IMS XML Database and XQuery

Did somebody say hierarchical data?

Christopher Holtz (holtz@us.ibm.com)

ON DEMAND BUSINESS℠ = Make it happen now

Important Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
- ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

Why XML…

- Standard Internet Data Exchange Format
  - Self-Describing
  - Handles encoding (internationalization)

```
<?xml version="1.1" encoding="ebcdic-cp-us"?>
```

  - Handles byte ordering

```
<OrderNumber>110203</OrderNumber>
```

  - Can Represent Almost Anything
  - Forces Syntax-Level Interoperability
  - Easily Parsed
  - Confers Longevity
  - **Standard!**

The XML Schema Definition Language

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

An XML Schema:

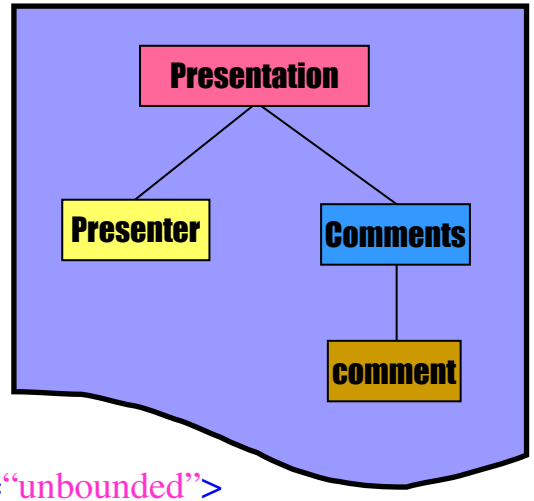
- defines elements and attributes that can appear in a document
- defines which elements are child elements
- defines the order and number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for 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.ibm.com/ims"
        targetNamespace="http://www.ibm.com/ims"
        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>
```



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

Standards built on XML (to name a few…)

- DTD
- DOM
- SAX
- SOAP
- SQL/XML
- VoiceXML
- WAP
- WSDL
- WS-Policy
- XForms
- XHTML
- XInclude
- XLink
- XML Base
- XML Encryption
- XML Key Management
- XML Processing
- XML Schema
- XML Signature
- XPath
- XPointer
- XQuery
- XSL and XSLT
- …

Financial

- FIXML
- FpML
- IFX
- MMDL
- OFX Schema
- RIXML
- SWIFTNet
- XBRL

Open Source and Java

- Xerces
- Xalan

- JAXB
- JAXP
- JAXR
- JAX-RPC
- JDOM
- XQJ

Other Field Vocabularies

- Accounting
- Advertising
- Astronomy
- Building
- Chemistry
- Construction
- Education
- Food
- Finance
- Government
- Healthcare
- Insurance
- Legal
- Manufacturing
- News
- Physics
- Telecommunications
- …

Why XML Databases…

- Physical Storage Benefits
  - Very much depends on the type of data and how it is going to be used!!

- Data coming in as XML and going out as XML
  - Need to convert to / from XML
    - Structure
    - Types / Encoding
  - Conceptual divide between XML and Physical Storage Layout
  - XML Standards (XML Schema's, XQuery, etc)

11.4

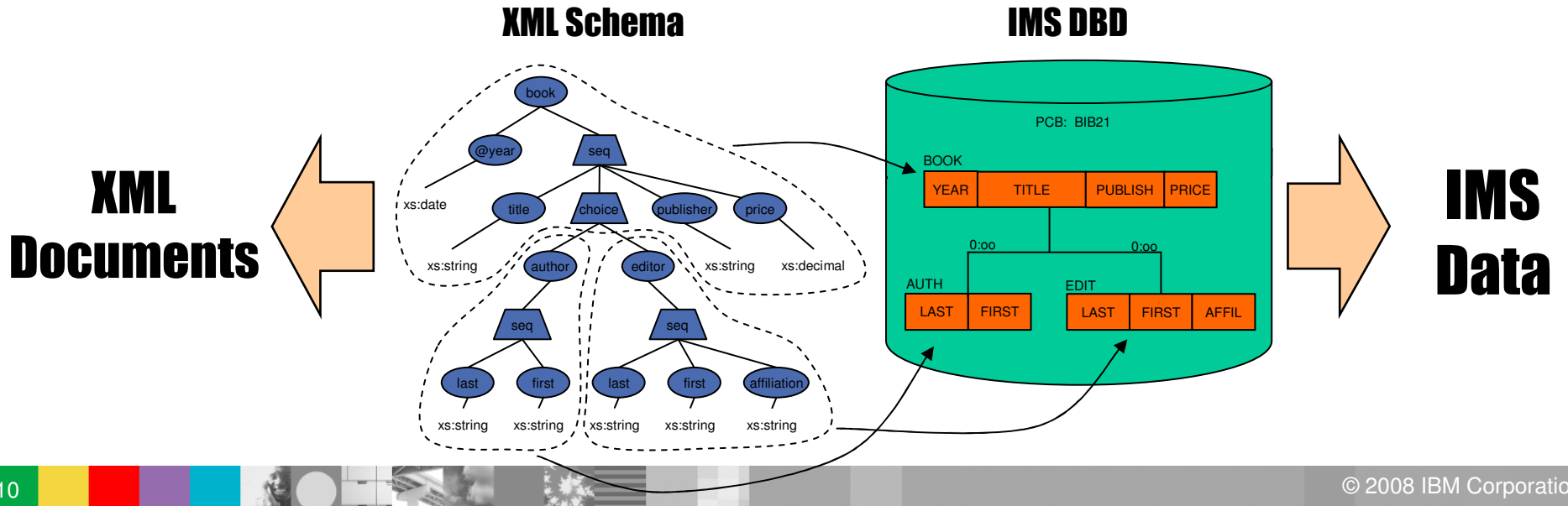
XML Databases really sound ideal for handling data in heterogeneous environments where I'm receiving and transmitting data as XML…

…but isn't IMS already the world's premier hierarchical database management system with nearly 40 years of proven experience…

…and couldn't I just take my terabytes of existing IMS hierarchical data and ***view*** it as XML when it fits to do so…

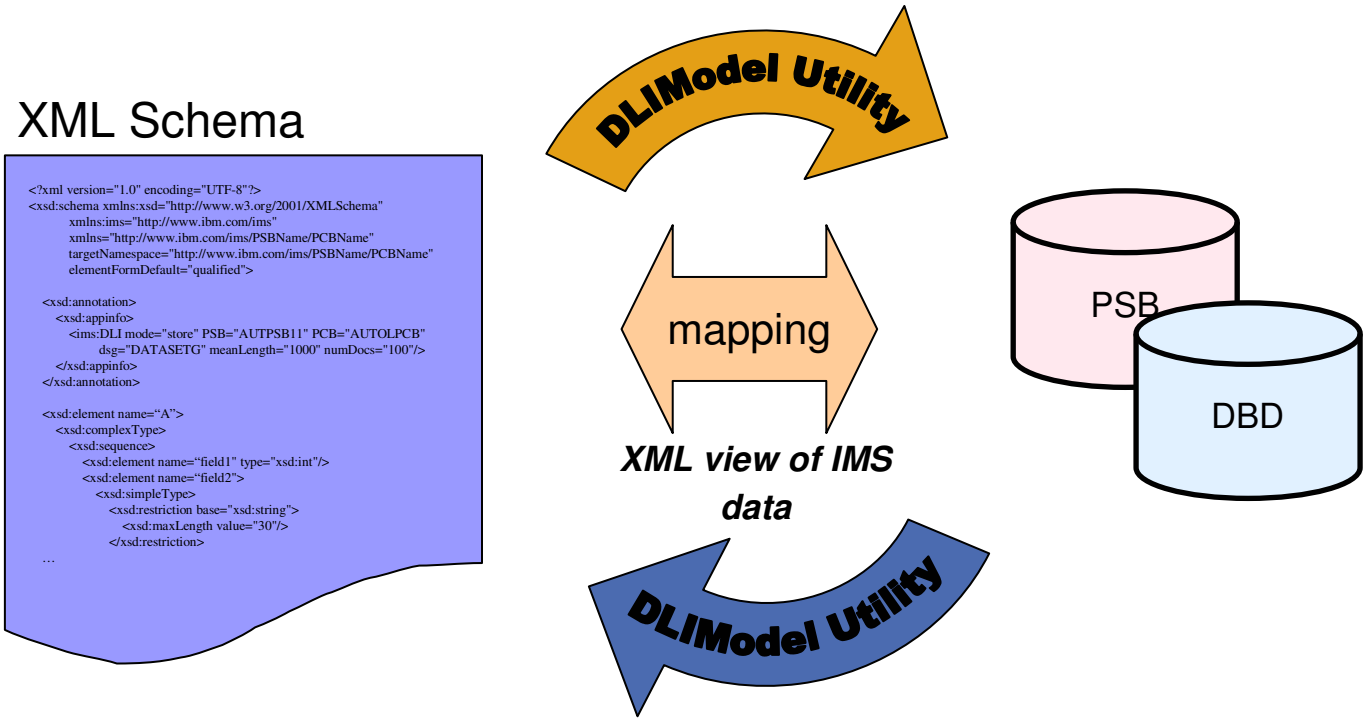
IMS XML Database

- Introduces a way to view/map *native* IMS hierarchical data to XML documents
- Aligns IMS Database (DBD) with XML Schema
- Allows the retrieval and storage of IMS Records as XML documents with ***no change*** to existing IMS databases

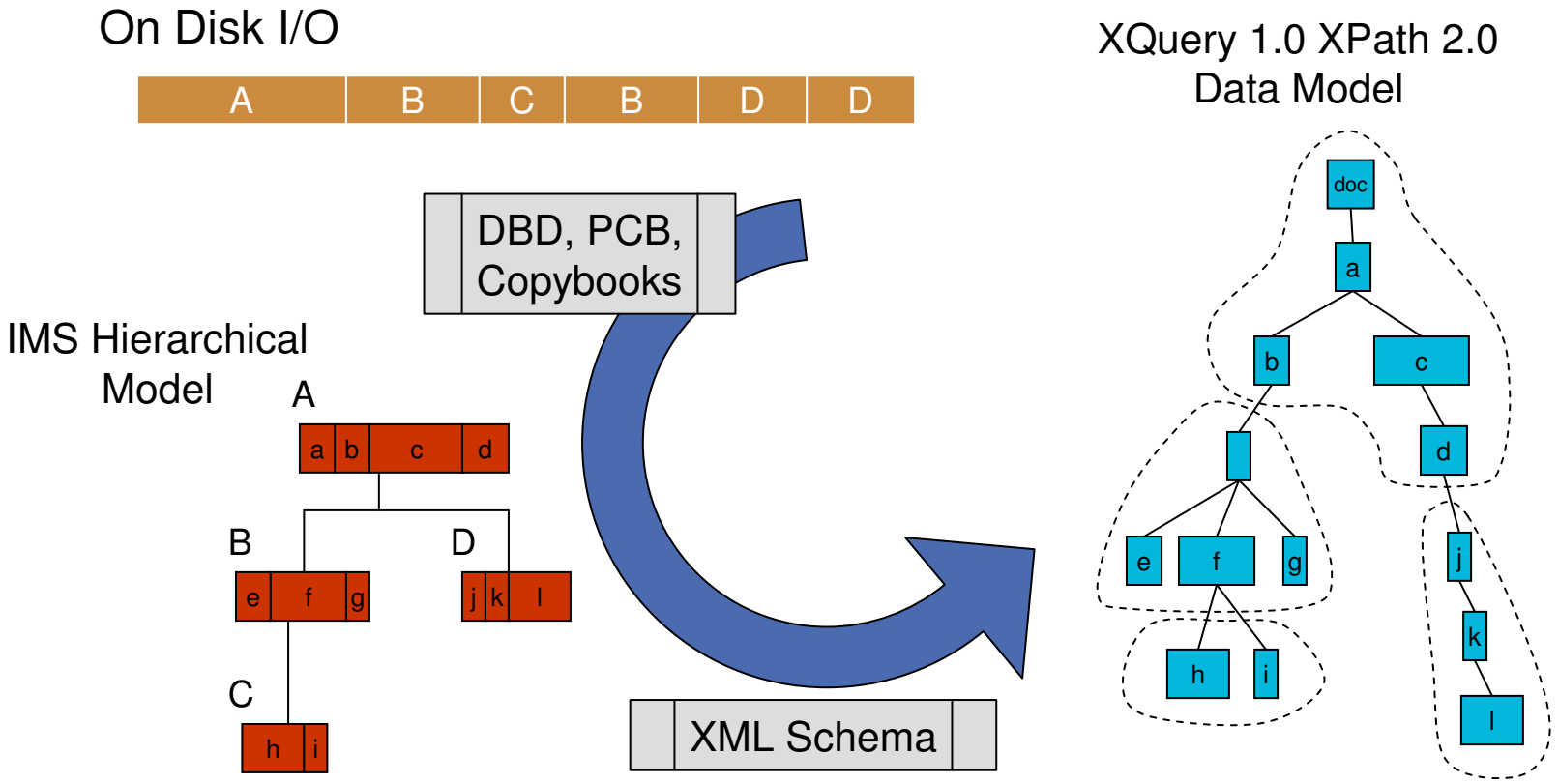


IMS XML-DB Metadata

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



XML Visualization



It's the Metadata, Stupid!

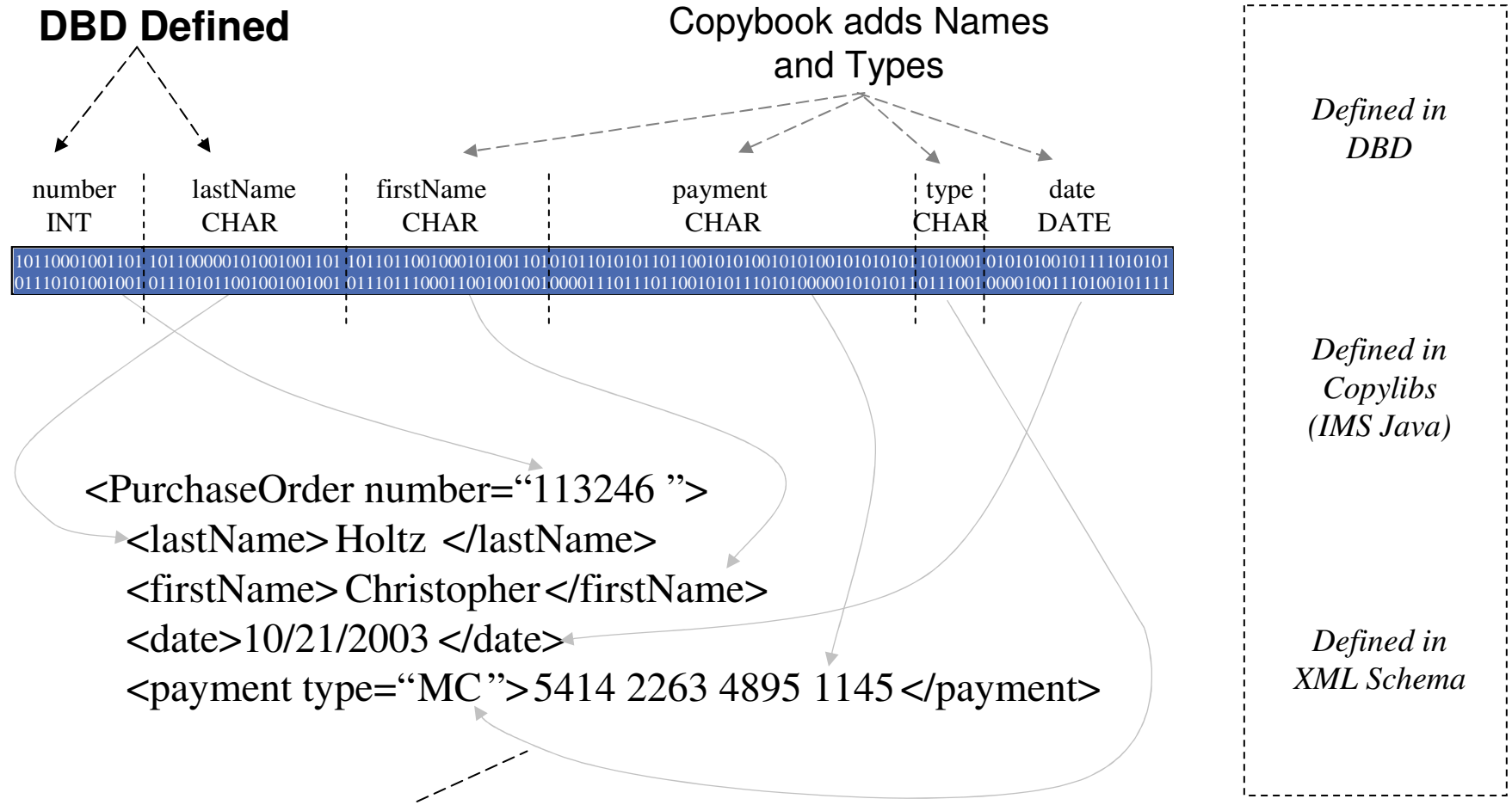
- Physical Metadata
  - Segment Sizes
  - Segment Hierarchy (*field relationships – 1-to-1, 1-to-n*)
  - DBD Defined Fields
  - Application Defined Fields
  - Field Type, Type Length, Byte Ordering, Encoding, etc.
  - Offer Field/Segment Renaming *(lift 8 char restriction)*
- Structural Metadata
  -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

Defined in DBD

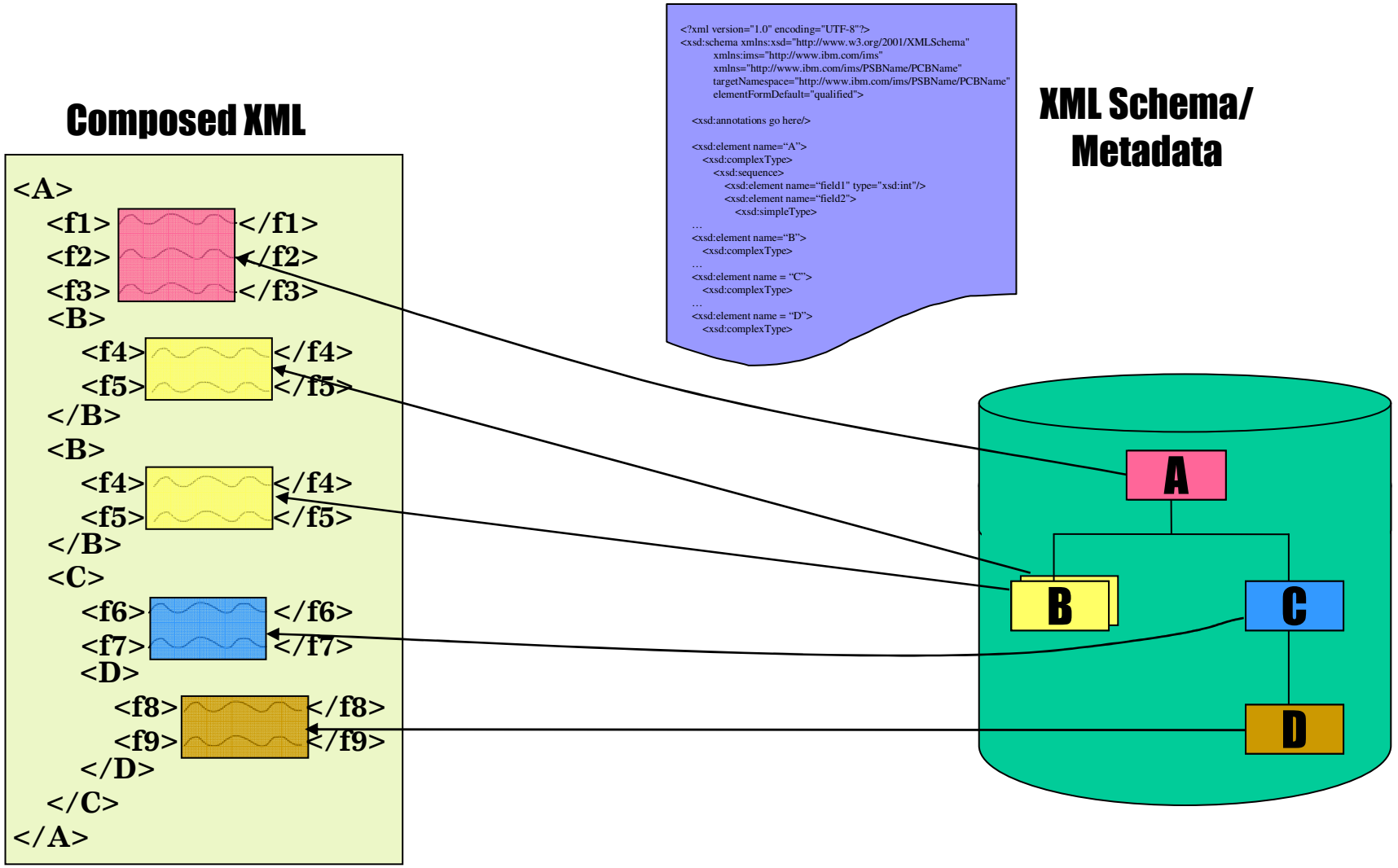
Defined in Copylibs (IMS Java)

Defined in XML Schema

It's the Metadata, Stupid!



Decomposed XML Retrieval in IMS

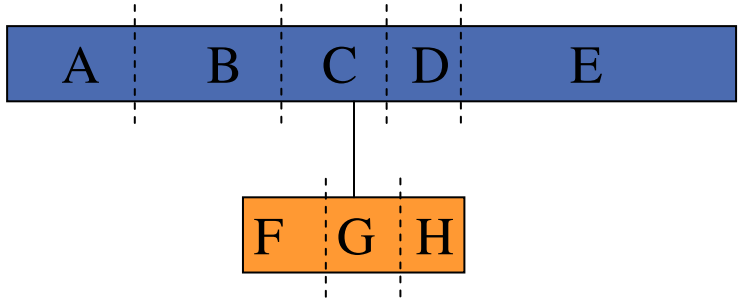


Decomposed Storage

- XML document must be parsed and validated.
- Data is 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.

Simple XML Structuring

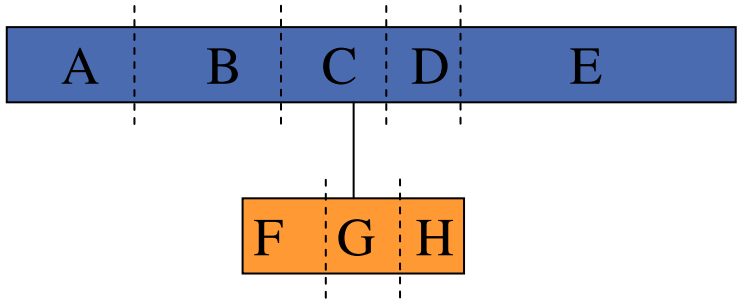
- Uses Default structure
- All data is represented as elements




```
<?xml version="1.0"?>
<SegName>
    <A> [data] </A>
    <B> [data] </B>
    <C> [data] </C>
    <D> [data] </D>
    <E> [data] </E>
    <Seg2Name>
        <F> [data] </F>
        <G> [data] </G>
        <H> [data] </H>
    </Seg2Name>
    <Seg2Name>
        <F> [data] </F>
        <G> [data] </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>
        </another>
    </sub>
</document>
```

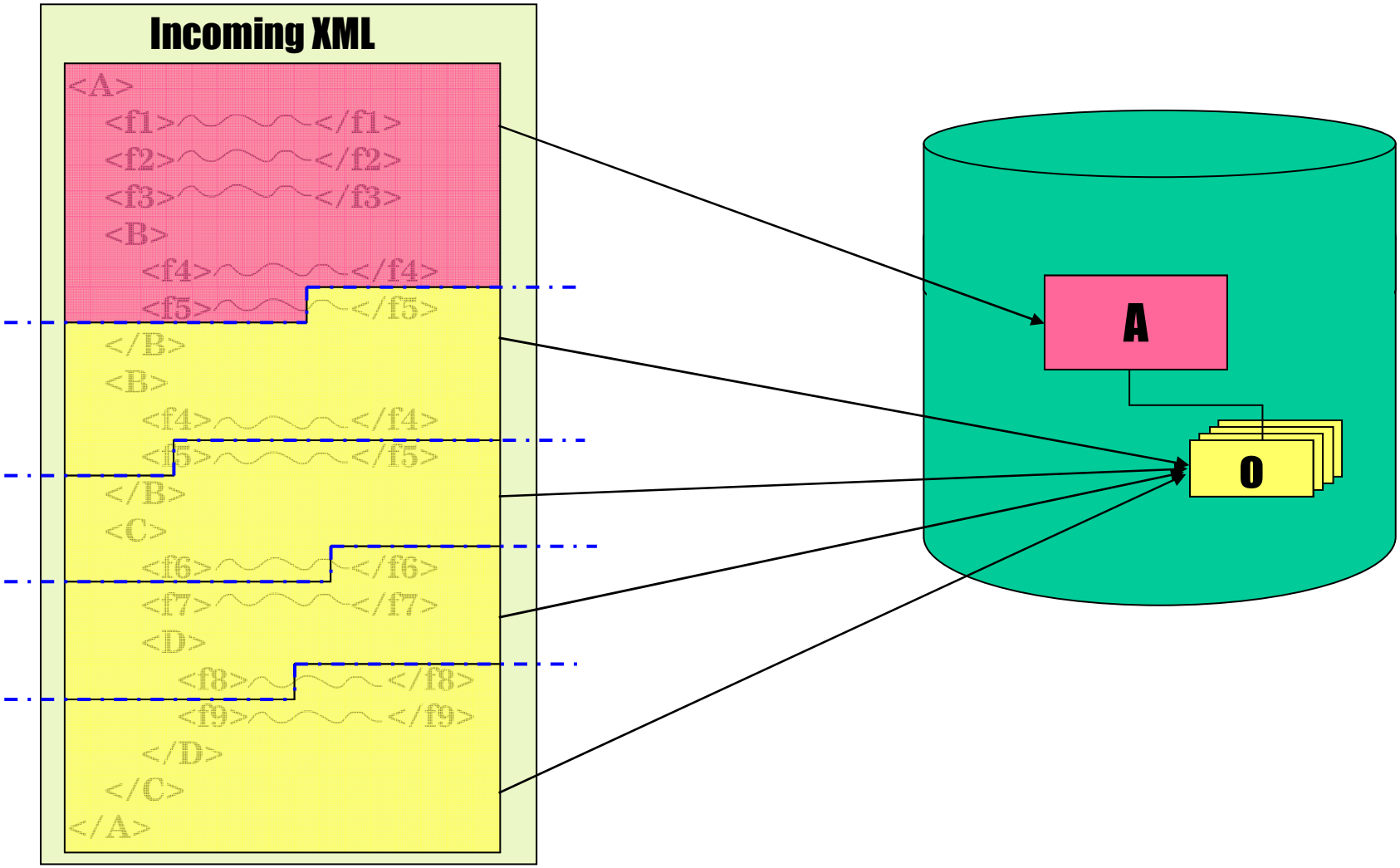
Two general types of XML documents

- 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.

Intact Storage

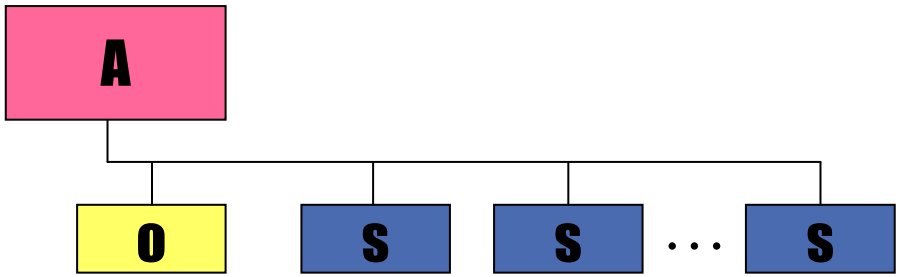
- No (or little) XML Parsing or Schema validation
  - Storage and Retrieval Performance
- No (or little) data type conversions
  - Unicode storage
- Stored documents are no longer searchable by IMS and only accessible to XML-aware applications

Intact XML Storage in IMS



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.

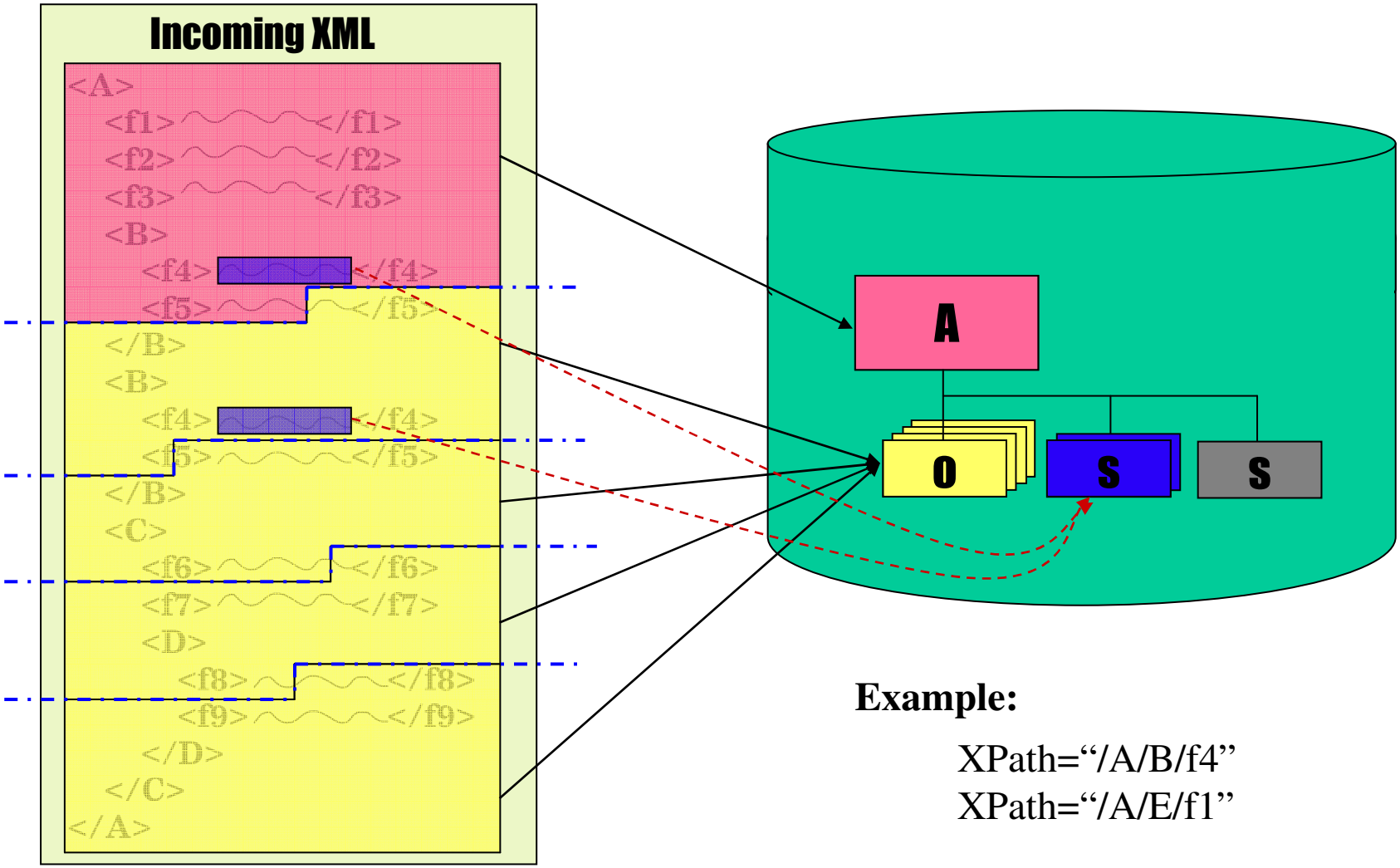


Example:

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

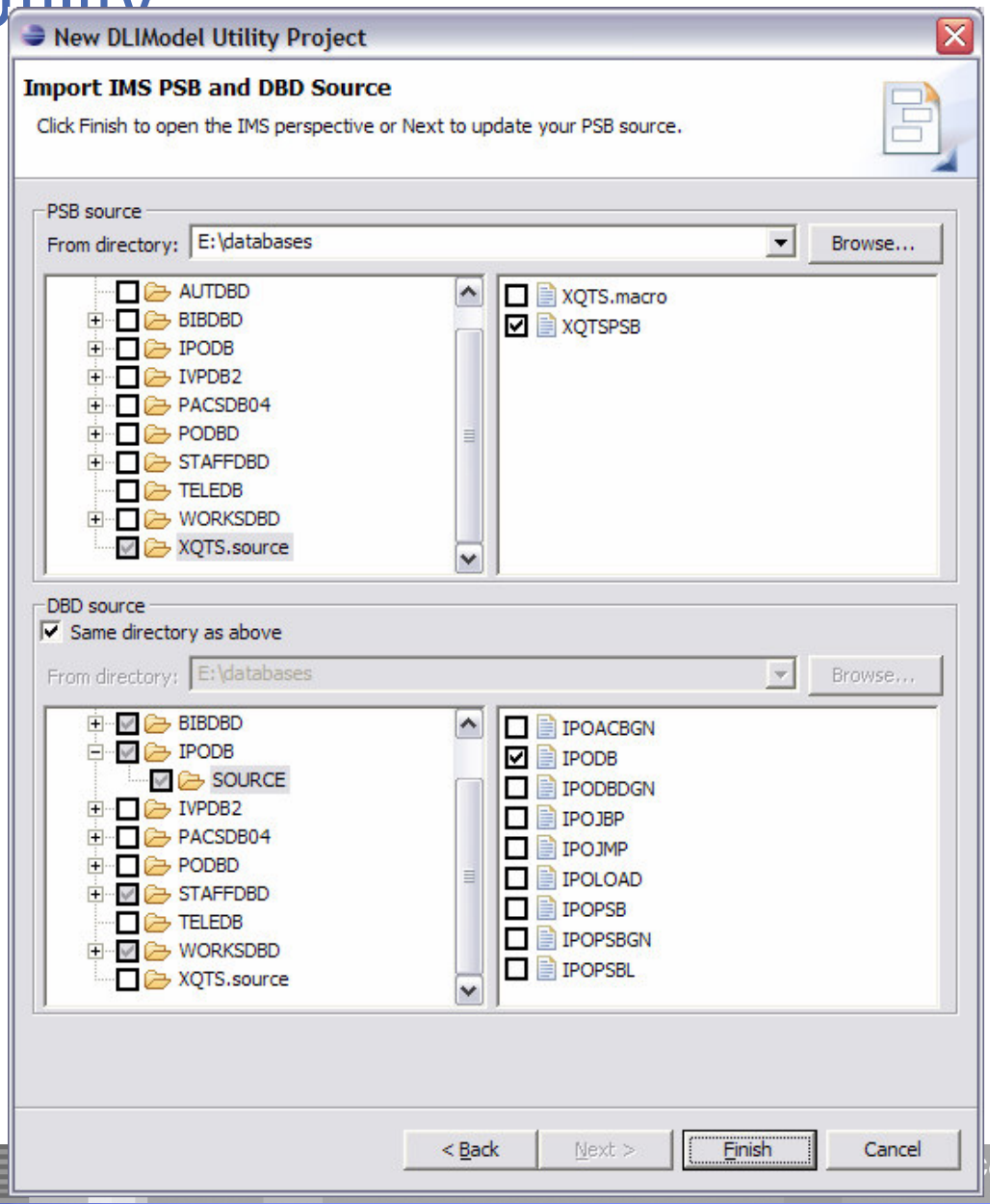
- Decomposed (*data-centric storage*)
  - XML tags are stripped from XML data
  - Identical as current IMS storage
  - Strict data-centric XML Schema validated data
  - EBCDIC (or local) encoding
  - Searching on IMS Search Fields
  - Transparently accessible by non-XML enabled applications

- 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
  - Only accessible by XML enabled applications

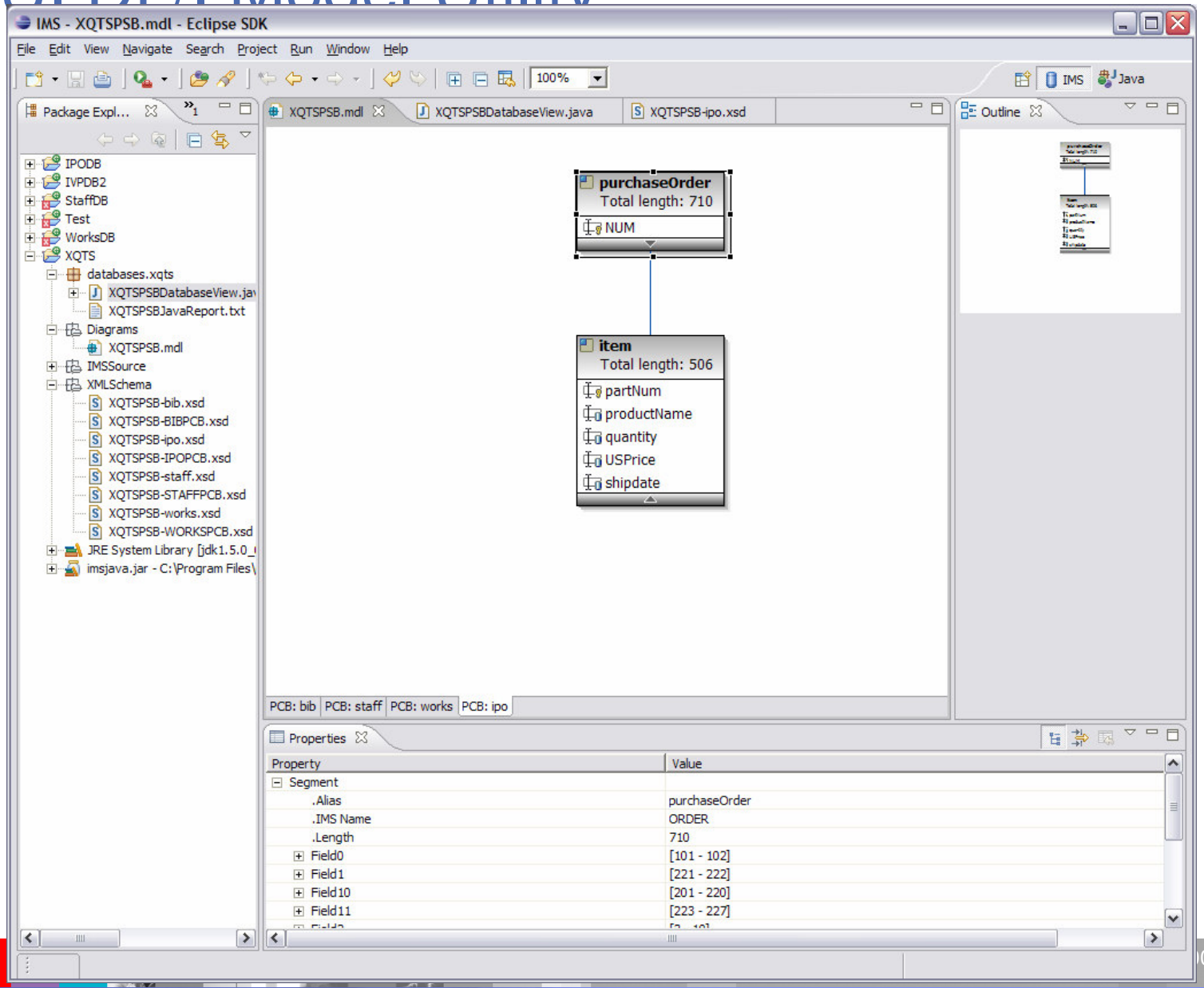
Wonderful…but this is all high level
abstract internal mumbo jumbo
…how do I actually use it

GUI DL/I Model Utility

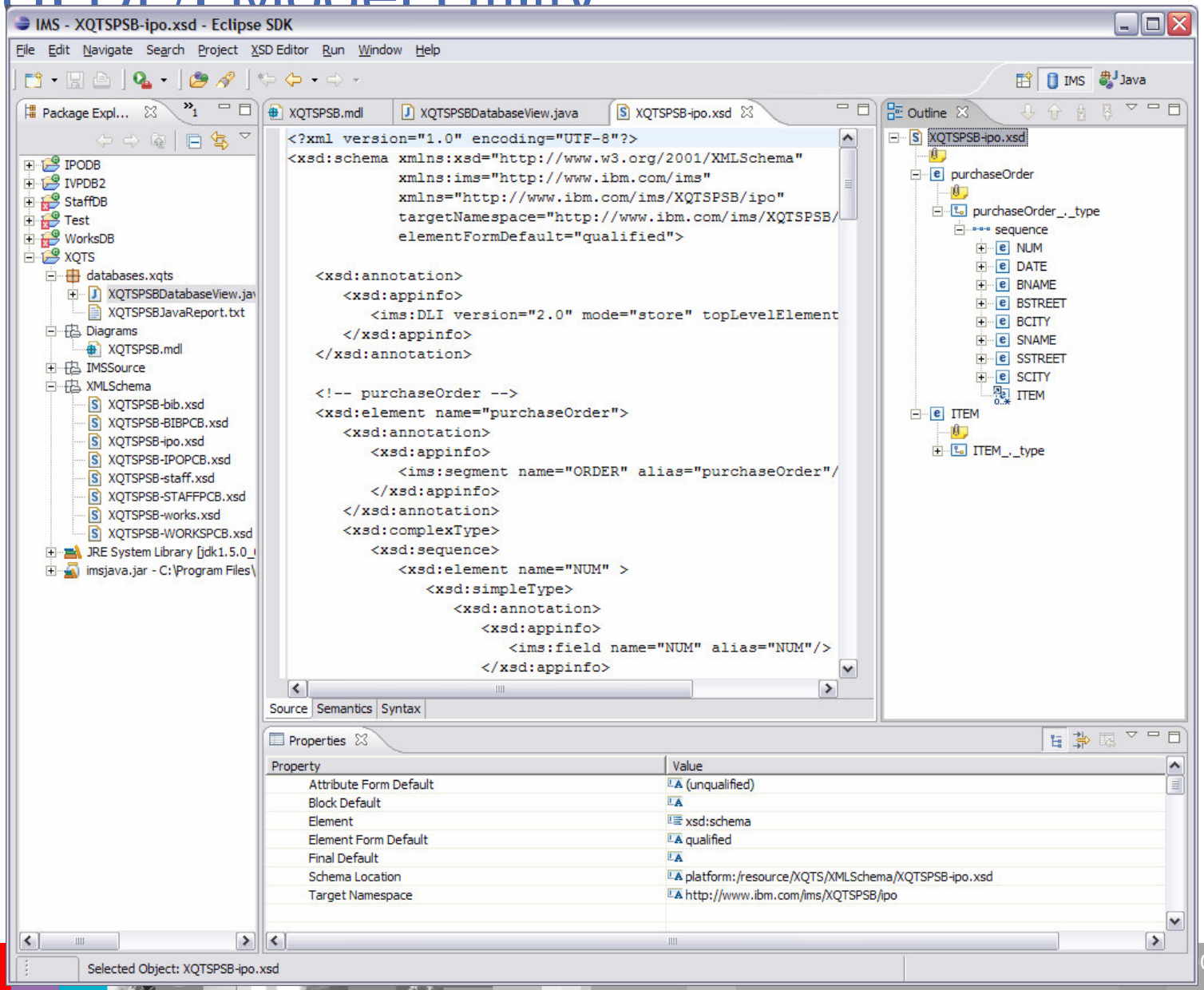
- Select DBDs / PSBs
  - must be FTPed locally
- Source is Parsed
  - any errors are reported
- XMI Metamodel
  - generated
  - opened for editing



GUI DL/I Model Utility

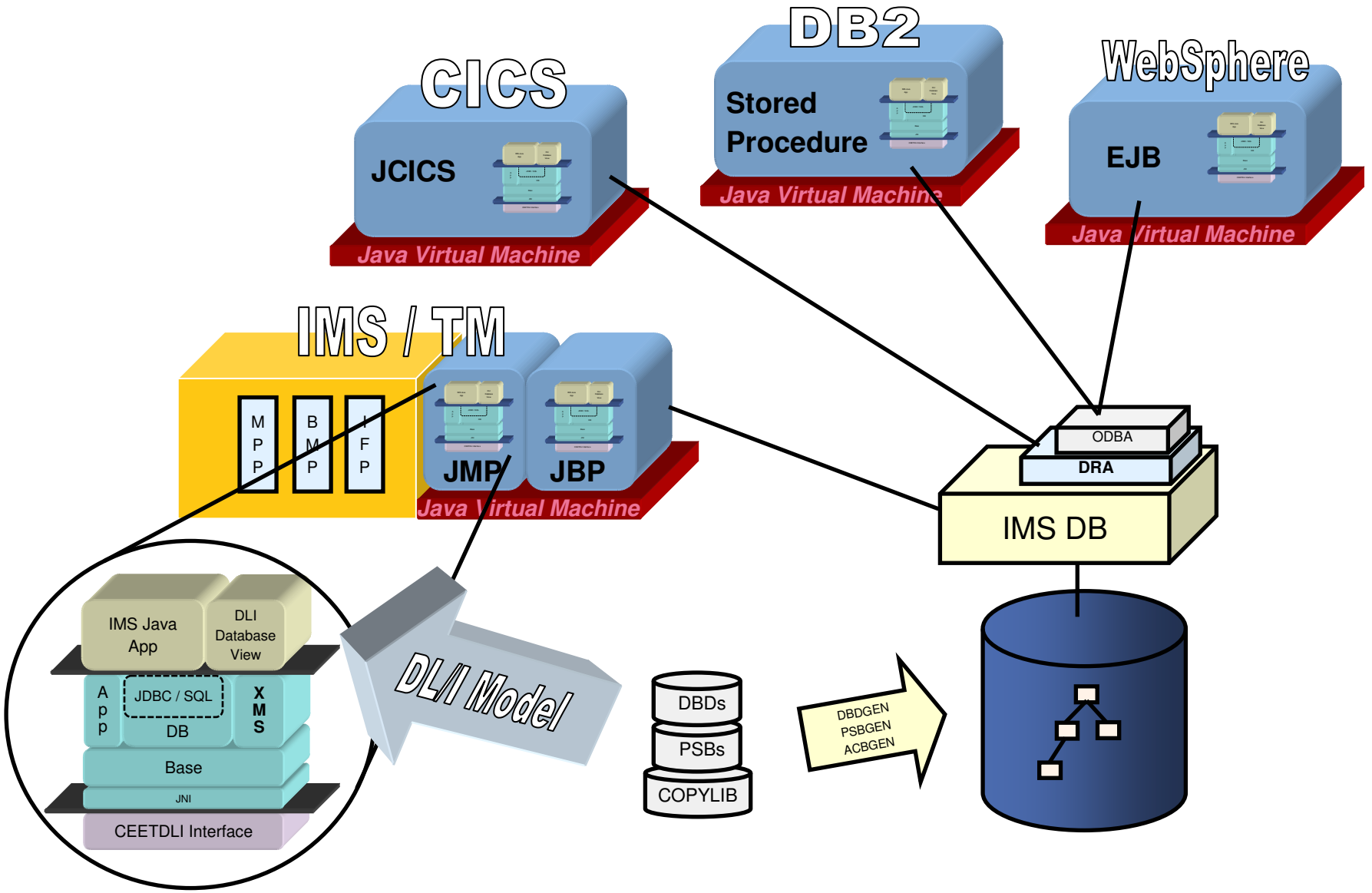


GUI DL/I Model Utility


IMS - XQTSPSB-ipo.xsd - Eclipse SDK
File Edit Navigate Search Project XSD Editor Run Window Help
IMS
Java
Package Expl...
IPODB
IVPDB2
StaffDB
Test
WorksDB
XQTS
databases.xqts
XQTSPSBDatabaseView.jav
XQTSPSBJavaReport.txt
Diagrams
XQTSPSB.mdl
IMSSource
XMLSchema
XQTSPSB-bib.xsd
XQTSPSB-BIBPCB.xsd
XQTSPSB-ipo.xsd
XQTSPSB-IPOPCB.xsd
XQTSPSB-staff.xsd
XQTSPSB-STAFFPCB.xsd
XQTSPSB-works.xsd
XQTSPSB-WORKSPCB.xsd
JRE System Library [jdk1.5.0_
imsjava.jar - C:\Program Files\
XQTSPSB.mdl
XQTSPSBDatabaseView.java
XQTSPSB-ipo.xsd
<?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/XQTSPSB/ipo"
targetNamespace="http://www.ibm.com/ims/XQTSPSB/
elementFormDefault="qualified">
<xsd:annotation>
<xsd:appinfo>
<ims:DLI version="2.0" mode="store" topLevelElement
</xsd:appinfo>
</xsd:annotation>
<!-- purchaseOrder -->
<xsd:element name="purchaseOrder">
<xsd:annotation>
<xsd:appinfo>
<ims:segment name="ORDER" alias="purchaseOrder"/
</xsd:appinfo>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element name="NUM" >
<xsd:simpleType>
<xsd:annotation>
<xsd:appinfo>
<ims:field name="NUM" alias="NUM"/>
</xsd:appinfo>
Source Semantics Syntax
Outline
XQTSPSB-ipo.xsd
purchaseOrder
purchaseOrder_._type
sequence
NUM
DATE
BNAME
BSTREET
BCITY
SNAME
SSTREET
SCITY
ITEM
ITEM
ITEM_._type
Properties
Property
Value
Attribute Form Default
(unqualified)
Block Default
Element
xsd:schema
Element Form Default
qualified
Final Default
Schema Location
platform:/resource/XQTS/XMLSchema/XQTSPSB-ipo.xsd
Target Namespace
http://www.ibm.com/ims/XQTSPSB/ipo
Selected Object: XQTSPSB-ipo.xsd

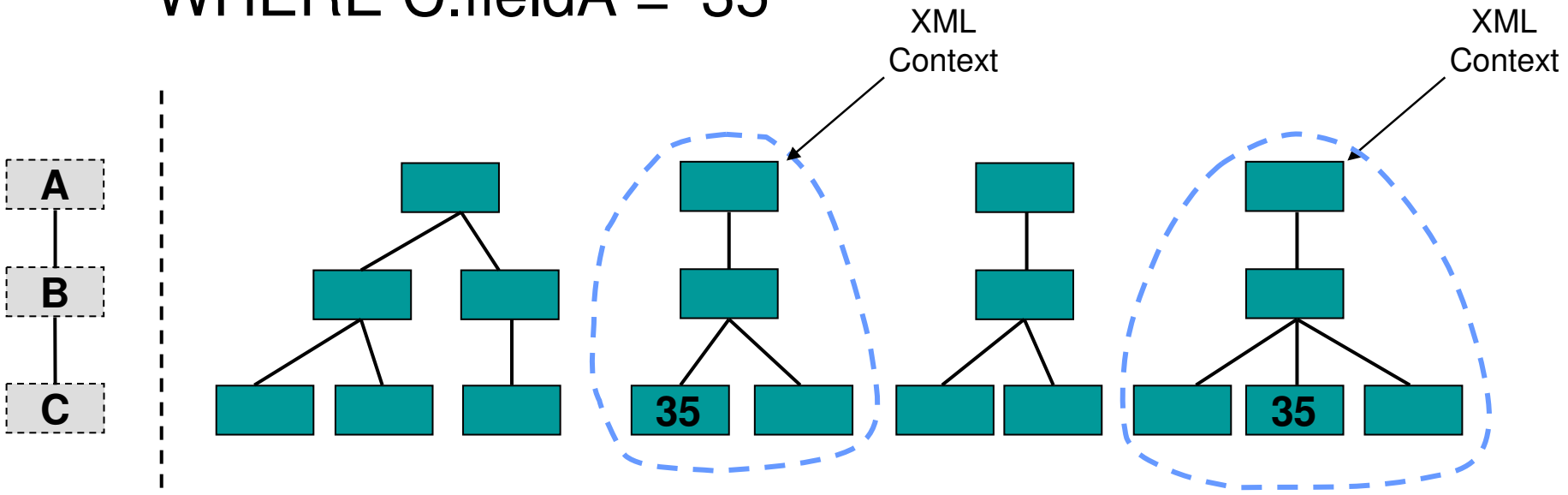
IMS JDBC interface for XML

IMS JDBC Runtime



retrieveXML() UDF

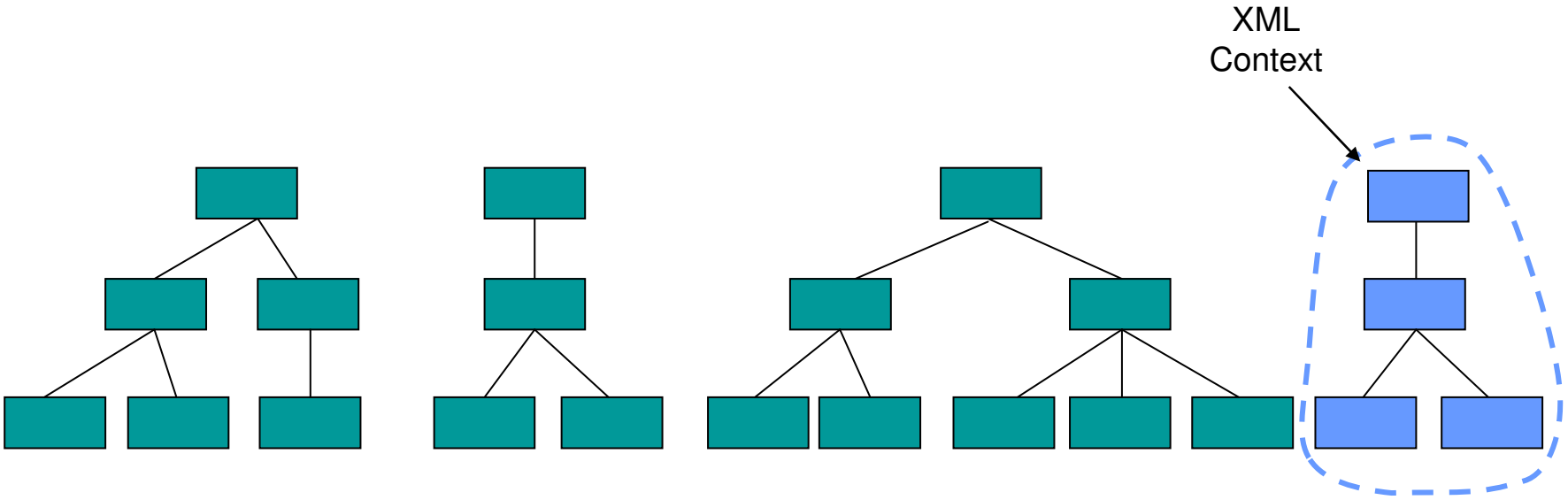
```
SELECT retrieveXML(A)
FROM C
WHERE C.fieldA = '35'
```



**Two Rows of XML CLOBs in the ResultSet*

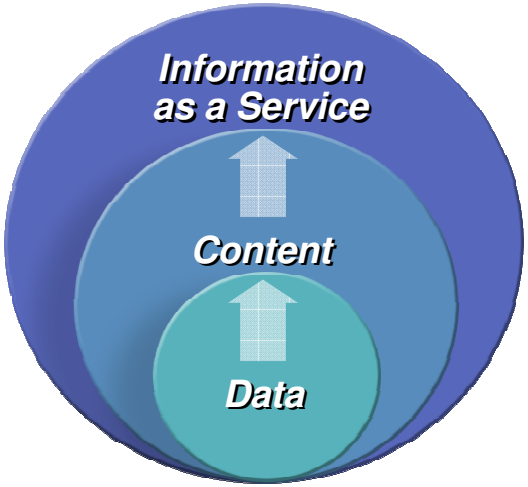
storeXML() UDF

```
INSERT INTO A (storeXML())
VALUES (?)
```



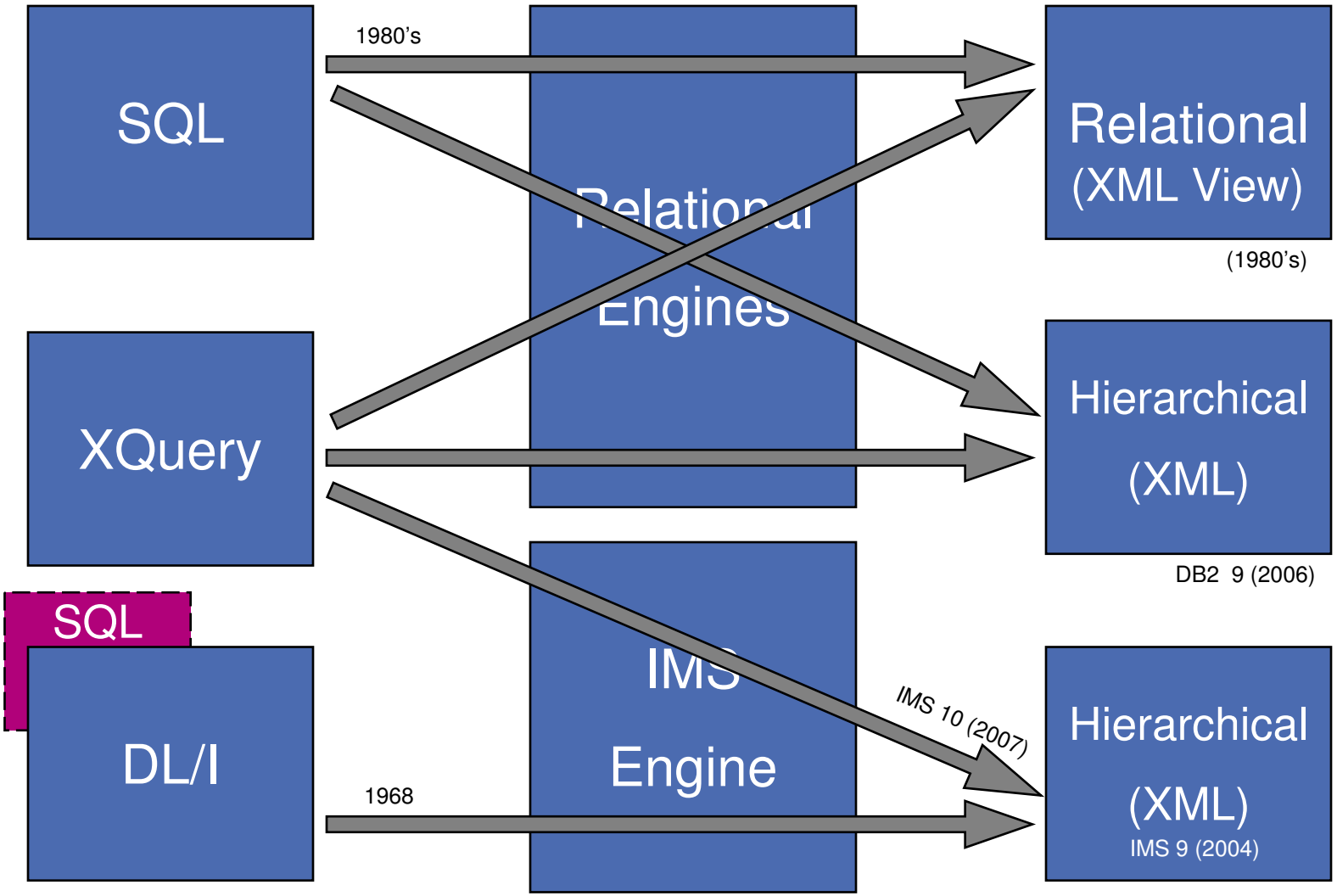
**Insert Statement must be a Prepared Statement*

XQuery support in IMS V10



- Designed to address the querying of collections of XML data.
- Semantically similar to SQL
- XQuery 1.0 became a W3C recommendation in Jan 2007
- More natural fit for querying hierarchical data
- Enables customers to leverage emerging standard skill set
- Enhanced product and tooling integration
- Transparently query existing IMS data as XML documents

Road to Interoperability through XQuery



XQuery FLWOR Expressions

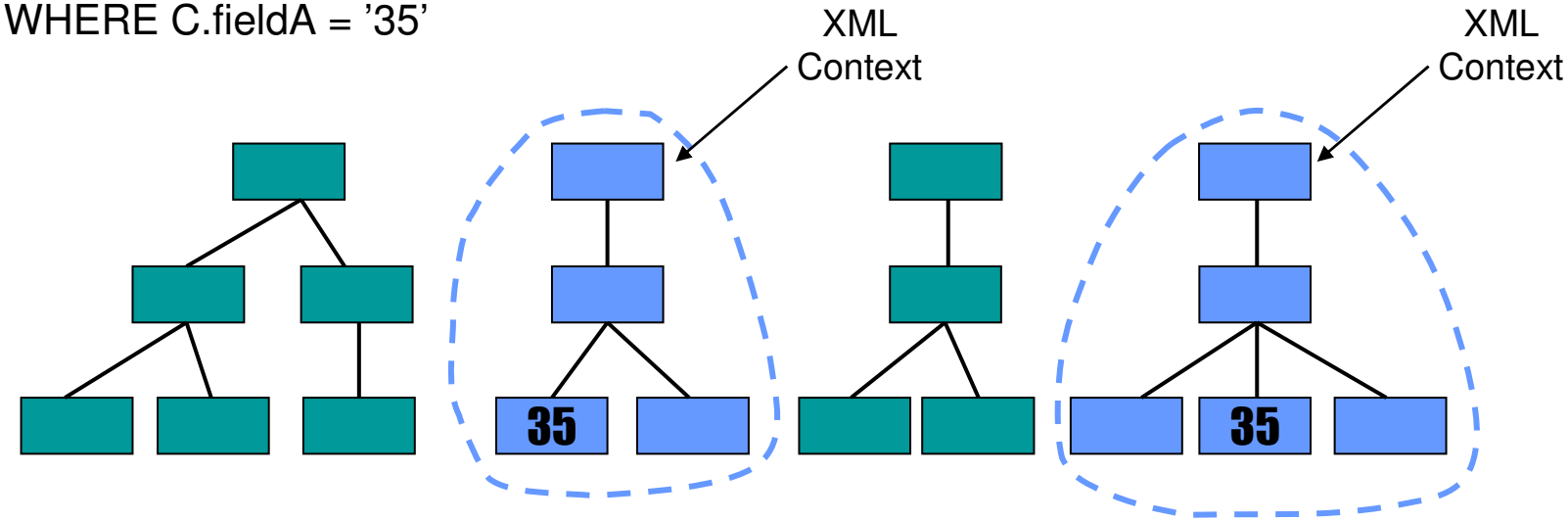
- **F**OR: iterates through a sequence, bind variable to items
- **L**ET: binds a variable to a sequence
- **W**HERE: eliminates items of the iteration
- **O**RDER BY: reorders items of the iteration
- **R**ETURN: constructs query results

```
<bib> {
   for $b in /bib/book
   let $title := $b/title
   where $b/publisher = "Addison-Wesley“
   order by $b/@year
   return
      <book year="{ $b/@year }">
         { $title }
      </book>
} </bib>
```

```
<bib>
   <book year=“1992">
      <title>Advanced Programming in the Unix
   </book>
   <book year=“1994">
      <title>TCP/IP Illustrated</title>
   </book>
</bib>
```

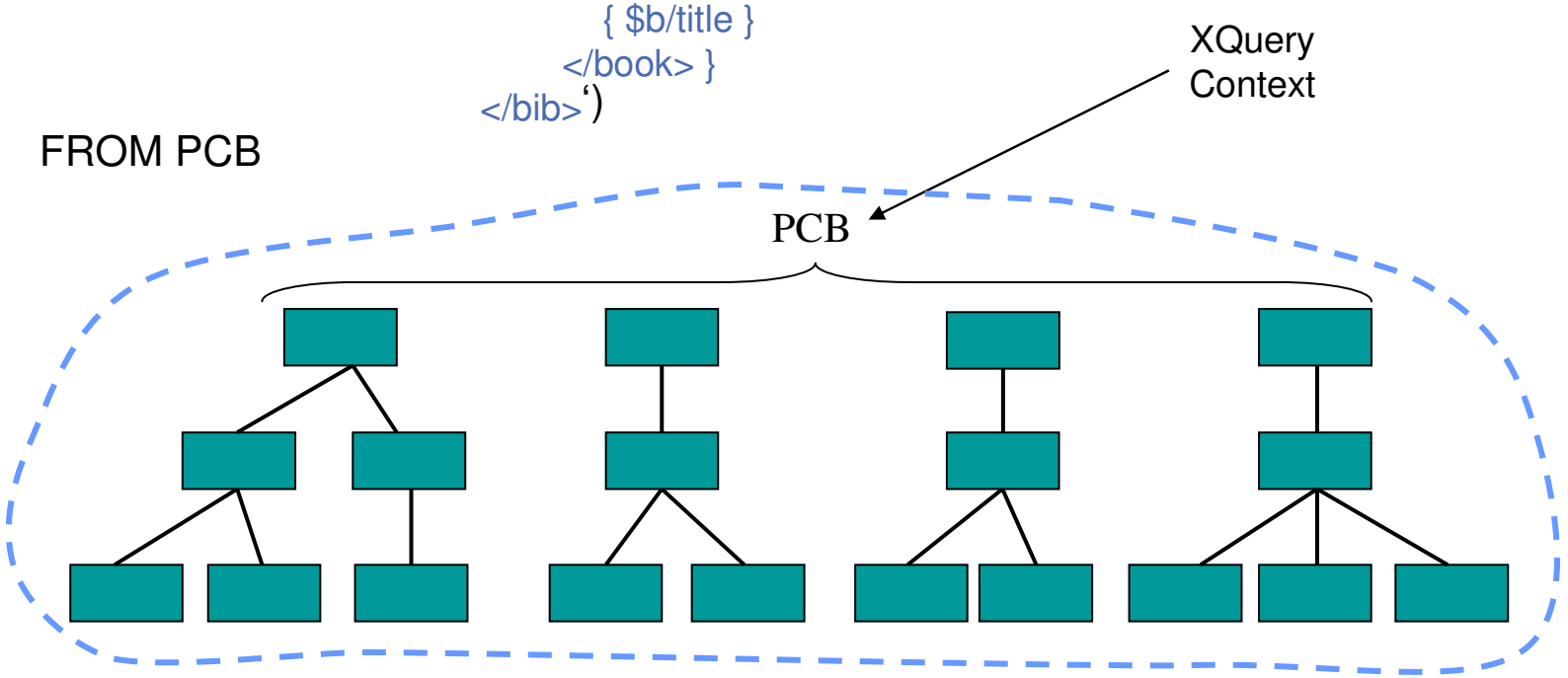
Extended JDBC interface…

```
SELECT retrieveXML( A,‘<bib> {
                          for $b in book
                          where $b/publisher = 'Addison-Wesley                ‘
                             and $b/@year > 1991
                          return
                             <book year="{ $b/@year }">
                                { $b/title }
                             </book> }
                       </bib> ‘)
FROM C
WHERE C.fieldA = ’35’
```



Extended JDBC interface…

```
SELECT retrieveXML( '<bib> {
                        for $b in /bib/book
                        where $b/publisher = 'Addison-Wesley
                            and $b/@year > 1991
                        return
                            <book year="{ $b/@year }">
                               { $b/title }
                            </book> }
                    </bib>')
FROM PCB
```



Why XML Databases…

- Data coming in as XML and going out as XML
  - Need to convert to / from XML
    - Structure
    - Types / Encoding
  - Conceptual divide between XML and Physical Storage Layout
  - XML Standards (XML Schema's, XQuery, etc)

- Very much depends on the type of data and how it is going to be used!!

XML can be a better choice than relational for...

- Data that's inherently hierarchical or nested in nature
  - Example: Medical data, Bill-of-materials, etc.
- Data sets with sparsely populated attributes
  - Example: FIXML, FpML, Customer profiles

These are already IMS's Strengths

- Schema evolution
  - Example: Frequently changing services/products/processes
- Variable schemas, many schemas
  - Example: Data integration, consolidation
- Combining structured & unstructured data
  - Example: CM, Life Sciences, News & Media

These are not supported (well) in IMS

SEPA – IMS Support

Proof of Concept

ON DEMAND BUSINESS℠ = Make it happen now

Standards built on XML (to name a few…)

- DTD
- DOM
- SAX
- SOAP
- SQL/XML
- VoiceXML
- WAP
- WSDL
- WS-Policy
- XForms
- XHTML
- XInclude
- XLink
- XML Base
- XML Encryption
- XML Key Management
- XML Processing
- XML Schema
- XML Signature
- XPath
- XPointer
- XQuery
- XSL and XSLT
- …

Financial

- FIXML
- FpML
- IFX
- MMDL
- OFX Schema
- RIXML
- SWIFTNet
- XBRL

Open Source and Java

- Xerces
- Xalan

- JAXB
- JAXP
- JAXR
- JAX-RPC
- JDOM
- XQJ

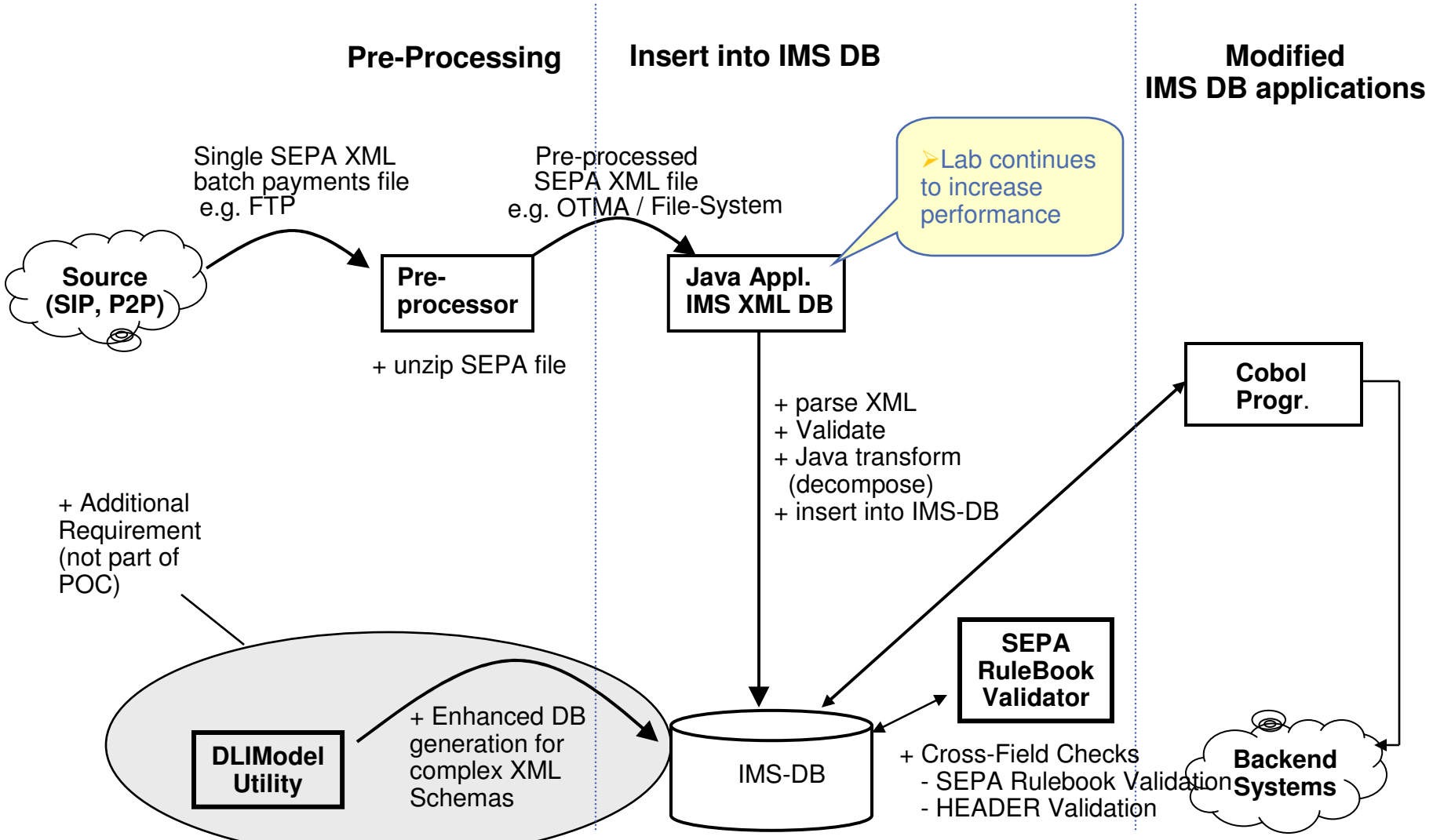
Other Field Vocabularies

- Accounting
- Advertising
- Astronomy
- Building
- Chemistry
- Construction
- Education
- Food
- Finance
- Government
- Healthcare
- Insurance
- Legal
- Manufacturing
- News
- Physics
- Telecommunications
- …

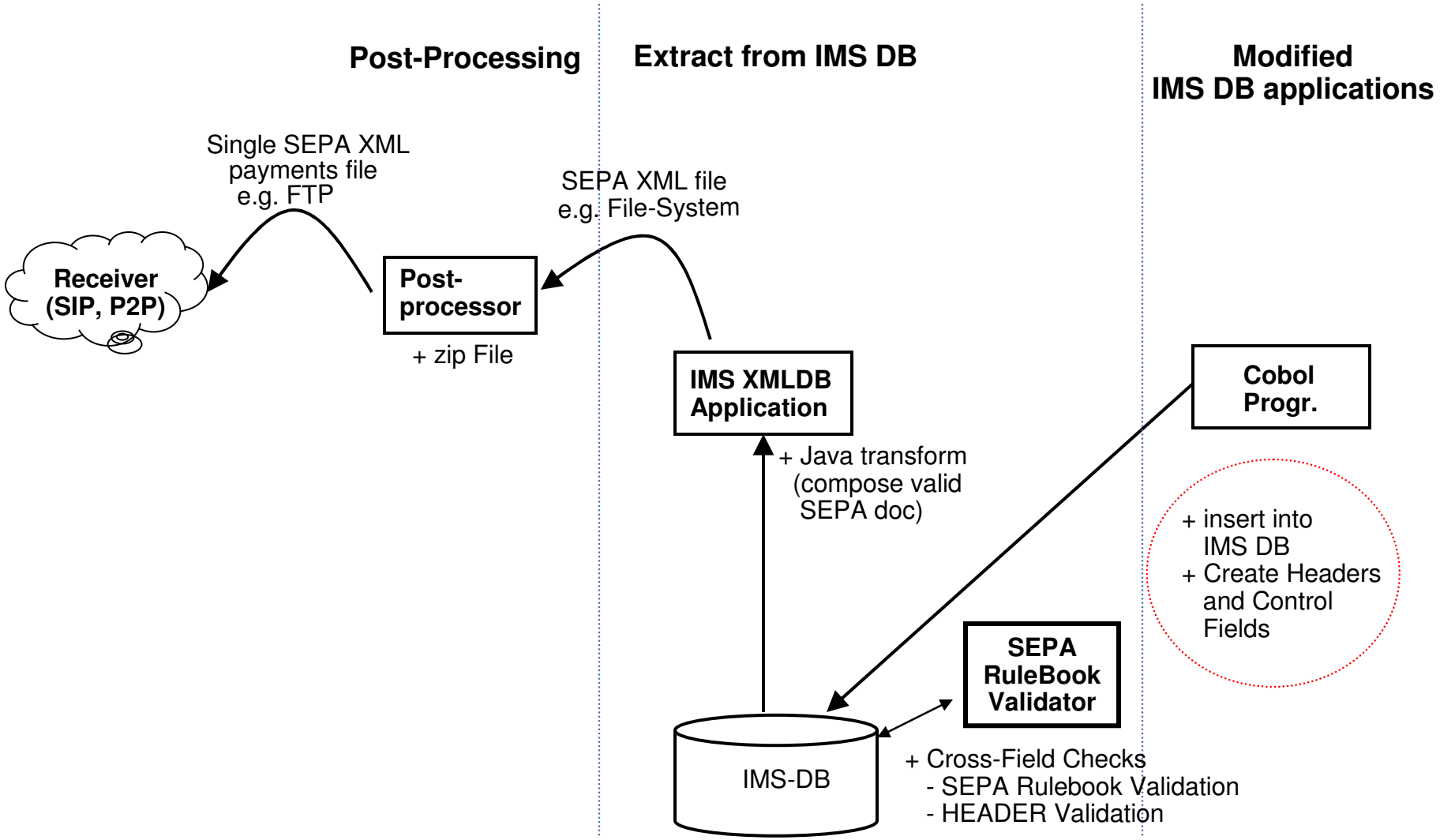
Non-Functional Requirements

- **Processing Load:**
  - SEPA file can be up to 250MB compressed (zip/gzip) – size varies
  - SEPA file uncompressed ~ 10 GBytes
  - Payment instruction messages contained: up to 10 Million
- **Processing Latency:**
  - ~ 1 [hr]
- **Scalability**
  - Easy and linear / near-linear
  - Possibility for parallel processing with synchronous output
- **Security**
  - not applicable yet

Future proposed by IMS Lab



IMS Outbound support for SEPA



Thank You for Joining Us today!

Go to **www.ibm.com/software/systemz** to:

- Replay this teleconference
- Replay previously broadcast teleconferences
- Register for upcoming events