E43

XML Storage in IMS: What's Next

Christopher Holtz

IMS technical conference

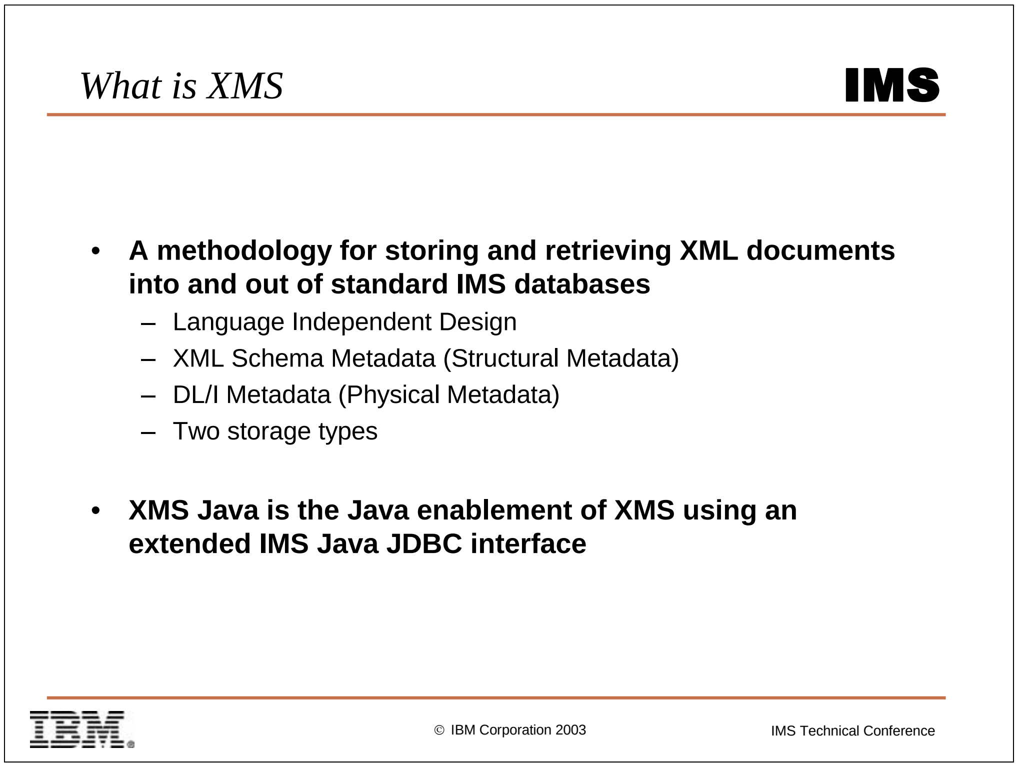
Las Vegas, NV **September 15 – September 18, 2003**

Abstract and Title for this talk is a bit misleading because at the time of writing it was not for sure that this code was going to make version 9.

Overview

IMS

- **Introduction**
  - What is XMS?
  - What is XML?
  - What are XML Schemas?
- **XMS Methodology**
  - Decomposed Storage
  - Intact Storage
- **XMS Tooling**
  - Metadata Generation
- **XMS Java Implementation**
  - SQL UDF Interface
  - Future



What is XMS

IMS

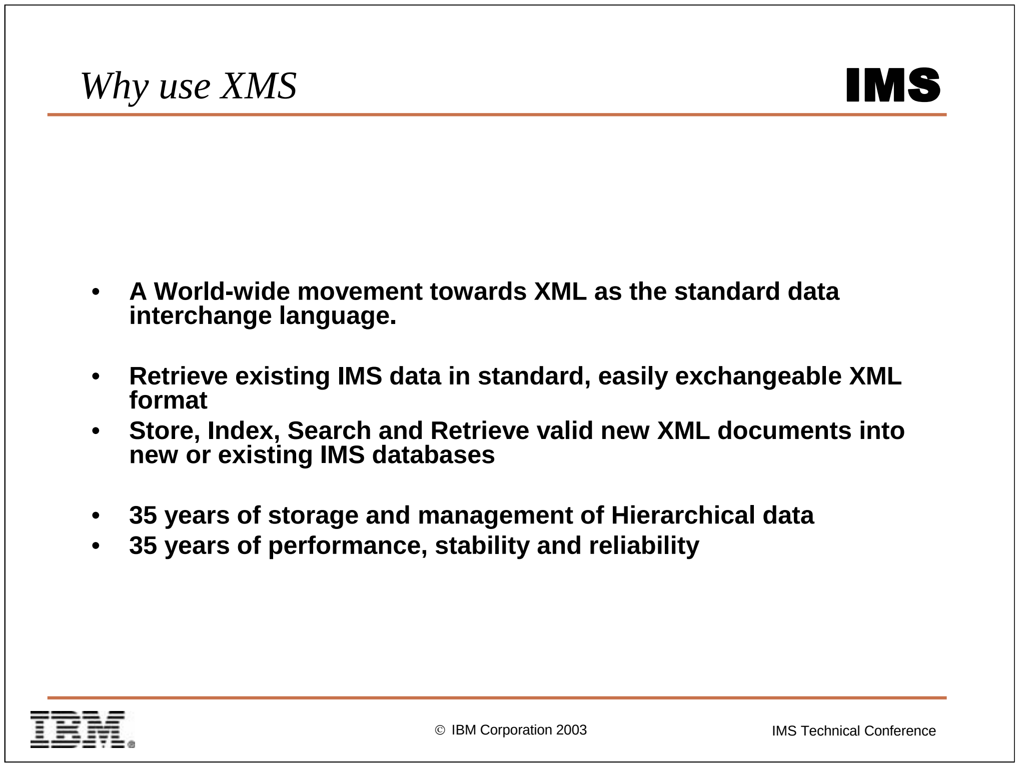
- **A methodology for storing and retrieving XML documents into and out of standard IMS databases**
  - Language Independent Design
  - XML Schema Metadata (Structural Metadata)
  - DL/I Metadata (Physical Metadata)
  - Two storage types
- **XMS Java is the Java enablement of XMS using an extended IMS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

IBM © IBM Corporation 2003 IMS Technical Conference

We will talk about the IMS XML Database methodology. This methodology is tied to no specific language, just as DL/I segments and fields are not tied to any particular language. The XML Schema metadata is used to map the between the structure of DL/I Segments and fields and the structure of XML elements and attributes.

The DL/I Metadata (IMS Java Metadata for Java) describes physical characteristics of the database (Name aliases, DL/I type storage (COMP-1, COMP-2, PIC Strings, etc.), Segment Sizes, etc.

XMS Java is a Java JDBC implementation building upon IMS Java.



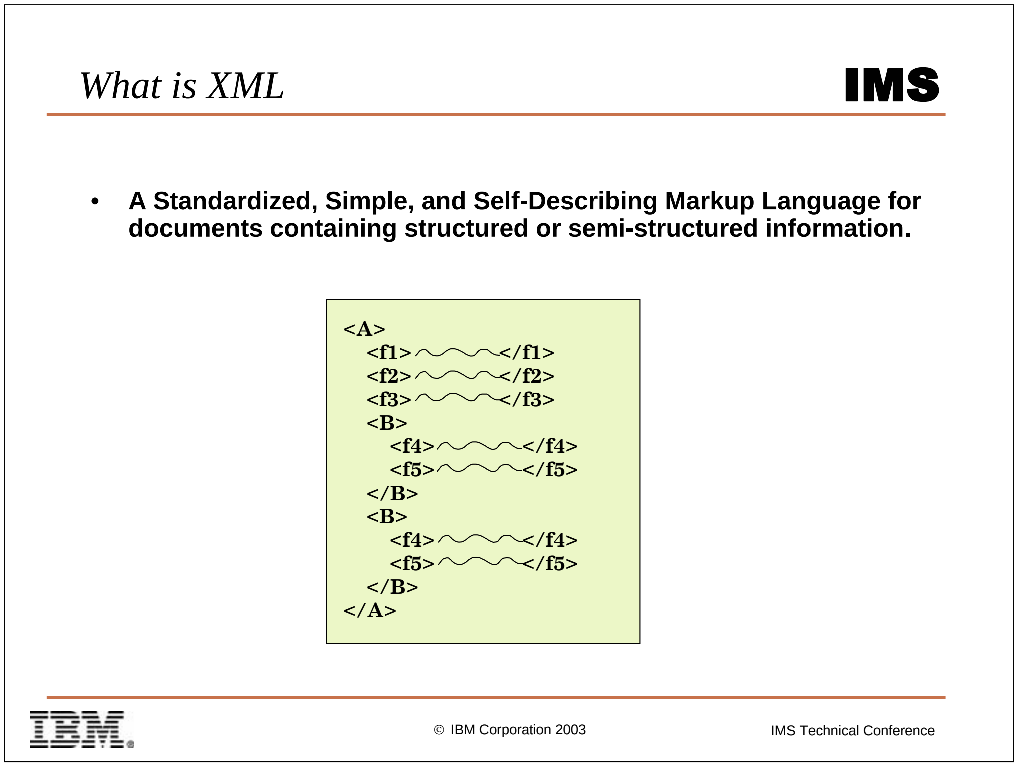
Why use XMS IMS

- 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

© IBM Corporation 2003 IMS Technical Conference

Exponential growth of XML data in transactions, purchase orders, invoices, etc. Growing need to store and manage all this data.

You don’t want to split your data manage two databases: IMS and an XML database. Especially, when this data is so tightly bound (could be impossible without serious replication or migration)



What is XML

IMS

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

```
<A>
  <f1>~~~~</f1>
  <f2>~~~~</f2>
  <f3>~~~~</f3>
  <B>
    <f4>~~~~</f4>
    <f5>~~~~</f5>
  </B>
  <B>
    <f4>~~~~</f4>
    <f5>~~~~</f5>
  </B>
</A>
```

© IBM Corporation 2003 IMS Technical Conference

Everyone should have some idea already about XML.

An XML Document is nothing more than a structured document. A means of separating data from presentation.

It is so successful because a) it is an agreed upon standard b) it handles encoding problems and byte ordering **c) it is easily parsable**

Why is XML…

IMS

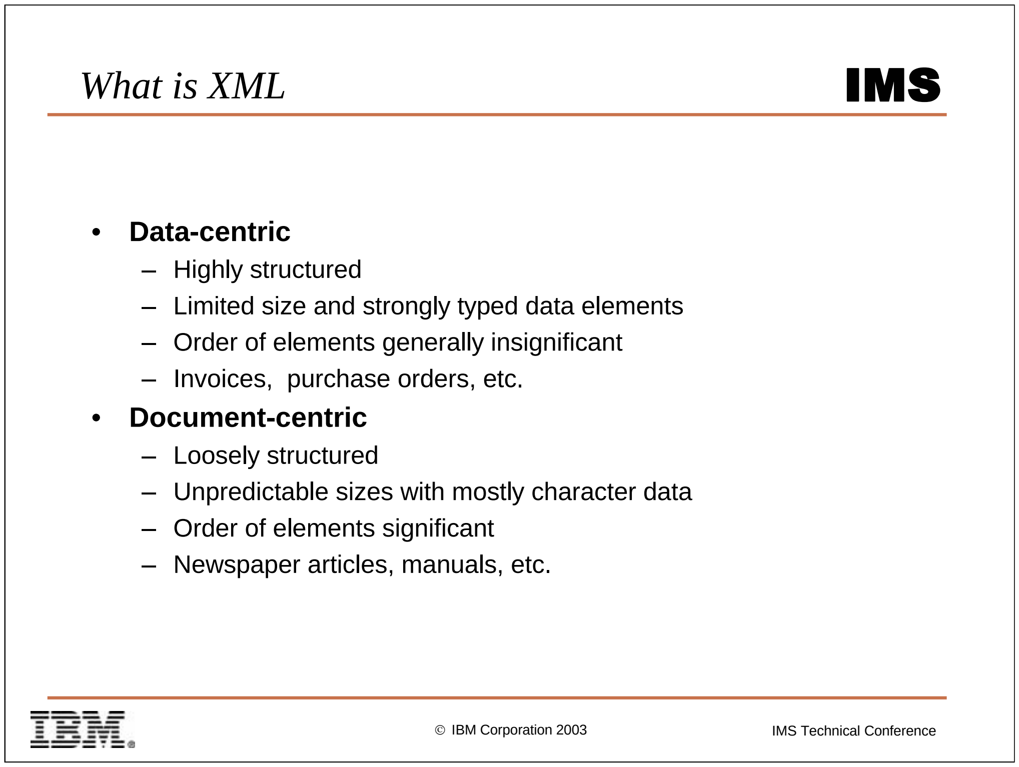
- **Standard Internet Data Exchange Format**
  - Handles encoding

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

  - Handles byte ordering

    <OrderNumber>**110203**</OrderNumber>

  - Human Legible?
  - Easily Parsed
  - Standard



What is XML

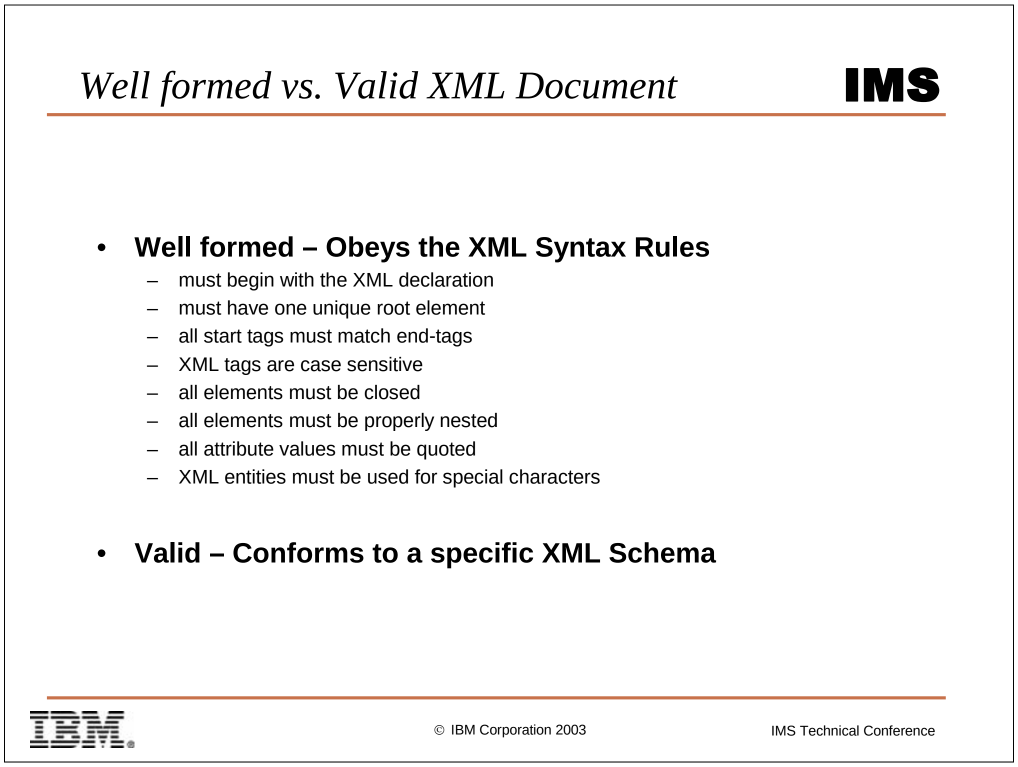
IMS

- **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.

We are going to hit on this data vs. document centric concept often.

Data-centric: invoices, purchase orders, parts listings,

Document-centric: newspaper articles, manuals, Shakespeare (all his plays have been converted to XML on the Web – like what HTML did for Lewis Carol)



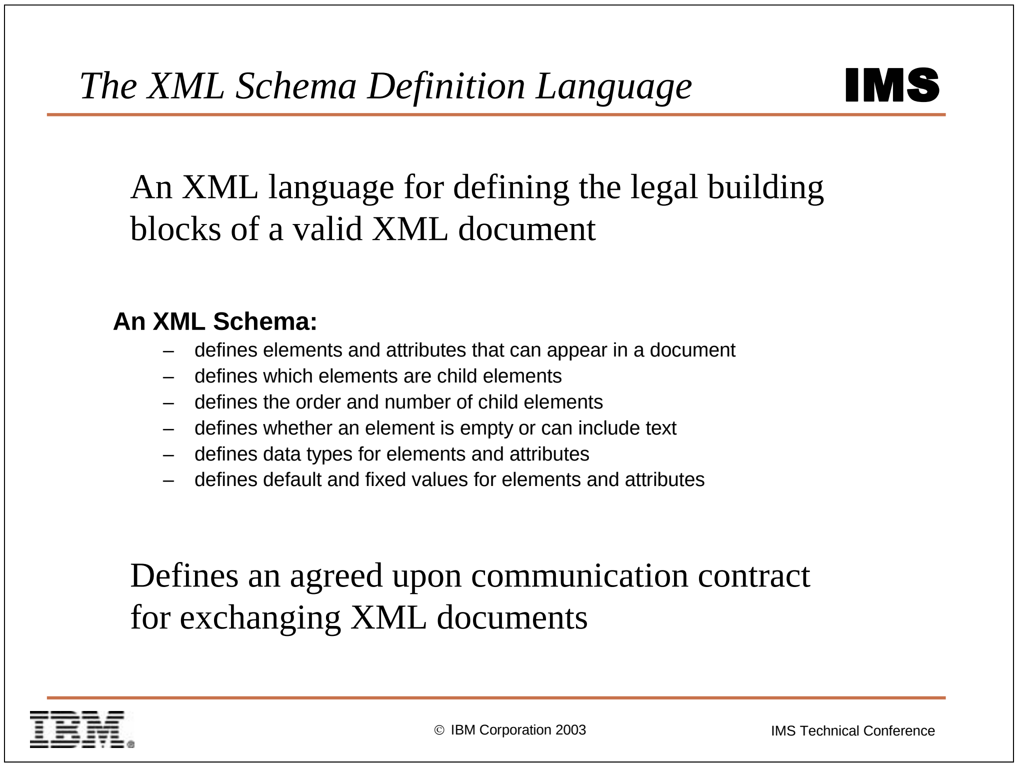
Well formed vs. Valid XML Document

IMS

- **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**

Well formed – follows the XML syntax rules (analogy: no more compiler errors – however, doesn't mean your program works)

Valid – Data matches the template (XML Schema) including structure, types, etc.



The XML Schema Definition Language IMS

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

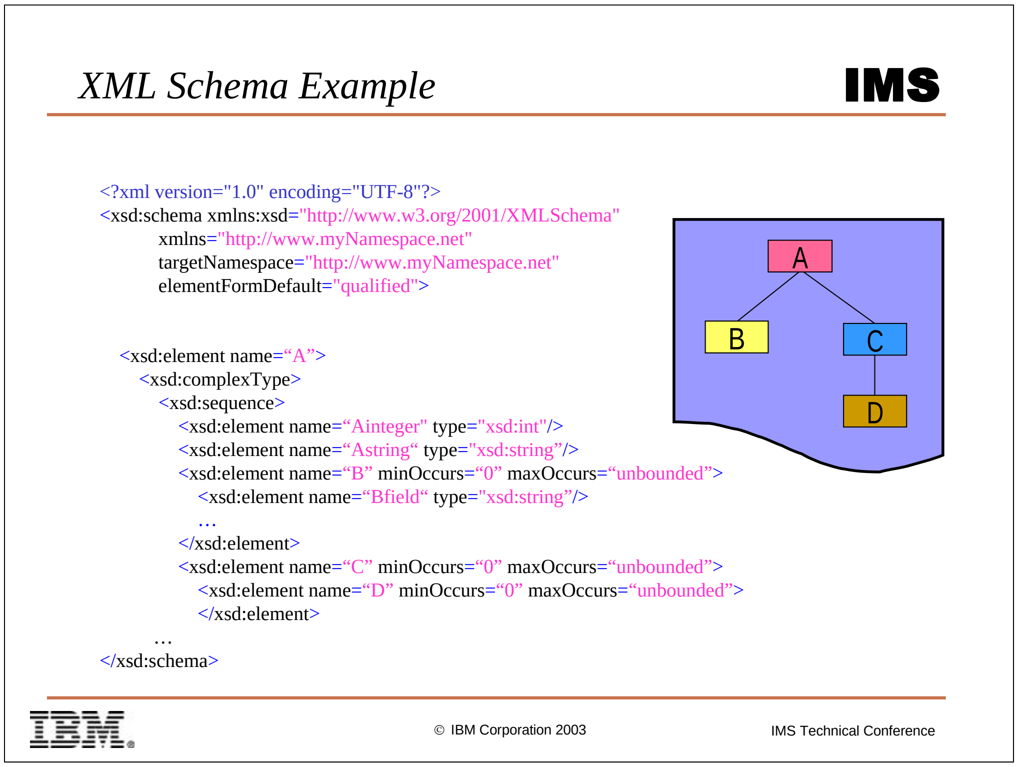
Defines an agreed upon communication contract for exchanging XML documents

It is the XML blueprints defining the full set of XML instance documents.

XML Schema's are themselves XML documents (therefore, there is an XML Schema that describes what an XML Schema can look like – and its like 4 pages long).

An XML Schema can be handed to a supplier or consumer saying "This is what I expect, or this is what you can expect from me"

XML Schema can be as specific or flexible as you could possible want (an XML Schema that only allows one possible XML document, an XML Schema that allows anything ("any") keyword).

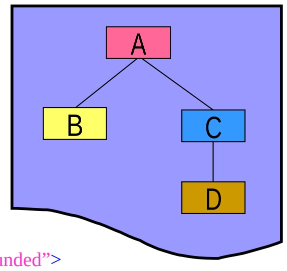


XML Schema Example

IMS

```
<?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="A">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Ainteger" type="xsd:int"/>
            <xsd:element name="Astring" type="xsd:string"/>
            <xsd:element name="B" minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="Bfield" type="xsd:string"/>
               …
            </xsd:element>
            <xsd:element name="C" minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="D" minOccurs="0" maxOccurs="unbounded">
               </xsd:element>
         …
</xsd:schema>
```



© IBM Corporation 2003 IMS Technical Conference

Things to point out:

XML header – required. Shows this is a well-formed XML document.

xmlns:xsd – designates the XML Schema Namespace

xmln – designates default Namespace. So we don't need to refer to its elements with a namespace tag.

targetNamespace – designates which Namespace we are defining a structure for.

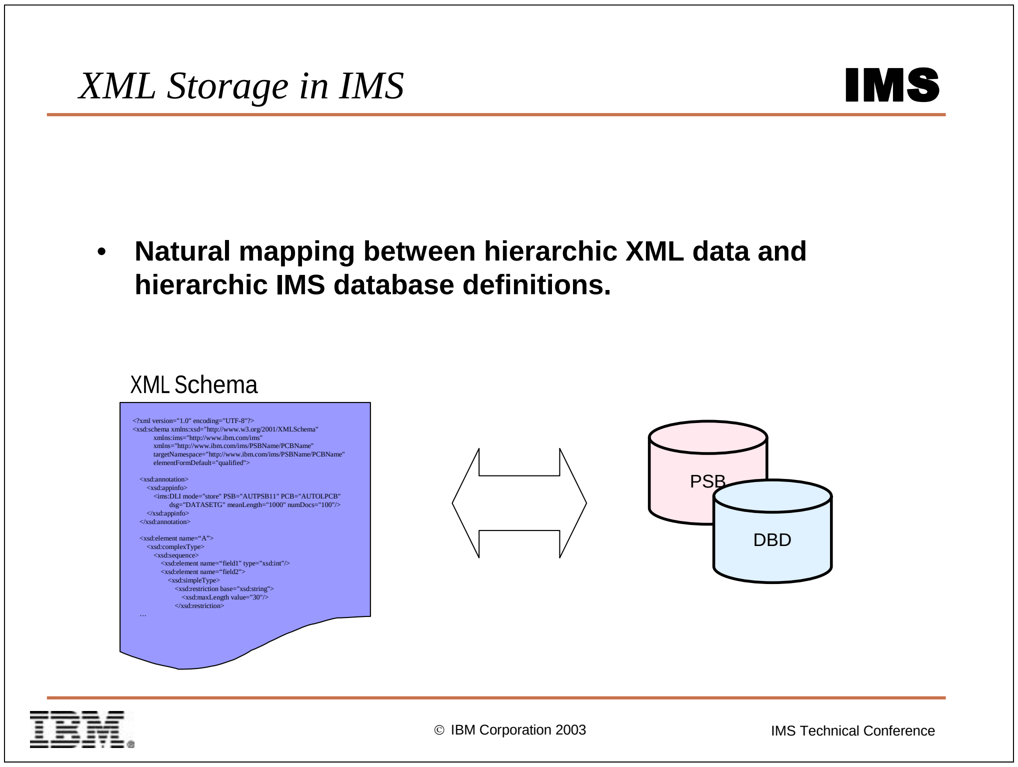
elementFormDefault – simply means everything in the instance doc needs to be qualified.

The A element is made up of a sequence of elements, including an int, a string (restricted to be 30 char max), and B, and C elements.

The C element has D element.

Particularly notice unbounded 1:n relationships vs. 1:1 relationships

Extra detail has been left off.



XML Storage in IMS

IMS

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

XML Schema

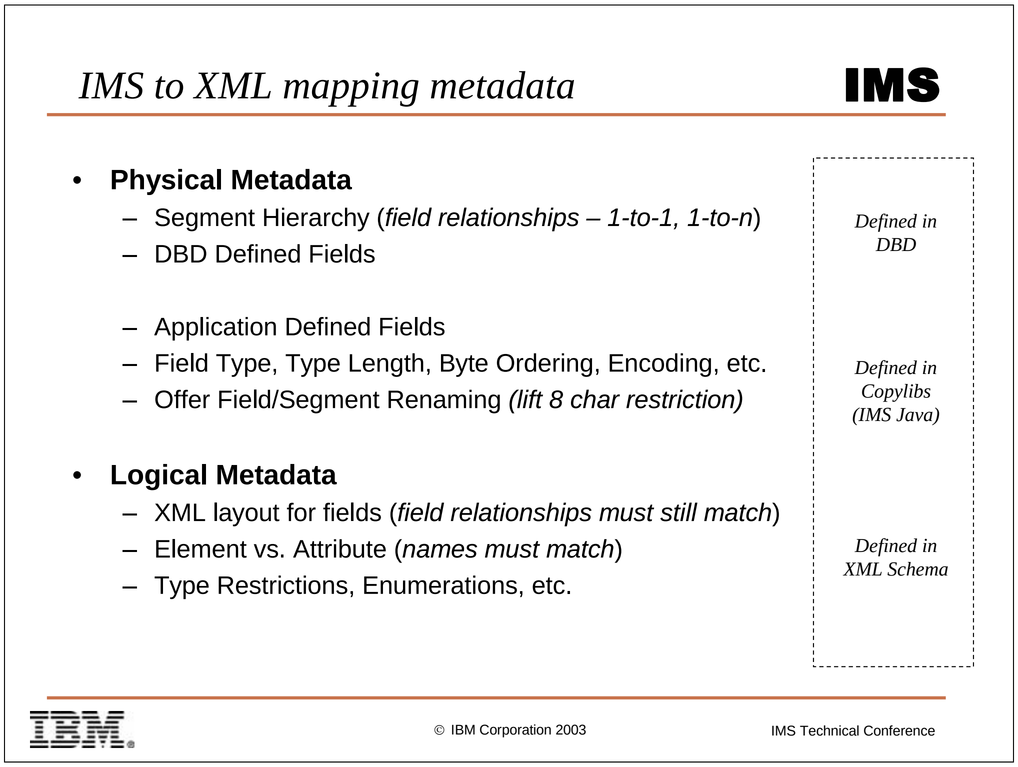
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:ims="http://www.ibm.com/ims"
          xmlns="http://www.ibm.com/ims/PSBName/PCBName"
          targetNamespace="http://www.ibm.com/ims/PSBName/PCBName"
          elementFormDefault="qualified">

   <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>
   …
```

© IBM Corporation 2003 IMS Technical Conference

Natural mapping (kind of – what about fairly generic XML Schemas? Especially document-centric XML)…but almost a no-brainer for fairly rigid XML Schema's, like good ole' data-centric. Go back and look at the XML Schema talk about how unbounded means another segment.



IMS to XML mapping metadata

IMS

- **Physical Metadata**
  - 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)*
- **Logical Metadata**
  - XML layout for fields (*field relationships must still match*)
  - Element vs. Attribute (*names must match*)
  - Type Restrictions, Enumerations, etc.

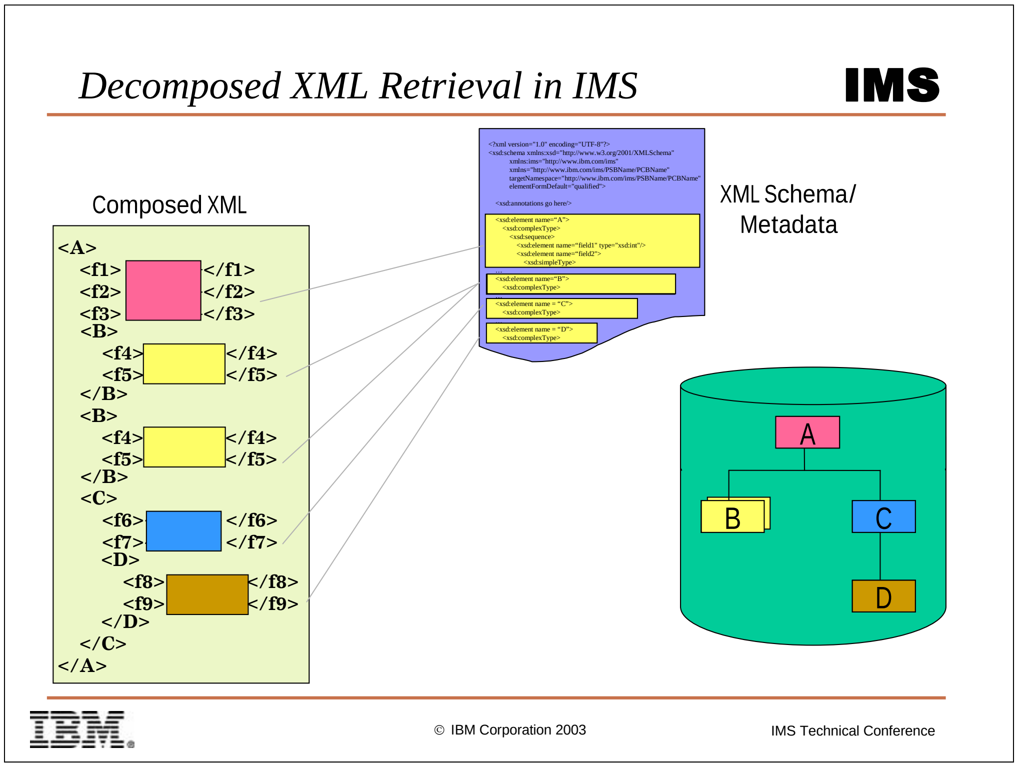
Defined in DBD

Defined in Copylibs (IMS Java)

Defined in XML Schema

IBM © IBM Corporation 2003 IMS Technical Conference

Physical Metadata – is often broken down into two groups (Type Layout, Type Length, byte ordering etc…would be for hardware metadata),



Decomposed XML Retrieval in IMS

IMS

Composed XML

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

XML Schema/ Metadata

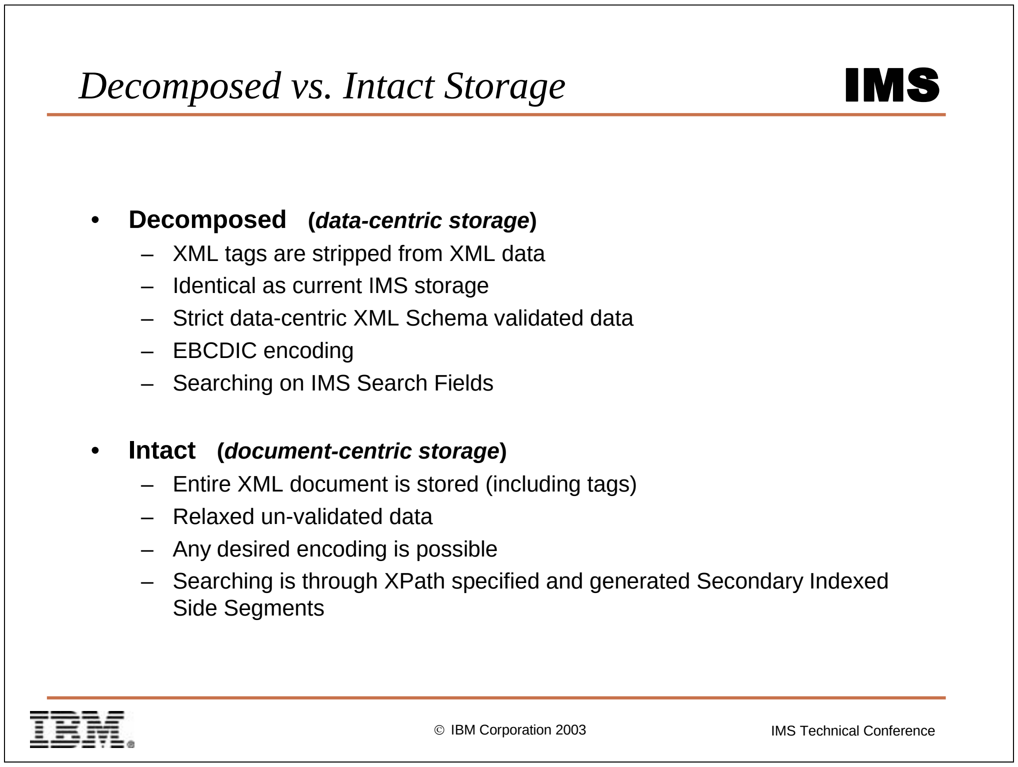
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:ims="http://www.ibm.com/ims"
        xmlns="http://www.ibm.com/ims/PSBName/PCBName"
        targetNamespace="http://www.ibm.com/ims/PSBName/PCBName"
        elementFormDefault="qualified">

   <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>
```

© IBM Corporation 2003

IMS Technical Conference

Show how XML documents are created from traditional data.

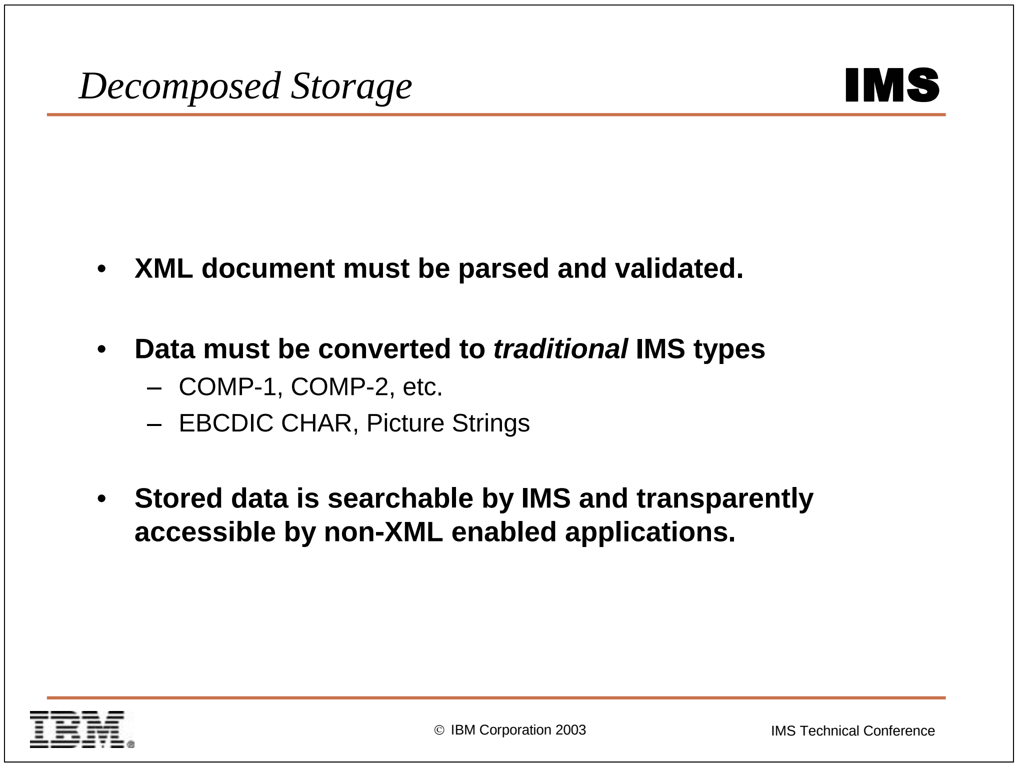


Decomposed vs. Intact Storage

IMS

- **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

Two different storage types for two different types of XML Schemas (notice we say two different types of XML Schemas and not two different types of XML documents – you may have extremely rigid and narrow data-centric XML documents, but all we have to go off of is the XML Schema…so if the XML Schema allows a lot of flexibility we have to allow for it).



Decomposed Storage

IMS

- **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.**

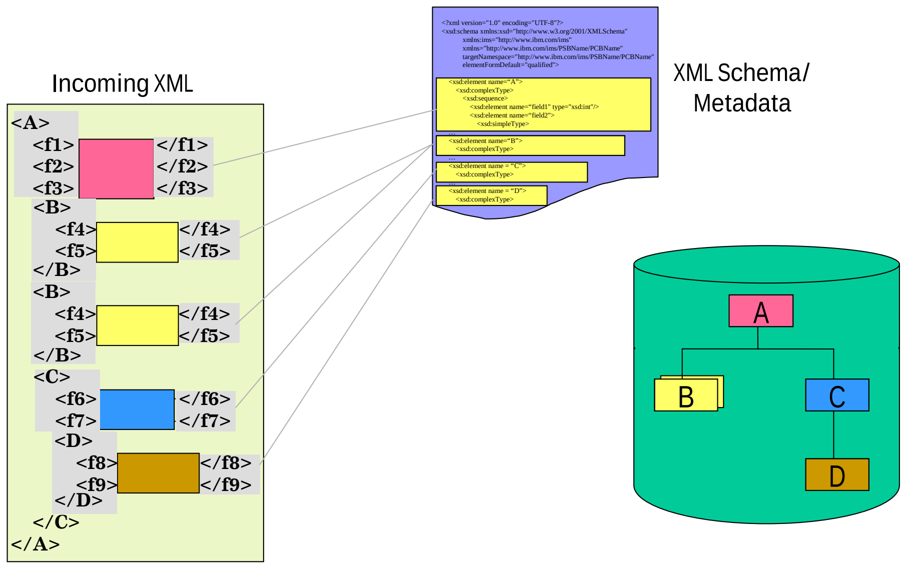
Parsing and especially validation is slow – in inverse implies recomposition is also slow.

Encoding and type conversions are slow.

Huge plus – the data is easily searchable and accessible to legacy applications. Both XML and legacy, non-XML can play together

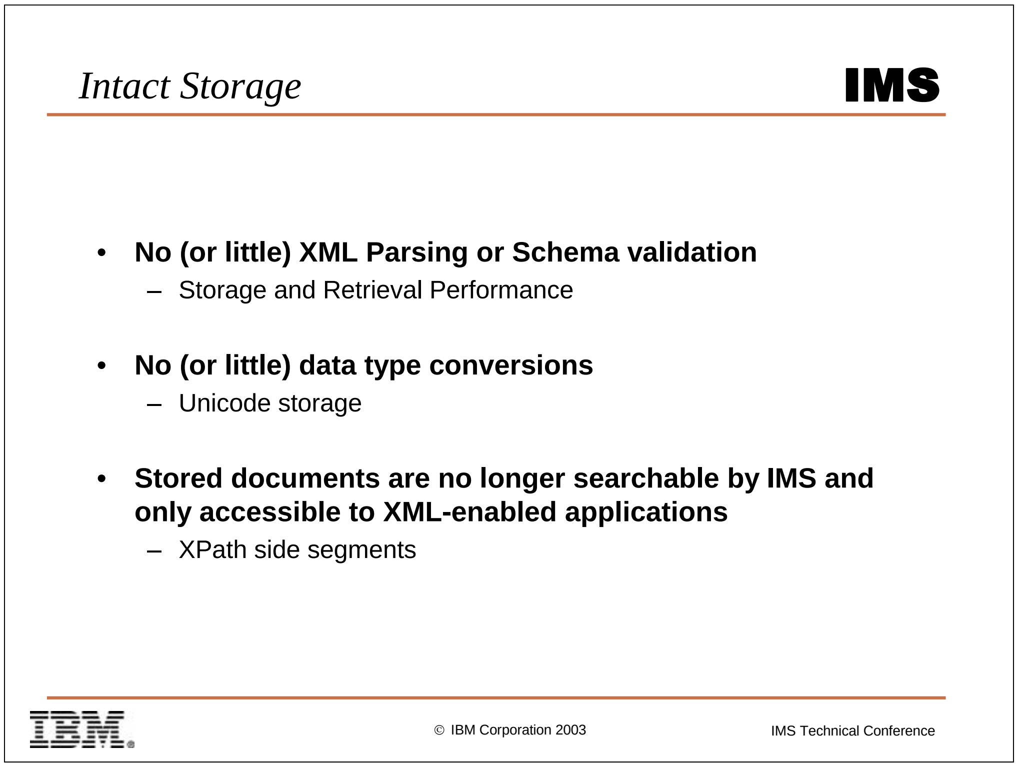
Decomposed XML Storage in IMS

IMS



```
<A>
  <f1>  </f1>
  <f2>  </f2>
  <f3>  </f3>
  <B>
    <f4>  </f4>
    <f5>  </f5>
  </B>
  <B>
    <f4>  </f4>
    <f5>  </f5>
  </B>
  <C>
    <f6>  </f6>
    <f7>  </f7>
    <D>
      <f8>  </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: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>
```



Intact Storage

IMS

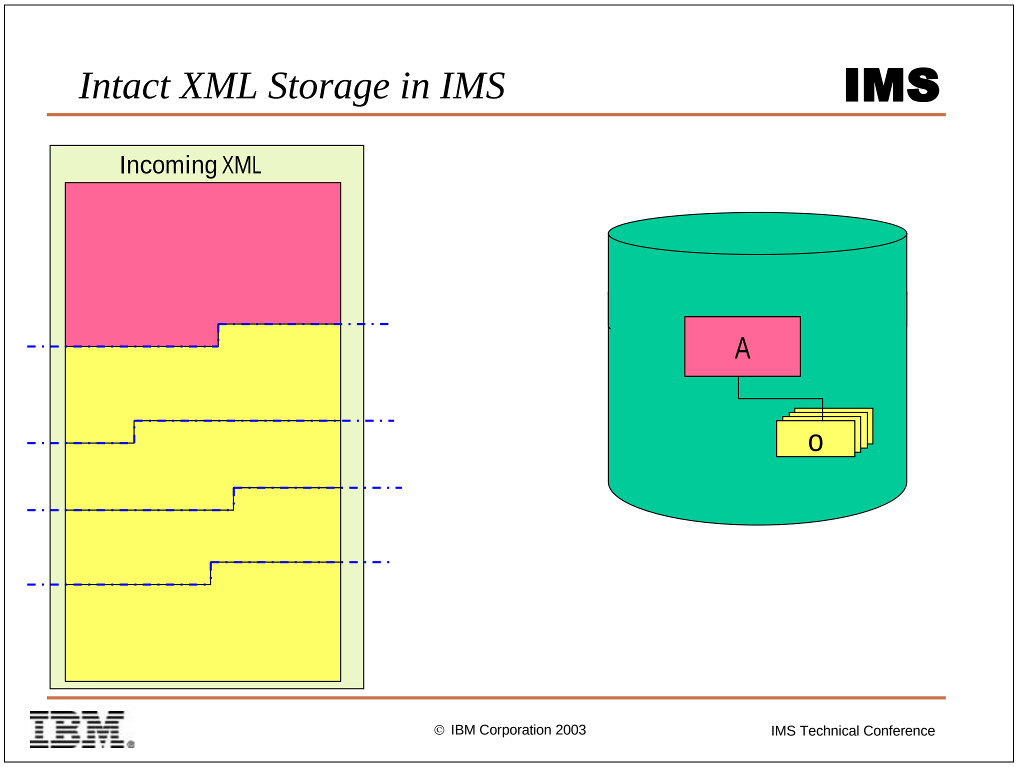
- **No (or little) XML Parsing or Schema validation**
  - Storage and Retrieval Performance
- **No (or little) data type conversions**
  - Unicode storage
- **Stored documents are no longer searchable by IMS and only accessible to XML-enabled applications**
  - XPath side segments

Both the top two mean better storage and retrieval.

Bottom one is natural consequence

The "(or little)" refer to possible XPath side segments.

We will discuss these secondary indexed side segments later.



Intact XML Storage in IMS

IMS

© IBM Corporation 2003

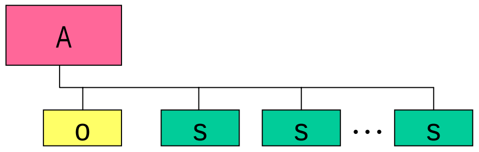
IMS Technical Conference

Although we don’t need the XML Schema for validation (optionally) we do need it to indicate the document is stored intact.

Intact Storage Secondary Indexing

IMS

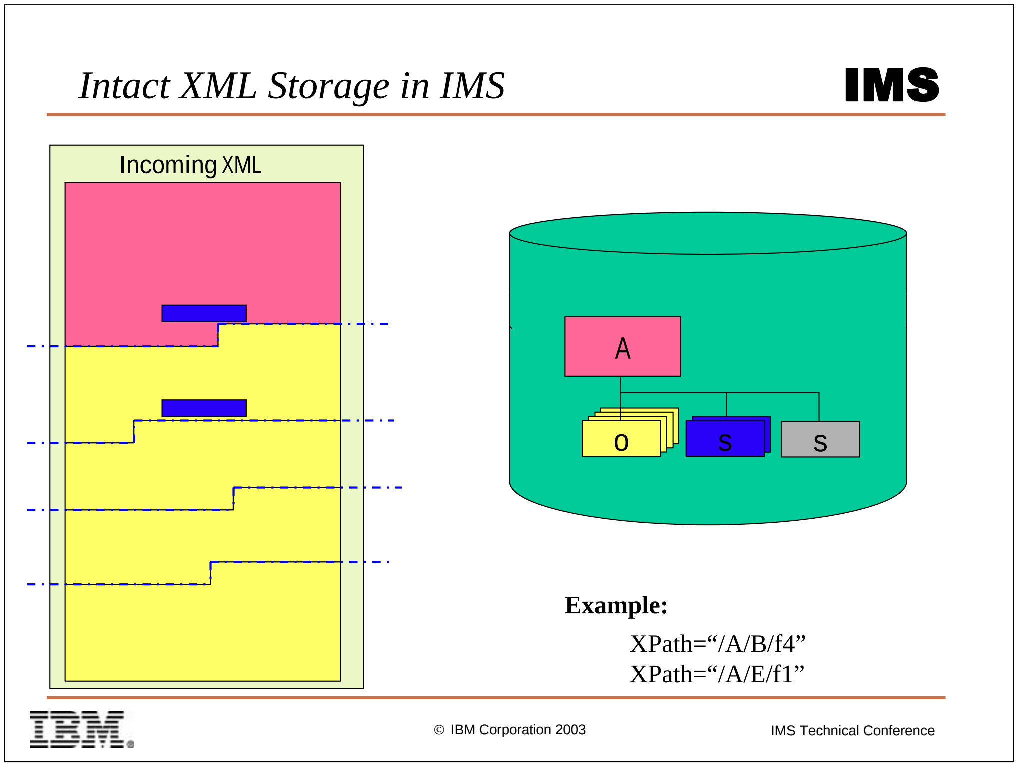
- **XPath expression identifying Side Segments**
  - Side segment is converted to *traditional* data type and copied into segment.
- **Side Segments are secondary indexed with documents root as target.**



Example:

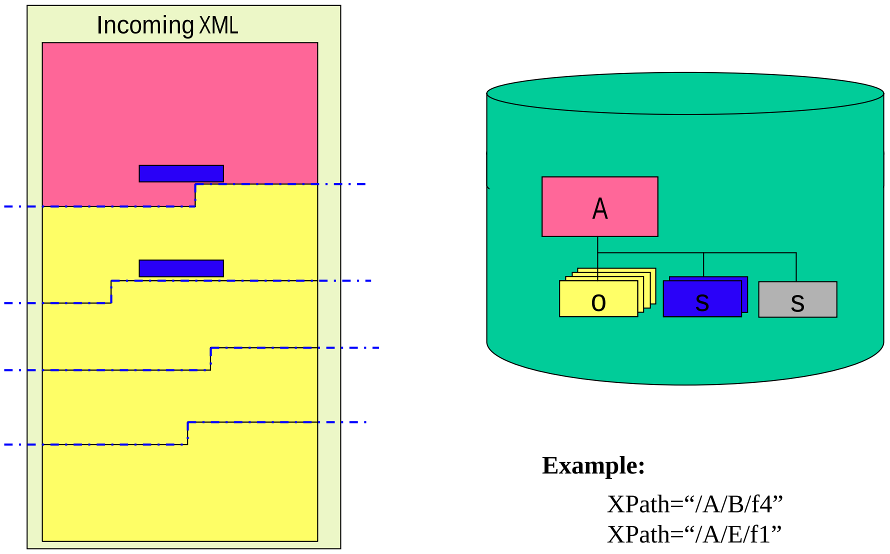
XPath="/Dealer/DealerName"

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



Intact XML Storage in IMS

IMS



Example:

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

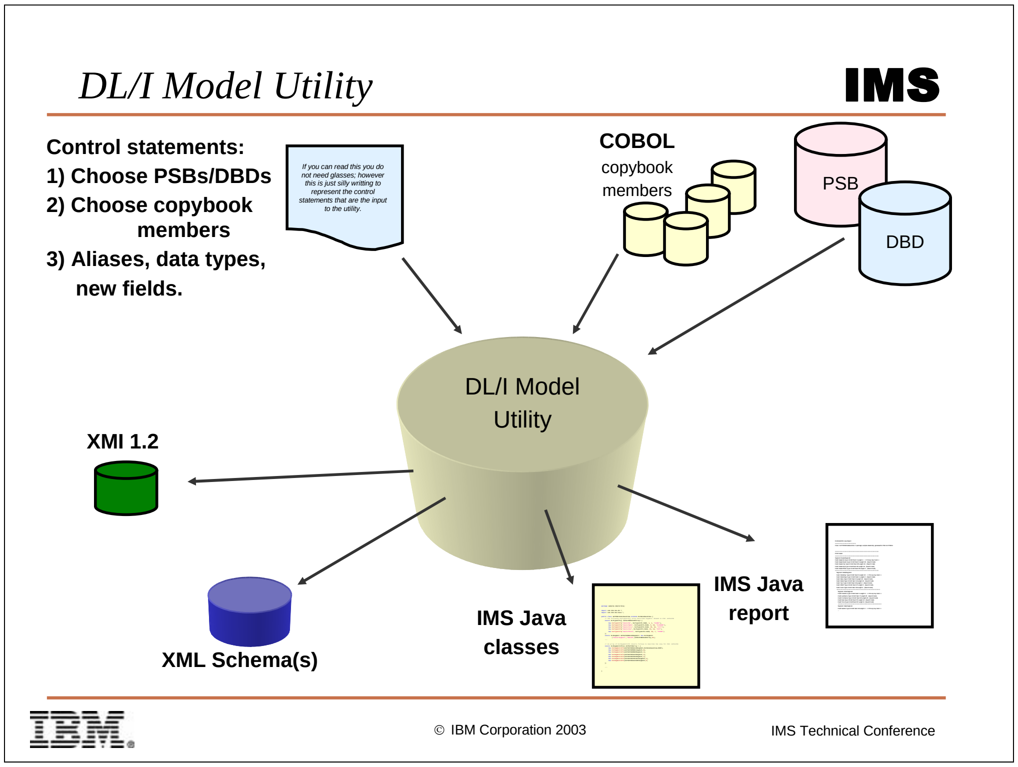
Although we don't need the XML Schema for validation (optionally) we do need it to indicate the document is stored intact.

When using Side Segments for XML indexing, we need a Schema to at least validate the XPath expressions used for indexing.

Overview

IMS

- **Introduction**
  - What is XMS?
  - What is XML?
  - What are XML Schemas?
- **XMS Methodology**
  - Decomposed Storage
  - Intact Storage
- **XMS Tooling**
  - Metadata Generation
- **XMS Java Implementation**
  - SQL UDF Interface
  - Future



DL/I Model Utility

IMS

Control statements:

1) Choose PSBs/DBDs

2) Choose copybook members

3) Aliases, data types, new fields.

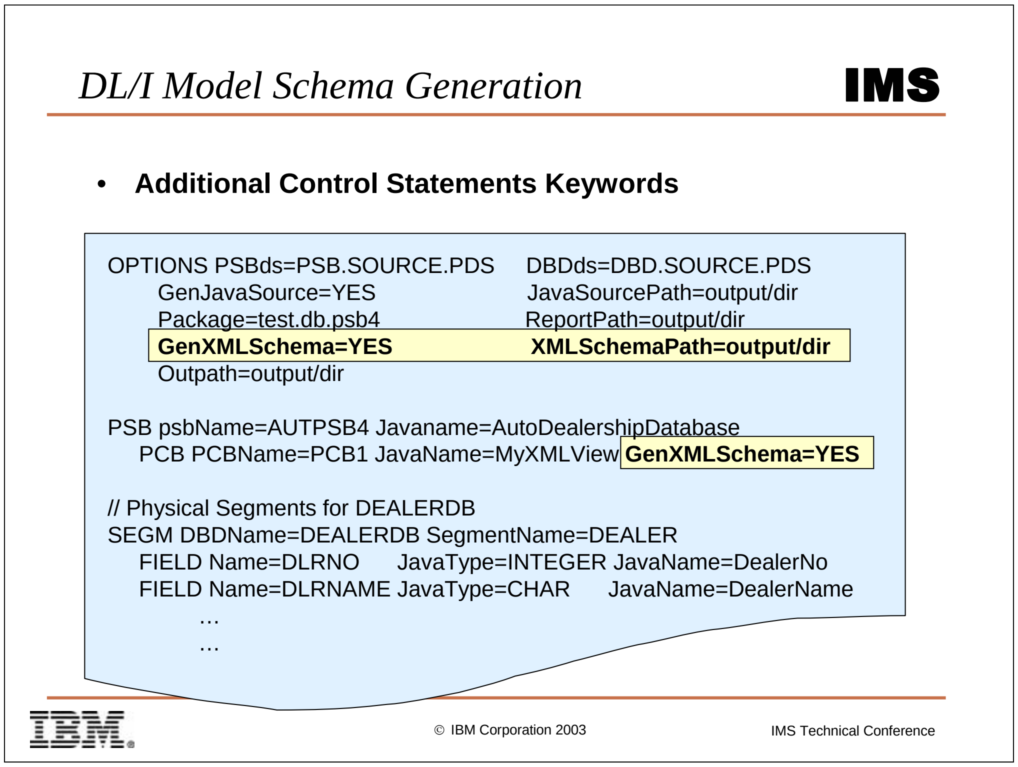
If you can read this you do not need glasses; however this is just silly writting to represent the control statements that are the input to the utility.
COBOL
copybook members
PSB
DBD
DL/I Model Utility
XMI 1.2
XML Schema(s)
IMS Java classes
IMS Java report

© IBM Corporation 2003

IMS Technical Conference

XMI is the future of the physical DL/I metadata.

The XML Schema is the structural XML / DL/I metadata mapping



DL/I Model Schema Generation

IMS

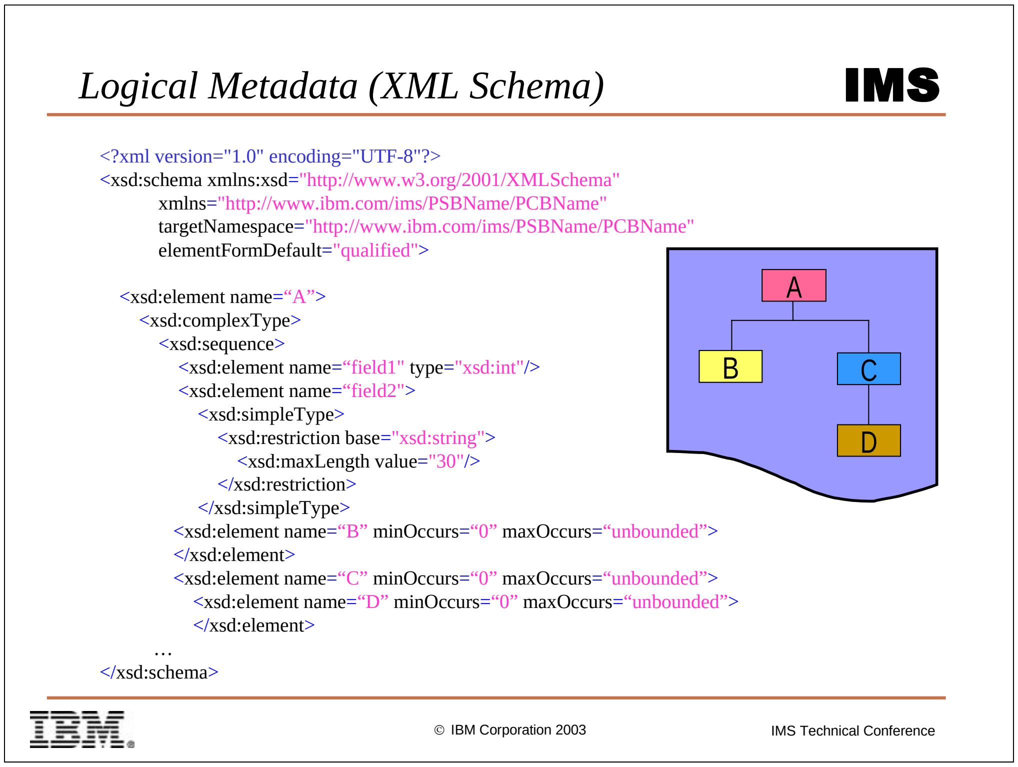
- **Additional Control Statements Keywords**

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

PSB psbName=AUTPSB4 Javaname=AutoDealershipDatabase
    PCB PCBName=PCB1 JavaName=MyXMLView GenXMLSchema=YES

// Physical Segments for DEALERDB
SEGM DBDName=DEALERDB SegmentName=DEALER
    FIELD Name=DLRNO      JavaType=INTEGER JavaName=DealerNo
    FIELD Name=DLRNAME JavaType=CHAR      JavaName=DealerName
            …
            …
```

IBM

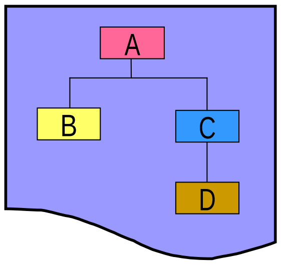


Logical Metadata (XML Schema)

IMS

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

   <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>
               </xsd:simpleType>
           <xsd:element name="B" minOccurs="0" maxOccurs="unbounded">
           </xsd:element>
           <xsd:element name="C" minOccurs="0" maxOccurs="unbounded">
              <xsd:element name="D" minOccurs="0" maxOccurs="unbounded">
              </xsd:element>
        …
</xsd:schema>
```



© IBM Corporation 2003 IMS Technical Conference

Things to point out:

XML header – required. Shows this is a well-formed XML document.

xmlns:xsd – designates the XML Schema Namespace

xmln – designates default Namespace. So we don't need to refer to its elements with a namespace tag.

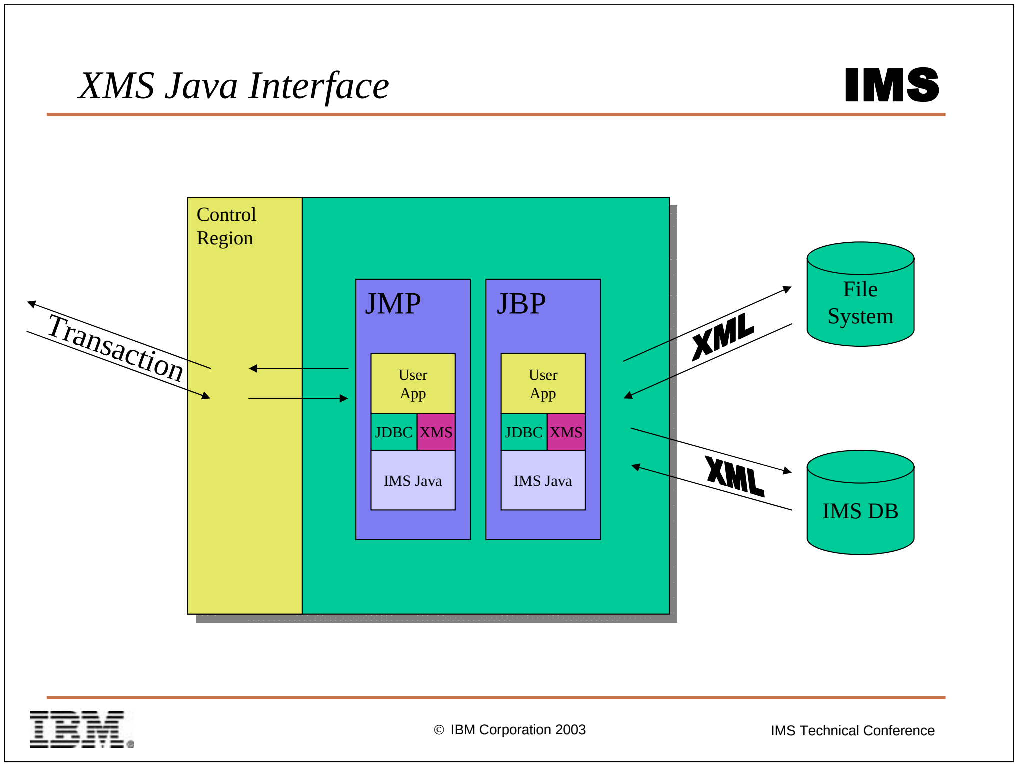
targetNamespace – Target Namespace is PSB and PCB (is unique per IMS)

elementFormDefault – simply means everything in the instance doc needs to be qualified.

The A element is made up of a sequence of elements, including an int, a string (restricted to be 30 char max), and B, and C elements.

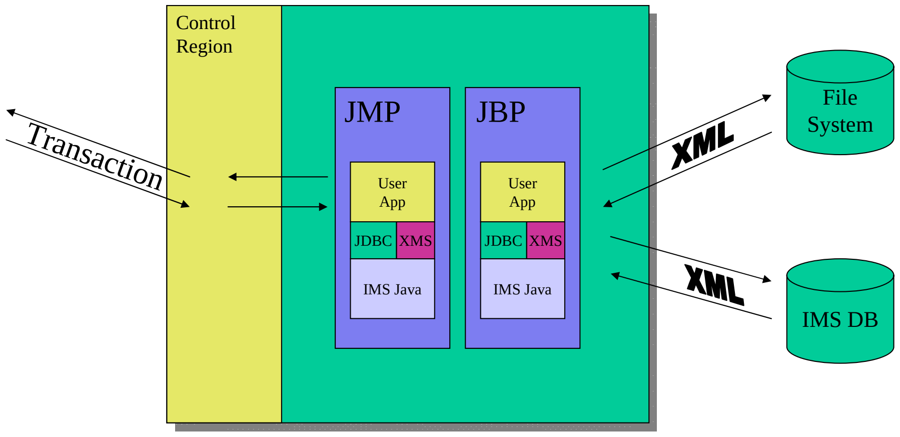
The C element has D element.

Extra detail has been left off.



XMS Java Interface

IMS



© IBM Corporation 2003

IMS Technical Conference

This is the applications point of view that it is now storing and retrieving XML documents from the Database and File system.

XMS Java Interface

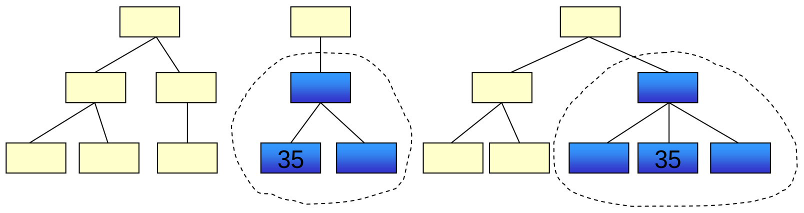
IMS

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

RetrieveXML() UDF

IMS

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

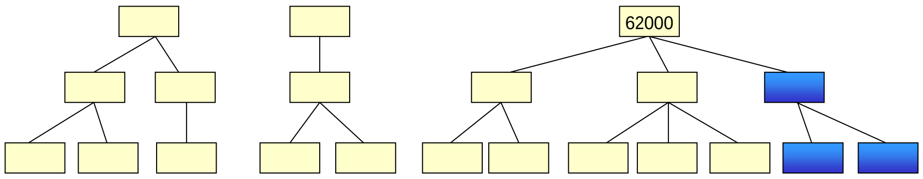


**Two Rows of XML CLOBs in the ResultSet*

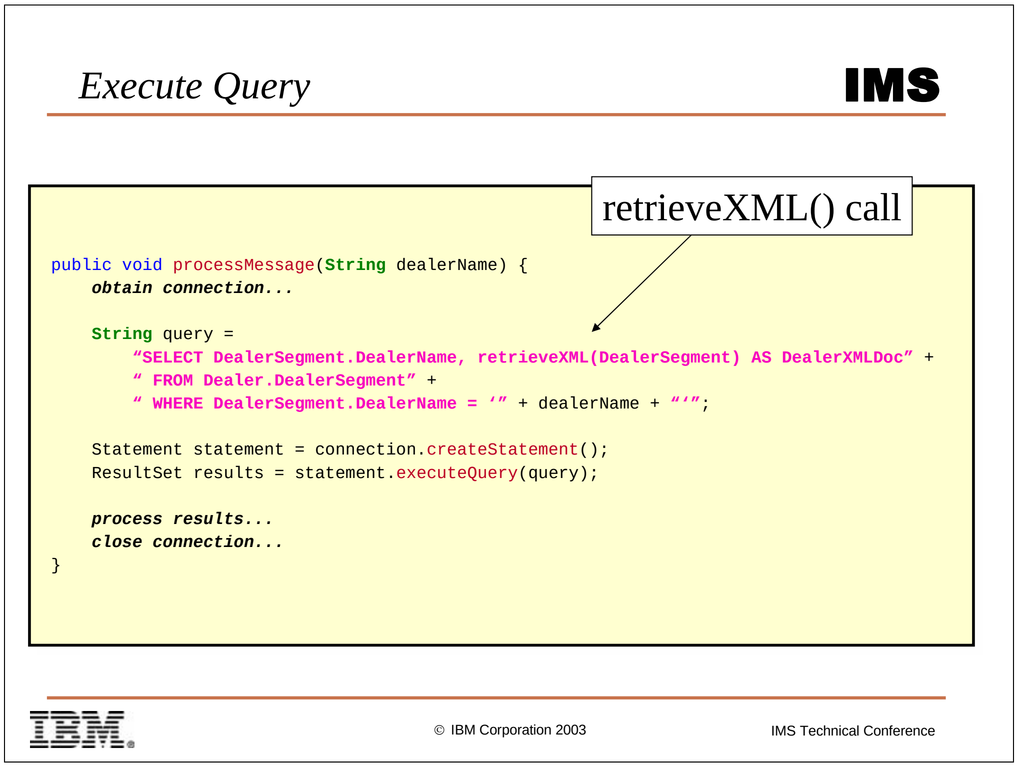
StoreXML() UDF

IMS

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



**Insert Statement must be a Prepared Statement*



Execute Query

IMS

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...
}
```

© IBM Corporation 2003

IMS Technical Conference

We handle the front end the same as always per environment

We create a connection the same way (either Managed or Non-managed)

But…we issue a new SQL and process the results differently.

Process Results

IMS

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

IMS

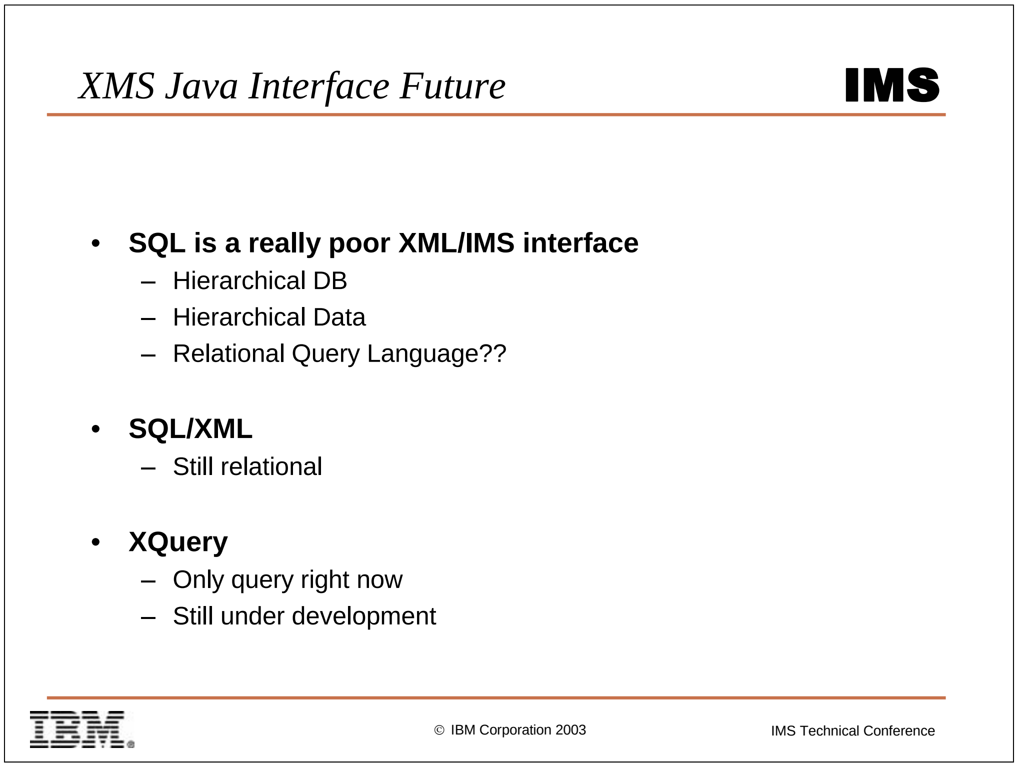
getCharacterStream() or
getAsciiStream()

```
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();
}
```



XMS Java Interface Future

IMS

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

© IBM Corporation 2003 IMS Technical Conference

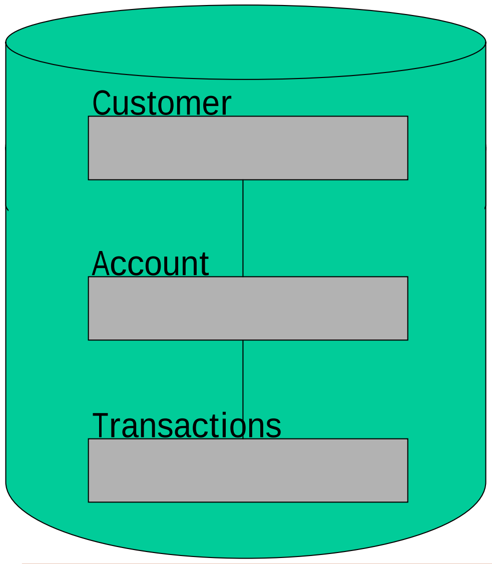
Clearly this is bad….but it was fast and makes this available now rather than later.

SQL/XML stems from the inability to map XML queries (initially XQuery) directly onto relational, so it started its own track. Its fairly well developed and allows you to dynamically build an XML document out of the underlying data (notice it assumes the underlying data is not already XML)

XQuery is still under development and not a completed standard, however it is more the direction of XMS for the future, and we (IMS) are involved in its review (have access to team room and discussions, since it is mostly being developed here at IBM).

Hypothetical Bank DB

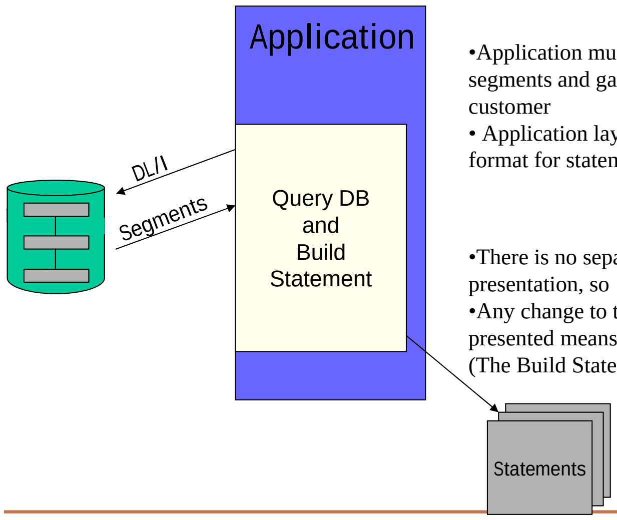
IMS



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

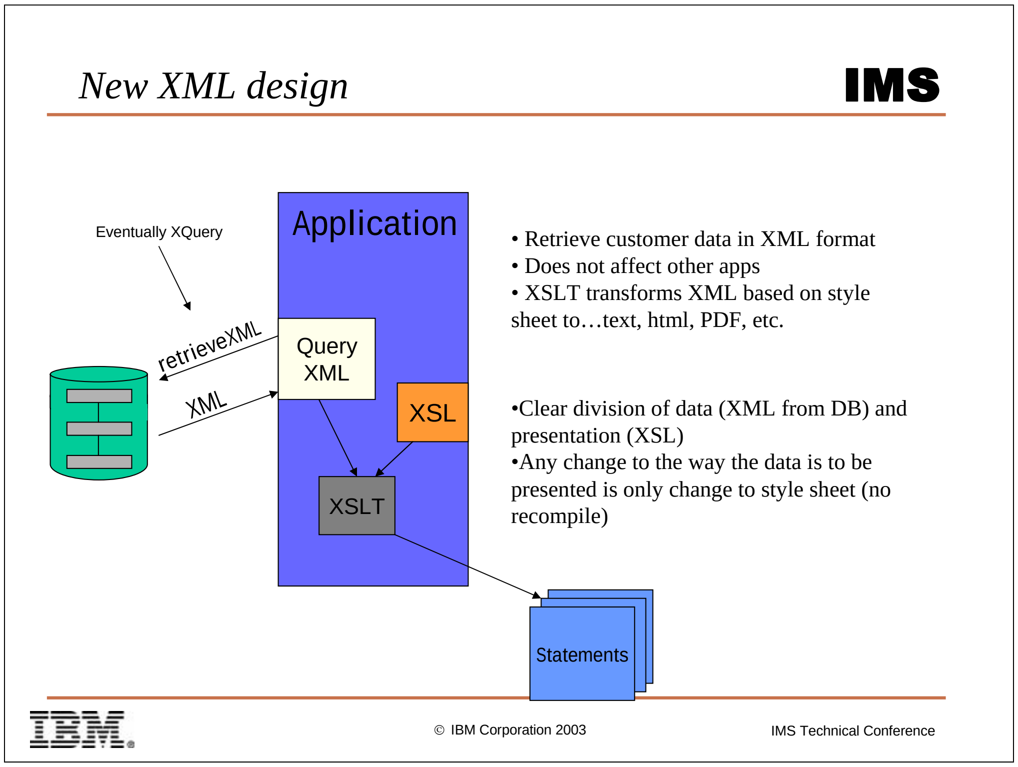
Current tedious design

IMS



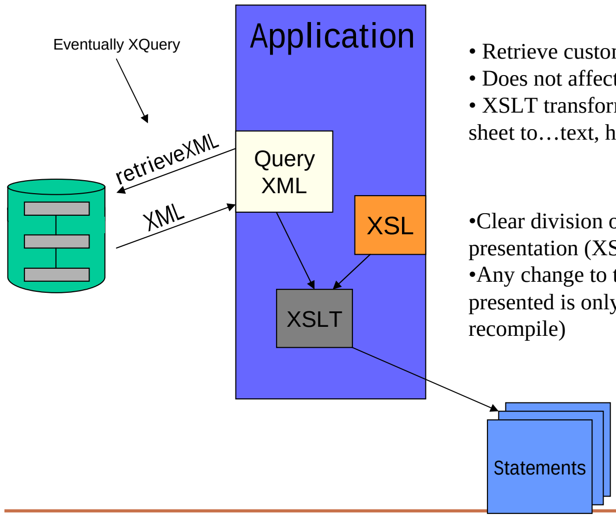
•Application must query all needed DL/I segments and gather needed data for each customer
• Application lays out data in desired output format for statement or web page.

•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

IMS

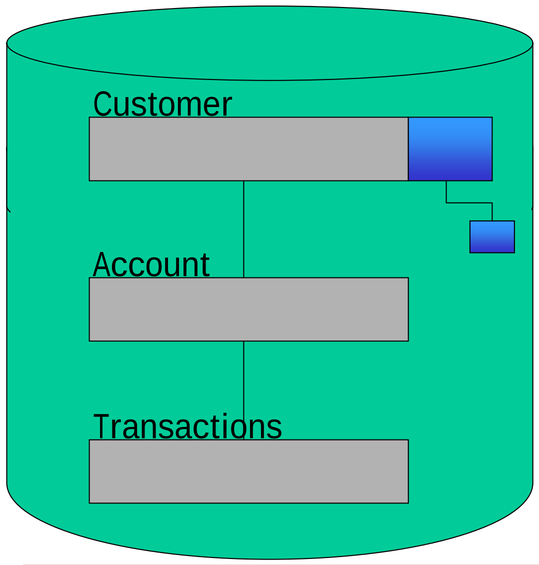


• Retrieve customer data in XML format
• Does not affect other apps
• XSLT transforms XML based on style sheet to…text, html, PDF, etc.

•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

IMS



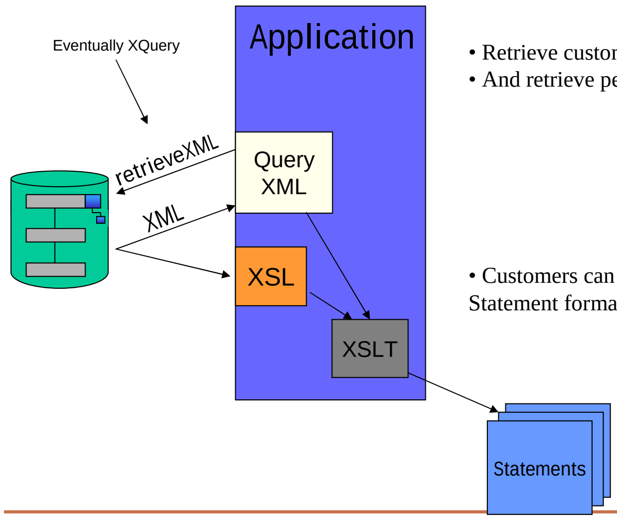
- **Extend Customer with intact XML for each customers own personalized style sheet**

© IBM Corporation 2003

IMS Technical Conference

New XML design

IMS

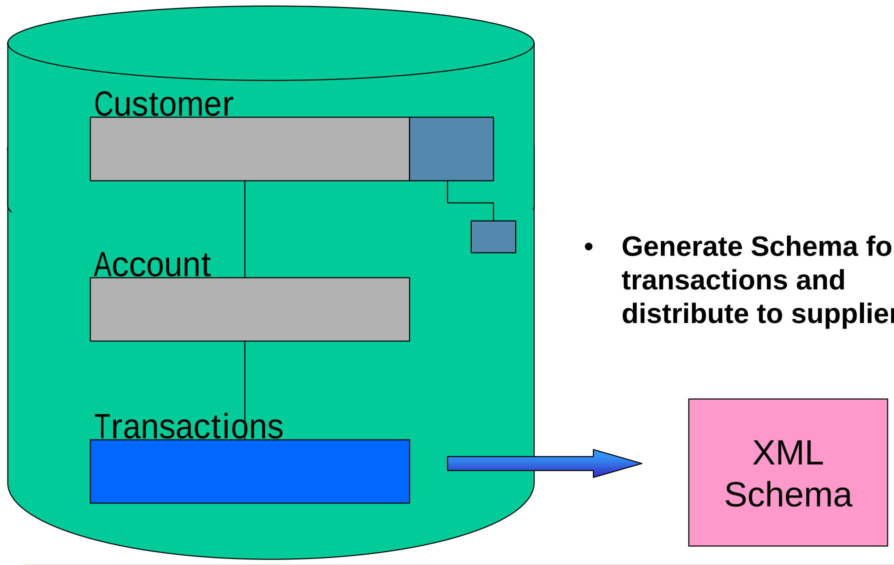


• Retrieve customer data in XML format
• And retrieve personalized XSL

• Customers can change their own Bank Statement format

Possible further extension

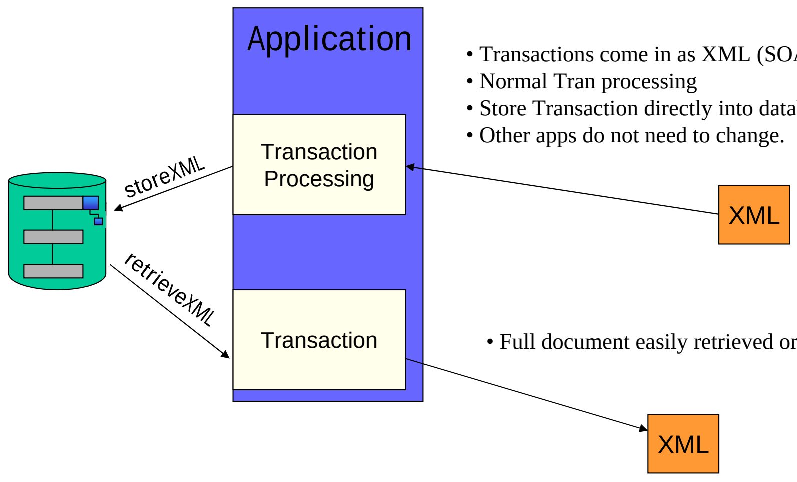
IMS



- **Generate Schema for transactions and distribute to suppliers.**

New XML storage and retrieval

IMS



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

- Full document easily retrieved or searched

End

IMS