UNIVERSITÄT LEIPZIG

# Mainframe Internet Integration

**Prof. Dr. Martin Bogdan**
**Prof. Dr.-Ing. Wilhelm G. Spruth**

**SS2013**

## Service Oriented Architecture Teil 1

## XML und DB2

# XML
# Extensible Markup Language

**Die Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache“), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird bevorzugt für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell über das Internet.**

**Die vom World Wide Web Consortium (W3C) herausgegebene XML-Spezifikation definiert eine Metasprache, auf deren Basis durch strukturelle und inhaltliche Einschränkungen Anwendungs-spezifische Dialekte definiert werden können. Diese Einschränkungen werden durch zwei alternative Schemasprachen (entweder DTD oder XML-Schema (XSD)) ausgedrückt. Bei der Verarbeitung von Textdokumenten wird meistens DTD eingesetzt. Für die Datenverarbeitung, besonders für Web Services, wird in der Regel XSD benutzt um den verwendeten XML Dialekt zu beschreiben.**

**Ein XML-Dokument besteht aus Textzeichen, in der Regel im ASCII-Code oder Unicode (meist UTF-8) , und ist damit lesbar durch einen menschlichen Benutzer. Binärdaten enthält es per Definition nicht.**

# HTML und XML

**HTML wurde entworfen, um Daten visuell in einem Web Browser darzustellen. Für ein Anwendungsprogramm würde es meistens schwierig sein, allgemein Daten aus einer HTML Seite zur Weiterberarbeitung zu extrahieren, da es nicht weis, wie die Daten in der HTML Seite strukturiert sind.**

**Bei XML besteht eine absolute Trennung von Inhalt und Formatierung. Die Tags beschreiben den Inhalt, so dass geeignete Programme den Inhalt entsprechend den Vorschriften verarbeiten können, die dem Tag irgendwie zugeordnet werden. XML-Dokumente sind nicht dazu bestimmt, direkt ausgegeben zu werden, sondern werden mit geeigneten Programmen weiterverarbeitet.**

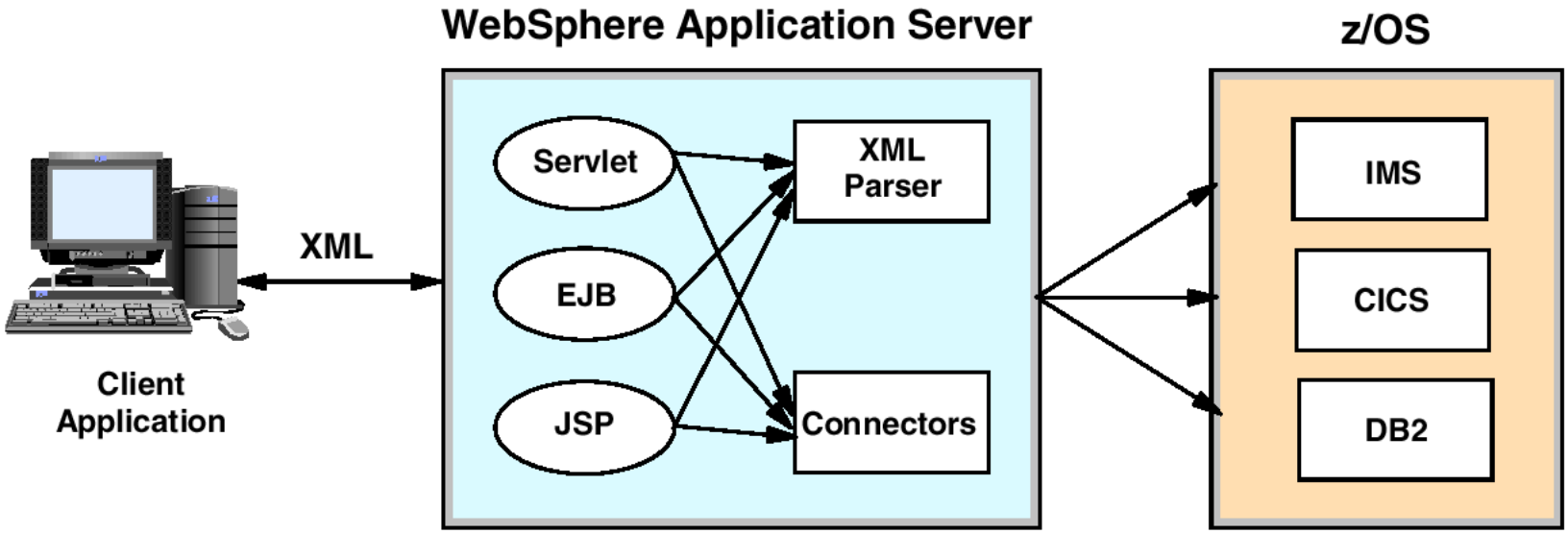
**Da die XML-Tags inhaltlich definiert sind, können diese Programme zur Weiterverarbeitung ganz unterschiedliche Themen betreffen:**

- **Formatierung und Ausgabe als HTML** **z.B. suchen in Such-Maschinen**
- **Eintrag in Datenbanken** **z.B. Output mit maschineller Verarbeitung**
- **inhaltlich korrekte Archivierung** **z.B. Umformatieren auf beliebige Datenformate**

**XML kann weiterhin semantische Information in ein Dokument einbauen.**

**Was ist Semantische information (or meaning)? Nehmen Sie als Beispiel eine Buchbeschreibung in www.amazon.com. Ein Leser kann entscheiden, ob der Ausdruck “Braun” sich auf die Farbe des Einbandes oder den Namen des Verfassers bezieht. Eine HTML-basierte Web Search Engine kann das nicht, weil für diese Unterscheidung nicht genügend semantische Information in der HTML Seite vorhanden ist.**

**Eine spezielle XML Version, XHTML, vereinigt die Möglichkeiten von HTML und XML.**




# Client/Server Operation mit XML

**Die Client Anwendung übermittelt Daten im XML Format an den WebSphere Applicationserver. Die Daten können von einer Serverkomponente wie einem Servlet, einer Java Server Page (JSP), einer Java Bean oder einer Enterprise Java Bean (EJB) verarbeitet werden. Ein "XML Parser" extrahiert die relevante Information von der empfangenen XML Nachricht.**

**Als Ergebnis der Parsing Operation kann z.B. eine IMS Message, eine CICS COMMAREA, oder eine JDBC Request entstehen, die an die Business Logik des entsprechenden Backend Systems weitergeleitet wird.**


# Eine Standard Interface zwischen unterschiedlichen Systemen

**XML kann Anwendungen voneinander isolieren. Es stellt ein universelles Datenformat bereit, welches für Input und/oder Output zwischen beliebigen Anwendungsprogrammen benutzt werden kann. Da XML Daten self-defining sind, erhält das empfangene Anwendungsprogramm alle Information die es braucht, um die übermittelten Daten zu verarbeiten, ohne Wissen über das sendende Anwendungsprogramm. Es ist damit eine Alternative zu der Interface Definition Language (IDL) im klassischen oder im CORBA RPC.**

**Der Hauptvorteil von XML ist, dass es den Kontext mit den Daten überträgt.**

**Damit ist XML gut geeignet für die Integration von Geschäftsprozessen und deren Anwendungen innerhalb eines Unternehmens.**

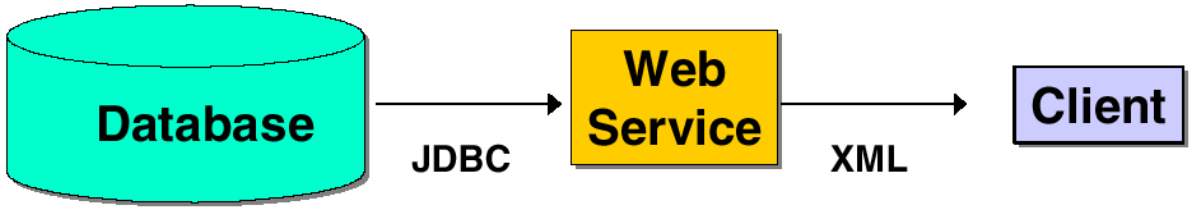
# XSLT

**Mit Hilfe von entweder DTD oder (häufiger) XML-Schema (XSD) können unterschiedliche XML Dialekte für spezielle Anwendungen definiert werden.**

**XSL Transformation, kurz XSLT, ist eine Programmiersprache zur Transformation von XML-Dokumenten. Hiermit ist eine Übersetzung von einem mittels DTD oder XSD definierten XML Dialekt in einen anderen Dialekt möglich.**

**XSLT baut auf der logischen Baumstruktur eines XML-Dokumentes auf und dient zur Definition von Umwandlungsregeln. XSLT-Programme, sogenannte XSLT-Stylesheets, sind dabei selbst nach den Regeln des XML-Standards aufgebaut.**

**Die Stylesheets werden von spezieller Software, den XSLT-Prozessoren, eingelesen, die mit diesen Anweisungen ein oder mehrere XML-Dokumente in das gewünschte Ausgabeformat umwandeln.**




**Es existieren mehrere Gründe, XML mit einer Datenbank (z.B. DB2 oder IMS) zu benutzen. Besonders gebräuchlich ist es, auf Daten im XML Format zuzugreifen, die in einer existierenden Datenbank untergebracht sind. Nehmen wir z.B. an, eine Datenbank enthält information über Aktienkurse. Ein Web Service könnte auf Anfrage den derzeitigen Aktionkurs als ein XML Dokument wiedergegeben.**


**Ähnlich kann ein Klientenprogramm Daten in der Form eines XML Dokuments an eine Datenbank übertragen. Ein Versicherungsvertreter könnte die Daten für eine neue Versicherung in der Form eines XML Dokumentes eingeben. Innerhalb des Versicherungsunternehmens würde ein Anwendungsprogramm die Daten aus dem XML Dokument extrahieren und in einer Datenbank (z.B. DB2 oder IMS) speichern.**

# XML und DB2

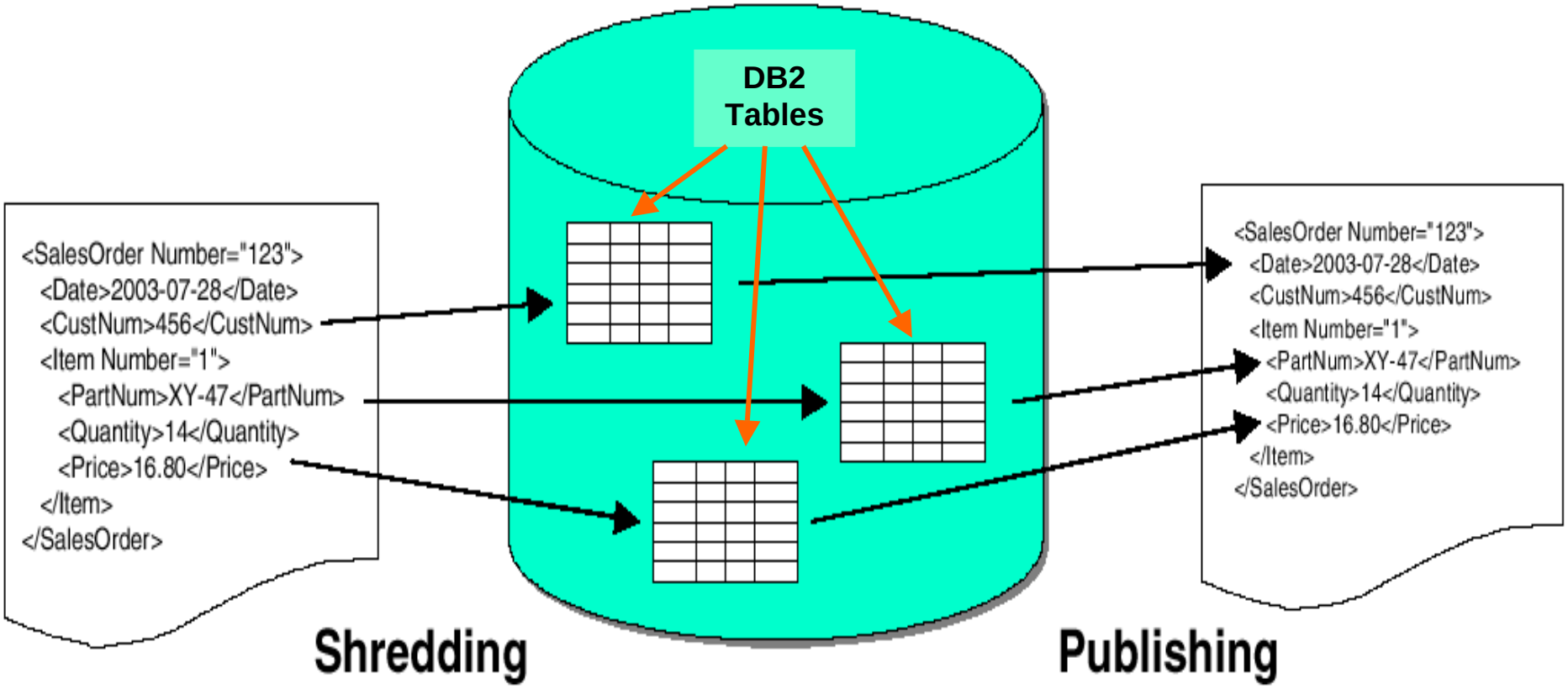
**Mit XML entsteht ein neues Datenmodell in der Welt der Datenbanken, und zwar zusätzlich zu den existierenden hierarchischen, relationalen, object-orientierten und anderen Datenmodellen. Ein XML-Dokument speichert beliebige Daten nach den Regeln eines bestimmten XML Datenmodells ab, ist also so etwas wie eine Datenbank, welche die eigentlichen Rohdaten enthält. Das XML Datenmodell ist ein Baum mit Ästen, Zweigen und Blättern, wobei die Blätter die eigentlichen Daten speichern.**

**So, wie mit SQL aus einer DB2 Datenbank Aggregationen und Verdichtungen extrahiert werden, können mit dem Werkzeug XSLT aus demselben XML-Dokument verschiedenartigste Outputs erzeugt werden.**

- **Eine XML-enabled Database verwendet ein non-XML Datenmodell und bildet ab (maps) bei jedem Zugriff Instanzen dieses Modells (z.B. SQL Tables mit den gewünschten Datan) auf eine XML Darstellung (Instanzen des XML Datenmodells).**
- **Eine native XML Database benutzt das XML Datenmodell direkt (speichert Daten im XML Format).**

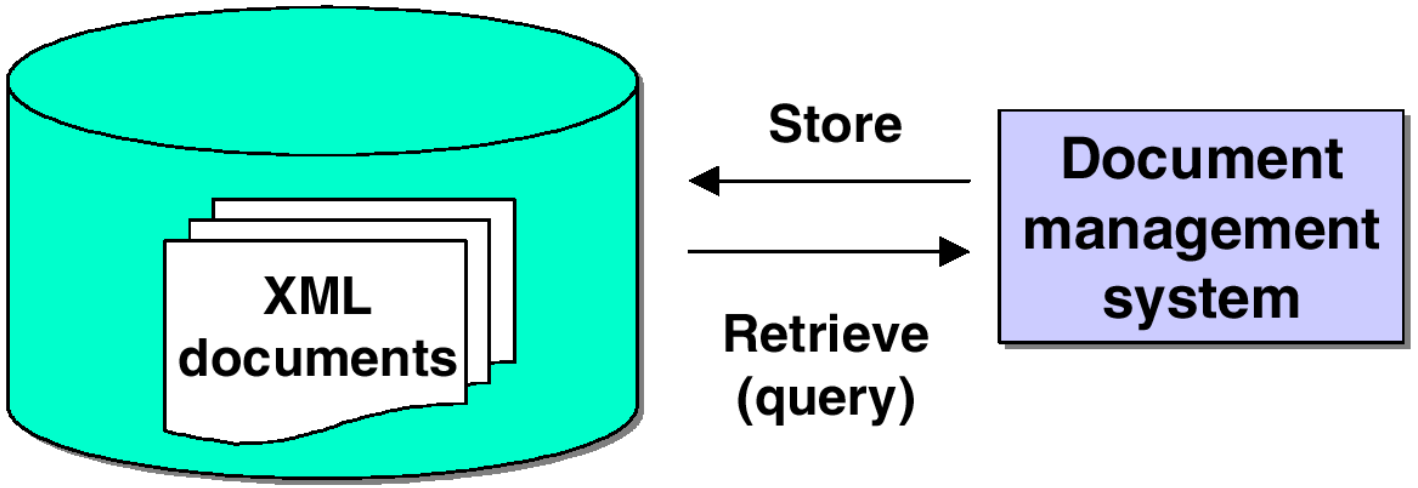
**Bei einer XML-enabled Database existiert allerdings eine Abhängigkeit von den Eigentümlichkeiten des Datenbank-Systems und der darunter liegenden Betriebssystem-Plattform. SQL Datenbanken sind auf Grund von proprietären Erweiterungen des SQL Standards durch Hersteller wie Oracle oder IBM nicht miteinander kompatibel. Bei XML/XSLT sind dagegen sowohl die Daten (das XML-Dokument) als auch die Transformationsregeln (eine oder mehrere XSLT-Dateien) reine Textdateien, die mit jedem Standard-Editor auf jedem Betriebssystem bearbeitet werden können.**

**Einfache bzw. validierende Parser sowie Parser, die zusätzlich das XSLT-Dokument laden und es auf das XML-Dokument anwenden, gibt es ebenfalls auf jeder Plattform. Damit ist die eigentliche Entwicklung eigener Dokument Type Definitions, das Schreiben der gewünschten XML-Dokumente sowie die Entwicklung des XSLT-Codes möglich, ohne Betriebssystem-spezifische Besonderheiten beachten zu müssen.**




**XML Dokumente werden benutzt, um Daten zwischen einer Datenbank und einem Anwendungsprogramm (oder einer anderen Datenbank) auszutauschen.**

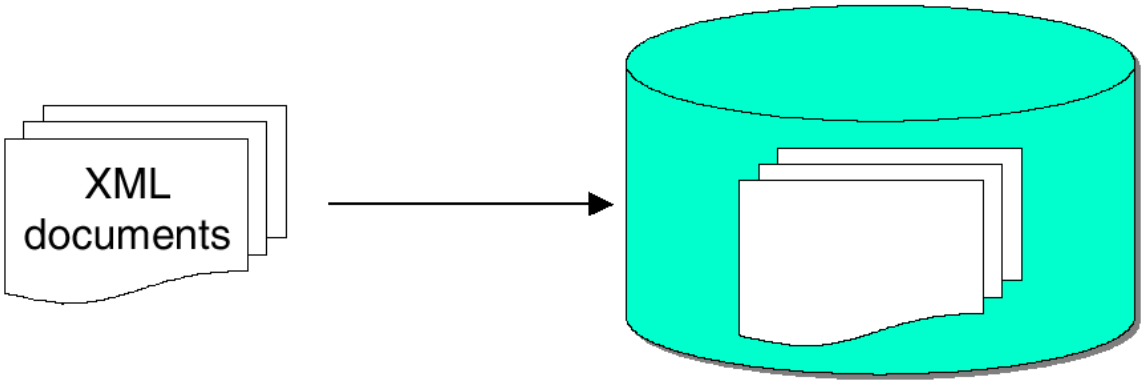
**Angenommen eine XML-enabled Database, z.B. DB2. Ein Prozess, der Daten aus einer Datenbank extrahiert und daraus ein XML Dokument erzeugt wird als publishing oder composition bezeichnet. Der umgekehrte Prozess (Daten aus einem XML Dokument extrahieren und in einer DB2 Datenbank speichern) wird als shredding oder decomposition bezeichnet.**




**Die XML Dokumente in dem obigen Beispiel sind data-centric . In anderen Worten, die Dokumente enthalten Daten mit einer regulären Struktur.**

**Andersartig sind document-centric Dokumente. Beispiele sind Endbenutzer Dokumentation, Vertriebsbroschüren und Webseiten. Document-centric Dokumente haben eine irreguläre Struktur. Hier kann eine native XML Database nützlich sein.**

**Verwaltung und Abfrage ist die am häufigsten benutzte Funktion bei document-centric Dokumenten. Beispielsweise erfordern große Projekte wie z.B. die Entwicklung eines neuen Flugzeuges einen sehr großen Umfang an Endbenutzer Dokumentation. Die Verwaltung dieser Dokumente in einer Datenbank ist eine kritische Aufgabe. Mittels XML ist es leicht, neue Dokumente aus Fragmenten bisheriger Dokumente zu erstellen. Strukturierte Abfragen über den Inhalt dieser Dokumente sind möglich.**




# Zwei unterschiedliche Arten ein XML Dokument zu speichern

```
<SalesOrder Number="123">
   <OrderDate>2003-07-28</OrderDate>
   <CustomerNumber>456</CustomerNumber>
   <Item Number="1">
      <PartNumber>XY-47</PartNumber>
      <Quantity>14</Quantity>
      <Price>16.80</Price>
   </Item>
   <Item Number="2">
      <PartNumber>B-987</PartNumber>
      <Quantity>6</Quantity>
      <Price>2.34</Price>
   </Item>
</SalesOrder>
```

## XML Version des Sales Order Dokuments

**Zwei unterschiedliche Arten, um das XML Sales Order Document zu speichern:**

1. **XML- enabled Database**
2. **native XML Database**

| Number | Date | Customer |
|---|---|---|
| 123 | 2330-07-28 | 456 |
| ... | ... | ... |

**DB2 sales order table**

| SONumber | Number | PartNumber | Quantity | Price |
|---|---|---|---|---|
| 123 | 1 | "XY-47" | 14 | 16.80 |
| 123 | 2 | "B-987" | 6 | 2.34 |
| ... | ... | ... | ... | ... |

**DB2 items table**

# XML- enabled DB2 Version des Sales Order Dokumentes

**Wird das Dokument nur für den Datenausteausch benutzt, können wir es in die Datenbank shredden. Wir extrahieren die Daten und speichern sie in 2 DB2 Tabellen. Wir können auch jederzeit unter Benutzung dieser Daten ein Sales Order Dokument im XML Format erstellen.**

# Wichtige Eigenschaften

**In der hier gezeigten XML- enabled DB2 Version des Sales Order Dokumentes ist kein XML innerhalb der DB2 Datenbank sichtbar. Das XML Dokument existiert nur außerhalb der Datenbank. Es kann jederzeit aus den Daten in der Datenbank erstellt werden. Ebenso kann es als Quelle für neu in der Datenbank zu speichernde Daten benutzt werden.**

**Ein Schema ist eine formale Beschreibung der Struktur von Daten. Bei einer XML-enabled Database stimmt das Datenbank Schema mit dem XML Schema überein. Ein unterschiedliches XML Schema wird für jedes Datenbank Schema benötigt.**

**Als Alternative kann das XML Dokument selbst in der Datenbank gespeichert werden. Hierzu werden 4 DB2 Tabellen benötigt, wie auf den beiden folgenden Seiten dargestellt.**

**Wird das Dokument selbst in einer native XML DB2 Datenbank gespeichert, benötigen wir 4 Tabellen:**

| ID | Name |
|---|---|
| 34 | "SalesOrder123.xml" |

**1. Documents table**

| DocumentID | ElementID | ParentID | Name | OrderInParent |
|---|---|---|---|---|
| 34 | 1 | NULL | "SalesOrder" | 1 |
| 34 | 2 | 1 | "OrderDate" | 1 |
| 34 | 3 | 1 | "CustomerNumber" | 2 |
| 34 | 4 | 1 | "Item" | 3 |
| 34 | 5 | 4 | "PartNumber" | 4 |
| 34 | 6 | 4 | "Quantity" | 2 |
| 34 | 7 | 4 | "Price" | 3 |
| 34 | 8 | 1 | "Item" | 4 |
| 34 | 9 | 8 | "PartNumber" | 1 |
| 34 | 10 | 8 | "Quantity" | 2 |
| 34 | 11 | 8 | "Price" | 3 |

**2. Elements table**

| DocumentID | AttributeID | ParentID | Name | Value |
|---|---|---|---|---|
| 34 | 1 | 1 | "Number" | 123 |
| 34 | 2 | 4 | "Number" | 1 |
| 34 | 3 | 8 | "Number" | 2 |

**3. Attributes table**

| DocumentID | TextID | ParentID | Value | OrderInParent |
|---|---|---|---|---|
| 34 | 1 | 2 | "2003-07-28" | 1 |
| 34 | 2 | 3 | "456" | 1 |
| 34 | 3 | 5 | "XY-47" | 1 |
| 34 | 4 | 6 | "14" | 1 |
| 34 | 5 | 7 | "16.80" | 1 |
| 34 | 6 | 9 | "B-987" | 1 |
| 34 | 7 | 10 | "6" | 1 |
| 34 | 8 | 11 | "234" | 1 |

**4. Text table**

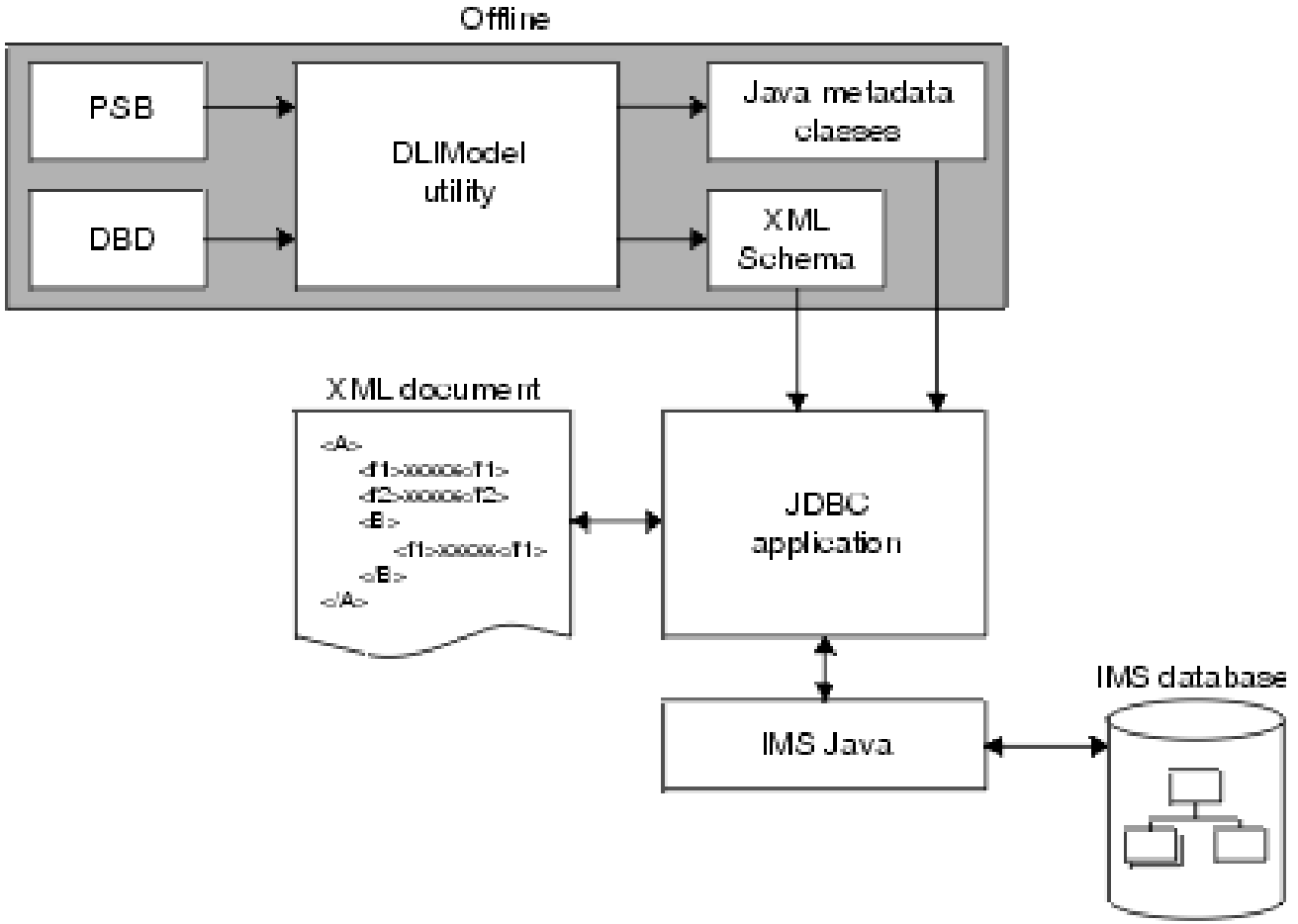
**In diesem Fall ist XML innerhalb der Datenbank sichtbar. Die Datenbank enthält Information wie Element Type und Attribute Namen. Ein einziges Datenbank Schema reicht für das Speichern aller XML Dokumente aus. In anderen Worten, das Datenbank Schema modelliert XML Dokumente, nicht die Daten in diesen Dokumenten. Datenbanken, welche XML auf diese Art benutzen werden als native XML database bezeichnet.**

# IMS

**DB2 und IMS sind die beiden wichtigsten Mainframe Datenbanken. IMS ist nicht relational; Daten werden in der Form von hierarchischen Bäumen gespeichert. Es existieren viele Situationen, wo das Speichern und Auslesen von Daten mit IMS schneller abläuft als mit DB2. Auf der anderen Seite ist die Flexibilität von DB2 viel besser.**

**Da XML und IMS Datenbanken beide hierarchisch sind, ist es besonders einfach, XML Dokumente in einer IMS Datenbank zu verwalten. Beispielsweise kann man:**

- **XML Dokumente aus allen Arten von existierenden IMS Datenbanken erzeugen. Das ist z.B. nützlich für Business-to-Business on Demand Transaktionen, oder für den Datenaustausch innerhalb einer Organisation.**
- **Eintreffende XML Dokumente innerhalb einer IMS Datenbank abspeichern. XML Dokumente werden decomposed (shredded) abgespeichert. Hierzu werden eintreffende Dokumente geparsed; die Datenelemente und Attribute werden in Feldern der IMS-Segmente als normale IMS Daten gespeichert. Dieses Verfahren ist für data-centric, weniger für document-centric XML Dokumente geeignet.**




# XML Datenspeicherung in IMS

**Ein IMS PSB, (Program Specification Block) spezifiziert die Datenbank, auf welche das Anwendungsprogramm zugreift.**

**Ein IMS DBD ist der Database Description Control Block.**