DB2 Data Management Software

IBM

e business software

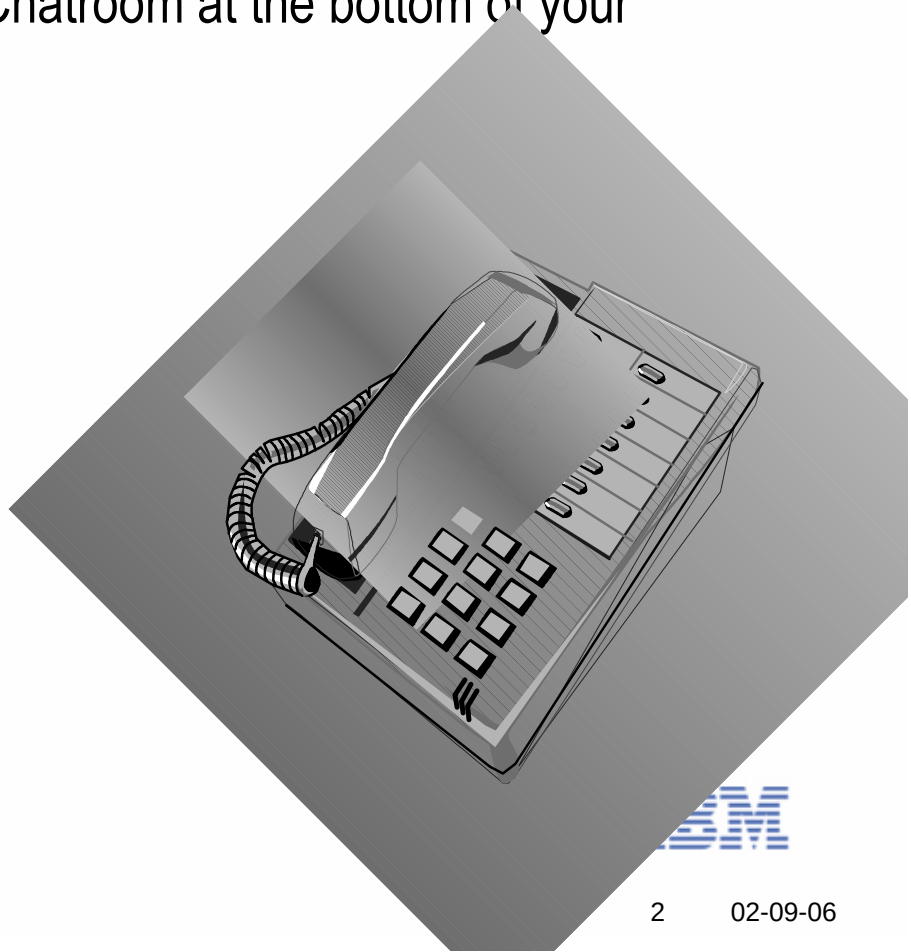
Welcome to the e-learning workshop "Using the DB2 XML Extender"

Part 1: Building DADs

IBM Software Group

Classroom Etiquette - Telephone Conferencing

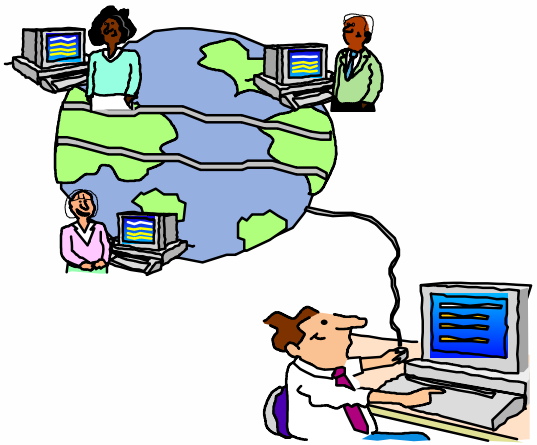
- To improve audio quality by reducing background noise - you will be placed on MUTE
- We will take questions in the Sametime Chatroom at the bottom of your screen



Web Conferencing - Sametime Instant Messaging

- Using Sametime Instant Messaging at the bottom of the screen, you can send a text message to a colleague, to the moderator or to the entire group.

- We can't see your faces for visual clues - provide us with feedback to assist you.
  - You can ask us to speak louder / slower ....
  - You can ask us to give more details for a specific section
  - Ask a question



eLearning Workshop Evaluation (help us to help you!)

By the time you complete this eLearning workshop, we would like you to be "delighted".

We hope you are able to actively participate in the class. Please ask questions / ask for clarification as required.

We will ask for your evaluation - your evaluation is very important to us:

- Review Quality / Value of PIC Workshops
- Added to the PIC **N**et **S**atisfaction **I**ndex (NSI)
- Our Target is 85 (or higher!)

Your input to the PIC NSI Calculation

1(100)	= VERY SATISFIED
2 (75)	= SATISFIED
3 (50)	= NEITHER SATISFIED NOR DISSATISFIED
4 (25)	= DISSATISFIED
5 (0)	= VERY DISSATISFIED

PLEASE complete your Evaluation form which has been sent to your Notes Id

Overview of education series

- Workshop 1 : DB2, XML and Web services
  - Part 1 : Intro to XML and DB2 XML Extender
  - Part 2 : DB2 and Webservices

- Workshop 2 : Using the DB2 XML Extender
  - Part 1 : Mapping XML to DB2 databases, Building DADs
  - Part 2: Administration, Storing and Retrieving data

Agenda

- DB2 XML Extender Overview
- XML and DTDs
- Mapping XML to databases
- Building DADs (including guidelines)
- Websphere Studio Overview
- Hands on exercises

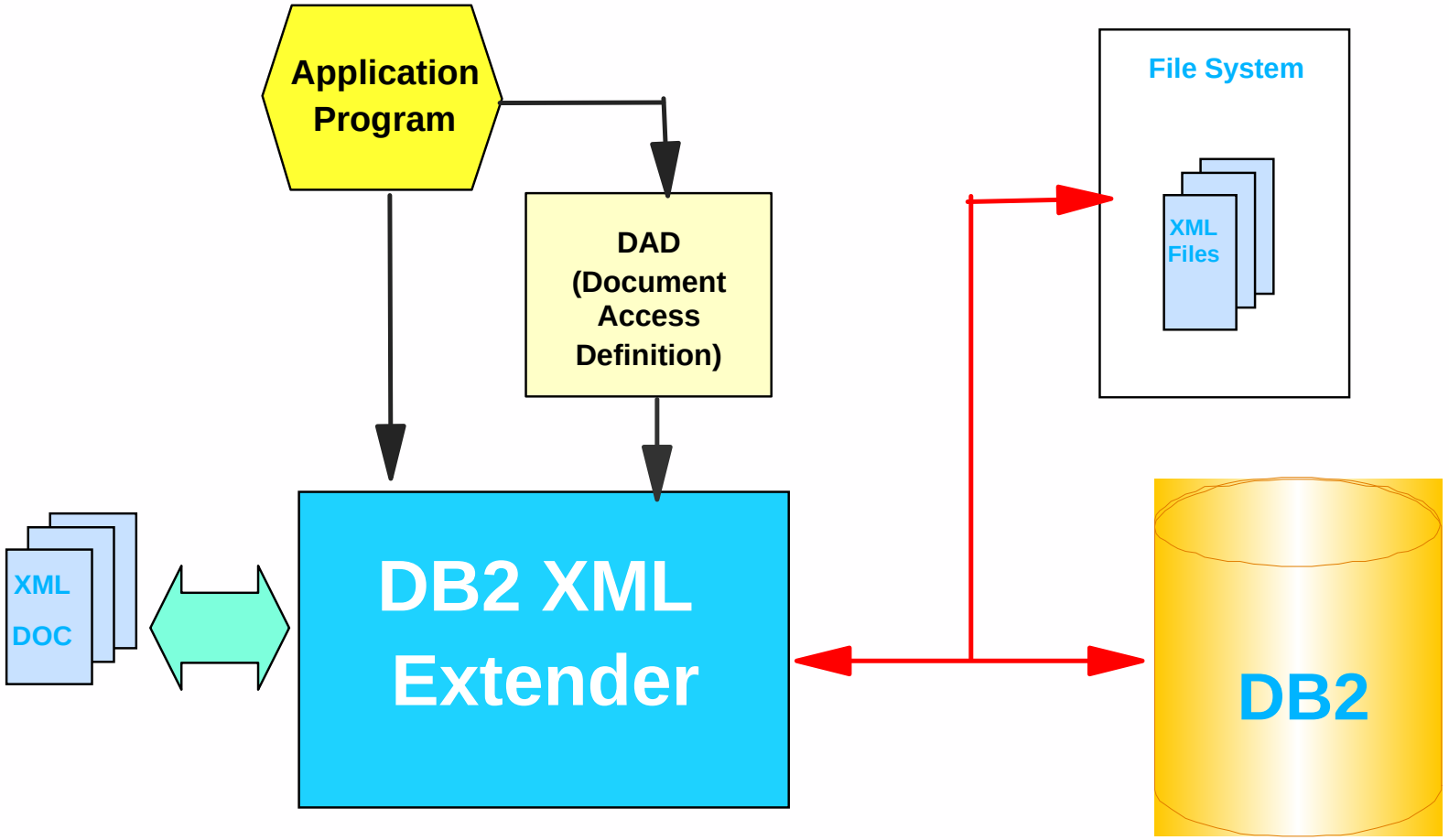
DB2 XML Extender Overview

A reminder.....

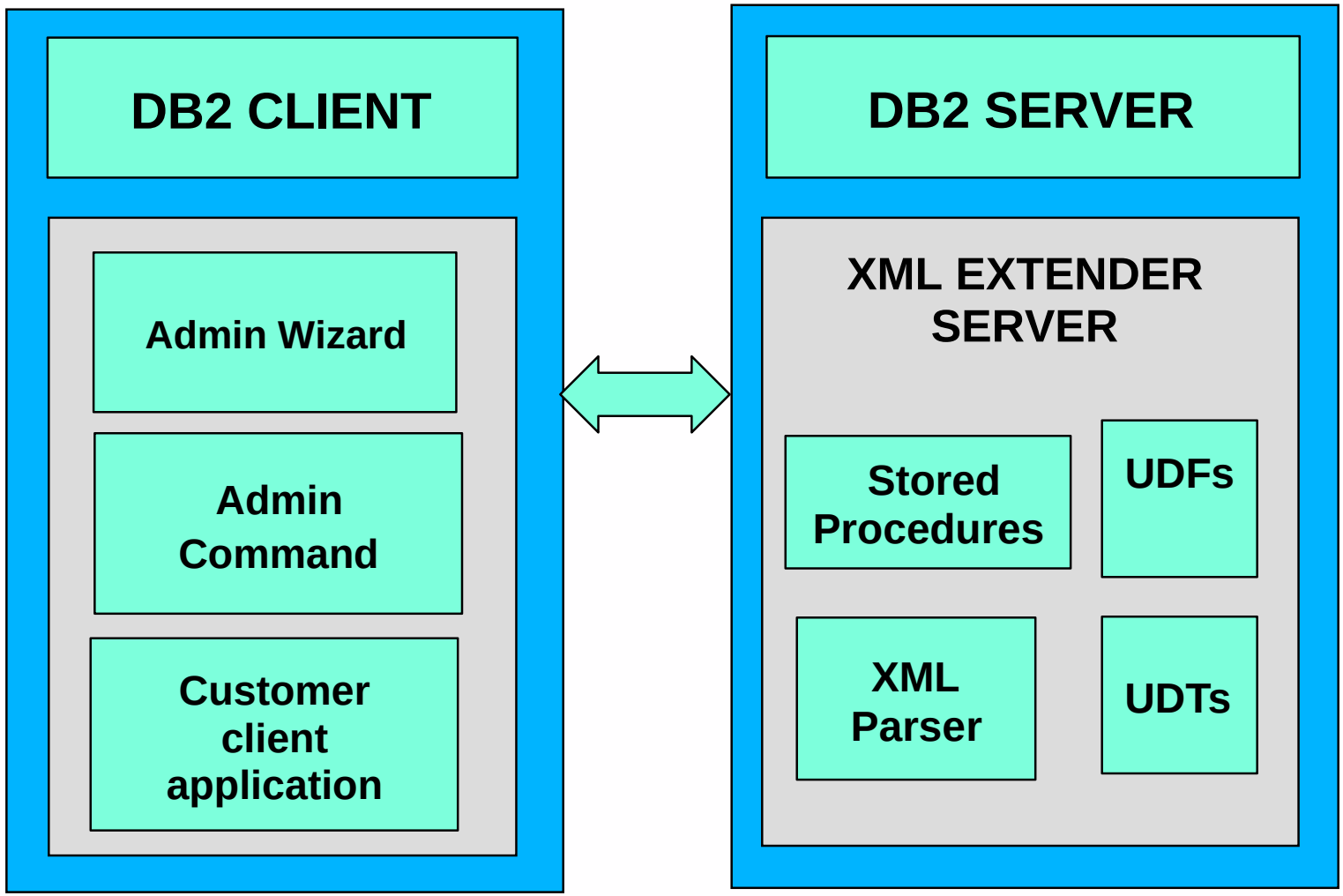
XML and DB2 - What can you do?

- Build an XML document from DB2 data
- Store an XML document in DB2
  - through decomposition (without tags)
  - entirely (with tags)
- Store an entire XML document in an external file
- Retrieve an entire XML document from DB2
- Reconstruct a decomposed document
- Extract XML elements or attribute values or fragments
- Provide search and indexing on XML

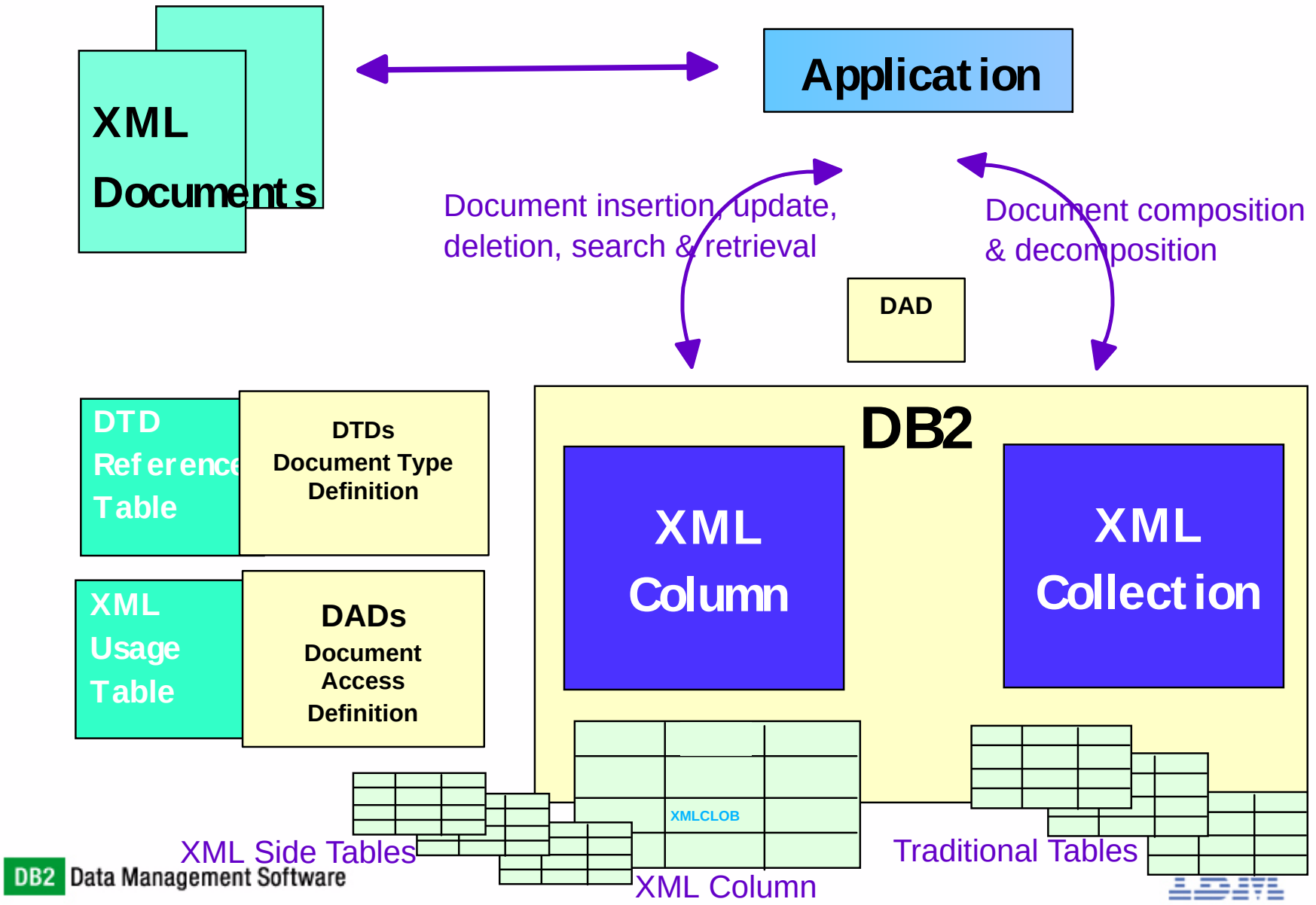
XML Extender Overview



XML Extender Architecture



DB2 Access and Storage Methods



DAD: Document Access Definition

- The DAD is an XML document used by XML columns and XML collections
- Maps XML document structures to DB2 data structures
- For XML columns, it defines how documents are indexed by identifying the elements to be extracted into side tables
- For XML collections, it maps the structure of an XML document to DB2 tables and columns.
- WSAD (WebSphere Studio Application Developer) assists in DAD creation

Summary of the 3 options

- Indexing XML documents using XML columns
  - stores XML document intact with side tables providing index functionality
- Composing and Decomposing XML documents
  - using XML collections through stored procedures
  - ability to store and retrieve data from relational tables
- Extracting data from XML documents
  - using the UDFs
  - adhoc extraction of data elements and attributes
- To summarise the XML Extender is very flexible!

A word on platforms...

- DB2 XML Extender works on the following platforms
  - Windows, AIX, Sun Solaris, Linux, AS/400 and OS/390
  - Note that with EEE some of the composition stored procedures cannot be used but there are alternatives in Fixpack 4
  - HP/UX not currently supported
- Most of the information in this presentation is platform independent
  - OS/390 and Windows specific information is included where appropriate

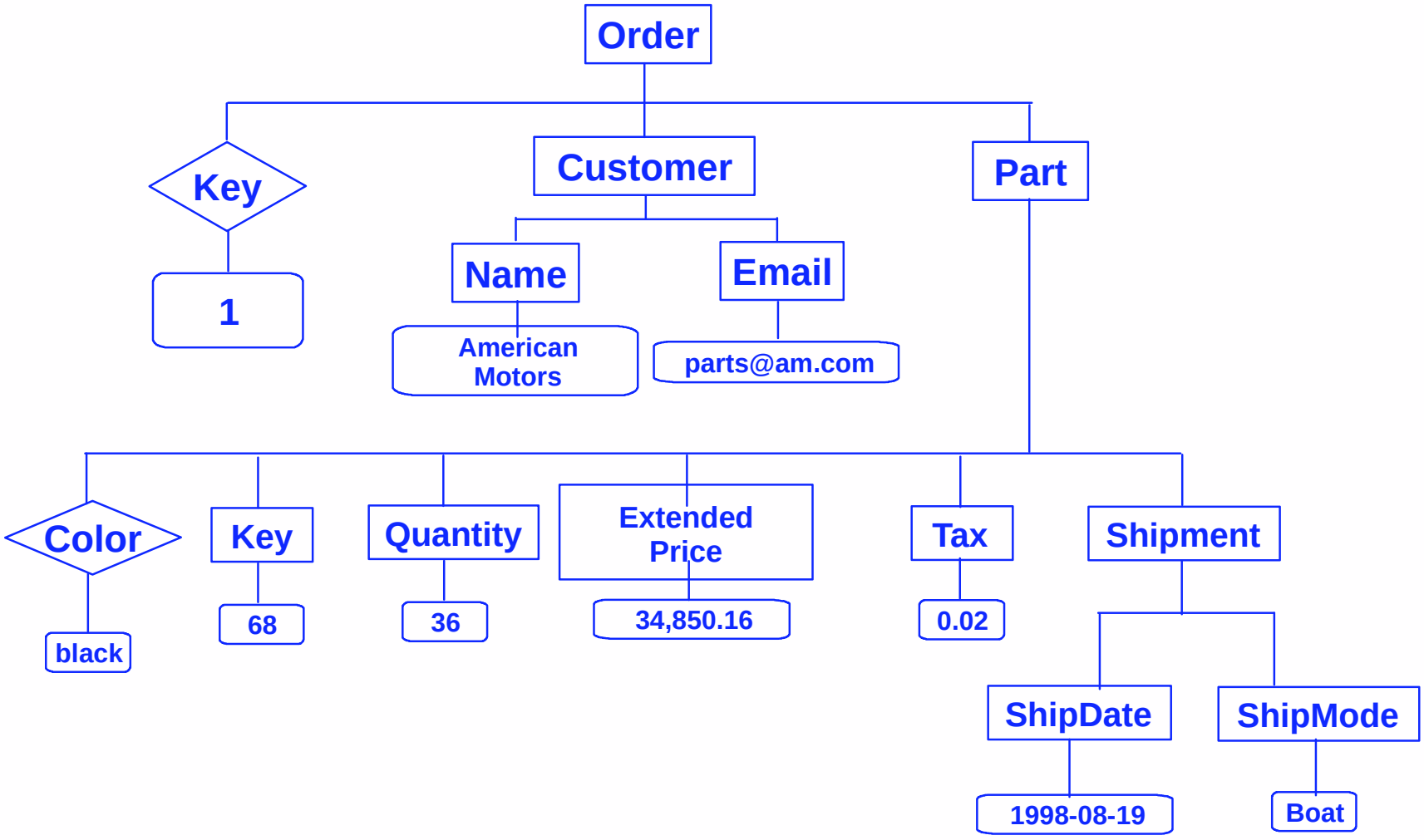
XML and DTDs

Elements vs Attributes

- Element types
  - simple - a piece of data eg <name>Joe Bloggs </name>
  - complex - comprise of other elements eg Customer
- Attributes
  - an attribute describes an element
  - an attribute can appear at most once for a given element
  - can specify allowable values, ranges, default values in the DTD
    - <!ATTLIST person band (A|B|C|D|E|F) #REQUIRED>
- Note that there are no rules/guidance as to what data should be elements and which attributes - they are all just pieces of data

XML IS DOCUMENT RATHER THAN DATA BASED!

Sample XML document hierarchy



Building up a sample XML document

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "order.dtd">
```

These first two lines denote it is an Order XML document and the DTD that should be used

Building up a sample XML document

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "order.dtd">
<Order>

</Order>
```

Order root element

Note that every XML document must be enclosed within a set of tags (root)

Order element closing tag

Building up a sample XML document

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "order.dtd">
<Order key="1">

</Order>
```

Order key attribute

Building up a sample XML document

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "order.dtd">
<Order key="1">
  <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>




</Order>
```

Customer complex element

Simple elements

Customer closing tag

Building up a sample XML document

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "order.dtd">
<Order key="1">
  <Customer>
     <Name>American Motors</Name>
     <Email>parts@am.com</Email>
  </Customer
  <Part color="red   ">
     <key>156</key>
  </Part>

  <Part color="black">
     <key>68</key>
  </Part>

</Order>
```

Part element and color attribute

Key element

Closing tag for first Part element

Second part element

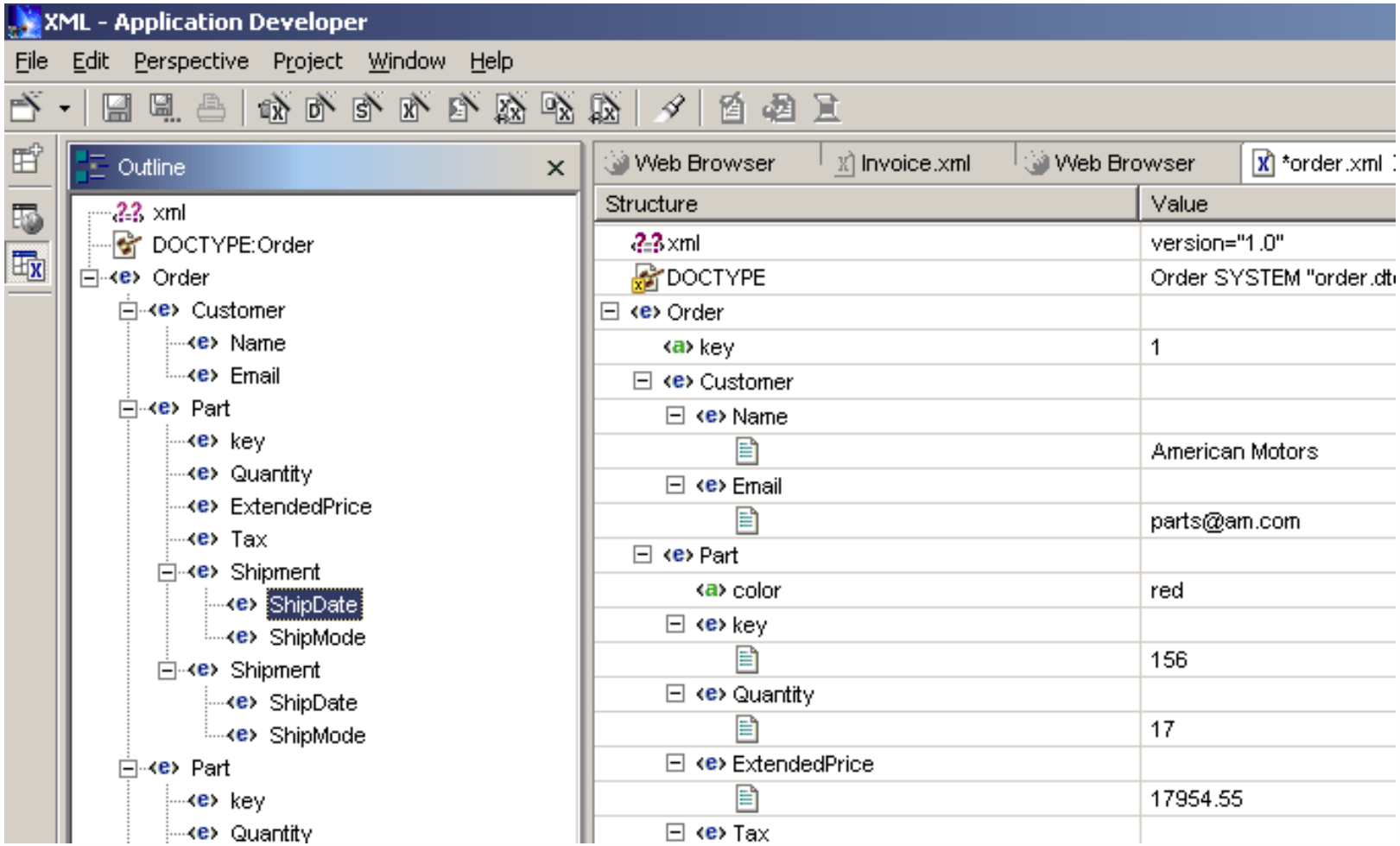
NB Note that keys and other data can be elements **OR** attributes

Building up a sample XML document

```
<Part color="red   ">
<key>156</key>
<Quantity>17</Quantity>
  <ExtendedPrice>17954.55</ExtendedPrice>
  <Tax>2.000000e-02</Tax>
  <Shipment>
    <ShipDate>1998-03-13</ShipDate>
    <ShipMode>TRUCK </ShipMode>
  </Shipment>
  <Shipment>
    <ShipDate>1999-01-16</ShipDate>
    <ShipMode>FEDEX </ShipMode>
  </Shipment>
</Part>
<Part color="black">
<key>68</key>
<Quantity>36</Quantity> etc
```

Elements within each Part

Websphere Studio XML design view

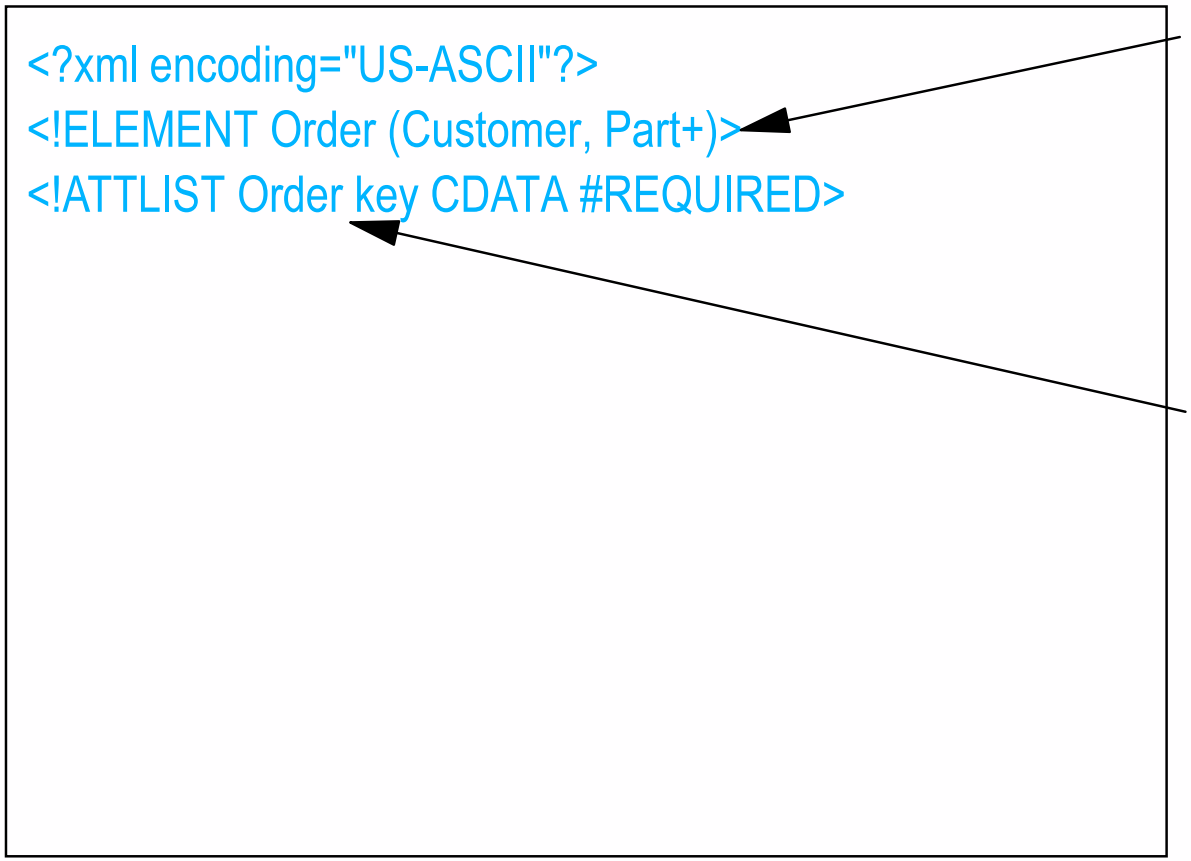


NB Ability to edit XML and flip between XML source and design view

Document Type Definition - DTD

- Defines the structure of an XML document
  - defines how elements relate to one another within the documents tree structure
  - defines the tags that can or must appear
  - how often the tags can appear
  - how often the tags can be nested
  - allowable, required or default attributes
- XML references DTD in <!DOCTYPE> XML prolog
  - Allows a validating parser to detect deviations from a vocabulary
  - Use is optional - can parse a well-formed XML document without a DTD
- DTDs can be shared across organizations
- **Defines the rules of the language we create**

Building up a sample DTD



Order element is a complex element comprising customer and any number of part elements

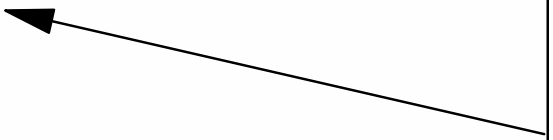
Order key attribute (mandatory)

Note that the component elements are described before any attributes (in the XML the key attribute was listed first)

Building up a sample DTD

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email, Phone)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
```

Customer element comprises these simple elements



#PCDATA is the XML method of denoting a parsable text string (Note that no length is specified)

Building up a sample DTD

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email, Phone)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
```

Denotes one or more shipments - Repeating group!

? would denote optional

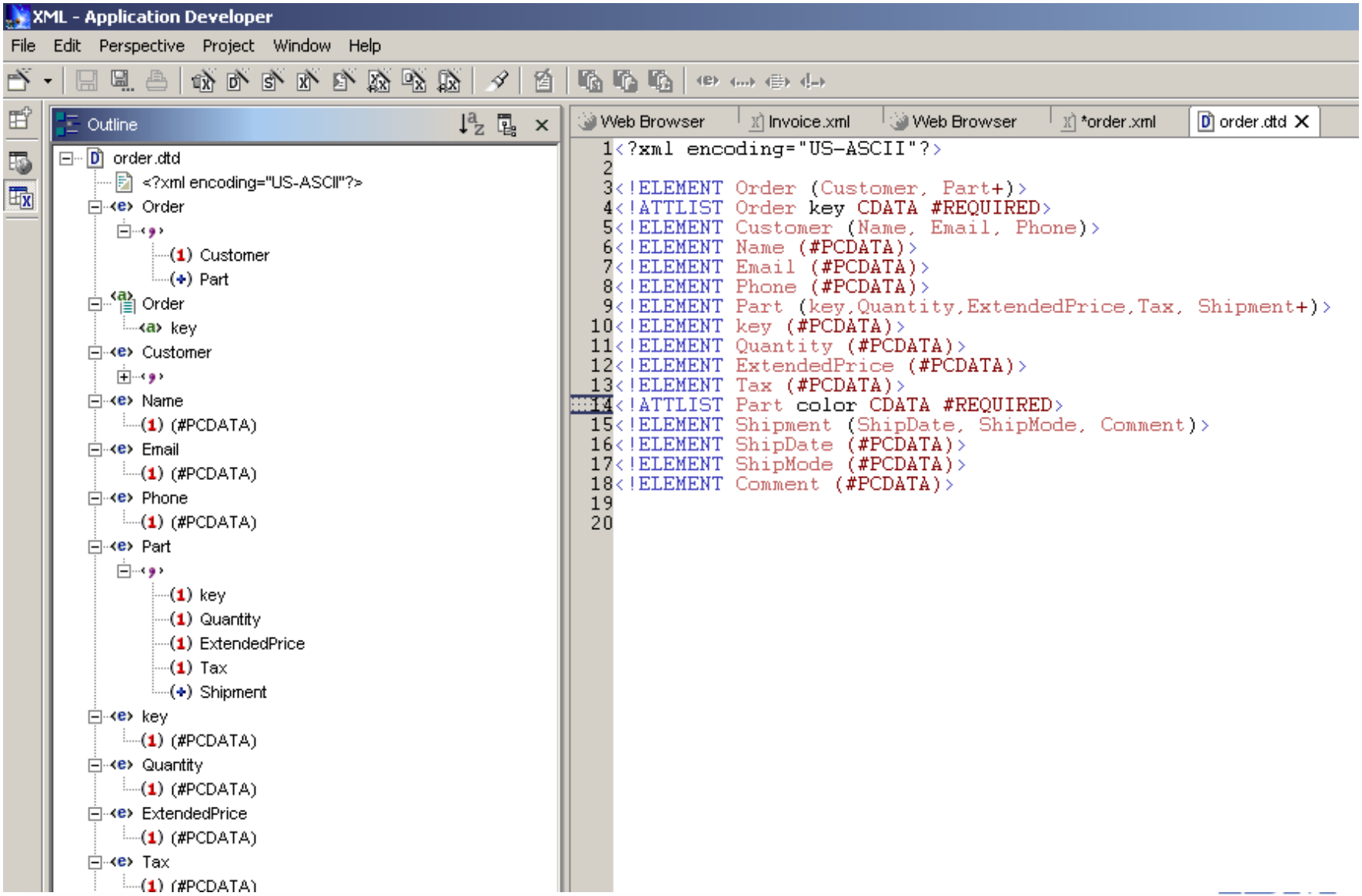
Note that in this case key is an element rather than an attribute

Denotes a character string - mandatory

Complete sample DTD

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email, Phone)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode, Comment)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
<!ELEMENT Comment (#PCDATA)>
```

Websphere Studio view of DTD



Mapping XML to the database

Before we start discussing the DB2 XML Extender in more detail we should discuss some concepts of mapping XML to databases...

Mapping XML to DB2

- XML and DTDs include the following Metadata
  - Element and attribute hierarchy
  - Element and attribute names
  - Allowable values and defaults
  - Optionality (nullability)
  - Repeating groups

Intact documents or relational format?

- Do you want to store data or documents?
  - Is XML used purely as a transport mechanism or is it important for document information to be retained?
  - Is it acceptable for reconstruction to create a different document?
- Does data have regular structure with fine-grained data or lots of text?
- How often are the documents read as opposed to the data?
- How often is the data updated? Update of individual element or attribute values is possible with column but not collection
- Do the documents or DB2 tables already exist?

Key factors for mapping the data hierarchy

- Do the DB2 tables already exist?
  - If so, map each XML element or attribute to a column in a table
  - If not, produce a data model based on information in the XML documents and DTDs

- Do the XML documents already exist and is their structure outside your control?
  - XML Extender assumes all elements and attributes in an XML document have unique names even if they are in different paths
    - If duplicate names exist, you may wish to either transform the document using XSL, or consider the use of temporary table and triggers
    - For example, elements SHIPMENT and DELIVERY could both have sub-elements called DATE which need to map to SHIPMENT_DATE AND DELIVERY_DATE
  - A particular element or attribute can populate just one column (except for columns specified in join conditions)

Populating multiple columns

- An element or attribute can only populate a single column
  - Except where the XML data is destined for a column that is part of a primary key-foreign key relationship with another column of another table
- Possible options are
  - Change XML document to have repeating child elements
  - Keep original structure but have multiple documents
  - Restructure the table to resemble document
  - Consider setting up triggers to populate the additional columns
- If you want to combine element values into a single column eg Day, Month Year elements into a DB2 DATE you should consider using a temporary table and triggers

Mapping XML names to DB2

- How this is done will depend on whether the XML documents and/or tables already exist
- It makes sense to use the same names where possible but XML names do not allow some special characters eg @
- Element and attribute names can be up to 63 bytes long which may prove an issue for some databases (DB2 for OS/390 currently restricts names to 18 characters)
- Note that element and attribute names are case sensitive whereas DB2 names are not unless in quotes

Mapping XML types to SQL types

- PCDATA can be any string of characters
  - Knowledge of data required to determine DB2 type and length
  - Eg Name element to CHAR, Quantity element to INTEGER
- DB2 XML Extender will introduce support for schemas which have type support, including allowable values, ranges etc
- Note that user-defined data types are not currently supported by the XML extender
- ID and IDREF can be used within XML documents to uniquely identify elements in an XML document and to associate one element to another
  - Note that there is no special handling within DB2 at this stage

Mapping optionality

- Take care when defining the nullability of columns to avoid failures on insert!
- Beware defining columns as NOT NULL, as the element or attribute must be present in the XML document regardless of whether it has a real value
  - Elements or attributes can be specified as empty by
    - <elem></elem>, </elem> or attrib=""
  - Could consider a trigger program to set default values
- #REQUIRED could map to NOT NULL
- ? denotes an element can be optional ie nullable
  - eg Customer(Name,Email?)
- If a default is specified for an attribute in the DTD and validation is turned on, it will be inserted during decomposition

Repeating groups in XML

- * and + denote that an element can be multi-occurring and therefore a candidate for normalization
- Denote which columns are part of a repeating group by enclosing them in a wrapper element
  - They have no child attribute or text nodes
  - They have one or more element nodes that have attributes or text nodes that map to the same table
  - Attribute hints are an alternative method
    - An attribute_node can only be specified on the first element_node that corresponds to a table
- If you want to decompose elements into the same row, use multi-occurrence =yes
  - Multi-occurrence no = repeat within parent tag, yes means repeat with children
  - Specify yes to identify the beginning of a new table

Repeating groups example

```
<element_node name="Shipment" multi_occurrence="YES">
        <element_node name="ShipDate">
          <text_node>
            <RDB_node>
              <table name="SHIP_TAB"/>
              <column name="DATE" type="Date"/>
            </RDB_node>
          </text_node>
        </element_node>
        <element_node name="ShipMode">
          <text_node>
            <RDB_node>
              <table name="SHIP_TAB"/>
              <column name="MODE" type="Character(6)"/>
            </RDB_node>
          </text_node>
        </element_node>
        <element_node name="Comment">
          <text_node>
            <RDB_node>
              <table name="SHIP_TAB"/>
              <column name="COMMENT" type="VarChar(64)"/>
            </RDB_node>
          </text_node>
        ......
```

Wrapper element

Multi-occurrence =yes means shipdate, shipmode and comment will all be inserted into same row

NB If multi-occurrence=no was specified 3 rows would have been inserted

Sample mapping to DB2

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email, Phone)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode, Comment)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
<!ELEMENT Comment (#PCDATA)>
```

+ denotes Part element is a repeating group

+ denotes Shipment is a repeating group

Data types mapped to DB2 types based on knowledge of data eg Quantity to Int

Color could be defined as not null

Sample mapping to DB2

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email, Phone)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode, Comment)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
<!ELEMENT Comment (#PCDATA)>
```

ORDER_TAB

ORDER_KEY	INT
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)
CUSTOMER_PHONE	VARCHAR(16)

PART_TAB

PART_KEY	INT
COLOR	CHAR(6)
QUANTITY	INT
PRICE	DEC(10,2
TAX	REAL
ORDER_KEY	INT

SHIP_TAB

DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(64)
PART_KEY	INT

Example of document not suited to XML extender

- XML extender requires distinguishable element and attribute names for column indexing and collection decomposition
- Example of unsuitable document

```
<root>
   <param id="firstname">John</param>
   <param id="lastname">Smith</param>
</root>
```

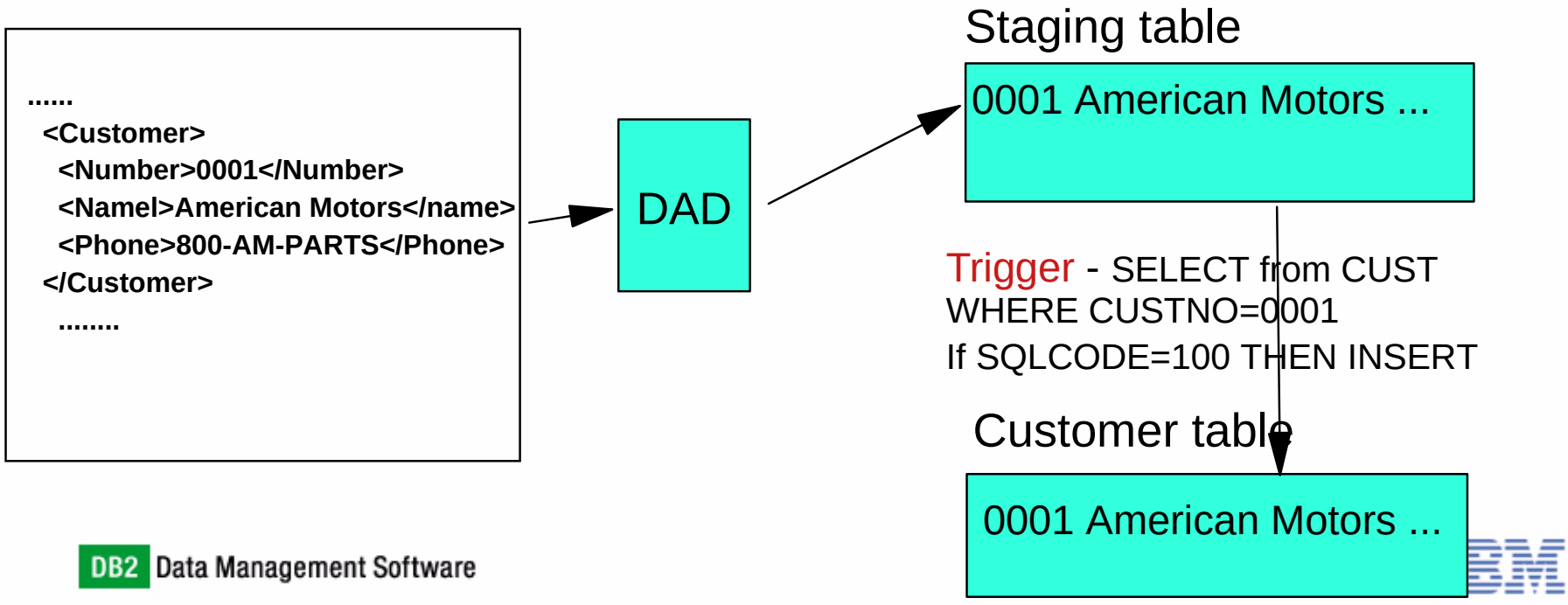
- Better structure would be

```
<root>
   <firstname>John</firstname>
   <lastname>Smith</lastname>
</root>
```

- Could transform using a stylesheet first
  - Planned to provide support for the XML extender to use stylesheets

Inserting rows only if unique

- You may wish to insert a row into a table only if the data does not already exist in the table
  - Set up a staging table with exactly the same column definitions
  - Create a trigger which selects from the actual table to check if the data already exists and insert/update if appropriate
  - Populate the staging table via the DAD which will automatically cause the actual table to be inserted if appropriate



Building DADs for XML collections

Reasons for choosing XML collections

- You want to generate XML documents from DB2 tables (new or existing)
- Your XML documents contain a collection of data that map well to relational
- You wish to incorporate document content into existing or new relational systems
- You need to create different views of relational data using different mapping schemes
- Your XML documents come from other data sources and you just want to store pure data
- A subset of your XML document needs to be updated often and performance is critical
- Need to process document content with analytical software

DB2 XML Collection

- XML documents decomposed
  - XML elements and attributes stored as SQL data types
  - with optional validation against DTDs
  - many rows in many tables can be created
- XML documents composed or reconstructed
  - from regular SQL content held in traditional tables
  - with optional validation against DTDs
  - multiple documents can be generated from one request
- Document Access Definition: To define mappings between relational and XML data
  - Two mapping types - SQL and RDB node

SQL Mapping

- **Composition only**
  - SQL syntax
  - Single SQL statement only
  - Specify the mapping between column and XML data in attribute_node and text_node

```
<DAD>
<dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
<validation>YES</validation>
<Xcollection>
<SQL_stmt>
        SELECT  ........
</SQL_stmt>
<root_node>
        <element_node name="Order">
        <attribute_node name="key">
        <column name="order_key"/>
        </attribute_node>

        ........
        </element_node>
</root_node>
</Xcollection>
</DAD>
```

Sample SQL mapping DAD

```
<?xml version="1.0"?>
<!DOCTYPE DAD PUBLIC "dadId" "dad.dtd">
<DAD>
  <dtdid>order.dtd</dtdid>
  <validation>NO</validation>
```

The DAD itself is an XML document so has an associated DTD

Specifies the DTD for the ORDER document and whether validation should be performed

- performance impact

Sample SQL mapping DAD

```
<?xml version="1.0"?>
<!DOCTYPE DAD PUBLIC "dadId" "dad.dtd">
<DAD>
  <dtdid>order.dtd</dtdid>
  <validation>NO</validation>
  <Xcollection>
    <SQL_stmt>
SELECT
   O.ORDER_KEY,  O.CUSTOMER_NAME,  O.CUSTOMER_EMAIL,  O.CUSTOMER_PHONE,
   P.PART_KEY,  P.COLOR,  P.QUANTITY,  P.PRICE,  P.TAX,
   S.DATE,  S.MODE,  S.COMMENT
FROM
   XMLDEMO.ORDER_TAB AS O, XMLDEMO.PART_TAB AS P, XMLDEMO.SHIP_TAB AS S
WHERE
   O.ORDER_KEY = P.ORDER_KEY
   AND P.PART_KEY = S.PART_KEY
   AND O.ORDER_KEY = 1
   AND P.PRICE > 20000
ORDER BY
   O.ORDER_KEY ASC,
   P.PART_KEY ASC
    </SQL_stmt>
```

Single SQL statement which selects all the data required for the XML document

ORDER BY maintains the top-down order of entity hierarchy - NOT the order of the rows selected

Sample SQL mapping DAD

```
<?xml version="1.0"?>
<!DOCTYPE DAD PUBLIC "dadId" "dad.dtd">
<DAD>
 <dtdid>order.dtd</dtdid>
 <validation>NO</validation>
 <Xcollection>
  <SQL_stmt>
SELECT
  O.ORDER_KEY,  O.CUSTOMER_NAME,  O.CUSTOMER_EMAIL,  O.CUSTOMER_PHONE,
  P.PART_KEY,  P.COLOR,  P.QUANTITY,  P.PRICE,  P.TAX,
  S.DATE,  S.MODE,  S.COMMENT
FROM
  XMLDEMO.ORDER_TAB AS O, XMLDEMO.PART_TAB AS P, XMLDEMO.SHIP_TAB AS S
WHERE
  O.ORDER_KEY = P.ORDER_KEY
  AND P.PART_KEY = S.PART_KEY
  AND O.ORDER_KEY = 1
  AND P.PRICE > 20000
ORDER BY
  O.ORDER_KEY ASC,
  P.PART_KEY ASC
  </SQL_stmt>
  <root_node>
    <element_node name="Order">
      <attribute_node name="key">
        <column name="ORDER_KEY"/>
      </attribute_node> etc
```

Beginning of the root and attribute node definitions which tell the Extender what the document should look like

Discussed in more detail on next slide

Sample SQL mapping DAD continued

```
<element_node name="Customer">
  <element_node name="Name">
    <text_node>
      <column name="CUSTOMER_NAME"/>
    </text_node>
  </element_node>
  <element_node name="Email">
    <text_node>
      <column name="CUSTOMER_EMAIL"/>
    </text_node>
  </element_node>
  <element_node name="Phone">
    <text_node>
      <column name="CUSTOMER_PHONE"/>
    </text_node>
  </element_node>
</element_node>
<element_node name="Part" multi_occurrence="YES">
  <attribute_node name="color">
    <column name="COLOR"/>
  </attribute_node>
  <element_node name="key">
    <text_node>
      <column name="PART_KEY"/>
    </text_node>
  </element_node>
    etc.....................
</root_node>
</Xcollection>
</DAD>
```

Element and attribute names

Name of column selected in SQL statement

Multi-occurrence="YES" has to be specified for Part and Shipment so the Extender knows there could be more than one

NB - NO column types and lengths required as composition only

SQL Mapping guidelines

- Columns should be specified in top-down order in order of XML hierarchy
- Columns belonging to the same table must be grouped together
- Use the AS clause if you need to specify a name for the column due to duplicate column names in the SQL statement or to refer to derived data
  - eg SELECT O.DATE AS ORDER_DATE, S.DATE AS SHIP_DATE
- Test SQL for performance beforehand - tune bufferpools, heap sizes etc
  - Run explain to check access path

Composing a single document from multiple rows

- If your SQL statement returns multiple rows but you want all the data in a single XML document, change your SQL slightly as follows:

Original SQL

```
SELECT EMPNO, FIRSTNAME,LASTNAME FROM EMP
ORDER BY EMPNO
```

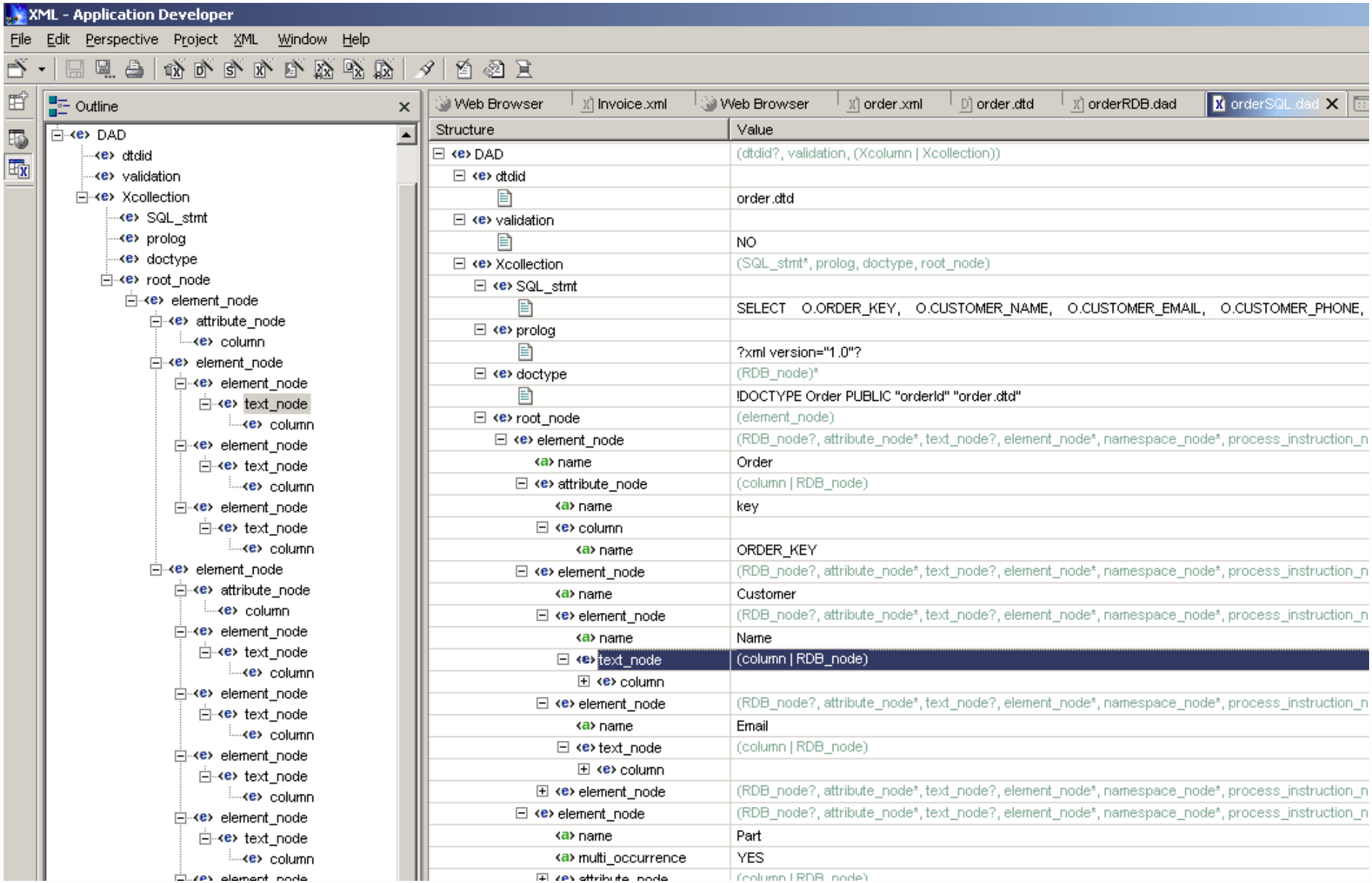
Produced 32 documents

Revised SQL

```
SELECT "2" AS CONST, EMPNO,FIRSTNAME, LASTNAME
FROM EMP
ORDER BY CONST,EMPNO
```

Produces a single document containing all the data

Websphere Studio Design view



RDB Node Mapping

- **Composition and decomposition**
  - XML syntax
  - Specify tables and relationship among tables in the RDB_node of the root element_node
  - Specify RDB_node with table name, column name and optional condition for attribute_node and text_node

```
<DAD>
<Xcollection>
    ....
<element_node name="Order">
 <RDB_node>
  <table name="order_tab"/>
 <table name="part_tab"/>
  <condition>
     order_tab.order_key =
          part_tab.part_key
  </condition>
 </RDB_node>
......
<attribute_node name="key">
<RDB_node>
<table name="order_tab"/>
<column name="order_key"/>
</RDB_node>
</attribute_node>
....
</Xcollection>
</DAD>
```

Sample RDB mapping DAD

```
<?xml version="1.0"?>
<!DOCTYPE DAD PUBLIC "dadId" "dad.dtd">
<DAD>
  <dtdid>order.dtd</dtdid>
  <validation>NO</validation>
```

The DAD itself is an XML document so has an associated DTD

Specifies the DTD for the ORDER document and whether validation should be performed

- performance impact

Sample RDB node mapping DAD

```
<?xml version="1.0"?>
<!DOCTYPE DAD PUBLIC "dadId" "dad.dtd">
<DAD>
 <dtdid>order.dtd</dtdid>
 <validation>NO</validation>
 <Xcollection>
   <root_node>
    <element_node name="Order">
     <RDB_node>
      <table name="ORDER_TAB"/>
      <table name="PART_TAB"/>
      <table name="SHIP_TAB"/>
      <condition>
       ORDER_TAB.ORDER_KEY=PART_TAB.ORDER_KEY AND PART_TAB.PART_KEY=SHIP_TAB.PART_KEY
      </condition>
     </RDB_node>
```

Top element name is root element

At this level you must define the DB2 source/target tables involved and the join condition

Sample RDB mapping DAD

```
<?xml version="1.0"?>
<!DOCTYPE DAD PUBLIC "dadId" "dad.dtd">
<DAD>
 <dtdid>order.dtd</dtdid>
 <validation>NO</validation>
 <Xcollection>
   <root_node>
    <element_node name="Order">
     <RDB_node>
      <table name="ORDER_TAB"/>
      <table name="PART_TAB"/>
      <table name="SHIP_TAB"/>
      <condition>
       ORDER_TAB.ORDER_KEY=PART_TAB.ORDER_KEY AND PART_TAB.PART_KEY=SHIP_TAB.PART_KEY
      </condition>
     </RDB_node>
     <attribute_node name="key">
      <RDB_node>
       <table name="ORDER_TAB"/>
       <column name="ORDER_KEY" type="Integer"/>
      </RDB_node>
     </attribute_node>
```

Example of attribute definition - defines the table/column name and data type

Sample RDB mapping DAD

```
<element_node name="Customer">
  <element_node name="Name">
    <text_node>
      <RDB_node>
        <table name="ORDER_TAB"/>
        <column name="CUSTOMER_NAME" type="VarChar(16)"/>
      </RDB_node>
    </text_node>
  </element_node>
  <element_node name="Email">
    <text_node>
      <RDB_node>
        <table name="ORDER_TAB"/>
        <column name="CUSTOMER_EMAIL" type="VarChar(16)"/>
      </RDB_node>
    </text_node>
  </element_node>
  <element_node name="Phone">
    <text_node>
      <RDB_node>
        <table name="ORDER_TAB"/>
        <column name="CUSTOMER_PHONE" type="VarChar(16)"/>
      </RDB_node>
    </text_node>
  </element_node>
</element_node>
```

DB2 Data Management Software

Sample RDB mapping DAD

```
      <element_node name="Part" multi_occurrence="YES">
       <attribute_node name="color">
        <RDB_node>
         <table name="PART_TAB"/>
         <column name="COLOR" type="Character(6)"/>
        </RDB_node>
       </attribute_node>
       <element_node name="key">
        <text_node>
         <RDB_node>
          <table name="PART_TAB"/>
          <column name="PART_KEY" type="Integer"/>
         </RDB_node>
         .........
   </root_node>
  </Xcollection>
</DAD>
```

Multi-occurrence=yes must be specified for Part and Shipment

RDB node mapping guidelines

- All elements except the root must be enclosed in text_node or attribute_node elements
- You can specify conditions so documents are only created for data meeting certain criteria
- Each table can have up to 10240 rows for each document
- Use wrapper elements to enclose elements forming a single row
  - Note that examples often use attribute hints
    - Example in DB2 for OS/390 Powering the worlds e-business solutions redbook gives an example of how to generate a key using SELECT DB2XML.generate_unique() as ID

Decomposition guidelines

- You must specify a primary key for each table
  - Eg <table name = "part_tab" key="part_key">
  - For composite keys put spaces in between column names
  - They do not need to be defined as PKs within DB2
- Cannot have sub-elements of an element mapping to columns in different tables
- Duplicate element or attribute names are disallowed
- Consider whether the order of multi-occurring elements needs to be preserved
  - If the tables exist prior to enabling the collection, the sequence order of multi-occuring elements and attributes is NOT preserved
  - If tables were created during enable collection, they include a sequence number DXX_SEQNO which will be used to preserve order

General DAD advice for collections

- You must fully qualify any mapped element in the DAD
- Order of element and attributes in DAD must match XML document
- For enable_collection and decomposition, column types and lengths must be specified in the DAD and they must match the DDL
  - enable collection uses types to create tables or check them if they already exist
- The tags in a DAD must be unique. Any tag with the same name must refer to the same column in the DAD
- Only the primary key of a primary key-foreign key join condition needs a mapping in the DAD

Dynamic mapping

- Ability for applications to create DADs on the fly
- Change query criteria via overriding query condition for document generation
- SQL_OVERRIDE for SQL composition:
  - replaces the entire SQL query
  - change conditions in the WHERE clause
  - designed for DBMS experts
- XML_OVERRIDE for RDB node composition:
  - Use XPath syntax to define element or attribute,
  - Overrides a condition in the DAD to specify the constraints on XML elements or attributes on the documents to be generated
  - An example:

    "/department/@id='E01'

XML_Override example

- Part of an RDB node mapping DAD

```
<element_node name="Price">
    <text_node>
      <RDB_node>
        <table name="PART_TAB"/>
        <column name="PRICE" type="Decimal(10,2)"/>
          <condition>
              Price > 2500.00
          </condition
      </RDB_node>
    </text_node>
</element_node>
```

- The price condition can be overriden when composing XML documents by specifying values for the stored procedure override input parameters

  OVERRIDETYPE : XML_OVERRIDE

  OVERRIDE : "/Order/Part/Price = 5000"

Building DADs - XML Columns

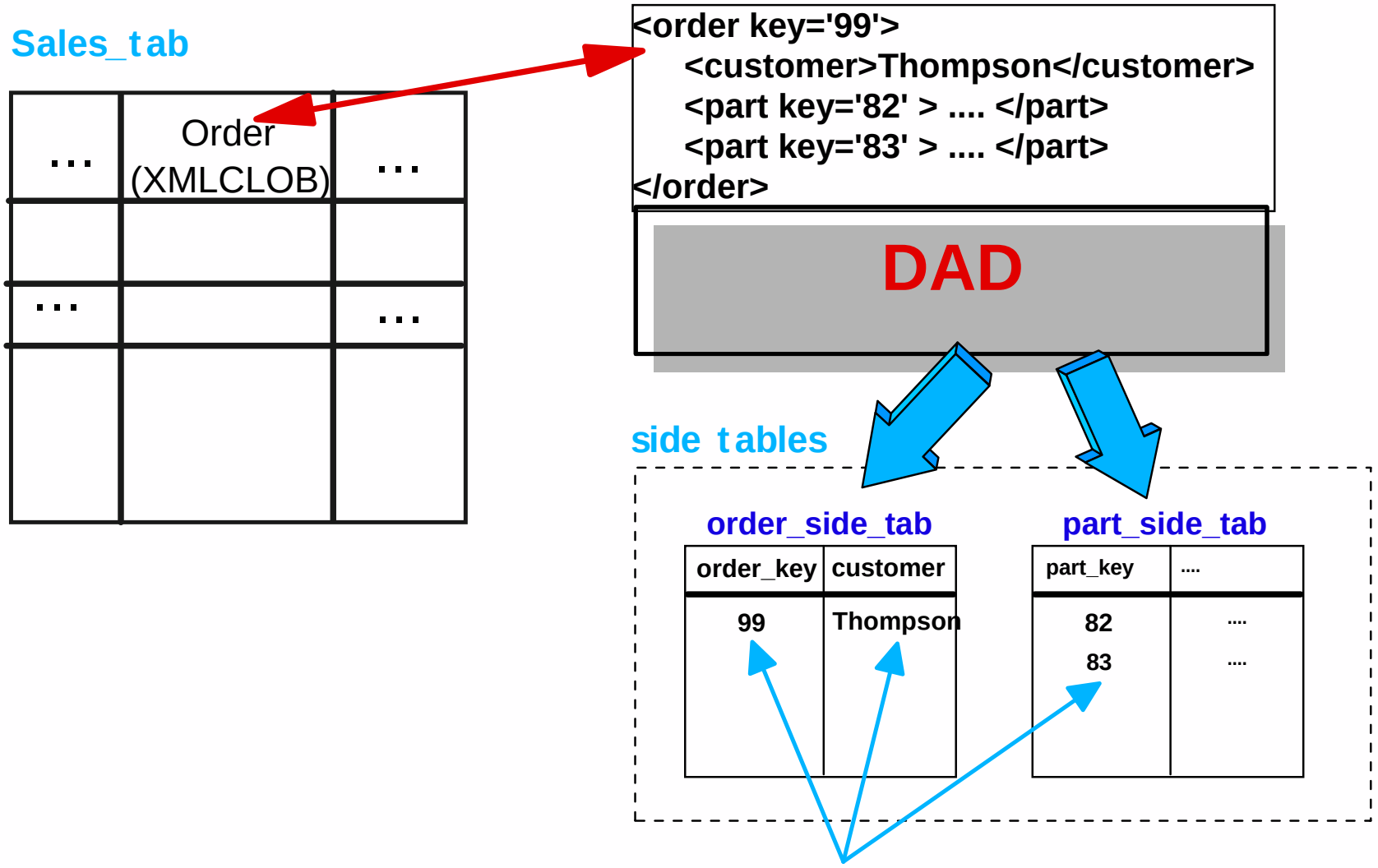
XML Columns - reasons for choosing

- XML documents already exist and you want to archive them
- Want to store intact XML documents
- Documents are read often and updated rarely
- Know which elements or attributes will be frequently searched
- Performance of updates is not critical
- Want to keep XML documents external to DB2 on a file system

DB2 XML Columns

- XML documents stored intact in DB2
  - Optional validation against DTDs
  - Can also use text extender search facilities
- Supplied User Defined Types
  - XMLVarchar, XMLCLOB, XMLFile
- Supplied User Defined Functions:
  - To store or retrieve entire XML documents into/from DB2
  - To extract or update XML elements or attribute values using XPath notation
  - To extract XML fragments using XPath notation
- Document Access Definition: To define special indexes {side tables}
  - for speedy structured search of XML document content

Using Side Tables for Fast Searches in XML Documents



Indexes can be created for frequently accessed tables

Side tables

- Improve query performance
- DB2 tables created to store indexes and frequently searched data
- The tables are synchronized with the XML document using DB2 triggers
  - NB if you add a side table, you will need to populate it yourself with existing data
- **Indexes for side tables must be created manually after column has been enabled**

DAD Side table Example

- Table name tag for each side table
- Define location path (XPath) of element/attribute to be indexed
- Specify data type if you want a different data type in the DB2 tables

```
<DAD>
<dtdid>c:\samples\dtd\getstart.dtd</dtdid>
<validation>YES</validation>
<Xcolumn>
  <table name="order_side_tab">
    <column name="order_key"
      type="integer"
      path="/Order/@key"
      multi_occurrence="NO"/>
    <column name="customer"
      type="varchar(50)"
      path="/Order/Customer/Name"
      multi_occurrence="NO"/>
  </table>
 </Xcolumn>
</DAD>
```

Side table consideration

- If Multi-occurrence=yes for an element or attribute
  - it must have it's own side table - no other columns allowed
  - DB2 XML extender will automatically assign a sequence number to make them unique
- Recommend side table names are fully qualified in the DAD
  - else the userid for the enable column becomes the default schema
- Beware of management of side tables - eg if you drop table, side tables and entry in XML_USAGE will remain
- If you are planning on using -v to create a default view ensure each of the side tables have unique column names
- Do not use rowid as the key for side tables

DAD Checker

- Standalone program which validates the DAD is semantically correct
- Produces two output files with messages - one XML and one plain text
- Checks include
  - DAD against DTD validation
  - Duplicate leaf element and attribute detection
  - Missing type attribute detection
  - Missing table declaration
  - Missing text node or attribute detection
  - For fixpak 3 and before attribute node tags must be prior to element node tags for the same table
- Download from
  http://www.ibm.com/software/data/db2/extenders/xmlext/download/beta/dadchecker.html
  - will ship with next releases of WSAD and DB2

DB2 Data Management Software

IBM

e business software

Websphere Studio Application Developer
An Overview

Joan Haggarty,
WSAD XML Tools Development, IBM Toronto

IBM Software Group

Agenda

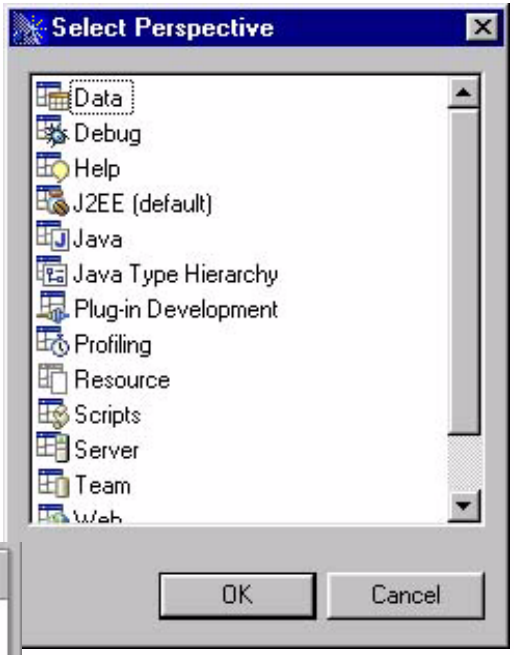
- Overview of WSAD
- General WSAD Concepts
- Working with XML
- Relational Data Access
- Java Development
- More information & Questions

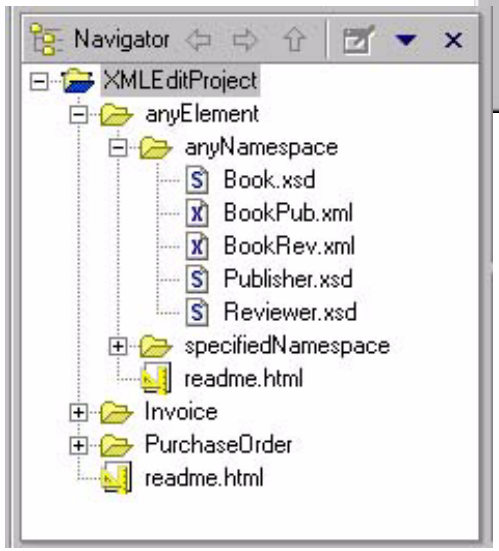
WebSphere Studio in a Nutshell

- Comprehensive Development Environment
- End-to-end support for application development:
  - **Java/J2EE**
  - **XML**
  - **Web Applications**
  - **Web Services**
  - **Relational Data**
  - **Debugging & Profiling**
  - **Team support**
  - **Server configuration**
  - **Unit testing**
  - **Plug-in development**
- Highly customizable
- Extensible

What's What and What's Where

- Standard application stuff - menus, toolbars...
- Multi-panel environment
- Views
- Perspectives - canned and customized
- Navigator - Projects, folders, resources...
- Editors - source and design
- "New" wizard

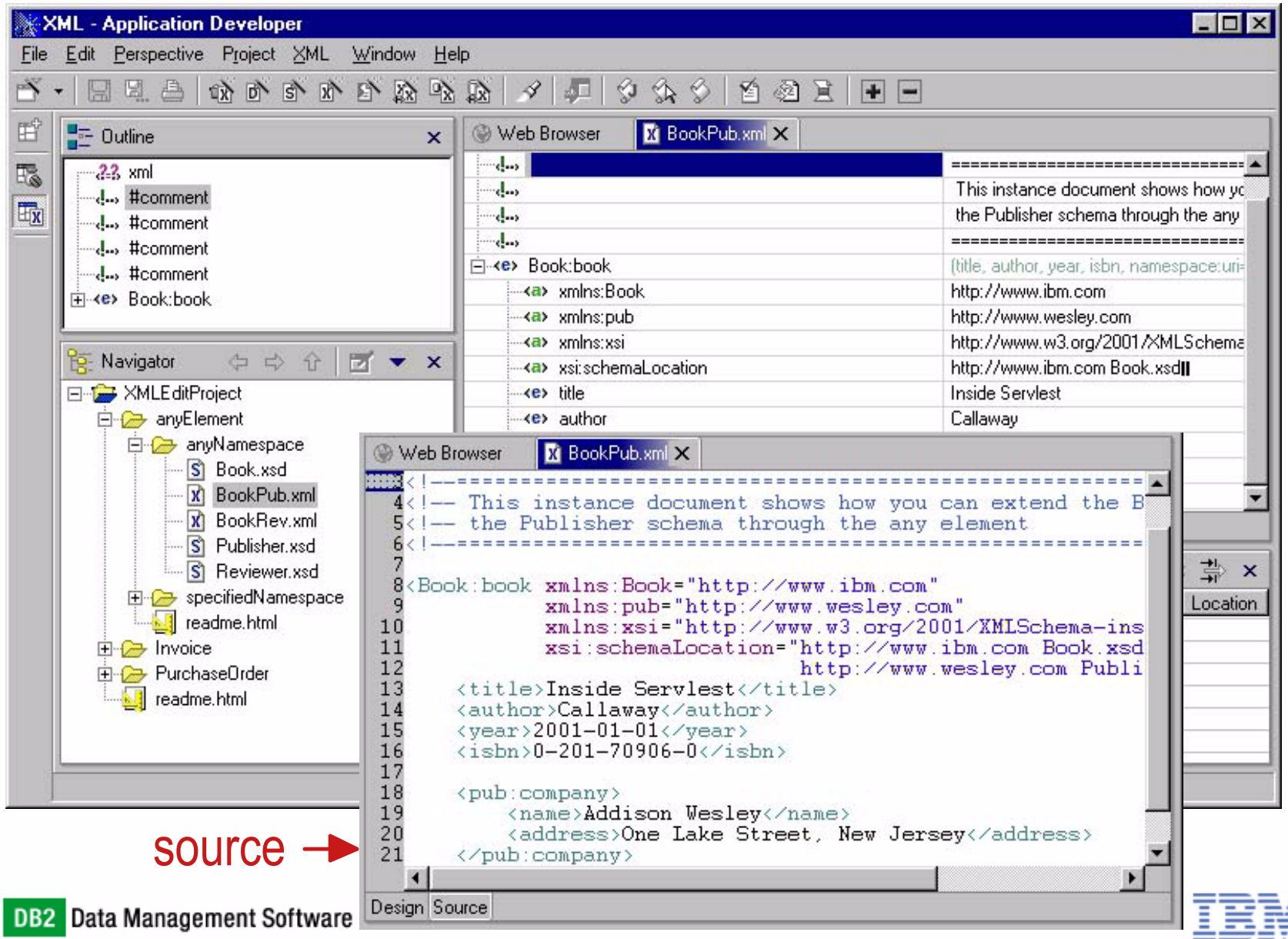




XML Perspective

- Outline, Navigator & Tasks views and editors
- XML, DTD, XML Schema, Java Bean Gen, XSL Trace, XML Mapping, XML from SQL
- Using the XML Perspective
  - DTD Editor
  - XML Editor
  - RDB to XML Mapping

XML Perspective - In pictures



DB2 Data Management Software

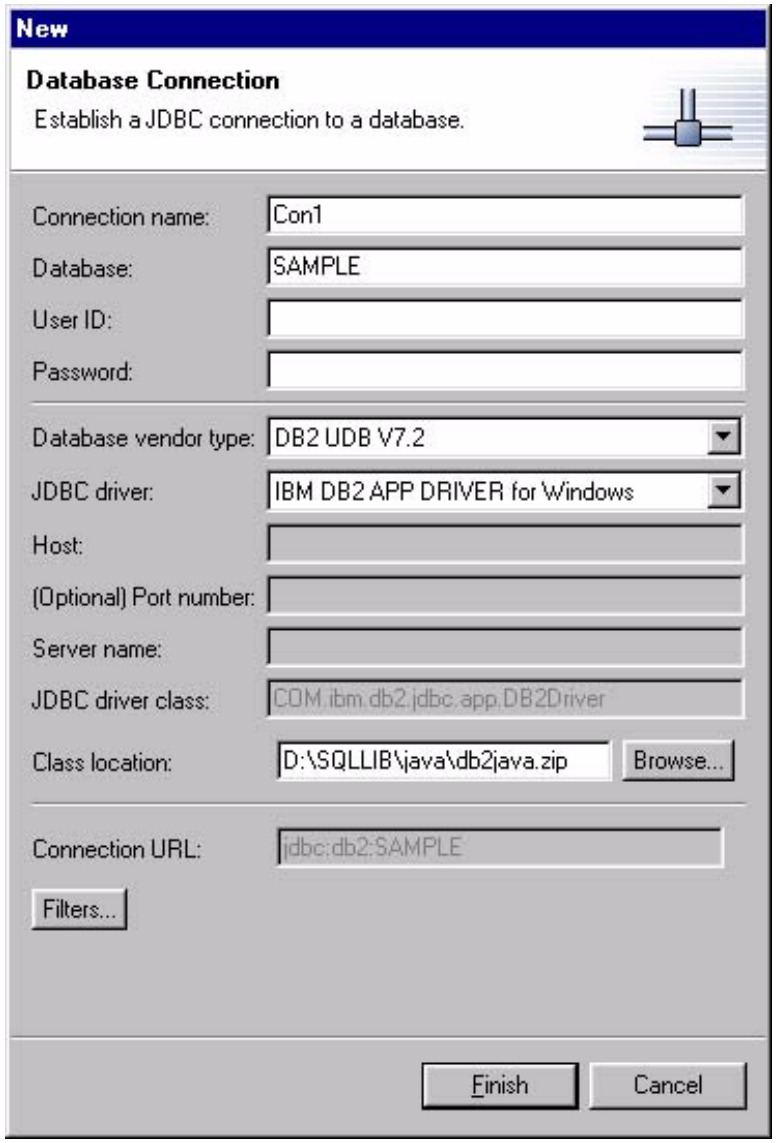
IBM

Data Perspective

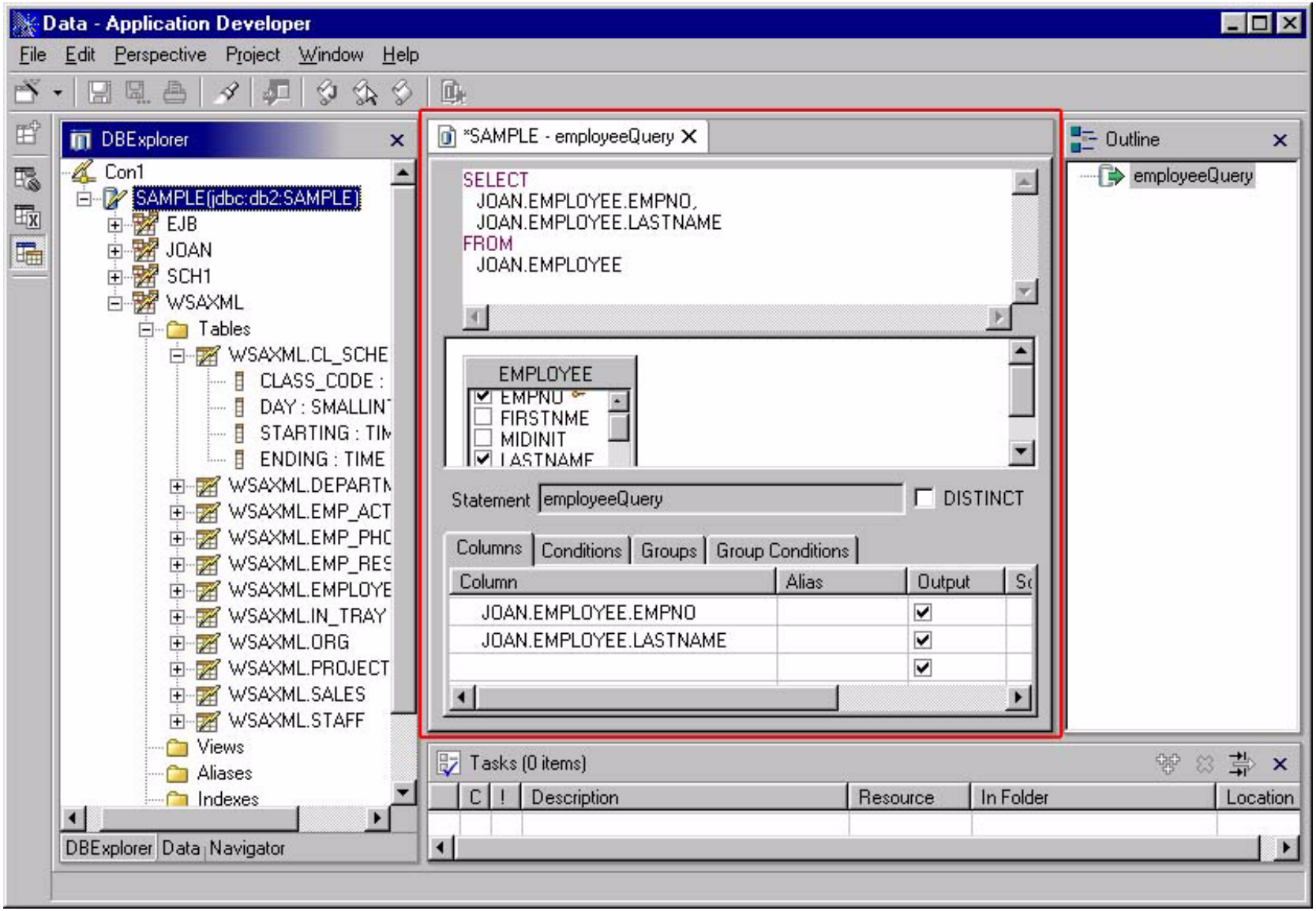
- DB Explorer, Data, Navigator & Tasks views and editors

- Tools:
  - Relational Schema Centre
  - SQL Builder
  - DDL Generation

- Using the Data Perspective
  - database connection
  - import to local
  - browse database schema
  - edit database schema
  - generate DDL
  - create SQL statements

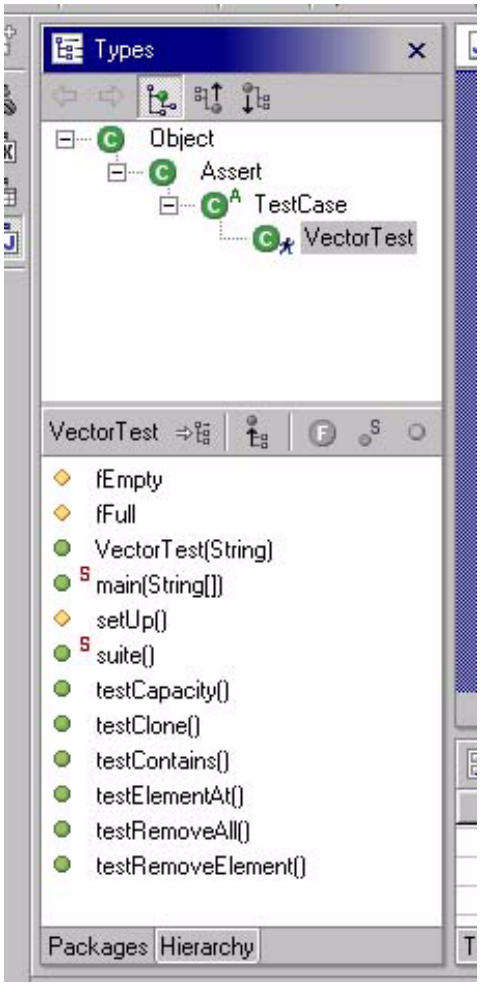


Data Perspective - In Pictures

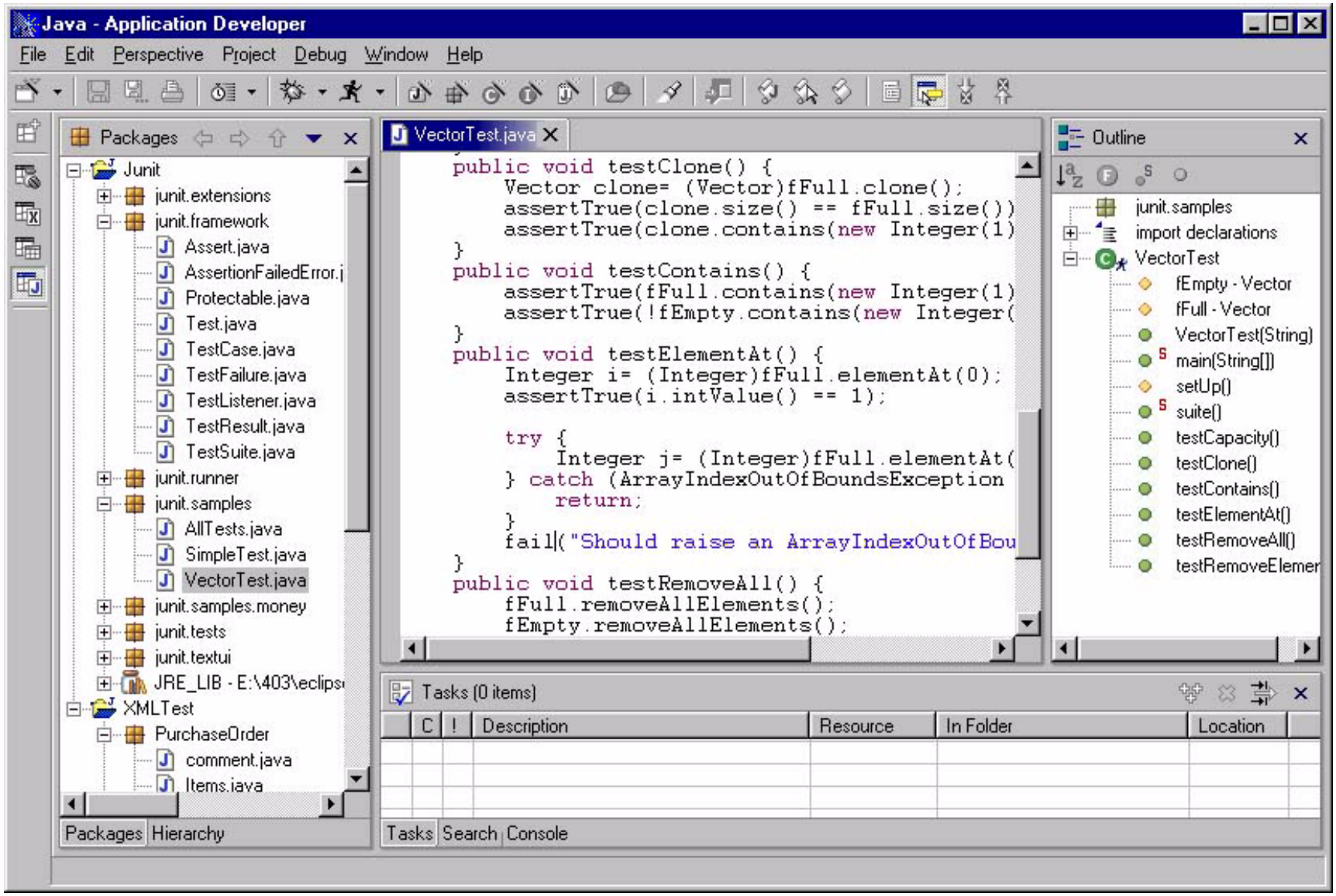


Java Perspective

- Packages, Hierarchy & Java Beans views and Java editor
- Tools:
  - wizards - packages, classes, interfaces, methods, source folder
  - browsing support
  - debugger & profiler
  - project properties - define source folders, build path and output
- Java features
  - class navigation
  - editor content assist & syntax highlighting
  - incremental build - manual or automatic
  - tight integration with Tasks view



Java Perspective - In Pictures



Packaging Information

- Beta downloadable from
  - http://www.ibm.com/software/ad/studioappdev/
- Future plans are to package a restricted use copy of of WSSD with the DB2 XML Extender (UWO)
  - WSSD is a lower-priced version containing a subset of the functionality in WSAD
- You can obtain DB2 XML Extender samples by
  - New - Other - Examples - XML - DB2 XML Extender
- As an aside - you can generate DDL from a DTD (Via schema)
  - Could be used where tables do not already exist

More Information...

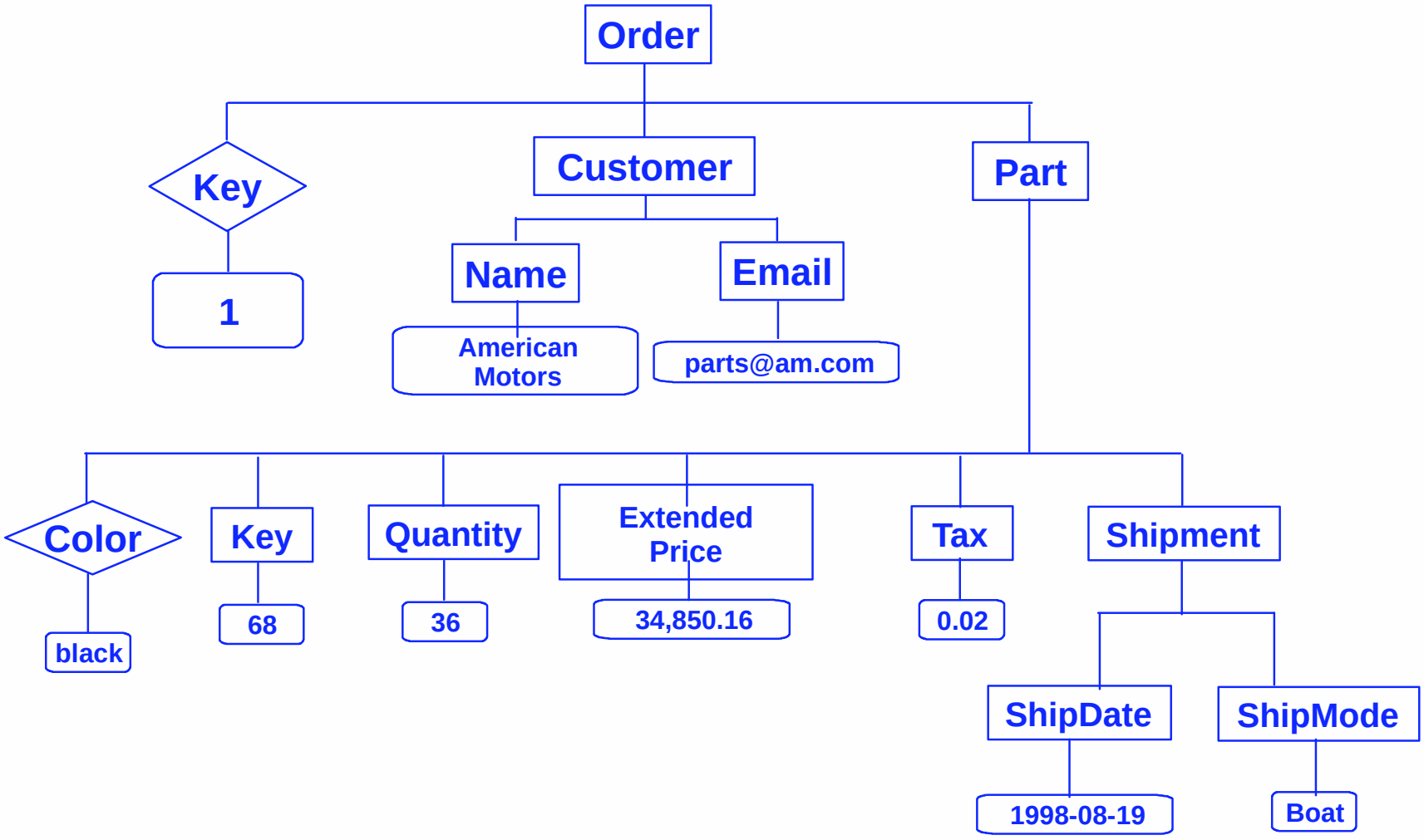
- WebSphere Developer Technical Journal
  - http://www7b.boulder.ibm.com/wsdd/techjournal/
- XML Article Series
  - ✓ Part 1: Developing XML Schema (Nov 2001)
  - ✓ Part 2: Creating an SQL Query (Jan 2002)
  - ✓ Part 3: SQL and XML Together (Feb 2002)
  - ✓ Part 4: Exploring the XML Editor (March 2002)
  - ✓ Part 5: Exploring the RDB to XML Mapping Editor (April 2002)
  - Part 6: ... XML Namespace Demystified
  - ...

Overview to Lab Exercises : Part 1

Composing an XML document scenario

- Your task
  - Build XML documents from data in the Purchase Order database SALES_DB
  - Using Websphere Studio tooling
- Service department:
  - Will use this information when working with customer requests and complaints
  - Requested specific data for inclusion
  - Provided a recommended structure for the XML documents
  - Wants the document structured by the order number
- What will you do?
  - Build a DTD defining the structure of the XML document
  - Create a DAD file that maps DB2 table columns to an XML document structure
  - Create an XML collection
  - Compose an XML document from data in these tables

Hierarchical structure of XML document



DB2 Tables used for XML Document Composition

ORDER_TAB

Column Name	Data type
ORDER_KEY	INTEGER
CUSTOMER	VARCHAR(16)
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)

PART_TAB

Column Name	Data type
PART_KEY	INTEGER
COLOR	CHAR(6)
QUANTITY	INTEGER
PRICE	DECIMAL(10,2)
TAX	REAL
ORDER_KEY	INTEGER

SHIP_TAB

Column Name	Data type
DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(128)
PART_KEY	INTEGER

SQL Statement to be used

```
SELECT
        O.ORDER_KEY,
        O.CUSTOMER_NAME,
        O.CUSTOMER_EMAIL,
        O.CUSTOMER_PHONE,
        P.PART_KEY,
        P.COLOR,
        P.QUANTITY,
        P.PRICE,
        P.TAX,
        S.DATE,
        S.MODE,
        S.COMMENT
  FROM
        XMLDEMO.ORDER_TAB AS O,  XMLDEMO.PART_TAB AS P,  XMLDEMO.SHIP_TAB AS S
  WHERE
        O.ORDER_KEY = P.ORDER_KEY
        AND P.PART_KEY = S.PART_KEY
        AND O.ORDER_KEY = 1
        AND P.PRICE > 20000
  ORDER BY
        ORDER_KEY ASC,
        PART_KEY ASC
```

Lab 1 Tasks - What you will do

- Familiarisation of WebSphere Studio
  - Create a Web Project
  - Create the sample XML Extender Project - **salesdb**
  - Import an existing XML file as **Myorder.xml**
  - Generate a DTD **Myorder.dtd** defining the structure of the XML document
  - Generate a DAD based on SQLMapping Technique
    - Create **Myorder.rmx** mapping file - Map columns to the XML document structure
    - Generate **Myorder.dad**
  - Optionally generate a DAD that uses RDB node Mapping
    - Create **Myorderrdb.rmx** mapping file
    - Generate **Myorderrdb.dad**

Finally...

- Tomorrow, we'll finish this series of education by covering how to enable columns and collections and store/retrieve XML data.
- We will also discuss setup and usage considerations, including code page issues and 390 specifics