# Integrating Oracle 10g XML: A Case Study

**Coleman Leviter**
**Arrow Electronics**
**IT Software Systems Engineer**
**cleviter@ieee.org**

# Project History

- WMS Group – Nine years
- VAX Rewrite (.for ) to UNIX (.c, .pc)
- VAX Forms to Oracle Forms
- TIFF file migration to Oracle
- WMS Development and Support

# Terminology

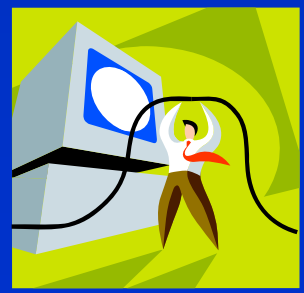
- WMS – Warehouse Management System
- TMS – Transportation Management System
- XML – Extensible Markup Language
- W3C - World Wide Web Consortium

# Presentation Objectives

- WMS – Current Transportation Management System
- TMS goals and objectives
- XML – W3C, Oracle XML DB
- Questions

# Project Goals – External TMS w/WMS

- Use  a third party Transportation Management System
- Eliminate maintenance updating table rates, adding new carriers, find best routes, rate shopping
- Fit the needs of a global WMS



# Project Goals – TMS Gains (cont'd)

- Eliminate the following warehouse equipment
  - GSS – Global (export) Shipping System
  - UPS
  - FEDEX
  - DHL
  - Supply Chain Solution – Gemini System
  - Bill of Lading System (BOL)
  - Rate Sheets – manual

# Oracle XML History

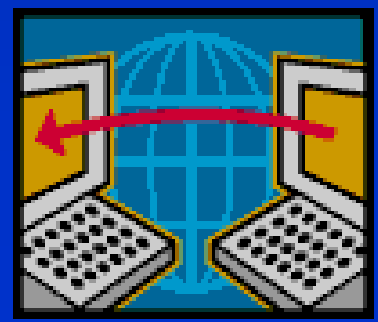
- 8i 1998 - XML Api
- 9i 2001 - XML Storage
- 10g 2004 - XPath
- 11g 2007 - Binary XML

# Architecture Challenges

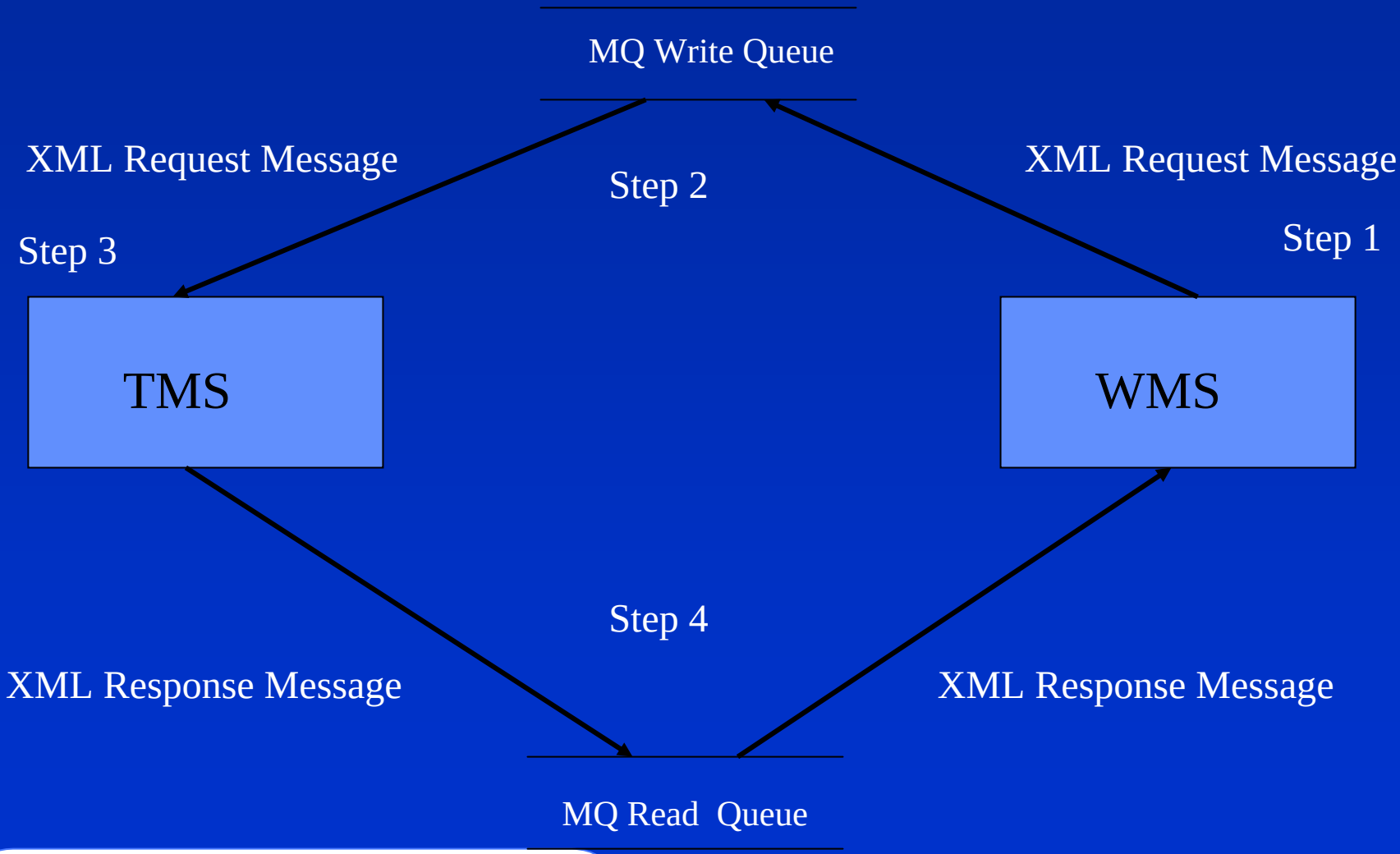
- Currently, WMS uses internal rates and routing
- Objective – wherever WMS performs routing and rate lookup, break that connection and use TMS
- Simple - 😐😐😐

# Steps

- Wherever rate and routing evaluation occur within WMS, connect to TMS
- Use MQ Series for the communications layer – guaranteed message delivery
- Use Oracle XML DB for messages

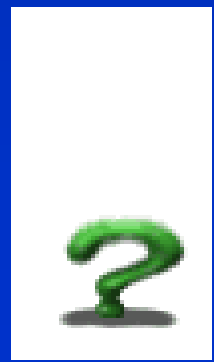


# WMS/TMS Data Flow



# What is XML

Short for *eXtensible Markup Language,* a specification developed by the W3C. XML is a pared-down version of SGML or Standard Generalized Markup Language, designed especially for Web documents. It allows designers to **create their own customized tags**, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.



# SQLX Definitions - Query

## New Terms

Defined by ISO/IEC 9075-14:2003.

- XMLAGG() is an aggregate function. It takes a collection of XML fragments and returns an aggregated XML document.
- XMLELEMENT() takes an element name for *identifier*, an optional collection of attributes for the element, and arguments that make up the content of the element
- XMLFOREST() converts each of its argument parameters to XML, and then returns an XML fragment that is the concatenation of these converted arguments.

# SQL Query

## Conventional SQL

```
SELECT   '1'   AS CARTONNUMBER,
         'CUSTOM'   AS PACKAGETYPE,
         '121' AS WEIGHT,
         ' '   AS LENGTH,
         ' '   AS WIDTH,
         ' '   AS HEIGHT
FROM DUAL
```

# SQL Query Results

## Conventional SQL Results

| CARTONNUMBER | PACKAGETYPE | WEIGHT | LENGTH | WIDTH | HEIGHT |
|---|---|---|---|---|---|
| 1 | CUSTOM | 121 | <blank> | <blank> | <blank> |

# SQLX Query

```
SELECT xmlagg (xmlelement("PACKAGES",
              xmlforest
                  ('1' AS "CARTON",
                   'CUSTOM' AS "PACKAGE",
                   '121' AS "WEIGHT",
                   ' ' AS "LENGTH" ,
                   ' ' AS "WIDTH",
                   ' ' AS "HEIGHT"
                   )
                )
              ) AS "XML"
FROM DUAL
```

# SQLX Query Results

## XML Fragment

```
-- from XMLAGG
<PACKAGES>              -- from XMLELEMENT
  <CARTON>1</CARTON> -- from XMLFOREST
  <PACKAGE>CUSTOM</PACKAGE>
  <WEIGHT>121</WEIGHT>
  <LENGTH/>
  <WIDTH/>
  <HEIGHT/>
</PACKAGES>             -- from XMLELEMENT
```

# SQLX Examples

## XMLElement()

```
SELECT XMLELEMENT("Emp", XMLATTRIBUTES(e.employee_id
   AS "ID", e.last_name),
   XMLELEMENT("Dept", e.department_id),
   XMLELEMENT("Salary", e.salary)) AS "Emp Element"
FROM employees e
WHERE e.employee_id = 206;

Emp Element
--------------------------------------------------------
   ------------
<Emp ID="206" LAST_NAME="Gietz">
   <Dept>110</Dept>
   <Salary>8300</Salary>
</Emp>
```

# SQLX Examples (cont'd)

## XMLForest()

```
SELECT XMLELEMENT("Emp", XMLFOREST(e.employee_id,
  e.last_name, e.salary)) "Emp Element"
FROM employees e
WHERE employee_id = 204;

Emp Element
--------------------------------------------------------------
<Emp>
  <EMPLOYEE_ID>204</EMPLOYEE_ID>
  <LAST_NAME>Baer</LAST_NAME>
  <SALARY>10000</SALARY>
</Emp>
```

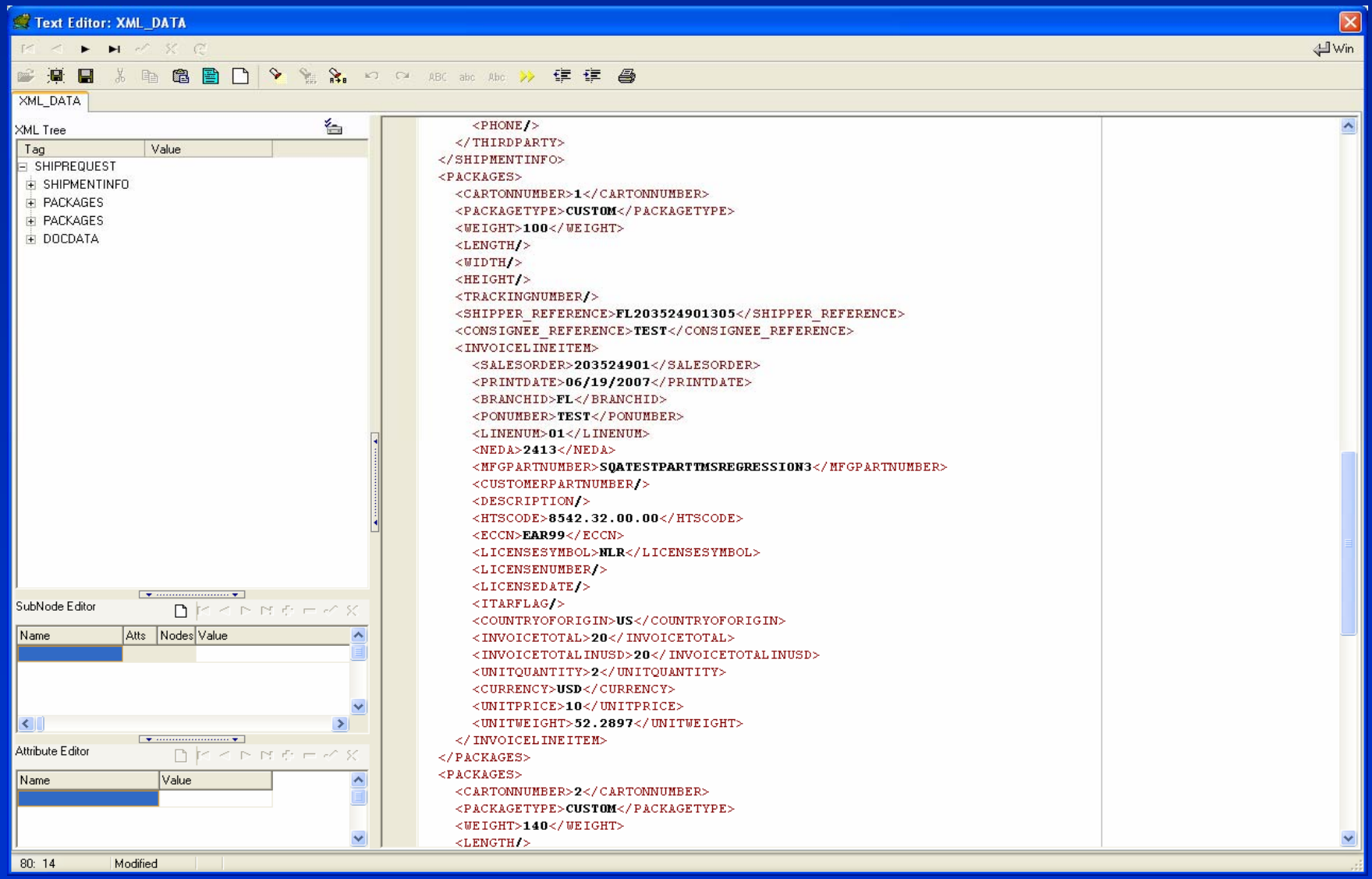
# SQLX Examples (cont'd)

## XMLAgg()

```
SELECT XMLELEMENT("Department",
                  XMLAGG(
                    XMLELEMENT("Employee", e.job_id||'
  '||e.last_name)
                    ORDER BY last_name)) as "Dept_list"
FROM employees e
WHERE e.department_id = 30;

Dept_list
 ----------------------------------------------------
  ----------
<Department>
   <Employee>PU_CLERK Baida</Employee>
   <Employee>PU_CLERK Colmenares</Employee>
   <Employee>PU_CLERK Khoo</Employee>
   <Employee>PU_MAN Raphaely</Employee>
   <Employee>PU_CLERK Tobias</Employee>
</Department>
```

# XML Document – using TOAD



# XPath Definition - Shredding

- **XPath** (XML Path Language) is an expression language for extracting portions of an XML document, or for calculating values (strings, numbers, or Boolean values) based on the content of an XML document. Used for document shredding. Insert data into relational table.
- **Expression** **Description**

| | |
|---|---|
| *nodename* | Selects all child nodes of the named node |
| / | Selects from the root node |
| // | Selects all nodes in the document from the current node that match the selection no matter where they are |
| . | Selects the current node |
| .. | Selects the parent of the current node |
| @ | Selects attributes |

- /A/B[1] Select the first B node from the A node

# XPath Vocabulary

- XMLTYPE - CLOB under the covers
- extract() - returns document fragment
- XMLType Constructor() - converts CLOB to XMLTYPE
- isFragment() - Returns true (1) if the XMLType contains a document fragment. A document fragment is an XML document without a Root Node. Document fragments are typically generated using the extract() function and method.
- getClobVal() - Returns a CLOB containing an XML document based on the contents of the XMLType.
- getRootElement() - Returns the name of the root element of the XML document contained in the XMLType.

# XML Document Shredding - XPath

- Table Using XMLType Column

```
CREATE TABLE TMS_OUTBOUND_INTERFACE

  TMS_SEQ     NUMBER(10), (PK1)

  XMIT_DATE   TIMESTAMP(6), (PK2)

  SORD_NUM    VARCHAR2(12 BYTE),

  SORD_REL    VARCHAR2(4 BYTE),

  PRNT_DATE   DATE,

  MSG_TYPE    VARCHAR2(10 BYTE),

  XML_DATA    SYS.XMLTYPE  (CLOB under the covers)
```

# XML Document Shredding - XPath

```
SELECT

   tabx.XML_DATA.extract('//INVOICELINEITEM/SALESORDER/text()').
      getstringval() AS salesorder,

   tabx.XML_DATA.extract('//PACKAGES/CARTONNUMBER/text()').
      getstringval()  AS cartonnumber,

   tabx.XML_DATA.extract('/SHIPREQUEST/SHIPMENTINFO/FACILITYCODE/
      text()').getstringval()  AS facilitycode,

    tabx.XML_DATA

FROM tms_outbound_interface tabx

WHERE tabx.MSG_TYPE = 'SHIP_TEST'

AND tabx.XML_DATA.extract('//INVOICELINEITEM/SALESORDER/text()').
   getstringval() = '987654321'

AND tabx.tms_seq = 122
```

# XML Document Shredding (cont'd)

## XML Results

| SALESORDER | CARTONNUMBER | FACILITYCODE | XML_DATA |
|---|---|---|---|
| 987654321 | 1 | RNO | <xmltype> |

# XML Document Shredding (cont'd)

## XPATH Results

```
<SHIPREQUEST>
  <!-- this is 182, svia J4 -->
  <SHIPMENTINFO>
    <FACILITYCODE>RNO</FACILITYCODE>
    <TARGETDATE>05/10/2007</TARGETDATE>
  …
  …
     <NUMBEROFSKIDS/>
    <BOLDESCRIPTION>ELECTRONIC COMPONENTS</BOLDESCRIPTION>
    <BOLCLASS>85</BOLCLASS>
    <BOLCOMMENTS/>
  </DOCDATA>
</SHIPREQUEST>
```

# XPath

```
SELECT  tabx.XML_DATA.extract('
  //PACKAGES[1]
  /INVOICELINEITEM/SALESORDER/text()').
  getstringval() AS salesorder,
  tabx.TMS_SEQ,
  tabx.XML_DATA

FROM tms_outbound_interface tabx

WHERE tabx.MSG_TYPE = 'SHIP'

AND tabx.XML_DATA.extract('
  //PACKAGES[1]/INVOICELINEITEM
  /SALESORDER/text()').getstringval()
  LIKE '203524901%'   (predicate: like)
```

# XML Data Transfer Efficiency

Efficiency is the ratio of meaningful valid data to total data

$$\text{Efficiency} = \frac{\text{Valid Data}}{\text{Valid Data + XML Elements}} \times 100$$

# XML Data Transfer

**Excerpt from a typical XML Document (447 bytes):**

```
<SHIPREQUEST>
  <!-- this is 182, svia J4 -->   <----------------------- comment
  <SHIPMENTINFO>
    <FACILITYCODE>RNO</FACILITYCODE>
    <TARGETDATE>05/10/2007</TARGETDATE>
     …                          ^------------------------  typical data…
     <NUMBEROFSKIDS/>   <------------------------ empty element
    <BOLDESCRIPTION>ELECTRONIC COMPONENTS</BOLDESCRIPTION>
    <BOLCLASS>85</BOLCLASS>          ^-------- typical data
    <BOLCOMMENTS/>
  </DOCDATA>
</SHIPREQUEST>
```

**Using the Efficiency Formula**

$$\frac{\text{447 Bytes Message Data}}{\text{447 Bytes Message Data + 3437 Bytes XML Element Definition (= 3884 total bytes)}} \times 100$$

**= 12 % efficient**

# Binary Data Transfer

## Sales Complete Message (SCP)

| Field Name | Field Type | Size | Bit Position |
|---|---|---|---|
| Overhead Data | Alphanumeric | 32 | 1 - 32 |
| Message Code | 'SCP* ' | 4 | 33-36 |
| Entering Location | Alphanumeric | 3 | 37-39 |
| Sales Number | Alphanumeric | 6 | 40-45 |
| Version Number | Alphanumeric | 2 | 46-47 |
| Cartons for Shipping | Numeric | 2 | 48-49 |
| Shipping Charges | Alphanumeric | 9/2 | 50-58 |
| Date (YYMMDD) | Alphanumeric | 6 | 59-64 |
| Carrier | Alphanumeric | 16 | 65-80 |
| Carrier Number | Alphanumeric | 19 | 81-99 |
| Shipment Weight | Alphanumeric | 4 | 100-103 |
| Number of Orders | Alphanumeric | 2 | 104-105 |
| Ship Code | Alphanumeric | 2 | 106-107 |
| Spare 1 | Alphanumeric | 3 | 108-110 |
| Spare 2 | Numeric | 7 | 111-117 |
| Spare 3 | Alphanumeric | 4 | 118-121 |
| Spare 4 | Alphanumeric | 1 | 122-122 |
| Spare 5 | Alphanumeric | 1 | 123-123 |
| End | Alphanumeric | 27 | 124-150 |

# Binary Data Transfer Efficiency

Efficiency is the ratio of meaningful valid data to total data

$$\text{Efficiency} = \frac{\text{Valid Data}}{\text{Valid Data + XML Elements}} \times 100$$

Example Binary Data Message

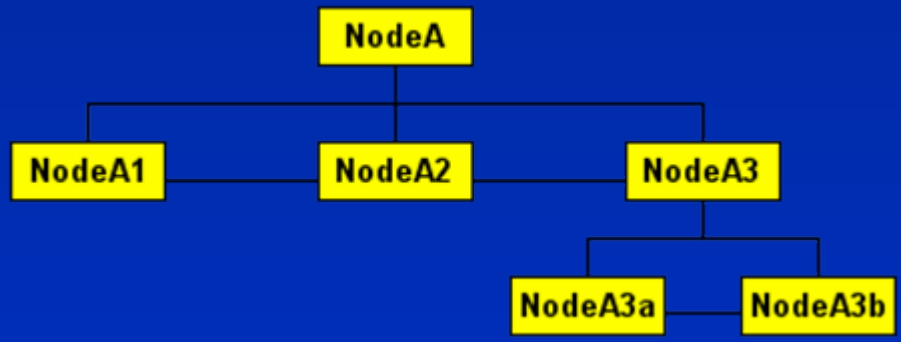
$$\frac{\textbf{118 bits (pos 33- 150)}}{\textbf{(Message data) 118 bits + 32 bits (overhead data)}} \times 100 = \textbf{79\% efficient}$$

In other words almost 90% of each transmission is not pure data transfer, it is Meta Data

Using element compression: 676 (26**2) + 26 elements: <ab> <data> </ab> = 9 characters

# Document Object Model (DOM)

- XML access defined as a tree structure
- Available with Oracle's 10g XDK (XML Devloper's Kit) JAVA/C/C+
- Navigation
  - NodeA.firstChild = NodeA1
  - NodeA.lastChild = NodeA3
  - NodeA.childNodes.length = 3
  - NodeA.childNodes[0] = NodeA1
  - NodeA.childNodes[1] = NodeA2
  - NodeA.childNodes[2] = NodeA3
  - NodeA1.parentNode = NodeA
  - NodeA1.nextSibling = NodeA2
  - NodeA3.prevSibling = NodeA2
  - NodeA3.nextSibling = null
  - NodeA.lastChild.firstChild = NodeA3a
  - NodeA3b.parentNode.parentNode = NodeA




- Methods
  - insertBefore()
  - replaceChild()
  - removeChild()
  - appendChild()
  - cloneNode()

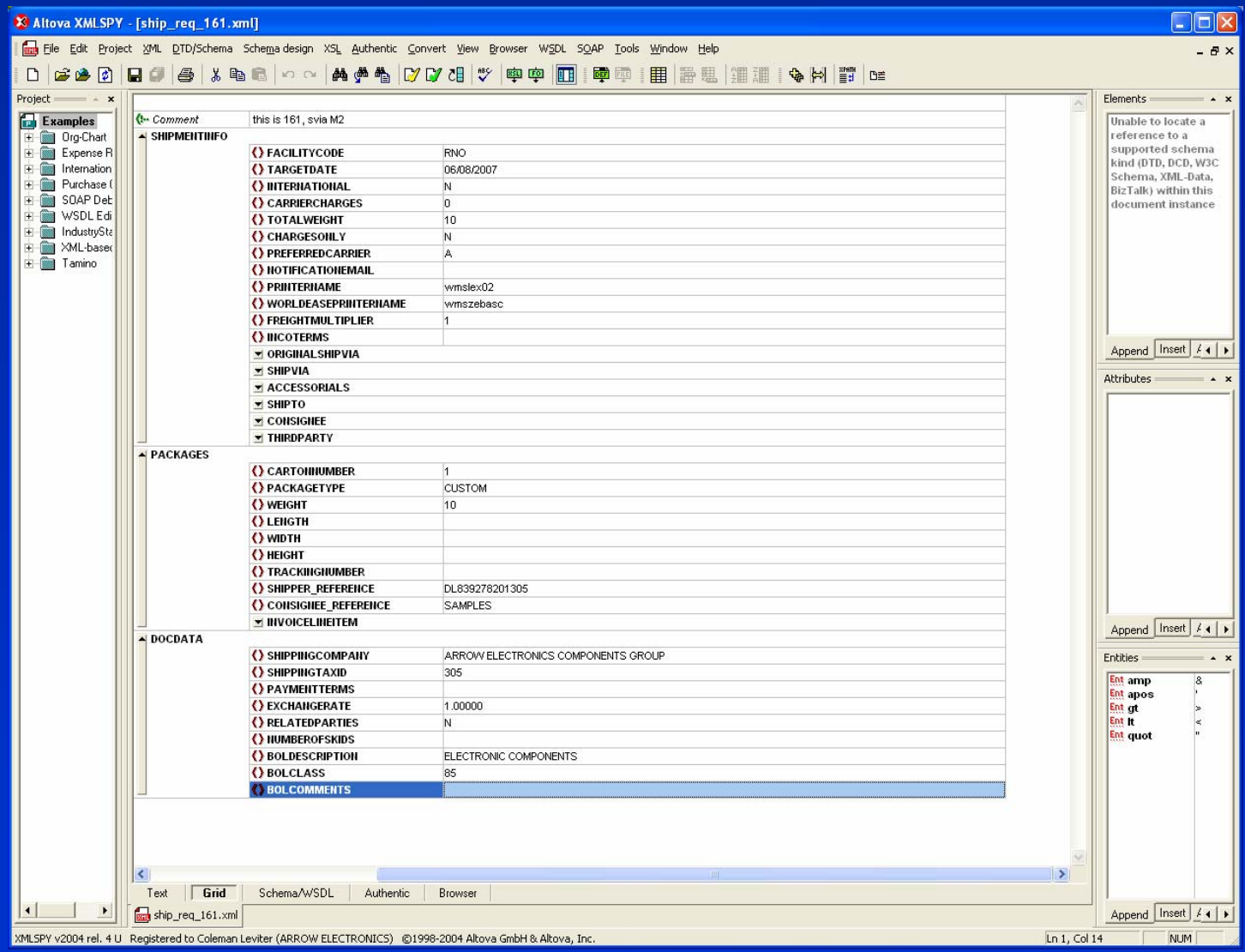
# Atomic Commits

- Use pragma autonomous_transaction for individual table commits. Similar to a sequence object.
- Introduced in oracle 8i
- Instructs the PL/SQL compiler to mark a routine as *autonomous* (independent).

```
*********************************************************************/

  FUNCTION INSERT_TMS_INTERFACE ( p_so_num            IN
   soh_orders_all.so_num%TYPE,
                                   p_so_rel            IN
   soh_orders_all.so_rel_num%TYPE,
                                   p_prnt_date         IN
   soh_orders_all.prnt_date%TYPE,
                                   p_msg_type          IN  VARCHAR2,
                                   p_outbound_message  IN  CLOB,
                                   p_response_message  IN  CLOB
                                 ) RETURN NUMBER
   IS
     PRAGMA autonomous_transaction;
     lcl_seq_num              NUMBER;
     lcl_invalid_xml_message VARCHAR2(100) DEFAULT NULL;
   BEGIN
      ...
```

# XMLSPY – Design XML



# XML Schema Confirmation

Steps for XML Schema Definition (XSD)

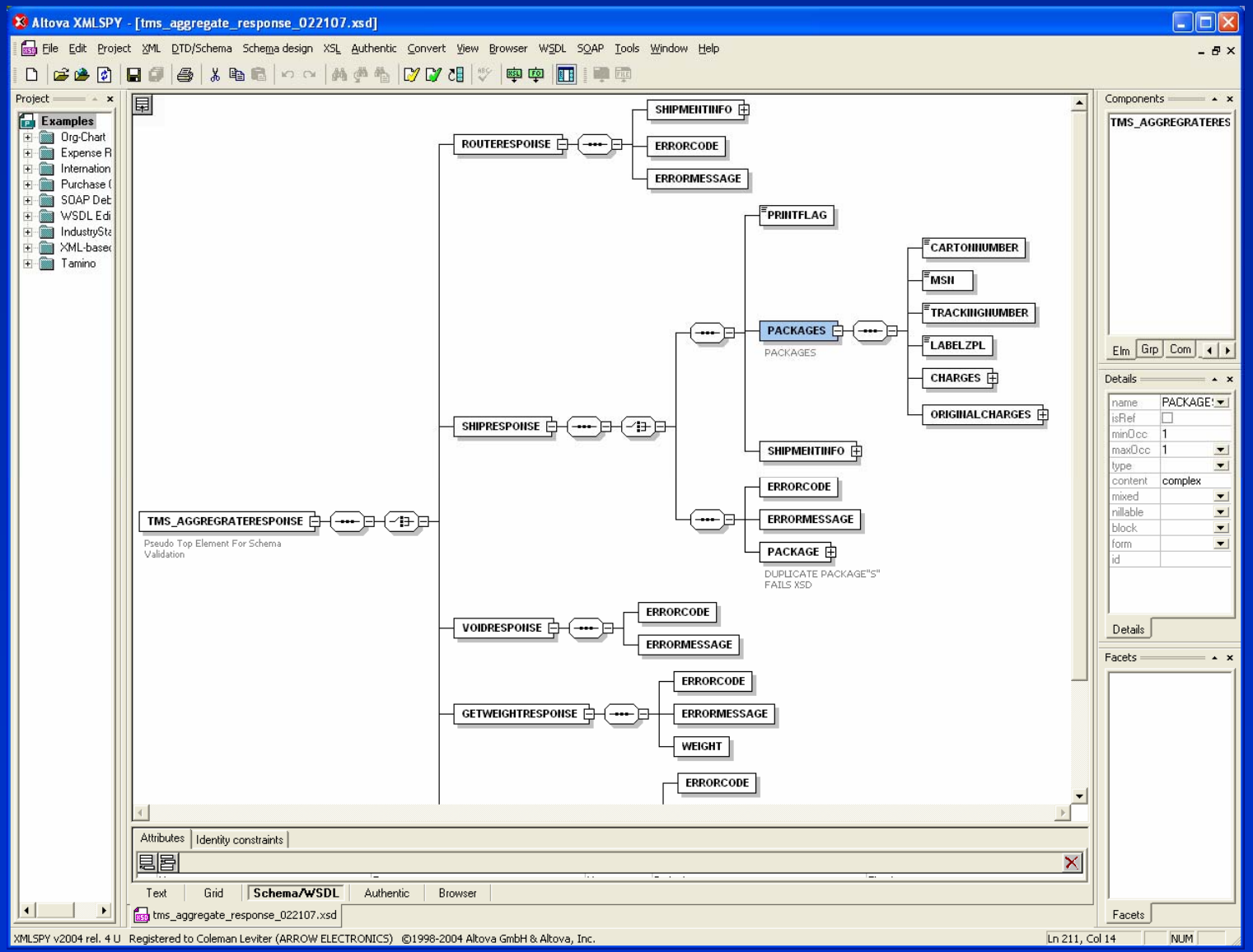
1) Delete the Schema: dbms_xmlschema.deleteSchema
2) Register the Schema:
   dbms_xmlschema.registerSchema('http://www.example.com/schemas/xml_tab_xsd.xsd'
3) Create table to hold XML documents

```
DROP TABLE xml_tab_xsd;
CREATE TABLE xml_tab_xsd (id number, xmlcol XMLType)
 XMLTYPE COLUMN xmlcol
 XMLSCHEMA "http://www.example.com/schemas/xml_tab_xsd.xsd"
 ELEMENT "purchaseOrder";
```

4) Populate the XML Document:

```
INSERT INTO xml_tab_xsd VALUES(xml_tab_seq.nextval, xmltype(
'<?xml version="1.0"?>
<ipo:purchaseOrder
```

# XMLSPY - XSD


Altova XMLSPY - [tms_aggregate_response_022107.xsd]
File Edit Project XML DTD/Schema Schema design XSL Authentic Convert View Browser WSDL SOAP Tools Window Help
Project
Examples
Org-Chart
Expense R
Internation
Purchase (
SOAP Det
WSDL Edi
IndustrySta
XML-base
Tamino
TMS_AGGREGRATERESPONSE
Pseudo Top Element For Schema Validation
ROUTERESPONSE
SHIPMENTINFO
ERRORCODE
ERRORMESSAGE
SHIPRESPONSE
PRINTFLAG
PACKAGES
PACKAGES
CARTONNUMBER
MSN
TRACKINGNUMBER
LABELZPL
CHARGES
ORIGINALCHARGES
SHIPMENTINFO
ERRORCODE
ERRORMESSAGE
PACKAGE
DUPLICATE PACKAGE"S" FAILS XSD
VOIDRESPONSE
ERRORCODE
ERRORMESSAGE
GETWEIGHTRESPONSE
ERRORCODE
ERRORMESSAGE
WEIGHT
ERRORCODE
Components
TMS_AGGREGRATERES
Elm Grp Com
Details
name PACKAGE
isRef
minOcc 1
maxOcc 1
type
content complex
mixed
nillable
block
form
id
Details
Facets
Facets
Attributes Identity constraints
Text Grid Schema/WSDL Authentic Browser
tms_aggregate_response_022107.xsd
XMLSPY v2004 rel. 4 U Registered to Coleman Leviter (ARROW ELECTRONICS) ©1998-2004 Altova GmbH & Altova, Inc.
Ln 211, Col 14
NUM

# References

- **Design Tools:  XMLSPY – 30 day free trial**

  **http://www.altova.com/simpledownload1.html?gclid=CJuhrey1m40CFRGsGgodPwqM5w**

- **Oracle XML DB Docs: full implementation plus extensions of XML**

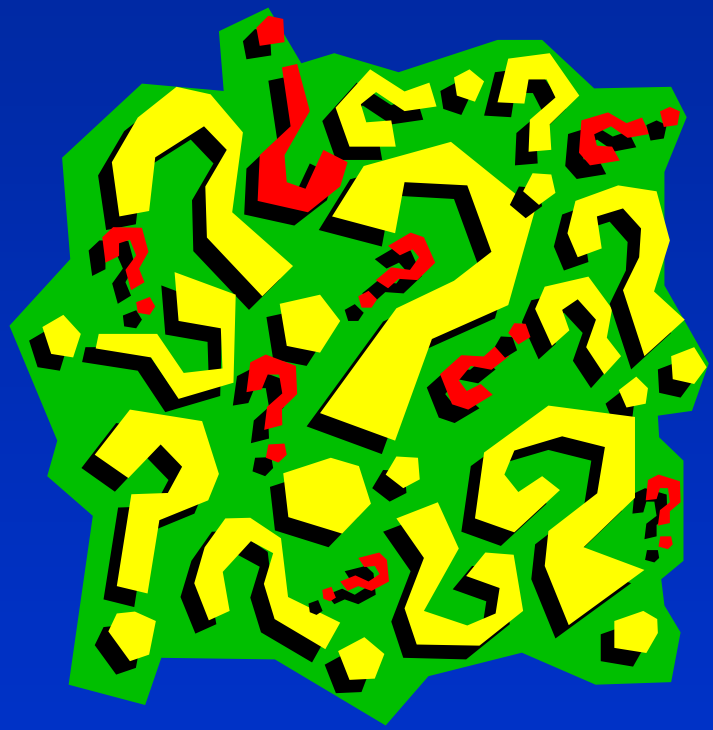
  **http://www.oracle.com/technology/tech/xml/xmldb/index.html**

- **Oracle's SQL 10g Doc**

  **http://oraclesvca2.oracle.com/docs/cd/B14117_01/server.101/b10759/functions204.htm**

- **Oracle Database 10g XML & SQL – Oracle Press**
- **Wikipedia.com (xpath, xml, xmlelement, etc.)**
- **XPath  - http://www.w3schools.com/xpath/xpath_syntax.asp**

# Questions



# Contact Information

- Coleman Leviter
- cleviter@ieee.com
- Integrating Oracle 10g XML: A Case Study