EZXMLSPY

PROJECT WORK DONE AT P-759

Premier Technology Group Pvt. Ltd.

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

OF BHARATHIAR UNIVERSITY, COIMBATORE

SUBMITTED BY

Anil.M.Jain 9938M0598

GUIDED BY

Mr. Suresh Sirigineedi, B.Tech.	Mr. Ramasubramanian K. MCA
EXTERNAL GUIDE	INTERNAL GUIDE

Department of Computer Science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore – 641 006

May 2002

Department of Computer Science and Engineering

Kumaraguru College of Technology

(Affiliated to Bharathiar University)

Coimbatore – 641 006

CERTIFICATE

This is to certify that the project work entitled

EZXMLSPY

Done by

Anil.M.Jain

9938M0598

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications of Bharathiar University.

[illegible] 30/4/02
Professor and Head

[illegible] 30/4/02
Internal Guide

Submitted for the University Examination held on 10-5-2002

[illegible] 10/5/02
Internal Examiner

[illegible]
External Examiner

Date : April 22, 2002.

CERTIFICATE

This is to Certify that Mr. **Anil M. Jain** of **(Kumaraguru College of Technology)**, did his project training in partial fulfillment of the Academic requirement of **MCA** curriculum with us from **November 2001 to April 2002**.

His Project Training title was **"EZXMLSpy"**, which he has completed successfully in the above said period

[illegible]

(Arun Arumugham)
Manager – HR & Admin

S. Suresh Babu

Suresh Sirigineedi
Sr.Project Leader

DECLARATION

I hereby declare that the project entitled **'EzXMLSpy – Generation of database from XML documents'**, submitted to Bharathiar University, Coimbatore as the project work of **MASTER OF COMPUTER APPLICATIONS** Degree during May 2002, is a record of original work done by me under the supervision and guidance of **Mr. Suresh Sirigineedi, Senior Project Leader, Premier Technology Group Pvt. Ltd., Chennai,** and **Mr. K. Ramasubramanian, Lecturer, Kumaraguru College of Technology, Coimbatore,** and this Project has not found the basis for the award of any Degree/ Diploma/ Associateship/ Fellowship or similar title to any candidate of any University.

Place: Coimbatore

Date: 30-04-2002

[illegible]

(Anil. M . Jain)

[illegible] 30/4/02

(Internal Guide)

S. Suresh Babu

(External Guide)

DECLARATION

I have not disclosed nor attempted to disclose any proprietary and/or confidential information related to the business of Premier Technology Group Pvt. Ltd. in this Project report that is being submitted by me after completion of my Project Training in PTG.

Signature: [illegible]

Name: Anil . M . Jain

Date: 30-04-2002

Place: Coimbatore

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude to **Dr. K. K. Padmanabhan, BSc., (Engg.), M. Tech, Ph. D., Principal, Dr. S. Thangasamy, Ph. D., Head Of the Department, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore** for providing me opportunity to take up this project.

I would like to place on record my thanks to my internal guide **Mr. K.Ramasubramanian, Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology,** whose encouragement proved to be stimulating. I am also grateful to all staff members of the Department for their support and guidance throughout this project work.

I am grateful to **Mr. Arun Arumugham, Manager - Human Resource and Administration,** and **Mr. Suresh Sirigineedi, Senior Project Leader, Premier Technology Group Pvt., Ltd., Chennai,** for giving me the opportunity to work on this project. I also wish to express my appreciation and thanks to **Mr. Krishna Perumal** for his keen interest and valuable suggestions throughout the project regarding XML.

My special thanks to **Ms. Sureka Rajeshwari,** Project Coordinator for her encouragement and support during the project period.

I am proud of my family for encouraging me whenever I was depressed, to face the challenges in the life and made the project a great success.

I am also grateful to my friends for their valuable comments and constructive criticisms, without which the success of this project would not have been possible.

SYNOPSIS

eXtensible Markup Language(XML) is currently the most promising language for delivering information on the World Wide Web. XML has a highly flexible syntax that allows us to use it to describe virtually any kind of information from a simple page to complex database. As XML becomes the data standard for e-business the amount of XML data being exchanged will grow exponentially.

This will lead to the requirement of persistent XML storage and hence the need for a database management system will also be required. The project aims to generate relational database from existing XML documents.

The XML documents to be uploaded to the database will contain information regarding table structures and the actual data to be stored. Regardless of which database format the XML documents represent the tool uploads it to any database format for which a Java DataBase Connectivity(JDBC) driver is available.

CONTENTS

1.1 Project Overview

EzXMLSpy is intended at building a tool that converts the XML data to its corresponding relational database.

XML a subset of SGML is a markup language designed specifically for delivering of information over the World Wide Web. It is the standard way to identify and describe data on the web. It is widely implemented and easy to deploy.

XML can be used to represent each field of information within each database record. This enables us to display the data in a variety of ways and search, sort and process the data in other ways. The response time will be considerably improved when data is accessed from a XML document than directly from the database.

XML can be used as an intermediary tool for transfer of database information in one platform to another and from one database format to any database format. Since XML consists of only text it can be easily transported over the Internet.

1.2 Organization Profile

About Premier Technology Group Private Limited

Premier Technology Group (PTG) is a provider of large-scale e-Business applications for customers in the insurance, investment and banking industries.

PTG delivers e-Business solutions that meet customers' needs in rapidly changing business environments. Utilizing a proven framework, customers are able to tap new markets, increase presence or provide better service in the industries they serve. PTG has developed the right infrastructure to do Quality Software Development & Research. It is registered with Software Technology Parks of India (STPI). It is also ISO 9001 certified. PTG has a strong strategic relationship with US based E-Z Data Inc., which also happens to be the parent Company of PTG, E-Z Data is a leader in Practice Management & Sales Force Automation Solutions for Insurance Industry. It has total user base of about 75,000 users, it has customer base in US, Canada and Australia.

With a total of over 75,000 user base, the company's clients include approximately 30 of the largest insurance, investment and banking companies in the United States, Canada and Australia. The company is a leader in Practice Management & Sales Force Automation Solutions for Insurance Industry. It is the leading provider of front office systems for insurance providers, investment advisors and managers.

In 1997 E-Z Data, Inc. facing competition and the shortage of IT resources in the U.S. established a development arm offshore. The result was the creation of Premier Software Technology Group Inc. (PSTG) in United States and Premier Technology Group Pvt. Ltd. (PTG) in India. The corporate office and main development center of PTG are in Nagpur. It also has a development center in Chennai. Other offshore offices are in Kuwait and Japan.

Mission

"Premier Technology Group's mission is to provide vital IT support for rapidly expanding organizations. In today's competitive global economy you need a partner with proven technology implementation experience."

Platforms

- Windows based applications using VC++, MFC, ODBC, OLE Automation, COM, DCOM, Windows CE, and VB.
- Virtually supporting all back-end RDBMS (UDB, MS SQL, Oracle, Sybase-System 11, Sybase SQL Anywhere, Sybase Adaptive Server, Sybase Ultralite).
- Internet Application using Java, XML, DHTML, Active X, ASP.
- Developing applications for hand – held devices like Palm.
- Lotus Notes & Group ware

Services offered

- Comprehensive consulting service
- Turnkey development projects
- Web/Internet development
- Mobile/Wireless connectivity
- Handheld device application development
- Customer Relationship Management
- ERP implementation and configuration
- Data Warehousing

2.1 Existing System

The rapid growth of Web services has led to a situation where companies and individuals rely more and more on material that is available over the Internet. An increasing number of people use web services both at work and at home.

Most of the web services rely upon the information transfer over the net, which is accessed from traditional databases. This involves the overheads of establishing connection and accessing complex database structures and converting to the format in which it can be displayed over the Internet.

When a user wants to transfer his database in one platform to another he has to write a conversion program. If he has to transfer database across multiple platforms he has to write a separate program for each platform.

The limitations of the above approaches include:

- Unnecessary delay in response time for accessing databases.
- Separate programs for transferring of database information into another platform.
- When the user needs database representation of the XML the conversion is done manually which is quite time consuming.

2.2 Proposed system

The objective of the project is to develop a tool, which extracts the XML database information and converts it into the corresponding relational database format. The output generated will be database tables populated with the data. The project aims at a tool, which automatically generates database after establishing the connection with the database.

This product can be used by organizations, which want to migrate their database to another platform. Online information systems, which provide information depending upon the request submitted, can use the output of this tool. Accessing the data in XML is far less processor intensive.

While accessing the information, which is stored in relational databases, much of the processor time is wasted in unwanted security checks and frequent accessing of complex database structures, which is not needed all the time. Efficient XML parsers are available in the market, which access the data at higher speeds from XML than the traditional database engines.

The user has to provide database connection parameters through the easy to use wizard and establish connection with the database. Then the user has to select the table names. This will lead to the creation of table structures and/or population of tables with data.

One more option of editing the datatypemapping XML file will enable the user to make the transformation to any database for which a JDBC driver is available without modifying the programs.

2.3 Requirements on new system

Requirement Specification is focused specifically on functioning of the new system. It allows the developer to understand the system, functions to be carried out, performance levels to be obtained and corresponding interfaces to be established. Some of the attributes of the requirement specification are unambiguous, complete, verifiable, consistent and modifiable.

Scope

This project is to develop a tool, which converts the data stored in the XML document to relational database. It has a wizard-based interface and it is an easy to use tool. Since it is developed in Java it can be run on any platform with Java Runtime Environment support.

Product perspective

The output of the product is the relational database tables populated with the data from the XML documents. Database data is different from a XML document, so it introduces several conversion issues. The product reads information from the XML documents and converts into corresponding relational database.

Product function

The product converts the XML documents into corresponding relational database. The user can choose from the tables available in the XML document or create and/or populate all the tables in the XML document.

User characteristics

There is a large potential set of users for the product. Any person who wishes to convert his XML documents to relational database can use this product.

The product can be used with Online Information Systems that submit information to the database over the Internet through XML documents.

A Database Administrator who wants to migrate his database contents to another geographical area can also use the product in an effective manner. The reason for this being XML can be easily transported over the Internet than the complex database structures.

A User who wishes to copy his database to another platform uses this as an intermediary tool for transfer of the database.

The product has a wizard-based interface through which the user can virtually click his way to the output database. He should have knowledge of XML and relational databases.

Functional requirements

The library provides a Java 2 classes which establishes connection with the database generates database and closes the connection. Other functions include editing the datatypemapping XML file, viewing of XML as text file and hierarchical tree.

Input

Input to the product include:

- Database driver to be used
- Database URL
- Username
- Password
- Selected list of tables to be created and/or populated
- Metadata XML file name
- Data XML file name
- Datatypemapping XML file name

Information processing required

1. Establish connection with the database.
2. Read the metadata XML and data XML files.
3. Validate the XML files against DTDs.
4. Map the metaXML to database table structures .
5. Datatype conversion, if needed using datatypemapping XML.
6. Map foreign key references.
7. Map integrity constraint rules.
8. Execute SQL create statement.
9. Map the dataXML to database data.
10. Execute SQL insert statement.

Design constraints

Standards compliance

Java 2 Standard
W3C XML 1.0 Specification
W3C DOM (Level 1) Specification

Hardware limitations

The proposed system should work on any hardware with lower or higher configurations, which support Java Runtime Environment.

Hardware interfaces and software interfaces with other systems

The product establishes connection with the database which the user wants to create and/or populate with XML data.

3.1 Hardware Configuration

Processor	-	Pentium III, 833 MHz
RAM	-	128 MB
Hard Disk	-	Seagate 20 GB
Monitor	-	Samsung 14" color
Keyboard	-	Logitech
Mouse	-	Logitech

3.2 Software configuration

Operating System	-	Windows 2000 Professional
Language	-	JAVA 2
XML DOM parser	-	JAXP 1.1
XML Editor	-	XML Spy 1.0
Others	-	Internet Explorer 5.5

4.1 Input Design

List of methods used

XMLtoDB() /*Constructor*/

Description: This method constructs a XMLtoDB object and establishes connection with the database.
Parameters: Driver name, URL, Username, password

readXML()

Description: This method reads a XML file.
Parameters: XML file name.

getTableNames()

Description: This method parses metadataXML or dataXML to fetch the table names in the input XML file.
Parameters: metadataXML or dataXML.
Return Values: Vector containing names of tables.

getValidDatatype()

Description: Parses the datatypemapping XML file and returns the corresponding datatype for the database to be generated.
Parameter: Datatype attribute in the metadata XML file.
Return value: Datatype for the database to be generated.

getValidRule()

Description: Parses the metadata XML file and returns the corresponding integrity constraint rule.

Parameter: Value of the ruletype attribute in the metadata XML file.

Return value: Valid integrity constraint rule.

generateCreateStmt()

Description: Constructs the complete Create statement by fetching information from the matadataXML.

Parameters: Table name.

Return value: string containing Create statement.

getConnectTime()

Description: Gets the connection time of the current connection for log information.

Return value: String containing date and connection time.

getDisConnectTime()

Description: Gets the disconnection time of the current connection for log information.

Return value: String containing date and disconnection time.

exeuteCreateStmt()

Description: Execute SQL Create statement.

Parameters: String containing Create statement.

Return value: Integer indicating successful or unsuccessful operation.

InsertIntoTable()

Description: Parses the data XML file to constuct all the insert statements for the input table name by fetching data from the dataXML and execution of the SQL insert statements.
Parameters: Table name.
Return value: Integer indicating Successful or unsuccessful operation.

closeDBConnection()

Description: Closes active database connection.

getAttribute()

Description: Method in interface org.w3c.dom.Element.Retrieves an attribute value by name.
Parameters: The name of the attribute to retrieve.
Return value: The attribute value as a string or the empty string if that attribute does not have a specified or default value.

getChildNodes()

Description: Method in interface org.w3c.dom.Node. Returns a NodeList containing the Child Nodes of this node.
Return value: NodeList containing the child nodes.

GetDocumentElement()

Description: Method in interface org.w3c.dom.Document. Fetches the root Element of this XML document.
Return value: Element containing the root.

GetElementsByTagName()

Description: Method in interface rg.w3c.dom.Element. Returns a NodeList containing all the descendants elements of this element. With the given tag name in the order in which they would encountered in the preorder traversal of the element tree.
Parameters: Tag name.
Return value: NodeList.

getFirstChild()

Description: Method in interface rg.w3c.dom.Node. Returns the first child of this node.
Return value: Node.

GetLength()

Description: Method in interfaceorg.w3c.NodeList. Returns the length of the NodeList.
Return value: Integer.

GetNodeName()

Description: Method in interface rg.w3c.dom.Node. Returns the tag name of this node.
Return value: String containing the tag name.

GetNodetype()

Description: Method in interface rg.w3c.dom.Node. Returns the type of the node.
Return value: Short integer indicating the type.

GetNodeValue()

Description: Method in interface rg.w3c.dom.Node. Returns the value of this node.

Return value: String containing the value of this node.

GetNextSibling()

Description: Method in interface rg.w3c.dom.Node. Returns the next sibling of this node.
Return value: Node.

GetPreviousSibling()

Description: Method in interface rg.w3c.dom.Node. Returns the previous sibling of this node.
Return value: Node.

getParentNode()

Description: Method in interface rg.w3c.dom.Node.Returns the parent node of this node.
Return value: Node.

hasChildNodes()

Description: Method in interface rg.w3c.dom.Node.Check whether if this node has children.
Return value: True if it has child nodes else false.

parse()

Description: Method in class javax.parsers.DocumentBuilder.Parse the content of the given file as an XML document and return a new DOM Document object.
Parameters: XML file.

Return value: Document object.

The content of the **metadata DTD** file is,

```
<!ENTITY%DATA_TYPE_CODES
"(BIGINT|BINARY|BINARY|BIT|BYTE|CHAR|COUNTER|CURRENCY|D
ATE|DATETIME|DECIMAL|DOUBLE|FLOAT|GUID|IMAGE|INT|INTEG
ER|LONG|LONGRAW|LONGBINARY|LONGCHAR|MONEY|NCHAR|NT
EXT|NUMBER|NVARCHAR|RAW|REAL|ROWID|SMALLDATETIME|S
MALLINT|SMALLMONEY|SQL_VARIANT|TEXT|TIMESTAMP|TINYIN
T|UNIQUEIDENTIFIER|VARBINARY|VARCHAR|VARCHAR2)">
<!ENTITY % YESNO_TYPE_CODES (YES|NO|UNKNOWN)">
<!ENTITY%RULES_TYPE_CODES
"(UPDATE_NO_ACTION|UPDATE_KEY_CASCADE|UPDATE_KEY_SE
T_NULL|UPDATE_KEY_SET_DEFAULT|DELETE_NO_ACTION|DELE
TE_KEY_CASCADE|DELETE_KEY_SET_NULL|DELETE_KEY_SET_D
EFAULT)">
<!ELEMENT DBMETA (Table*)>
<!ELEMENT Table (Column*)>
<!ATTLIST Table
        name CDATA #REQUIRED>
<!ELEMENT Column EMPTY>
<!ATTLIST Column
        name CDATA #REQUIRED
        type %DATA_TYPE_CODES; #REQUIRED
        size CDATA #REQUIRED
        scale CDATA #IMPLIED
        defvalue CDATA #IMPLIED
        isprimary %YESNO_TYPE_CODES; #IMPLIED
        isnotnull %YESNO_TYPE_CODES; #IMPLIED
        isfkey %YESNO_TYPE_CODES; #IMPLIED
```

```
ptablename CDATA #REQUIRED
pcolname CDATA #REQUIRED
updaterule %RULES_TYPE_CODES; #REQUIRED
deleterule %RULES_TYPE_CODES; #REQUIRED>
```

The general format of the **metadataXML** is,

```
<?xml version="1.0"?>
<!DOCTYPE DBMETA SYSTEM "md.dtd">
<DBMETA>
        <Table name="table name 1">
                <Column name="column 1 name"
                            type="column 1 type"
                            size="column 1 size">
                            .
                            .
                <Column name="column n name"
                            type="column n type"
                            size="column n size">
        </Table>
        .
        .
        <Table name="table name n">
                <!— column details -->
        </Table>
</DBMETA>
```

Design of dataXML

The content of **dataDTD** is,

```
<!ELEMENT DBDATA (Table*)>
<!ELEMENT Table (Row*)>
<!ATTLIST Table name CDATA #REQUIRED>
<!ELEMENT Row (Col*)>
<!ELEMENT Col (#PCDATA)>
```

The general format of the **dataXML** is,

```
<?xml version="1.0"?>
<!DOCTYPE DBMETA SYSTEM "data.dtd">
<DBDATA>
<Table name="table name 1">
        <Row 1>
                <Col 1> value in column 1   </Col 1>
                .
                .
                <Col n> value in column n   </Col n>
        </Row 1>
        .
        .
        <Row n>
                <!— column n details -->
        </Row n>
</Table>
<Table name="table name n">
        <!—table n details -->
</Table>
</DBDATA>
```

Design of datatypeMappingXML

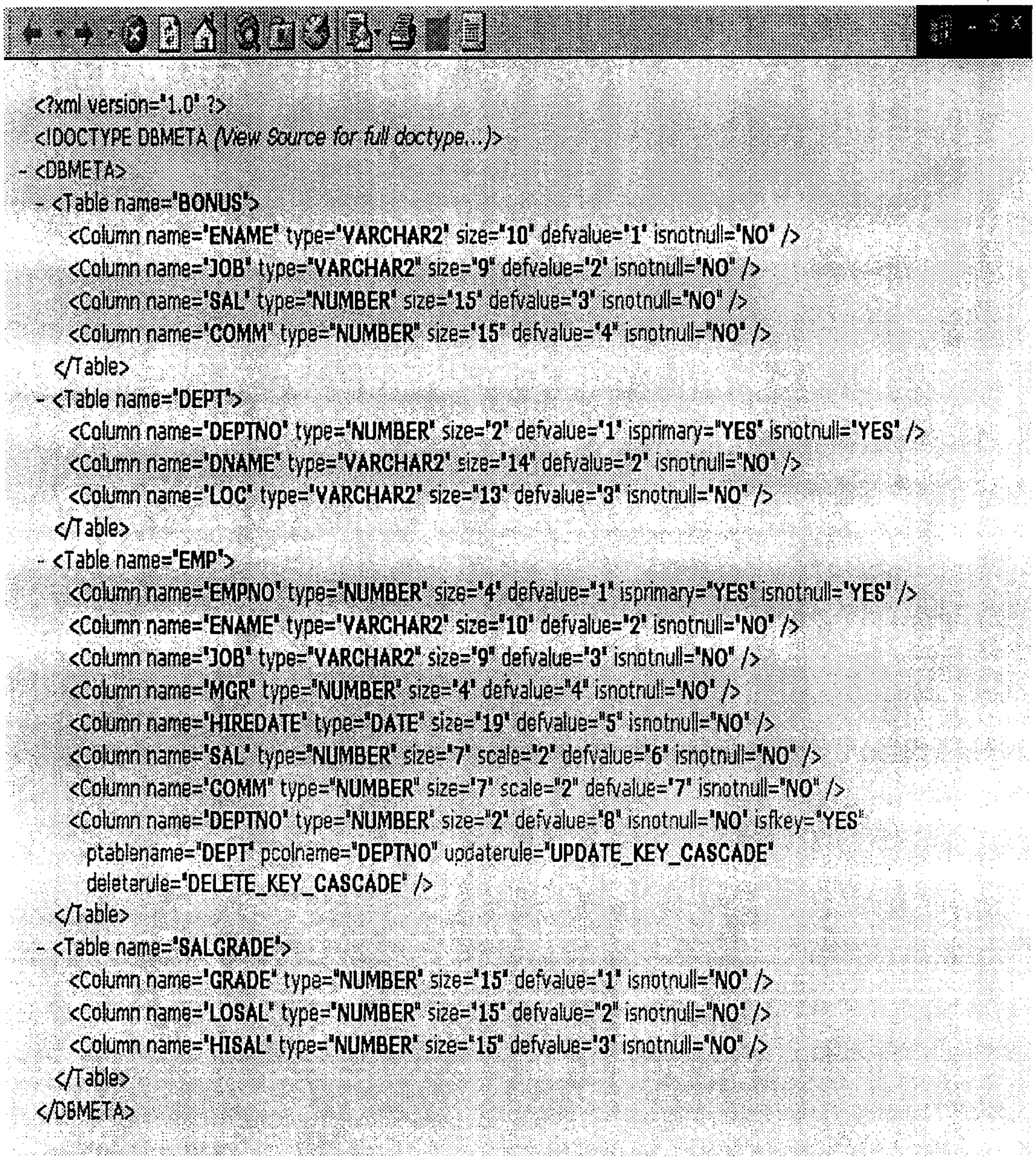
The general format of the **datatypemappingXML** is,

```
<?xml version="1.0"?>
<DatatypesMapping>
<SQLSERVER>
        <datatype1>validdatatype1</datatype1>
        <datatype2>validdatatype2</datatype2>
         .
         .
        <datatype n>validdatatype n</datatype n>
</SQLSERVER>
<ORACLE>
        <datatype1>validdatatype1</datatype1>
        <datatype2>validdatatype2</datatype2>
         .
         .
        <datatype n>validdatatype n</datatype n>
</ORACLE>
<ACCESS>
</ACCESS>
.
.
</DatatypesMapping>
```

4.2 Output design

The output of the product is the relational database table structures and populated database tables .The user selects the names of the input metadataXML, dataXML in the EzXMLSpy Wizard which takes user through a sequence of steps leading to the generation of selected database .The input XML files are valid and they conform to the metadataDTD and dataDTD files.This is shown below with the practical test samples:

Input Metadata XML File



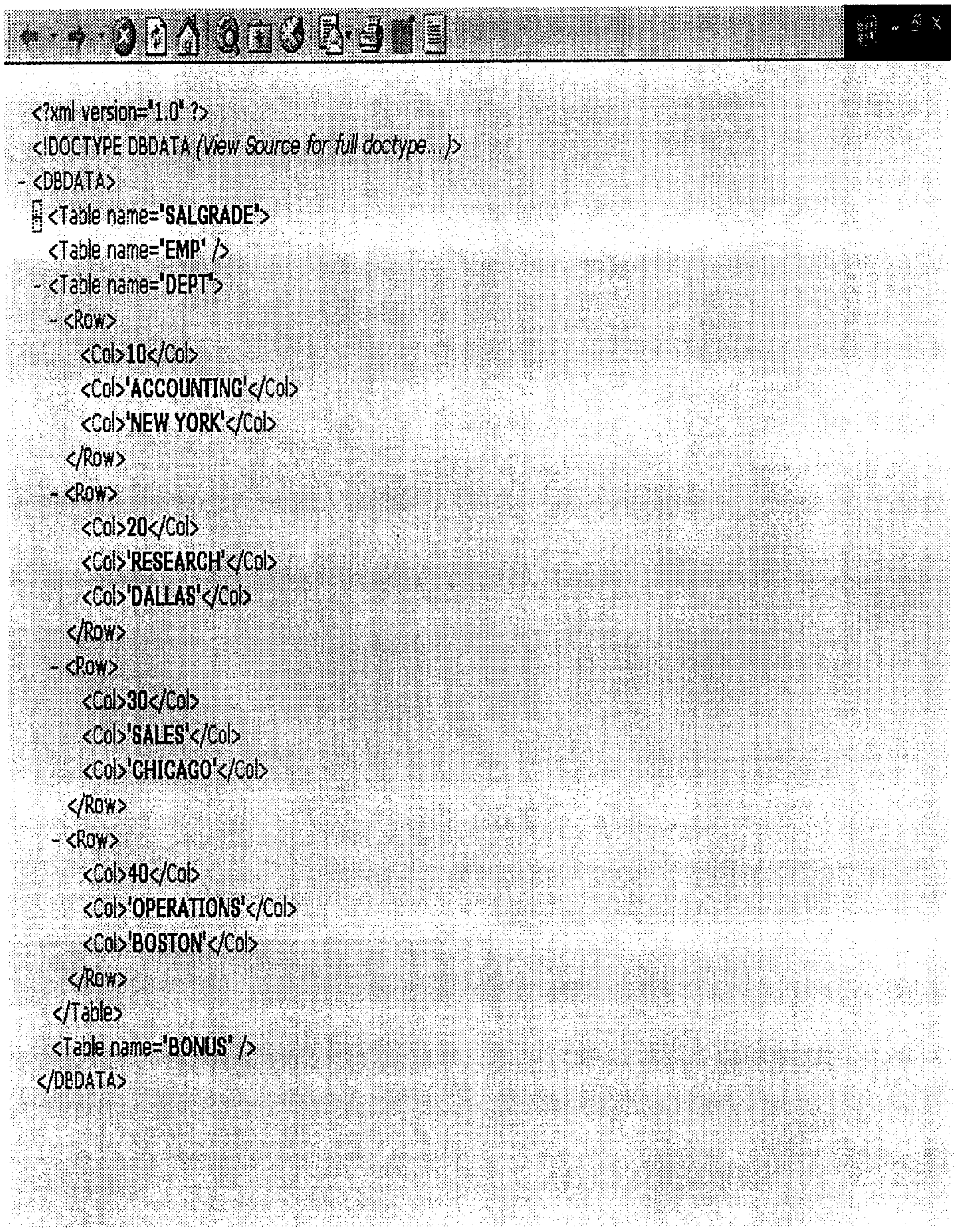
```
  <?xml version="1.0" ?>
  <!DOCTYPE DBMETA (View Source for full doctype...)>
- <DBMETA>
  - <Table name="BONUS">
      <Column name="ENAME" type="VARCHAR2" size="10" defvalue="1" isnotnull="NO" />
      <Column name="JOB" type="VARCHAR2" size="9" defvalue="2" isnotnull="NO" />
      <Column name="SAL" type="NUMBER" size="15" defvalue="3" isnotnull="NO" />
      <Column name="COMM" type="NUMBER" size="15" defvalue="4" isnotnull="NO" />
    </Table>
  - <Table name="DEPT">
      <Column name="DEPTNO" type="NUMBER" size="2" defvalue="1" isprimary="YES" isnotnull="YES" />
      <Column name="DNAME" type="VARCHAR2" size="14" defvalue="2" isnotnull="NO" />
      <Column name="LOC" type="VARCHAR2" size="13" defvalue="3" isnotnull="NO" />
    </Table>
  - <Table name="EMP">
      <Column name="EMPNO" type="NUMBER" size="4" defvalue="1" isprimary="YES" isnotnull="YES" />
      <Column name="ENAME" type="VARCHAR2" size="10" defvalue="2" isnotnull="NO" />
      <Column name="JOB" type="VARCHAR2" size="9" defvalue="3" isnotnull="NO" />
      <Column name="MGR" type="NUMBER" size="4" defvalue="4" isnotnull="NO" />
      <Column name="HIREDATE" type="DATE" size="19" defvalue="5" isnotnull="NO" />
      <Column name="SAL" type="NUMBER" size="7" scale="2" defvalue="6" isnotnull="NO" />
      <Column name="COMM" type="NUMBER" size="7" scale="2" defvalue="7" isnotnull="NO" />
      <Column name="DEPTNO" type="NUMBER" size="2" defvalue="8" isnotnull="NO" isfkey="YES"
        ptablename="DEPT" pcolname="DEPTNO" updaterule="UPDATE_KEY_CASCADE"
        deleterule="DELETE_KEY_CASCADE" />
    </Table>
  - <Table name="SALGRADE">
      <Column name="GRADE" type="NUMBER" size="15" defvalue="1" isnotnull="NO" />
      <Column name="LOSAL" type="NUMBER" size="15" defvalue="2" isnotnull="NO" />
      <Column name="HISAL" type="NUMBER" size="15" defvalue="3" isnotnull="NO" />
    </Table>
  </DBMETA>
```

Output Table Structures

```
SQL> describe EMP;
 Name                           Null?     Type
 ------------------------------ -------- --------------------------------
 EMPNO                          NOT NULL  NUMBER(4)
 ENAME                                    VARCHAR2(10)
 JOB                                      VARCHAR2(9)
 MGR                                      NUMBER(4)
 HIREDATE                                 DATE
 SAL                                      NUMBER (7,2)
 COMM                                     NUMBER (7,2)
 DEPTNO                                   NUMBER(2)

SQL> describe  SALGRADE;
 Name                           Null?     Type
 ------------------------------ -------- ----------------------------
 GRADE                                    NUMBER(15)
 LOSAL                                    NUMBER(15)
 HISAL                                    NUMBER(15)

SQL> describe BONUS;
 Name                           Null?     Type
 ------------------------------ -------------------------------- ----
 ENAME                                    VARCHAR2(10)
 JOB                                      VARCHAR2(9)
 SAL                                      NUMBER(15)
 COMM                                     NUMBER(15)
```

```
SQL> describe DEPT;
 Name                          Null?    Type
 ------------------------------ -------- ----------------------------------
 DEPTNO                                 NOT NULL NUMBER(2)
 DNAME                                  VARCHAR2(14)
 LOC                                    VARCHAR2(13)
```

Input Data XML File



```
  <?xml version="1.0" ?>
  <!DOCTYPE DBDATA (View Source for full doctype...)>
- <DBDATA>
  + <Table name='SALGRADE'>
    <Table name='EMP' />
  - <Table name='DEPT'>
    - <Row>
        <Col>10</Col>
        <Col>'ACCOUNTING'</Col>
        <Col>'NEW YORK'</Col>
      </Row>
    - <Row>
        <Col>20</Col>
        <Col>'RESEARCH'</Col>
        <Col>'DALLAS'</Col>
      </Row>
    - <Row>
        <Col>30</Col>
        <Col>'SALES'</Col>
        <Col>'CHICAGO'</Col>
      </Row>
    - <Row>
        <Col>40</Col>
        <Col>'OPERATIONS'</Col>
        <Col>'BOSTON'</Col>
      </Row>
    </Table>
    <Table name='BONUS' />
  </DBDATA>
```

Output Database tables Populated with Data

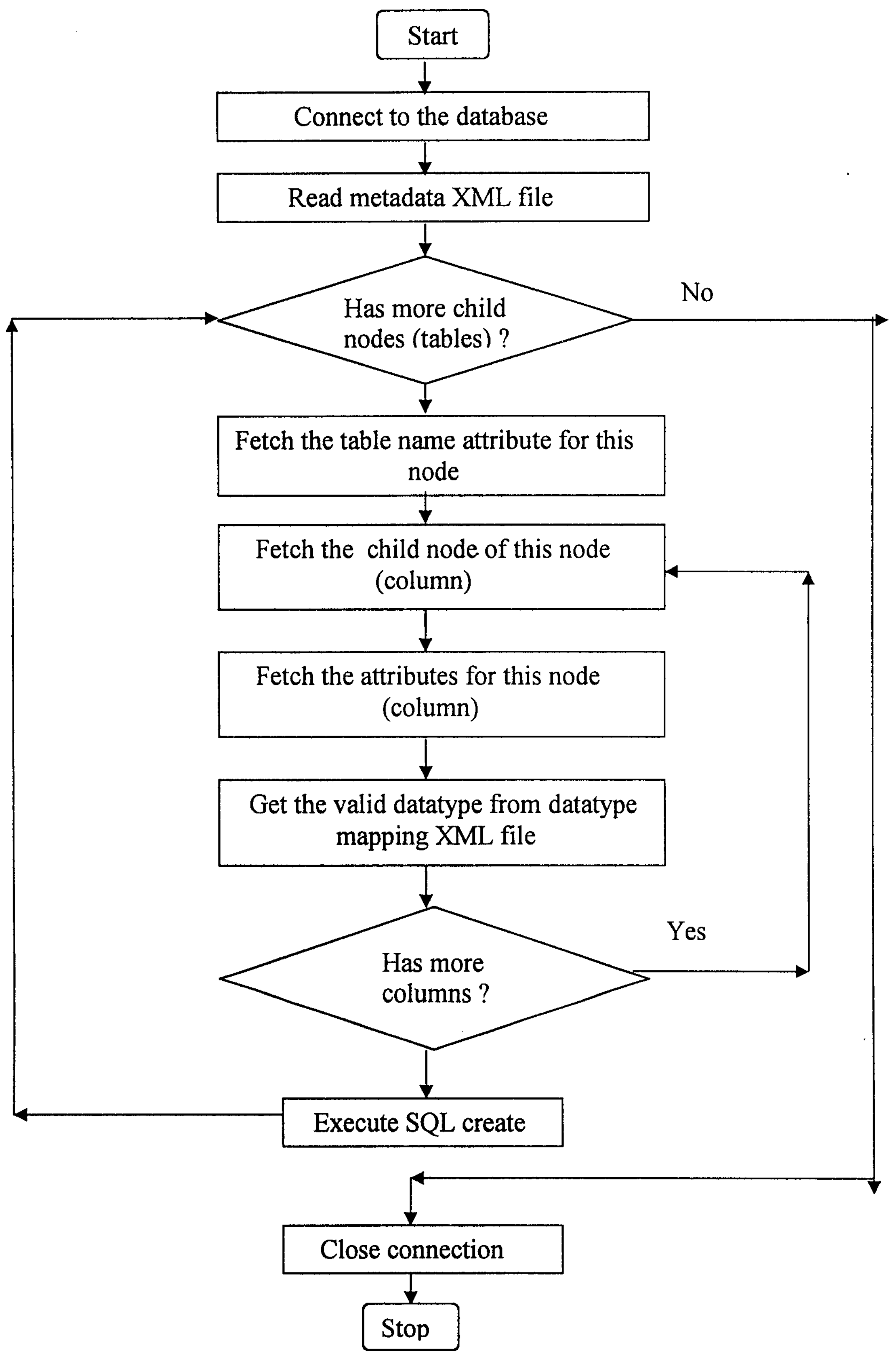
```
SQL> select * from BONUS;

no rows selected

SQL> select * from DEPT;

 DEPTNO          DNAME                     LOC
--------- -------------- ----------------------------------------------
   10            ACCOUNTING                NEW YORK
   20            RESEARCH                  DALLAS
   30            SALES                     CHICAGO
   40            OPERATIONS                BOSTON

SQL> select * from SALGRADE;

  GRADE    LOSAL        HISAL
--------- --------------------- ---------
    1        700          1200
    2        1201         1400
    3        1401         2000
    4        2001         3000
    5        3001         9999

SQL> select * from EMP;

no rows selected
```

4.3 Database design

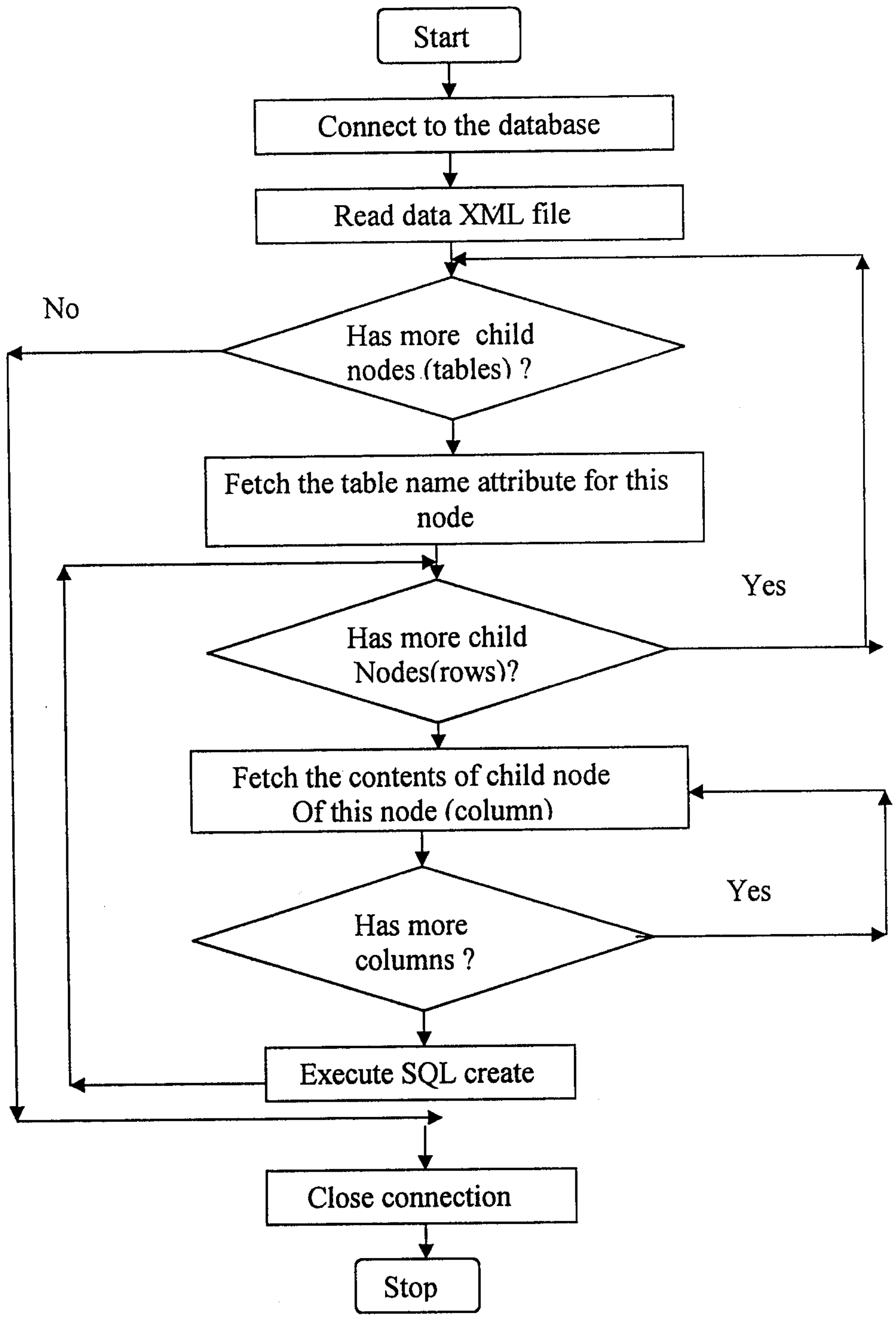
Since the product is cross database software there is no specific database design.

4.4 Process Design

Metadata XML


Start
Connect to the database
Read metadata XML file
Has more child nodes (tables) ?
No
Fetch the table name attribute for this node
Fetch the child node of this node (column)
Fetch the attributes for this node (column)
Get the valid datatype from datatype mapping XML file
Has more columns ?
Yes
Execute SQL create
Close connection
Stop

Data XML


Start
Connect to the database
Read data XML file
No
Has more child nodes (tables) ?
Fetch the table name attribute for this node
Yes
Has more child Nodes(rows)?
Fetch the contents of child node Of this node (column)
Yes
Has more columns ?
Execute SQL create
Close connection
Stop

5.1 System Testing

Software testing is a critical element of software quality assurance and follows the ultimate review of specification, design and coding. In addition, data collected during testing is used to provide a good indication of software quality as a whole. Some severe errors that require design modification are encountered with regularity, software functions appear to be working properly and the errors encountered are easily corrected.

The objectives of testing are as follows:

- Testing is the process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding as yet undiscovered error.

The testing phase involves the testing of developed system using various kinds of data. The following tests were conducted and desirable results were found.

Black Box Testing

In Black box testing the incorrect or missing functions, interface errors, performance errors, initialization and termination errors are tested by giving proper inputs and the errors which are found, are corrected.

White Box Testing

In white box testing, the following were tested successfully for the product. Checking whether all independent paths within each module have been exercised at least once.

- Exercise all logical decisions on their true and false sides.

- Execute all loops at their boundaries and within their bounds.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design. Important control paths were tested to uncover errors within the boundary of the module. The relative complexity of tests and errors detected as a result is limited by the constraints.

This testing was carried out during programming stage itself and found to be working satisfactorily as regard to the expected output from the module.

Integration Testing

Data can be lost across an interface. One module can have an adverse effect on another. Sub functions, when combined, may not produce the desired output for the major functions. Integration testing is a systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the structure. All the modules are combined and tested as a whole.

This system has been tested by using integration testing and found to be working satisfactorily.

Validation Testing

Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably expected.

The proposed system gets the input XML files and generates the output database. The outputs are tested for validity and found to be satisfactory.

5.2 System Implementation

The implementation phase of the software development is concerned with translating design specification to source code. The primary goal of implementation, testing and modification enables us to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and hence debugging is easily done. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are indications of adequate design and misdirected thinking.

- All the code modules that are developed for this system is single entry and single exit constructs.
- Static variables and methods are used whenever necessary for efficient execution of code.
- No goto statement is used during the development.
- The comments are added at all the parts of the source code to make the code easily understandable to the other programmers.

6 Conclusion

The project has aimed to satisfy the requirements of the system to the maximum extent. All the required changes during the implementation are being resolved to the best of abilities.

Maximum care and concentration has been focused to troubleshoot this project and provide an efficient system.

Invaluable experience has been gained in the areas of system design, new technologies like XML, DOM, system testing and implementation.

7 Scope for future development

Further enhancement of the product can be done in the following areas:

- Storing multimedia data.
- Storing vector graphics.
- Storage of WML documents.
- Storing RNA, DNA and protein sequence information.
- Storage of voice scripts for delivery over phones.

References

Books

Step by step XML

Author: Michael J. Young

Publication: Prentice-Hall of India Private Ltd.

XML in Record Time

Author: Natanya Pitts

Publication: BPB publications

SAMS Teach Yourself XML in 21 days

Authors: Simon North and Paul Hermans

Publication: Techmedia

Professional XML databases

Authors: Group of authors

Publication: Wrox Press

Web references

http://www.w3.org/XML/

http://www.alphaworks.ibm.com/

Java Overview

The application level safety features of Java make it possible to develop new kinds of applications not necessarily feasible before. A web browser that implements the Java run-time system can incorporate Java applets as executable content inside the documents. This means that web pages can contain not only hypertext information but also full-fledged interactive applications. The added potential for use of WWW is enormous.

A user can retrieve and use software simply by navigating with a web browser. Formerly static information can be paired with portable software for interpreting and using the information. Instead of providing some data for a spreadsheet, for example, a web document might contain a fully functional spreadsheet application embedded within it that allows users to view and manipulate the information.

Other main features of Java:

- Compiled and interpreted
- Complete and portable
- Object oriented
- Safety of design
- Error handling
- Multithreading

XML Overview

XML is subset of the Standard Generalized Markup Language (SGML) defined in ISO standard 8879:1986 that is designed to make it easy to interchange structured documents over the Internet. XML files always clearly mark where the start and end of each of the logical parts (called *elements*) of an interchanged document occurs. XML restricts the use of SGML constructs to ensure that fallback options are available when access to certain components of the document is not currently possible over the Internet. It also defines how Internet Uniform Resource Locators can be used to identify component parts of XML data streams.

By defining the role of each element of text in a formal model, known as a *Document Type Definition* (DTD), users of XML can check that each component of document occurs in a valid place within the interchanged data stream.

XML DTD allows computers to check, for example, that users do not accidentally enter a third-level heading without first having entered a second-level heading, something that cannot be checked using the Hyper Text Markup Language (HTML) previously used to code documents that form part of the World Wide Web (WWW) of documents accessible through the Internet.

However, unlike SGML, XML does not require the presence of a DTD. If no DTD is available, either because all or part of it is not accessible over the Internet or because the user failed to create it, an XML system can assign a default definition for undeclared components of the markup.

XML allows users to:

- Bring multiple files together to form compound documents.
- Identify where illustrations are to be incorporated into text files, and the format used to encode each illustration.
- Provide processing control information to supporting programs, such as document validators and browsers.
- Add editorial comments to a file.

It is important to note, however, that XML is not:

- A predefined set of tags, of the type defined for HTML, that can be used to markup documents.
- A standardized template for producing particular types of documents.

XML was not designed to be a standardized way of coding text: in fact it is impossible to devise a single coding scheme that would be suit all languages and all applications. Instead XML is formal language that can be used to pass information about the component parts of a document to another computer system. XML is flexible enough to be able to describe any logical text structure, whether it be a form, memo, letter, report, book, encyclopedia, dictionary or database.

The Components of XML

XML is based on the concept of *documents* composed of a series of *entities*. (`Entity' is the English spelling of the French word `entité', the Teutonic equivalent of which is `thing'. Those familiar with modern programming techniques will be probably be more comfortable using the word `object'. All these terms are synonymous.) Each entity can contain one or more logical *elements*. Each of these elements can have certain *attributes* (properties) that describe the way in which it is to be processed. XML provides a formal syntax for describing the relationships between the entities, elements and attributes that make up an XML document, which can be used to tell the computer how it can recognize the component parts of each document.

XML differs from other markup languages in that it does not simply indicate where a change of appearance occurs, or where a new element starts. XML sets out to clearly identify the boundaries of every part of a document, whether it can be a new chapter, a piece of boilerplate text, or a reference to another publication.

To allow the computer to check the structure of a document users must provide it with a document type definition that declares each of the permitted entities, elements and attributes, and the relationships between them.

How is XML Used?

To use a set of *markup tags* that has been defined by a trade association or similar body, users need to know how the markup tags are delimited from normal text and in which order the various elements should be used in. Systems that understand XML can provide users with lists of the elements that are valid at each point in the document, and will automatically add the required delimiters to the name to produce a markup tag. Where the data capture system does not understand XML, users can enter the XML tags manually for later validation. Elements and their attributes are entered between matched pairs of angle brackets (<. . . >) while entity references start with an ampersand and end with a semicolon (`&...;`).

Because XML tag sets are based on the logical structure of the document they are somewhat easier to understand, and remember, than physically based markup schemes of the type typically provided by word processors.

Well-Formed and Valid XML Documents

An XML document that conforms to the structural and notational rules of XML is considered well-formed. A well-formed XML document does not have to contain or reference a DTD, but rather can implicitly define its data elements and their relationships.

Well-formed XML documents must follow these rules:

- The document must start with the XML declaration`<?xml version="1.0">`.
- All elements must be contained within one root element.
- Elements must be nested in a tree structure without overlapping.
- All non-empty elements must have start and end tags.

Well-formed XML documents that also conform to a DTD are considered valid. When an XML document containing or referencing a DTD is parsed, the parsing application verifies that the XML conforms to the DTD and is therefore valid, which allows the parsing application to process it with the assurance that all of the data follows the rules defined in the DTD. When storing data from an XML document in a database, the DTD can be used to validate its structure ensuring its data elements will map to the corresponding columns in the database table. If an XML document is generated by reading data from the database and constructing the XML based upon its schema, the resulting XML document will be implicitly valid. By design, it will conform to the underlying table structure that generated it.

XML in Internet Applications

There are many potential uses of XML in Internet applications. Two of the most compelling uses that involve database applications are customizing the presentation of data and exchanging business data among applications.

XML enables customized presentation of data for different browsers, devices, and users. By using XML documents along with XSL stylesheets on either the client, middle-tier, or server, you can transform, organize, and present XML data tailored to individual users for a variety of client devices, including graphical and non-graphical Web browsers, personal digital assistants (PDAs) like the Palm Pilot, digital cell phones, and pagers, among others. In doing so, you can focus your business applications on business operations, without concern for the kind of output devices that will present the data, now or in the future. Using XML and XSL also makes it easier to create and manage dynamic Web sites. You can change the look and feel simply by changing the XSL stylesheet, without having to modify the underlying business logic or database code. As you target new users and devices, you can simply design new XSL stylesheets as needed.

XML and Databases Overview

There are many reasons why we might wish to expose our database content as XML, or to store our XML documents in the database. One obvious advantage to XML is that it provides a way to represent structured data without any additional information. Because this structure is inherent in the XML document rather than needing to be driven by an additional document that describes how the structure appears as wee do with, say, a flat file, it becomes very easy to send structured information between systems. Since XML documents are simply text files, they may also be produced and consumed by legacy systems allowing these systems to expose their legacy data in a way that can easily be accessed by different consumers.

Another advantage to the use of XML is the ability to leverage tools, either available, or starting to appear, that use XML to drive more sophisticated behavior. For example XSLT may be used to style XML documents, producing HTML documents, WML decks, or any other type of text document. XML servers such as BizTalk allow XML to be encapsulated in routing information, which then may be used to drive documents to their appropriate consumers in our workflow.

Data serialized in an XML format provides flexibility with regard to transmission and presentation. With the recent boom in wireless computing, one challenge that many developers are facing is how to easily reuse their data to drive both traditional presentation layers (such as HTML) and new technologies such as (such as WML-aware cell phones). XML provides a great way to decouple the structure of data from exact syntactical presentation of that data. Additionally, since XML contains both data and structure, it avoids some of the typical data transmission issues that arise when sending normalized data from one system to another.

One caveat to remember is that, at least at this time, relational databases will perform better than XML documents. This means that for many uses, if there are no network or usage barriers, relational databases will be a better home for our data than XML. This is especially important if we intend to perform queries across our data – in this case a relational database is much better suited to the task than XML documents should be.

If we imagine that we are running an e-commerce system and that we take orders as XML, perhaps some of information needs to be sent to the some internal source (such as customer service department) as well as to some external partner (an external service department). In this case we might want to store past customer order details in a relational database but make it available to both parties, and XML would be the ideal format for exposing the data. It could be read no matter what language the application was written or what platform it is running on. It makes the system more loosely coupled and does not requires us to write code that ties us to either part of the application.

Clearly, in the case where numerous users (especially B2B and B2C) need different views of same data, the XML can provide a huge advantage.

XML Parsers

The XML recommendation assumes that there is a separate software module, called the processor or the parser, that converts the physical content of the document into a data structure or a sequence of events and callbacks. The output of the processor is made available to a larger application.

The purpose of parsing an XML document is to make some interfaces available to an application that needs to make use of the document; using those interfaces, the application can inspect, retrieve, and modify the document's contents. The XML parser thus sits between an XML document and an application that uses it.

The interactions between the processor and the application are codified in two specifications: Document Object Model (DOM) and Simple API for XML (SAX).

To process an existing document, you need an "input source" that delivers the document's contents. Once an input source is in place, the lexical analyzer can convert its sequence of characters into a sequence of tokens, and the parser can get to work. The application that uses that uses the parser wants to access various components of the document for the purpose of displaying or modifying or rearranging them.

Simple API for XML (SAX)

Here, we usually don't know or care how the parsing process unfolds and in what order lexical analysis or production rules are applied. However, we can visualize that process as a steady progression through the text that sends notifications of certain important events: the document has stated, an element has started, an element has ended, a character sequence between two elements has been found, and so on. SAX provides standard names for callback functions that are triggered by these events. Writing a SAX application mostly consists of implementing these callbacks.

When you use SAX, you have to think in terms of an unfolding process, a sweep through the text to be parsed. If you come across an element or an attribute list that you want to use later, you have to save a persistent reference to it, because it is not yet part of any data structure.

Document Object Model (DOM)

By contrast, DOM is all about a data structure: the result of parsing an XML file is not the emission of events (a la SAX), but a generic object tree.

COMPARISION

Because the parser reports events as it visits different parts of the document, it does not have to build any internal structure. That reduces the strain on system resources, which makes the parser attractive for large documents. For XML documents received as continuous streams, an event-based API is the only choice.

The DOM API, on the other hand, follows a treelike construct. Elements have parent-child relations with other elements. With this API, the parser builds an internal structure such that an application can navigate it in a treelike fashion. DOM allows an application to have random access to the tree-structured document at the cost of increased memory usage.

XML DOM Parser

The DOM recommendation defines an API (Application Programmer Interface) for the description and manipulation of HTML/XML documents. It is said to be an "Object Model" because it describes an object view of the components of the document.

The DOM specification itself only defines the interfaces that will be used to represent an XML document, by presenting one abstraction for each of the components of a document.

DOM History

Back in the early days of Web, people started to realize the need to manipulate the content and presentation of HTML pages on the client side. This involved the creation of scripting languages that could manipulate the document, and of course, some program-accessible representation of the document itself. As is usual in polarized markets with time constraints, the first approach to the problem was the creation of proprietary, browser-specific solutions. Both Netscape and Microsoft came up with their own "Object Document Model".

The DOM is the result of standardization of a common document view of HTML/XML documents. The uses of DOM are not restricted to Web browsers and the current recommendation (Level 2) includes interfaces not only for HTML but also for CSS, stylesheets, user inteface events, and XML.

Features of DOM

- The result of parsing an XML document is an object tree.
- Before you are able to see the content of even the first element, the whole document is loaded into the memory.
- The object structure reflects the view of your XML file as a document object, not as a domain-specific object.

- You can easily go back and forth in the tree, visiting elements and attributes more than once.
- The process of manipulating your XML is now based on the traversing and manipulation of objects.
- Whether or not this model of-processing is better depends on the problem at hand.
- Using the DOM classes, one can represent almost any XML data as a DOM tree.

DOM Structure

DOM interfaces are grouped in eight categories:

Core interfaces.
HTML.
Views.
Stylesheets.
CSS.
Events.
Traversal.
Range.

Core DOM Interfaces

Node

It defines the set of methods necessary to navigate, inspect, and modify any node(e.g. getChildren, removeChild, getFirstChild).

Element

It contains methods to access and manipulate the name and attributes of the element, as well as the inherited methods of the Node interface.

Attr

It represents an attribute/value pair. Attr objects are not considered subnodes of the tree but rather mere attributes of the element in which they appear.

Document

It serves a dual purpose: first, it represents the whole XML document and is therefore the root of the DOM tree. Second it provides the factory methods used to create elements and attributes programmatically.

Definitions, Acronyms and Abbreviation

Document Object Model (DOM)

An in-memory tree-based object representation of an XML document that enables programmatic access to its elements and attributes. The DOM object and its interface is a W3C recommendation. It specifies the Document Object Model of an XML Document including the APIs for programmatic access. DOM views the parsed document as a tree of objects.

Document Type Definition (DTD)

A set of rules that define the allowable structure of an XML document. DTDs are text files that derive their format from SGML and can either be included in an XML document by using the DOCTYPE element or by using an external file through a DOCTYPE reference.

Element

The basic logical unit of an XML document that may serve as a container for other elements as children, data, attributes, and their values. Elements are identified by start-tags, <name> and end-tags</name> or in the case of empty elements, <name/>.

eXtensible Markup Language (XML)

An open standard for describing data developed by the W3C using a subset of the SGML syntax and designed for Internet use. Version 1.0 is the current standard, having been published as a W3C Recommendation in February 1998.

EzXMLSpy

The Software that is being developed.

Hypertext Markup Language (HTML)

The markup language used to create the files sent to Web browsers and that serves as the basis of the World Wide Web.

Java

A high-level programming language developed and maintained by Sun Microsystems where applications run in a virtual machine known as a JVM. The JVM is responsible for all interfaces to the operating system. This architecture permits developers to create Java applications and applets that can run on any operating system or platform that has a JVM.

Java Database Connectivity (JDBC)

The programming API that enables Java applications to access a database through the SQL. JDBC drivers are written in Java for platform independence but are specific to each database.

Java Developer's Kit (JDK)

The collection of Java classes, runtime, compiler, debugger and usually source code for a version of Java that makes up a Java development environment. JDKs are designated by versions, and Java 2 is used to designate versions from 1.2 onward.

Java virtual machine (JVM)

The Java interpreter that converts compiled Java byte code into the machine language of the platform and runs it.

Parser

In XML, a software program that accepts as input an XML document and determines whether it is well formed and optionally, valid.

Product

EzXMLSpy

Structured Query Language (SQL)

The standard language used to access and process data in a relational database.

Supplier

Organization for which the developer is developing the product.

User

The person who uses EzXMLSpy to generate valid XML documents from the database.

Valid

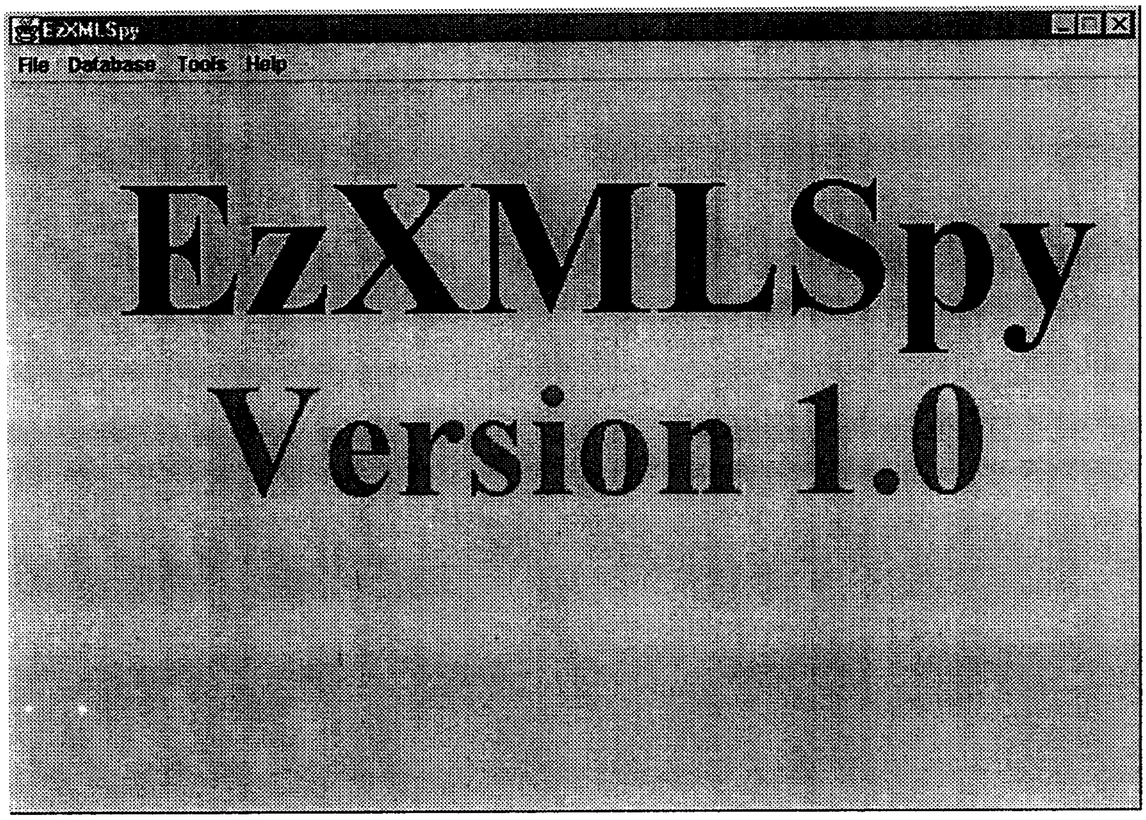
The term used to refer to an XML document when its structure and element content is - consistent with that declared in its referenced or included DTD.

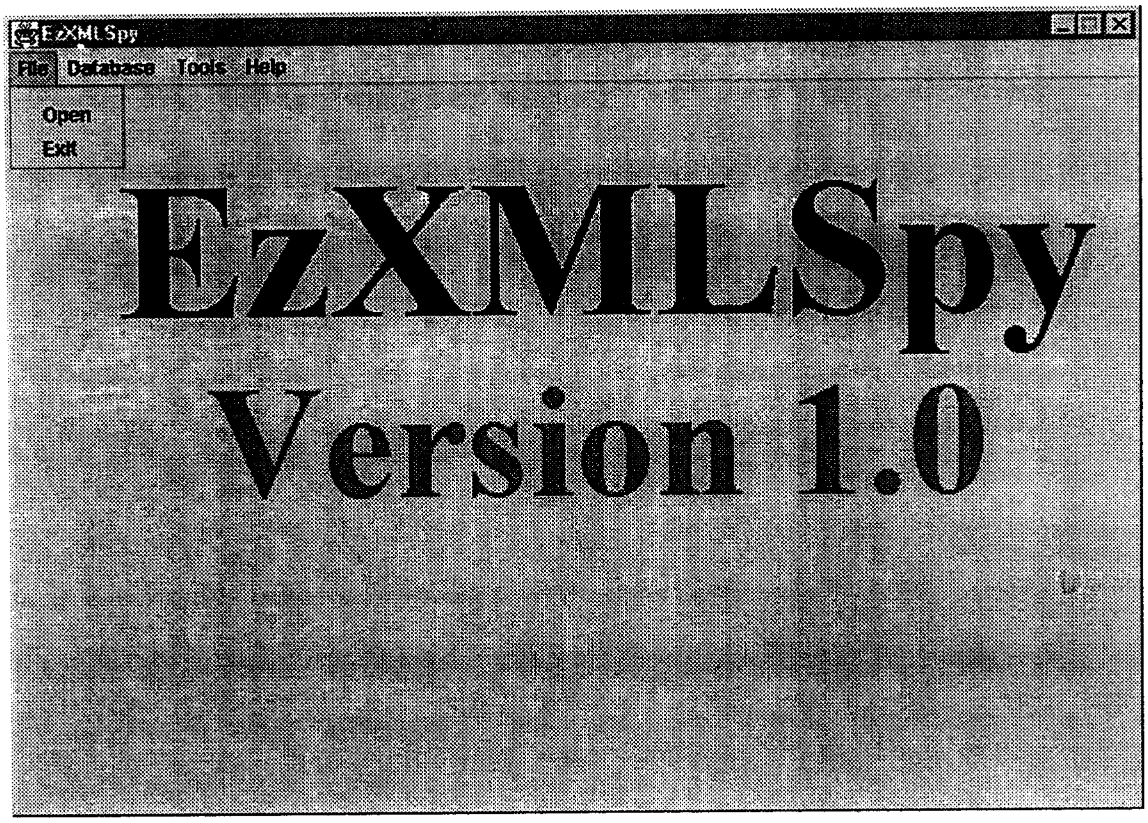
Well formed

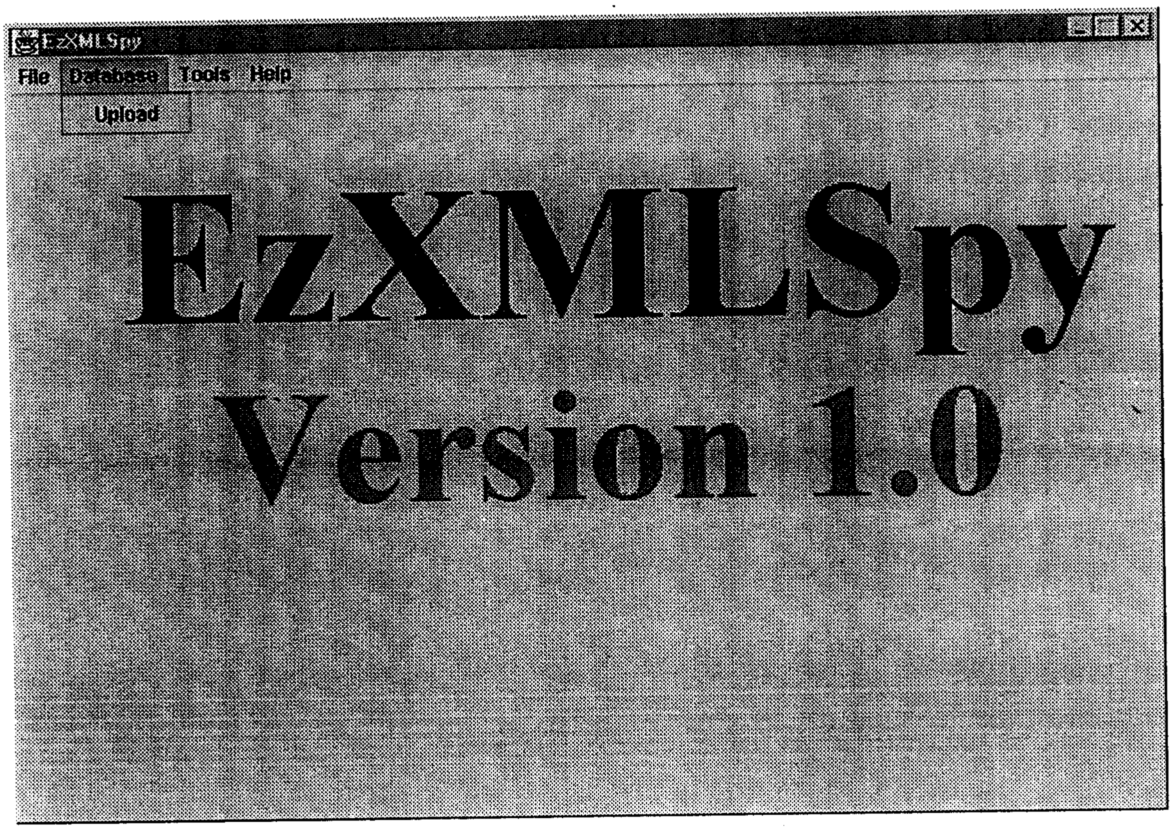
The term used to refer to an XML document that conforms to the syntax of the XML version declared in its XML declaration. This includes having a single root element, properly nested tags, and so forth.

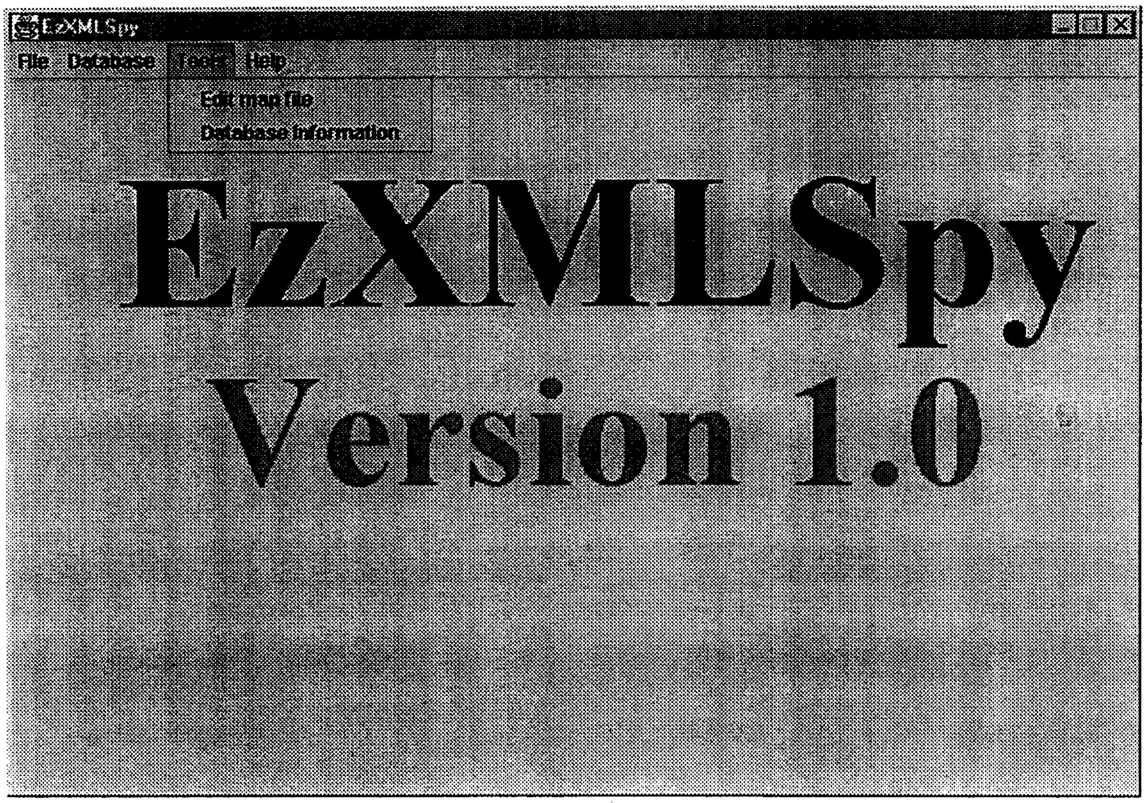
World Wide Web Consortium (W3C)

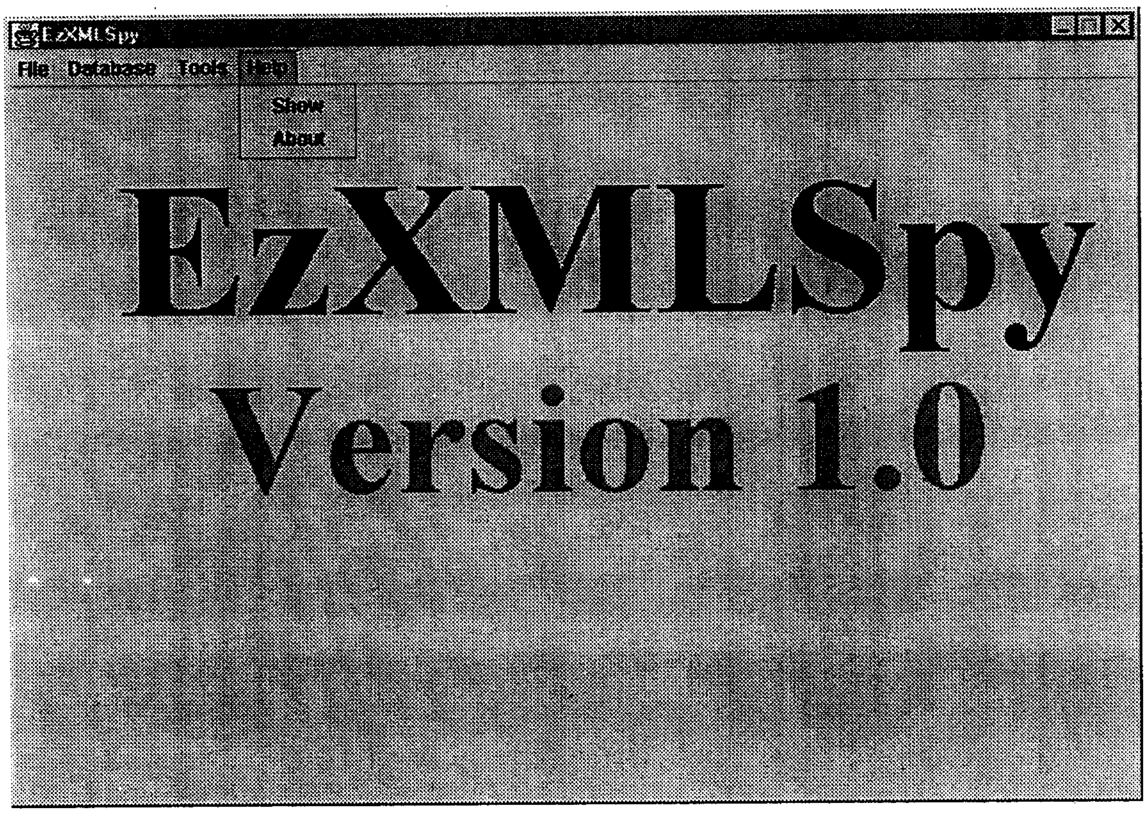
An international industry consortium started in 1994 to develop standards for the World Wide Web. It is located at http://www.w3c.org/.

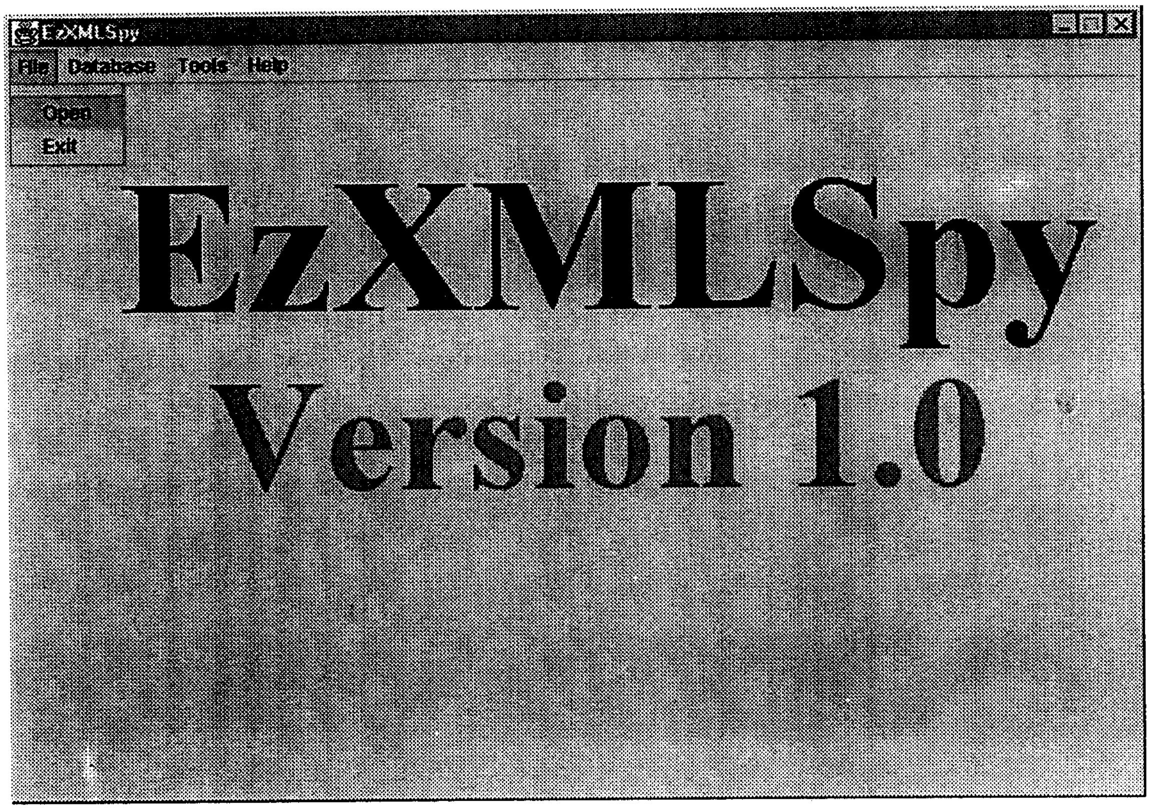

EzXMLSpy
File
Database
Tools
Help
EzXMLSpy
Version 1.0

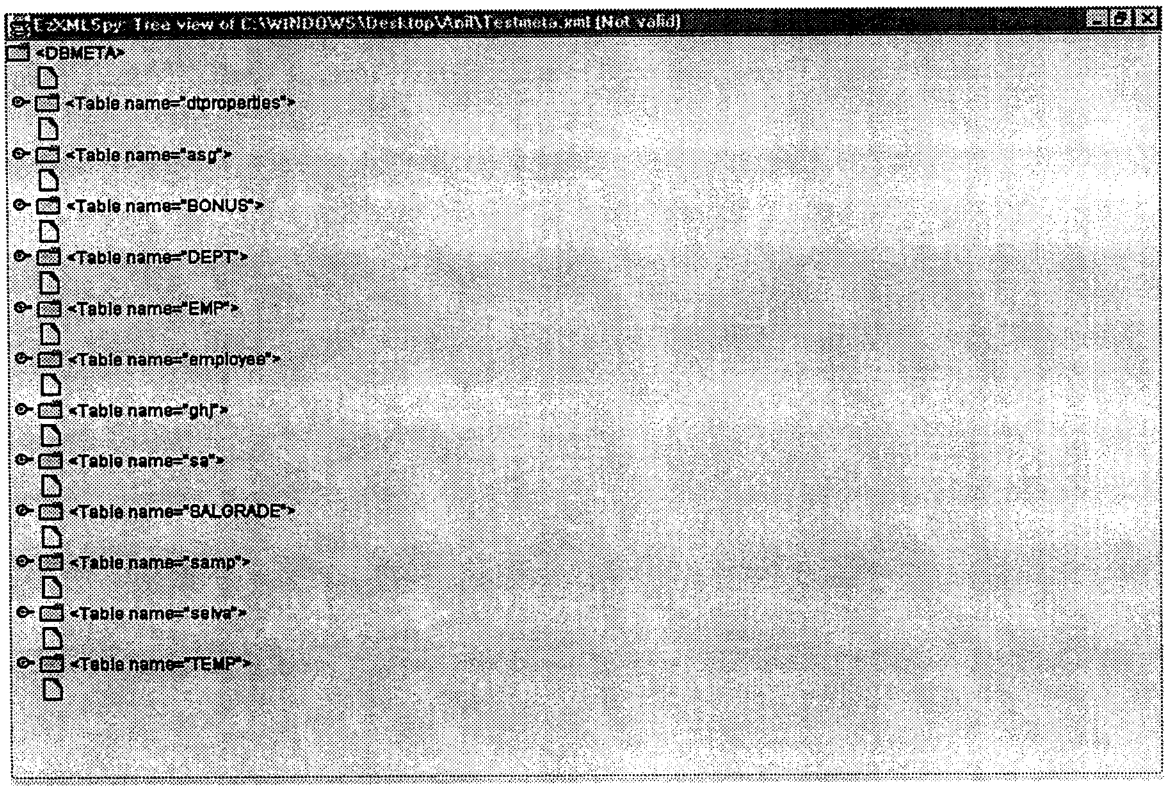

EzXMLSpy
File
Database
Tools
Help
Open
Exit
EzXMLSpy
Version 1.0


EzXMLSpy
File
Database
Tools
Help
Upload
EzXMLSpy
Version 1.0


EzXMLSpy
File
Database
Tools
Help
Edit map file
Database information
EzXMLSpy
Version 1.0

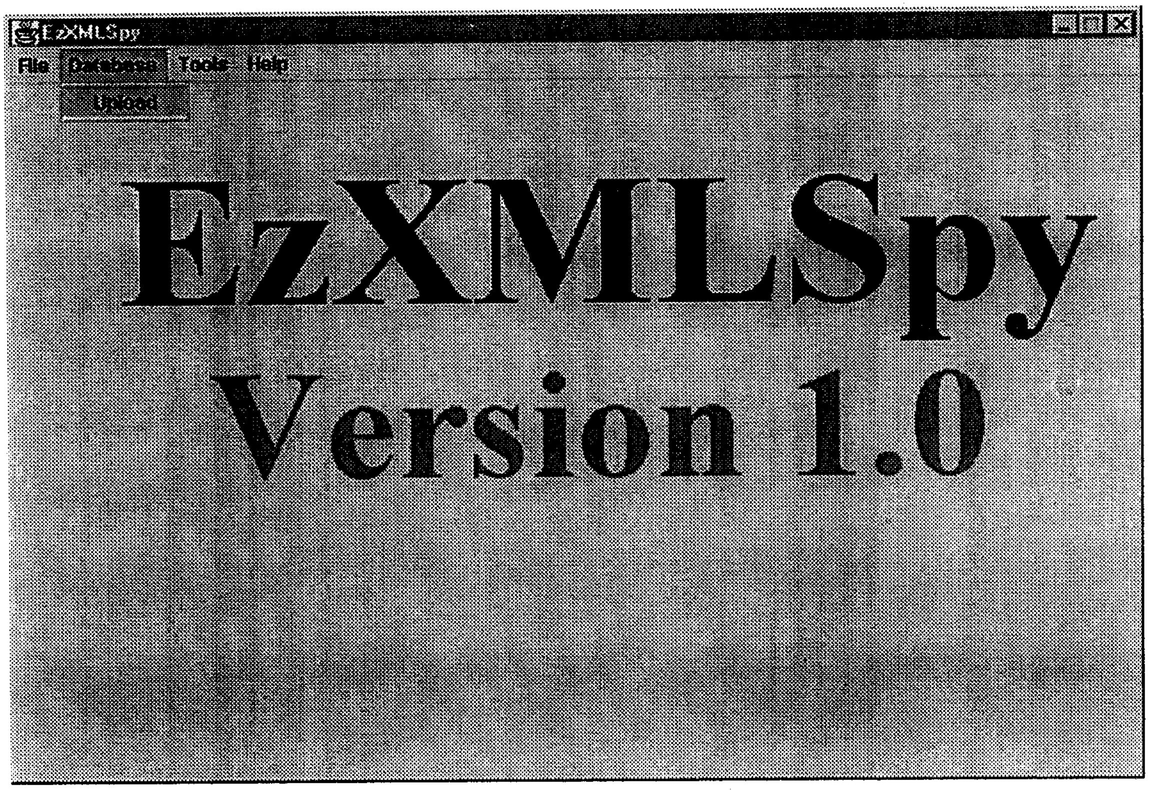

EzXMLSpy
File
Database
Tools
Help
Show
About
EzXMLSpy
Version 1.0

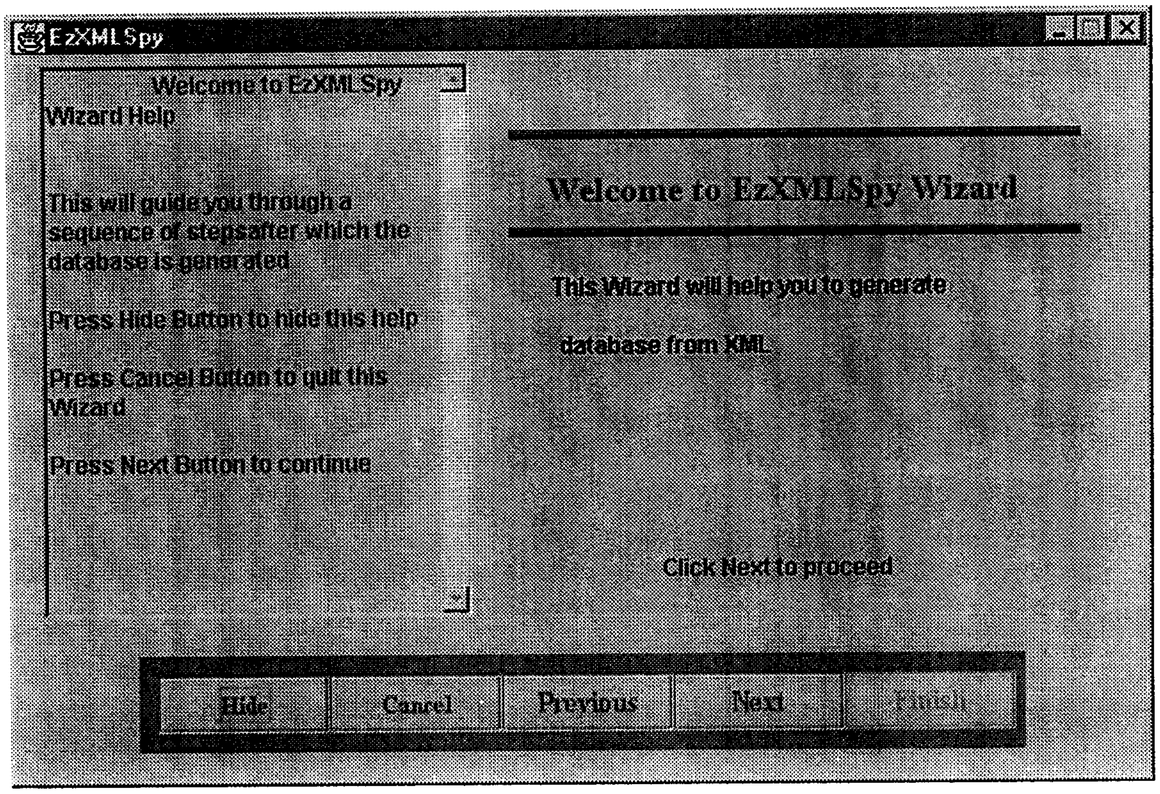

EzXMLSpy
File
Database
Tools
Help
Open
Exit
EzXMLSpy
Version 1.0



EzXMLSpy: Tree view of C:\WINDOWS\Desktop\Anil\Testmeta.xml (Not valid)

```
<DBMETA>
  <Table name="dtproperties">
  <Table name="asg">
  <Table name="BONUS">
  <Table name="DEPT">
  <Table name="EMP">
  <Table name="employee">
  <Table name="ghj">
  <Table name="sa">
  <Table name="SALGRADE">
  <Table name="samp">
  <Table name="selva">
  <Table name="TEMP">
```


EzXMLSpy
File
Database
Tools
Help
Upload
EzXMLSpy
Version 1.0



EzXMLSpy

Welcome to EzXMLSpy Wizard Help

This will guide you through a sequence of stepsafter which the database is generated

Press Hide Button to hide this help

Press Cancel Button to quit this Wizard

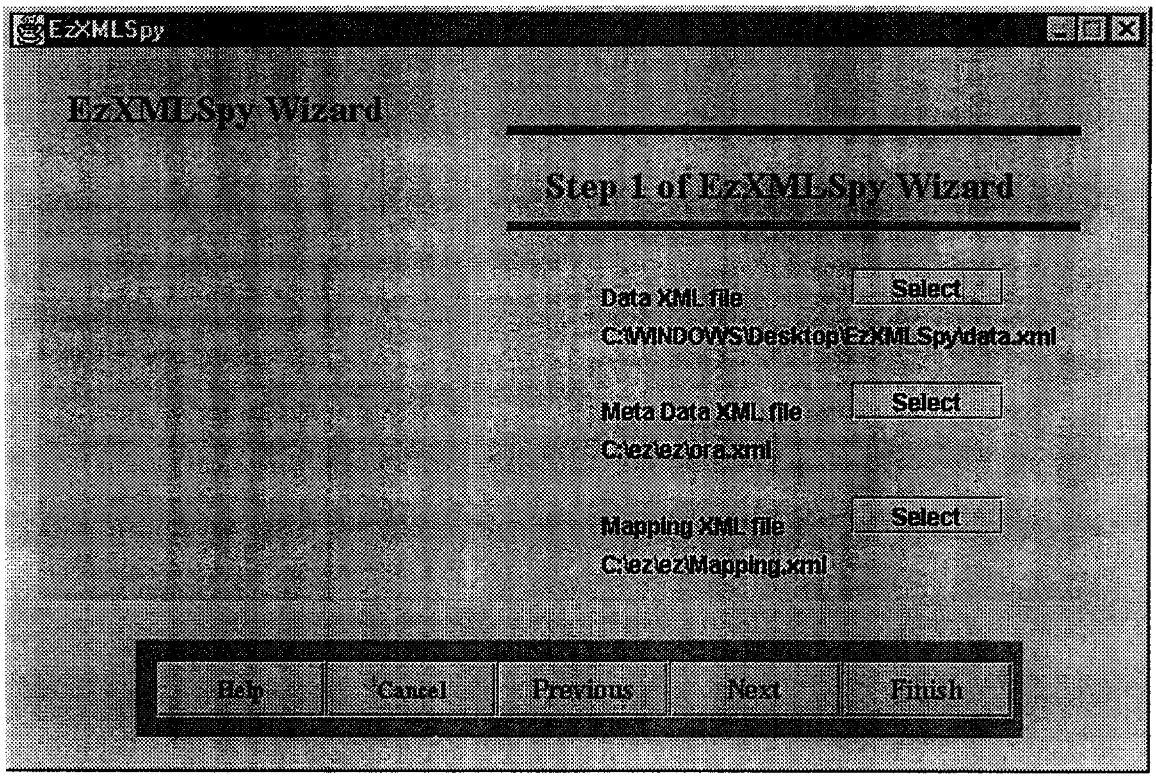
Press Next Button to continue

Welcome to EzXMLSpy Wizard

This Wizard will help you to generate

database from XML.

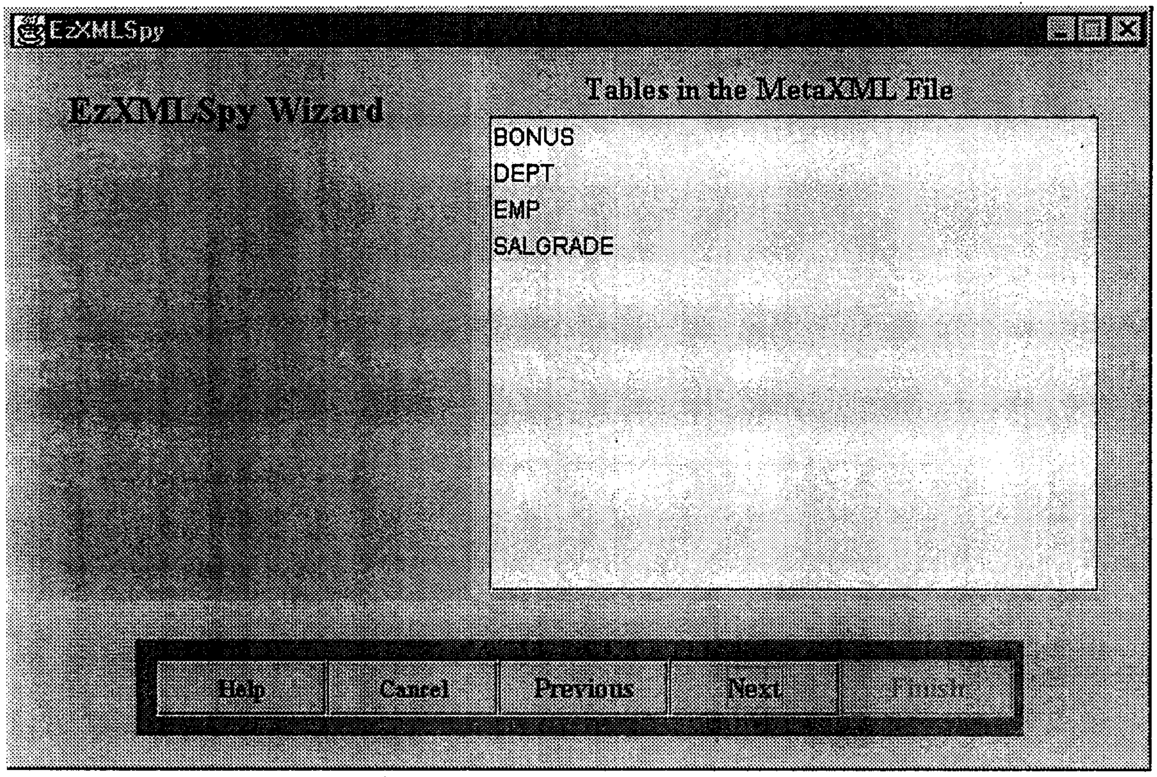
Click Next to proceed



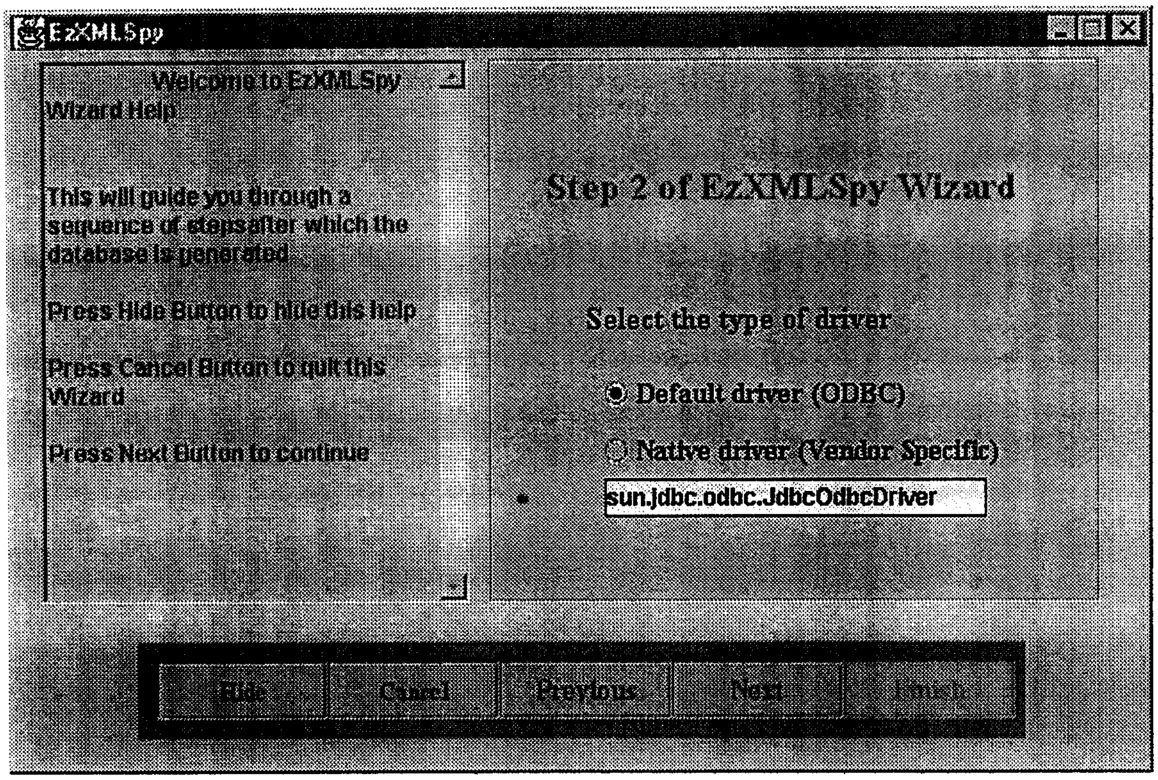
Data XML file
C:\WINDOWS\Desktop\EzXMLSpy\data.xml

Meta Data XML file
C:\ez\ez\ora.xml

Mapping XML file
C:\ez\ez\Mapping.xml



BONUS
DEPT
EMP
SALGRADE


EzXMLSpy

Welcome to EzXMLSpy Wizard Help

This will guide you through a sequence of stepsafter which the database is generated.

Press Hide Button to hide this help

Press Cancel Button to quit this Wizard

Press Next Button to continue

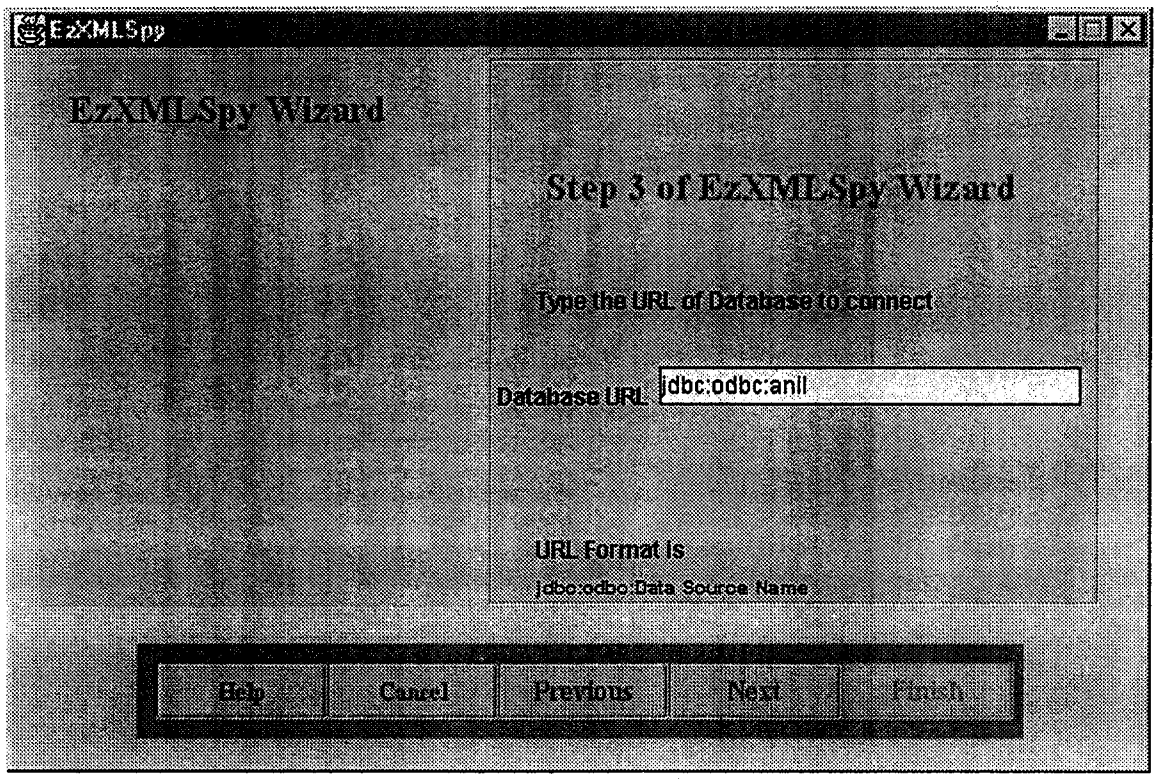
Step 2 of EzXMLSpy Wizard

Select the type of driver

[✓ Default driver (ODBC)]

Native driver (Vendor Specific)

sun.jdbc.odbc.JdbcOdbcDriver

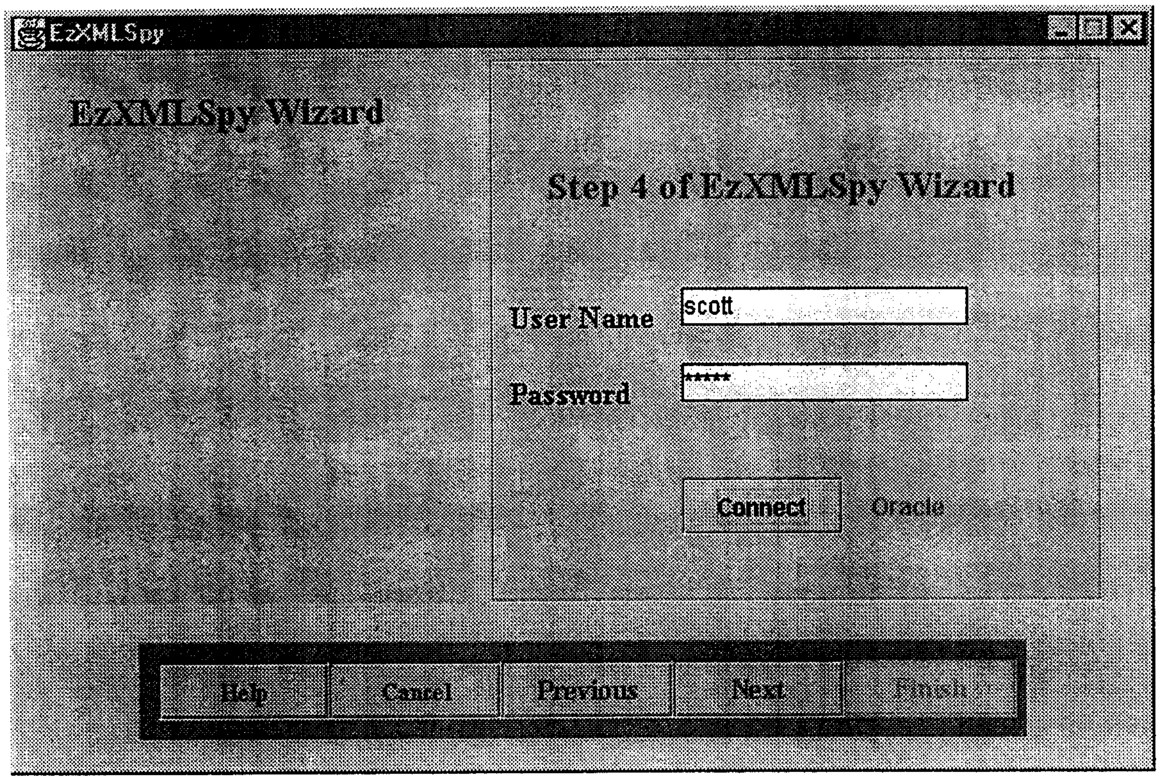


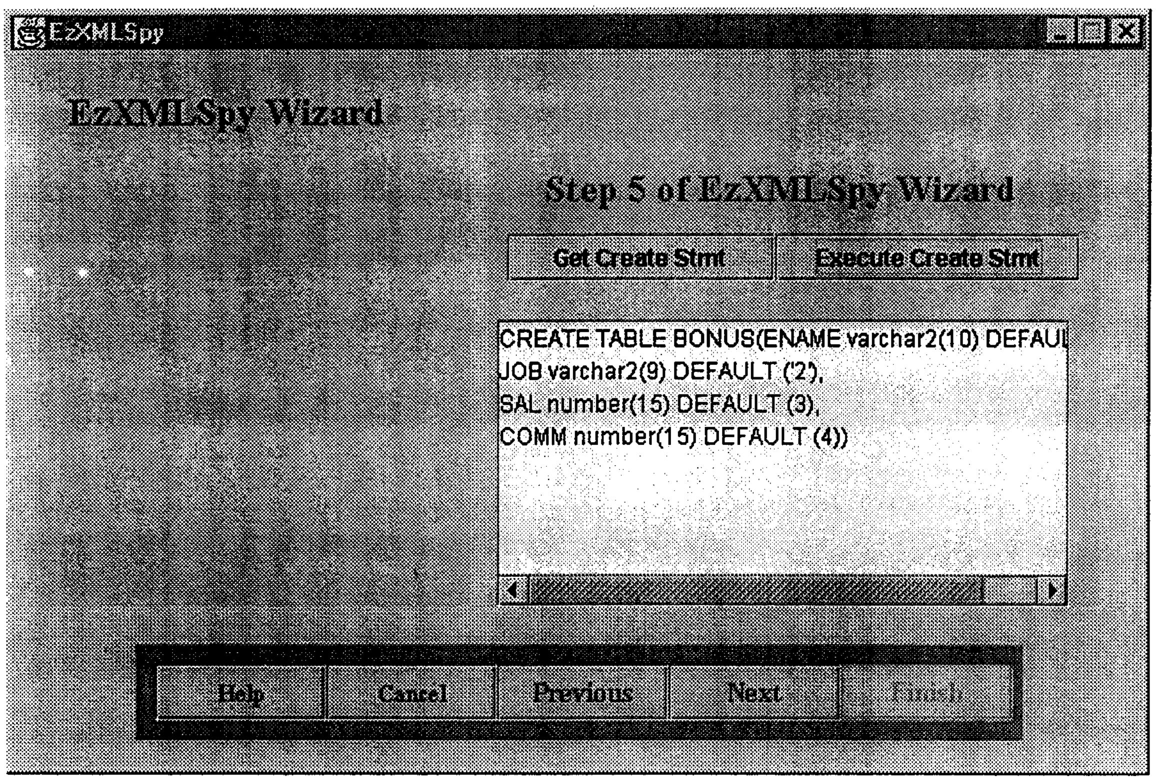
Step 3 of EzXMLSpy Wizard

Type the URL of Database to connect

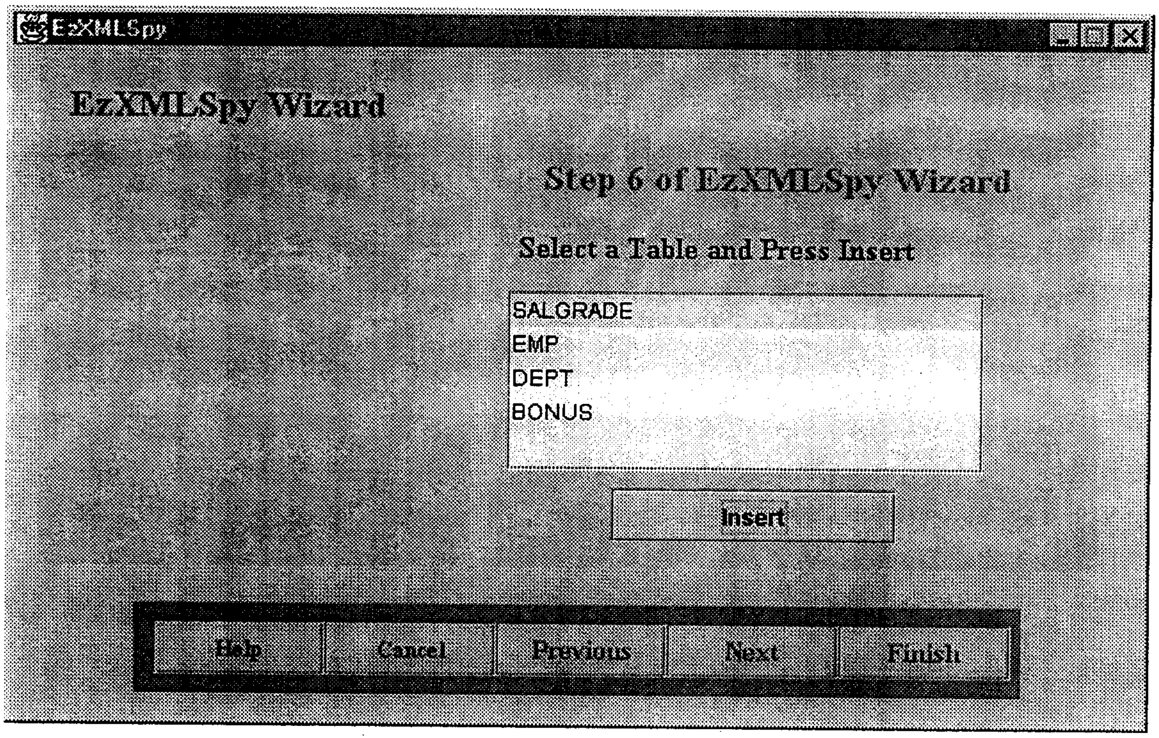
Database URL jdbc:odbc:anil

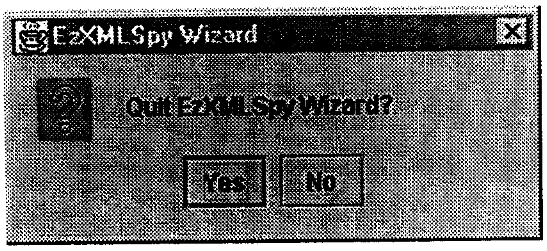
URL Format is
jdbc:odbc:Data Source Name

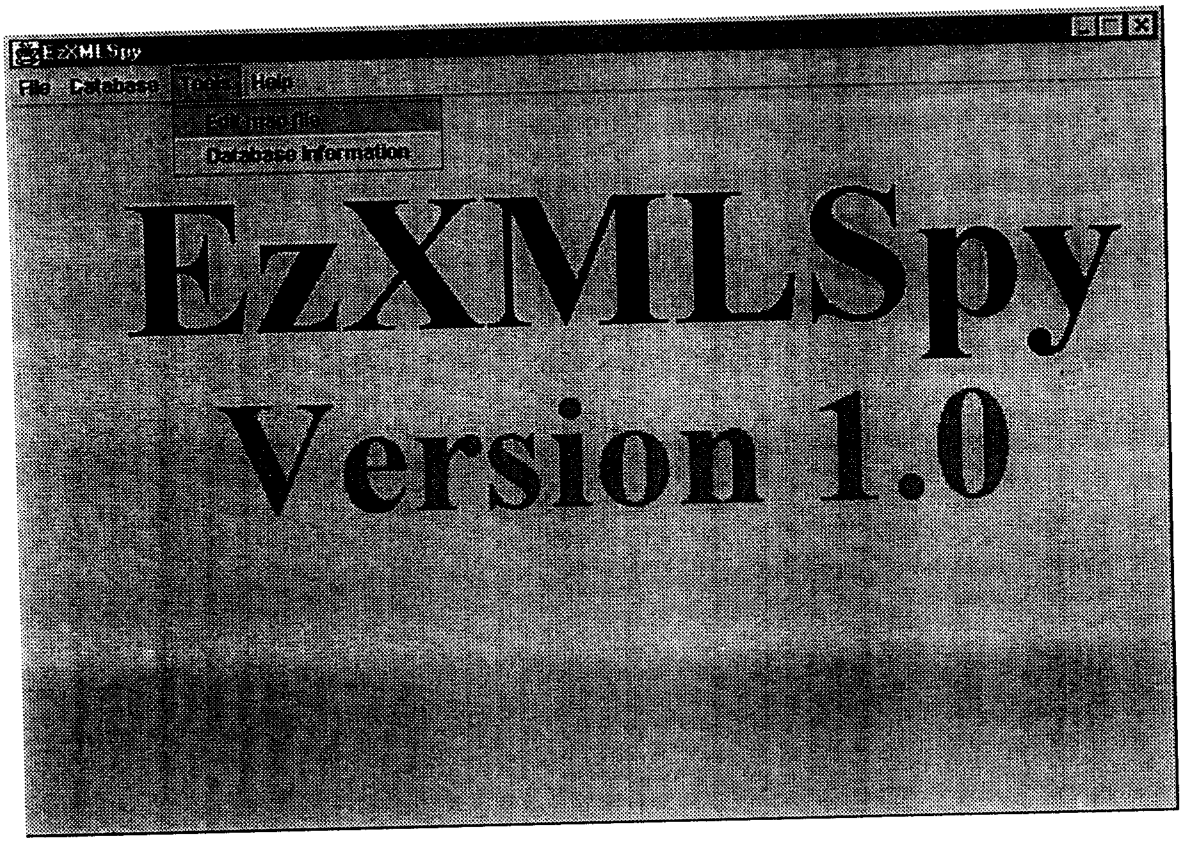

EzXMLSpy
EzXMLSpy Wizard
Step 4 of EzXMLSpy Wizard
User Name
scott
Password
*****
Connect
Oracle
Help
Cancel
Previous
Next
Finish

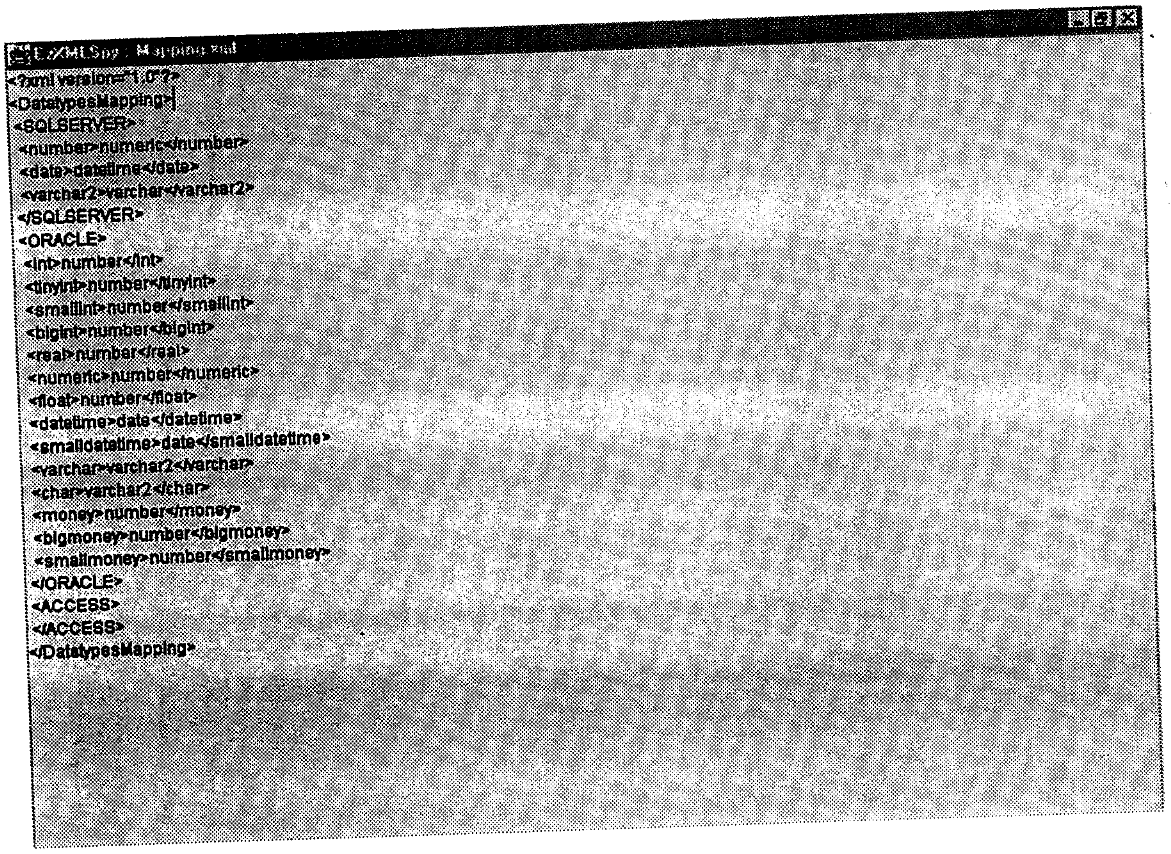


```
CREATE TABLE BONUS(ENAME varchar2(10) DEFAUL
JOB varchar2(9) DEFAULT ('2'),
SAL number(15) DEFAULT (3),
COMM number(15) DEFAULT (4))
```

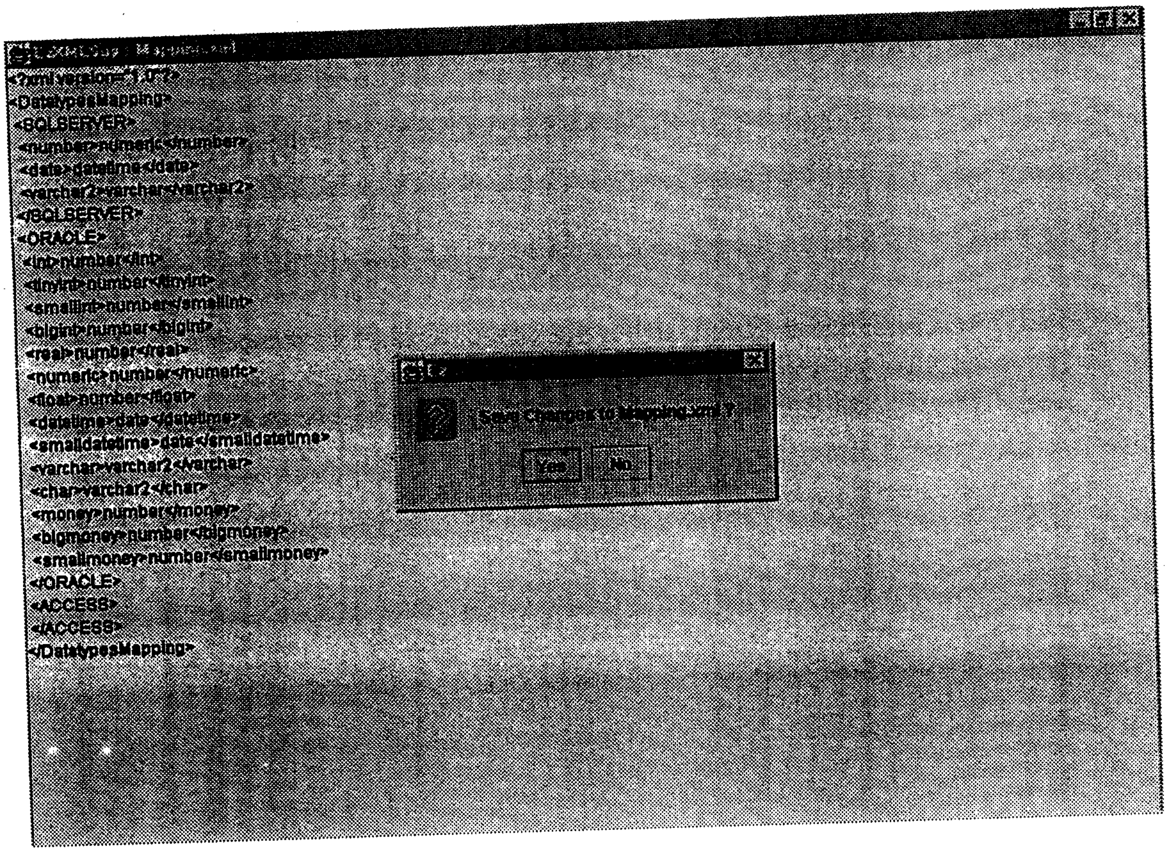

EzXMLSpy
EzXMLSpy Wizard
Step 6 of EzXMLSpy Wizard
Select a Table and Press Insert
SALGRADE
EMP
DEPT
BONUS
Insert
Help
Cancel
Previous
Next
Finish


EzXMLSpy Wizard
Quit EzXMLSpy Wizard?
Yes
No


EzXMLSpy
File
Database
Tools
Help
Edit map file
Database information
EzXMLSpy
Version 1.0

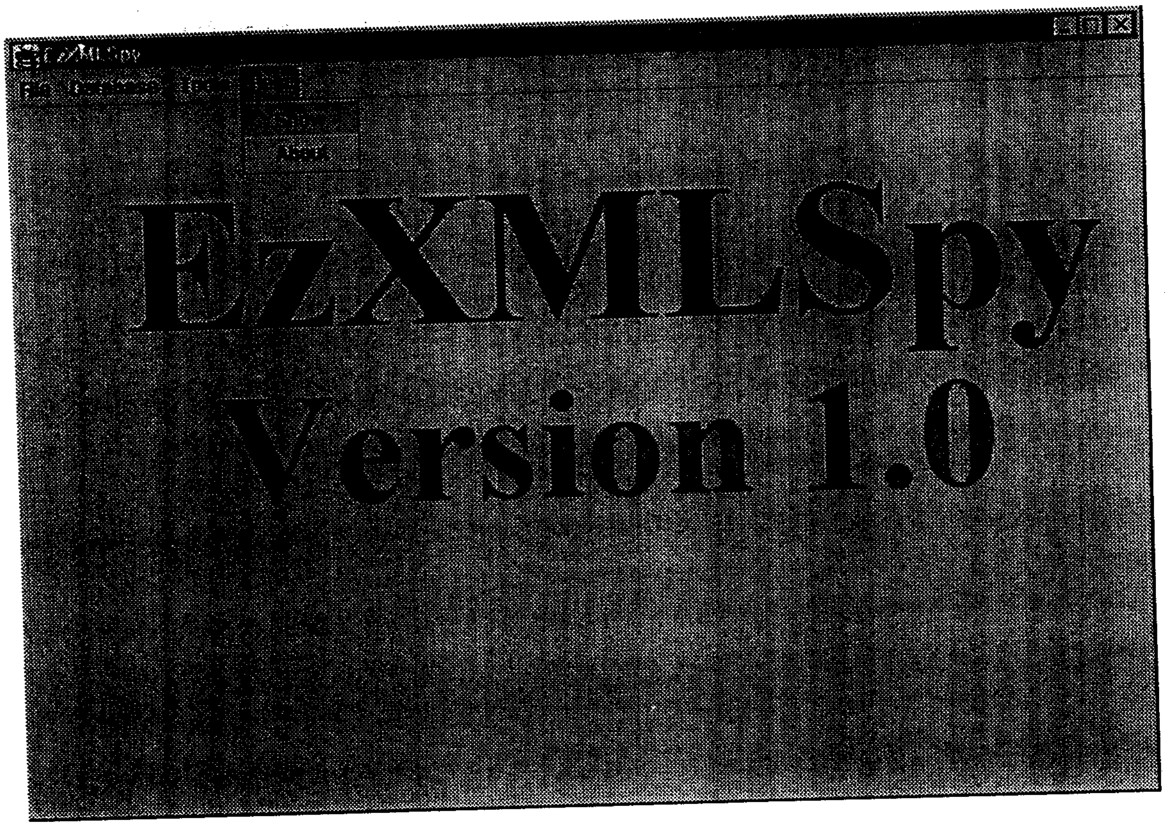


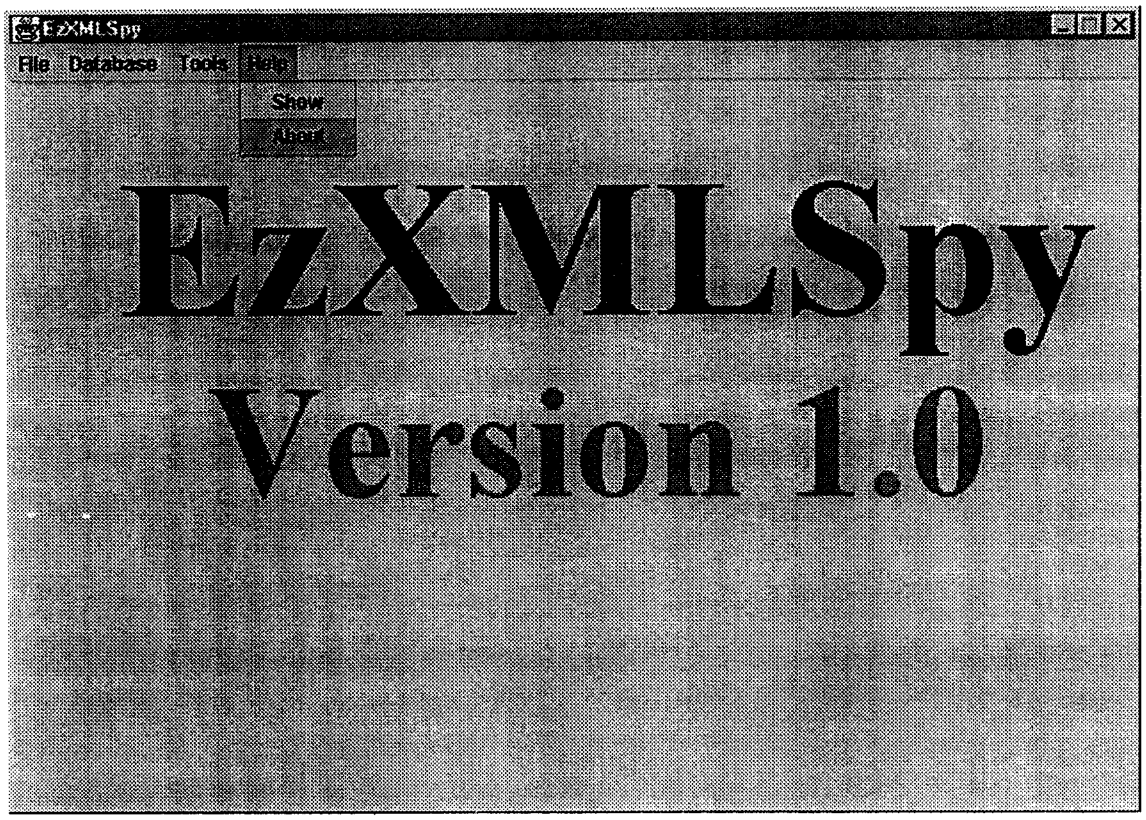
```
<?xml version="1.0"?>
<DatatypesMapping>
 <SQLSERVER>
 <number>numeric</number>
 <date>datetime</date>
 <varchar2>varchar</varchar2>
 </SQLSERVER>
 <ORACLE>
 <int>number</int>
 <tinyint>number</tinyint>
 <smallint>number</smallint>
 <bigint>number</bigint>
 <real>number</real>
 <numeric>number</numeric>
 <float>number</float>
 <datetime>date</datetime>
 <smalldatetime>date</smalldatetime>
 <varchar>varchar2</varchar>
 <char>varchar2</char>
 <money>number</money>
 <bigmoney>number</bigmoney>
 <smallmoney>number</smallmoney>
 </ORACLE>
 <ACCESS>
 </ACCESS>
</DatatypesMapping>
```

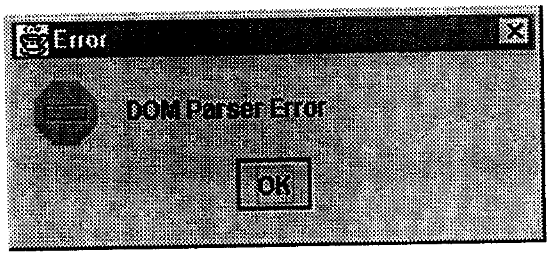


```
<?xml version="1.0"?>
<DatatypesMapping>
 <SQLSERVER>
  <number>numeric</number>
  <date>datetime</date>
  <varchar2>varchar</varchar2>
 </SQLSERVER>
 <ORACLE>
  <int>number</int>
  <tinyint>number</tinyint>
  <smallint>number</smallint>
  <bigint>number</bigint>
  <real>number</real>
  <numeric>number</numeric>
  <float>number</float>
  <datetime>date</datetime>
  <smalldatetime>date</smalldatetime>
  <varchar>varchar2</varchar>
  <char>varchar2</char>
  <money>number</money>
  <bigmoney>number</bigmoney>
  <smallmoney>number</smallmoney>
 </ORACLE>
 <ACCESS>
 </ACCESS>
</DatatypesMapping>
```

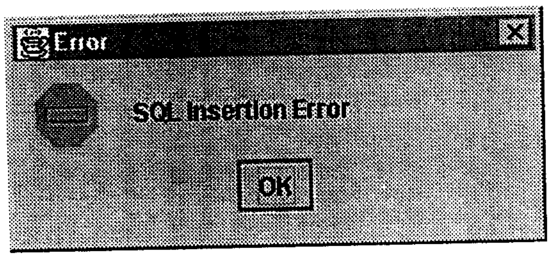
Save Changes to Mapping.xml ?


EzXMLSpy
About
EzXMLSpy
Version 1.0

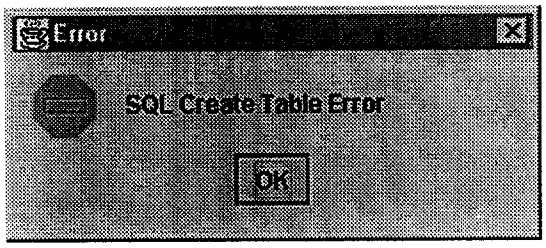

EzXMLSpy
File
Database
Tools
Help
Show
About
EzXMLSpy
Version 1.0



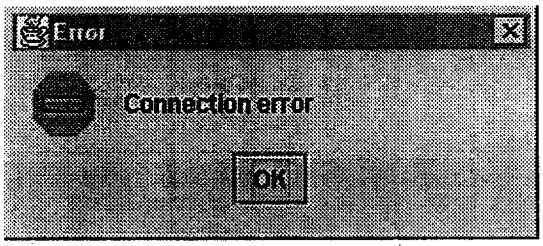
DOM Parser Error



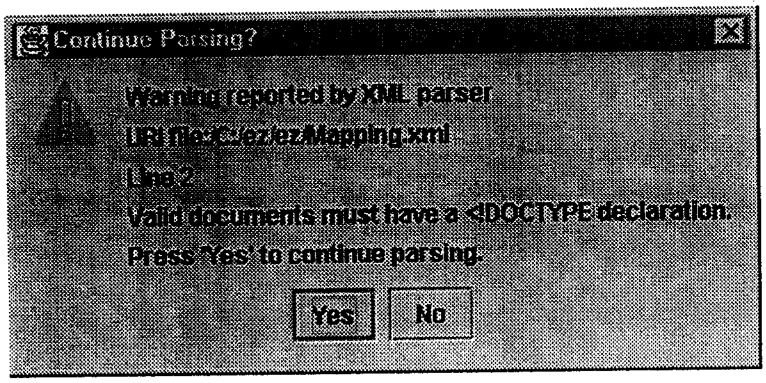
SQL Insertion Error



SQL Create Table Error



Connection error



Warning reported by XML parser

URI file:C:/ez/ez/Mapping.xml

Line 2

Valid documents must have a <!DOCTYPE declaration.

Press 'Yes' to continue parsing.