

# **EZXMLSPY**

P-792

**PROJECT WORK DONE AT**

**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

**Selvamani K. 9938M0638**

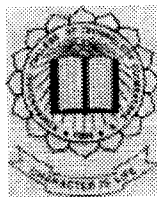
GUIDED BY

Mr. Suresh Sirigineedi, B.Tech.

EXTERNAL GUIDE

Mr. Ramasubramanian K. MCA

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**

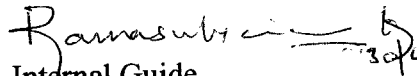
Done by

**Selvamani K.**


**9938M0638**

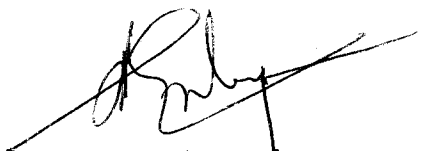
Submitted in partial fulfillment of the requirements for the award of the degree of  
**Master of Computer Applications of Bharathiar University.**

  
Professor and Head 30/4/02

  
Internal Guide 30/4/02

Submitted for the University Examination held on 10.5.2002

  
Internal Examiner 10/5/02

  
External Examiner

# PREMIER TECHNOLOGY GROUP PVT. LTD.

'Riaz Garden', 4<sup>th</sup> Floor, No. 29 Kodambakkam High Road, (Near Palm Grove Hotel)  
Nungambakkam, CHENNAI - 600 034. Ph : 8213945

---

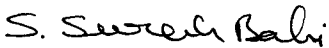
Date : April 22, 2002.

## CERTIFICATE

This is to Certify that Mr. K.Selvamani, 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

  
(Arun Arumugham)  
Manager - HR & Admin

  
Suresh Sirigineedi  
Sr.Project Leader

## **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.

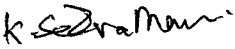
I am also grateful to my friends for their valuable comments and constructive criticisms, without which the success of this project would not have been possible.

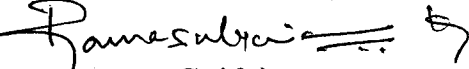
## DECLARATION


I hereby declare that the project entitled 'EzXMLSpy – Generation of XML documents from databases', 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: 30.4.2002

  
(K. Selvamani)

  
(Internal Guide)

  
(External Guide)

## DECLARATION

I have not disclosed not 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: k. Selvamani  
Name: SELVAMANI K  
Date: 30.4.2002  
Place: COIMBATORE

## **SYNOPSIS**

The project aims to generate eXtensible Markup Language (XML) from existing databases and represent the data in the database using the XML syntax.

Normally a database has two types of information, one the values stored in the database and the other is the information about those values stored, which is called metadata.

The XML document generated by the proposed system, which contains the database values, is called data XML. The XML document, which contains the metadata about the database, is called metadata XML.

The proposed system generates these two XML documents for the databases for which the user wants the XML representation.

The data XML and metadata XML documents generated conform to the predefined data Document Type Definition (DTD) and metadata DTD files respectively.

## CONTENTS

<b>1. Introduction</b>	
<b>1.1 Project Overview</b>	<b>1</b>
<b>1.2 Organization Profile</b>	<b>2</b>
<b>2. System Study and Analysis</b>	
<b>2.1 Existing System</b>	<b>4</b>
<b>2.2 Proposed System</b>	<b>5</b>
<b>2.3 Requirements on new System</b>	<b>6</b>
<b>2.4 Feasibility Study</b>	<b>9</b>
<b>3. Programming Environment</b>	
<b>3.1 Hardware Configuration</b>	<b>10</b>
<b>3.2 Software Used</b>	<b>10</b>
<b>4. System Design and Development</b>	
<b>4.1 Input Design</b>	<b>11</b>
<b>4.2 Output Design</b>	<b>12</b>
<b>4.3 Database Design</b>	<b>18</b>
<b>4.4 Process Design</b>	<b>19</b>
<b>5. System Testing and Implementation</b>	
<b>5.1 System Testing</b>	<b>20</b>
<b>5.2 System Implementation</b>	<b>22</b>
<b>6. Conclusion</b>	<b>23</b>
<b>7. Scope for future development</b>	<b>24</b>
<b>References</b>	<b>25</b>
<b>Appendices</b>	<b>26</b>



## 1.1 Project Overview

EzXMLSpy is intended at building a tool that converts the data in the database to its corresponding eXtensible Markup Language (XML) document and vice versa.

XML is a markup language designed specifically for delivering of information over the World Wide Web (WWW). It is the standard way to identify and describe data on the web. It is widely implementable 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 intermediate tool for transfer of database information in one platform to a database in another platform or from a database of one vendor to another vendor's database.

Since XML consists of only text it can be easily transported over the Internet. XML is currently the most promising language for storing and delivering information on the World Wide Web (WWW).

Although Hyper Text Markup Language (HTML) is presently the most common language used to create web pages, HTML has a limited capacity for storing information. In contrast XML has a highly flexible syntax that allows us to use it to describe virtually any kind of information from a simple page to a complex database.

Because an XML document so effectively structures and labels the information it contains, the browser can find, extract, sort, filter, arrange and manipulate that information in highly flexible ways.

XML thus provides an ideal situation for handling the rapidly expanding quantity and complexity of information that needs to be put on the web.

Normally a database consists of two types of data. The first is the actual data stored in the database and the second is the information about the actual data, which is the metadata.

The proposed system translates the database information into the corresponding XML document, which contains the data and metadata.

## **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. PTG belongs to Private Sector and is a MNC company.

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 (CRM)
- Enterprise Resource Planning (ERP) implementation and configuration
- Data migration
- Data warehousing

# **System Study and Analysis**

## **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 through Internet.
- Separate programs for transferring of database information into another platform.
- When the user needs XML representation of the database 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 database information and converts it into the corresponding XML format. The output generated will contain both the data and the metadata about the database. The project aims at a tool, which automatically generates XML 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 connection parameters like database Uniform Resource Locator (URL), username, password and the database driver to be used through the easy to use wizard and establish connection with the database. After the connection is successfully established the wizard leads the user to a screen, which shows the list of, all table names. Then the user has to select the table names.

This will lead to the Generation wizard in which the user provides the generation options. In the Generation Wizard the user specifies whether he or she wants to generate XML for all tables or for only the selected tables. The user also specifies whether he or she wants data XML or the metadata XML or both.

One more option of getting XML document for a SELECT statement is also available. Here the user can send queries to the database using the WHERE clause. Using this option the user can select specific rows and generate XML documents for those rows only.

## **2.3 Requirements on new system**

Requirement Specification is focused specifically on functioning of the 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 database to XML documents. 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 Java enabled platform.

### **Product perspective**

The output of the product is the XML form of the database information. Database values are different from a XML document, so it introduces several conversion issues. The product reads information from the database and converts into corresponding XML documents.

### **Product function**

The product converts the Database information into corresponding XML documents. The user can choose from the tables available in the database or generate XML for all the tables in the database.

### **User characteristics**

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

The product can be use with Online Information Systems that read information from the database and display it over the Internet through rendered 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 intermediate 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 files.

## **Functional requirements**

The main class of the product is the one, which establishes connection with the database generates XML and closes the connection.

Other functions include generating XML for a SELECT statement, 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,

Output filenames.

## **Information processing required**

1. Establish connection with the database
2. Read the database data and metadata
3. Map the data in database to data XML
4. Map the metadata of database to metadata XML
5. Map foreign key references
6. Generate metadata XML which conforms to metadata DTD (Document Type Definition)
7. Generate data XML which conforms to data DTD
8. Display the output XML files by using appropriate parsers

## **Design constraints**

### **Standards compliance**

Java 2 Standard

World Wide Web Consortium (W3C) XML 1.0 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 for which the user wants to generate XML documents.

## **2.4 Feasibility Study**

### **Economic Feasibility**

The cost in developing the proposed system involves cost of hardware, software and power consumption. Hardware is already present, the software is freely downloadable and the only cost involved is for the power consumption.

### **Technical Feasibility**

The proposed product is to be developed on Windows 2000 platform with Java 2.0. It can be ported to any platform.

(a) **Development Risk:**

The proposed product can be built, but doing it within the deadline of 4 months is a risk.

(b) **Resources Available:**

Suitable hardware and software.

Windows, JDK 1.3 and XML parser library.

(c) **Technology:**

High-speed processors and softwares are available and no bar for technology is seen.

# **Programming Environment**

### **3.1 Hardware Configuration**

- Processor** - Pentium III, 833 Mega Hertz
- RAM** - 128 Mega Bytes
- Hard Disk** - Seagate 20 Giga Bytes
- Monitor** - Samsung 14" color
- Keyboard** - Logitech
- Mouse** - Logitech

### **3.2 Software Used**

- Operating System** - Windows 2000 Professional
- Development** - JDK 1.3, Java Project X
- Tools** - Technology 2 Release XML Parser.
- Others** - Internet Explorer 5.5

# **System Design and Development**

## 4.1 Input Design

### List of methods used

#### **DTX()** /\*Constructor\*/

**Description:** This method constructs a Database to XML object and establishes connection with the database.

**Parameters:** Driver name, URL, Username, password

#### **generateXML()**

**Description:** This method generates XML for the table names in the list. It stores XML in a String buffer.

**Parameters:** Vector containing names of tables.

#### **generateMetaXML()**

**Description:** This method generates metadata XML for the table names in the list. It stores metadata XML in a String buffer.

**Parameters:** Vector containing names of tables.

#### **getValidTableList()**

**Description:** Gives the list of tables, which are accessible by a particular user.

**Parameter:** None

**Return value:** Vector containing the list of accessible table.

#### **foreignKeyStatus()**

**Description:** Stores the foreign key status of a particular key in a table in metadata XML buffer.

**Parameters:** Table name and column name.

#### **getConnectionTime()**

**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.

## 4.2 Output design

The output of the product is the metadata and data XML documents. The user gives the names of the output XML file names in the final step of the generation wizard.

The output XML files are valid and they conform to the metadata DTD file and data DTD file.

### Design of metadata output XML

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

```
<!ENTITY%DATA_TYPE_CODES
"(BIGINT|BINARY|BINARY|BIT|BYTE|CHAR|COUNTER|CURRENCY|DATE|DATE
TIME|DECIMAL|DOUBLE|FLOAT|GUID|IMAGE|INT|INTEGER|LONG|LONGRA
W|LONGBINARY|LONGCHAR|MONEY|NCHAR|NTEXT|NUMBER|NVARCHAR|RA
W|REAL|ROWID|SMALLDATETIME|SMALLINT|SMALLMONEY|SQL_VARIAN
T|TEXT|TIMESTAMP|TINYINT|UNIQUEIDENTIFIER|VARBINARY|VARCHAR|V
ARCHAR2)">
<!ENTITY % YESNO_TYPE_CODES (YES|NO|UNKNOWN)">
<!ENTITY%RULES_TYPE_CODES
"(UPDATE_NO_ACTION|UPDATE_KEY_CASCADE|UPDATE_KEY_SET_NULL|U
PDATE_KEY_SET_DEFAULT|DELETE_NO_ACTION|DELETE_KEY_CASCADE|D
ELETE_KEY_SET_NULL|DELETE_KEY_SET_DEFAULT)">
<!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 **metadata XML** output 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 data output XML

The content of **data DTD** is,

```

<!ELEMENT DBDATA (Table*)>

```



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

The general format of the **data XML** output is,

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

## Sample Table Structure and Values

SQL> desc dept

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

SQL> select \* from dept;

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

SQL> desc 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> select \* from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

14 rows selected.

SQL> desc salgrade

Name	Null?	Type
GRADE		NUMBER
LOSAL		NUMBER
HISAL		NUMBER

SQL> select \* from salgrade;

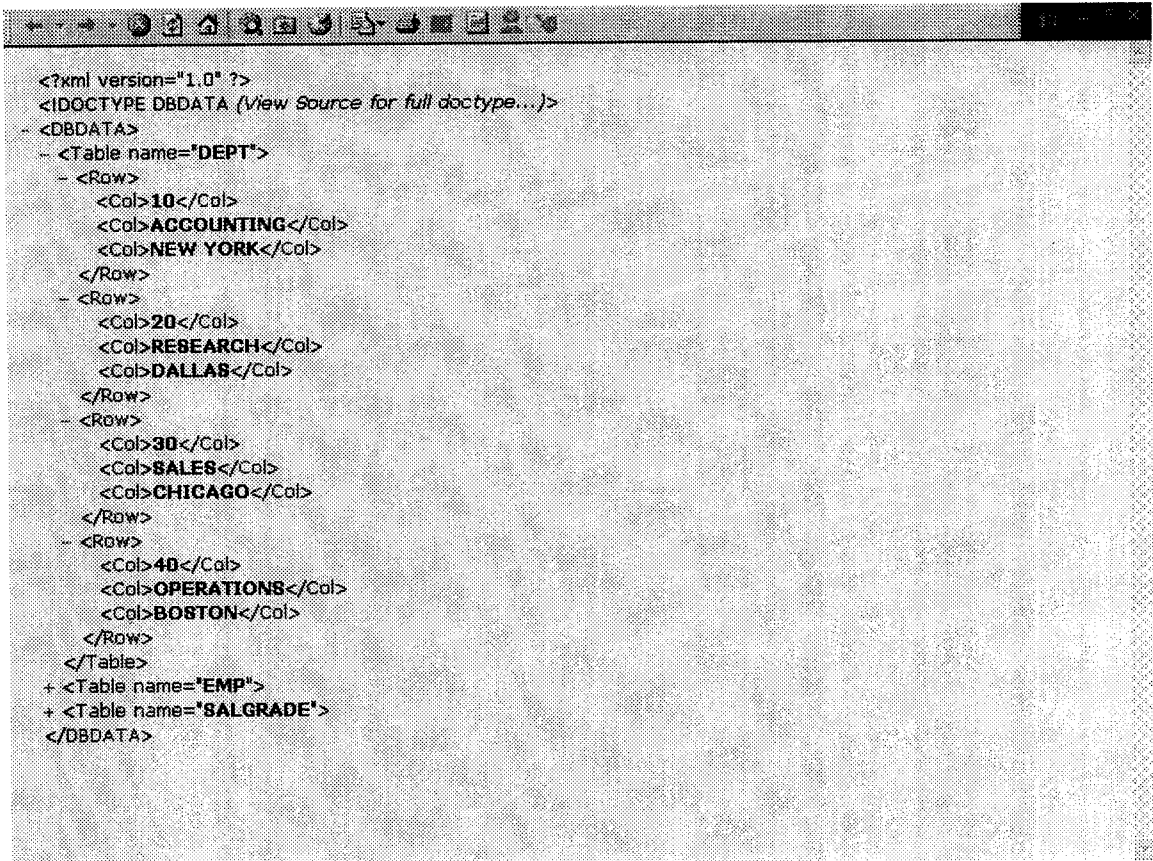
	GRADE	LOSAL	HISAL
1	700	1200	
2	1201	1400	
3	1401	2000	
4	2001	3000	
5	3001	9999	

SQL>

**Sample metadata XML Output as seen on Internet Explorer 5.5 for the sample tables**

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

## Sample data XML Output as seen on Internet Explorer 5.5 for the sample tables

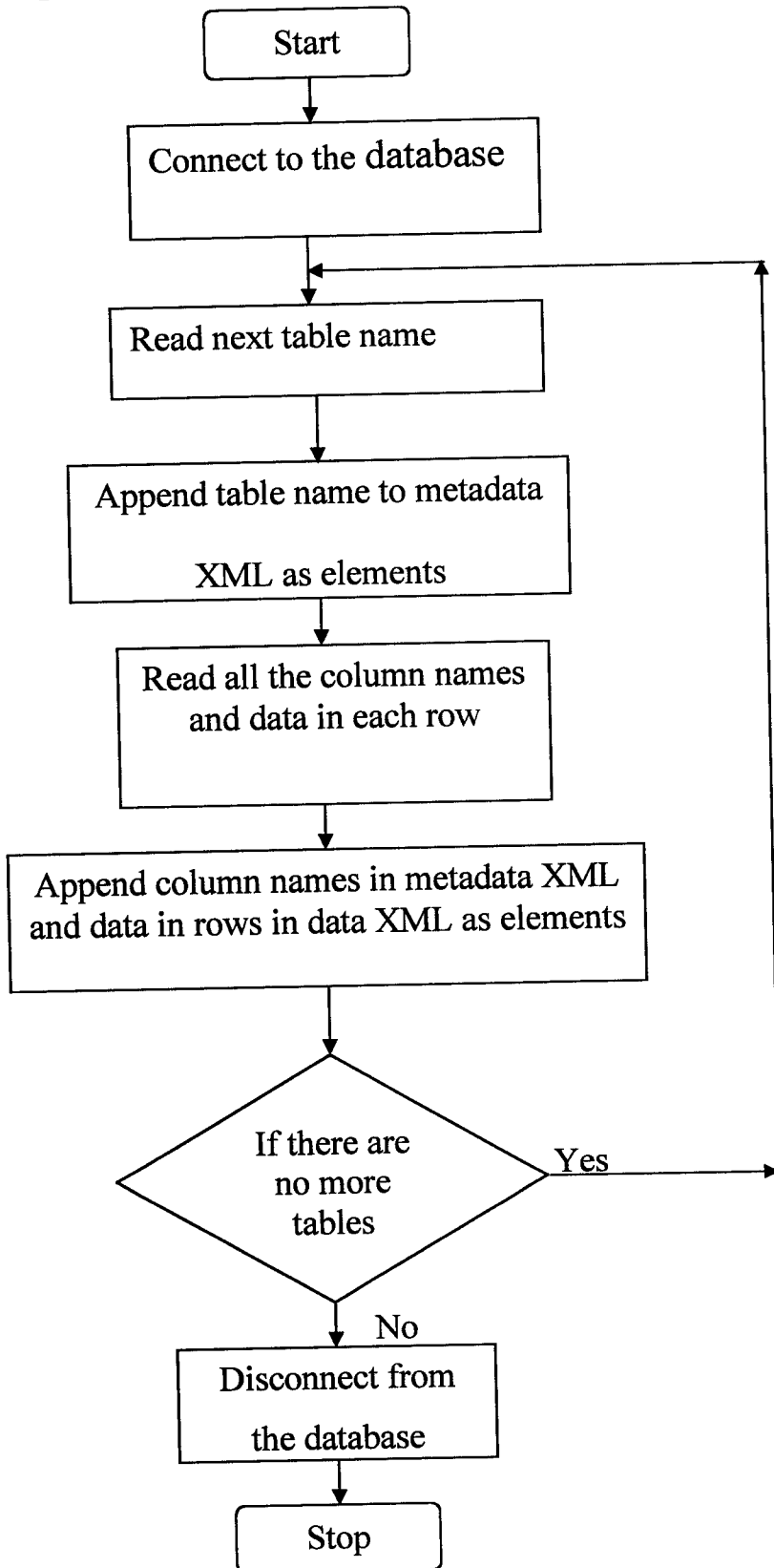


```
<?xml version="1.0" ?>
<!DOCTYPE DBDATA (View Source for full doctype...)>
- <DBDATA>
- <Table name="DEPT">
- <Row>
  <Col>10</Col>
  <Col>ACCOUNTING</Col>
  <Col>NEW YORK</Col>
</Row>
- <Row>
  <Col>20</Col>
  <Col>RESEARCH</Col>
  <Col>DALLAS</Col>
</Row>
- <Row>
  <Col>30</Col>
  <Col>SALES</Col>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

### **4.3 Database design**

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

#### 4.4 Process design



## **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 and error.
- A good test case is one that has probability of finding an 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 for the product. Checking whether all independent paths within each module have been exercised at least once.

- Exercise all logical decisions on their true 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 database parameters and generates the output XML files. 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 so that debugging is easily done. This goal can be achieved by making the source code as clear and straightforward 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.

The product converts the data in the database to XML elements, attributes and values in the output XML documents. XML Parser is used to display the output XML in a tree view format.

**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, system testing and implementation.

**Scope for future  
development**

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

- Reading multimedia data.
- Storing vector graphics.
- Generation of WML decks.
- Encoding and displaying RNA, DNA and protein sequence information.
- Generation 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:** Kevin Williams and other Group of authors

**Publication:** Wrox Press

### **Web references**

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

<http://www.w3schools.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 include,

- Platform independent
- Compiled and interpreted
- Complete and portable
- Object oriented
- Safety of design
- Exception handling
- Multithreading

## XML Overview

eXtensible Markup Language (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 fall back 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. An 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 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 we 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 uses 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 Overview**

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 started, 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 interface 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.



## **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**

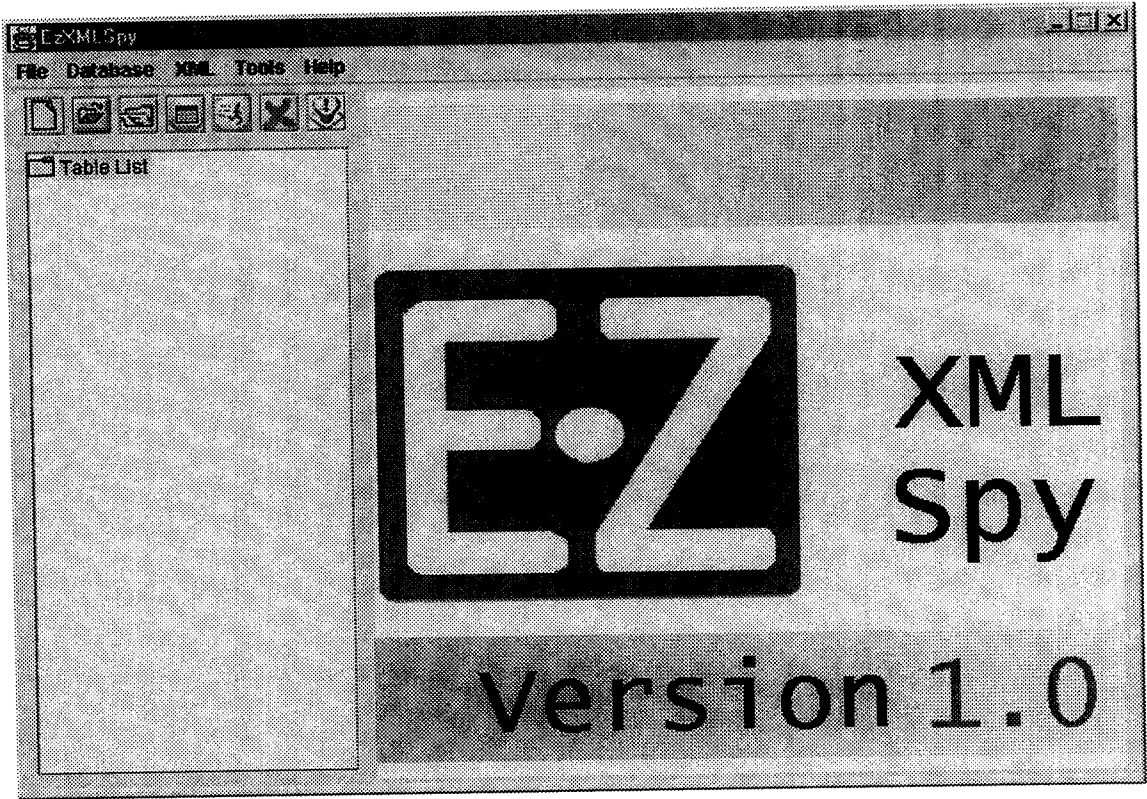
The term used to refer to an XML document when its structure and element content is consistent with that declared in its referenced or included DTD.

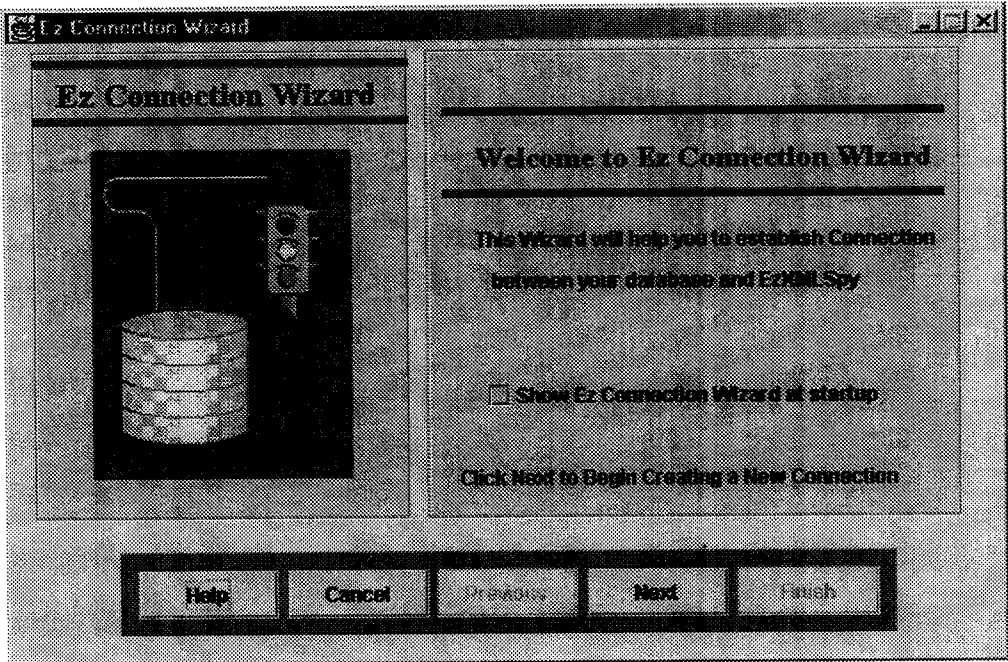
## **Well-formed**

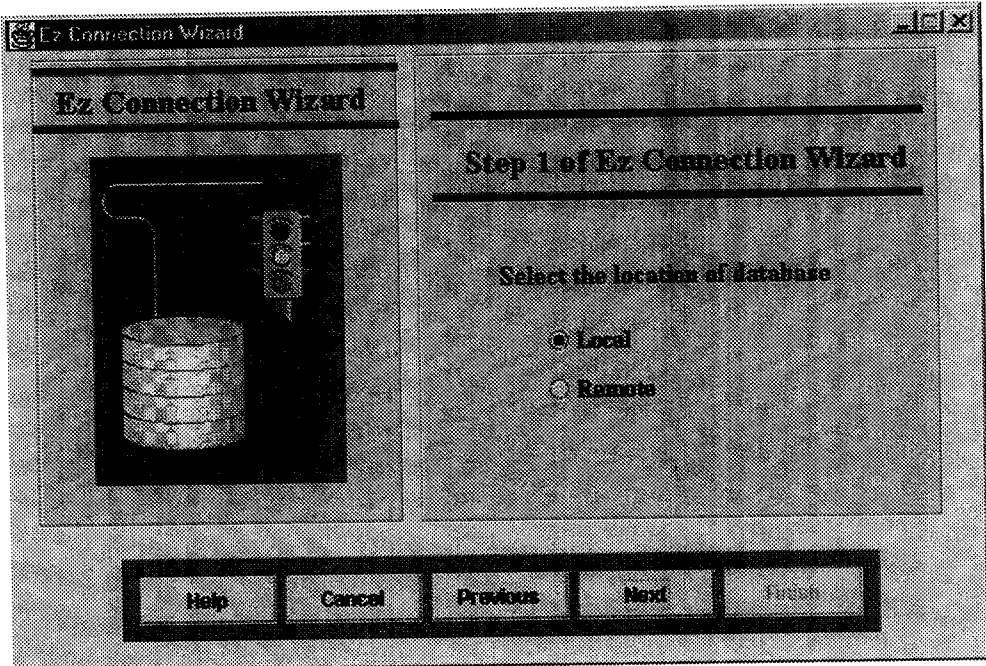
The term used to refer to an XML document that conforms to the syntax of the XML version declared in its XML declaration. This includes having a single root element, properly nested tags, and so forth.

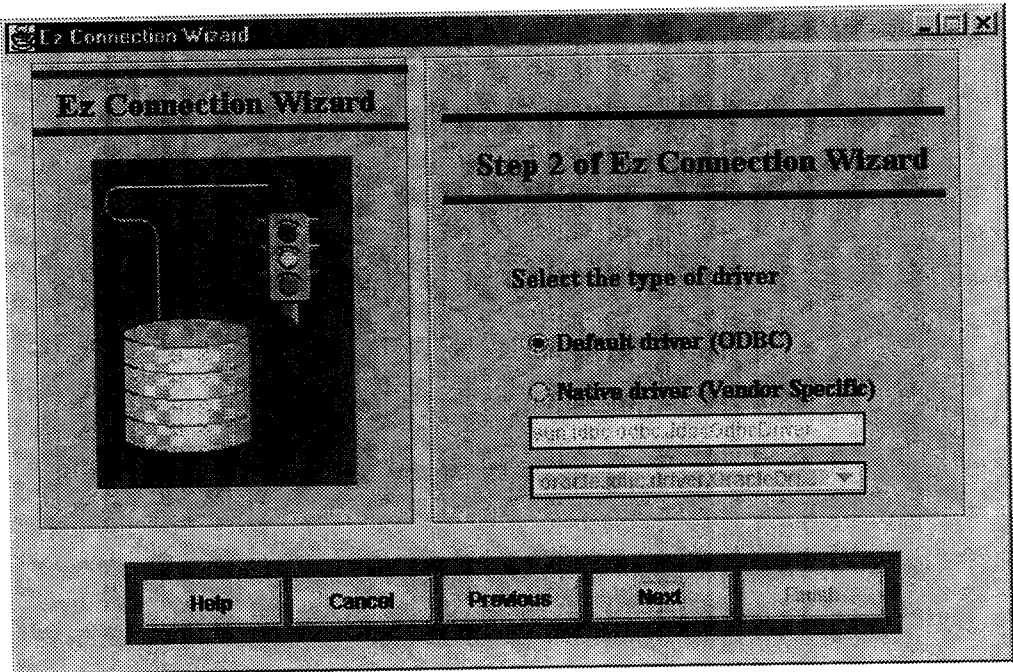
## **World Wide Web Consortium (W3C)**

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

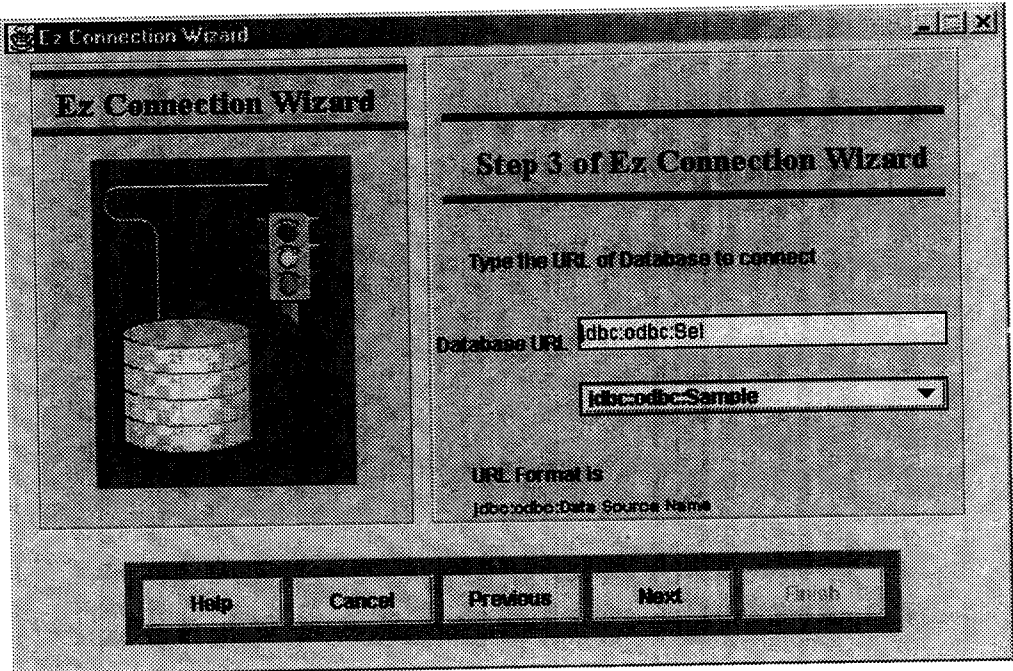


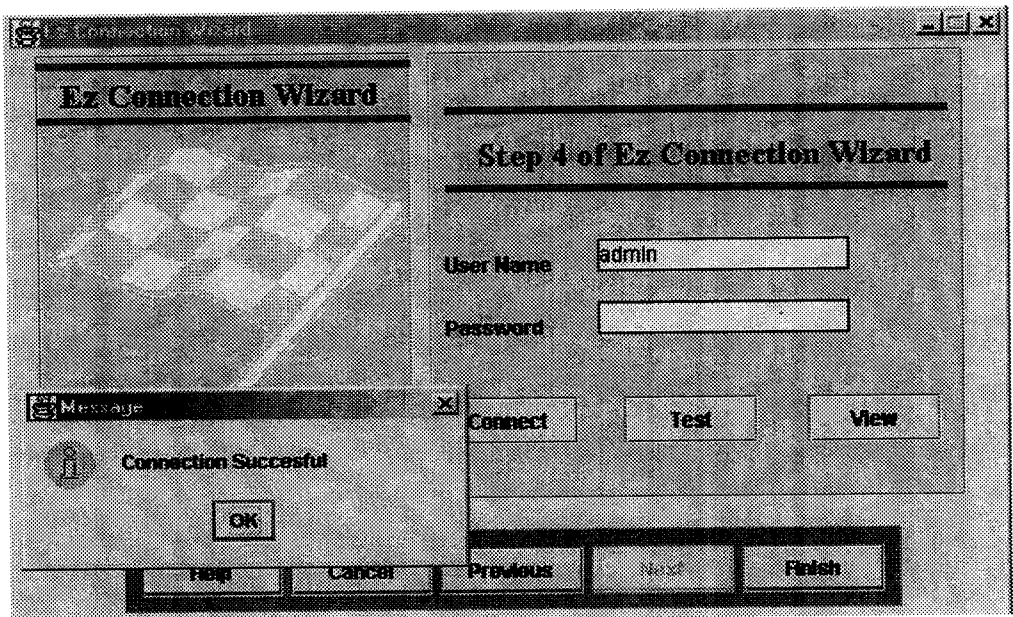


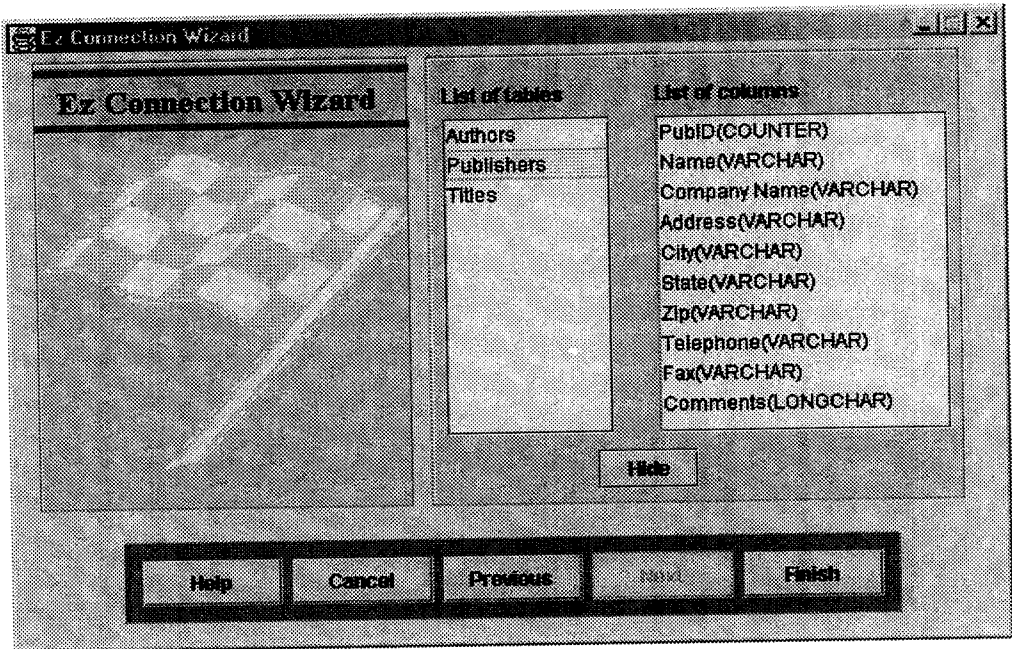


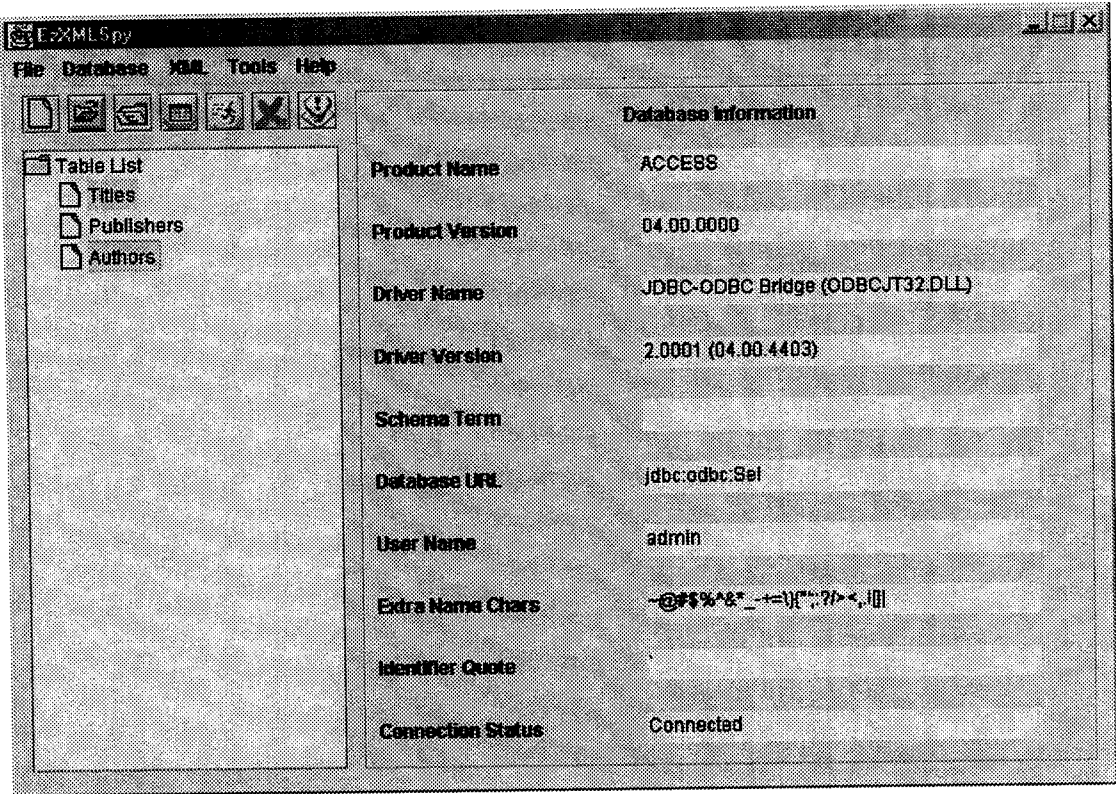


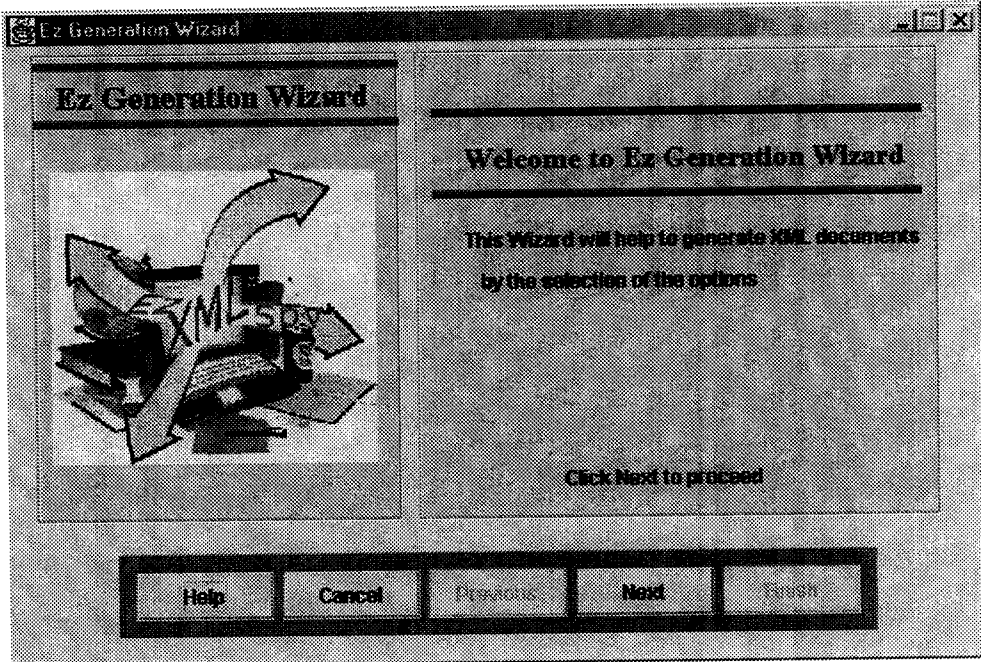


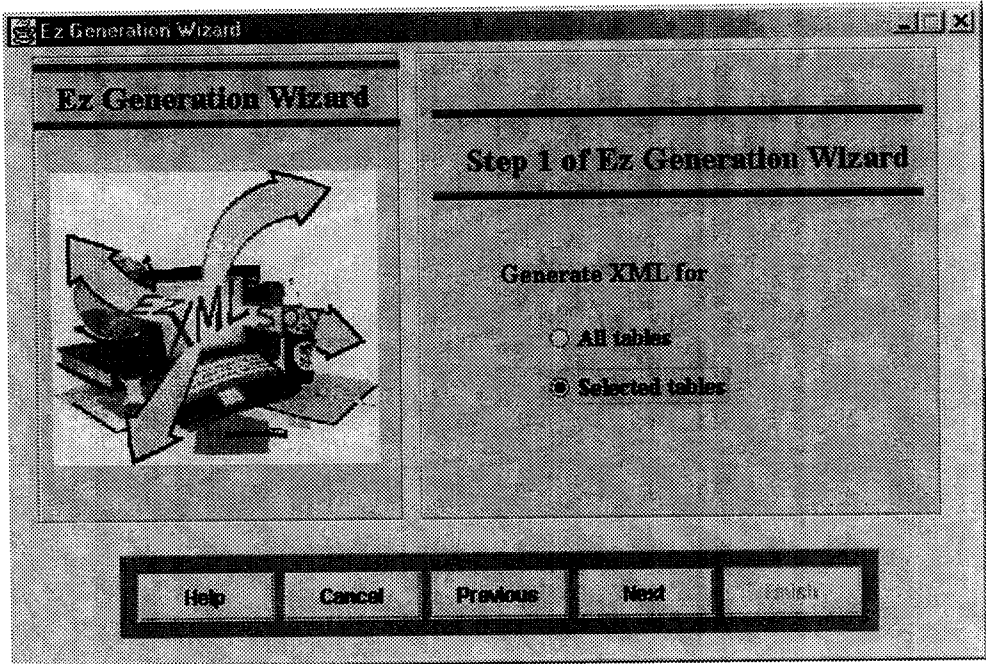


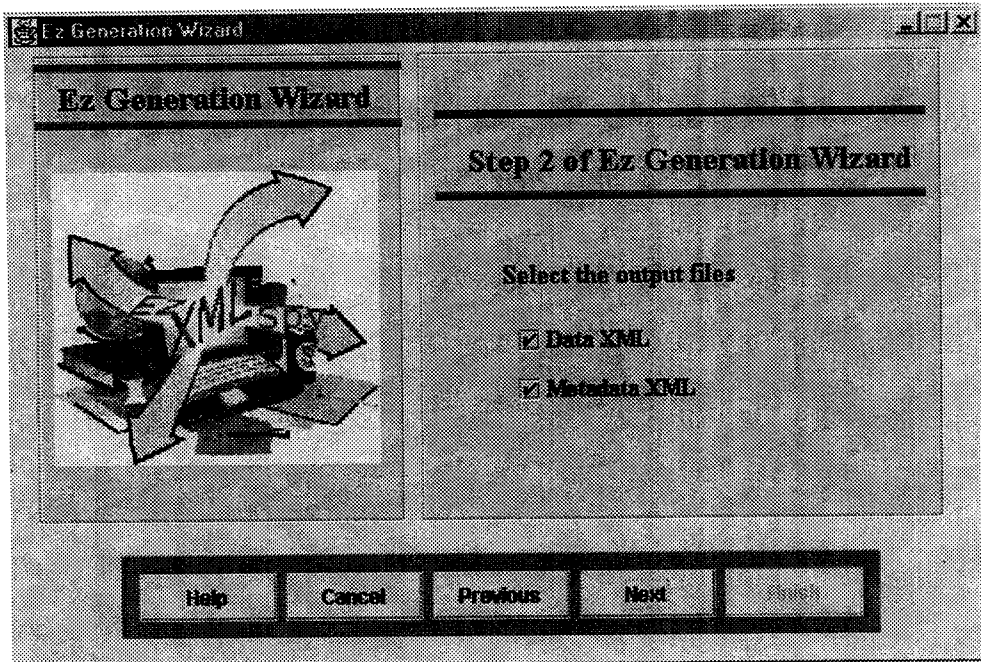


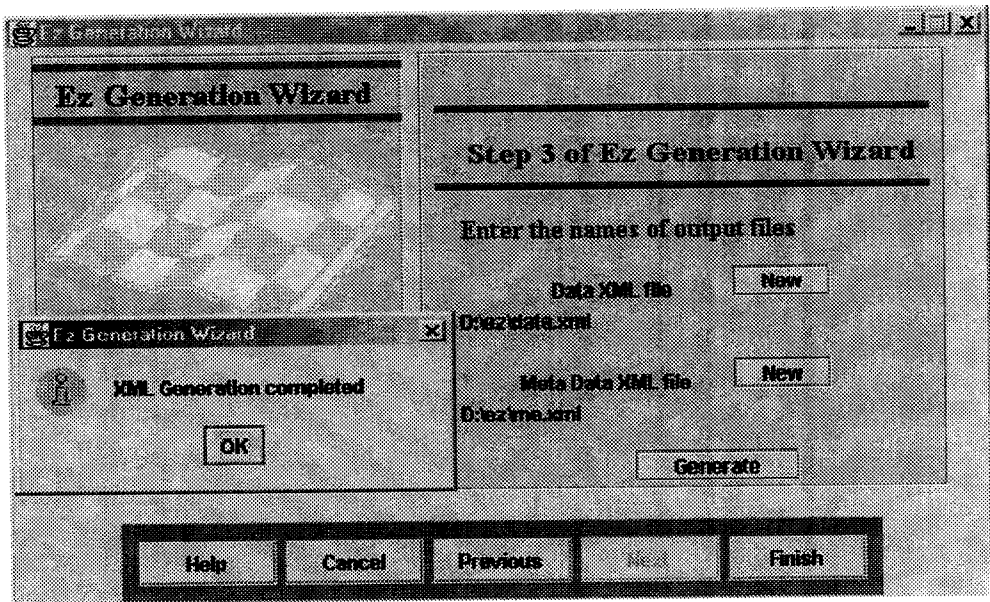














```
XMLSpy - Plain text view of D:\As\Adata and
<?xml version="1.0"?>
<!DOCTYPE DBDATA SYSTEM "data.dtd">
<DBDATA>
  <Table name="Titles">
    <Row>
      <Col>
        8088/8088, 80286, 80386, And 80486 Assembly Language Programming'
      </Col>
      <Col>
        1993
      </Col>
      <Col>
        D-0231424-7-2'
      </Col>
      <Col>
        321
      </Col>
      <Col>
        41'
      </Col>
      <Col>
        005.28520'
      </Col>
      <Col>
        .
      </Col>
      <Col>
        QA76.B.I2674B73 1993 (93000093)'
      </Col>
    </Row>
  </Table>
</DBDATA>
```

-DBMETA-

- ☐ <Table name="Titles">
  - ☐ <Column name="Title" type="VARCHAR" size="255" isnotnull="NO">
  - ☐ <Column name="Year Published" type="SMALLINT" size="5" isnotnull="NO">
  - ☐ <Column name="ISBN" type="VARCHAR" size="20" isnotnull="NO">
  - ☐ <Column name="PubID" type="INTEGER" size="10" isnotnull="NO">
  - ☐ <Column name="Description" type="VARCHAR" size="50" isnotnull="NO">
  - ☐ <Column name="Notes" type="VARCHAR" size="50" isnotnull="NO">
  - ☐ <Column name="Subject" type="VARCHAR" size="50" isnotnull="NO">
  - ☐ <Column name="Comments" type="LONGCHAR" size="1073741823" isnotnull="NO">
- ☐ <Table name="Authors">
  - ☐ <Column name="Au\_ID" type="COUNTER" size="10" isnotnull="YES">
  - ☐ <Column name="Author" type="VARCHAR" size="50" isnotnull="NO">
  - ☐ <Column name="Year Born" type="SMALLINT" size="5" isnotnull="NO">

EtXML Spy

File Database XML Tools Help

Et Query  
Enter a single table query with all columns to generate EtXML Spy compliant XML

SELECT \* FROM Publishers where pubid>600

Execute Table XML

PubID	Name	Comp	Address	City	State	Zip	Telep	Fax	Com
607	FUTU	FUTU	92 Su	Syos	NY	11791	516		
612	LOGI	LOGI	585 B	Roch	NY	14610	716		
669	RES	RES	1918	Hom	IL	60430	708		
697	MAN	MAN	P O	Carmel	IN	46032			
710	Addis	Addis	Rte 1	Read	MA	01867	617	617-9	
711	Banta	Banta	666 F	New	NY	10103	800	212-7	OEN
712	Benja	Benja	390 B	Redw	CA	94065	800	415-5	
713	Interl	Interl	2633	Anch	AK	99508			
714	McGr	McGr	1221	New	NY	10020	212		DICTI
715	Prenti	Prenti	15 C	New	NY	10023	800		
716	Prenti	Prenti	Rte 9	Engle	NJ	07032	201		
717	SRA	Scien	155	Chica	IL	60606	800	312-9	
718	Yourd	Yourd	15 C	New	NY	10023	212		
719	Micro	Micro	165 V	San	CA	9206	619		
720	McGr	Osbo	1221	New	NY	10020	800		DICTI
721	Sams	Sam	1171	Carmel	IN	46032			
722	SYBEX	SYBE	2021	Alam	CA	94501	800	415-5	PAPE
723	McGr	TAP	1221	New	NY	10020	800		

Table List  
Titles  
Publishers  
Authors

