

SEPTEMBER 21-25, 2008 | MOSCONE CENTER | SAN FRANCISCO

# ORACLE DEVELOP

## The Premier Conference for Developers

# EVELOP

# Oracle XML DB – Managing Structured and Unstructured XML with Oracle Database 11g



## Table of Content

<b>Lesson 1: Using Oracle XML DB to Store, Query, and Access XML and Relational Data .....</b>	<b>1</b>
Overview .....	1
Prerequisites .....	3
Reviewing the XML Schema in Enterprise Manager .....	4
Reviewing the XML Schema in JDeveloper .....	22
Creating a Binary XML Table .....	32
Improving Performance of XQuery Expressions through Index Creation .....	40
Using XMLType Views with XQuery .....	49
Using Relational Views over Binary XML Table .....	54
Summary .....	57
<b>Lesson 2: Performing In-Place XML Schema Evolution .....</b>	<b>58</b>
Overview .....	58
Prerequisites .....	59
Performing In-Place XML Schema Evolution .....	60
Summary .....	71
<b>Lesson 3: Using Oracle XML DB Web Services for Service-Oriented Architecture .....</b>	<b>72</b>
Overview .....	72
Creating a Binary XML Table .....	73
Using Oracle XML DB Web Services for Service-Oriented Architecture .....	79
Summary .....	112
<b>Lesson 4: Using Binary XML and XMLIndex to Aggregate and Query Unstructured XML Data Sources .....</b>	<b>114</b>
Overview .....	114
Creating a new user and grant necessary privileges .....	116
Creating a table to store information about RSS feeds .....	121
Creating a table to store RSS news items .....	125
Creating an XMLIndex on News Items .....	128
Creating an RSS View with Aggregated and Normalized News Items from Diverse News Feeds .....	129
Reading the Normalized RSS Feed with a Firefox 2.0 Browser .....	132
Summary .....	134
<b>Lesson 5: Using Binary XML, XMLIndex, and XQuery to Store and Query Unstructured XML Data .....</b>	<b>136</b>
Overview .....	136
Prerequisites .....	138
Loading Medline Data .....	139
Improving Performance of XQuery Expressions through Index Creation .....	145
Summary .....	153





# Lesson 1: Using Oracle XML DB to Store, Query, and Access XML and Relational Data

## Purpose









This tutorial shows you how to store, query, and access XML and relational data in Oracle XML DB.

## Time to Complete

Approximately 40 minutes.

## Topics

This tutorial covers the following topics:

-  [Overview](#)
-  [Prerequisites](#)
-  [Reviewing the XML Schema in Enterprise Manager](#)
-  [Reviewing the XML Schema in JDeveloper](#)
-  [Improving Performance of XQuery Expressions through Index Creation](#)
-  [Using `XMLType` Views with XQuery](#)
-  [Using Relational Views over Binary XML Table](#)
-  [Summary](#)

## Overview

Since Oracle 9i Database Release 2, Oracle XML DB has been seamlessly integrated with the Oracle database to provide high-performance database-native storage, retrieval, and management of XML data. With the new Oracle Database 11g release, Oracle XML DB is taking another leap ahead with a rich set of new capabilities to simplify DBAs' tasks in managing XML data while further empowering XML and SOA application developers. Oracle XML DB now supports multiple database-native XML storage models and XML indexing schemes, SQL/XML standard operations, W3C standard XQuery data model and XQuery/XPath languages, database-native web services, high performance XML publishing, XML DB repository, and versioning and access control. This tutorial covers these topics along with key new features for using Oracle XML DB to store, query, transform, and access XML and relational data.

### Binary XML Storage Model:

Binary XML is a new storage model for abstract data type `XMLType`, joining the existing native storage models of structured (object-relational) and unstructured (CLOB) storage. Binary XML storage provides more efficient database storage, updating, indexing, and fragment extraction than unstructured storage. It can provide better query performance than unstructured storage—it does not suffer from the XML parsing bottleneck (it is a post-parse persistence model). Like structured storage, binary XML storage is aware of XML Schema data types and can take advantage of native database data types. Like unstructured storage, no data conversion is needed during database insertion or retrieval. Like structured storage, binary XML storage allows for piecewise updates. Because binary XML data can also be used outside the database, it can serve as an efficient XML exchange medium, and you can offload work from the database to increase overall performance in many cases. Like unstructured storage, binary XML data is kept in document order. Like structured storage, data and metadata can, using binary storage, be separated at the database level, for efficiency. Like unstructured storage, however, binary storage allows for intermingled data and metadata, which lets instance structures vary. Binary XML storage allows for very complex and variable data, which in the structured-storage model could necessitate using many database tables and joins. Unlike the other `XMLType` storage models, you can use binary storage for XML schema-based data even if the XML schema is not known beforehand, and you can store multiple XML schemas in the same table and query across common elements.

### **XMLIndex Indexing for Binary XML and Unstructured XML Storage Models:**

B-Tree indexes can be used advantageously with structured storage. They provide sharp focus by targeting the underlying objects directly. They are generally ineffective, however, in addressing the detailed structure (elements and attributes) of an XML document stored in a binary XML or a CLOB instance. That is the special domain of XMLIndex: binary XML and unstructured storage models. Unlike a B-Tree index, which you define for a specific column that represents an individual XML element or attribute, an XMLIndex index is very general: indexing with XMLIndex applies to all possible XPath expressions for your XML data. An XMLIndex index presents the following advantages over other indexing methods:

- An XMLIndex index can be used for SQL/XML functions `XMLExists()`, `XMLTable()`, and `XMLQuery()`, and it is effective in any part of a query; it is not limited to use in a `WHERE` clause. This is not the case for any of the other kinds of indexes you might use with XML data.
- XMLIndex can thus speed access to `SELECT` list data and `FROM` list data, making it useful for XML fragment extraction, in particular. Function-based indexes and CTXXPath indexes
- You need no prior knowledge of the XPath expressions that will be used in queries. XMLIndex is completely general. This is not the case for function-based indexes.
- You can use an XMLIndex index with either XML schema-based or non-schema-based data. It can be used with binary XML and unstructured storage models. B-Tree indexing is appropriate only for schema-based data stored object-relationally (structured storage); it is ineffective for XML schema-based data stored in a binary XML or a CLOB instance.
- You can use an XMLIndex index for searches with XPath expressions that target collections, that is, nodes that occur multiple times within a document. This is not the case for functional indexes.

### **Oracle Database-Native XQuery:**

Since XQuery is now a W3C standard, the IT community has started adopting the business uses of XML and XQuery. As the innovation leader in commercial database technology, Oracle Database 11g provides a full-featured native XQuery engine integrated with the traditional Oracle database server. On the SQL side, the SQL/XML standard has defined a way to encapsulate XML in SQL and to integrate the querying of XML using XQuery. This is being accomplished by introducing new SQL functions: `XMLQuery`, `XMLTable`, `XMLExists`, and `XMLCast`, which operate on XML and SQL values using XQuery. Oracle Database 11g enables XQuery support in the database server through these SQL standard functions. A new `XQUERY` command has also been implemented in SQL\*Plus to allow users to enter XQuery expressions on the command line. With standards-based implementation of XQuery in Oracle Database 11g, application developers can use their favorite APIs (e.g., JDBC, ODP.NET, and web service) to access Oracle Database XQuery capabilities.

### **Benefits of Oracle XQuery:**

Using SQL/XML XQuery functions along with indexing schemes for structured, unstructured, and binary XML storage models, XML DB can perform uniform XML queries across different storage models with orders of magnitude performance improvement over DOM-based functional evaluation of XML queries. Furthermore, XML queries can be seamlessly merged with SQL relational queries to handle all query scenarios. Finally, the XML query capabilities of Oracle XML DB are built on the solid foundation of industry's best relational database that is highly reliable, available, scalable, and secure. In short, the XML DB query capabilities in Oracle Database 11g provide the most comprehensive and efficient functionality for versatile, scalable, concurrent, and high performance XML applications.

## Prerequisites

Before you perform this tutorial, you should first complete the following steps:

1. Check your Oracle Database 11g installation and make sure the **OE**, **HR** users are unlocked.
2. Check your Oracle SQL Developer (version 1.5.1) installation
3. Check your Oracle JDeveloper 10.1.3 installation
4. Check the files in your working directory (/HOL/xmlldb\_1/files)

## Reviewing the XML Schema in Enterprise Manager

An XML Schema has been supplied as part of the OE Schema in the Sample Schema that is provided with Oracle Database 11g. In this section, you will review its contents using Enterprise Manager. Perform the following steps:

1. Open your browser and enter the following URL:

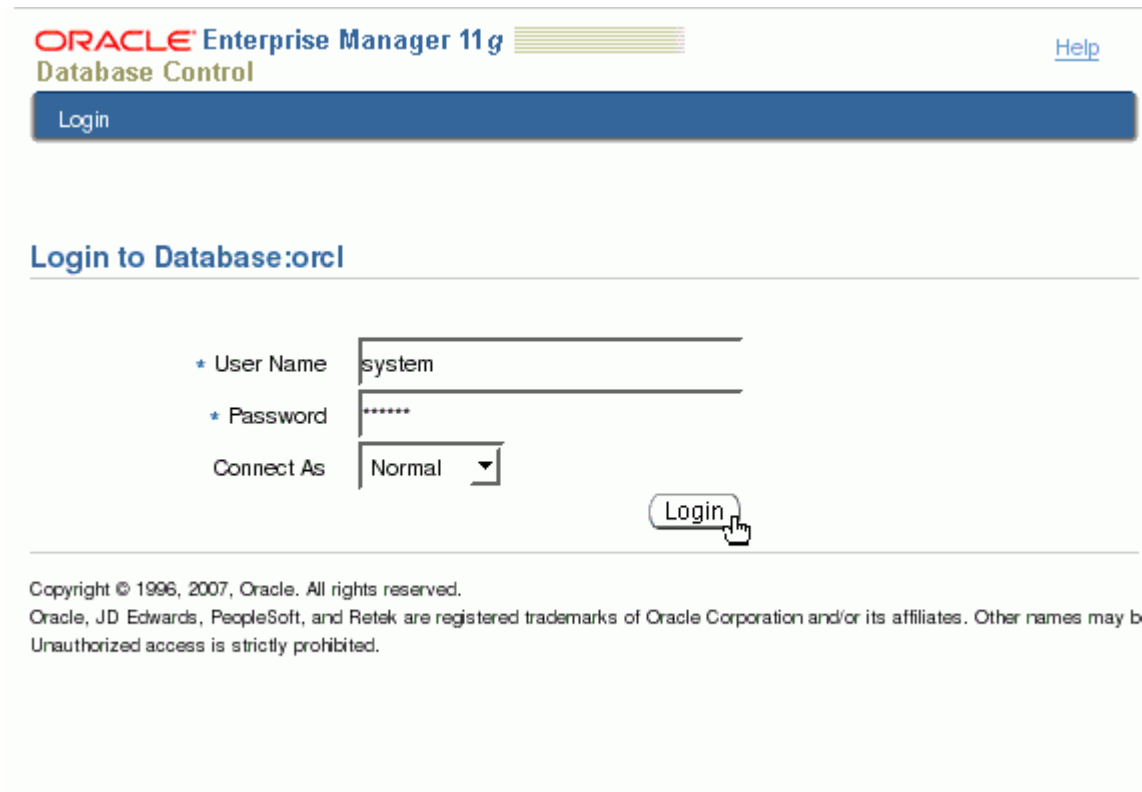
`https://localhost:1158/em`

Enter the following details, and accept the default value for **Connect As**.

User Name: **system**

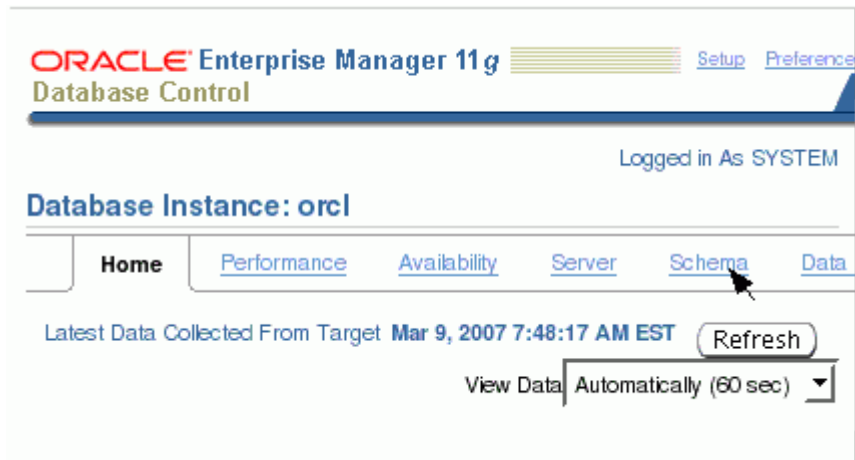
Password: **oracle**

Then, click **Login**.



The screenshot shows the Oracle Enterprise Manager 11g Database Control login interface. At the top, the header reads "ORACLE Enterprise Manager 11g Database Control" with a "Help" link on the right. Below the header is a blue "Login" button. The main section is titled "Login to Database:orcl". It contains three input fields: "User Name" with the value "system", "Password" with masked characters "\*\*\*\*\*", and "Connect As" with a dropdown menu showing "Normal". A "Login" button is positioned to the right of these fields, with a mouse cursor hovering over it. At the bottom, there is a copyright notice: "Copyright © 1996, 2007, Oracle. All rights reserved. Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be Unauthorized access is strictly prohibited."

2. Oracle Enterprise Manager 11g Database Control window is displayed. Click the **Schema** tab.



3. Under **XML Database**, select the **Configuration** link.

Database Instance: orcl

[Home](#)

[Performance](#)

[Availability](#)

[Server](#)

**Schema**

[Data Movement](#)

[Software and Supp](#)

**Database Objects**

[Tables](#)  
[Indexes](#)  
[Views](#)  
[Synonyms](#)  
[Sequences](#)  
[Database Links](#)  
[Directory Objects](#)  
[Reorganize Objects](#)

**Programs**

[Packages](#)  
[Package Bodies](#)  
[Procedures](#)  
[Functions](#)  
[Triggers](#)  
[Java Classes](#)  
[Java Sources](#)

**Change Management**

[Dictionary Baselines](#)  
[Dictionary Comparisons](#)

**Materialized Views**

[Materialized Views](#)  
[Materialized View Logs](#)  
[Refresh Groups](#)  
[Dimensions](#)

**User Defined Types**

[Array Types](#)  
[Object Types](#)  
[Table Types](#)

**XML Database**

[Configuration](#)  
[Resources](#)  
[Access Control Lists](#)  
[XML Schemas](#)  
[XMLType Tables](#)  
[XMLType Views](#)  
[XML Indexes](#)  
[XML Repository Events](#)

**Workspace Manager**

[Workspaces](#)

**Text Manager**

[Text Indexes](#)

4. Enter **2100** for FTP Port and **8080** for HTTP port. Then, click **OK**.

Database Instance: orcl >

Logged in As SYSTEM

### XDB Configuration

CancelOK

The configuration parameters listed here are from the XML Database Resource /xdbConfig.xml for configuring the Oracle XML DB protocol server.

☒ **TIP** \* indicates parameters that need a database restart for the changes to take effect.

#### Common Parameters for FTP, HTTP, and WebDAV Protocols

Session * Pool Size <small>Number of sessions in pool</small>	50	Call Timeout * (seconds)	60
Session * Timeout (seconds)	60	Default Lock * Timeout (seconds)	36

#### FTP Protocol Parameters

Session Timeout (seconds)	60	FTP Port	2100
------------------------------	----	-------------	------

Maximum Request Body Size (MB)	<input type="text" value="1907.3486328125"/> <small>Data beyond this size is ignored</small>	(seconds)	Maximum Header Size (KB)	<input type="text" value="16.0"/>
HTTP Port	<input type="text" value="8080"/>	HTTP Protocol	<input type="text" value="tcp"/>	
HTTPS Port	<input type="text"/>	HTTPS Protocol	<input type="text"/>	
<div>Welcome File List</div>				
<b>NFS Protocol Parameters</b>				
<div>Enable</div>				
NFS Port	<input type="text" value="0"/>			
<b>Related Links</b>				
<a href="#">Mime Mappings</a>				
<a href="#">Language, Encoding and Character Set Mappings</a>				
				<div>Cancel OK</div>



5. Your configuration has been set. Click **Database**.

The screenshot shows the Oracle Enterprise Manager 11g Database Control interface. At the top, the title bar reads "ORACLE Enterprise Manager 11g Database Control" with a "Database" tab selected. Navigation links include "Setup", "Preferences", "Help", and "Logout". The user is logged in as "SYSTEM". A confirmation message states: "XDB Configuration has been modified successfully". Below this, the "XDB Configuration" section is titled, with "Cancel" and "OK" buttons. A note explains that the parameters are from the XML Database Resource /xdbConfig.xml. A tip indicates that parameters marked with an asterisk (\*) need a database restart. The configuration parameters are listed in two columns:

Common Parameters for FTP, HTTP, and WebDAV Protocols	
Session * Pool Size <small>Number of sessions in pool</small>	Call Timeout * <small>(seconds)</small>
Session * Timeout <small>(seconds)</small>	Default Lock * Timeout <small>(seconds)</small>

6. Go to **Schema > XML Database**, and select the **Resources** link.

The screenshot shows the Oracle Enterprise Manager 11g Database Control interface. The title bar reads "Enterprise Manager 11g Database Control" with a "Database" tab selected. Navigation links include "Setup", "Preferences", "Help", and "Logout". The user is logged in as "SYSTEM". The "Instance: orcl" section is visible. Below it, a navigation pane shows links for "Performance", "Availability", "Server", "Schema", "Data Movement", and "Software and Support". The "Schema" link is highlighted with a mouse cursor. Below the navigation pane, the text "Latest Data Collected From Target Feb 19, 2007 10:22:17 AM EST" is displayed, along with a "Refresh" button and a "View Data" link. A dropdown menu shows "Automatically (60 sec)".

Database Instance: orcl

[Home](#)

[Performance](#)

[Availability](#)

[Server](#)

**Schema**

[Data Movement](#)

[Software and Support](#)

Database Objects

[Tables](#)  
[Indexes](#)  
[Views](#)  
[Synonyms](#)  
[Sequences](#)  
[Database Links](#)  
[Directory Objects](#)  
[Reorganize Objects](#)

Programs

[Packages](#)  
[Package Bodies](#)  
[Procedures](#)  
[Functions](#)  
[Triggers](#)  
[Java Classes](#)  
[Java Sources](#)

Change Management

[Dictionary Baselines](#)  
[Dictionary Comparisons](#)

Materialized Views

[Materialized Views](#)  
[Materialized View Logs](#)  
[Refresh Groups](#)  
[Dimensions](#)

User Defined Types

[Array Types](#)  
[Object Types](#)  
[Table Types](#)

XML Database

[Configuration](#)  
[Resources](#)  
[Access Control Lists](#)  
[XML Schemas](#)  
[XMLType Tables](#)  
[XMLType Views](#)  
[XML Indexes](#)  
[XML Diagnostic Events](#)

Workspace Manager

[Workspaces](#)

Text Manager

[Text Indexes](#)

7. All the resources are displayed. Expand **home**.

**ORACLE® Enterprise Manager 11g** Database Control [Setup](#) [Preferences](#) [Help](#) [Logout](#) **Database**

Database Instance: orcl > XML Database Resources Logged in As SYSTEM

[View Flat List](#)

XML Database Resource Name  [Go](#)

Only those XML Databases Resources for which the logged in user has permission will be shown. [Create](#)

[Edit](#) [View](#) [Delete](#)

Select	Name	Owner	Created	Last Modified
<input type="radio"/>	/			
<input checked="" type="radio"/>	<a href="#">OLAP_XDS</a>	SYS	Oct 15, 2007 10:48:52 AM CDT	Oct 15, 2007 10:48:52 AM CDT
<input type="radio"/>	<a href="#">home</a>	SYS	Aug 27, 2008 11:04:51 PM CDT	Aug 28, 2008 4:26:55 AM CDT
<input type="radio"/>	<a href="#">images</a>	SYS	Oct 15, 2007 11:24:16 AM CDT	Oct 15, 2007 11:31:14 AM CDT

8. Expand **OE - PurchaseOrders - 2002 - Apr** to show the list of XML documents. Click on the first XML document in the list.

Select	Name	Owner	Created	Last Modified
<input type="radio"/>	▼ /			
<input checked="" type="radio"/>	▶ OLAP_XDS	SYS	Oct 15, 2007 10:48:52 AM CDT	Oct 15, 2007 10:48:52 AM CDT
<input type="radio"/>	▼ home	SYS	Aug 27, 2008 11:04:51 PM CDT	Aug 28, 2008 4:26:55 AM CDT
<input type="radio"/>	▼ OE	OE	Aug 27, 2008 11:04:51 PM CDT	Aug 27, 2008 11:04:55 PM CDT
<input type="radio"/>	▼ PurchaseOrders	OE	Aug 27, 2008 11:04:53 PM CDT	Aug 27, 2008 11:05:01 PM CDT
<input type="radio"/>	▼ 2002	OE	Aug 27, 2008 11:05:01 PM CDT	Aug 27, 2008 11:05:01 PM CDT
<input type="radio"/>	▼ Apr	OE	Aug 27, 2008 11:05:01 PM CDT	Aug 27, 2008 11:05:02 PM CDT
<input type="radio"/>	AMCEWEN-20021009123336171PDT.xml	OE	Aug 27, 2008 11:05:02 PM CDT	Aug 27, 2008 11:05:02 PM CDT
<input type="radio"/>	AMCEWEN-20021009123336271PDT.xml	OE	Aug 27, 2008 11:05:02 PM CDT	Aug 27, 2008 11:05:02 PM CDT
<input type="radio"/>	EABEL-20021009123336251PDT.xml	OE	Aug 27, 2008 11:05:02 PM CDT	Aug 27, 2008 11:05:02 PM CDT

9. General information about the document is shown. To see the actual contents of the document, click **Display Contents**.

**ORACLE** Enterprise Manager 11g  
Database Control

[Setup](#) [Preferences](#) [Help](#) [Logout](#)

Database

Database Instance: orcl > XML Database Resources >

Logged in As SYS

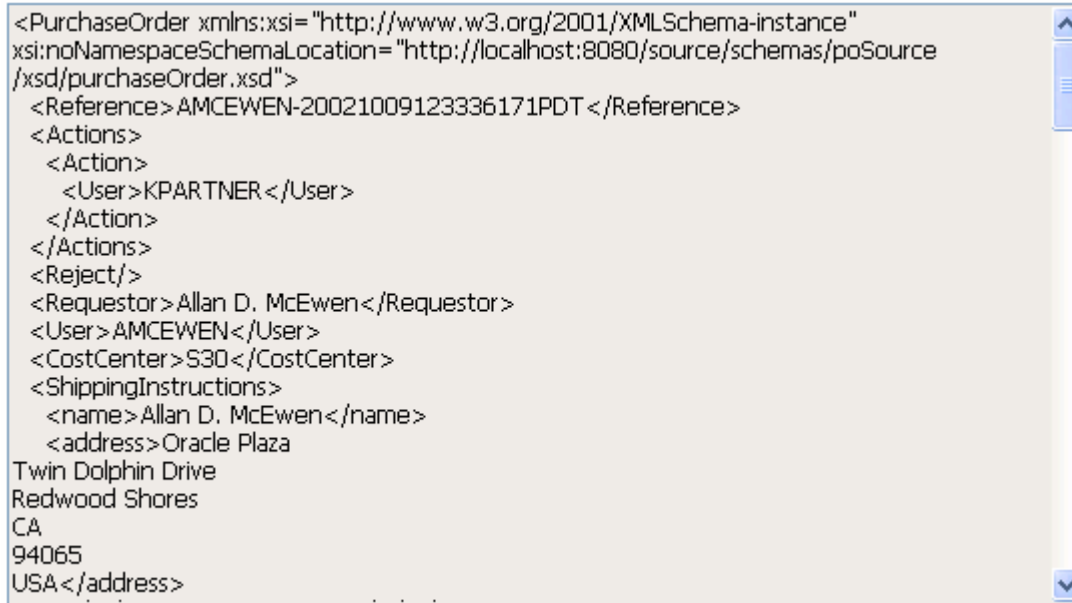
View XML Database Resource: AMCEWEN-20021009123336171PDT.xml

Display Contents Display Event Listeners Edit OK

General

Path Name	/home/OE/PurchaseOrders /2002/Apr/AMCEWEN- 20021009123336171PDT.xml
Display Name	AMCEWEN- 20021009123336171PDT.xml
Owner	OE
Creator	OE
Last Modifier	OE
Created	Aug 27, 2008 11:05:02 PM CDT
Last Modified	Aug 27, 2008 11:05:02 PM CDT
Type	XML Database Resource File
Content Type	text/xml
Character Set	US-ASCII
Language	en-US
Author	
Comment	

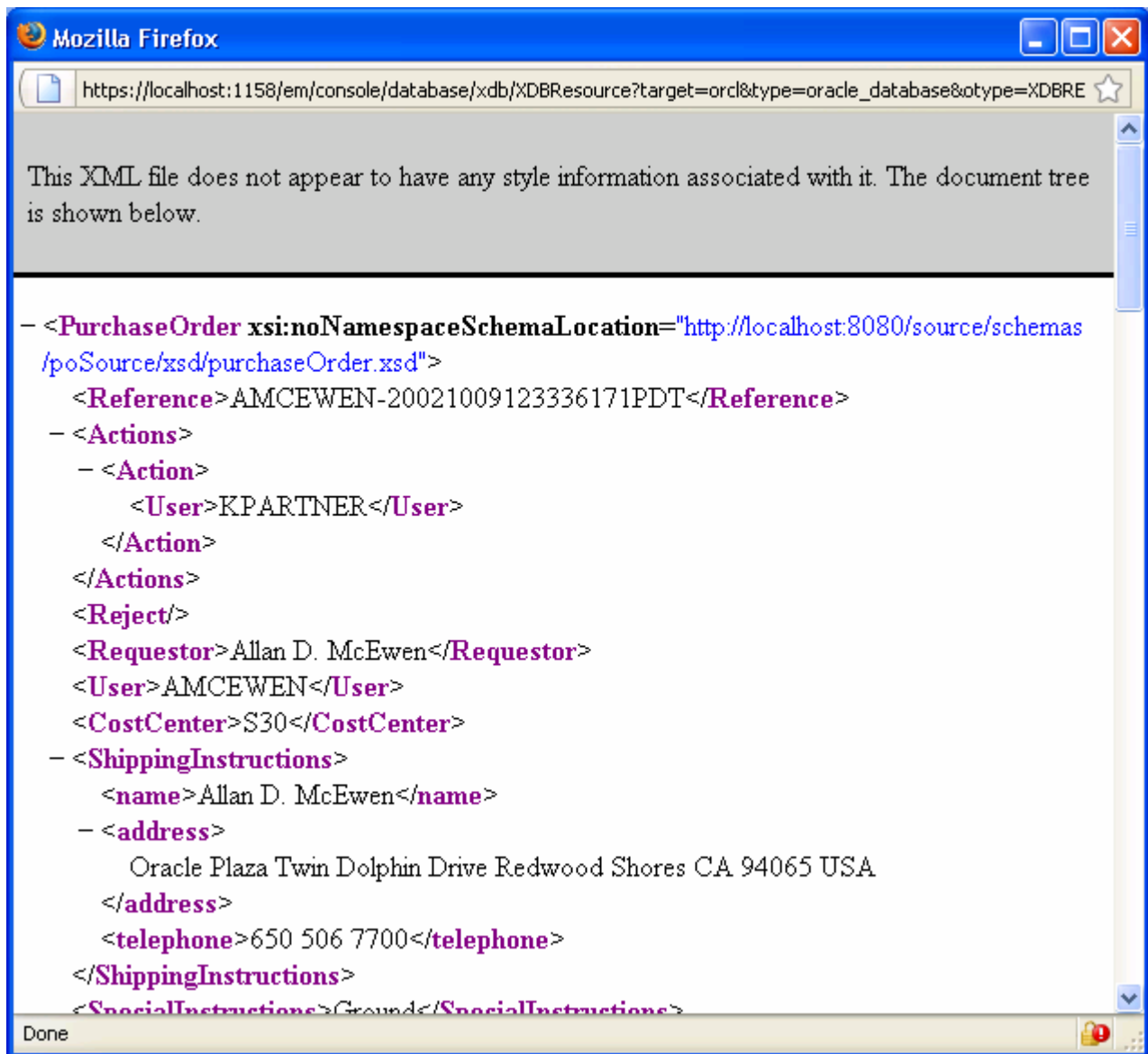
10. The file contents are shown. Click **Show formatted XML Content**.



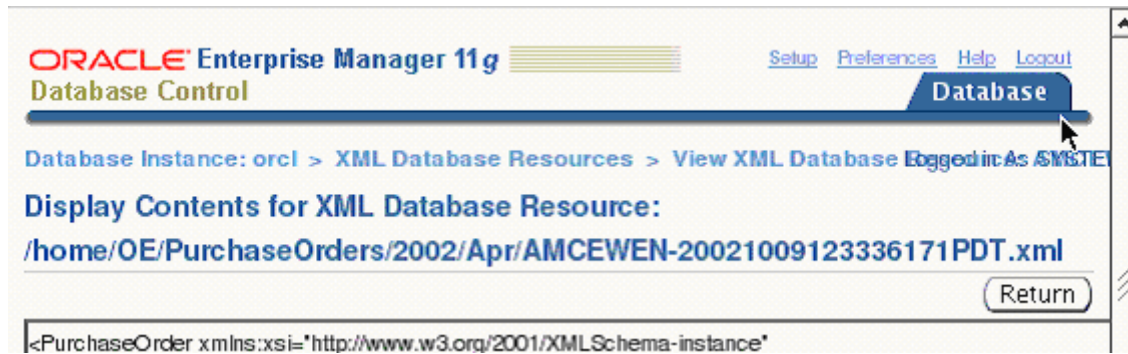
```
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://localhost:8080/source/schemas/poSource
/xsd/purchaseOrder.xsd">
  <Reference>AMCEWEN-20021009123336171PDT </Reference>
  <Actions>
    <Action>
      <User>KPARTNER </User>
    </Action>
  </Actions>
  <Reject/>
  <Requestor>Allan D. McEwen</Requestor>
  <User>AMCEWEN</User>
  <CostCenter>S30</CostCenter>
  <ShippingInstructions>
    <name>Allan D. McEwen</name>
    <address>Oracle Plaza
Twin Dolphin Drive
Redwood Shores
CA
94065
USA</address>
```

[Show Formatted XML Content](#)

11. Review the formatted XML document. When you are done, close the window.



12. Click on **Database**





13. Go to **Schema > XML Database**. Then, click **XMLType Tables**

Oracle Enterprise Manager 11g Database Control

Setup Preferences Help Logout

Database

Logged in As SYSTEM

Database Instance: orcl

Home Performance Availability Server Schema Data Movement Software and Support

Latest Data Collected From Target Feb 19, 2007 10:37:17 AM EST Refresh View Data

Automatically (60 sec)

Oracle Enterprise Manager 11g Database Control

Setup Preferences Help Logout

Database

Logged in As SYS

Database Instance: orcl

Home Performance Availability Server Schema Data Movement Software and Support

**Database Objects**

- Tables
- Indexes
- Views
- Synonyms
- Sequences
- Database Links
- Directory Objects
- Reorganize Objects

**Programs**

- Packages
- Package Bodies
- Procedures
- Functions
- Triggers
- Java Classes
- Java Sources

**Change Management**

- Dictionary Baselines
- Dictionary Comparisons

**Materialized Views**

- Materialized Views
- Materialized View Logs
- Refresh Groups
- Dimensions

**User Defined Types**

- Array Types
- Object Types
- Table Types

**XML Database**

- Configuration
- Resources
- Access Control Lists
- XML Schemas
- XMLType Tables
- XMLType Views
- XML Indexes
- XML Repository Events

**Workspace Manager**

- Workspaces

**Text Manager**

- Text Indexes

14. In the XMLType Tables search window, the Object Name displays SYSTEM. Delete **SYSTEM**, and click **Go**.

ORACLE Enterprise Manager 11g Database Control

Setup Preferences Help Logout

Database

Database Instance: orcl > Logged in As SYSTEM

### XMLType Tables

**Search**

Schema

Object Name

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Select	Owner	Table Name	XML Schema	Element Name	Storage Type
	No search conducted				

- Click the table name **PURCHASEORDER**.

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive search, use the exact match or case-sensitive search buttons. You can also use the wildcard symbol (%) in a double quoted string.

Selection Mode Single

Edit View Delete Actions Create Like Go

Select	Owner	Table Name	XML Schema
	OE	<a href="#">PURCHASEORDER</a>	http://localhost:8080/source/schemas/poSource/xsd/purchaseorder.xsd
	XDB	<a href="#">Folder42_TAB</a>	http://xmlns.oracle.com/xdb/XDBFolderListing.xsd
	XDB	<a href="#">SERVLET</a>	http://xmlns.oracle.com/xdb/XDBStandard.xsd
	XDB	<a href="#">XDB\$ACL</a>	http://xmlns.oracle.com/xdb/acl.xsd
	XDB	<a href="#">XDB\$ALL_MODEL</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$ANY</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$ANYATTR</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$ATTRGROUP_DEF</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$ATTRGROUP_REF</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$ATTRIBUTE</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$CHOICE_MODEL</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd
	XDB	<a href="#">XDB\$COMPLEX_TYPE</a>	http://xmlns.oracle.com/xdb/XDBSchema.xsd

16. The table definition is displayed. Scroll down to see more information. Then, click **OK**.

**ORACLE® Enterprise Manager 11g**
Database Control

[Setup](#)
[Preferences](#)
[Help](#)
[Logout](#)

Database

Database Instance: orcl > XMLType Tables >

View Table: OE.PURCHASEORDER

Logged in As SYSTEM

Actions
Create Like
Go
Edit
OK

General

Name PURCHASEORDER  
Schema OE  
Tablespace USERS  
Organization Standard (Heap Organized)

XML Type

Type XML Schema Based  
Storage Type OBJECT-RELATIONAL

Constraints

Previous 1-3 of 3 Next

Name	Type	Table Columns	Disabled	Deferrable	Initially Deferred	Validate	RELY	Check Condition	Referenced Schema	Referenced Table	Referenced Table Columns	Cascade on Delete
SYS_C009610	UNIQUE	SYS_NC0003400035\$	NO	NO	NO	YES	NO					NO
SYS_C009611	UNIQUE	SYS_NC0001300014\$	NO	NO	NO	YES	NO					NO
SYS_C009612	UNIQUE	SYS_NC_OID\$	NO	NO	NO	YES	NO					NO

Indexes

Number of Indexes 3

Schema.Index	Indexed Columns	Column Position
OE.ACTION TABLE MEMBERS	"XMLDATA"."ACTIONS"."ACTION"	1
OE.LINEITEM TABLE MEMBERS	"XMLDATA"."LINEITEMS"."LINEITEM"	1
OE.SYS_C009612	SYS_NC_OID\$	1

Storage

Tablespace

Name USERS  
Extent Management Local  
Segment Management Automatic  
Allocation Type SYSTEM  
Logging Yes

Space Usage

Free Space (PCTFREE)(%) 10

Number of Transactions

Initial 1  
Maximum 255

Buffer Pool

Buffer Pool DEFAULT

17. To log out of Oracle Enterprise Manager 11g, click **Logout**.

The screenshot shows the Oracle Enterprise Manager 11g Database Control interface. At the top, the title bar reads "ORACLE Enterprise Manager 11g Database Control". On the right, there are links for "Setup", "Preferences", "Help", and "Logout". Below the title bar, the "Database Instance: orcl" is displayed, and the user is logged in as "SYSTEM". The main section is titled "XMLType Tables". Under the "Search" heading, there are input fields for "Schema" and "Object Name", a "Go" button, and a search icon. A note states: "By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive wildcard symbol (%) in a double quoted string." Below this, the "Selection Mode" is set to "Single". At the bottom, there are buttons for "Edit", "View", "Delete", "Actions", and "Create Like", along with another "Go" button. A table displays the search results with columns: "Select", "Owner", "Table Name", and "XML Schema".

Select	Owner	Table Name	XML Schema
	OE	<a href="#">PURCHASEORDER</a>	http://localhost:8080/source/schemas/poSource/xsd/purchaseorder.xsd
	XDB	<a href="#">Folder42_TAB</a>	http://xmlns.oracle.com/xdb/XDBFolderListing.xsd
	XDB	<a href="#">SERVLET</a>	http://xmlns.oracle.com/xdb/XDBStandard.xsd

## Reviewing the XML Schema in JDeveloper

You need to perform the following tasks:

-  [Create a WebDAV Connection](#)
-  [Review the XML Schema](#)

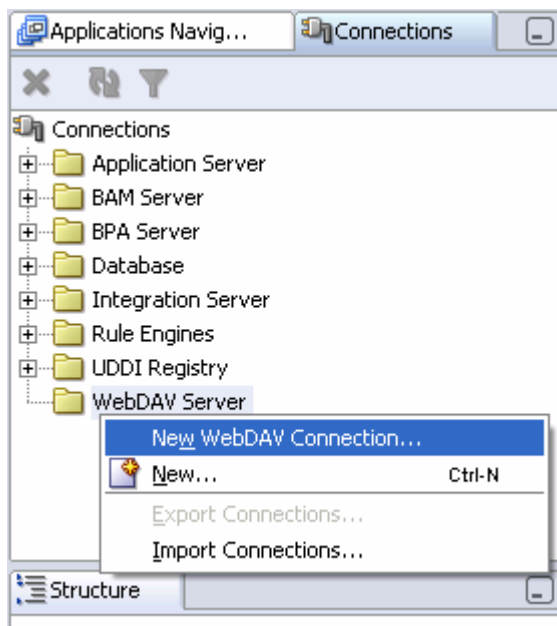
### Create a WebDAV Connection

Before you access the XML Schema documents in XML DB, you need to create a WebDAV connection. Perform the following steps:

1. Click on the **JDeveloper** icon on the desktop to start the application

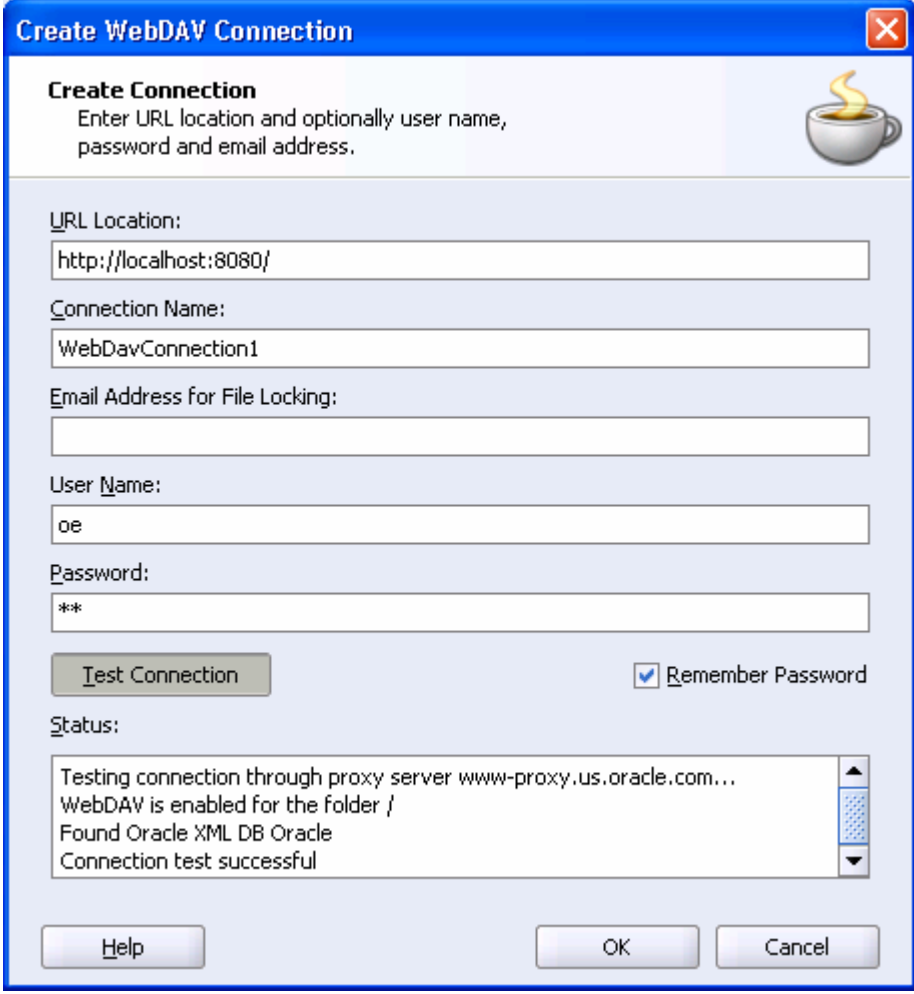


2. Click the **Connections Navigator** tab. In the Connections window, right-click **WebDAV Server**, and select **New WebDAV Connection...**



3. The Create WebDAV Connection window appears. Enter the following details, then click **Test Connection**.

URL location: **localhost:8080/**  
Connection Name: **WebDavConnection1**  
User Name: **oe**  
Password: **oe**

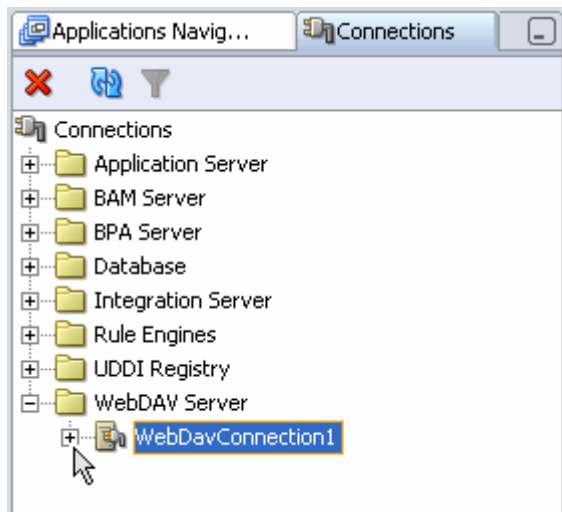


The image shows a Windows-style dialog box titled "Create WebDAV Connection". It has a blue title bar with a close button (X) in the top right corner. The main area is light blue and contains the following elements:

- Create Connection** section with a subtitle: "Enter URL location and optionally user name, password and email address." and a small icon of a steaming cup of coffee.
- URL Location:** A text input field containing "http://localhost:8080/".
- Connection Name:** A text input field containing "WebDavConnection1".
- Email Address for File Locking:** An empty text input field.
- User Name:** A text input field containing "oe".
- Password:** A text input field containing "\*\*" (masked characters).
- Test Connection** button and a checked checkbox labeled **Remember Password**.
- Status:** A text area displaying the following text:  
Testing connection through proxy server www-proxy.us.oracle.com...  
WebDAV is enabled for the folder /  
Found Oracle XML DB Oracle  
Connection test successful
- At the bottom are three buttons: **Help**, **OK**, and **Cancel**.

4. The test connection status shows successful. Click **OK**.

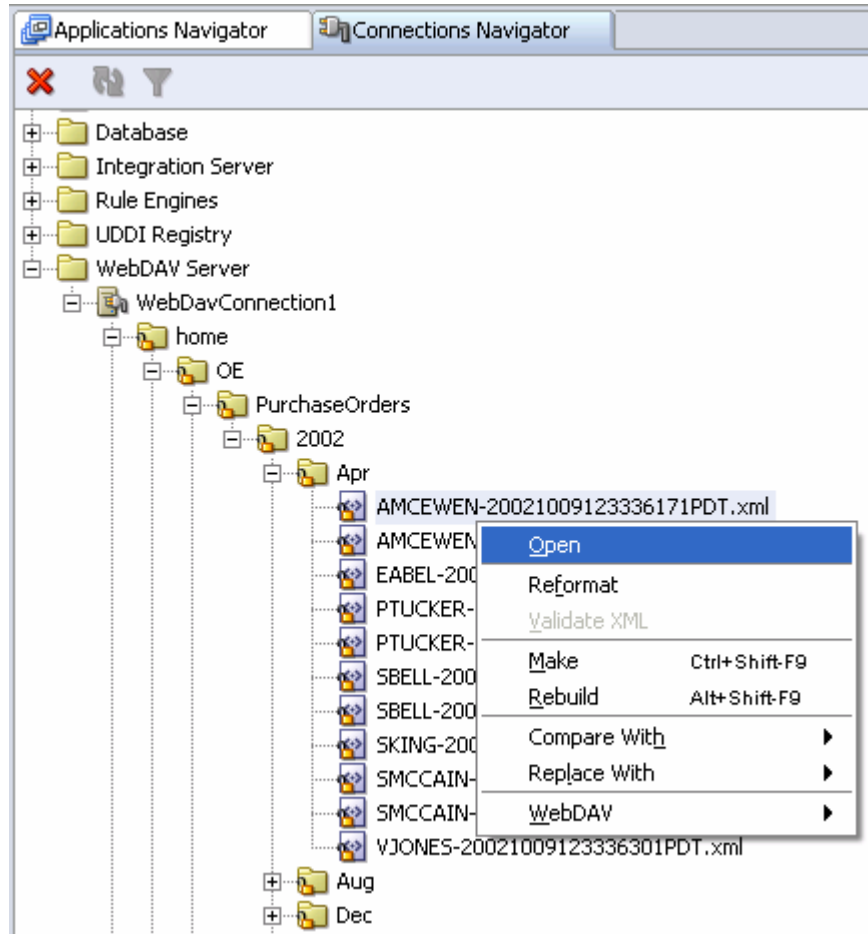
5. Expand **WebDavConnection1** connection that you just created.





Review the XML Schema Now you can review the XML Schema . Perform the following steps:

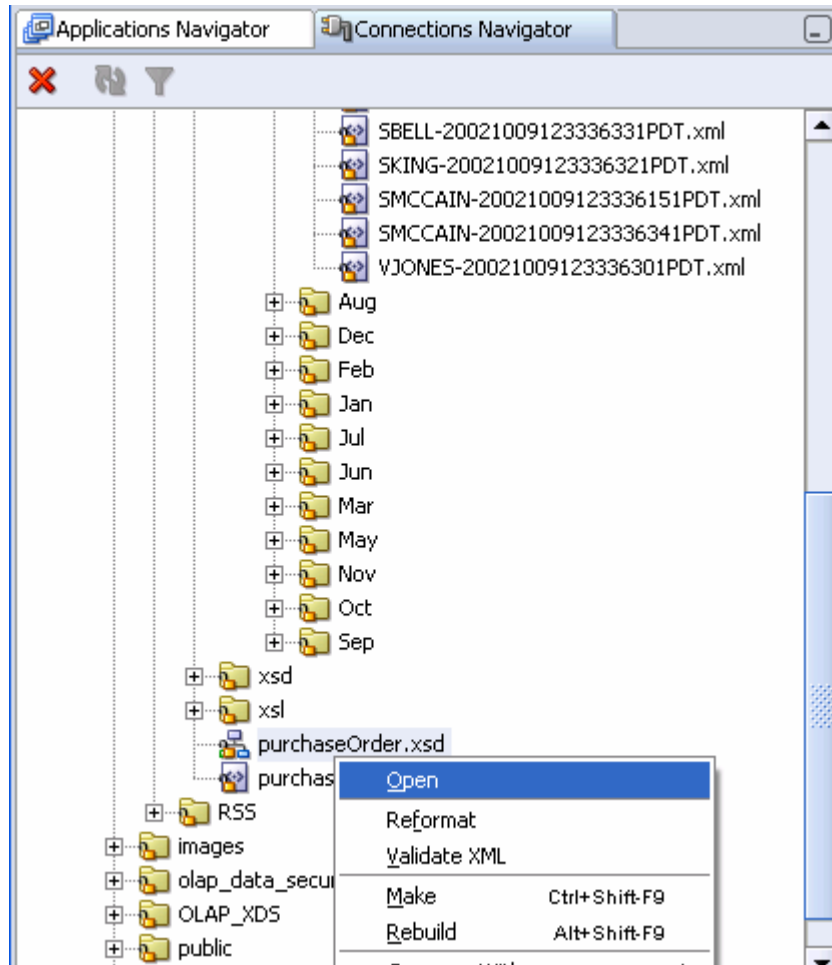
1. Expand **home - OE - PurchaseOrders - 2002 - Apr**. In the list, right click the first document ,and select **Open**. This opens the XML document.



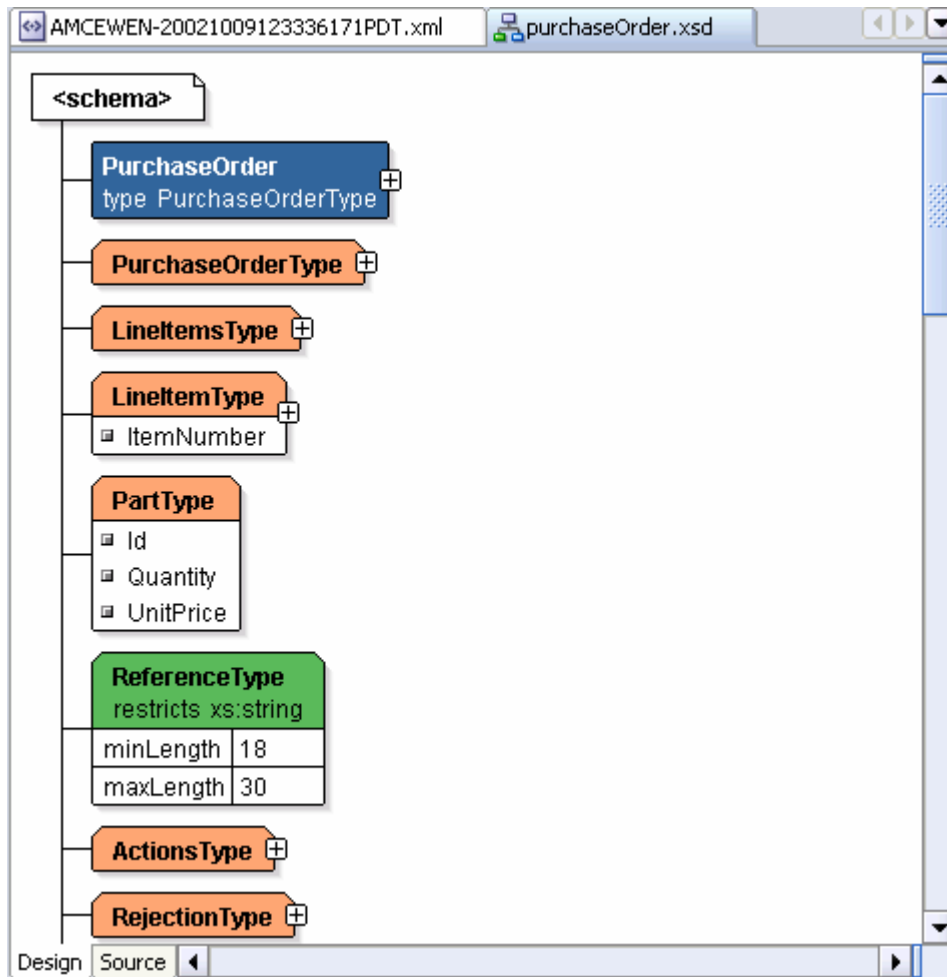
```
<?xml version="1.0" encoding="UTF-8" ?>
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="PurchaseOrder.xsd">
  <Reference>AMCEWEN-20021009123336171PDT</Reference>
  <Actions>
    <Action>
      <User>KPARTNER</User>
    </Action>
  </Actions>
  <Reject/>
  <Requestor>Allan D. McEwen</Requestor>
  <User>AMCEWEN</User>
  <CostCenter>S30</CostCenter>
  <ShippingInstructions>
    <name>Allan D. McEwen</name>
    <address>Oracle Plaza
Twin Dolphin Drive
Redwood Shores
CA
94065
USA</address>
    <telephone>650 506 7700</telephone>
  </ShippingInstructions>
  <SpecialInstructions>Ground</SpecialInstructions>
  <LineItems>
    <LineItem ItemNumber="1">
      <Description>Salesman</Description>
      <Part Id="37429158920" UnitPrice="39.95" Quantity="2"/>
    </LineItem>
  </LineItems>
</PurchaseOrder>
```

Source

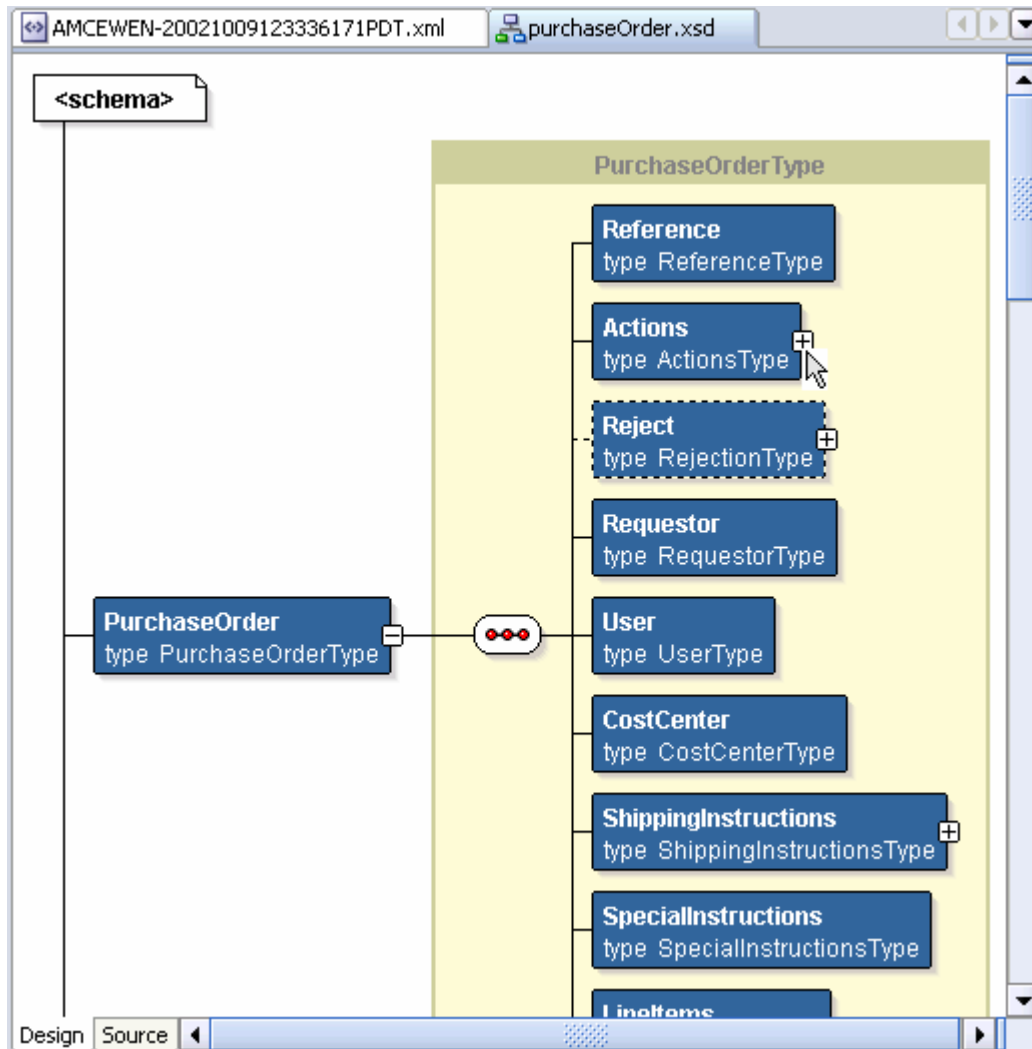
2. Scroll down in the Connections window, and right-click **purchaseOrder.xsd**. Then, select **Open**.



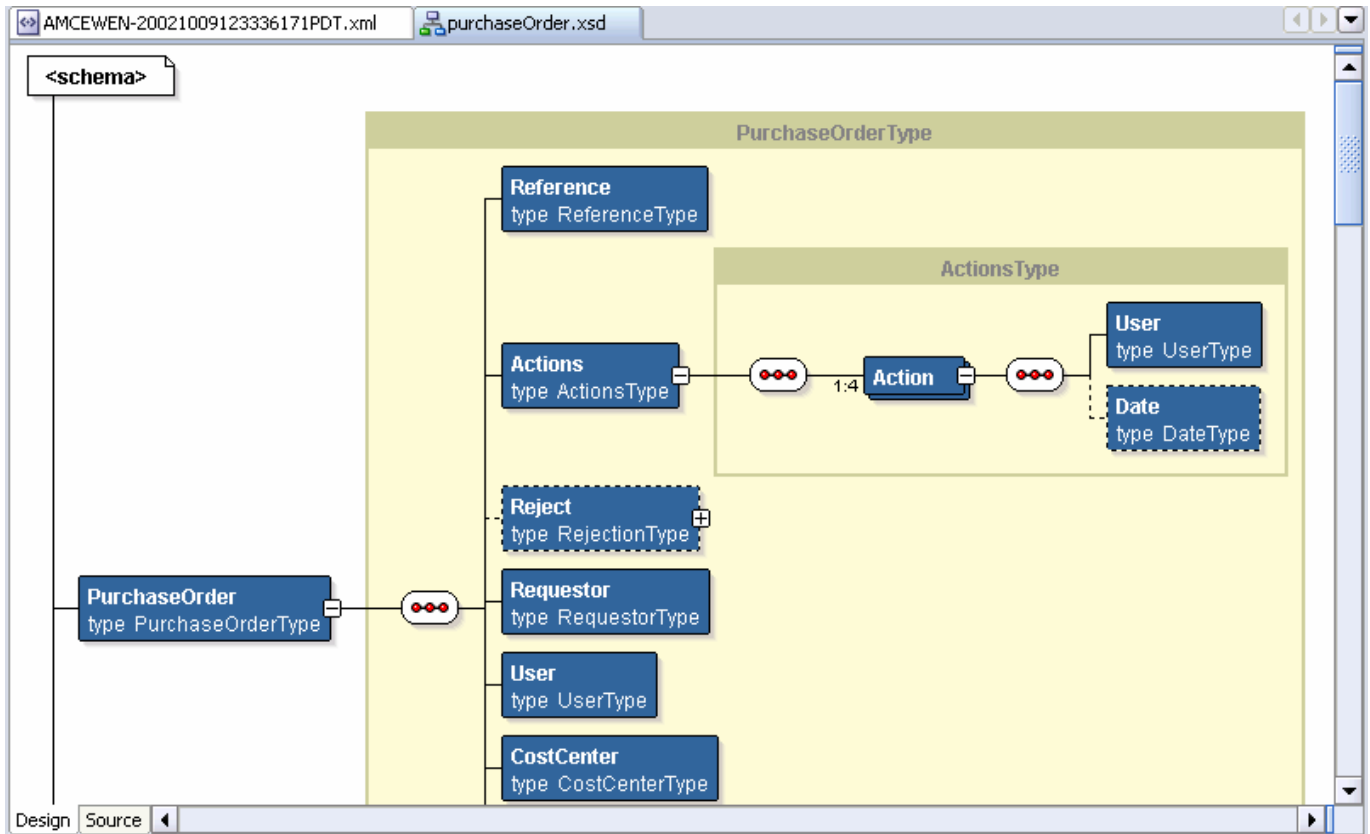
3. The XML Schema is shown in a graphical layout. Expand **PurchaseOrder**.



4. Expand **Actions**.



5. Expand **Action** and you see that there are two type objects, User and Date.



6. Now, go back to the XML Document to see the definition. Click the **XML Document** tab. Note that the Action object contains a user type of KPARTNER.

```
<PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="purchaseOrder.xsd">
  <Reference>AMCEWEN-20021009123336171PDT</Reference>
  <Actions>
    <Action>
      <User>KPARTNER</User>
    </Action>
  </Actions>
  <Reject/>
  <Requestor>Allan D. McEwen</Requestor>
  <User>AMCEWEN</User>
  <CostCenter>$30</CostCenter>
  <ShippingInstructions>
    <name>Allan D. McEwen</name>
    <address>Oracle Plaza
Twin Dolphin Drive
Redwood Shores
CA
94065
USA</address>
    <telephone>650 506 7700</telephone>
  </ShippingInstructions>
  <SpecialInstructions>Ground</SpecialInstructions>
  <LineItems>
    <LineItem ItemNumber="1">
      <Description>Salesman</Description>
      <Part Id="37429158920" UnitPrice="39.95" Quantity="2"/>
    </LineItem>
  </LineItems>
</PurchaseOrder>
```

## Creating a Binary XML Table

In this section, you create a binary XML table, and insert rows with data selected from the `PURCHASEORDER` table. You use Oracle SQL Developer throughout this tutorial. Perform the following steps:

[Start SQL Developer](#)

[Create Binary XML Table](#)

### Start SQL Developer

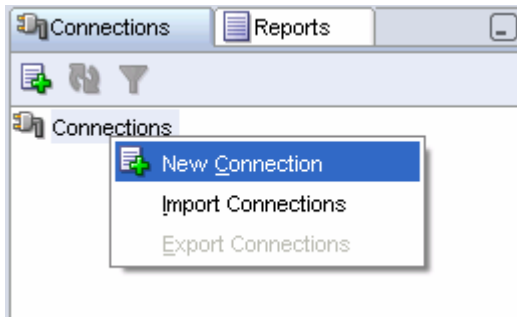
Perform the following steps:

1. Click on the SQL Developer icon on the desktop to start the application.



2. You must create a database connection as **system** user. Perform the following steps.

- a. In the Connections tab, right-click **Connections** and select **New Connection**.



- b. The New/Select Database Connection window appears. Enter the following details, and click **Test** to make sure that the connection has been set correctly.

Connection Name: **system**

UserName: **system**

Password: **oracle**

Hostname: **localhost**

Port: **1521**

SID: **orcl**



**New / Select Database Connection**

Connection Name	Connection Details
system	<p>Username: system</p> <p>Password: *****</p> <p><input checked="" type="checkbox"/> Save Password</p> <p>Oracle Access</p> <p>Role: default <input type="checkbox"/> OS Authentication</p> <p>Connection Type: Basic <input type="checkbox"/> Proxy Connection</p> <p>Hostname: localhost</p> <p>Port: 1521</p> <p><input checked="" type="radio"/> SID <input type="radio"/> Service name</p>

Status: Success

Help Save Clear Test Connect Cancel

c. The test status shows success. Click **Connect**.

d. Click on the Files tab and then open the **accounts.sql** file from the **C:\HOL\xmlldb\_1\files\** directory and click on the **Run Script** button to grant privileges to user **oe**.

**Oracle SQL Developer : C:\HOL\xmlldb\_1\files\accounts.sql**

File Edit View Navigate Run Source Versioning Migration Tools Help

Connections system accounts.sql

SQL Worksheet History

0.51018447 seconds system

Enter SQL Run Script (F5)

```

ALTER USER hr IDENTIFIED BY hr ACCOUNT UNLOCK;
ALTER USER oe IDENTIFIED BY oe ACCOUNT UNLOCK;

GRANT SELECT_CATALOG_ROLE TO oe;
GRANT SELECT ANY DICTIONARY TO oe;

```

Results Script Output Explain Autotrace DBMS Output OWA Output

```

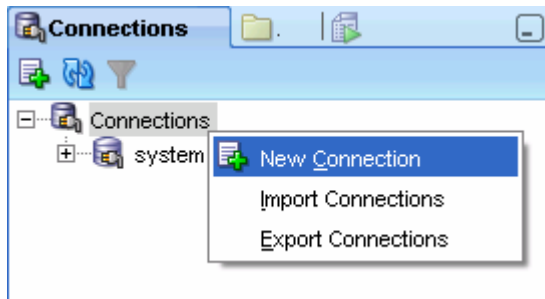
ALTER USER hr succeeded.
ALTER USER oe succeeded.
GRANT SELECT_CATALOG_ROLE succeeded.
GRANT SELECT succeeded.

```

SQL History

Script Finished | Line 1 Column 1 | Insert | Windows: CR/... Editing

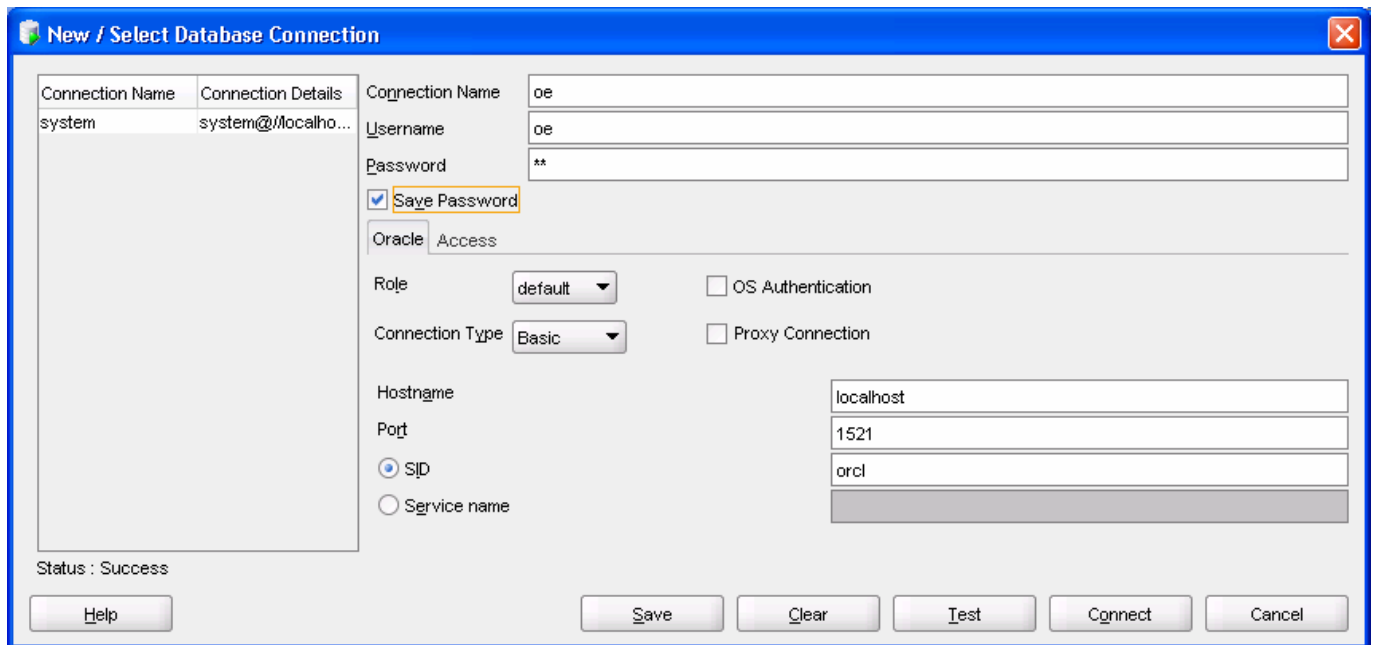
e. In the Connections tab, right-click **Connections** and select **New Connection**.



f. The New/Select Database Connection window appears. Enter the following details, and click **Test** to make sure that the connection has been set correctly.

Connection Name: **oe**  
UserName: **oe**  
Password: **oe**  
Hostname: **localhost**  
Port: **1521**  
SID: **orcl**

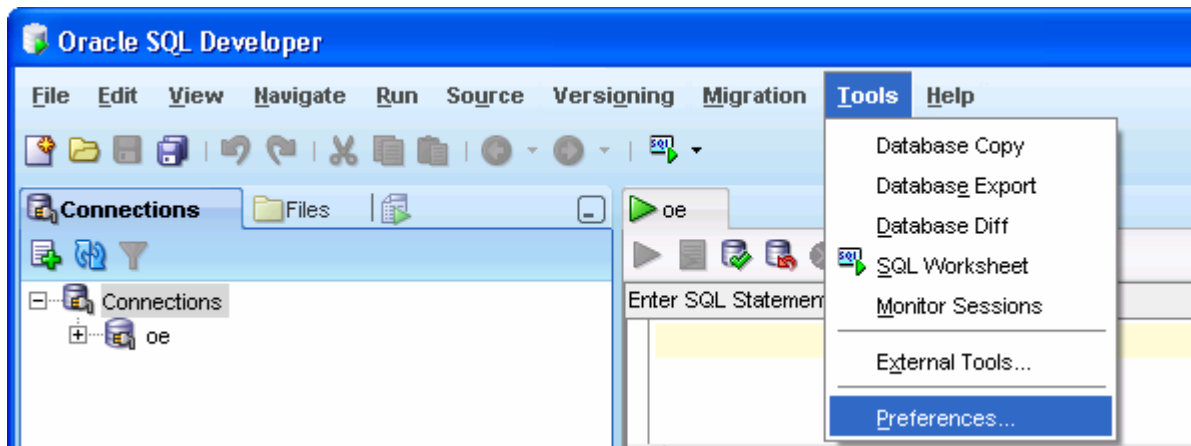
If you select the Save Password check box, the password is saved to an XML file. Therefore, once you close SQL Developer connection and open again, you will not be prompted for the password.



g. The test status shows success. Click **Connect**.

3. Set the Autotrace parameters. Perform the following steps:

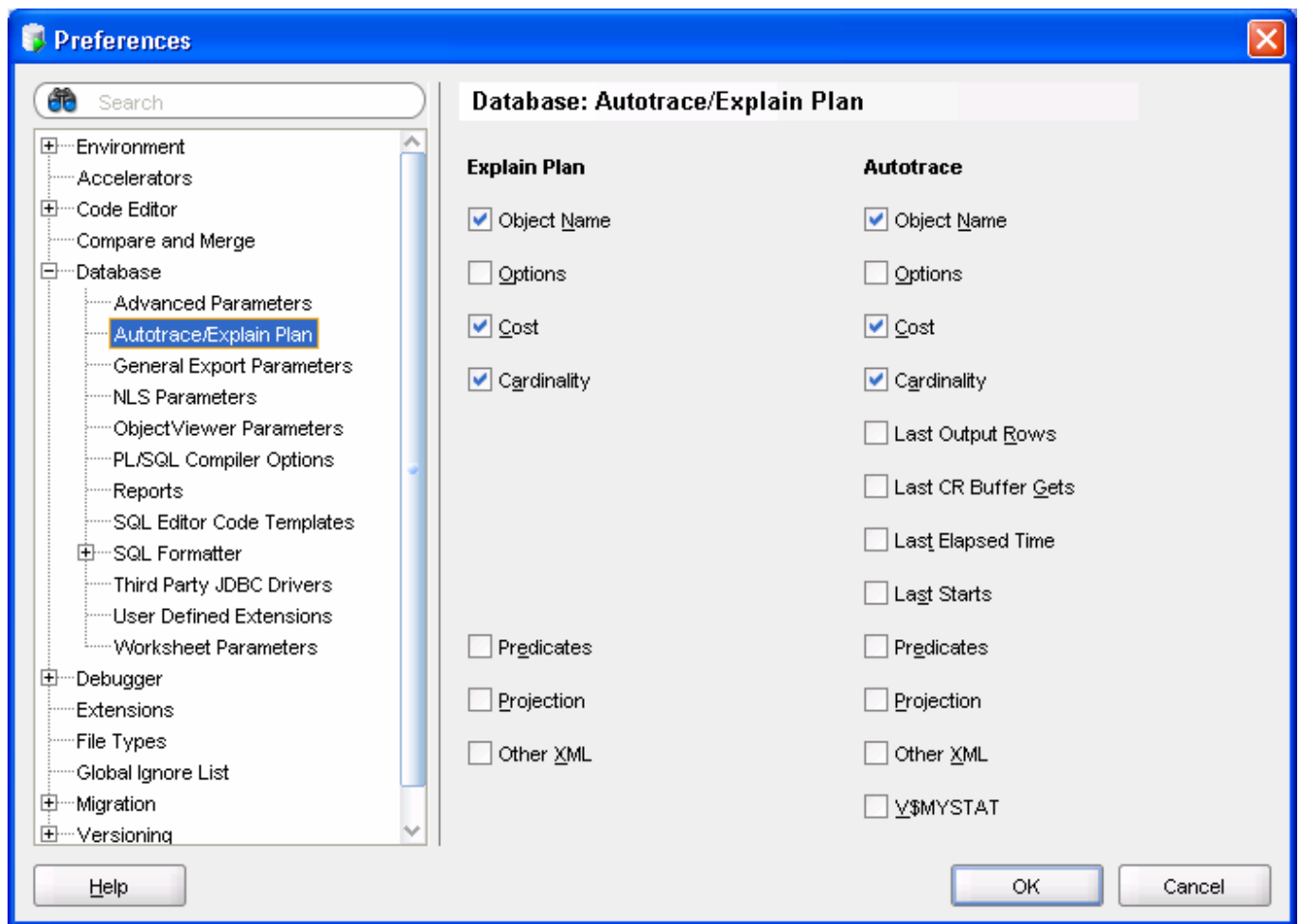
a. Go to **Tools > Preferences**.



b. Expand **Database**, and select **Autotrace Parameters**.

c. Make sure to select the following check boxes and click **OK**.

Object\_Name  
Cost  
Cardinality

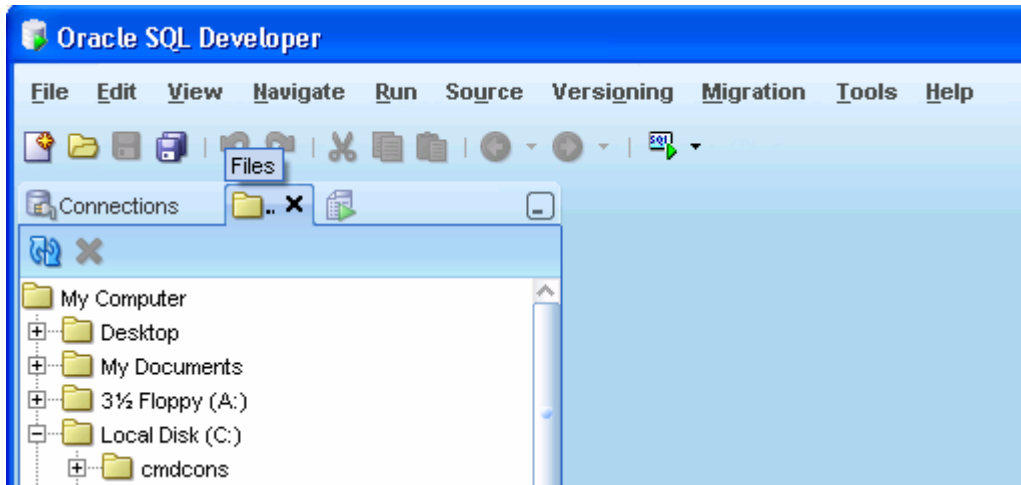


In the above section, you learned how to connect to SQL Developer, and set Autotrace parameters.

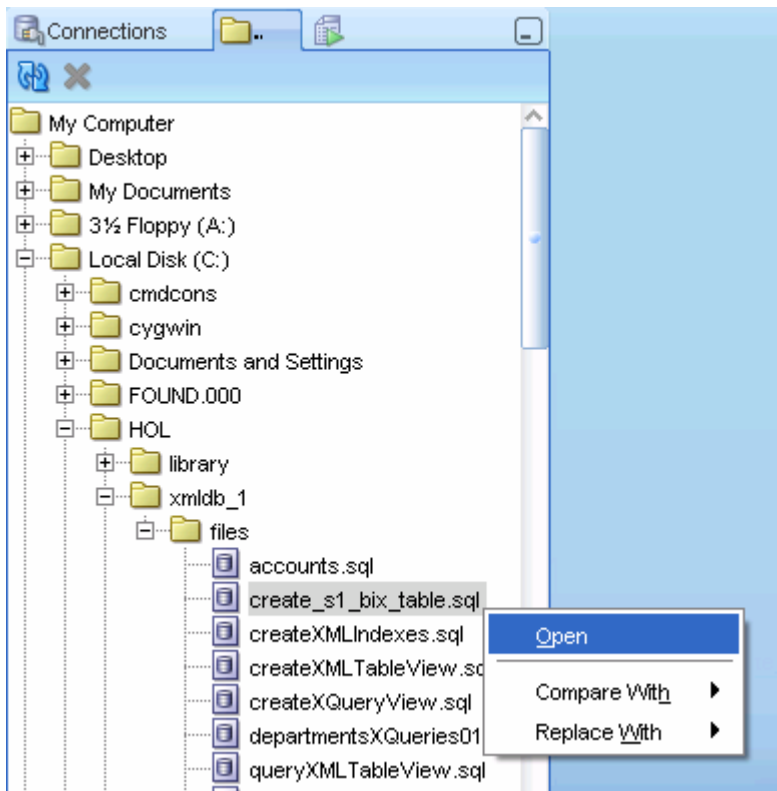
## Create Binary XML Table

1. Create an `XMLType` table with binary XML storage, and populate the table with data selected from the `PURCHASEORDERS` table. Run the script `create_s1_bix_table.sql`. Perform the following steps:

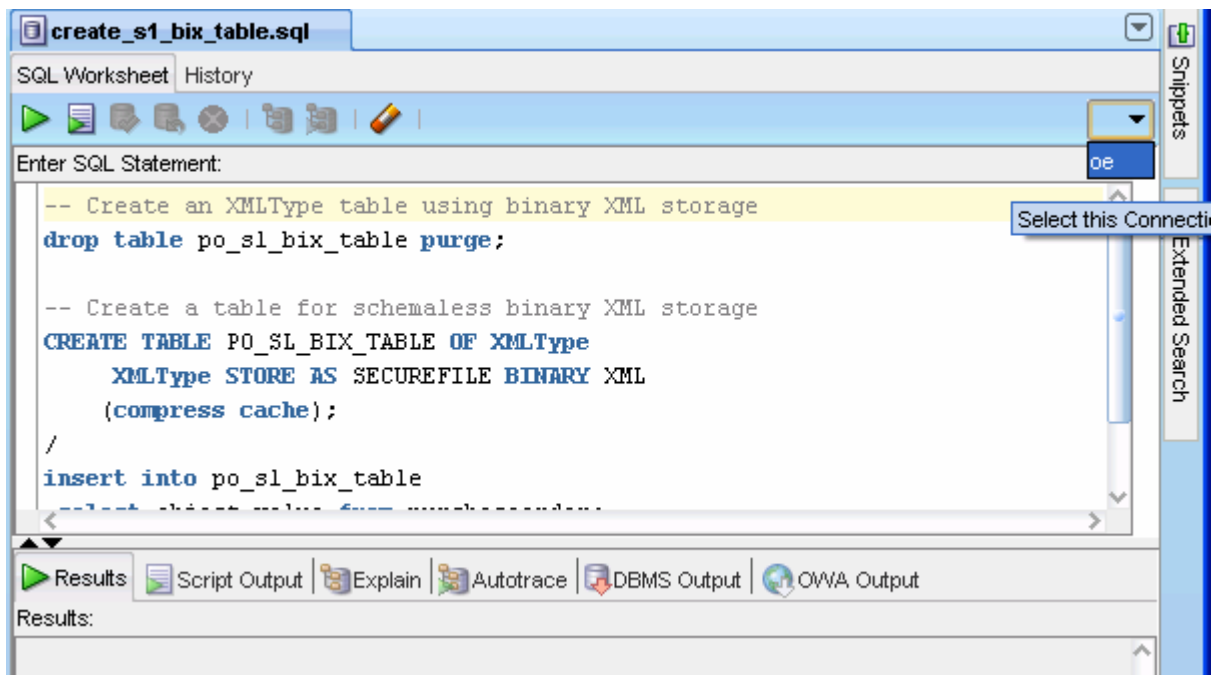
a. Click in the **Files** tab,.



b. Browse to the location of your working directory, and select the file `create_s1_bix_table.sql`. Then, right-click **Open**.



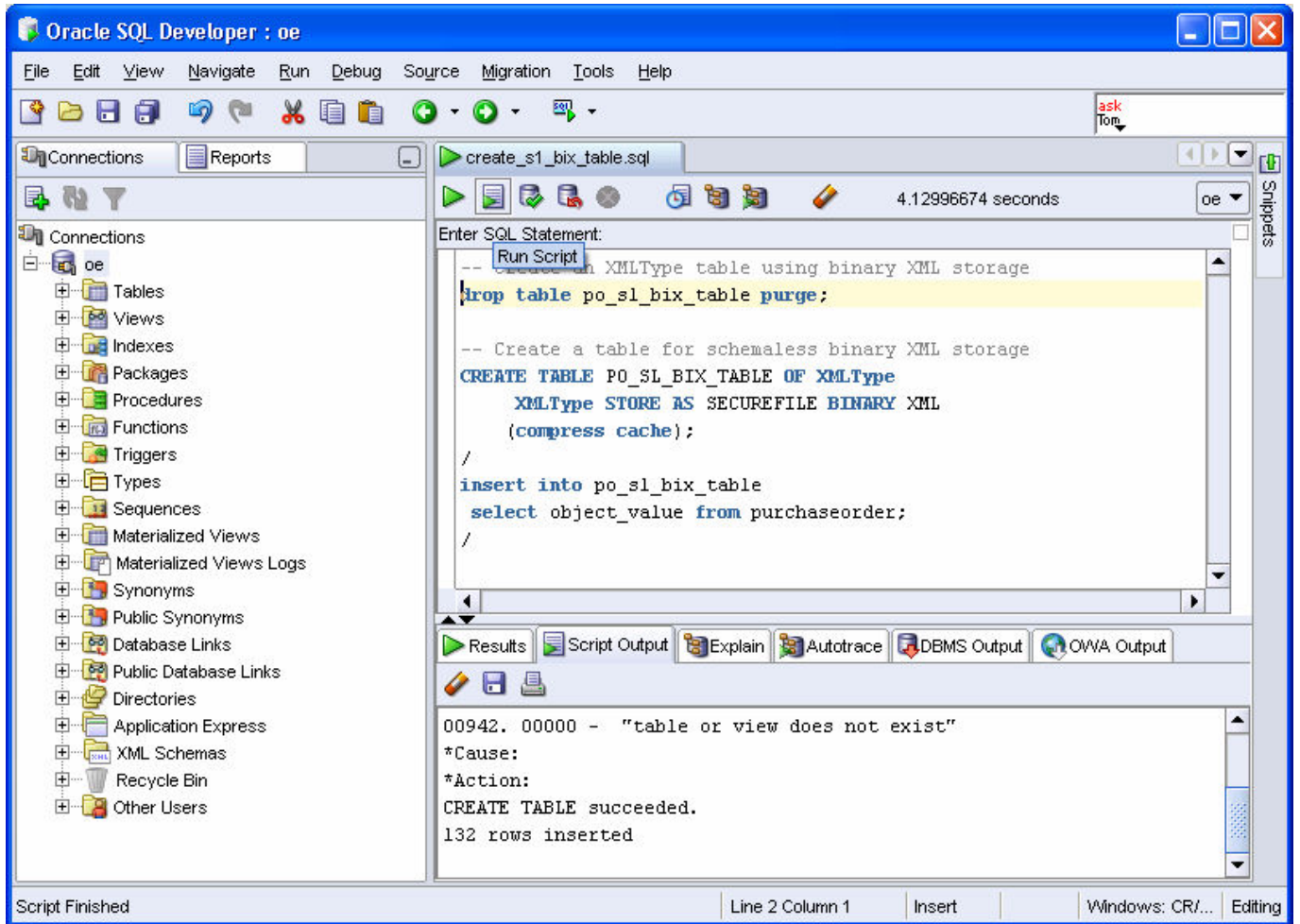
c. Select the **oe** connection from the drop down box on the right.



d. The code is displayed in the SQL Worksheet. Click the **Run Script** icon. Alternatively, you can press **F5**. Note the results that are displayed under the Script Output pane.

**create\_sl\_bix\_table.sql**

```
DROP table po_sl_bix_table purge;
/
CREATE TABLE PO_SL_BIX_TABLE OF XMLType
XMLType STORE AS SECUREFILE BINARY XML
(compress cache);
/
INSERT INTO po_sl_bix_table
SELECT object_value from purchaseorder;
/
```



Note: Throughout this tutorial, to execute the script files in SQL Developer, you must follow the above steps a through c. If you want to run a single statement at the mouse pointer, click the **Execute Statement** icon. Alternatively, move the cursor over the statement, and press **F9**.

In the above topic, you learned how to create an `XMLType` table with binary XML storage. You also learned how to populate data into this table.

## Improving Performance of XQuery Expressions through Index Creation

You can increase the performance of your XQuery by creating an index. In this section, you will create B-Tree indexes on object relational storage and `XMLIndex` index on binary storage table. You will then run the SQL/XML, XQuery expressions against both object-relational and binary XML tables to see the explain plan and note that the performance has improved. Perform the following steps:

1. In your SQLDeveloper session, connect as OE user. Then, execute the script **createXMLIndexes.sql**.

```
set echo on
-- B-tree indexes for O-R storage
drop index iPurchaseOrderUser;
drop index iPurchaseOrderRef;
drop index iLineItemPartNumber;
drop index iPartNumber;
drop index iDESCRIPTION_FULL_TEXT;

-- XMLIndex indexes for binary XML storage
drop index po_sl_xmlindex_bix_ix force;
drop index po_sl_bix_text_ix;

create index iPurchaseOrderUser on PurchaseOrder
(extractValue(object_value, '/PurchaseOrder/User'))
/
create index iPurchaseOrderRef on PurchaseOrder
(extractValue(object_value, '/PurchaseOrder/Reference'))
/

create index iLineItemPartNumber on LINEITEM_TABLE
(ITEMNUMBER, PART.PART_NUMBER, NESTED_TABLE_ID) compute statistics
/
create index iPartNumber on LINEITEM_TABLE
(PART.PART_NUMBER, NESTED_TABLE_ID) compute statistics
/

-- Create XML index on the binary XML table
create index po_sl_xmlindex_bix_ix on po_sl_bix_table(object_value) indextype is xdb.xmlindex
parameters ('PATH TABLE po_sl_bix_path_table
PATH ID INDEX po_sl_bix_path_id_ix
ORDER KEY INDEX po_sl_bix_order_key_ix
VALUE INDEX po_sl_bix_value_ix')
/

-- Create a secondary text index on the VALUE column of the path table
CREATE INDEX po_sl_bix_text_ix ON po_sl_bix_path_table (VALUE) INDEXTYPE IS CTXSYS.CONTEXT
parameters ('transactional')
/

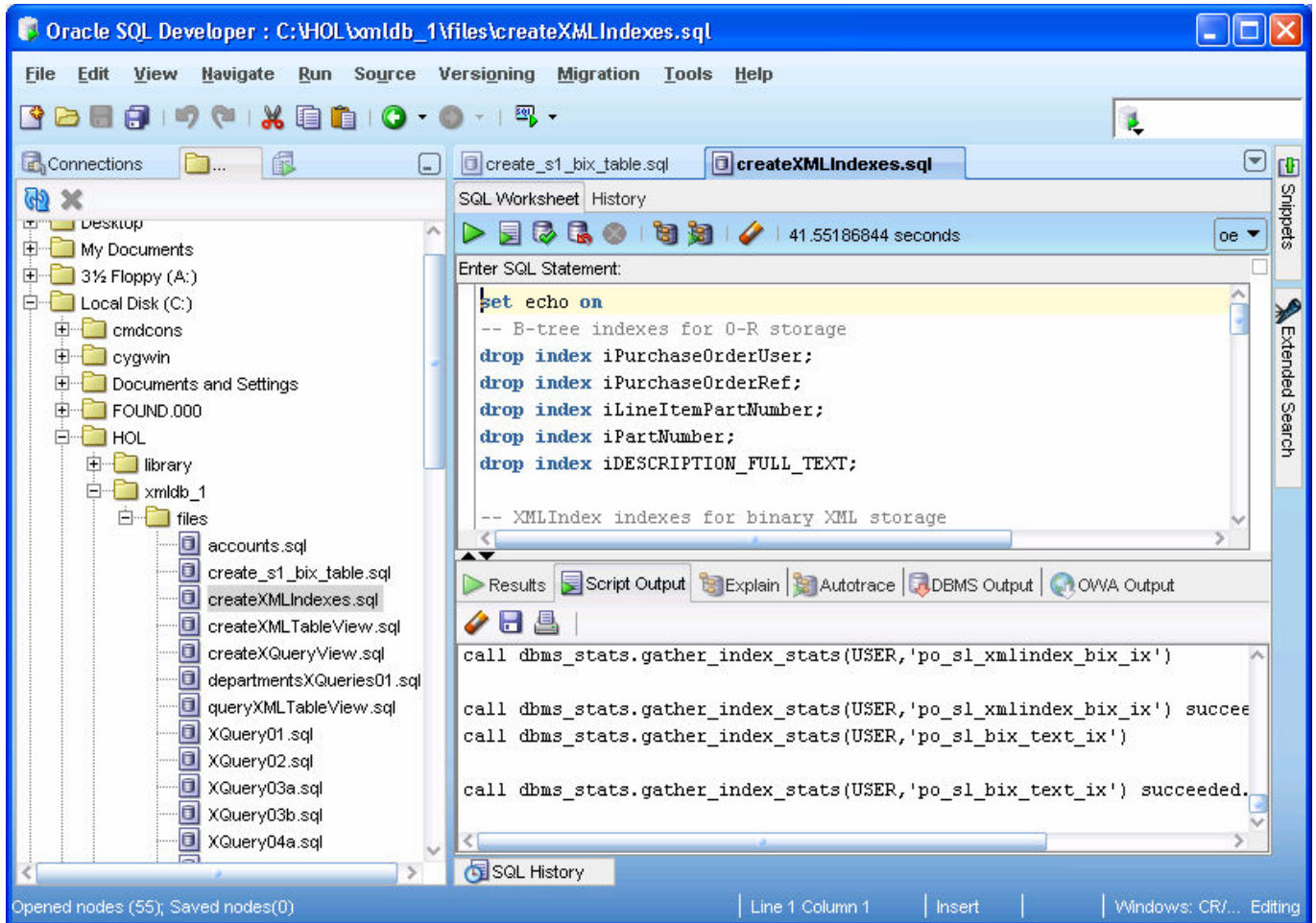
call dbms_stats.gather_table_stats(USER, 'PURCHASEORDER')
/
call dbms_stats.gather_table_stats(USER, 'LINEITEM_TABLE')
/
call dbms_stats.gather_table_stats(USER, 'po_sl_bix_table')
/
call dbms_stats.gather_index_stats(USER, 'iPurchaseOrderUser')
/
call dbms_stats.gather_index_stats(USER, 'iPurchaseOrderRef')
/
```



```

call dbms_stats.gather_index_stats(USER,'iLineItemPartNumber')
/
call dbms_stats.gather_index_stats(USER,'iPartNumber')
/
call dbms_stats.gather_index_stats(USER,'po_sl_xmlindex_bix_ix')
/
call dbms_stats.gather_index_stats(USER,'po_sl_bix_text_ix')
/

```



2. Now that you created indexes, you view the explain plan to observe the performance of SQL/XML, XQuery expressions. Observe that the explain plan picks up the applicable indexes.

First, view the execution plan of the query that reviews a specific purchase order. In the **Enter SQL Statement** box, perform the following steps:

- a. Open the file **XQuery03a.sql**. The code is displayed in the **Enter SQL Statement** box. Now, click the **Autotrace** icon. Note the usage of the index **IPURCHASEORDERREF**.

Code in XQuery03a.sql:

```
--Xquery: Review a specific purchase order
```

```
SELECT XMLQuery('/PurchaseOrder/ShippingInstructions/name' passing object_value returning
content)
FROM PURCHASEORDER
WHERE XMLEXISTS('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing object_value)
/
```

The screenshot shows the Oracle SQL Developer interface. The left pane displays the file explorer with the 'files' folder expanded, showing various SQL files including 'XQuery03a.sql'. The main window is titled 'XQuery03a.sql' and contains the SQL statement: `--Xquery: Review a specific purchase order` followed by the query. The 'Autotrace' icon is highlighted. Below the SQL statement, the 'Results' tab is active, displaying the execution plan. The plan shows the following operations:

OPERATION	OBJECT_NAME	COS
SELECT STATEMENT		
TABLE ACCESS BY INDEX ROWID	PURCHASEORDER	
INDEX RANGE SCAN	IPURCHASEORDERREF	

- b. Open the file **XQuery03b.sql**, and click the **Autotrace** icon. Note the usage of **XMLIndex** index.

Code in XQuery03b.sql:

```
-- Same query on a binary storage table
SELECT XMLQuery('/PurchaseOrder/ShippingInstructions/name' passing object_value returning
content).getStringVal()
FROM PO_SL_BIX_TABLE
WHERE XMLEXISTS('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing object_value)
```

Oracle SQL Developer : C:\HOL\bmldb\_1\files\XQuery03b.sql

File Edit View Navigate Run Source Versing Migration Tools Help

Connections

SQL Worksheet History

2.0697732 seconds

Enter SQL Statement:

Autotrace (F10)

```
-- Same query on a binary storage table
SELECT XMLQuery('/PurchaseOrder/ShippingInstructions/name' passing object_value re
FROM PO_SL_BIX_TABLE
WHERE XMLEXISTS('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing ob
/
```

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION	OBJECT_NAME
SELECT STATEMENT	
SORT GROUP BY	
TABLE ACCESS BY INDEX ROWID	PO_SL_BIX_PATH_TABLE
INDEX RANGE SCAN	PO_SL_BIX_PATH_ID_IX
NESTED LOOPS	
VIEW	VW_SQ_1
HASH UNIQUE	
NESTED LOOPS	
NESTED LOOPS	
TABLE ACCESS BY INDEX ROWID	PO_SL_BIX_PATH_TABLE
BITMAP CONVERSION TO ROWIDS	
BITMAP AND	
BITMAP CONVERSION FROM BY	

SQL History

Opened nodes (55); Saved nodes(0)

Line 5 Column 1 | Insert | Windows: CR/... Editing

3. View the execution plan of the query that reviews all the purchase orders having a particular part id.

a. Open the file **XQuery04a.sql**. The code is displayed in the **Enter SQL Statement** box. Now, click the **Autotrace** icon. Note the usage of **IPARTNUMBER**.

Code in XQuery04a.sql:

```
--List the purchase orders References having a particular part id

SELECT XMLQuery('/PurchaseOrder/Reference' passing object_value returning content)
FROM PURCHASEORDER
WHERE XMExists('/PurchaseOrder/LineItems/LineItem/Part[@Id="717951002372"]' passing
object_value)
/
```

Oracle SQL Developer : C:\HOL\xmlldb\_1\files\XQuery04a.sql

File Edit View Navigate Run Source Versioning Migration Tools Help

Connections

XQuery04a.sql

SQL Worksheet History

0.19984627 seconds

Enter SQL Statement:

```
--List the purchase orders References having a particular part id

SELECT XMLQuery('/PurchaseOrder/Reference' passing object_value returning content)
FROM PURCHASEORDER
WHERE XMExists('/PurchaseOrder/LineItems/LineItem/Part[@Id="717951002372"]' passing
object_value)
/
```

Results Script Output Explain Autotrace DBMS Output OWVA Output

OPERATION	OBJECT_NAME	COS
SELECT STATEMENT		
NESTED LOOPS SEMI		
TABLE ACCESS FULL	PURCHASEORDER	
TABLE ACCESS BY INDEX ROWID	LINEITEM_TABLE	
INDEX RANGE SCAN	PARTNUMBER	

V\$STATNAME Name V\$MYSTAT Value

SQL History

All Rows Fetched: 14

Line 6 Column 1 | Insert | Windows: CR/... Editing

b. Open the file **XQuery04b.sql**, and click the **Autotrace** icon. Note the usage of **XMLIndex** index.

Code in XQuery04b.sql:

```
--Same query on binary storage table
SELECT XMLQuery('/PurchaseOrder/Reference' passing object_value returning
content).getStringVal()
FROM PO_SL_BIX_TABLE
WHERE XMExists('/PurchaseOrder/LineItems/LineItem/Part[@Id="717951002372"]' passing
object_value)
/
```



Oracle SQL Developer : C:\HOL\omldb\_1\files\XQuery04b.sql

File Edit View Navigate Run Source Versing Migration Tools Help

Connections

SQL Worksheet History

Enter SQL Statement:

```
--Same query on binary storage table
SELECT XMLQuery('/PurchaseOrder/Reference' passing object_value returning content)
FROM PO_SL_BIX_TABLE
WHERE XMLExists('/PurchaseOrder/LineItems/LineItem/Part[@Id="717951002372"]' pass:
/
```

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION	OBJECT_NAME	CC
SELECT STATEMENT		
SORT GROUP BY		
TABLE ACCESS BY INDEX ROWID	PO_SL_BIX_PATH_TABLE	
INDEX RANGE SCAN	PO_SL_BIX_PATH_ID_IX	
NESTED LOOPS		
VIEW	VW_SQ_1	
HASH UNIQUE		
NESTED LOOPS		

SQL History

Opened nodes (55); Saved nodes(0)

Line 5 Column 1 | Insert | Windows: CR/... Editing

4. View the execution plan of the query that lists the description for each line item on a particular purchase order.

a. Open the file **XQuery05a.sql**. The code is displayed in the **Enter SQL Statement** box. Now, click the **Autotrace** icon.

Code in XQuery05a.sql:

```
-- 1. List the description for each item on the purchase order

SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem/Description' passing object_value returning
content)
FROM PURCHASEORDER
WHERE XMLEExists('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing object_value)
/
```

Oracle SQL Developer : C:\HOL\xmlldb\_1\files\XQuery05a.sql

File Edit View Navigate Run Source Versioning Migration Tools Help

Connections

XQuery05a.sql

SQL Worksheet History

0.04119012 seconds

Enter SQL Statement:

```
-- 1. List the description for each item on the purchase order

SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem/Description' passing object_value returning
content)
FROM PURCHASEORDER
WHERE XMLEExists('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing object_value)
/
```

Results Script Output Explain Autotrace DBMS Output OWBA Output

OPERATION	OBJECT_NAME	COS
SELECT STATEMENT		
SORT GROUP BY		
TABLE ACCESS BY INDEX ROWID	LINEITEM_TABLE	
INDEX RANGE SCAN	SYS_C009609	
TABLE ACCESS BY INDEX ROWID	PURCHASEORDER	
INDEX RANGE SCAN	PURCHASEORDERREF	

SQL History

All Rows Fetched: 1

Line 7 Column 1 | Insert | Windows: CR/... Editing

b. Open the file **XQuery05b.sql**. The code is displayed in the **Enter SQL Statement** box. Now, click the **Autotrace** icon.

Code in XQuery05b.sql:

```
--same query on binary storage table
SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem/Description' passing object_value returning
content).getStringVal()
FROM PO_SL_BIX_TABLE
WHERE XMLEExists('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing object_value)
/
```

Oracle SQL Developer : C:\HOL\omldb\_1\files\XQuery05b.sql

File Edit View Navigate Run Source Versing Migration Tools Help

Connections

SQL Worksheet History

0.17509364 seconds

Enter SQL Statement:

```
--same query on binary storage table
SELECT XMLQuery('/PurchaseOrder/LineItems/LineItem/Description' passing object_va
FROM PO_SL_BIX_TABLE
WHERE XMLExists('/PurchaseOrder[Reference="SBELL-2002100912333601PDT"]' passing ob
/
```

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION	OBJECT_NAME
SELECT STATEMENT	
SORT GROUP BY	
TABLE ACCESS BY INDEX ROWID	PO_SL_BIX_PATH_TABLE
INDEX RANGE SCAN	PO_SL_BIX_PATH_ID_IX
NESTED LOOPS	
VIEW	VW_SQ_1
HASH UNIQUE	
NESTED LOOPS	

V\$STATNAME Name V\$MYSTAT Value

SQL History

Opened nodes (55); Saved nodes(0)

Line 5 Column 1 Insert Windows: CR/... Editing

- View the execution plan of the query that lists the references for LineItem 20 with a description containing picnic on a particular purchase order. Open the file **XQuery06b.sql**. The code is displayed in the **Enter SQL Statement** box. Now, click the **Autotrace** icon.

Code in XQuery06b.sql:

```
--same query on binary storage table
SELECT XMLQuery('/PurchaseOrder/Reference' passing object_value returning content).getStringVal()
FROM PO_SL_BIX_TABLE
WHERE
XMLExists('/PurchaseOrder/LineItems/LineItem[@ItemNumber=20]/Description/text() [ora:contains(., "Picnic"
> 0]' passing object_value)
/
```

Oracle SQL Developer : C:\HOL\bmldb\_1\files\XQuery06b.sql

File Edit View Navigate Run Source Versioning Migration Tools Help

Connections

SQL Worksheet History

Enter SQL Statement:

```
--same query on binary storage table
SELECT XMLQuery('/PurchaseOrder/Reference' passing object_value returning content).getStringVal()
FROM PO_SL_BIX_TABLE
WHERE XMLExists('/PurchaseOrder/LineItems/LineItem[@ItemNumber=20]/Description/text() [ora:contains(., "Picnic"
> 0]' passing object_value)
/
```

Autotrace (F10)

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION	OBJECT_NAME
SELECT STATEMENT	
SORT GROUP BY	
TABLE ACCESS BY INDEX ROWID	PO_SL_BIX_PATH_TABLE
INDEX RANGE SCAN	PO_SL_BIX_PATH_ID_IX
NESTED LOOPS	
VIEW	VW_SQ_1
HASH UNIQUE	
NESTED LOOPS	
NESTED LOOPS	
TABLE ACCESS BY INDEX ROWID	PO_SL_BIX_PATH_TABLE

SQL History

Opened nodes (56); Saved nodes(0)

Line 5 Column 1 | Insert | Windows: CR/... Editing

In the above topic, you learned how to create B-Tree indexes on object relational storage and `XMLIndex` index on binary XML table. You also learned how to observe the performance of SQL/XML, XQuery expressions by viewing the explain plan.



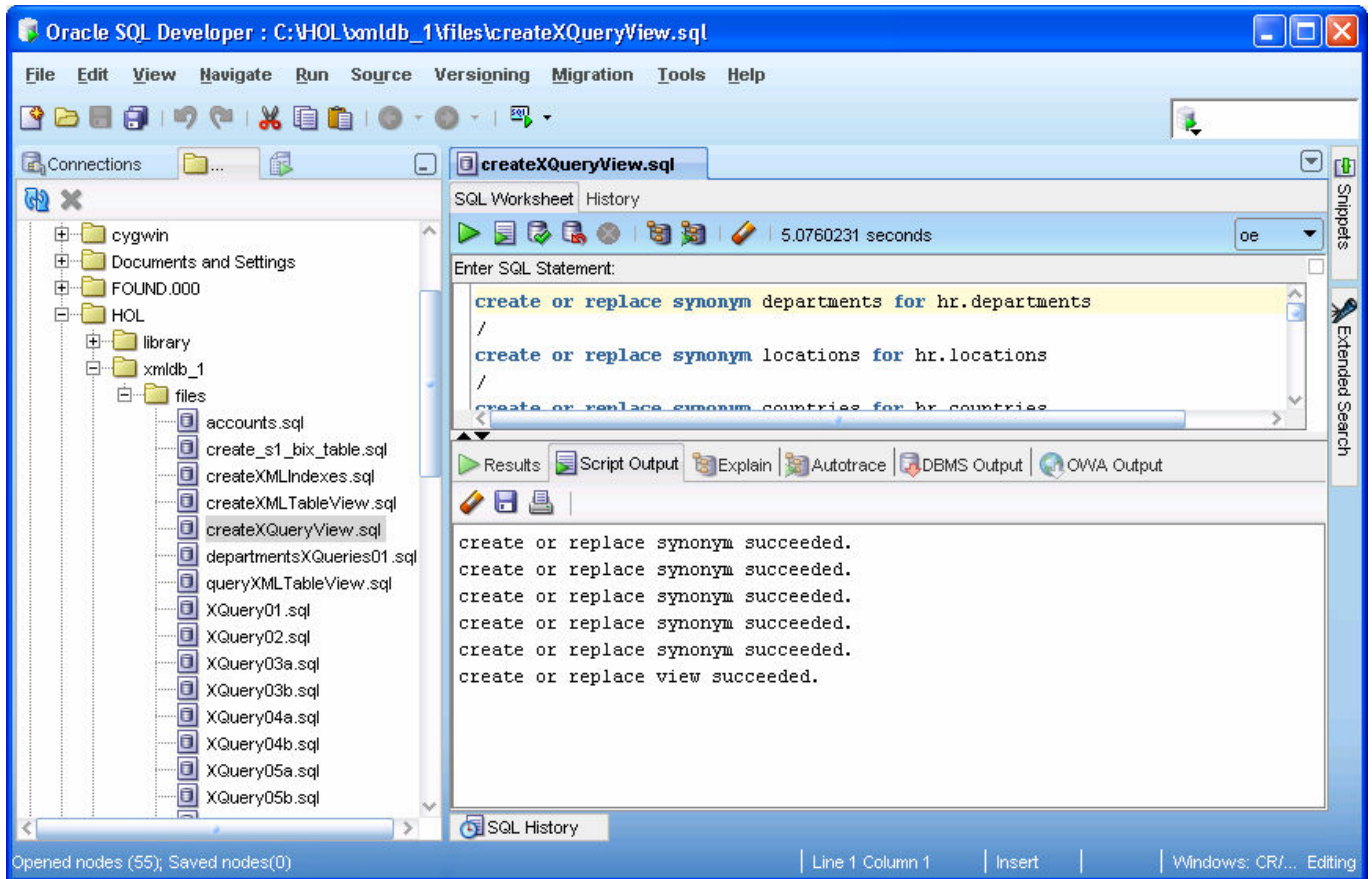
## Using XMLType Views with XQuery

You can use XQuery to generate XML from SQL data using Views. Perform the following steps:

1. You first will create an XML view over relational tables by using a `XMLTable()` SQL/XML function and an XQuery expression. Execute the following script:

### CreateXQueryView.sql

```
create or replace synonym departments for hr.departments
/
create or replace synonym locations for hr.locations
/
create or replace synonym countries for hr.countries
/
create or replace synonym employees for hr.employees
/
create or replace synonym jobs for hr.jobs
/
create or replace view DEPARTMENT_XML of xmltype
with object id
(extract(object_value, '/Department/@DepartmentId').getnumberVal())
as
select column_value from XMLTable
('for $d in ora:view("DEPARTMENTS"),
 $l in ora:view("LOCATIONS"),
 $c in ora:view("COUNTRIES")
where $d/ROW/LOCATION_ID = $l/ROW/LOCATION_ID
and $l/ROW/COUNTRY_ID = $c/ROW/COUNTRY_ID
return
<Department DepartmentId= "{ $d/ROW/DEPARTMENT_ID/text() }" >
<Name>{ $d/ROW/DEPARTMENT_NAME/text() }</Name>
<Location>
<Address>{ $l/ROW/STREET_ADDRESS/text() }</Address>
<City>{ $l/ROW/CITY/text() }</City>
<State>{ $l/ROW/STATE_PROVINCE/text() }</State>
<Zip>{ $l/ROW/POSTAL_CODE/text() }</Zip>
<Country>{ $c/ROW/COUNTRY_NAME/text() }</Country>
</Location>
<EmployeeList>
{
for $e in ora:view("EMPLOYEES"),
 $m in ora:view("EMPLOYEES"),
 $j in ora:view("JOBS")
where $e/ROW/DEPARTMENT_ID = $d/ROW/DEPARTMENT_ID
and $j/ROW/JOB_ID = $e/ROW/JOB_ID
and $m/ROW/EMPLOYEE_ID = $e/ROW/MANAGER_ID
return
<Employee employeeNumber="{ $e/ROW/EMPLOYEE_ID/text() }" >
<FirstName>{ $e/ROW/FIRST_NAME/text() }</FirstName>
<LastName>{ $e/ROW/LAST_NAME/text() }</LastName>
<EmailAddress>{ $e/ROW/EMAIL/text() }</EmailAddress>
<Telephone>{ $e/ROW/PHONE_NUMBER/text() }</Telephone>
<StartDate>{ $e/ROW/HIRE_DATE/text() }</StartDate>
<JobTitle>{ $j/ROW/JOB_TITLE/text() }</JobTitle>
<Salary>{ $e/ROW/SALARY/text() }</Salary>
<Manager>{ $m/ROW/LAST_NAME/text(), " ", " ", $m/ROW/FIRST_NAME/text() }</Manager>
</Employee>
}
</EmployeeList>
</Department>')
```



2. Now you can show XQuery over the XML view you just created.

a. Execute the script **xqueryXQLView.sql**.

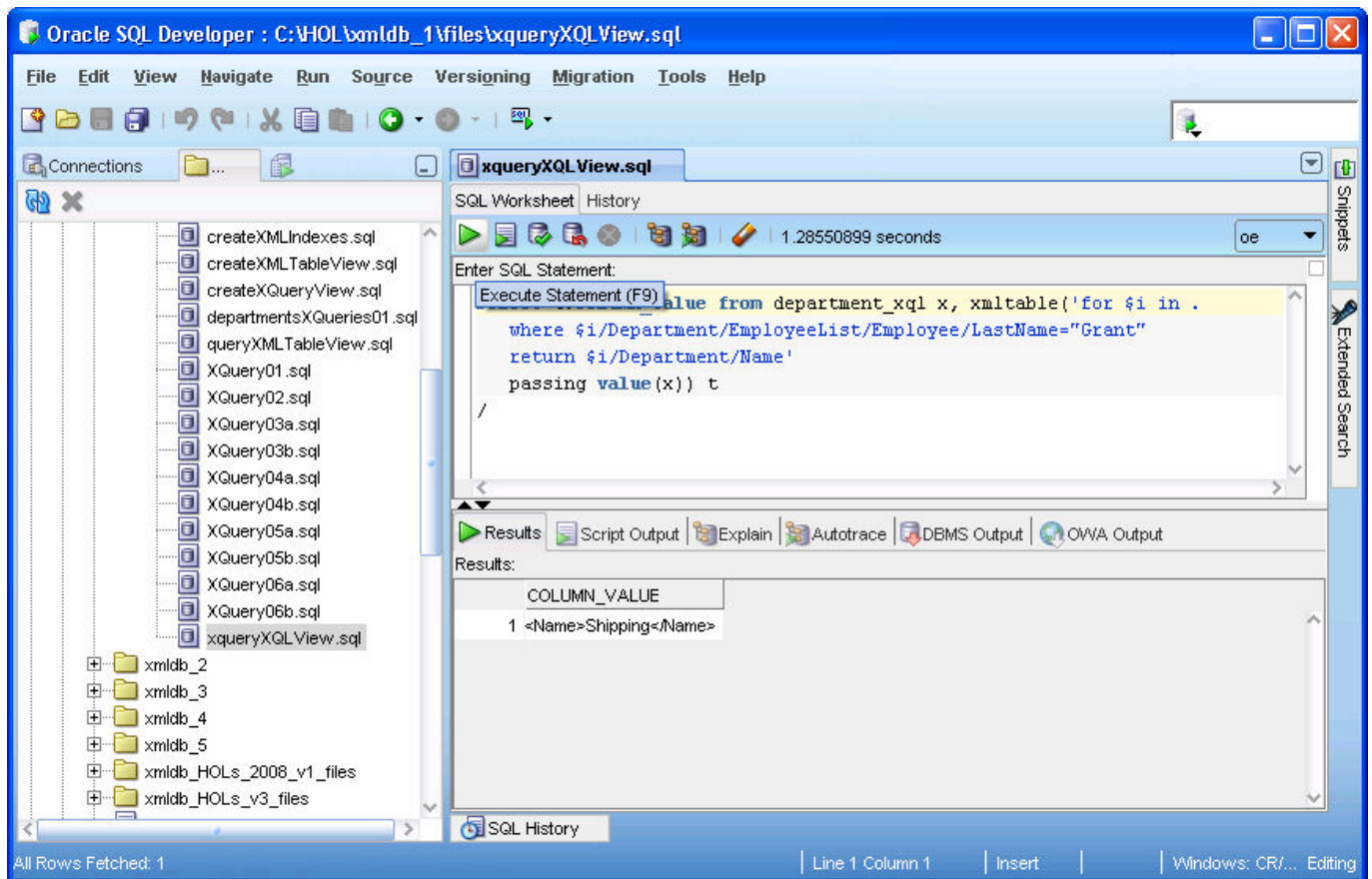
```
select t.column_value from department_xql x, xmltable('for $i in .

where $i/Department/EmployeeList/Employee/LastName="Grant"

return $i/Department/Name'

passing value(x)) t

/
```



b. View the execution plan of the query. Click the **Autotrace** icon.

Oracle SQL Developer : C:\HOL\xmldb\_1\files\xqueryXQLView.sql

File Edit View Navigate Run Source Versioning Migration Tools Help

Connections

- createXMLIndexes.sql
- createXMLTableView.sql
- createXQueryView.sql
- departmentsXQueries01.sql
- queryXMLTableView.sql
- XQuery01.sql
- XQuery02.sql
- XQuery03a.sql
- XQuery03b.sql
- XQuery04a.sql
- XQuery04b.sql
- XQuery05a.sql
- XQuery05b.sql
- XQuery06a.sql
- XQuery06b.sql
- xqueryXQLView.sql
- xmldb\_2
- xmldb\_3
- xmldb\_4
- xmldb\_5
- xmldb\_HOLs\_2008\_v1\_files
- xmldb\_HOLs\_v3\_files
- ~\ldb\_HOLs\_2008\_v1.htm
- S298830-1\_Structured.doc
- S298832-3\_Unstructured.doc

xqueryXQLView.sql

SQL Worksheet History

0.51767266 seconds

Enter SQL Statement:

```
select t.column_value from department_xql x, xmltable('for $i in .
where $i/Department/EmployeeList/Employee/LastName="Grant"
return $i/Department/Name'
passing value(x)) t
/
```

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION	OBJECT_NAME
SELECT STATEMENT	
FILTER	
MERGE JOIN	
TABLE ACCESS BY INDEX ROWID	DEPARTMENTS
INDEX FULL SCAN	DEPT_LOCATION_IX
SORT JOIN	
TABLE ACCESS BY INDEX ROWID	LOCATIONS
INDEX FULL SCAN	LOC_COUNTRY_IX
TABLE ACCESS BY INDEX ROWID	EMPLOYEES
INDEX RANGE SCAN	EMP_NAME_IX

SQL History

All Rows Fetched: 1

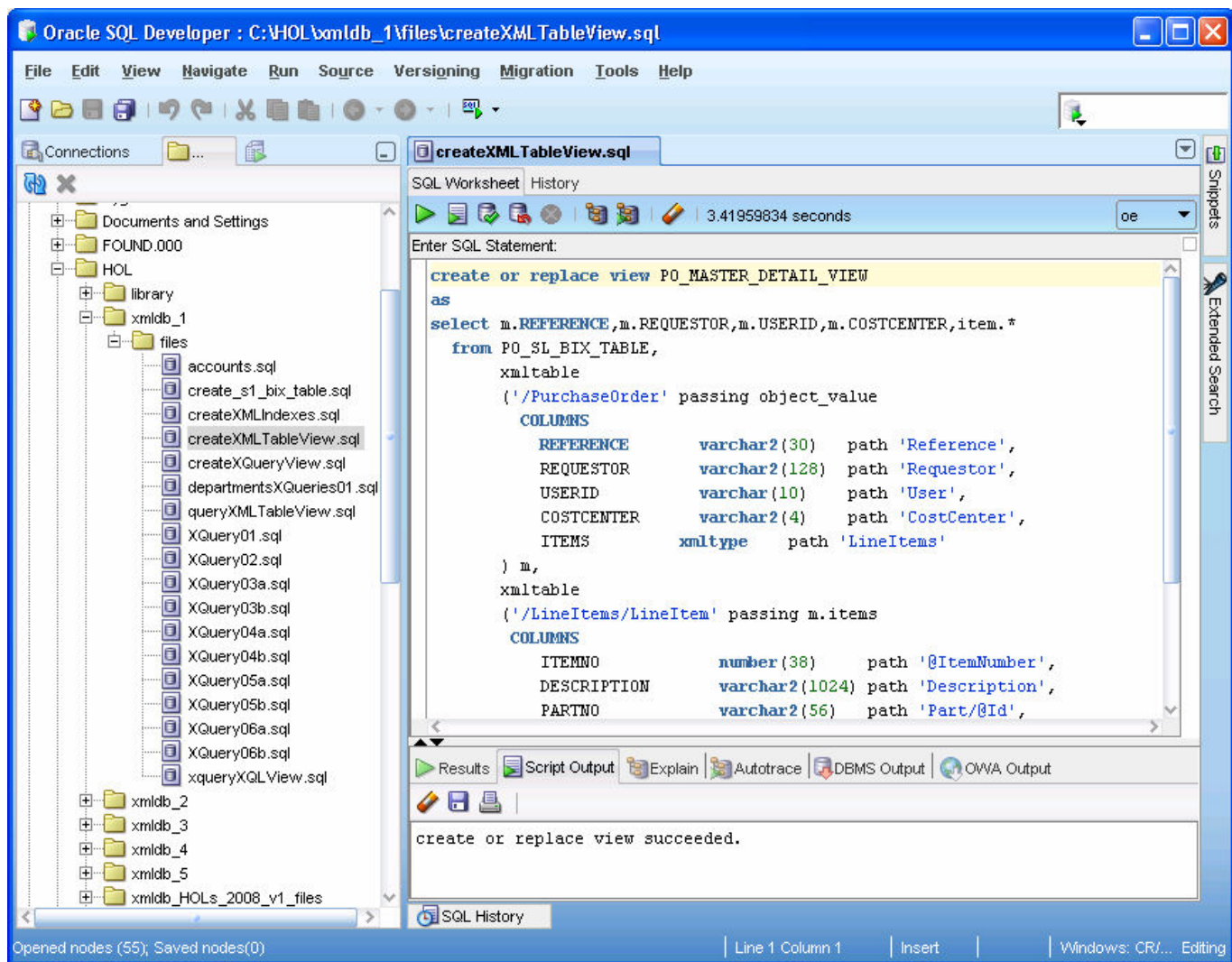
Line 1 Column 1 | Insert | Windows: CR/... Editing

## Using Relational Views over Binary XML Table

You can use the `XMLTable()` function to create and efficiently query relational views over binary XML tables. Perform the following steps:

1. You first will create a relational view over a binary XML by using a `XMLTable()` SQL/XML function. Execute the script **createXMLTableView.sql**.

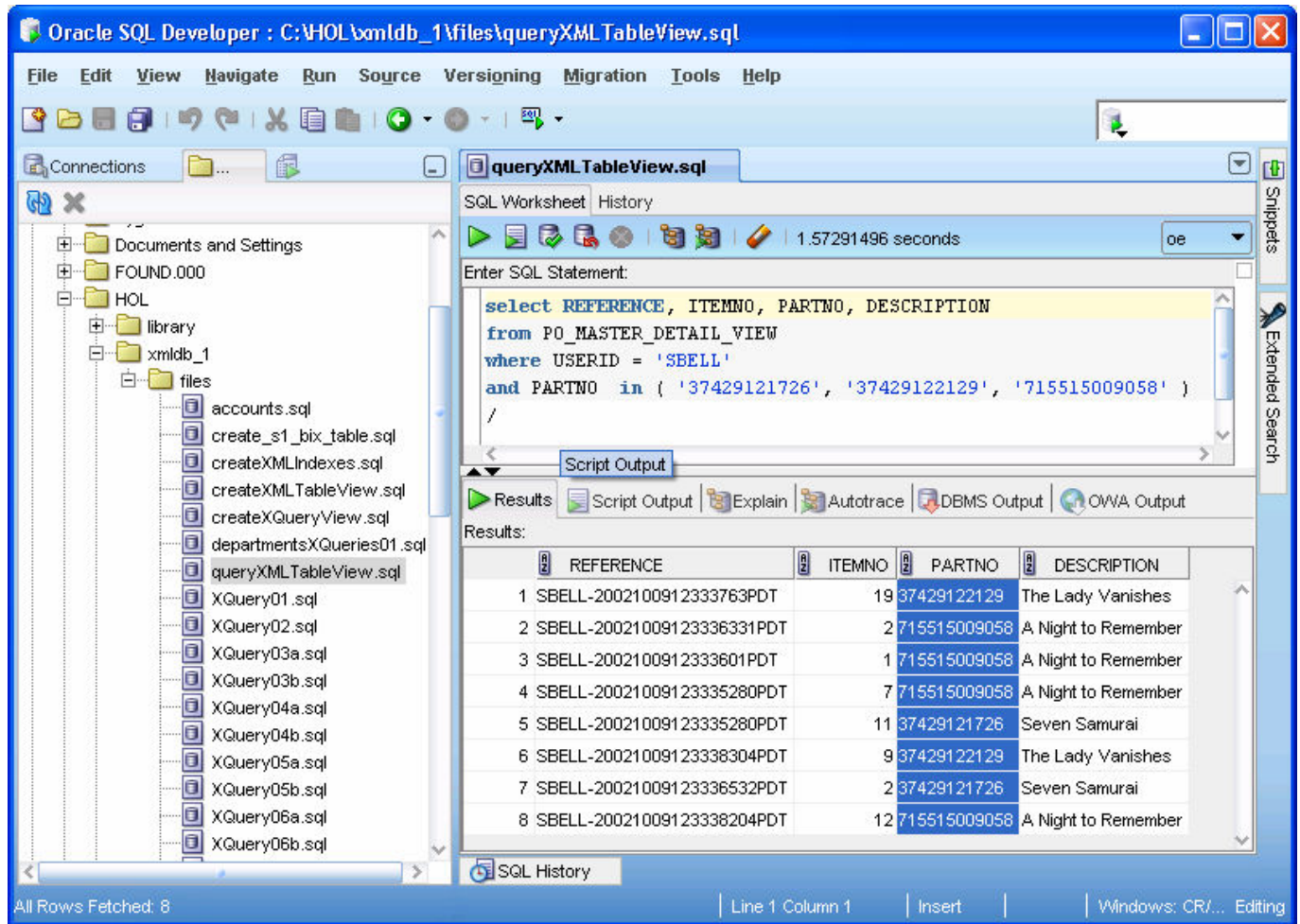
```
create or replace view PO_MASTER_DETAIL_VIEW
as
select m.REFERENCE,m.REQUESTOR,m.USERID,m.COSTCENTER,item.*
from PO_SL_BIX_TABLE,
xmltable
('/PurchaseOrder' passing object_value
COLUMNS
REFERENCE varchar2(30) path 'Reference',
REQUESTOR varchar2(128) path 'Requestor',
USERID varchar(10) path 'User',
COSTCENTER varchar2(4) path 'CostCenter',
ITEMS xmltype path 'LineItems'
) m,
xmltable
('/LineItems/LineItem' passing m.items
COLUMNS
ITEMNO number(38) path '@ItemNumber',
DESCRIPTION varchar2(1024) path 'Description',
PARTNO varchar2(56) path 'Part/@Id',
QUANTITY number(38) path 'Part/@Quantity',
UNITPRICE number(12,2) path 'Part/@UnitPrice'
) item
/
```





2. Now you can use SQL queries on the relational view. Execute the script **queryXMLTableView.sql**.

```
select REFERENCE, ITEMNO, PARTNO, DESCRIPTION
from PO_MASTER_DETAIL_VIEW
where USERID = 'SBELL'
and PARTNO in ( '37429121726', '37429122129', '715515009058' )
/
```



Oracle SQL Developer : C:\HOL\oradb\_1\files\queryXMLTableView.sql

File Edit View Navigate Run Source Versioning Migration Tools Help

Connections

Documents and Settings  
FOUND.000  
HOL  
library  
xmldb\_1  
files  
accounts.sql  
create\_s1\_bix\_table.sql  
createXMLIndexes.sql  
createXMLTableView.sql  
createXQueryView.sql  
departmentsXQueries01.sql  
queryXMLTableView.sql  
XQuery01.sql  
XQuery02.sql  
XQuery03a.sql  
XQuery03b.sql  
XQuery04a.sql  
XQuery04b.sql  
XQuery05a.sql  
XQuery05b.sql  
XQuery06a.sql  
XQuery06b.sql

queryXMLTableView.sql

SQL Worksheet History

1.57291496 seconds

Enter SQL Statement:

```
select REFERENCE, ITEMNO, PARTNO, DESCRIPTION
from PO_MASTER_DETAIL_VIEW
where USERID = 'SBELL'
and PARTNO in ( '37429121726', '37429122129', '715515009058' )
/
```

Script Output

Results

	REFERENCE	ITEMNO	PARTNO	DESCRIPTION
1	SBELL-2002100912333763PDT	19	37429122129	The Lady Vanishes
2	SBELL-20021009123336331PDT	2	715515009058	A Night to Remember
3	SBELL-2002100912333601PDT	1	715515009058	A Night to Remember
4	SBELL-20021009123335280PDT	7	715515009058	A Night to Remember
5	SBELL-20021009123335280PDT	11	37429121726	Seven Samurai
6	SBELL-20021009123338304PDT	9	37429122129	The Lady Vanishes
7	SBELL-20021009123336532PDT	2	37429121726	Seven Samurai
8	SBELL-20021009123338204PDT	12	715515009058	A Night to Remember

SQL History

All Rows Fetched: 8

Line 1 Column 1 | Insert | Windows: CR/... Editing

In the above topic, you learned how to create a relational view over a binary XML table. You also learned how to use query the relational view.



## Summary

In this tutorial, you learned how to:

- ▣ Review an XML Schema in Enterprise Manager
- ▣ Create a binary XML table and store data
- ▣ Add indexes to improve the performance of XQuery expressions
- ▣ Create an `XMLType` view with XQuery
- ▣ Use relational views over binary XML table

# Lesson 2: Performing In-Place XML Schema Evolution

## Purpose





This tutorial shows you how to make changes to an XML schema without requiring that existing data to be copied, deleted, and reinserted.

## Time to Complete

Approximately 20 minutes.

## Topics

This tutorial covers the following topics:

-  [Overview](#)
-  [Prerequisites](#)
-  [Performing In-Place XML Schema Evolution](#)
-  [Summary](#)

## Overview

Since Oracle 9i Database Release 2, Oracle XML DB has been seamlessly integrated with the Oracle database to provide high-performance database-native storage, retrieval, and management of XML data. With the new Oracle Database 11g release, Oracle XML DB is taking another leap ahead with a rich set of new capabilities to simplify DBAs' tasks in managing XML data while further empowering XML and SOA application developers. Oracle XML DB now supports multiple database-native XML storage models and XML indexing schemes, SQL/XML standard operations, W3C standard XQuery data model and XQuery/XPath languages, In-place XML schema evolution, database-native web services, high performance XML publishing, XML DB repository, and versioning and access control. This tutorial covers the In-place XML schema evolution and using Oracle XML DB Web services for Service-Oriented Architecture.

## Schema Evolution

A major challenge for developers using an XML schema with Oracle XML DB is how to deal with changes in the content or structure of XML documents. In some environments, the need for changes may be frequent or extensive, arising from new regulations, internal needs, or external opportunities. For example, new elements or attributes may need to be added to an XML schema definition, a data type may need to be modified, or certain minimum and maximum occurrence requirements may need to be relaxed or tightened.

In such cases, you need to "evolve" the XML schema so that new requirements are accommodated, while any existing instance documents (the data) remain valid (or can be made valid), and existing applications can continue to run.

If you do not care about any existing documents, you can of course simply drop the XMLType tables that are dependent on the XML schema, delete the old XML schema, and register the new XML schema at the same URL. In most cases, however, you need to keep the existing documents, possibly transforming them to accommodate the new XML schema.

Oracle XML DB supports two kinds of schema evolution. Each approach has its own PL/SQL procedure:

`DBMS_XMLSCHEMA.copyEvolve` for copy-based evolution and `DBMS_XMLSCHEMA.inPlaceEvolve` for in-place evolution, which is introduced in the Oracle Database 11g release.

## In-place Schema Evolution

In-place XML schema evolution makes changes to an XML schema without requiring that existing data be copied, deleted, and reinserted. In-place evolution is therefore much faster than copy-based evolution. In general, in-place evolution is permitted if you are not changing the storage model and if the changes do not invalidate existing documents (that is, if existing documents are conformant with the new schema or can be made conformant with it).

In-place XML schema evolution constructs a new version of an XML schema by applying changes specified in a `diffXML` document, validates that new XML schema (against the XML schema for XML schemas), constructs DDL statements to evolve the disk structures used to store the XML instance documents associated with the XML schema, executes these DDL statements, and replaces the old version of the XML schema with the new, in that order.

## Prerequisites

Before you perform this tutorial, you should first complete the following steps:

1. Install Oracle Database 11g and make sure the `OE`, `HR` users are unlocked.
2. Check your Oracle SQL Developer (version 1.5.1) installation
3. Check the files in the working directory (`/HOL/xmlldb_2/files`) for this lesson.

**Note:** If you use an earlier version of Oracle JDeveloper, the screenshots may slightly differ.

## Performing In-Place XML Schema Evolution

You will use the `DBMS_XMLSCHEMA.inPlaceEvolve` procedure to perform in-place XML schema evolution. Using this procedure, you identify the changes to be made to an existing XML schema by specifying an XML schema-differences document.

Perform the following tasks:

-  [Start SQL Developer](#)
-  [Use In-Place XML Schema Evolution](#)

### Start SQL Developer

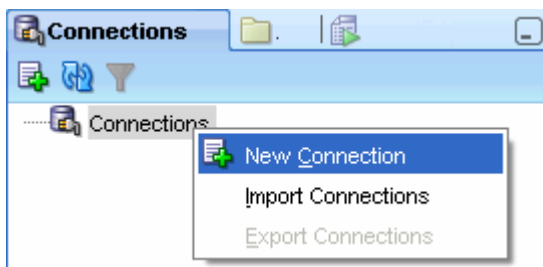
In this tutorial, you use the SQL Developer tool. After creating a database connection, you set autotrace parameters and script pathing reference in SQL Developer. Perform the following steps:

1. Click on the SQL Developer icon on the desktop to start the application.



2. In SQL Developer, you must create a database connection as OE user. Perform the following steps.

- a. In the Connections tab, right-click **Connections** and select **New Connection**.



- b. The New/Select Database Connection window appears. Enter the following details, and click **Test** to make sure that the connection has been set correctly.

Connection Name: **oe**  
UserName: **oe**  
Password: **oe**  
Hostname: **localhost**  
Port: **1521**  
SID: **orcl**

If you select the Save Password check box, the password is saved to an XML file. Therefore, once you close SQL Developer connection and open again, you will not be prompted for the password.

**New / Select Database Connection**

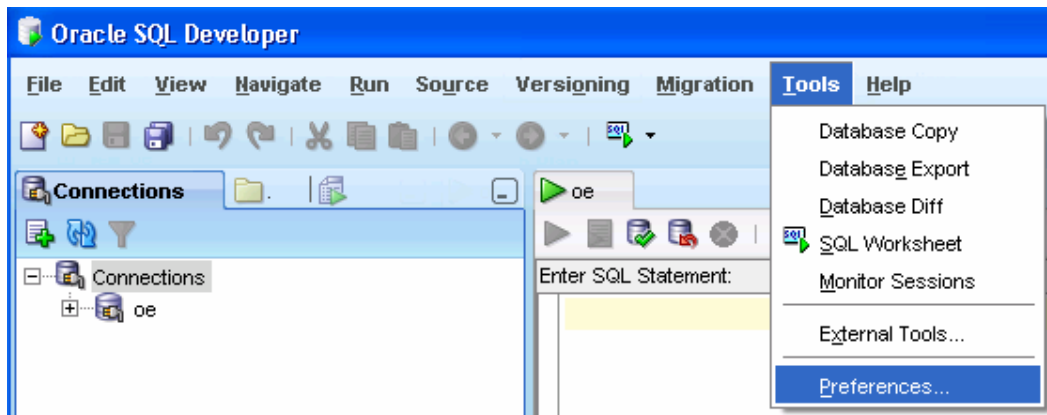
Connection Name	Connection Details
	<p>Connection Name: <input type="text" value="oe"/></p> <p>Username: <input type="text" value="oe"/></p> <p>Password: <input type="text" value="**"/></p> <p><input checked="" type="checkbox"/> Save Password</p> <p><input type="button" value="Oracle"/> <input type="button" value="Access"/></p> <p>Role: <input type="text" value="default"/> <input type="checkbox"/> OS Authentication</p> <p>Connection Type: <input type="text" value="Basic"/> <input type="checkbox"/> Proxy Connection</p> <p>Hostname: <input type="text" value="localhost"/></p> <p>Port: <input type="text" value="1521"/></p> <p><input checked="" type="radio"/> SID <input type="radio"/> Service name</p> <p>SID: <input type="text" value="orcl"/></p> <p>Service name: <input type="text"/></p>

Status : Success

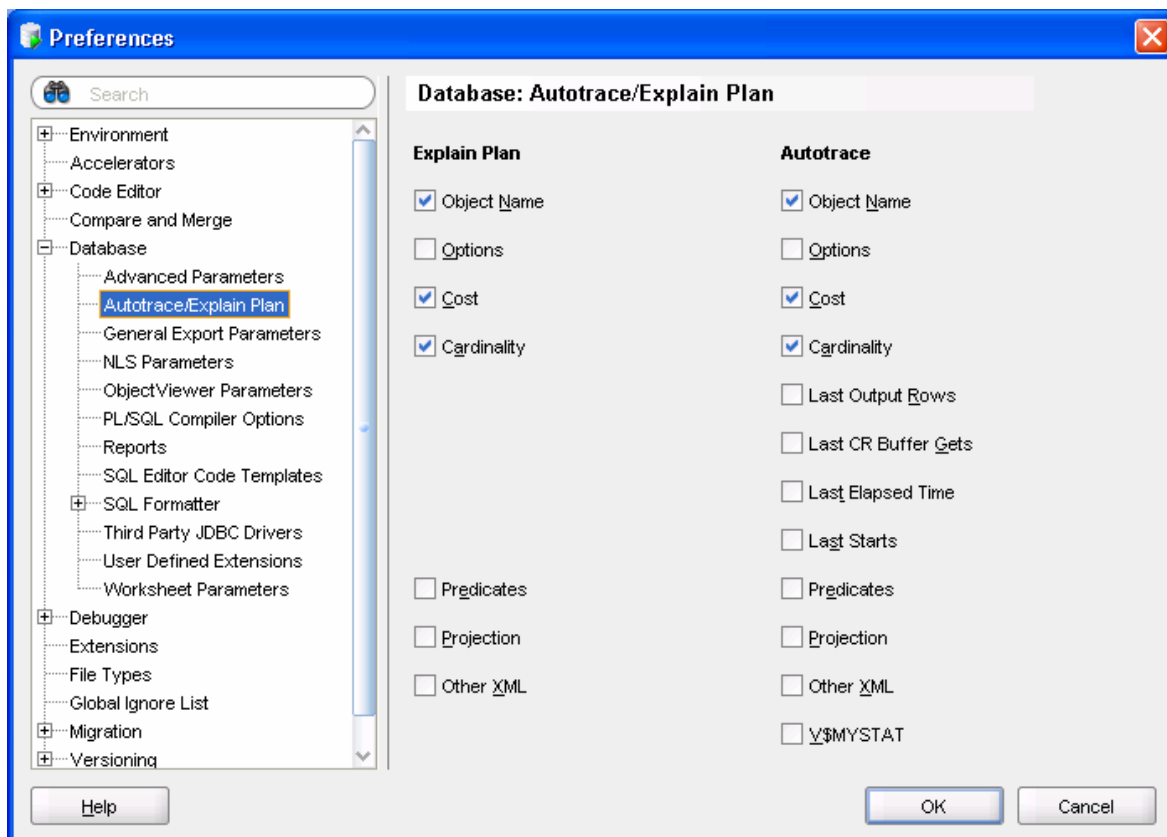
c. The test status shows success. Click **Connect**.

3. Set the Autotrace parameters. Perform the following steps:

a. Go to **Tools > Preferences**.



b. Expand **Database**, and select **Autotrace Parameters**.

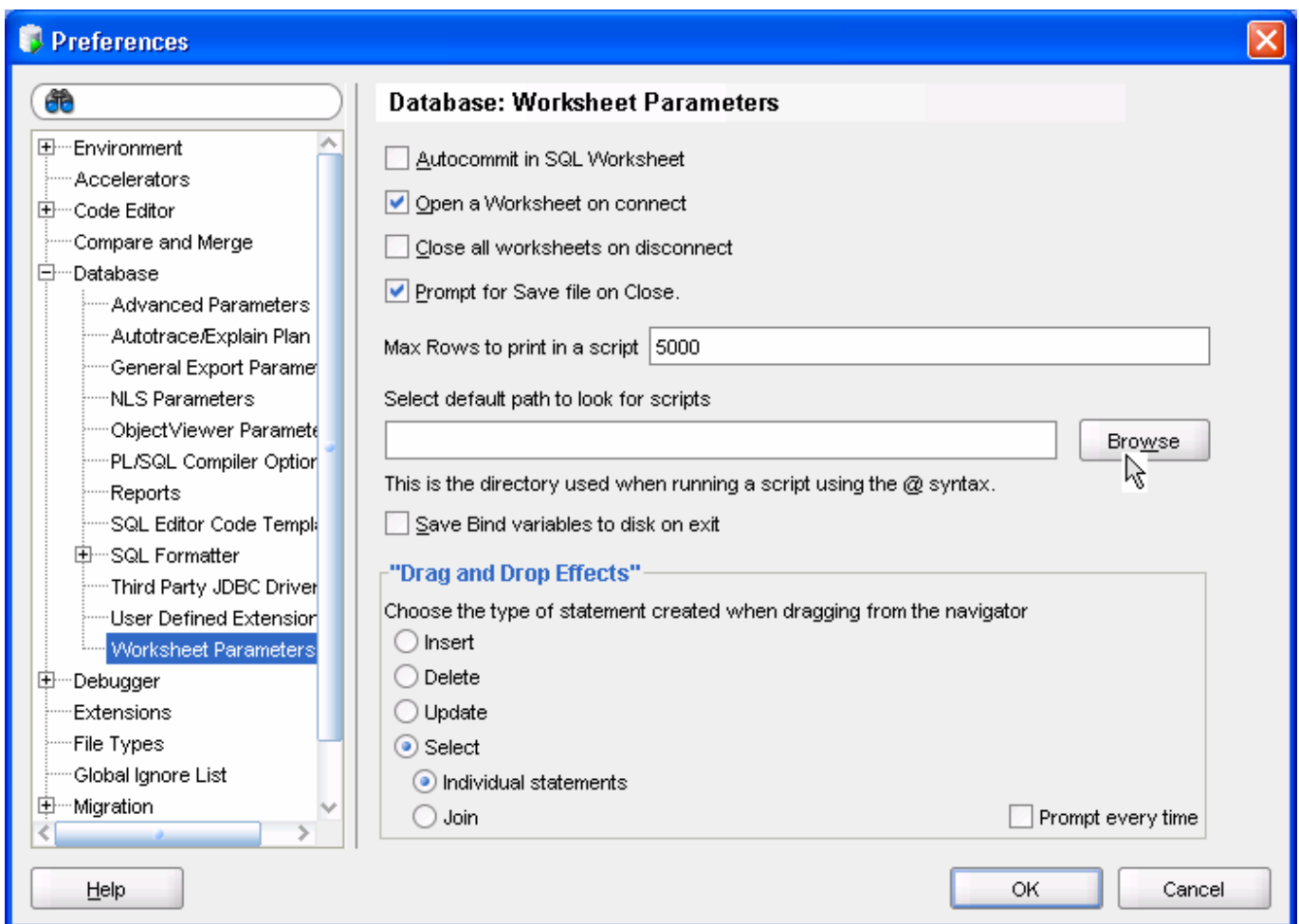
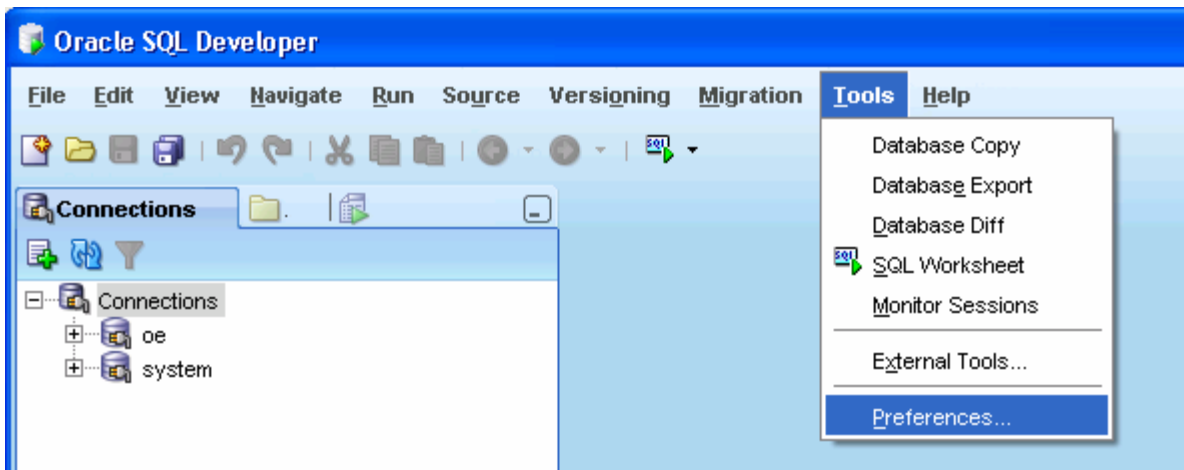


c. In the Preferences window, make sure to select the following check boxes. Then, click **OK**.

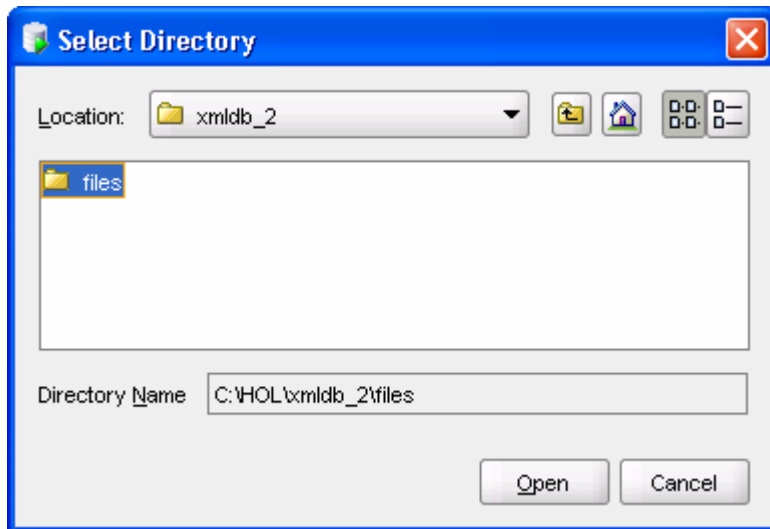
Object\_Name, Cost, and Cardinality

4. To run the scripts by using the @ syntax, you can set the script pathing reference in SQL Developer. Perform the following steps:

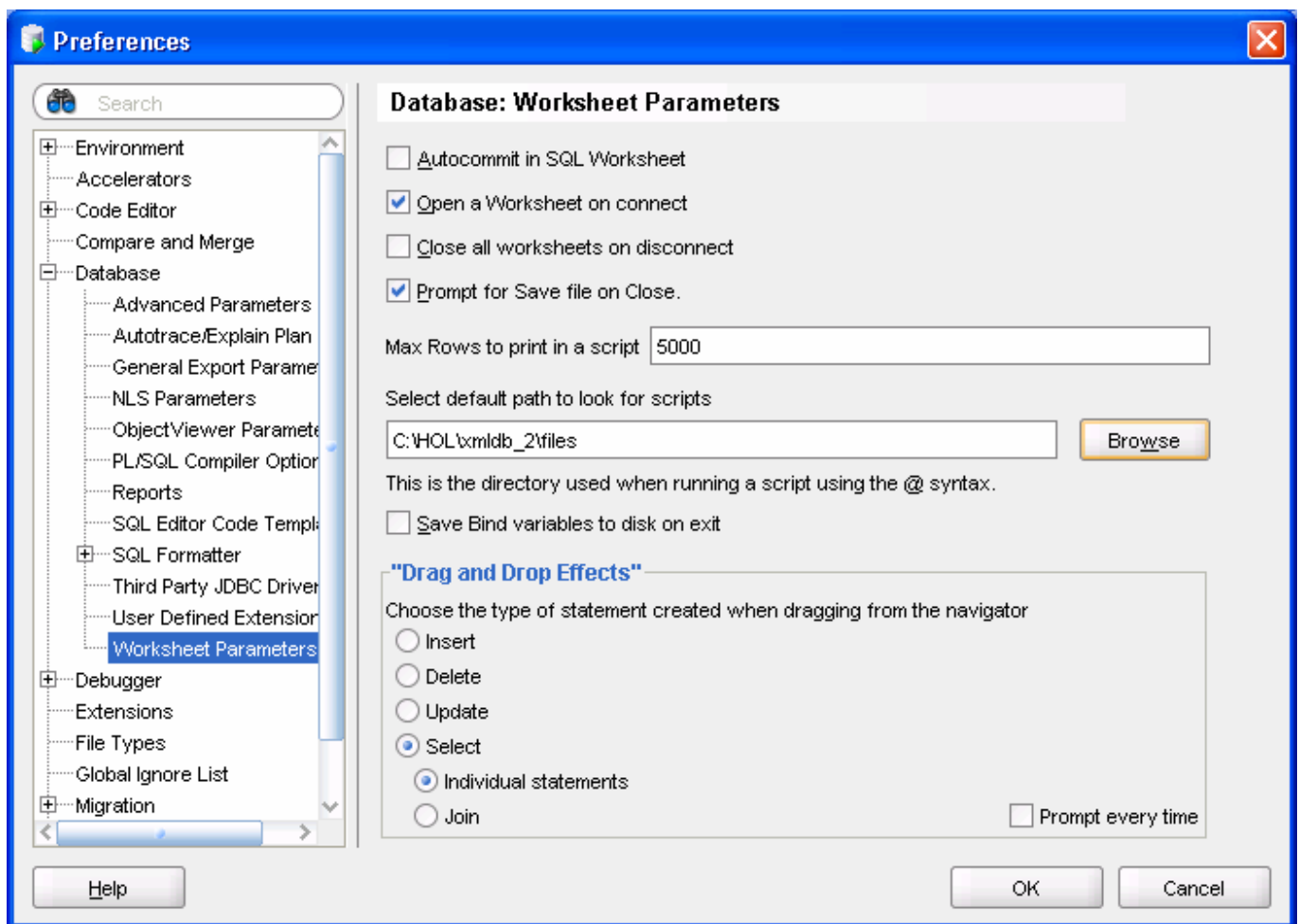
- a. Select **Tools > Preferences > Database > Worksheet Parameters**. Then, click **Browse**.



b. Browse to the location of your working directory that has the SQL scripts. Then, click **Open**.



c In the Preferences window, verify the script path in the Select default path to look for scripts field. Click **OK**.





In the above section, you learned how to connect to SQL Developer, set autotrace parameters and set script pathing reference.

## Use In-Place XML Schema Evolution

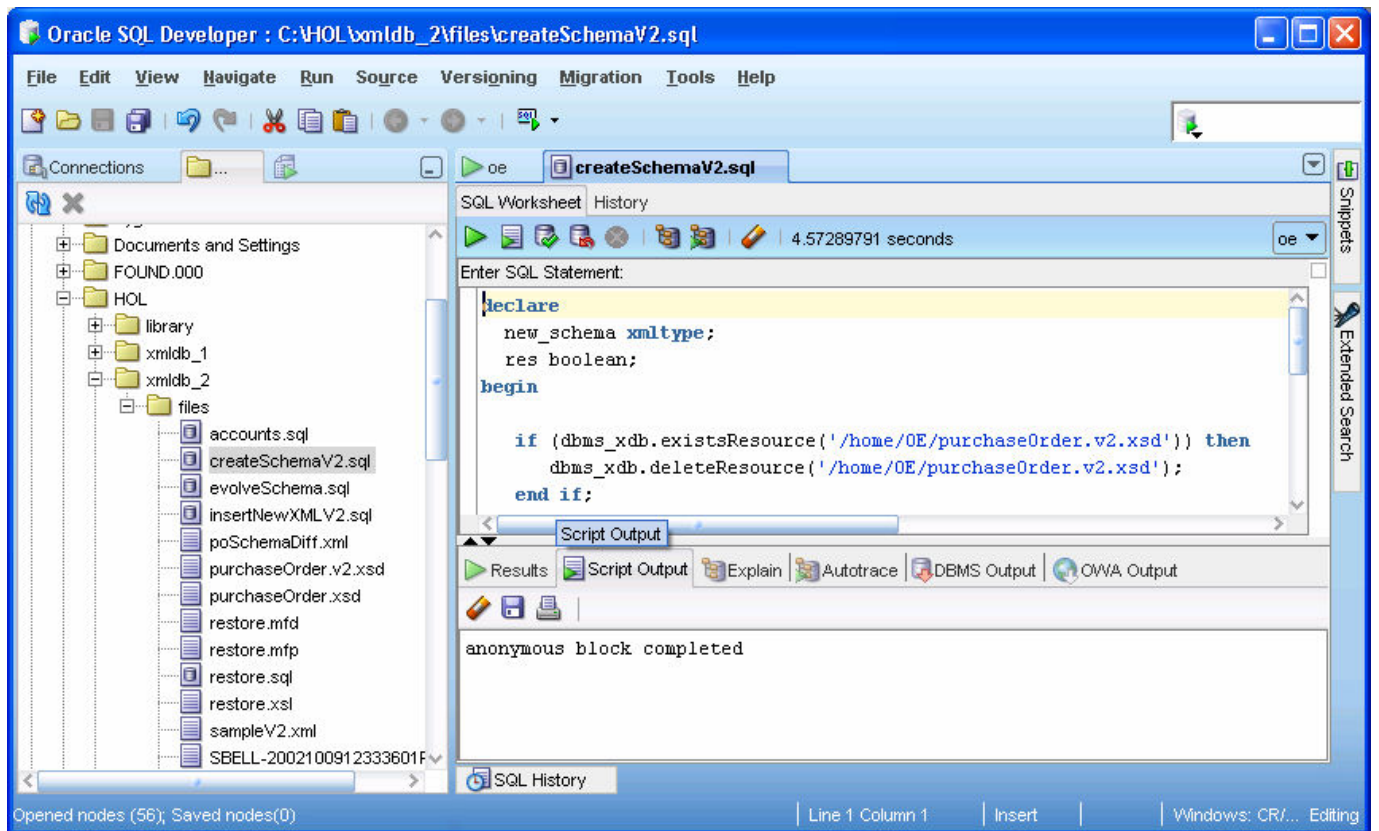
1. Create a new version of an XML schema from an existing one. Execute the script `createSchemaV2.sql`.

```
@createSchemaV2.sql

declare
new_schema xmltype;
res boolean;
begin
if (dbms_xdb.existsResource('/home/OE/purchaseOrder.v2.xsd')) then
dbms_xdb.deleteResource('/home/OE/purchaseOrder.v2.xsd');
end if;
select appendChildXML (xdbUriType('/home/OE/purchaseOrder.xsd').getXML(),
'/xs:schema/xs:complexType[@name="LineItemType"]/xs:sequence',
xmltype('<xs:element xmlns:xs="http://www.w3.org/2001/XMLSchema" name="Unit" type="xs:string"
minOccurs="0" />'),
'xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xdb="http://xmlns.oracle.com/xdb"')
into new_schema from dual;
res := dbms_xdb.createResource('/home/OE/purchaseOrder.v2.xsd',new_schema);
commit;

end;
/
```

To execute a SQL script in SQL Developer, you can open the file first. Then, click **Run Script** or press **[F5]**.



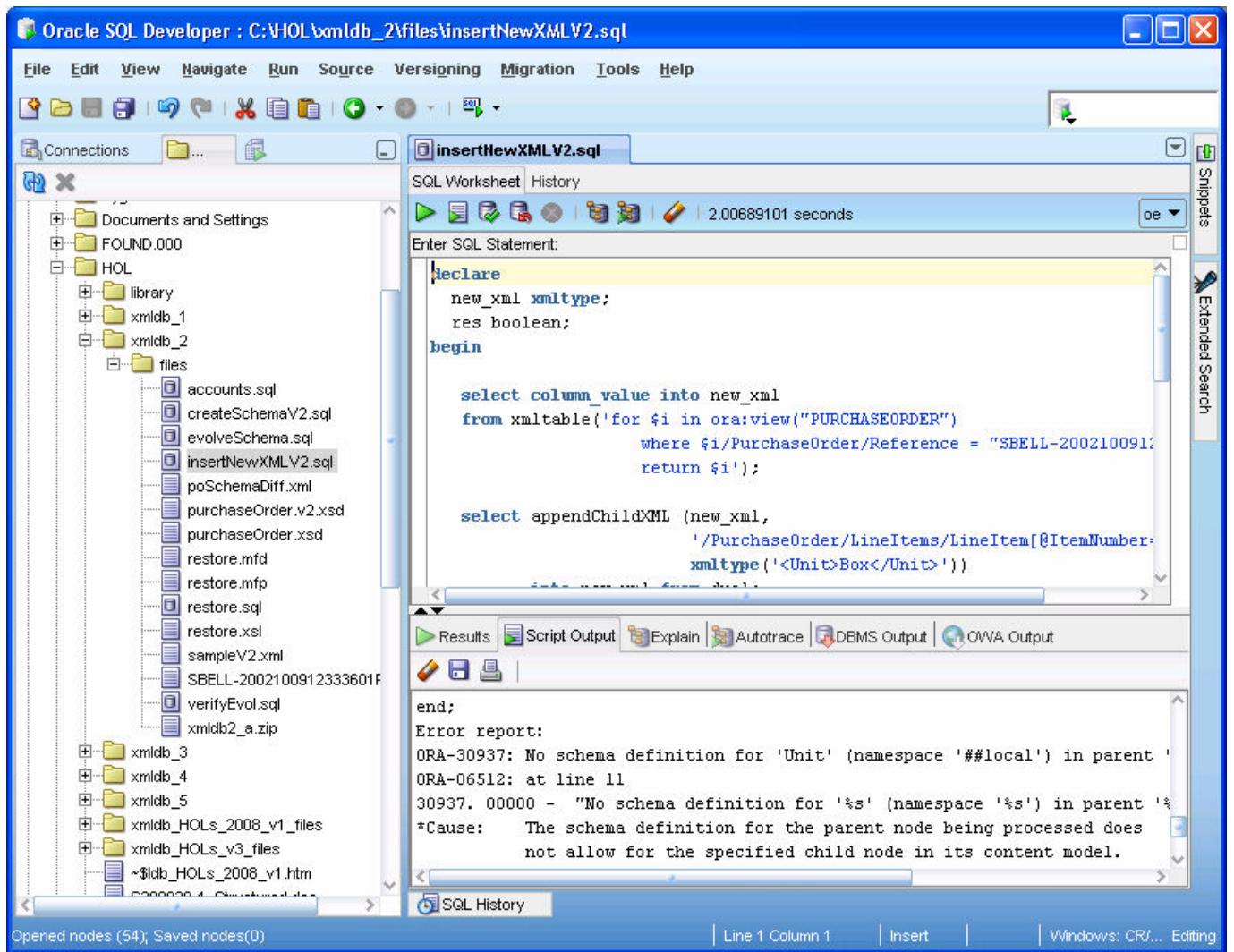
2. Insert a new XML document conforming to the new version of the XML schema. Execute the script `insertNewXMLV2.sql`. Observe the error in the script output pane.

```
@insertNewXMLV2.sql

declare
new_xml xmltype;
res boolean;
begin
select column_value into new_xml
from xmltable('for $i in ora:view("PURCHASEORDER")
where $i/PurchaseOrder/Reference = "SBELL-2002100912333601PDT"
return $i');
select appendChildXML (new_xml,
'/PurchaseOrder/LineItems/LineItem[@ItemNumber="1"]',
xmltype('<Unit>Box</Unit>'))
into new_xml from dual;
select appendChildXML (new_xml,
'/PurchaseOrder/LineItems/LineItem[@ItemNumber="2"]',
xmltype('<Unit>Carton</Unit>'))
into new_xml from dual;
select appendChildXML (new_xml,
'/PurchaseOrder/LineItems/LineItem[@ItemNumber="3"]',
xmltype('<Unit>Case</Unit>'))
into new_xml from dual;
select updateXML (new_xml,
'/PurchaseOrder/Reference/text()',
'SBELL-2002100912333601PDT-V2')
into new_xml from dual;
insert into PURCHASEORDER values new_xml;

commit;

end;
/
```



- Now, perform in-place XML schema evolution. Execute **evolveSchema.sql**. Notice the **xmlDiff** SQL function call in the script that compares the differences between the old and the new schemas.

@evolveSchema.sql

```

set timing on
declare
schemaDiff XMLType;
res boolean;
begin
select xmlDiff
(
  xdburitype('/home/OE/purchaseOrder.xsd').getXML(),
  xdburitype('/home/OE/purchaseOrder.v2.xsd').getXML()
)
into schemaDiff
from dual;

if (dbms_xdb.existsResource('/home/OE/poSchemaDiff.xml')) then
dbms_xdb.deleteResource('/home/OE/poSchemaDiff.xml');
end if;

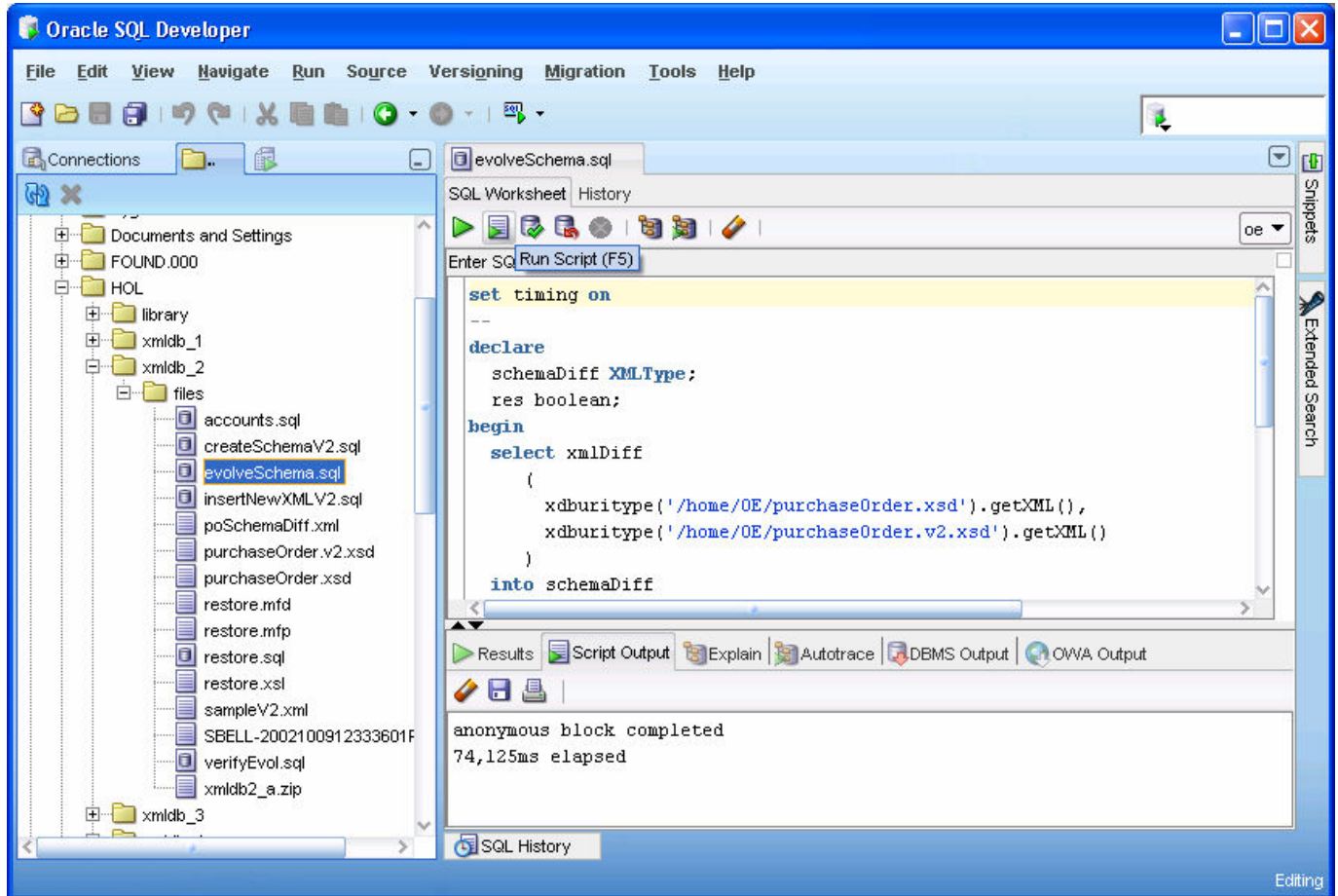
```

```

res := dbms_xdb.createResource('/home/OE/poSchemaDiff.xml',schemaDiff);

dbms_xmlschema.inPlaceEvolve('http://localhost:8080/source/schemas/poSource/xsd/purchaseOrder.xsd',
schemaDiff, 1);
end;
/

```



4. Try inserting again the a new XML document conforming to the new version of the XML schema. Execute the script **insertNewXMLV2.sql**. Observe that there are no errors this time.

```

@insertNewXMLV2.sql

declare
new_xml xmltype;
res boolean;
begin
select column_value into new_xml
from xmltable('for $i in ora:view("PURCHASEORDER")
where $i/PurchaseOrder/Reference = "SBELL-2002100912333601PDT"
return $i');
select appendChildXML (new_xml,
'/PurchaseOrder/LineItems/LineItem[@ItemNumber="1"]',
xmltype('<Unit>Box</Unit>'))
into new_xml from dual;
select appendChildXML (new_xml,
'/PurchaseOrder/LineItems/LineItem[@ItemNumber="2"]',
xmltype('<Unit>Carton</Unit>'))

```

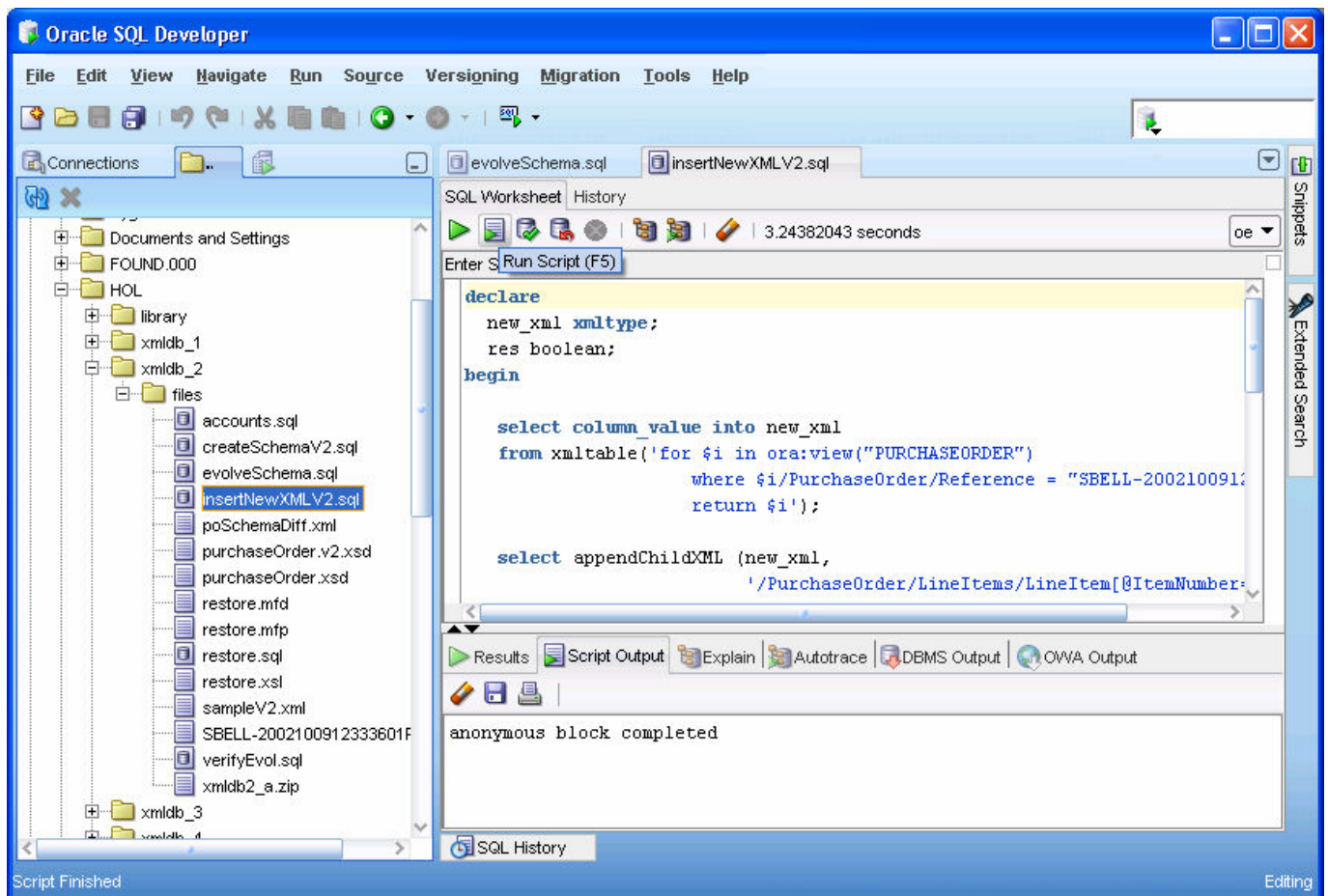
```

into new_xml from dual;
select appendChildXML (new_xml,
'/PurchaseOrder/LineItems/LineItem[@ItemNumber="3"]',
xmltype('<Unit>Case</Unit>'))
into new_xml from dual;
select updateXML (new_xml,
'/PurchaseOrder/Reference/text()',
'SBELL-2002100912333601PDT-V2')
into new_xml from dual;
insert into PURCHASEORDER values new_xml;

commit;

end;
/

```



5. Verify that the new XML document has been successfully inserted. Execute **verifyEvol.sql**.

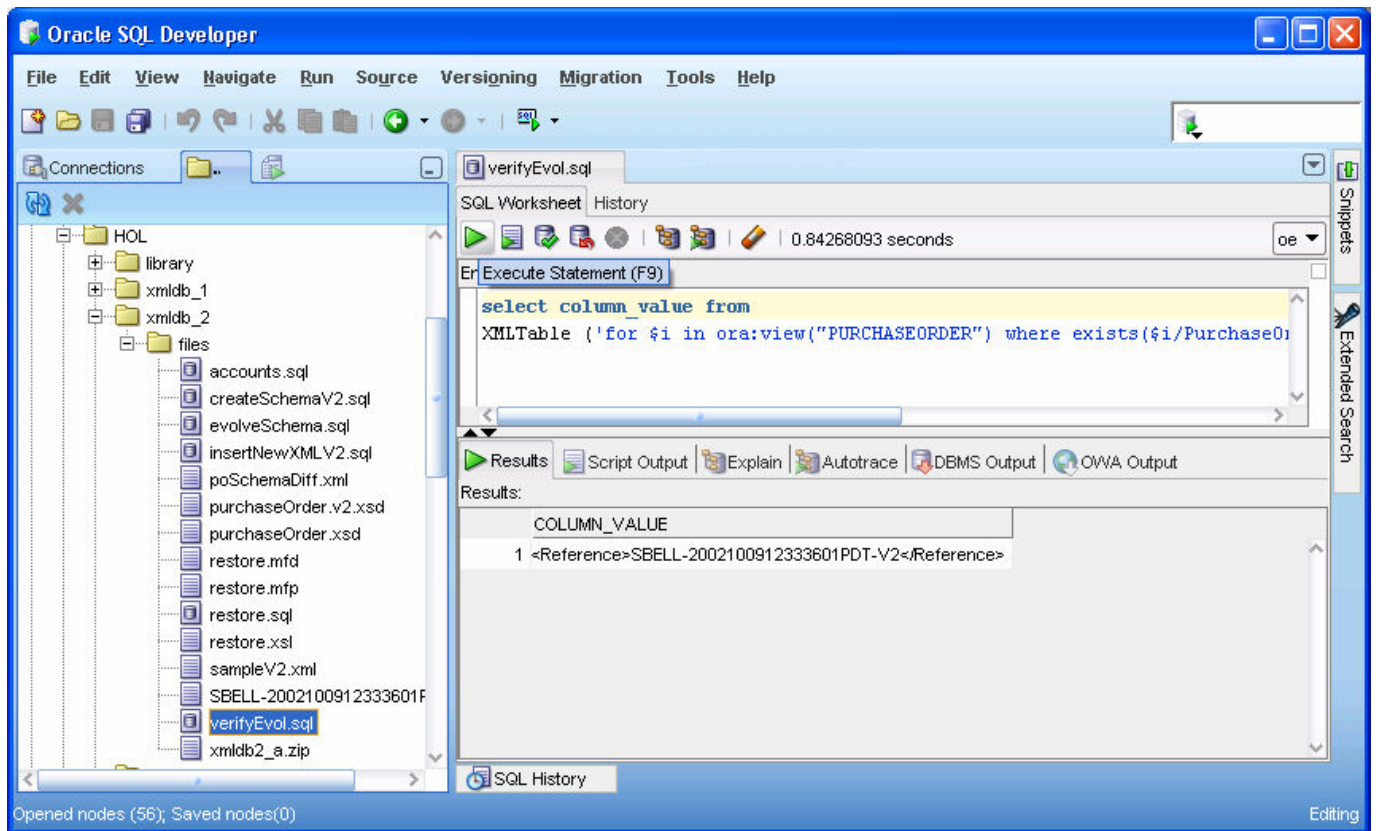
@verifyEvol.sql

```

select column_value from
XMLTable ('for $i in ora:view("PURCHASEORDER") where exists($i/PurchaseOrder/LineItems/LineItem/Unit)
return $i/PurchaseOrder/Reference');

```





## Summary

In this tutorial, you learned how to use in-place XML schema evolution.

# Lesson 3: Using Oracle XML DB Web Services for Service-Oriented Architecture

## Purpose






This tutorial also shows you how to use Oracle XML DB Web services for Service-Oriented Architecture.

## Time to Complete

Approximately 20 minutes.

## Topics

This tutorial covers the following topics:

-  [Overview](#)
-  [Prerequisites](#)
-  [Creating a Binary XML Table](#)
-  [Using Oracle XML DB Web Services for Service-Oriented Architecture](#)
-  [Summary](#)

## Overview

Since Oracle 9i Database Release 2, Oracle XML DB has been seamlessly integrated with the Oracle database to provide high-performance database-native storage, retrieval, and management of XML data. With the new Oracle Database 11g release, Oracle XML DB is taking another leap ahead with a rich set of new capabilities to simplify DBAs' tasks in managing XML data while further empowering XML and SOA application developers. Oracle XML DB now supports multiple database-native XML storage models and XML indexing schemes, SQL/XML standard operations, W3C standard XQuery data model and XQuery/XPath languages, In-place XML schema evolution, database-native web services, high performance XML publishing, XML DB repository, and versioning and access control. This tutorial covers using Oracle XML DB Web services for Service-Oriented Architecture.

## Oracle XML DB Web Services

Web Services have become an industry- standard way for both exchanging information and giving access to business logic. The new Oracle XML DB web services capability provides access to SQL, XQuery, PL/SQL, and other database elements as a web service. For example, one web service allows users to issue SQL and XQuery queries and receive the results as XML, and another can provide access to all PL/SQL functions and procedures stored inside the DB as web services.

## Prerequisites

Before you perform this tutorial, you should first complete the following steps:

1. Check your Oracle Database 11g installation and make sure the **OE**, **HR** users are unlocked.
2. Check your Oracle SQL Developer (version 1.5.1) installation
3. Check your Oracle Jdeveloper 10.1.3.4 installation
4. Check the files in your working directory (/HOL/xmlldb\_3/files)

**Note:** If you use an earlier version of Oracle JDeveloper, the screenshots may slightly differ.



## Creating a Binary XML Table

In this tutorial, you use the SQL Developer tool. After creating a database connection, you set the script pathing reference in SQL Developer. You create a binary XML table and insert rows with data selected from the `PURCHASEORDER` table.

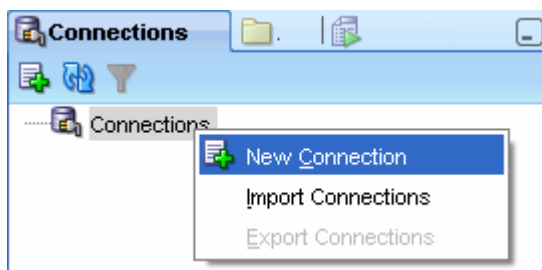
Perform the following steps:

1. Click on the SQL Developer icon on the desktop to start the application.



2. In SQL Developer, you must create a database connection as OE user. Perform the following steps.

- a. In the Connections tab, right-click **Connections** and select **New Connection**.



- b. The New/Select Database Connection window appears. Enter the following details, and click **Test** to make sure that the connection has been set correctly.

Connection Name: **oe**  
UserName: **oe**  
Password: **oe**  
Hostname: **localhost**  
Port: **1521**  
SID: **orcl**

If you select the Save Password check box, the password is saved to an XML file. Therefore, once you close SQL Developer connection and open again, you will not be prompted for the password.

**New / Select Database Connection**

Connection Name	Connection Details
system	system@Mocalho...

Connection Name:

Username:

Password:

☒ Save Password

Oracle Access

Role:  ☐ OS Authentication

Connection Type:  ☐ Proxy Connection

Hostname:

Port:

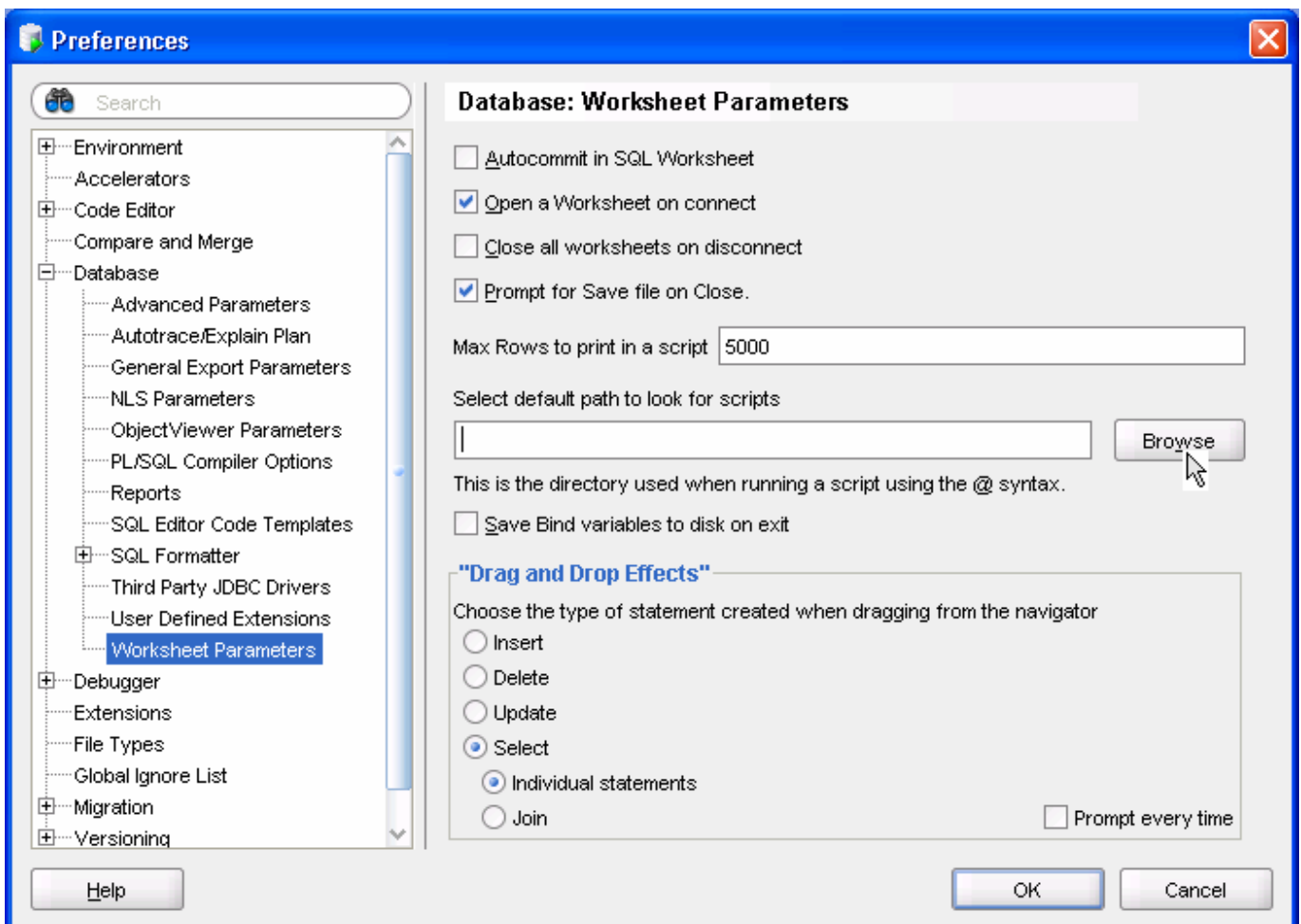
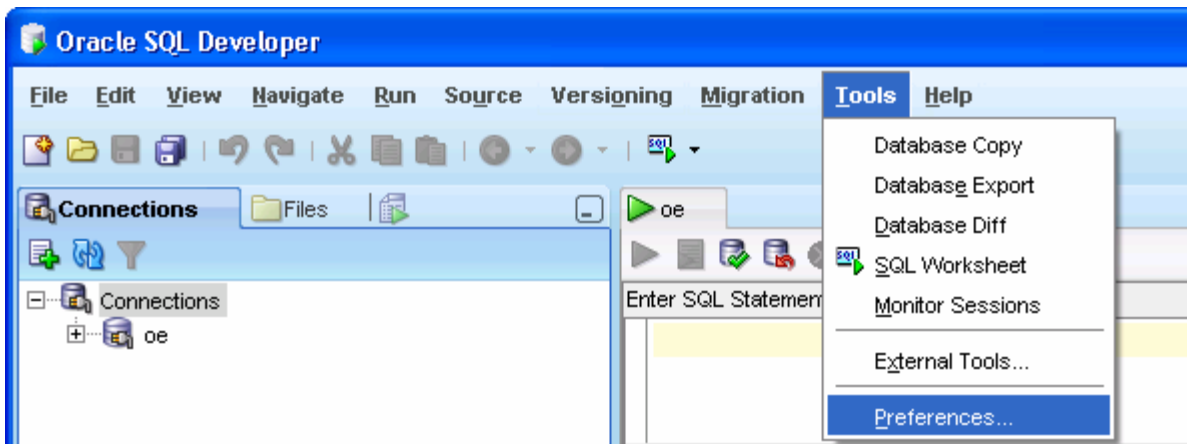
☒ SID ☐ Service name

Status : Success

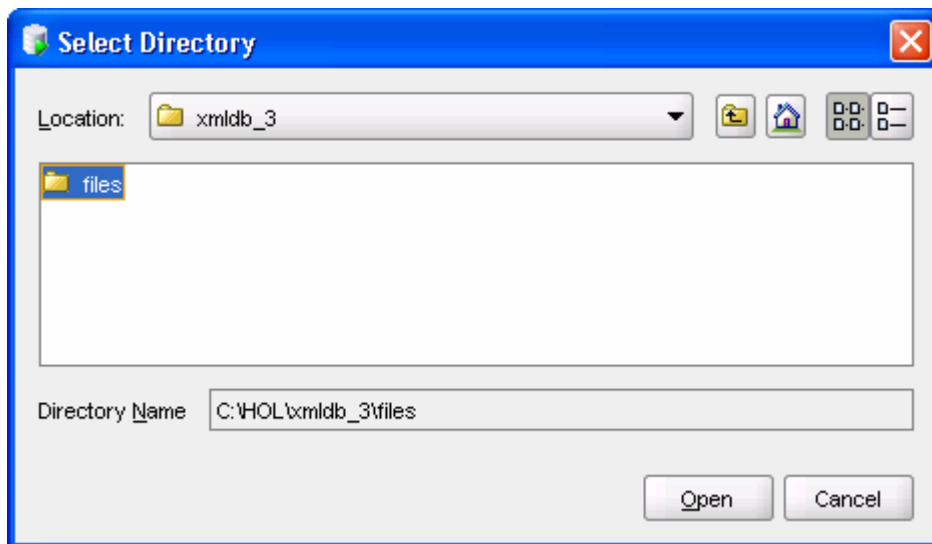
c. The test status shows success. Click **Connect**.

3. To run the scripts by using the @ syntax, you can set the script pathing reference in SQL Developer. Perform the following steps:

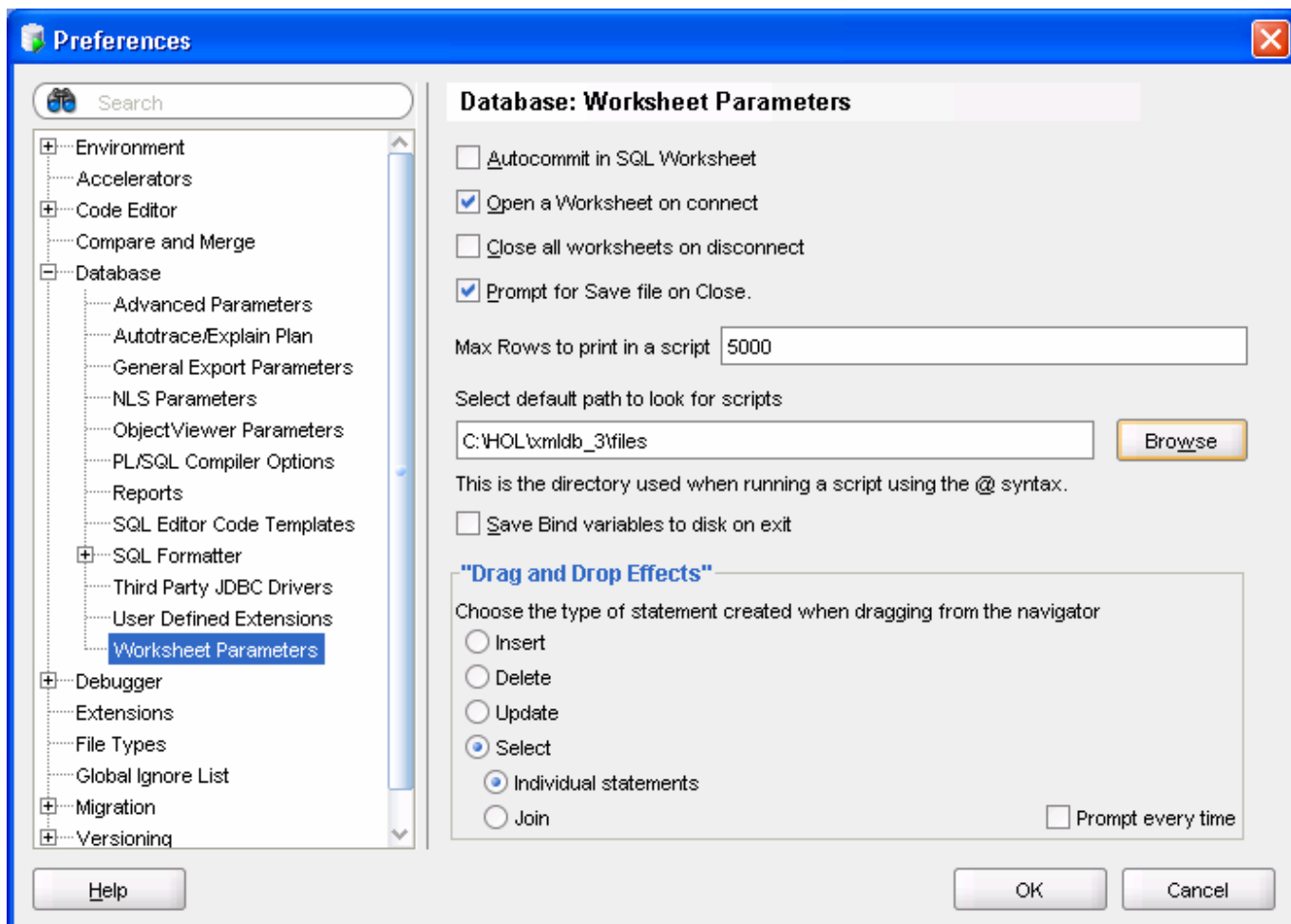
a. Select **Tools > Preferences > Database > Worksheet Parameters**. Then, click **Browse**.



b. Browse to the location of your working directory that has the SQL scripts. Then, click **Open**.



c In the Preferences window, verify the script path in the Select default path to look for scripts field. Click **OK**.



4. Run the script `create_sl_bix_table.sql`.

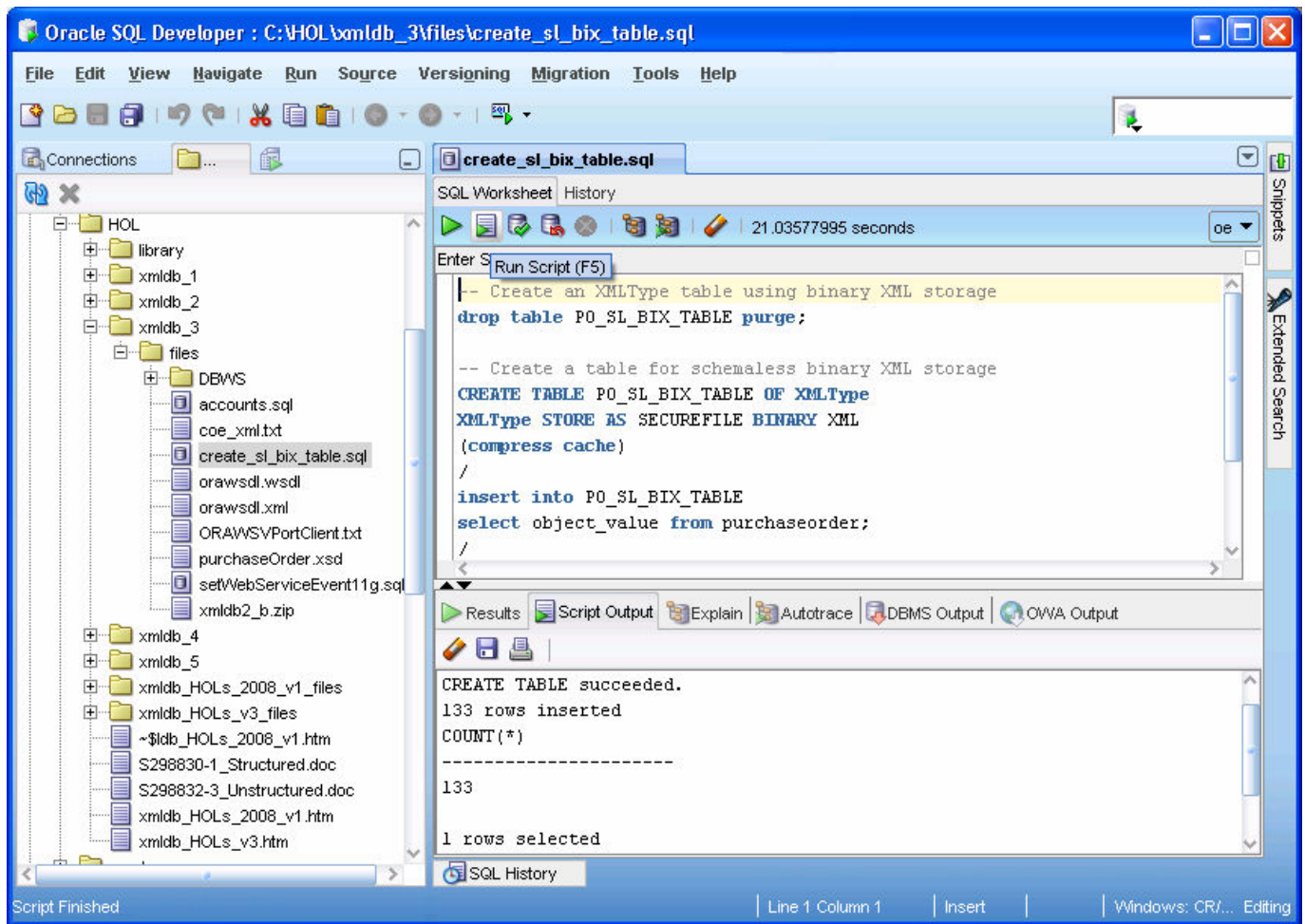
```
@create_sl_bix_table.sql

DROP TABLE po_sl_bix_table PURGE;

CREATE TABLE PO_SL_BIX_TABLE OF XMLType
XMLType STORE AS SECUREFILE BINARY XML
(compress cache)
/
INSERT INTO po_sl_bix_table
SELECT object_value FROM purchaseorder;
/

SELECT COUNT(*) FROM po_sl_bix_table
/

COMMIT
/
```



In the above section, you learned how to connect to SQL Developer and set script pathing reference. You also learned how to create an `XMLType` table with binary XML storage, and populate the table with data.

## Using Oracle XML DB Web Services for Service-Oriented Architecture

You can implement a service-oriented architecture using a new web service endpoint native to Oracle XML DB. Perform the following tasks:

[Enable the database-native Web service end point in Oracle XMLDB](#)

[Develop a Web service client application](#)

[View Web service request and response using Http Analyzer](#)

### Enabling the database-native Web service end point in Oracle XML DB

1. Open a terminal window and enter the following commands:

```
cd HOL\xmlldb_3\files
sqlplus sys/oracle as sysdba
```

2. Enable the database-native Web service endpoint in Oracle XML DB. Execute the following script:

**@setWebServiceEvent11g.sql**

```
create or replace procedure addServletMapping (pattern varchar2,
servletname varchar2,
dispname varchar2,
servletclass varchar2,
servletschema varchar2,
language varchar2,
description varchar2,
securityRole xmltype) as
xdbcconfig xmltype;
begin
xdbcconfig := dbms_xdb.cfg_get();
select deleteXML
(
xdbcconfig,
'/xdbcconfig/sysconfig/protocolconfig/httpconfig/webappconfig/servletconfig/servlet-
list/servlet[servlet-name=' || servletname || ']'
)
into xdbcconfig
from dual;

if (language = 'C') then
select insertChildXML
(
xdbcconfig,
'/xdbcconfig/sysconfig/protocolconfig/httpconfig/webappconfig/servletconfig/servlet-list',
'servlet',
xmlElement
(
"servlet",
xmlAttributes('http://xmlns.oracle.com/xdb/xdbcconfig.xsd' as "xmlns"),
xmlForest
(
servletname as "servlet-name",
language as "servlet-language",
dispname as "display-name",
description as "description"
),
securityRole
)
)
into xdbcconfig
```

```

from dual;
else
select insertChildXML
(
xdbcconfig,
'/xdbcconfig/sysconfig/protocolconfig/httpconfig/webappconfig/servletconfig/servlet-list',
'servlet',
XMLElement
(
"servlet",
xmlAttributes('http://xmlns.oracle.com/xdm/xdmconfig.xsd' as "xmlns"),
xmlForest
(
servletname as "servlet-name",
language as "servlet-language",
dispname as "display-name",
description as "description",
servletclass as "servlet-class",
servletschema as "servlet-schema"
)
)
)
into xdbcconfig
from dual;
end if;

select deleteXML
(
xdbcconfig,
'/xdbcconfig/sysconfig/protocolconfig/httpconfig/webappconfig/servletconfig/servlet-mappings/servlet-
mapping[servlet-name="' || servletname || '"]'
)
into xdbcconfig
from dual;

select insertChildXML
(
xdbcconfig,
'/xdbcconfig/sysconfig/protocolconfig/httpconfig/webappconfig/servletconfig/servlet-mappings',
'servlet-mapping',
xmltype
(
'<servlet-mapping xmlns="http://xmlns.oracle.com/xdm/xdmconfig.xsd">
<servlet-pattern>' || pattern || '</servlet-pattern>
<servlet-name>' || servletname || '</servlet-name>
</servlet-mapping>'
)
)
into xdbcconfig
from dual;

dbms_xdb.cfg_update(xdbcconfig);

end;
/

call addServletMapping(
'/orawsv/*',
'orawsv',
'Oracle Query Web Service',
null,
null,
'C',
'Web Services Servlet',
xmltype(
'<security-role-ref>
<role-name>XDB_WEBSERVICES</role-name>'
)
)

```



```

<role-link>XDB_WEBSERVICES</role-link>
</security-role-ref>'
)
)
/

call addServletMapping(
'/orawsdl/*',
'orawsdl',
'Oracle WSDLs',
null,
null,
'C',
'WSDL Servlet',
xmltype(
'<security-role-ref>
<role-name>XDBWEBSERVICES</role-name>
<role-link>XDBWEBSERVICES</role-link>
</security-role-ref>'
)
)
/

grant XDB_WEBSERVICES to oe
/

-- For 11g only
grant XDB_WEBSERVICES_OVER_HTTP to oe
/
grant XDB_WEBSERVICES_WITH_PUBLIC to oe
/

-- Clean up afterward
drop procedure addServletMapping
/

```

```
Command Prompt - sqlplus sys/oracle as sysdba
C:\H01\xmldb_3\files>
C:\H01\xmldb_3\files>sqlplus sys/oracle as sysdba

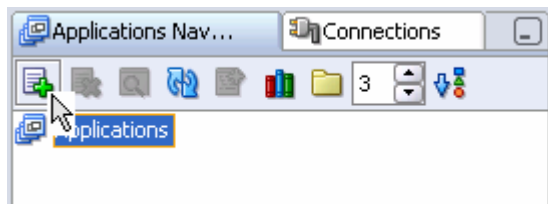
SQL*Plus: Release 11.1.0.6.0 - Production on Sat Aug 30 17:48:35 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

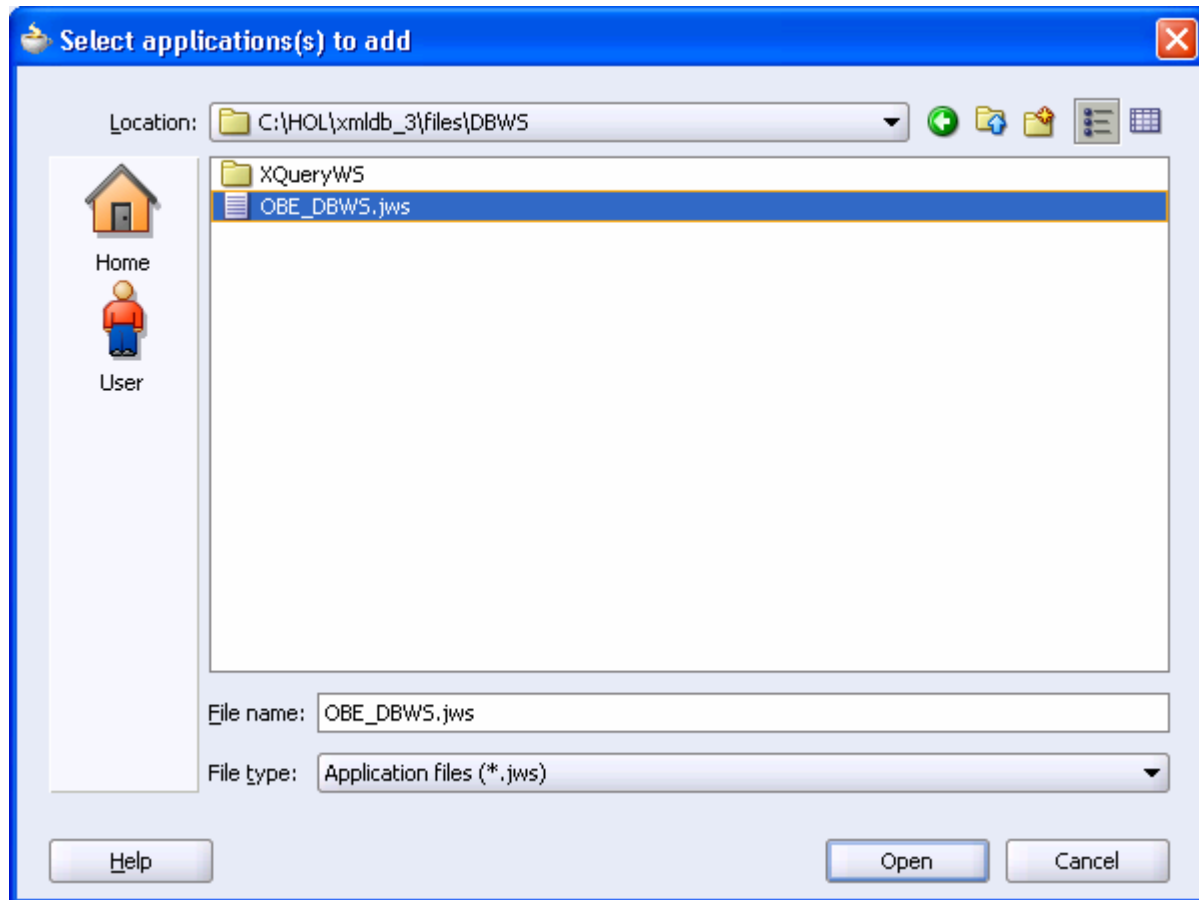
SQL> @setWebServiceEvent11g.sql
SQL> -- conn sys/oracle as sysdba
SQL>
SQL> create or replace procedure addServletMapping (pattern varchar2,
2          servletname varchar2,
3          dispname varchar2,
4          servletclass varchar2,
5          servletschema varchar2,
6          language varchar2,
7          description varchar2,
8          securityRole xmltype) as
9     xdbconfig xmltype;
10  begin
11     xdbconfig := dbms_xdb.cfg_get();
12     select deleteXML
13     <
14         xdbconfig,
15         '/xdbconfig/sysconfig/protocolconfig/httpconfig/webappconfig/
servletconfig/servlet-list/servlet[servlet-name="' || servletname || '"]'
16     >
    >
```

## Developing a Web service client Application

1. Switch to JDeveloper. In the navigator window, click the **Add to Applications** icon.

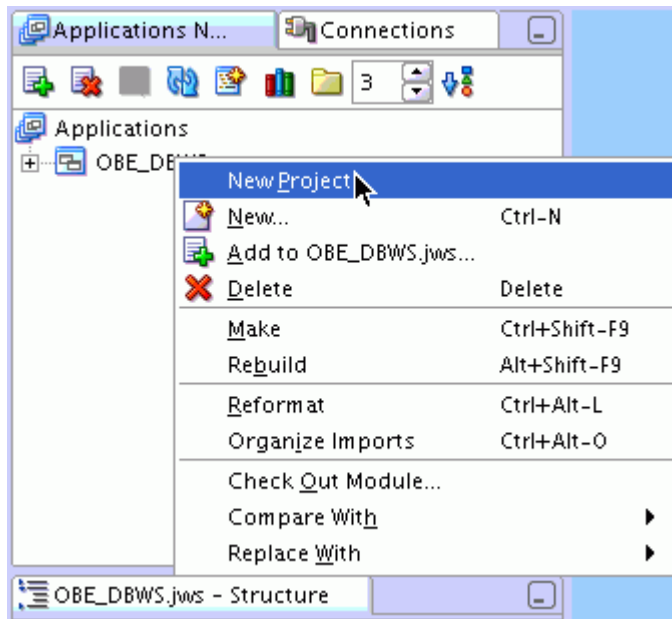


2. In the **Select application(s) to add** window, go to your **HOL\xmlb\_3\files\DBWS** directory. Then, select **OBE\_DBWS.jws**, and click **Open**.

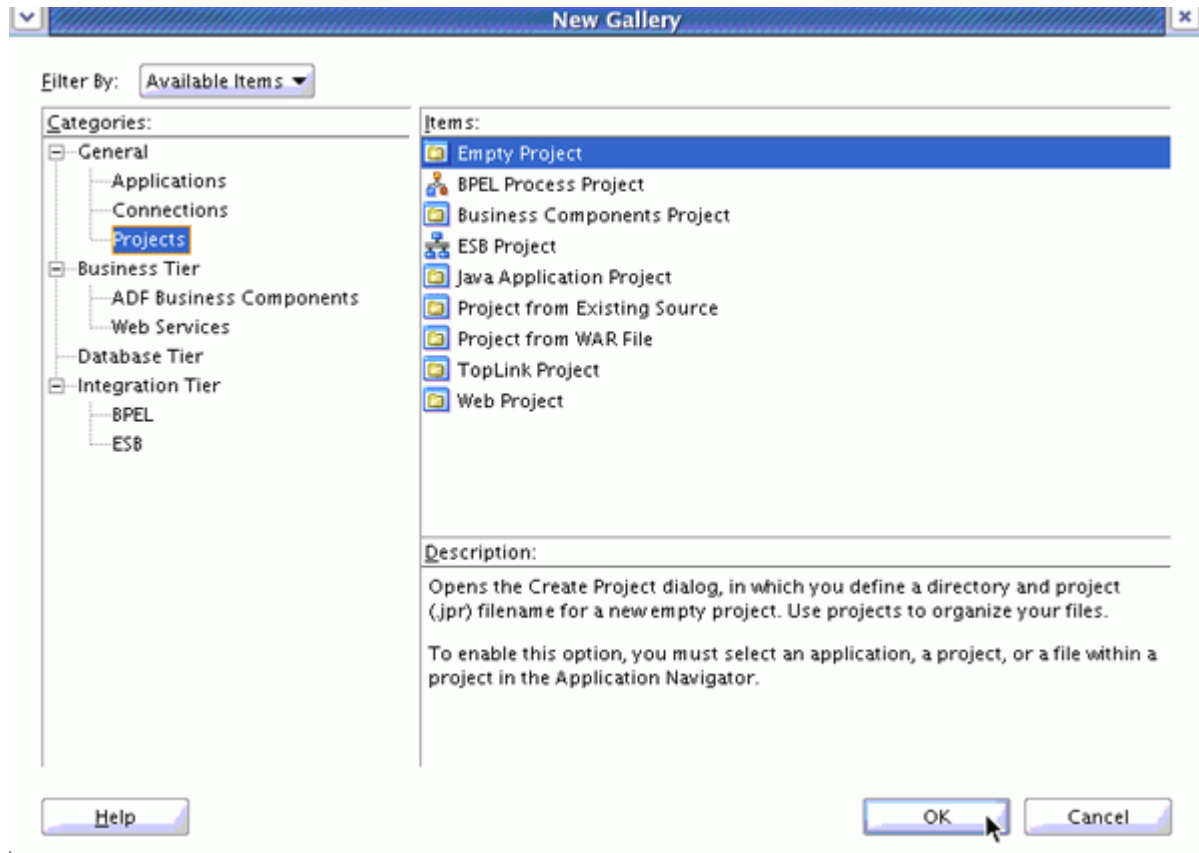


At this point, if you see a warning window, click **Yes**. You may see this warning because of the application migration to the latest JDeveloper version file format.

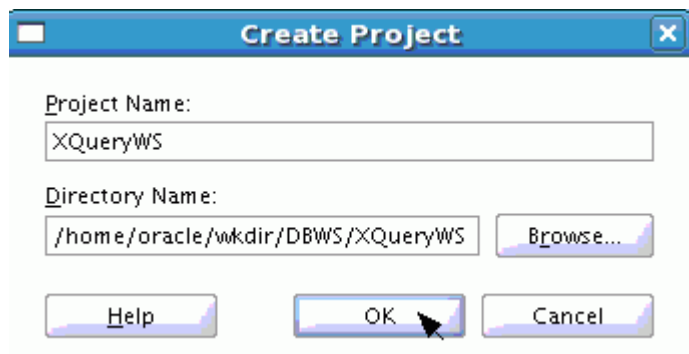
3. Right-click **OBE\_DBWS**, and select **New Project...**



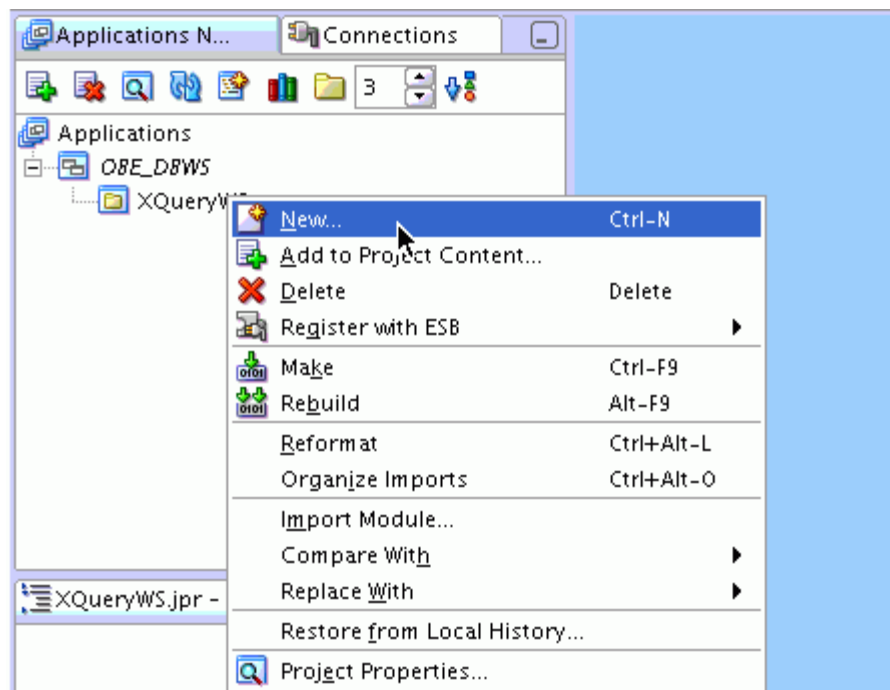
4. The New Gallery window appears. From the Items list, select **Empty Project**, and click **OK**.



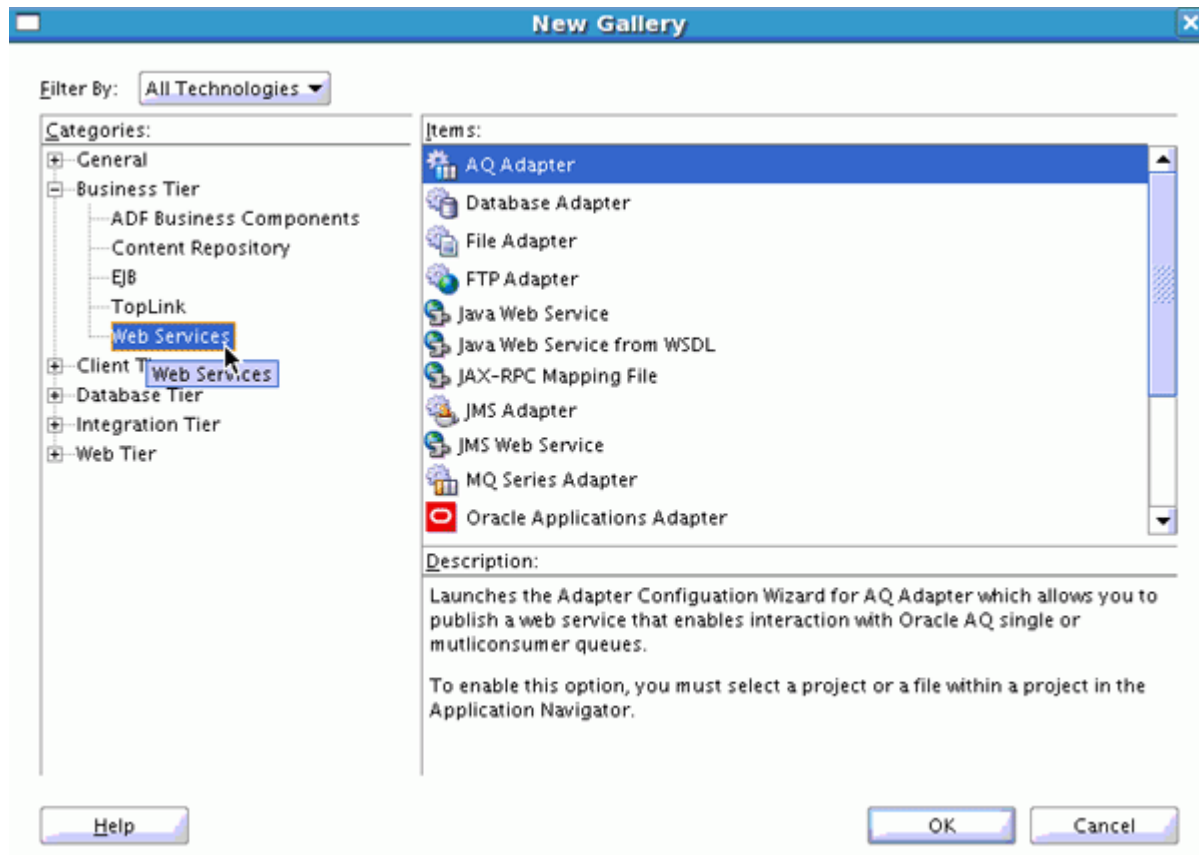
5. In the Create Project window, enter **XQueryWS** for Project Name. Accept the default Directory Name, and click **OK**.  
Note: The default directory name will be <working directory>/DBWS/XQueryWS.



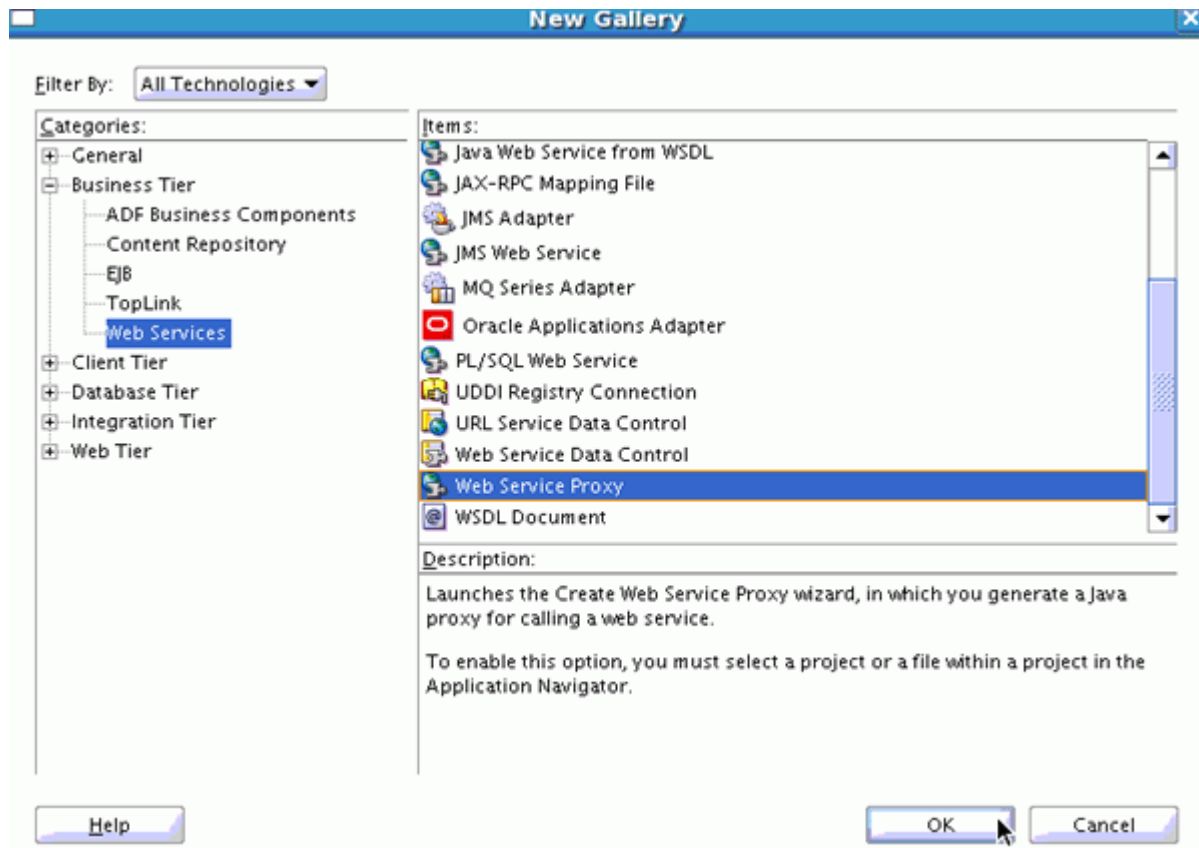
6. Expand **OBE\_DBWS**. Right-click **XQueryWS**, and select **New...**



7. In the Categories section, Expand **Business Tier**, and select **Web Services**.

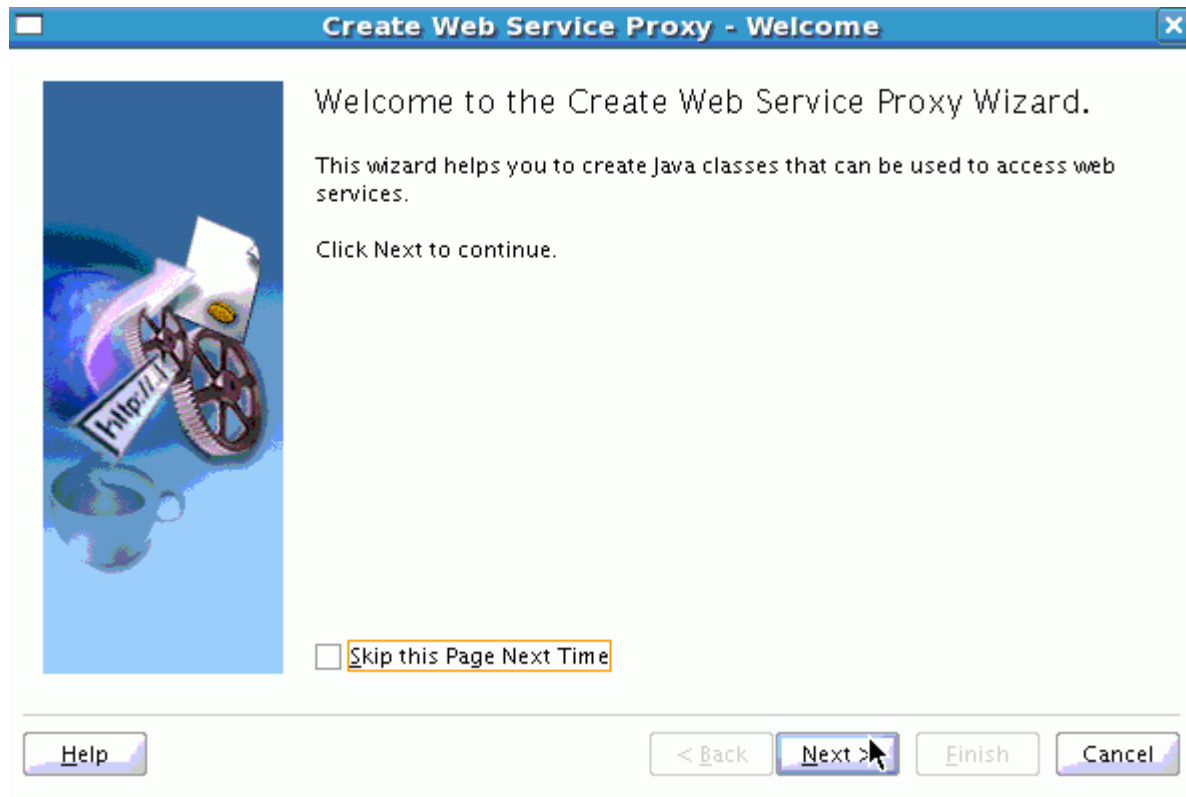


8. The list of Items is displayed in the right window pane. Select **Web Service Proxy**, and click **OK**.





- 9 This launches a Create Web Service Proxy wizard, in which you generate a java proxy for calling the web service. Click **Next**.



10. You should select a WSDL document containing service descriptions, and then choose service in the document to create a proxy for. Browse in your **<working directory>**, and select **orawSDL.xml** file. Then, click **Open**.

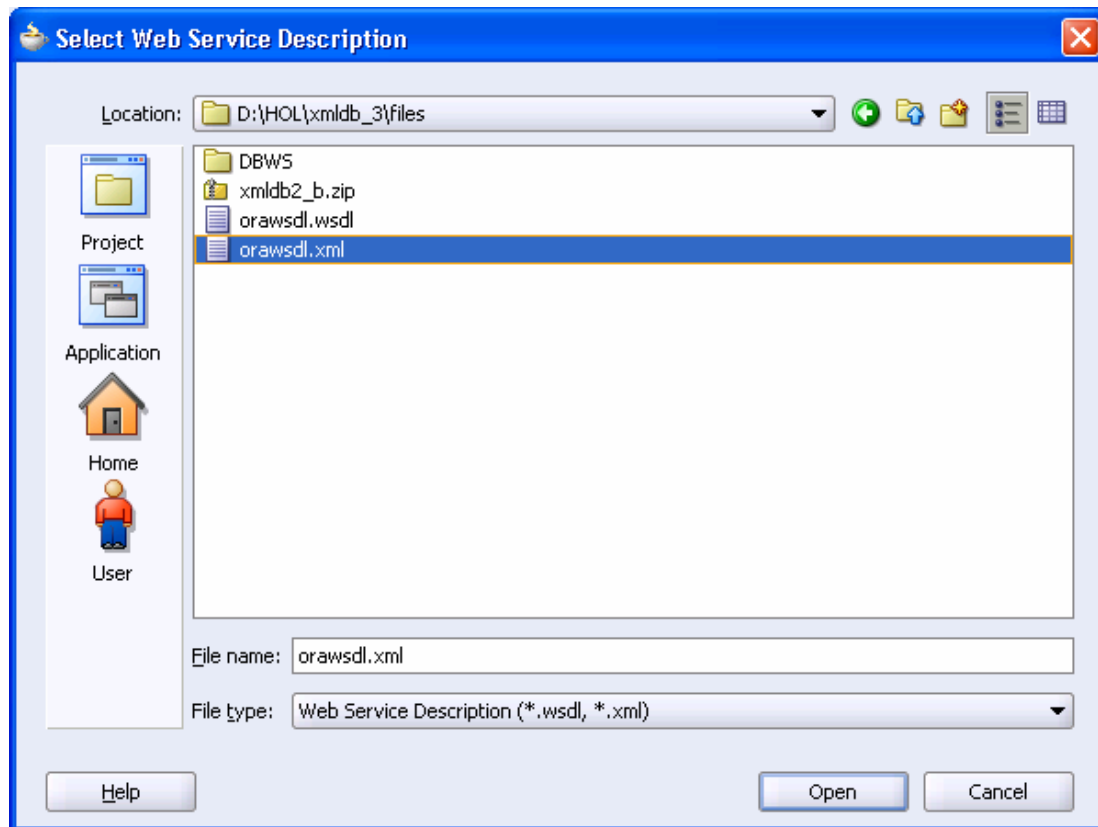
**Create Web Service Proxy - Step 1 of 7: Web Service Description**

Select a WSDL document containing service descriptions, and then choose the service in the document to create a proxy for. You can also optionally select a mapping file for use when mapping from WSDL to Java.

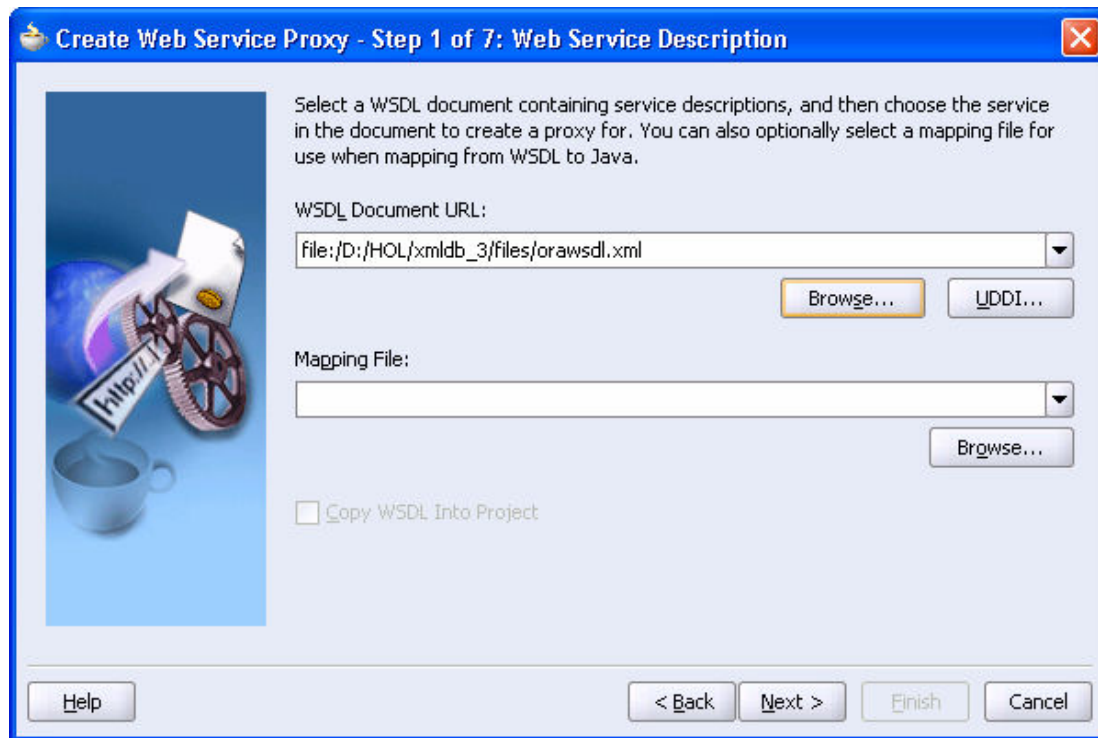
WSDL Document URL:

Mapping File:

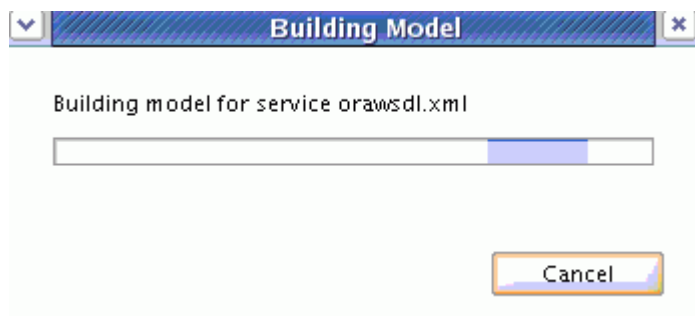
☐ Copy WSDL Into Project

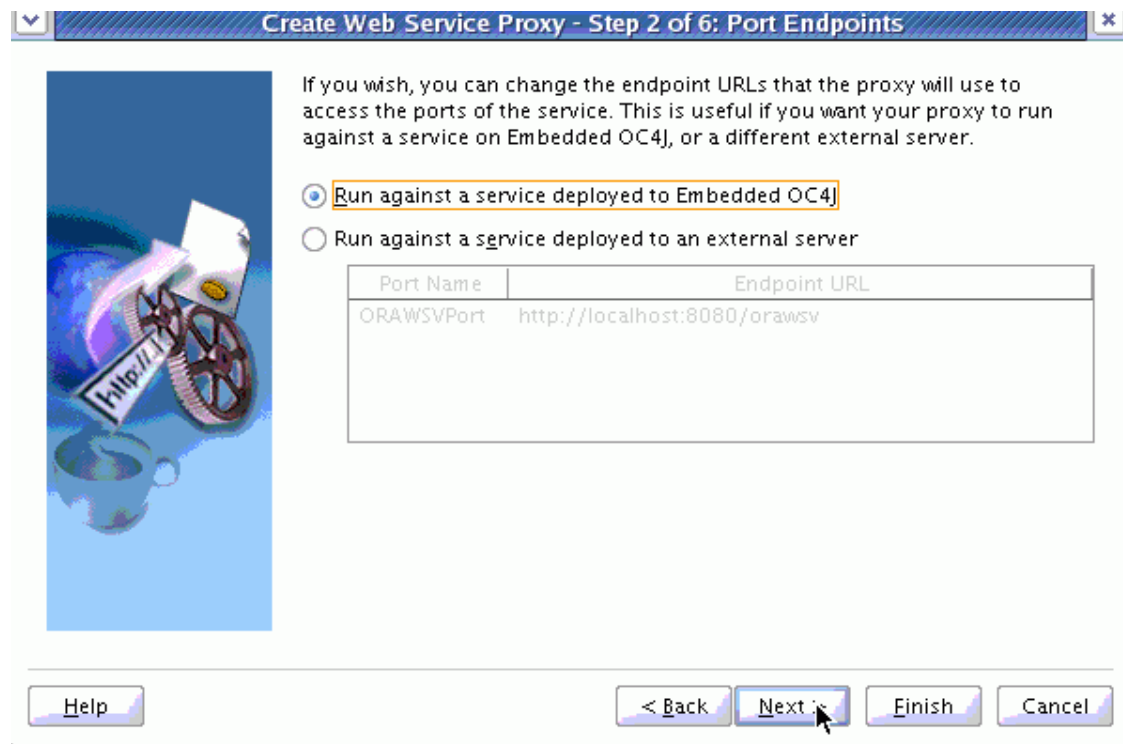


11. Click **Next**.

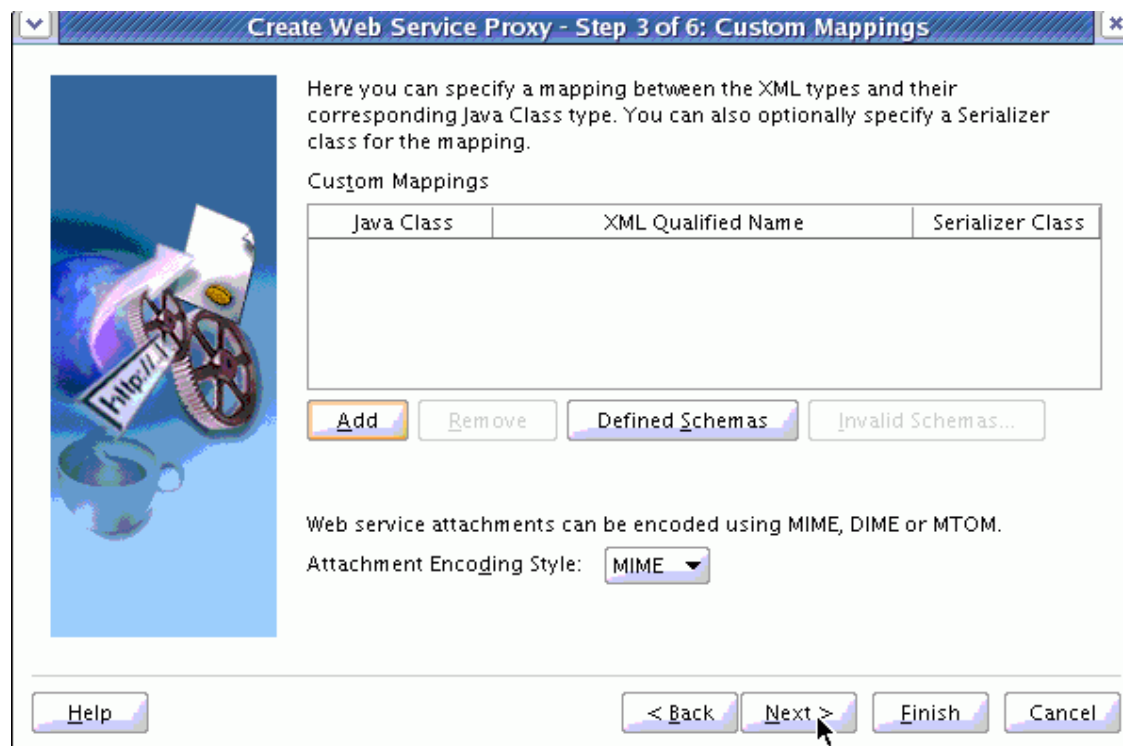


12. The Building Model window pops up for a while and shows the building model status for service **orawsdl.xml**. The next step is to select the endpoint URL that the proxy will use to accept the ports of the service. Accept the default selection of the radio button **Run against a service deployed to Embedded OC4J**. Click **Next**.

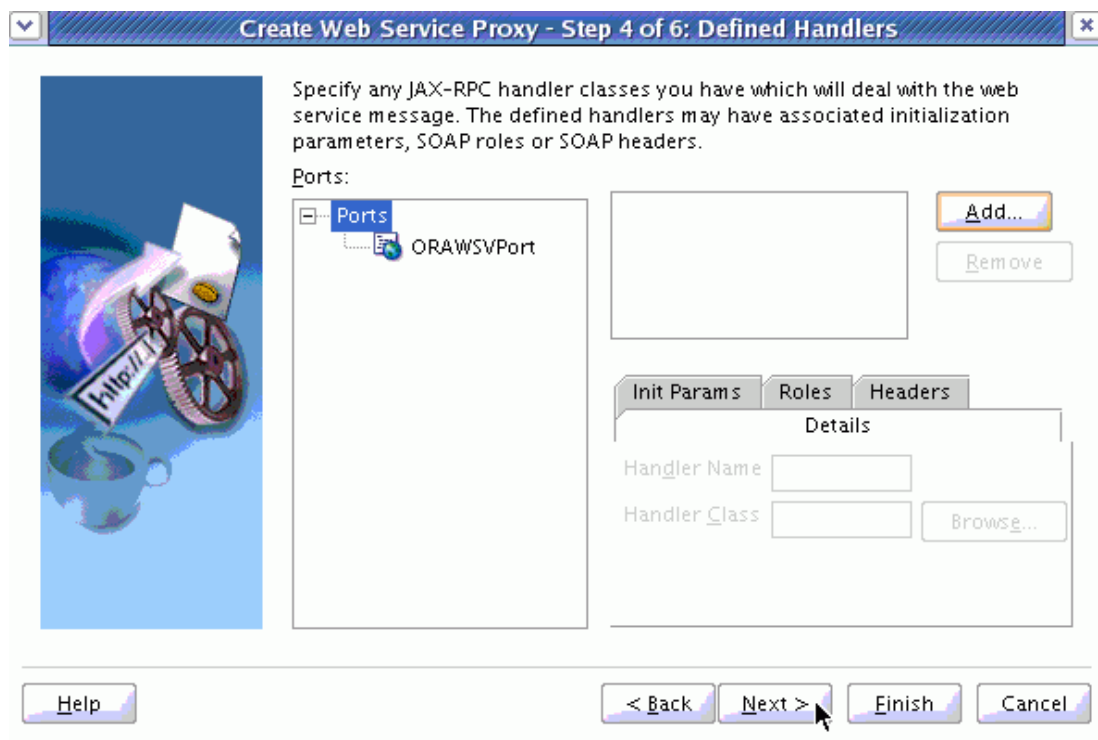




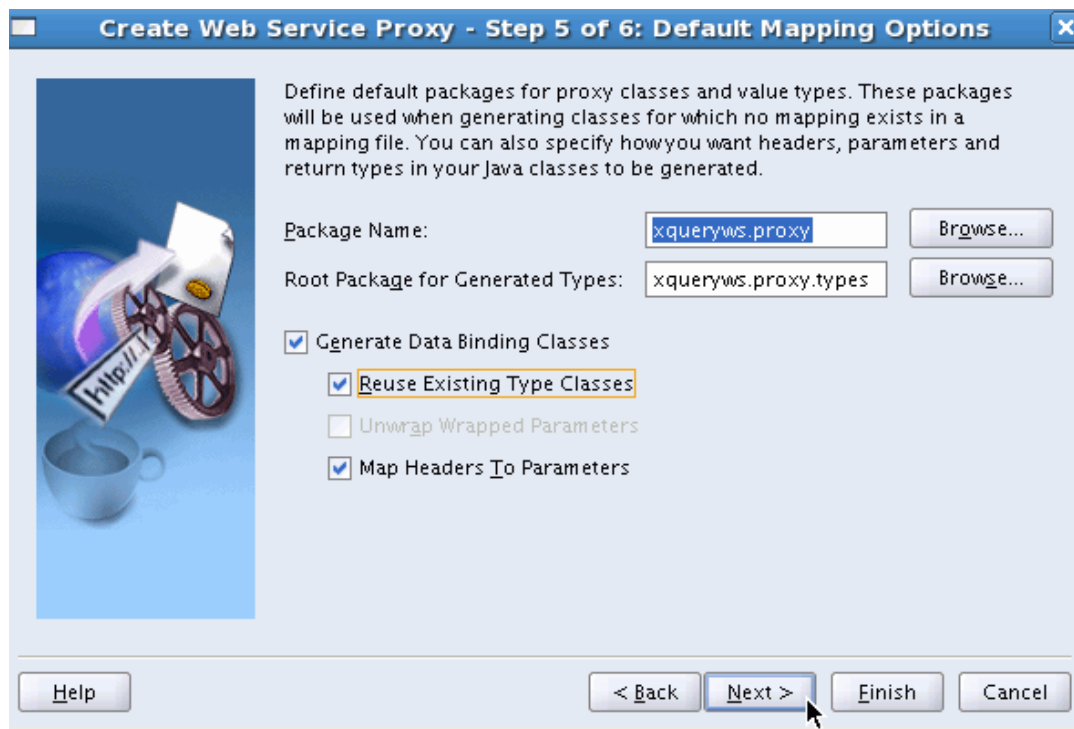
13. Click **Next**.



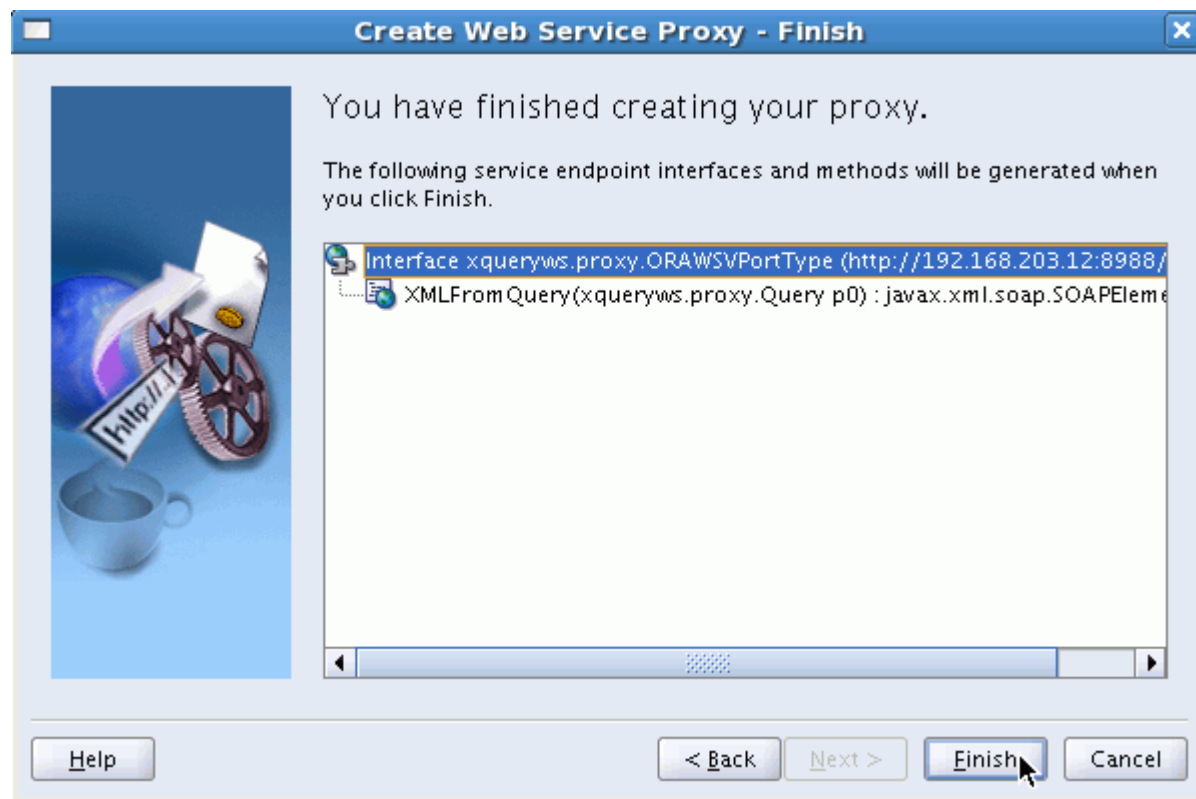
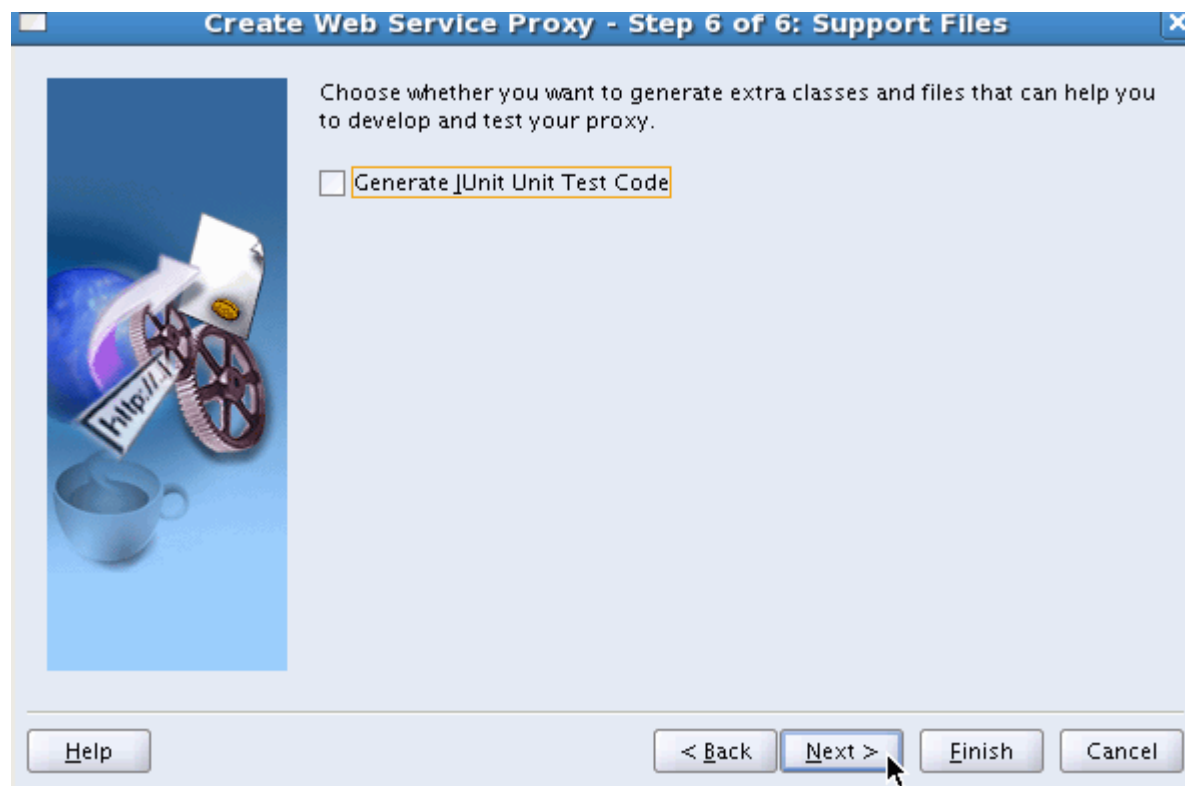
14. In this step, you see ORAWSVPort under the Ports node. Click **Next**.



15. Accept the default mapping options, and select the **Reuse Existing Type Classes** check box. Click **Next**.



16. Click **Next**. Then, click **Finish**. You have finished creating your proxy.



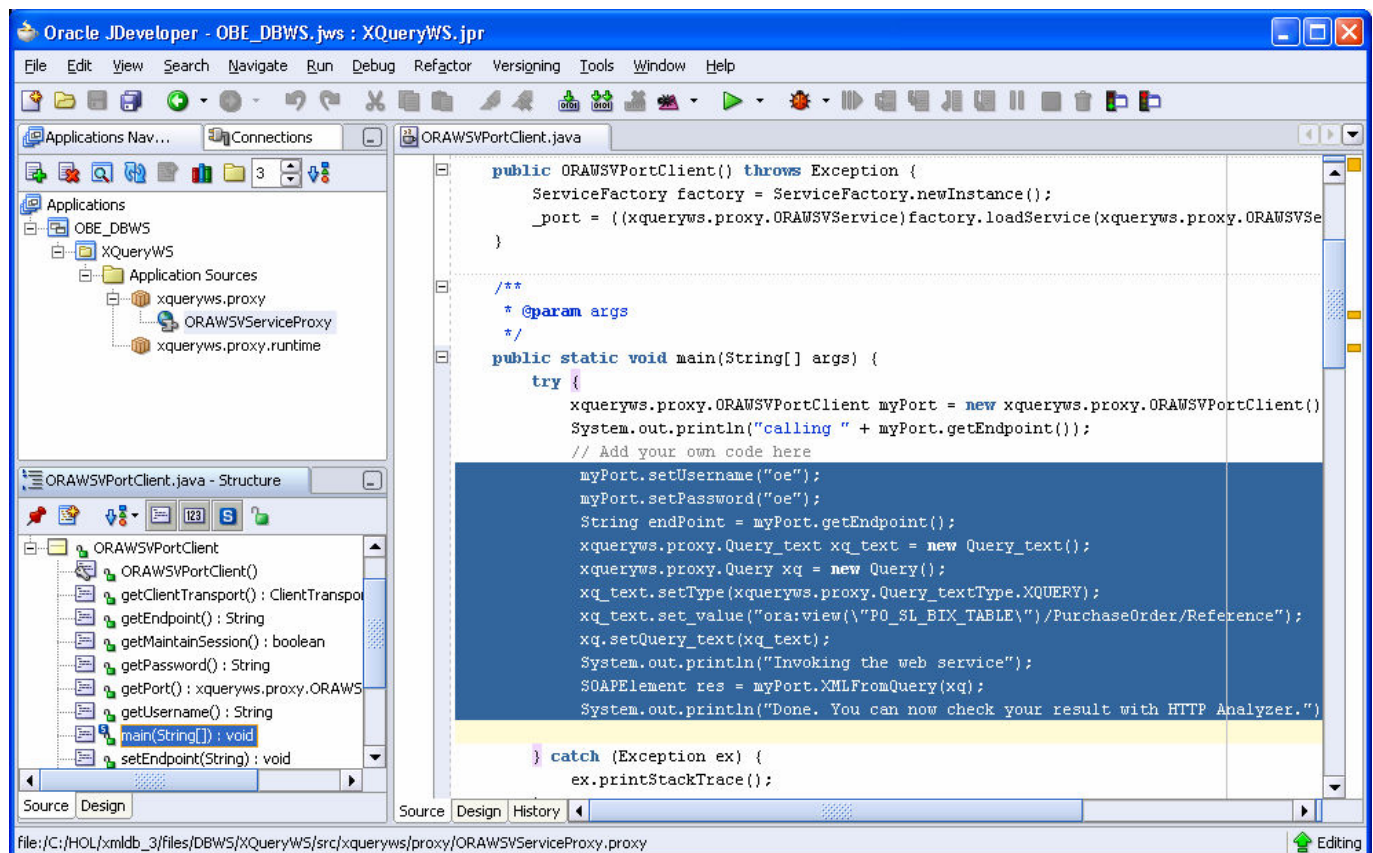
Now, you see the **Generation in Progress** window.



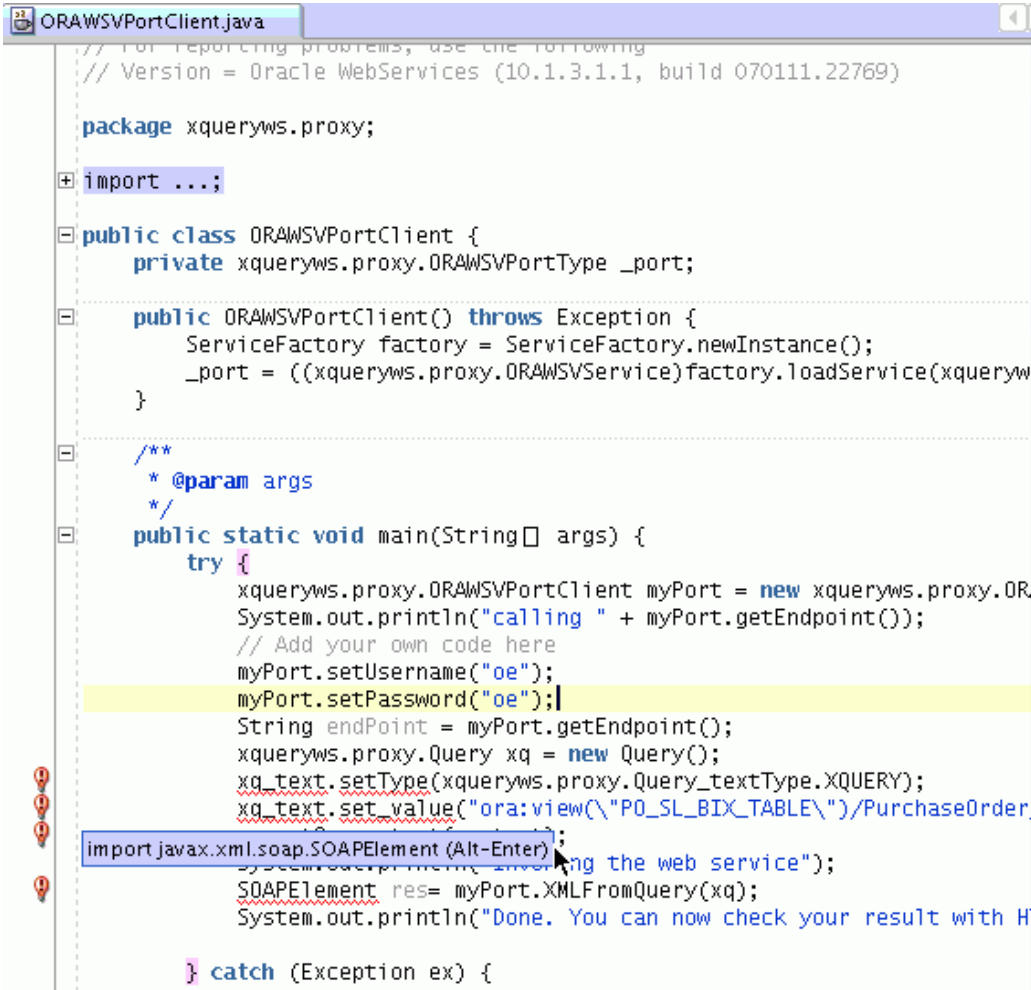
17. JDeveloper generates code for invoking web services defined in the WSDL file. In this lesson, the main client program file is the ORAWSVPortClient.java.

Open the file ORAWSVPortClient.txt. Use a text editor to copy the content of this file. In the JDeveloper window, paste this code after the line that reads **//Add your own code here**.

```
myPort.setUsername("oe");
myPort.setPassword("oe");
String endPoint = myPort.getEndpoint();
xqueryws.proxy.Query_text xq_text = new Query_text();
xqueryws.proxy.Query xq = new Query();
xq_text.setType(xqueryws.proxy.Query_textType.XQUERY);
xq_text.set_value("ora:view(\"PO_SL_BIX_TABLE\")/PurchaseOrder/Reference");
xq.setQuery_text(xq_text);
System.out.println("Invoking the web service");
SOAPElement res = myPort.XMLFromQuery(xq);
System.out.println("Done. You can now check your result with HTTP Analyzer.");
```



18. To compile the code successfully, you should import `javax.xml.soap.SOAPElement`. At this point, you receive a code assist in JDeveloper to import `javax.xml.soap.SOAPElement`. Press **Alt-Enter** to import.



```
// For reporting problems, use the following
// Version = Oracle WebServices (10.1.3.1.1, build 070111.22769)

package xqueryws.proxy;

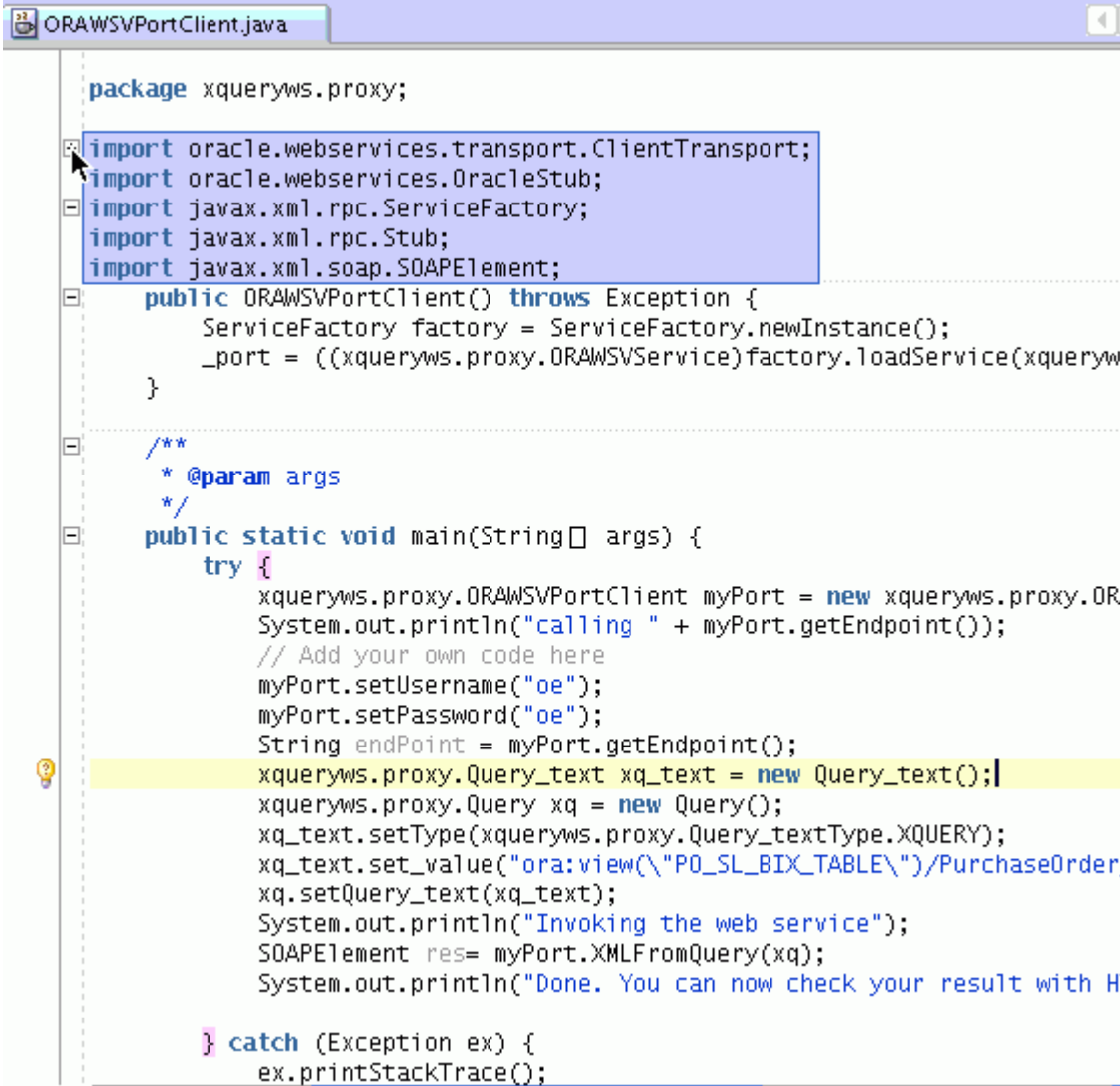
import ...;

public class ORAWSVPortClient {
    private xqueryws.proxy.ORAWSVPortType _port;

    public ORAWSVPortClient() throws Exception {
        ServiceFactory factory = ServiceFactory.newInstance();
        _port = ((xqueryws.proxy.ORAWSVService)factory.loadService(xqueryws.proxy.
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        try {
            xqueryws.proxy.ORAWSVPortClient myPort = new xqueryws.proxy.ORAWSVPortClient();
            System.out.println("calling " + myPort.getEndpoint());
            // Add your own code here
            myPort.setUsername("oe");
            myPort.setPassword("oe");
            String endPoint = myPort.getEndpoint();
            xqueryws.proxy.Query xq = new Query();
            xq_text.setType(xqueryws.proxy.Query_textType.XQUERY);
            xq_text.set_value("ora:view(\"PO_SL_BIX_TABLE\")/PurchaseOrder");
            import javax.xml.soap.SOAPElement (Alt-Enter);
            System.out.println("Invoking the web service");
            SOAPElement res= myPort.XMLFromQuery(xq);
            System.out.println("Done. You can now check your result with H");
        } catch (Exception ex) {
```

19. Now, expand the **import...**, node. You see the `javax.xml.soap.SOAPElement` added.



```
package xqueryws.proxy;

import oracle.webservices.transport.ClientTransport;
import oracle.webservices.OracleStub;
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.Stub;
import javax.xml.soap.SOAPElement;

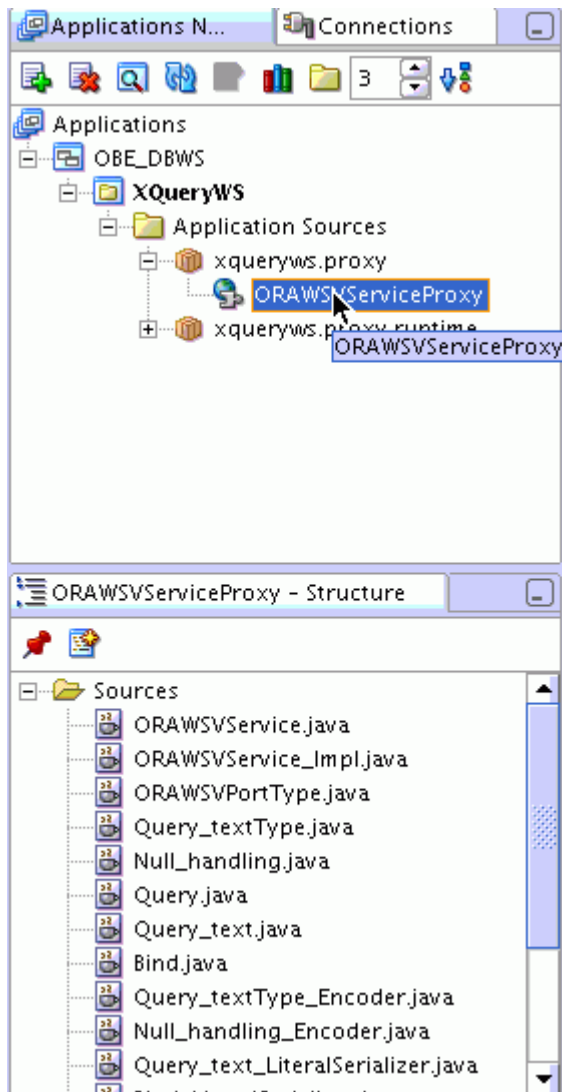
public ORAWSVPortClient() throws Exception {
    ServiceFactory factory = ServiceFactory.newInstance();
    _port = ((xqueryws.proxy.ORAWSVService)factory.loadService(xqueryw

}

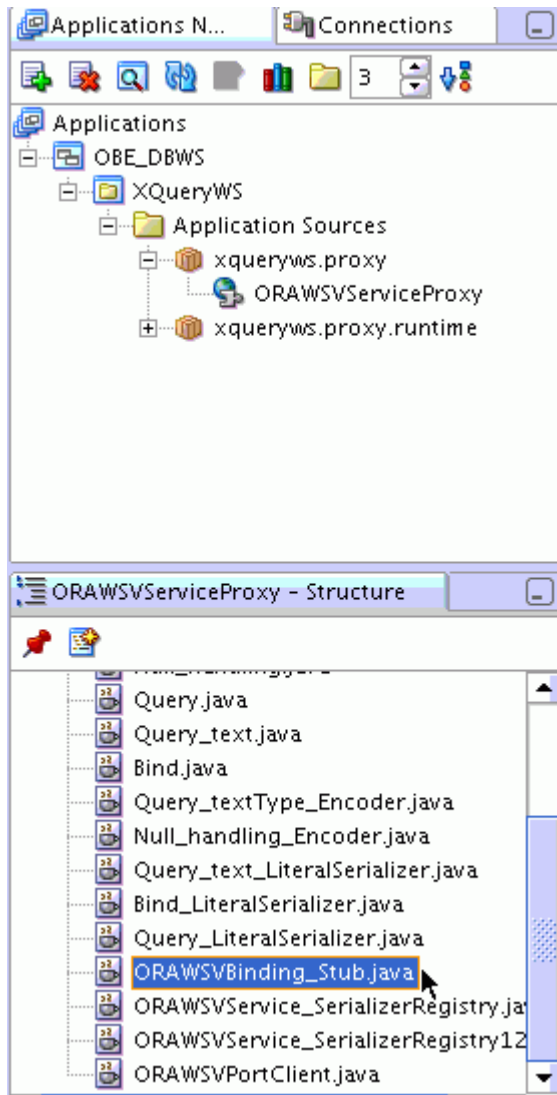
/**
 * @param args
 */
public static void main(String[] args) {
    try {
        xqueryws.proxy.ORAWSVPortClient myPort = new xqueryws.proxy.OR
        System.out.println("calling " + myPort.getEndpoint());
        // Add your own code here
        myPort.setUsername("oe");
        myPort.setPassword("oe");
        String endPoint = myPort.getEndpoint();
        xqueryws.proxy.Query_text xq_text = new Query_text();
        xqueryws.proxy.Query xq = new Query();
        xq_text.setType(xqueryws.proxy.Query_textType.XQUERY);
        xq_text.set_value("ora:view(\"PO_SL_BIX_TABLE\")/PurchaseOrder
        xq.setQuery_text(xq_text);
        System.out.println("Invoking the web service");
        SOAPElement res= myPort.XMLFromQuery(xq);
        System.out.println("Done. You can now check your result with H

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

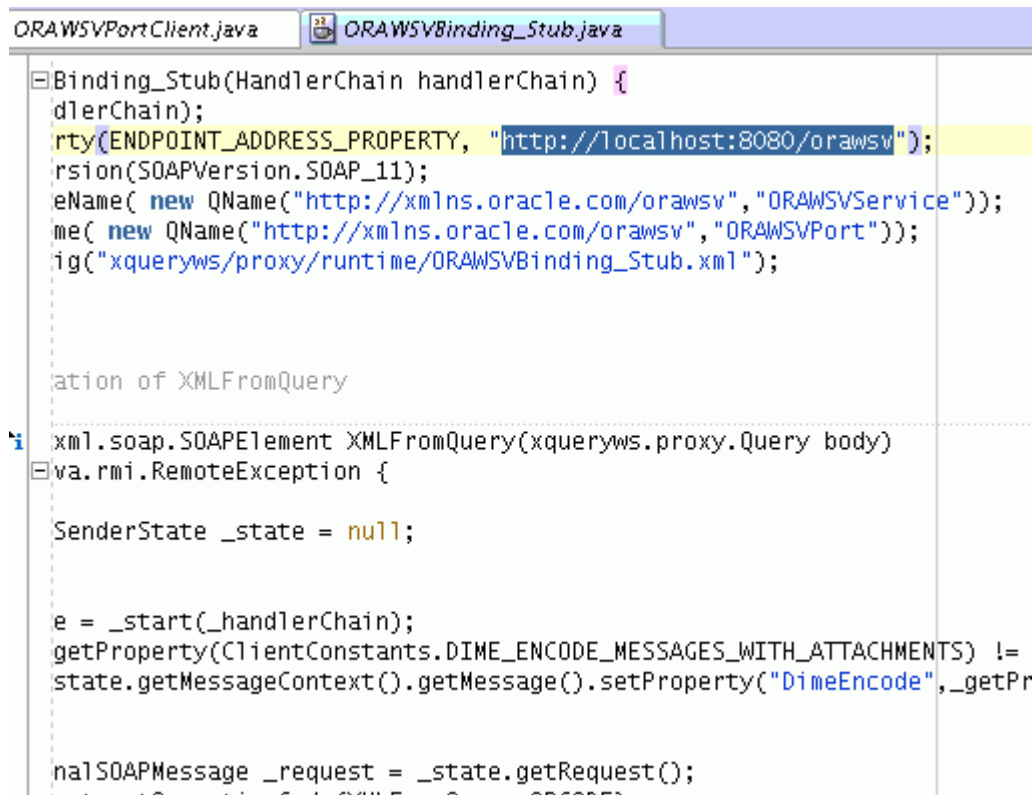
20. In the navigator, expand **OBE-DBWS > XQueryWS > Application Sources > xqueryws.proxy**. Click **ORAWSVServiceProxy** to see the structure at the bottom window pane.



21. In the ORAWSVServiceProxy-Structure window pane, double-click **ORAWSVBinding\_Stub.java**.



22. The code in `ORAWSVBinding_Stub.java` is displayed in the right window. Make sure all occurrences of the `ENDPOINT_ADDRESS_PROPERTY` are set to `http://localhost:8080/orawsv`.



```
ORAWSVPortClient.java  ORAWSVBinding_Stub.java
Binding_Stub(HandlerChain handlerChain) {
    dlerChain);
    rty(ENDPOINT_ADDRESS_PROPERTY, "http://localhost:8080/orawsv");
    rsion(SOAPVersion.SOAP_11);
    eName( new QName("http://xmlns.oracle.com/orawsv", "ORAWSVService"));
    me( new QName("http://xmlns.oracle.com/orawsv", "ORAWSVPort"));
    ig("xqueryws/proxy/runtime/ORAWSVBinding_Stub.xml");

    ation of XMLFromQuery
}

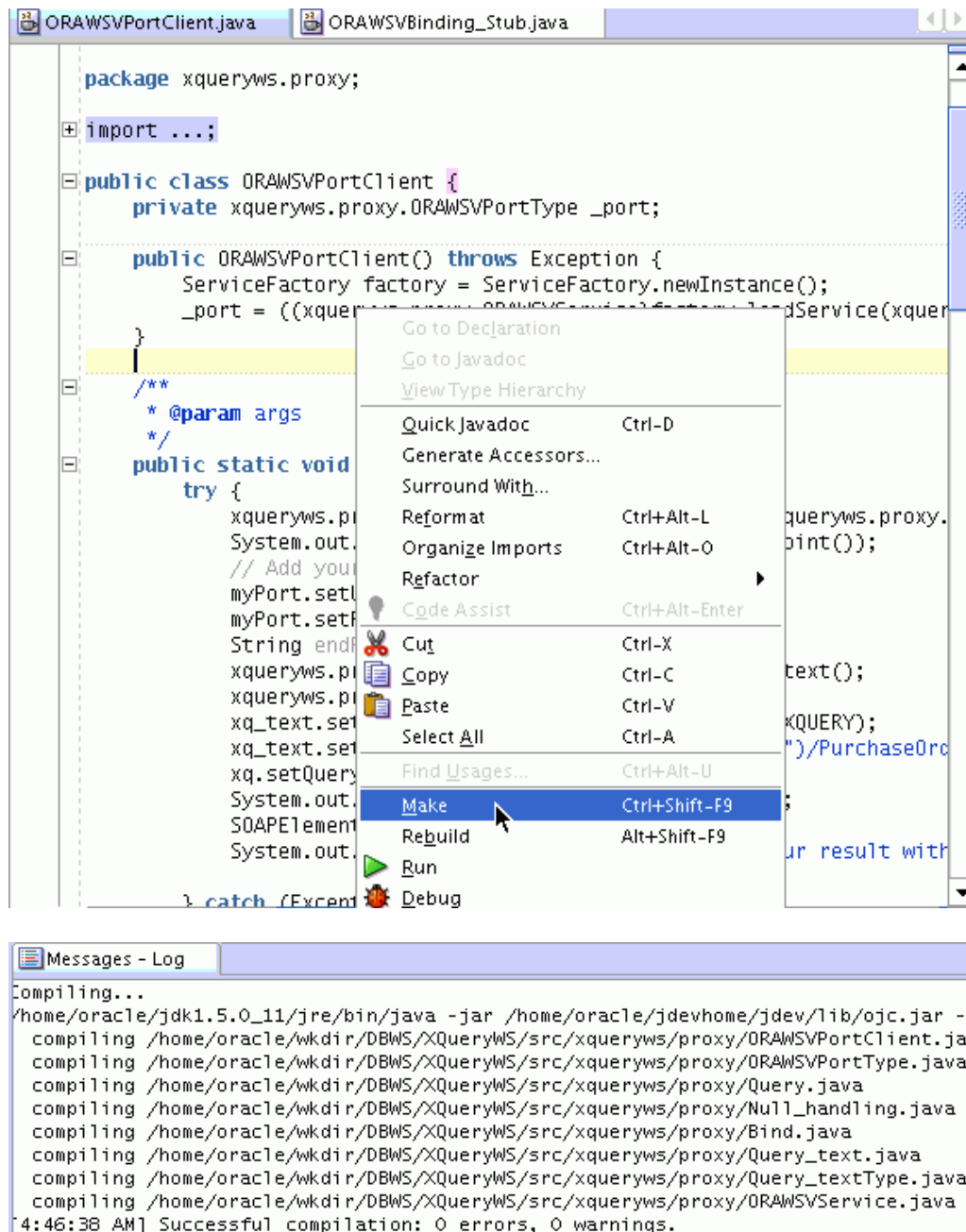
xml.soap.SOAPElement XMLFromQuery(xqueryws.proxy.Query body)
{
    va.rmi.RemoteException {

        SenderState _state = null;

        e = _start(_handlerChain);
        getProperty(ClientConstants.DIME_ENCODE_MESSAGES_WITH_ATTACHMENTS) !=
        state.getMessageContext().getMessage().setProperty("DimeEncode", _getPr

        nalSOAPMessage _request = _state.getRequest();
        setPropertyCode(XMLFromQuery, ORAWSVBinding_Stub.java)
```

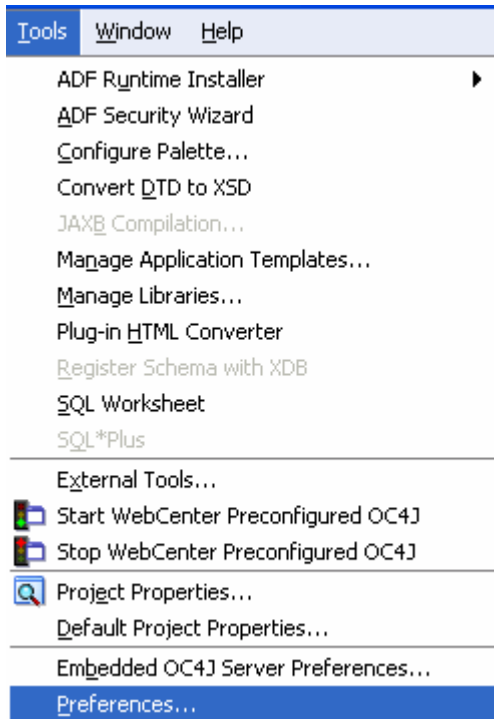
23. Now, switch to ORAWSVPortClient.java window pane. Right-click on the code, and select **Make**. At the bottom of the window, Messages- Log shows successful compilation.



## View Web service request and response using Http Analyzer

1. To view the Webservice request and response using HTTP Analyzer, first, you should set HTTP Analyser settings in JDeveloper.

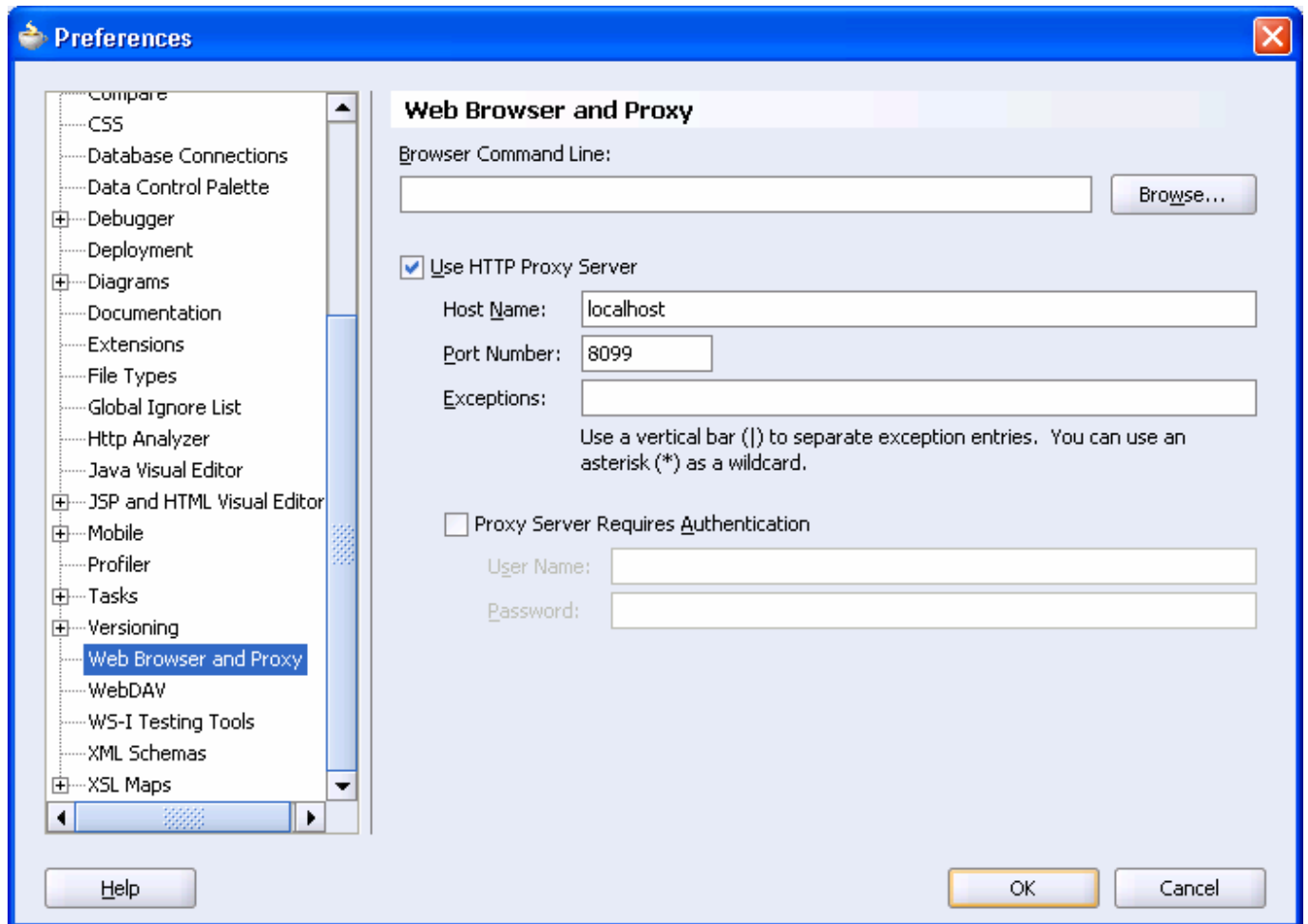
In the JDeveloper window, go to **Tools**, and select **Preferences...**



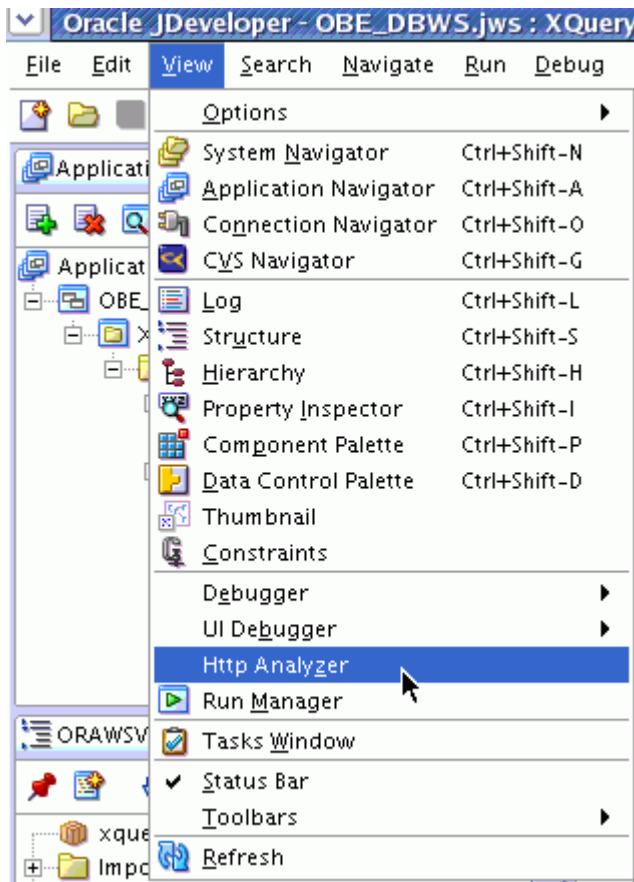


2. In the Preferences window, select **Web Browser and Proxy**. Make sure of the following, and click **OK**.

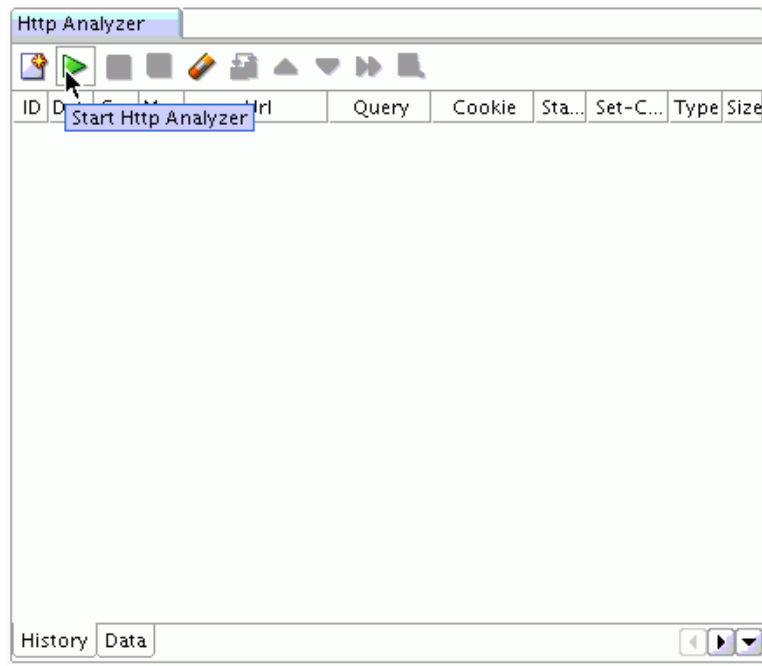
- Use default HTTP Proxy Server check box is selected
- Host Name is **localhost**
- Port Number is **8099**.



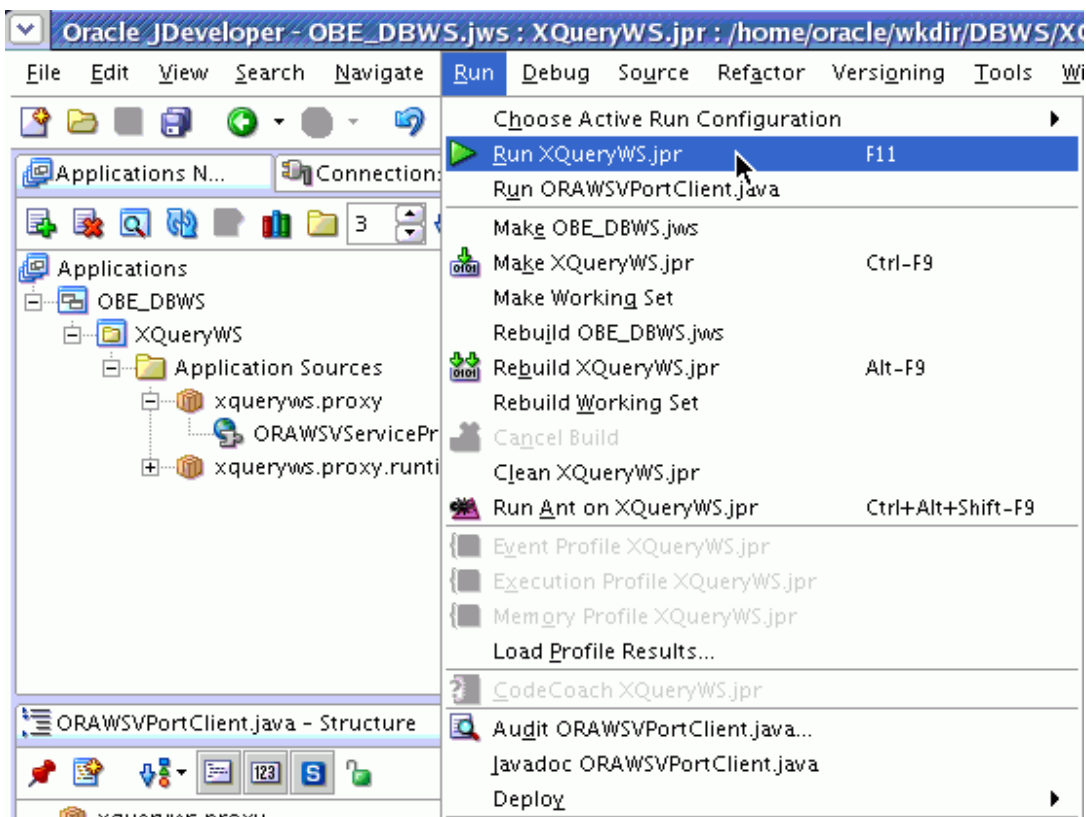
3. In the JDeveloper window, go to **View**, and select **Http Analyzer**.



4. The Http Analyzer appears at the bottom of the window. Click the **Start Http Analyzer** icon.

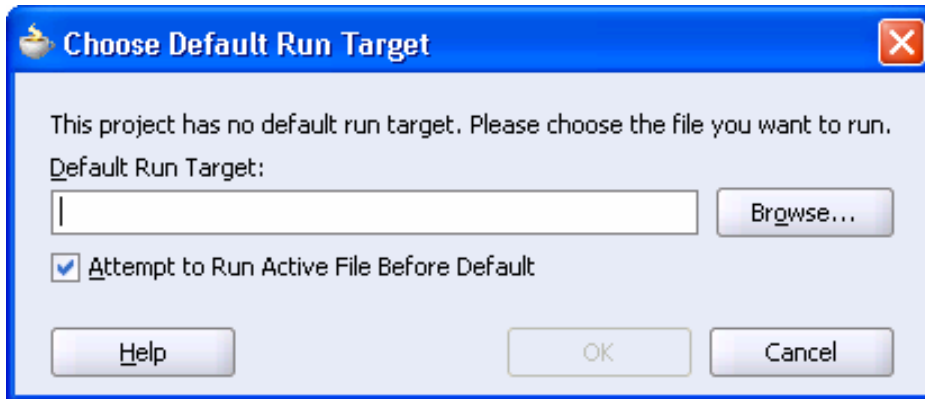


5. Now, you run the project. In the JDeveloper window, go to **Run**, and select **Run XQueryWS.jpr**.

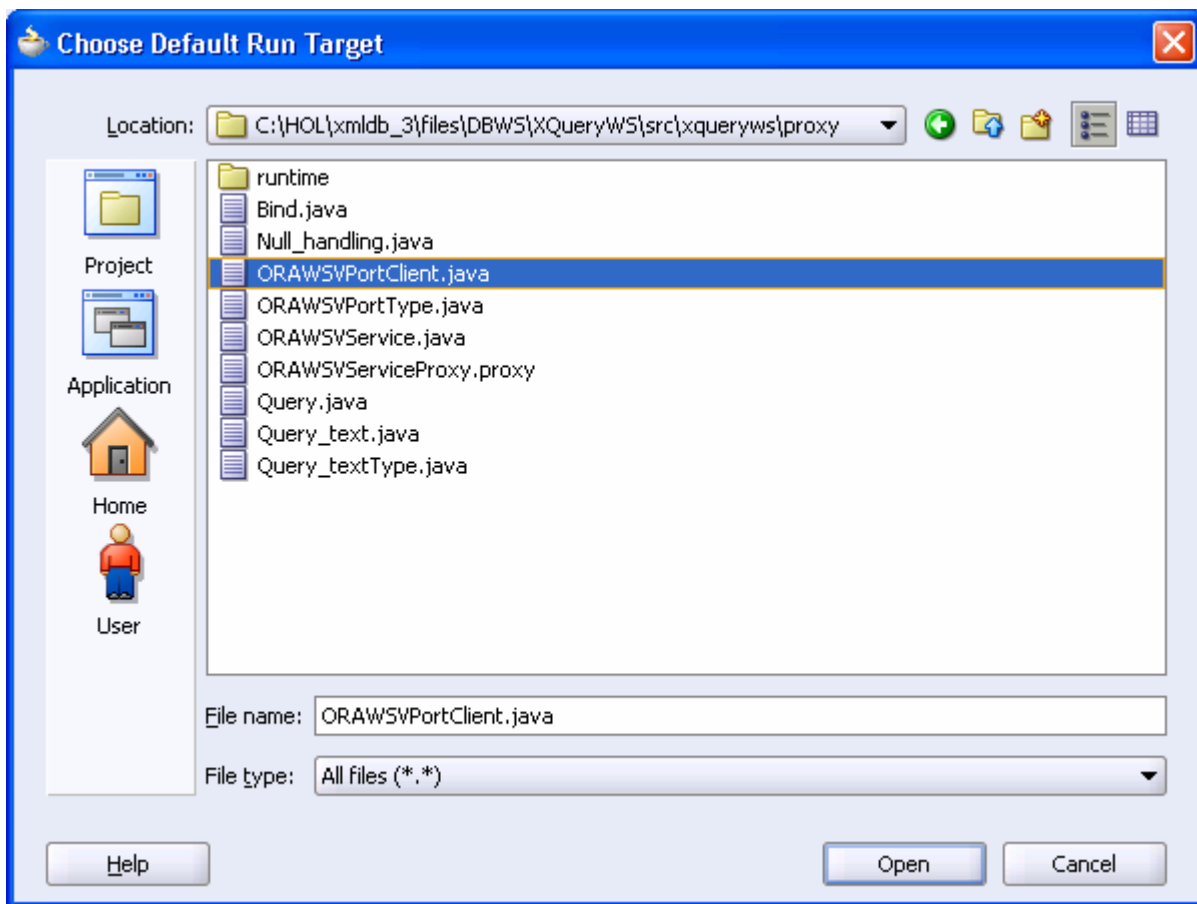


6. In the Choose Default Run Target Window that appears, perform the following steps:

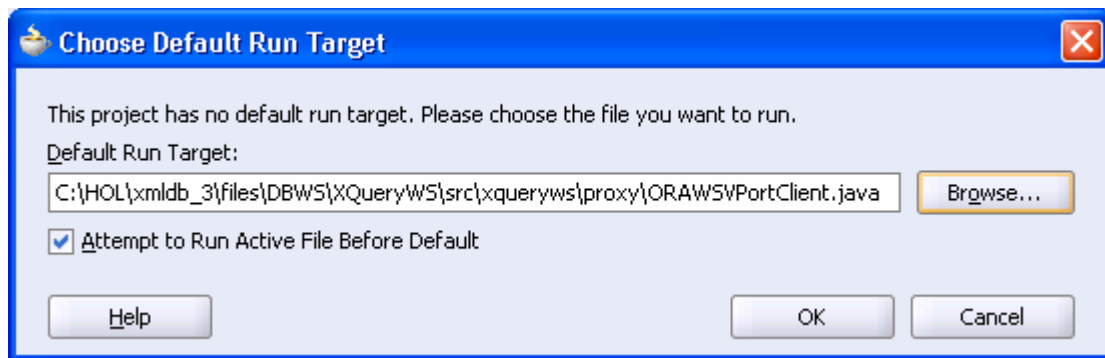
a. Click **Browse**.



b. Go to <working directory>/DBWS/XQueryWS/src/xqueryws/proxy. Then, select ORAWSVPortClient.java, and click **Open**.

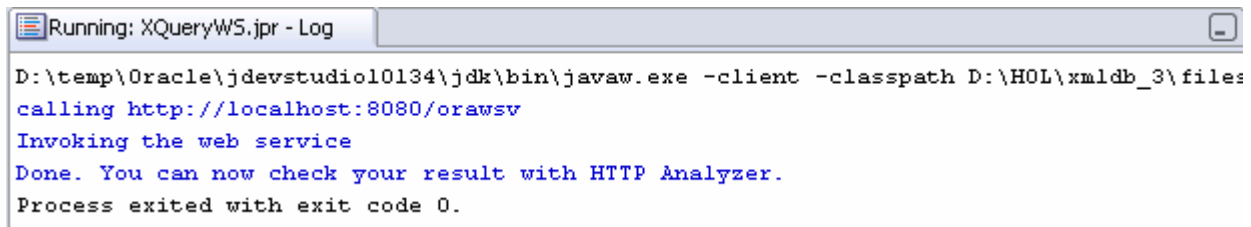


c. Click **OK**.



7. In the Log window that appears at the bottom, you see the following message.

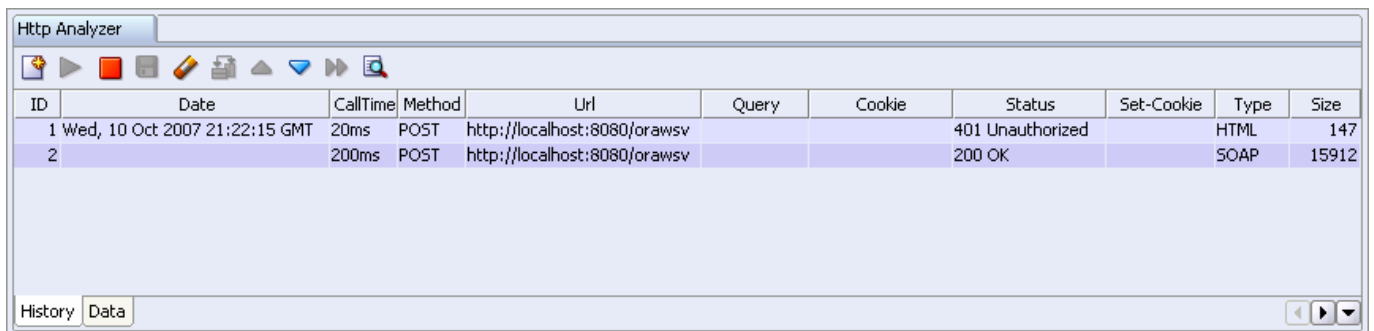
calling http://localhost:8080/orawsv  
Invoking the web service  
Done. You can now check your result with HTTP Analyzer.  
Process exited with exit code 0.



```
D:\temp\Oracle\jdevstudio10134\jdk\bin\javaw.exe -client -classpath D:\H01\xml\db_3\files:
calling http://localhost:8080/orawsv
Invoking the web service
Done. You can now check your result with HTTP Analyzer.
Process exited with exit code 0.
```

8. Switch to Http Analyzer at the bottom window pane. Double-click on the last item.

Alternatively, you can select the last item and click **Data**.



ID	Date	CallTime	Method	Url	Query	Cookie	Status	Set-Cookie	Type	Size
1	Wed, 10 Oct 2007 21:22:15 GMT	20ms	POST	http://localhost:8080/orawsv			401 Unauthorized		HTML	147
2		200ms	POST	http://localhost:8080/orawsv			200 OK		SOAP	15912

9. In the Http Analyzer window, the bottom left pane displays the web service request, and the bottom right pane displays the response.

The screenshot shows the 'Http Analyzer' application window. It is divided into four main panes. The top-left pane shows the request headers for 'localhost'. The top-right pane shows the response headers for 'localhost'. The bottom-left pane displays the raw XML of the SOAP request. The bottom-right pane displays the raw XML of the SOAP response. The status bar at the bottom indicates 'History' and 'Data' tabs.

**Request: localhost**

Header	Value
Host	localhost:8080
Proxy-Connection	Keep-Alive
Connection	TE
TE	trailers, deflate, gzip, compress
User-Agent	Oracle HTTPClient Version 10h
SOAPAction	"http://localhost:8080/orawsv"
Accept-Encoding	gzip, x-gzip, compress, x-compress
Authorization	Basic b2U6b2U=
Content-type	text/xml; charset=UTF-8
Content-length	407

**Response: localhost**

Header	Value
MS-Author-Via	DAV
DAV	1,2, <http://www.oracle.com/xdm/webdav/props>
Server	Oracle XML DB/Oracle Database
Content-Type	text/xml; charset=UTF-8
Transfer-Encoding	chunked

**Request XML:**

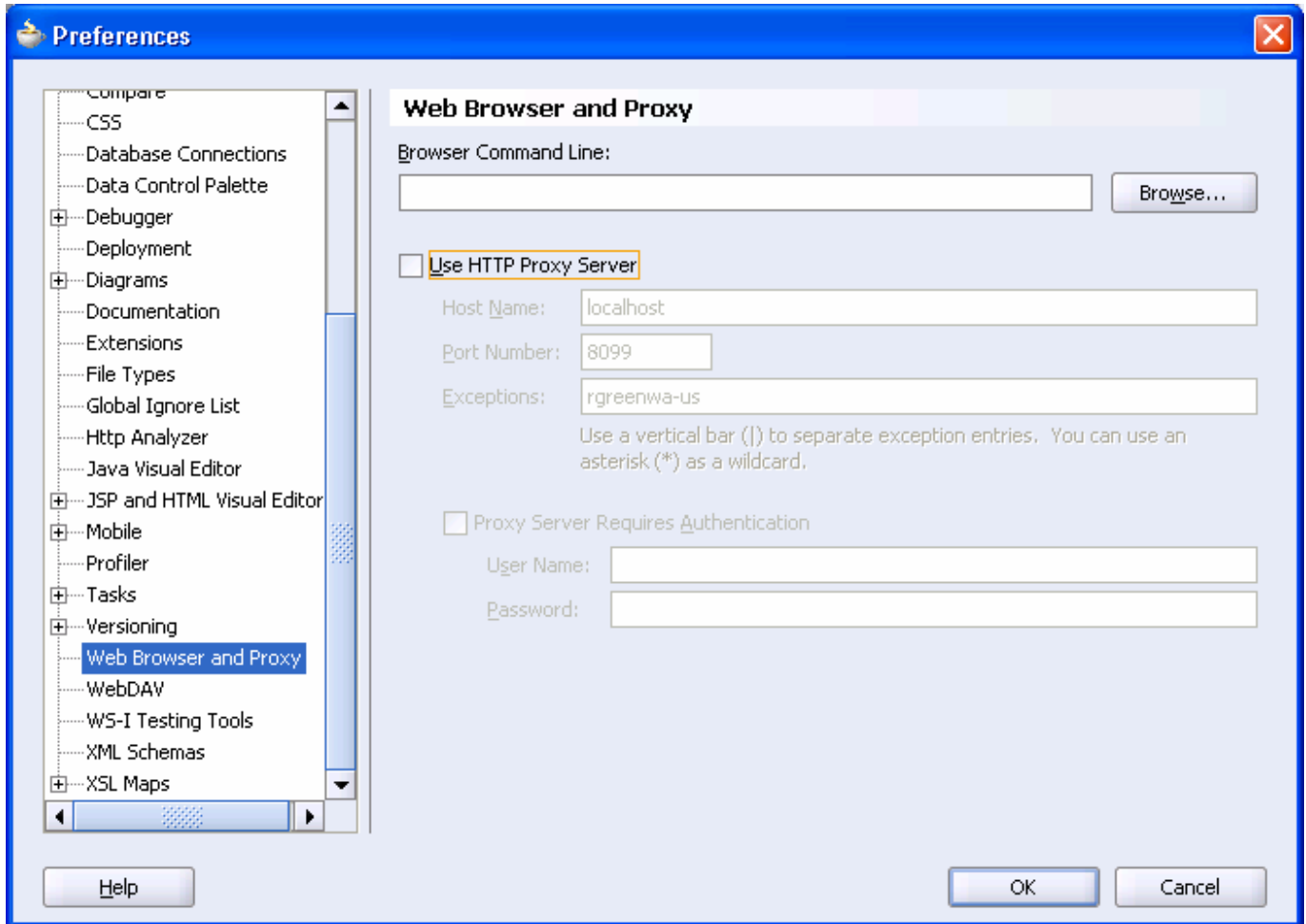
```
<?xml version = '1.0' encoding = 'UTF-8' ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <ns0:query>
      <ns0:query_text type="XQUERY">ora:view("PO_SL_BIX_TABLE") / Po
    </ns0:query>
  </env:Body>
</env:Envelope>
```

**Response XML:**

```
<?xml version = '1.0' ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <queryOut xmlns="http://xmlns.oracle.com/orawsv">
      <ROWSET>
        <ROW>
          <COLUMN_VALUE>
            <Reference>TF0X-20021009123336872PDT</Reference>
          </COLUMN_VALUE>
        </ROW>
        <ROW>
          <COLUMN_VALUE>
            <Reference>TF0X-20021009123336912PDT</Reference>
          </COLUMN_VALUE>
        </ROW>
      </ROWSET>
    </queryOut>
  </soap:Body>
</soap:Envelope>
```

## Reset the Proxy Server Setting in JDeveloper

1. In the “Preferences” dialog box, select the “Web Browser and Proxy” from the list, and make sure the “Use HTTP Proxy Server” checkbox is unchecked.



## Summary

In this tutorial, you learned how to:

- Create a binary XML table
- Review the XML Schema in JDeveloper
- Use XML DB Web Services for Service-Oriented Architecture





# Lesson 4: Using Binary XML and XMLIndex to Aggregate and Query Unstructured XML Data Sources

## Purpose












This tutorial shows you how to use binary XML storage model, XMLIndex, XQuery, SQL/XML, and many other salient capabilities of Oracle XML DB to aggregate and query unstructured XML data sources.

## Time to Complete

Approximately 40 minutes

## Topics

This tutorial covers the following topics:

-  [Overview](#)
-  [Prerequisites](#)
-  [Setting Environment Variables](#)
-  [Creating a new user and grant necessary privileges](#)
-  [Creating a table to store information about RSS feeds](#)
-  [Creating a table to store RSS news items](#)
-  [Creating an XMLIndex on News Items](#)
-  [Creating an RSS View with Aggregated and Normalized News Items from Diverse News Feeds](#)
-  [Reading the Normalized RSS Feed with a Firefox 2.0 Browser](#)
-  [Cleaning up the machine](#)
-  [Summary](#)

## Overview

Oracle Database 11g XML DB introduces a new binary XML storage model, a new XMLIndex index, and many other salient new capabilities for high performance storage and retrieval of structured and unstructured XML documents. This tutorial uses a real world use case of an RSS feed aggregator to demonstrate these versatile and high performance capabilities.

### Binary XML Storage Model:

Binary XML is a new storage model for abstract data type XMLType, joining the existing native storage models of structured (object-relational) and unstructured (CLOB) storage. Binary XML storage provides more efficient database storage, updating, indexing, and fragment extraction than unstructured storage. It can provide better query performance than unstructured storage—it does not suffer from the XML parsing bottleneck (it is a post-parse persistence model). Like structured storage, binary XML storage is aware of XML Schema data types and can take advantage of native database data types. Like unstructured storage, no data conversion is needed during database insertion or retrieval. Like structured storage, binary XML storage allows for piecewise updates. Because binary XML data can also be used outside the database, it can serve as an efficient XML exchange medium, and you can offload work from the database to increase overall performance in many cases. Like unstructured storage, binary XML data is kept in document order. Like structured storage, data and metadata can, using binary storage, be separated at the database level, for efficiency. Like unstructured storage, however, binary storage allows for intermingled data and metadata, which lets instance structures vary. Binary XML storage allows for very complex and variable data, which in the structured-storage model could necessitate using many database tables and joins. Unlike the other XMLType storage models, you can use binary storage for XML schema-based data even if the XML schema is not known beforehand, and you can store multiple XML schemas in the same table and query across common elements.

### **XMLIndex Indexing for Binary XML and Unstructured XML Storage Models:**

B-Tree indexes can be used advantageously with structured storage. They provide sharp focus by targeting the underlying objects directly. They are generally ineffective, however, in addressing the detailed structure (elements and attributes) of an XML document stored in a binary XML or a CLOB instance. That is the special domain of XMLIndex: binary XML and unstructured storage models. Unlike a B-Tree index, which you define for a specific column that represents an individual XML element or attribute, an XMLIndex index is very general: indexing with XMLIndex applies to all possible XPath expressions for your XML data. An XMLIndex index presents the following advantages over other indexing methods:

An XMLIndex index can be used for SQL/XML functions `XMLExists()`, `XMLTable()`, and `XMLQuery()`, and it is effective in any part of a query; it is not limited to use in a WHERE clause. This is not the case for any of the other kinds of indexes you might use with XML data.

XMLIndex can thus speed access to SELECT list data and FROM list data, making it useful for XML fragment extraction, in particular. Function-based indexes and CTXPath indexes

You need no prior knowledge of the XPath expressions that will be used in queries. XMLIndex is completely general. This is not the case for function-based indexes.

You can use an XMLIndex index with either XML schema-based or non-schema-based data. It can be used with binary XML and unstructured storage models. B-Tree indexing is appropriate only for schema-based data stored object-relationally (structured storage); it is ineffective for XML schema-based data stored in a binary XML or a CLOB instance.

You can use an XMLIndex index for searches with XPath expressions that target collections, that is, nodes that occur multiple times within a document. This is not the case for functional indexes.

### **Oracle Database-Native XQuery:**

Since XQuery is now a W3C standard, the IT community has started adopting the business uses of XML and XQuery. As the innovation leader in commercial database technology, Oracle Database 11g provides a full-featured native XQuery engine integrated with the traditional Oracle database server. On the SQL side, the SQL/XML standard has defined a way to encapsulate XML in SQL and to integrate the querying of XML using XQuery. This is being accomplished by introducing new SQL functions: `XMLQuery`, `XMLTable`, `XMLExists`, and `XMLCast`, which operate on XML and SQL values using XQuery. Oracle Database 11g enables XQuery support in the database server through these SQL standard functions. A new `XQUERY` command has also been implemented in SQL\*Plus to allow users to enter XQuery expressions on the command line. With standards-based implementation of XQuery in Oracle Database 11g, application developers can use their favorite APIs (e.g., JDBC, ODP.NET, and web service) to access Oracle Database XQuery capabilities.

### **Benefits of Oracle XQuery:**

Using SQL/XML XQuery functions along with indexing schemes for structured, unstructured, and binary XML storage models, XML dB can perform uniform XML queries across different storage models with orders of magnitude performance improvement over DOM-based functional evaluation of XML queries. Furthermore, XML queries can be seamlessly merged with SQL relational queries to handle all query scenarios. Finally, the XML query capabilities of Oracle XML dB are built on the solid foundation of industry's best relational database that is highly reliable, available, scalable, and secure. In short, the XML dB query capabilities in Oracle Database 11g provide the most comprehensive and efficient functionality for versatile, scalable, concurrent, and high performance XML applications.

## **Prerequisites**

Before you perform this tutorial, you should:

1. Check your Oracle Database 11g installation
2. Check your Firefox browser version 3.0 installation
3. Check your Oracle SQL Developer 1.5.1 installation
4. Check the files in your working directory (/HOL/xmlldb\_4/files)

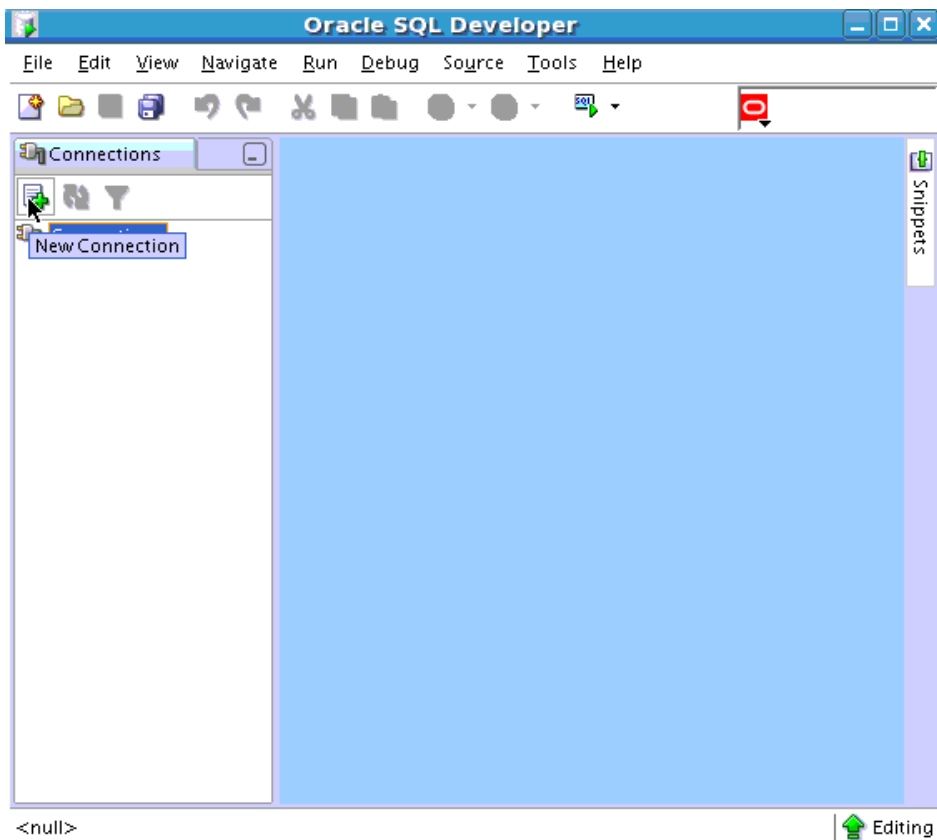
**NOTE:** To accommodate the hands-on lab environment (i.e., no internet access), all the steps have been pre-run during installation to populate news items. You should focus on examining the scripts and running the queries.

## Creating a new user and grant necessary privileges

Using the following steps you would create a new user and use the new security mechanism in Oracle Database 11g to grant the new user access to internet.

### 1. Connect as user system

1. Open SQL Developer.
2. In the Connections tab select New Connection.



3. Connect as system/oracle.

New / Select Database Connection

Connectio...	Connectio...
system	system@...

Connection Name: system

Username: system

Password: \*\*\*\*\*

☐ Save Password

Oracle Access MySQL SQLServer

Role: default

Connection Type: ☒ Basic ☐ TNS ☐ Advanced

Hostname: localhost

Port: 1521

☒ SID

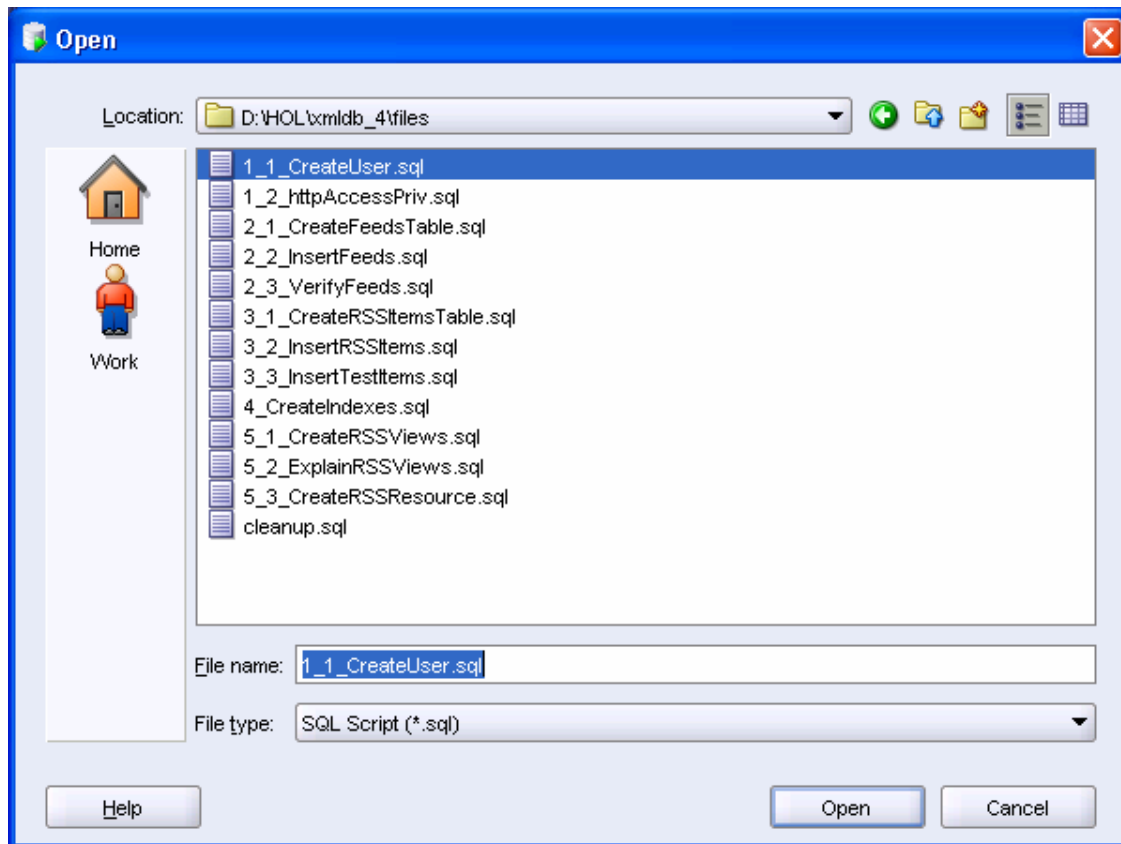
☐ Service name

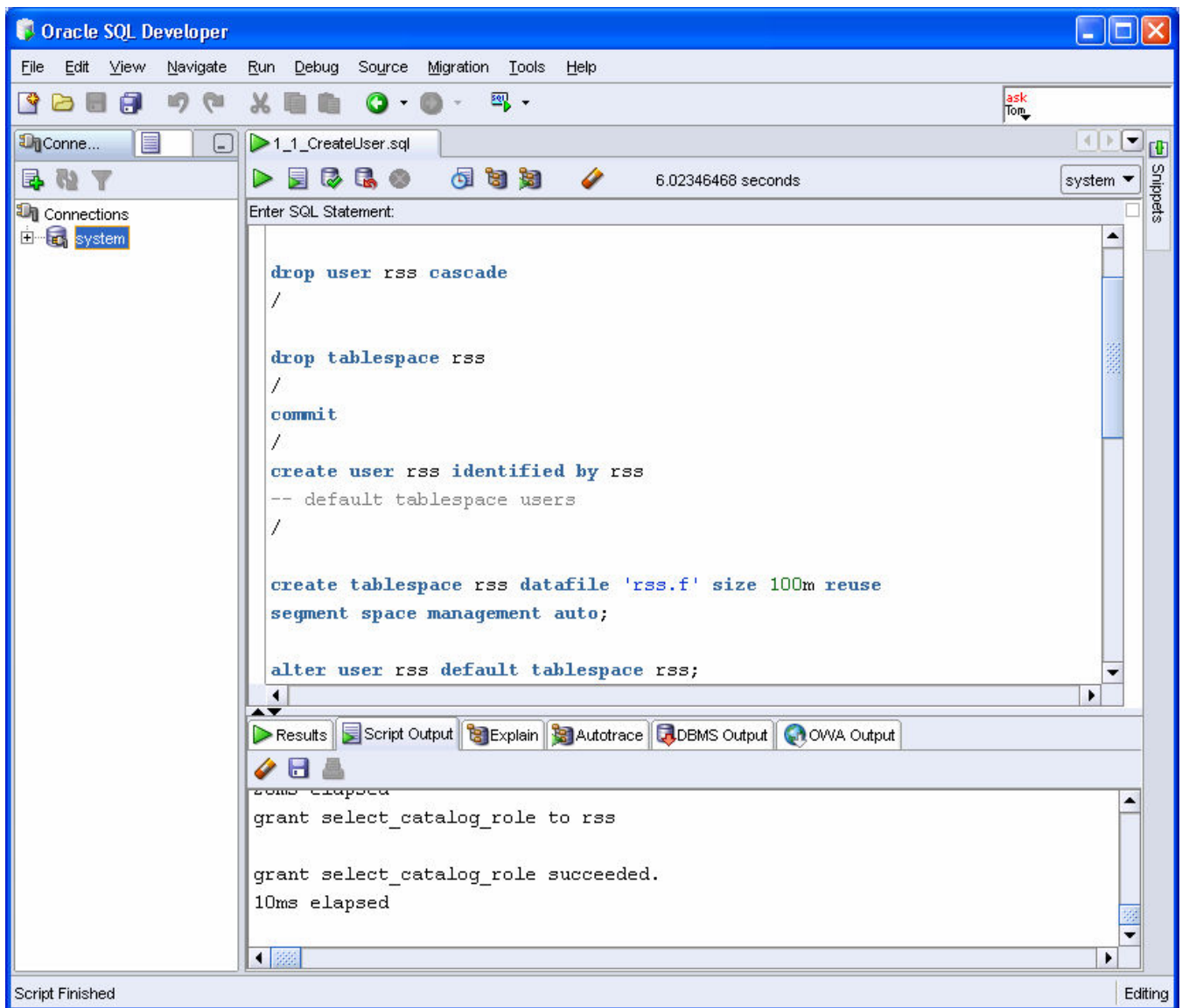
orcl

Status :

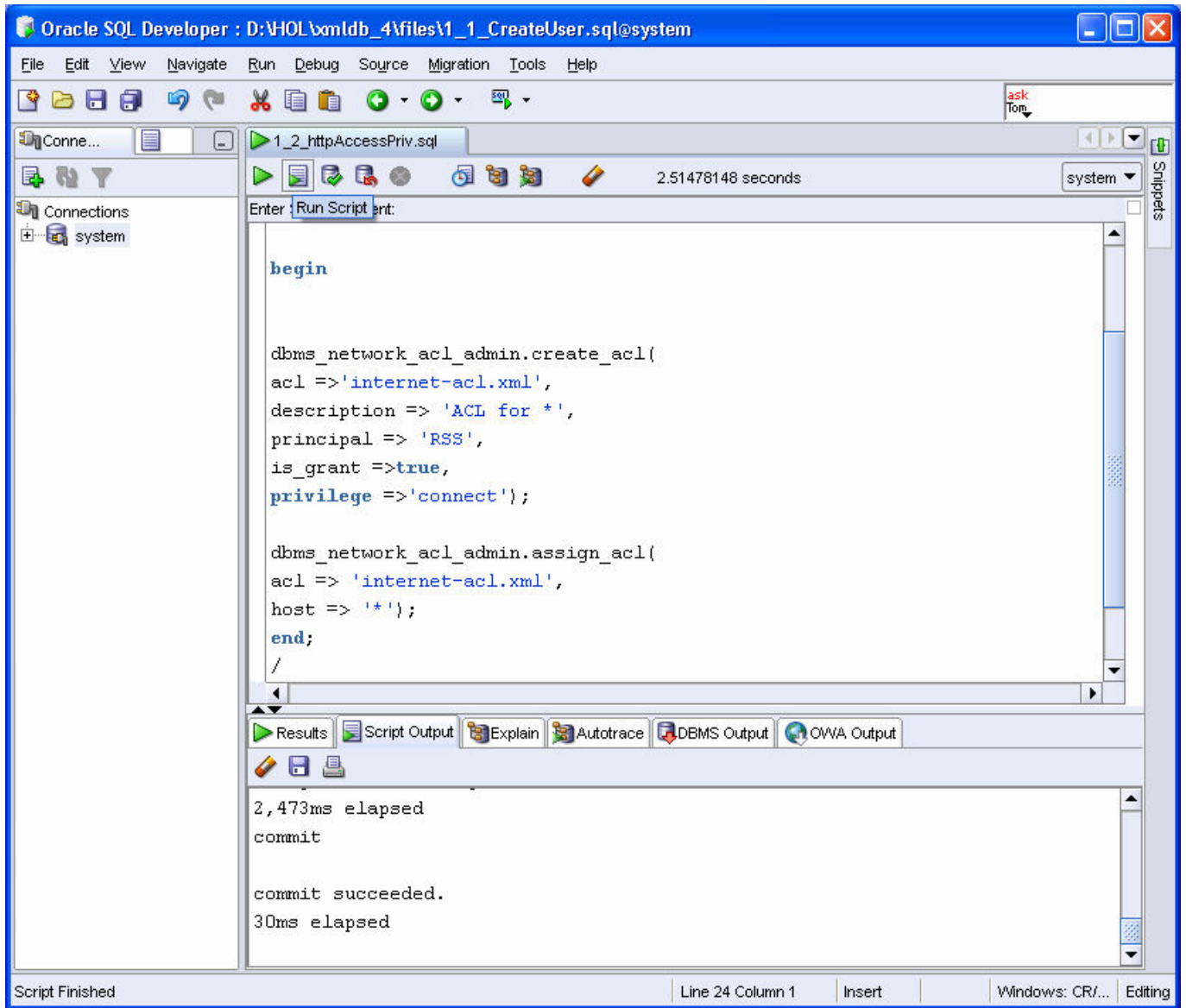
Help Save Clear Test Connect Cancel

2. Open and execute the script **1\_1\_CreateUser.sql**.





3. Open and execute **1\_2\_httpAccessPriv.sql** create a new Internet access control list (ACL) and assigned to the new user.





## Creating a table to store information about RSS feeds

An RSS (Real Simple Syndication) feed is an URL pointing to an XML document in one of the many variants of RSS formats. The XML document of an RSS feed contains news items. In this section, you will create a table to store information about subscribed RSS feeds.

1. Create a connection with user rss/rss

**New / Select Database Connection**

Connection Name	Connection Name
system	system@...

Connection Name:

Username:

Password:

☐ Save Password

Database: ☒ Oracle ☐ Access ☐ MySQL ☐ SQLServer

Role:

Connection Type: ☒ Basic ☐ TNS ☐ Advanced

Hostname:

Port:

☒ SID:

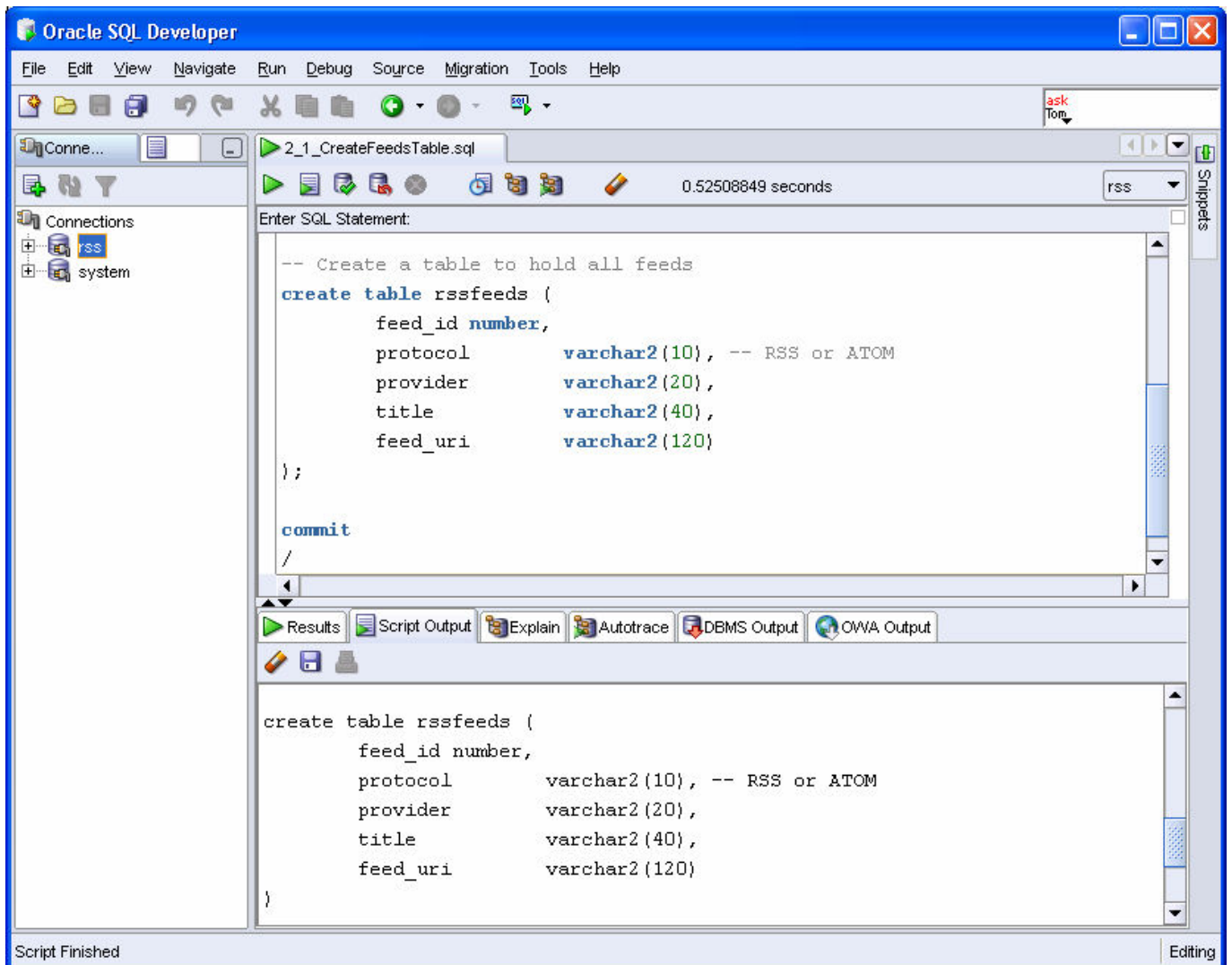
☐ Service name:

Status: Success

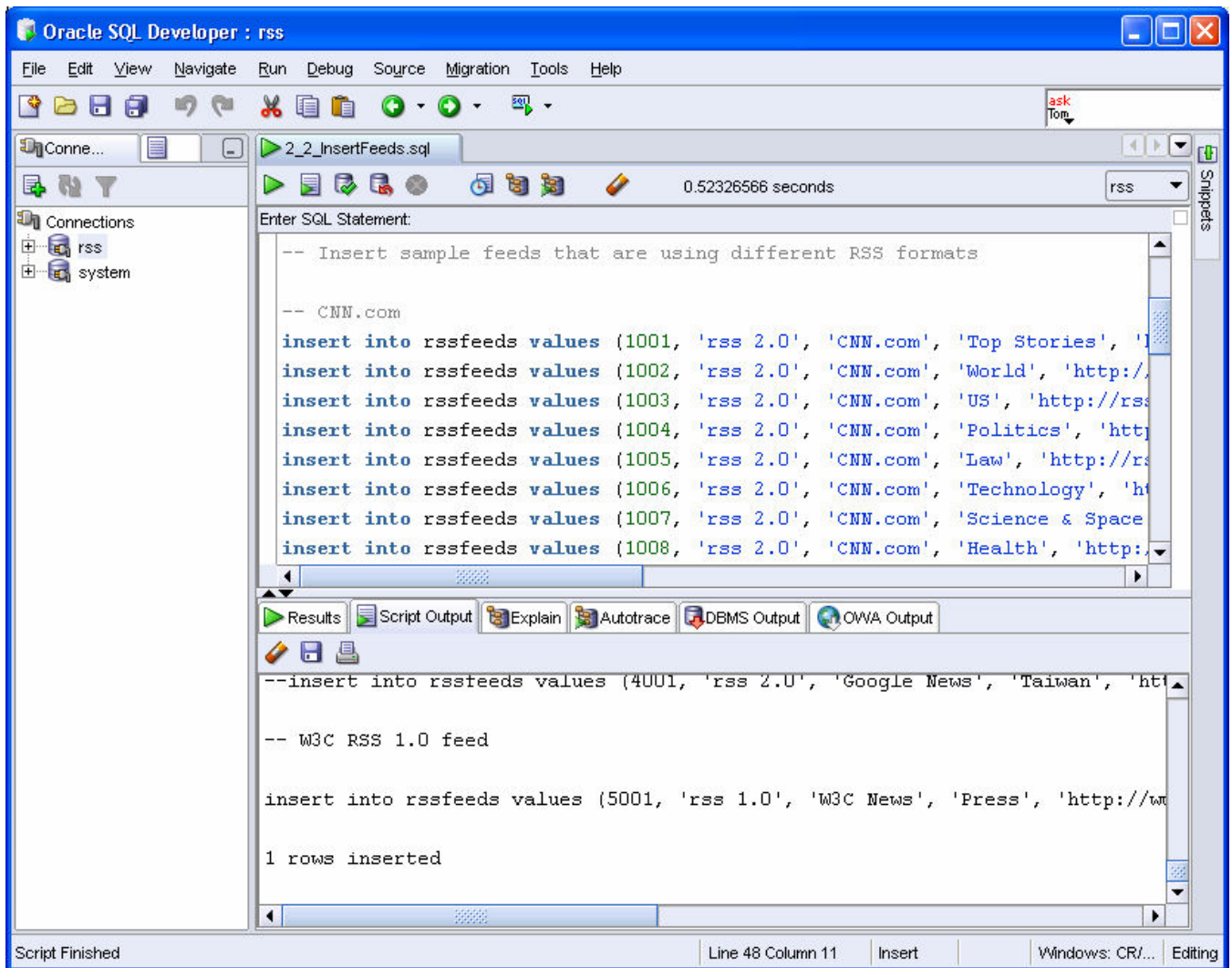
Buttons:

Click on the "Connect" button to connect as user RSS.

2. As user RSS, execute the script **2\_1\_CreateFeedsTable.sql** to create a new rssfeeds table.



3. Open and execute the script `2_2_InsertFeeds.sql` to insert a list of news feeds into the `rssfeeds` table.



4. Open and execute the script `2_3_VerifyFeeds.sql` to verify that a list of news feeds have been inserted.

The screenshot shows the Oracle SQL Developer interface. The main window displays the script `2_3_VerifyFeeds.sql` with the following content:

```
conn rss/rss
-- Check current feeds
select * from rssfeeds
/
```

The script has been executed, and the results are displayed in the 'Results' pane. The results show a table with 12 rows of news feeds. The columns are: FEED\_ID, FEED\_PROTOCOL, PROVIDER, TITLE, and FEED\_URI. The data is as follows:

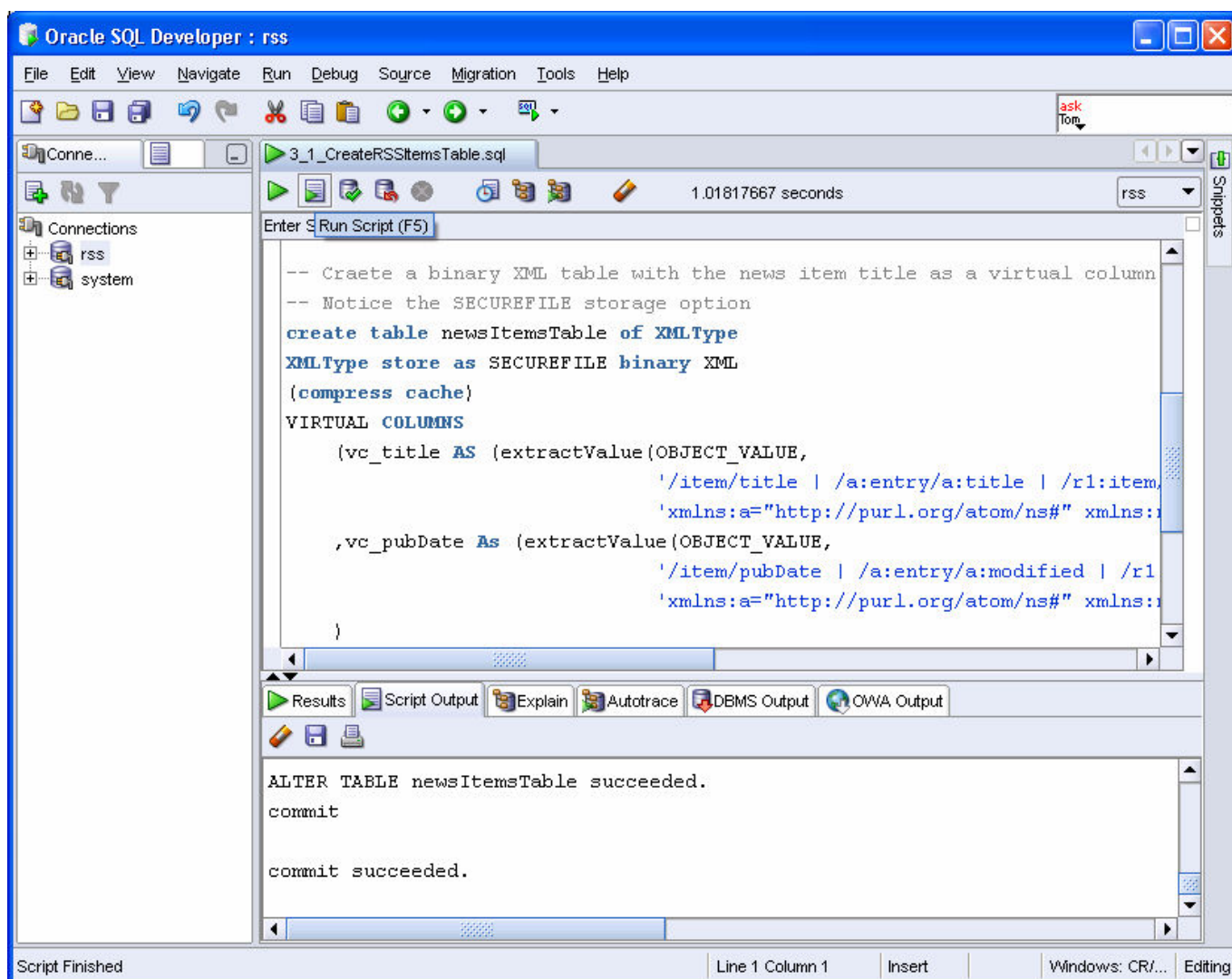
FEED_ID	FEED_PROTOCOL	PROVIDER	TITLE	FEED_URI
1	1001 rss 2.0	CNN.com	Top Stories	http://rss.cnn.com/rss/cnn_topstories.rss
2	1002 rss 2.0	CNN.com	World	http://rss.cnn.com/rss/cnn_world.rss
3	1003 rss 2.0	CNN.com	US	http://rss.cnn.com/rss/cnn_us.rss
4	1004 rss 2.0	CNN.com	Politics	http://rss.cnn.com/rss/cnn_allpolitics.rss
5	1005 rss 2.0	CNN.com	Law	http://rss.cnn.com/rss/cnn_law.rss
6	1006 rss 2.0	CNN.com	Technology	http://rss.cnn.com/rss/cnn_tech.rss
7	1007 rss 2.0	CNN.com	Science & Space	http://rss.cnn.com/rss/cnn_space.rss
8	1008 rss 2.0	CNN.com	Health	http://rss.cnn.com/rss/cnn_health.rss
9	1009 rss 2.0	CNN.com	Entertainment	http://rss.cnn.com/rss/cnn_showbiz.rss
10	1010 rss 2.0	CNN.com	Travel	http://rss.cnn.com/rss/cnn_travel.rss
11	1006 rss 2.0	CNN.com	Education	http://rss.cnn.com/rss/cnn_education.rss
12	1008 rss 2.0	CNN.com	Offbeat	http://rss.cnn.com/rss/cnn_offbeat.rss

The status bar at the bottom indicates 'All Rows Fetched: 37', 'Line 9 Column 1', 'Insert', and 'VWindows: CR/... Editing'.

## Creating a table to store RSS news items

Different RSS formats used different XML structures to represent news items. This type of highly variable XML structures are best stored with the binary XML storage model. By taking advantage of the binary XML storage model along with the SECUREFILE LOB storage format, you can gain improved performance while reducing storage space.

1. As user `RSS`, execute the script `3_1_CreateRSSItemsTable.sql` to create a new **newsItemsTable**. Notice the creation of “virtual columns” to enforce unique constraints. Also notice how Oracle XML DB can handle different namespaces defined by different RSS formats.



2. Open `3_2_InsertRSSItems.sql`. Depending on whether you are accessing the Internet via a proxy server, modify the script to change the parameter of the `utl_http.set_proxy()` call.

```
conn rss/rss

-- Need one of the following to access RSS feeds on the Internet
--call utl_http.set_proxy('www-proxy.us.oracle.com')
call utl_http.set_proxy(NULL)
/
```

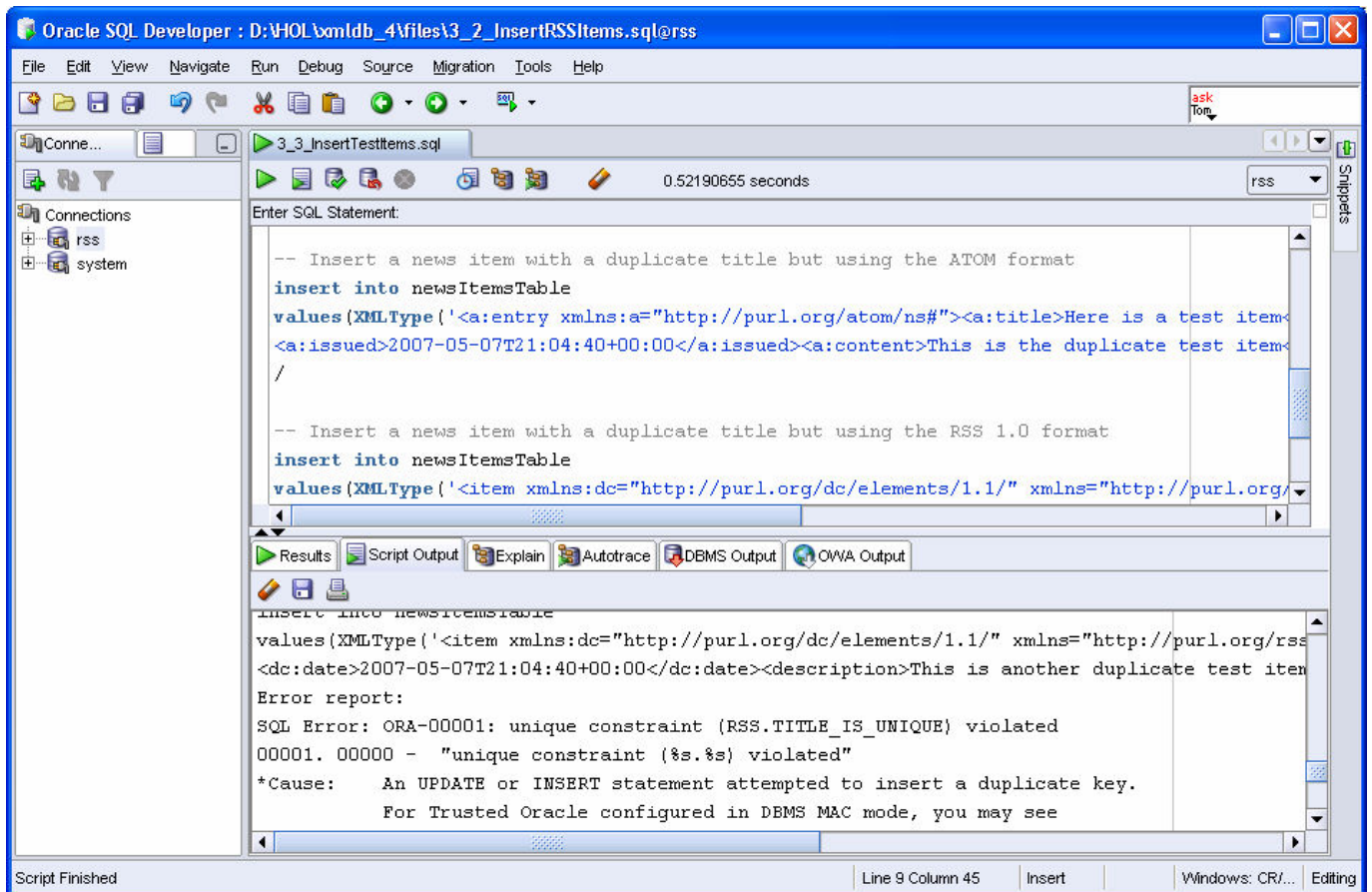
3. You can now execute `3_2_InsertRSSItems.sql` to insert news items into the `newsItemsTable`. Notice how the script uses the `XMLTable()` function to extract news items for insertion into the `newItemsTable`.

```
BEGIN
  sql_stmt :=
    'select column_value from xmltable (''declare namespace a = "http://purl.org/atom/ns#";
    declare namespace r1="http://purl.org/rss/1.0/"
    for $i in . return
      if ($i//item) then $i//item
      else if ($i//r1:item) then $i//r1:item
      else //a:entry'' passing xmlparse (document
    'httpuritype(:1).getclob()) ' ');
  DBMS_OUTPUT.put_line(sql_stmt);
END;
```

Script Finished



4. With a unique constraint on the virtual column of news item title, Oracle XML DB can enforce the uniqueness of a news item title. You can execute `3_3_InsertTestItems.sql` to insert news items with duplicate item titles to confirm the existence of a unique constraint on news item title.



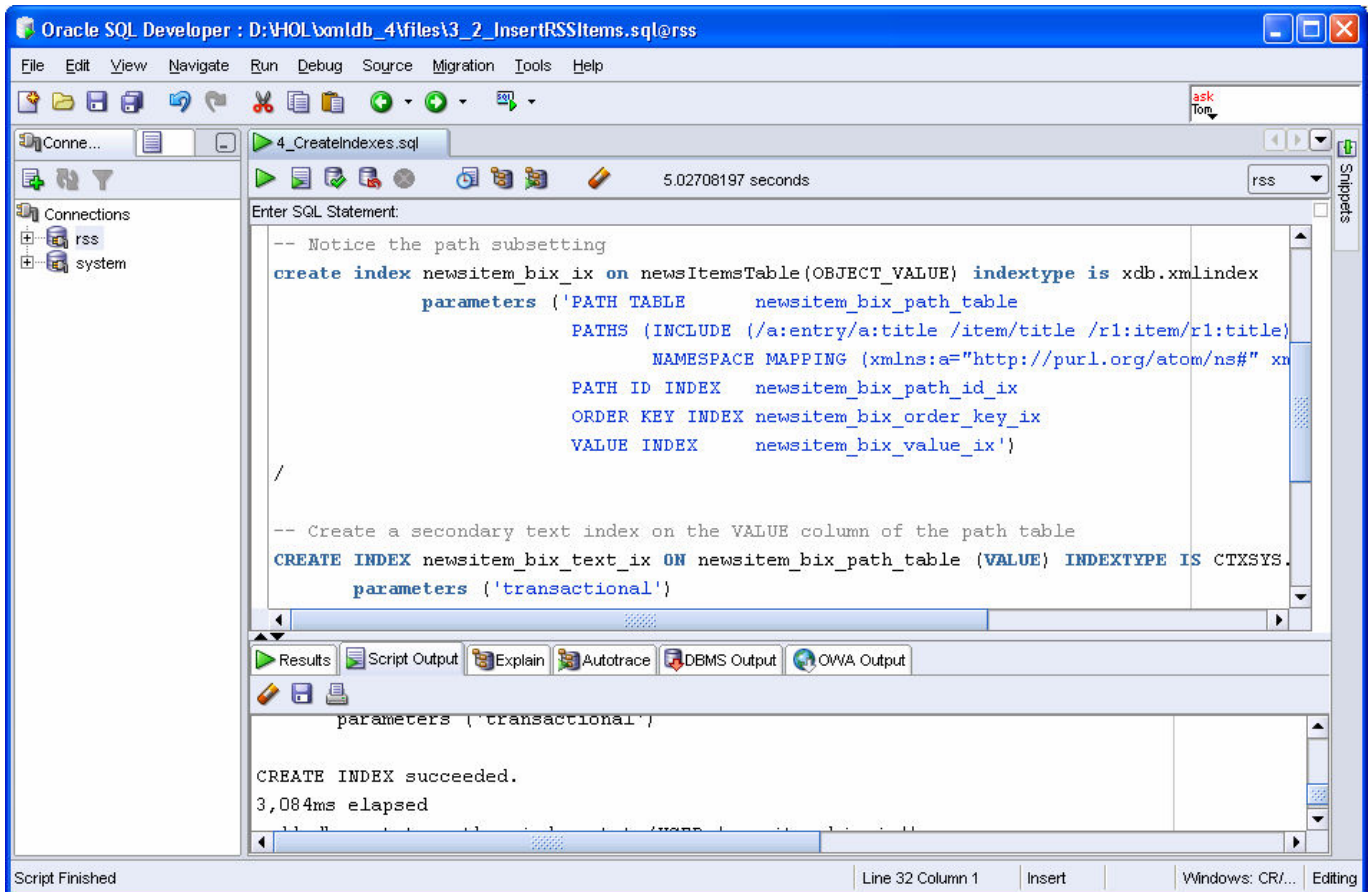
The screenshot shows the Oracle SQL Developer interface. The title bar indicates the file path: `D:\HOL\omldb_4\files\3_2_InsertRSSItems.sql@rss`. The menu bar includes File, Edit, View, Navigate, Run, Debug, Source, Migration, Tools, and Help. The toolbar contains icons for file operations, execution, and debugging. The left sidebar shows a 'Connections' tree with 'rss' and 'system' nodes. The main editor window displays the script `3_3_InsertTestItems.sql` with a run time of 0.52190655 seconds. The script contains two `INSERT` statements for the `newsItemsTable`. The first statement uses the ATOM format, and the second uses the RSS 1.0 format. Both statements attempt to insert items with duplicate titles. Below the script, the 'Results' tab shows the error report:   
SQL Error: ORA-00001: unique constraint (RSS.TITLE\_IS\_UNIQUE) violated  
00001. 00000 - "unique constraint (%s.%s) violated"  
\*Cause: An UPDATE or INSERT statement attempted to insert a duplicate key.  
For Trusted Oracle configured in DBMS MAC mode, you may see

Script Finished | Line 9 Column 45 | Insert | Windows: CR/... | Editing

## Creating an XMLIndex on News Items

With news items stored with binary XML storage model, creating an XMLIndex on news item titles will provide much improved query performance. To allow full text searches on news item titles, a secondary text index can be created on the path table of XMLIndex.

1. As user **RSS**, execute the script **4\_CreateIndexes.sql** to create an XMLIndex using a subset of XPath pointing to news item titles. A secondary text index is also created. Notice how Oracle XML DB handles multiple namespaces used by different RSS formats.

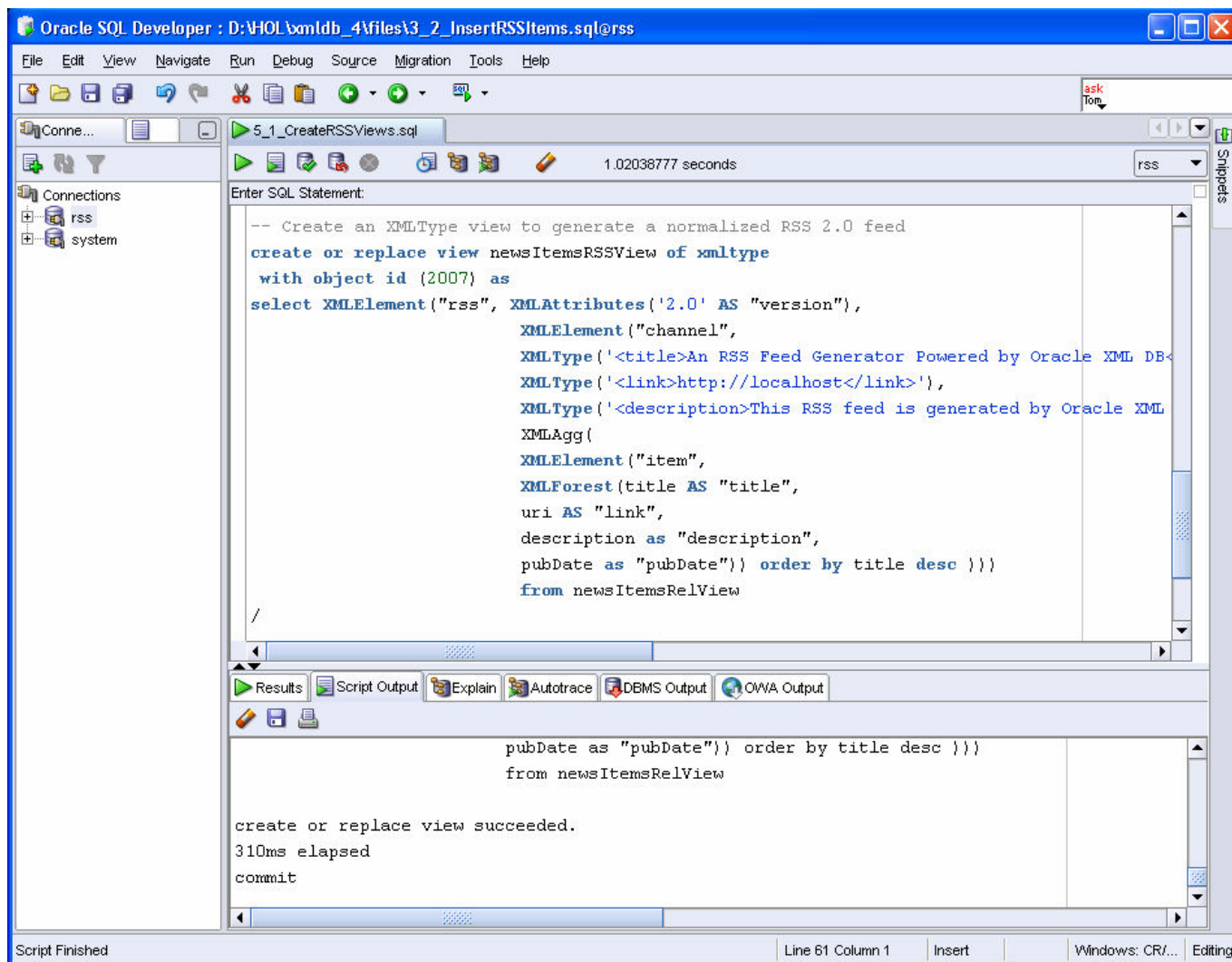




## Creating an RSS View with Aggregated and Normalized News Items from Diverse News Feeds

The main function of a RSS (Real Simple Syndication) aggregator is to aggregate and normalize news items from multiple feeds using diverse RSS formats. In this section, you will see how you can create an RSS view normalizing diverse RSS formats.

1. As user RSS, execute the script `5_1_CreateRSSViews.sql` to create a relational view `newsItemsRelView` which normalizes news items using diverse RSS formats. Notice the UNION ALL clause in the SQL statement for creating this view. Also notice the usage of `ora:contains()` XQuery function for full text searches on news item titles. An XMLType view `newsItemsRSSView` is also created to complete the normalization of diverse news feed RSS formats into a single RSS 2.0 format.



2. Open and execute 5\_2\_ExplainsRssViews.sql to see if the SQL/XML queries in the normalized views result in optimized query execution where the XMLIndex and the secondary text index are used.

The screenshot displays the Oracle SQL Developer interface. The left pane shows a file tree with a folder named 'files' containing several SQL files, including '5\_2\_ExplainsRssViews.sql'. The main window shows the SQL Worksheet with the following SQL statement:

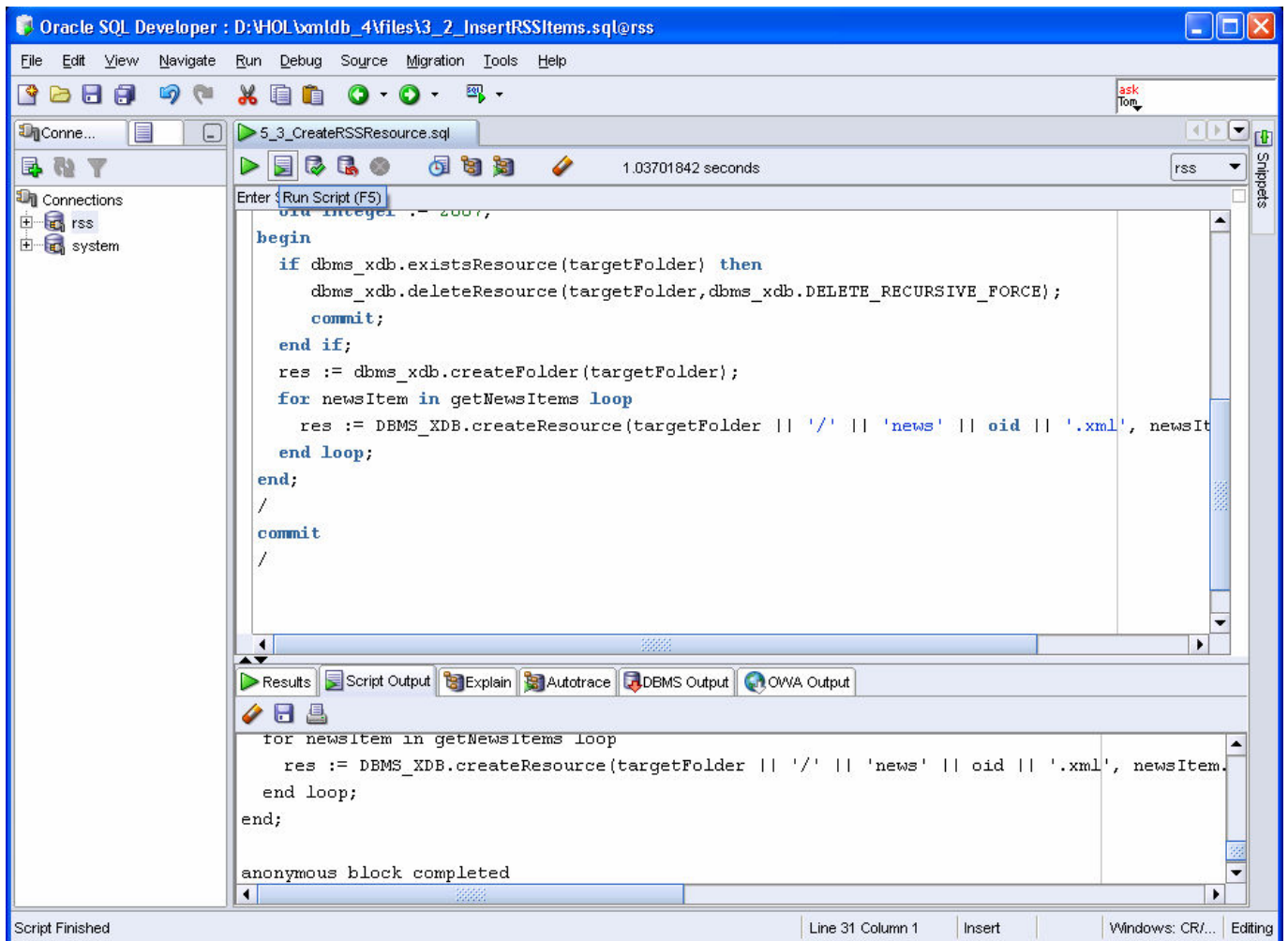
```
-- Run an explain plan to verify that the normalized RSS 2.0 feed
-- has been generated with optimal query execution
explain plan for
select * from newsItemsRSSView
/
set echo off
select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));
set echo on
```

The execution time is 7.68915796 seconds. The 'Autotrace (F10)' button is visible. Below the SQL statement, the 'Results' tab is active, showing the execution plan. The plan is a tree structure with the following operations and object names:

OPERATION	OBJECT_NAME
SORT UNIQUE	
TABLE ACCESS BY INDEX ROWID	NEWSITEM_BIX_PATH_TABLE
BITMAP CONVERSION TO ROWIDS	
BITMAP AND	
BITMAP CONVERSION FROM	
SORT ORDER BY	
DOMAIN INDEX	NEWSITEM_BIX_TEXT_IX
BITMAP CONVERSION FROM	
SORT ORDER BY	
INDEX RANGE SCAN	NEWSITEM_BIX_PATH_ID_IX
TABLE ACCESS BY USER ROWID	NEWSITEMSTABLE

The 'DOMAIN INDEX' operation is highlighted in blue. The bottom status bar shows 'Line 11 Column 1 | Insert | Windows: CR/... Editing'.

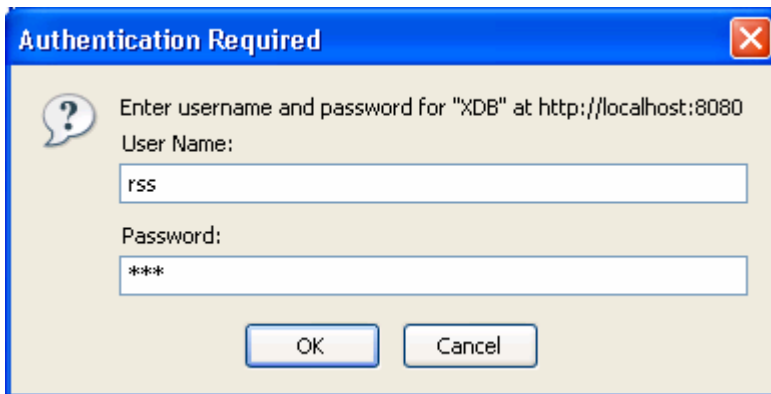
3. Open and execute 5\_3\_CreateRSSResource.sql to create an Oracle XML dB repository resource to represent a normalized RSS feed with aggregated and normalized news items from diverse RSS feeds.

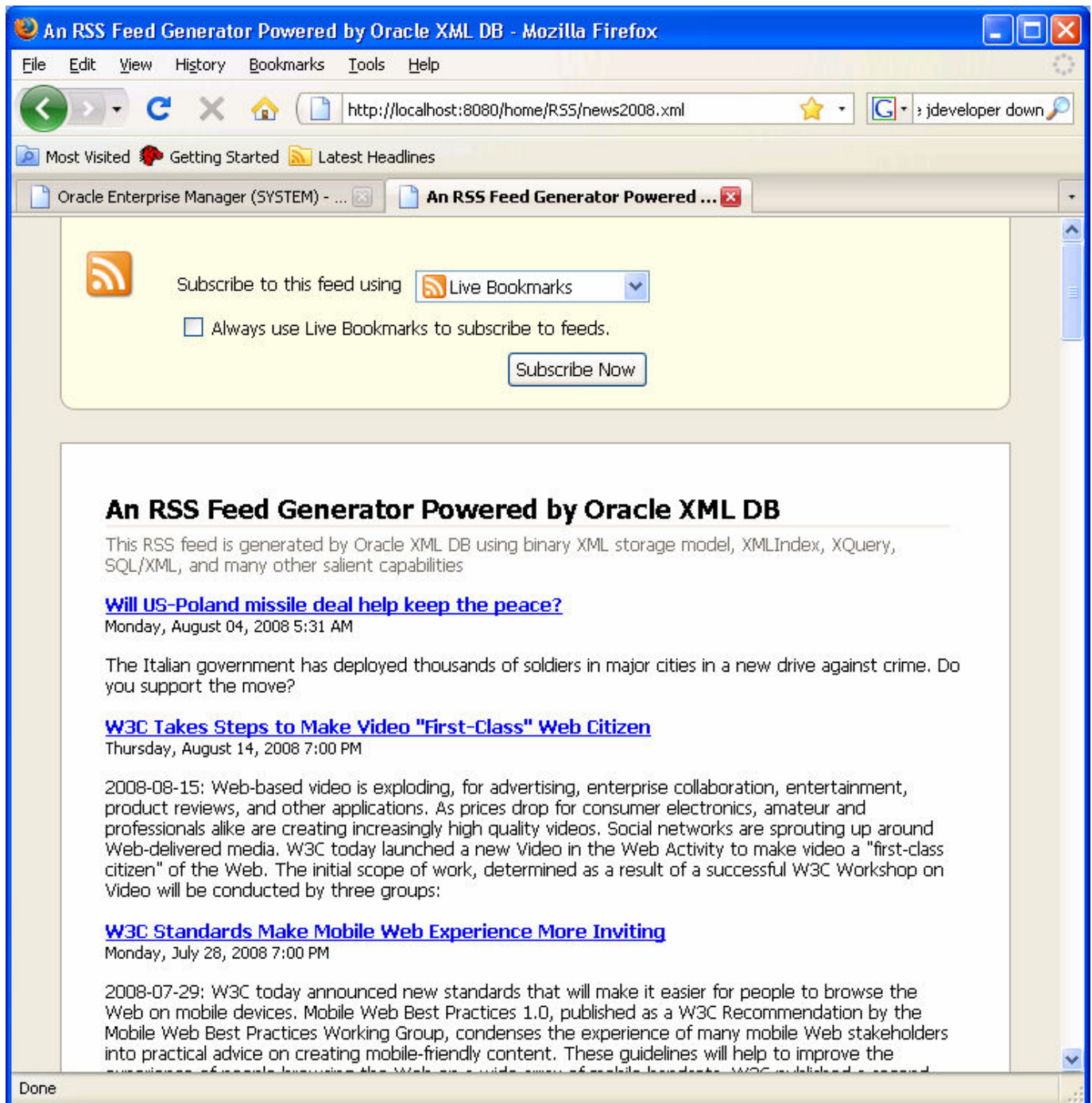


## Reading the Normalized RSS Feed with a Firefox 2.0 Browser

There are several news readers capable of processing RSS 2.0 format. This section uses Firefox 2.0 browser to read the normalized RSS feed.

1. You can open a Firefox 2.0 browser and point to the URL `http://localhost:8080/home/RSS/news2007.xml`. You will see a list of news items aggregated from multiple news feeds. Enter `rss/rss` when prompted for User Name and Password.





Note :The screen shot will be different from what a user gets due to changes in the news.

## Summary

In this tutorial, you learned about the following:

- Creating a new user and grant necessary privileges
- Creating a table to store information about RSS feeds
- Creating a table to store RSS news items
- Creating an XMLIndex on News Items
- Creating an RSS View with Aggregated and Normalized News Items from Diverse News Feeds



# Lesson 5: Using Binary XML, XMLIndex, and XQuery to Store and Query Unstructured XML Data

## Purpose








This workshop shows you how to load, index, and query Medline data in Oracle XML DB.

## Time to Complete

Approximately 60 minutes.

## Topics

This workshop covers the following topics:

-  [Overview](#)
-  [Prerequisites](#)
-  [Loading Medline data into Oracle XML DB](#)
-  [Createing XMLIndex and Full-Text indexes on Medline data](#)
-  [Executing a script of SQL/XML statements with XQuery expressions](#)
-  [Executing a script of updateXML statements](#)
-  [Summary](#)

## Overview

Since Oracle 9i Database Release 2, Oracle XML DB has been seamlessly integrated with the Oracle database to provide high-performance database-native storage, retrieval, and management of XML data. With the new Oracle Database 11g release, Oracle XML DB is taking another leap ahead with a rich set of new capabilities to simplify DBAs' tasks in managing XML data while further empowering XML and SOA application developers. Oracle XML DB now supports multiple database-native XML storage models and XML indexing schemes, SQL/XML standard operations, W3C standard XQuery data model and XQuery/XPath languages, database-native web services, high performance XML publishing, XML DB repository, and versioning and access control. This tutorial covers these topics along with key new features for using Oracle XML DB to store, query, transform, and access XML and relational data.

### Binary XML Storage Model:

Binary XML is a new storage model for abstract data type `XMLType`, joining the existing native storage models of structured (object-relational) and unstructured (CLOB) storage. Binary XML storage provides more efficient database storage, updating, indexing, and fragment extraction than unstructured storage. It can provide better query performance than unstructured storage- it does not suffer from the XML parsing bottleneck (it is a post-parse persistence model). Like structured storage, binary XML storage is aware of XML Schema data types and can take advantage of native database data types. Like unstructured storage, no data conversion is needed during database insertion or retrieval. Like structured storage, binary XML storage allows for piecewise updates. Because binary XML data can also be used outside the database, it can serve as an efficient XML exchange medium, and you can offload work from the database to increase overall performance in many cases. Like unstructured storage, binary XML data is kept in document order. Like structured storage, data and metadata can, using binary storage, be separated at the database level, for efficiency. Like unstructured storage, however, binary storage allows for intermingled data and metadata, which lets instance structures vary. Binary XML storage allows for very complex and variable data, which in the structured-storage model could necessitate using many database tables and joins. Unlike the other `XMLType` storage models, you can use binary storage for XML schema-based data even if the XML schema is not known beforehand, and you can store multiple XML schemas in the same table and query across common elements.



### **XMLIndex Indexing for Binary XML and Unstructured XML Storage Models:**

B-Tree indexes can be used advantageously with structured storage. They provide sharp focus by targeting the underlying objects directly. They are generally ineffective, however, in addressing the detailed structure (elements and attributes) of an XML document stored in a binary XML or a CLOB instance. That is the special domain of XMLIndex: binary XML and unstructured storage models. Unlike a B-Tree index, which you define for a specific column that represents an individual XML element or attribute, an XMLIndex index is very general: indexing with XMLIndex applies to all possible XPath expressions for your XML data. An XMLIndex index presents the following advantages over other indexing methods:

- An XMLIndex index can be used for SQL/XML functions `XMLExists()`, `XMLTable()`, and `XMLQuery()`, and it is effective in any part of a query; it is not limited to use in a `WHERE` clause. This is not the case for any of the other kinds of indexes you might use with XML data.
- XMLIndex can thus speed access to `SELECT` list data and `FROM` list data, making it useful for XML fragment extraction, in particular. Function-based indexes and CTXXPath indexes
- You need no prior knowledge of the XPath expressions that will be used in queries. XMLIndex is completely general. This is not the case for function-based indexes.
- You can use an XMLIndex index with either XML schema-based or non-schema-based data. It can be used with binary XML and unstructured storage models. B-Tree indexing is appropriate only for schema-based data stored object-relationally (structured storage); it is ineffective for XML schema-based data stored in a binary XML or a CLOB instance.
- You can use an XMLIndex index for searches with XPath expressions that target collections, that is, nodes that occur multiple times within a document. This is not the case for functional indexes.

### **Oracle Database-Native XQuery:**

Since XQuery is now a W3C standard, the IT community has started adopting the business uses of XML and XQuery. As the innovation leader in commercial database technology, Oracle Database 11g provides a full-featured native XQuery engine integrated with the traditional Oracle database server. On the SQL side, the SQL/XML standard has defined a way to encapsulate XML in SQL and to integrate the querying of XML using XQuery. This is being accomplished by introducing new SQL functions: `XMLQuery`, `XMLTable`, `XMLExists`, and `XMLCast`, which operate on XML and SQL values using XQuery. Oracle Database 11g enables XQuery support in the database server through these SQL standard functions. A new `XQUERY` command has also been implemented in SQL\*Plus to allow users to enter XQuery expressions on the command line. With standards-based implementation of XQuery in Oracle Database 11g, application developers can use their favorite APIs (e.g., JDBC, ODP.NET, and web service) to access Oracle Database XQuery capabilities.

### **Benefits of Oracle XQuery:**

Using SQL/XML XQuery functions along with indexing schemes for structured, unstructured, and binary XML storage models, XML DB can perform uniform XML queries across different storage models with orders of magnitude performance improvement over DOM-based functional evaluation of XML queries. Furthermore, XML queries can be seamlessly merged with SQL relational queries to handle all query scenarios. Finally, the XML query capabilities of Oracle XML DB are built on the solid foundation of industry's best relational database that is highly reliable, available, scalable, and secure. In short, the XML DB query capabilities in Oracle Database 11g provide the most comprehensive and efficient functionality for versatile, scalable, concurrent, and high performance XML applications.

## Prerequisites

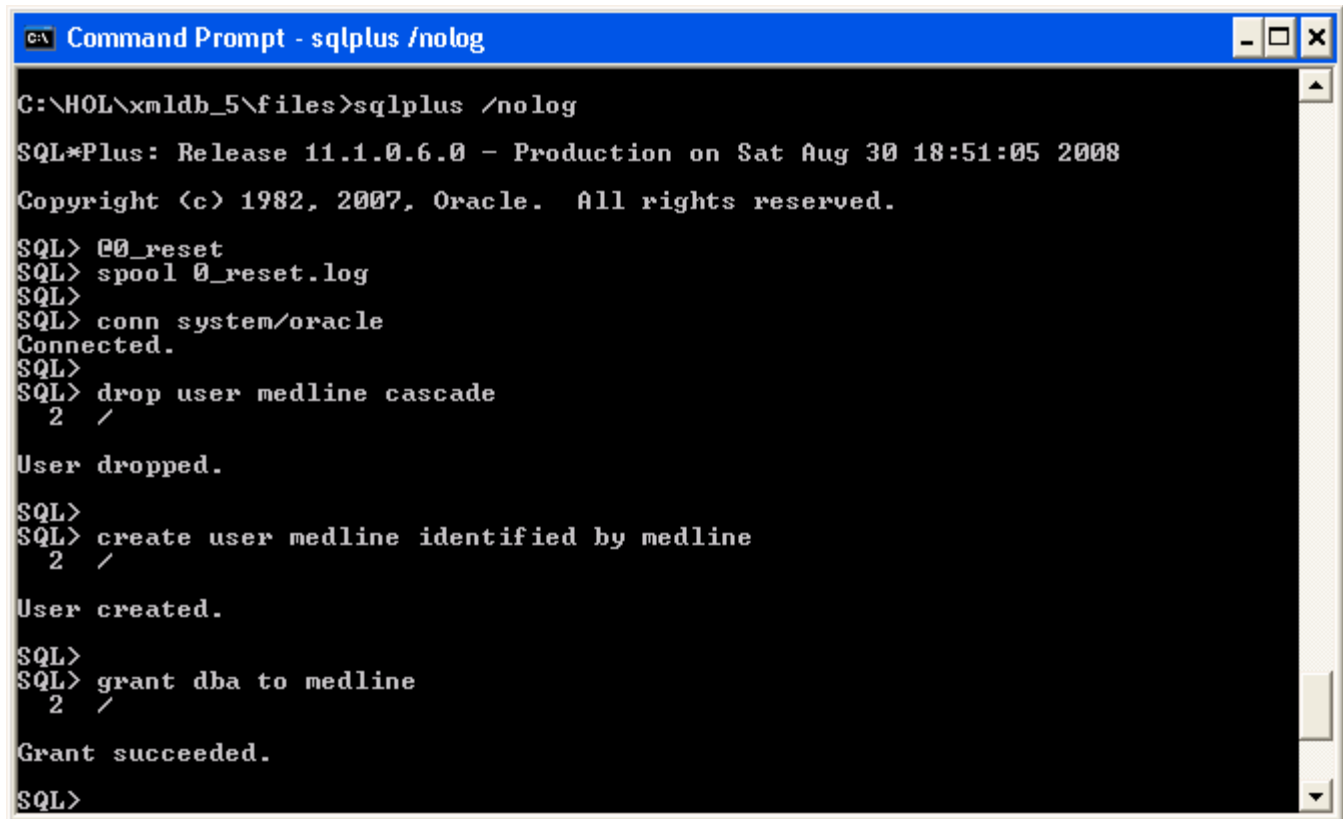
Before you perform this tutorial, you should first complete the following steps:

1. Check your Oracle Database 11g installation
2. Check your Oracle SQL Developer (version 1.5.1) installation
3. Check the files in your working directory (c:\HOL\xmldb\_5\files)

## Loading Medline Data

In this section, you create a binary XML table, and insert rows from a Medline distribution. You use Oracle SQL Developer throughout this tutorial. Perform the following steps:

Using SQLPlus to run a setup script `0_reset.sql` located in `C:\HOL\xmldb_5\files`:



```
C:\HOL\xmldb_5\files>sqlplus /nolog
SQL*Plus: Release 11.1.0.6.0 - Production on Sat Aug 30 18:51:05 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.

SQL> @0_reset
SQL> spool 0_reset.log
SQL>
SQL> conn system/oracle
Connected.
SQL>
SQL> drop user medline cascade
2 /

User dropped.

SQL>
SQL> create user medline identified by medline
2 /

User created.

SQL>
SQL> grant dba to medline
2 /

Grant succeeded.

SQL>
```

[Start SQL Developer](#)

[Loading Medline Data into a Binary XML Storage Column](#)

### Start SQL Developer

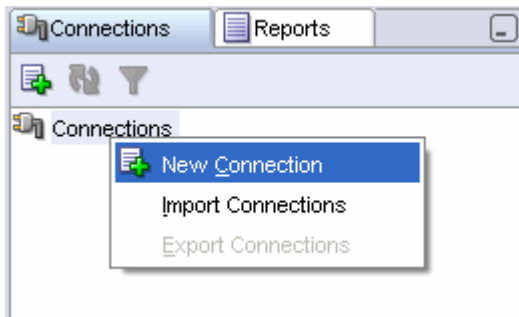
Perform the following steps:

1. Click on the SQL Developer icon on the desktop to start the application.



2. You must create a database connection as OE user. Perform the following steps.

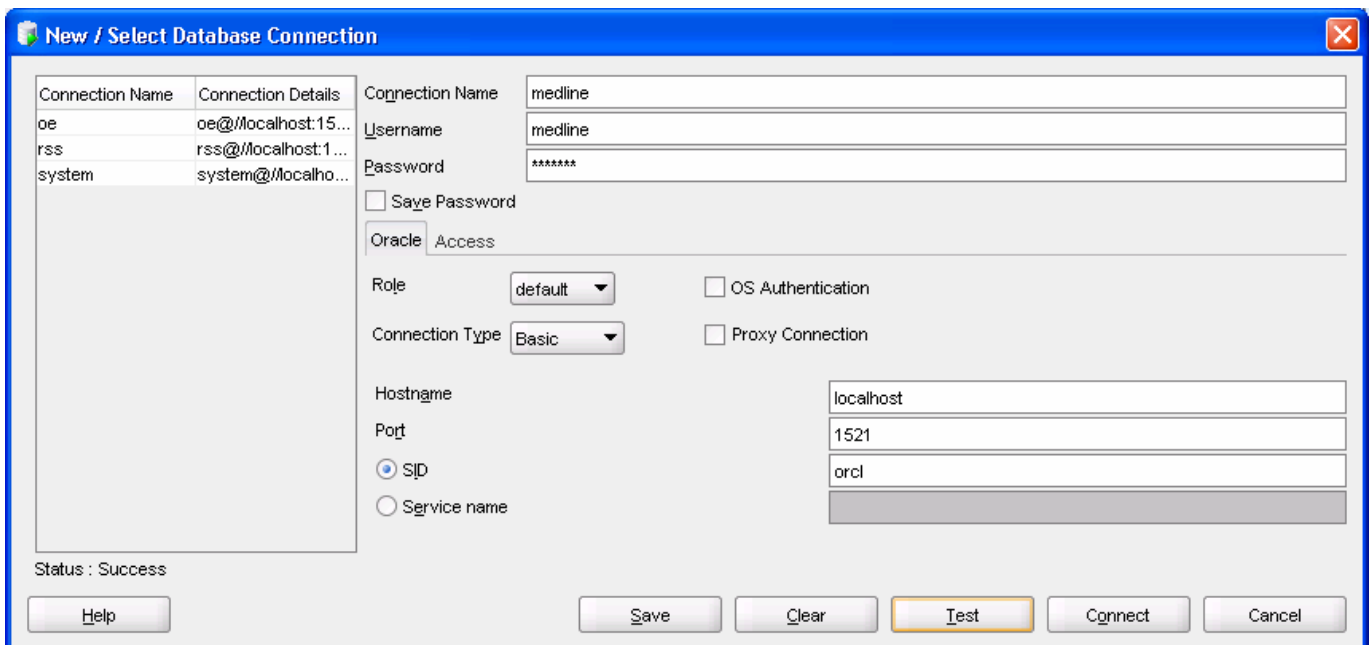
- a. In the Connections tab, right-click **Connections** and select **New Connection**.



b. The New/Select Database Connection window appears. Enter the following details, and click **Test** to make sure that the connection has been set correctly.

Connection Name: **medline**  
 Username: **medline**  
 Password: **medline**  
 Hostname: **localhost**  
 Port: **1521**  
 SID: **orcl**

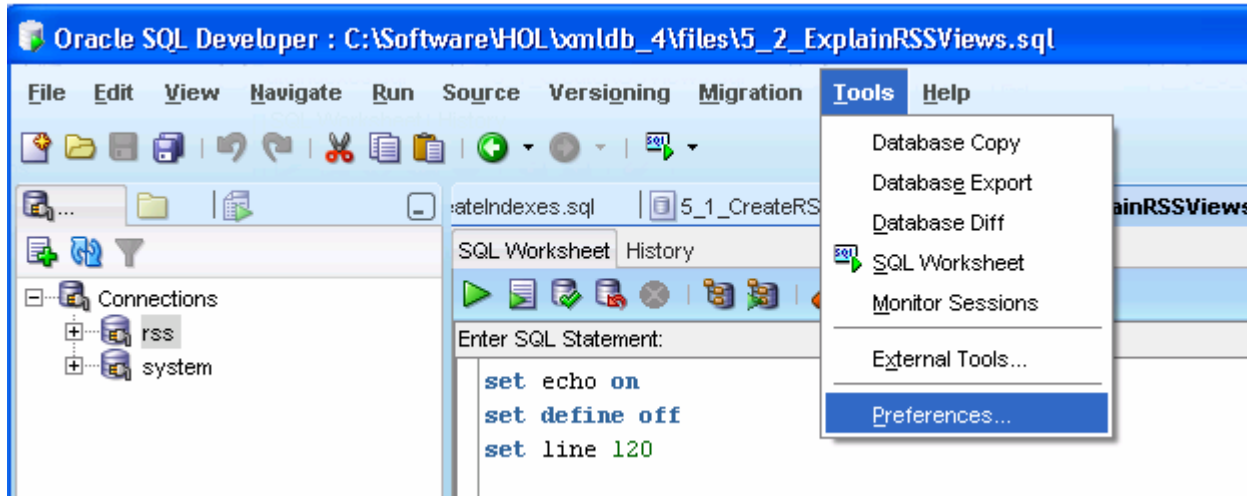
If you select the Save Password check box, the password is saved to an XML file. Therefore, once you close SQL Developer connection and open again, you will not be prompted for the password.



c. The test status shows success. Click **Connect**.

3. Set the Autotrace parameters. Perform the following steps:

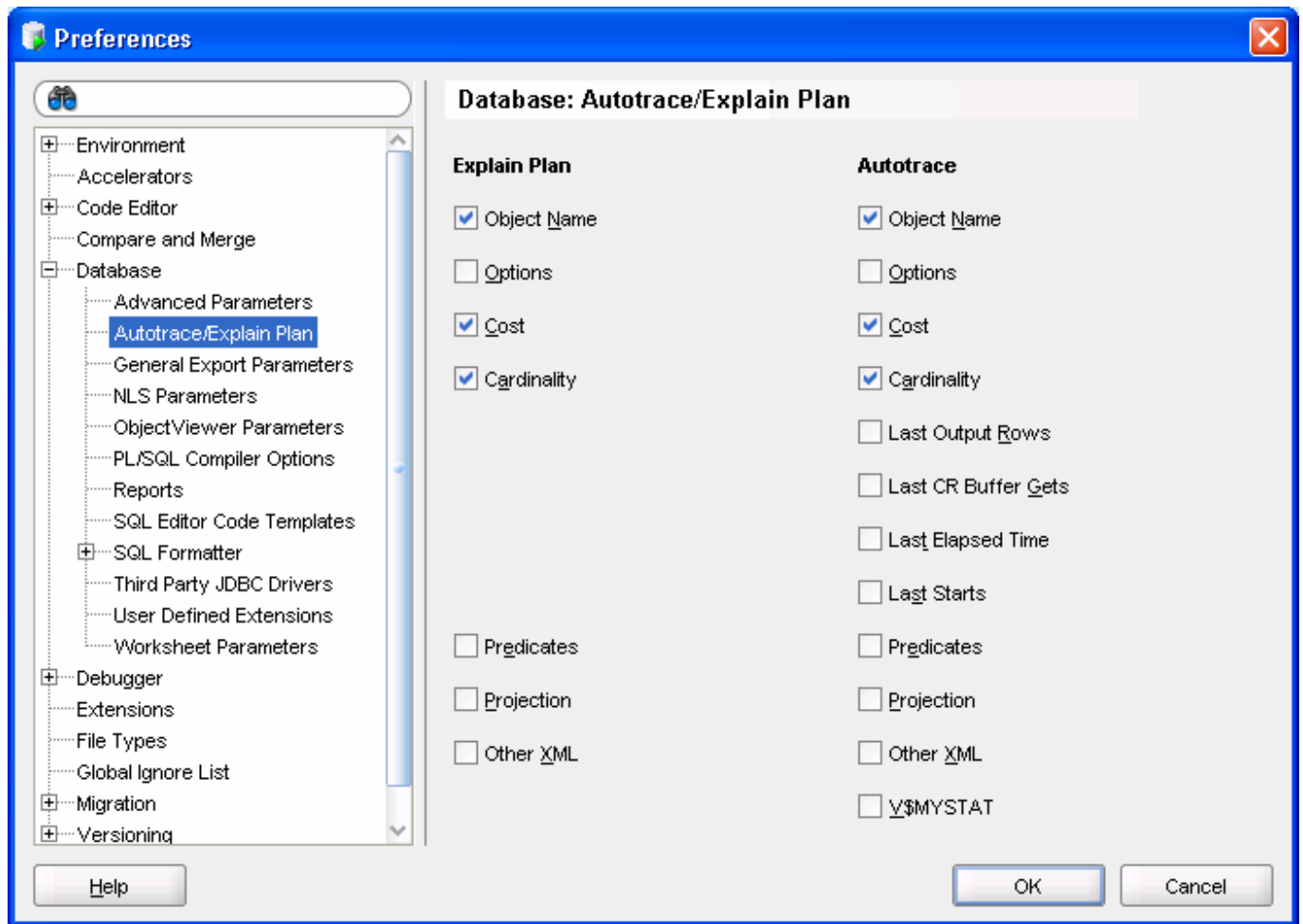
a. Go to **Tools > Preferences**.



b. Expand **Database**, and select **Autotrace Parameters**.

c. Make sure to select the following check boxes and click **OK**.

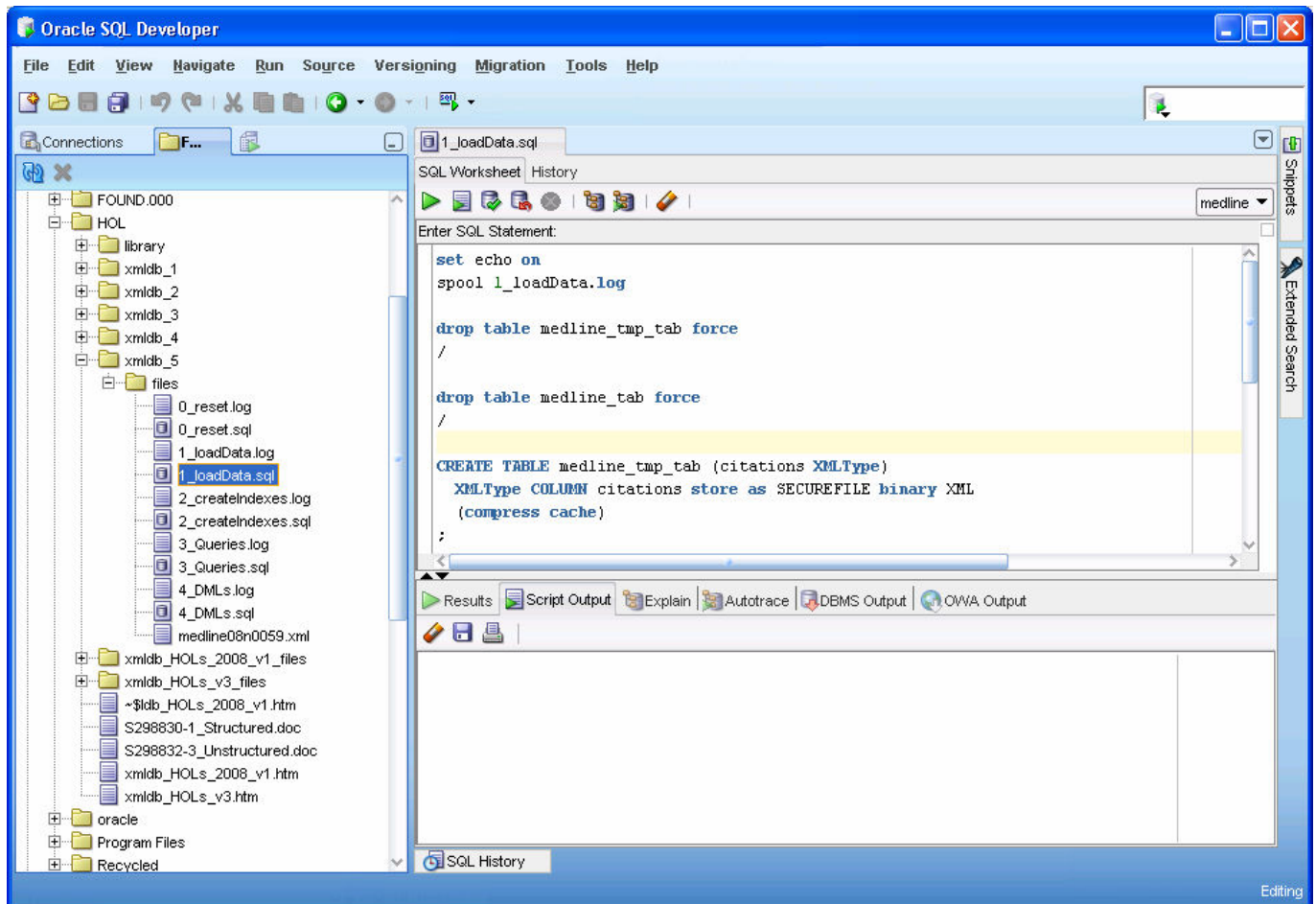
Object\_Name  
Cost  
Cardinality



In the above section, you learned how to connect to SQL Developer, and set Autotrace parameters.

## Load Medline Data into Binary XML Storage

1. Create a temporary table with an `XMLType` column using the binary XML storage to load the Medline data, and create another table with an `XMLType` column (using binary XML storage) to store all the citations extracted from the Medline data. Run the script `1_loadData.sql` after selecting it from the **Files** tab:



c. The code is displayed in the Enter SQL Statement box of the SQL Worksheet. Click the **Run Script** icon. Alternatively, you can press **F5**. Note the results that are displayed under the Script Output pane.

### 1\_loadData.sql

```
set echo on
spool 1_loadData.log

conn oraxdb/oraxdb

drop table medline_tmp_tab force
/

drop table medline_tab force
/

CREATE TABLE medline_tmp_tab (citations XMLType)
XMLType COLUMN citations store as SECUREFILE binary XML
(compress cache)
```

```

;

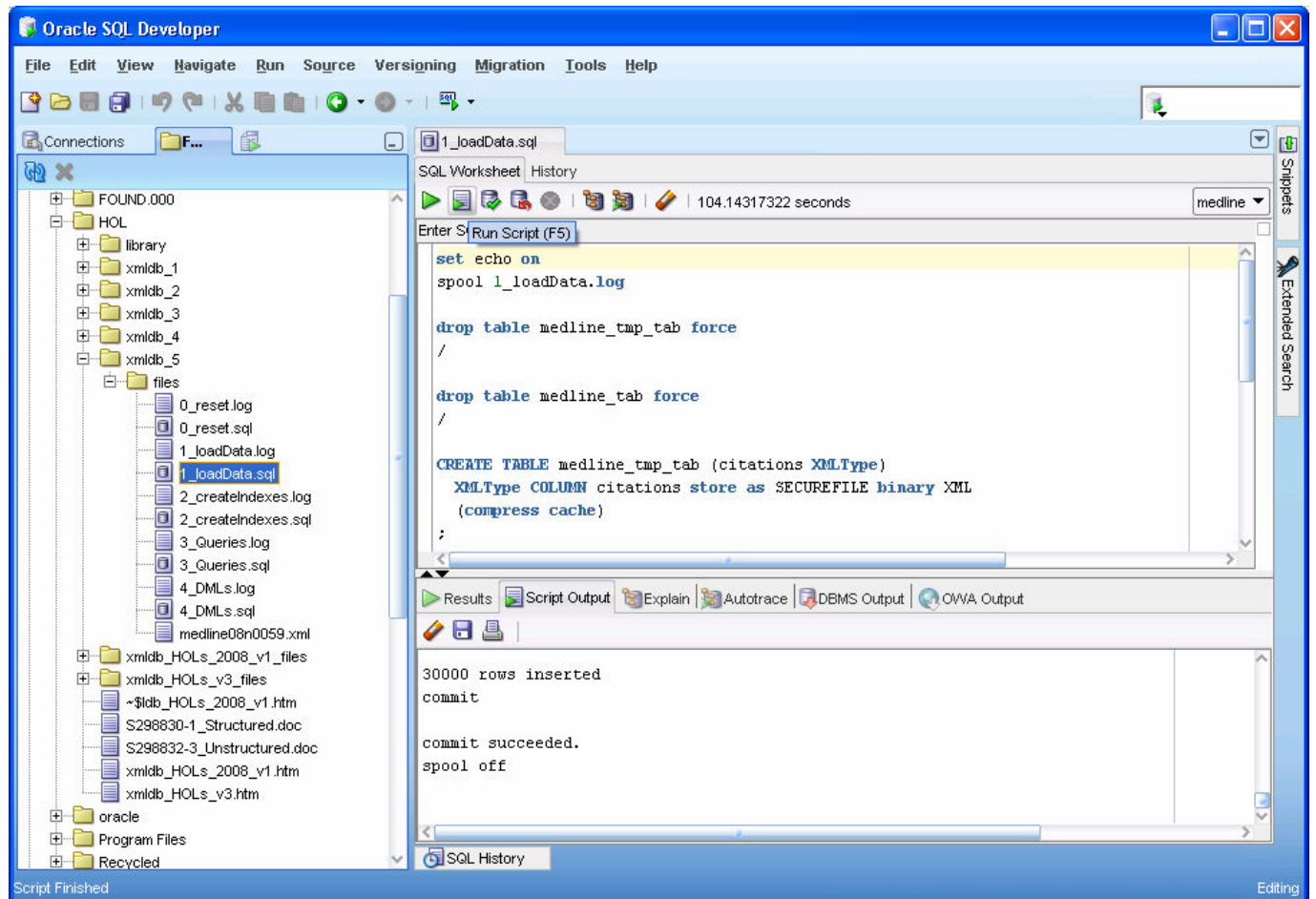
create or replace directory xmldir as 'c:\OLSUG 2008\Oracle XML DB'
/

insert into medline_tmp_tab values(xmltype(bfilename('XMLDIR','medline08n0059.xml'),
nls_charset_id('AL32UTF8')));

create table medline_tab (citation xmltype)
xmltype column citation store as securefile binary xml
(compress cache)
;

insert into medline_tab select x.column_value from medline_tmp_tab,
xmltable('//MedlineCitation' passing citations) x
/
commit
/
spool off

```



Note: Throughout this workshop, to execute the script files in SQL Developer, you must follow the above steps. If you want to run a single statement at the mouse pointer, click the **Execute Statement** icon. Alternatively, move the cursor over the statement, and press **F9**.



## Improving Performance of XQuery Expressions through Index Creation

You can improve the performance of your XQuery by creating indexes. In this section, you will create an `XMLIndex` index on the binary XML column and a secondary Full-Text index on the value of the XMLIndex. You will then run the SQL/XML scripts with XQuery expressions against the binary XML column to see the explain plan and note how the query and DML executions have taken advantage of the XMLIndex and full-text index. Perform the following steps:

1. In your SQLDeveloper session, connect as `OE` user. Then, execute the script `2_createIndexes.sql`.

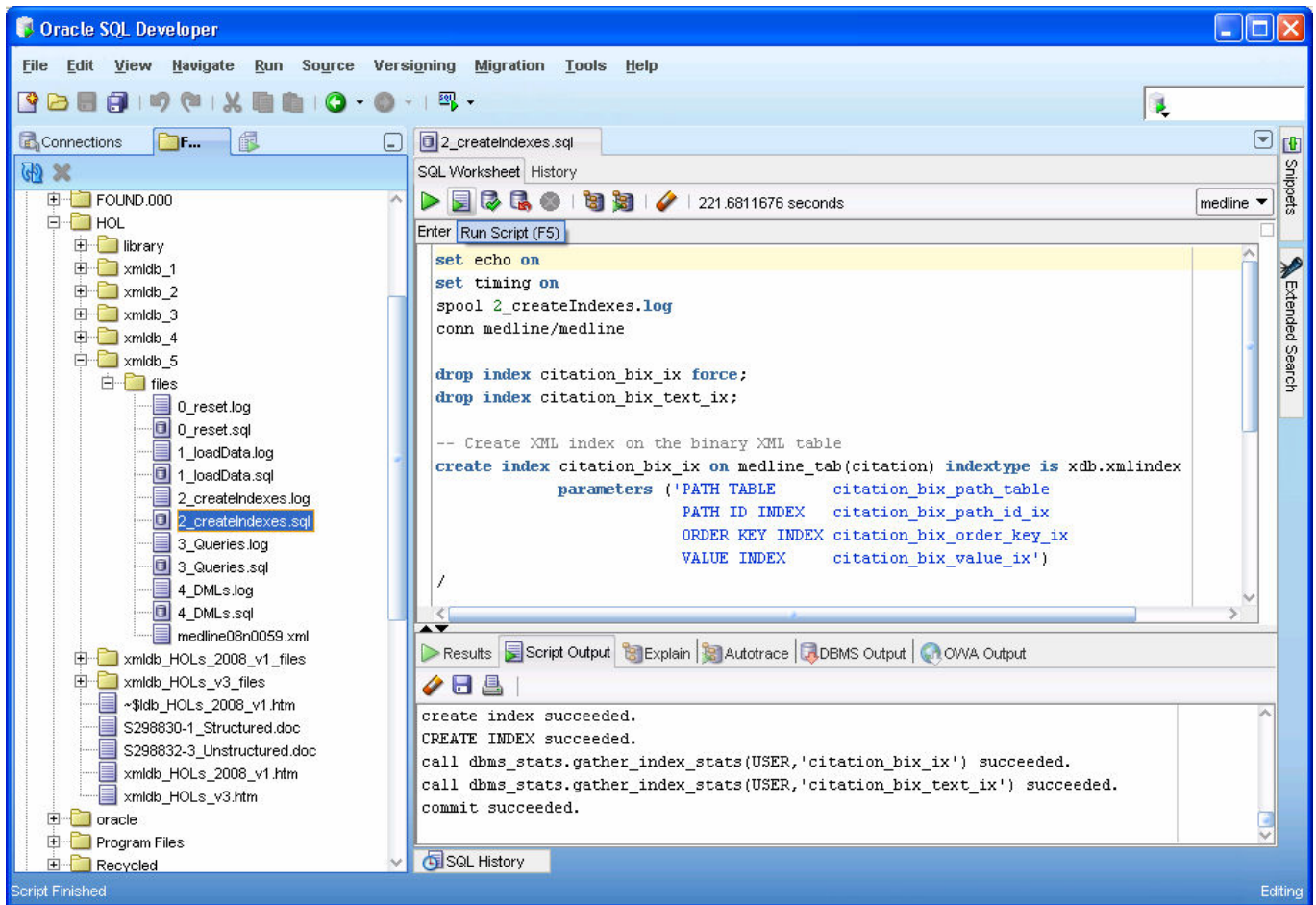
```
set echo on
set timing on
spool 2_createIndexes.log
conn oraxdb/oraxdb

drop index citation_bix_ix force;
drop index citation_bix_text_ix;

-- Create XML index on the binary XML table
create index citation_bix_ix on medline_tab(citation) indextype is xdb.xmlindex
    parameters ('PATH TABLE      citation_bix_path_table
                  PATH ID INDEX    citation_bix_path_id_ix
                  ORDER KEY INDEX  citation_bix_order_key_ix
                  VALUE INDEX      citation_bix_value_ix')
/

-- Create a secondary text index on the VALUE column of the path table
CREATE INDEX citation_bix_text_ix ON citation_bix_path_table (VALUE)
    INDEXTYPE IS CTXSYS.CONTEXT
    parameters ('transactional')
/

call dbms_stats.gather_index_stats(USER,'citation_bix_ix')
/
call dbms_stats.gather_index_stats(USER,'citation_bix_text_ix')
/
commit
/
spool off
```



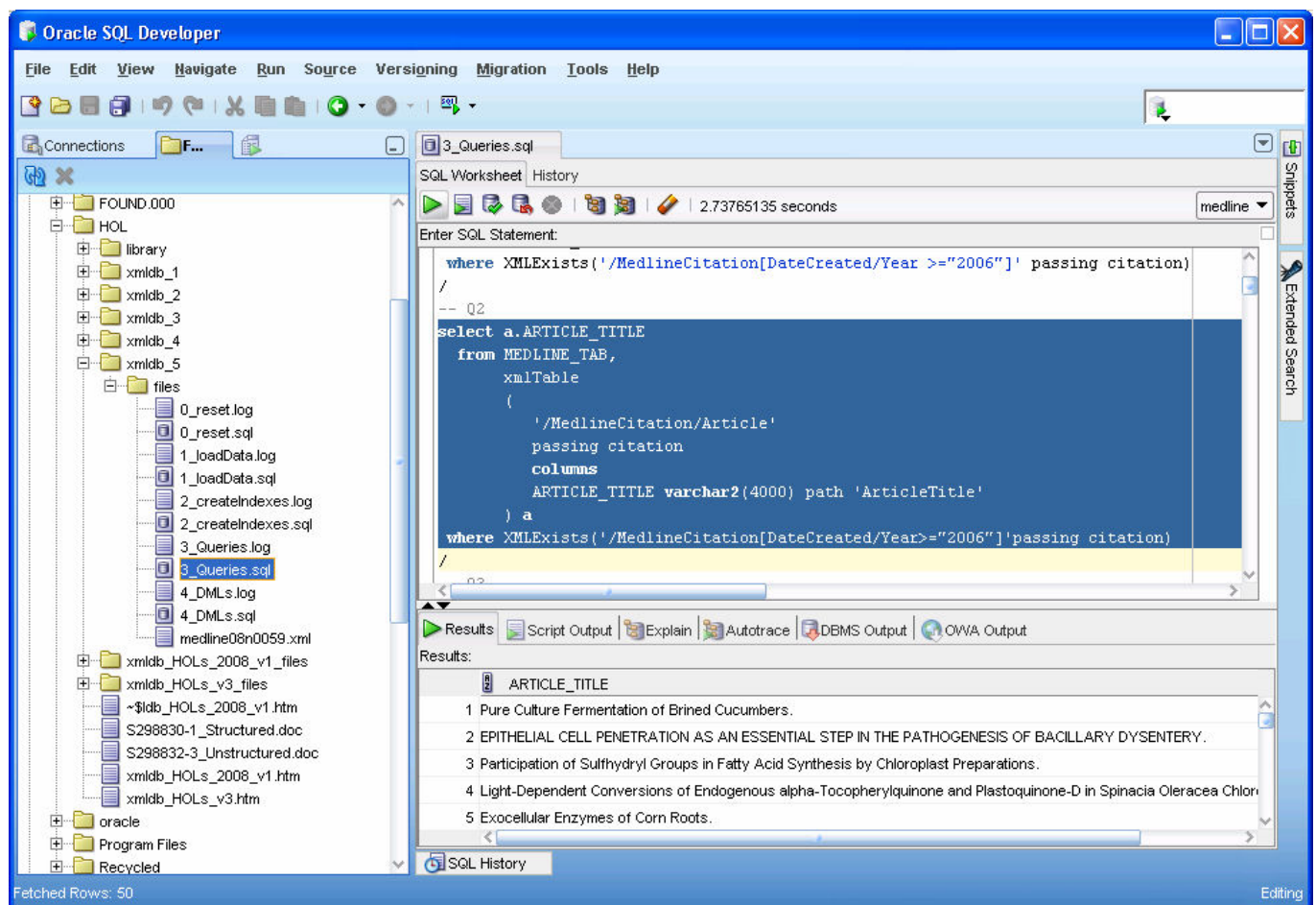
2. Now that you created indexes, you can view the explain plan to observe the performance of SQL/XML scripts with XQuery expressions. Observe that the explain plan picks up the applicable indexes.

First, view the execution of the query that retrieves the article titles of Medline citations. In the **Enter SQL Statement** box, perform the following steps:

- a. Open the file **3\_Queries.sql**. The code is displayed in the **Enter SQL Statement** box. Select the No. 2 query (-- Q2) and then click the **Execute Statement** button.

Code snippet in 3\_Queries.sql:

```
-- Q2
select a.ARTICLE_TITLE
  from MEDLINE_TAB,
       xmlTable
       (
         '/MedlineCitation/Article'
         passing citation
         columns
         ARTICLE_TITLE varchar2(4000) path 'ArticleTitle'
       ) a
 where XMLEExists('/MedlineCitation[DateCreated/Year>="2006"]' passing citation)
/
```



The screenshot shows the Oracle SQL Developer interface. The left pane displays a file tree with the following structure:

- FOUND.000
  - HOL
    - library
    - xmladb\_1
    - xmladb\_2
    - xmladb\_3
    - xmladb\_4
    - xmladb\_5
    - files
      - 0\_reset.log
      - 0\_reset.sql
      - 1\_loadData.log
      - 1\_loadData.sql
      - 2\_createIndexes.log
      - 2\_createIndexes.sql
      - 3\_Queries.log
      - 3\_Queries.sql
      - 4\_DMLs.log
      - 4\_DMLs.sql
      - medline08n0059.xml
  - xmladb\_HOLs\_2008\_v1\_files
  - xmladb\_HOLs\_v3\_files
  - ~\$ldb\_HOLs\_2008\_v1.htm
  - S298830-1\_Structured.doc
  - S298832-3\_Unstructured.doc
  - xmladb\_HOLs\_2008\_v1.htm
  - xmladb\_HOLs\_v3.htm
- oracle
- Program Files
- Recycled

The main pane shows the SQL Worksheet with the following code:

```
where XMLEExists('/MedlineCitation[DateCreated/Year >="2006"]' passing citation)
/
-- Q2
select a.ARTICLE_TITLE
  from MEDLINE_TAB,
       xmlTable
       (
         '/MedlineCitation/Article'
         passing citation
         columns
         ARTICLE_TITLE varchar2(4000) path 'ArticleTitle'
       ) a
 where XMLEExists('/MedlineCitation[DateCreated/Year>="2006"]' passing citation)
/
```

The Results pane displays the following data:

ARTICLE_TITLE
1 Pure Culture Fermentation of Brined Cucumbers.
2 EPITHELIAL CELL PENETRATION AS AN ESSENTIAL STEP IN THE PATHOGENESIS OF BACILLARY DYSENTERY.
3 Participation of Sulfhydryl Groups in Fatty Acid Synthesis by Chloroplast Preparations.
4 Light-Dependent Conversions of Endogenous alpha-Tocopherylquinone and Plastoquinone-D in Spinacia Oleracea Chloroplasts.
5 Exocellular Enzymes of Corn Roots.

Fetches Rows: 50

b. With the same selection of the Q2 query, now click the **Autotrace** icon. Note the usage of XMLIndex index.

The screenshot shows the Oracle SQL Developer interface. The left pane displays a file tree with folders like 'FOUND.000', 'HOL', 'library', and 'files'. The main window shows an SQL worksheet with a query named '3\_Queries.sql'. The query is as follows:

```
-- Q2
select a.ARTICLE_TITLE
  from MEDLINE_TAB,
       xmlTable
       (
         '/MedlineCitation/Article'
         passing citation
         columns
         ARTICLE_TITLE varchar2(4000) path 'ArticleTitle'
       ) a
 where XMLExists('/MedlineCitation[DateCreated/Year >="2006"]' passing citation)
```

The query has been executed, and the Autotrace icon (F10) has been clicked. The bottom pane shows the execution plan for the query. The plan is as follows:

OPERATION	OBJECT_NAME	CC
SELECT STATEMENT		
FILTER		
TABLE ACCESS BY INDEX ROWID	CITATION_BIX_PATH_TABLE	
INDEX RANGE SCAN	CITATION_BIX_PATH_ID_IX	
NESTED LOOPS		
NESTED LOOPS		
NESTED LOOPS		
VIEW	VW_SQ_1	

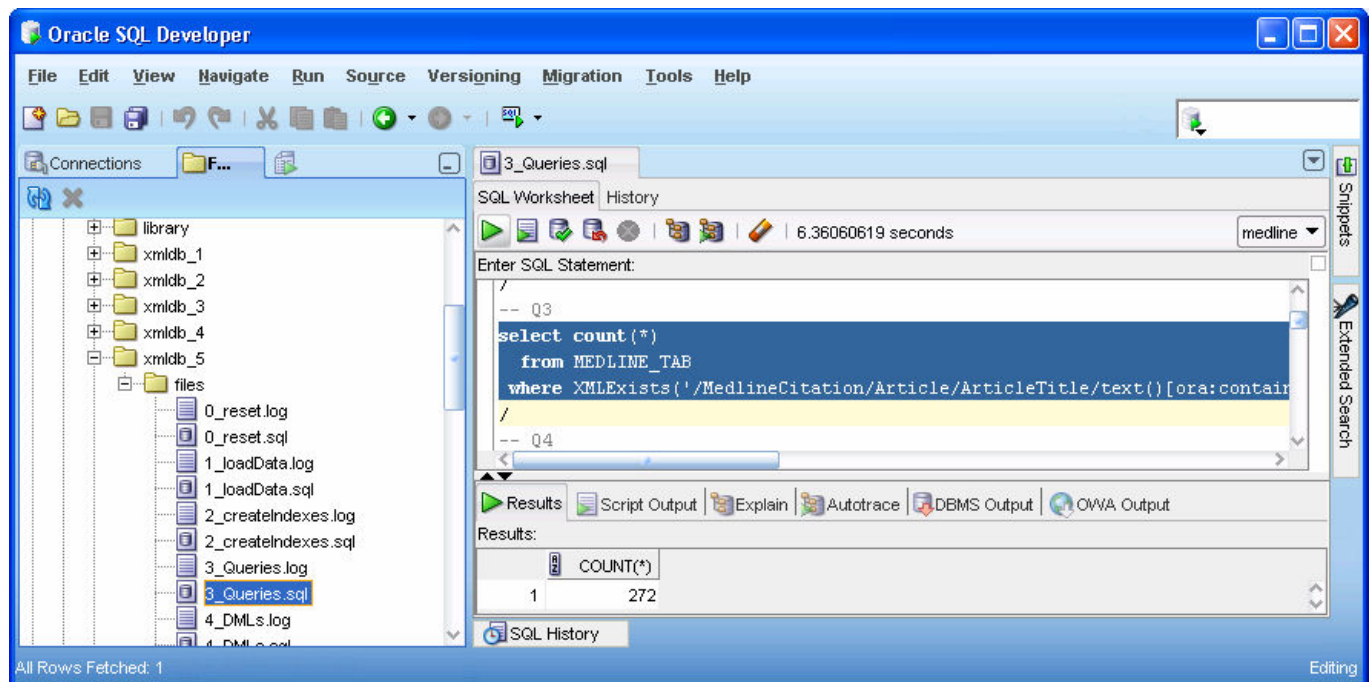
The status bar at the bottom indicates 'Line 23 Column 1 | Insert | Windows: CR/... Editing'.

3. We can follow the same steps described above on another query to view the query results and the execution plan of the query.

a. Scroll down to select query No. 3 (-- Q3) in the opened file **3\_Queries.sql**. The code is highlighted in the **Enter SQL Statement** box. Now, click the **Executed Statement** button to see the results of this query.

Code snippet in 3\_Queries.sql:

```
-- Q3
select count(*)
  from MEDLINE_TAB
 where XMLEExists('/MedlineCitation/Article/ArticleTitle/text() [ora:contains(., "tuberculosis")
 > 0]' passing citation)
/
```



b. With the same selection of the Q16 query, now click the **Autotrace** button. Note the usage of **XMLIndex** and Full-Text indexes.

Oracle SQL Developer

File Edit View Navigate Run Source Versing Migration Tools Help

Connections F...

library  
xmldb\_1  
xmldb\_2  
xmldb\_3  
xmldb\_4  
xmldb\_5  
files  
0\_reset.log  
0\_reset.sql  
1\_loadData.log  
1\_loadData.sql  
2\_createIndexes.log  
2\_createIndexes.sql  
3\_Queries.log  
3\_Queries.sql  
4\_DMLs.log  
4\_DMLs.sql  
medline08n0059.xml  
xmldb\_HOLs\_2008\_v1\_files  
xmldb\_HOLs\_v3\_files  
~\$ldb\_HOLs\_2008\_v1.htm  
S298830-1\_Structured.doc  
S298832-3\_Unstructured.doc  
xmldb\_HOLs\_2008\_v1.htm  
xmldb\_HOLs\_v3.htm  
oracle

3\_Queries.sql

SQL Worksheet History

0.85325152 seconds medline

Enter SQL Statement:

Autotrace (F10)

```
-- Q3
select count(*)
from MEDLINE_TAB
where XMLElementExists('/MedlineCitation/Article/ArticleTitle/text()')ora:contains(., "t
/
-- Q4
select count(*)
```

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION

OPERATION	OBJECT_NAME
TABLE ACCESS BY INDEX ROWID	CITATION_BIX_PATH_TABLE
BITMAP CONVERSION TO ROWIDS	
BITMAP AND	
BITMAP CONVERSION FROM ROWIDS	
SORT ORDER BY	
INDEX RANGE SCAN	CITATION_BIX_PATH_ID_IX
BITMAP CONVERSION FROM ROWIDS	
SORT ORDER BY	
DOMAIN INDEX	CITATION_BIX_TEXT_IX
TABLE ACCESS BY USER ROWID	MEDLINE_TAB

SQL History

All Rows Fetched: 1

Editing



4. We can now take a look at examples that perform DML operations on the Medline data and see how their executions are also optimized.

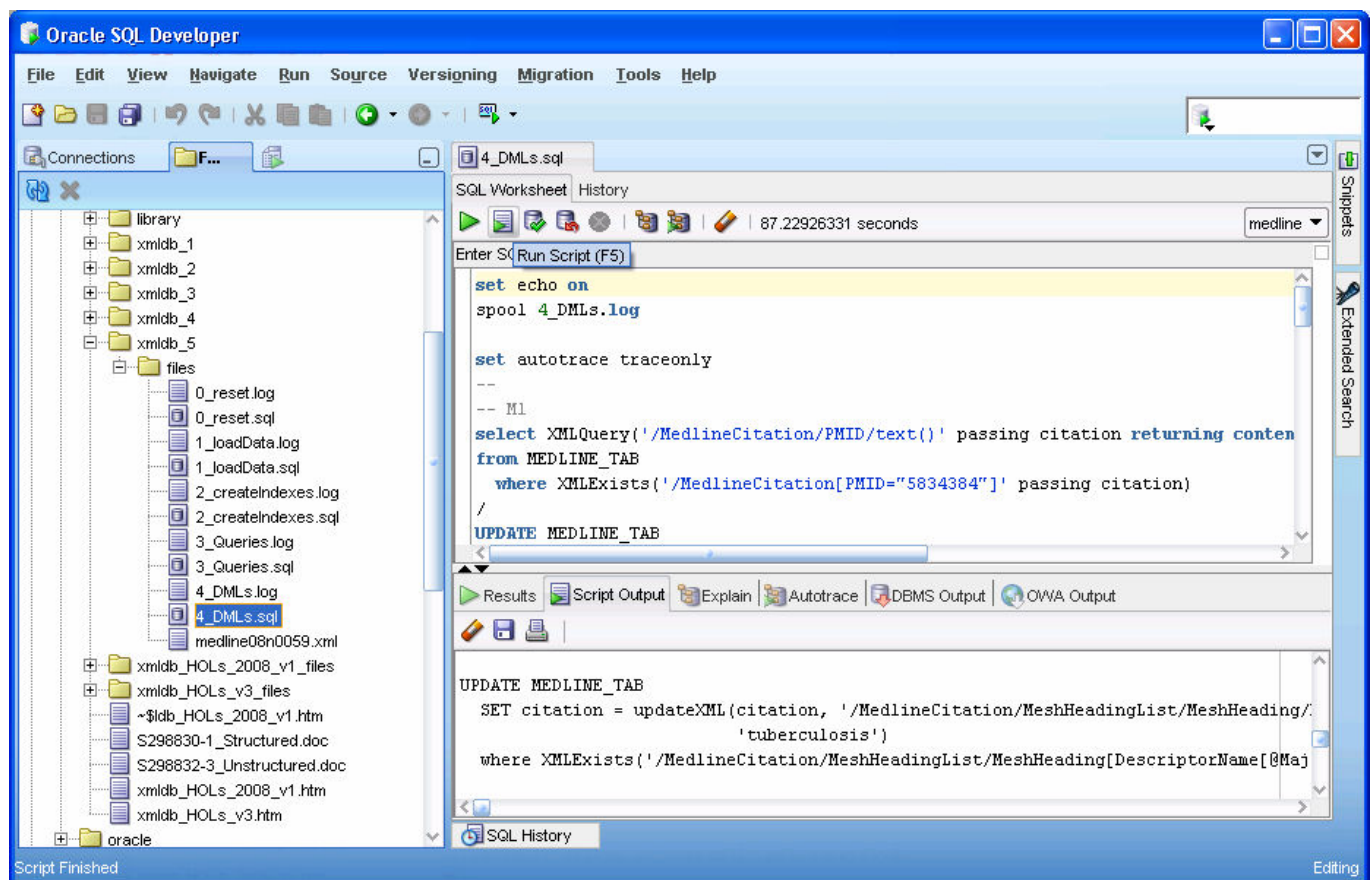
a. Open the file **4\_DMLs.sql**. The code is displayed in the **SQL Worksheet** box. Now, click the **Autotrace** icon.

Code snippet in 4\_DMLs.sql:

```
-- M1

UPDATE MEDLINE_TAB
  SET citation = updateXML(citation, '/MedlineCitation/PMID/text() [.="5834384"]', '5834384_0')
  where XMLEExists('/MedlineCitation[PMID="5834384"]' passing citation)
/

commit
/
```



b. Using the same selection of the Update statement in the **Enter SQL Statement** box, click the **Autotrace** button. You can see how

Oracle SQL Developer

File Edit View Navigate Run Source Versing Migration Tools Help

Connections F...

library  
xmldb\_1  
xmldb\_2  
xmldb\_3  
xmldb\_4  
xmldb\_5  
files  
0\_reset.log  
0\_reset.sql  
1\_loadData.log  
1\_loadData.sql  
2\_createIndexes.log  
2\_createIndexes.sql  
3\_Queries.log  
3\_Queries.sql  
4\_DMLs.log  
4\_DMLs.sql  
medline08n0059.xml  
xmldb\_HOLs\_2008\_v1\_files  
xmldb\_HOLs\_v3\_files  
~\$ldb\_HOLs\_2008\_v1.htm  
S298830-1\_Structured.doc  
S298832-3\_Unstructured.doc  
xmldb\_HOLs\_2008\_v1.htm  
xmldb\_HOLs\_v3.htm  
oracle

4\_DMLs.sql

SQL Worksheet History

2.01800728 seconds medline

Enter SQL Statement:

```

select XMLQuery('/MedlineCitation/PMID/text()' passing citation returning conten
from MEDLINE_TAB
where XMLEExists('/MedlineCitation[PMID="5834384"]' passing citation)
/

UPDATE MEDLINE_TAB
SET citation = updateXML(citation, '/MedlineCitation/PMID/text()[.="5834384"]'
where XMLEExists('/MedlineCitation[PMID="5834384"]' passing citation)
/

```

Autotrace (F10)

Results Script Output Explain Autotrace DBMS Output OWA Output

OPERATION	OBJECT_NAME
NESTED LOOPS	
NESTED LOOPS	
FAST DUAL	
TABLE ACCESS BY INDEX ROWID	CITATION_BIX_PATH_TABLE
INDEX RANGE SCAN	CITATION_BIX_VALUE_IX
TABLE ACCESS BY INDEX ROWID	CITATION_BIX_PATH_TABLE
INDEX RANGE SCAN	CITATION_BIX_PATH_ID_IX
TABLE ACCESS BY USER ROWID	MEDLINE_TAB

SQL History

1 rows updated

Editing



## Summary

In this workshop, you learned how to:

- Load Medline data file into Oracle XML DB
- Create a binary XML storage to store Medline data
- Add XMLIndex and Full-Text indexes to improve the performance of XQuery expressions and XML updates
- Execute SQL/XML scripts with XQuery expressions to observe optimal execution plans
- Execute updateXML statements to observe optimal execution plans