

Oracle XML DB 11gR1 Advanced Features

An Oracle White Paper
October 2007

Introduction	2
Architecture	2
XMLType Partitioning	2
XMLSchema Evolution	3
Installation	4
Server Pre-requisites	4
Oracle Software	4
Client Pre-requisites	4
Oracle Software	4
Non Oracle Software	4
Database Configuration Changes	5
Installing the Application	5
Using the demonstration framework	9
Performing the demonstration	11
0.1 Initialize Demo	12
1.1 View XML Schema (XFILES)	13
1.2 Register XML Schema	15
1.3 Add Partitions	17
1.4 SQL-Loader Control File	18
1.5 Load Data	19
1.6 View Log	20
1.7 Count Documents by Partition	20
2.1 PurchaseOrder from Cost-Center E999	21
2.2 Diff XML Schemas	23
2.3 View Differences	24
2.4 Add Cost-Center E999	25
3.1 Revised Purchase Order Documents	27
3.2 Load Revised PurchaseOrders	28
3.3 Diff XML Schemas	29
3.4 View Differences	29
3.5 Evolve XML Schema	32
3.6 Load Revised PurchaseOrders	33

Introduction

The Oracle XML DB basic demonstration high-lights some of the advanced features of Oracle XML DB in Oracle Database 11g Release 1 including

- Partitioning of data stored in a table based on XML Schema-optimized XMLType.
- Evolution of the XML Schema under-pinning an XML-Schema-optimized storage model.
- Use of SQL*Loader to load data into an XMLType table.

Architecture

Oracle Database 11g introduces two new capabilities that can be used with Oracle XML DB's schema-optimized storage model: These are

- The ability to use the Oracle Partitioning option with XMLType tables.
- The ability to make changes to an XML Schema that has been used for Schema-optimized XMLType storage without unloading and reloading data in tables and columns that are bound to the XML Schema.

XMLType Partitioning

In Oracle Database 10g Release 2 and earlier, an XMLType table or column can be partitioned, but a nested table cannot. This means that Oracle Partitioning can only be used with XMLType tables or columns that use CLOB or LOB based storage models. This is a significant limitation, since many applications will require a nested table storage model to ensure satisfactory query performance.

When a nested table model is used for XMLType storage, it is not uncommon for the bulk of the XML documents to be mapped into the nested tables. This means that the nested tables are often much larger than the XMLType tables they are associated with. If the storage model for the XMLType object uses nested tables, the fact that the nested table is not partitioned effectively disables all partition maintenance operations on the parent table. Consequently the full benefit of Oracle Partitioning can only be realized when all the tables used to store XML content are partitioned.

In Oracle Database 11g Oracle Partitioning option includes a new option for REF-Based partitioning. This allows a table to be partitioned based on a partition key inherited from another table. Oracle XML DB is able to use Ref-based partitioned to partition of nested tables based on the partitioning schema of the parent XMLType table. Since the nested tables now share a partitioning schema that is inherited from the XMLType table, when a partition maintenance operation is performed on the XMLType table, a similar operation is automatically performed on the associated partitions of the nested table.

XML Schema Evolution

Oracle Database 10g Release 2 included support for copy-based XML Schema evolution. Copy based XML Schema evolution allows modifications to made to an XML Schema that is being used for XML Schema-optimized storage without any loss of data. All content in XMLType tables and columns bound to the XML Schema is preserved intact by the evolution process. The advantage of copy-based evolution is that it is extremely flexible; it places no limitations on the kind of changes that can be made to the XML Schema. Copy based XML Schema evolution is performed using procedure copyEvolve in package DBMS_XMLSCHEMA.

Copy based evolution has two major problems. First, the time taken to complete the copy-evolve operation is directly proportional to the amount of data being managed. Second, significant amounts tablespace are required in order for the operation to complete successfully. The reason for this is that all of the data is unloaded and reloaded as part of the process. The unloaded data is stored using a textual representation of the XML which is transformed and re-parsed into the object-relational representation as it is reloaded.

Oracle Database 11g Release 1 addresses the issues with copy based XML Schema evolution by adding support for In-place XML Schema evolution. In-place evolution also allows modifications to made to an XML Schema that is being used for XML Schema-optimized storage without any loss of data. The big advantage of In-place evolution is that it does not need to unload and reload the data in the XMLType tables and columns that are bound to the XML Schema being modified.

The downside of in-place XML Schema is that it can only be used for changes that do not require data migration and which do not invalidate existing documents. Since no data is migrated as part of the In-place XML Schema evolution processing the time taken by the operation is minimal and constant, regardless of the amount of data that being managed. In-place XML Schema evolution is performed using procedure inplaceEvolve in package DBMS_XMLSCHEMA.

Installation

Server Pre-requisites

The following software is required to run the Oracle XML DB basic demonstration

Oracle Software

- Oracle Database 11g release 11.1.0.6.0 or later, with the XML DB, Oracle Text and Oracle JVM features installed.

Client Pre-requisites

The installation process uses an HTML application, VB Scripting and the HTTP protocol to upload the source code into the Oracle XML DB repository, SQL*PLUS scripts are used to re-configure the Oracle XML DB repository to support the demonstration.

The following software is required to install the Oracle XML DB Repository features demonstration.

Oracle Software

- Oracle Client (SQL*PLUS and Oracle Net Services) 11.1.0.6.0 (Production) or later. The application can be installed into a remote database, however both SQL*PLUS and Oracle Net Services must be installed on the client machine in order to perform a remote install. Currently remote installs are only supported on the Windows platform.
- Oracle XML DB X-Files application. The basic demonstration runs inside an AJAX-framework that is included as part of the Oracle XML DB X-Files application. Starting with Oracle Database 11g the X-Files demonstration must be downloaded and installed before installing the basic features demonstration.

Non Oracle Software

- Microsoft Internet Explorer 7.0 with the latest service packs. The X-Files application has not been tested with any release of other browsers including Firefox, Mozilla, Netscape, Safari or Opera.
- Microsoft Windows Scripting Technologies version 5. Windows Scripting is used by the installation process. You can verify the version of Windows Scripting installed on your machine by opening a command prompt and typing the command cscript.
- Microsoft Core XML Services (MSXML) versions 4.0sp2 and 6.0 are required in-order install the demonstration. It is also required by the AJAX based framework that is used to run the demonstration.

At the time of writing the latest version of this software can be downloaded from <http://www.microsoft.com/downloads/Search.aspx?displaylang=en>

- XMLSPY: XMLSPY is an IDE for XML from Altova Corporation. An evaluation copy of this product can be obtained from Altova's website at <http://www.altova.com>.
- Microsoft Windows XP Professional with Service Pack 2.

Database Configuration Changes

Installing the Repository Features demonstration application will make the following changes to the configuration of the target database.

- **Oracle XML DB HTTP and FTP Server:** Installing the basic demonstration will enable the database's native HTTP and FTP Servers. Ensure that you have read the XML DB documentation regarding the use of the XML DB HTTP Server before installing this application into a database that contains production data. This information can be found in the Oracle XML DB Developers guide.
- **Database Native Web Services:** The basic demonstration is executed using an AJAX-based framework. Ensure that you have read the XML DB documentation regarding the use of the Database Native Web Services before installing this application into a database that contains production data. This information can be found in the Oracle XML DB Developers guide.
- **Database Schema XDBMETADATA:** This database owns the objects used by the demonstration. This includes the global XML Schema <http://xmlns.oracle.com/demo/imageMetadata.xsd>, which defines the metadata managed by the demonstration, as well as the tables and packages required to create and manage the metadata. XDBMETADATA is a locked account with XDBADMIN role. It also has read access to some catalog tables owned by XDB and SYS. This account should remain locked.

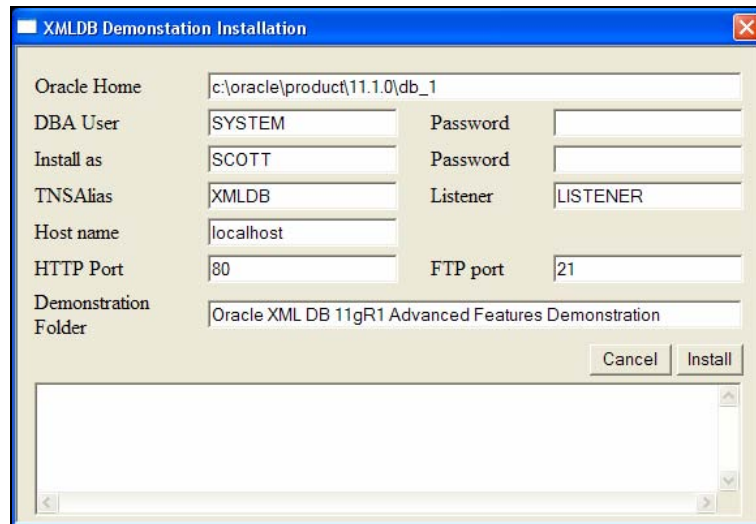
Installing the Application

To install the Oracle XML DB 11g Release 1 Advanced Features demonstration unzip the file XMLDB_11GR1_Advanced.zip into a folder of your choice. Ensure that there are no spaces in any of the parent folder names. After unzipping this file the target folder should contain a subfolder called advancedFeatures. This folder will contain subfolders SQL, setup and Install.

To start the installation, execute the file install.hta found in the install subfolder.

```
C:\advancedFeatures\install>install.hta
```

This will launch the installer dialog. The installation process is an HTML application.

The image shows a Windows-style dialog box titled "XMLDB Demonstration Installation". It contains several input fields for configuration. The fields and their values are: Oracle Home (c:\oracle\product\11.1.0\db_1), DBA User (SYSTEM), Password (empty), Install as (SCOTT), Password (empty), TNSAlias (XMLDB), Listener (LISTENER), Host name (localhost), HTTP Port (80), FTP port (21), and Demonstration Folder (Oracle XML DB 11gR1 Advanced Features Demonstration). There are "Cancel" and "Install" buttons at the bottom right. A large empty text area is at the bottom of the dialog.

The default values for the dialog are obtained from the file InstallationParameters.xml. The contents of this file are as follows:

```
<installationParameters>
  <shortCutFolderName>
    Oracle XML DB 11gR1 Advanced Features Demonstration
  </shortCutFolderName>
  <oracleHome>c:\oracle\product\11.1.0\db_1</oracleHome>
  <dba>SYSTEM</dba>
  <oracleUser>SCOTT</oracleUser>
  <oraclePassword/>
  <tnsAlias>XMLDB</tnsAlias>
  <listener>LISTENER</listener>
  <sqlPort/>
  <hostName>xmldb</hostName>
  <httpPort>80</httpPort>
  <ftpPort>21</ftpPort>
  <parameter name="%BASEFOLDER%" value="/publishedContent/imageMetadata" />
</installationParameters>
```

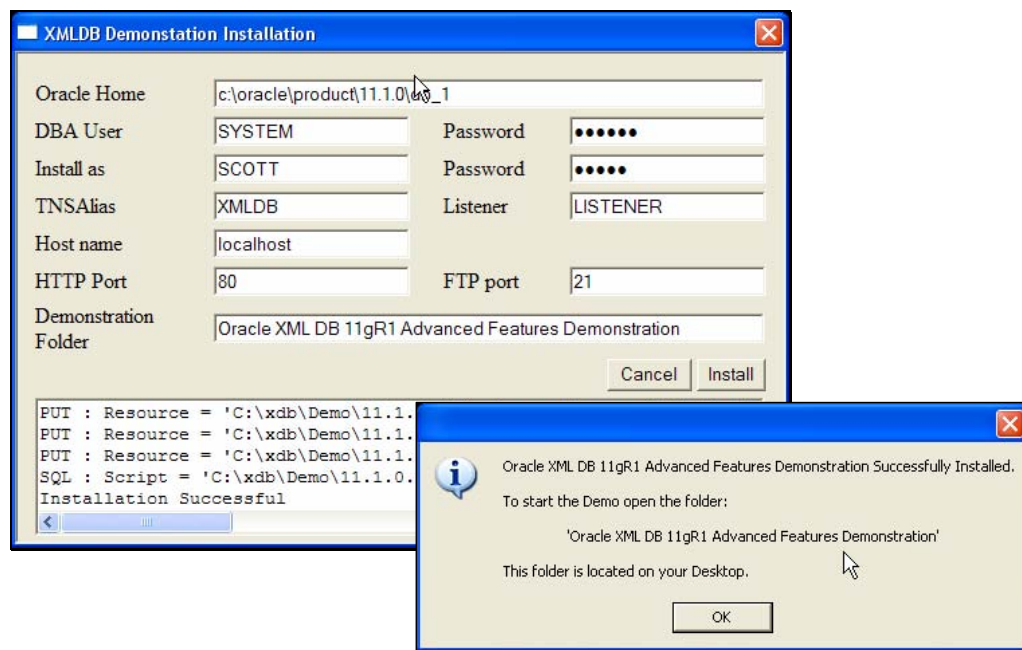
You can modify the default values by modifying the contents of InstallationParameters.xml before starting the installation. To install the demonstration, start the installer, modify any values that are not correct for your environment, enter the DBA and installation user's passwords and click install. Remember that in Oracle Database 11g passwords are case-sensitive. To cancel the installation click cancel.

The meaning of each parameter / field is given in the following table

< shortCutFolderName >	The name of the folder that will contain the set of Icons used to run the demo. This folder will be placed on the user's desktop.
<oracleHome>	The location of the Oracle Home on the local computer. An Oracle client installation is required to run the installation process.
<dba>	The name of a user with DBA capabilities which can be used to install the demonstration. Normally this will be SYSTEM, but any DBA is acceptable. The password for the DBA user can only be entered using the installation dialog.
<oracleUser>	The database user that will be used to run the demo. This user should already exist and be able to connect to the database. The installation process will grant this user the following privileges: session, unlimited tablespace, create table, create view, create any directory, drop any directory. These privileges are required to run the demonstration,
<oraclePassword>	The password for the demonstration user.
<tnsAlias>	The tnsAlias that can be used to connect to the target database instance
<listener>	The name of the listener associated with the database instance. A listener should not service more than one database when Oracle XML DB protocols are in use.
<hostname>	The name of the machine running the Listener.
httpPort	The port used by the Oracle XML DB HTTP Service. The port must not already be in use by any other service. The installation process will configure the database to use this HTTP port. If this port is a privileged port on a unix system the listener.ora must be configured appropriately.
ftpPort	The port used by the Oracle XML DB FTP Service. The port must not already be in use by any other service. The installation process will configure the database to use this HTTP port. If this port is a privileged port on a unix system the listener.ora must be configured appropriately.

Clicking Install will start the installation. The installation will verify that it can connect as the DBA, and as the demonstration user using SQL and HTTP. Once connectivity has been verified the demonstration will be installed. If the connectivity tests fail the installation will not proceed.

The progress of the installation will be shown in the status window at the bottom of the installer dialog. When the installation is complete the following message will be displayed.



Click OK to dismiss the dialog and then cancel to exit the installer.

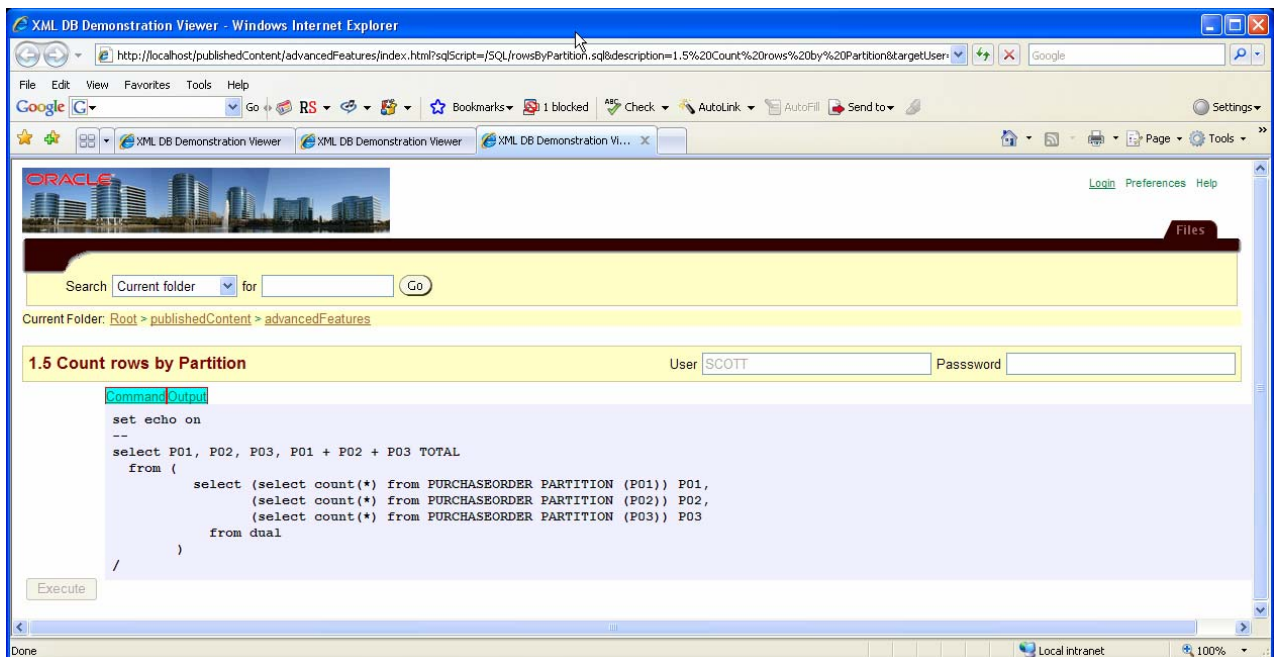
Using the demonstration framework

The SQL based portions of the demo are presented using the Oracle XML DB demonstration framework. This is an AJAX-based application uses the Database Native Web Services (DBNWS) feature of Oracle Database 11g to execute SQL scripts. The demonstration framework is installed as part of the X-Files application, which can be downloaded from the XML DB page on OTN. Please make sure you have the latest version of the X-Files application installed before running this demonstration.

The XML DB demonstration framework is launched by clicking the icons in the demonstration folder. If the browser currently owns an authenticated HTTP connection to the Oracle XML DB Database, the framework will automatically execute the SQL script. If the browser does not own an authenticated HTTP connection to the Oracle Database, or the current session does not belong to the correct user, the framework will prompt for a password before running the script. Entering the correct password will execute the script.

If the demonstration framework encounters a pause command, the CONTINUE button will be enabled. Click the CONTINUE button to continue executing the script. When the script is complete the CLOSE button will be enabled. Click the close button will close the framework session. If at least one framework session is left open, subsequent windows will be able to inherit the HTTP connection, avoiding the need to enter a password each time a new framework session is opened.

The following screen shot shows the demonstration framework ready to run a script. The framework is waiting for the password to be entered. The first SQL command in the script is displayed in the command area. The cursor is positioned in the Password field and the EXECUTE button is disabled.



Entering a valid password will automatically execute the SQL statement.

The framework consists of interleaved command and output areas.

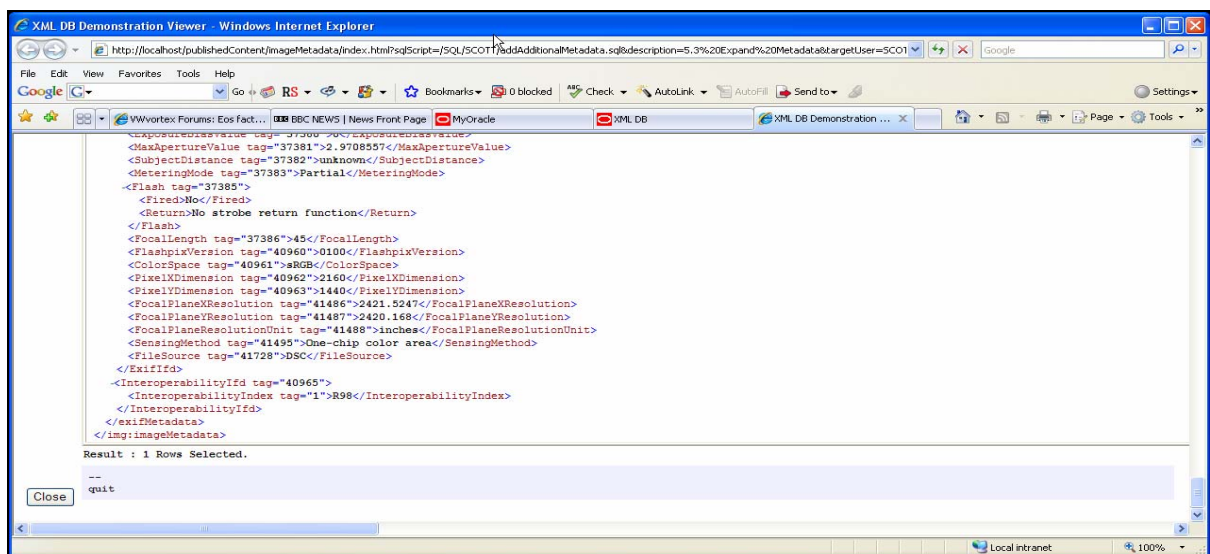
The command area shows the current SQL command. If the SQL command contains more lines than can be displayed in the default command area a vertical scroll bar will appear. The scroll bar can be used to scroll the contents of the command area. When the scroll bar is present clicking the Command tab will expand the command area to show the entire command. Once the command area is expanded clicking the Command tab again will revert to the default size.

The output area shows the results of the query. If the script includes a **set autotrace on explain** command the output area can also show the query plan for the current query. When the query plan is available two additional tabs, Result and Plan will be displayed. Click Result to see the query output, Click Plan to see the query plan.

If the command generates more output than can be displayed by the default output area a vertical scroll bar will appear. The scroll bar can be used to scroll the contents of the output area. When the scroll bar is present clicking the Output tab will expand the output area to show as much of the output as possible. Once the output area has been expanded clicking the Output tab again will revert to the default size. This behavior also occurs when the query plan is displayed in the output area.

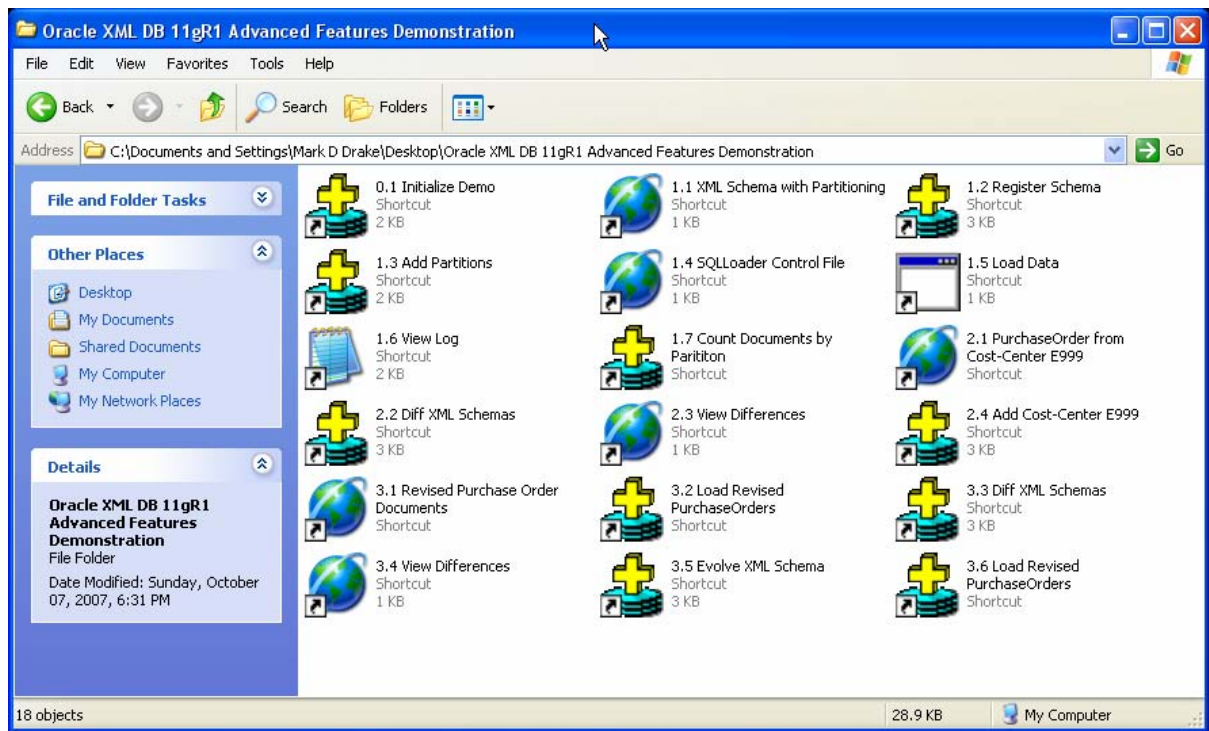
One major advantage of the demonstration framework is that it is completely XML aware. When a query executes XML, the XML will be displayed complete with all of its tag information intact. Elements with complex content will be marked with a - icon. Clicking the - icon will close up the children of the element, and replace the - icon with a + icon. Clicking the + icon will expand the subtree for the element.

The following screen shows the demonstration framework after completion of a script. The output area contains an XML document. Since the script is complete the CLOSE button is enabled.



Performing the demonstration

The installation creates the folder “Oracle 11gR1 XML DB Basic Features Demonstration”. This folder is located on the Desktop. The folder contains the following Icons.



The icons are numbered. To run the demonstration, click each icon in turn.

The first part of this demonstration focuses creating and maintaining Schema-optimized XMLType tables that use the Oracle Partitioning option. The second part of the demonstration shows how to use SQL*LOADER to load data in an XMLType table. The final part of the demonstration shows how to use In-place XML Schema evolution to make changes to the XML Schema without having to unload and reload data.

0.1 Initialize Demo

This step initializes the demonstration.

Click the icon to launch the XML DB demonstration framework and run the SQL script. The username field will be pre-filled with the name of the demonstration user and the password field will be empty. Type the password and hit enter or click outside of the password field. If the password is entered correctly the form will refresh and the script will execute. If an HTTP authentication dialog appears after entering the correct password then Database Native Web Services have not been correctly configured for the user.

The script undoes all of the actions performed while running the demonstration.

```
Command Output
set echo on
--
declare
  unregistered_schema exception;
  PRAGMA EXCEPTION_INIT( unregistered_schema , -31000 );
begin
  dbms_xmlschema.deleteSchema
  (
    'http://localhost:80/home/SCOTT/poSource/xsd/purchaseOrder.xsd',
    dbms_xmlschema.DELETE_CASCADE_FORCE);
exception
  when unregistered_schema then
    null;
end;
/

PL/SQL procedure successfully completed.
Command Output
--
--
-- Create home directory for target user...
--
call xdb_utilities.createHomeFolder()
/

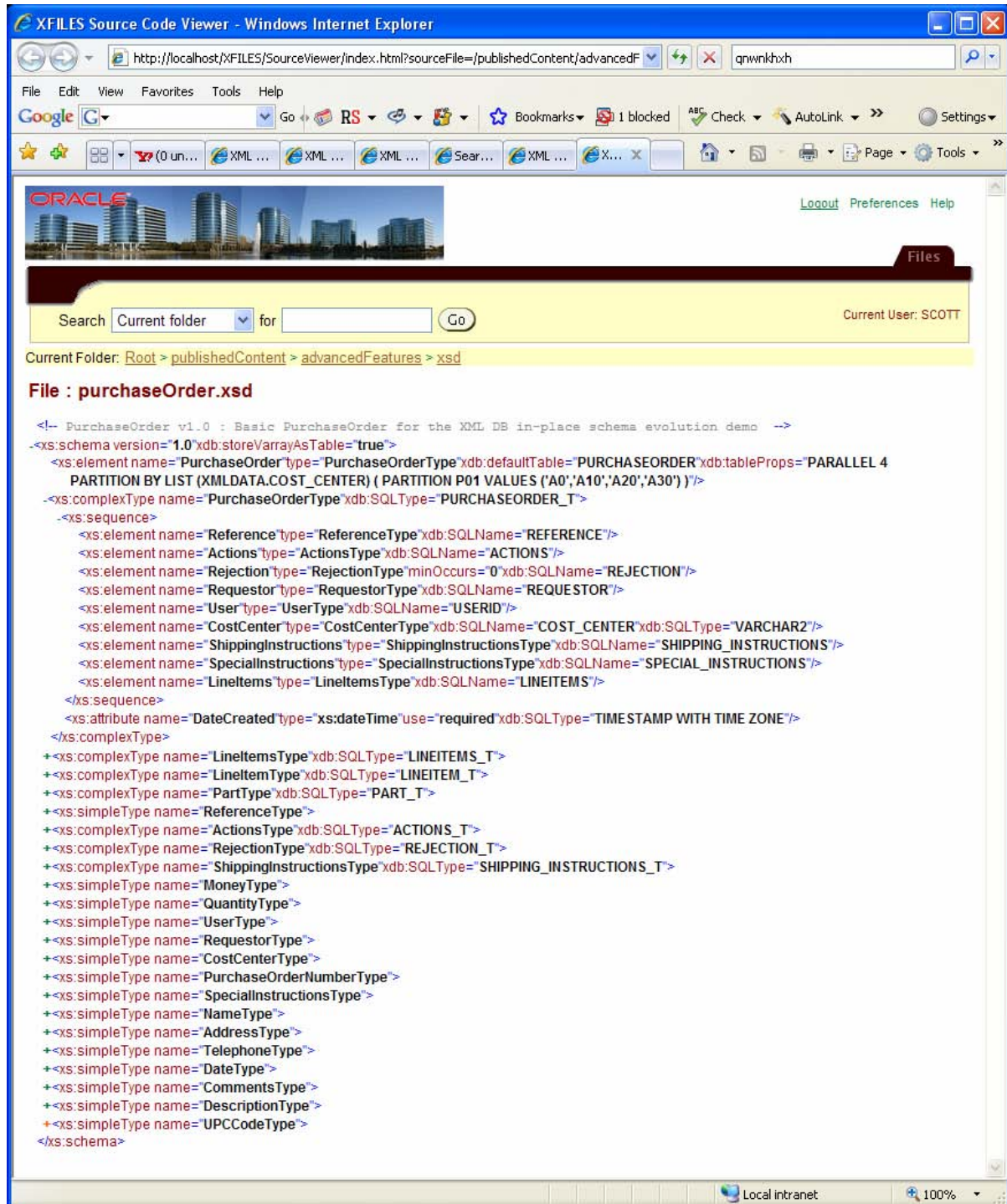
PL/SQL procedure successfully completed.
Command Output
declare
  res boolean;
begin
  if (dbms_xdb.existsResource('/home/SCOTT/diff')) then
    dbms_xdb.deleteResource('/home/SCOTT/diff',dbms_xdb.DELETE_RECURSIVE_FORCE);
  end if;
  res := dbms_xdb.createFolder('/home/SCOTT/diff');
  dbms_xdb.changeOwner('/home/SCOTT/diff','SCOTT');
end;
/

PL/SQL procedure successfully completed.
```

- Delete the PurchaseOrder XML Schema and all associated objects.
- Reset the status and content of all documents in the demonstration user's home folder
- Delete folder /home/SCOTT/diff and all of its contents. This folder contains the diff documents that are generated by the in-place XML Schema evolution process.

1.1 View XML Schema (XFILES)

This step uses the source code viewer component of the X-Files application to show the PurchaseOrder XML Schema. Click the icon to launch the XML DB Source Code Viewer and display the contents of the file.



- The default table for element PurchaseOrder is PURCHASEORDER
- Table PURCHASEORDER will be a partitioned XMLType table. The table will be list partitioned based on element CostCenter.
- The element / attribute chosen as the partition key can only occur once in each document
- In order to use XMLType partitioning the Oracle Partitioning option must be licensed. Once the partitioning option is licensed no additional licenses are required to use XMLType partitioning.
- The partitioning specification is supplied using the xdb:tableProps annotation. The syntax used to specify the partitioning scheme is standard Oracle Partitioning DDL.
- The partition key has to be specified using object-relational notation. .
- Only specify the first partition and degree of parallelism using attribute xdb:tableProps. Due to limitations on the amount of text permitted in attribute xdb:tableProps add the remaining partitions using ALTER TABLE ADD PARTITION statements after XML Schema registration has generated the default table.

1.2 Register XML Schema

This step registers the PurchaseOrder XML Schema with Oracle XML DB. Registering an XML Schema with the database enables optimized processing for the instance documents that belong to the XML Schema.

Click the icon to launch the XML DB demonstration framework and run the SQL script.

```
Command Output
--
Set echo on
--
declare
  logDDLOperations varchar2(4000) := 'alter session Set events=''31098 trace name context forever'';
  enablePartitioning varchar2(4000) := 'alter session Set events=''2321 trace name context forever, level 1'';
begin
  execute immediate enablePartitioning;
  execute immediate logDDLOperations;

  dbms_xmlschema.registerSchema
  (
    schemaURL => 'http://localhost:80/home/SCOTT/poSource/xsd/purchaseOrder.xsd',
    schemaDoc => xdbURIType('/publishedContent/advancedFeatures/xsd/purchaseOrder.xsd').getXML(),
    local => TRUE,
    genTypes => TRUE,
    genTables => TRUE,
    ENABLEHIERARCHY => DBMS_XMLSCHEMA.ENABLE_HIERARCHY_NONE
  );
end;
/

PL/SQL procedure successfully completed.

pause

Command Output
--
call xdb_analyze_xmlschema.renameCollectionTable ('PURCHASEORDER','XMLDATA"."LINEITEMS"."LINEITEM','LINEITEM')
/

PL/SQL procedure successfully completed.

Command Output
call xdb_analyze_xmlschema.renameCollectionTable ('PURCHASEORDER','XMLDATA"."ACTIONS"."ACTION','ACTION')
/

PL/SQL procedure successfully completed.

Command Output
select TABLE_NAME, PARTITION_NAME
  from USER_TAB_PARTITIONS
/
```

TABLE_NAME	PARTITION_NAME
ACTION_TABLE	P01
LINEITEM_TABLE	P01
PURCHASEORDER	P01

Result : 3 Rows Selected.

- Schema registration will create all the types and tables need to provide an XML Schema optimized storage model for PurchaseOrder documents
- Nested tables are generated for the collection of Action elements and the collection of LineItem elements. Procedure registerSchema uses system generated names for these tables. Procedure renameCollectionTable in package XDB_ANALYZE_SCHEMA can be used to specify meaningful names for these tables.

- Procedure `renameCollectionTable` takes three arguments: the name of the parent table; the name of the SQL attribute corresponding to the repeating element; and the new name for the nested table. The new table name is automatically be suffixed with `_TABLE`. Renaming nested tables makes it much easier to create indexes and interpret query plans.
- The default table will be partitioned based on the content of attribute `xdb:tableProps`.
- The nested tables will be partitioned using a REF based partitioning. This ensures that the nested table rows are stored in same partition as the parent table row. This allows partition maintenance operations on the parent table to be cascaded to the nested tables.
- Following XML Schema registration there are three partitioned tables, each with one partition. The partitioned tables are the XMLType table `PURCHASEORDER` and the nested tables `LINEITEM_TABLE` and `ACTION_TABLE`

1.3 Add Partitions

This step adds additional partition to table PURCHASEORDER.

Click the icon to launch the XML DB demonstration framework and run the SQL script.

Command Output

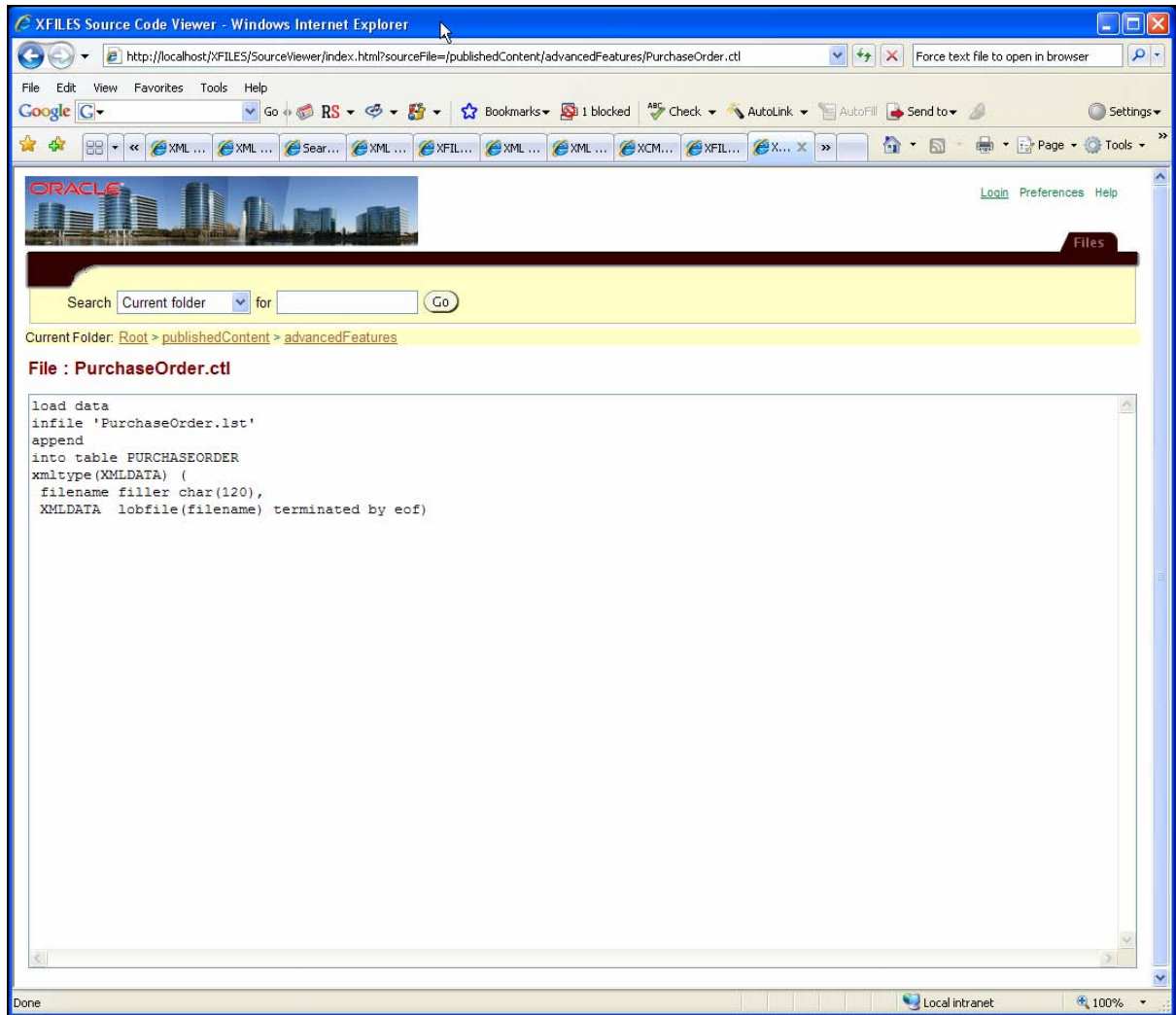
```
set echo on
--
ALTER TABLE PURCHASEORDER
  ADD PARTITION P02
    VALUES ('A40','A50','A60','A70')
/

Table altered.
```

Command Output

1.4 SQL-Loader Control File

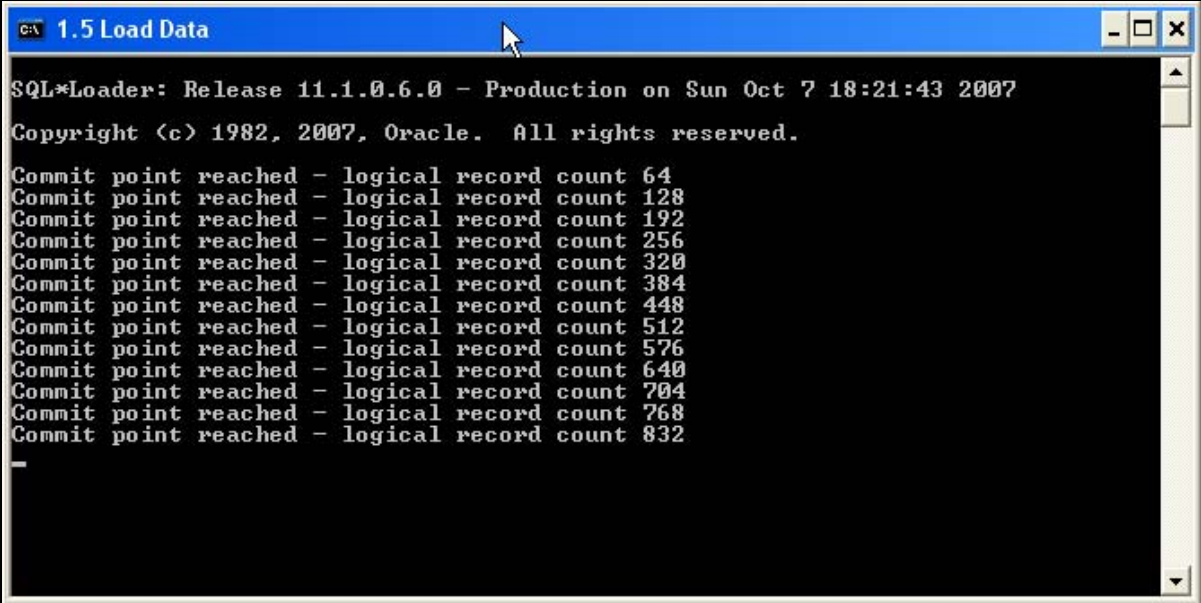
This step uses the source code viewer component of the X-Files application to show the SQL*LOADER control file used to load 10,000 instance documents into table PURCHASEORDER. Click the icon to launch the XML DB Source Code Viewer and display the contents of the file.



- The list of documents to be loaded will be read from the file PurchaseOrder.lst.

1.5 Load Data

This step uses SQL*Loader to load 10,000 documents into table PURCHASEORDER.

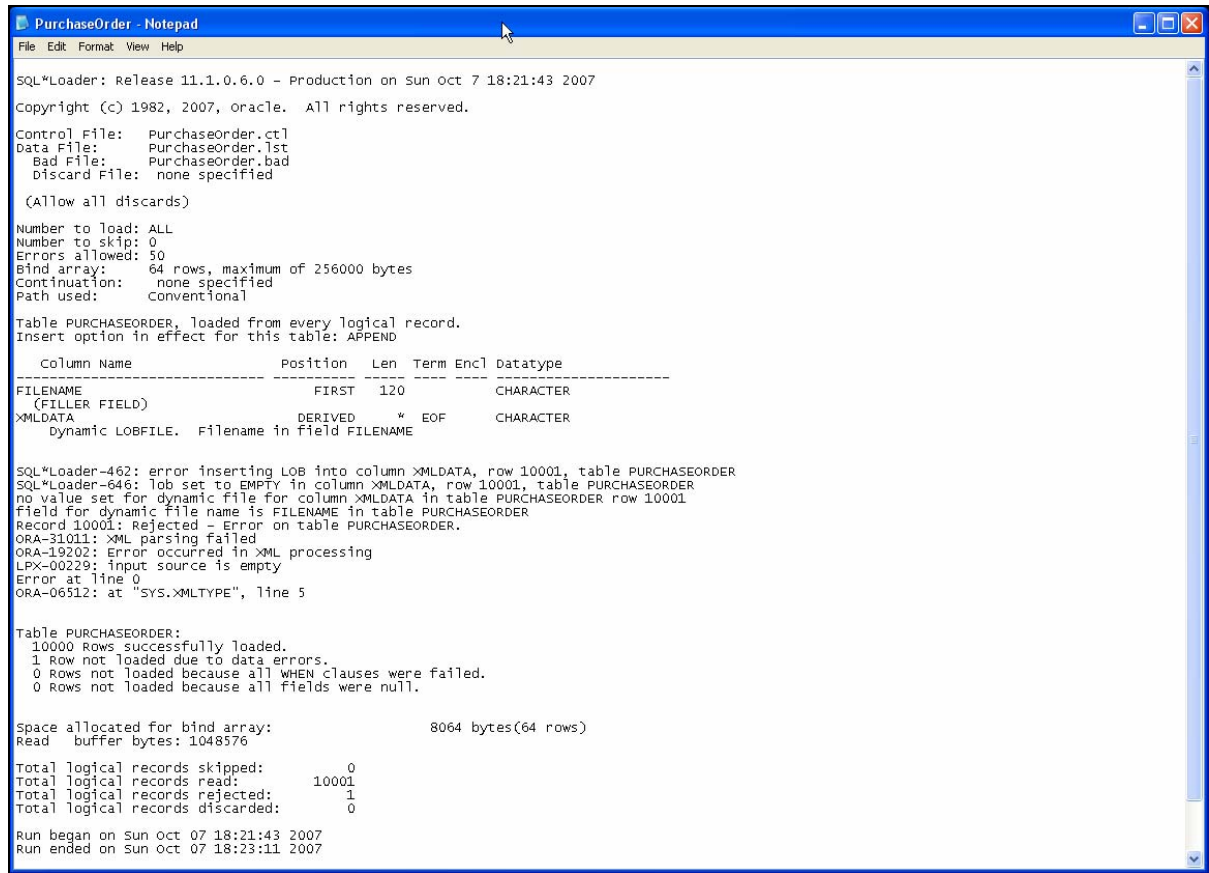


```
C:\ 1.5 Load Data
SQL*Loader: Release 11.1.0.6.0 - Production on Sun Oct 7 18:21:43 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Commit point reached - logical record count 64
Commit point reached - logical record count 128
Commit point reached - logical record count 192
Commit point reached - logical record count 256
Commit point reached - logical record count 320
Commit point reached - logical record count 384
Commit point reached - logical record count 448
Commit point reached - logical record count 512
Commit point reached - logical record count 576
Commit point reached - logical record count 640
Commit point reached - logical record count 704
Commit point reached - logical record count 768
Commit point reached - logical record count 832
-
```

- SQL*Loader conventional path load is used to load the documents from the local file system in the Oracle database. Direct path load is not currently supported with partitioned XMLType tables.

1.6 View Log

This step uses Windows Notepad to review the SQL*LOADER log file.



```
SQL*Loader: Release 11.1.0.6.0 - Production on Sun Oct 7 18:21:43 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.

Control File:   Purchaseorder.ctl
Data File:      Purchaseorder.lst
Bad File:       Purchaseorder.bad
Discard File:   none specified

(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 256000 bytes
Continuation:   none specified
Path used:      Conventional

Table PURCHASEORDER, loaded from every logical record.
Insert option in effect for this table: APPEND

  Column Name          Position  Len  Term  Enc1  Datatype
-----
FILENAME              FIRST    120             CHARACTER
(FILLER FIELD)
XMLDATA                DERIVED    *  EOF             CHARACTER
dynamic LOBFILE.  filename in field FILENAME

SQL*Loader-462: error inserting LOB into column XMLDATA, row 10001, table PURCHASEORDER
SQL*Loader-646: lob set to EMPTY in column XMLDATA, row 10001, table PURCHASEORDER
no value set for dynamic file for column XMLDATA in table PURCHASEORDER row 10001
field for dynamic file name is FILENAME in table PURCHASEORDER
Record 10001: Rejected - Error on table PURCHASEORDER.
ORA-31011: XML parsing failed
ORA-19202: error occurred in XML processing
LPX-00229: input source is empty
Error at line 0
ORA-06512: at "SYS.XMLTYPE", line 5

Table PURCHASEORDER:
10000 Rows successfully loaded.
1 Row not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

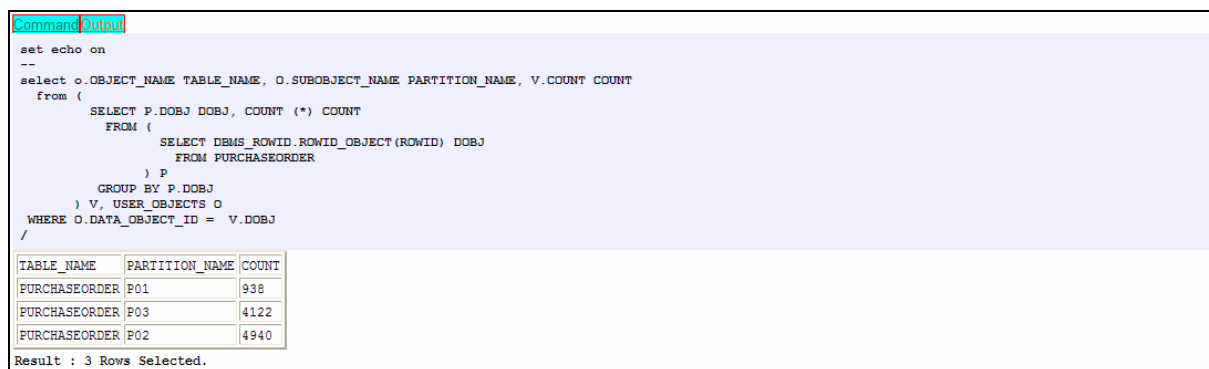
space allocated for bind array:                8064 bytes(64 rows)
Read  buffer bytes: 1048576

Total logical records skipped:      0
Total logical records read:         10001
Total logical records rejected:     1
Total logical records discarded:    0

Run began on Sun Oct 07 18:21:43 2007
Run ended on Sun Oct 07 18:23:11 2007
```

1.7 Count Documents by Partition

This step shows distribution of the 10,000 documents by partition. Click the icon to launch the XML DB demonstration framework and run the SQL script.



```
Command Output

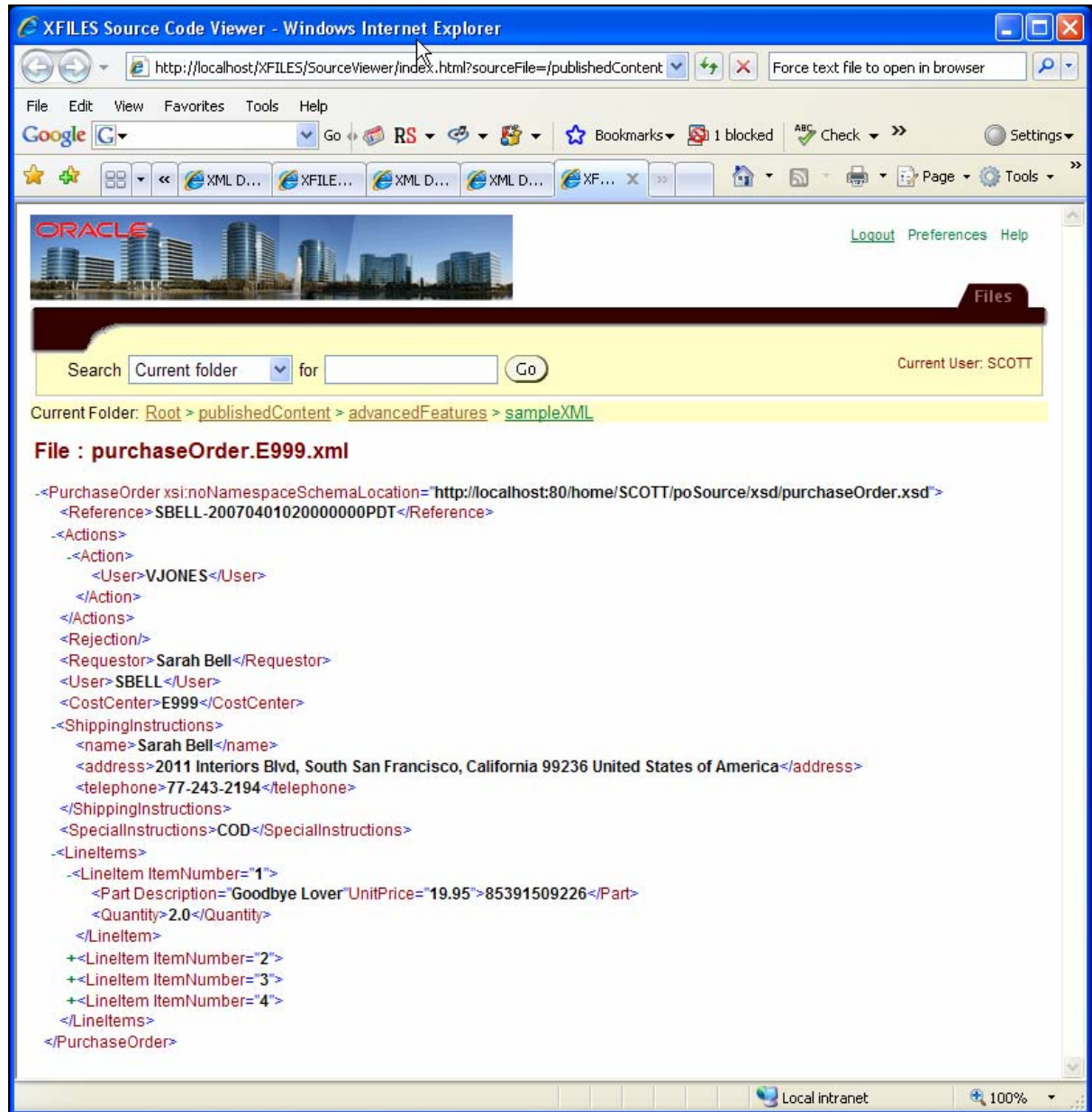
set echo on
--
select o.OBJECT_NAME TABLE_NAME, o.SUBOBJECT_NAME PARTITION_NAME, v.COUNT COUNT
from (
  SELECT P.DOBJ DOBJ, COUNT (*) COUNT
  FROM (
    SELECT DBMS_ROWID.ROWID_OBJECT(ROWID) DOBJ
    FROM PURCHASEORDER
  ) P
  GROUP BY P.DOBJ
) V, USER_OBJECTS O
WHERE O.DATA_OBJECT_ID = V.DOBJ
/

TABLE_NAME | PARTITION_NAME | COUNT
-----
PURCHASEORDER | P01 | 938
PURCHASEORDER | P03 | 4122
PURCHASEORDER | P02 | 4940

Result : 3 Rows Selected.
```

2.1 PurchaseOrder from Cost-Center E999

This step uses the source code viewer component of the X-Files application view a PurchaseOrder document where element CostCenter contains a value that does map to any of the existing partitions of table PURCHASEORDER. Click the icon to launch the XML DB Source Code Viewer and display the contents of the file.



Introducing a new value for element CostCenter introduces two challenges.

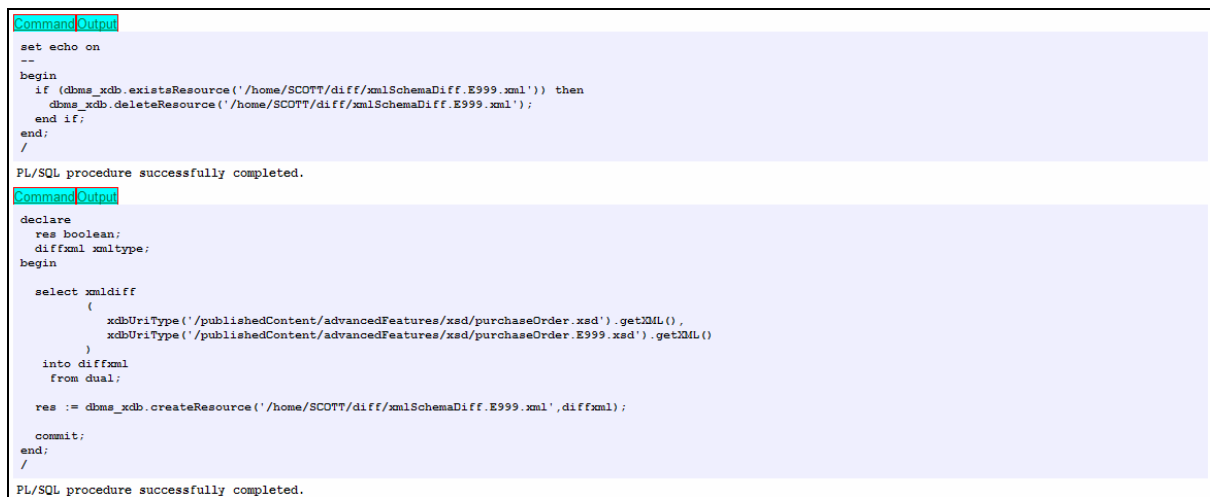
- CostCenter is defined as an instance of CostCenterType which includes an enumeration that define the list of valid values for CostCenter. The current set of valid values for the enumeration does not include E999.
- The current set of partitions defined for table PURCHASEORDER do not support the case where value of element CostCenter is E999. It will be necessary to introduce a new partition to manage this value once the XML Schema has been updated.

2.2 Diff XML Schemas

Adding a new value to the CostCenterType enumeration does not invalidate the existing instance documents, so the change can be implemented using in-place XML Schema evolution.

In-place schema evolution requires an XML document which identifies the differences between the old and new versions of the XML Schema. This document is generated by processing the current and new versions of the XML Schema using the new Oracle XML DB 11g's XMLDiff feature.

This step shows how to use XMLDiff to generate the document that is required to perform an in-place evolution of the PurchaseOrder XML Schema. Click the icon to launch the XML DB demonstration framework and run the SQL script.



```
Command Output
set echo on
--
begin
  if (dbms_xdb.existsResource('/home/SCOTT/diff/xmlSchemaDiff.E999.xml')) then
    dbms_xdb.deleteResource('/home/SCOTT/diff/xmlSchemaDiff.E999.xml');
  end if;
end;
/

PL/SQL procedure successfully completed.

Command Output
declare
  res boolean;
  diffxml xmltype;
begin
  select xmldiff
    (
      xdbUriType('/publishedContent/advancedFeatures/xsd/purchaseOrder.xsd').getXML(),
      xdbUriType('/publishedContent/advancedFeatures/xsd/purchaseOrder.E999.xsd').getXML()
    )
  into diffxml
  from dual;

  res := dbms_xdb.createResource('/home/SCOTT/diff/xmlSchemaDiff.E999.xml', diffxml);

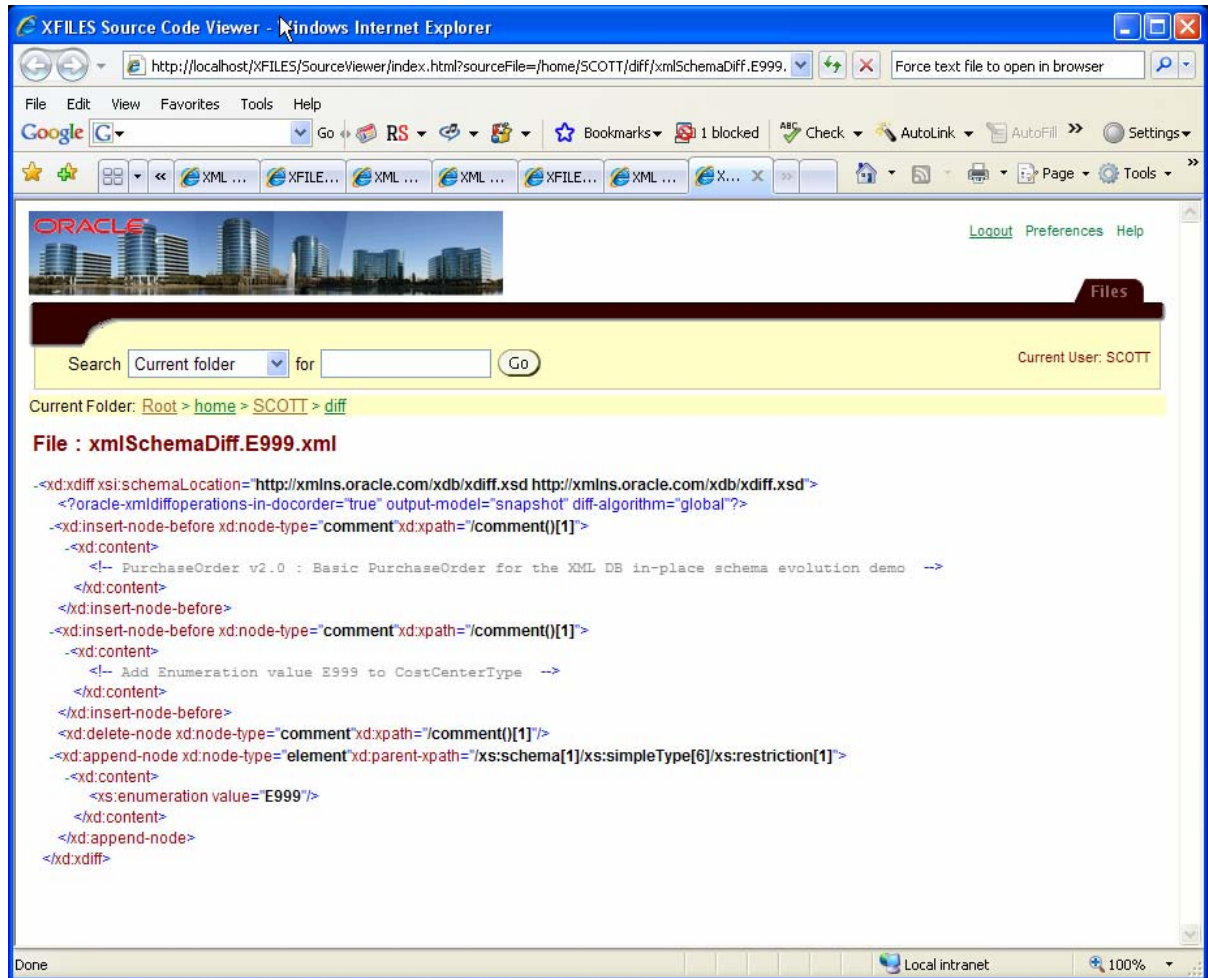
  commit;
end;
/

PL/SQL procedure successfully completed.
```

- The document /publishedContent/advancedFeatures/xsd/PurchaseOrder.xsd is the original version XML Schema
- The document /publishedContent/advancedFeatures/xsd/PurchaseOrder.E999.xsd is the new version of the XML Schema.
- The result of running XMLDiff on the two XML Schemas an XML document. In the result of XMLDiff is stored as /home/SCOTT/diff/xmlSchemaDiff.E999.xml

2.3 View Differences

This step uses the source code viewer component of the X-Files application view the document generated by the XMLDiff process. Click the icon to launch the XML DB Source Code Viewer and display the contents of the file.



The XMLDiff output shows that differences between the two XML Schemas

- Two comments have been inserted before first comment.
- The first comment in the original document has been deleted.
- A new enumeration element has been added to the element node identified by the path /xs:schema[1]/xs:simpleType[6]/xs:restriction[1]

2.4 Add Cost-Center E999

This step shows how to use the XMLDiff document to evolve the XML schema so that an instance document for CostCenter E999 can be stored in table PURCHASEORDER. Click the icon to launch the XML DB demonstration framework and run the SQL script.

```
Command Output
set echo on
set timing on
set long 1000000 pages 0
--
select count(*)
  from PURCHASEORDER
/

COUNT(*)
10000

Result : 1 Rows Selected.
Elapsed Time : 703 milliseconds.

Command Output
insert into PURCHASEORDER
values
(
  xdburitype('/publishedContent/advancedFeatures/sampleXML/purchaseOrder.E999.xml').getXML()
)
/

ORA-19202: Error occurred in XML processing
ORA-14400: inserted partition key does not map to any partition
ORA-06512: at "XDBPM.XDB_DEMO_HELPER_i1100", line 31
ORA-06512: at line 1
Elapsed Time : 47 milliseconds.
```

- The first statement shows that table PURCHASEORDER currently contains 10,000 documents.
- The second statement shows that an instance document where element CostCenter contains the value E999 cannot be inserted into table PURCHASEORDER. The insert fails since the value E999 does not map to any of the available partitions.

```
Command Output
--
begin
  dbms_xmlschema.inPlaceEvolve
  (
    'http://localhost:80/home/SCOTT/poSource/xsd/purchaseOrder.xsd',
    xdburitype('/home/SCOTT/diff/xmlSchemaDiff.E999.xml').getXML()
  );
end;
/

PL/SQL procedure successfully completed.
Elapsed Time : 437 milliseconds.

Command Output
select count(*)
  from PURCHASEORDER
/

COUNT(*)
10000

Result : 1 Rows Selected.
Elapsed Time : 438 milliseconds.
```

- The third statement evolves the XML Schema using the contents of the XMLDiff document. Procedure inPlaceEvolve in package DBMS_XMLSCHEMA is used to modify the registered version of the XML Schema. Even though table PURCHASEORDER contains 10,000 rows the operation takes less in less than 0.5 seconds, since no data migration is required.

- The fourth statement shows that after the XML Schema evolution is complete table PURCHASEORDER still contains 10,000 rows.

```

Command Output
--
ALTER TABLE PURCHASEORDER
ADD PARTITION P04
VALUES ('E999')
/

Table altered.
Elapsed Time : 94 milliseconds.
Command Output
select TABLE_NAME, PARTITION_NAME
from USER_TAB_PARTITIONS
/

```

TABLE_NAME	PARTITION_NAME
ACTION_TABLE	P01
LINEITEM_TABLE	P01
PURCHASEORDER	P01
PURCHASEORDER	P02
ACTION_TABLE	P02
LINEITEM_TABLE	P02
PURCHASEORDER	P03
ACTION_TABLE	P03
LINEITEM_TABLE	P03
PURCHASEORDER	P04
ACTION_TABLE	P04
LINEITEM_TABLE	P04

```

Result : 12 Rows Selected.
Elapsed Time : 15 milliseconds.

```

- The fifth statement adds a new partition to table PURCHASEORDER that will manage instance documents where element CostCenter contains the value E999.
- The sixth statement shows that additional partitions have been added the new partition to table PURCHASEORDER automatically added new partitions to the nested tables LINEITEM_TABLE and ACTION_TABLE.

```

Command Output
--
insert into PURCHASEORDER
values
(
  xdburitype('/publishedContent/advancedFeatures/sampleXML/purchaseOrder.E999.xml').getXML()
)
/

1 row Inserted.
Elapsed Time : 156 milliseconds.
Command Output
select count(*)
from PURCHASEORDER
/

```

COUNT(*)
10001

```

Result : 1 Rows Selected.
Elapsed Time : 453 milliseconds.

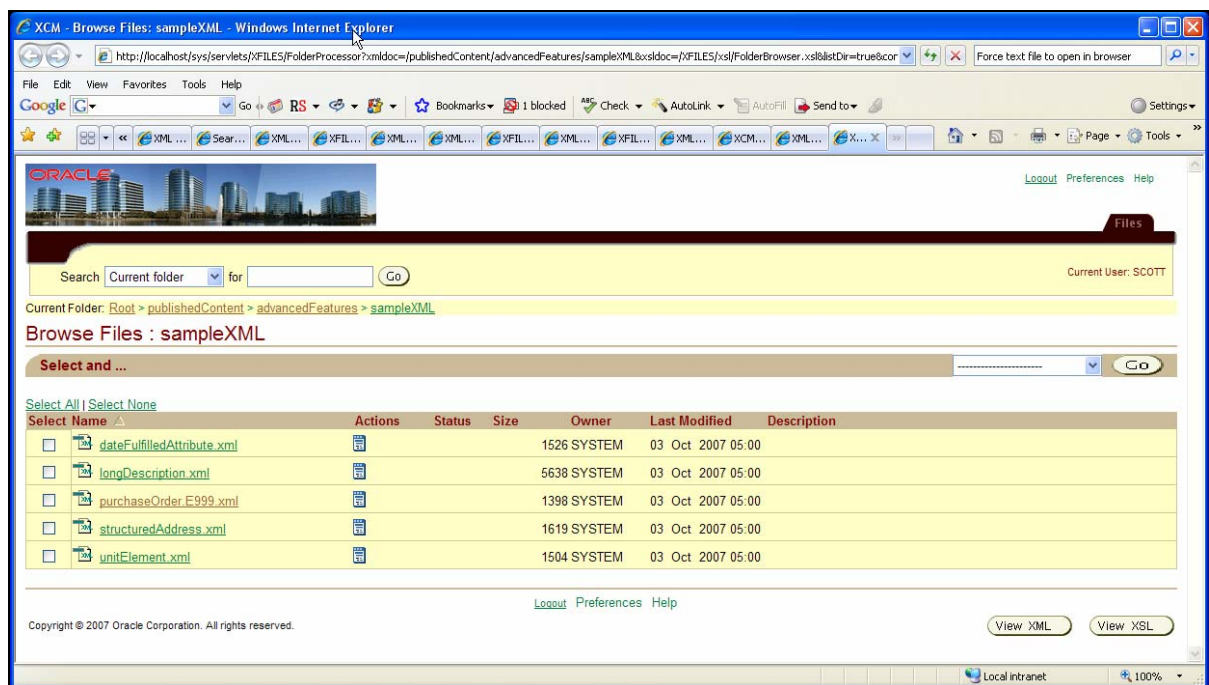
```

- The seventh statement shows that the instance document containing CostCenter E999 can be inserted into table PURCHASEORDER without an errors being raised.
- The final statement shows that after the insert completes table PURCHASEORDER contains 10,001 rows.

3.1 Revised Purchase Order Documents

Adding a new value for an enumeration is a very simple change. It does not require any changes to the types and tables generated when the original XML Schema was registered with the database. This step shows that in-place evolution can be used to make more complex changes to the XML Schema which do require modifications to the underlying types and tables, as long as the changes are additive in nature.

This step uses X-Files application to open a folder containing set of four instance documents that do not conform to the current version of the PurchaseOrder XML Schema. Click the icon to launch the X-Files application and view the contents of the folder. Click any of the documents to view its content in the source code viewer.



- Element PurchaseOrder in document dateFulfilledAttribute.xml contains attribute DateFulfilled and element Notes element. The current version of the XML Schema does not permit these nodes to appear as children on element PurchaseOrder.
- Elements Special Instructions and Description in document longDescription.xml contain far more text that is permitted by the current version of the XML Schema.
- Element ShippingInstructions in structuredAddress.xml contains element newAddress with child elements StreetLine1, City, State, ZipCode and Country. The current version of the XML Schema only allows elements name, telephone and address to appear as children on

element ShippingInstructions. Element address is defined as type string, so it is not permitted to contain child elements.

- Element LineItem in unitElement.xml contains element Unit. The current version of the XML schema only allows elements Part and Quantity to appear as children of element LineItem.

3.2 Load Revised PurchaseOrders

This step attempts to load the four new PurchaseOrder documents into table PURCHASEORDER. Click the icon to launch the XML DB demonstration framework and run the SQL script.

```
Command Output
--
set echo on
--
select count(*)
  from PURCHASEORDER
/

COUNT(*)
10001
Result : 1 Rows Selected.

Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/dateFulfilledAttribute.xml').getXML())
/

ORA-19202: Error occurred in XML processing
ORA-30937: No schema definition for 'DateFulfilled' (namespace '##local') in parent '/PurchaseOrder'
ORA-06512: at "XDBPM.XDB_DEMO_HELPER_11100", line 31
ORA-06512: at line 1

Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/unitElement.xml').getXML())
/

ORA-19202: Error occurred in XML processing
ORA-30937: No schema definition for 'Unit' (namespace '##local') in parent '/PurchaseOrder/LineItems/LineItem[1]'
ORA-06512: at "XDBPM.XDB_DEMO_HELPER_11100", line 31
ORA-06512: at line 1

Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/longDescription.xml').getXML())
/

ORA-19202: Error occurred in XML processing
ORA-30951: Element or attribute at Xpath /PurchaseOrder/SpecialInstructions exceeds maximum length
ORA-06512: at "XDBPM.XDB_DEMO_HELPER_11100", line 31
ORA-06512: at line 1

Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/structuredAddress.xml').getXML())
/

ORA-19202: Error occurred in XML processing
ORA-30937: No schema definition for 'newAddress' (namespace '##local') in parent '/PurchaseOrder/ShippingInstructions'
ORA-06512: at "XDBPM.XDB_DEMO_HELPER_11100", line 31
ORA-06512: at line 1

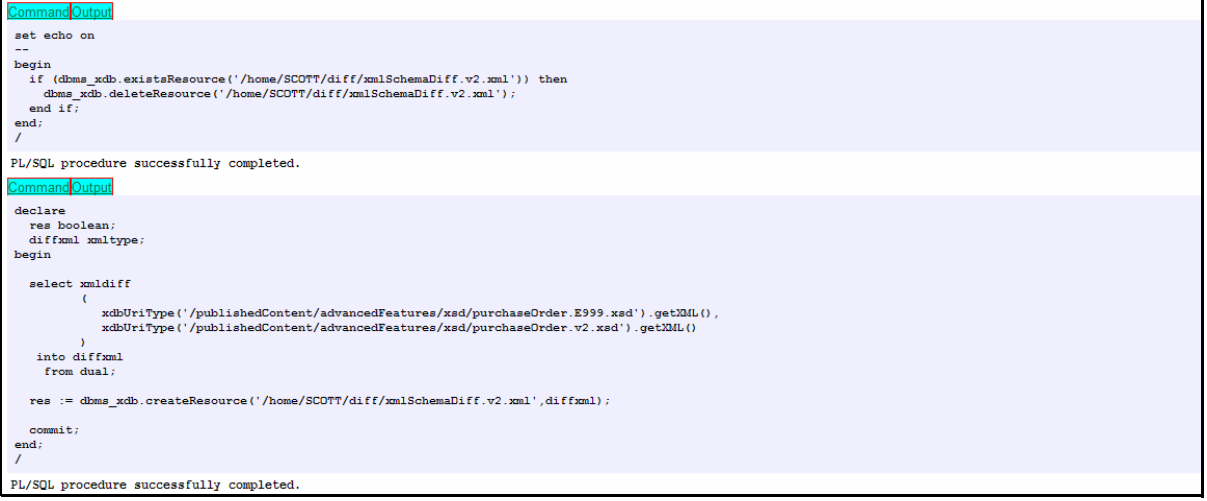
Command Output
select count(*)
  from PURCHASEORDER
/

COUNT(*)
10001
Result : 1 Rows Selected.
```

- All of the documents fail to load since they do not conform to the current version of the PurchaseOrder XML Schema.

3.3 Diff XML Schemas

The document `/publishedContent/advancedFeatures/xsd/PurchaseOrder.v2.xsd` contains an version of the XML Schema that includes the changes required to support all of the new PurchaseOrder documents. This step shows how to use XMLDiff to generate the document required to perform an in-place evolution of the PurchaseOrder XML Schema. Click the icon to launch the XML DB demonstration framework and run the SQL script.



```
Command Output
set echo on
--
begin
  if (dbms_xdb.existsResource('/home/SCOTT/diff/xmlSchemaDiff.v2.xml')) then
    dbms_xdb.deleteResource('/home/SCOTT/diff/xmlSchemaDiff.v2.xml');
  end if;
end;
/

PL/SQL procedure successfully completed.

Command Output
declare
  res boolean;
  diffxml xmltype;
begin
  select xmldiff
    (
      xdbUriType('/publishedContent/advancedFeatures/xsd/purchaseOrder.E999.xsd').getXML(),
      xdbUriType('/publishedContent/advancedFeatures/xsd/purchaseOrder.v2.xsd').getXML()
    )
  into diffxml
  from dual;

  res := dbms_xdb.createResource('/home/SCOTT/diff/xmlSchemaDiff.v2.xml',diffxml);

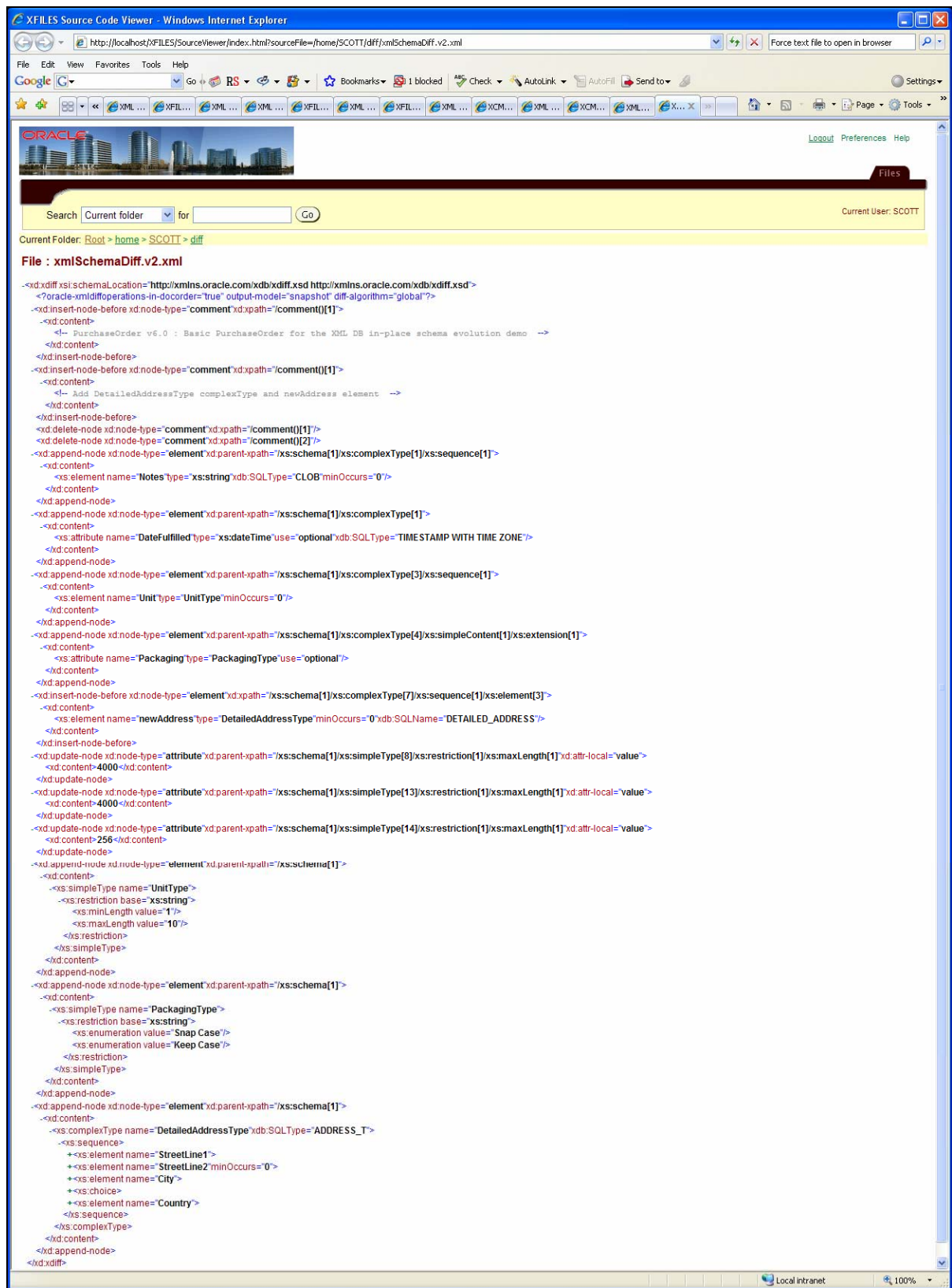
  commit;
end;
/

PL/SQL procedure successfully completed.
```

- The document `/publishedContent/advancedFeatures/xsd/PurchaseOrder.E999.xsd` is the current version XML Schema
- The document `/publishedContent/advancedFeatures/xsd/PurchaseOrder.v2.xsd` is the new version of the XML Schema.
- The result of running XMLDiff on the two XML Schemas an XML document. In the result of XMLDiff is stored as `/home/SCOTT/diff/xmlSchemaDiff.v2.xml`

3.4 View Differences

This step uses the source code viewer component of the X-Files application view the document generated by the XMLDiff process. Click the icon to launch the XML DB Source Code Viewer and display the contents of the file.

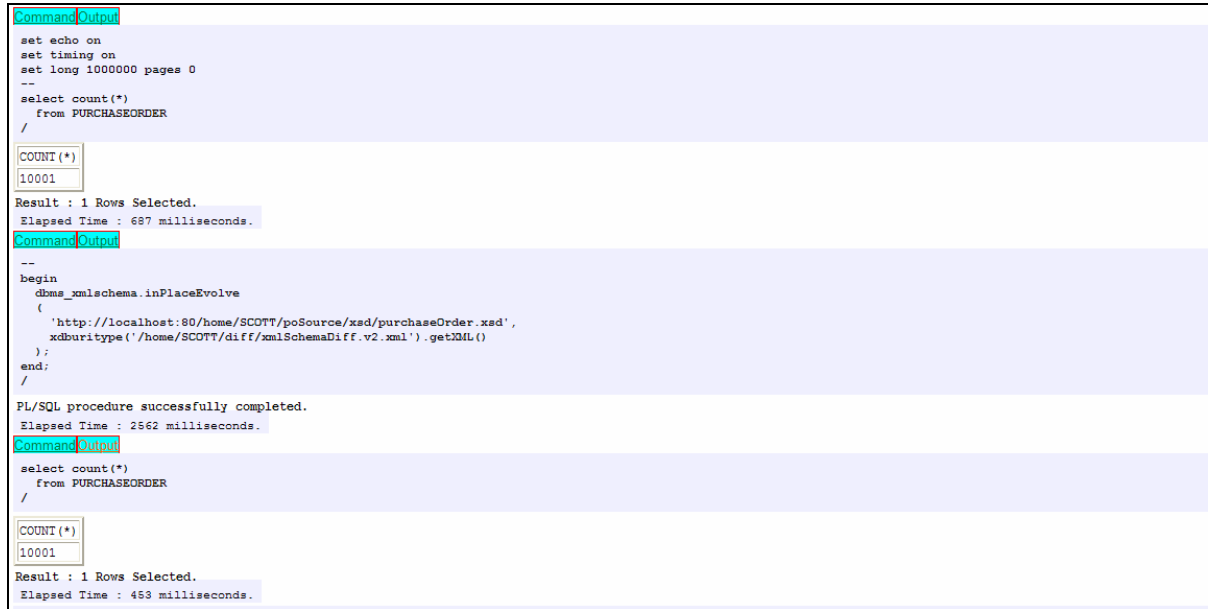


The differences between the current and new version of the XML schema are as follows

- The comments have been replaced
- An element called notes has been added to the first complexType (PurchaseOrderType)
- An attribute called DateFulfilled has been added to the first complexType (PurchaseOrderType)
- An element called Unit has been added to the third complexType (LineItemType)
- An attribute called Packaging has been added to the fourth complexType (PartType).
- A new element has called newAddress has been added to the seventh complexType (ShippingInstructionsType). Element newAddress is based on complexType DetailedAddressType.
- The maxLength attribute of the eighth simpleType (SpecialInstructionsType) has been changed to 4000 characters.
- The maxLength attribute of the twelfth simpleType (CommentsType) has been increased to 4000 characters.
- The maxLength attribute of the fourteenth simpleType (DescriptionType) has been increased to 256 characters.
- ComplexType DetailedAddressType has been added to the XML Schema.
- This set of modifications shows the type of changes that can be managed with in-place schema evolution.

3.5 Evolve XML Schema

This step shows how to use the XMLDiff document to evolve the XML schema so the new PurchaseOrder documents can be stored in table PURCHASEORDER. Click the icon to launch the XML DB demonstration framework and run the SQL script.



```
Command Output
set echo on
set timing on
set long 1000000 pages 0
--
select count(*)
from PURCHASEORDER
/

COUNT(*)
10001
Result : 1 Rows Selected.
Elapsed Time : 687 milliseconds.

Command Output
--
begin
  dbms_xmlschema.inPlaceEvolve
  (
    'http://localhost:80/home/SCOTT/poSource/xsd/purchaseOrder.xsd',
    xdburitype('/home/SCOTT/diff/xmlSchemaDiff.v2.xml').getXML()
  );
end;
/

PL/SQL procedure successfully completed.
Elapsed Time : 2562 milliseconds.

Command Output
select count(*)
from PURCHASEORDER
/

COUNT(*)
10001
Result : 1 Rows Selected.
Elapsed Time : 453 milliseconds.
```

- With 10,001 rows loaded into the PurchaseOrder table the in-place evolution takes 2.5 seconds to run. The time taken is related purely to the complexity of the changes being made. Since no data migration takes place the time taken is not proportional to the amount of data being managed.

3.6 Load Revised PurchaseOrders

This step show that once the in-place evolution is completed the four new PurchaseOrder documents can be successfully loaded into table PURCHASEORDER. Click the icon to launch the XML DB demonstration framework and run the SQL script

```
Command Output
set echo on
--
select count(*)
  from PURCHASEORDER
/

COUNT(*)
10001
Result : 1 Rows Selected.
Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/dateFulfilledAttribute.xml').getXML())
/

1 row Inserted.
Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/unitElement.xml').getXML())
/

1 row Inserted.
Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/longDescription.xml').getXML())
/

1 row Inserted.
Command Output
insert into PURCHASEORDER values ( xdburitype('/publishedContent/advancedFeatures/sampleXML/structuredAddress.xml').getXML())
/

1 row Inserted.
Command Output
select count(*)
  from PURCHASEORDER
/

COUNT(*)
10005
Result : 1 Rows Selected.
```

- Once the XML Schema evolution process is completed the documents can be successfully loaded into the table.



Oracle Database 11g Oracle XML DB

October 2007

Author: Mark D Drake

Contributing Authors: Oracle XML DB Development Team

Oracle Corporation

World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

oracle.com

Copyright © 2007, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.