Storing your Oracle Report in the database

Preface

This document describes two different ways to store your reports generated by Oracle Reports in your database.

Index

Introduction

One can come up with several reasons to store your Oracle Reports output. Especially in cases where it needed to prove to authorities that the document you delivered was not manipulated by the end user. Consider a pharmaceutical customer, they will need to apply to FDA regulation 21CFR11. Or perhaps even better, to prove some end user manipulated the document. Perhaps you need to deliver a copy of an already printed report and are not able to run the report again since it concerns intermediate data which can not be reproduced anymore. One thousand reasons to consider storing your output in the database.

The two methods described below demonstrate how you could set up the Oracle Reports server so it will store the report in the database.

Store your report using PL/SQL

Introduction

To store your report using PL/SQL you will need to write code that actually uploads you report to the database. The ability to use the code is fully depending upon the availability of the functions/procedures in your database.

Setup

1. You will need to create a directory on the server and make it known to your instance.

```
SQL> create or replace directory REP_FILE_DIR
     as 'C:\oracle\product\10.1.0\Db_1\demo\schema\REP_FILES\';
```

   Where the bold printed words are your variables and can be replaced for any name\directory path.

2. You will need to have a table which will actually store your file

```
SQL> create table BLOBTAB
(
  IND       VARCHAR2(10), -- primary key
  BLOBCOL   BLOB,
  FILE_NAME VARCHAR2(255),
 Timestamp date default sysdate,
 Userstamp varchar2(40) default user
)
/
```

3. Create a package in the database that actually does the file upload:

```
Create or replace package uploadBLobFile as
procedure Upload (pDirectory IN VARCHAR2, pIndex IN NUMBER, pFileName
IN VARCHAR2);
End uploadBLobFile;

Create or replace package body uploadBLobFile as

procedure Upload (pDirectory IN VARCHAR2, pIndex IN NUMBER, pFileName
IN VARCHAR2)
is

lBLob BLOB;
lBFileName BFILE;

Begin
  INSERT INTO BLOBTAB(IND,BLOBCOL,FILENAME)
  VALUES (pIndex,EMPTY_BLOB(),pFileName)
  RETURNING BLOBCOL
  INTO lBlob;
  -- insert the Blob
  lBfileName:=BFILENAME(pDirectory, pFileName);
  DBMS_LOB.fileopen(lBfileName);
  DBMS_LOB.loadfromfile(lblob,
                        lBfileName,
                        DBMS_LOB.getlength(lBfilename)
                        );
  DBMS_LOB.fileclose (lBfilename);
  COMMIT;
  Exception
    When others then
      If DBMS_LOB.ISOPEN(lBfileName) then
        DBMS_LOB.fileclose (lBfilename);
End upload;

End uploadBLobFile;
```

4. Create the proper synonyms and grant the correct privileges

```
SQL> Create public synonym uploadBlobFile for uploadBlobFile;
SQL> Grant execute on uploadBlobFile to public;
```

5. Create an after reports trigger that executes the following code:

```
Begin
 UploadBlobFile('REP_FILE_DIR', 1,'rep.pdf');
End;
```

Where the first argument equals to the name of the directory, the second argument represents the primary key you want to give the row and the third argument stands for the name of the fire which is created on the OS by the reports engine.

6. Make sure that when you run the report you have selected as destype=FILE, desname=C:\oracle\product\10.1.0\Db_1\demo\schema\REP_FILES\rep.pdf and desformat=PDF.

Note: make sure the reports engine writes the file to the server OS. If the file does not exist in the specified directory an error will occur. The report file will not be deleted from the OS after the upload. You will need to write your own java class to do so.

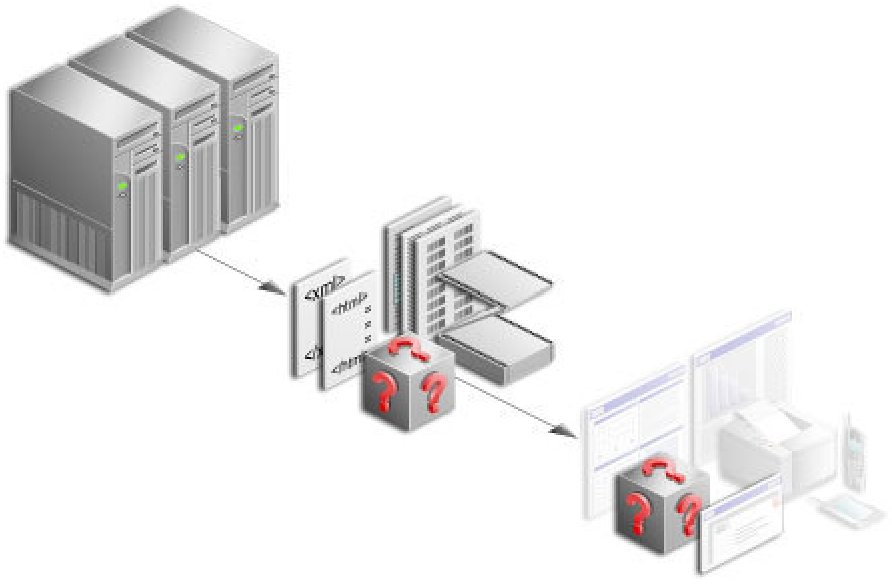
Store your report using a pluggable destination

Introduction

Pluggable destinations can be used to distribute any content that an engine (not only the reports engine) has created in the Reports Server's cache. Oracle Reports provides several out-of-the-box destinations:

- Web
- printer
- e-mail
- file
- OracleAS Portal

However it is possible to create your own custom destination by using the Oracle Reports Java APIs.



Oracle Reports enables you to "plug" in any destination you wish, using the provided API.

Setup

1. Copy the BLOBDestination.jar file to *ORACLE_HOME*\reports\jlib folder.
2. Deploy the new destination class.
   You need to make the BLOB destination jar file (BLOBDestination.jar) available to

Oracle Reports via the classpath. This procedure depends on whether you are using the in-process reports server or a standalone reports server. If you are using the in-process reports server, you need to modify the classpath for the OC4J instance. The reason being that the in-process server inherits the class path from its OC4J instance.

If you are using the in-process reports server:

a. Go to the Enterprise Manager. Under 'System Components', click OC4J_BI_Forms. On the OC4J_BI_Forms page click Applications. Click on the application named Reports.
b. Under Properties, click on General.
c. Under Library Paths add the following entry
   ```
   <library
   path="../../../../reports/jlib/BLOBDestination.jar" />
   ```

   Note that instead of a relative path, it is also possible to write the absolute path of the JAR.

d. Restart OC4J_BI_Forms from the Enterprise Manager.

If you are using a standalone reports server:

a. Add the following entry to the REPORTS_CLASSPATH environment variable: `ORACLE_HOME`/reports/jlib/BLOBDestination.jar

   Note: On Windows, use the registry to specify the value. On Unix/Linux, set the environment variable in reports.sh.

b. Register the new destination with the Oracle Reports Services.
   To register the new destination with the Reports Server, you need to add an entry in the Reports Server configuration file (`server_name.conf`). The configuration file is in *ORACLE_HOME*`/reports/conf` directory. Specifically, add the following entry:
   ```
   <destination
   destype="BLOBDestination"
   class="oracle.reports.plugin.destination.blob.BLOBDestinati
   on">
   </destination>
   ```
c. Restart the Reports Server for your changes to take effect.

To send your report output to this destination, specify BLOBDestination as the value for the DESTYPE parameter. The syntax for the command line is:

```
...&DESTYPE=blobdestination&DESFORMAT=pdf&DESNAME=http://user:pwd@DBhos
tname:port/sid/table/blobcolumn/pkcolumn/pkvalue/filename_column
```

DESTYPE=blobdestination&DESFORMAT=PDF&DESNAME=http://user:password@DBhostname:port/sid/table/blobcolumn/pkColumn/pkValue/filename_column

The description of the parameters is as follows:

Parameter	**Description**
User	Username for logging into the database where the report output must be stored
Pwd	Password for logging into the database where the report output must be stored
DBHostname:port	Hostname and port number for the database host machine
SID	Oracle SID for the database
Table	Table name where you would like to store the report output
Blobcolumn	Name of the column which will hold the report output as a BLOB datatype
Pkcolumn	Primary key column of the table
Pkvalue	The desired value of primary key for the row that will hold this report's output
filename_column	Name of the column that stores the file name of the report output. This file name is the same as generated by Oracle Reports

Example:
```
http://repserver_host/reports/rwservlet?report=test.rdf&userid=scott/ti
ger@demo&DESTYPE=blobdestination&DESFORMAT=PDF&DESNAME=http://scott:tig
er@database_servername:1521/iasdb/blobtab/blobcol/ind/5/file_name
```

There is no restriction on the number of columns that may be available in the table, however, this sample writes into only the above 3 columns.

Conclusion

Both examples show that it is possible to store your Oracle report in the database.
The PL/SQL example is useful for you if you do not have the availability of Oracle Application Server 9i or above.
If you do have the ability to use the Oracle pluggable destination solution you are best of using it. It is the best solution since it does not leave files on the OS and, if running in three tier (application server and database server not on the same machine), no mapping is required from the reports server to the database OS.