



Copying data from SQL Server database to an Oracle Schema

White Paper

© Copyright Decipher Information Systems, 2005. All rights reserved.

The information in this publication is furnished for information use only, does not constitute a commitment from Decipher Information Systems of any features or functions discussed and is subject to change without notice. Decipher Information Systems assumes no responsibility or liability for any errors or inaccuracies that may appear in this publication.

Last revised: June 2006

Table of Contents

| | |
|--|---|
| Copying data from SQL Server database to an Oracle Schema..... | 4 |
| Abstract | 4 |
| Different Options Available..... | 4 |
| Script Framework and the steps involved | 5 |
| Summary | 8 |

Copying data from SQL Server database to an Oracle Schema

Abstract

Usage of multiple RDBMS is becoming a norm and a necessity in many IT shops. Oracle, DB2 UDB and Microsoft SQL Server are the three top-most RDBMS that are deployed in the shops today. In future whitepapers we will cover the hurdles that are faced in such shops that use heterogeneous RDBMS and the different tools and techniques that can be used to make work easier. One of the many asks that DBAs in these shops face, is data movement between an Oracle Schema to a SQL Server database or vice-versa. That needs to be done efficiently and in the minimum amount of time. In this whitepaper, we will look at a framework of scripts that will make the movement of data from a SQL Server Database to an Oracle schema feasible and we will look at the different options that can be deployed to make that happen and why we opted for choosing the framework scripts approach.

Different Options Available

There are a couple of options that are available for copying over the data from a SQL Server database to an Oracle Schema:

1. **Linked Servers:** Linked servers can be set up on the SQL Server instance and then data can be pushed out to an Oracle schema. This works fine when you are dealing with smaller sub-sets of data but has several limitations in terms of:
 - a. the data-types supported by the provider/ODBC driver that has been used for creating the linked server,
 - b. Performance is not good for decent size tables.Advantage of this approach is that this is all SQL based and you can specify different criteria, put filters etc. in order to classify your data sub-set.
2. **Oracle Heterogeneous Services:** This is Oracle's answer to Microsoft's linked servers. By setting it up, one can use it to pull data from a SQL Server database and load it up in an Oracle Schema. This has the same limitations and advantages as those of option #1 mentioned above.
3. **DTS (Data Transformation Services) or SSIS (SQL Server Integration Services):** Microsoft SQL Server 2000 ships with DTS and SQL 2005 ships with SSIS. SSIS is way robust than DTS for this kind of work and can be used to load the data up in an Oracle Schema. These tools can be used to provide a real ETL process for the data loading process. Usage of SSIS will be covered in a future white-paper.
4. **Scripting Framework:** Scripts can be written to bcp the data out from a SQL Server database and then use SQL *LDR (SQL Loader) to load the data into an Oracle Schema.
5. **Third party tools:** A couple of third party tools also provide these options.

In this white-paper, we will introduce a scripting framework that can be used to copy data from any SQL Server database to any Oracle schema as long as the schema structures for the tables and their columns are the same i.e. the underlying assumption of this scripting framework is that the DBA in the IT shop is maintaining the application that runs on multiple RDBMS and as a result of that the Oracle schema (structure wise) is the same as the one used for SQL Server or DB2 UDB.

Script Framework and the steps involved

Assumptions:

- 1) As mentioned before, the assumption is that the schema is the same between SQL Server and Oracle.
- 2) The script framework uses a python script to parse through the output to generate multiple control files for the Oracle schema to facilitate the easy loading of the data using sql *ldr. You can download python (for windows) software from:

<http://www.python.org/download>

If you do not intend to use the parser DISBulk.py, then there is no need to download it. In that case, you will need to either manually split the control files or write a parser of your own.

Steps:

- 1) Take the Script: "Create_DIS_Conversion_Objects.sql" and run it in the destination Oracle schema to create the truncate_table, disable_fks and other objects.
 - a. Run truncate_table in the Oracle Schema: `exec truncate_table`
 - b. Run disable_fks and disable_triggers in the Oracle Schema:
 - i. `exec disable_fks`
 - ii. `exec disable_trigger`
- 2) Take the files supplied and copy them to the SQL Server database Server or Alternatively, you can also run this from any client machine (the only difference would be that once the data files have been generated, then you would need to copy over the data files into that client machine folder).

For the sake of Simplicity, you can adhere to this folder naming convention (though if you want, you can change it and then change the scripts accordingly):

```
C:\DIS_CONV
    \Oracle
        \Bad
        \Data
```

\Log

Copy all the scripts under the \Oracle directory.

- 3) In Query Analyzer, copy the "GEN_BCP_DATA_FILES.SQL" script and change the name of the database to be the name of the source database that resides on that instance.
 - a. Make the changes for the configuration information i.e. the login-id, path where the data files should be generated etc.
 - b. Execute it.

This will dump all the tables from that database into their separate data files in the folder that you specify (this folder will be on the DB Server) --> alternatively a UNC naming path can be followed as well if the client has an issue (though I don't see why anyone would have an issue with this). Copy all these data files into the \Oracle\Data folder mentioned above.

- 4) Now, take the SQL script: "GEN_CONTROL_FILES.SQL" and change the name of the database to be the name of the source database that resides on that instance. Run that in Query Analyzer using the text mode (Ctrl-T).

This script will generate all the control files that you need for loading up of the data that you generated in the previous step into Oracle. Copy the entire set and put them into a file called: "DISBULK.def" (a sample file has been provided --replace the contents from your execution run).

NOTE: Please use WORDPAD for copying and pasting since notepad will ruin the formatting.

Then copy this file in the Oracle directory above.

- 5) After the file DISBULK.def has been copied, double-click on the DISBULK.py python script which will parse through that file and dump out the control files for each of the tables in the same directory i.e. under the Oracle folder.
- 6) Now, take the script: "GEN_BATCH_FILES.SQL" and change the name of the database to be the name of the source database that resides on that instance. Run that in Query Analyzer using the text mode (Ctrl-T).
 - a. Make the changes for the configuration information i.e. path where the data files should be generated, the Oracle connection information etc.
 - b. Execute it.

This will generate the batch file contents. Copy these contents into the "DIS_ORA_DATA_LOAD.BAT" file as is.

NOTE: Turn off Word-Wrap (under the format option in notepad).

- 7) Double-click the batch file to start the data load from SQL Server to Oracle.
 - a. Check for the bad data records and the logs for any errors.
 - b. Fix any issues that are noticed and selectively re-import the data.
- 8) Run enable_fks and enable_triggers in the Oracle Schema and check for any violations that result from it :
 - a. exec enable_fks
 - b. exec enable_trigger

Run DIS_update_sequences in the Oracle Schema:

Exec DIS_update_sequences

- 9) Run the script "Gather_Stats.sql" to gather the stats on the tables and their indexes.
- 10) Once that has been done, you can get the number of rows per table using this SQL:

```
select table_name, num_rows
from user_tables
where num_rows > 0
order by table_name
/
```

Checking for Errors: You should check the contents of the BAD and the LOG folders to fix any data related errors that you might have encountered during the data loading process.

Checking for the rowcounts: You can also establish a link between SQL Server and Oracle and run this SQL to ensure that you have the right object counts for your tables. You can use the script: DIS_Check_Rowcounts.sql in order to establish the linked server between SQL Server instance and Oracle and then query the schemas and get the information. Be sure to change the names of the instance, the user-name etc. appropriately. Also, that particular script is dependent upon whether proper statistics have been collected on the SQL Server database and the Oracle Schema. There are other ways also to count the number of rows in the tables in a schema (look into the "How-To" documents on our site : <http://www.decipherinfosys.com/faq-home.htm>).

All the scripts that are used for this framework are present as a separate download – see Het_RDBMS_1.zip file.

Summary

We introduced a scripting framework in this white-paper that can be used to copy data from a SQL Server database to an Oracle Schema given that the schema structures are the same between the two.

Further information on the bcp command utility can be obtained from SQL Server Books Online and information on SQL *LDR (SQL Loader) utility from Oracle can be obtained from the Oracle online documentation at OTN. We hope that you would find this useful for your work.