Source – *A.Kishore*
http:/www.appsdba.info

DATAPUMP

Examples

1> Simple export and import using DATAPUMP
2> Performing a table mode export
3> Estimating how much disk space will be consumed in a schema mode export
4> Performing a schema mode export
5> Performing a full database export using four parallel processes
6> Attaching to and stopping an existing job
7> Attaching to and restarting a stopped job
8> Performing a data-only table mode import
9> Performing a schema mode import
10>Using DBMS_DATAPUMP
11> Import data via a network link in Oracle 10g
12> Reorganize tablespaces using Oracle 10g Data Pump
13> Moving data between versions
14> Monitor DataPump jobs

ORACLE
Certified Professional

ORACLE
E-Business Suite 11*i*
Certified Professional

Source – *A.Kishore*
http:/www.appsdba.info

Example 1

Simple export and import using DATAPUMP

1. Create a copy of the employees table under the HR schema

```
$ sqlplus hr/hr
SQL> CREATE TABLE EMP2 AS SELECT * FROM EMPLOYEES;
```

2. Create a DIRECTORY

```
$ sqlplus  / as sysdba
SQL> CREATE DIRECTORY testdir AS '/home/oracle10g/datapump';
SQL> GRANT READ ON DIRECTORY testdir  TO hr;
SQL> GRANT WRITE ON DIRECTORY testdir TO hr;
```

3. Create directory at the OS Level

```
$  cd
$ mkdir datadump
```

4. Export the EMP2 table from the HR user

```
$ expdp hr/hr directory=TESTDIR tables=hr.emp2
```

5. Logon and drop the EMP2 table

```
$ sqlplus hr/hr
SQL> Drop table emp2 purge;
SQL> EXIT
```

6. Verify that a .DMP file and a .LOG file exists in the DATDUMP directory.
7. Examine the .LOG file with any text editor
8. Import the .DMP file back into the HR user.

```
$ impdp hr/hr directory=testdir tables=emp2
```

9. Verify that the emp2 table is re-created and re-loaded with data.

```
$ sqlplus hr/hr
Sql> select * from emp2;
```

Example 2

Performing a table mode export

```
DROP DIRECTORY datadir1;
DROP DIRECTORY datadir2;

CREATE DIRECTORY datadir1 AS 'c:\data_pump1';
CREATE DIRECTORY datadir2 AS 'c:\data_pump2';

GRANT READ,WRITE ON DIRECTORY datadir1 TO sh;
GRANT READ,WRITE ON DIRECTORY datadir2 TO sh;


expdp system/<password> TABLES=sh.costs,sh.sales DUMPFILE=datadir2:table.dmp
NOLOGFILE=y
```

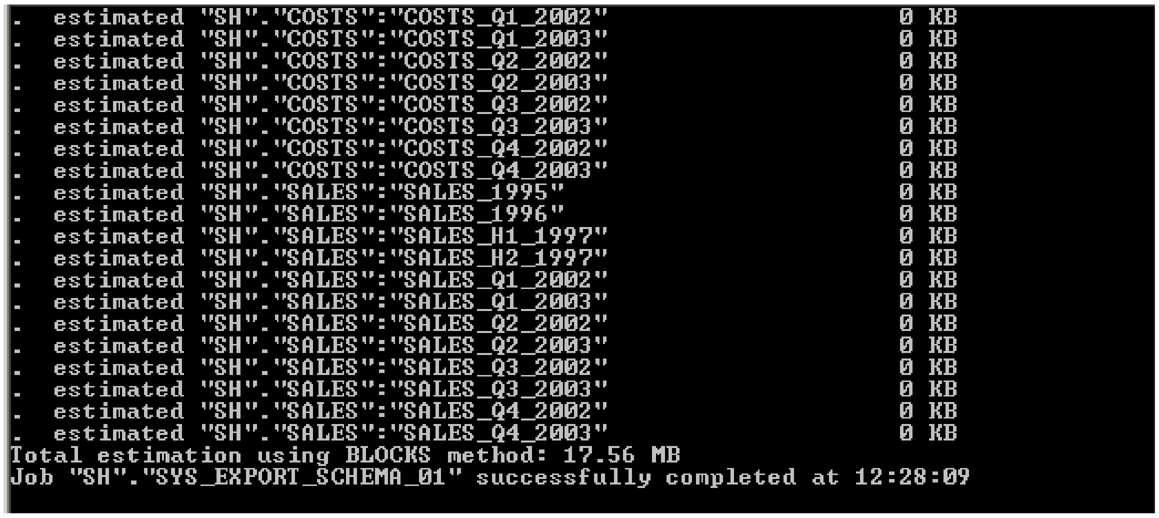
Example 3

Estimating How Much Disk Space Will Be Consumed in a Schema Mode Export

The ESTIMATE_ONLY parameter estimates the space that would be consumed in a schema export, but stops without actually performing the export operation. The estimate is printed in the log file and displayed on the client's standard output device. The estimate is for table row data only; it does not include metadata.

The INCLUDE parameter allows you to filter the metadata that is exported by specifying objects and object types for the current export mode. The specified objects and all their dependent objects are exported. Grants on these objects are also exported.

```
expdp sh/sh INCLUDE=table:\"IN \( \'SALES\',\'PRODUCTS\',\'COSTS\'\) \"
DIRECTORY=datadir2 ESTIMATE_ONLY=y
```



```
.  estimated "SH"."COSTS":"COSTS_Q1_2002"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q1_2003"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q2_2002"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q2_2003"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q3_2002"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q3_2003"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q4_2002"                    0 KB
.  estimated "SH"."COSTS":"COSTS_Q4_2003"                    0 KB
.  estimated "SH"."SALES":"SALES_1995"                       0 KB
.  estimated "SH"."SALES":"SALES_1996"                       0 KB
.  estimated "SH"."SALES":"SALES_H1_1997"                    0 KB
.  estimated "SH"."SALES":"SALES_H2_1997"                    0 KB
.  estimated "SH"."SALES":"SALES_Q1_2002"                    0 KB
.  estimated "SH"."SALES":"SALES_Q1_2003"                    0 KB
.  estimated "SH"."SALES":"SALES_Q2_2002"                    0 KB
.  estimated "SH"."SALES":"SALES_Q2_2003"                    0 KB
.  estimated "SH"."SALES":"SALES_Q3_2002"                    0 KB
.  estimated "SH"."SALES":"SALES_Q3_2003"                    0 KB
.  estimated "SH"."SALES":"SALES_Q4_2002"                    0 KB
.  estimated "SH"."SALES":"SALES_Q4_2003"                    0 KB
Total estimation using BLOCKS method: 17.56 MB
Job "SH"."SYS_EXPORT_SCHEMA_01" successfully completed at 12:28:09
```

Example 4

Performing a Schema Mode Export

```
expdp system/oracle SCHEMAS=sh
DUMPFILE=datadir1:schema1%U.dmp,datadir2:schema2%U.dmp
LOGFILE=datadir1:expschema.log
```

Example 5

Performing a Parallel Full Database Export

The FULL parameter indicates that the export is a full database mode export. All data and metadata in the database are exported.

The PARALLEL parameter specifies the maximum number of threads of active execution operating on behalf of the export job. This parameter enables you to make trade-offs between resource consumption and elapsed time. For best performance, the value specified for PARALLEL should be at least as large as the number of output files specified with the DUMPFILE parameter. Each Data Pump execution thread writes exclusively to one file at a time.

The PARALLEL parameter is valid only in the Enterprise Edition of the Oracle database. To increase or decrease the value of PARALLEL during job execution, use interactive-command mode that is described in the example below.

The FILESIZE parameter will limit the maximum size of each dump file to 2 gigabytes.

```
expdp system/oracle FULL=y
DUMPFILE=datadir1:full1%U.dmp,datadir2:full2%U.dmp FILESIZE=2g
PARALLEL=4 LOGFILE=datadir1:expfull.log JOB_NAME=expfull
```

Example 6
Attaching to and Stopping an Existing Job

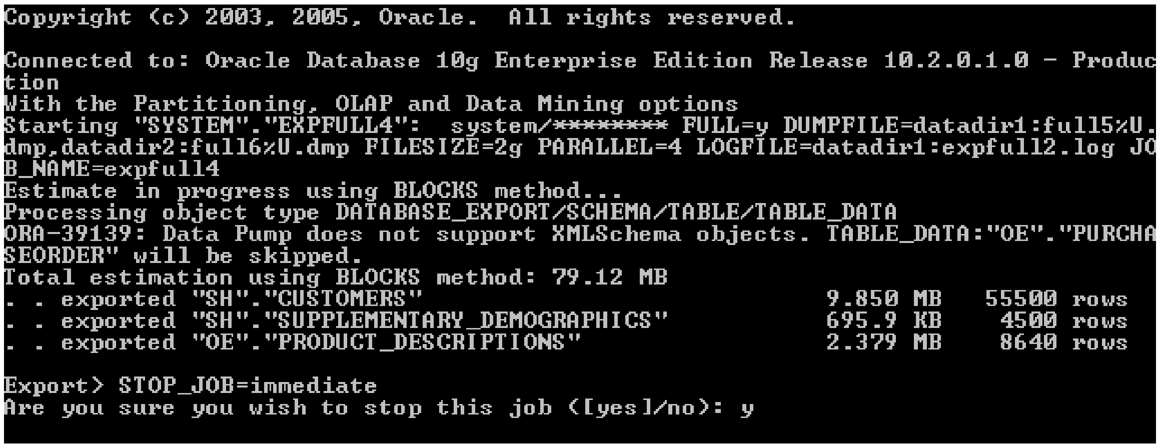
The ATTACH command attaches the client session to an existing export job and automatically places you in the interactive-command interface. Export displays a description of the job to which you are attached and also displays the export prompt. A job name does not have to be specified if there is only one export job that is associated with your schema. The job you attach to can be either currently executing or stopped

Run the full export again. While the export is running, press [Ctrl + C], to connect to the interactive-command interface, which is required for the next example. The interactive-command interface stops logging to the terminal and displays the Export prompt, from which you can enter various commands, some of which are specific to interactive mode.

```
expdp system/oracle FULL=y
DUMPFILE=datadir1:full5%U.dmp,datadir2:full6%U.dmp FILESIZE=2g
PARALLEL=4 LOGFILE=datadir1:expfull2.log JOB_NAME=expfull4
```

Press Ctrl + C

```
Export> STOP_JOB=immediate
Are you sure you wish to stop this job (y/n): y
```



```
Copyright (c) 2003, 2005, Oracle.  All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Produc
tion
With the Partitioning, OLAP and Data Mining options
Starting "SYSTEM"."EXPFULL4":  system/******** FULL=y DUMPFILE=datadir1:full5%U.
dmp,datadir2:full6%U.dmp FILESIZE=2g PARALLEL=4 LOGFILE=datadir1:expfull2.log JO
B_NAME=expfull4
Estimate in progress using BLOCKS method...
Processing object type DATABASE_EXPORT/SCHEMA/TABLE/TABLE_DATA
ORA-39139: Data Pump does not support XMLSchema objects. TABLE_DATA:"OE"."PURCHA
SEORDER" will be skipped.
Total estimation using BLOCKS method: 79.12 MB
. . exported "SH"."CUSTOMERS"                          9.850 MB   55500 rows
. . exported "SH"."SUPPLEMENTARY_DEMOGRAPHICS"         695.9 KB    4500 rows
. . exported "OE"."PRODUCT_DESCRIPTIONS"               2.379 MB    8640 rows

Export> STOP_JOB=immediate
Are you sure you wish to stop this job ([yes]/no): y
```

Example 7
Attaching to and Restarting a Stopped Job

```
expdp system/oracle ATTACH=expfull4
Export> PARALLEL=10
Export> START_JOB
Export> STATUS=600
Export> CONTINUE_CLIENT
```

Example 8
Performing a data-only table mode import

The CONTENT parameter enables you to filter the data and metadata that Import loads. The DATA_ONLY value loads only table row data; no database object definitions (metadata) are re-created.

```
impdp system/oracle TABLES=sh.costs CONTENT=data_only
DUMPFILE=datadir2:table.dmp NOLOGFILE=y
```

Example 9
Performing a Schema Mode Import

The EXCLUDE parameter enables you to filter the metadata that is imported by specifying database objects that you want to exclude from the import job. For the given mode of import, all the objects contained within the source, and all their dependent objects, are included except those specified in an EXCLUDE statement. If an object is excluded, all of its dependent objects are also excluded.

TABLE_EXISTS_ACTION instructs import about what to do if the table it is trying to create already exists. When TABLE_EXISTS_ACTION=REPLACE is specified, the import drops the existing table and then re-creates and loads it using the source database contents.

From your terminal window, issue the following import command to perform a schema import that excludes constraints, referential constraints, indexes, and materialized views using the dump file set created by the schema mode export in the Export section.

```
impdp system/oracle \
SCHEMAS=sh \
REMAP_SCHEMA=sh:sh2 \
DUMPFILE=datadir1:schema1%U.dmp,datadir2:schema2%U.dmp \
EXCLUDE=constraint, ref_constraint, index,materialized_view \
TABLE_EXISTS_ACTION=replace \
logfile=datadir1:impschema.log
```

Example 10

Not only is the Data Pump running inside the database, but also, most of the command-line features are exposed from inside the database through a PL/SQL api, DBMS_DATAPUMP. For example, you can start the export job from a PL/SQL package with the following PL/SQL code:

```
declare
   handle  number;
begin
   handle := dbms_datapump.open('EXPORT','SCHEMA');
   dbms_datapump.add_file(handle,'SCOTT3.DMP','DUMPDIR');
   dbms_datapump.metadata_filter(handle,'SCHEMA_EXPR','= ''SCOTT''');
   dbms_datapump.set_parallel(handle,4);
   dbms_datapump.start_job(handle);
   dbms_datapump.detach(handle);
end;
/
```

Example 11
Import data via a network link in Oracle 10g

In Oracle 10g, the Data Pump version of import can eliminate the dump file entirely by importing directly from another database instance.

The first step is to define a database link object to identify the source database and provide login credentials. For example, a source database in Chicago might be identified by the Oracle network service name CHI. A user in that instance, ADMIN1, logs in using the password WINDY and has the correct privileges to access the data to be imported. The following CREATE DATABASE LINK command, then, could be used to define the source database:

```
CREATE DATABASE LINK chicago
   CONNECT TO admin1 IDENTIFIED BY windy
   USING 'CHI';
```

The Data Pump import command, impdp, can now use this database link to directly access remote data. The command line parameter NETWORK_LINK points to the source database via its database link. On the local database instance in Seattle, user ADMIN2 executes the following command (all one line):

```
impdp admin2/market TABLES=customers,sales DIRECTORY=dpump1
  NETWORK_LINK=chicago
```

Example 12
Reorganize tablespaces using Oracle 10g Data Pump

Export tablespaces as a unit

In the past, the export (exp) and import (imp) utilities had three modes: You could export a single table and its dependent objects such as indexes; you could export all objects owned by a specific user; or you could export the entire database. But tablespaces were a problem. Objects owned by many different users could be stored in a given tablespace, but some of their objects might be stored in other tablespaces. So, the only solution was to query the data dictionary to find the exact list of tables and their owners and use table-mode export to export the objects individually.

In Oracle 10g, the Data Pump version of export (expdp) lets you directly export all the objects in a tablespace. The TABLESPACES parameter lets you specify which tablespace(s) you want to export.

```
TABLESPACES=name [,...]
```

This is particularly useful if you've inherited a database with a lot of dictionary-based tablespaces, and you want to reduce fragmentation by recreating the tablespaces as locally managed, and then re-import the contents.

Rename datafile names during import

When migrating a database from one platform to another prior to 10g, the DBA was required to pre-create the tablespaces and their datafiles before importing. Why? Because the dump file created by export contained datafile pathnames in the format of the original database's operating system. These pathnames would cause errors if used with a different operating system on import.

In the 10g Data Pump version of import (impdp), the REMAP_DATAFILE parameter can be used to rename these datafiles on the fly. The format is:

```
REMAP_DATAFILE=source_datafile:target_datafile
```

This option is used with FULL imports only, and the userID you specify must have the IMP_FULL_DATABASE role.

This is a very useful feature when you move databases
between platforms that have different file naming conventions. This parameter changes the source datafile name to the target datafile name in all SQL statements where the source datafile is referenced. Because the REMAP_DATAFILE value uses quotation marks, it's best to specify the parameter within a parameter file.
Example:
The parameter file, payroll.par, has the following content:

```
DIRECTORY=dpump_dir1
FULL=Y
DUMPFILE=db_full.dmp
REMAP_DATAFILE="'C:\DB1\HRDATA\PAYROLL\tbs6.dbf':'/db1/hrdata/payroll/t
bs6.dbf'"
```

You can then issue the following command:

```
> impdp username/password PARFILE=payroll.par
```

Change tablespace names during import

The impdp utility also lets you load objects into different tablespaces than they came from originally. Before 10g, the way to do this was complex. First, you had to remove your quota on the original tablespace so that you had no privileges to write there. Then, you set your default tablespace to the desired one. During the import, objects that were in the original tablespace would be stored in the default tablespace for the user. Then you had to remember to set everything back again when you were done.

In 10g import, the REMAP_TABLESPACE parameter makes this a lot easier. You still need to have quota on the destination tablespace, but no other preparation is required. Simply add the parameter:

```
REMAP_TABLESPACE=source_tablespace:target_tablespace
```

Objects will be automatically sent to the new tablespace.

Example 13
Moving data between versions

The Data Pump method for moving data between different database versions is different from the method used by original Export and Import. With original Export, you had to run an older version of Export to produce a dump file that was compatible with an older database version. With Data Pump, you use the current Export version and simply use the VERSION parameter to specify the target database version. You cannot specify versions earlier than Oracle Database 10g (since Data Pump did not exist before 10g).
Example:

```
> expdp username/password TABLES=hr.employees VERSION=10.1
```

DIRECTORY=dpump_dir1 DUMPFILE=emp.dmp

14> Monitor DataPump jobs

In interactive mode, you can get a lot of detail through the STATUS command. In SQL, you can query the following views:

- DBA_DATAPUMP_JOBS - all active Data Pump jobs and the state of each job
– USER_DATAPUMP_JOBS – summary of the user's active Data Pump jobs
– DBA_DATAPUMP_SESSIONS – all active user sessions that are attached to a Data Pump job
– V$SESSION_LONGOPS – shows all progress on each active Data Pump job

Reference:

http://www.nyoug.org/Presentations/2006/September_NYC_Metro_Meeting/200609Nanda_Data%20Pump.pdf
http://articles.techrepublic.com.com/5100-10878_11-6111806.html?tag=rbxccnbtr1
http://www.oracle.com/technology/obe/obe10gdb/integrate/datapump/datapump.htm
http://www.voug.org/handouts/10gNewFeatures_Anderson.pdf