Do More with Data Pump: Tips and Techniques

Biju Thomas
OneNeck IT Solutions
Dallas, Texas, USA

Keywords:

DBA, Data Pump, Datapump, export, import

Introduction

Data Pump was introduced in Oracle Database 10*g* and has evolved over the versions with various features and performance improvements in moving data and objects. The legacy export/import (exp/imp) utility is still maintained in all versions of database, including Oracle 12*c*, for backward compatibility and to migrate legacy data to newer versions of the database.

The Basics

- There are two sets of export/import tools available in Oracle: original export and import utilities (exp/imp) is the oldest and is maintained in the newer versions. New Oracle Data Pump export and import utilities (expdp/impdp) is available since Oracle 10*g*.
- Legacy exp/imp is a client-side tool, meaning the dump file is written to (and read from) a local directory on the machine where you run the tool. Data Pump is a server-side utility, meaning the dump file is written to (and read from) a directory on the database server. By default, the file is written to (or read from) the directory defined by the `DATA_PUMP_DIR` directory (in `dba_directories`) location.
- Data Pump export and import jobs can be paused, stopped, and restarted. Legacy exp/imp does not have this option.
- Data Pump provides a very powerful interactive command-line mode that allows you to monitor and control Data Pump export and import operations on a different terminal.
- Data Pump jobs can be initiated using the expdp/impdp tools or by using the Procedural Language/Structured Query Language (PL/SQL) package `DBMS_DATAPUMP`. The expdp and impdp tools are part of the Oracle client install.

Though table, schema, and full are the most commonly used modes, two more modes are available in Data Pump export/import:

- **Tablespace mode:** Using the `TABLESPACES=` parameter, you can specify the tablespace names, and only objects belonging to the tablespace are exported. The dependent objects, even if they belong to another tablespace, are exported. For example, when exporting a table, if the indexes belong to another tablespace, they will still be exported.
- **Transport tablespace mode:** When the `TRANSPORT_TABLESPACES=` parameter is used, only the metadata for the tables and their dependent objects within a specified set of tablespaces is exported. This mode is used to copy a tablespace physically (by copying the datafiles belonging to the tablespace) from one tablespace to another.

Oracle documentation and invoking expdp/impdp with HELP=Y provides you help with the modes of export. This session shows examples of not-so-common object and data migration requirements accomplished using Data Pump.

Let us quickly review the common parameters used in expdp and impdp:

- DIRECTORY: This is a database directory location with permission for the Oracle database owner to read and write. The default value and location for this parameter is defined by the DATA_PUMP_DIR directory.
- DUMPFLE: This is the name of the dump file. The default value is expdat.dmp.
- LOGFILE: This is the name of the log file. The default value is export.log.
- FULL/SCHEMAS/TABLES: These are mutually exclusive parameters to specify the export/import mode.
- CONTENT: This specifies if the export/import should include metadata only, data only, or both metadata and data.
- EXCLUDE/INCLUDE: These are mutually exclusive parameters to refine the object types and names of objects to include or exclude in an export/import.

Let's get into the meat of the presentation… tips and techniques!

Exporting Public Objects

Public objects do not belong to any schema, and there is no PUBLIC schema per se for you to specify PUBLIC as the schema name while exporting to get all PUBLIC objects. Almost all objects that come under a schema can be considered private to the schema owner. One very special private object is a database link, as you cannot create a database link qualifying the link name with a schema name. Pretty much all other objects can be created by a DBA qualifying the object name with a schema name to create under a specific schema. Private database links are always created under the current user only.

The EXCLUDE and INCLUDE parameters of Data Pump decides which objects to include or exclude. These parameters are very powerful and granular enough to export or import any type of object, including a database link.

The database links for the whole database can be saved to a dump file using

```
$ expdp dumpfile=uatdblinks.dmp directory=clone_save full=y
     include=db_link userid =\"/ as sysdba\"
```

The code exports public database links and all private database links. If you are interested only in the private database links belonging few schema names, you could export using

```
$ expdp dumpfile=uatdblinks.dmp directory=clone_save
     schemas=APPS,XX,XXL include=db_link userid =\"/ as sysdba\"
```

What if we need to save only the public database links in the export, or we want to follow a pattern matching for schema names? The EXCLUDE and INCLUDE parameters are very flexible and powerful, and they accept a subquery instead of hard-coded values. It is easy to figure out the public synonyms or public database links by querying the DBA_SYNONYMS or DBA_DB_LINKS views. Here is how you export only the public database links:

```
full=y
include=db_link:"IN (SELECT db_link
                     from dba_db_links Where Owner = 'PUBLIC')"
```

Since a query is used, you can imagine the possibilities are endless if you can write the appropriate query.

The next example shows how to export the public synonyms defined on the schema HR. This is useful when you have to copy a schema from one database to another. The schema level export only copies the

database objects owned by the schemas you list in the SCHEMAS parameter. Public database links and public synonyms are included in the export only if full database export (FULL=Y) is performed.

```
full=y
include=DATABASE_EXPORT/SCHEMA/PUBLIC_SYNONYM/SYNONYM:"IN (SELECT
        synonym_name from dba_synonyms
        where Owner = 'PUBLIC' and table_owner = 'HR')"
```

Finding Valid INCLUDE and EXCLUE values

Metadata filtering is implemented through the EXCLUDE and INCLUDE parameters, and these parameters are mutually exclusive. In the previous examples, you saw the use of the EXCLUDE parameter. You can specify any valid database object in the EXCLUDE and INCLUDE parameters. So, for example, you can exclude constraints, but how do you know if you can exclude NOT NULL constraints from being exported? That is, how do you determine what are the valid values for EXCLUDE and INCLUDE parameters?

Depending on the type of export performed, you can query a database view to find the valid values:

- To find the valid object types that can be excluded and included during full database export/import, query the DATABASE_EXPORT_OBJECTS view.
- To find the valid object types that can be excluded and included during schema-level export/import, query the SCHEMA_EXPORT_OBJECTS view.
- Similarly, use the TABLE_EXPORT_OBJECTS view object types during table-level or tablespace-level export/import.

Following is an example query from DATABASE_EXPORT_OBJECTS, looking for valid options to exclude constraints:

```
SQL> SELECT object_path, comments
     FROM   database_export_objects
     WHERE  object_path like '%CONSTRAINT%'
     AND    object_path not like '%/%';

OBJECT_PATH
COMMENTS
--------------------------------------------------
CONSTRAINT
Constraints (including referential constraints)

REF_CONSTRAINT
Referential constraints
```

Output shows that it is possible to exclude and include all constraints, or just referential constraints (foreign key constraints). By the way, when you exclude or include constraints, it does not consider NOT NULL constraints.

The syntax of the EXCLUDE and INCLUDE parameters allows a name clause along with the object type. The name clause evaluates the expression. The following examples help you understand the options and power of EXCLUDE and INCLUDE:

- The following exports only the foreign key constraints belonging to HR schema:

  ```
  schemas=HR
  include='REF_CONSTRAINT'
  ```

- The following exports a full database, except HR and SH schemas; notice the use of the IN expression:

  ```
  full=y
  ```

```
exclude=schema:"in ('HR','SH')"
```

- The following validates the full export did exclude HR and SH (it uses the impdp with SQLFILE). This validated if there is a CREATE USER statement for HR and SH users. INCLUDE=USER only does the user creation; no schema objects are imported.

```
sqlfile=validate.txt
include=USER
```

- The following exports all tables in the database that end with TEMP and do not belong to the SYSTEM and APPS schemas. Remember, we cannot use both INCLUDE and EXCLUDE at the same time.

```
full=y
include=table:"IN (select table_name from dba_tables
        where table_name like '%TEMP'
        and owner not in ('SYSTEM', 'APPS'))"
```

Improving Performance of Data Pump Jobs

While hardware resources, a faster disk system, and proper setup improve database performance, they will also help Data Pump jobs. Additionally, there are certain parameters you can consider to further boost performance:

- As with any database operation, Data Pump also benefits from accurate statistics available in the database for dictionary objects. Collect DBMS_STATS.GATHER_DICTIONARY_STATS before a Data Pump job for better performance if dictionary statistics were not collected recently.
- If you are using Data Pump import on Oracle Database 12*c*, consider using the option to not generate archive logs:

```
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y
```

  Thus, the objects are created by the Data Pump in NOLOGGING mode and are switched to their appropriate LOGGING or NOLOGGING mode once import completes. If your database is running in FORCE LOGGING mode, this Data Pump NOLOGGING will have no impact.
- Many more will be discussed in the presentation.

What else?

The presentation will also include tips and techniques on

- Estimating space required
- Exporting objects like foreign key constraints, and temporary tables in a schema, etc.
- Verifying what is exported by examining the contents
- Exporting subset of data using 3 different methods
- Changing table partition properties
- Changing table name, tablespace name and other properties during import
- Ignoring storage attributes during import
- Data masking
- Using data pump in legacy mode
- How to export data to import into a lower version of database
- Interactive commands
- Using data pump in PL/SQL

Contact address:

Biju Thomas
OneNeck IT Solutions
5301 N Pima Road #100
Scottsdale, AZ 85250 USA

Phone:	+1(817)374-4411
Fax:	-
Email	Biju.Thomas@OneNeck.Com
Internet:	www.oneneck.com, www.bijoos.com