3
Oracle Data Pump Import

With Oracle Data Pump Import, you can load an export dump file set into a target database, or load a target database directly from a source database with no intervening files.

- What Is Oracle Data Pump Import?
  Oracle Data Pump Import is a utility for loading an Oracle export dump file set into a target system.
- Starting Oracle Data Pump Import
  Start the Oracle Data Pump Import utility by using the `impdp` command.
- Filtering During Import Operations
  Oracle Data Pump Import provides data and metadata filtering capability, which can help you limit the type of information that you import.
- Parameters Available in Oracle Data Pump Import Command-Line Mode
  Use Oracle Data Pump parameters for Import (`impdp`) to manage your data imports.
- Commands Available in Oracle Data Pump Import Interactive-Command Mode
  In interactive-command mode, the current job continues running, but logging to the terminal is suspended, and the Import prompt (`Import>`) is displayed.
- Examples of Using Oracle Data Pump Import
  You can use these common scenario examples to learn how you can use Oracle Data Pump Import to move your data.
- Syntax Diagrams for Oracle Data Pump Import
  You can use syntax diagrams to understand the valid SQL syntax for Oracle Data Pump Import.

3.1 What Is Oracle Data Pump Import?

Oracle Data Pump Import is a utility for loading an Oracle export dump file set into a target system.

An export dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format. During an Oracle Data Pump import operation, the Import utility uses these files to locate each database object in the dump file set.

You can also use Import to load a target database directly from a source database with no intervening dump files. This type of import is called a **network import**.

Import enables you to specify whether a job should move a subset of the data and metadata from the dump file set or the source database (in the case of a network import), as determined by the import mode. This is done by using data filters and metadata filters, which are implemented through Import commands.

3.2 Starting Oracle Data Pump Import

Start the Oracle Data Pump Import utility by using the `impdp` command.

The characteristics of the import operation are determined by the import parameters you specify. These parameters can be specified either on the command line or in a parameter file.

> **Note:**
>
> - Do not start Import as `SYSDBA`, except at the request of Oracle technical support. `SYSDBA` is used internally and has specialized functions; its behavior is not the same as for general users.
> - Be aware that if you are performing a Data Pump Import into a table or tablespace created with the `NOLOGGING` clause enabled, then a redo log file may still be generated. The redo that is generated in such a case is generally for maintenance of the Data Pump control table, or related to underlying recursive space transactions, data dictionary changes, and index maintenance for indices on the table that require logging.
> - If the timezone version used by the export database is older than the version used by the import database, then loading columns with data type `TIMESTAMP WITH TIMEZONE` takes longer than it would otherwise. This additional time is required because the database must check to determine if the new timezone rules change the values being loaded.

- Oracle Data Pump Import Interfaces
  You can interact with Oracle Data Pump Import by using a command line, a parameter file, or an interactive-command mode.
- Oracle Data Pump Import Modes
  The import mode that you use for Oracle Data Pump determines what is imported.
- Network Considerations for Oracle Data Pump Import
  Learn how Oracle Data Pump Import utility `impdp` identifies instances with connect identifiers in the connection string using Oracle*Net or a net service name, and how they are different from import operations using the `NETWORK_LINK` parameter.

3.2.1 Oracle Data Pump Import Interfaces

You can interact with Oracle Data Pump Import by using a command line, a parameter file, or an interactive-command mode.

- Command-Line Interface: Enables you to specify the Import parameters directly on the command line. For a complete description of the parameters available in the command-line interface.
- Parameter File Interface: Enables you to specify command-line parameters in a parameter file. The only exception is the `PARFILE` parameter because parameter files cannot be nested. The use of parameter files is recommended if you are using parameters whose values require quotation marks.
- Interactive-Command Interface: Stops logging to the terminal and displays the Import prompt, from which you can enter various commands, some of which are specific to interactive-command mode. This mode is enabled by pressing Ctrl+C during an import operation started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.

Related Topics

- Parameters Available in Import's Command-Line Mode
- Commands Available in Import's Interactive-Command Mode

3.2.2 Oracle Data Pump Import Modes

The import mode that you use for Oracle Data Pump determines what is imported.

- About Oracle Data Pump Import Modes
  Learn how Oracle Data Pump Import modes operate during the import.
- Full Import Mode
  To specify a full import with Oracle Data Pump, use the `FULL` parameter.
- Schema Mode
  To specify a schema import with Oracle Data Pump, use the `SCHEMAS` parameter.
- Table Mode
  To specify a table mode import with Oracle Data Pump, use the `TABLES` parameter.
- Tablespace Mode
  To specify a tablespace mode import with Oracle Data Pump, use the `TABLESPACES` parameter.
- Transportable Tablespace Mode
  To specify a transportable tablespace mode import with Oracle Data Pump, use the `TRANSPORT_TABLESPACES` parameter.

3.2.2.1 About Oracle Data Pump Import Modes

Learn how Oracle Data Pump Import modes operate during the import.

The Oracle Data Pump import mode that you specify for the import applies to the source of the operation. If you specify the `NETWORK_LINK` parameter, then that source is either a dump file set, or another database.

When the source of the import operation is a dump file set, specifying a mode is optional. If you do not specify a mode, then Import attempts to load the entire dump file set in the mode in which the export operation was run.

The mode is specified on the command line, using the appropriate parameter.

> **Note:**
>
> When you import a dump file that was created by a full-mode export, the import operation attempts to copy the password for the `SYS` account from the source database. This copy sometimes fails (For example, if the password is in a shared password file). If it does fail, then after the import completes, you must set the password for the `SYS` account at the target database to a password of your choice.

3.2.2.2 Full Import Mode

To specify a full import with Oracle Data Pump, use the `FULL` parameter.

In full import mode, the entire content of the source (dump file set or another database) is loaded into the target database. This mode is the default for file-based imports. If the source is another database containing schemas other than your own, then you must have the `DATAPUMP_IMP_FULL_DATABASE` role.

Cross-schema references are not imported for non-privileged users. For example, a trigger defined on a table within the schema of the importing user, but residing in another user schema, is not imported.

The `DATAPUMP_IMP_FULL_DATABASE` role is required on the target database. If the `NETWORK_LINK` parameter is used for a full import, then the `DATAPUMP_EXP_FULL_DATABASE` role is required on the source database

A full export does not export triggers owned by schema `SYS`. You must manually recreate `SYS` triggers either before or after the full import. Oracle recommends that you recreate them after the import in case they define actions that would impede progress of the import.

Using the Transportable Option During Full Mode Imports

You can use the transportable option during a full-mode import to perform a full transportable import.

Network-based full transportable imports require use of the `FULL=YES`, `TRANSPORTABLE=ALWAYS`, and `TRANSPORT_DATAFILES=`*`datafile_name`* parameters.

File-based full transportable imports only require use of the `TRANSPORT_DATAFILES=`*`datafile_name`* parameter. Data Pump Import infers the presence of the `TRANSPORTABLE=ALWAYS` and `FULL=Y` parameters.

There are several requirements when performing a full transportable import:

- Either you must also specify the `NETWORK_LINK` parameter, or the dump file set being imported must have been created using the transportable option during export.
- If you are using a network link, then the database specified on the `NETWORK_LINK` parameter must be Oracle Database 11g release 2 (11.2.0.3) or later, and the Oracle Data Pump `VERSION` parameter must be set to at least 12. (In a non-network import, `VERSION=12` is implicitly determined from the dump file.)
- If the source platform and the target platform are of different endianness, then you must convert the data being transported so that it is in the format of the target platform. To convert the data, you can use either the `DBMS_FILE_TRANSFER` package or the `RMAN CONVERT` command.
- If the source and target platforms do not have the same endianness, then a full transportable import of encrypted tablespaces is not supported in network mode or in dump file mode

For a detailed example of performing a full transportable import, see *Oracle Database Administrator's Guide*.

Related Topics

- FULL
- TRANSPORTABLE
- Transporting Tablespaces Between Databases in *Oracle Database Administrator's Guide*

3.2.2.3 Schema Mode

To specify a schema import with Oracle Data Pump, use the `SCHEMAS` parameter.

In a schema import, only objects owned by the specified schemas are loaded. The source can be a full, table, tablespace, or a schema-mode export dump file set, or another database. If you have the `DATAPUMP_IMP_FULL_DATABASE` role, then you can specify a list of schemas, and the schemas themselves (including system privilege grants) are created in the database in addition to the objects contained within those schemas.

Cross-schema references are not imported for non-privileged users unless the other schema is remapped to the current schema. For example, a trigger defined on a table within the importing user's schema, but residing in another user's schema, is not imported.

Related Topics

- SCHEMAS

3.2.2.4 Table Mode

To specify a table mode import with Oracle Data Pump, use the `TABLES` parameter.

A table-mode import is specified using the `TABLES` parameter. In table mode, only the specified set of tables, partitions, and their dependent objects are loaded. The source can be a full, schema, tablespace, or table-mode export dump file set, or another database. You must have the `DATAPUMP_IMP_FULL_DATABASE` role to specify tables that are not in your own schema.

You can use the transportable option during a table-mode import by specifying the `TRANPORTABLE=ALWAYS` parameter with the `TABLES` parameter. If you use this option, then you must also use the `NETWORK_LINK` parameter.

To recover tables and table partitions, you can also use `RMAN` backups, and the `RMAN RECOVER TABLE` command. During this process, `RMAN` creates (and optionally imports) an Oracle Data Pump export dump file that contains the recovered objects.

Related Topics

- TABLES
- TRANSPORTABLE
- *Oracle Database Backup and Recovery User's Guide*

3.2.2.5 Tablespace Mode

To specify a tablespace mode import with Oracle Data Pump, use the `TABLESPACES` parameter.

A tablespace-mode import is specified using the `TABLESPACES` parameter. In tablespace mode, all objects contained within the specified set of tablespaces are loaded, along with the dependent objects. The source can be a full, schema, tablespace, or table-mode export dump file set, or another database. For unprivileged users, objects not remapped to the current schema will not be processed.

Related Topics

- TABLESPACES

3.2.2.6 Transportable Tablespace Mode

To specify a transportable tablespace mode import with Oracle Data Pump, use the `TRANSPORT_TABLESPACES` parameter.

In transportable tablespace mode, the metadata from another database is loaded by using either a database link (specified with the `NETWORK_LINK` parameter), or by specifying a dump file that contains the metadata. The actual data files, specified by the `TRANSPORT_DATAFILES` parameter, must be made available from the source system for use in the target database, typically by copying them over to the target system.

When transportable jobs are performed, Oracle recommends that you keep a copy of the data files on the source system until the import job has successfully completed on the target system. With a copy of the data files, if the import job should fail for some reason, then you still have uncorrupted copies of the data files.

Using this mode requires the `DATAPUMP_IMP_FULL_DATABASE` role.

> **Note:**
>
> You cannot export transportable tablespaces and then import them into a database at a lower release level. The target database must be at the same or later release level as the source database.

Related Topics

- How Does Oracle Data Pump Handle Timestamp Data?
- Using Data File Copying to Move Data

3.2.3 Network Considerations for Oracle Data Pump Import

Learn how Oracle Data Pump Import utility `impdp` identifies instances with connect identifiers in the connection string using Oracle*Net or a net service name, and how they are different from import operations using the `NETWORK_LINK` parameter.

When you start `impdp`, you can specify a connect identifier in the connect string that can be different from the current instance identified by the current Oracle System ID (SID).

You can specify a connect identifier by using either an Oracle*Net connect descriptor, or by using a net service name (usually defined in the `tnsnames.ora` file) that maps to a connect descriptor. Use of a connect identifier requires that you have Oracle Net Listener running (to start the default listener, enter `lsnrctl start`).

The following example shows this type of connection, in which `inst1` is the connect identifier:

```
impdp hr@inst1 DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp TABLES=employees
```

Import then prompts you for a password:

```
Password: password
```

To specify an Easy Connect string, the connect string must be an escaped quoted string. The Easy Connect string in its simplest form consists of a string `database_host[:port][/[service_name]`. For example, if the host is `inst1`, and you run Export on `pdb1`, then the Easy Connect string can be:

```
impdp hr@\"inst1@example.com/pdb1" DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp
TABLES=employees
```

If you prefer to use an unquoted string, then you can specify the Easy Connect connect string in a parameter file.

The local Import client connects to the database instance identified by the connect identifier `inst1` (a net service name), and imports the data from the dump file `hr.dmp` to `inst1`.

Specifying a connect identifier when you start the Import utility is different from performing an import operation using the `NETWORK_LINK` parameter. When you start an import operation and specify a connect identifier, the local Import client connects to the database instance identified by the connect identifier and imports the data from the dump file named on the command line to that database instance.

By contrast, when you perform an import using the `NETWORK_LINK` parameter, the import is performed using a database link, and there is no dump file involved. (A database link is a connection between two physical database servers that allows a client to access them as one logical database.)

Related Topics

- NETWORK_LINK
- Database Links
- Understanding the Easy Connect Naming Method

3.3 Filtering During Import Operations

Oracle Data Pump Import provides data and metadata filtering capability, which can help you limit the type of information that you import.

- Oracle Data Pump Import Data Filters
  You can specify restrictions on the table rows that you import by using Oracle Data Pump Data-specific filtering through the `QUERY` and `SAMPLE` parameters.
- Oracle Data Pump Import Metadata Filters
  To exclude or include objects in an import operation, use Oracle Data Pump metadata filters.

3.3.1 Oracle Data Pump Import Data Filters

You can specify restrictions on the table rows that you import by using Oracle Data Pump Data-specific filtering through the `QUERY` and `SAMPLE` parameters.

Oracle Data Pump can also implement Data filtering indirectly because of metadata filtering, which can include or exclude table objects along with any associated row data.

Each data filter can be specified once for each table within a job. If different filters using the same name are applied to both a particular table and to the whole job, then the filter parameter supplied for the specific table takes precedence.

3.3.2 Oracle Data Pump Import Metadata Filters

To exclude or include objects in an import operation, use Oracle Data Pump metadata filters.

Metadata filtering is implemented through the `EXCLUDE` and `INCLUDE` parameters. Metadata filters identify a set of objects that you want to be included or excluded from an Oracle Data Pump operation. For example: You can request a full import, but without Package Specifications or Package Bodies. Oracle Data Pump Import provides much greater metadata filtering capability than was provided by the original Import utility.

To use filters correctly, and to obtain the results that you expect, remember that dependent objects of an identified object are processed along with the identified object.

For example, if a filter specifies that a package is to be included in an operation, then grants upon that package will also be included. Likewise, if a table is excluded by a filter, then indexes, constraints, grants, and triggers upon the table will also be excluded by the filter.

Starting with Oracle Database 21c, Oracle Data Pump permits you to set both `INCLUDE` and `EXCLUDE` parameters in the same command. When you include both parameters in a command, Oracle Data Pump processes the `INCLUDE` parameter first, and includes all objects identified by the parameter. Then it processes the exclude parameters. Specifically, the `EXCLUDE_PATH_EXPR`, `EXCLUDE_PATH_LIST` and `EXCLUDE_TABLE` parameters are processed last. Any objects specified by the `EXCLUDE` parameter that are in the list of include objects are removed as the command executes.

If multiple filters are specified for an object type, then an implicit `AND` operation is applied to them. That is, objects participating in the job must pass *all* of the filters applied to their object types.

The same filter name can be specified multiple times within a job.

To see a list of valid object types, query the following views: `DATABASE_EXPORT_OBJECTS` for full mode, `SCHEMA_EXPORT_OBJECTS` for schema mode, `TABLE_EXPORT_OBJECTS` for table mode, `TABLESPACE_EXPORT_OBJECTS` for tablespace mode and `TRANSPORTABLE_EXPORT_OBJECTS` for transportable tablespace mode. The values listed in the `OBJECT_PATH` column are the valid object types. Note that full object path names are determined by the export mode, not by the import mode.

Related Topics

- EXCLUDE
- INCLUDE

3.4 Parameters Available in Oracle Data Pump Import Command-Line Mode

Use Oracle Data Pump parameters for Import (`impdp`) to manage your data imports.

- About Import Command-Line Mode
  Learn how to use Oracle Data Pump Import parameters in command-line mode, including case sensitivity, quotation marks, escape characters, and information about how to use examples.

- ABORT_STEP
  The Oracle Data Pump Import command-line mode `ABORT_STEP` parameter stops the job after it is initialized. Stopping the job enables the Data Pump control job table to be queried before any data is imported.
- ACCESS_METHOD
  The Oracle Data Pump Import command-line mode `ACCESS_METHOD` parameter instructs Import to use a particular method to load data
- ATTACH
  The Oracle Data Pump Import command-line mode `ATTACH` parameter attaches a worker session to an existing Data Pump control import job, and automatically places you in interactive-command mode.
- CLUSTER
  The Oracle Data Pump Import command-line mode `CLUSTER` parameter determines whether Data Pump can use Oracle Real Application Clusters (Oracle RAC) resources, and start workers on other Oracle RAC instances.
- CONTENT
  The Oracle Data Pump Import command-line mode `CONTENT` parameter enables you to filter what is loaded during the import operation.
- CREDENTIAL
  The Oracle Data Pump Import command-line mode `CREDENTIAL` parameter specifies the credential object name owned by the database user that Import uses to process files in the dump file set imported into cloud storage.
- DATA_OPTIONS
  The Oracle Data Pump Import command-line mode `DATA_OPTIONS` parameter designates how you want certain types of data to be handled during import operations.
- DIRECTORY
  The Oracle Data Pump Import command-line mode `DIRECTORY` parameter specifies the default location in which the import job can find the dump file set, and create log and SQL files.
- DUMPFILE
  The Oracle Data Pump Import command-line mode `DUMPFILE` parameter specifies the names, and optionally, the directory objects of the dump file set that Export created.
- ENABLE_SECURE_ROLES
  The Oracle Data Pump Import command-line utility `ENABLE_SECURE_ROLES` parameter prevents inadvertent use of protected roles during exports.
- ENCRYPTION_PASSWORD
  The Oracle Data Pump Import command-line mode `ENCRYPTION_PASSWORD` parameter specifies a password for accessing encrypted column data in the dump file set.
- ENCRYPTION_PWD_PROMPT
  The Oracle Data Pump Import command-line mode `ENCRYPTION_PWD_PROMPT` parameter specifies whether Data Pump should prompt you for the encryption password.
- ESTIMATE
  The Oracle Data Pump Import command-line mode `ESTIMATE` parameter instructs the source system in a network import operation to estimate how much data is generated during the import.
- EXCLUDE
  The Oracle Data Pump Import command-line mode `EXCLUDE` parameter enables you to filter the metadata that is imported by specifying objects and object types to exclude from the import job.

- FLASHBACK_SCN
  The Oracle Data Pump Import command-line mode `FLASHBACK_SCN` specifies the system change number (SCN) that Import uses to enable the Flashback utility.
- FLASHBACK_TIME
  The Oracle Data Pump Import command-line mode `FLASHBACK_TIME` parameter specifies the system change number (SCN) that Import uses to enable the Flashback utility.
- FULL
  The Oracle Data Pump Import command-line mode `FULL` parameter specifies that you want to perform a full database import.
- HELP
  The Oracle Data Pump Import command-line mode `HELP` parameter displays online help for the Import utility.
- INCLUDE
  The Oracle Data Pump Import command-line mode `INCLUDE` parameter enables you to filter the metadata that is imported by specifying objects and object types for the current import mode.
- INDEX_THRESHOLD
- JOB_NAME
  The Oracle Data Pump Import command-line mode `JOB_NAME` parameter is used to identify the import job in subsequent actions.
- KEEP_MASTER
  The Oracle Data Pump Import command-line mode `KEEP_MASTER` parameter indicates whether the Data Pump control job table should be deleted or retained at the end of an Oracle Data Pump job that completes successfully.
- LOGFILE
  The Oracle Data Pump Import command-line mode `LOGFILE` parameter specifies the name, and optionally, a directory object, for the log file of the import job.
- LOGTIME
  The Oracle Data Pump Import command-line mode `LOGTIME` parameter specifies that you want to have messages displayed with timestamps during import.
- MASTER_ONLY
  The Oracle Data Pump Import command-line mode `MASTER_ONLY` parameter indicates whether to import just the Data Pump control job table, and then stop the job so that the contents of the Data Pump control job table can be examined.
- METRICS
  The Oracle Data Pump Import command-line mode `METRICS` parameter indicates whether additional information about the job should be reported to the log file.
- NETWORK_LINK
  The Oracle Data Pump Import command-line mode `NETWORK_LINK` parameter enables an import from a source database identified by a valid database link.
- NOLOGFILE
  The Oracle Data Pump Import command-line mode `NOLOGFILE` parameter specifies whether to suppress the default behavior of creating a log file.
- ONESTEP_INDEX
- PARALLEL
  The Oracle Data Pump Import command-line mode `PARALLEL` parameter sets the maximum number of worker processes that can load in parallel.

- PARALLEL_THRESHOLD
  The Oracle Data Pump Import command-line utility `PARALLEL_THRESHOLD` parameter specifies the size of the divisor that Data Pump uses to calculate potential parallel DML based on table size.
- PARFILE
  The Oracle Data Pump Import command-line mode `PARFILE` parameter specifies the name of an import parameter file.
- PARTITION_OPTIONS
  The Oracle Data Pump Import command-line mode `PARTITION_OPTIONS` parameter specifies how you want table partitions created during an import operation.
- QUERY
  The Oracle Data Pump Import command-line mode `QUERY` parameter enables you to specify a query clause that filters the data that is imported.
- REMAP_DATA
  The Oracle Data Pump Import command-line mode `REMAP_DATA` parameter enables you to remap data as it is being inserted into a new database.
- REMAP_DATAFILE
  The Oracle Data Pump Import command-line mode `REMAP_DATAFILE` parameter changes the name of the source data file to the target data file name in all SQL statements where the source data file is referenced.
- REMAP_DIRECTORY
  The Oracle Data Pump Import command-line mode `REMAP_DIRECTORY` parameter lets you remap directories when you move databases between platforms.
- REMAP_SCHEMA
  The Oracle Data Pump Import command-line mode `REMAP_SCHEMA` parameter loads all objects from the source schema into a target schema.
- REMAP_TABLE
  The Oracle Data Pump Import command-line mode `REMAP_TABLE` parameter enables you to rename tables during an import operation.
- REMAP_TABLESPACE
  The Oracle Data Pump Import command-line mode `REMAP_TABLESPACE` parameter remaps all objects selected for import with persistent data in the source tablespace to be created in the target tablespace.
- SCHEMAS
  The Oracle Data Pump Import command-line mode `SCHEMAS` parameter specifies that you want a schema-mode import to be performed.
- SERVICE_NAME
  The Oracle Data Pump Import command-line mode `SERVICE_NAME` parameter specifies a service name that you want to use in conjunction with the `CLUSTER` parameter.
- SKIP_UNUSABLE_INDEXES
  The Oracle Data Pump Import command-line mode `SKIP_UNUSABLE_INDEXES` parameter specifies whether Import skips loading tables that have indexes that were set to the Index Unusable state (by either the system or the user).
- SOURCE_EDITION
  The Oracle Data Pump Import command-line mode `SOURCE_EDITION` parameter specifies the database edition on the remote node from which objects are fetched.

- SQLFILE
  The Oracle Data Pump Import command-line mode `SQLFILE` parameter specifies a file into which all the SQL DDL that Import prepares to execute is written, based on other Import parameters selected.
- STATUS
  The Oracle Data Pump Import command-line mode `STATUS` parameter specifies the frequency at which the job status is displayed.
- STREAMS_CONFIGURATION
  The Oracle Data Pump Import command-line mode `STREAMS_CONFIGURATION` parameter specifies whether to import any GoldenGate Replication metadata that may be present in the export dump file.
- TABLE_EXISTS_ACTION
  The Oracle Data Pump Import command-line mode `TABLE_EXISTS_ACTION` parameter specifies for Import what to do if the table it is trying to create already exists.
- REUSE_DATAFILES
  The Oracle Data Pump Import command-line mode `REUSE_DATAFILES` parameter specifies whether you want the import job to reuse existing data files for tablespace creation.
- TABLES
  The Oracle Data Pump Import command-line mode `TABLES` parameter specifies that you want to perform a table-mode import.
- TABLESPACES
  The Oracle Data Pump Import command-line mode `TABLESPACES` parameter specifies that you want to perform a tablespace-mode import.
- TARGET_EDITION
  The Oracle Data Pump Import command-line mode `TARGET_EDITION` parameter specifies the database edition into which you want objects imported.
- TRANSFORM
  The Oracle Data Pump Import command-line mode `TRANSFORM` parameter enables you to alter object creation DDL for objects being imported.
- TRANSPORT_DATAFILES
  The Oracle Data Pump Import command-line mode `TRANSPORT_DATAFILES` parameter specifies a list of data files that are imported into the target database when `TRANSPORTABLE=ALWAYS` is set during the export.
- TRANSPORT_FULL_CHECK
  The Oracle Data Pump Import command-line mode `TRANSPORT_FULL_CHECK` parameter specifies whether to verify that the specified transportable tablespace set is being referenced by objects in other tablespaces.
- TRANSPORT_TABLESPACES
  The Oracle Data Pump Import command-line mode `TRANSPORT_TABLESPACES` parameter specifies that you want to perform an import in transportable-tablespace mode over a database link.
- TRANSPORTABLE
  The optional Oracle Data Pump Import command-line mode `TRANSPORTABLE` parameter specifies either that transportable tables are imported with `KEEP_READ_ONLY`, or `NO_BITMAP_REBUILD`.
- VERIFY_CHECKSUM
  The Oracle Data Pump Import command-line utility `VERIFY_CHECKSUM` parameter specifies whether to verify dump file checksums.

- VERIFY_ONLY
  The Oracle Data Pump Import command-line utility `VERIFY_ONLY` parameter enables you to verify the checksum for the dump file.
- VERSION
  The Oracle Data Pump Import command-line mode `VERSION` parameter specifies the version of database objects that you want to import.
- VIEWS_AS_TABLES (Network Import)
  The Oracle Data Pump Import command-line mode `VIEWS_AS_TABLES` (Network Import) parameter specifies that you want one or more views to be imported as tables.

3.4.1 About Import Command-Line Mode

Learn how to use Oracle Data Pump Import parameters in command-line mode, including case sensitivity, quotation marks, escape characters, and information about how to use examples.

Before using Oracle Data Pump import parameters, read the following sections:

- Specifying Import Parameter
- Use of Quotation Marks On the Data Pump Command Line

Many of the descriptions include an example of how to use the parameter. For background information on setting up the necessary environment to run the examples, see:

- Using the Import Parameter Examples

Specifying Import Parameters

For parameters that can have multiple values specified, the values can be separated by commas or by spaces. For example, you could specify `TABLES=employees,jobs` or `TABLES=employees jobs`.

For every parameter you enter, you must enter an equal sign (=) and a value. Data Pump has no other way of knowing that the previous parameter specification is complete and a new parameter specification is beginning. For example, in the following command line, even though `NOLOGFILE` is a valid parameter, it would be interpreted as another dump file name for the `DUMPFILE` parameter:

```
impdp DIRECTORY=dpumpdir DUMPFILE=test.dmp NOLOGFILE TABLES=employees
```

This would result in two dump files being created, `test.dmp` and `nologfile.dmp`.

To avoid this, specify either `NOLOGFILE=YES` or `NOLOGFILE=NO`.

Case Sensitivity When Specifying Parameter Values

For tablespace names, schema names, table names, and so on that you enter as parameter values, Oracle Data Pump by default changes values entered as lowercase or mixed-case into uppercase. For example, if you enter `TABLE=hr.employees`, then it is changed to `TABLE=HR.EMPLOYEES`. To maintain case, you must enclose the value within quotation marks. For example, `TABLE="hr.employees"` would preserve the table name in all lower case. The name you enter must exactly match the name stored in the database.

Use of Quotation Marks On the Data Pump Command Line

Some operating systems treat quotation marks as special characters and will therefore not pass them to an application unless they are preceded by an escape character, such as the backslash (\). This is true both on the command line and within parameter files. Some

operating systems may require an additional set of single or double quotation marks on the command line around the entire parameter value containing the special characters.

The following examples are provided to illustrate these concepts. Be aware that they may not apply to your particular operating system and that this documentation cannot anticipate the operating environments unique to each user.

Suppose you specify the `TABLES` parameter in a parameter file, as follows:

```
TABLES = \"MixedCaseTableName\"
```

If you were to specify that on the command line, then some operating systems would require that it be surrounded by single quotation marks, as follows:

```
TABLES = '\"MixedCaseTableName\"'
```

To avoid having to supply additional quotation marks on the command line, Oracle recommends the use of parameter files. Also, note that if you use a parameter file and the parameter value being specified does not have quotation marks as the first character in the string (for example, `TABLES=scott."EmP"`), then the use of escape characters may not be necessary on some systems.

Using the Import Parameter Examples

If you try running the examples that are provided for each parameter, then be aware of the following:

- After you enter the username and parameters as shown in the example, Import is started and you are prompted for a password. You must supply a password before a database connection is made.
- Most of the examples use the sample schemas of the seed database, which is installed by default when you install Oracle Database. In particular, the human resources (`hr`) schema is often used.
- Examples that specify a dump file to import assume that the dump file exists. Wherever possible, the examples use dump files that are generated when you run the Export examples.
- The examples assume that the directory objects, `dpump_dir1` and `dpump_dir2`, already exist and that `READ` and `WRITE` privileges have been granted to the `hr` user for these directory objects.
- Some of the examples require the `DATAPUMP_EXP_FULL_DATABASE` and `DATAPUMP_IMP_FULL_DATABASE` roles. The examples assume that the `hr` user has been granted these roles.

If necessary, ask your DBA for help in creating these directory objects and assigning the necessary privileges and roles.

Unless specifically noted, these parameters can also be specified in a parameter file.

> **See Also:**
>
> *Oracle Database Sample Schemas*
>
> Your Oracle operating system-specific documentation for information about how special and reserved characters are handled on your system.

3.4.2 ABORT_STEP

The Oracle Data Pump Import command-line mode `ABORT_STEP` parameter stops the job after it is initialized. Stopping the job enables the Data Pump control job table to be queried before any data is imported.

Default

Null

Purpose

Stops the job after it is initialized. Stopping the job enables the Data Pump control job table to be queried before any data is imported.

Syntax and Description

```
ABORT_STEP=[n | -1]
```

The possible values correspond to a process order number in the Data Pump control job table. The result of using each number is as follows:

- *n*: If the value is zero or greater, then the import operation is started. The job is stopped at the object that is stored in the Data Pump control job table with the corresponding process order number.
- `-1` The import job uses a `NETWORK_LINK`: Abort the job after setting it up but before importing any objects.
- `-1` The import job does not use `NETWORK_LINK`: Abort the job after loading the master table and applying filters.

Restrictions

- None

Example

```
> impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 LOGFILE=schemas.log
DUMPFILE=expdat.dmp ABORT_STEP=-1
```

3.4.3 ACCESS_METHOD

The Oracle Data Pump Import command-line mode `ACCESS_METHOD` parameter instructs Import to use a particular method to load data

Default

```
AUTOMATIC
```

Purpose

Instructs Import to use a particular method to load data.

Syntax and Description

```
ACCESS_METHOD=[AUTOMATIC | DIRECT_PATH | EXTERNAL_TABLE | CONVENTIONAL |
INSERT_AS_SELECT]
```

The ACCESS_METHOD parameter is provided so that you can try an alternative method if the default method does not work for some reason. If the data for a table cannot be loaded with the specified access method, then the data displays an error for the table and continues with the next work item.

The available options are:

- AUTOMATIC: This access method is the default. Data Pump determines the best way to load data for each table. Oracle recommends that you use AUTOMATIC whenever possible, because it enables Data Pump to automatically select the most efficient method.
- DIRECT_PATH: Data Pump uses direct path load for every table.
- EXTERNAL_TABLE: Data Pump creates an external table over the data stored in the dump file, and uses a SQL INSERT AS SELECT statement to load the data into the table. Data Pump applies the APPEND hint to the INSERT statement.
- CONVENTIONAL: Data Pump creates an external table over the data stored in the dump file and reads rows from the external table one at a time. Every time it reads a row, Data Pump executes an insert statement that loads that row into the target table. This method takes a long time to load data, but it is the only way to load data that cannot be loaded by direct path and external tables.
- INSERT_AS_SELECT: Data Pump loads tables by executing a SQL INSERT AS SELECT statement that selects data from the remote database and inserts it into the target table. This option is available only for network mode imports. It is used to disable use of DIRECT_PATH when data is moved over the network.

Restrictions

- The valid options for network mode import are AUTOMATIC, DIRECT_PATH and INSERT_AS_SELECT .
- The only valid options when importing from a dump file are AUTOMATIC, DIRECT_PATH, EXTERNAL_TABLE and CONVENTIONAL.
- To use the ACCESS_METHOD parameter with network imports, you must be using Oracle Database 12c Release 2 (12.2.0.1) or later
- The ACCESS_METHOD parameter for Oracle Data Pump Import is not valid for transportable tablespace jobs.

Example

The following example enables Oracle Data Pump to load data for multiple partitions of the pre-existing table SALES at the same time.

```
impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 LOGFILE=schemas.log
DUMPFILE=expdat.dmp ACCESS_METHOD=CONVENTIONAL
```

3.4.4 ATTACH

The Oracle Data Pump Import command-line mode `ATTACH` parameter attaches a worker session to an existing Data Pump control import job, and automatically places you in interactive-command mode.

Default

If there is only one running job, then the current job in user's schema.

Purpose

This command attaches the client worker session to an existing import job, and automatically places you in interactive-command mode.

Syntax and Description

```
ATTACH [=[schema_name.]job_name]
```

Specify a *schema_name* if the schema to which you are attaching is not your own. To do this, you must have the `DATAPUMP_IMP_FULL_DATABASE` role.

A *job_name* does not have to be specified if only one running job is associated with your schema, and if the job is active. If the job you are attaching to is stopped, then you must supply the job name. To see a list of Oracle Data Pump job names, you can query the `DBA_DATAPUMP_JOBS` view or the `USER_DATAPUMP_JOBS` view.

When you are attached to the job, Import displays a description of the job, and then displays the Import prompt.

Restrictions

- When you specify the `ATTACH` parameter, the only other Oracle Data Pump parameter you can specify on the command line is `ENCRYPTION_PASSWORD`.
- If the job you are attaching to was initially started using an encryption password, then when you attach to the job, you must again enter the `ENCRYPTION_PASSWORD` parameter on the command line to re-specify that password.
- You cannot attach to a job in another schema unless it is already running.
- If the dump file set or master table for the job have been deleted, then the attach operation fails.
- Altering the Data Pump control table in any way can lead to unpredictable results.

Example

The following is an example of using the `ATTACH` parameter.

```
> impdp hr ATTACH=import_job
```

This example assumes that a job named `import_job` exists in the `hr` schema.

Related Topics

- Commands Available in Oracle Data Pump Import Interactive-Command Mode
  In interactive-command mode, the current job continues running, but logging to the terminal is suspended, and the Import prompt (`Import>`) is displayed.

3.4.5 CLUSTER

The Oracle Data Pump Import command-line mode `CLUSTER` parameter determines whether Data Pump can use Oracle Real Application Clusters (Oracle RAC) resources, and start workers on other Oracle RAC instances.

Default

`YES`

Purpose

Determines whether Oracle Data Pump can use Oracle Real Application Clusters (Oracle RAC) resources, and start workers on other Oracle RAC instances.

Syntax and Description

```
CLUSTER=[YES | NO]
```

To force Data Pump Import to use only the instance where the job is started and to replicate pre-Oracle Database 11g Release 2 (11.2) behavior, specify `CLUSTER=NO`.

To specify a specific, existing service, and constrain worker processes to run only on instances defined for that service, use the `SERVICE_NAME` parameter with the `CLUSTER=YES` parameter.

Using the `CLUSTER` parameter can affect performance, because there is some additional overhead in distributing the import job across Oracle RAC instances. For small jobs, it can be better to specify `CLUSTER=NO`, so that the job is constrained to run on the instance where it is started. Jobs that obtain the most performance benefits from using the `CLUSTER` parameter are those involving large amounts of data.

Example

```
> impdp hr DIRECTORY=dpump_dir1 SCHEMAS=hr CLUSTER=NO PARALLEL=3
NETWORK_LINK=dbs1
```

This example performs a schema-mode import of the `hr` schema. Because `CLUSTER=NO` is used, the job uses only the instance where it is started. Up to 3 parallel processes can be used. The `NETWORK_LINK` value of `dbs1` would be replaced with the name of the source database from which you were importing data. (Note that there is no dump file generated, because this is a network import.)

In this example, the `NETWORK_LINK` parameter is only used as part of the example. It is not required when using the `CLUSTER` parameter.

Related Topics

- SERVICE_NAME
  The Oracle Data Pump Import command-line mode `SERVICE_NAME` parameter specifies a service name that you want to use in conjunction with the `CLUSTER` parameter.
- Understanding How to Use Oracle Data Pump with Oracle RAC
  Using Oracle Data Pump in an Oracle Real Application Clusters (Oracle RAC) environment requires you to perform a few checks to ensure that you are making cluster member nodes available.

3.4.6 CONTENT

The Oracle Data Pump Import command-line mode `CONTENT` parameter enables you to filter what is loaded during the import operation.

Default

`ALL`

Purpose

Enables you to filter what is loaded during the import operation.

Syntax and Description

```
CONTENT=[ALL | DATA_ONLY | METADATA_ONLY]
```

- `ALL`: loads any data and metadata contained in the source. This is the default.
- `DATA_ONLY`: loads only table row data into existing tables; no database objects are created.
- `METADATA_ONLY`: loads only database object definitions. It does not load table row data. Be aware that if you specify `CONTENT=METADATA_ONLY`, then any index or table statistics imported from the dump file are locked after the import operation is complete.

Restrictions

- The `CONTENT=METADATA_ONLY` parameter and value cannot be used in conjunction with the `TRANSPORT_TABLESPACES` (transportable-tablespace mode) parameter or the `QUERY` parameter.
- The `CONTENT=ALL` and `CONTENT=DATA_ONLY` parameter and values cannot be used in conjunction with the `SQLFILE` parameter.

Example

The following is an example of using the `CONTENT` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp CONTENT=METADATA_ONLY
```

This command runs a full import that loads only the metadata in the `expfull.dmp` dump file. It runs a full import, because a full import is the default for file-based imports in which no import mode is specified.

Related Topics

- FULL
  The Oracle Data Pump Export command-line utility `FULL` parameter specifies that you want to perform a full database mode export.

3.4.7 CREDENTIAL

The Oracle Data Pump Import command-line mode `CREDENTIAL` parameter specifies the credential object name owned by the database user that Import uses to process files in the dump file set imported into cloud storage.

Default

`none`.

Purpose

Specifies the credential object name owned by the database user that Import uses to process files in the dump file set imported into Oracle Cloud Infrastructure cloud storage.

Syntax and Description

`CREDENTIAL=`*`credential_object_name`*
The import operation reads and processes files in the dump file set stored in the cloud the same as files stored on local file systems.

If the `CREDENTIAL` parameter is specified, then the value for the `DUMPFILE` parameter is a list of comma-delimited strings that Import treats as `URI` values. Starting with Oracle Database 19c, the `URI` files in the dump file set can include templates that contain the Data Pump substitution variables, such as `%U`, `%L`, and so on. For example: *`urlpath`*`exp%U.dmp`.

> **Note:**
>
> Substitution variables are only allowed in the filename portion of the `URI`.

The `DUMPFILE` parameter enables you to specify an optional directory object, using the format *`directory_object _name`*`:`*`file_name`*. However, if you specify the `CREDENTIAL` parameter, then Import does not attempt to look for a directory object name in the strings passed for `DUMPFILE`. Instead, the strings are treated as `URI` strings.

The `DIRECTORY` parameter is still used as the location of log files and `SQL` files. Also, you can still specify directory object names as part of the file names for `LOGFILE` and `SQLFILE`.

Oracle Data Pump import is no longer constrained to using the `default_credential` value in Oracle Autonomous Database. The Import `CREDENTIAL` parameter now accepts any Oracle Cloud Infrastructure (OCI) Object Storage credential created in the Oracle Autonomous Database that is added to the database using the `DBMS_CLOUD.CREATE_CREDENTIAL()` procedure. Oracle Data Pump validates if the credential exists, and if the user has access to read the credential. Any errors are returned back to the `impdp` client.

Starting with Oracle Database 21c, Oracle Data Pump Import and Export support use of Object Storage URIs for the `DUMPFILE` parameter. To use this feature for exports or imports from an object store, the CREDENTIAL parameter must be set to the Object Storage URI. This feature eases migration to and from Oracle Cloud, because it relieves you of the extra step of transferring a dumpfile to or from the object store. Note that export and import performance is slower when accessing the object store, compared to local disk access, but the process is simpler. In addition, the process should be faster than running two separate export operations from Oracle Cloud, and transferring the dumpfile from the object store to an on premises

location, or transferring the dumpfile from on premises to the object store, and then importing into Oracle Cloud.

Restrictions

The credential parameter cannot be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account.

Example: Using the Import CREDENTIAL Parameter

The following is an example of using the Import `CREDENTIAL` parameter. You can create the dump files used in this example by running the example provided for the Export `DUMPFILE` parameter, and then uploading the dump files into your cloud storage.

```
> impdp hr/your_password DIRECTORY=dpump_dir1
CREDENTIAL=user_accessible_credential
      DUMPFILE='https://objectstorage.example.com/exp1.dmp',
  'https://objectstorage.example.com/exp201.dmp',
     'https://objectstorage.example.com/exp202.dmp'
```

The import job looks in the specified cloud storage for the dump files. The log file is written to the path associated with the directory object, `dpump_dir1`, that was specified with the `DIRECTORY` parameter.

Example: Specifying a User-Defined Credential

The following example creates a new user-defined credential in the Oracle Autonomous Database, and uses the same credential in an `impdp` command:

```
   BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'MY_CRED_NAME',
    username => 'adwc_user@example.com',
    password => 'Auth token' ); END;

   > impdp admin/password@ADWC1_high
    directory=data_pump_dir
    credential=MY_cred_name …
```

Example: Importing Into Autonomous Data Warehouse Using an Object Store Credential

```
impdp admin/password@ADWC1_high \
     directory=data_pump_dir \
     credential=def_cred_name \
     dumpfile= https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/export%u.dmp \
     parallel=16 \
     encryption_pwd_prompt=yes \
     partition_options=merge \
     transform=segment_attributes:n \
     transform=dwcs_cvt_iots:y transform=constraint_use_default_index:y \

exclude=index,cluster,indextype,materialized_view,materialized_view_log,materi
alized_zonemap,db_link
```

3.4.8 DATA_OPTIONS

The Oracle Data Pump Import command-line mode DATA_OPTIONS parameter designates how you want certain types of data to be handled during import operations.

Default

There is no default. If this parameter is not used, then the special data handling options it provides simply do not take effect.

Purpose

The DATA_OPTIONS parameter designates how you want certain types of data to be handled during import operations.

Syntax and Description

```
DATA_OPTIONS = [DISABLE_APPEND_HINT | SKIP_CONSTRAINT_ERRORS |
REJECT_ROWS_WITH_REPL_CHAR | GROUP_PARTITION_TABLE_DATA |
TRUST_EXISTING_TABLE_PARTITIONS |
VALIDATE_TABLE_DATA | ENABLE_NETWORK_COMPRESSION |
CONTINUE_LOAD_ON_FORMAT_ERROR]
```

- CONTINUE_LOAD_ON_FORMAT_ERROR: Directs Oracle Data Pump to skip forward to the start of the next granule when a stream format error is encountered while loading table data.

  Stream format errors typically are the result of corrupt dump files. If Oracle Data Pump encounters a stream format error, and the original export database is not available to export the table data again, then you can use CONTINUE_LOAD_ON_FORMAT_ERROR. If Oracle Data Pump skips over data, then not all data from the source database is imported, which potentially skips hundreds or thousands of rows.

- DISABLE_APPEND_HINT: Specifies that you do not want the import operation to use the APPEND hint while loading the data object. Disabling the APPEND hint can be useful to address duplicate data. For example, you can use DISABLE_APPEND_HINT when there is a small set of data objects to load that exists already in the database, and some other application can be concurrently accessing one or more of the data objects.

  DISABLE_APPEND_HINT: Changes the default behavior, so that the APPEND hint is not used for loading data objects. When not set, the default is to use the APPEND hint for loading data objects.

- DISABLE_STATS_GATHERING: Oracle Data Pump does not gather statistics on load by default. However, some environments, such as Oracle Autonomous Database, do gather statistics on load by default. When this parameter is used, statistics gathering is suspended during the import job. The tradeoff for gathering statistics while loading data is incurring potentially significant overhead for the short-term benefit of having basic statistics gathered until you can gather full statistics on the table.

- ENABLE_NETWORK_COMPRESSION: Used for network imports in which the Oracle Data Pump ACCESS_METHOD parameter is set to DIRECT_PATH to load remote table data.

  When ENABLE_NETWORK_COMPRESSION is specified, Oracle Data Pump compresses data on the remote node before it is sent over the network to the target database, where it is decompressed. This option is useful if the network connection between the remote and local database is slow, because it reduces the amount of data sent over the network.

Setting ACCESS_METHOD=AUTOMATIC enables Oracle Data Pump to set ENABLE_NETWORK_COMPRESSION automatically during the import if Oracle Data Pump uses DIRECT_PATH for a network import.

The ENABLE_NETWORK_COMPRESSION option is ignored if Oracle Data Pump is importing data from a dump file, if the remote data base is earlier than Oracle Database 12c Release 2 (12.2), or if an INSERT_AS_SELECT statement is being used to load data from the remote database.

- GROUP_PARTITION_TABLE_DATA: Tells Oracle Data Pump to import the table data in all partitions of a table as one operation. The default behavior is to import each table partition as a separate operation. If you know that the data for a partition will not move, then choose this parameter to accelerate the import of partitioned table data. There are cases when Oracle Data Pump attempts to load only one partition at a time. It does this when the table already exists, or when there is a risk that the data for one partition might be moved to another partition.
- REJECT_ROWS_WITH_REPL_CHAR: Specifies that you want the import operation to reject any rows that experience data loss because the default replacement character was used during character set conversion.

  If REJECT_ROWS_WITH_REPL_CHAR is not set, then the default behavior is to load the converted rows with replacement characters.
- SKIP_CONSTRAINT_ERRORS: Affects how non-deferred constraint violations are handled while a data object (table, partition, or subpartition) is being loaded.

  If deferred constraint violations are encountered, then SKIP_CONSTRAINT_ERRORS has no effect on the load. Deferred constraint violations always cause the entire load to be rolled back.

  The SKIP_CONSTRAINT_ERRORS option specifies that you want the import operation to proceed even if non-deferred constraint violations are encountered. It logs any rows that cause non-deferred constraint violations, but does not stop the load for the data object experiencing the violation.

  SKIP_CONSTRAINT_ERRORS: Prevents roll back of the entire data object when non-deferred constraint violations are encountered.

  If SKIP_CONSTRAINT_ERRORS is not set, then the default behavior is to roll back the entire load of the data object on which non-deferred constraint violations are encountered.
- TRUST_EXISTING_TABLE_PARTITIONS: Tells Data Pump to load partition data in parallel into existing tables.

  Use this option when you are using Data Pump to create the table from the definition in the export database before the table data import is started. Typically, you use this parameter as part of a migration when the metadata is static, and you can move it before the databases are taken off line to migrate the data. Moving the metadata separately minimizes downtime. If you use this option, and if other attributes of the database are the same (for example, character set), then the data from the export database goes to the same partitions in the import database.

  You can create the table outside of Oracle Data Pump. However, if you create tables as a separate option from using Oracle Data Pump, then the partition attributes and partition names must be identical to the export database.

> **Note:**
>
> This option can be used for import no matter the source version of the export.

- `VALIDATE_TABLE_DATA`: Directs Oracle Data Pump to validate the number and date data types in table data columns.

  If the import encounters invalid data, then an `ORA-39376` error is written to the `.log` file. The error text includes the column name. The default is to do no validation. Use this option if the source of the Oracle Data Pump dump file is not trusted.

Restrictions

- If you use `DISABLE_APPEND_HINT`, then it can take longer for data objects to load.
- If you use `SKIP_CONSTRAINT_ERRORS`, and if a data object has unique indexes or constraints defined on it at the time of the load, then the `APPEND` hint is not used for loading that data object. Therefore, loading such data objects can take longer when the `SKIP_CONSTRAINT_ERRORS` option is used.
- Even if `SKIP_CONSTRAINT_ERRORS` is specified, it is not used unless a data object is being loaded using the external table access method.

Example

This example shows a data-only table mode import with `SKIP_CONSTRAINT_ERRORS` enabled:

```
> impdp hr TABLES=employees CONTENT=DATA_ONLY
DUMPFILE=dpump_dir1:table.dmp DATA_OPTIONS=skip_constraint_errors
```

If any non-deferred constraint violations are encountered during this import operation, then they are logged. The import continues on to completion.

3.4.9 DIRECTORY

The Oracle Data Pump Import command-line mode `DIRECTORY` parameter specifies the default location in which the import job can find the dump file set, and create log and SQL files.

Default

`DATA_PUMP_DIR`

Purpose

Specifies the default location in which the import job can find the dump file set and where it should create log and SQL files.

Syntax and Description

`DIRECTORY=`*`directory_object`*
The *`directory_object`* is the name of a database directory object. It is not the file path of an actual directory. Privileged users have access to a default directory object named `DATA_PUMP_DIR`. The definition of the `DATA_PUMP_DIR` directory can be changed by Oracle during upgrades, or when patches are applied.

Users with access to the default `DATA_PUMP_DIR` directory object do not need to use the `DIRECTORY` parameter.

A directory object specified on the DUMPFILE, LOGFILE, or SQLFILE parameter overrides any directory object that you specify for the DIRECTORY parameter. You must have Read access to the directory used for the dump file set. You must have Write access to the directory used to create the log and SQL files.

Example

The following is an example of using the DIRECTORY parameter. You can create the expfull.dmp dump file used in this example by running the example provided for the Export FULL parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
LOGFILE=dpump_dir2:expfull.log
```

This command results in the import job looking for the expfull.dmp dump file in the directory pointed to by the dpump_dir1 directory object. The dpump_dir2 directory object specified on the LOGFILE parameter overrides the DIRECTORY parameter so that the log file is written to dpump_dir2. Refer to *Oracle Database SQL Language Reference* for more information about the CREATE DIRECTORY command.

Related Topics

- Understanding Dump, Log, and SQL File Default Locations
  Oracle Data Pump is server-based, rather than client-based. Dump files, log files, and SQL files are accessed relative to server-based directory paths.
- Understanding How to Use Oracle Data Pump with Oracle RAC
  Using Oracle Data Pump in an Oracle Real Application Clusters (Oracle RAC) environment requires you to perform a few checks to ensure that you are making cluster member nodes available.
- CREATE DIRECTORY in *Oracle Database SQL Language Reference*

3.4.10 DUMPFILE

The Oracle Data Pump Import command-line mode DUMPFILE parameter specifies the names, and optionally, the directory objects of the dump file set that Export created.

Default

expdat.dmp

Purpose

Specifies the names, and, if you choose, the directory objects or default credential of the dump file set that was created by Export.

Syntax and Description

```
DUMPFILE=[directory_object:]file_name [, ...]
```

Or

```
DUMPFILE=[DEFAULT_CREDENTIAL:]URI_file [, ...]
```

The *directory_object* is optional if one is already established by the DIRECTORY parameter. If you do supply a value, then it must be a directory object that already exists, and to which you have access. A database directory object that is specified as part of the DUMPFILE parameter overrides a value specified by the DIRECTORY parameter.

The *file_name* is the name of a file in the dump file set. The file names can also be templates that contain the substitution variable %U. The Import process checks each file that matches the template to locate all files that are part of the dump file set, until no match is found. Sufficient information is contained within the files for Import to locate the entire set, provided that the file specifications defined in the DUMPFILE parameter encompass the entire set. The files are not required to have the same names, locations, or order used at export time.

The possible substitution variables are described in the following table.

Substitution Variable	Description
%U	If %U is used, then the%U expands to a 2-digit incrementing integer starting with 01.
%l, %L	Specifies a system-generated unique file name. The file names can contain a substitution variable (%L), which implies that multiple files may be generated. The substitution variable is expanded in the resulting file names into a 2-digit, fixed-width, incrementing integer starting at 01 and ending at 99 which is the same as (%U). In addition, the substitution variable is expanded in the resulting file names into a 3-digit to 10-digit, variable-width, incrementing integers starting at 100 and ending at 2147483646. The width field is determined by the number of digits in the integer. For example if the current integer is 1, then exp%Laa%L.dmp resolves to the following sequence order exp01aa01.dmp exp02aa02.dmp The 2-digit increment continues increasing, up to 99. Then, the next file names substitute a 3-digit increment: exp100aa100.dmp exp101aa101.dmp The 3-digit increments continue up until 999. Then, the next file names substitute a 4-digit increment. The substitutions continue up to the largest number substitution allowed, which is 2147483646.

Restrictions

- Dump files created on Oracle Database 11g releases with the Oracle Data Pump parameter VERSION=12 can only be imported on Oracle Database 12c Release 1 (12.1) and later.

Example of Using the Import DUMPFILE Parameter

You can create the dump files used in this example by running the example provided for the Export DUMPFILE parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=dpump_dir2:exp1.dmp, exp2%U.dmp
```

Because a directory object (dpump_dir2) is specified for the exp1.dmp dump file, the import job looks there for the file. It also looks in dpump_dir1 for dump files of the form exp2*nn*.dmp. The log file is written to dpump_dir1.

If you use the alternative DEFAULT_CREDENTIAL keyword syntax for the Import DUMPFILE parameter, then a default credential with user access must already exist. The import operation uses the default credential to read and process files in the dump file set that is stored in the cloud at the specified *URI_file* location.

The variable *URI_file* represents the name of a URI file in the dump file set. The file name cannot be the same as templates that contain the Data Pump substitution variables, such as %U, %L, and so on.

The DUMPFILE parameter DEFAULT_CREDENTIAL keyword syntax is mutually exclusive to the *directory_object* syntax. Only one form can be used in the same command line.

Example of Using the Import DUMPFILE with User-Defined Credentials

This example specifies the default location in which the import job can find the dump file set, and create log and SQL files, and specifies the credential object name owned by the database user that Import uses to process files in the dump file set that were previously imported into cloud storage.

```
> impdp admin/password@ADWC1_high
      directory=data_pump_dir
      credential=MY_cred_name …
```

Example of Using the Import DUMPFILE parameter with DEFAULT_CREDENTIAL Keywords.

You can create the dump files used in this example by running the example provided for the Export DUMPFILE parameter.

```
> impdp hr/your_password DIRECTORY=dpump_dir1
      DUMPFILE='DEFAULT_CREDENTIAL:https://objectstorage.example.com/
exp1.dmp',
  'DEFAULT_CREDENTIAL:https://objectstorage.example.com/exp201.dmp',
     'DEFAULT_CREDENTIAL:https://objectstorage.example.com/exp202.dmp'
```

The import job looks in the specified *URI_file* location for the dump files using the default credential which has already been setup for the user. The log file is written to the path associated with the directory object, *dpump_dir1* that was specified with the DIRECTORY parameter.

Example of Using the Import DUMPFILE parameter with User-Defined Credentials

This example specifies the default location in which the import job can find the dump file set, and create log and SQL files, and specifies the credential object name owned by the database user that Import uses to process files in the dump file set that were previously imported into cloud storage.

```
> impdp impdp admin/password@ADWC1_high  DIRECTORY=data_pump_dir
      DUMPFILE='MY_cred_name:https://objectstorage.example.com/exp1.dmp',
  'MY_cred_name:https://objectstorage.example.com/exp201.dmp',
     'MY_cred_name:https://objectstorage.example.com/exp202.dmp'
```

Related Topics

- DUMPFILE
- File Allocation with Oracle Data Pump
- Performing a Data-Only Table-Mode Import

3.4.11 ENABLE_SECURE_ROLES

The Oracle Data Pump Import command-line utility `ENABLE_SECURE_ROLES` parameter prevents inadvertent use of protected roles during exports.

Default

In Oracle Database 19c and later releases, the default value is `NO`.

Purpose

Some Oracle roles require authorization. If you need to use these roles with Oracle Data Pump imports, then you must explicitly enable them by setting the `ENABLE_SECURE_ROLES` parameter to `YES`.

Syntax

```
ENABLE_SECURE_ROLES=[NO|YES]
```

- `NO` Disables Oracle roles that require authorization.
- `YES` Enables Oracle roles that require authorization.

Example

```
impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 DUMPFILE=dpump_dir2:imp1.dmp,
 imp2%U.dmp ENABLE_SECURE_ROLES=YES
```

3.4.12 ENCRYPTION_PASSWORD

The Oracle Data Pump Import command-line mode `ENCRYPTION_PASSWORD` parameter specifies a password for accessing encrypted column data in the dump file set.

Default

There is no default; the value is user-supplied.

Purpose

Specifies a password for accessing encrypted column data in the dump file set. Using passwords prevents unauthorized access to an encrypted dump file set.

This parameter is also required for the transport of keys associated with encrypted tablespaces, and transporting tables with encrypted columns during a full transportable export or import operation.

The password that you enter is echoed to the screen. If you do not want the password shown on the screen as you enter it, then use the `ENCRYPTION_PWD_PROMPT` parameter.

Syntax and Description

```
ENCRYPTION_PASSWORD = password
```

If an encryption password was specified on the export operation, then this parameter is required on an import operation. The password that is specified must be the same one that was specified on the export operation.

Restrictions

- The export operation using this parameter requires the Enterprise Edition release of Oracle Database 11g or later, It is not possible to use `ENCRYPTION_PASSWORD` for an export from Standard Edition, so you cannot use this this parameter for a migration from Standard Edition to Enterprise Edition. You can use this parameter for migrations from Enterprise Edition to Standard Edition.
- Oracle Data Pump encryption features require that you have the Oracle Advanced Security option enabled. Refer to *Oracle Database Licensing Information* for information about licensing requirements for the Oracle Advanced Security option.
- The `ENCRYPTION_PASSWORD` parameter is not valid if the dump file set was created using the transparent mode of encryption.
- The `ENCRYPTION_PASSWORD` parameter is required for network-based full transportable imports where the source database has encrypted tablespaces or tables with encrypted columns.
- If the source table and target tables have different column encryption attributes, then import can fail to load the source table rows into the target table. If this issue occurs, then an error indicating a difference in column encryption properties is raised.

Example

In the following example, the encryption password, `123456`, must be specified, because it was specified when the `dpcd2be1.dmp` dump file was created.

```
> impdp hr TABLES=employee_s_encrypt DIRECTORY=dpump_dir
  DUMPFILE=dpcd2be1.dmp ENCRYPTION_PASSWORD=123456
```

During the import operation, any columns in the `employee_s_encrypt` table encrypted during the export operation are decrypted before being imported.

Related Topics

- Oracle Database Options and Their Permitted Features

3.4.13 ENCRYPTION_PWD_PROMPT

The Oracle Data Pump Import command-line mode `ENCRYPTION_PWD_PROMPT` parameter specifies whether Data Pump should prompt you for the encryption password.

Default

```
NO
```

Purpose

Specifies whether Oracle Data Pump should prompt you for the encryption password.

Syntax and Description

```
ENCRYPTION_PWD_PROMPT=[YES | NO]
```

Specify `ENCRYPTION_PWD_PROMPT=YES` on the command line to instruct Oracle Data Pump to prompt you for the encryption password. If you do not specify the value to `YES`, then you must enter the encryption password on the command line with the `ENCRYPTION_PASSWORD` parameter. The advantage to setting the parameter to `YES` is that the encryption password is not echoed to the screen when it is entered at the prompt. By contrast, if you enter the password on the command line using the `ENCRYPTION_PASSWORD` parameter, then the password appears in plain text.

The encryption password that you enter at the prompt is subject to the same criteria described for the `ENCRYPTION_PASSWORD` parameter.

If you specify an encryption password on the export operation, then you must also supply it on the import operation.

Restrictions

Concurrent use of the `ENCRYPTION_PWD_PROMPT` and `ENCRYPTION_PASSWORD` parameters is prohibited.

Example

The following example shows Oracle Data Pump first prompting for the user password, and then for the encryption password.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp ENCRYPTION_PWD_PROMPT=YES
.
.
.
Copyright (c) 1982, 2017, Oracle and/or its affiliates.  All rights reserved.

Password:

Connected to: Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 -
Development
Version 18.1.0.0.0

Encryption Password:

Master table "HR"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "HR"."SYS_IMPORT_FULL_01":  hr/******** directory=dpump_dir1
dumpfile=hr.dmp encryption_pwd_prompt=Y
.
.
.
```

3.4.14 ESTIMATE

The Oracle Data Pump Import command-line mode `ESTIMATE` parameter instructs the source system in a network import operation to estimate how much data is generated during the import.

Default

`STATISTICS`

Purpose

Instructs the source system in a network import operation to estimate how much data is generated during the import.

Syntax and Description

```
ESTIMATE=[BLOCKS | STATISTICS]
```

The valid choices for the `ESTIMATE` parameter are as follows:

- `BLOCKS`: The estimate is calculated by multiplying the number of database blocks used by the source objects times the appropriate block sizes.
- `STATISTICS`: The estimate is calculated using statistics for each table. For this method to be as accurate as possible, all tables should have been analyzed recently. (Table analysis can be done with either the SQL `ANALYZE` statement or the `DBMS_STATS` PL/SQL package.)

You can use the estimate that is generated to determine a percentage of the import job that is completed throughout the import.

Restrictions

- The Import `ESTIMATE` parameter is valid only if the `NETWORK_LINK` parameter is also specified.
- When the import source is a dump file set, the amount of data to be loaded is already known, so the percentage complete is automatically calculated.
- The estimate may be inaccurate if either the `QUERY` or `REMAP_DATA` parameter is used.

Example

In the following syntax example, you replace the variable *`source_database_link`* with the name of a valid link to the source database.

```
> impdp hr TABLES=job_history NETWORK_LINK=source_database_link
  DIRECTORY=dpump_dir1 ESTIMATE=STATISTICS
```

The `job_history` table in the `hr` schema is imported from the source database. A log file is created by default and written to the directory pointed to by the `dpump_dir1` directory object. When the job begins, an estimate for the job is calculated based on table statistics.

3.4.15 EXCLUDE

The Oracle Data Pump Import command-line mode `EXCLUDE` parameter enables you to filter the metadata that is imported by specifying objects and object types to exclude from the import job.

Default

There is no default.

Purpose

Enables you to filter the metadata that is imported by specifying objects and object types to exclude from the import job.

Syntax and Description

```
EXCLUDE=object_type[:name_clause] [, ...]
```

The *object_type* specifies the type of object to be excluded. To see a list of valid values for *object_type*, query the following views: `DATABASE_EXPORT_OBJECTS` for full mode, `SCHEMA_EXPORT_OBJECTS` for schema mode, `TABLE_EXPORT_OBJECTS` for table mode, `TABLESPACE_EXPORT_OBJECTS` for tablespace mode and `TRANSPORTABLE_EXPORT_OBJECTS` for transportable tablespace mode. The values listed in the `OBJECT_PATH` column are the valid object types.

For the given mode of import, all object types contained within the source (and their dependents) are included, except those specified in an `EXCLUDE` statement. If an object is excluded, then all of its dependent objects are also excluded. For example, excluding a table will also exclude all indexes and triggers on the table.

The *name_clause* is optional. It allows fine-grained selection of specific objects within an object type. It is a SQL expression used as a filter on the object names of the type. It consists of a SQL operator and the values against which the object names of the specified type are to be compared. The *name_clause* applies only to object types whose instances have names (for example, it is applicable to `TABLE` and `VIEW`, but not to `GRANT`). It must be separated from the object type with a colon and enclosed in double quotation marks, because single quotation marks are required to delimit the name strings. For example, you could set `EXCLUDE=INDEX:"LIKE 'DEPT%'"` to exclude all indexes whose names start with `dept`.

The name that you supply for the *name_clause* must exactly match, including upper and lower casing, an existing object in the database. For example, if the *name_clause* you supply is for a table named `EMPLOYEES`, then there must be an existing table named `EMPLOYEES` using all upper case. If the *name_clause* were supplied as `Employees` or `employees` or any other variation, then the table would not be found.

More than one `EXCLUDE` statement can be specified.

Depending on your operating system, the use of quotation marks when you specify a value for this parameter may also require that you use escape characters. Oracle recommends that you place this parameter in a parameter file, which can reduce the number of escape characters that might otherwise be needed on the command line.

As explained in the following sections, you should be aware of the effects of specifying certain objects for exclusion, in particular, `CONSTRAINT`, `GRANT`, and `USER`.

Excluding Constraints

The following constraints cannot be excluded:

- Constraints needed for the table to be created and loaded successfully (for example, primary key constraints for index-organized tables or `REF SCOPE` and `WITH ROWID` constraints for tables with `REF` columns).

This means that the following `EXCLUDE` statements will be interpreted as follows:

- `EXCLUDE=CONSTRAINT` excludes all constraints, except for any constraints needed for successful table creation and loading.
- `EXCLUDE=REF_CONSTRAINT` excludes referential integrity (foreign key) constraints.

Excluding Grants and Users

Specifying `EXCLUDE=GRANT` excludes object grants on all object types and system privilege grants.

Specifying `EXCLUDE=USER` excludes only the definitions of users, not the objects contained within users' schemas.

To exclude a specific user and all objects of that user, specify a command such as the following, where `hr` is the schema name of the user you want to exclude.

```
impdp FULL=YES DUMPFILE=expfull.dmp EXCLUDE=SCHEMA:"='HR'"
```

Note that in this example, the `FULL` import mode is specified. If no mode is specified, then `SCHEMAS` is used, because that is the default mode. However, with this example, if you do not specify `FULL`, and instead use `SCHEMAS`, followed by the `EXCLUDE=SCHEMA` argument, then that causes an error, because in that case you are indicating that you want the schema both to be imported and excluded at the same time.

If you try to exclude a user by using a statement such as `EXCLUDE=USER:"= 'HR'"`, then only `CREATE USER hr` DDL statements are excluded, which can return unexpected results.

Starting with Oracle Database 21c, Oracle Data Pump permits you to set both `INCLUDE` and `EXCLUDE` parameters in the same command. When you include both parameters in a command, Oracle Data Pump processes the `INCLUDE` parameter first, and include all objects identified by the parameter. Then it processes the exclude parameters. Any objects specified by the `EXCLUDE` parameter that are in the list of include objects are removed as the command executes.

Restrictions

- Exports of SQL firewall metadata (captures and allow-lists) with the object `SQL_FIREWALL` are supported starting with Oracle Database 23ai. However, Oracle Data Pump supports the export or import of all the existing SQL Firewall as a whole. You cannot import or export a specific capture or a specific allow-list.

Example

Assume the following is in a parameter file, `exclude.par`, being used by a DBA or some other user with the `DATAPUMP_IMP_FULL_DATABASE` role. (To run the example, you must first create this file.)

```
EXCLUDE=FUNCTION
EXCLUDE=PROCEDURE
EXCLUDE=PACKAGE
EXCLUDE=INDEX:"LIKE 'EMP%' "
```

You then issue the following command:

```
> impdp system DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp PARFILE=exclude.par
```

You can create the `expfull.dmp` dump file used in this command by running the example provided for the Export `FULL` parameter. in the FULL reference topic. All data from the `expfull.dmp` dump file is loaded, except for functions, procedures, packages, and indexes whose names start with `emp`.

Related Topics

- FULL
- Oracle Data Pump Import Metadata Filters
- Filtering During Import Operations
- About Import Command-Line Mode

3.4.16 FLASHBACK_SCN

The Oracle Data Pump Import command-line mode `FLASHBACK_SCN` specifies the system change number (SCN) that Import uses to enable the Flashback utility.

Default

There is no default

Purpose

Specifies the system change number (SCN) that Import will use to enable the Flashback utility.

Syntax and Description

```
FLASHBACK_SCN=scn_number
```

The import operation is performed with data that is consistent up to the specified *`scn_number`*.

Starting with Oracle Database 12c Release 2 (12.2), the SCN value can be a big SCN (8 bytes). See the following restrictions for more information about using big SCNs.

Restrictions

- The `FLASHBACK_SCN` parameter is valid only when the `NETWORK_LINK` parameter is also specified.

- The `FLASHBACK_SCN` parameter pertains only to the Flashback Query capability of Oracle Database. It is not applicable to Flashback Database, Flashback Drop, or Flashback Data Archive.
- `FLASHBACK_SCN` and `FLASHBACK_TIME` are mutually exclusive.
- You cannot specify a big SCN for a network export or network import from a version that does not support big SCNs.

Example

The following is a syntax example of using the `FLASHBACK_SCN` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 FLASHBACK_SCN=123456
NETWORK_LINK=source_database_link
```

When using this command, replace the variables *`123456`* and *`source_database_link`* with the SCN and the name of a source database from which you are importing data.

> **Note:**
>
> If you are on a logical standby system, then the `FLASHBACK_SCN` parameter is ignored, because SCNs are selected by logical standby. See *Oracle Data Guard Concepts and Administration* for information about logical standby databases.

Related Topics

- Logical Standby Databases in *Oracle Data Guard Concepts and Administration*

3.4.17 FLASHBACK_TIME

The Oracle Data Pump Import command-line mode `FLASHBACK_TIME` parameter specifies the system change number (SCN) that Import uses to enable the Flashback utility.

Default

There is no default

Purpose

Specifies the system change number (SCN) that Import will use to enable the Flashback utility.

Syntax and Description

```
FLASHBACK_TIME="TO_TIMESTAMP()"
```

The SCN that most closely matches the specified time is found, and this SCN is used to enable the Flashback utility. The import operation is performed with data that is consistent up to this SCN. Because the `TO_TIMESTAMP` value is enclosed in quotation marks, it would be best to put this parameter in a parameter file.

> **Note:**
>
> If you are on a logical standby system, then the `FLASHBACK_TIME` parameter is ignored because SCNs are selected by logical standby. See *Oracle Data Guard Concepts and Administration* for information about logical standby databases.

Restrictions

- This parameter is valid only when the `NETWORK_LINK` parameter is also specified.
- The `FLASHBACK_TIME` parameter pertains only to the flashback query capability of Oracle Database. It is not applicable to Flashback Database, Flashback Drop, or Flashback Data Archive.
- `FLASHBACK_TIME` and `FLASHBACK_SCN` are mutually exclusive.

Example

You can specify the time in any format that the `DBMS_FLASHBACK.ENABLE_AT_TIME` procedure accepts,. For example, suppose you have a parameter file, `flashback_imp.par`, that contains the following:

```
FLASHBACK_TIME="TO_TIMESTAMP('27-10-2012 13:40:00', 'DD-MM-YYYY HH24:MI:SS')"
```

You could then issue the following command:

```
> impdp hr DIRECTORY=dpump_dir1 PARFILE=flashback_imp.par
NETWORK_LINK=source_database_link
```

The import operation will be performed with data that is consistent with the SCN that most closely matches the specified time.

> **Note:**
>
> See *Oracle Database Development Guide* for information about using flashback

Related Topics

- About Import Command-Line Mode
  Learn how to use Oracle Data Pump Import parameters in command-line mode, including case sensitivity, quotation marks, escape characters, and information about how to use examples.
- Logical Standby Databases in *Oracle Data Guard Concepts and Administration*
- Using Oracle Flashback Query (SELECT AS OF) in *Oracle Database Development Guide*

3.4.18 FULL

The Oracle Data Pump Import command-line mode `FULL` parameter specifies that you want to perform a full database import.

Default

`YES`

Purpose

Specifies that you want to perform a full database import.

Syntax and Description

```
FULL=YES
```

A value of `FULL=YES` indicates that all data and metadata from the source is imported. The source can be a dump file set for a file-based import or it can be another database, specified with the `NETWORK_LINK` parameter, for a network import.

If you are importing from a file and do not have the `DATAPUMP_IMP_FULL_DATABASE` role, then only schemas that map to your own schema are imported.

If the `NETWORK_LINK` parameter is used and the user executing the import job has the `DATAPUMP_IMP_FULL_DATABASE` role on the target database, then that user must also have the `DATAPUMP_EXP_FULL_DATABASE` role on the source database.

Filtering can restrict what is imported using this import mode.

`FULL` is the default mode, and does not need to be specified on the command line when you are performing a file-based import, but if you are performing a network-based full import then you must specify `FULL=Y` on the command line.

You can use the transportable option during a full-mode import to perform a full transportable import.

Restrictions

- The Automatic Workload Repository (AWR) is not moved in a full database export and import operation. (See *Oracle Database Performance Tuning Guide* for information about using Data Pump to move AWR snapshots.)
- The XDB repository is not moved in a full database export and import operation. User created XML schemas are moved.
- If the target is Oracle Database 12c Release 1 (12.1.0.1) or later, and the source is Oracle Database 11g Release 2 (11.2.0.3) or later, then Full imports performed over a network link require that you set `VERSION=12`

Example

The following is an example of using the `FULL` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr DUMPFILE=dpump_dir1:expfull.dmp FULL=YES
LOGFILE=dpump_dir2:full_imp.log
```

This example imports everything from the `expfull.dmp` dump file. In this example, a `DIRECTORY` parameter is not provided. Therefore, a directory object must be provided on both the `DUMPFILE` parameter and the `LOGFILE` parameter. The directory objects can be different, as shown in this example.

Related Topics

- Transporting Automatic Workload Repository Data in *Oracle Database Performance Tuning Guide*

- Transporting Databases in *Oracle Database Administrator's Guide*
- FULL

3.4.19 HELP

The Oracle Data Pump Import command-line mode `HELP` parameter displays online help for the Import utility.

Default

`NO`

Purpose

Displays online help for the Import utility.

Syntax and Description

```
HELP=YES
```

If `HELP=YES` is specified, then Import displays a summary of all Import command-line parameters and interactive commands.

Example

This example displays a brief description of all Import parameters and commands.

```
> impdp HELP = YES
```

3.4.20 INCLUDE

The Oracle Data Pump Import command-line mode `INCLUDE` parameter enables you to filter the metadata that is imported by specifying objects and object types for the current import mode.

Default

There is no default.

Purpose

Enables you to filter the metadata that is imported by specifying objects and object types for the current import mode.

Syntax and Description

```
INCLUDE = object_type[:name_clause] [, ...]
```

The variable *object_type* in the syntax specifies the type of object that you want to include. To see a list of valid values for *object_type*, query the following views:

- Full mode: `DATABASE_EXPORT_OBJECTS`
- Schema mode: `SCHEMA_EXPORT_OBJECTS`
- Table mode: `TABLE_EXPORT_OBJECTS`
- Tablespace mode: `TABLESPACE_EXPORT_OBJECTS`
- Transportable tablespace mode: `TRANSPORTABLE_EXPORT_OBJECTS`

In the query result, the values listed in the OBJECT_PATH column are the valid object types. (See "Metadata Filters" for an example of how to perform such a query.)

Only object types in the source (and their dependents) that you explicitly specify in the INCLUDE statement are imported.

The variable *name_clause* in the syntax is optional. It enables you to perform fine-grained selection of specific objects within an object type. It is a SQL expression used as a filter on the object names of the type. It consists of a SQL operator, and the values against which the object names of the specified type are to be compared. The *name_clause* applies only to object types whose instances have names (for example, it is applicable to TABLE, but not to GRANT). It must be separated from the object type with a colon, and enclosed in double quotation marks. You must use double quotation marks, because single quotation marks are required to delimit the name strings.

The name string that you supply for the *name_clause* must exactly match an existing object in the database, including upper and lower case. For example, if the *name_clause* that you supply is for a table named EMPLOYEES, then there must be an existing table named EMPLOYEES, using all upper case characters. If the *name_clause* is supplied as Employees, or employees, or uses any other variation from the existing table names string, then the table is not found.

You can specify more than one INCLUDE statement.

Depending on your operating system, when you specify a value for this parameter with the use of quotation marks, you can also be required to use escape characters. Oracle recommends that you place this parameter in a parameter file, which can reduce the number of escape characters that you otherwise must use in the command line..

To see a list of valid paths for use with the INCLUDE parameter, query the following views:

- Full mode: DATABASE_EXPORT_OBJECTS
- Schema mode: SCHEMA_EXPORT_OBJECTS
- Table and Tablespace mode: TABLE_EXPORT_OBJECTS

Starting with Oracle Database 21c, the following additional enhancements are available:

- You can set both INCLUDE and EXCLUDE parameters in the same command.

  When you include both parameters in a command, Oracle Data Pump processes the INCLUDE parameter first, and includes all objects identified by the parameter. Then it processes the exclude parameters. Any objects specified by the EXCLUDE parameter that are in the list of include objects are removed as the command executes.

Restrictions

- Grants on objects owned by the SYS schema are never imported.
- Exports of SQL firewall metadata (captures and allow-lists) with the object SQL_FIREWALL are supported starting with Oracle Database 23ai. However, Oracle Data Pump supports the export or import of all the existing SQL Firewall as a whole. You cannot import or export a specific capture or a specific allow-list.

Example

Assume the following is in a parameter file named `imp_include.par`. This parameter file is being used by a DBA or some other user that is granted the role `DATAPUMP_IMP_FULL_DATABASE`:

```
INCLUDE=FUNCTION
INCLUDE=PROCEDURE
INCLUDE=PACKAGE
INCLUDE=INDEX:"LIKE 'EMP%' "
```

With the aid of this parameter file, you can then issue the following command:

```
> impdp system SCHEMAS=hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
PARFILE=imp_include.par
```

You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

The Import operation will load only functions, procedures, and packages from the `hr` schema and indexes whose names start with `EMP`. Although this is a privileged-mode import (the user must have the `DATAPUMP_IMP_FULL_DATABASE` role), the schema definition is not imported, because the `USER` object type was not specified in an `INCLUDE` statement.

Related Topics

- Oracle Data Pump Metadata Filters
- About Import Command-Line Mode
- FULL

3.4.21 INDEX_THRESHOLD

The Oracle Data Pump Import command-line mode `INDEX_THRESHOLD` parameter sets a size threshold for creating large indexes with a degree of parallelism greater than 1. It is used in conjunction with the `ONESTEP_INDEX` parameter.

Default

`150M`

Purpose

Sets the index size threshold. Indexes of a size equal to and above the threshold can be created with a degree of parallelism (DOP) greater than 1 if the parameter `ONESTEP_INDEX=FALSE`.

Syntax and Description

```
INDEX_THRESHOLD=string value
```

Indexes on tables that are smaller than the threshold value will be created with a degree of parallelism (DOP) of 1. An index on a table equal to or larger than the threshold will incur some additional overhead to determine the optimal DOP for creating that index, and the index will be

created with that DOP unless it is limited by the ' setting for the job itself, as discussed for the ONESTEP_INDEX parameter.

Indexes on tables that are smaller than the threshold value will be created with a DOP of 1. An index on a table equal to or larger than the threshold will incur some additional overhead to determine the optimal DOP for creating that index. The index will be created with that optimal DOP unless it is limited by the PARALLEL setting for the job itself..as discussed for the ONESTEP_INDEX parameter.

The default works well in almost all cases. It can be adjusted by experimentation in database configurations that warrant it.

An invalid threshold will produce an error.

> **Note:**
>
> You can also call the parameter by using the subprogram dbms_datapump.set_parameter(*handle*,*parameter_name*,*value*).

Restrictions

The INDEX_THRESHOLD value must be specified as a string, such as 1000B, 100k, 200kb, 100M, 200mb, 100G, 200gb, 100t, 200TB.

Example

The following is an example of using the ONESTEP_INDEX and INDEX_THRESHOLD parameters with their default settings. This command specifies that indexes equal to or larger than 150MB can be created with a DOP greater than 1. It balances large index creation DOP with the overall import job DOP for importing all objects in the job, up to the PARALLEL=16 setting for the job.

```
> impdp hr DIRECTORY=dpump_dir1 LOGFILE=parallel_import.log
JOB_NAME=imp_par3 DUMPFILE=par_exp%L.dmp PARALLEL=16 ONESTEP_INDEX=FALSE
INDEX_THRESHOLD=150M
```

Related Topics

- SET_PARAMETER Procedures in *Oracle Database PL/SQL Packages and Types Reference*

3.4.22 JOB_NAME

The Oracle Data Pump Import command-line mode JOB_NAME parameter is used to identify the import job in subsequent actions.

Default

A system-generated name of the form SYS_*IMPORT or SQLFILE_mode_NN*

Purpose

Use the JOB_NAME parameter when you want to identify the import job in subsequent actions. For example, when you want to use the ATTACH parameter to attach to a job, you use the

JOB_NAME parameter to identify the job that you want to attach. You can also use JOB_NAME to identify the job by using the views DBA_DATAPUMP_JOBS or USER_DATAPUMP_JOBS.

Syntax and Description

```
JOB_NAME=jobname_string
```

The variable *jobname_string* specifies a name of up to 128 bytes for the import job. The bytes must represent printable characters and spaces. If the string includes spaces, then the name must be enclosed in single quotation marks (for example, 'Thursday Import'). For additional information about job name restrictions, see "Database Object Names and Qualifiers" item 7 in *Oracle Database SQL Language Reference*. The job name is implicitly qualified by the schema of the user performing the import operation. The job name is used as the name of the Data Pump control import job table, which controls the export job.

The default job name is system-generated in the form SYS_IMPORT_mode_*NN* or SYS_SQLFILE_mode_*NN*, where *NN* expands to a 2-digit incrementing integer, starting at 01. For example, SYS_IMPORT_TABLESPACE_02' is a default job name.

Example

The following is an example of using the JOB_NAME parameter. You can create the expfull.dmp dump file that is used in this example by running the example provided in the Export FULL parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp JOB_NAME=impjob01
```

Related Topics

- Database Object Names and Qualifiers in *Oracle Database SQL Language Reference*
- FULL

3.4.23 KEEP_MASTER

The Oracle Data Pump Import command-line mode KEEP_MASTER parameter indicates whether the Data Pump control job table should be deleted or retained at the end of an Oracle Data Pump job that completes successfully.

Default

```
NO
```

Purpose

Indicates whether the Data Pump control job table should be deleted or retained at the end of an Oracle Data Pump job that completes successfully. The Data Pump control job table is automatically retained for jobs that do not complete successfully.

Syntax and Description

```
KEEP_MASTER=[YES | NO]
```

Restrictions

- None

Example

```
> impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 LOGFILE=schemas.log
DUMPFILE=expdat.dmp KEEP_MASTER=YES
```

3.4.24 LOGFILE

The Oracle Data Pump Import command-line mode `LOGFILE` parameter specifies the name, and optionally, a directory object, for the log file of the import job.

Default

`import.log`

Purpose

Specifies the name, and optionally, a directory object, for the log file of the import job.

Syntax and Description

`LOGFILE=[`*`directory_object`*`:]`*`file_name`*

If you specify a *`directory_object`*, then it must be one that was previously established by the DBA, and to which you have access. This parameter overrides the directory object specified with the `DIRECTORY` parameter. The default behavior is to create `import.log` in the directory referenced by the directory object specified in the `DIRECTORY` parameter.

Starting with Oracle Database 23ai, If an existing file that has a name matching the one specified with this parameter, it is overwritten only if the existing file extension is one of the following: `log`, `LOG`, `lst`, or `LST`. If the existing file extension does not match one of these extensions, then you receive the message `ORA-02604: 'file already exists'`. However, if no existing file with a matching name is found, then there is no file extension restriction.

All messages regarding work in progress, work completed, and errors encountered are written to the log file. (For a real-time status of the job, use the `STATUS` command in interactive mode.)

A log file is always created, unless you specify the `NOLOGFILE` parameter. As with the dump file set, the log file is relative to the server, and not the client.

> **Note:**
>
> Oracle Data Pump Import writes the log file using the database character set. If your client `NLS_LANG` environment sets up a different client character set from the database character set, then it is possible that table names can be different in the log file than they are when displayed on the client output screen.

Restrictions

- To perform an Oracle Data Pump Import using Oracle Automatic Storage Management (Oracle ASM), you must specify a `LOGFILE` parameter that includes a directory object that does not include the Oracle ASM + notation. That is, the log file must be written to a disk file, and not written into the Oracle ASM storage. Alternatively, you can specify `NOLOGFILE=YES`. However, this prevents the writing of the log file.

Example

The following is an example of using the LOGFILE parameter. You can create the expfull.dmp dump file used in this example by running the example provided for the Export FULL parameter.

```
> impdp hr SCHEMAS=HR DIRECTORY=dpump_dir2 LOGFILE=imp.log
 DUMPFILE=dpump_dir1:expfull.dmp
```

Because no directory object is specified on the LOGFILE parameter, the log file is written to the directory object specified on the DIRECTORY parameter.

Related Topics

- STATUS
- Using Directory Objects When Oracle Automatic Storage Management Is Enabled
- FULL

3.4.25 LOGTIME

The Oracle Data Pump Import command-line mode LOGTIME parameter specifies that you want to have messages displayed with timestamps during import.

Default

No timestamps are recorded

Purpose

Specifies that you want to have messages displayed with timestamps during import.. You can use the timestamps to figure out the elapsed time between different phases of a Data Pump operation. Such information can be helpful in diagnosing performance problems and estimating the timing of future similar operations.

Syntax and Description

```
LOGTIME=[NONE | STATUS | LOGFILE | ALL]
```

The available options are defined as follows:

- NONE: No timestamps on status or log file messages (same as default)
- STATUS: Timestamps on status messages only
- LOGFILE: Timestamps on log file messages only
- ALL: Timestamps on both status and log file messages

Restrictions

If the file specified by LOGFILE exists and it is not identified as a Data Pump LOGFILE, such as using more than one dot in the filename (specifically, a compound suffix), then it cannot be overwritten. You must specify a different filename.

Example

The following example records timestamps for all status and log file messages that are displayed during the import operation:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr LOGTIME=ALL
TABLE_EXISTS_ACTION=REPLACE
```

For an example of what the LOGTIME output looks like, see the Export LOGTIME parameter.

Related Topics

- LOGTIME

3.4.26 MASTER_ONLY

The Oracle Data Pump Import command-line mode MASTER_ONLY parameter indicates whether to import just the Data Pump control job table, and then stop the job so that the contents of the Data Pump control job table can be examined.

Default

NO

Purpose

Indicates whether to import just the Data Pump control job table and then stop the job so that the contents of the Data Pump control job table can be examined.

Syntax and Description

```
MASTER_ONLY=[YES | NO]
```

Restrictions

- If the NETWORK_LINK parameter is also specified, then MASTER_ONLY=YES is not supported.

Example

```
> impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 LOGFILE=schemas.log
DUMPFILE=expdat.dmp MASTER_ONLY=YES
```

3.4.27 METRICS

The Oracle Data Pump Import command-line mode METRICS parameter indicates whether additional information about the job should be reported to the log file.

Default

NO

Purpose

Indicates whether additional information about the job should be reported to the Oracle Data Pump log file.

Syntax and Description

```
METRICS=[YES | NO]
```

When METRICS=YES is used, the number of objects and the elapsed time are recorded in the Oracle Data Pump log file.

Restrictions

- None

Example

```
> impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 LOGFILE=schemas.log
DUMPFILE=expdat.dmp METRICS=YES
```

3.4.28 NETWORK_LINK

The Oracle Data Pump Import command-line mode `NETWORK_LINK` parameter enables an import from a source database identified by a valid database link.

Default:

There is no default

Purpose

Enables an import from a source database identified by a valid database link. The data from the source database instance is written directly back to the connected database instance.

Syntax and Description

`NETWORK_LINK=`*`source_database_link`*

The `NETWORK_LINK` parameter initiates an import using a database link. This means that the system to which the `impdp` client is connected contacts the source database referenced by the *`source_database_link`*, retrieves data from it, and writes the data directly to the database on the connected instance. There are no dump files involved.

The *`source_database_link`* provided must be the name of a database link to an available database. If the database on that instance does not already have a database link, then you or your DBA must create one using the SQL `CREATE DATABASE LINK` statement.

When you perform a network import using the transportable method, you must copy the source data files to the target database before you start the import.

If the source database is read-only, then the connected user must have a locally managed tablespace assigned as the default temporary tablespace on the source database. Otherwise, the job will fail.

This parameter is required when any of the following parameters are specified: `FLASHBACK_SCN`, `FLASHBACK_TIME`, `ESTIMATE`, `TRANSPORT_TABLESPACES`, or `TRANSPORTABLE`.

The following types of database links are supported for use with Oracle Data Pump Import:

- Public fixed user
- Public connected user
- Public shared user (only when used by link owner)
- Private shared user (only when used by link owner)
- Private fixed user (only when used by link owner)

> **Caution:**
>
> If an import operation is performed over an unencrypted network link, then all data is imported as clear text even if it is encrypted in the database. See *Oracle Database Security Guide* for more information about network security.

Restrictions

- The following types of database links are not supported for use with Oracle Data Pump Import:
  - Private connected user
  - Current user
- The Import `NETWORK_LINK` parameter is not supported for tables containing SecureFiles that have ContentType set, or that are currently stored outside of the SecureFiles segment through Oracle Database File System Links.
- Network imports do not support the use of evolved types.
- When operating across a network link, Data Pump requires that the source and target databases differ by no more than two versions. For example, if one database is Oracle Database 12c, then the other database must be 12c, 11g, or 10g. Note that Oracle Data Pump checks only the major version number (for example, 10g, 11g, 12c), not specific release numbers (for example, 12.1, 12.2, 11.1, 11.2, 10.1, or 10.2).
- If the `USERID` that is executing the import job has the `DATAPUMP_IMP_FULL_DATABASE` role on the target database, then that user must also have the `DATAPUMP_EXP_FULL_DATABASE` role on the source database.
- Network mode import does not use parallel query (PQ) child processes.
- Metadata cannot be imported in parallel when the `NETWORK_LINK` parameter is also used
- When transporting a database over the network using full transportable import, auditing cannot be enabled for tables stored in an administrative tablespace (such as `SYSTEM` and `SYSAUX`) if the audit trail information itself is stored in a user-defined tablespace.

Example

In the following syntax example, replace *`source_database_link`* with the name of a valid database link.

```
> impdp hr TABLES=employees DIRECTORY=dpump_dir1
NETWORK_LINK=source_database_link EXCLUDE=CONSTRAINT
```

This example results in an import of the `employees` table (excluding constraints) from the source database. The log file is written to `dpump_dir1`, specified on the `DIRECTORY` parameter.

Related Topics

- PARALLEL

> **See Also:**
>
> - *Oracle Database Administrator's Guide* for more information about database links
> - *Oracle Database SQL Language Reference* for more information about the `CREATE DATABASE LINK` statement
> - *Oracle Database Administrator's Guide* for more information about locally managed tablespaces

3.4.29 NOLOGFILE

The Oracle Data Pump Import command-line mode `NOLOGFILE` parameter specifies whether to suppress the default behavior of creating a log file.

Default

`NO`

Purpose

Specifies whether to suppress the default behavior of creating a log file.

Syntax and Description

```
NOLOGFILE=[YES | NO]
```

If you specify `NOLOGFILE=YES` to suppress creation of a log file, then progress and error information is still written to the standard output device of any attached clients, including the client that started the original export operation. If there are no clients attached to a running job, and you specify `NOLOGFILE=YES`, then you run the risk of losing important progress and error information.

Example

The following is an example of using the `NOLOGFILE` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp NOLOGFILE=YES
```

This command results in a full mode import (the default for file-based imports) of the `expfull.dmp` dump file. No log file is written, because `NOLOGFILE` is set to `YES`.

3.4.30 ONESTEP_INDEX

The Oracle Data Pump Import command-line mode `ONESTEP_INDEX` parameter attempts to optimize index creation concurrency and balance it with job parallelism. It is used in conjunction with the `INDEX_THRESHOLD` parameter.

Default

`FALSE`

Purpose

Allows creation of large indexes (determined by the INDEX_THRESHOLD parameter) with a degree of parallelism (DOP) greater than 1.

Syntax and Description

```
ONESTEP_INDEX=[TRUE | FALSE]
```

ONESTEP_INDEX=FALSE, in conjunction with the INDEX_THRESHOLD parameter starts a two-step process to balance parallelism for creating very large indexes with parallelism for importing all objects in the overall import job, up to the PARALLEL parameter setting for the job. Oracle Data Pump attempts to create all the indexes in the shortest amount of time, given the constraints of the job. Large index creation is prioritized first because large indexes take longer to create. In most cases, FALSE will yield better performance than TRUE.

ONESTEP_INDEX=TRUE starts index creation using a single step process with a DOP of 1 for each index created during import. However, multiple indexes can be created in parallel up to the DOP setting for the import job. TRUE can be optimal if all indexes are relatively small, below the INDEX_THRESHOLD default value. TRUE is also the default for jobs with PARALLEL=1.

You can also call the parameter by using the DBMS_DATAPUMP.SET_PARAMETER() subprogram. When you use the DBMS_DATAPUMP subprogram, you then specify ONESTEP_INDEX = [1,0] instead of [TRUE,FALSE]

Restrictions

After a method is selected, it cannot be changed for the job. This restriction also applies to a job restart.

Example

The following is an example of using the ONESTEP_INDEX and INDEX_THRESHOLD parameters with their default settings. This command specifies that indexes equal to or larger than 150MB can be created with a DOP greater than 1. It balances large index creation DOP with the overall import job DOP for importing all objects in the job, up to the PARALLEL=16 setting for the job.

```
> impdp hr DIRECTORY=dpump_dir1 LOGFILE=parallel_import.log
JOB_NAME=imp_par3 DUMPFILE=par_exp%L.dmp PARALLEL=16 ONESTEP_INDEX=FALSE
INDEX_THRESHOLD=150M
```

Related Topics

- SET_PARAMETER Procedures in *Oracle Database PL/SQL Packages and Types Reference*

3.4.31 PARALLEL

The Oracle Data Pump Import command-line mode PARALLEL parameter sets the maximum number of worker processes that can load in parallel.

Default

1

Purpose

Specifies the maximum number of worker processes of active execution operating on behalf of the Data Pump control import job.

Syntax and Description

```
PARALLEL=integer
```

The value that you specify for *integer* specifies the maximum number of processes of active execution operating on behalf of the import job. This execution set consists of a combination of worker processes and parallel input/output (I/O) server processes. The Data Pump control process, idle worker processes, and worker processes acting as parallel execution coordinators in parallel I/O operations do not count toward this total. This parameter enables you to make trade-offs between resource consumption and elapsed time.

If the source of the import is a dump file set consisting of files, then multiple processes can read from the same file, but performance can be limited by I/O contention.

To increase or decrease the value of `PARALLEL` during job execution, use interactive-command mode.

Using PARALLEL During a Network Mode Import

During a network mode import, the `PARALLEL` parameter defines the maximum number of worker processes that can be assigned to the job. To understand the effect of the `PARALLEL` parameter during a network import mode, it is important to understand the concept of "`table_data` objects" as defined by Oracle Data Pump. When Oracle Data Pump moves data, it considers the following items to be individual "`table_data` objects:"

- a complete table (one that is not partitioned or subpartitioned)
- partitions, if the table is partitioned but not subpartitioned
- subpartitions, if the table is subpartitioned

For example:

- A nonpartitioned table, `scott.non_part_table`, has one `table_data` object:

  `scott.non_part_table`

- A partitioned table, `scott.part_table` (having partition `p1` and partition `p2`), has two `table_data` objects:

  `scott.part_table:p1`

  `scott.part_table:p2`

- A subpartitioned table, `scott.sub_part_table` (having partition `p1` and `p2`, and subpartitions `p1s1`, `p1s2`, `p2s1`, and `p2s2`) has four `table_data` objects:

  `scott.sub_part_table:p1s1`

  `scott.sub_part_table:p1s2`

  `scott.sub_part_table:p2s1`

  `scott.sub_part_table:p2s2`

During a network mode import, each `table_data` object is assigned its own worker process, up to the value specified for the `PARALLEL` parameter. No parallel query (PQ) worker processes are

assigned because network mode import does not use parallel query (PQ) worker processes. Multiple `table_data` objects can be unloaded at the same time. However, each `table_data` object is unloaded using a single process.

Using PARALLEL During An Import In An Oracle RAC Environment

In an Oracle Real Application Clusters (Oracle RAC) environment, if an import operation has `PARALLEL=1`, then all Oracle Data Pump processes reside on the instance where the job is started. Therefore, the directory object can point to local storage for that instance.

If the import operation has `PARALLEL` set to a value greater than 1, then Oracle Data Pump processes can reside on instances other than the one where the job was started. Therefore, the directory object must point to shared storage that is accessible by all Oracle RAC cluster member nodes.

Restrictions

- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later.
- Transportable tablespace metadata cannot be imported in parallel.
- To import a table or table partition in parallel (using parallel query worker processes), you must have the `DATAPUMP_IMP_FULL_DATABASE` role.
- In addition, the following objects cannot be imported in parallel:
  - `TRIGGER`
  - `VIEW`
  - `OBJECT_GRANT`
  - `SEQUENCE`
  - `CONSTRAINT`
  - `REF_CONSTRAINT`

Example

The following is an example of using the `PARALLEL` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 LOGFILE=parallel_import.log
JOB_NAME=imp_par3 DUMPFILE=par_exp%U.dmp PARALLEL=3
```

This command imports the dump file set that is created when you run the example for the Export `PARALLEL` parameter) The names of the dump files are `par_exp01.dmp`, `par_exp02.dmp`, and `par_exp03.dmp`.

Related Topics

- PARALLEL

3.4.32 PARALLEL_THRESHOLD

The Oracle Data Pump Import command-line utility `PARALLEL_THRESHOLD` parameter specifies the size of the divisor that Data Pump uses to calculate potential parallel DML based on table size.

Default

`250MB`

Purpose

`PARALLEL_THRESHOLD` should only be used with export or import jobs of a single unpartitioned table, or one partition of a partitioned table. When you specify `PARALLEL` in the job, you can specify `PARALLEL_THRESHOLD` to modify the size of the divisor that Oracle Data Pump uses to determine if a table should be exported or imported using parallel data manipulation statements (PDML) during imports and exports. If you specify a lower value than the default, then it enables a smaller table size to use the Oracle Data Pump parallel algorithm. For example, if you have a 100MB table and you want it to use PDML of 5, to break it into five units, then you specify `PARALLEL_THRESHOLD=20M`

Syntax and Description

The parameter value specifies the threshold size in bytes:

`PARALLEL_THRESHOLD=`*`size-in-bytes`*

For a single table export or import, if you want a higher degree of parallelism, then you may want to set `PARALLEL_THRESHOLD` to lower values, to take advantage of parallelism for a smaller table or table partition. However, the benefit of this resource allocation can be limited by the performance of the I/O of the file systems to which you are loading or unloading. Also, if the job involves more than one object, for both tables and metadata objects, then the PQ allocation request specified by `PARALLEL` with `PARALLEL_THRESHOLD` is of limited value. The actual amount of PQ processes allocated to a table is impacted by how many operations Oracle Data Pump is running concurrently, where the amount of parallelism has to be shared. The database, the optimizer, and the execution plan produced by the optimizer for the SQL determine the actual degree of parallelism used to load or unload the object specified in the job.

One use case for this parameter: Using Oracle Data Pump to load a large table from one database into a larger table in another database. For example: Uploading weekly sales data from an OLTP database into a reporting or business analytics data warehouse database.

Restrictions

`PARALLEL_THRESHOLD` is used only in conjunction when the `PARALLEL` parameter is specified with a value greater than 1.

Example

The following is an example of using the `PARALLEL_THRESHOLD` parameter to export the table `table_to_use_PDML`, where the size of the divisor for PQ processes is set to 1 KB, the

variables *user* and *user-password* are the user and password of the user running Import (impdp), and the job name is parathresh_example.

```
impdp user/user-password \
    directory=dpump_dir \
    dumpfile=parathresh_example.dmp
    tables=table_to_use_PDML \
    parallel=8 \
    parallel_threshold=1K \
    job_name=parathresh_example
```

3.4.33 PARFILE

The Oracle Data Pump Import command-line mode PARFILE parameter specifies the name of an import parameter file.

Default

There is no default

Purpose

Specifies the name of an import parameter file, also known as a **parfile**.

Syntax and Description

```
PARFILE=[directory_path]file_name
```

A parameter file allows you to specify Oracle Data Pump parameters within a file. Whe you create a parameter file, that file can be specified on the command line instead of entering all the individual commands. This option can be useful if you use the same parameter combination many times. The use of parameter files is also highly recommended if you are using parameters whose values require the use of quotation marks.

A directory object is not specified for the parameter file because unlike dump files, log files, and SQL files which are created and written by the server, the parameter file is opened and read by the impdp client. The default location of the parameter file is the user's current directory.

Within a parameter file, a comma is implicit at every newline character so you do not have to enter commas at the end of each line. If you have a long line that wraps, such as a long table name, enter the backslash continuation character (\) at the end of the current line to continue onto the next line.

The contents of the parameter file are written to the Oracle Data Pump log file.

Restrictions

- The PARFILE parameter cannot be specified within a parameter file.

Example

Suppose the content of an example parameter file, hr_imp.par, are as follows:

```
TABLES= countries, locations, regions
DUMPFILE=dpump_dir2:exp1.dmp,exp2%U.dmp
```

```
DIRECTORY=dpump_dir1
PARALLEL=3
```

You can then issue the following command to execute the parameter file:

```
> impdp hr PARFILE=hr_imp.par
```

As a result of the command, the tables named `countries`, `locations`, and `regions` are imported from the dump file set that is created when you run the example for the Export `DUMPFILE` parameter. (See the Export `DUMPFILE` parameter.) The import job looks for the `exp1.dmp` file in the location pointed to by `dpump_dir2`. It looks for any dump files of the form `exp2`*`nn`*`.dmp` in the location pointed to by `dpump_dir1`. The log file for the job is also written to `dpump_dir1`.

Related Topics

- DUMPFILE
- About Import Command-Line Mode

3.4.34 PARTITION_OPTIONS

The Oracle Data Pump Import command-line mode `PARTITION_OPTIONS` parameter specifies how you want table partitions created during an import operation.

Default

The default is `departition` when partition names are specified on the `TABLES` parameter and `TRANPORTABLE=ALWAYS` is set (whether on the import operation or during the export). Otherwise, the default is `none`.

Purpose

Specifies how you want table partitions created during an import operation.

Syntax and Description

```
PARTITION_OPTIONS=[NONE | DEPARTITION | MERGE]
```

A value of `NONE` creates tables as they existed on the system from which the export operation was performed. If the export was performed with the transportable method, with a partition or subpartition filter, then you cannot use either the `NONE` option or the `MERGE` option. In that case, you must use the `DEPARTITION` option.

A value of `DEPARTITION` promotes each partition or subpartition to a new individual table. The default name of the new table is the concatenation of the table and partition name, or the table and subpartition name, as appropriate.

A value of `MERGE` combines all partitions and subpartitions into one table.

Parallel processing during import of partitioned tables is subject to the following:

- If a partitioned table is imported into an existing partitioned table, then Data Pump only processes one partition or subpartition at a time, regardless of any value specified with the `PARALLEL` parameter.

- If the table into which you are importing does not already exist, and Data Pump has to create it, then the import runs in parallel up to the parallelism specified on the `PARALLEL` parameter when the import is started.

Restrictions

- You use departitioning to create and populate tables that are based on the source tables partitions.

  To avoid naming conflicts, when the value for `PARTITION_OPTIONS` is set to `DEPARTITION`, then the dependent objects, such as constraints and indexes, are not created along with these tables. This error message is included in the log file if any tables are affected by this restriction: `ORA-39427: Dependent objects of partitioned tables will not be imported`. To suppress this message, you can use the `EXCLUDE` parameter to exclude dependent objects from the import.
- When the value for `PARTITION_OPTIONS` is set to `MERGE`, domain indexes are not created with these tables. If this event occurs, then the error is reported in the log file: `ORA-39426: Domain indexes of partitioned tables will not be imported`. To suppress this message, you can use the `EXCLUDE` parameter to exclude the indexes: `EXCLUDE=DOMAIN_INDEX`.
- If the export operation that created the dump file was performed with the transportable method, and if a partition or subpartition was specified, then the import operation must use the `DEPARTITION` option.
- If the export operation that created the dump file was performed with the transportable method, then the import operation cannot use `PARTITION_OPTIONS=MERGE`.
- If there are any grants on objects being departitioned, then an error message is generated, and the objects are not loaded.

Example

The following example assumes that the `sh.sales` table has been exported into a dump file named `sales.dmp`. It uses the `merge` option to merge all the partitions in `sh.sales` into one non-partitioned table in `scott` schema.

```
> impdp system TABLES=sh.sales PARTITION_OPTIONS=MERGE
DIRECTORY=dpump_dir1 DUMPFILE=sales.dmp REMAP_SCHEMA=sh:scott
```

Related Topics

- TRANSPORTABLE

> **See Also:**
>
> The Export `TRANSPORTABLE` parameter for an example of performing an import operation using `PARTITION_OPTIONS=DEPARTITION`

3.4.35 QUERY

The Oracle Data Pump Import command-line mode `QUERY` parameter enables you to specify a query clause that filters the data that is imported.

Default

There is no default

Purpose

Enables you to specify a query clause that filters the data that is imported.

Syntax and Description

`QUERY=[[schema_name.]table_name:]query_clause`

The *query_clause* typically is a SQL `WHERE` clause for fine-grained row selection. However, it can be any SQL clause. For example, you can use an `ORDER BY` clause to speed up a migration from a heap-organized table to an index-organized table. If a schema and table name are not supplied, then the query is applied to (and must be valid for) all tables in the source dump file set or database. A table-specific query overrides a query applied to all tables.

When you want to apply the query to a specific table, you must separate the table name from the query cause with a colon (:). You can specify more than one table-specific query , but only one query can be specified per table.

If the `NETWORK_LINK` parameter is specified along with the `QUERY` parameter, then any objects specified in the *query_clause* that are on the remote (source) node must be explicitly qualified with the `NETWORK_LINK` value. Otherwise, Data Pump assumes that the object is on the local (target) node; if it is not, then an error is returned and the import of the table from the remote (source) system fails.

For example, if you specify `NETWORK_LINK=dblink1`, then the *query_clause* of the `QUERY` parameter must specify that link, as shown in the following example:

```
QUERY=(hr.employees:"WHERE last_name IN(SELECT last_name
FROM hr.employees@dblink1)")
```

Depending on your operating system, the use of quotation marks when you specify a value for this parameter may also require that you use escape characters. Oracle recommends that you place this parameter in a parameter file, which can reduce the number of escape characters that might otherwise be needed on the command line. See ”About Import Command-Line Mode."

If you use the `QUERY` parameter , then the external tables method (rather than the direct path method) is used for data access.

To specify a schema other than your own in a table-specific query, you must be granted access to that specific table.

Restrictions

- When trying to select a subset of rows stored in the export dump file, the `QUERY` parameter cannot contain references to virtual columns for import

The reason for this restriction is that virtual column values are only present in a table in the database. Such a table does not contain the virtual column data in an Oracle Data Pump export file, so having a reference to a virtual column in an import QUERY parameter does not match any known column in the source table in the dump file. However, you can include the virtual column in an import QUERY parameter if you use a network import link (NETWORK_LINK=*dblink to source db*) that imports directly from the source table in the remote database.

- You cannot use the QUERY parameter with the following parameters:
  - CONTENT=METADATA_ONLY
  - SQLFILE
  - TRANSPORT_DATAFILES
- When the QUERY parameter is specified for a table, Oracle Data Pump uses external tables to load the target table. External tables uses a SQL INSERT statement with a SELECT clause. The value of the QUERY parameter is included in the WHERE clause of the SELECT portion of the INSERT statement. If the QUERY parameter includes references to another table with columns whose names match the table being loaded, and if those columns are used in the query, then you must use a table alias to distinguish between columns in the table being loaded, and columns in the SELECT statement with the same name.

  For example, suppose you are importing a subset of the sh.sales table based on the credit limit for a customer in the sh.customers table. In the following example, the table alias used by Data Pump for the table being loaded is KU$. KU$ is used to qualify the cust_id field in the QUERY parameter for loading sh.sales. As a result, Data Pump imports only rows for customers whose credit limit is greater than $10,000.

  ```
  QUERY='sales:"WHERE EXISTS (SELECT cust_id FROM customers c
  WHERE cust_credit_limit > 10000 AND ku$.cust_id = c.cust_id)"'
  ```

  If KU$ is not used for a table alias, then all rows are loaded:

  ```
  QUERY='sales:"WHERE EXISTS (SELECT cust_id FROM customers c
  WHERE cust_credit_limit > 10000 AND cust_id = c.cust_id)"'
  ```

- The maximum length allowed for a QUERY string is 4000 bytes, including quotation marks, which means that the actual maximum length allowed is 3998 bytes.

Example

The following is an example of using the QUERY parameter. You can create the expfull.dmp dump file used in this example by running the example provided for the Export FULL parameter. See the Export FULL parameter. Because the QUERY value uses quotation marks, Oracle recommends that you use a parameter file.

Suppose you have a parameter file, query_imp.par, that contains the following:

```
QUERY=departments:"WHERE department_id < 120"
```

You can then enter the following command:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
  PARFILE=query_imp.par NOLOGFILE=YES
```

All tables in `expfull.dmp` are imported, but for the `departments` table, only data that meets the criteria specified in the `QUERY` parameter is imported.

Related Topics

- About Import Command-Line Mode
- FULL

3.4.36 REMAP_DATA

The Oracle Data Pump Import command-line mode `REMAP_DATA` parameter enables you to remap data as it is being inserted into a new database.

Default

There is no default

Purpose

The `REMAP_DATA` parameter enables you to remap data as it is being inserted into a new database. A common use is to regenerate primary keys to avoid conflict when importing a table into a pre-existing table on the target database.

You can specify a remap function that takes as a source the value of the designated column from either the dump file or a remote database. The remap function then returns a remapped value that replaces the original value in the target database.

The same function can be applied to multiple columns being dumped. This function is useful when you want to guarantee consistency in remapping both the child and parent column in a referential constraint.

Syntax and Description

```
REMAP_DATA=[schema.]tablename.column_name:[schema.]pkg.function
```

The following is a list of each syntax element, in the order in which they appear in the syntax:

schema: the schema containing the table that you want remapped. By default, this schema is the schema of the user doing the import.

tablename: the table whose column is remapped.

column_name: the column whose data is to be remapped.

schema: the schema containing the PL/SQL package you created that contains the remapping function. As a default, this is the schema of the user doing the import.

pkg: the name of the PL/SQL package you created that contains the remapping function.

function: the name of the function within the PL/SQL that is called to remap the column table in each row of the specified table.

Restrictions

- The data types and sizes of the source argument and the returned value must both match the data type and size of the designated column in the table.
- Remapping functions should not perform commits or rollbacks except in autonomous transactions.

- The use of synonyms as values for the REMAP_DATA parameter is not supported. For example, if the regions table in the hr schema had a synonym of regn, an error would be returned if you specified regn as part of the REMPA_DATA specification.
- Remapping LOB column data of a remote table is not supported.
- REMAP_DATA does not support columns of the following types: User-Defined Types, attributes of User-Defined Types, LONG, REF, VARRAY, Nested Tables, BFILE, and XMLtype.

Example

The following example assumes a package named remap has been created that contains a function named plusx that changes the values for first_name in the employees table.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expschema.dmp
TABLES=hr.employees REMAP_DATA=hr.employees.first_name:hr.remap.plusx
```

3.4.37 REMAP_DATAFILE

The Oracle Data Pump Import command-line mode REMAP_DATAFILE parameter changes the name of the source data file to the target data file name in all SQL statements where the source data file is referenced.

Default

There is no default

Purpose

Changes the name of the source data file to the target data file name in all SQL statements where the source data file is referenced: CREATE TABLESPACE, CREATE LIBRARY, and CREATE DIRECTORY.

Syntax and Description

```
REMAP_DATAFILE=source_datafile:target_datafile
```

Remapping data files is useful when you move databases between platforms that have different file naming conventions. The *source_datafile* and *target_datafile* names should be exactly as you want them to appear in the SQL statements where they are referenced. Oracle recommends that you enclose data file names in quotation marks to eliminate ambiguity on platforms for which a colon is a valid file specification character.

Depending on your operating system, escape characters can be required if you use quotation marks when you specify a value for this parameter. Oracle recommends that you place this parameter in a parameter file, which can reduce the number of escape characters that you otherwise would require on the command line.

You must have the DATAPUMP_IMP_FULL_DATABASE role to specify this parameter.

Example

Suppose you had a parameter file, payroll.par, with the following content:

```
DIRECTORY=dpump_dir1
FULL=YES
DUMPFILE=db_full.dmp
```

```
REMAP_DATAFILE="'DB1$:[HRDATA.PAYROLL]tbs6.dbf':'/db1/hrdata/payroll/
tbs6.dbf'"
```

You can then issue the following command:

```
> impdp hr PARFILE=payroll.par
```

This example remaps a VMS file specification (`DR1$:[HRDATA.PAYROLL]tbs6.dbf`) to a Unix file specification, (`/db1/hrdata/payroll/tbs6.dbf`) for all SQL DDL statements during the import. The dump file, `db_full.dmp`, is located by the directory object, `dpump_dir1`.

Related Topics

- About Import Command-Line Mode

3.4.38 REMAP_DIRECTORY

The Oracle Data Pump Import command-line mode `REMAP_DIRECTORY` parameter lets you remap directories when you move databases between platforms.

Default

There is no default.

Purpose

The `REMAP_DIRECTORY` parameter changes the source directory string to the target directory string in all SQL statements where the source directory is the left-most portion of a full file or directory specification: `CREATE TABLESPACE`, `CREATE LIBRARY`, and `CREATE DIRECTORY`.

Syntax and Description

```
REMAP_DIRECTORY=source_directory_string:target_directory_string
```

Remapping a directory is useful when you move databases between platforms that have different directory file naming conventions. This provides an easy way to remap multiple data files in a directory when you only *want* to change the directory file specification while preserving the original data file names.

The *`source_directory_string`* and *`target_directory_string`* should be exactly as you want them to appear in the SQL statements where they are referenced. In addition, Oracle recommends that the directory be properly terminated with the directory file terminator for the respective source and target platform. Oracle recommends that you enclose the directory names in quotation marks to eliminate ambiguity on platforms for which a colon is a valid directory file specification character.

Depending on your operating system, escape characters can be required if you use quotation marks when you specify a value for this parameter. Oracle recommends that you place this parameter in a parameter file, which can reduce the number of escape characters that you otherwise would require on the command line.

You must have the `DATAPUMP_IMP_FULL_DATABASE` role to specify this parameter.

Restrictions

- The `REMAP_DIRECTORY` and `REMAP_DATAFILE` parameters are mutually exclusive.

Example

Suppose you want to remap the following data files:

```
DB1$:[HRDATA.PAYROLL]tbs5.dbf
DB1$:[HRDATA.PAYROLL]tbs6.dbf
```

In addition, you have a parameter file, `payroll.par`, with the following content:

```
DIRECTORY=dpump_dir1
FULL=YES
DUMPFILE=db_full.dmp
REMAP_DIRECTORY="'DB1$:[HRDATA.PAYROLL]':'/db1/hrdata/payroll/'"
```

You can issue the following command:

```
> impdp hr PARFILE=payroll.par
```

This example remaps the VMS file specifications (`DB1$:[HRDATA.PAYROLL]tbs5.dbf`, and `DB1$:[HRDATA.PAYROLL]tbs6.dbf`) to UNIX file specifications, (`/db1/hrdata/payroll/tbs5.dbf`, and `/db1/hrdata/payroll/tbs6.dbf`) for all SQL DDL statements during the import. The dump file, `db_full.dmp`, is located by the directory object, `dpump_dir1`.

3.4.39 REMAP_SCHEMA

The Oracle Data Pump Import command-line mode `REMAP_SCHEMA` parameter loads all objects from the source schema into a target schema.

Default

There is no default

Purpose

Loads all objects from the source schema into a target schema.

Syntax and Description

`REMAP_SCHEMA=`*`source_schema`*`:`*`target_schema`*

Multiple `REMAP_SCHEMA` lines can be specified, but the source schema must be different for each one. However, different source schemas can map to the same target schema. The mapping can be incomplete; see the Restrictions section in this topic.

If the schema you are remapping to does not exist before the import, then the import operation can create it, except in the case of `REMAP_SCHEMA` for the `SYSTEM` user. The target schema of the `REMAP_SCHEMA` must exist before the import. To create the schema, the dump file set must contain the necessary `CREATE USER` metadata for the source schema, and you must be carrying out the import with enough privileges. For example, the following Export commands create dump file sets with the necessary metadata to create a schema, because the user `SYSTEM` has the necessary privileges:

```
> expdp system SCHEMAS=hr
Password: password
```

```
> expdp system FULL=YES
Password: password
```

If your dump file set does not contain the metadata necessary to create a schema, or if you do not have privileges, then the target schema must be created before the import operation is performed. You must have the target schema created before the import, because the unprivileged dump files do not contain the necessary information for the import to create the schema automatically.

For Oracle Database releases earlier than Oracle Database 11g, if the import operation does create the schema, then after the import is complete, you must assign it a valid password to connect to it. You can then use the following SQL statement to assign the password; note that you require privileges:

```
SQL> ALTER USER schema_name IDENTIFIED BY new_password
```

In Oracle Database releases after Oracle Database 11g Release 1 (11.1.0.1), it is no longer necessary to reset the schema password; the original password remains valid.

Restrictions

- Unprivileged users can perform schema remaps only if their schema is the target schema of the remap. (Privileged users can perform unrestricted schema remaps.) For example, `SCOTT` can remap his `BLAKE`'s objects to `SCOTT`, but `SCOTT` cannot remap `SCOTT`'s objects to `BLAKE`.
- The mapping can be incomplete, because there are certain schema references that Import is not capable of finding. For example, Import does not find schema references embedded within the body of definitions of types, views, procedures, and packages.
- For triggers, `REMAP_SCHEMA` affects only the trigger owner.
- If any table in the schema being remapped contains user-defined object types, and that table changes between the time it is exported and the time you attempt to import it, then the import of that table fails. However, the import operation itself continues.
- By default, if schema objects on the source database have object identifiers (OIDs), then they are imported to the target database with those same OIDs. If an object is imported back into the same database from which it was exported, but into a different schema, then the OID of the new (imported) object is the same as that of the existing object and the import fails. For the import to succeed, you must also specify the `TRANSFORM=OID:N` parameter on the import. The transform `OID:N` causes a new OID to be created for the new object, which allows the import to succeed.

Example

Suppose that, as user `SYSTEM`, you run the following Export and Import commands to remap the `hr` schema into the `scott` schema:

```
> expdp system SCHEMAS=hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp

> impdp system DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp REMAP_SCHEMA=hr:scott
```

In this example, if user `scott` already exists before the import, then the Import `REMAP_SCHEMA` command adds objects from the `hr` schema into the existing `scott` schema. You can connect to the `scott` schema after the import by using the existing password (without resetting it).

If user `scott` does not exist before you execute the import operation, then Import automatically creates it with an unusable password. This action is possible because the dump file, `hr.dmp`, was created by `SYSTEM`, which has the privileges necessary to create a dump file that contains the metadata needed to create a schema. However, you cannot connect to `scott` on completion of the import, unless you reset the password for `scott` on the target database after the import completes.

3.4.40 REMAP_TABLE

The Oracle Data Pump Import command-line mode `REMAP_TABLE` parameter enables you to rename tables during an import operation.

Default

There is no default

Purpose

Enables you to rename tables during an import operation.

Syntax and Description

You can use either of the following syntaxes (see the Usage Notes):

```
REMAP_TABLE=[schema.]old_tablename[.partition]:new_tablename
```

OR

```
REMAP_TABLE=[schema.]old_tablename[:partition]:new_tablename
```

If the table is being departitioned, then you can use the `REMAP_TABLE` parameter to rename entire tables, or to rename table partitions (See `PARTITION_OPTIONS`).

You can also use `REMAP_TABLE` to override the automatic naming of exported table partitions.

Usage Notes

With the first syntax, if you specify `REMAP_TABLE=A.B:C`, then Import assumes that `A` is a schema name, `B` is the old table name, and `C` is the new table name. To use the first syntax to rename a partition that is being promoted to a nonpartitioned table, you must specify a schema name.

To use the second syntax to rename a partition being promoted to a nonpartitioned table, you qualify it with the old table name. No schema name is required.

Restrictions

- The `REMAP_TABLE` parameter only handles user-created tables. Data Pump does not have enough information for any dependent tables created internally. Therefore, the `REMAP_TABLE` parameter cannot remap internally created tables.
- Only objects created by the Import are remapped. In particular, pre-existing tables are not remapped.
- If the table being remapped has named constraints in the same schema, and the constraints must be created when the table is created, then `REMAP_TABLE` parameter does not work

Example

The following is an example of using the REMAP_TABLE parameter to rename the employees table to a new name of emps:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expschema.dmp
TABLES=hr.employees REMAP_TABLE=hr.employees:emps
```

Related Topics

- PARTITION_OPTIONS

3.4.41 REMAP_TABLESPACE

The Oracle Data Pump Import command-line mode REMAP_TABLESPACE parameter remaps all objects selected for import with persistent data in the source tablespace to be created in the target tablespace.

Default

There is no default

Purpose

Remaps all objects selected for import with persistent data in the source tablespace to be created in the target tablespace.

Syntax and Description

```
REMAP_TABLESPACE=source_tablespace:target_tablespace
```

Multiple REMAP_TABLESPACE parameters can be specified, but no two can have the same source tablespace. The target schema must have sufficient quota in the target tablespace.

The Data Pump Import method of using the REMAP_TABLESPACE parameter works for all objects, including the CREATE USER statement.

With Oracle Database 19c and later releases, the % wildcard can used in place of the source tablespaces for the REMAP_TABLESPACE parameter. When you specify the source database using the % wildcard, Oracle Data Pump combines the tablespaces from the source database export dumpfile into a target permanent tablespace. It is applied to the object types USER, TABLE, INDEX, MVIEW, MVIEW_LOG, MVIEW_ZONEMAP, and CLUSTERS.

A production database can have multiple tablespaces. You may want to consolidate those tablespaces during migration into a particular target tablespace. For example, you may want to consolidate tablespaces when migrating to Oracle Autonomous Database where only the USER tablespace is available for applications. Using this parameter with the % wildcard makes it easy to do so without specifying all of the source tablespaces.

The % wildcard can also replace the source tablespace specified in place of 'TBS_OLD' in this API example: DBMS_METADATA.SET_REMAP_PARAM(*handle*, 'REMAP_TABLESPACE','TBS_OLD','TBS_NEW','*object-type*');

Restrictions

- Oracle Data Pump Import can only remap tablespaces for transportable imports in databases where the compatibility level is set to 10.1 or later.

- Only objects created by the Import are remapped. In particular, if TABLE_EXISTS_ACTION is set to SKIP, TRUNCATE, or APPEND, then the tablespaces for pre-existing tables are not remapped.
- You cannot use REMAP_TABLESPACE with domain indexes to exclude the storage clause of the source metadata. If you customized the tablespace using storage clauses, then REMAP_TABLESPACE does not apply to those storage clauses. If you used a default tablespace without storage clauses, then REMAP_TABLESPACE should work for that tablespace.
- If the index preferences have customized tablespaces in the storage clauses at the source table, then you must recreate those customized tablespaces on the target before attempting to import those tablespaces. If you do not recreate the customized tablespaces on the target database, then the Text index rebuild will fail.
- The target tablespace must be a permanent tablespace, and it must exist before the import.
- The target tablespace cannot be a temporary tablespace.
- The % wildcard cannot be used with multiple REMAP_TABLESPACE parameters.
- The REMAP_TABLESPACE parameter and TRANSFORM=TABLESPACE:N transform parameter are mutually exclusive.

Example

The following is an example of using the REMAP_TABLESPACE parameter.

```
> impdp hr REMAP_TABLESPACE=tbs_1:tbs_6 DIRECTORY=dpump_dir1
  DUMPFILE=employees.dmp
```

3.4.42 SCHEMAS

The Oracle Data Pump Import command-line mode SCHEMAS parameter specifies that you want a schema-mode import to be performed.

Default

There is no default

Purpose

Specifies that you want a schema-mode import to be performed.

Syntax and Description

```
SCHEMAS=schema_name [,...]
```

If you have the DATAPUMP_IMP_FULL_DATABASE role, then you can use this parameter to perform a schema-mode import by specifying a list of schemas to import. First, the user definitions are imported (if they do not already exist), including system and role grants, password history, and so on. Then all objects contained within the schemas are imported. Unprivileged users can specify only their own schemas, or schemas remapped to their own schemas. In that case, no information about the schema definition is imported, only the objects contained within it.

To restrict what is imported by using this import mode, you can use filtering.

Schema mode is the default mode when you are performing a network-based import.

Example

The following is an example of using the SCHEMAS parameter. You can create the expdat.dmp file used in this example by running the example provided for the Export SCHEMAS parameter.

```
> impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 LOGFILE=schemas.log
DUMPFILE=expdat.dmp
```

The hr schema is imported from the expdat.dmp file. The log file, schemas.log, is written to dpump_dir1.

Related Topics

- Filtering During Import Operations
- SCHEMAS

3.4.43 SERVICE_NAME

The Oracle Data Pump Import command-line mode SERVICE_NAME parameter specifies a service name that you want to use in conjunction with the CLUSTER parameter.

Default

There is no default

Purpose

Used to specify a service name to be used with the CLUSTER parameter.

Syntax and Description

```
SERVICE_NAME=name
```

The SERVICE_NAME parameter can be used with the CLUSTER=YES parameter to specify an existing service associated with a resource group that defines a set of Oracle Real Application Clusters (Oracle RAC) instances belonging to that resource group, typically a subset of all the Oracle RAC instances.

The service name is only used to determine the resource group and instances defined for that resource group. The instance where the job is started is always used, regardless of whether it is part of the resource group.

The SERVICE_NAME parameter is ignored whenCLUSTER=NO is also specified.

Suppose you have an Oracle RAC configuration containing instances A, B, C, and D. Also suppose that a service named my_service exists with a resource group consisting of instances A, B, and C only. In such a scenario, the following would be true:

- If you start an Oracle Data Pump job on instance A, and specify CLUSTER=YES (or accept the default, which is YES), and you do not specify the SERVICE_NAME parameter, then Oracle Data Pump creates workers on all instances: A, B, C, and D, depending on the degree of parallelism specified.
- If you start an Oracle Data Pump job on instance A, and specify CLUSTER=YES and SERVICE_NAME=my_service, then workers can be started on instances A, B, and C only.

- If you start an Oracle Data Pump job on instance D, and specify `CLUSTER=YES` and `SERVICE_NAME=my_service`, then workers can be started on instances A, B, C, and D. Even though instance D is not in `my_service` it is included because it is the instance on which the job was started.
- If you start an Oracle Data Pump job on instance A, and specify `CLUSTER=NO`, then any `SERVICE_NAME` parameter that you specify is ignored, and all processes start on instance A.

Example

```
> impdp system DIRECTORY=dpump_dir1 SCHEMAS=hr
  SERVICE_NAME=sales NETWORK_LINK=dbs1
```

This example starts a schema-mode network import of the `hr` schema. Even though `CLUSTER=YES` is not specified on the command line, it is the default behavior, so the job uses all instances in the resource group associated with the service name `sales`. The `NETWORK_LINK` value of `dbs1` is replaced with the name of the source database from which you are importing data. (Note that there is no dump file generated with a network import.)

The `NETWORK_LINK` parameter is simply being used as part of the example. It is not required when using the `SERVICE_NAME` parameter.

Related Topics

- CLUSTER

3.4.44 SKIP_UNUSABLE_INDEXES

The Oracle Data Pump Import command-line mode `SKIP_UNUSABLE_INDEXES` parameter specifies whether Import skips loading tables that have indexes that were set to the Index Unusable state (by either the system or the user).

Default

The value of the Oracle Database configuration parameter, `SKIP_UNUSABLE_INDEXES`.

Purpose

Specifies whether Import skips loading tables that have indexes that were set to the Index Unusable state (by either the system or the user).

Syntax and Description

```
SKIP_UNUSABLE_INDEXES=[YES | NO]
```

If `SKIP_UNUSABLE_INDEXES` is set to `YES`, and a table or partition with an index in the Unusable state is encountered, then the load of that table or partition proceeds anyway, as if the unusable index did not exist.

If `SKIP_UNUSABLE_INDEXES` is set to `NO`, and a table or partition with an index in the Unusable state is encountered, then that table or partition is not loaded. Other tables, with indexes not previously set Unusable, continue to be updated as rows are inserted.

If the `SKIP_UNUSABLE_INDEXES` parameter is not specified, then the setting of the Oracle Database configuration parameter, `SKIP_UNUSABLE_INDEXES` is used to determine how to handle unusable indexes. The default value for that parameter is `y`).

If indexes used to enforce constraints are marked unusable, then the data is not imported into that table.

> **Note:**
>
> `SKIP_UNUSABLE_INDEXES` is useful only when importing data into an existing table. It has no practical effect when a table is created as part of an import. In that case, the table and indexes are newly created, and are not marked unusable.

Example

The following is an example of using the `SKIP_UNUSABLE_INDEXES` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp LOGFILE=skip.log
SKIP_UNUSABLE_INDEXES=YES
```

Related Topics

- FULL

3.4.45 SOURCE_EDITION

The Oracle Data Pump Import command-line mode `SOURCE_EDITION` parameter specifies the database edition on the remote node from which objects are fetched.

Default

The default database edition on the remote node from which objects are fetched.

Purpose

Specifies the database edition on the remote node from which objects are e fetched.

Syntax and Description

`SOURCE_EDITION=`*`edition_name`*

If `SOURCE_EDITION=`*`edition_name`* is specified, then the objects from that edition are imported. Oracle Data Pump selects all inherited objects that have not changed, and all actual objects that have changed.

If this parameter is not specified, then the default edition is used. If the specified edition does not exist or is not usable, then an error message is returned.

Restrictions

- The `SOURCE_EDITION` parameter is valid on an import operation only when the `NETWORK_LINK` parameter is also specified.
- This parameter is only useful if there are two or more versions of the same versionable objects in the database.
- The job version must be set to 11.2 or later.

Example

The following is an example of using the import SOURCE_EDITION parameter:

```
> impdp hr DIRECTORY=dpump_dir1 SOURCE_EDITION=exp_edition
NETWORK_LINK=source_database_link EXCLUDE=USER
```

In this example, we assume the existence of an edition named exp_edition on the system from which objects are being imported. Because no import mode is specified, the default, which is schema mode, is used. Replace *source_database_link* with the name of the source database from which you are importing data. The EXCLUDE=USER parameter excludes only the definitions of users, not the objects contained within user schemas. No dump file is generated, because this is a network import.

Related Topics

- NETWORK_LINK
- VERSION

> **See Also:**
>
> - CREATE EDITION in *Oracle Database SQL Language Reference* for information about how editions are created
> - Editions in *Oracle Database Development Guide* for more information about the editions feature, including inherited and actual objects

3.4.46 SQLFILE

The Oracle Data Pump Import command-line mode SQLFILE parameter specifies a file into which all the SQL DDL that Import prepares to execute is written, based on other Import parameters selected.

Default

There is no default

Purpose

Specifies a file into which all the SQL DDL that Import prepares to execute is written, based on other Import parameters selected.

Syntax and Description

```
SQLFILE=[directory_object:]file_name
```

The *file_name* specifies where the import job writes the DDL that is prepared to run during the job. The SQL is not actually run, and the target system remains unchanged. The file is written to the directory object specified in the DIRECTORY parameter, unless you explicitly specify another directory object.

> **Note:**
>
> If an existing file that has a name matching the one specified with this parameter, it is overwritten only if the existing file extension is one of the following: `sql`, `SQL`, `log`, `LOG`, `lst`, or `LST`. If the existing file extension does not match one of these extensions, then you receive the message `ORA-02604: 'file already exists'`. However, if no existing file with a matching name is found, then there is no file extension restriction.

Note that passwords are not included in the SQL file. For example, if a `CONNECT` statement is part of the DDL that was run, then it is replaced by a comment with only the schema name shown. In the following example, the dashes (`--`) indicate that a comment follows. The `hr` schema name is shown, but not the password.

```
-- CONNECT hr
```

Therefore, before you can run the SQL file, you must edit it by removing the dashes indicating a comment, and adding the password for the `hr` schema.

Oracle Data Pump places any `ALTER SESSION` statements at the top of the SQL file created by the Oracle Data Pump import. If the import operation has different connection statements, then you must manually copy each of the `ALTER SESSION` statements, and paste them after the appropriate `CONNECT` statements.

For some Oracle Database options, anonymous PL/SQL blocks can appear within the `SQLFILE` output. Do not run these PL/SQL blocks directly.

Restrictions

- If `SQLFILE` is specified, then the `CONTENT` parameter is ignored if it is set to either `ALL` or `DATA_ONLY`.
- To perform an Oracle Data Pump Import to a SQL file using Oracle Automatic Storage Management (Oracle ASM), the `SQLFILE` parameter that you specify must include a directory object that does not use the Oracle `ASM +` notation. That is, the SQL file must be written to a disk file, not into the Oracle ASM storage.
- You cannot use the `SQLFILE` parameter in conjunction with the `QUERY` parameter.
- When you specify the same filename, the `SQLFILE` filename you provide must have a file extension (`SQL`, `sql`, `LOG`, `log`, `LST`, `lst`). The file name you provide cannot have multiple dots in the filename (specifically, a compound suffix). Compound suffixes are not supported.

Example

The following is an example of using the `SQLFILE` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
SQLFILE=dpump_dir2:expfull.sql
```

A SQL file named `expfull.sql` is written to `dpump_dir2`.

Related Topics

- FULL

3.4.47 STATUS

The Oracle Data Pump Import command-line mode `STATUS` parameter specifies the frequency at which the job status is displayed.

Default

`0`

Purpose

Specifies the frequency at which the job status is displayed.

Syntax and Description

`STATUS[=integer]`

If you supply a value for *integer*, then it specifies how frequently, in seconds, job status should be displayed in logging mode. If no value is entered, or if the default value of 0 is used, then no additional information is displayed beyond information about the completion of each object type, table, or partition.

This status information is written only to your standard output device, not to the log file (if one is in effect).

Example

The following is an example of using the `STATUS` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter..

```
> impdp hr NOLOGFILE=YES STATUS=120 DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
```

In this example, the status is shown every two minutes (120 seconds).

Related Topics

- FULL

3.4.48 STREAMS_CONFIGURATION

The Oracle Data Pump Import command-line mode `STREAMS_CONFIGURATION` parameter specifies whether to import any GoldenGate Replication metadata that may be present in the export dump file.

Default

`YES`

Purpose

Specifies whether to import any GoldenGate Replication metadata that can be present in the export dump file.

Syntax and Description

```
STREAMS_CONFIGURATION=[YES | NO]
```

Example

The following is an example of using the STREAMS_CONFIGURATION parameter. You can create the expfull.dmp dump file used in this example by running the example provided for the Export FULL parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp STREAMS_CONFIGURATION=NO
```

3.4.49 TABLE_EXISTS_ACTION

The Oracle Data Pump Import command-line mode TABLE_EXISTS_ACTION parameter specifies for Import what to do if the table it is trying to create already exists.

Default

```
SKIP
```

> **Note:**
>
> If CONTENT=DATA_ONLY is specified, then the default is APPEND, not SKIP.

Purpose

Specifies for Import what to do if the table it is trying to create already exists.

Syntax and Description

```
TABLE_EXISTS_ACTION=[SKIP | APPEND | TRUNCATE | REPLACE]
```

The possible values have the following effects:

- SKIP leaves the table as is, and moves on to the next object. This option is not valid when the CONTENT parameter is set to DATA_ONLY.
- APPEND loads rows from the source and leaves existing rows unchanged.
- TRUNCATE deletes existing rows and then loads rows from the source.
- REPLACE drops the existing table, and then creates and loads it from the source. This option is not valid when the CONTENT parameter is set to DATA_ONLY.

When you are using these options, be aware of the following:

- When you use TRUNCATE or REPLACE, ensure that rows in the affected tables are not targets of any referential constraints.
- When you use SKIP, APPEND, or TRUNCATE, existing table-dependent objects in the source, such as indexes, grants, triggers, and constraints, are not modified. For REPLACE, the dependent objects are dropped and recreated from the source, if they are not explicitly or implicitly excluded (using EXCLUDE) and if they exist in the source dump file or system.
- When you use APPEND or TRUNCATE, Import checks that rows from the source are compatible with the existing table before performing any action.

If the existing table has active constraints and triggers, then it is loaded using the external tables access method. If any row violates an active constraint, then the load fails and no data is loaded. You can override this behavior by specifying `DATA_OPTIONS=SKIP_CONSTRAINT_ERRORS` on the Import command line.

If you have data that must be loaded, but that can cause constraint violations, then consider disabling the constraints, loading the data, and then deleting the problem rows before re-enabling the constraints.

- When you use `APPEND`, the data is always loaded into new space; existing space, even if available, is not reused. For this reason, you may want to compress your data after the load.
- If you use parallel processing, then review the description of the Import `PARTITION_OPTIONS` parameter for information about how parallel processing of partitioned tables is affected, depending on whether the target table already exists or not.
- If you are importing into an existing table (`TABLE_EXISTS_ACTION=REPLACE` or `TRUNCATE`), then follow these guidelines, depending on the table partitioning scheme:
  - If the partitioning scheme matches between the source and target, then use `DATA_OPTIONS=TRUST_EXISTING_TABLE_PARTITIONS` on import.
  - If the partitioning scheme differs between source and target, then use `DATA_OPTIONS=GROUP_PARTITION_TABLE_DATA` on export.

> **Note:**
>
> When Oracle Data Pump detects that the source table and target table do not match (the two tables do not have the same number of columns or the target table has a column name that is not present in the source table), it then compares column names between the two tables. If the tables have at least one column in common, then the data for the common columns is imported into the table (assuming the data types are compatible). The following restrictions apply:
>
> - This behavior is not supported for network imports.
> - The following types of columns cannot be dropped: object columns, object attributes, nested table columns, and ref columns based on a primary key.

Restrictions

- `TRUNCATE` cannot be used on clustered tables.

Example

The following is an example of using the `TABLE_EXISTS_ACTION` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr TABLES=employees DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
TABLE_EXISTS_ACTION=REPLACE
```

Related Topics

- PARTITION_OPTIONS
- FULL

3.4.50 REUSE_DATAFILES

The Oracle Data Pump Import command-line mode `REUSE_DATAFILES` parameter specifies whether you want the import job to reuse existing data files for tablespace creation.

Default

`NO`

Purpose

Specifies whether you want the import job to reuse existing data files for tablespace creation.

Syntax and Description

```
REUSE_DATAFILES=[YES | NO]
```

If you use the default (`n`), and the data files specified in `CREATE TABLESPACE` statements already exist, then an error message from the failing `CREATE TABLESPACE` statement is issued, but the import job continues.

If this parameter is specified as `y`, then the existing data files are reinitialized.

> **Caution:**
>
> Specifying `REUSE_DATAFILES=YES` can result in a loss of data.

Example

The following is an example of using the `REUSE_DATAFILES` parameter. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp LOGFILE=reuse.log
REUSE_DATAFILES=YES
```

This example reinitializes data files referenced by `CREATE TABLESPACE` statements in the `expfull.dmp` file.

Related Topics

- FULL

3.4.51 TABLES

The Oracle Data Pump Import command-line mode `TABLES` parameter specifies that you want to perform a table-mode import.

Default

There is no default.

Purpose

Specifies that you want to perform a table-mode import.

Syntax and Description

```
TABLES=[schema_name.]table_name[:partition_name]
```

In a table-mode import, you can filter the data that is imported from the source by specifying a comma-delimited list of tables and partitions or subpartitions.

If you do not supply a *schema_name*, then it defaults to that of the current user. To specify a schema other than your own, you must either have the DATAPUMP_IMP_FULL_DATABASE role or remap the schema to the current user.

If you want to restrict what is imported, you can use filtering with this import mode.

If you specify *partition_name*, then it must be the name of a partition or subpartition in the associated table.

You can specify table names and partition names by using the wildcard character %.

The following restrictions apply to table names:

- By default, table names in a database are stored as uppercase characters. If you have a table name in mixed-case or lowercase characters, and you want to preserve case sensitivity for the table name, then you must enclose the name in quotation marks. The name must exactly match the table name stored in the database.

  Some operating systems require that quotation marks on the command line be preceded by an escape character. The following are examples of how case-sensitivity can be preserved in the different Import modes.

  – In command-line mode:

    ```
    TABLES='\"Emp\"'
    ```

  – In parameter file mode:

    ```
    TABLES='"Emp"'
    ```

- Table names specified on the command line cannot include a pound sign (#), unless the table name is enclosed in quotation marks. Similarly, in the parameter file, if a table name includes a pound sign (#), then unless the table name is enclosed in quotation marks, the Import utility interprets the rest of the line as a comment.

  For example, if the parameter file contains the following line, then Import interprets everything on the line after emp# as a comment, and does not import the tables dept and mydata:

  ```
  TABLES=(emp#, dept, mydata)
  ```

  However, if the parameter file contains the following line, then the Import utility imports all three tables because emp# is enclosed in quotation marks:

  ```
  TABLES=('"emp#"', dept, mydata)
  ```

> **Note:**
>
> Some operating systems require single quotation marks rather than double quotation marks, or the reverse; see your operating system documentation. Different operating systems also have other restrictions on table naming.
>
> For example, the Unix C shell attaches a special meaning to a dollar sign ($) or pound sign (#), or certain other special characters. You must use escape characters to use these special characters in the names so that the operating system shell ignores them, and they can be used with Import.

Restrictions

- The use of synonyms as values for the `TABLES` parameter is not supported. For example, if the `regions` table in the `hr` schema had a synonym of `regn`, then it would not be valid to use `TABLES=regn`. An error would be returned.
- You can only specify partitions from one table if `PARTITION_OPTIONS=DEPARTITION` is also specified on the import.
- If you specify `TRANSPORTABLE=ALWAYS`, then all partitions specified on the `TABLES` parameter must be in the same table.
- The length of the table name list specified for the `TABLES` parameter is limited to a maximum of 4 MB, unless you are using the `NETWORK_LINK` parameter to an Oracle Database release 10.2.0.3 or earlier or to a read-only database. In such cases, the limit is 4 KB.

Example

The following example shows a simple use of the `TABLES` parameter to import only the `employees` and `jobs` tables from the `expfull.dmp` file. You can create the `expfull.dmp` dump file used in this example by running the example provided for the Export `FULL` parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp TABLES=employees,jobs
```

The following example is a command to import partitions using the `TABLES`:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp
TABLES=sh.sales:sales_Q1_2012,sh.sales:sales_Q2_2012
```

This example imports the partitions `sales_Q1_2012` and `sales_Q2_2012` for the table `sales` in the schema `sh`.

Related Topics

- Filtering During Import Operations
- FULL

3.4.52 TABLESPACES

The Oracle Data Pump Import command-line mode TABLESPACES parameter specifies that you want to perform a tablespace-mode import.

Default

There is no default

Purpose

Specifies that you want to perform a tablespace-mode import.

Syntax and Description

```
TABLESPACES=tablespace_name [, ...]
```

Use TABLESPACES to specify a list of tablespace names whose tables and dependent objects are to be imported from the source (full, schema, tablespace, or table-mode export dump file set or another database).

During the following import situations, Data Pump automatically creates the tablespaces into which the data will be imported:

- The import is being done in FULL or TRANSPORT_TABLESPACES mode
- The import is being done in table mode with TRANSPORTABLE=ALWAYS

In all other cases, the tablespaces for the selected objects must already exist on the import database. You could also use the Import REMAP_TABLESPACE parameter to map the tablespace name to an existing tablespace on the import database.

If you want to restrict what is imported, you can use filtering with this import mode.

Restrictions

- The length of the list of tablespace names specified for the TABLESPACES parameter is limited to a maximum of 4 MB, unless you are using the NETWORK_LINK parameter to a 10.2.0.3 or earlier database or to a read-only database. In such cases, the limit is 4 KB.

Example

The following is an example of using the TABLESPACES parameter. It assumes that the tablespaces already exist. You can create the expfull.dmp dump file used in this example by running the example provided for the Export FULL parameter.

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=expfull.dmp
TABLESPACES=tbs_1,tbs_2,tbs_3,tbs_4
```

This example imports all tables that have data in tablespaces tbs_1, tbs_2, tbs_3, and tbs_4.

Related Topics

- Filtering During Import Operations
- FULL

3.4.53 TARGET_EDITION

The Oracle Data Pump Import command-line mode `TARGET_EDITION` parameter specifies the database edition into which you want objects imported.

Default

The default database edition on the system.

Purpose

Specifies the database edition into which you want objects imported.

Syntax and Description

```
TARGET_EDITION=name
```

If you specify `TARGET_EDITION=name`, then Data Pump Import creates all of the objects found in the dump file. Objects that are not editionable are created in all editions.

For example, tables are not editionable, so if there is a table in the dump file, then the table is created, and all editions see it. Objects in the dump file that are editionable, such as procedures, are created only in the specified target edition.

If this parameter is not specified, then Import uses the default edition on the target database, even if an edition was specified in the export job. If the specified edition does not exist, or is not usable, then an error message is returned.

Restrictions

- This parameter is only useful if there are two or more versions of the same versionable objects in the database.
- The job version must be 11.2 or later.

Example

The following is an example of using the `TARGET_EDITION` parameter:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=exp_dat.dmp
TARGET_EDITION=exp_edition
```

This example assumes the existence of an edition named `exp_edition` on the system to which objects are being imported. Because no import mode is specified, the default of schema mode will be used.

See *Oracle Database SQL Language Reference* for information about how editions are created. See *Oracle Database Development Guide* for more information about the editions features.

Related Topics

- VERSION
- CREATE EDITION in *Oracle Database SQL Language Reference*
- Editions in *Oracle Database Development Guide*

3.4.54 TRANSFORM

The Oracle Data Pump Import command-line mode `TRANSFORM` parameter enables you to alter object creation DDL for objects being imported.

Default

The default value for this parameter is N and constraints are validated during import.

Purpose

Enables you to alter object creation DDL for objects being imported.

Syntax and Description

```
TRANSFORM = transform_name:value[:object_type]
```

The *transform_name* specifies the name of the transform.

Specifying *object_type* is optional. If supplied, this parameter designates the object type to which the transform is applied. If no object type is specified, then the transform applies to all valid object types.

The available transforms are as follows, in alphabetical order:

- `CONSTRAINT_NAME_FROM_INDEX:[Y|N]`

  This transform is valid for the following object types:`TABLE` and `CONSTRAINT` object types.

  This transform parameter affects the generation of the `pk` or `fk` constraint which reference user created indexes. If set to `Y`, then it forces the name of the constraint to match the name of the index.

  If set to `N` (the default), then the constraint is created as named on the source database.

- `CONSTRAINT_NOVALIDATE:[Y|N]`

  The default value for this parameter is `N`. If the parameter is set to `Y`, then constraints are not validated during import. Validating constraints during import that were valid on the source can be unnecessary and slow the migration process. Validation can be done after import.

  You cannot choose `CONSTRAINT_NOVALIDATE = Y` for tables with the following properties because these constraints must be in the `VALIDATE` state to complete the import:

  – Reference partitioned table
  – Reference partitioned child table
  – Table with Primary key OID
  – Table is clustered

- `CONSTRAINT_USE_DEFAULT_INDEX:[Y|N ]`

  This transform is valid for the following object types:`TABLE` and `CONSTRAINT` object types.

  This transform parameter affects the generation of index relating to the `pk` or `fk` constraint. If set to `Y`, then the transform parameter forces the name of an index automatically created to enforce the constraint to be identical to the constraint name. In addition, the index is

created using the default constraint definition for the target database, and will not use any special characteristics that might have been defined in the source database.

Default Indexes are not allowed unless they use standard schema integrity constraints, such as `UNIQUE`, `PRIMARY KEY`, or `FOREIGN KEY`. Accordingly, if you run an Oracle Data Pump import from a system where no restrictions exist, and you have additional constraints in the source index (for example, user generated constraints, such as a hash-partitioned index), then these additional constraints are removed during the import.

If set to `N` (the default), then the index is created as named on the source database.

- `DISABLE_ARCHIVE_LOGGING:[Y|N]`

  This transform is valid for the following object types: `INDEX` and `TABLE`.

  If set to `Y`, then the logging attributes for the specified object types (`TABLE` and/or `INDEX`) are disabled before the data is imported. If set to `N` (the default), then archive logging is not disabled during import. After the data has been loaded, the logging attributes for the objects are restored to their original settings. If no object type is specified, then the `DISABLE_ARCHIVE_LOGGING` behavior is applied to both `TABLE` and `INDEX` object types. This transform works for both file mode imports and network mode imports. It does not apply to transportable tablespace imports.

  > **Note:**
  >
  > If the database is in `FORCE LOGGING` mode, then the `DISABLE_ARCHIVE_LOGGING` option does not disable logging when indexes and tables are created.

- `DWCS_CVT_IOTS:[Y|N]`

  This transform is valid for `TABLE` object types.

  If set to `Y`, it directs Oracle Data Pump to transform Index Organized tables to heap organized tables by suppressing the `ORGANIZATION INDEX` clause when creating the table.

  If set to `N` (the default), the generated DDL retains the table characteristics of the source object.

- `DWCS_CVT_CONSTRAINTS:[Y|N]`

  This transform is valid for the following object types:`TABLE` and `CONSTRAINT` object types.

  If set to `Y`, it directs Oracle Data Pump to create `pk`, `fk`, or `uk` constraints as disabled.

  If set to `N` (the default), it directs Oracle Data Pump to create`pk`, `fk`, or `uk` constraints based on the source database status.

- `INDEX_COMPRESSION_CLAUSE [NONE | compression_clause]`

  This transform is valid for the object type `INDEX`. As with `TABLE_COMPRESSION_CLAUSE`, using `INDEX_COMPRESSION_CLAUSE` enables you to control index compression on import.

  If `NONE` is specified, then the index compression clause is omitted (and the index is given the default compression for the tablespace). However, if you use compression, then Oracle recommends that you use `COMPRESS ADVANCED LOW`). Indexes are created with the specified compression. See *Oracle Database SQL Language Reference* for information about valid table compression syntax.

If the index compression clause is more than one word, then it must be contained in single or double quotation marks. Also, your operating system can require you to enclose the clause in escape characters, such as the backslash character. For example:

```
TRANSFORM=INDEX_COMPRESSION_CLAUSE:\"COMPRESS ADVANCED LOW\"
```

Specifying this transform changes the type of compression for all indexes in the job.

- `INCLUDE_SHARDING_CLAUSES:[Y|N]`

  The default for this transform is `N`. When set to `Y`, `get_ddl()` generates shard syntax, if the dictionary values in the imported document contain the shard syntax.

- `INMEMORY:[Y|N]`

  This transform is valid for the following object types: `TABLE` and `TABLESPACE`

  The `INMEMORY` transform is related to the In-Memory Column Store (IM column store). The IM column store is an optional portion of the system global area (SGA) that stores copies of tables, table partitions, and other database objects. In the IM column store, data is populated by column rather than row as it is in other parts of the SGA, and data is optimized for rapid scans. The IM column store does not replace the buffer cache, but acts as a supplement so that both memory areas can store the same data in different formats. The IM column store is included with the Oracle Database In-Memory option.

  If `Y` (the default value) is specified on import, then Data Pump keeps the IM column store clause for all objects that have one. When those objects are recreated at import time, Data Pump generates the IM column store clause that matches the setting for those objects at export time.

  If `N` is specified on import, then Data Pump drops the IM column store clause from all objects that have one. If there is no IM column store clause for an object that is stored in a tablespace, then the object inherits the IM column store clause from the tablespace. So if you are migrating a database, and you want the new database to use IM column store features, then you can pre-create the tablespaces with the appropriate IM column store clause and then use `TRANSFORM=INMEMORY:N` on the import command. The object then inherits the IM column store clause from the new pre-created tablespace.

  If you do not use the `INMEMORY` transform, then you must individually alter every object to add the appropriate IM column store clause.

  > **Note:**
  >
  > The `INMEMORY` transform is available only in Oracle Database 12c Release 1 (12.1.0.2) or later releases.
  >
  > See *Oracle Database Administrator's Guide* for information about using the In-Memory Column Store (IM column store).

- `INMEMORY_CLAUSE:"string with a valid in-memory parameter"`

  This transform is valid for the following object types: `TABLE` and `TABLESPACE`.

  The `INMEMORY_CLAUSE` transform is related to the In-Memory Column Store (IM column store). The IM column store is an optional portion of the system global area (SGA) that stores copies of tables, table partitions, and other database objects. In the IM column store, data is populated by column rather than row as it is in other parts of the SGA, and data is optimized for rapid scans. The IM column store does not replace the buffer cache,

but acts as a supplement so that both memory areas can store the same data in different formats. The IM column store is included with the Oracle Database In-Memory option.

When you specify this transform, Data Pump uses the contents of the string as the `INMEMORY_CLAUSE` for all objects being imported that have an IM column store clause in their DDL. This transform is useful when you want to override the IM column store clause for an object in the dump file.

The string that you supply must be enclosed in double quotation marks. If you are entering the command on the command line, be aware that some operating systems can strip out the quotation marks during parsing of the command, which causes an error. You can avoid this error by using backslash escape characters (\). For example:

```
transform=inmemory_clause:\"INMEMORY MEMCOMPRESS FOR DML PRIORITY
CRITICAL\"
```

Alternatively you can put parameters in a parameter file. Quotation marks in the parameter file are maintained during processing.

> **Note:**
>
> The `INMEMORY_CLAUSE` transform is available only with Oracle Database 12c Release 1 (12.1.0.2) or later releases.
>
> See *Oracle Database Administrator's Guide* for information about using the In-Memory Column Store (IM column store). See *Oracle Database Reference* for a listing and description of parameters that can be specified in an IM column store clause

- `LOB_STORAGE:[SECUREFILE | BASICFILE | DEFAULT | NO_CHANGE]`

  This transform is valid for the object type `TABLE`.

  LOB segments are created with the storage data type that you specify, either `SECUREFILE` or `BASICFILE`. (Note that Oracle recommends that you migrate all legacy binary data types to SecureFile LOBs.) If the value is `NO_CHANGE` (the default), then the LOB segments are created with the same storage that they had in the source database. If the value is `DEFAULT`, then the keyword (`SECUREFILE` or `BASICFILE`) is omitted, and the LOB segment is created with the default storage.

  Specifying this transform changes LOB storage for all tables in the job, including tables that provide storage for materialized views.

  The `LOB_STORAGE` transform is not valid in transportable import jobs.

- `LONG_TO_LOB:[Y|N]`

  The default value is `N`. This parameter changes all `LONG` data types to `CLOB` and all `LONG RAW` data types to `BLOB`. Using this transform enables you to migrate deprecated `LONG` and `LONG RAW` data types, transparently and automatically converting them to `CLOB`s and `BLOB`s.

- `OID:[Y|N]`

  This transform is valid for the following object types: `INC_TYPE`, `TABLE`, and `TYPE`.

  If `Y` (the default value) is specified on import, then the exported OIDs are assigned to new object tables and types. Data Pump also performs OID checking when looking for an existing matching type on the target database.

If N is specified on import, then:

  - The assignment of the exported OID during the creation of new object tables and types is inhibited. Instead, a new OID is assigned. Inhibiting assignment of exported OIDs can be useful for cloning schemas, but does not affect referenced objects.
  - Before loading data for a table associated with a type, Data Pump skips normal type OID checking when looking for an existing matching type on the target database. Other checks using a hash code for a type, version number, and type name are still performed.

- OMIT_ACDR_METADATA:[Y|N]

  The default value is N. When set to Y (true), Oracle Data Pump Import excludes invisible columns from importing replicated tables deletes tombstone tables, and deletes all the automatic conflict detection and resolution (ACDR) instance procedural actions.

- OMIT_ENCRYPTION_CLAUSE:[Y|N]

  This transform is valid for TABLE object types.

  If set to Y, it directs Oracle Data Pump to suppress column encryption clauses. Columns which were encrypted in the source database are not encrypted in imported tables.

  If set to N (the default), it directs Oracle Data Pump to create column encryption clauses, as in the source database.

- PCTSPACE:*some_number_greater_than_zero*

  This transform is valid for the following object types: CLUSTER, CONSTRAINT, INDEX, ROLLBACK_SEGMENT, TABLE, and TABLESPACE.

  The value supplied for this transform must be a number greater than zero. It represents the percentage multiplier used to alter extent allocations and the size of data files.

  You can use the PCTSPACE transform with the Data Pump Export SAMPLE parameter so that the size of storage allocations matches the sampled data subset. (See the SAMPLE export parameter.)

- SEGMENT_ATTRIBUTES:[Y|N]

  This transform is valid for the following object types: CLUSTER, CONSTRAINT, INDEX, ROLLBACK_SEGMENT, TABLE, and TABLESPACE.

  If the value is specified as Y, then segment attributes (physical attributes, storage attributes, tablespaces, and logging) are included, with appropriate DDL. The default is Y.

- SEGMENT_CREATION:[Y|N]

  This transform is valid for the object type TABLE.

  If set to Y (the default), then this transform causes the SQL SEGMENT CREATION clause to be added to the CREATE TABLE statement. That is, the CREATE TABLE statement explicitly says either SEGMENT CREATION DEFERRED or SEGMENT CREATION IMMEDIATE. If the value is N, then the SEGMENT CREATION clause is omitted from the CREATE TABLE statement. Set this parameter to N to use the default segment creation attributes for the tables being loaded. This functionality is available with Oracle Database 11g release 2 (11.2.0.2) and later releases.

- STORAGE:[Y|N]

  This transform is valid for the following object types: CLUSTER, CONSTRAINT, INDEX, ROLLBACK_SEGMENT, and TABLE.

If the value is specified as `Y`, then the storage clauses are included, with appropriate DDL. The default is `Y`. This parameter is ignored if `SEGMENT_ATTRIBUTES=N`.

- `TABLESPACE:[Y|N]`

  The `TABLESPACE` option for the `TRANSFORM` parameter `(transform=TABLESPACE:[Y|N])`, allows you to associate many source tablespaces with the user default tablespace for the target database.

  If you set the `TABLESPACE` parameter to No during import, then the tablespace space clause is omitted from the DDL for creating the object types `TABLE`, `INDEX`, `CONSTRAINT`, `CLUSTER`, `MATERIALIZED VIEW`, `MATERIALIZED VIEW LOG`, and `MATERIALIZED ZONEMAP`. If you set the `TABLESPACE` parameter to Yes (the default), then the tablespace space clause is emitted for these object types. A production database can have multiple tablespaces. You may want to consolidate those tablespaces during migration into a particular target tablespace. For example, you may want to consolidate tablespaces when migrating to Oracle Autonomous Database where only the `USER` tablespace is available for applications. Using this parameter with the `%` wildcard makes it easy to do so without specifying all of the source tablespaces.

- `TABLE_COMPRESSION_CLAUSE:[NONE|`*`compression_clause`*`]`

  This transform is valid for the object type `TABLE`.

  If `NONE` is specified, then the table compression clause is omitted (and the table is given the default compression for the tablespace). Otherwise, the value is a valid table compression clause (for example, `NOCOMPRESS`, `COMPRESS BASIC`, and so on). Tables are created with the specified compression. See *Oracle Database SQL Language Reference* for information about valid table compression syntax.

  If the table compression clause is more than one word, then it must be contained in single or double quotation marks. Also, your operating system can require you to enclose the clause in escape characters, such as the backslash character. For example:

  ```
  TRANSFORM=TABLE_COMPRESSION_CLAUSE:\"COLUMN STORE COMPRESS FOR QUERY HIGH\"
  ```

  Specifying this transform changes the type of compression for all tables in the job, including tables that provide storage for materialized views.

- `XMLTYPE_STORAGE_CLAUSE:[TRANSPORTABLE BINARY XML | BINARY XML]`

  There is no default. If the transform is not used, then the source data type in the dump file is the data type defined on the target, and the `NOT TRANSPORTABLE` clauses remain as they are.

  Oracle recommends that you use the `TRANSPORTABLE BINARY XML` XMLType with Oracle Database 23ai to store data in a self-contained binary format. This format supports sharding and greater scalability. It does not store the metadata used to encode or decode XML data in a central table (central token tables and schema registries), which simplifies the XML data storage and makes it easier to transport. If `TRANSPORTABLE BINARY XML` is set, then it forces the `TRANSPORTABLE` clause to be present in table creation DDLs for Binary XML data. This data type is available for Oracle Database 21c and Oracle Database 19c in Oracle Cloud Infrastructure.

  Use the `BINARY XML` storage XMLType (which is non-transportable) to store the data in a post-parse, binary format designed specifically for XML data. Binary XML is compact, post-parse, XML schema-aware XML data. The metadata used to encode or decode XML data is stored efficiently in a central table. When `BINARY XML` is set, it forces the `NOT TRANSPORTABLE` clause to be present in table creation DDLs for Binary XML data. When

tables with Binary XML data have neither `TRANSPORTABLE` nor `NOT TRANSPORTABLE` clauses, the default is `NOT TRANSPORTABLE`, and the XMLType column remains stored as Binary XML.

You can export and import data of type XMLType regardless of the source database XMLType storage format (object-relational, binary XML or CLOB). However, Oracle Data Pump exports and imports XML data as binary XML data only. The underlying tables and columns used for object-relational storage of XMLType are consequently not exported. Instead, they are converted to binary form and exported as self-describing binary XML data with a token map preamble.

Because XMLType data is exported and imported as XML data, the source and target databases can use different XMLType storage models for that data. You can export data from a database that stores XMLType data one way and import it into a database that stores XMLType data a different way. For details, see *Oracle XML DB Developer's Guide*

Restrictions

- You cannot use `TRANSFORM` with domain indexes to exclude the storage clause of the source metadata.
- You cannot use `REMAP_TABLESPACE` or `TRANSFORM ATTRIBUTE` with Oracle Text indexes.
- Using `TRANSFORM=TABLESPACE:N` is mutually exclusive with `REMAP_TABLESPACE`
- For the `XMLTYPE_STORAGE_CLAUSE` data type, the following restrictions apply:
  - Do not use the option `table_exists_action=append` to import more than once from the same dump file into an XMLType table, regardless of the XMLType storage model used. Doing so raises a unique-constraint violation error, because rows in XMLType tables are always exported and imported using a unique object identifier.
  - Transportable Binary XML can only be stored using SecureFile `LOB`. If a `BasicFile` clause is specified for TBX, then an error is raised. The only exceptions are for the `SYS` and XDB users, which are permitted to use the BasicFile clause.
  - Binary XML defaults to SecureFiles storage option. However, if either of the following is true, it is not possible to use SecureFiles LOB storage. In that case, BasicFile is the default option for binary XML data:
    - The tablespace for the XMLType table does not use automatic segment space management.
    - A setting in file `init.ora` prevents SecureFiles LOB storage. For example, see parameter `DB_SECUREFILE`.
- The use of the unstructured (`CLOB`) storage model for XMLType is deprecated in Oracle Database 12c Release 1 (12.1.0.1), and later releases.
- Oracle Data Pump for Oracle Database 11g Release 1 (11.1) does not support the export of XML schemas, XML schema-based XMLType columns, or binary XML data to database releases prior to 11.1.

Examples

TRANSFORM use case example

The following is a common example of using `TRANSFORM`. For the following example, assume that you have exported the `employees` table in the `hr` schema. The SQL `CREATE TABLE` statement that results when you then import the table is similar to the following:

```
CREATE TABLE "HR"."EMPLOYEES"
   ( "EMPLOYEE_ID" NUMBER(6,0),
     "FIRST_NAME" VARCHAR2(20),
     "LAST_NAME" VARCHAR2(25) CONSTRAINT "EMP_LAST_NAME_NN" NOT NULL ENABLE,
     "EMAIL" VARCHAR2(25) CONSTRAINT "EMP_EMAIL_NN" NOT NULL ENABLE,
     "PHONE_NUMBER" VARCHAR2(20),
     "HIRE_DATE" DATE CONSTRAINT "EMP_HIRE_DATE_NN" NOT NULL ENABLE,
     "JOB_ID" VARCHAR2(10) CONSTRAINT "EMP_JOB_NN" NOT NULL ENABLE,
     "SALARY" NUMBER(8,2),
     "COMMISSION_PCT" NUMBER(2,2),
     "MANAGER_ID" NUMBER(6,0),
     "DEPARTMENT_ID" NUMBER(4,0)
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 10240 NEXT 16384 MINEXTENTS 1 MAXEXTENTS 121
  PCTINCREASE 50 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "SYSTEM" ;
```

If you do not want to retain the `STORAGE` clause or `TABLESPACE` clause, then you can remove them from the `CREATE STATEMENT` by using the Import `TRANSFORM` parameter. Specify the value of `SEGMENT_ATTRIBUTES` as `N`. This results in the exclusion of segment attributes (both storage and tablespace) from the table.

```
> impdp hr TABLES=hr.employees DIRECTORY=dpump_dir1 DUMPFILE=hr_emp.dmp
  TRANSFORM=SEGMENT_ATTRIBUTES:N:table
```

The resulting `CREATE TABLE` statement for the `employees` table then looks similar to the following. It does not contain a `STORAGE` or `TABLESPACE` clause; the attributes for the default tablespace for the `HR` schema are used instead.

```
CREATE TABLE "HR"."EMPLOYEES"
   ( "EMPLOYEE_ID" NUMBER(6,0),
     "FIRST_NAME" VARCHAR2(20),
     "LAST_NAME" VARCHAR2(25) CONSTRAINT "EMP_LAST_NAME_NN" NOT NULL ENABLE,
     "EMAIL" VARCHAR2(25) CONSTRAINT "EMP_EMAIL_NN" NOT NULL ENABLE,
     "PHONE_NUMBER" VARCHAR2(20),
     "HIRE_DATE" DATE CONSTRAINT "EMP_HIRE_DATE_NN" NOT NULL ENABLE,
     "JOB_ID" VARCHAR2(10) CONSTRAINT "EMP_JOB_NN" NOT NULL ENABLE,
     "SALARY" NUMBER(8,2),
     "COMMISSION_PCT" NUMBER(2,2),
     "MANAGER_ID" NUMBER(6,0),
     "DEPARTMENT_ID" NUMBER(4,0)
   );
```

As shown in the previous example, the SEGMENT_ATTRIBUTES transform applies to both storage and tablespace attributes. To omit only the STORAGE clause and retain the TABLESPACE clause, you can use the STORAGE transform, as follows:

```
> impdp hr TABLES=hr.employees DIRECTORY=dpump_dir1 DUMPFILE=hr_emp.dmp
  TRANSFORM=STORAGE:N:table
```

The SEGMENT_ATTRIBUTES and STORAGE transforms can be applied to all applicable table and index objects by not specifying the object type on the TRANSFORM parameter, as shown in the following command:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp SCHEMAS=hr TRANSFORM=SEGMENT_ATTRIBUTES:N
```

Related Topics

- XMLTYPE_STORAGE_CLAUSE: Export/Import Limitations for Oracle XML DB Repository in *Oracle XML DB Developer's Guide*
- XMLTYPE_STORAGE_CLAUSE: Oracle XML DB Features in *Oracle XML DB Developer's Guide*
- CREATE INDEX in *Oracle Database Administrator's Guide*
- Improved Analytics Using the In-Memory Column Store in *Oracle Database Data Warehousing Guide*
- SAMPLE
  The Oracle Data Pump Export command-line utility SAMPLE parameter specifies a percentage of the data rows that you want to be sampled and unloaded from the source database.
- CREATE TABLE in *Oracle Database SQL Language Reference*

3.4.55 TRANSPORT_DATAFILES

The Oracle Data Pump Import command-line mode TRANSPORT_DATAFILES parameter specifies a list of data files that are imported into the target database when TRANSPORTABLE=ALWAYS is set during the export.

Default

There is no default

Purpose

Specifies a list of data files that are imported into the target database by a transportable-tablespace mode import, or by a table-mode or full-mode import, when TRANSPORTABLE=ALWAYSis set during the export. The data files must already exist on the target database system.

Syntax and Description

```
TRANSPORT_DATAFILES=datafile_name
```

The *datafile_name* must include an absolute directory path specification (not a directory object name) that is valid on the system where the target database resides.

The *datafile_name* can also use wildcards in the file name portion of an absolute path specification. An Asterisk (*) matches 0 to *N* characters. A question mark (?) matches exactly

one character. You cannot use wildcards in the directory portions of the absolute path specification. If a wildcard is used, then all matching files must be part of the transport set. If any files are found that are not part of the transport set, then an error is displayed, and the import job terminates.

At some point before the import operation, you must copy the data files from the source system to the target system. You can copy the data files by using any copy method supported by your operating system. If desired, you can rename the files when you copy them to the target system. See Example 2.

If you already have a dump file set generated by any transportable mode export, then you can perform a transportable-mode import of that dump file by specifying the dump file (which contains the metadata) and the `TRANSPORT_DATAFILES` parameter. The presence of the `TRANSPORT_DATAFILES` parameter tells import that it is a transportable-mode import and where to get the actual data.

Depending on your operating system, the use of quotation marks when you specify a value for this parameter can also require that you use escape characters. Oracle recommends that you place this parameter in a parameter file, which can reduce the number of escape characters that you would otherwise be required to use on the command line.

Restrictions

- You cannot use the `TRANSPORT_DATAFILES` parameter in conjunction with the `QUERY` parameter.
- The `TRANSPORT_DATAFILES` directory portion of the absolute file path cannot contain wildcards. However, the file name portion of the absolute file path can contain wildcards

Example 1

The following is an example of using the `TRANSPORT_DATAFILES` parameter. Assume you have a parameter file, `trans_datafiles.par`, with the following content:

```
DIRECTORY=dpump_dir1
DUMPFILE=tts.dmp
TRANSPORT_DATAFILES='/user01/data/tbs1.dbf'
```

You can then issue the following command:

```
> impdp hr PARFILE=trans_datafiles.par
```

Example 2

This example illustrates the renaming of data files as part of a transportable tablespace export and import operation. Assume that you have a data file named `employees.dat` on your source system.

1. Using a method supported by your operating system, manually copy the data file named `employees.dat` from your source system to the system where your target database resides. As part of the copy operation, rename it to `workers.dat`.
2. Perform a transportable tablespace export of tablespace `tbs_1`.

   ```
   > expdp hr DIRECTORY=dpump_dir1 DUMPFILE=tts.dmp
   TRANSPORT_TABLESPACES=tbs_1
   ```

The metadata only (no data) for `tbs_1` is exported to a dump file named `tts.dmp`. The actual data was copied over to the target database in step 1.

3. Perform a transportable tablespace import, specifying an absolute directory path for the data file named `workers.dat`:

```
> impdp hr DIRECTORY=dpump_dir1 DUMPFILE=tts.dmp
TRANSPORT_DATAFILES='/user01/data/workers.dat'
```

The metadata contained in `tts.dmp` is imported and Data Pump then assigns the information in the `workers.dat` file to the correct place in the database.

Example 3

This example illustrates use of the asterisk (`*`) wildcard character in the file name when used with the `TRANSPORT_DATAFILES` parameter.

```
TRANSPORT_DATAFILES='/db1/hrdata/payroll/emp*.dbf'
```

This parameter use results in Oracle Data Pump validating that all files in the directory `/db1/hrdata/payroll/` of type `.dbf` whose names begin with `emp` are part of the transport set.

Example 4

This example illustrates use of the question mark (`?`) wildcard character in the file name when used with the `TRANSPORT_DATAFILES` parameter.

```
TRANSPORT_DATAFILES='/db1/hrdata/payroll/m?emp.dbf'
```

This parameter use results in Oracle Data Pump validating that all files in the directory `/db1/hrdata/payroll/` of type `.dbf` whose name begins with `m`, followed by any other single character, and ending in `emp` are part of the transport set. For example, a file named `myemp.dbf` is included, but `memp.dbf` is not included.

Related Topics

- About Import Command-Line Mode

3.4.56 TRANSPORT_FULL_CHECK

The Oracle Data Pump Import command-line mode `TRANSPORT_FULL_CHECK` parameter specifies whether to verify that the specified transportable tablespace set is being referenced by objects in other tablespaces.

Default

`NO`

Purpose

Specifies whether to verify that the specified transportable tablespace set is being referenced by objects in other tablespaces.

Syntax and Description

```
TRANSPORT_FULL_CHECK=[YES | NO]
```

If TRANSPORT_FULL_CHECK=YES, then Import verifies that there are no dependencies between those objects inside the transportable set and those outside the transportable set. The check addresses two-way dependencies. For example, if a table is inside the transportable set but its index is not, then a failure is returned and the import operation is terminated. Similarly, a failure is also returned if an index is in the transportable set but the table is not.

If TRANSPORT_FULL_CHECK=NO, then Import verifies only that there are no objects within the transportable set that are dependent on objects outside the transportable set. This check addresses a one-way dependency. For example, a table is not dependent on an index, but an index *is* dependent on a table, because an index without a table has no meaning. Therefore, if the transportable set contains a table, but not its index, then this check succeeds. However, if the transportable set contains an index, but not the table, then the import operation is terminated.

In addition to this check, Import always verifies that all storage segments of all tables (and their indexes) defined within the tablespace set specified by TRANSPORT_TABLESPACES are actually contained within the tablespace set.

Restrictions

- This parameter is valid for transportable mode (or table mode or full mode when TRANSPORTABLE=ALWAYS was specified on the export) only when the NETWORK_LINK parameter is specified.

Example

In the following example, *source_database_link* would be replaced with the name of a valid database link. The example also assumes that a data file named tbs6.dbf already exists.

Assume you have a parameter file, full_check.par, with the following content:

```
DIRECTORY=dpump_dir1
TRANSPORT_TABLESPACES=tbs_6
NETWORK_LINK=source_database_link
TRANSPORT_FULL_CHECK=YES
TRANSPORT_DATAFILES='/wkdir/data/tbs6.dbf'
```

You can then issue the following command:

```
> impdp hr PARFILE=full_check.par
```

3.4.57 TRANSPORT_TABLESPACES

The Oracle Data Pump Import command-line mode TRANSPORT_TABLESPACES parameter specifies that you want to perform an import in transportable-tablespace mode over a database link.

Default

There is no default.

Purpose

Specifies that you want to perform an import in transportable-tablespace mode over a database link (as specified with the NETWORK_LINK parameter.)

Syntax and Description

```
TRANSPORT_TABLESPACES=tablespace_name [, ...]
```

Use the TRANSPORT_TABLESPACES parameter to specify a list of tablespace names for which object metadata are imported from the source database into the target database.

Because this import is a transportable-mode import, the tablespaces into which the data is imported are automatically created by Data Pump.You do not need to pre-create them. However, copy the data files to the target database before starting the import.

When you specify TRANSPORT_TABLESPACES on the import command line, you must also use the NETWORK_LINK parameter to specify a database link. A database link is a connection between two physical database servers that allows a client to access them as one logical database. Therefore, the NETWORK_LINK parameter is required, because the object metadata is exported from the source (the database being pointed to by NETWORK_LINK) and then imported directly into the target (database from which the impdp command is issued), using that database link. There are no dump files involved in this situation. If you copied the actual data to the target in a separate operation using some other means, then specify the TRANSPORT_DATAFILES parameter and indicate where the data is located.

> **Note:**
>
> If you already have a dump file set generated by a transportable-tablespace mode export, then you can perform a transportable-mode import of that dump file, but in this case you do not specify TRANSPORT_TABLESPACES or NETWORK_LINK. Doing so would result in an error. Rather, you specify the dump file (which contains the metadata) and the TRANSPORT_DATAFILES parameter. The presence of the TRANSPORT_DATAFILES parameter tells import that it's a transportable-mode import and where to get the actual data.

When transportable jobs are performed, it is best practice to keep a copy of the data files on the source system until the import job has successfully completed on the target system. If the import job fails, then you still have uncorrupted copies of the data files.

Restrictions

- You cannot export transportable tablespaces and then import them into a database at a lower release level. The target database into which you are importing must be at the same or later release level as the source database.
- The TRANSPORT_TABLESPACES parameter is valid only when the NETWORK_LINK parameter is also specified.
- To use the TRANSPORT_TABLESPACES parameter to perform a transportable tablespace import, the COMPATIBLE initialization parameter must be set to at least 11.0.0.
- Depending on your operating system, the use of quotation marks when you specify a value for this parameter can also require that you use escape characters. Oracle recommends that you place this parameter in a parameter file. If you use a parameter file, then that can reduce the number of escape characters that you have to use on a command line.
- Transportable tablespace jobs do not support the ACCESS_METHOD parameter for Data Pump Import.

Example

In the following example, the *source_database_link* would be replaced with the name of a valid database link. The example also assumes that a data file named `tbs6.dbf` has already been copied from the source database to the local system. Suppose you have a parameter file, `tablespaces.par`, with the following content:

```
DIRECTORY=dpump_dir1
NETWORK_LINK=source_database_link
TRANSPORT_TABLESPACES=tbs_6
TRANSPORT_FULL_CHECK=NO
TRANSPORT_DATAFILES='user01/data/tbs6.dbf'
```

You can then issue the following command:

```
> impdp hr PARFILE=tablespaces.par
```

Related Topics

- Database Links in *Oracle Database Administrator's Guide*
- Using Data File Copying to Move Data
- How Does Oracle Data Pump Handle Timestamp Data?
- About Import Command-Line Mode

3.4.58 TRANSPORTABLE

The optional Oracle Data Pump Import command-line mode `TRANSPORTABLE` parameter specifies either that transportable tables are imported with `KEEP_READ_ONLY`, or `NO_BITMAP_REBUILD`.

Default

None.

Purpose

This optional parameter enables you to specify two values to control how transportable table imports are managed: `KEEP_READ_ONLY` and `NO_BITMAP_REBUILD`.There is no default value for the `TRANSPORTABLE` parameter.

Syntax and Description

```
TRANSPORTABLE = [ALWAYS|NEVER|KEEP_READ_ONLY|NO_BITMAP_REBUILD]
```

The definitions of the allowed values are as follows:

- `ALWAYS` (valid for Full and Table Export) indicates a transportable export. If specified, then only the metadata is exported, and data files are plugged into the target database during the import.
- `NEVER` indicates that only a traditional data export is enabled.

- KEEP_READ_ONLY: Valid with transportable mode imports (table, tablespace, full). If specified, then tablespaces and data files remain in read-only mode. Keeping tablespaces and data files in read-only mode enables the transportable data file set to be available to be plugged in to multiple target databases. When data files are in read-only mode, this disables updating tables containing TSTZ column data, if that data needs to be updated, to avoid issues with different TSTZ versions. For this reason, tables with TSTZ columns are dropped from the transportable import. Placing data files in read-only mode also disables rebuilding of tablespace storage bitmaps to reclaim segments.
- NO_BITMAP_REBUILD: Indicates that you do not want Oracle Data Pump to reclaim storage segments by rebuilding tablespace storage bitmaps during the transportable import. Not rebuilding the bitmaps can speed up the import. You can reclaim segments at a later time by using the DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_BITMAPS() procedure.

APIs or Classes

You can set the TRANSPORTABLE parameter value by using the existing procedure DBMS_DATAPUMP.SET_PARAMETER.

Restrictions

- The Import TRANSPORTABLE parameter is valid only if the NETWORK_LINK parameter is also specified.
- The TRANSPORTABLE parameter is only valid in table mode imports and full mode imports.
- The user performing a transportable import requires both the DATAPUMP_EXP_FULL_DATABASE role on the source database, and the DATAPUMP_IMP_FULL_DATABASE role on the target database.
- All objects with storage that are selected for network import must have all of their storage segments on the source system either entirely within administrative, non-transportable tablespaces (SYSTEM / SYSAUX), or entirely within user-defined, transportable tablespaces. Storage for a single object cannot straddle the two kinds of tablespaces.
- To use the TRANSPORTABLE parameter to perform a network-based full transportable import, the Data Pump VERSION parameter must be set to at least 12.0 if the source database is release 11.2.0.3. If the source database is release 12.1 or later, then the VERSION parameter is not required, but the COMPATIBLE database initialization parameter must be set to 12.0.0 or later.

Example of a Network Link Import

The following example shows the use of the TRANSPORTABLE parameter during a network link import, where *datafile_name* is the data file that you want to import.

```
> impdp system TABLES=hr.sales TRANSPORTABLE=ALWAYS
  DIRECTORY=dpump_dir1 NETWORK_LINK=dbs1 PARTITION_OPTIONS=DEPARTITION
  TRANSPORT_DATAFILES=datafile_name
```

Example of a Full Transportable Import

The following example shows the use of the TRANSPORTABLE parameter when performing a full transportable import over the database link dbs1. The import specifies a password for the tables with encrypted columns.

```
> impdp import_admin FULL=Y TRANSPORTABLE=ALWAYS VERSION=12 NETWORK_LINK=dbs1
  ENCRYPTION_PASSWORD=password TRANSPORT_DATAFILES=datafile_name
  LOGFILE=dpump_dir1:fullnet.log
```

Example of Setting NEVER or ALWAYS

Setting the TRANSPORTABLE parameter with string values is limited to NEVER or ALWAYS values:

```
SYS.DBMS_DATAPUMP.SET_PARAMETER(jobhdl, 'TRANSPORTABLE','ALWAYS');
SYS.DBMS_DATAPUMP.SET_PARAMETER(jobhdl, 'TRANSPORTABLE','NEVER');
```

The new TRANSPORTABLE parameter options are set using the new numeric bitmask values:

DBMS_DATAPUMP.KU$_TTS_NEVER is the value 1

DBMS_DATAPUMP.KU$_TTS_ALWAYS is the value 2

DBMS_DATAPUMP.KU$_TTS_KEEP_READ_ONLY is the value 4

DBMS_DATAPUMP.KU$_TTS_NO_BITMAP_REBUILD is the value 8

```
SYS.DBMS_DATAPUMP.SET_PARAMETER(jobhdl, 'TRANSPORTABLE',
DBMS_DATAPUMP.KU$_TTS_ALWAYS+DBMS_DATAPUMP.KU$_TTS_KEEP_READ_ONLY);
```

Example of a File-Based Transportable Tablespace Import

The following example shows the use of the TRANSPORTABLE parameter during a file-based transportable tablespace import. The specified KEEP_READ_ONLY option indicates that the data file remains in read–only access throughout the import operation. The required data files are reported by the transportable tablespace export.

```
impdp system DIRECTORY=dpump_dir DUMPFILE=dumpfile_name
TRANSPORT_DATAFILES=datafile_name TRANSPORTABLE=KEEP_READ_ONLY
```

Related Topics

- About Import Command-Line Mode
- Using Data File Copying to Move Data

3.4.59 VERIFY_CHECKSUM

The Oracle Data Pump Import command-line utility VERIFY_CHECKSUM parameter specifies whether to verify dump file checksums.

Default

If checksums were generated when the export dump files were first produced, then the default value is YES.

Purpose

Specifies whether Oracle Data Pump verifies dump file checksums before proceeding with the import operation.

Syntax and Description

```
VERIFY_CHECKSUM=[YES|NO]
```

- `YES` Specifies that Oracle Data Pump performs file checksum verification for each dump file in the export dump file set.
- `NO` Specifies that Oacle Data Pump does not perform checksum verification for the dump file set.

Restrictions

- To use this checksum feature, the COMPATIBLE initialization parameter must be set to at least 20.0.
- The VERIFY_CHECKSUM and VERIFY_ONLY parameters are mutually exclusive.

Example

This example performs a schema-mode load of the `HR` schema. Checksum verification of the dump files is performed before the actual import operation begins.

```
impdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp VERIFY_CHECKSUM=YES
```

3.4.60 VERIFY_ONLY

The Oracle Data Pump Import command-line utility `VERIFY_ONLY` parameter enables you to verify the checksum for the dump file.

Default

```
NO
```

Purpose

Specifies whether Oracle Data Pump verifies the dump file checksums.

Syntax and Description

```
VERIFY_ONLY=[YES|NO]
```

When set to `YES`, Oracle Data Pump verifies the checksum. If there are no errors, then you can issue another import command for the dump file set.

Restrictions

- When you set the `VERIFY_ONLY` parameter to `YES`, no actual import operation is performed. The Oracle Data Pump Import job only completes the listed verification checks.
- The `VERIFY_CHECKSUM` and `VERIFY_ONLY` parameters are mutually exclusive.

Example

This example performs a verification check of the `hr.dmp` dump file. Beyond the verification checks, no actual import of data is performed.

```
impdp system directory=dpump_dir1 dumpfile=hr.dmp verify_checksum=yes
```

3.4.61 VERSION

The Oracle Data Pump Import command-line mode `VERSION` parameter specifies the version of database objects that you want to import.

Default

You should rarely have to specify the `VERSION` parameter on an import operation. Oracle Data Pump uses whichever of the following is earlier:

- The version associated with the dump file, or source database in the case of network imports
- The version specified by the `COMPATIBLE` initialization parameter on the target database

Purpose

Specifies the version of database objects that you want to be imported (that is, only database objects and attributes that are compatible with the specified release will be imported). Note that this does not mean that Oracle Data Pump Import can be used with releases of Oracle Database earlier than 10.1. Oracle Data Pump Import only works with Oracle Database 10g release 1 (10.1) or later. The `VERSION` parameter simply allows you to identify the version of the objects being imported.

Syntax and Description

```
VERSION=[COMPATIBLE | LATEST | version_string]
```

This parameter can be used to load a target system whose Oracle Database is at an earlier compatibility release than that of the source system. When the `VERSION` parameter is set, database objects or attributes on the source system that are incompatible with the specified release are not moved to the target. For example, tables containing new data types that are not supported in the specified release are not imported. Legal values for this parameter are as follows:

- `COMPATIBLE` - This is the default value. The version of the metadata corresponds to the database compatibility level. Database compatibility must be set to 9.2.0 or later.
- `LATEST` - The version of the metadata corresponds to the database release. Specifying `VERSION=LATEST` on an import job has no effect when the target database's actual version is later than the version specified in its `COMPATIBLE` initialization parameter.
- *`version_string`* - A specific database release (for example, `12.2.0`).

Restrictions

- If the Oracle Data Pump `VERSION` parameter is specified as any value earlier than 12.1, then the Oracle Data Pump dump file excludes any tables that contain `VARCHAR2` or `NVARCHAR2` columns longer than 4000 bytes and any `RAW` columns longer than 2000 bytes.

- Full imports performed over a network link require that you set `VERSION=12` if the target is Oracle Database 12c Release 1 (12.1.0.1) or later and the source is Oracle Database 11g Release 2 (11.2.0.3) or later.
- Dump files created on Oracle Database 11g releases with the Oracle Data Pump parameter `VERSION=12` can only be imported on Oracle Database 12c Release 1 (12.1) and later.
- The value of the `VERSION` parameter affects the import differently depending on whether data-bound collation (DBC) is enabled.

> **Note:**
>
> Database objects or attributes that are incompatible with the release specified for `VERSION` are not exported. For example, tables containing new data types that are not supported in the specified release are not exported. If you attempt to export dump files into an Oracle Cloud Infrastructure (OCI) Native credential store where `VERSION=19`, then the export fails, and you receive the following error:
>
> ```
> ORA-39463 "header block format is not supported for object-store URI dump file"
> ```

Example

In the following example, assume that the target is an Oracle Database 12c Release 1 (12.1.0.1) database and the source is an Oracle Database 11g Release 2 (11.2.0.3) database. In that situation, you must set `VERSION=12` for network-based imports. Also note that even though full is the default import mode, you must specify it on the command line when the `NETWORK_LINK` parameter is being used.

```
> impdp hr FULL=Y DIRECTORY=dpump_dir1
  NETWORK_LINK=source_database_link VERSION=12
```

Related Topics

- Oracle Data Pump Behavior with Data-Bound Collation
- Exporting and Importing Between Different Oracle Database Releases

3.4.62 VIEWS_AS_TABLES (Network Import)

The Oracle Data Pump Import command-line mode `VIEWS_AS_TABLES` (Network Import) parameter specifies that you want one or more views to be imported as tables.

Default

There is no default.

> **Note:**
>
> This description of `VIEWS_AS_TABLES` is applicable during network imports, meaning that you supply a value for the Data Pump Import `NETWORK_LINK` parameter.

Purpose

Specifies that you want one or more views to be imported as tables.

Syntax and Description

```
VIEWS_AS_TABLES=[schema_name.]view_name[:table_name], ...
```

Oracle Data Pump imports a table with the same columns as the view and with row data fetched from the view. Oracle Data Pump also imports objects dependent on the view, such as grants and constraints. Dependent objects that do not apply to tables (for example, grants of the `UNDER` object privilege) are not imported. You can use the `VIEWS_AS_TABLES` parameter by itself, or along with the `TABLES` parameter. If either is used, then Oracle Data Pump performs a table-mode import.

The syntax elements are defined as follows:

`schema_name`: The name of the schema in which the view resides. If a schema name is not supplied, it defaults to the user performing the import.

`view_name`: The name of the view to be imported as a table. The view must exist and it must be a relational view with only scalar, non-LOB columns. If you specify an invalid or non-existent view, the view is skipped and an error message is returned.

`table_name`: The name of a table that you want to serve as the source of the metadata for the imported view. By default, Oracle Data Pump automatically creates a temporary "template table" with the same columns and data types as the view, but no rows. If the database is read-only, then this default creation of a template table fails. In such a case, you can specify a table name. The table must be in the same schema as the view. It must be a non-partitioned relational table with heap organization. It cannot be a nested table.

If the import job contains multiple views with explicitly specified template tables, then the template tables must all be different. For example, in the following job (in which two views use the same template table), one of the views is skipped:

```
impdp hr DIRECTORY=dpump_dir NETWORK_LINK=dblink1
VIEWS_AS_TABLES=v1:employees,v2:employees
```

An error message is returned reporting the omitted object.

Template tables are automatically dropped after the import operation is completed. While they exist, you can perform the following query to view their names (which all begin with `KU$VAT`):

```
SQL> SELECT * FROM user_tab_comments WHERE table_name LIKE 'KU$VAT%';
TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
COMMENTS
--------------------------------------------------------
KU$VAT_63629                   TABLE
Data Pump metadata template table for view HR.EMPLOYEESV
```

Restrictions

- The `VIEWS_AS_TABLES` parameter cannot be used with the `TRANSPORTABLE=ALWAYS` parameter.

- Tables created using the `VIEWS_AS_TABLES` parameter do not contain any hidden columns that were part of the specified view.
- The `VIEWS_AS_TABLES` parameter does not support tables that have columns with a data type of `LONG`.

Example

The following example performs a network import to import the contents of the view `hr.v1` from a read-only database. The `hr` schema on the source database must contain a template table with the same geometry as the view `view1` (call this table `view1_tab`). The `VIEWS_AS_TABLES` parameter lists the view name and the table name separated by a colon:

```
> impdp hr VIEWS_AS_TABLES=view1:view1_tab NETWORK_LINK=dblink1
```

The view is imported as a table named `view1` with rows fetched from the view. The metadata for the table is copied from the template table `view1_tab`.

3.5 Commands Available in Data Pump Export Interactive-Command Mode

Check which command options are available to you when using Data Pump Export in interactive mode.

- About Oracle Data Pump Export Interactive Command Mode
  Learn about commands you can use with Oracle Data Pump Export in interactive command mode while your current job is running.
- ADD_FILE
  The Oracle Data Pump Export interactive command mode `ADD_FILE` parameter adds additional files or substitution variables to the export dump file set.
- CONTINUE_CLIENT
  The Oracle Data Pump Export interactive command mode `CONTINUE_CLIENT` parameter changes the Export mode from interactive-command mode to logging mode.
- EXIT_CLIENT
  The Oracle Data Pump Export interactive command mode `EXIT_CLIENT` parameter stops the export client session, exits Export, and discontinues logging to the terminal, but leaves the current job running.
- FILESIZE
  The Oracle Data Pump Export interactive command mode `FILESIZE` parameter redefines the maximum size of subsequent dump files.
- HELP
  The Oracle Data Pump Export interactive command mode `HELP` parameter provides information about Data Pump Export commands available in interactive-command mode.
- KILL_JOB
  The Oracle Data Pump Export interactive command mode `KILL_JOB` parameter detaches all currently attached worker client sessions, and then terminates the current job. It exits Export, and returns to the terminal prompt.
- PARALLEL
  The Export Interactive-Command Mode `PARALLEL` parameter enables you to increase or decrease the number of active processes (child and parallel child processes) for the current job.

- START_JOB
  The Oracle Data Pump Export interactive command mode `START_JOB` parameter starts the current job to which you are attached.
- STATUS
  The Oracle Data Pump Export interactive command `STATUS` parameter displays status information about the export, and enables you to set the display interval for logging mode status.
- STOP_JOB
  The Oracle Data Pump Export interactive command mode `STOP_JOB` parameter stops the current job. It stops the job either immediately, or after an orderly shutdown, and exits Export.

2.5.1 About Oracle Data Pump Export Interactive Command Mode

Learn about commands you can use with Oracle Data Pump Export in interactive command mode while your current job is running.

In interactive command mode, the current job continues running, but logging to the terminal is suspended, and the Export prompt (`Export>`) is displayed.

To start interactive-command mode, do one of the following:

- From an attached client, press Ctrl+C.
- From a terminal other than the one on which the job is running, specify the `ATTACH` parameter in an `expdp` command to attach to the job. `ATTACH` is a useful feature in situations in which you start a job at one location, and need to check on it at a later time from a different location.

The following table lists the activities that you can perform for the current job from the Data Pump Export prompt in interactive-command mode.

Table 3-1 Supported Activities in Data Pump Export's Interactive-Command Mode

Activity	Command Used
Add additional dump files.	`ADD_FILE`
Exit interactive mode and enter logging mode.	`CONTINUE_CLIENT`
Stop the export client session, but leave the job running.	EXIT_CLIENT
Redefine the default size to be used for any subsequent dump files.	`FILESIZE`
Display a summary of available commands.	HELP
Detach all currently attached client sessions and terminate the current job.	KILL_JOB
Increase or decrease the number of active worker processes for the current job. This command is valid only in the Enterprise Edition of Oracle Database 11*g* or later.	PARALLEL
Restart a stopped job to which you are attached.	START_JOB
Display detailed status for the current job and/or set status interval.	STATUS
Stop the current job for later restart.	STOP_JOB

2.5.2 ADD_FILE

The Oracle Data Pump Export interactive command mode `ADD_FILE` parameter adds additional files or substitution variables to the export dump file set.

Purpose

Adds additional files or substitution variables to the export dump file set.

Syntax and Description

```
ADD_FILE=[directory_object:]file_name [,...]
```

Each file name can have a different directory object. If no directory object is specified, then the default is assumed.

The *`file_name`* must not contain any directory path information. However, it can include a substitution variable, `%U`, which indicates that multiple files can be generated using the specified file name as a template.

The size of the file being added is determined by the setting of the `FILESIZE` parameter.

Example

The following example adds two dump files to the dump file set. A directory object is not specified for the dump file named `hr2.dmp`, so the default directory object for the job is assumed. A different directory object, `dpump_dir2`, is specified for the dump file named `hr3.dmp`.

```
Export> ADD_FILE=hr2.dmp, dpump_dir2:hr3.dmp
```

Related Topics

- File Allocation with Oracle Data Pump

2.5.3 CONTINUE_CLIENT

The Oracle Data Pump Export interactive command mode `CONTINUE_CLIENT` parameter changes the Export mode from interactive-command mode to logging mode.

Purpose

Changes the Export mode from interactive-command mode to logging mode.

Syntax and Description

```
CONTINUE_CLIENT
```

In logging mode, status is continually output to the terminal. If the job is currently stopped, then `CONTINUE_CLIENT` also causes the client to attempt to start the job.

Example

```
Export> CONTINUE_CLIENT
```

2.5.4 EXIT_CLIENT

The Oracle Data Pump Export interactive command mode `EXIT_CLIENT` parameter stops the export client session, exits Export, and discontinues logging to the terminal, but leaves the current job running.

Purpose

Stops the export client session, exits Export, and discontinues logging to the terminal, but leaves the current job running.

Syntax and Description

```
EXIT_CLIENT
```

Because `EXIT_CLIENT` leaves the job running, you can attach to the job at a later time. To see the status of the job, you can monitor the log file for the job, or you can query the `USER_DATAPUMP_JOBS` view, or the `V$SESSION_LONGOPS` view.

Example

```
Export> EXIT_CLIENT
```

2.5.5 FILESIZE

The Oracle Data Pump Export interactive command mode `FILESIZE` parameter redefines the maximum size of subsequent dump files.

Purpose

Redefines the maximum size of subsequent dump files. If the size is reached for any member of the dump file set, then that file is closed and an attempt is made to create a new file, if the file specification contains a substitution variable or if additional dump files have been added to the job.

Syntax and Description

```
FILESIZE=integer[B | KB | MB | GB | TB]
```

The *integer* can be immediately followed (do not insert a space) by `B`, `KB`, `MB`, `GB`, or `TB` (indicating bytes, kilobytes, megabytes, gigabytes, and terabytes respectively). Bytes is the default. The actual size of the resulting file may be rounded down slightly to match the size of the internal blocks used in dump files.

A file size of 0 is equivalent to the maximum file size of 16 TB.

Restrictions

- The minimum size for a file is ten times the default Oracle Data Pump block size, which is 4 kilobytes.
- The maximum size for a file is 16 terabytes.

Example

```
Export> FILESIZE=100MB
```

2.5.6 HELP

The Oracle Data Pump Export interactive command mode `HELP` parameter provides information about Data Pump Export commands available in interactive-command mode.

Purpose

Provides information about Oracle Data Pump Export commands available in interactive-command mode.

Syntax and Description

```
HELP
```

Displays information about the commands available in interactive-command mode.

Example

```
Export> HELP
```

2.5.7 KILL_JOB

The Oracle Data Pump Export interactive command mode `KILL_JOB` parameter detaches all currently attached worker client sessions, and then terminates the current job. It exits Export, and returns to the terminal prompt.

Purpose

Detaches all currently attached child client sessions, and then terminates the current job. It exits Export and returns to the terminal prompt.

Syntax and Description

```
KILL_JOB
```

A job that is terminated using `KILL_JOB` cannot be restarted. All attached clients, including the one issuing the `KILL_JOB` command, receive a warning that the job is being terminated by the current user and are then detached. After all child clients are detached, the job's process structure is immediately run down and the Data Pump control job table and dump files are deleted. Log files are not deleted.

Example

```
Export> KILL_JOB
```

2.5.8 PARALLEL

The Export Interactive-Command Mode `PARALLEL` parameter enables you to increase or decrease the number of active processes (child and parallel child processes) for the current job.

Purpose

Enables you to increase or decrease the number of active processes (child and parallel child processes) for the current job.

Syntax and Description

```
PARALLEL=integer
```

`PARALLEL` is available as both a command-line parameter, and as an interactive-command mode parameter. You set it to the desired number of parallel processes (child and parallel child processes). An increase takes effect immediately if there are sufficient files and resources. A decrease does not take effect until an existing process finishes its current task. If the value is decreased, then child processes are idled but not deleted until the job exits.

Restrictions

- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later releases.
- Transportable tablespace metadata cannot be imported in parallel.
- Metadata cannot be imported in parallel when the `NETWORK_LINK` parameter is used.

In addition, the following objects cannot be imported in parallel:

- `TRIGGER`
- `VIEW`
- `OBJECT_GRANT`
- `SEQUENCE`
- `CONSTRAINT`
- `REF_CONSTRAINT`

Example

```
Export> PARALLEL=10
```

Related Topics

- PARALLEL

2.5.9 START_JOB

The Oracle Data Pump Export interactive command mode `START_JOB` parameter starts the current job to which you are attached.

Purpose

Starts the current job to which you are attached.

Syntax and Description

```
START_JOB
```

The `START_JOB` command restarts the current job to which you are attached. The job cannot be running at the time that you enter the command. The job is restarted with no data loss or corruption after an unexpected failure or after you issued a `STOP_JOB` command, provided the dump file set and parent job table have not been altered in any way.

Example

```
Export> START_JOB
```

2.5.10 STATUS

The Oracle Data Pump Export interactive command `STATUS` parameter displays status information about the export, and enables you to set the display interval for logging mode status.

Purpose

Displays cumulative status of the job, a description of the current operation, and an estimated completion percentage. It also allows you to reset the display interval for logging mode status.

Syntax and Description

```
STATUS[=integer]
```

You have the option of specifying how frequently, in seconds, this status should be displayed in logging mode. If no value is entered, or if the default value of 0 is used, then the periodic status display is turned off, and status is displayed only once.

This status information is written only to your standard output device, not to the log file (even if one is in effect).

Example

The following example displays the current job status, and changes the logging mode display interval to five minutes (300 seconds):

```
Export> STATUS=300
```

2.5.11 STOP_JOB

The Oracle Data Pump Export interactive command mode `STOP_JOB` parameter stops the current job. It stops the job either immediately, or after an orderly shutdown, and exits Export.

Purpose

Stops the current job, either immediately, or after an orderly shutdown, and exits Export.

Syntax and Description

```
STOP_JOB[=IMMEDIATE]
```

If the Data Pump control job table and dump file set are not disturbed when or after the `STOP_JOB` command is issued, then the job can be attached to and restarted at a later time with the `START_JOB` command.

To perform an orderly shutdown, use `STOP_JOB` (without any associated value). A warning requiring confirmation will be issued. An orderly shutdown stops the job after worker processes have finished their current tasks.

To perform an immediate shutdown, specify `STOP_JOB=IMMEDIATE`. A warning requiring confirmation will be issued. All attached clients, including the one issuing the `STOP_JOB` command, receive a warning that the job is being stopped by the current user and they will be detached. After all clients are detached, the process structure of the job is immediately run down. That is, the Data Pump control job process will not wait for the child processes to finish their current tasks. There is no risk of corruption or data loss when you specify `STOP_JOB=IMMEDIATE`. However, some tasks that were incomplete at the time of shutdown may have to be redone at restart time.

Example

```
Export> STOP_JOB=IMMEDIATE
```

3.5.1 Commands Available in Oracle Data Pump Import Interactive-Command Mode

In interactive-command mode, the current job continues running, but logging to the terminal is suspended, and the Import prompt (`Import>`) is displayed.

- About Oracle Data Pump Import Interactive Command Mode
  Learn how to run Oracle Data Pump commands from an attached client, or from a terminal other than the one on which the job is running.
- CONTINUE_CLIENT
  The Oracle Data Pump Import interactive command mode `CONTINUE_CLIENT` parameter changes the mode from interactive-command mode to logging mode.
- EXIT_CLIENT
  The Oracle Data Pump Import interactive command mode `EXIT_CLIENT` parameter stops the import client session, exits Import, and discontinues logging to the terminal, but leaves the current job running.

- HELP
  The Oracle Data Pump Import interactive command mode `HELP` parameter provides information about Import commands available in interactive-command mode.
- KILL_JOB
  The Oracle Data Pump Import interactive command mode `KILL_JOB` parameter detaches all currently attached client sessions and then terminates the current job. It exits Import and returns to the terminal prompt.
- PARALLEL
  The Oracle Data Pump Import interactive command mode `PARALLEL` parameter enables you to increase or decrease the number of active child processes, PQ child processes, or both, for the current job.
- START_JOB
  The Oracle Data Pump Import interactive command mode `START_JOB` parameter starts the current job to which you are attached.
- STATUS
  The Oracle Data Pump Import interactive command `STATUS` parameter displays job status, and enables update of the display intervals for logging mode status.
- STOP_JOB
  The Oracle Data Pump Import interactive command mode `STOP_JOB` parameter stops the current job, either immediately or after an orderly shutdown, and exits Import.

About Oracle Data Pump Import Interactive Command Mode

Learn how to run Oracle Data Pump commands from an attached client, or from a terminal other than the one on which the job is running.

To start interactive-command mode, do one of the following:

- From an attached client, press **Ctrl+C**.
- From a terminal other than the one on which the job is running, use the `ATTACH` parameter to attach to the job. This feature is useful in situations in which you start a job at one location, and must check it at a later time from a different location.

Commands for Oracle Data Pump Interactive Mode

The following table lists the activities that you can perform for the current job from the Oracle Data Pump Import prompt in interactive-command mode.

Table 3-2 Supported Activities in Oracle Data Pump Import's Interactive-Command Mode

Activity	Command Used
Exit interactive-command mode.	`CONTINUE_CLIENT`
Stop the import client session, but leave the current job running.	`EXIT_CLIENT`
Display a summary of available commands.	`HELP`
Detach all currently attached client sessions and terminate the current job.	`KILL_JOB`
Increase or decrease the number of active worker processes for the current job. This command is valid only in Oracle Database Enterprise Edition.	`PARALLEL`
Restart a stopped job to which you are attached.	`START_JOB`

Table 3-2 (Cont.) Supported Activities in Oracle Data Pump Import's Interactive-Command Mode

Activity	Command Used
Display detailed status for the current job.	STATUS
Stop the current job.	STOP_JOB

3.5.2 CONTINUE_CLIENT

The Oracle Data Pump Import interactive command mode CONTINUE_CLIENT parameter changes the mode from interactive-command mode to logging mode.

Purpose

Changes the mode from interactive-command mode to logging mode.

Syntax and Description

```
CONTINUE_CLIENT
```

In logging mode, the job status is continually output to the terminal. If the job is currently stopped, then CONTINUE_CLIENT also causes the client to attempt to start the job.

Example

```
Import> CONTINUE_CLIENT
```

3.5.3 EXIT_CLIENT

The Oracle Data Pump Import interactive command mode EXIT_CLIENT parameter stops the import client session, exits Import, and discontinues logging to the terminal, but leaves the current job running.

Purpose

Stops the import client session, exits Import, and discontinues logging to the terminal, but leaves the current job running.

Syntax and Description

```
EXIT_CLIENT
```

Because EXIT_CLIENT leaves the job running, you can attach to the job at a later time if the job is still running, or if the job is in a stopped state. To see the status of the job, you can monitor the log file for the job, or you can query the USER_DATAPUMP_JOBS view or the V$SESSION_LONGOPS view.

Example

```
Import> EXIT_CLIENT
```

3.5.4 HELP

The Oracle Data Pump Import interactive command mode `HELP` parameter provides information about Import commands available in interactive-command mode.

Purpose

Provides information about Oracle Data Pump Import commands available in interactive-command mode.

Syntax and Description

```
HELP
```

Displays information about the commands available in interactive-command mode.

Example

```
Import> HELP
```

3.5.5 KILL_JOB

The Oracle Data Pump Import interactive command mode `KILL_JOB` parameter detaches all currently attached client sessions and then terminates the current job. It exits Import and returns to the terminal prompt.

Purpose

Detaches all currently attached client sessions and then terminates the current job. It exits Import and returns to the terminal prompt.

Syntax and Description

```
KILL_JOB
```

A job that is terminated using `KILL_JOB` cannot be restarted. All attached clients, including the one issuing the `KILL_JOB` command, receive a warning that the job is being terminated by the current user, and are then detached. After all clients are detached, the job process structure is immediately run down, and the Data Pump control job table is deleted. Log files are not deleted.

Example

```
Import> KILL_JOB
```

3.5.6 PARALLEL

The Oracle Data Pump Import interactive command mode `PARALLEL` parameter enables you to increase or decrease the number of active child processes, PQ child processes, or both, for the current job.

Purpose

Enables you to increase or decrease the number of active child processes, parallel query (PQ) child processes, or both, for the current job.

Syntax and Description

```
PARALLEL=integer
```

`PARALLEL` is available as both a command-line parameter and an interactive-mode parameter. You set it to the desired number of parallel processes. An increase takes effect immediately if there are enough resources, and if there is enough work requiring parallelization. A decrease does not take effect until an existing process finishes its current task. If the integer value is decreased, then child processes are idled but not deleted until the job exits.

Restrictions

- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later releases.
- Transportable tablespace metadata cannot be imported in parallel.
- Metadata cannot be imported in parallel when the `NETWORK_LINK` parameter is also used
- The following objects cannot be imported in parallel:
  - `TRIGGER`
  - `VIEW`
  - `OBJECT_GRANT`
  - `SEQUENCE`
  - `CONSTRAINT`
  - `REF_CONSTRAINT`

Example

```
Import> PARALLEL=10
```

3.5.7 START_JOB

The Oracle Data Pump Import interactive command mode `START_JOB` parameter starts the current job to which you are attached.

Purpose

Starts the current job to which you are attached.

Syntax and Description

```
START_JOB[=SKIP_CURRENT=YES]
```

The `START_JOB` command restarts the job to which you are currently attached (the job cannot be currently running). The job is restarted with no data loss or corruption after an unexpected failure, or after you issue a `STOP_JOB` command, provided the dump file set and Data Pump control job table remain undisturbed.

The `SKIP_CURRENT` option enables you to restart a job that previously failed, or that is hung or performing slowly on a particular object. The failing statement or current object being processed is skipped, and the job is restarted from the next work item. For parallel jobs, this option causes each worker to skip whatever it is currently working on and to move on to the next item at restart.

You cannot restart `SQLFILE` jobs.

Example

```
Import> START_JOB
```

3.5.8 STATUS

The Oracle Data Pump Import interactive command `STATUS` parameter displays job status, and enables update of the display intervals for logging mode status.

Purpose

Displays cumulative status of the job, a description of the current operation, and an estimated completion percentage. It also allows you to reset the display interval for logging mode status.

Syntax and Description

```
STATUS[=integer]
```

You have the option of specifying how frequently, in seconds, this status should be displayed in logging mode. If no value is entered or if the default value of `0` is used, then the periodic status display is turned off and status is displayed only once.

This status information is written only to your standard output device, not to the log file (even if one is in effect).

Example

The following example displays the current job status, and changes the logging mode display interval to two minutes (120 seconds).

```
Import> STATUS=120
```

3.5.9 STOP_JOB

The Oracle Data Pump Import interactive command mode `STOP_JOB` parameter stops the current job, either immediately or after an orderly shutdown, and exits Import.

Purpose

Stops the current job, either immediately or after an orderly shutdown, and exits Import.

Syntax and Description

```
STOP_JOB[=IMMEDIATE]
```

After you run `STOP_JOB`, you can attach and restart jobs later with `START_JOB`. To attach and restart jobs, the master table and dump file set must not be disturbed, either when you issue the command, or after you issue the command.

To perform an orderly shutdown, use `STOP_JOB` (without any associated value). A warning requiring confirmation is then issued. An orderly shutdown stops the job after worker processes have finished their current tasks.

To perform an immediate shutdown, specify `STOP_JOB=IMMEDIATE`. A warning requiring confirmation is then issued. All attached clients, including the one issuing the `STOP_JOB` command, receive a warning that the current user is stopping the job. They are then detached. After all clients are detached, the process structure of the job is immediately run down. That is, the Data Pump control job process does not wait for the worker processes to finish their current tasks. When you specify `STOP_JOB=IMMEDIATE`, there is no risk of corruption or data loss. However, you can be required to redo some tasks that were incomplete at the time of shutdown at restart time.

Example

```
Import> STOP_JOB=IMMEDIATE
```

3.6 Examples of Using Oracle Data Pump Import

You can use these common scenario examples to learn how you can use Oracle Data Pump Import to move your data.

- Performing a Data-Only Table-Mode Import
  See how to use Oracle Data Pump to perform a data-only table-mode import.
- Performing a Schema-Mode Import
  See how to use Oracle Data Pump to perform a schema-mode import.
- Performing a Network-Mode Import
  See how to use Oracle Data Pump to perform a network-mode import.
- Using Wildcards in URL-Based Dumpfile Names
  Oracle Data Pump simplifies importing multiple dump files into Oracle Autonomous Database from the Oracle Object Store Service by allowing wildcards for URL-based dumpfile names.

3.6.1 Performing a Data-Only Table-Mode Import

See how to use Oracle Data Pump to perform a data-only table-mode import.

In the example, the table is named `employees`. It uses the dump file created in "Performing a Table-Mode Export.".

The `CONTENT=DATA_ONLY` parameter filters out any database object definitions (metadata). Only table row data is loaded.

Example 3-1 Performing a Data-Only Table-Mode Import

```
> impdp hr TABLES=employees CONTENT=DATA_ONLY DUMPFILE=dpump_dir1:table.dmp
NOLOGFILE=YES
```

Related Topics

- Performing a Table-Mode Export

3.6.2 Performing a Schema-Mode Import

See how to use Oracle Data Pump to perform a schema-mode import.

The example is a schema-mode import of the dump file set created in "Performing a Schema-Mode Export".

Example 3-2 Performing a Schema-Mode Import

```
> impdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 DUMPFILE=expschema.dmp
 EXCLUDE=CONSTRAINT,REF_CONSTRAINT,INDEX TABLE_EXISTS_ACTION=REPLACE
```

The `EXCLUDE` parameter filters the metadata that is imported. 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, then all of its dependent objects are also excluded. The `TABLE_EXISTS_ACTION=REPLACE` parameter tells Import to drop the table if it already exists and to then re-create and load it using the dump file contents.

Related Topics

- Performing a Schema-Mode Export

3.6.3 Performing a Network-Mode Import

See how to use Oracle Data Pump to perform a network-mode import.

The network-mode import uses as its source the database specified by the `NETWORK_LINK` parameter.

Example 3-3 Network-Mode Import of Schemas

```
> impdp hr TABLES=employees REMAP_SCHEMA=hr:scott DIRECTORY=dpump_dir1
NETWORK_LINK=dblink
```

This example imports the `employees` table from the `hr` schema into the `scott` schema. The `dblink` references a source database that is different than the target database.

To remap the schema, user `hr` must have the `DATAPUMP_IMP_FULL_DATABASE` role on the local database and the `DATAPUMP_EXP_FULL_DATABASE` role on the source database.

`REMAP_SCHEMA` loads all the objects from the source schema into the target schema.

Related Topics

- NETWORK_LINK

3.6.4 Using Wildcards in URL-Based Dumpfile Names

Oracle Data Pump simplifies importing multiple dump files into Oracle Autonomous Database from the Oracle Object Store Service by allowing wildcards for URL-based dumpfile names.

Example 3-4 Wildcards Used in a URL-based Filename

This example shows how to use wildcards in the file name for importing multiple dump files into Oracle Autonomous Database from the Oracle Object Store Service.

```
> impdp admin/password@ATPC1_high
    directory=data_pump_dir credential=my_cred_name
    dumpfile= https://objectstorage.example.com/v1/atpc/atpc_user/exp%u.dmp"
```

> **Note:**
>
> You cannot use wildcard characters in the bucket-name component of the URL.

3.7 Syntax Diagrams for Oracle Data Pump Import

You can use syntax diagrams to understand the valid SQL syntax for Oracle Data Pump Import.

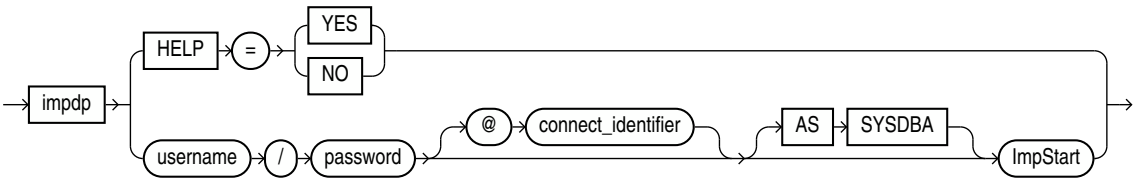
How to Read Graphic Syntax Diagrams

Syntax diagrams are drawings that illustrate valid SQL syntax. To read a diagram, trace it from left to right, in the direction shown by the arrows.

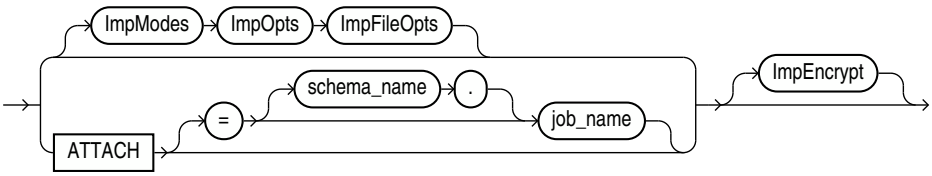
For more information about standard SQL syntax notation, see:

How to Read Syntax Diagrams in *Oracle Database SQL Language Reference*

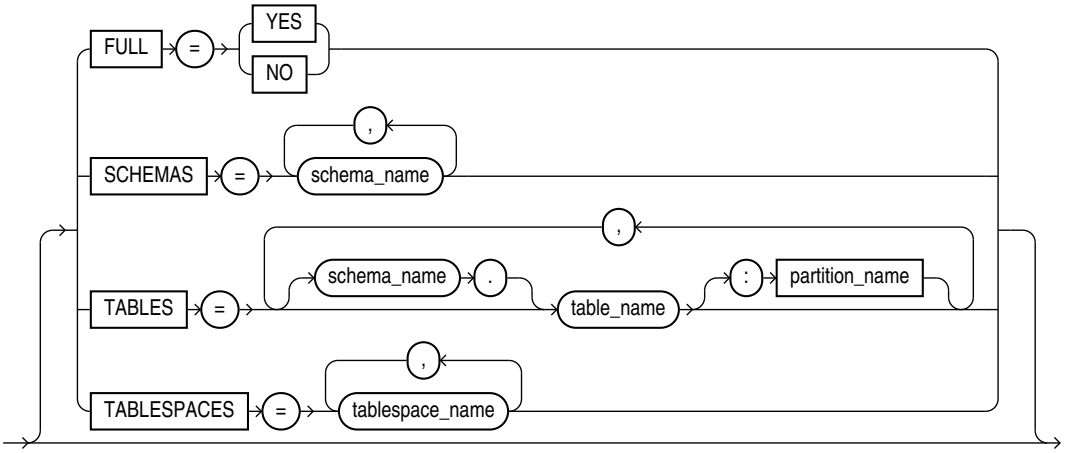
ImpInit



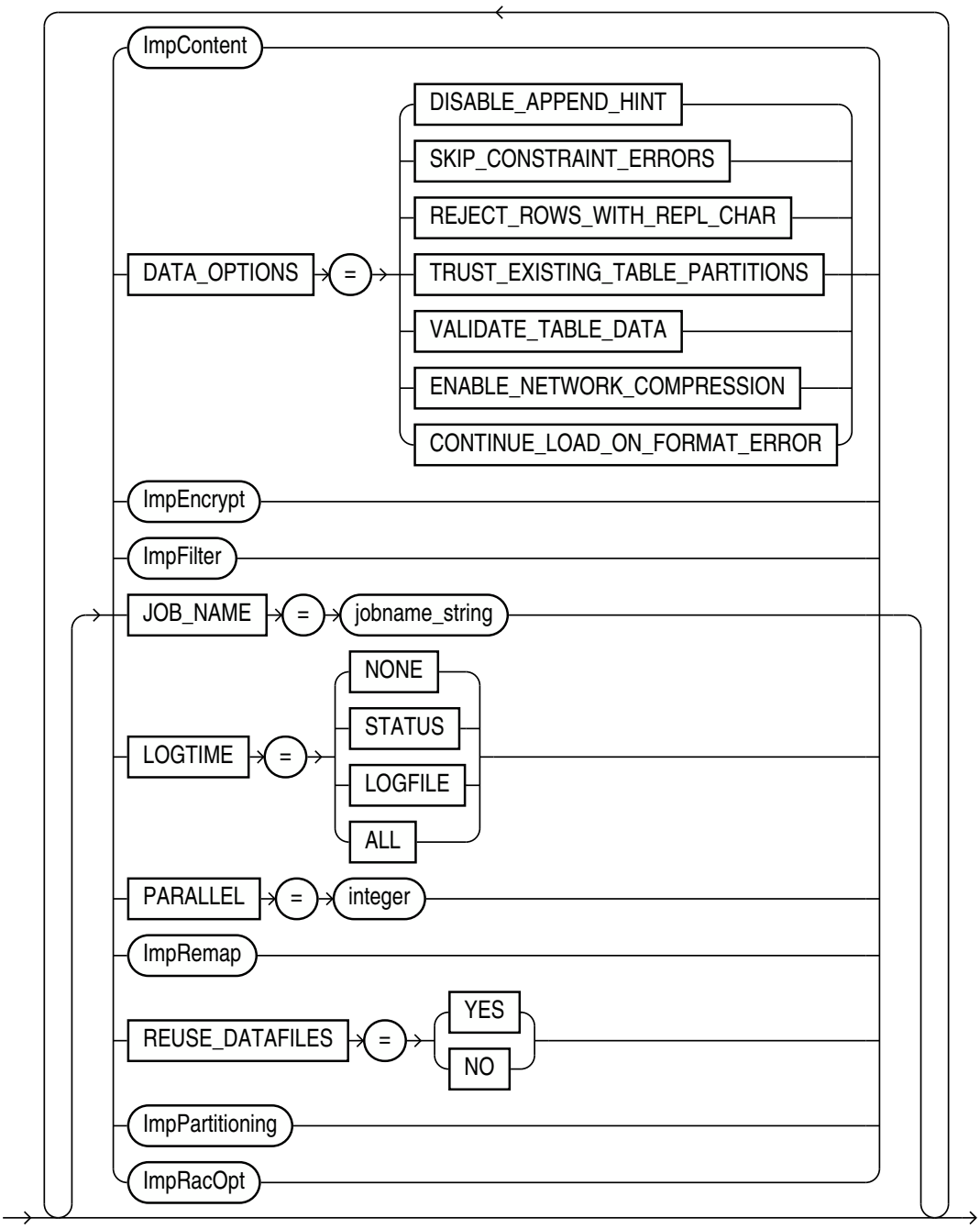
ImpStart



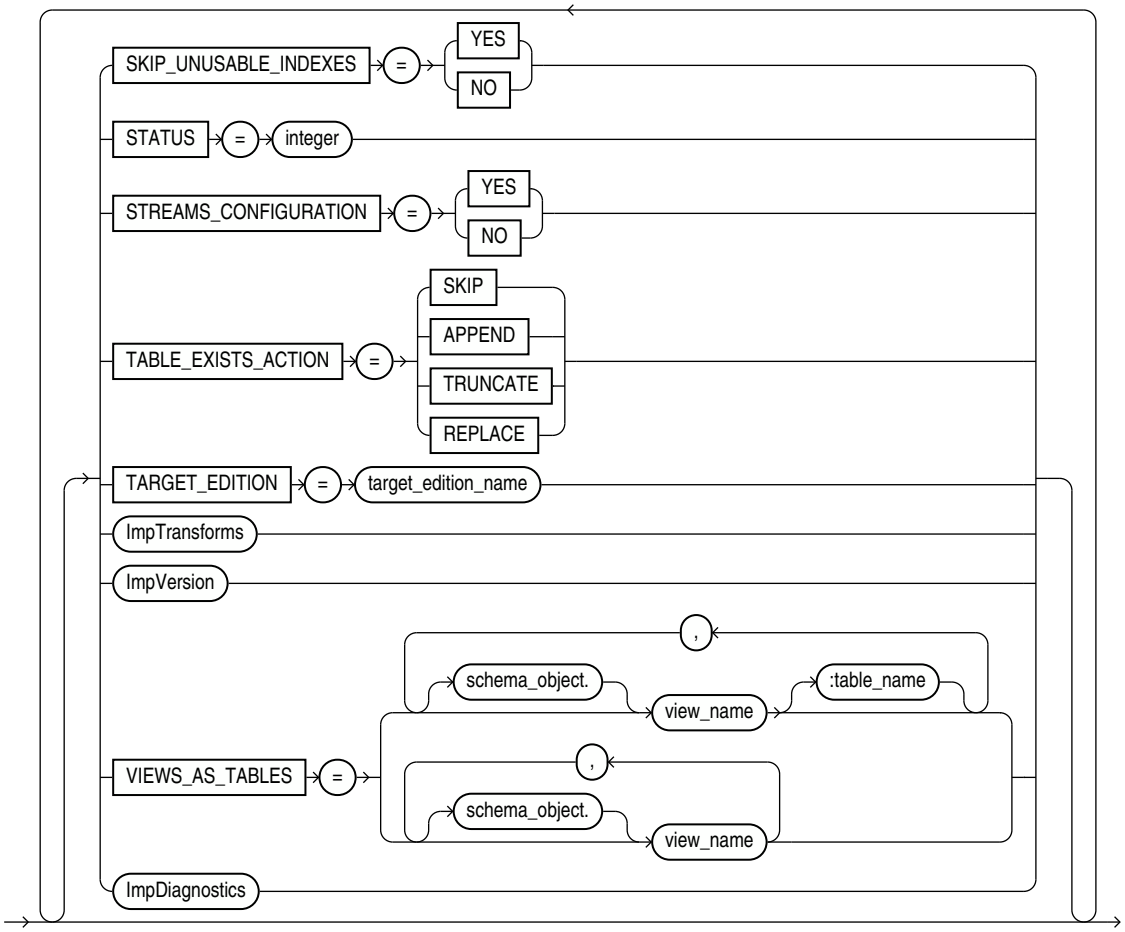
ImpModes



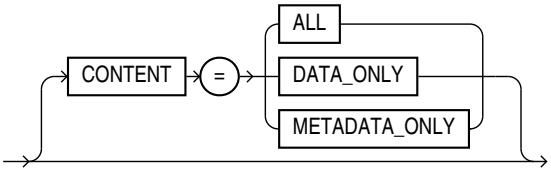
ImpOpts


ImpContent
DISABLE_APPEND_HINT
SKIP_CONSTRAINT_ERRORS
REJECT_ROWS_WITH_REPL_CHAR
DATA_OPTIONS
=
TRUST_EXISTING_TABLE_PARTITIONS
VALIDATE_TABLE_DATA
ENABLE_NETWORK_COMPRESSION
CONTINUE_LOAD_ON_FORMAT_ERROR
ImpEncrypt
ImpFilter
JOB_NAME
=
jobname_string
NONE
LOGTIME
=
STATUS
LOGFILE
ALL
PARALLEL
=
integer
ImpRemap
REUSE_DATAFILES
=
YES
NO
ImpPartitioning
ImpRacOpt

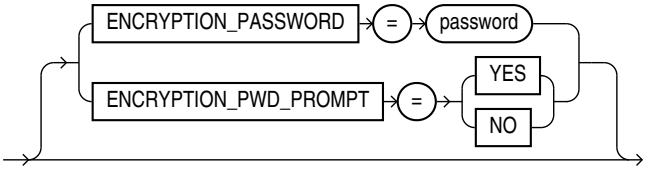
ImpOpts_Cont



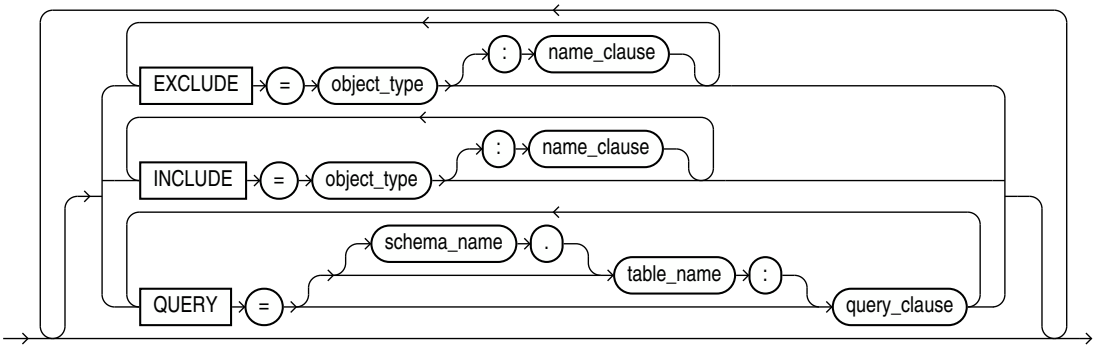
ImpContent



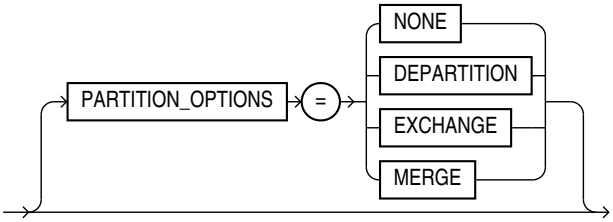
ImpEncrypt



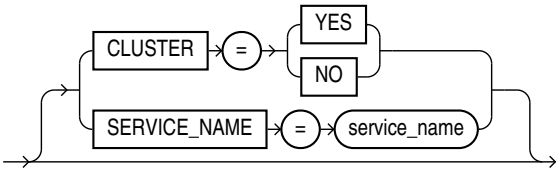
ImpFilter



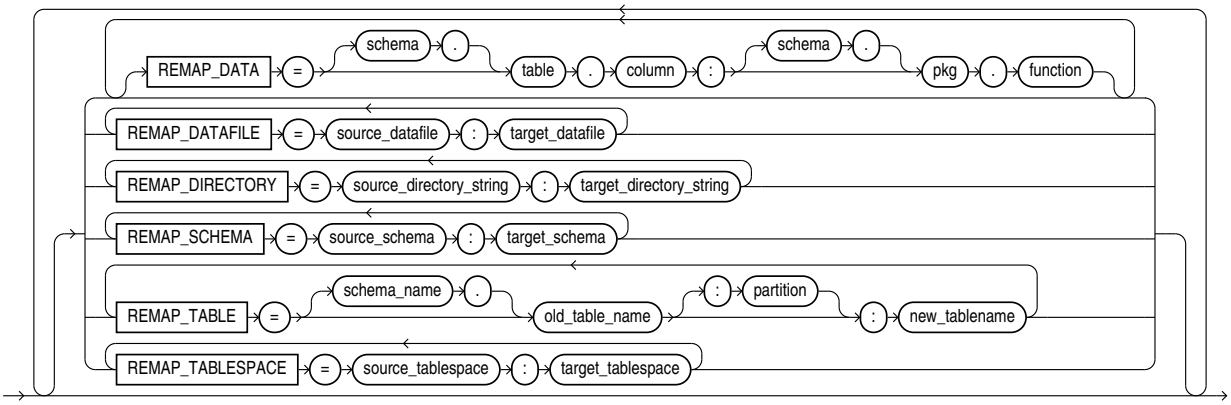
ImpPartitioning



ImpRacOpt

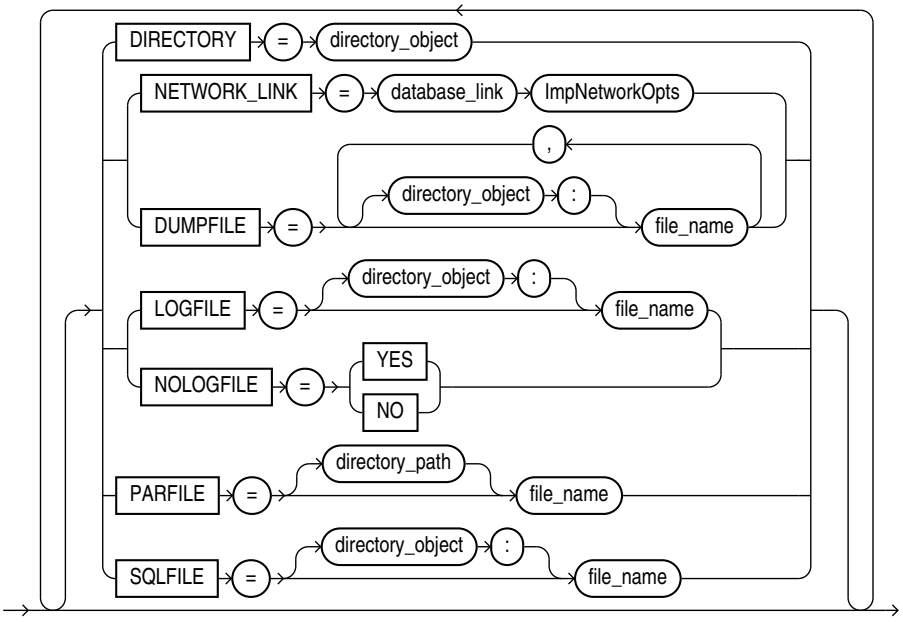


ImpRemap

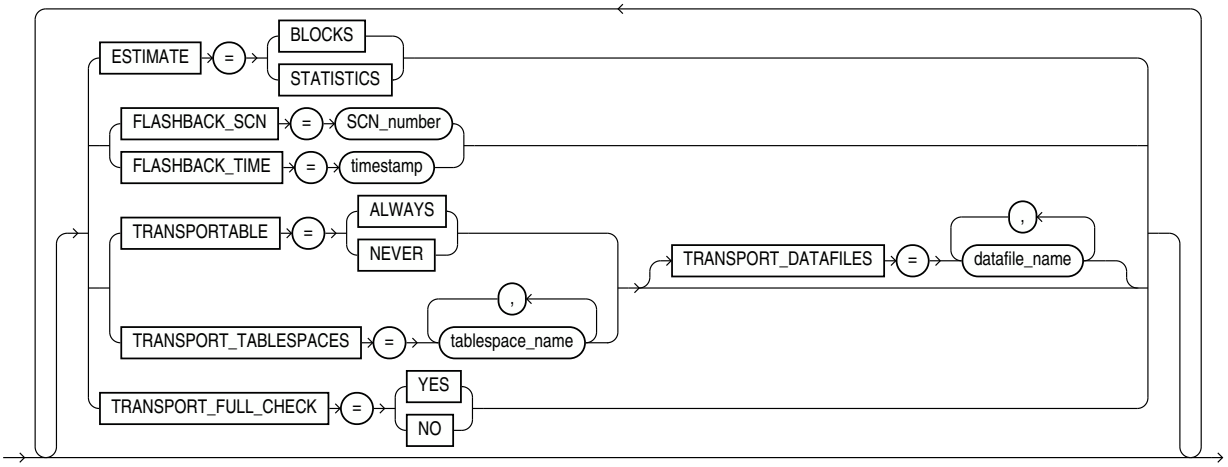


Note: The `REMAP_DATAFILE` and `REMAP_DIRECTORY` parameters are mutually exclusive.

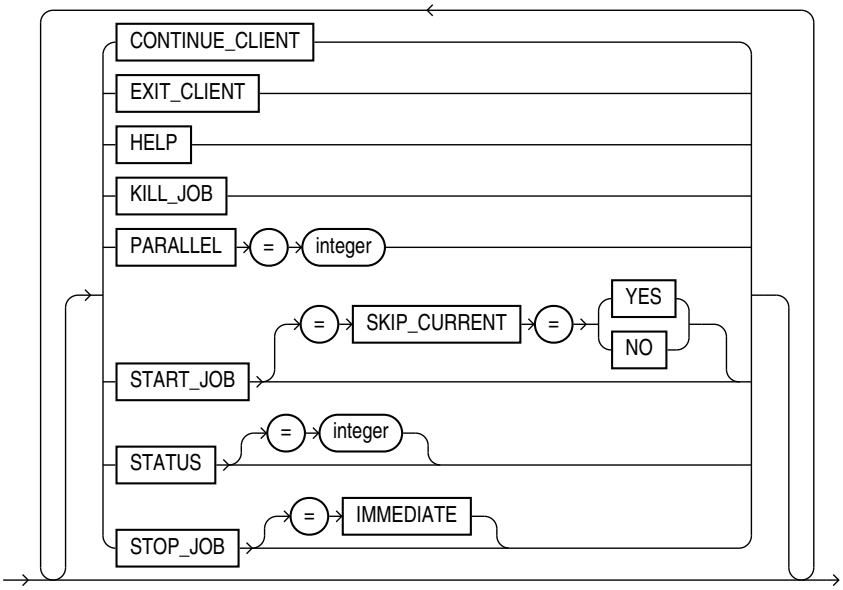
ImpFileOpts



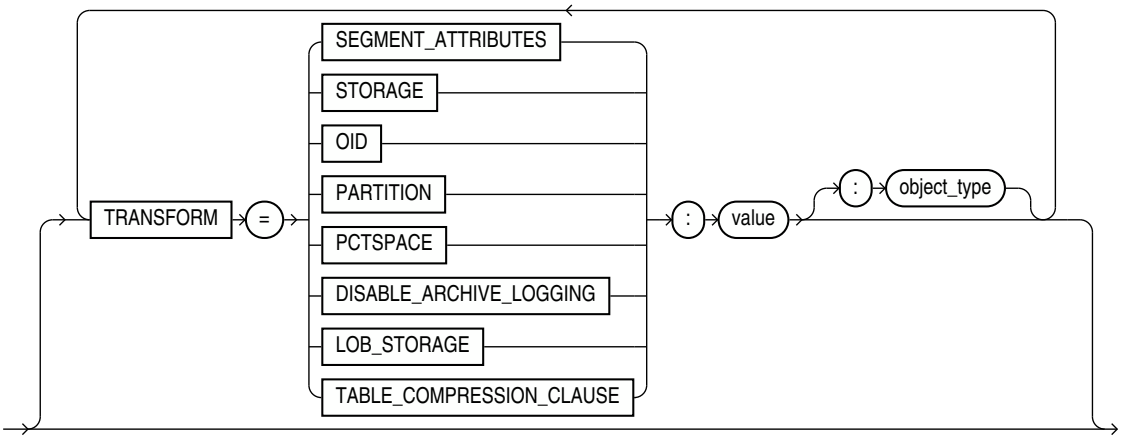
ImpNetworkOpts



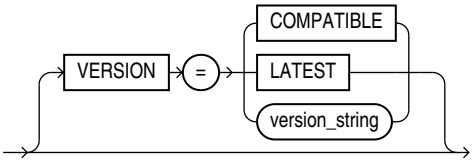
ImpDynOpts



ImpTransforms



ImpVersion



ImpDiagnostics