2

Oracle Data Pump Export

The Oracle Data Pump Export utility is used to unload data and metadata into a set of operating system files, which are called a dump file set.

- What Is Oracle Data Pump Export?
  Oracle Data Pump Export is a utility for unloading data and metadata into a set of operating system files that are called a **dump file set**.
- Starting Oracle Data Pump Export
  Start the Oracle Data Pump Export utility by using the `expdp` command.
- Filtering During Export Operations
  Oracle Data Pump Export provides data and metadata filtering capability. This capability helps you limit the type of information that is exported.
- Parameters Available in Data Pump Export Command-Line Mode
  Use Oracle Data Pump parameters for Export (`expdp`) to manage your data exports.
- 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.
- Examples of Using Oracle Data Pump Export
  You can use these common scenario examples to learn how you can create parameter files and use Oracle Data Pump Export to move your data.
- Syntax Diagrams for Oracle Data Pump Export
  You can use syntax diagrams to understand the valid SQL syntax for Oracle Data Pump Export.

2.1 What Is Oracle Data Pump Export?

Oracle Data Pump Export is a utility for unloading data and metadata into a set of operating system files that are called a **dump file set**.

You can import a dump file set only by using the Oracle Data Pump Import utility. You can import the dump file set on the same system, or import it to another system, and load the dump file set there.

The 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 import operation, the Oracle Data Pump Import utility uses these files to locate each database object in the dump file set.

Because the dump files are written by the server, rather than by the client, you must create directory objects that define the server locations to which files are written.

Oracle Data Pump Export enables you to specify that you want a job to move a subset of the data and metadata, as determined by the export mode. This subset selection is done by using data filters and metadata filters, which are specified through Oracle Data Pump Export parameters.

Note:

Several system schemas cannot be exported, because they are not user schemas; they contain Oracle-managed data and metadata. Examples of schemas that are not exported include `SYS`, `ORDSYS`, and `MDSYS`. Secondary objects are also not exported, because the `CREATE INDEX` run at import time will recreate them.

Related Topics

- Understanding Dump_ Log_ and SQL File Default Locations
- Filtering During Export Operations
- Export Utility (exp or expdp) does not Export DR${name}$% or DR#{name}$% Secondary Tables of Text Indexes (Doc ID 139388.1)
- Examples of Using Oracle Data Pump Export

2.2 Starting Oracle Data Pump Export

Start the Oracle Data Pump Export utility by using the `expdp` command.

The characteristics of the Oracle Data Pump export operation are determined by the Export parameters that you specify. You can specify these parameters either on the command line, or in a parameter file.

Caution:

Do not start Export 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.

- Oracle Data Pump Export Interfaces
  You can interact with Oracle Data Pump Export by using a command line, a parameter file, or an interactive-command mode.
- Oracle Data Pump Export Modes
  Export provides different modes for unloading different portions of Oracle Database data.
- Network Considerations for Oracle Data Pump Export
  Learn how Oracle Data Pump Export utility `expdp` identifies instances with connect identifiers in the connection string using Oracle*Net or a net service name, and how they are different from export operations using the `NETWORK_LINK` parameter.

2.2.1 Oracle Data Pump Export Interfaces

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

Choose among the three options:

- Command-Line Interface: Enables you to specify most of the Export parameters directly on the command line.

- 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. If you are using parameters whose values require quotation marks, then Oracle recommends that you use parameter files.
- Interactive-Command Interface: Stops logging to the terminal and displays the Export prompt, from which you can enter various commands, some of which are specific to interactive-command mode. This mode is enabled by pressing Ctrl+C during an export 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.

2.2.2 Oracle Data Pump Export Modes

Export provides different modes for unloading different portions of Oracle Database data.

Specify export modes on the command line, using the appropriate parameter.

Note:

You cannot export several Oracle-managed system schemas for Oracle Database, because they are not user schemas; they contain Oracle-managed data and metadata. Examples of system schemas that are not exported include `SYS`, `ORDSYS`, and `MDSYS`.

- Full Export Mode
  You can use Oracle Data Pump to carry out a full database export by using the `FULL` parameter.
- Schema Mode
  You can specify a schema export with Data Pump by using the `SCHEMAS` parameter. A schema export is the default export mode.
- Table Mode
  You can use Oracle Data Pump to carry out a table mode export by specifying the table using the `TABLES` parameter.
- Tablespace Mode
  You can use Data Pump to carry out a tablespace export by specifying tables using the `TABLESPACES` parameter.
- Transportable Tablespace Mode
  You can use Oracle Data Pump to carry out a transportable tablespace export by using the `TRANSPORT_TABLESPACES` parameter.

2.2.2.1 Full Export Mode

You can use Oracle Data Pump to carry out a full database export by using the `FULL` parameter.

In a full database export, the entire database is unloaded. This mode requires that you have the `DATAPUMP_EXP_FULL_DATABASE` role.

Using the Transportable Option During Full Mode Exports

If you specify the `TRANSPORTABLE=ALWAYS` parameter along with the `FULL` parameter, then Data Pump performs a full transportable export. A full transportable export exports all objects and data necessary to create a complete copy of the database. A mix of data movement methods is used:

- Objects residing in transportable tablespaces have only their metadata unloaded into the dump file set; the data itself is moved when you copy the data files to the target database. The data files that must be copied are listed at the end of the log file for the export operation.
- Objects residing in non-transportable tablespaces (for example, `SYSTEM` and `SYSAUX`) have both their metadata and data unloaded into the dump file set, using direct path unload and external tables.

Restrictions

Performing a full transportable export has the following restrictions:

- The user performing a full transportable export requires the `DATAPUMP_EXP_FULL_DATABASE` privilege.
- The default tablespace of the user performing the export must not be set to one of the tablespaces being transported.
- If the database being exported contains either encrypted tablespaces or tables with encrypted columns (either Transparent Data Encryption (TDE) columns or SecureFiles LOB columns), then the `ENCRYPTION_PASSWORD` parameter must also be supplied.
- The source and target databases must be on platforms with the same endianness if there are encrypted tablespaces in the source database.
- 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. You can use the `DBMS_FILE_TRANSFER` package or the `RMAN CONVERT` command to convert the data.
- All objects with storage that are selected for export must have all of their storage segments 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.
- When transporting a database over the network using full transportable export, 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.
- If both the source and target databases are running Oracle Database 12c, then to perform a full transportable export, either the Oracle Data Pump `VERSION` parameter must be set to at least 12.0. or the `COMPATIBLE` database initialization parameter must be set to at least 12.0 or later.

Full Exports from Oracle Database 11.2.0.3

Full transportable exports are supported from a source database running at least release 11.2.0.3. To run full transportable exports set the Oracle Data Pump `VERSION` parameter to at least 12.0, as shown in the following syntax example, where *`user_name`* is the user performing a full transportable export:

```
> expdp user_name FULL=y DUMPFILE=expdat.dmp DIRECTORY=data_pump_dir
      TRANSPORTABLE=always VERSION=12.0 LOGFILE=export.log
```

Full Exports and Imports Using Extensibility Filters

In the following example, you use a full export to copy just the `audit_trails` metadata and data from the source database to the target database:

```
> expdp user/pwd directory=mydir full=y include=AUDIT_TRAILS
> impdp user/pwd directory=mydir
```

If you have completed an export from the source database in Full mode, then you can also import just the audit trails from the full export:

```
> expdp user/pwd directory=mydir full=y
> impdp user/pwd directory=mydir include=AUDIT_TRAILS
```

To obtain a list of valid extensibility tags, use this query:

```
SELECT OBJECT_PATH FROM DATABASE_EXPORT_PATHS WHERE tag=1 ORDER BY 1;
```

Related Topics

- CONVERT
- Scenarios for Full Transportable Export/import

2.2.2.2 Schema Mode

You can specify a schema export with Data Pump by using the `SCHEMAS` parameter. A schema export is the default export mode.

If you have the `DATAPUMP_EXP_FULL_DATABASE` role, then you can specify a list of schemas, optionally including the schema definitions themselves and also system privilege grants to those schemas. If you do not have the `DATAPUMP_EXP_FULL_DATABASE` role, then you can export only your own schema.

The `SYS` schema cannot be used as a source schema for export jobs.

Cross-schema references are not exported unless the referenced schema is also specified in the list of schemas to be exported. For example, a trigger defined on a table within one of the specified schemas, but that resides in a schema not explicitly specified, is not exported. Also, external type definitions upon which tables in the specified schemas depend are not exported. In such a case, it is expected that the type definitions already exist in the target instance at import time.

Related Topics

- SCHEMAS

2.2.2.3 Table Mode

You can use Oracle Data Pump to carry out a table mode export by specifying the table using the `TABLES` parameter.

In table mode, only a specified set of tables, partitions, and their dependent objects are unloaded. Any object required to create the table, such as the owning schema, or types for columns, must already exist.

If you specify the `TRANSPORTABLE=ALWAYS` parameter with the `TABLES` parameter, then only object metadata is unloaded. To move the actual data, you copy the data files to the target database. This results in quicker export times. If you are moving data files between releases or platforms, then the data files need to be processed by Oracle Recovery Manager (RMAN).

You must have the `DATAPUMP_EXP_FULL_DATABASE` role to specify tables that are not in your own schema. Note that type definitions for columns are *not* exported in table mode. It is expected that the type definitions already exist in the target instance at import time. Also, as in schema exports, cross-schema references are not exported.

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) a Data Pump export dump file that contains the recovered objects. Refer to *Oracle Database Backup and Recovery Guide* for more information about transporting data across platforms.

Carrying out a table mode export has the following restriction:

- When using `TRANSPORTABLE=ALWAYS` parameter with the `TABLES` parameter, the `ENCRYPTION_PASSWORD` parameter must also be used if the table being exported contains encrypted columns, either Transparent Data Encryption (TDE) columns or SecureFiles LOB columns.

Related Topics

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

2.2.2.4 Tablespace Mode

You can use Data Pump to carry out a tablespace export by specifying tables using the `TABLESPACES` parameter.

In tablespace mode, only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, then its dependent objects are also unloaded. Both object metadata and data are unloaded. In tablespace mode, if any part of a table resides in the specified set, then that table and all of its dependent objects are exported. Privileged users get all tables. Unprivileged users get only the tables in their own schemas.

Related Topics

- TABLESPACES

2.2.2.5 Transportable Tablespace Mode

You can use Oracle Data Pump to carry out a transportable tablespace export by using the `TRANSPORT_TABLESPACES` parameter.

In transportable tablespace mode, only the metadata for the tables (and their dependent objects) within a specified set of tablespaces is exported. The tablespace data files are copied in a separate operation. Then, a transportable tablespace import is performed to import the dump file containing the metadata and to specify the data files to use.

Transportable tablespace mode requires that the specified tables be completely self-contained. That is, all storage segments of all tables (and their indexes) defined within the tablespace set must also be contained within the set. If there are self-containment violations, then Export identifies all of the problems without actually performing the export.

Type definitions for columns of tables in the specified tablespaces are exported and imported. The schemas owning those types must be present in the target instance.

Starting with Oracle Database 21c, transportable tablespace exports can be done with degrees of parallelism greater than 1.

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.

Using Oracle Data Pump to carry out a transportable tablespace export has the following restrictions:

- If any of the tablespaces being exported contains tables with encrypted columns, either Transparent Data Encryption (TDE) columns or SecureFiles LOB columns, then the `ENCRYPTION_PASSWORD` parameter must also be supplied..
- If any of the tablespaces being exported is encrypted, then the use of the `ENCRYPTION_PASSWORD` is optional but recommended. If the `ENCRYPTION_PASSWORD` is omitted in this case, then the following warning message is displayed:

  ```
  ORA-39396: Warning: exporting encrypted data using transportable option
  without password
  ```

  This warning points out that in order to successfully import such a transportable tablespace job, the target database wallet must contain a copy of the same database access key used in the source database when performing the export. Using the `ENCRYPTION_PASSWORD` parameter during the export and import eliminates this requirement.

Related Topics

- How Does Oracle Data Pump Handle Timestamp Data?

2.2.3 Network Considerations for Oracle Data Pump Export

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

When you start `expdp`, 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).

To specify a connect identifier manually by using either an Oracle*Net connect descriptor, or an Easy Connect identifier, or a net service name (usually defined in the `tnsnames.ora` file) that maps to a connect descriptor.

To use a connect identifier, you must 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:

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

Export 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:

```
expdp 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 Export client connects to the database instance defined by the connect identifier `inst1` (a Net service name), retrieves data from `inst1`, and writes it to the dump file `hr.dmp` on `inst1`.

Specifying a connect identifier when you start the Export utility is different from performing an export operation using the `NETWORK_LINK` parameter. When you start an export operation and specify a connect identifier, the local Export client connects to the database instance identified by the connect identifier, retrieves data from that database instance, and writes it to a dump file set on that database instance. By contrast, when you perform an export using the `NETWORK_LINK` parameter, the export is performed using 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.)

Related Topics

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

2.3 Filtering During Export Operations

Oracle Data Pump Export provides data and metadata filtering capability. This capability helps you limit the type of information that is exported.

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

2.3.1 Oracle Data Pump Export Data Filters

You can specify restrictions on the table rows that you export 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.

2.3.2 Oracle Data Pump Metadata Filters

To exclude or include objects in an export 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 Export or Import operation. For example, you can request a full export, but without Package Specifications or Package Bodies.

To use filters correctly and to obtain the results you expect, remember that dependent objects of an identified object are processed along with the identified object. For example, if a filter specifies that you want an index included in an operation, then statistics from that index are also included. Likewise, if a table is excluded by a filter, then indexes, constraints, grants, and triggers upon the table are also 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, such that the Oracle Data Pump job includes only objects identified as included. Then it processes the `EXCLUDE` parameters, which can further restrict the objects processed by the job. As the command runs, any objects specified by the `EXCLUDE` parameter that are in the list of `INCLUDE` objects are removed.

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

You can specify the same metadata filter name 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. For example, you could perform the following query:

```
SQL> SELECT OBJECT_PATH, COMMENTS FROM SCHEMA_EXPORT_OBJECTS
  2  WHERE OBJECT_PATH LIKE '%GRANT' AND OBJECT_PATH NOT LIKE '%/%';
```

The output of this query looks similar to the following:

```
OBJECT_PATH
------------------------------------------------------------------------------
--
COMMENTS
------------------------------------------------------------------------------
--
GRANT
Object grants on the selected tables

OBJECT_GRANT
Object grants on the selected tables

PROCDEPOBJ_GRANT
```

```
Grants on instance procedural objects

PROCOBJ_GRANT
Schema procedural object grants in the selected schemas

ROLE_GRANT
Role grants to users associated with the selected schemas

SYSTEM_GRANT
System privileges granted to users associated with the selected schemas
```

Related Topics

- EXCLUDE
- INCLUDE
- OPEN Function

2.4 Parameters Available in Data Pump Export Command-Line Mode

Use Oracle Data Pump parameters for Export (`expdp`) to manage your data exports.

- About Oracle Data Pump Export Parameters
  Learn how to use Oracle Data Pump Export 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 Export command-line utility `ABORT_STEP` parameter stops the job after it is initialized.
- ACCESS_METHOD
  The Oracle Data Pump Export command-line utility `ACCESS_METHOD` parameter instructs Export to use a particular method to unload data.
- ATTACH
  The Oracle Data Pump Export command-line utility `ATTACH` parameter attaches a worker or client session to an existing export job, and automatically places you in the interactive-command interface.
- CHECKSUM
  The Oracle Data Pump Export command-line utility `CHECKSUM` parameter enables the export to perform checksum validations for exports.
- CHECKSUM_ALGORITM
  The Oracle Data Pump Export command-line utility `CHECKSUM_ALGORITHM` parameter specifies which checksum algorithm to use when calculating checksums.
- CLUSTER
  The Oracle Data Pump Export command-line utility `CLUSTER` parameter determines whether Data Pump can use Oracle RAC, resources, and start workers on other Oracle RAC instances.
- COMPRESSION
  The Oracle Data Pump Export command-line utility `COMPRESSION` parameter specifies which data to compress before writing to the dump file set.

- COMPRESSION_ALGORITHM
  The Oracle Data Pump Export command-line utility `COMPRESSION_ALGORITHM` parameter specifies the compression algorithm that you want to use when compressing dump file data.
- CONTENT
  The Oracle Data Pump Export command-line utility `CONTENT` parameter enables you to filter what Export unloads: data only, metadata only, or both.
- CREDENTIAL
  The Oracle Data Pump Export command-line utility `CREDENTIAL` parameter enables the export to write data stored into object stores.
- DATA_OPTIONS
  The Oracle Data Pump Export command-line utility `DATA_OPTIONS` parameter designates how you want certain types of data handled during export operations.
- DIRECTORY
  The Oracle Data Pump Export command-line utility `DIRECTORY` parameter specifies the default location to which Export can write the dump file set and the log file.
- DUMPFILE
  The Oracle Data Pump Export command-line utility `DUMPFILE` parameter specifies the names, and optionally, the directory objects of dump files for an export job.
- ENABLE_SECURE_ROLES
  The Oracle Data Pump Export command-line utility `ENABLE_SECURE_ROLES` parameter prevents inadvertent use of protected roles during exports.
- ENCRYPTION
  The Oracle Data Pump Export command-line utility `ENCRYPTION` parameter specifies whether to encrypt data before writing it to the dump file set.
- ENCRYPTION_ALGORITHM
  The Oracle Data Pump Export command-line utility `ENCRYPTION_ALGORITHM` parameter specifies which cryptographic algorithm should be used to perform the encryption.
- ENCRYPTION_MODE
  The Oracle Data Pump Export command-line utility `ENCRYPTION_MODE` parameter specifies the type of security to use when encryption and decryption are performed.
- ENCRYPTION_PASSWORD
  The Oracle Data Pump Export command-line utility `ENCRYPTION_PASSWORD` parameter prevents unauthorized access to an encrypted dump file set.
- ENCRYPTION_PWD_PROMPT
  The Oracle Data Pump Export command-line utility `ENCRYPTION_PWD_PROMPT` specifies whether Oracle Data Pump prompts you for the encryption password.
- ESTIMATE
  The Oracle Data Pump Export command-line utility `ESTIMATE` parameter specifies the method that Export uses to estimate how much disk space each table in the export job will consume (in bytes).
- ESTIMATE_ONLY
  The Oracle Data Pump Export command-line utility `ESTIMATE_ONLY` parameter instructs Export to estimate the space that a job consumes, without actually performing the export operation.

- EXCLUDE
  The Oracle Data Pump Export command-line utility `EXCLUDE` parameter enables you to filter the metadata that is exported by specifying objects and object types that you want to exclude from the export operation.
- FILESIZE
  The Oracle Data Pump Export command-line utility `FILESIZE` parameter specifies the maximum size of each dump file.
- FLASHBACK_SCN
  The Oracle Data Pump Export command-line utility `FLASHBACK_SCN` parameter specifies the system change number (SCN) that Export uses to enable the Flashback Query utility.
- FLASHBACK_TIME
  The Oracle Data Pump Export command-line utility `FLASHBACK_TIME` parameter finds the SCN that most closely matches the specified time.
- FULL
  The Oracle Data Pump Export command-line utility `FULL` parameter specifies that you want to perform a full database mode export.
- HELP
  The Oracle Data Pump Export command-line utility `HELP` parameter displays online help for the Export utility.
- INCLUDE
  The Oracle Data Pump Export command-line utility `INCLUDE` parameter enables you to filter the metadata that is exported by specifying objects and object types for the current export mode.
- JOB_NAME
  The Oracle Data Pump Export command-line utility `JOB_NAME` parameter identifies the export job in subsequent actions.
- KEEP_MASTER
  The Oracle Data Pump Export command-line utility `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 Export command-line utility `LOGFILE` parameter specifies the name, and optionally, a directory, for the log file of the export job.
- LOGTIME
  The Oracle Data Pump Export command-line utility `LOGTIME` parameter specifies that messages displayed during export operations are timestamped.
- METRICS
  The Oracle Data Pump Export command-line utility `METRICS` parameter indicates whether you want additional information about the job reported to the Data Pump log file.
- NETWORK_LINK
  The Oracle Data Pump Export command-line utility `NETWORK_LINK` parameter enables an export from a (source) database identified by a valid database link.
- NOLOGFILE
  The Oracle Data Pump Export command-line utility `NOLOGFILE` parameter specifies whether to suppress creation of a log file.
- PARALLEL
  The Oracle Data Pump Export command-line utility `PARALLEL` parameter specifies the maximum number of processes of active execution operating on behalf of the export job.

- PARALLEL_THRESHOLD
  The Oracle Data Pump Export 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 Export command-line utility `PARFILE` parameter specifies the name of an export parameter file.
- QUERY
  The Oracle Data Pump Export command-line utility `QUERY` parameter enables you to specify a query clause that is used to filter the data that gets exported.
- REMAP_DATA
  The Oracle Data Pump Export command-line utility `REMAP_DATA` parameter enables you to specify a remap function that takes as a source the original value of the designated column and returns a remapped value that replaces the original value in the dump file.
- REUSE_DUMPFILES
  The Oracle Data Pump Export command-line utility `REUSE_DUMPFILES` parameter specifies whether to overwrite a preexisting dump file.
- 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.
- SCHEMAS
  The Oracle Data Pump Export command-line utility `SCHEMAS` parameter specifies that you want to perform a schema-mode export.
- SERVICE_NAME
  The Oracle Data Pump Export command-line utility `SERVICE_NAME` parameter specifies a service name that you want to use in conjunction with the `CLUSTER` parameter.
- SOURCE_EDITION
  The Oracle Data Pump Export command-line utility `SOURCE_EDITION` parameter specifies the database edition from which objects are exported.
- STATUS
  The Oracle Data Pump Export command-line utility `STATUS` parameter specifies the frequency at which the job status display is updated.
- TABLES
  The Oracle Data Pump Export command-line utility `TABLES` parameter specifies that you want to perform a table-mode export.
- TABLESPACES
  The Oracle Data Pump Export command-line utility `TABLESPACES` parameter specifies a list of tablespace names that you want to be exported in tablespace mode.
- TRANSPORT_DATAFILES_LOG
  The Oracle Data Pump Export command-line mode `TRANSPORT_DATAFILES_LOG` parameter specifies a file into which the list of data files associated with a transportable export is written.
- TRANSPORT_FULL_CHECK
  The Oracle Data Pump Export command-line utility `TRANSPORT_FULL_CHECK` parameter specifies whether to check for dependencies between objects

- TRANSPORT_TABLESPACES
  The Oracle Data Pump Export command-line utility `TRANSPORT_TABLESPACES` parameter specifies that you want to perform an export in transportable-tablespace mode.
- TRANSPORTABLE
  The Oracle Data Pump Export command-line utility `TRANSPORTABLE` parameter specifies whether the transportable option should be used during a table mode or full mode export.
- TTS_CLOSURE_CHECK
  The Oracle Data Pump Export command-line mode `TTS_CLOSURE_CHECK` parameter is used to indicate the degree of closure checking to be performed as part of a Data Pump transportable tablespace operation.
- VERSION
  The Oracle Data Pump Export command-line utility `VERSION` parameter specifies the version of database objects that you want to export.
- VIEWS_AS_TABLES
  The Oracle Data Pump Export command-line utility `VIEWS_AS_TABLES` parameter specifies that you want one or more views exported as tables.

2.4.1 About Oracle Data Pump Export Parameters

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

Use these examples to understand how you can use Oracle Data Pump Export at the command line.

Specifying Export Parameters

For parameters that can have multiple values specified, you can specify the values by commas, or by spaces. For example, you can 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, Export interprets the string as another dump file name for the `DUMPFILE` parameter:

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

This command results in two dump files being created, `test.dmp` and `nologfile.dmp`.

To avoid this result, 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. These operating systems therefore do not pass quotation marks on to an application unless quotation marks are preceded by an escape character, such as the backslash (\). This requirement is true both on the command lin, and within parameter files. Some operating systems can 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. Note that your particular operating system can have different requirements. The documentation examples cannot fully anticipate operating environments, which are unique to each user.

In this example, the `TABLES` parameter is specified in a parameter file:

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

If you specify that value on the command line, then some operating systems require that you surround the parameter file name using single quotation marks, as follows:

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

To avoid having to supply more 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 some operating systems do not require the use of escape characters.

Using the Export Parameter Examples

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

- After you enter the user name and parameters as shown in the example, Export is started, and you are prompted for a password. You are required to enter the 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.
- The examples assume that the directory objects, `dpump_dir1` and `dpump_dir2`, already exist, and that `READ` and `WRITE` privileges are 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 is 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, you can also specify these parameters in a parameter file.

Related Topics

- Introduction to Sample Schemas

See Also:

Your operating system-specific documentation for information about how special and reserved characters are handled on your system

2.4.2 ABORT_STEP

The Oracle Data Pump Export command-line utility `ABORT_STEP` parameter stops the job after it is initialized.

Default

Null

Purpose

Used to stop the job after it is initialized. Stopping a job after it is initialized enables you to query the Data Pump control job table that you want to query before any data is exported.

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 export operation is started, and the job is stopped at the object that is stored in the Data Pump control job table with the corresponding process order number.
- `-1`: If the value is negative one (`-1`), then abort the job after setting it up, but before exporting any objects or data.

Restrictions

- None

Example

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr ABORT_STEP=-1
```

2.4.3 ACCESS_METHOD

The Oracle Data Pump Export command-line utility `ACCESS_METHOD` parameter instructs Export to use a particular method to unload data.

Default

`AUTOMATIC`

Purpose

Instructs Export to use a particular method to unload data.

Syntax and Description

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

The `ACCESS_METHOD` parameter is provided so that you can try an alternative method if the default method does not work for some reason. All methods can be specified for a network export. If the data for a table cannot be unloaded 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 as follows:

- `AUTOMATIC` — Oracle Data Pump determines the best way to unload data for each table. Oracle recommends that you use `AUTOMATIC` whenever possible because it allows Data Pump to automatically select the most efficient method.
- `DIRECT_PATH` — Oracle Data Pump uses direct path unload for every table.
- `EXTERNAL_TABLE` — Oracle Data Pump uses a SQL `CREATE TABLE AS SELECT` statement to create an external table using data that is stored in the dump file. The `SELECT` clause reads from the table to be unloaded.

Restrictions

- To use the `ACCESS_METHOD` parameter with network exports, you must be using Oracle Database 12c Release 2 (12.2.0.1) or later.
- The `ACCESS_METHOD` parameter for Oracle Data Pump Export is not valid for transportable tablespace jobs.

Example

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr
ACCESS_METHOD=EXTERNAL_TABLE
```

2.4.4 ATTACH

The Oracle Data Pump Export command-line utility `ATTACH` parameter attaches a worker or client session to an existing export job, and automatically places you in the interactive-command interface.

Default

The default is the job currently in the user schema, if there is only one.

Purpose

Attaches the worker session to an existing Data Pump control export job, and automatically places you in the interactive-command interface. Export displays a description of the job to which you are attached, and also displays the Export prompt.

Syntax and Description

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

The *`schema_name`* is optional. To specify a schema other than your own, you must have the `DATAPUMP_EXP_FULL_DATABASE` role.

The *job_name* is optional if only one export job is associated with your schema and the job is active. To attach to a stopped job, you must supply the job name. To see a list of 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, Export displays a description of the job and then displays the Export prompt.

Restrictions

- When you specify the ATTACH parameter, the only other Data Pump parameter you can specify on the command line is ENCRYPTION_PASSWORD.
- If the job to which you are attaching 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 respecify that password. The only exception to this requirement is if the job was initially started with the ENCRYPTION=ENCRYPTED_COLUMNS_ONLY parameter. In that case, the encryption password is not needed when attaching to the job.
- You cannot attach to a job in another schema unless it is already running.
- If the dump file set or Data Pump control table for the job have been deleted, then the attach operation fails.
- Altering the Data Pump control table in any way leads to unpredictable results.

Example

The following is an example of using the ATTACH parameter. It assumes that the job hr.export_job is an existing job.

```
> expdp hr ATTACH=hr.export_job
```

2.4.5 CHECKSUM

The Oracle Data Pump Export command-line utility CHECKSUM parameter enables the export to perform checksum validations for exports.

Default

The default value depends upon the combination of checksum-related parameters that are used. To enable checksums, you must specify either the CHECKSUM or the CHECKSUM_ALGORITHM parameter.

If you specify only the CHECKSUM_ALGORITHM parameter, then CHECKSUM defaults to YES.

If you specify neither the CHECKSUM nor the CHECKSUM_ALGORITHM parameters, then CHECKSUM defaults to NO.

Purpose

Specifies whether Oracle Data Pump calculates checksums for the export dump file set.

The checksum is calculated at the end of the job, so the time scales according to the size of the file. Multiple files can be processed in parallel. You can use this parameter to validate that a dumpfile is complete and not corrupted after copying it over the network to an object store, or using it to validate an old dumpfile.

Syntax and Description

```
CHECKSUM=[YES|NO]
```

- `YES` Specifies that Oracle Data Pump calculates a file checksum for each dump file in the export dump file set.
- `NO` Specifies that Oacle Data Pump does not calculate file checksums.

Restrictions

To use this checksum feature, the `COMPATIBLE` initialization parameter must be set to at least `20.0`.

Example

This example performs a schema-mode unload of the `HR` schema, and generates an `SHA256` (the default `CHECKSUM_ALGORITHM`) checksum for each dump file in the dump file set.

```
expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp CHECKSUM=YES
```

2.4.6 CHECKSUM_ALGORITM

The Oracle Data Pump Export command-line utility `CHECKSUM_ALGORITHM` parameter specifies which checksum algorithm to use when calculating checksums.

Default

The default value depends upon the combination of checksum-related parameters that are used. To enable checksums, you must specify either the `CHECKSUM` or the `CHECKSUM_ALGORITHM` parameter.

If the `CHECKSUM` parameter is set to `YES`, and you have not specified a value for `CHECKSUM_ALGORITHM`, then `CHECKSUM_ALGORITHM` defaults to the `SHA256` Secure Hash Algorithm.

Purpose

Helps to ensure the integrity of the contents of a dump file beyond the header block by using a cryptographic hash to ensure that there are no unintentional errors in a dump file, such as can occur with a transmission error. Setting the value specifies whether Oracle Data Pump calculates checksums for the export dump file set, and which hash algorithm is used to calculate the checksum.

Syntax and Description

```
CHECKSUM_ALGORITHM = [CRC32|SHA256|SHA384|SHA512]
```

- `CRC32` Specifies that Oracle Data Pump genrerates a 32-bit checksum.
- `SHA256` Specifies that Oracle Data Pump generates a 256-bit checksum.
- `SHA384` Specifies that Oracle Data Pump generates a 384-bit checksum.
- `SHA512` Specifies that Oracle Data Pump generates a 512-bit checksum.

Restrictions

To use this checksum feature, the `COMPATIBLE` initialization parameter must be set to at least `20.0`.

Example

This example performs a schema-mode unload of the HR schema, and generates an `SHA384` checksum for each dump file in the dump file set that is generated.

```
expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp CHECKSUM_ALGORITHM=SHA384
```

2.4.7 CLUSTER

The Oracle Data Pump Export command-line utility `CLUSTER` parameter determines whether Data Pump can use 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 Oracle Data Pump Export 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.

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

Example

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

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_clus%U.dmp CLUSTER=NO PARALLEL=3
```

This example starts a schema-mode export (the default) of the `hr` schema. Because `CLUSTER=NO` is specified, the job uses only the instance on which it started. (If you do not specify the `CLUSTER` parameter, then the default value of `Y` is used. With that value, if necessary, workers are started on other instances in the Oracle RAC cluster). The dump files are written to the location specified for the `dpump_dir1` directory object. The job can have up to 3 parallel processes.

Related Topics

- SERVICE_NAME
- Understanding How to Use Oracle Data Pump with Oracle RAC

2.4.8 COMPRESSION

The Oracle Data Pump Export command-line utility `COMPRESSION` parameter specifies which data to compress before writing to the dump file set.

Default

`METADATA_ONLY`

Purpose

Specifies which data to compress before writing to the dump file set.

Syntax and Description

```
COMPRESSION=[ALL | DATA_ONLY | METADATA_ONLY | NONE]
```

- `ALL` enables compression for the entire export operation. The `ALL` option requires that the Oracle Advanced Compression option is enabled.
- `DATA_ONLY` results in all data being written to the dump file in compressed format. The `DATA_ONLY` option requires that the Oracle Advanced Compression option is enabled.
- `METADATA_ONLY` results in all metadata being written to the dump file in compressed format. This is the default.
- `NONE` disables compression for the entire export operation.

Restrictions

- To make full use of all these compression options, the `COMPATIBLE` initialization parameter must be set to at least 11.0.0.
- The `METADATA_ONLY` option can be used even if the `COMPATIBLE` initialization parameter is set to 10.2.
- Compression of data using `ALL` or `DATA_ONLY` is valid only in the Enterprise Edition of Oracle Database 11g or later, and requires that the Oracle Advanced Compression option is enabled.

Example

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

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_comp.dmp COMPRESSION=METADATA_ONLY
```

This command runs a schema-mode export that compresses all metadata before writing it out to the dump file, `hr_comp.dmp`. It defaults to a schema-mode export, because no export mode is specified.

See *Oracle Database Licensing Information* for information about licensing requirements for the Oracle Advanced Compression option.

Related Topics

- Oracle Database Options and Their Permitted Features

2.4.9 COMPRESSION_ALGORITHM

The Oracle Data Pump Export command-line utility `COMPRESSION_ALGORITHM` parameter specifies the compression algorithm that you want to use when compressing dump file data.

Default

`BASIC`

Purpose

Specifies the compression algorithm to be used when compressing dump file data.

Syntax and Description

```
COMPRESSION_ALGORITHM = [BASIC | LOW | MEDIUM | HIGH]
```

The parameter options are defined as follows:

- `BASIC`: Offers a good combination of compression ratios and speed; the algorithm used is the same as in previous versions of Oracle Data Pump.
- `LOW`: Least impact on export throughput. This option is suited for environments where CPU resources are the limiting factor.
- `MEDIUM`: Recommended for most environments. This option, like the `BASIC` option, provides a good combination of compression ratios and speed, but it uses a different algorithm than `BASIC`.
- `HIGH`: Best suited for situations in which dump files are copied over slower networks, where the limiting factor is network speed.

You characterize the performance of a compression algorithm by its CPU usage, and by the compression ratio (the size of the compressed output as a percentage of the uncompressed input). These measures vary, based on the size and type of inputs, as well as the speed of the compression algorithms used. The compression ratio generally increases from low to high, with a trade-off of potentially consuming more CPU resources.

Oracle recommends that you run tests with the different compression levels on the data in your environment. Choosing a compression level based on your environment, workload characteristics, and size and type of data is the only way to ensure that the exported dump file set compression level meets your performance and storage requirements.

Restrictions

- To use this feature, database compatibility must be set to 12.0.0 or later.
- This feature requires that you have the Oracle Advanced Compression option enabled.

Example 1

This example performs a schema-mode unload of the `HR` schema, and compresses only the table data using a compression algorithm with a low level of compression. Using this command option can result in fewer CPU resources being used, at the expense of a less than optimal compression ratio.

```
    > expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp COMPRESSION=DATA_ONLY
COMPRESSION_ALGORITHM=LOW
```

Example 2

This example performs a schema-mode unload of the `HR` schema, and compresses both metadata and table data using the basic level of compression. Omitting the `COMPRESSION_ALGORITHM` parameter altogether is equivalent to specifying `BASIC` as the value.

```
    > expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp COMPRESSION=ALL
COMPRESSION_ALGORITHM=BASIC
```

2.4.10 CONTENT

The Oracle Data Pump Export command-line utility `CONTENT` parameter enables you to filter what Export unloads: data only, metadata only, or both.

Default

`ALL`

Purpose

Enables you to filter what Export unloads: data only, metadata only, or both.

Syntax and Description

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

- `ALL` unloads both data and metadata. This option is the default.
- `DATA_ONLY` unloads only table row data; no database object definitions are unloaded.
- `METADATA_ONLY` unloads only database object definitions; no table row data is unloaded. Be aware that if you specify `CONTENT=METADATA_ONLY`, then afterward, when the dump file is imported, any index or table statistics imported from the dump file are locked after the import.

Restrictions

- The `CONTENT=METADATA_ONLY` parameter cannot be used with the `TRANSPORT_TABLESPACES` (transportable-tablespace mode) parameter or with the `QUERY` parameter.

Example

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

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp CONTENT=METADATA_ONLY
```

This command executes a schema-mode export that unloads only the metadata associated with the `hr` schema. It defaults to a schema-mode export of the `hr` schema, because no export mode is specified.

2.4.11 CREDENTIAL

The Oracle Data Pump Export command-line utility `CREDENTIAL` parameter enables the export to write data stored into object stores.

Default

none.

Purpose

Enables Oracle Data Pump exports to write data files to object stores. For a data file, you can specify the URI for the data file that you want to be stored on the object store. The `CREDENTIAL` values specifies credentials granted to the user starting the export. These permissions enable the Oracle Data Pump export to access and write to the object store, so that data files can be written to Oracle Cloud Infrastructure object stores.

Syntax and Description

```
CREDENTIAL=user-credential
```

Usage Notes

The `CREDENTIAL` parameter changes how `expdp` interprets the text string in `DUMPFILE`. If the `CREDENTIAL` parameter is not specified, then the `DUMPFILE` parameter can specify an optional directory object and file name in *`directory-object-name`*`:`*`file-name`* format. If the `CREDENTIAL` parameter is used, then it provides authentication and authorization for `expdp` to write to one or more object storage URIs specified by `DUMPFILE`.

If you do not specify the `CREDENTIAL` parameter, then the dumpfile value is not treated as a URI, but instead treated as a file specification. The dumpfile specification only contains the file name; it cannot contain a path. As a result, if you do not specify the `CREDENTIAL` parameter, then you receive the following errors:

```
ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39088: file name cannot contain a path specification
```

Restrictions

- The credential parameter cannot be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account.
- For Cloud systems, `UTIL_FILE` does not support writing to the cloud. In that case, the export continues to use the value set by the `DEFAULT_DIRECTORY` parameter as the location of the log files. Also, you can specify directory object names as part of the file names for `LOGFILE`.
- If you attempt to specify a URI for a dump file, and the `CREDENTIAL` parameter is not specified, then you encounter the error `ORA-39000 bad dumpfile specification`, as shown in the preceding usage notes.

Examples

The following example provides a credential, "`sales-dept`" and `DUMPFILE` specifies an Object Storage URI in which to export:

```
expdp hr DUMPFILE=https://objectstorage.example.com/images_basic.dmp
CREDENTIAL=sales-dept
```

The following example does not specify a credential:

```
expdp hr DUMPFILE=dir obj:filename
```

2.4.12 DATA_OPTIONS

The Oracle Data Pump Export command-line utility `DATA_OPTIONS` parameter designates how you want certain types of data handled during export operations.

Default

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

Purpose

The `DATA_OPTIONS` parameter designates how certain types of data should be handled during export operations.

Syntax and Description

```
DATA_OPTIONS= [GROUP_PARTITION_TABLE_DATA | VERIFY_STREAM_FORMAT]
```

- `GROUP_PARTITION_TABLE_DATA`: Tells Oracle Data Pump to unload all table data in one operation rather than unload each table partition as a separate operation. As a result, the definition of the table will not matter at import time because Import will see one partition of data that will be loaded into the entire table.
- `VERIFY_STREAM_FORMAT`: Validates the format of a data stream before it is written to the Oracle Data Pump dump file. The verification checks for a valid format for the stream after it is generated but before it is written to disk. This assures that there are no errors when the dump file is created, which in turn helps to assure that there will not be errors when the stream is read at import time.

Restrictions

The Export `DATA_OPTIONS` parameter requires the job version to be set to `11.0.0` or later. See `VERSION`.

Example

This example shows an export operation in which data for all partitions of a table are unloaded together instead of the default behavior of unloading the data for each partition separately.

```
> expdp hr TABLES=hr.tab1 DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp VERSION=11.2
GROUP_PARTITION_TABLE_DATA
```

See *Oracle XML DB Developer's Guide* for information specific to exporting and importing `XMLType` tables.

Related Topics

- VERSION

2.4.13 DIRECTORY

The Oracle Data Pump Export command-line utility `DIRECTORY` parameter specifies the default location to which Export can write the dump file set and the log file.

Default

`DATA_PUMP_DIR`

Purpose

Specifies the default location to which Export can write the dump file set and the log file.

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` or `LOGFILE` parameter overrides any directory object that you specify for the `DIRECTORY` parameter.

Example

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

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=employees.dmp CONTENT=METADATA_ONLY
```

In this example, the dump file, `employees.dump` is written to the path that is associated with the directory object `dpump_dir1`.

Related Topics

- Understanding Dump_ Log_ and SQL File Default Locations
- Understanding How to Use Oracle Data Pump with Oracle RAC

- *Oracle Database SQL Language Reference*

2.4.14 DUMPFILE

The Oracle Data Pump Export command-line utility `DUMPFILE` parameter specifies the names, and optionally, the directory objects of dump files for an export job.

Default

`expdat.dmp`

Purpose

Specifies the names, and if you choose to do so, the directory objects of dump files for an export job.

Syntax and Description

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

Specifying *`directory_object`* is optional if you have already specified the directory object by using the `DIRECTORY` parameter. If you supply a value here, then it must be a directory object that 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, or by the default directory object.

You can supply multiple *`file_name`* specifications as a comma-delimited list, or in separate `DUMPFILE` parameter specifications. If no extension is given for the file name, then Export uses the default file extension of `.dmp`. The file names can contain a substitution variable. The following table lists the available substitution variables.

Substitution Variable	Meaning
`%U`	The substitution variable is expanded in the resulting file names into a 2-digit, fixed-width, incrementing integer that starts at 01 and ends at 99. If a file specification contains two substitution variables, then both are incremented at the same time. For example, `exp%Uaa%U.dmp` resolves to `exp01aa01.dmp`, `exp02aa02.dmp`, and so forth.
`%d`, `%D`	Specifies the current day of the month from the Gregorian calendar in format `DD`. Note: This substitution variable cannot be used in an import file name.
`%m`, `%M`	Specifies the month in the Gregorian calendar in format `MM`. Note: This substitution variable cannot be used in an import file name.
`%t`, `%T`	Specifies the year, month, and day in the Gregorian calendar in this format: `YYYYMMDD`. Note: This substitution variable cannot be used in an import file name.

Substitution Variable	Meaning

`%l`, `%L`	Specifies a system-generated unique file name.
	The file names can contain a substitution variable (`%L`), which implies that multiple files can 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:
	`exp01aa01.dmp`
	`exp02aa02.dmp`
	and so forth, up until 99. Then, the next file name has 3 digits substituted:
	`exp100aa100.dmp`
	`exp101aa101.dmp`
	and so forth, up until 999, where the next file has 4 digits substituted. The substitutions continue up to the largest number substitution allowed, which is 2147483646.

`%y`, `%Y`	Specifies the year in this format: YYYY. Note: This substitution variable cannot be used in an import file name.

If the `FILESIZE` parameter is specified, then each dump file has a maximum of that size and be nonextensible. If more space is required for the dump file set, and a template with a substitution variable was supplied, then a new dump file is automatically created of the size specified by the `FILESIZE` parameter, if there is room on the device.

As each file specification or file template containing a substitution variable is defined, it is instantiated into one fully qualified file name, and Export attempts to create the file. The file specifications are processed in the order in which they are specified. If the job needs extra files because the maximum file size is reached, or to keep parallel workers active, then more files are created if file templates with substitution variables were specified.

Although it is possible to specify multiple files using the `DUMPFILE` parameter, the export job can only require a subset of those files to hold the exported data. The dump file set displayed at the end of the export job shows exactly which files were used. It is this list of files that is required to perform an import operation using this dump file set. Any files that were not used can be discarded.

When you specify the `DUMPFILE` parameter, it is possible to introduce conflicting file names, regardless of whether substitution variables are used. The following are some examples of `expdp` commands that would produce file name conflicts. For all these examples, an `ORA-27308 created file already exists` error is returned:

```
expdp system/manager directory=dpump_dir schemas=hr
DUMPFILE=foo%U.dmp,foo%U.dmp
```

```
expdp system/manager directory=dpump_dir schemas=hr
DUMPFILE=foo%U.dmp,foo%L.dmp
```

```
expdp system/manager directory=dpump_dir schemas=hr
DUMPFILE=foo%U.dmp,foo%D.dmp
```

```
expdp system/manager directory =dpump_dir schemas=hr
DUMPFILE=foo%tK_%t_%u_%y_P,foo%TK_%T_%U_%Y_P
```

Restrictions

- Any resulting dump file names that match preexisting dump file names generate an error, and the preexisting dump files are not overwritten. You can override this behavior by specifying the Export parameter `REUSE_DUMPFILES=YES`.
- 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.

Note:

The Data Pump Export `DUMPFILE` parameter gives you the option to specify an optional directory object using `directory-object-name:filename`. However, if `CREDENTIAL` is specified, then this overrides the `DUMPFILE` parameter specification.

Example

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

```
> expdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 DUMPFILE=dpump_dir2:exp1.dmp,
 exp2%U.dmp PARALLEL=3
```

The dump file, `exp1.dmp`, is written to the path associated with the directory object `dpump_dir2`, because `dpump_dir2` was specified as part of the dump file name, and therefore overrides the directory object specified with the `DIRECTORY` parameter. Because all three parallel processes are given work to perform during this job, dump files named `exp201.dmp` and `exp202.dmp` is created, based on the specified substitution variable `exp2%U.dmp`. Because no directory is specified for them, they are written to the path associated with the directory object, `dpump_dir1`, that was specified with the `DIRECTORY` parameter.

Related Topics

- Using Substitution Variables with Oracle Data Pump Exports

2.4.15 ENABLE_SECURE_ROLES

The Oracle Data Pump Export 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 exports, 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

```
expdp hr SCHEMAS=hr DIRECTORY=dpump_dir1 DUMPFILE=dpump_dir2:exp1.dmp,
 exp2%U.dmp ENABLE_SECURE_ROLES=YES
```

2.4.16 ENCRYPTION

The Oracle Data Pump Export command-line utility `ENCRYPTION` parameter specifies whether to encrypt data before writing it to the dump file set.

Default

The default value depends upon the combination of encryption-related parameters that are used. To enable encryption, either the `ENCRYPTION` or `ENCRYPTION_PASSWORD` parameter, or both, must be specified.

If only the `ENCRYPTION_PASSWORD` parameter is specified, then the `ENCRYPTION` parameter defaults to `ALL`.

If only the `ENCRYPTION` parameter is specified and the Oracle encryption wallet is open, then the default mode is `TRANSPARENT`. If only the `ENCRYPTION` parameter is specified and the wallet is closed, then an error is returned.

If neither `ENCRYPTION` nor `ENCRYPTION_PASSWORD` is specified, then `ENCRYPTION` defaults to `NONE`.

Purpose

Specifies whether to encrypt data before writing it to the dump file set.

Syntax and Description

```
ENCRYPTION = [ALL | DATA_ONLY | ENCRYPTED_COLUMNS_ONLY | METADATA_ONLY | NONE]
```

- `ALL` enables encryption for all data and metadata in the export operation.

- `DATA_ONLY` specifies that only data is written to the dump file set in encrypted format.
- `ENCRYPTED_COLUMNS_ONLY` specifies that only encrypted columns are written to the dump file set in encrypted format. This option cannot be used with the `ENCRYPTION_ALGORITHM` parameter because the columns already have an assigned encryption format and by definition, a column can have only one form of encryption.

  To use the `ENCRYPTED_COLUMNS_ONLY` option, you must also use the `ENCRYPTION_PASSWORD` parameter.

  To use the `ENCRYPTED_COLUMNS_ONLY` option, you must have Oracle Advanced Security Transparent Data Encryption (TDE) enabled. See *Oracle Database Advanced Security Guide* for more information about TDE.
- `METADATA_ONLY` specifies that only metadata is written to the dump file set in encrypted format.
- `NONE` specifies that no data is written to the dump file set in encrypted format.

SecureFiles Considerations for Encryption

If the data being exported includes SecureFiles that you want to be encrypted, then you must specify `ENCRYPTION=ALL` to encrypt the entire dump file set. Encryption of the entire dump file set is the only way to achieve encryption security for SecureFiles during a Data Pump export operation. For more information about SecureFiles, see *Oracle Database SecureFiles and Large Objects Developer's Guide*.

Oracle Database Vault Considerations for Encryption

When an export operation is started, Data Pump determines whether Oracle Database Vault is enabled. If it is, and dump file encryption has not been specified for the job, a warning message is returned to alert you that secure data is being written in an insecure manner (clear text) to the dump file set:

```
ORA-39327: Oracle Database Vault data is being stored unencrypted in dump
file set
```

You can stop the current export operation and start a new one, specifying that you want the output dump file set to be encrypted.

Restrictions

- To specify the `ALL`, `DATA_ONLY`, or `METADATA_ONLY` options, the `COMPATIBLE` initialization parameter must be set to at least 11.0.0.
- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later.
- To use the `ALL, DATA_ONLY` or `METADATA_ONLY` options without also using an encryption password, you must have the Oracle Advanced Security option enabled. See *Oracle Database Licensing Information* for information about licensing requirements for the Oracle Advanced Security option.

Example

The following example performs an export operation in which only data is encrypted in the dump file:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc.dmp JOB_NAME=enc1
ENCRYPTION=data_only ENCRYPTION_PASSWORD=foobar
```

Related Topics

- *Oracle Database Security Guide*
- SecureFiles LOB Storage
- Oracle Database Options and Their Permitted Features

2.4.17 ENCRYPTION_ALGORITHM

The Oracle Data Pump Export command-line utility `ENCRYPTION_ALGORITHM` parameter specifies which cryptographic algorithm should be used to perform the encryption.

Default

`AES256`

Purpose

Specifies which cryptographic algorithm should be used to perform the encryption.

Syntax and Description

```
ENCRYPTION_ALGORITHM = 256
```

Restrictions

- To use this encryption feature, the `COMPATIBLE` initialization parameter must be set to at least 11.0.0.
- The `ENCRYPTION_ALGORITHM` parameter requires that you also specify either the `ENCRYPTION` or `ENCRYPTION_PASSWORD` parameter; otherwise an error is returned.
- The `ENCRYPTION_ALGORITHM` parameter cannot be used in conjunction with `ENCRYPTION=ENCRYPTED_COLUMNS_ONLY` because columns that are already encrypted cannot have an additional encryption format assigned to them.
- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later.
- The `ENCRYPTION _ALGORITHM` parameter does not require that you have the Oracle Advanced Security enabled, but it can be used in conjunction with other encryption-related parameters that do require that option. See *Oracle Database Licensing Information* for information about licensing requirements for the Oracle Advanced Security option.

Example

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc3.dmp
ENCRYPTION_PASSWORD=foobar ENCRYPTION_ALGORITHM=AES256
```

Related Topics

- About Oracle Database Native Network Encryption and Data Integrity
- Oracle Database Options and Their Permitted Features

2.4.18 ENCRYPTION_MODE

The Oracle Data Pump Export command-line utility `ENCRYPTION_MODE` parameter specifies the type of security to use when encryption and decryption are performed.

Default

The default mode depends on which other encryption-related parameters are used. If only the `ENCRYPTION` parameter is specified and the Oracle encryption wallet is open, then the default mode is `TRANSPARENT`. If only the `ENCRYPTION` parameter is specified and the wallet is closed, then an error is returned.

If the `ENCRYPTION_PASSWORD` parameter is specified and the wallet is open, then the default is `DUAL`. If the `ENCRYPTION_PASSWORD` parameter is specified and the wallet is closed, then the default is `PASSWORD`.

Purpose

Specifies the type of security to use when encryption and decryption are performed.

Syntax and Description

```
ENCRYPTION_MODE = [DUAL | PASSWORD | TRANSPARENT]
```

`DUAL` mode creates a dump file set that can later be imported either transparently or by specifying a password that was used when the dual-mode encrypted dump file set was created. When you later import the dump file set created in `DUAL` mode, you can use either the wallet or the password that was specified with the `ENCRYPTION_PASSWORD` parameter. `DUAL` mode is best suited for cases in which the dump file set will be imported on-site using the wallet, but which may also need to be imported offsite where the wallet is not available.

`PASSWORD` mode requires that you provide a password when creating encrypted dump file sets. You will need to provide the same password when you import the dump file set. `PASSWORD` mode requires that you also specify the `ENCRYPTION_PASSWORD` parameter. The `PASSWORD` mode is best suited for cases in which the dump file set will be imported into a different or remote database, but which must remain secure in transit.

`TRANSPARENT` mode enables you to create an encrypted dump file set without any intervention from a database administrator (DBA), provided the required wallet is available. Therefore, the `ENCRYPTION_PASSWORD` parameter is not required. The parameter will, in fact, cause an error if it is used in `TRANSPARENT` mode. This encryption mode is best suited for cases in which the dump file set is imported into the same database from which it was exported.

Restrictions

- To use `DUAL` or `TRANSPARENT` mode, the `COMPATIBLE` initialization parameter must be set to at least 11.0.0.
- When you use the `ENCRYPTION_MODE` parameter, you must also use either the `ENCRYPTION` or `ENCRYPTION_PASSWORD` parameter. Otherwise, an error is returned.
- When you use the `ENCRYPTION=ENCRYPTED_COLUMNS_ONLY`, you cannot use the `ENCRYPTION_MODE` parameter. Otherwise, an error is returned.
- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later.

- The use of `DUAL` or `TRANSPARENT` mode requires that the Oracle Advanced Security option is enabled. See *Oracle Database Licensing Information* for information about licensing requirements for the Oracle Advanced Security option.

Example

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_enc4.dmp
ENCRYPTION=all ENCRYPTION_PASSWORD=secretwords
ENCRYPTION_ALGORITHM=AES256 ENCRYPTION_MODE=DUAL
```

Related Topics

- Oracle Database Options and Their Permitted Features

2.4.19 ENCRYPTION_PASSWORD

The Oracle Data Pump Export command-line utility `ENCRYPTION_PASSWORD` parameter prevents unauthorized access to an encrypted dump file set.

Default

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

Purpose

Specifies a password for encrypting encrypted column data, metadata, or table data in the export dump file. Using this parameter prevents unauthorized access to an encrypted dump file set.

Note:

Oracle Data Pump encryption functionality changed as of Oracle Database 11g release 1 (11.1). Before release 11.1, the `ENCRYPTION_PASSWORD` parameter applied only to encrypted columns. However, as of release 11.1, the new `ENCRYPTION` parameter provides options for encrypting other types of data. As a result of this change, if you now specify `ENCRYPTION_PASSWORD` without also specifying `ENCRYPTION` and a specific option, then all data written to the dump file is encrypted (equivalent to specifying `ENCRYPTION=ALL`). To re-encrypt only encrypted columns, you must now specify `ENCRYPTION=ENCRYPTED_COLUMNS_ONLY` in addition to `ENCRYPTION_PASSWORD`.

Syntax and Description

```
ENCRYPTION_PASSWORD = password
```

The *`password`* value that is supplied specifies a key for re-encrypting encrypted table columns, metadata, or table data so that they are not written as clear text in the dump file set. If the export operation involves encrypted table columns, but an encryption password is not supplied, then the encrypted columns are written to the dump file set as clear text and a warning is issued.

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.

The maximum length allowed for an encryption password is usually 128 bytes. However, the limit is 30 bytes if `ENCRYPTION=ENCRYPTED_COLUMNS_ONLY` and either the `VERSION` parameter or database compatibility is set to less than 12.2.

For export operations, this parameter is required if the `ENCRYPTION_MODE` parameter is set to either `PASSWORD` or `DUAL`.

Note:

There is no connection or dependency between the key specified with the Oracle Data Pump `ENCRYPTION_PASSWORD` parameter and the key specified with the `ENCRYPT` keyword when the table with encrypted columns was initially created. For example, suppose that a table is created as follows, with an encrypted column whose key is `xyz`:

```
CREATE TABLE emp (col1 VARCHAR2(256) ENCRYPT IDENTIFIED BY "xyz");
```

When you export the `emp` table, you can supply any arbitrary value for `ENCRYPTION_PASSWORD`. It does not have to be `xyz`.

Restrictions

- This parameter is valid only in Oracle Database Enterprise Edition 11g or later.
- The `ENCRYPTION_PASSWORD` parameter is required for the transport of encrypted tablespaces and tablespaces containing tables with encrypted columns in a full transportable export.
- If `ENCRYPTION_PASSWORD` is specified but `ENCRYPTION_MODE` is not specified, then it is not necessary to have Oracle Advanced Security Transparent Data Encryption enabled, because `ENCRYPTION_MODE` defaults to `PASSWORD`.
- If the requested encryption mode is `TRANSPARENT`, then the `ENCRYPTION_PASSWORD` parameter is not valid.
- If `ENCRYPTION_MODE` is set to `DUAL`, then to use the `ENCRYPTION_PASSWORD` parameter, you must have Oracle Advanced Security Transparent Data Encryption (TDE) enabled. See *Oracle Database Advanced Security Guide* for more information about TDE.
- For network exports, the `ENCRYPTION_PASSWORD` parameter in conjunction with `ENCRYPTION=ENCRYPTED_COLUMNS_ONLY` is not supported with user-defined external tables that have encrypted columns. The table is skipped, and an error message is displayed, but the job continues.

Example

In the following example, an encryption password, `123456`, is assigned to the dump file, `dpcd2be1.dmp`.

```
> expdp hr TABLES=employee_s_encrypt DIRECTORY=dpump_dir1
DUMPFILE=dpcd2be1.dmp ENCRYPTION=ENCRYPTED_COLUMNS_ONLY
ENCRYPTION_PASSWORD=123456
```

Encrypted columns in the `employee_s_encrypt` table are not written as clear text in the `dpcd2be1.dmp` dump file. Afterward, if you want to import the `dpcd2be1.dmp` file created by this example, then you must supply the same encryption password.

Related Topics

- *Oracle Database Licensing Information User Manual*
- *Oracle Database Advanced Security Guide*

2.4.20 ENCRYPTION_PWD_PROMPT

The Oracle Data Pump Export command-line utility `ENCRYPTION_PWD_PROMPT` specifies whether Oracle Data Pump prompts you for the encryption password.

Default

`NO`

Purpose

Specifies whether 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 Data Pump to prompt you for the encryption password, rather than you entering it on the command line with the `ENCRYPTION_PASSWORD` parameter. The advantage to doing this is that the encryption password is not echoed to the screen when it is entered at the prompt. Whereas, when it is entered on the command line using the `ENCRYPTION_PASSWORD` parameter, it 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, 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 syntax example shows Data Pump first prompting for the user password and then for the encryption password.

```
> expdp 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 -
```

```
Production
Version 18.1.0.0.0

Encryption Password:

Starting "HR"."SYS_EXPORT_SCHEMA_01":  hr/******** directory=dpump_dir1
dumpfile=hr.dmp encryption_pwd_prompt=Y
.
.
.
```

2.4.21 ESTIMATE

The Oracle Data Pump Export command-line utility `ESTIMATE` parameter specifies the method that Export uses to estimate how much disk space each table in the export job will consume (in bytes).

Default

`STATISTICS`

Purpose

Specifies the method that Export will use to estimate how much disk space each table in the export job will consume (in bytes). The estimate is printed in the log file and displayed on the client's standard output device. The estimate is for table row data only; it does not include metadata.

Syntax and Description

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

- `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.)

Restrictions

- If the Data Pump export job involves compressed tables, then when you use `ESTIMATE=BLOCKS`, the default size estimation given for the compressed table is inaccurate. This inaccuracy results because the size estimate does not reflect that the data was stored in a compressed form. To obtain a more accurate size estimate for compressed tables, use `ESTIMATE=STATISTICS`.
- If either the `QUERY` or `REMAP_DATA` parameter is used, then the estimate can also be inaccurate.

Example

The following example shows a use of the `ESTIMATE` parameter in which the estimate is calculated using statistics for the `employees` table:

```
> expdp hr TABLES=employees ESTIMATE=STATISTICS DIRECTORY=dpump_dir1
 DUMPFILE=estimate_stat.dmp
```

2.4.22 ESTIMATE_ONLY

The Oracle Data Pump Export command-line utility `ESTIMATE_ONLY` parameter instructs Export to estimate the space that a job consumes, without actually performing the export operation.

Default

`NO`

Purpose

Instructs Export to estimate the space that a job consumes, without actually performing the export operation.

Syntax and Description

```
ESTIMATE_ONLY=[YES | NO]
```

If `ESTIMATE_ONLY=YES`, then Export estimates the space that would be consumed, but quits without actually performing the export operation.

Restrictions

- The `ESTIMATE_ONLY` parameter cannot be used in conjunction with the `QUERY` parameter.

Example

The following shows an example of using the `ESTIMATE_ONLY` parameter to determine how much space an export of the `HR` schema requires.

```
> expdp hr ESTIMATE_ONLY=YES NOLOGFILE=YES SCHEMAS=HR
```

2.4.23 EXCLUDE

The Oracle Data Pump Export command-line utility `EXCLUDE` parameter enables you to filter the metadata that is exported by specifying objects and object types that you want to exclude from the export operation.

Default

There is no default

Purpose

Enables you to filter the metadata that is exported by specifying objects and object types that you want to exclude from the export operation.

Syntax and Description

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

The *`object_type`* specifies the type of object that you want to exclude. 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.

All object types for the given mode of export are included in the export, except object types specified in an `EXCLUDE` statement. If an object is excluded, then all dependent objects are also excluded. For example, excluding a table also excludes all indexes and triggers on the table.

The *`name_clause`* is optional. Using this parameter enables selection of specific objects within an object type. It is a SQL expression used as a filter on the object names of that type. It consists of a SQL operator, and the values against which you want to compare the object names of the specified type. 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, because single quotation marks are required to delimit the name strings. For example, you can set `EXCLUDE=INDEX:"LIKE 'EMP%'"` to exclude all indexes whose names start with `EMP`.

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 you supplied the *`name_clause`* as `Employees` or `employees` or any other variation that does not match the existing table, then the table is not found.

If no *`name_clause`* is provided, then all objects of the specified type are excluded.

You can specify more than one `EXCLUDE` statement.

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 otherwise can be needed on the command line.

If the *`object_type`* you specify is `CONSTRAINT`, `GRANT`, or `USER`, then be aware of the effects, as described in the following paragraphs.

Excluding Constraints

The following constraints cannot be explicitly 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

For example, the following `EXCLUDE` statements are 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 user 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.

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

In this example, the export mode `FULL` is specified. If no mode is specified, then the default mode is used. The default mode is `SCHEMAS`. But if the default mode is used, then in this example, the default causes an error, because if `SCHEMAS` is used, then the command indicates that you want the schema both exported and excluded at the same time.

If you try to exclude a user by using a statement such as `EXCLUDE=USER:"='HR'"`, then only the information used in `CREATE USER hr` DDL statements is excluded, and you can obtain 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 includes all objects identified by the parameter. Then it processes the exclude parameters, eliminating the excluded objects from the included set.

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

The following is an example of using the `EXCLUDE` statement.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_exclude.dmp EXCLUDE=VIEW,
PACKAGE, FUNCTION
```

This example results in a schema-mode export (the default export mode) in which all the `hr` schema is exported except its views, packages, and functions.

Related Topics

- Oracle Data Pump Export Metadata Filters
- Filtering During Export Operations
- INCLUDE
  The Oracle Data Pump Export command-line utility `INCLUDE` parameter enables you to filter the metadata that is exported by specifying objects and object types for the current export mode.

2.4.24 FILESIZE

The Oracle Data Pump Export command-line utility `FILESIZE` parameter specifies the maximum size of each dump file.

Default

`0` (equivalent to the maximum size of 16 terabytes)

Purpose

Specifies the maximum size of each dump file. 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 more 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 can be rounded down slightly to match the size of the internal blocks used in dump files.

Restrictions

- The minimum size for a file is 10 times the default Data Pump block size, which is 4 kilobytes.
- The maximum size for a file is 16 terabytes.

Example

The following example shows setting the size of the dump file to 3 megabytes:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_3m.dmp FILESIZE=3MB
```

In this scenario, if the 3 megabytes allocated was not sufficient to hold all the exported data, then the following error results, and displayed and the job stops:

```
ORA-39095: Dump file space has been exhausted: Unable to allocate 217088 bytes
```

The actual number of bytes that cannot be allocated can vary. Also, this number does not represent the amount of space required complete the entire export operation. It indicates only the size of the current object that was being exported when the job ran out of dump file space. You can correct this problem by first attaching to the stopped job, adding one or more files using the `ADD_FILE` command, and then restarting the operation.

2.4.25 FLASHBACK_SCN

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

Default

Default: There is no default

Purpose

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

Syntax and Description

```
FLASHBACK_SCN=scn_value
```

The export operation is performed with data that is consistent up to the specified SCN. If the `NETWORK_LINK` parameter is specified, then the SCN refers to the SCN of the source database.

As of Oracle Database 12c release 2 (12.2) and later releases, the SCN value can be a big SCN (8 bytes). You can also specify a big SCN when you create a dump file for an earlier version that does not support big SCNs, because actual SCN values are not moved.

Restrictions

- `FLASHBACK_SCN` and `FLASHBACK_TIME` are mutually exclusive.
- 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.
- 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 example assumes that an existing SCN value of `384632` exists. It exports the `hr` schema up to SCN 384632.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_scn.dmp FLASHBACK_SCN=384632
```

Note:

If you are on a logical standby system and using a network link to access the logical standby primary, 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*

2.4.26 FLASHBACK_TIME

The Oracle Data Pump Export command-line utility `FLASHBACK_TIME` parameter finds the SCN that most closely matches the specified time.

Default

There is no default.

Purpose

Finds the system change number (SCN) that most closely matches the specified time. This SCN is used to enable the Flashback utility. The export operation is performed with data that is consistent up to this SCN.

Syntax and Description

```
FLASHBACK_TIME="TO_TIMESTAMP(time-value)"
```

Because the `TO_TIMESTAMP` value is enclosed in quotation marks, it is best to put this parameter in a parameter file.

Alternatively, you can enter the following parameter setting. This setting initiate a consistent export that is based on current system time:

```
FLASHBACK_TIME=systimestamp
```

Restrictions

- `FLASHBACK_TIME` and `FLASHBACK_SCN` are mutually exclusive.
- 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.

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.par`, with the following contents:

```
DIRECTORY=dpump_dir1
DUMPFILE=hr_time.dmp
FLASHBACK_TIME="TO_TIMESTAMP('27-10-2012 13:16:00', 'DD-MM-YYYY HH24:MI:SS')"
```

You can then issue the following command:

```
> expdp hr PARFILE=flashback.par
```

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

Note:

If you are on a logical standby system and using a network link to access the logical standby primary, then the `FLASHBACK_SCN` parameter is ignored, because the logical standby selects the SCNs. See *Oracle Data Guard Concepts and Administration* for information about logical standby databases.

See *Oracle Database Development Guide* for information about using Flashback Query.

Related Topics

- Logical Standby Databases in *Oracle Data Guard Concepts and Administration*
- Using Oracle Flashback Query (SELECT AS OF) in *Oracle Database Development Guide*

2.4.27 FULL

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

Default

`NO`

Purpose

Specifies that you want to perform a full database mode export.

Syntax and Description

```
FULL=[YES | NO]
```

`FULL=YES` indicates that all data and metadata are to be exported. To perform a full export, you must have the `DATAPUMP_EXP_FULL_DATABASE` role.

Filtering can restrict what is exported using this export mode.

You can perform a full mode export using the transportable option (`TRANSPORTABLE=ALWAYS`). This is referred to as a full transportable export, which exports all objects and data necessary to create a complete copy of the database. See

Note:

Be aware that when you later 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 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.

Restrictions

- To use the `FULL` parameter in conjunction with `TRANSPORTABLE` (a full transportable export), either the Data Pump `VERSION` parameter must be set to at least 12.0. or the `COMPATIBLE` database initialization parameter must be set to at least 12.0 or later.
- A full export does not, by default, export system schemas that contain Oracle-managed data and metadata. Examples of system schemas that are not exported by default include `SYS`, `ORDSYS`, and `MDSYS`.
- Grants on objects owned by the `SYS` schema are never exported.
- A full export operation exports objects from only one database edition; by default it exports the current edition but you can use the Export `SOURCE_EDITION` parameter to specify a different edition.
- If you are exporting data that is protected by a realm, then you must have authorization for that realm.
- 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 Oracle 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.

Example

The following is an example of using the `FULL` parameter. The dump file, `expfull.dmp` is written to the `dpump_dir2` directory.

```
> expdp hr DIRECTORY=dpump_dir2 DUMPFILE=expfull.dmp FULL=YES NOLOGFILE=YES
```

To see a detailed example of how to perform a full transportable export, see *Oracle Database Administrator's Guide*. For information about configuring realms, see *Oracle Database Vault Administrator's Guide*.

Related Topics

- Full Export Mode
- Gathering Database Statistics
- Transporting Databases
- Configuring Realms

2.4.28 HELP

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

Default

`NO`

Purpose

Displays online help for the Export utility.

Syntax and Description

```
HELP = [YES | NO]
```

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

Example

```
> expdp HELP = YES
```

This example display a brief description of all Export parameters and commands.

2.4.29 INCLUDE

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

Default

There is no default

Purpose

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

Syntax and Description

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

The `object_type` specifies the type of object to be included. 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.

Only object types explicitly specified in `INCLUDE` statements, and their dependent objects, are exported. No other object types, including the schema definition information that is normally part of a schema-mode export when you have the `DATAPUMP_EXP_FULL_DATABASE` role, are exported.

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`, 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.

The name that you supply for the `name_clause` must exactly match an existing object in the database, including upper- and lower- case letters. 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 letters. If the `name_clause` is provided as `Employees` or `employees` or any other variation, then the table is not found.

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 otherwise need to enter on the command line.

For example, suppose you have a parameter file named `hr.par` with the following content:

```
SCHEMAS=HR
DUMPFILE=expinclude.dmp
DIRECTORY=dpump_dir1
LOGFILE=expinclude.log
INCLUDE=TABLE:"IN ('EMPLOYEES', 'DEPARTMENTS')"
INCLUDE=PROCEDURE
INCLUDE=INDEX:"LIKE 'EMP%'"
```

You can then use the `hr.par` file to start an export operation, without having to enter any other parameters on the command line. The `EMPLOYEES` and `DEPARTMENTS` tables, all procedures, and all index names with an EMP prefix, are included in the export.

```
> expdp hr PARFILE=hr.par
```

Including Constraints

If the *`object_type`* that you specify is a `CONSTRAINT`, then be aware of the effects of using a constraint..

You cannot include explicitly the following constraints:

- `NOT NULL` constraints
- Constraints that are required for the table to be created and loaded successfully. For example: you cannot include primary key constraints for index-organized tables, or `REF SCOPE` and `WITH ROWID` constraints for tables with `REF` columns.

For example, the following `INCLUDE` statements are interpreted as follows:

- `INCLUDE=CONSTRAINT` includes all (nonreferential) constraints, except for `NOT NULL` constraints, and any constraints needed for successful table creation and loading.
- `INCLUDE=REF_CONSTRAINT` includes referential integrity (foreign key) constraints.

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 exported.
- 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

The following example performs an export of all tables (and their dependent objects) in the `hr` schema:

```
> expdp hr INCLUDE=TABLE DUMPFILE=dpump_dir1:exp_inc.dmp NOLOGFILE=YES
```

Related Topics

- Oracle Data Pump Metadata Filters
- Parameters Available in Data Pump Export Command-Line Mode

2.4.30 JOB_NAME

The Oracle Data Pump Export command-line utility `JOB_NAME` parameter identifies the export job in subsequent actions.

Default

A system-generated name of the form `SYS_EXPORT_`*`EXPORT or SQLFILE_mode_NN`*

Purpose

Use the `JOB_NAME` parameter when you want to identify the export 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 name of 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 *`jobname_string`* specifies a name of up to 128 bytes for the export job. The bytes must represent printable characters and spaces. If the name includes spaces or other non-alphanumeric characters (for example, hyphens), then the name must be enclosed in single quotation marks. Examples: 'Thursday Export', 'Thursday-Export'. 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 export 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_EXPORT_`*`mode_NN`*, where *`NN`* expands to a 2-digit incrementing integer starting at 01. An example of a default name is `'SYS_EXPORT_TABLESPACE_02'`.

Example

The following example shows an export operation that is assigned a job name of `exp_job`:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=exp_job.dmp JOB_NAME=exp_job
NOLOGFILE=YES
```

Related Topics

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

2.4.31 KEEP_MASTER

The Oracle Data Pump Export command-line utility `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

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr KEEP_MASTER=YES
```

2.4.32 LOGFILE

The Oracle Data Pump Export command-line utility `LOGFILE` parameter specifies the name, and optionally, a directory, for the log file of the export job.

Default

`export log`.

Purpose

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

Syntax and Description

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

You can specify a database *`directory_object`* previously established by the DBA, assuming that you have access to it. This setting overrides the directory object specified with the `DIRECTORY` parameter.

The *`file_name`* specifies a name for the log file. The default behavior is to create a file named `export.log` in the directory referenced by the directory object specified in the `DIRECTORY` parameter.

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 for an export job unless the `NOLOGFILE` parameter is specified. As with the dump file set, the log file is relative to the server and not the client.

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: `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.

Restrictions

- To perform an Oracle Data Pump Export 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, if you specify `NOLOGFILE=YES`, then that setting prevents the writing of the log file.

Example

The following example shows how to specify a log file name when you do not want to use the default:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp LOGFILE=hr_export.log
```

Note:

Oracle Data Pump Export writes the log file using the database character set. If your client `NLS_LANG` environment setting 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.

Related Topics

- STATUS
  The Oracle Data Pump Export command-line utility `STATUS` parameter specifies the frequency at which the job status display is updated.
- Using Directory Objects When Oracle Automatic Storage Management Is Enabled

2.4.33 LOGTIME

The Oracle Data Pump Export command-line utility `LOGTIME` parameter specifies that messages displayed during export operations are timestamped.

Default

No timestamps are recorded

Purpose

Specifies that messages displayed during export operations are timestamped. 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 export operation:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr LOGTIME=ALL
```

The output looks similar to the following:

```
10-JUL-12 10:12:22.300: Starting "HR"."SYS_EXPORT_SCHEMA_01":  hr/********
directory=dpump_dir1 dumpfile=expdat.dmp schemas=hr logtime=all
10-JUL-12 10:12:22.915: Estimate in progress using BLOCKS method...
10-JUL-12 10:12:24.422: Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
10-JUL-12 10:12:24.498: Total estimation using BLOCKS method: 128 KB
10-JUL-12 10:12:24.822: Processing object type SCHEMA_EXPORT/USER
10-JUL-12 10:12:24.902: Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
10-JUL-12 10:12:24.926: Processing object type SCHEMA_EXPORT/ROLE_GRANT
10-JUL-12 10:12:24.948: Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
10-JUL-12 10:12:24.967: Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
10-JUL-12 10:12:25.747: Processing object type SCHEMA_EXPORT/PRE_SCHEMA/
PROCACT_SCHEMA
10-JUL-12 10:12:32.762: Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
10-JUL-12 10:12:46.631: Processing object type SCHEMA_EXPORT/TABLE/TABLE
10-JUL-12 10:12:58.007: Processing object type SCHEMA_EXPORT/TABLE/GRANT/
OWNER_GRANT/OBJECT_GRANT
10-JUL-12 10:12:58.106: Processing object type SCHEMA_EXPORT/TABLE/COMMENT
10-JUL-12 10:12:58.516: Processing object type SCHEMA_EXPORT/PROCEDURE/
PROCEDURE
10-JUL-12 10:12:58.630: Processing object type SCHEMA_EXPORT/PROCEDURE/
ALTER_PROCEDURE
10-JUL-12 10:12:59.365: Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
10-JUL-12 10:13:01.066: Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/
CONSTRAINT
10-JUL-12 10:13:01.143: Processing object type SCHEMA_EXPORT/TABLE/INDEX/
```

```
STATISTICS/INDEX_STATISTICS
10-JUL-12 10:13:02.503: Processing object type SCHEMA_EXPORT/VIEW/VIEW
10-JUL-12 10:13:03.288: Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/
REF_CONSTRAINT
10-JUL-12 10:13:04.067: Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
10-JUL-12 10:13:05.251: Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/
TABLE_STATISTICS
10-JUL-12 10:13:06.172: . . exported
"HR"."EMPLOYEES"                          17.05 KB     107 rows
10-JUL-12 10:13:06.658: . . exported
"HR"."COUNTRIES"                          6.429 KB      25 rows
10-JUL-12 10:13:06.691: . . exported
"HR"."DEPARTMENTS"                        7.093 KB      27 rows
10-JUL-12 10:13:06.723: . . exported
"HR"."JOBS"                               7.078 KB      19 rows
10-JUL-12 10:13:06.758: . . exported
"HR"."JOB_HISTORY"                        7.164 KB      10 rows
10-JUL-12 10:13:06.794: . . exported
"HR"."LOCATIONS"                          8.398 KB      23 rows
10-JUL-12 10:13:06.824: . . exported
"HR"."REGIONS"                            5.515 KB       4 rows
10-JUL-12 10:13:07.500: Master table "HR"."SYS_EXPORT_SCHEMA_01" successfully
loaded/unloaded
10-JUL-12 10:13:07.503:
******************************************************************************
```

2.4.34 METRICS

The Oracle Data Pump Export command-line utility `METRICS` parameter indicates whether you want additional information about the job reported to the Data Pump log file.

Default

`NO`

Purpose

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

Restrictions

None

Example

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr METRICS=YES
```

2.4.35 NETWORK_LINK

The Oracle Data Pump Export command-line utility `NETWORK_LINK` parameter enables an export from a (source) database identified by a valid database link.

Default

There is no default

Purpose

Enables an export from a (source) database identified by a valid database link. The data from the source database instance is written to a dump file set on the connected database instance.

Syntax and Description

```
NETWORK_LINK=source_database_link
```

The `NETWORK_LINK` parameter initiates an export using a database link. This export setting means that the system to which the `expdp` client is connected contacts the source database referenced by the *`source_database_link`*, retrieves data from it, and writes the data to a dump file set back on the connected system.

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.

If the source database is read-only, then the user on the source database must have a locally managed temporary tablespace assigned as the default temporary tablespace. Otherwise, the job will fail. The database where you are running the Oracle Data Pump job has its own time stamp with time zone (`TSTZ`) version. This version may or may not be the same version as the source database. If the versions are different, then the export job will convert the source data to be compatible with the target database `TSTZ` version where the export job is running.

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

- 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 export operation is performed over an unencrypted network link, then all data is exported as clear text, even if it is encrypted in the database. See *Oracle Database Security Guide* on strong authentication for more information about network security.

Restrictions

- The following types of database links are not supported for use with Data Pump Export:
  - Private connected user

  – Current user

- 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 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).
- When transporting a database over the network using full transportable export, 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.
- Metadata cannot be exported or imported in parallel when the `NETWORK_LINK` parameter is also used

Example

The following is a syntax example of using the `NETWORK_LINK` parameter. Replace the variable *`source_database_link`* with the name of a valid database link that must already exist.

```
> expdp hr DIRECTORY=dpump_dir1 NETWORK_LINK=source_database_link
  DUMPFILE=network_export.dmp LOGFILE=network_export.log
```

Related Topics

- Introduction to Strong Authentication
- Database Links
- CREATE DATABASE LINK

2.4.36 NOLOGFILE

The Oracle Data Pump Export command-line utility `NOLOGFILE` parameter specifies whether to suppress creation of a log file.

Default

`NO`

Purpose

Specifies whether to suppress creation of a log file.

Syntax and Description

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

Specify `NOLOGFILE=YES` to suppress the default behavior of creating a log file. 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:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp NOLOGFILE=YES
```

This command results in a schema-mode export (the default), in which no log file is written.

2.4.37 PARALLEL

The Oracle Data Pump Export command-line utility `PARALLEL` parameter specifies the maximum number of processes of active execution operating on behalf of the export job.

Default

`1`

Purpose

Specifies the maximum number of processes of active execution operating on behalf of the export job. This execution set consists of a combination of worker processes and parallel input/output (I/O) server processes. The Data Pump control process and worker processes acting as query coordinators in parallel query operations do not count toward this total.

This parameter enables you to make trade-offs between resource consumption and elapsed time.

Syntax and Description

```
PARALLEL=integer
```

The value that you specify for *`integer`* should be less than, or equal to, the number of files in the dump file set (or you should specify either the `%U` or `%L` substitution variables in the dump file specifications). Because each active worker processor I/O server process writes exclusively to one file at a time, an insufficient number of files can have adverse effects. For example, some of the worker processes can be idle while waiting for files, thereby degrading the overall performance of the job. More importantly, if any member of a cooperating group of parallel I/O server processes cannot obtain a file for output, then the export operation is stopped with an `ORA-39095` error. Both situations can be corrected by attaching to the job using the Data Pump Export utility, adding more files using the `ADD_FILE` command while in interactive mode, and in the case of a stopped job, restarting the job.

To increase or decrease the value of `PARALLEL` during job execution, use interactive-command mode. Decreasing parallelism does not result in fewer worker processes associated with the job; it decreases the number of worker processes that are running at any given time. Also, any ongoing work must reach an orderly completion point before the decrease takes effect. Therefore, it can take a while to see any effect from decreasing the value. Idle worker processes are not deleted until the job exits.

If there is work that can be performed in parallel, then increasing the parallelism takes effect immediately .

Using PARALLEL During An Export In An Oracle RAC Environment

In an Oracle Real Application Clusters (Oracle RAC) environment, if an export 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 export 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 members.

Restrictions

- This parameter is valid only in the Enterprise Edition of Oracle Database 11g or later.
- To export a table or table partition in parallel (using parallel query, or PQ, worker processes), you must have the `DATAPUMP_EXP_FULL_DATABASE` role.
- Transportable tablespace metadata cannot be exported in parallel.
- Metadata cannot be exported in parallel when the `NETWORK_LINK` parameter is also used.
- The following objects cannot be exported in parallel:
  - `TRIGGER`
  - `VIEW`
  - `OBJECT_GRANT`
  - `SEQUENCE`
  - `CONSTRAINT`
  - `REF_CONSTRAINT`

Example

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

```
> expdp hr DIRECTORY=dpump_dir1 LOGFILE=parallel_export.log
JOB_NAME=par4_job DUMPFILE=par_exp%u.dmp PARALLEL=4
```

This results in a schema-mode export (the default) of the `hr` schema, in which up to four files can be created in the path pointed to by the directory object, `dpump_dir1`.

Related Topics

- DUMPFILE
  The Oracle Data Pump Export command-line utility `DUMPFILE` parameter specifies the names, and optionally, the directory objects of dump files for an export job.
- Performing a Parallel Full Database Export

2.4.38 PARALLEL_THRESHOLD

The Oracle Data Pump Export 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`. Note that 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.

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.

You can use this parameter to assist with particular data movement issues. For example:

- When you want to use Oracle Data Pump to load a large table from one database into a larger table in another database. One possible use case: Uploading weekly sales data from an OLTP database into a reporting or business analytics data warehouse database.
- When you want to export a single large table, but you have not gathered RDBMS stats recently. The default size is determined from the table's statistics. However, suppose that the statistics are old (or have never been run). In that case, the value used by Oracle Data Pump could underrepresent the table's actual size. To compensate for a case such as this, you can specify a smaller `parallel_threshold` value, so that the algorithm for the degree of parallelism (table size divided by threshold amount) can yield a more reasonable degree of parallelism value.

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 Export (`expdp`), and the job name is `parathresh_example`.

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

2.4.39 PARFILE

The Oracle Data Pump Export command-line utility `PARFILE` parameter specifies the name of an export parameter file.

Default

There is no default

Purpose

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

Syntax and Description

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

A parameter file enables you to specify Oracle Data Pump parameters within a file. You can then specify that file on the command line, instead of entering all of the individual commands. Using a parameter file can be useful if you use the same parameter combination many times. The use of parameter files is also highly recommended when you use parameters whose values require the use of quotation marks.

A directory object is not specified for the parameter file. You do not specify a directory object, because the parameter file is opened and read by the `expdp` client, unlike dump files, log files, and SQL files which are created and written by the server. 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, then 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 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.par`, is as follows:

```
SCHEMAS=HR
DUMPFILE=exp.dmp
DIRECTORY=dpump_dir1
LOGFILE=exp.log
```

You can then issue the following Export command to specify the parameter file:

```
> expdp hr PARFILE=hr.par
```

Related Topics

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

2.4.40 QUERY

The Oracle Data Pump Export command-line utility `QUERY` parameter enables you to specify a query clause that is used to filter the data that gets exported.

Default

There is no default.

Purpose

Enables you to specify a query clause that is used to filter the data that gets exported.

Syntax and Description

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

The *`query_clause`* is typically a SQL `WHERE` clause for fine-grained row selection, but could 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 export job. A table-specific query overrides a query applied to all tables.

When the query is to be applied to a specific table, a colon must separate the table name from the query clause. More than one table-specific query can be specified, 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, when you specify a value for this parameter that the uses quotation marks, it 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 might otherwise be needed on the command line. .

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

Restrictions

- The `QUERY` parameter cannot be used with the following parameters:

  - `CONTENT=METADATA_ONLY`
  - `ESTIMATE_ONLY`
  - `TRANSPORT_TABLESPACES`
- When the `QUERY` parameter is specified for a table, Data Pump uses external tables to unload the target table. External tables uses a SQL `CREATE TABLE AS SELECT` statement. The value of the `QUERY` parameter is the `WHERE` clause in the `SELECT` portion of the `CREATE TABLE` statement. If the `QUERY` parameter includes references to another table with columns whose names match the table being unloaded, and if those columns are used in the query, then you will need to use a table alias to distinguish between columns in the table being unloaded and columns in the `SELECT` statement with the same name. The table alias used by Data Pump for the table being unloaded is `KU$`.

  For example, suppose you want to export a subset of the `sh.sales` table based on the credit limit for a customer in the `sh.customers` table. In the following example, `KU$` is used to qualify the `cust_id` field in the `QUERY` parameter for unloading `sh.sales`. As a result, Data Pump exports 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)"'
  ```

  In the following query, `KU$` is not used for a table alias. The result is that all rows are unloaded:

  ```
  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, which includes quotation marks. This restriction means that the actual maximum length allowed is 3998 bytes.

Example

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

```
> expdp hr PARFILE=emp_query.par
```

The contents of the `emp_query.par` file are as follows:

```
QUERY=employees:"WHERE department_id > 10 AND salary > 10000"
NOLOGFILE=YES
DIRECTORY=dpump_dir1
DUMPFILE=exp1.dmp
```

This example unloads all tables in the `hr` schema, but only the rows that fit the query expression. In this case, all rows in all tables (except `employees`) in the `hr` schema are unloaded. For the `employees` table, only rows that meet the query criteria are unloaded.

Related Topics

- About Oracle Data Pump Export Parameters

2.4.41 REMAP_DATA

The Oracle Data Pump Export command-line utility `REMAP_DATA` parameter enables you to specify a remap function that takes as a source the original value of the designated column and returns a remapped value that replaces the original value in the dump file.

Default

There is no default

Purpose

The `REMAP_DATA` parameter enables you to specify a remap function that takes as a source the original value of the designated column, and returns a remapped value that will replace the original value in the dump file. A common use for this option is to mask data when moving from a production system to a test system. For example, a column of sensitive customer data, such as credit card numbers, could be replaced with numbers generated by a `REMAP_DATA` function. Replacing the sensitive data with numbers enables the data to retain its essential formatting and processing characteristics, without exposing private data to unauthorized personnel.

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 description of each syntax element, in the order in which they appear in the syntax, is as follows:

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

tablename: the table whose column you want to be remapped.

column_name: the column whose data you want to be remapped.

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

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

function: the name of the function within the PL/SQL that will be 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 `REMAP_DATA` specification.
- Remapping LOB column data of a remote table is not supported.

- Columns of the following types are not supported byREMAP_DATA: User Defined Types, attributes of User Defined Types, LONGs, REFs, VARRAYs, Nested Tables, BFILEs, and XMLtype.

Example

The following example assumes a package named remap has been created that contains functions named minus10 and plusx. These functions change the values for employee_id and first_name in the employees table.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=remap1.dmp TABLES=employees
REMAP_DATA=hr.employees.employee_id:hr.remap.minus10
REMAP_DATA=hr.employees.first_name:hr.remap.plusx
```

2.4.42 REUSE_DUMPFILES

The Oracle Data Pump Export command-line utility REUSE_DUMPFILES parameter specifies whether to overwrite a preexisting dump file.

Default

NO

Purpose

Specifies whether to overwrite a preexisting dump file.

Syntax and Description

```
REUSE_DUMPFILES=[YES | NO]
```

Normally, Data Pump Export will return an error if you specify a dump file name that already exists. The REUSE_DUMPFILES parameter allows you to override that behavior and reuse a dump file name. For example, if you performed an export and specified DUMPFILE=hr.dmp and REUSE_DUMPFILES=YES, then hr.dmp is overwritten if it already exists. Its previous contents are then lost, and it instead contains data for the current export.

Starting with Oracle Database 23ai when you set REUSE_DUMPFILES=YES for an export, Data Pump Export verifies that the file specified by DUMPFILE is actually an Oracle Data Pump dump file, so that it is allowed to be overwritten. If the dump file cannot be verified as an Oracle Data Pump (expdp) dump file, then you receive the message ORA-31619: 'invalid dump file'.

Example

The following export operation creates a dump file named enc1.dmp, even if a dump file with that name already exists.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=enc1.dmp
TABLES=employees REUSE_DUMPFILES=YES
```

2.4.43 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.

Default

There is no default.

Purpose

Specifies a percentage of the data rows that you want to be sampled and unloaded from the source database.

Syntax and Description

```
SAMPLE=[[schema_name.]table_name:]sample_percent
```

This parameter allows you to export subsets of data by specifying the percentage of data to be sampled and exported. The *`sample_percent`* indicates the probability that a row will be selected as part of the sample. It does not mean that the database will retrieve exactly that amount of rows from the table. The value you supply for *`sample_percent`* can be anywhere from .000001 up to, but not including, 100.

You can apply the *`sample_percent`* to specific tables. In the following example, 50% of the `HR.EMPLOYEES` table is exported:

```
SAMPLE="HR"."EMPLOYEES":50
```

If you specify a schema, then you must also specify a table. However, you can specify a table without specifying a schema. In that scenario, the current user is assumed. If no table is specified, then the *`sample_percent`* value applies to the entire export job.

You can use this parameter with the Data Pump Import `PCTSPACE` transform, so that the size of storage allocations matches the sampled data subset. (See the Import `TRANSFORM` parameter).

Restrictions

- The `SAMPLE` parameter is not valid for network exports.

Example

In the following example, the value `70` for `SAMPLE` is applied to the entire export job because no table name is specified.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=sample.dmp SAMPLE=70
```

Related Topics

- TRANSFORM

2.4.44 SCHEMAS

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

Default

The current user's schema

Purpose

Specifies that you want to perform a schema-mode export. This is the default mode for Export.

Syntax and Description

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

If you have the `DATAPUMP_EXP_FULL_DATABASE` role, then you can specify a single schema other than your own or a list of schema names. The `DATAPUMP_EXP_FULL_DATABASE` role also allows you to export additional nonschema object information for each specified schema so that the schemas can be re-created at import time. This additional information includes the user definitions themselves and all associated system and role grants, user password history, and so on. Filtering can further restrict what is exported using schema mode.

Restrictions

- If you do not have the `DATAPUMP_EXP_FULL_DATABASE` role, then you can specify only your own schema.
- The `SYS` schema cannot be used as a source schema for export jobs.

Example

The following is an example of using the `SCHEMAS` parameter. Note that user `hr` is allowed to specify more than one schema, because the `DATAPUMP_EXP_FULL_DATABASE` role was previously assigned to it for the purpose of these examples.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=expdat.dmp SCHEMAS=hr,sh,oe
```

This results in a schema-mode export in which the schemas, `hr`, `sh`, and `oe` will be written to the `expdat.dmp` dump file located in the `dpump_dir1` directory.

Related Topics

- Filtering During Export Operations

2.4.45 SERVICE_NAME

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

Default

There is no default.

Purpose

Specifies a service name that you want to use in conjunction with the `CLUSTER` parameter.

Syntax and Description

```
SERVICE_NAME=name
```

You can use the `SERVICE_NAME` parameter 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, the resource group is a subset of all the Oracle RAC instances.

The service name is only used to determine the resource group, and the 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.

If `CLUSTER=NO` is also specified, then the `SERVICE_NAME` parameter is ignored

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 is true:

- If you start an Oracle Data Pump job on instance A, and specify `CLUSTER=YES` (or accept the default, which is `Y`), 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 a 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 a 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 a Data Pump job on instance A, and specify `CLUSTER=NO`, then any `SERVICE_NAME` parameter that you specify is ignored. All processes start on instance A.

Example

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

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=hr_svname2.dmp SERVICE_NAME=sales
```

This example starts a schema-mode export (the default mode) 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`. A dump file named `hr_svname2.dmp` is written to the location specified by the `dpump_dir1` directory object.

Related Topics

- CLUSTER

2.4.46 SOURCE_EDITION

The Oracle Data Pump Export command-line utility `SOURCE_EDITION` parameter specifies the database edition from which objects are exported.

Default: the default database edition on the system

Purpose

Specifies the database edition from which objects are exported.

Syntax and Description

```
SOURCE_EDITION=edition_name
```

If `SOURCE_EDITION=`*`edition_name`* is specified, then the objects from that edition are exported. 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

- 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 `SOURCE_EDITION` parameter:

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=exp_dat.dmp SOURCE_EDITION=exp_edition
EXCLUDE=USER
```

This example assumes the existence of an edition named `exp_edition` on the system from which objects are being exported. Because no export mode is specified, the default of schema mode will be used. The `EXCLUDE=user` parameter excludes only the definitions of users, not the objects contained within users' schemas.

Related Topics

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

2.4.47 STATUS

The Oracle Data Pump Export command-line utility `STATUS` parameter specifies the frequency at which the job status display is updated.

Default

```
0
```

Purpose

Specifies the frequency at which the job status display is updated.

Syntax and Description

`STATUS=[`*`integer`*`]`

If you supply a value for *`integer`*, 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.

```
> expdp hr DIRECTORY=dpump_dir1 SCHEMAS=hr,sh STATUS=300
```

This example exports the `hr` and `sh` schemas, and displays the status of the export every 5 minutes (60 seconds x 5 = 300 seconds).

2.4.48 TABLES

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

Default

There is no default.

Purpose

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

Syntax and Description

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

Filtering can restrict what is exported using this mode. You can filter the data and metadata that is exported by specifying a comma-delimited list of tables and partitions or subpartitions. If a partition name is specified, then it must be the name of a partition or subpartition in the associated table. Only the specified set of tables, partitions, and their dependent objects are unloaded.

If an entire partitioned table is exported, then it is imported in its entirety as a partitioned table. The only case in which this is not true is if `PARTITION_OPTIONS=DEPARTITION` is specified during import.

The table name that you specify can be preceded by a qualifying schema name. The schema defaults to that of the current user. To specify a schema other than your own, you must have the `DATAPUMP_EXP_FULL_DATABASE` role.

Use of the wildcard character (`%`) to specify table names and partition names is supported.

The following restrictions apply to table names:

- By default, table names in a database are stored as uppercase. If you have a table name in mixed-case or lowercase, 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 are preceded by an escape character. The following examples show of how case-sensitivity can be preserved in the different Export 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 the Data Pump Export utility interprets the rest of the line as a comment, unless the table name is enclosed in quotation marks.

  For example, if the parameter file contains the following line, then Data Pump Export interprets everything on the line after `emp#` as a comment, and does not export the tables `dept` and `mydata`:

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

  However, if the parameter file contains the following line, then the Data Pump Export utility exports all three tables, because `emp#` is enclosed in quotation marks:

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

> **Note:**
>
> Some operating systems use single quotation marks as escape characters, rather than double quotation marks, and others the reverse. Refer to your operating system-specific 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 be able to use such characters in the name and have them ignored by the shell, and used by Export.

Using the Transportable Option During Table-Mode Export

To use the transportable option during a table-mode export, specify the `TRANSPORTABLE=ALWAYS` parameter with the `TABLES` parameter. Metadata for the specified tables, partitions, or subpartitions is exported to the dump file. To move the actual data, you copy the data files to the target database.

If only a subset of a table's partitions are exported and the `TRANSPORTABLE=ALWAYS` parameter is used, then on import each partition becomes a non-partitioned table.

Restrictions

- Cross-schema references are not exported. For example, a trigger defined on a table within one of the specified schemas, but that resides in a schema not explicitly specified, is not exported.
- Types used by the table are not exported in table mode. This restriction means that if you subsequently import the dump file, and the type does not already exist in the destination database, then the table creation fails.
- 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 is not valid to use `TABLES=regn`. If you attempt to use the synonym, then an error is returned.
- The export of tables that include a wildcard character (`%`) in the table name is not supported if the table has partitions.
- 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.
- You can only specify partitions from one table if `TRANSPORTABLE=ALWAYS` is also set on the export.

Examples

The following example shows a simple use of the `TABLES` parameter to export three tables found in the `hr` schema: `employees`, `jobs`, and `departments`. Because user `hr` is exporting tables found in the `hr` schema, the schema name is not needed before the table names.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=tables.dmp
TABLES=employees,jobs,departments
```

The following example assumes that user `hr` has the `DATAPUMP_EXP_FULL_DATABASE` role. It shows the use of the `TABLES` parameter to export partitions.

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

This example exports the partitions, `sales_Q1_2012` and `sales_Q2_2012`, from the table `sales` in the schema `sh`.

Related Topics

- Filtering During Export Operations
- TRANSPORTABLE
  The Oracle Data Pump Export command-line utility `TRANSPORTABLE` parameter specifies whether the transportable option should be used during a table mode or full mode export.
- REMAP_TABLE
  The Oracle Data Pump Import command-line mode `REMAP_TABLE` parameter enables you to rename tables during an import operation.
- Using Data File Copying to Move Data

2.4.49 TABLESPACES

The Oracle Data Pump Export command-line utility `TABLESPACES` parameter specifies a list of tablespace names that you want to be exported in tablespace mode.

Default

There is no default.

Purpose

Specifies a list of tablespace names that you want to be exported in tablespace mode.

Syntax and Description

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

In tablespace mode, only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, then its dependent objects are also unloaded. Both object metadata and data are unloaded. If any part of a table resides in the specified set, then that table and all of its dependent objects are exported. Privileged users get all tables. Unprivileged users obtain only the tables in their own schemas

Filtering can restrict what is exported using this mode.

Restrictions

The length of the tablespace name list specified for the `TABLESPACES` parameter is limited to a maximum of 4 MB, unless you are using the `NETWORK_LINK` 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 is an example of using the `TABLESPACES` parameter. The example assumes that tablespaces `tbs_4`, `tbs_5`, and `tbs_6` already exist.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=tbs.dmp
TABLESPACES=tbs_4, tbs_5, tbs_6
```

This command results in a tablespace export in which tables (and their dependent objects) from the specified tablespaces (`tbs_4`, `tbs_5`, and `tbs_6`) is unloaded.

Related Topics

- Filtering During Export Operations

2.4.50 TRANSPORT_DATAFILES_LOG

The Oracle Data Pump Export command-line mode `TRANSPORT_DATAFILES_LOG` parameter specifies a file into which the list of data files associated with a transportable export is written.

Default

None

Purpose

Specifies a file into which the list of data files associated with a transportable export is written.

Syntax and Description

```
TRANSPORT_DATAFILES_LOG=[directory_object:]file_name
```

If you specify a `directory_object`, then it must be an object that was previously established in the database and to which you have access. This parameter overrides the directory object specified with the `DIRECTORY` parameter. There is no default for the log file *`file_name`*. If specified, the file is created in the directory object specified in the `DIRECTORY` parameter, unless you explicitly specify another `directory_object`.

> **Note:**
>
> 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: `tdl`, `TDL`, `log`, `LOG`, `lst`, or `LST`. If the file name 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.

Usage Notes

The specified file written to as the `TRANSPORT_DATAFILES_LOG` file is formatted as an Oracle Data Pump parameter file. You can modify this file to add any other parameters you want to use, and specify this file as the value of the `PARFILE` parameter on a subsequent import.

Restrictions

This parameter is valid for transportable mode exports

Example

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

```
 > expdp hr DIRECTORY=dpump_dir DUMPFILE=tts.dmp
TRANSPORT_TABLESPACE=tbs_1, tbs_2 TRANSPORT_DATAFILES_LOG=tts.tdl
```

The following is an example of a file generated as the output using the `TRANSPORT_DATAFILES_LOG` parameter. In the example, *`target_database_area_path`* is the path to the tablespace file::

```
#
#
*******************************************************************************
#  The dump file set and data files must be copied to the target database
area.
#  The data file paths must be updated accordingly before initiating the
Import.
#
```

```
******************************************************************************
#
# Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
#   dpumpdir1:ttbs.dmp
#
# Datafiles required for transportable tablespace TBS1:
#   /oracle/dbs/tbs1.dbf
#
# Datafiles required for transportable tablespace TBS2:
#   /oracle/dbs/tbs2.dbf
#
#
TRANSPORT_DATAFILES=
'target_database_area_pathtbs1.dbf'
'target_database_area_pathtbs2.dbf'
```

2.4.51 TRANSPORT_FULL_CHECK

The Oracle Data Pump Export command-line utility `TRANSPORT_FULL_CHECK` parameter specifies whether to check for dependencies between objects

Default

`NO`

Purpose

Specifies whether to check for dependencies between those objects inside the transportable set and those outside the transportable set. This parameter is applicable only to a transportable-tablespace mode export.

Syntax and Description

```
TRANSPORT_FULL_CHECK=[YES | NO]
```

If `TRANSPORT_FULL_CHECK=YES`, then the Data Pump Export 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 export 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 Export 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 export operation is terminated.

There are other checks performed as well. For instance, Data Pump Export 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.

There are two current command line parameters that control full closure check:

```
TTS_FULL_CHECK=[YES|NO]
TRANSPORT_FULL_CHECK=[YES|NO]
```

`[TTS|TRANSPORT]_FULL_CHECK=YES` is interpreted as `TTS_CLOSURE_CHECK=FULL`. `[TTS|TRANSPORT]_FULL_CHECK=NO` is interpreted as `TTS_CLOSURE_CHECK=ON`.

Example

The following is an example of using the `TRANSPORT_FULL_CHECK` parameter. It assumes that tablespace `tbs_1` exists.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=tts.dmp
TRANSPORT_TABLESPACES=tbs_1 TRANSPORT_FULL_CHECK=YES LOGFILE=tts.log
```

2.4.52 TRANSPORT_TABLESPACES

The Oracle Data Pump Export command-line utility `TRANSPORT_TABLESPACES` parameter specifies that you want to perform an export in transportable-tablespace mode.

Default

There is no default.

Purpose

Specifies that you want to perform an export in transportable-tablespace mode.

Syntax and Description

```
TRANSPORT_TABLESPACES=tablespace_name [, ...]
```

Use the `TRANSPORT_TABLESPACES` parameter to specify a list of tablespace names for which object metadata will be exported from the source database into the target database.

The log file for the export lists the data files that are used in the transportable set, the dump files, and any containment violations.

The `TRANSPORT_TABLESPACES` parameter exports metadata for all objects within the specified tablespaces. If you want to perform a transportable export of only certain tables, partitions, or subpartitions, then you must use the `TABLES` parameter with the `TRANSPORTABLE=ALWAYS` parameter.

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.

Restrictions

- Transportable tablespace jobs are no longer restricted to a degree of parallelism of 1.

- Transportable tablespace mode requires that you have the `DATAPUMP_EXP_FULL_DATABASE` role.
- The default tablespace of the user performing the export must not be set to one of the tablespaces being transported.
- The `SYSTEM` and `SYSAUX` tablespaces are not transportable in transportable tablespace mode.
- All tablespaces in the transportable set must be set to read-only.
- If the Data Pump Export `VERSION` parameter is specified along with the `TRANSPORT_TABLESPACES` parameter, then the version must be equal to or greater than the Oracle Database `COMPATIBLE` initialization parameter.
- The `TRANSPORT_TABLESPACES` parameter cannot be used in conjunction with the `QUERY` parameter.
- Transportable tablespace jobs do not support the `ACCESS_METHOD` parameter for Data Pump Export.

Example

The following is an example of using the `TRANSPORT_TABLESPACES` parameter in a file-based job (rather than network-based). The tablespace `tbs_1` is the tablespace being moved. This example assumes that tablespace `tbs_1` exists and that it has been set to read-only. This example also assumes that the default tablespace was changed before this export command was issued.

```
> expdp hr DIRECTORY=dpump_dir1 DUMPFILE=tts.dmp
TRANSPORT_TABLESPACES=tbs_1 TRANSPORT_FULL_CHECK=YES LOGFILE=tts.log
```

See *Oracle Database Administrator's Guide* for detailed information about transporting tablespaces between databases

Related Topics

- Transportable Tablespace Mode
- Using Data File Copying to Move Data
- How Does Oracle Data Pump Handle Timestamp Data?
- Transporting Tablespaces Between Databases in *Oracle Database Administrator’s Guide*

2.4.53 TRANSPORTABLE

The Oracle Data Pump Export command-line utility `TRANSPORTABLE` parameter specifies whether the transportable option should be used during a table mode or full mode export.

Default

`NEVER`

Purpose

Specifies whether the transportable option should be used during a table mode export (specified with the `TABLES` parameter) or a full mode export (specified with the `FULL` parameter).

Syntax and Description

```
TRANSPORTABLE = [ALWAYS | NEVER]
```

The definitions of the allowed values are as follows:

`ALWAYS` - Instructs the export job to use the transportable option. If transportable is not possible, then the job fails.

In a table mode export, using the transportable option results in a transportable tablespace export in which metadata for only the specified tables, partitions, or subpartitions is exported.

In a full mode export, using the transportable option results in a full transportable export which exports all objects and data necessary to create a complete copy of the database.

`NEVER` - Instructs the export job to use either the direct path or external table method to unload data rather than the transportable option. This is the default.

> **Note:**
>
> To export an entire tablespace in transportable mode, use the `TRANSPORT_TABLESPACES` parameter.

- If only a subset of a table's partitions are exported and the `TRANSPORTABLE=ALWAYS` parameter is used, then on import each partition becomes a non-partitioned table.
- If only a subset of a table's partitions are exported and the `TRANSPORTABLE` parameter is *not* used at all or is set to `NEVER` (the default), then on import:
  - If `PARTITION_OPTIONS=DEPARTITION` is used, then each partition included in the dump file set is created as a non-partitioned table.
  - If `PARTITION_OPTIONS` is not used, then the complete table is created. That is, all the metadata for the complete table is present, so that the table definition looks the same on the target system as it did on the source. But only the data that was exported for the specified partitions is inserted into the table.

Restrictions

- The `TRANSPORTABLE` parameter is only valid in table mode exports and full mode exports.
- To use the `TRANSPORTABLE` parameter, the `COMPATIBLE` initialization parameter must be set to at least 11.0.0.
- To use the `FULL` parameter in conjunction with `TRANSPORTABLE` (to perform a full transportable export), the Data Pump `VERSION` parameter must be set to at least 12.0. If the `VERSION` parameter is not specified, then the `COMPATIBLE` database initialization parameter must be set to at least 12.0 or later.
- The user performing a transportable export requires the `DATAPUMP_EXP_FULL_DATABASE` privilege.
- Tablespaces associated with tables, partitions, and subpartitions must be read-only.
- A full transportable export uses a mix of data movement methods. Objects residing in a transportable tablespace have only their metadata unloaded; data is copied when the data files are copied from the source system to the target system. The data files that must be copied are listed at the end of the log file for the export operation. Objects residing in non-

transportable tablespaces (for example, `SYSTEM` and `SYSAUX`) have both their metadata and data unloaded into the dump file set. (See *Oracle Database Administrator's Guide* for more information about performing full transportable exports.)

- The default tablespace of the user performing the export must not be set to one of the tablespaces being transported.

Example

The following example assumes that the `sh` user has the `DATAPUMP_EXP_FULL_DATABASE` role and that table `sales2` is partitioned and contained within tablespace `tbs2`. (The `tbs2` tablespace must be set to read-only in the source database.)

```
> expdp sh DIRECTORY=dpump_dir1 DUMPFILE=tto1.dmp
TABLES=sh.sales2 TRANSPORTABLE=ALWAYS
```

After the export completes successfully, you must copy the data files to the target database area. You could then perform an import operation using the `PARTITION_OPTIONS` and `REMAP_SCHEMA` parameters to make each of the partitions in `sales2` its own table.

```
> impdp system PARTITION_OPTIONS=DEPARTITION
TRANSPORT_DATAFILES=oracle/dbs/tbs2 DIRECTORY=dpump_dir1
DUMPFILE=tto1.dmp REMAP_SCHEMA=sh:dp
```

Related Topics

- Transporting Databases in *Oracle Database Administrator's Guide*
- Full Export Mode
- Using Data File Copying to Move Data

2.4.54 TTS_CLOSURE_CHECK

The Oracle Data Pump Export command-line mode `TTS_CLOSURE_CHECK` parameter is used to indicate the degree of closure checking to be performed as part of a Data Pump transportable tablespace operation.

Default

There is no default.

Purpose

Specifies the level of closure check that you want to be performed as part of the transportable export operation. The `TTS_CLOSURE_CHECK` parameter can also be used to indicate that tablespaces can remain read-write during a test mode transportable tablespace operation. This option is used to obtain the timing requirements of the export operation. It is for testing purposes only. The dump file is unavailable for import.

Syntax and Description

```
TTS_CLOSURE_CHECK = [ ON | OFF | FULL | REKEY_OFF | TEST_MODE ]
```

The `TTS_CLOSURE_CHECK` parameter supports the following options:

- `ON` - specifies that the self-containment closure check is performed
- `OFF` - specifies that no closure check is performed
- `FULL` - specifies that full bidirectional closure check is performed

- `REKEY_OFF` - specifies that the concurrent rekey check is not performed as part of the transportable allowable checks
- `TEST_MODE` - specifies that tablespaces are not required to be in read-only mode

The `ON`,`OFF`, `FULL` and `REKEY_OFF` options are mutually exclusive. `TEST_MODE` and `REKEY_OFF` are only Oracle Data Pump Export options.

Example

```
TTS_CLOSURE_CHECK=FULL
```

2.4.55 VERSION

The Oracle Data Pump Export command-line utility `VERSION` parameter specifies the version of database objects that you want to export.

Default

```
COMPATIBLE
```

Purpose

Specifies the version of database objects that you want to export. Only database objects and attributes that are compatible with the specified release are exported. You can use the `VERSION` parameter to create a dump file set that is compatible with a previous release of Oracle Database. You cannot use Data Pump Export with releases of Oracle Database before Oracle Database 10g release 1 (10.1). Data Pump Export only works with Oracle Database 10g release 1 (10.1) or later. The `VERSION` parameter simply allows you to identify the version of objects that you export.

Starting with Oracle Database 23ai, if you want to use Header Blocks for dump files, then you must use `VERSION` to specify a compatible version Dump files created with `VERSION=23` cannot be imported into an earlier relese. However, Data Pump can continue to import from earlier releases using Header Blocks into Oracle Database 23ai.

On Oracle Database 11g release 2 (11.2.0.3) or later, you can specify the `VERSION` parameter as `VERSION=12` with `FULL=Y` to generate a full export dump file that is ready for import into Oracle Database 12c. The export with the later release target `VERSION` value includes information from registered database options and components. The dump file set specifying a later release version can only be imported into Oracle Database 12c Release 1 (12.1.0.1) and later. For example, if `VERSION=12` is used with `FULL=Y` and also with `TRANSPORTABLE=ALWAYS`, then a full transportable export dump file is generated that is ready for import into Oracle Database 12c. For more information, refer to the `FULL` export parameter option.

Syntax and Description

```
VERSION=[COMPATIBLE | LATEST | version_string]
```

The legal values for the `VERSION` parameter are as follows:

- `COMPATIBLE` - This value is the default value. The version of the metadata corresponds to the database compatibility level as specified on the `COMPATIBLE` initialization parameter.

  Note: Database compatibility must be set to 9.2 or later.
- `LATEST` - The version of the metadata and resulting SQL DDL corresponds to the database release, regardless of its compatibility level.

- `version_string` - A specific database release (for example, 11.2.0). In Oracle Database 11g, this value cannot be lower than 9.2.

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"
```

Restrictions

- Exporting a table with archived LOBs to a database release earlier than 11.2 is not allowed.
- If the Data Pump Export `VERSION` parameter is specified with the `TRANSPORT_TABLESPACES` parameter, then the value for `VERSION` must be equal to or greater than the Oracle Database `COMPATIBLE` initialization parameter.
- If the Data Pump `VERSION` parameter is specified as any value earlier than 12.1, then the 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.
- Dump files created on Oracle Database 11g releases with the Data Pump parameter `VERSION=12` can only be imported on Oracle Database 12c Release 1 (12.1) and later.

Example

The following example shows an export for which the version of the metadata corresponds to the database release:

```
> expdp hr TABLES=hr.employees VERSION=LATEST DIRECTORY=dpump_dir1
DUMPFILE=emp.dmp NOLOGFILE=YES
```

Related Topics

- Full Export Mode
- Exporting and Importing Between Different Oracle Database Releases

2.4.56 VIEWS_AS_TABLES

The Oracle Data Pump Export command-line utility `VIEWS_AS_TABLES` parameter specifies that you want one or more views exported as tables.

Default

There is no default.

Caution:

The `VIEWS_AS_TABLES` parameter unloads view data in unencrypted format, and creates an unencrypted table. If you are unloading sensitive data, then Oracle strongly recommends that you enable encryption on the export operation, and that you ensure the table is created in an encrypted tablespace. You can use the `REMAP_TABLESPACE` parameter to move the table to such a tablespace.

Purpose

Specifies that you want one or more views exported as tables.

Syntax and Description

```
VIEWS_AS_TABLES=[schema_name.]view_name[:table_name], ...
```

Oracle Data Pump exports a table with the same columns as the view, and with row data obtained from the view. Oracle Data Pump also exports 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 exported. You can use the `VIEWS_AS_TABLES` parameter by itself, or use it with the `TABLES` parameter. Either way you use the parameter, Oracle Data Pump performs a table-mode export.

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, then it defaults to the user performing the export.

`view_name`: The name of the view that you want exported as a table.

`table_name`: The name of a table that you want to serve as the source of the metadata for the exported view. By default, Oracle Data Pump automatically creates a temporary "template table" with the same columns and data types as the view, but with 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.

If the export 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:

```
expdp scott/password directory=dpump_dir dumpfile=a.dmp
views_as_tables=v1:emp,v2:emp
```

An error message is returned reporting the omitted object.

Template tables are automatically dropped after the export 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 SCOTT.EMPV
```

Restrictions

- The `VIEWS_AS_TABLES` parameter cannot be used with the `TRANSPORTABLE=ALWAYS` parameter.
- Tables that you want to serve as the source of the metadata for the exported view must be in the same schema as the view.
- Tables that you want to serve as the source of the metadata for the exported view must be non-partitioned relational tables with heap organization.
- Tables that you want to serve as the source of the metadata for the exported view cannot be nested tables.
- Tables created using the `VIEWS_AS_TABLES` parameter do not contain any hidden or invisible columns that were part of the specified view.
- Views that you want exported as tables must exist, and must be relational views with only scalar columns. If you specify an invalid or non-existent view, then the view is skipped, and an error message is returned.
- The `VIEWS_AS_TABLES` parameter does not support tables that have columns with a data type of `LONG`.

Example

The following example exports the contents of view `scott.view1` to a dump file named `scott1.dmp`.

```
> expdp scott/password views_as_tables=view1 directory=data_pump_dir
dumpfile=scott1.dmp
```

The dump file contains a table named `view1` with rows obtained from the view.

2.5 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 2-1 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

Table 2-1 (Cont.) Supported Activities in Data Pump Export's Interactive-Command Mode

Activity	Command Used
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
```

2.6 Examples of Using Oracle Data Pump Export

You can use these common scenario examples to learn how you can create parameter files and use Oracle Data Pump Export to move your data.

- Performing a Table-Mode Export
  This example shows a table-mode export, specified using the `TABLES` parameter.
- Data-Only Unload of Selected Tables and Rows
  This example shows data-only unload of selected tables and rows.
- Estimating Disk Space Needed in a Table-Mode Export
  This example shows how to estimate the disk space needed in a table-mode export.
- Performing a Schema-Mode Export
  This example shows you how to perform a schema-mode export.
- Performing a Parallel Full Database Export
  To learn how to perform a parallel full database export, use this example to understand the syntax.
- Using Interactive Mode to Stop and Reattach to a Job
  This example shows you how to use interactive mode to stop and reattach to a job.

- Continuing Table Loads when LOB Data Type Corruptions are Found
  This example shows you how to address ORA-1555 errors with an Oracle Data Pump export job.

2.6.1 Performing a Table-Mode Export

This example shows a table-mode export, specified using the `TABLES` parameter.

In this example, the Data Pump export command performs a table export of the tables `employees` and `jobs` from the human resources (`hr`) schema.

Because user `hr` is exporting tables in his own schema, it is not necessary to specify the schema name for the tables. The `NOLOGFILE=YES` parameter indicates that an Export log file of the operation is not generated.

Example 2-1   Performing a Table-Mode Export

```
expdp hr TABLES=employees,jobs DUMPFILE=dpump_dir1:table.dmp NOLOGFILE=YES
```

2.6.2 Data-Only Unload of Selected Tables and Rows

This example shows data-only unload of selected tables and rows.

The example shows the contents of a parameter file (`exp.par`), which you can use to perform a data-only unload of all the tables in the human resources (`hr`) schema, except for the tables `countries` and `regions`. Rows in the `employees` table are unloaded that have a `department_id` other than 50. The rows are ordered by `employee_id`.

You can issue the following command to execute the `exp.par` parameter file:

```
> expdp hr PARFILE=exp.par
```

This export performs a schema-mode export (the default mode), but the `CONTENT` parameter effectively limits the export to an unload of just the table data. The DBA previously created the directory object `dpump_dir1`, which points to the directory on the server where user `hr` is authorized to read and write export dump files. The dump file `dataonly.dmp` is created in `dpump_dir1`.

Example 2-2   Data-Only Unload of Selected Tables and Rows

```
DIRECTORY=dpump_dir1
DUMPFILE=dataonly.dmp
CONTENT=DATA_ONLY
EXCLUDE=TABLE:"IN ('COUNTRIES', 'REGIONS')"
QUERY=employees:"WHERE department_id !=50 ORDER BY employee_id"
```

2.6.3 Estimating Disk Space Needed in a Table-Mode Export

This example shows how to estimate the disk space needed in a table-mode export.

In this example, the `ESTIMATE_ONLY` parameter is used to estimate the space that is consumed in a table-mode export, without actually performing the export operation. Issue the following command to use the `BLOCKS` method to estimate the number of bytes required to export the data in the following three tables located in the human resource (`hr`) schema: `employees`, `departments`, and `locations`.

The estimate is printed in the log file and displayed on the client's standard output device. The estimate is for table row data only; it does not include metadata.

Example 2-3 Estimating Disk Space Needed in a Table-Mode Export

```
> expdp hr DIRECTORY=dpump_dir1 ESTIMATE_ONLY=YES TABLES=employees,
departments, locations LOGFILE=estimate.log
```

2.6.4 Performing a Schema-Mode Export

This example shows you how to perform a schema-mode export.

The example shows a schema-mode export of the `hr` schema. In a schema-mode export, only objects belonging to the corresponding schemas are unloaded. Because schema mode is the default mode, it is not necessary to specify the `SCHEMAS` parameter on the command line, unless you are specifying more than one schema or a schema other than your own.

Example 2-4 Performing a Schema Mode Export

```
> expdp hr DUMPFILE=dpump_dir1:expschema.dmp LOGFILE=dpump_dir1:expschema.log
```

2.6.5 Performing a Parallel Full Database Export

To learn how to perform a parallel full database export, use this example to understand the syntax.

The example shows a full database Export that can use 3 parallel processes (worker or parallel query worker processes).

Example 2-5 Parallel Full Export

```
> expdp hr FULL=YES DUMPFILE=dpump_dir1:full1%U.dmp, dpump_dir2:full2%U.dmp
FILESIZE=2G PARALLEL=3 LOGFILE=dpump_dir1:expfull.log JOB_NAME=expfull
```

Because this export is a full database export, all data and metadata in the database is exported. Dump files `full101.dmp`, `full201.dmp`, `full102.dmp`, and so on, are created in a round-robin fashion in the directories pointed to by the `dpump_dir1` and `dpump_dir2` directory objects. For best performance, Oracle recommends that you place the dump files on separate input/output (I/O) channels. Each file is up to 2 gigabytes in size, as necessary. Initially, up to three files are created. If needed, more files are created. The job and Data Pump control process table has a name of `expfull`. The log file is written to `expfull.log` in the `dpump_dir1` directory.

2.6.6 Using Interactive Mode to Stop and Reattach to a Job

This example shows you how to use interactive mode to stop and reattach to a job.

To start this example, reexecute the parallel full export described here:

Performing a Parallel Full Database Export

While the export is running, press Ctrl+C. This keyboard command starts the interactive-command interface of Data Pump Export. In the interactive interface, logging to the terminal stops, and the Export prompt is displayed.

After the job status is displayed, you can issue the `CONTINUE_CLIENT` command to resume logging mode and restart the `expfull` job.

```
Export> CONTINUE_CLIENT
```

A message is displayed that the job has been reopened, and processing status is output to the client.

Example 2-6   Stopping and Reattaching to a Job

At the Export prompt, issue the following command to stop the job:

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

The job is placed in a stopped state, and exits the client.

To reattach to the job you just stopped, enter the following command:

```
> expdp hr ATTACH=EXPFULL
```

2.6.7 Continuing Table Loads when LOB Data Type Corruptions are Found

This example shows you how to address ORA-1555 errors with an Oracle Data Pump export job.

Suppose you have a table with large object datatype (LOB) columns (BLOB, CLOB, NCLOB or BFILE) that has a large number of rows that require several hours to complete. During the export job, Oracle Data Pump encounters an ORA-1555 error ("ORA-01555: snapshot too old: rollback segment number with name "" too small"). Oracle recommends that you do not attempt to export partial rows, because attempting this workaround can cause further corruption. Instead, before exporting the LOB table, Oracle recommends that you use the script in this example to find the corrupted LOB rowids, and then repair the table by either emptying those rows before export, or excluding those rows from the export.

Example 2-7   Finding LOB Corruption in Large Tables

Use this script to verify LOB corruption, and to find and store the corrupted LOB IDs in a temporary table so that you can then exclude them from the table you want to export.

1. Create a new temporary table for storing all `rowids` called `corrupt_lobs`

   ```
   SQL> create table corrupt_lobs (corrupt_rowid rowid, err_num number);
   ```

2. Make a `desc` on the large table *`<TABLE_NAME>`* containing the LOB column:

   ```
   DESC <TABLE_NAME>

   Name          Null?     Type
   ----------    --------- ------------
   <COL1>        NOT NULL  NUMBER
   <LOB_COLUMN>            BLOB
   ```

Run the following PL/SQL block:

```
declare
  error_1578 exception;
  error_1555 exception;
  error_22922 exception;
  pragma exception_init(error_1578,-1578);
  pragma exception_init(error_1555,-1555);
  pragma exception_init(error_22922,-22922);
  n number;
begin
  for cursor_lob in (select rowid r, <LOB_COLUMN> from <TABLE_NAME>) loop
  begin
    n:=dbms_lob.instr(cursor_lob.<LOB_COLUMN>,hextoraw('889911'));
  exception
    when error_1578 then
      insert into corrupt_lobs values (cursor_lob.r, 1578);
      commit;
    when error_1555 then
      insert into corrupt_lobs values (cursor_lob.r, 1555);
      commit;
    when error_22922 then
      insert into corrupt_lobs values (cursor_lob.r, 22922);
      commit;
    end;
  end loop;
end;
/
```

The result of this PL/SQL script is that all rowids of the corrupted LOBs will be inserted into the newly created `corrupt_lobs` table.

3. Resolve the issue with the corrupted LOB `rowids` either by emptying the corrupted LOB rows, or by exporting the table without the corrupted LOB rows.

   - **Empty the corrupted LOB rows**

     With this option, you run a SQL statement to empty the rows. In this example, the rows that you select are BLOB or BFILE columns, so we use `EMPTY_BLOB`. For CLOB and NCLOB columns, or use `EMPTY_CLOB`:

     ```
     SQL> update <TABLE_NAME> set <LOB_COLUMN> = empty_blob()
          where rowid in (select corrupt_rowid from corrupt_lobs);
     ```

   - **Export the table without the corrupted LOB rows**

     Use this script with values for your environment:

     ```
     $ expdp system/<PASSWORD> DIRECTORY=my_dir DUMPFILE=<dump_name>.dmp
     LOGFILE=<logfile_name>.log TABLES=<SCHEMA_NAME>.<TABLE_NAME> QUERY=\"WHERE
     rowid NOT IN \(\'<corrupt_rowid>\'\)\"
     ```

For more information about identifying and resolving ORA-1555 errors, and distinguishing them from LOB segment issues due to LOB PCTVERSION or RETENTION being low, see the My Oracle Support document "Export Receives The Errors ORA-1555 ORA-22924 ORA-1578 ORA-22922 (Doc ID 787004.1"

Related Topics

- Export Receives The Errors ORA-1555 ORA-22924 ORA-1578 ORA-22922 (Doc ID 787004.1

2.7 Syntax Diagrams for Oracle Data Pump Export

You can use syntax diagrams to understand the valid SQL syntax for Oracle Data Pump Export.

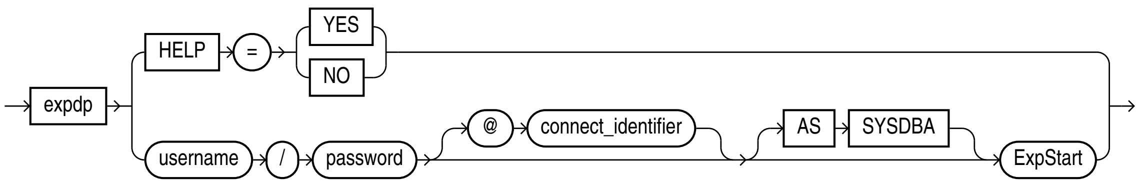
How to Read Graphic Syntax Diagrams

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.

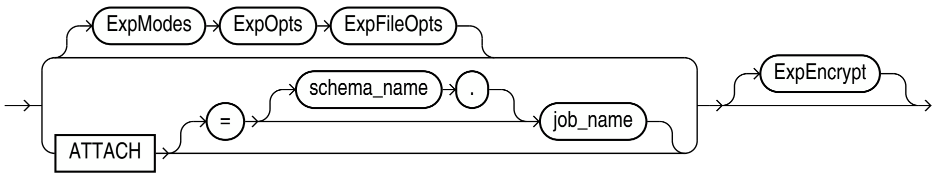
For more information about standard SQL syntax notation, see:

How to Read Syntax Diagrams in *Oracle Database SQL Language Reference*

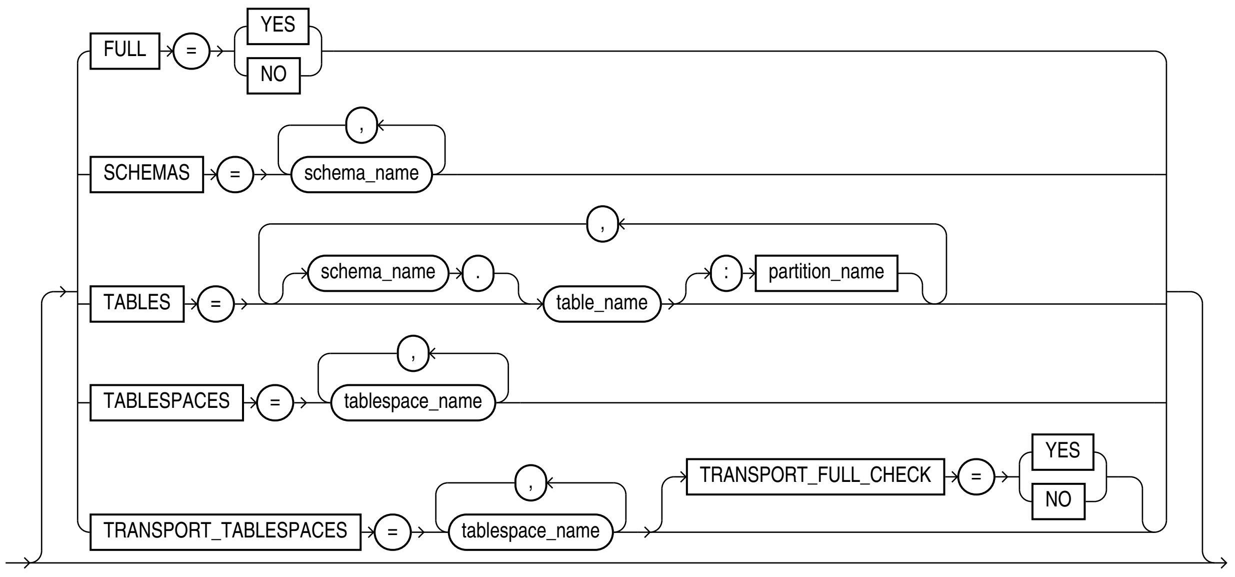
ExpInit



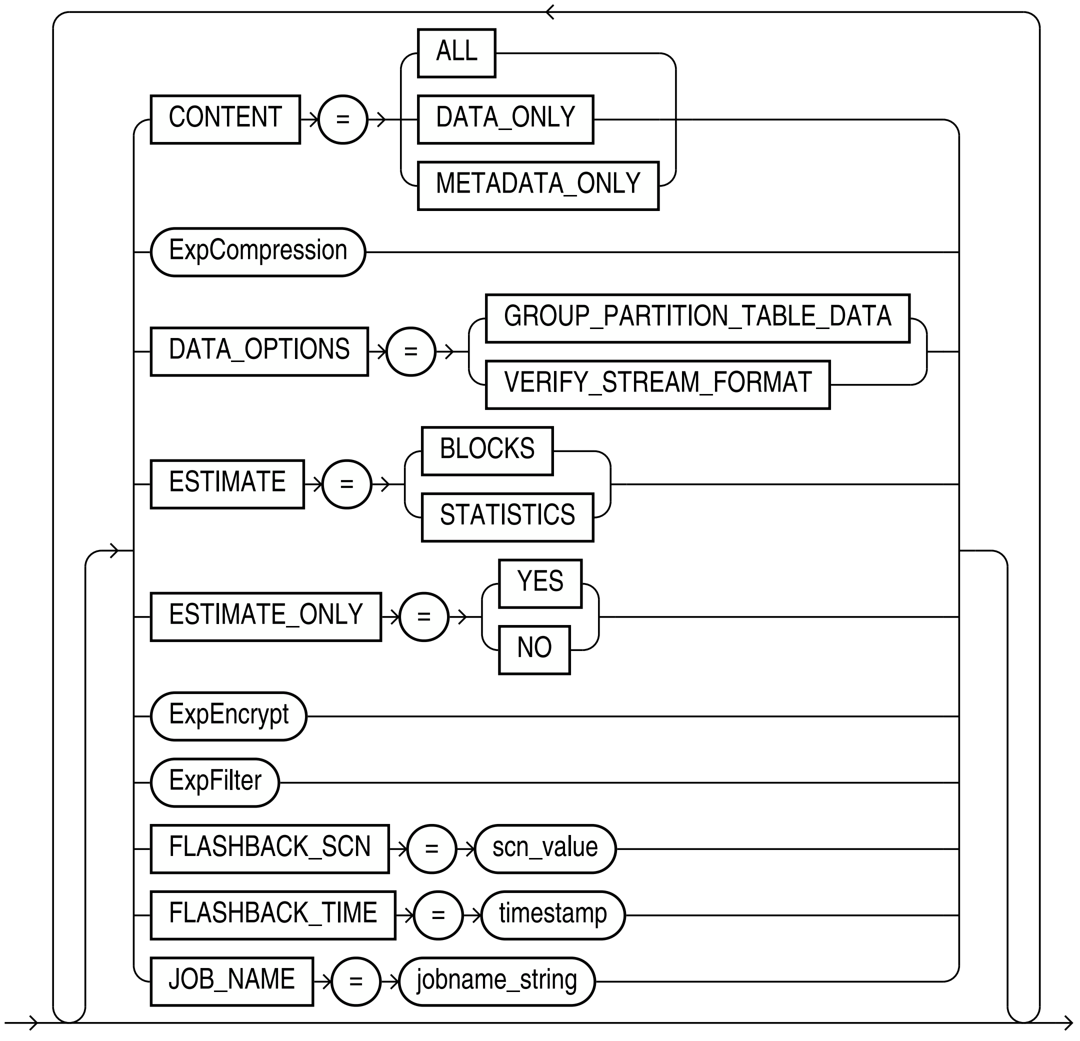
ExpStart



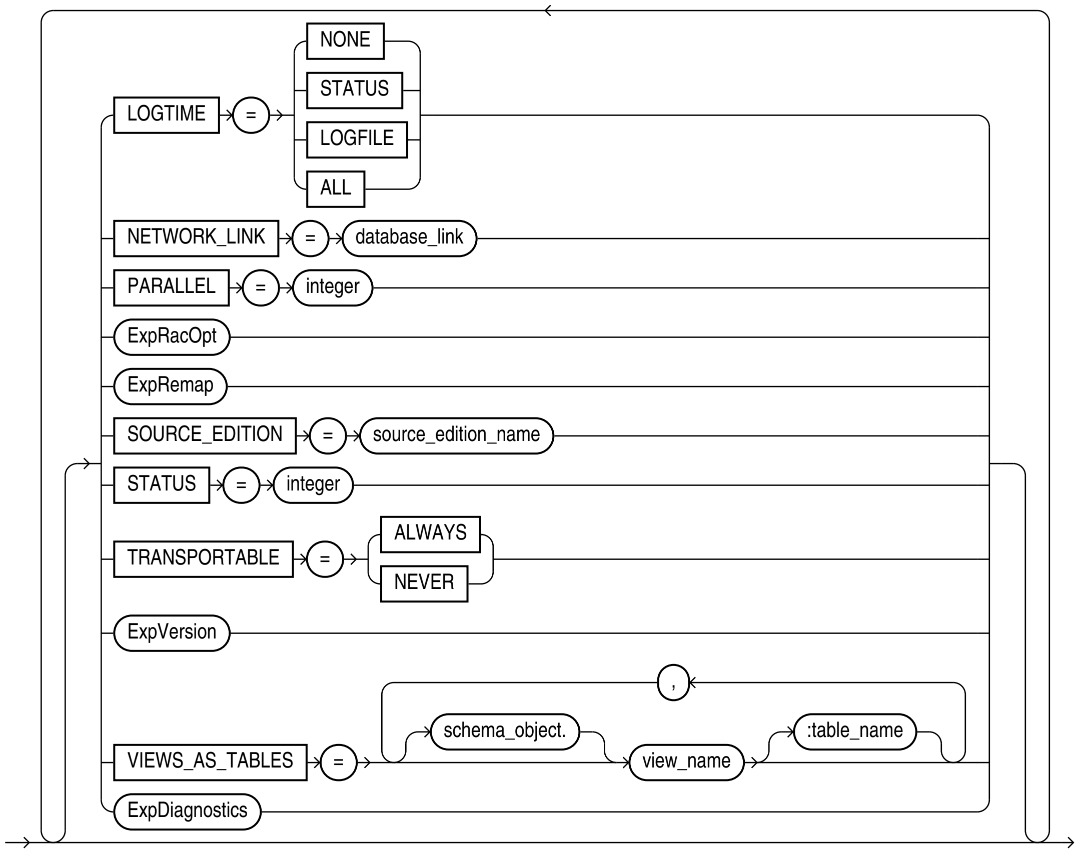
ExpModes


FULL
=
YES
NO
SCHEMAS
=
,
schema_name
TABLES
=
,
schema_name
.
table_name
:
partition_name
TABLESPACES
=
,
tablespace_name
TRANSPORT_TABLESPACES
=
,
tablespace_name
TRANSPORT_FULL_CHECK
=
YES
NO

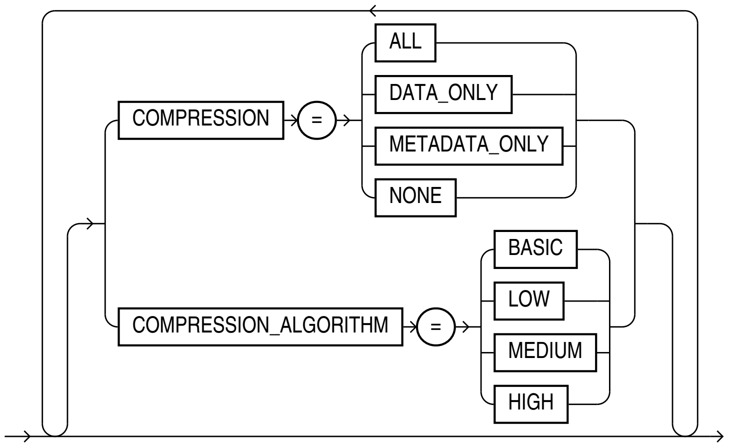
ExpOpts


CONTENT
=
ALL
DATA_ONLY
METADATA_ONLY
ExpCompression
DATA_OPTIONS
=
GROUP_PARTITION_TABLE_DATA
VERIFY_STREAM_FORMAT
ESTIMATE
=
BLOCKS
STATISTICS
ESTIMATE_ONLY
=
YES
NO
ExpEncrypt
ExpFilter
FLASHBACK_SCN
=
scn_value
FLASHBACK_TIME
=
timestamp
JOB_NAME
=
jobname_string

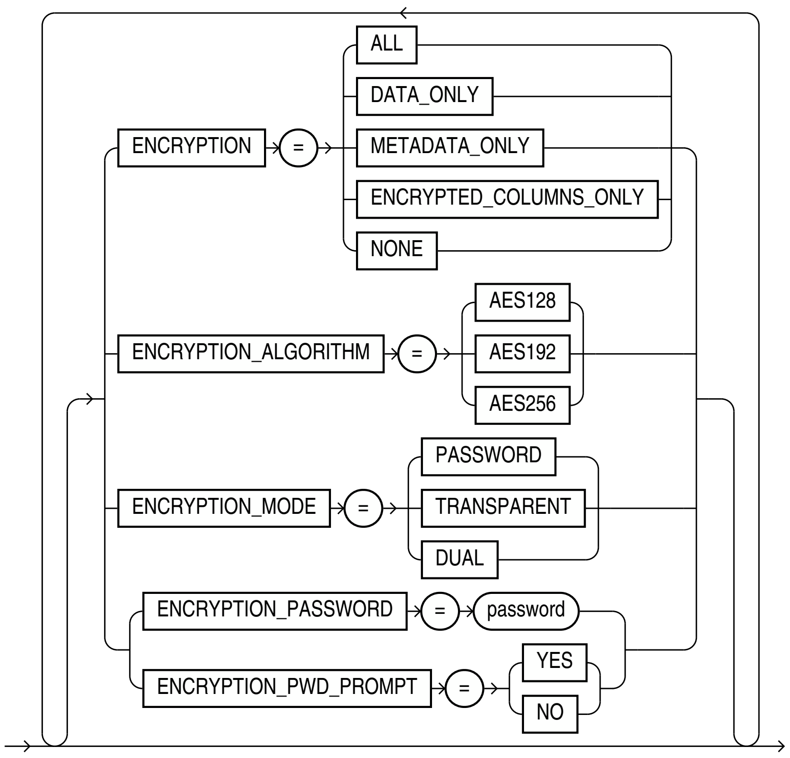
ExpOpts_Cont



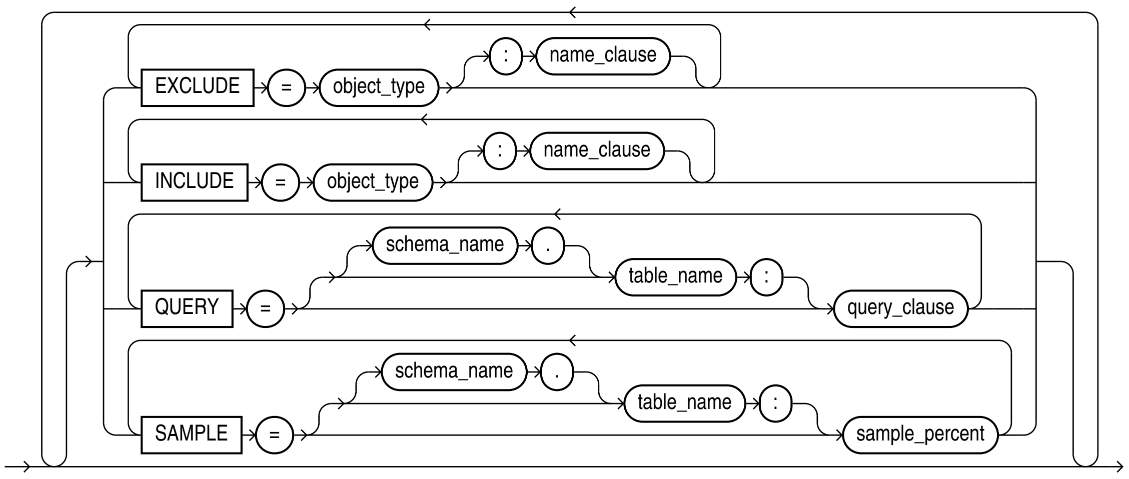
ExpCompression



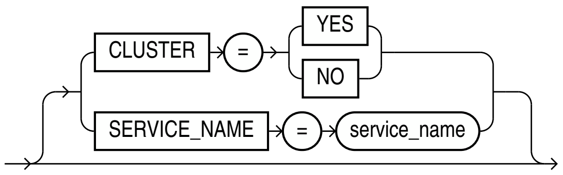
ExpEncrypt



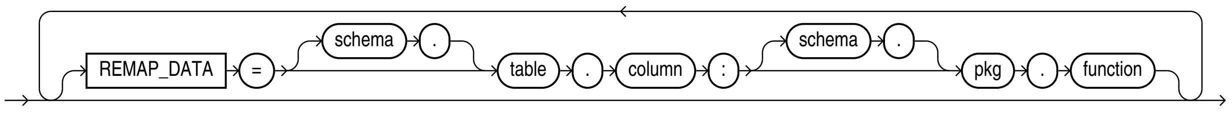
ExpFilter



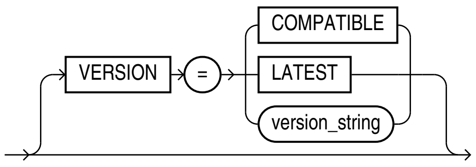
ExpRacOpt



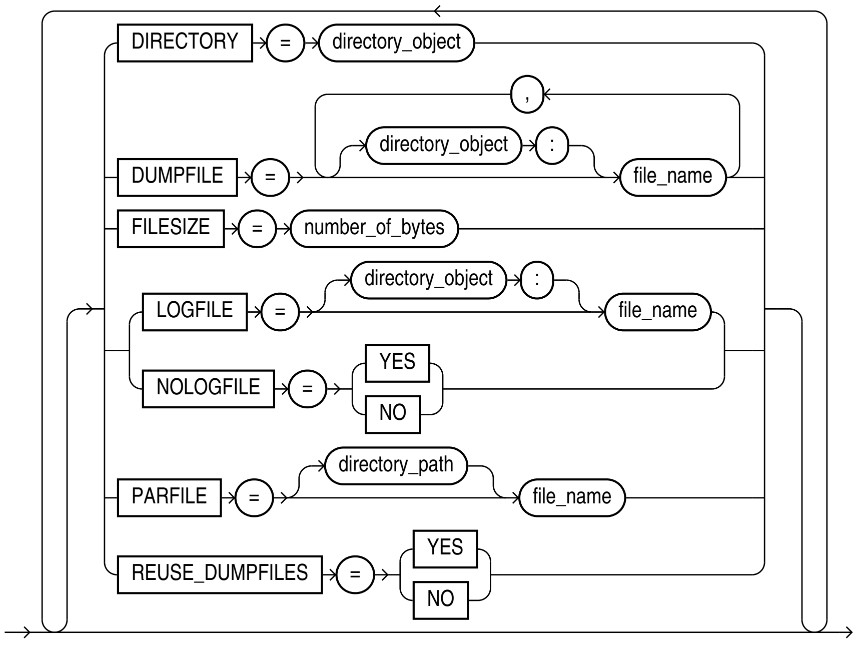
ExpRemap



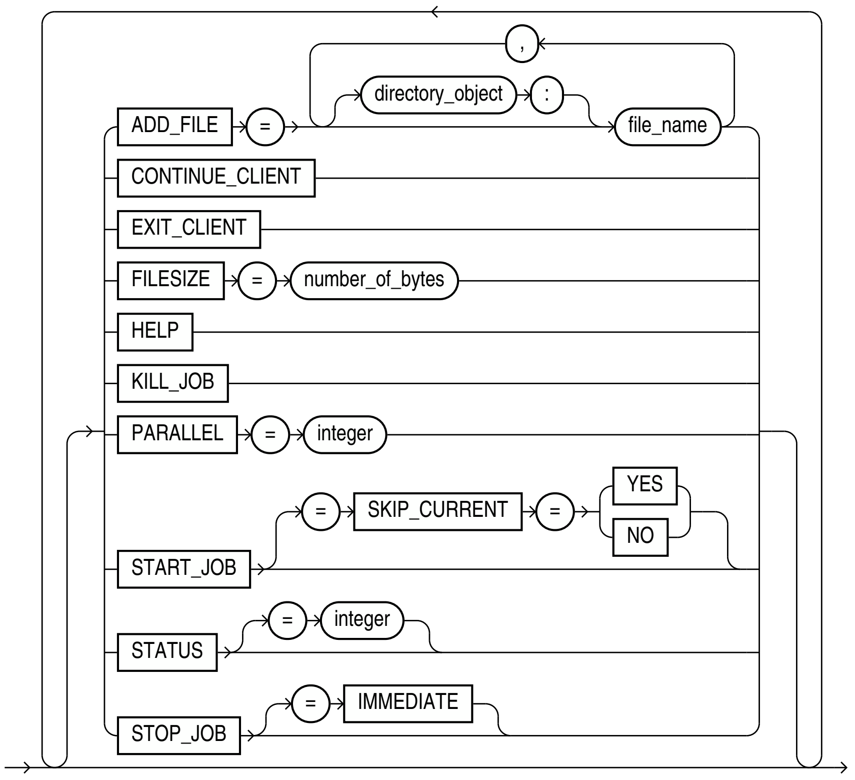
ExpVersion



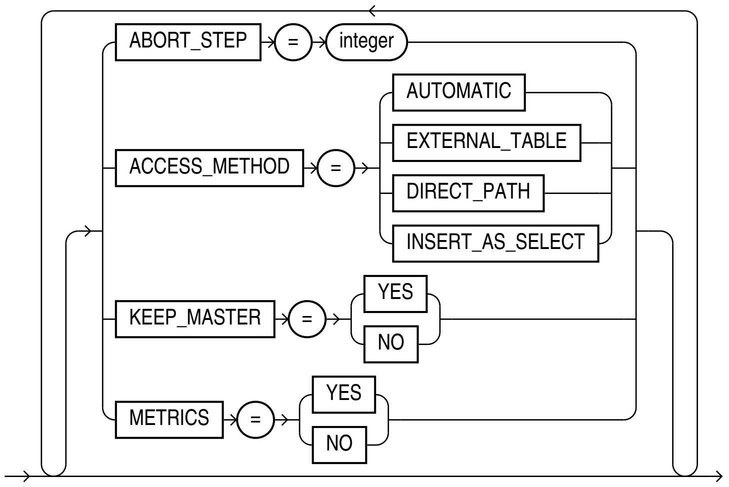
ExpFileOpts



ExpDynOpts



ExpDiagnostics


ABORT_STEP
=
integer
ACCESS_METHOD
=
AUTOMATIC
EXTERNAL_TABLE
DIRECT_PATH
INSERT_AS_SELECT
KEEP_MASTER
=
YES
NO
METRICS
=
YES
NO