1

Overview of Oracle Data Pump

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another.

An understanding of the following topics can help you to successfully use Oracle Data Pump to its fullest advantage:

- Oracle Data Pump Components
  Oracle Data Pump is made up of three distinct components: Command-line clients, `expdp` and `impdp`; the `DBMS_DATAPUMP` PL/SQL package (also known as the Data Pump API); and the `DBMS_METADATA` PL/SQL package (also known as the Metadata API).
- How Does Oracle Data Pump Move Data?
  There are several Oracle Data Pump methods that you can use to move data in and out of databases. You can select the method that best fits your use case.
- Using Oracle Data Pump With CDBs
  Oracle Data Pump can migrate all, or portions of, a database from a non-CDB into a PDB, between PDBs within the same or different CDBs, and from a PDB into a non-CDB.
- Cloud Premigration Advisor Tool
  The Cloud Premigration Advisor tool can assist you to migrate a database to the Oracle Cloud.
- Required Roles for Oracle Data Pump Export and Import Operations
  The roles `DATAPUMP_EXP_FULL_DATABASE` and `DATAPUMP_IMP_FULL_DATABASE` are required for many Export and Import operations.
- What Happens During the Processing of an Oracle Data Pump Job?
  Oracle Data Pump jobs use a Data Pump control job table, a Data Pump control job process, and worker processes to perform the work and keep track of progress.
- How to Monitor Status of Oracle Data Pump Jobs
  The Oracle Data Pump Export and Import client utilities can attach to a job in either logging mode or interactive-command mode.
- How to Monitor the Progress of Running Jobs with V$SESSION_LONGOPS
  To monitor table data transfers, you can use the `V$SESSION_LONGOPS` dynamic performance view to monitor Oracle Data Pump jobs.
- File Allocation with Oracle Data Pump
  You can modify how Oracle Data Pump allocates and handles files by using commands in interactive mode.
- Exporting and Importing Between Different Oracle Database Releases
  You can use Oracle Data Pump to migrate all or any portion of an Oracle Database between different releases of the database software.
- Exporting and Importing Blockchain Tables with Oracle Data Pump
  To export or import blockchain tables, review these minimum requirements, restrictions, and guidelines.
- Unload and Load Vectors Using Oracle Data Pump
  Starting with Oracle Database 23ai, Oracle Data Pump enables you to use multiple components to load and unload vectors to databases.

- Managing SecureFiles Large Object Exports with Oracle Data Pump
  Exports of SecureFiles large objects (LOBs) are affected by the content type, the `VERSION` parameter, and other variables.
- Oracle Data Pump Process Exit Codes
  To check the status of your Oracle Data Pump export and import operations, review the process exit codes in the log file.
- How Oracle Data Pump Manages Dump File Blocks
  In releases before Oracle Database 23ai, Oracle Data Pump uses Header Blocks. Starting with Oracle Database 23ai, Oracle Data Pump uses Trailer Blocks.
- How to Monitor Oracle Data Pump Jobs with Unified Auditing
  To monitor and record specific user database actions, perform auditing on Data Pump jobs with unified auditing.
- Encrypted Data Security Warnings for Oracle Data Pump Operations
  Oracle Data Pump warns you when encrypted data is exported as unencrypted data.
- How Does Oracle Data Pump Handle Timestamp Data?
  Learn about factors that can affect successful completion of export and import jobs that involve the timestamp data types `TIMESTAMP WITH TIMEZONE` and `TIMESTAMP WITH LOCAL TIMEZONE`.
- Character Set and Globalization Support Considerations
  Learn about Globalization support of Oracle Data Pump Export and Import using character set conversion of user data, and data definition language (DDL).
- Oracle Data Pump Behavior with Data-Bound Collation
  Oracle Data Pump supports data-bound collation (DBC).

1.1 Oracle Data Pump Components

Oracle Data Pump is made up of three distinct components: Command-line clients, `expdp` and `impdp`; the `DBMS_DATAPUMP` PL/SQL package (also known as the Data Pump API); and the `DBMS_METADATA` PL/SQL package (also known as the Metadata API).

The Oracle Data Pump clients, `expdp` and `impdp`, start the Oracle Data Pump Export utility and Oracle Data Pump Import utility, respectively.

The `expdp` and `impdp` clients use the procedures provided in the `DBMS_DATAPUMP` PL/SQL package to run export and import commands, using the parameters entered at the command line. These parameters enable the exporting and importing of data and metadata for a complete database, or for subsets of a database.

> **Note:**
>
> Beginning with the Oracle Database 21c Data Pump client, you can use later release Oracle Data Pump clients with earlier Oracle Database server versions. Oracle Database 12c Release 1 (12.1) and later releases no longer require that you use the same version of the `expdp` and `impdp` clients as the Oracle Database server from which you are exporting data, or to which you are importing data.
>
> When you use the later release Data Pump client to move data on earlier release servers, new parameters that the later release client supports may not be supported by the earlier release server. If the parameters are not supported by the earlier release server, then the server will return an error and the client will report the error.
>
> It is also possible that a new feature can be added in a patch to an older release server, and a later release Data Pump client does not support that new feature. If you encounter that problem, then Oracle recommends that you use the most recent Instant Client kit for the most recent Oracle Database release. The latest release Data Pump client should have support for the new feature.

When metadata is moved, Data Pump uses functionality provided by the `DBMS_METADATA` PL/SQL package. The `DBMS_METADATA` package provides a centralized facility for the extraction, manipulation, and re-creation of dictionary metadata.

The `DBMS_DATAPUMP` and `DBMS_METADATA` PL/SQL packages can be used independently of the Data Pump clients.

> **Note:**
>
> All Oracle Data Pump Export and Import processing, including the reading and writing of dump files, is done on the system (server) selected by the specified database connect string. **This means that for unprivileged users, the database administrator (DBA) must create directory objects for the Data Pump files that are read and written on that server file system.** (For security reasons, DBAs must ensure that only approved users are allowed access to directory objects.) For privileged users, a default directory object is available.

Starting with Oracle Database 18c, you can include the unified audit trail in either full or partial export and import operations using Oracle Data Pump. There is no change to the user interface. When you perform the export or import operations of a database, the unified audit trail is automatically included in the Oracle Data Pump dump files. See *Oracle Database PL/SQL Packages and Types Reference* for a description of the `DBMS_DATAPUMP` and the `DBMS_METADATA` packages. See *Oracle Database Security Guide* for information about exporting and importing the unified audit trail using Oracle Data Pump.

Related Topics

- Understanding Dump_ Log_ and SQL File Default Locations
- DBMS_DATAPUMP in *Oracle Database PL/SQL Packages and Types Reference*
- Exporting and Importing the Unified Audit Trail Using Oracle Data Pump in *Oracle Database Security Guide*

1.2 How Does Oracle Data Pump Move Data?

There are several Oracle Data Pump methods that you can use to move data in and out of databases. You can select the method that best fits your use case.

> **Note:**
>
> The `UTL_FILE_DIR` desupport in Oracle Database 18c and later releases affects Oracle Data Pump. This desupport can affect any feature from an earlier release using symbolic links, including (but not restricted to) Oracle Data Pump, BFILEs, and External Tables. If you attempt to use an affected feature configured with symbolic links, then you encounter `ORA-29283: invalid file operation: path traverses a symlink`. Oracle recommends that you instead use directory objects in place of symbolic links.
>
> Data Pump does not load tables with disabled unique indexes. To load data into the table, the indexes must be either dropped or reenabled.

- Using Data File Copying to Move Data
  The fastest method of moving data is to copy the database data files to the target database without interpreting or altering the data.
- Using Direct Path to Move Data
  After data file copying, direct path is the fastest method of moving data. In this method, the SQL layer of the database is bypassed and rows are moved to and from the dump file with only minimal interpretation.
- Using External Tables to Move Data
  If you do not select data file copying, and the data cannot be moved using direct path, you can use the external tables mechanism.
- Using Conventional Path to Move Data
  Where there are conflicting table attributes, Oracle Data Pump uses conventional path to move data.
- Using Network Link Import to Move Data
  When the Import `NETWORK_LINK` parameter is used to specify a network link for an import operation, the direct path method is used by default. Review supported database link types.
- Using a Parameter File (Parfile) with Oracle Data Pump
  To help to simplify Oracle Data Pump exports and imports, you can create a **parameter** file, also known as a **parfile**.

1.2.1 Using Data File Copying to Move Data

The fastest method of moving data is to copy the database data files to the target database without interpreting or altering the data.

When you copy database data files to the target database with this method, Data Pump Export is used to unload only structural information (metadata) into the dump file.

- The `TRANSPORT_TABLESPACES` parameter is used to specify a transportable tablespace export. Only metadata for the specified tablespaces is exported.

- The `TRANSPORTABLE=ALWAYS` parameter is supplied on a table mode export (specified with the `TABLES` parameter) or a full mode export (specified with the `FULL` parameter) or a full mode network import (specified with the `FULL` and `NETWORK_LINK` parameters).

When an export operation uses data file copying, the corresponding import job always also uses data file copying. During the ensuing import operation, both the data files and the export dump file must be loaded.

> **Note:**
>
> Starting with Oracle Database 21c, transportable jobs are restartable at or near the point of failure During transportable imports tablespaces are temporarily made read/write and then set back to read-only. The temporary setting change was introduced with Oracle Database 12c Release 1 (12.1.0.2) to improve performance. However, be aware that this behavior also causes the SCNs of the import job data files to change. Changing the SCNs for data files can cause issues during future transportable imports of those files.
>
> For example, if a transportable tablespace import fails at any point after the tablespaces have been made read/write (even if they are now read-only again), then the data files at that section of the export become corrupt. *They cannot be recovered.*
>
> When transportable jobs are performed, it is best practice to keep a copy of the data files on the source system until the import job has successfully completed on the target system. If the import job fails for some reason, then keeping copies ensures that you can have uncorrupted copies of the data files.

When data is moved by using data file copying, there are some limitations regarding character set compatibility between the source and target databases.

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` PL/SQL package or the `RMAN CONVERT` command to convert the data.

> **See Also:**
>
> - *Oracle Database Backup and Recovery Reference* for information about the RMAN `CONVERT` command
> - *Oracle Database Administrator's Guide* for a description and example (including how to convert the data) of transporting tablespaces between databases

1.2.2 Using Direct Path to Move Data

After data file copying, direct path is the fastest method of moving data. In this method, the SQL layer of the database is bypassed and rows are moved to and from the dump file with only minimal interpretation.

Data Pump automatically uses the direct path method for loading and unloading data unless the structure of a table does not allow it. For example, if a table contains a column of type `BFILE`, then direct path cannot be used to load that table and external tables is used instead.

The following sections describe situations in which direct path cannot be used for loading and unloading.

Situations in Which Direct Path Load Is Not Used

If any of the following conditions exist for a table, then Data Pump uses external tables to load the data for that table, instead of direct path:

- A domain index that is not a `CONTEXT` type index exists for a LOB column.
- A global index on multipartition tables exists during a single-partition load. This case includes object tables that are partitioned.
- A table is in a cluster.
- There is an active trigger on a preexisting table.
- Fine-grained access control is enabled in insert mode on a preexisting table.
- A table contains `BFILE` columns or columns of opaque types.
- A referential integrity constraint is present on a preexisting table.
- A table contains `VARRAY` columns with an embedded opaque type.
- The table has encrypted columns.
- The table into which data is being imported is a preexisting table and at least one of the following conditions exists:
  - There is an active trigger
  - The table is partitioned
  - Fine-grained access control is in insert mode
  - A referential integrity constraint exists
  - A unique index exists
- Supplemental logging is enabled, and the table has at least one LOB column.
- The Data Pump command for the specified table used the `QUERY`, `SAMPLE`, or `REMAP_DATA` parameter.
- A table contains a column (including a `VARRAY` column) with a `TIMESTAMP WITH TIME ZONE` data type, and the version of the time zone data file is different between the export and import systems.

Situations in Which Direct Path Unload Is Not Used

If any of the following conditions exist for a table, then Data Pump uses external tables rather than direct path to unload the data:

- Fine-grained access control for `SELECT` is enabled.
- The table is a queue table.
- The table contains one or more columns of type `BFILE` or opaque, or an object type containing opaque columns.
- The table contains encrypted columns.
- The table contains a column of an evolved type that needs upgrading.
- The Data Pump command for the specified table used the `QUERY`, `SAMPLE`, or `REMAP_DATA` parameter.

- Before the unload operation, the table was altered to contain a column that is NOT NULL, and also has a default value specified.

1.2.3 Using External Tables to Move Data

If you do not select data file copying, and the data cannot be moved using direct path, you can use the external tables mechanism.

The external tables mechanism creates an external table that maps to the dump file data for the database table. The SQL engine is then used to move the data. If possible, use the `APPEND` hint on import to speed the copying of the data into the database. The representation of data for direct path data and external table data is the same in a dump file. Because they are the same, Oracle Data Pump can use the direct path mechanism at export time, but use external tables when the data is imported into the target database. Similarly, Oracle Data Pump can use external tables for the export, but use direct path for the import.

In particular, Oracle Data Pump can use external tables in the following situations:

- Loading and unloading very large tables and partitions in situations where it is advantageous to use parallel SQL capabilities
- Loading tables with global or domain indexes defined on them, including partitioned object tables
- Loading tables with active triggers or clustered tables
- Loading and unloading tables with encrypted columns
- Loading tables with fine-grained access control enabled for inserts
- Loading a table not created by the import operation (the table exists before the import starts)

> **Note:**
>
> When Oracle Data Pump uses external tables as the data access mechanism, it uses the `ORACLE_DATAPUMP` access driver. However, be aware that the files that Oracle Data Pump creates when it uses external tables are not compatible with files created when you manually create an external table using the SQL `CREATE TABLE ... ORGANIZATION EXTERNAL` statement.

Related Topics

- The ORACLE_DATAPUMP Access Driver
- APPEND Hint
- Loading LOBs with External Tables

1.2.4 Using Conventional Path to Move Data

Where there are conflicting table attributes, Oracle Data Pump uses conventional path to move data.

In situations where there are conflicting table attributes, Oracle Data Pump is not able to load data into a table using either direct path or external tables. In such cases, conventional path is used, which can affect performance.

1.2.5 Using Network Link Import to Move Data

When the Import `NETWORK_LINK` parameter is used to specify a network link for an import operation, the direct path method is used by default. Review supported database link types.

If direct path cannot be used (for example, because one of the columns is a `BFILE`), then SQL is used to move the data using an `INSERT SELECT` statement. (Before Oracle Database 12c Release 2 (12.2.0.1), the default was to use the `INSERT SELECT` statement.) The `SELECT` clause retrieves the data from the remote database over the network link. The `INSERT` clause uses SQL to insert the data into the target database. There are no dump files involved.

When the Export `NETWORK_LINK` parameter is used to specify a network link for an export operation, the data from the remote database is written to dump files on the target database. (Note that to export from a read-only database, the `NETWORK_LINK` parameter is required.)

Because the link can identify a remotely networked database, the terms database link and network link are used interchangeably.

Supported Link Types

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

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

Unsupported Link Types

The following types of database links are not supported for use with Data Pump Export and Import:

- Private connected user
- Current user
- Parallel export or import of metadata for network jobs.

  For conventional jobs, if you need parallel metadata import, then use a dumpfile instead of `NETWORK_LINK`.

See Also:

- The Export NETWORK_LINK parameter for information about performing exports over a database link
- The Import NETWORK_LINK parameter for information about performing imports over a database link
- *Oracle Database Administrator's Guide* for information about creating database links and the different types of links

1.2.6 Using a Parameter File (Parfile) with Oracle Data Pump

To help to simplify Oracle Data Pump exports and imports, you can create a **parameter** file, also known as a **parfile**.

Instead of typing in Oracle Data Pump parameters at the command line, when you run an export or import operation, you can prepare a parameter text file (also known as a parfile, after the parameter name) that provides the command-line parameters to the Oracle Data Pump client. You specify that Oracle Data Pump obtains parameters for the command by entering the `PARFILE` parameter, and then specifying the parameter name:

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

When the Oracle Data Pump Export or Import operation starts, the parameter file is opened and read by the client. The default location of the parameter file is the user's current directory.

For example:

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

When you create a parameter file, it makes it easier for you to reuse that file for multiple export or import operations, which can simplify these operations, particularly if you perform them regularly. Creating a parameter file also helps you to avoid typographical errors that can occur from typing long Oracle Data Pump commands on the command line, especially if you use parameters whose values require quotation marks that must be placed precisely. On some systems, if you use a parameter file and the parameter value being specified does not have quotation marks as the first character in the string (for example, `TABLES=scott."Emp"`), then the use of escape characters may not be necessary.

There is no required file name extension, but Oracle examples use `.par` as the extension. Oracle recommends that you also use this file extension convention. Using a consistent parameter file extension makes it easier to identify and use these files.

> **Note:**
>
> The `PARFILE` parameter cannot be specified within a parameter file.

For more information and examples, see the `PARFILE` parameters for Oracle Data Pump Import and Export.

Related Topics

- Oracle Data Pump Export PARFILE
- Oracle Data Pump Import PARFILE

1.3 Using Oracle Data Pump With CDBs

Oracle Data Pump can migrate all, or portions of, a database from a non-CDB into a PDB, between PDBs within the same or different CDBs, and from a PDB into a non-CDB.

- About Using Oracle Data Pump in a Multitenant Environment
  In general, using Oracle Data Pump with PDBs is identical to using Oracle Data Pump with a non-CDB.

- Using Oracle Data Pump to Move Data Into a CDB
  After you create an empty PDB, to move data into the PDB, you can use an Oracle Data Pump full-mode export and import operation.
- Using Oracle Data Pump to Move PDBs Within or Between CDBs
  Learn how to avoid `ORA-65094` user schema errors with Oracle Data Pump export and import operations on PDBs.

1.3.1 About Using Oracle Data Pump in a Multitenant Environment

In general, using Oracle Data Pump with PDBs is identical to using Oracle Data Pump with a non-CDB.

A multitenant container database (CDB) is an Oracle Database that includes zero, one, or many user-created pluggable databases (PDBs). A PDB is a portable set of schemas, schema objects, and non-schema objects that appear to an Oracle Net client as a non-CDB. A non-CDB is an Oracle Database that is not a CDB. Non-CDB architecture Oracle Database was deprecated in Oracle Database 12c Release 1 (12.1). Starting with Oracle Database 21c, non-CDB architecture deployments are desupported.

You can use Oracle Data Pump to migrate all or some of a database in the following scenarios:

- From a non-CDB into a PDB
- Between PDBs within the same or different CDBs
- From a PDB into an earlier release non-CDB

> **Note:**
>
> Oracle Data Pump does not support any operations across the entire CDB. If you are connected to the root or seed database of a CDB, then Oracle Data Pump issues the following warning:
>
> ```
> ORA-39357: Warning: Oracle Data Pump operations are not typically
> needed when connected to the root or seed of a container database.
> ```

1.3.2 Using Oracle Data Pump to Move Data Into a CDB

After you create an empty PDB, to move data into the PDB, you can use an Oracle Data Pump full-mode export and import operation.

You can import data with or without the transportable option. If you use the transportable option on a full mode export or import, then it is referred to as a full transportable export/import.

When the transportable option is used, export and import use both transportable tablespace data movement and conventional data movement; the latter for those tables that reside in non-transportable tablespaces such as `SYSTEM` and `SYSAUX`. Using the transportable option can reduce the export time, and especially, the import time. With the transportable option, table data does not need to be unloaded and reloaded, and index structures in user tablespaces do not need to be recreated.

Note the following requirements when using Oracle Data Pump to move data into a CDB:

- To administer a multitenant environment, you must have the `CDB_DBA` role.

- Full database exports from Oracle Database 11.2.0.2 and earlier can be imported into Oracle Database 12c or later (CDB or non-CDB). However, Oracle recommends that you first upgrade the source database to Oracle Database 11g Release 2 (11.2.0.3 or later), so that information about registered options and components is included in the export.
- When migrating Oracle Database 11g Release 2 (11.2.0.3 or later) to a CDB (or to a non-CDB) using either full database export or full transportable database export, you must set the Oracle Data Pump Export parameter at least to `VERSION=12` to generate a dump file that is ready for import into an Oracle Database 12c or later release. If you do not set `VERSION=12`, then the export file that is generated does not contain complete information about registered database options and components.
- Network-based full transportable imports require use of the `FULL=YES`, `TRANSPORTABLE=ALWAYS`, and `TRANSPORT_DATAFILES=`*`datafile_name`* parameters. When the source database is Oracle Database 11g Release 11.2.0.3 or later, but earlier than Oracle Database 12c Release 1 (12.1), the `VERSION=12` parameter is also required.
- File-based full transportable imports only require use of the `TRANSPORT_DATAFILES=`*`datafile_name`* parameter. Data Pump Import infers the presence of the `TRANSPORTABLE=ALWAYS` and `FULL=YES` parameters.
- As of Oracle Database 12c Release 2 (12.2), in a multitenant container database (CDB) environment, the default Oracle Data Pump directory object, `DATA_PUMP_DIR`, is defined as a unique path for each PDB in the CDB. This unique path is defined whether the `PATH_PREFIX` clause of the `CREATE PLUGGABLE DATABASE` statement is defined or is not defined for relative paths.
- Starting in Oracle Database 19c, the `credential` parameter of `impdp` specifies the name of the credential object that contains the user name and password required to access an object store bucket. You can also specify a default credential using the PDB property named `DEFAULT_CREDENTIAL`. When you run `impdb` with then default credential, you prefix the dump file name with `DEFAULT_CREDENTIAL:` and you do not specify the `credential` parameter.

Example 1-1 Importing a Table into a PDB

To specify a particular PDB for the export/import operation, supply a connect identifier in the connect string when you start Data Pump. For example, to import data to a PDB named `pdb1`, you could enter the following on the Data Pump command line:

```
impdp hr@pdb1 DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp TABLES=employees
```

Example 1-2 Specifying a Credential When Importing Data

This example assumes that you created a credential named `HR_CRED` using `DBMS_CREDENTIAL.CREATE_CREDENTIAL` as follows:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'HR_CRED',
    username => 'atpc_user@example.com',
    password => 'password'
  );
END;
/
```

The following command specifies credential `HR_CRED`, and specifies the file stored in an object store. The URL of the file is `https://example.com/ostore/dnfs/myt.dmp`.

```
impdp hr@pdb1 \
      table_exists_action=replace \
      credential=HR_CRED \
      parallel=16 \
      dumpfile=https://example.com/ostore/dnfs/myt.dmp
```

Example 1-3 Importing Data Using a Default Credential

1. You create a credential named `HR_CRED` using `DBMS_CREDENTIAL.CREATE_CREDENTIAL` as follows:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'HR_CRED',
       username => 'atpc_user@example.com',
       password => 'password'
     );
   END;
   /
   ```

2. You set the PDB property `DEFAULT_CREDENTIAL` as follows:

   ```
   ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.HR_CRED'
   ```

3. The following command specifies the default credential as a prefix to the dump file location `https://example.com/ostore/dnfs/myt.dmp`:

   ```
   impdp hr@pdb1 \
         table_exists_action=replace \
         parallel=16 \
         dumpfile=default_credential:https://example.com/ostore/dnfs/myt.dmp
   ```

   Note that the `credential` parameter is not specified.

See Also:

- *Oracle Database Security Guide* to learn how to configure SSL authentication, which is necessary for object store access
- Importing a Table to an Object Store Using Oracle Data Pump to learn about using Oracle Data Pump Import to load files to the object store

1.3.3 Using Oracle Data Pump to Move PDBs Within or Between CDBs

Learn how to avoid `ORA-65094` user schema errors with Oracle Data Pump export and import operations on PDBs.

If you create a common user in a CDB, then a full database or privileged schema export of that user from within any PDB in the CDB results in a standard `CREATE USER C##common name` DDL statement being performed upon import. However, the statement fails because of the common user prefix `C##` on the user name. The following error message is returned:

```
ORA-65094:invalid local user or role name
```

Example 1-4 Avoiding Invalid Local User Error

In the PDB being exported, if you have created local objects in that user's schema, and you want to import them, then either make sure a common user of the same name already exists in the target CDB instance, or use the Oracle Data Pump Import `REMAP_SCHEMA` parameter on the `impdp` command to remap the schema to a valid local user. For example:

```
REMAP_SCHEMA=C##common name:local user name
```

Related Topics

- Full Export Mode
  You can use Oracle Data Pump to carry out a full database export by using the `FULL` parameter.
- Full Import Mode
  To specify a full import with Oracle Data Pump, use the `FULL` parameter.

1.4 Cloud Premigration Advisor Tool

The Cloud Premigration Advisor tool can assist you to migrate a database to the Oracle Cloud.

- What is the Cloud Premigration Advisor Tool (CPAT)
  To determine if your On Premises Oracle Database data is suitable to migrate to an Oracle Cloud, you can use Oracle's Cloud Premigration Advisor Tool (CPAT).

1.4.1 What is the Cloud Premigration Advisor Tool (CPAT)

To determine if your On Premises Oracle Database data is suitable to migrate to an Oracle Cloud, you can use Oracle's Cloud Premigration Advisor Tool (CPAT).

The Cloud Premigration Advisor Tool (CPAT) is a Java application that assists you to analyze your On Premises Oracle Databases to determine whether you can migrate some or all of that database to one of the Oracle Cloud platform options, such as Autonomous Database, or other Cloud database options. The CPAT assists you to evaluate your specific migration scenario, to identify migration options, and assist you to prepare your migration plans from source On Premises Oracle Databases to the target Oracle Cloud database option to which you want to migrate.

How the CPAT Helps You to Avoid Issues

When you use the CPAT tool, and it discovers that there are potential environment issues with a Cloud migration, you are warned ahead of time of what these issues are. As a result, you are less likely to encounter an unforeseen issue with your migration. In addition to warning you about issues, the tool can also provide you with parameters for migration, including parameters for Oracle Data Pump, or other migration tools. These parameters are customized for your specific migration case, so that potential migration issues are either reduced, or avoided entirely.

To identify issues and create customized parameters, CPAT performs several checks on the source database and schema contents. These checks are guided by the target Oracle Cloud database option that you select, and the migration approach that you intend to use. The results of these checks are compiled and presented back to you, either in a machine-readable format (JSON), or a human readable format (plain text or HTML), or both. In addition, the CPAT check results are designed so that they can be used by other Oracle features, such as Oracle Zero Downtime Migration (Oracle ZDM) or Oracle Enterprise Manager.

> **Note:**
>
> CPAT is not itself a migration tool. It is intended to assist you to prepare for migrations. It does not suggest whether a particular migration approach using Oracle GoldenGate or Oracle Data Pump is the best option, but rather provides you with customized support for the option that you choose.

1.5 Required Roles for Oracle Data Pump Export and Import Operations

The roles `DATAPUMP_EXP_FULL_DATABASE` and `DATAPUMP_IMP_FULL_DATABASE` are required for many Export and Import operations.

> **Caution:**
>
> Do not run Oracle Data Pump jobs as the `SYS` user. Either use the schema `SYSTEM` (or `ADMIN` in Oracle Autonomous Database) for system management operations, or use a user account that is granted the Data Pump full privileges roles that are described below.

When you run Export or Import operations, the operation can require that the user account that you are using to run the operations on premises or in user-managed cloud services is granted either the `DATAPUMP_EXP_FULL_DATABASE` role, or the `DATAPUMP_IMP_FULL_DATABASE` role, or both roles. The corresponding roles for Autonomous Database are `DATAPUMP_CLOUD_EXP` and `DATAPUMP_CLOUD_IMP`. These roles are automatically defined for Oracle Database when you run the standard scripts that are part of database creation. (Note that although the names of these roles contain the word `FULL`, these roles actually apply to any privileged operations in any export or import mode, not only `Full` mode.)

The `DATAPUMP_EXP_FULL_DATABASE` role affects only export operations. The `DATAPUMP_IMP_FULL_DATABASE` role affects import operations and operations that use the

Import `SQLFILE` parameter. These roles allow users performing exports and imports to do the following:

- Perform the operation outside the scope of their schema
- Monitor jobs that were initiated by another user
- Export objects (such as tablespace definitions) and import objects (such as directory definitions) that unprivileged users cannot reference

These are powerful roles. As a database administrator, you should use caution when granting these roles to users.

Although the `SYS` schema does not have either of these roles assigned to it, all security checks performed by Oracle Data Pump that require these roles also grant access to the `SYS` schema.

> **Note:**
>
> If you receive an `ORA-39181: Only Partial Data Exported Due to Fine Grain Access Control` error message, then see My Oracle Support "ORA-39181:Only Partial Table Data Exported Due To Fine Grain Access Control (Doc ID 422480.1)" for information about security during an export of table data with fine-grained access control policies enabled.:
>
> ORA-39181:Only Partial Table Data Exported Due To Fine Grain Access Control (Doc ID 422480.1)

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

> **See Also:**
>
> *Oracle Database Security Guide* for more information about predefined roles in an Oracle Database installation

1.6 What Happens During the Processing of an Oracle Data Pump Job?

Oracle Data Pump jobs use a Data Pump control job table, a Data Pump control job process, and worker processes to perform the work and keep track of progress.

- Coordination of an Oracle Data Pump Job
  A Data Pump control process is created to coordinate every Oracle Data Pump Export and Import job.
- Tracking Progress Within an Oracle Data Pump Job
  While Oracle Data Pump transfers data and metadata, a Data Pump control job table is used to track the progress within a job.

- Filtering Data and Metadata During an Oracle Data Pump Job
  If you want to filter the types of objects that are exported and imported with Oracle Data Pump, then you can use the `EXCLUDE` and `INCLUDE` parameters.
- Transforming Metadata During an Oracle Data Pump Job
  When you move data from one database to another, you can perform transformations on the metadata by using Oracle Data Pump Import parameters.
- Maximizing Job Performance of Oracle Data Pump
  To increase job performance, you can use the Oracle Data Pump `PARALLEL` parameter to run multiple worker processes in parallel.
- Loading and Unloading Data with Oracle Data Pump
  Learn how Oracle Data Pump child processes operate during data imports and exports.

1.6.1 Coordination of an Oracle Data Pump Job

A Data Pump control process is created to coordinate every Oracle Data Pump Export and Import job.

The Data Pump control process controls the entire job, including communicating with the client processes, creating and controlling a pool of worker processes, and performing logging operations.

1.6.2 Tracking Progress Within an Oracle Data Pump Job

While Oracle Data Pump transfers data and metadata, a Data Pump control job table is used to track the progress within a job.

The Data Pump control table is implemented as a user table within the database. The specific function of the Data Pump control table for export and import jobs is as follows:

- For export jobs, the Data Pump control job table records the location of database objects within a dump file set. Export builds and maintains the Data Pump control table for the duration of the job. At the end of an export job, the content of the Data Pump control table is written to a file in the dump file set.
- For import jobs, the Data Pump control job table is loaded from the dump file set, and is used to control the sequence of operations for locating objects that need to be imported into the target database.

The Data Pump control job table is created in the schema of the current user performing the export or import operation. Therefore, that user must have the `CREATE TABLE` system privilege and a sufficient tablespace quota for creation of the Data Pump control job table. The name of the Data Pump control job table is the same as the name of the job that created it. Therefore, you cannot explicitly give an Oracle Data Pump job the same name as a preexisting table or view.

For all operations, the information in the master table is used to restart a job.

The Data Pump control job table is either retained or dropped, depending on the circumstances, as follows:

- Upon successful job completion, the Data Pump control job table is dropped. You can override this by setting the Oracle Data Pump `KEEP_MASTER=YES` parameter for the job.
- The Data Pump control job table is automatically retained for jobs that do not complete successfully.

- If a job is stopped using the `STOP_JOB` interactive command, then the Data Pump control job table is retained for use in restarting the job.
- If a job is killed using the `KILL_JOB` interactive command, then the Data Pump control job table is dropped, and the job cannot be restarted.
- If a job terminates unexpectedly, then the Data Pump control job table is retained. You can delete it if you do not intend to restart the job.
- If a job stops before it starts running (that is, before any database objects have been copied), then the Data Pump control job table is dropped.

Related Topics

- Oracle Data Pump Export command-line utility JOB_NAME parameter

1.6.3 Filtering Data and Metadata During an Oracle Data Pump Job

If you want to filter the types of objects that are exported and imported with Oracle Data Pump, then you can use the `EXCLUDE` and `INCLUDE` parameters.

Within the Data Pump control job table, specific objects are assigned attributes such as name or owning schema. Objects also belong to a class of objects (such as `TABLE`, `INDEX`, or `DIRECTORY`). The class of an object is called its object type. You can use the `EXCLUDE` and `INCLUDE` parameters to restrict the types of objects that are exported and imported. The objects can be based upon the name of the object, or the name of the schema that owns the object. You can also specify data-specific filters to restrict the rows that are exported and imported.

Related Topics

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

1.6.4 Transforming Metadata During an Oracle Data Pump Job

When you move data from one database to another, you can perform transformations on the metadata by using Oracle Data Pump Import parameters.

It is often useful to perform transformations on your metadata, so that you can remap storage between tablespaces, or redefine the owner of a particular set of objects. When you move data, you can perform transformations by using the Oracle Data Pump import parameters `REMAP_DATAFILE`, `REMAP_SCHEMA`, `REMAP_TABLE`, `REMAP_TABLESPACE`, `TRANSFORM`, and `PARTITION_OPTIONS`.

1.6.5 Maximizing Job Performance of Oracle Data Pump

To increase job performance, you can use the Oracle Data Pump `PARALLEL` parameter to run multiple worker processes in parallel.

The `PARALLEL` parameter enables you to set a degree of parallelism that takes maximum advantage of current conditions. For example, to limit the effect of a job on a production system, database administrators can choose to restrict the parallelism. The degree of parallelism can be reset at any time during a job. For example, during production hours, you can set `PARALLEL` to 2, so that you restrict a particular job to only two degrees of parallelism.

During non-production hours, you can reset the degree of parallelism to 8. The parallelism setting is enforced by the Data Pump control process, which allocates workloads to worker processes that perform the data and metadata processing within an operation. These worker processes operate in parallel. For recommendations on setting the degree of parallelism, refer to the Export `PARALLEL` and Import `PARALLEL` parameter descriptions.

> **Note:**
>
> The ability to adjust the degree of parallelism is available only in the Enterprise Edition of Oracle Database.

Related Topics

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

1.6.6 Loading and Unloading Data with Oracle Data Pump

Learn how Oracle Data Pump child processes operate during data imports and exports.

Oracle Data Pump child processes unload and load metadata and table data. For export, all metadata and data are unloaded in parallel, with the exception of jobs that use transportable tablespace. For import, objects must be created in the correct dependency order.

If there are enough objects of the same type to make use of multiple child processes, then the objects are imported by multiple child processes. Some metadata objects have interdependencies, which require one child process to create them serially to satisfy those dependencies. Child processes are created as needed until the number of child processes equals the value supplied for the `PARALLEL` command-line parameter. The number of active child processes can be reset throughout the life of a job. Worker processes can be started on different nodes in an Oracle Real Application Clusters (Oracle RAC) environment.

> **Note:**
>
> The value of `PARALLEL` is restricted to 1 in the Standard Edition of Oracle Database.

When a child process is assigned the task of loading or unloading a very large table or partition, to make maximum use of parallel execution, it can make use of the external tables access method. In such a case, the child process becomes a parallel execution coordinator. The actual loading and unloading work is divided among some number of parallel input/output (I/O) execution processes allocated from a pool of available processes in an Oracle Real Application Clusters (Oracle RAC) environment.

Related Topics

- PARALLEL
- PARALLEL

1.7 How to Monitor Status of Oracle Data Pump Jobs

The Oracle Data Pump Export and Import client utilities can attach to a job in either logging mode or interactive-command mode.

In logging mode, real-time detailed status about the job is automatically displayed during job execution. The information displayed can include the job and parameter descriptions, an estimate of the amount of data to be processed, a description of the current operation or item being processed, files used during the job, any errors encountered, and the final job state (Stopped or Completed).

In interactive-command mode, job status can be displayed on request. The information displayed can include the job description and state, a description of the current operation or item being processed, files being written, and a cumulative status.

You can also have a log file written during the execution of a job. The log file summarizes the progress of the job, lists any errors encountered during execution of the job, and records the completion status of the job.

As an alternative to determine job status or other information about Oracle Data Pump jobs, you can query the `DBA_DATAPUMP_JOBS`, `USER_DATAPUMP_JOBS`, or `DBA_DATAPUMP_SESSIONS` views. Refer to *Oracle Database Reference* for more information.

Related Topics

- *Oracle Database Reference*

1.8 How to Monitor the Progress of Running Jobs with V$SESSION_LONGOPS

To monitor table data transfers, you can use the `V$SESSION_LONGOPS` dynamic performance view to monitor Oracle Data Pump jobs.

Oracle Data Pump operations that transfer table data (export and import) maintain an entry in the `V$SESSION_LONGOPS` dynamic performance view indicating the job progress (in megabytes of table data transferred). The entry contains the estimated transfer size and is periodically updated to reflect the actual amount of data transferred.

Use of the `COMPRESSION`, `ENCRYPTION`, `ENCRYPTION_ALGORITHM`, `ENCRYPTION_MODE`, `ENCRYPTION_PASSWORD`, `QUERY`, and `REMAP_DATA` parameters are not reflected in the determination of estimate values.

The usefulness of the estimate value for export operations depends on the type of estimation requested when the operation was initiated, and it is updated as required if exceeded by the actual transfer amount. The estimate value for import operations is exact.

The `V$SESSION_LONGOPS` columns that are relevant to a Data Pump job are as follows:

- `USERNAME`: Job owner
- `OPNAME`: Job name
- `TARGET_DESC`: Job operation
- `SOFAR`: Megabytes transferred thus far during the job
- `TOTALWORK` Estimated number of megabytes in the job

- `UNITS`: Megabytes (MB)
- `MESSAGE`: A formatted status message that uses the following format:

```
'job_name: operation_name : nnn out of mmm MB done'
```

1.9 File Allocation with Oracle Data Pump

You can modify how Oracle Data Pump allocates and handles files by using commands in interactive mode.

- Understanding File Allocation in Oracle Data Pump
  Understanding how Oracle Data Pump allocates and handles files helps you to use Export and Import to their fullest advantage.
- Specifying Files and Adding Additional Dump Files
  For export operations, you can either specify dump files at the time you define the Oracle Data Pump job, or at a later time during the operation.
- Default Locations for Dump, Log, and SQL Files
  Learn about default Oracle Data Pump file locations, and how these locations are affected when you are using Oracle RAC, Oracle Automatic Storage Management, and multitenant architecture.
- Using Substitution Variables with Oracle Data Pump Exports
  If you want to specify multiple dump files during Oracle Data Pump export operations, then use the `DUMPFILE` parameter with a substitution variable in the file name.

1.9.1 Understanding File Allocation in Oracle Data Pump

Understanding how Oracle Data Pump allocates and handles files helps you to use Export and Import to their fullest advantage.

Oracle Data Pump jobs manage the following types of files:

- Dump files, to contain the data and metadata that is being moved.
- Log files, to record the messages associated with an operation.
- SQL files, to record the output of a `SQLFILE` operation. A `SQLFILE` operation is started using the Oracle Data Pump Import `SQLFILE` parameter. This operation results in all of the `SQL DDL` that Import would execute, based on other parameters, being written to a SQL file.
- Files specified by the `DATA_FILES` parameter during a transportable import.

> **Note:**
>
> If your Oracle Data Pump job generates errors related to Network File Storage (NFS), then consult the installation guide for your platform to determine the correct NFS mount settings.

1.9.2 Specifying Files and Adding Additional Dump Files

For export operations, you can either specify dump files at the time you define the Oracle Data Pump job, or at a later time during the operation.

If you discover that space is running low during an export operation, then you can add additional dump files by using the Oracle Data Pump Export `ADD_FILE` command in interactive mode.

For import operations, all dump files must be specified at the time the job is defined.

For dump files, you can use the Export `REUSE_DUMPFILES` parameter to specify whether to overwrite a preexisting dump file.

> **Note:**
>
> 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'`.

1.9.3 Default Locations for Dump, Log, and SQL Files

Learn about default Oracle Data Pump file locations, and how these locations are affected when you are using Oracle RAC, Oracle Automatic Storage Management, and multitenant architecture.

- Understanding Dump, Log, and SQL File Default Locations
  Oracle Data Pump is server-based, rather than client-based. Dump files, log files, and SQL files are accessed relative to server-based directory paths.
- Understanding How to Use Oracle Data Pump with Oracle RAC
  Using Oracle Data Pump in an Oracle Real Application Clusters (Oracle RAC) environment requires you to perform a few checks to ensure that you are making cluster member nodes available.
- Using Directory Objects When Oracle Automatic Storage Management Is Enabled
  If you use Oracle Data Pump Export or Import with Oracle Automatic Storage Management (Oracle ASM) enabled, then define the directory object used for the dump file.
- The DATA_PUMP_DIR Directory Object and Pluggable Databases
  The default Oracle Data Pump directory object, `DATA_PUMP_DIR`, is defined as a unique path for each PDB in the CDB.

1.9.3.1 Understanding Dump, Log, and SQL File Default Locations

Oracle Data Pump is server-based, rather than client-based. Dump files, log files, and SQL files are accessed relative to server-based directory paths.

Oracle Data Pump requires that directory paths are specified as directory objects. A directory object maps a name to a directory path on the file system. As a database administrator, you must ensure that only approved users are allowed access to the directory object associated with the directory path.

The following example shows a SQL statement that creates a directory object named `dpump_dir1` that is mapped to a directory located at `/usr/apps/datafiles`.

```
SQL> CREATE DIRECTORY dpump_dir1 AS '/usr/apps/datafiles';
```

The reason that a directory object is required is to ensure data security and integrity. For example:

- If you are allowed to specify a directory path location for an input file, then it is possible that you could be able to read data that the server has access to, but to which you should not.
- If you are allowed to specify a directory path location for an output file, then it is possible that you could overwrite a file that normally you do not have privileges to delete.

On Unix, Linux, and Windows operating systems, a default directory object, `DATA_PUMP_DIR`, is created at database creation, or whenever the database dictionary is upgraded. By default, this directory object is available only to privileged users. (The user `SYSTEM` has read and write access to the `DATA_PUMP_DIR` directory, by default.) Oracle can change the definition of the `DATA_PUMP_DIR` directory, either during Oracle Database upgrades, or when patches are applied.

If you are not a privileged user, then before you can run Oracle Data Pump Export or Import, a directory object must be created by a database administrator (DBA), or by any user with the `CREATE ANY DIRECTORY` privilege.

After a directory is created, the user creating the directory object must grant `READ` or `WRITE` permission on the directory to other users. For example, to allow Oracle Database to read and write files on behalf of user `hr` in the directory named by `dpump_dir1`, the DBA must run the following command:

```
SQL> GRANT READ, WRITE ON DIRECTORY dpump_dir1 TO hr;
```

Note that `READ` or `WRITE` permission to a directory object only means that Oracle Database can read or write files in the corresponding directory on your behalf. Outside of Oracle Database, uou are not given direct access to those files, unless you have the appropriate operating system privileges. Similarly, Oracle Database requires permission from the operating system to read and write files in the directories.

Oracle Data Pump Export and Import use the following order of precedence to determine a file's location:

1. If a directory object is specified as part of the file specification, then the location specified by that directory object is used. (The directory object must be separated from the file name by a colon.)
2. If a directory object is not specified as part of the file specification, then the directory object named by the `DIRECTORY` parameter is used.
3. If a directory object is not specified as part of the file specification, and if no directory object is named by the `DIRECTORY` parameter, then the value of the environment variable, `DATA_PUMP_DIR`, is used. This environment variable is defined by using operating system commands on the client system where the Data Pump Export and Import utilities are run. The value assigned to this client-based environment variable must be the name of a server-based directory object, which must first be created on the server system by a DBA. For example, the following SQL statement creates a directory object on the server system. The name of the directory object is `DUMP_FILES1`, and it is located at `'/usr/apps/dumpfiles1'`.

   ```
   SQL> CREATE DIRECTORY DUMP_FILES1 AS '/usr/apps/dumpfiles1';
   ```

   After this statement is run, a user on a Unix-based client system using `csh` can assign the value `DUMP_FILES1` to the environment variable `DATA_PUMP_DIR`. The `DIRECTORY` parameter

can then be omitted from the command line. The dump file `employees.dmp`, and the log file `export.log`, are written to `'/usr/apps/dumpfiles1'`.

```
%setenv DATA_PUMP_DIR DUMP_FILES1
%expdp hr TABLES=employees DUMPFILE=employees.dmp
```

4. If none of the previous three conditions yields a directory object, and you are a privileged user, then Oracle Data Pump attempts to use the value of the default server-based directory object, `DATA_PUMP_DIR`. This directory object is automatically created, either at database creation, or when the database dictionary is upgraded. To see the path definition for `DATA_PUMP_DIR`, you can use the following SQL query:

   ```
   SQL> SELECT directory_name, directory_path FROM dba_directories
   2 WHERE directory_name='DATA_PUMP_DIR';
   ```

   If you are not a privileged user, then access to the `DATA_PUMP_DIR` directory object must have previously been granted to you by a DBA.

   Do not confuse the default `DATA_PUMP_DIR` directory object with the client-based environment variable of the same name.

1.9.3.2 Understanding How to Use Oracle Data Pump with Oracle RAC

Using Oracle Data Pump in an Oracle Real Application Clusters (Oracle RAC) environment requires you to perform a few checks to ensure that you are making cluster member nodes available.

- To use Oracle Data Pump or external tables in an Oracle RAC configuration, you must ensure that the directory object path is on a cluster-wide file system.

  The directory object must point to shared physical storage that is visible to, and accessible from, all instances where Oracle Data Pump or external tables processes (or both) can run.

- The default Oracle Data Pump behavior is that child processes can run on any instance in an Oracle RAC configuration. Therefore, child processes on those Oracle RAC instances must have physical access to the location defined by the directory object, such as shared storage media. If the configuration does not have shared storage for this purpose, but you still require parallelism, then you can use the `CLUSTER=NO` parameter to constrain all child processes to the instance where the Oracle Data Pump job was started.

- Under certain circumstances, Oracle Data Pump uses parallel query child processes to load or unload data. In an Oracle RAC environment, Data Pump does not control where these child processes run. Therefore, these child processes can run on other cluster member nodes in the cluster, regardless of which instance is specified for `CLUSTER` and `SERVICE_NAME` for the Oracle Data Pump job. Controls for parallel query operations are independent of Oracle Data Pump. When parallel query child processes run on other instances as part of an Oracle Data Pump job, they also require access to the physical storage of the dump file set.

1.9.3.3 Using Directory Objects When Oracle Automatic Storage Management Is Enabled

If you use Oracle Data Pump Export or Import with Oracle Automatic Storage Management (Oracle ASM) enabled, then define the directory object used for the dump file.

You must define the directory object used for the dump file so that the Oracle ASM disk group name is used, instead of an operating system directory path.

For log file, use a separate directory object that points to an operating system directory path.

For example, you can create a directory object for the Oracle ASM dump file using this procedure.

```
SQL> CREATE or REPLACE DIRECTORY dpump_dir as '+DATAFILES/';
```

After you create the directory object, you then create a separate directory object for the log file:

```
SQL> CREATE or REPLACE DIRECTORY dpump_log as '/homedir/user1/';
```

To enable user `hr` to have access to these directory objects, you assign the necessary privileges for that user:

```
SQL> GRANT READ, WRITE ON DIRECTORY dpump_dir TO hr;
SQL> GRANT READ, WRITE ON DIRECTORY dpump_log TO hr;
```

Finally, you then can use use the following Data Pump Export command:

```
> expdp hr DIRECTORY=dpump_dir DUMPFILE=hr.dmp LOGFILE=dpump_log:hr.log
```

Before the command executes, you are prompted for the password.

> **Note:**
>
> If you simply want to copy Data Pump dump files between ASM and disk directories, you can use the `DBMS_FILE_TRANSFER` PL/SQL package.

Related Topics

- *Oracle Database SQL Language Reference*
- *Oracle Database PL/SQL Packages and Types Reference*

1.9.3.4 The DATA_PUMP_DIR Directory Object and Pluggable Databases

The default Oracle Data Pump directory object, `DATA_PUMP_DIR`, is defined as a unique path for each PDB in the CDB.

As of Oracle Database 12c Release 2 (12.2), in a multitenant container database (CDB) environment, the default Oracle Data Pump directory object, `DATA_PUMP_DIR`, is defined as a unique path for each PDB in the CDB, whether or not the `PATH_PREFIX` clause of the `CREATE PLUGGABLE DATABASE` statement is defined for relative paths.

1.9.4 Using Substitution Variables with Oracle Data Pump Exports

If you want to specify multiple dump files during Oracle Data Pump export operations, then use the `DUMPFILE` parameter with a substitution variable in the file name.

When you use substitution variables with file names, instead of or in addition to listing specific file names, then those filenames with a substitution variable are called **dump file templates**.

> **Note:**
>
> In the examples that follow, the substitution variable `%U` is used to explain how Oracle Data Pump uses substitution variables. You can view other available substitution variables under the Import or Export `DUMPFILE` parameter reference topics.

When you use dump file templates, new dump files are created as they are needed. For example, if you are using the substitution variable `%U`, then new dump files are created as needed beginning with `01` for `%U`, and then using `02`, `03`, and so on. Enough dump files are created to allow all processes specified by the current setting of the `PARALLEL` parameter to be active. If one of the dump files becomes full because its size has reached the maximum size specified by the `FILESIZE` parameter, then it is closed, and a new dump file (with a new generated name) is created to take its place.

If multiple dump file templates are provided, then they are used to generate dump files in a round-robin fashion. For example, if `expa%U`, `expb%U`, and `expc%U` are all specified for a job having a parallelism of 6, then the initial dump files created are `expa01.dmp`, `expb01.dmp`, `expc01.dmp`, `expa02.dmp`, `expb02.dmp`, and `expc02.dmp`.

For import and `SQLFILE` operations, if dump file specifications `expa%U`, `expb%U`, and `expc%U` are specified, then the operation begins by attempting to open the dump files `expa01.dmp`, `expb01.dmp`, and `expc01.dmp`. It is possible for the Data Pump control export table to span multiple dump files. For this reason, until all pieces of the Data Pump control table are found, dump files continue to be opened by incrementing the substitution variable, and looking up the new file names (For example: `expa02.dmp`, `expb02.dmp`, and `expc02.dmp`). If a dump file does not exist, then the operation stops incrementing the substitution variable for the dump file specification that was in error. For example, if `expb01.dmp` and `expb02.dmp` are found, but `expb03.dmp` is not found, then no more files are searched for using the `expb%U` specification. After the entire Data Pump control table is found, it is used to determine whether all dump files in the dump file set have been located.

Related Topics

- Oracle Data Pump Export command-line utility DUMPFILE parameter
- Oracle Data Pump Import command-line mode DUMPFILE parameter

1.10 Exporting and Importing Between Different Oracle Database Releases

You can use Oracle Data Pump to migrate all or any portion of an Oracle Database between different releases of the database software.

Typically, you use the Oracle Data Pump Export `VERSION` parameter to migrate between database releases. Using `VERSION` generates an Oracle Data Pump dump file set that is compatible with the specified version.

The default value for `VERSION` is `COMPATIBLE`. This value indicates that exported database object definitions are compatible with the release specified for the `COMPATIBLE` initialization parameter.

In an upgrade situation, when the target release of an Oracle Data Pump-based migration is higher than the source, you typically do not have to specify the `VERSION` parameter. When the target release is higher then the source, all objects in the source database are compatible with the higher target release. However, an exception is when an entire Oracle Database 11g (Release 11.2.0.3 or higher) is exported in preparation for importing into Oracle Database 12c Release 1 (12.1.0.1) or later. In this case, to include a complete set of Oracle Database internal component metadata, explicitly specify `VERSION=12` with `FULL=YES`.

In a downgrade situation, when the target release of an Oracle Data Pump-based migration is lower than the source, set the `VERSION` parameter value to be the same version as the target. An exception is when the target release version is the same as the value of the `COMPATIBLE` initialization parameter on the source system. In that case, you do not need to specify `VERSION`. In general, however, Oracle Data Pump import cannot read dump file sets created by an Oracle Database release that is newer than the current release, unless you explicitly specify the `VERSION` parameter.

Keep the following information in mind when you are exporting and importing between different database releases:

- On an Oracle Data Pump export, if you specify a database version that is older than the current database version, then a dump file set is created that you can import into that older version of the database. For example, if you are running Oracle Database 19c, and you specify `VERSION=12.2` on an export, then the dump file set that is created can be imported into an Oracle Database 12c (Release 12.2) database.

  > **Note:**
  >
  > – Database privileges that are valid only in Oracle Database 12c Release 1 (12.1.0.2) and later (for example, the `READ` privilege on tables, views, materialized views, and synonyms) cannot be imported into Oracle Database 12c Release 1 (12.1.0.1) or earlier. If an attempt is made to do so, then Import reports it as an error, and continues the import operation.
  >
  > – When you export to a release earlier than Oracle Database 12c Release 2 (12.2.0.1), Oracle Data Pump does not filter out object names longer than 30 bytes. The objects are exported. At import time, if you attempt to create an object with a name longer than 30 bytes, then an error is returned.

- If you specify an Oracle Database release that is older than the current Oracle Database release, then certain features and data types can be unavailable. For example, specifying `VERSION=10.1` causes an error if data compression is also specified for the job, because compression was not supported in Oracle Database 10g release 1 (10.1). Another example: If a user-defined type or Oracle-supplied type in the source Oracle Database release is a later version than the type in the target Oracle Database release, then that type is not loaded, because it does not match any version of the type in the target database.

- Oracle Data Pump Import can always read Oracle Data Pump dump file sets created by older Oracle Database releases.
- When operating across a network link, Oracle Data Pump requires that the source and target Oracle Database releases differ by no more than two versions.

  For example, if one database is Oracle Database 12c, then the other Oracle Database release must be 12c, 11g, or 10g. Oracle Data Pump checks only the major version number (for example, 10g,11g, 12c), not specific Oracle Database release numbers (for example, 12.2, 12.1, 11.1, 11.2, 10.1, or 10.2).
- Importing Oracle Database 11g dump files that contain table statistics into Oracle Database 12c Release 1 (12.1) or later Oracle Database releases can result in an Oracle ORA-39346 error. This error occurs because Oracle Database 11g dump files contain table statistics as metadata. Oracle Database 12c Release 1 (12.1) and later releases require table statistics to be presented as table data. The workaround is to ignore the error during the import operation. After the import operation completes, regather table statistics.
- All forms of `LONG` data types (`LONG`, `LONG RAW`, `LONG VARCHAR`, `LONG VARRAW`) were deprecated in Oracle8i Release 8.1.6. For succeeding releases, the `LONG` data type was provided for backward compatibility with existing applications. In new applications developed with later releases, Oracle strongly recommends that you use `CLOB` and `NCLOB` data types for large amounts of character data.

Related Topics

- Oracle Data Pump Export command-line utility VERSION parameter
- Oracle Data Pump Import command-line mode VERSION parameter

> **See Also:**
>
> - READ and SELECT Object Privileges in *Oracle Database Security Guide* for more information about the `READ` and `READ ANY TABLE` privileges

1.11 Exporting and Importing Blockchain Tables with Oracle Data Pump

To export or import blockchain tables, review these minimum requirements, restrictions, and guidelines.

If you use Oracle Data Pump with blockchain tables, then you can use only `CONVENTIONAL access_method` or, beginning with Oracle Database 23ai, Transportable Tablespaces (TTS).

Blockchain tables are exported only under the following conditions:

- The `VERSION` parameter for the export is explicitly set to `21.0.0.0.0` or later.
- The `VERSION` parameter is set to (or defaults to) `COMPATIBLE`, and the database compatibility is set to `21.0.0.0.0` or later.
- The `VERSION` parameter is set to `LATEST`, and the database release is set to `21.0.0.0.0` or later.

If you attempt to use Oracle Data Pump options that are not supported with blockchain tables, then you receive errors when you attempt to use those options.

The following options of Oracle Data Pump are not supported with blockchain tables:

- `ACCESS_METHOD=[DIRECT_PATH, EXTERNAL_TABLE, INSERT_AS_SELECT]`
- `TABLE_EXISTS_ACTION=[REPLACE | APPEND | TRUNCATE]`

  These options result in errors when you attempt to use them to import data into an existing blockchain table.
- `CONTENT=DATA_ONLY`

  This option results in error when you attempt to import data into a blockchain table.
- `PARTITION_OPTIONS= [DEPARTITIONING | MERGE]`

  If you request departitioning using this option with blockchain tables, then the blockchain tables are skipped during departitioning.
- `NETWORK IMPORT`
- `TRANSPORTABLE` in Oracle Database 23ai Free and earlier Oracle Database releases
- `SAMPLE`, `QUERY`, and `REMAP_DATA`

1.12 Unload and Load Vectors Using Oracle Data Pump

Starting with Oracle Database 23ai, Oracle Data Pump enables you to use multiple components to load and unload vectors to databases.

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another. Oracle Data Pump is made up of three distinct components: Command-line clients, expdp and impdp; the DBMS_DATAPUMP PL/SQL package (also known as the Data Pump API); and the DBMS_METADATA PL/SQL package (also known as the Metadata API).

Unloading and Loading a table with vector datatype columns is supported in all modes (`FULL`, `SCHEMA`, `TABLES`) using all the available access methods (`DIRECT_PATH`, `EXTERNAL_TABLE`, `AUTOMATIC`, `INSERT_AS_SELECT`).

Examples Vector Export and Import Syntax

```
expdp <username>/<password>@<Database-instance-TNS-alias>  dumpfile=<dumpfile-
name>.dmp directory=<directory-name> full=y metrics=y
access_method=direct_path

expdp <username>/<password>@<Database-instance-TNS-alias>  dumpfile=<dumpfile-
name>.dmp directory=<directory-name> schemas=<schema-name> metrics=y
access_method=external_table

expdp <username>/<password>@<Database-instance-TNS-alias>  dumpfile=<dumpfile-
name>.dmp directory=<directory-name> tables=<schema-name>.<table-name>
metrics=y access_method=direct_path

impdp <username>/<password>@<Database-instance-TNS-alias>  dumpfile=<dumpfile-
name>.dmp directory=<directory-name> metrics=y access_method=direct_path
```

Note:

- `TABLE_EXISTS_ACTION=APPEND | TRUNCATE` can only be used with the `EXTERNAL_TABLE` access method.
- `TABLE_EXISTS_ACTION=APPEND | TRUNCATE` can load `VECTOR` column data into a `VARCHAR2` column if the conversion can fit into that `VARCHAR2`.
- `TABLE_EXISTS_ACTION=APPEND | TRUNCATE` can only load a `VECTOR` column with the source `VECTOR` data dimasion that matches that loaded `VECTOR` column's dimension. If the dimension does not match, then an error is raised.
- `TABLE_EXISTS_ACTION=REPLACE` supports any access method.
- It is not possible to use a the transportable tablespace mode with vector indexes. However, this mode supports tables with the `VECTOR` datatype.

Related Topics

- Overview of Oracle Data Pump
- DBMS_DATAPUMP
- DBMS_METADATA

1.13 Managing SecureFiles Large Object Exports with Oracle Data Pump

Exports of SecureFiles large objects (LOBs) are affected by the content type, the `VERSION` parameter, and other variables.

LOBs are a set of data types that are designed to hold large amounts of data. When you use Oracle Data Pump Export to export SecureFiles LOBs, the export behavior depends on several things, including the Export `VERSION` parameter value, whether a content type (`ContentType`) is present, and whether the LOB is archived and data is cached.

The following scenarios cover different combinations of these variables:

- If a table contains SecureFiles LOBs with a `ContentType`, and the Export `VERSION` parameter is set to a value earlier than `11.2.0.0.0`, then the `ContentType` is not exported.
- If a table contains SecureFiles LOBs with a `ContentType`, and the Export `VERSION` parameter is set to a value of `11.2.0.0.0` or later, then the `ContentType` is exported and restored on a subsequent import.
- If a table contains a SecureFiles LOB that is currently archived, the data is cached, and the Export `VERSION` parameter is set to a value earlier than `11.2.0.0.0`, then the SecureFiles LOB data is exported and the archive metadata is dropped. In this scenario, if `VERSION` is set to `11.1` or later, then the SecureFiles LOB becomes a plain SecureFiles LOB. But if `VERSION` is set to a value earlier than `11.1`, then the SecureFiles LOB becomes a `BasicFiles` LOB.
- If a table contains a SecureFiles LOB that is currently archived, but the data is not cached, and the Export `VERSION` parameter is set to a value earlier than `11.2.0.0.0`, then an `ORA-45001` error is returned.

- If a table contains a SecureFiles LOB that is currently archived, the data is cached, and the Export `VERSION` parameter is set to a value of `11.2.0.0.0` or later, then both the cached data and the archive metadata is exported.

Refer to *Oracle Database SecureFiles and Large Objects Developer's Guide* for more information about SecureFiles LOBs.

Related Topics

- *Oracle Database SecureFiles and Large Objects Developer's Guide*

1.14 Oracle Data Pump Process Exit Codes

To check the status of your Oracle Data Pump export and import operations, review the process exit codes in the log file.

Oracle Data Pump provides the results of export and import operations immediately upon completion. In addition to recording the results in a log file, Oracle Data Pump can also report the outcome in a process exit code. Use the Oracle Data Pump exit code to check the outcome of an Oracle Data Pump job from the command line or a script:

Table 1-1 Oracle Data Pump Exit Codes

Exit Code	Meaning
`EX_SUCC 0`	The export or import job completed successfully. No errors are displayed to the output device or recorded in the log file, if there is one.
`EX_SUCC_ERR 5`	The export or import job completed successfully, but there were errors encountered during the job. The errors are displayed to the output device and recorded in the log file, if there is one.
`EX_FAIL 1`	The export or import job encountered one or more irrecoverable errors, including the following: • Errors on the command line or in command syntax • Oracle Database errors from which export or import cannot recover • Operating system errors (such as `malloc`) • Invalid parameter values that prevent the job from starting (for example, an invalid directory object specified in the `DIRECTORY` parameter) An irrecoverable error is displayed to the output device but may not be recorded in the log file. Whether it is recorded in the log file can depend on several factors, including: • Was a log file specified at the start of the job? • Did the processing of the job proceed far enough for a log file to be opened?

1.15 How Oracle Data Pump Manages Dump File Blocks

In releases before Oracle Database 23ai, Oracle Data Pump uses Header Blocks. Starting with Oracle Database 23ai, Oracle Data Pump uses Trailer Blocks.

- Dump Files for Exports
  Learn about dump file types, annd differences of export dump flies between Oracle Data Pump and SQL Mode dump files
- Trailer Block File Layout in Dump Files
  Starting with Oracle Database 23ai, by default both SQL-Mode and Data Pump Export files use a trailer block format that facilitates use with object stores in Oracle Cloud Infrastructure.

- Header Block File Layout in Dump Files
  In Oracle Database 21c and earlier releases, dump files use Header Blocks.
- Types of Dump File Trailer Blocks
  There are two types of trailer blocks that are used for Oracle Data Pump and SQL-Mode dump files.

1.15.1 Dump Files for Exports

Learn about dump file types, annd differences of export dump flies between Oracle Data Pump and SQL Mode dump files

Export dump files are created when you use either the PL/SQL ORACLE_DATAPUMP external access driver API, or the Oracle Data Pump Export (`expdp`) command-line utility.

Types of Export Dump Files

There are two types of dump files that Oracle Data Pump can create during an export operation:

- **Extensible Files** are export dump files that are extensible if the file size attribute specified is null or zero. With extensible files, Data Pump continues to write as much data to the file as is needed, or until the device runs out of physical space, or until the process reaches its assigned disk quota.
- **Fixed-Size Files** are export dump files where the file size attribute specified is greater than zero. When a file size greater than null or zero is specified, Data Pump only writes data to the dump file up to the specified file size. If fixed-size files are used, and the size of the object being exported exceeds the remaining available space specfied for dump file size, then that object can span over multiple dump files

> **Note:**
>
> It is possible to use both extensible and fixed-size files in an Oracle Data Pump export operation. However, you can only do this by using the `DBMS_DATAPUMP` PL/SQL API. If you use the `expdp` command line client, then you are permitted to specify only one file type type for a given export operation.

1.15.2 Trailer Block File Layout in Dump Files

Starting with Oracle Database 23ai, by default both SQL-Mode and Data Pump Export files use a trailer block format that facilitates use with object stores in Oracle Cloud Infrastructure.

In Oracle Database 21c and earlier releases, Header Blocks are the default layout format used with dump files. Dump files were required to be located on a local file system. In Oracle Database 23ai and later releases, the default format changes from Header Blocks to Trailer Blocks. This default format change facilitates your ability to write dump files to object stores in the cloud.

Overview of Trailer Blocks

Unlike Header Blocks, Trailer Blocks are not written until the file is being closed. An initial Header Block is written with limited information at file create. However, Trailer Blocks are not written to disk. Instead, the Trailer Block is maintained and updated with the Control Table until written to disk. After they are written to disk, these Trailer Blocks contain the information needed to correctly process the data in the files when they are later read.

> **Note:**
>
> Because this feature is new with Oracle Database 23ai, if you export dump files using the trailer block format, then the Data Pump export dump file set will only be readable by Data Pump servers running Oracle Database 23ai or a later release.
>
> The `VERSION` parameter value controls the dump file format by specifying whether the database `COMPATIBLE` setting is set to Oracle Database 23ai, with this changed default, or if the COMPATIBLE setting is set to an earlier Oracle Database release, where the default is to use Header Blocks. The credential used for the object store indicates which API is used (`Native|Swift`). The API used is what determines dump file format.

How Trailer Blocks Write to Cloud Object Stores

When Trailer Blocks are enabled, Oracle Data Pump writes and processes the `.dmp` files stored in the cloud the same way as it writes and processes `.dmp` files stored on local file systems. The procedure flow is as follows:

1. Log in as the user with a credential for the data store. The value for the credential used to connect to the object store is the name of a credential object owned by the database user that starts Data Dump export (`expdp`).
2. If the `CREDENTIAL` parameter is specified, then the value for the `DUMPFILE` parameter is a list of comma-delimited strings that Data Pump treats as separated strings that the data pump treats as Uniform Resource Identifiers (URIs) in the cloud storage.

> **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.

The log file location is set by the `DEFAULT_DIRECTORY` parameter. You can choose to specify directory object names as part of the file names for LOGFILE. If a URI is specified for a dump file, and the CREDENTIAL the parameter is not specified, then you will receive an error.

Prerequisite to Storing Dump Files on a Cloud Object Store

Before you can use Oracle Data Pump Export (`expdp`) to access an object store, you must first have the credentials for that object store in a wallet pointed to by the `WALLET_LOCATION` parameter in the `sqlnet.ora` file. You must provide a user name and password to authenticate to the cloud, and you must provide a location for a certificate for the object store in the wallet. In the following syntax, *`location`* is the location of the wallet, *`file-for-trusted-certificate`* is the file name of the certificate, and `walletpassword` is the password for the Oracle wallet:

```
orapki wallet add -wallet location -trusted_cert -cert file-for-trusted-
certificate -pwd walletpassword
```

The `CREDENTIAL` parameter contains the name of the credential that Data Pump export uses to build a key to look up in the wallet. In the preceding example, to you would specify `CREDENTIAL=obm` on the `expdp` command line.

Related Topics

- Using The Secure External Password Store (Doc ID 340559.1)
- Oracle Data Pump Export command-line utility CREDENTIAL parameter

1.15.3 Header Block File Layout in Dump Files

In Oracle Database 21c and earlier releases, dump files use Header Blocks.

> **Note:**
>
> Starting with Oracle Database 23ai, Header Blocks are a legacy format.

Dump file layout comes in different forms.

Data Pump Dump File Layout with Header Blocks (Oracle Database Releases 10.1 to 21c Default)

For Data Pump export files from Release 10.1 to Release 21c, the basic dump file layout has the following components:

1. A file header block containing various fields (for example, dump file version number, charset ID, offset and length to master table data, if present).
2. One or more blocks that contain system metadata, such as `USERS`, `INDEXES`, `GRANTS`, or other metadata.
3. One or more blocks that contain table streams for each user table that is being exported. For example: `SCOTT.EMP`.
4. One or more blocks that contain the table stream for the export job primary table.

The VERSION Parameter and Dump File Compatibility

The `VERSION` parameter specifies the version of the database object that are exported. It also specifies the dump file compatibility. By default, `VERSION` is set to `COMPATIBLE`, which corresponds to the database compatibility level as specified on the `COMPATIBLE` initialization parameter.

Starting with Oracle Database 23ai, if you update the `COMPATIBLE` initialization parameter to 23, and then want to export dump files to a database where COMPATIBLE is not set to 23 (that is, you want to us the legacy Header Block format), you must specify a version earlier than 23. For example, when `VERSION` is specified as `19`, then Header block (legacy) format is used for dump files, and the dump file version is 5.1

```
VERSION=19
```

For more details, see the Data Pump export (`expdp`) and `impdp`) `VERSION` parameter.

Related Topics

- Examples Using DataPump VERSION Parameter And Its Relationship To Database COMPATIBLE Parameter (Doc ID 864582.1)

1.15.4 Types of Dump File Trailer Blocks

There are two types of trailer blocks that are used for Oracle Data Pump and SQL-Mode dump files.

The type of Export option that you use affects what kind of trailer block type is used for dump files.

Disk-Based Trailer Blocks

Disk-based trailer blocks are blocks that are written to the actual dump file where where its corresponding header block and other data reside. SQL-Mode dump files can only use disk-based trailers.

Table-Based Trailer Blocks

Table-based trailer blocks are trailer blocks that are stored externally to the dump file, in the export job primary table. Storing the dump file block assists with two purposes:

1. The process that initially creates the dump file (the Primary export process) and formats the header block is not the same process that later will have to use the header block as the basis of the file trailer block. Instead, this is done by a Worker process. Because the Worker process writes sequentially to the trailer block, and has no need to seek and read the file header block, storing the file trailer block in the Primary table is simply a place to save the information until a Worker process can later fetch it and write it to disk, making it a disk-based trailer.
2. For the stream trailer block, storing file information in a table-based trailer block simplifies size allocation management. All blocks in a dump file are 4K in size. If a disk-based trailer block was used, then every table being exported would require adding a trailer block to the file itself, which potentially could result in a substantial increase in the size of the output dump file set. For user tables, the stream trailer is *always* table-based. This is true for all user tables *except* the primary table, which uses a disk-based stream trailer block.

Any file or trailer blocks stored in the primary table will be in compressed format. The 4K header and trailer blocks compress to around 200 bytes or less each.

1.16 How to Monitor Oracle Data Pump Jobs with Unified Auditing

To monitor and record specific user database actions, perform auditing on Data Pump jobs with unified auditing.

To monitor and record specific user database actions, you can perform auditing on Oracle Data Pump jobs. Oracle Data Pump uses unified auditing, in which all audit records are centralized in one place. To set up unified auditing, you create a unified audit policy, or alter an existing audit policy. An audit policy is a named group of audit settings that enable you to audit a particular aspect of user behavior in the database.

To create the policy, use the SQL `CREATE AUDIT POLICY` statement. After creating the audit policy, use the `AUDIT` SQL statement to enable the policy.

To disable the policy, use the `NOAUDIT` SQL statement.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about the SQL `CREATE AUDIT POLICY`,`ALTER AUDIT POLICY`, `AUDIT`, and `NOAUDIT` statements
> - *Oracle Database Security Guide* for more information about using auditing in an Oracle database

1.17 Encrypted Data Security Warnings for Oracle Data Pump Operations

Oracle Data Pump warns you when encrypted data is exported as unencrypted data.

During Oracle Data Pump export operations, you receive an `ORA-39173` warning when Oracle Data Pump encounters encrypted data specified when the export job was started. This `ORA-39173` warning ("ORA-39173: Encrypted data has been stored unencrypted in dump file set") is also written to the the audit record. You can view the ORA-39173 errors encountered during the export operation by checking the `DP_WARNINGS1` column in the unified audit trail. Obtain the audit information by running the following SQL statement:

```
SELECT DP_WARNINGS1 FROM UNIFIED_AUDIT_TRAIL WHERE ACTION_NAME = 'EXPORT'
ORDER BY 1;
```

1.18 How Does Oracle Data Pump Handle Timestamp Data?

Learn about factors that can affect successful completion of export and import jobs that involve the timestamp data types `TIMESTAMP WITH TIMEZONE` and `TIMESTAMP WITH LOCAL TIMEZONE`.

> **Note:**
>
> The information in this section applies only to Oracle Data Pump running on Oracle Database 12c and later.

- TIMESTAMP WITH TIMEZONE Restrictions
  Export and import jobs that have `TIMESTAMP WITH TIME ZONE` data are restricted.
- TIMESTAMP WITH LOCAL TIME ZONE Restrictions
  Moving tables using a transportable mode is restricted.

1.18.1 TIMESTAMP WITH TIMEZONE Restrictions

Export and import jobs that have `TIMESTAMP WITH TIME ZONE` data are restricted.

- Understanding TIMESTAMP WITH TIME ZONE Restrictions
  Carrying out export and import jobs that have `TIMESTAMP WITH TIME ZONE` data requires understanding information about your time zone file data and Oracle Database release.

- Oracle Data Pump Support for TIMESTAMP WITH TIME ZONE Data
  Oracle Data Pump supports `TIMESTAMP WITH TIME ZONE` data during different export and import modes.
- Time Zone File Versions on the Source and Target
  Successful job completion can depend on whether the source and target time zone file versions match.

1.18.1.1 Understanding TIMESTAMP WITH TIME ZONE Restrictions

Carrying out export and import jobs that have `TIMESTAMP WITH TIME ZONE` data requires understanding information about your time zone file data and Oracle Database release.

When you import a dump file, the time zone version of the destination (target) database must be either the same version, or a more recent (higher) version than the time zone version of the source database from which the export was taken. Successful job completion can depend on the following factors:

- The version of the Oracle Database time zone files on the source and target databases.
- The export/import mode and whether the Data Pump version being used supports `TIMESTAMP WITH TIME ZONE` data. (Oracle Data Pump 11.2.0.1 and later releases provide support for `TIMESTAMP WITH TIME ZONE` data.)

To identify the time zone file version of a database, you can run the following SQL statement:

```
SQL> SELECT VERSION FROM V$TIMEZONE_FILE;
```

Related Topics

- Choosing a Time Zone File

1.18.1.2 Oracle Data Pump Support for TIMESTAMP WITH TIME ZONE Data

Oracle Data Pump supports `TIMESTAMP WITH TIME ZONE` data during different export and import modes.

Oracle Data Pump provides support for `TIMESTAMP WITH TIME ZONE` data during different export and import modes when versions of the Oracle Database time zone file are different on the source and target databases. Supported modes include non-transportable mode, transportable tablespace and transportable table mode, and full transportable mode.

Non-transportable Modes

- If the dump file is created with a Data Pump version that supports `TIMESTAMP WITH TIME ZONE` data (11.2.0.1 or later), then the time zone file version of the export system is recorded in the dump file. Oracle Data Pump uses that information to determine whether data conversion is necessary. If the target database knows about the source time zone version, but is actually using a later version, then the data is converted to the later version. `TIMESTAMP WITH TIME ZONE` data cannot be downgraded, so if you attempt to import to a target that is using an earlier version of the time zone file than the source used, the import fails.
- If the dump file was created with an Oracle Data Pump version earlier than Oracle Database 11g release 2 (11.2.0.1), then `TIMESTAMP WITH TIME ZONE` data is not supported. No conversion is done, and corruption may occur.

Transportable Tablespace and Transportable Table Modes

- In transportable tablespace and transportable table modes, if the source and target have different time zone file versions, tables with `TIMESTAMP WITH TIME ZONE` columns are not created. A warning is displayed at the beginning of the job that shows the source and target database time zone file versions. A message is also displayed for each table not created. This is true even if the Oracle Data Pump version used to create the dump file supports `TIMESTAMP WITH TIME ZONE` data. (Release 11.2.0.1 and later support `TIMESTAMP WITH TIMEZONE` data.)
- If the source is earlier than Oracle Database 11g release 2 (11.2.0.1), then the time zone file version must be the same on the source and target database for all transportable jobs, regardless of whether the transportable set uses `TIMESTAMP WITH TIME ZONE` columns.

Full Transportable Mode

Full transportable exports and imports are supported when the source database is at least Oracle Database 11g release 2 (11.2.0.3) and the target is at least Oracle Database 12c release 1 (12.1) or later.

Oracle Data Pump 11.2.0.1 and later provide support for `TIMESTAMP WITH TIME ZONE` data. Therefore, in full transportable operations, tables with `TIMESTAMP WITH TIME ZONE` columns are created. If the source and target database have different time zone file versions, then `TIMESTAMP WITH TIME ZONE` columns from the source are converted to the time zone file version of the target.

Related Topics

- Limitations on Transportable Tablespaces
- Full Export Mode
  You can use Oracle Data Pump to carry out a full database export by using the `FULL` parameter.
- Full Import Mode
  To specify a full import with Oracle Data Pump, use the `FULL` parameter.

1.18.1.3 Time Zone File Versions on the Source and Target

Successful job completion can depend on whether the source and target time zone file versions match.

- If the Oracle Database time zone file version is the same on the source and target databases, then conversion of `TIMESTAMP WITH TIME ZONE` data is not necessary. The export/import job should complete successfully.

  The exception to this is a transportable tablespace or transportable table export performed using a Data Pump release earlier than 11.2.0.1. In that case, tables in the dump file that have `TIMESTAMP WITH TIME ZONE` columns are not created on import even though the time zone file version is the same on the source and target.
- If the source time zone file version is not available on the target database, then the job fails. The version of the time zone file on the source may not be available on the target because the source may have had its time zone file updated to a later version but the target has not. For example, if the export is done on Oracle Database 11*g* release 2 (11.2.0.2) with a time zone file version of 17, and the import is done on 11.2.0.2 with only a time zone file of 16 available, then the job fails.

1.18.2 TIMESTAMP WITH LOCAL TIME ZONE Restrictions

Moving tables using a transportable mode is restricted.

If a table is moved using a transportable mode (transportable table, transportable tablespace, or full transportable), and the following conditions exist, then a warning is issued and the table is not created:

- The source and target databases have different database time zones.
- The table contains `TIMESTAMP WITH LOCAL TIME ZONE` data types.

To successfully move a table that was not created because of these conditions, use a non-transportable export and import mode.

1.19 Character Set and Globalization Support Considerations

Learn about Globalization support of Oracle Data Pump Export and Import using character set conversion of user data, and data definition language (DDL).

- Data Definition Language (DDL)
  The Export utility writes dump files using the database character set of the export system.
- Single-Byte Character Sets and Export and Import
  Ensure that the export database and the import database use the same character set.
- Multibyte Character Sets and Export and Import
  During an Oracle Data Pump export and import, the character set conversion depends on the importing Oracle Database character set.

1.19.1 Data Definition Language (DDL)

The Export utility writes dump files using the database character set of the export system.

When the dump file is imported, a character set conversion is required for DDL only if the database character set of the import system is different from the database character set of the export system.

To minimize data loss due to character set conversions, ensure that the import database character set is a superset of the export database character set.

1.19.2 Single-Byte Character Sets and Export and Import

Ensure that the export database and the import database use the same character set.

If the system on which the import occurs uses a 7-bit character set, and you import an 8-bit character set dump file, then some 8-bit characters may be converted to 7-bit equivalents. An indication that this has happened is when accented characters lose the accent mark.

To avoid this unwanted conversion, ensure that the export database and the import database use the same character set.

1.19.3 Multibyte Character Sets and Export and Import

During an Oracle Data Pump export and import, the character set conversion depends on the importing Oracle Database character set.

During character set conversion, any characters in the export file that have no equivalent in the import database character set are replaced with a default character. The import database character set defines the default character.

If the import system has to use replacement characters while converting DDL, then a warning message is displayed and the system attempts to load the converted DDL.

If the import system has to use replacement characters while converting user data, then the default behavior is to load the converted data. However, it is possible to instruct the import system to reject rows of user data that were converted using replacement characters. See the Import `DATA OPTIONS` parameter for details.

To guarantee 100% conversion, the import database character set must be a superset (or equivalent) of the character set used to generate the export file.

> **Caution:**
>
> When the database character set of the export system differs from that of the import system, the import system displays informational messages at the start of the job that show what the database character set is.
>
> When the import database character set is not a superset of the character set used to generate the export file, the import system displays a warning that possible data loss may occur due to character set conversions.

Related Topics

- DATA_OPTIONS

1.20 Oracle Data Pump Behavior with Data-Bound Collation

Oracle Data Pump supports data-bound collation (DBC).

Oracle Data Pump Export always includes all available collation metadata into the created dump file. This includes:

- Current default collations of exported users' schemas
- Current default collations of exported tables, views, materialized views and PL/SQL units (including user-defined types)
- Declared collations of all table and cluster character data type columns

When importing a dump file exported from an Oracle Database 12c Release 2 (12.2) database, Oracle Data Pump Import's behavior depends both on the effective value of the Oracle Data Pump `VERSION` parameter at the time of import, and on whether the data-bound collation (DBC) feature is enabled in the target database. The effective value of the `VERSION` parameter is determined by how it is specified. You can specify the parameter using the following:

- `VERSION=`*`n`*, which means the effective value is the specific version number *`n`*. For example: `VERSION=19`
- `VERSION=LATEST`, which means the effective value is the currently running database version
- `VERSION=COMPATIBLE`, which means the effective value is the same as the value of the database initialization parameter `COMPATIBLE`. This is also true if no value is specified for `VERSION`.

For the DBC feature to be enabled in a database, the initialization parameter `COMPATIBLE` must be set to 12.2 or higher, and the initialization parameter `MAX_STRING_SIZE` must be set to `EXTENDED`.

If the effective value of the Oracle Data Pump Import `VERSION` parameter is 12.2, and DBC is enabled in the target database, then Oracle Data Pump Import generates DDL statements with collation clauses referencing collation metadata from the dump file. Exported objects are created with the original collation metadata that they had in the source database.

No collation syntax is generated if DBC is disabled, or if the Oracle Data Pump Import `VERSION` parameter is set to a value lower than 12.2.