13
Transporting Data

Transporting data moves the data from one database to another.

> **Note:**
>
> A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

- About Transporting Data
  You can transport data at the following levels: database, tablespaces, tables, partitions, and subpartitions.
- Transporting Databases
  You can transport a database to a new Oracle Database instance.
- Transporting Tablespaces Between Databases
  You can transport tablespaces between databases.
- Transporting Tables, Partitions, or Subpartitions Between Databases
  You can transport tables, partitions, and subpartitions between databases.
- Converting Data Between Platforms
  To transport tables across platforms, check platform endianness, and review other restrictions.
- Guidelines for Transferring Data Files
  You should follow a set of guidelines when transferring the data files.

13.1 About Transporting Data

You can transport data at the following levels: database, tablespaces, tables, partitions, and subpartitions.

- Purpose of Transporting Data
  Transporting data is much faster than performing either an export/import or unload/load of the same data. It is faster because, for user-defined tablespaces, the data files containing all of the actual data are copied to the target location, and you use Data Pump to transfer only the metadata of the database objects to the new database.
- Transporting Data: Scenarios
  Transporting data is useful in several scenarios.
- Transporting Data Across Platforms
  You can transport data across platforms.

- General Limitations on Transporting Data
  There are general limitations on transporting data. There are also limitations that are specific to full transportable export/import, transportable tablespaces, or transportable tables.
- Compatibility Considerations for Transporting Data
  When transporting data, Oracle Database computes the lowest compatibility level at which the target database must run.

13.1.1 Purpose of Transporting Data

Transporting data is much faster than performing either an export/import or unload/load of the same data. It is faster because, for user-defined tablespaces, the data files containing all of the actual data are copied to the target location, and you use Data Pump to transfer only the metadata of the database objects to the new database.

You can transport data at any of the following levels:

- Database

  You can use the **full transportable export/import** feature to move an entire database to a different database instance.
- Tablespaces

  You can use the **transportable tablespaces** feature to move a set of tablespaces between databases.
- Tables, partitions, and subpartitions

  You can use the **transportable tables** feature to move a set of tables, partitions, and subpartitions between databases.

Transportable tablespaces and transportable tables only transports data that resides in user-defined tablespaces. However, full transportable export/import transports data that resides in both user-defined and administrative tablespaces, such as `SYSTEM` and `SYSAUX`. Full transportable export/import transports metadata for objects contained within the user-defined tablespaces and both the metadata and data for user-defined objects contained within the administrative tablespaces. Specifically, with full transportable export/import, the export dump file includes only the metadata for objects contained within the user-defined tablespaces, but it includes both the metadata and the data for user-defined objects contained within the administrative tablespaces.

13.1.2 Transporting Data: Scenarios

Transporting data is useful in several scenarios.

- Scenarios for Full Transportable Export/import
  The full transportable export/import feature is useful in several scenarios.
- Scenarios for Transportable Tablespaces or Transportable Tables
  The transportable tablespaces or transportable tables feature is useful in several scenarios.

13.1.2.1 Scenarios for Full Transportable Export/import

The full transportable export/import feature is useful in several scenarios.

- Moving a Non-CDB Into a CDB
  The multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes one or many customer-created pluggable databases (PDBs). You can move a non-CDB into a CDB by transporting the database.
- Moving a Database to a New Computer System
  You can use full transportable export/import to move a database from one computer system to another. You might want to move a database to a new computer system to upgrade the hardware or to move the database to a different platform.
- Upgrading to a New Release of Oracle Database
  You can use full transportable export/import to upgrade a database from an Oracle Database 11*g* Release 2 (11.2.0.3) or later to Oracle Database 19c.

13.1.2.1.1 Moving a Non-CDB Into a CDB

The multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes one or many customer-created pluggable databases (PDBs). You can move a non-CDB into a CDB by transporting the database.

The transported database becomes a pluggable database (PDB) associated with the CDB. Full transportable export/import can move an Oracle Database 11*g* Release 2 (11.2.0.3) or later into an Oracle Database 19c CDB efficiently.

See Also:

- "Transporting a Database Using an Export Dump File" for instructions that describe transporting a non-CDB into a CDB using an export dump file
- "Transporting a Database Over the Network" for instructions that describe transporting a non-CDB into a CDB over the network
- *Oracle Multitenant Administrator's Guide*

13.1.2.1.2 Moving a Database to a New Computer System

You can use full transportable export/import to move a database from one computer system to another. You might want to move a database to a new computer system to upgrade the hardware or to move the database to a different platform.

See Also:

- "Transporting Databases"
- "Transporting Data Across Platforms"

13.1.2.1.3 Upgrading to a New Release of Oracle Database

You can use full transportable export/import to upgrade a database from an Oracle Database 11*g* Release 2 (11.2.0.3) or later to Oracle Database 19c.

To do so, install Oracle Database 19c and create an empty database. Next, use full transportable export/import to transport the Oracle Database 11*g* Release 2 (11.2.0.3) or later database into the Oracle Database 19c database.

> **See Also:**
>
> - "Transporting Databases"
> - *Oracle Database Installation Guide*

13.1.2.2 Scenarios for Transportable Tablespaces or Transportable Tables

The transportable tablespaces or transportable tables feature is useful in several scenarios.

- Scenarios That Apply to Transportable Tablespaces or Transportable Tables
  For some scenarios, either transportable tablespaces or transportable tables can be useful. For other scenarios, only transportable tablespaces can be useful, or only transportable tables can be useful.
- Transporting and Attaching Partitions for Data Warehousing
  You can use transportable tables and tranportable tablespaces to attach partitions for data warehousing.
- Publishing Structured Data on CDs
  Transportable tablespaces and transportable tables both provide a way to publish structured data on CDs.
- Mounting the Same Tablespace Read-Only on Multiple Databases
  You can use transportable tablespaces to mount a tablespace read-only on multiple databases.
- Archiving Historical Data
  When you use transportable tablespaces or transportable tables, the transported data is a self-contained set of files that can be imported into any Oracle database. Therefore, you can archive old or historical data in an enterprise data warehouse using the transportable tablespaces and transportable tables procedures.
- Using Transportable Tablespaces to Perform TSPITR
  You can use transportable tablespaces to perform tablespace point-in-time recovery (TSPITR).
- Copying or Moving Individual Tables
  You can use transportable tables to move a table or a set of tables from one database to another without transporting the entire tablespaces that contain the tables. You can also copy or move individual partitions and subpartitions from one database to another using transportable tables.

13.1.2.2.1 Scenarios That Apply to Transportable Tablespaces or Transportable Tables

For some scenarios, either transportable tablespaces or transportable tables can be useful. For other scenarios, only transportable tablespaces can be useful, or only transportable tables can be useful.

Table 13-1 shows which feature can be used for each scenario.

Table 13-1 Scenarios for Transportable Tablespaces and Transportable Tables

Scenarios	Transportable Tablespaces	Transportable Tables
Transporting and Attaching Partitions for Data Warehousing	Yes	Yes
Publishing Structured Data on CDs	Yes	Yes
Archiving Historical Data	Yes	Yes
Using Transportable Tablespaces to Perform TSPITR	Yes	No
Copying or Moving Individual Tables	No	Yes

The following sections describe these scenarios in more detail.

13.1.2.2.2 Transporting and Attaching Partitions for Data Warehousing

You can use transportable tables and tranportable tablespaces to attach partitions for data warehousing.

Typical enterprise data warehouses contain one or more large fact tables. These fact tables can be partitioned by date, making the enterprise data warehouse a historical database. You can build indexes to speed up star queries. Oracle recommends that you build local indexes for such historically partitioned tables to avoid rebuilding global indexes every time you drop the oldest partition from the historical database.

Suppose every month you would like to load one month of data into the data warehouse. There is a large fact table in the data warehouse called `sales`, which has the following columns:

```
CREATE TABLE sales (invoice_no NUMBER,
   sale_year  INT NOT NULL,
   sale_month INT NOT NULL,
   sale_day   INT NOT NULL)
   PARTITION BY RANGE (sale_year, sale_month, sale_day)
     (partition jan2011 VALUES LESS THAN (2011, 2, 1),
      partition feb2011 VALUES LESS THAN (2011, 3, 1),
      partition mar2011 VALUES LESS THAN (2011, 4, 1),
      partition apr2011 VALUES LESS THAN (2011, 5, 1),
      partition may2011 VALUES LESS THAN (2011, 6, 1),
      partition jun2011 VALUES LESS THAN (2011, 7, 1));
```

You create a local non-prefixed index:

```
CREATE INDEX sales_index ON sales(invoice_no) LOCAL;
```

Initially, all partitions are empty, and are in the same default tablespace. Each month, you want to create one partition and attach it to the partitioned `sales` table.

Suppose it is July 2011, and you would like to load the July sales data into the partitioned table. In a staging database, you create a table, `jul_sales` with the same column types as the `sales` table. Optionally, you can create a new tablespace, `ts_jul`, before you create the table, and create the table in this tablespace. You can create the table `jul_sales` using the `CREATE TABLE ... AS SELECT` statement. After creating and populating `jul_sales`, you can also create an index, `jul_sale_index`, for the table, indexing the same column as the local index in the `sales` table. For detailed information about creating and populating a staging table in a data warehousing environment, see *Oracle Database Data Warehousing Guide*.

After creating the table and building the index, transport the table's data to the data warehouse in one of the following ways:

- You can use transportable tables to transport the `jul_sales` table to the data warehouse.
- If you created the `ts_jul` tablespace, then you can use transportable tablespaces to transport the tablespace `ts_jul` to the data warehouse.

In the data warehouse, add a partition to the `sales` table for the July sales data. This also creates another partition for the local non-prefixed index:

```
ALTER TABLE sales ADD PARTITION jul2011 VALUES LESS THAN (2011, 8, 1);
```

Attach the transported table `jul_sales` to the table `sales` by exchanging it with the new partition:

```
ALTER TABLE sales EXCHANGE PARTITION jul2011 WITH TABLE jul_sales
   INCLUDING INDEXES
   WITHOUT VALIDATION;
```

This statement places the July sales data into the new partition `jul2011`, attaching the new data to the partitioned table. This statement also converts the index `jul_sale_index` into a partition of the local index for the `sales` table. This statement should return immediately, because it only operates on the structural information and it simply switches database pointers. If you know that the data in the new partition does not overlap with data in previous partitions, you are advised to specify the `WITHOUT VALIDATION` clause. Otherwise, the statement goes through all the new data in the new partition in an attempt to validate the range of that partition.

If all partitions of the `sales` table came from the same staging database (the staging database is never destroyed), then the exchange statement always succeeds. In general, however, if data in a partitioned table comes from different databases, then the exchange operation might fail. For example, if the `jan2011` partition of `sales` did not come from the same staging database, then the preceding exchange operation can fail, returning the following error:

```
ORA-19728: data object number conflict between table JUL_SALES and partition JAN2011 in
table SALES
```

To resolve this conflict, move the offending partition by issuing the following statement:

```
ALTER TABLE sales MOVE PARTITION jan2011;
```

Then retry the exchange operation.

After the exchange succeeds, you can safely drop `jul_sales` and `jul_sale_index` (both are now empty). Thus you have successfully loaded the July sales data into your data warehouse.

13.1.2.2.3 Publishing Structured Data on CDs

Transportable tablespaces and transportable tables both provide a way to publish structured data on CDs.

You can copy the data to be published, including the data files and export dump file, to a CD. This CD can then be distributed. If you are using transportable tablespaces, then you must generate a transportable set before copying the data to the CD.

When customers receive this CD, they can add the CD contents to an existing database without having to copy the data files from the CD to disk storage. For example, suppose on a Microsoft Windows system D: drive is the CD drive. You can import the data in data file `catalog.f` and the export dump file `expdat.dmp` as follows:

```
impdp user_name/password DUMPFILE=expdat.dmp DIRECTORY=dpump_dir
   TRANSPORT_DATAFILES='D:\catalog.f'
```

You can remove the CD while the database is still up. Subsequent queries to the data return an error indicating that the database cannot open the data files on the CD. However, operations to other parts of the database are not affected. Placing the CD back into the drive makes the data readable again.

Removing the CD is the same as removing the data files of a read-only tablespace. If you shut down and restart the database, then the database indicates that it cannot find the removed data file and does not open the database (unless you set the initialization parameter `READ_ONLY_OPEN_DELAYED` to `TRUE`). When `READ_ONLY_OPEN_DELAYED` is set to `TRUE`, the database reads the file only when someone queries the data. Thus, when transporting data from a CD, set the `READ_ONLY_OPEN_DELAYED` initialization parameter to `TRUE`, unless the CD is permanently attached to the database.

13.1.2.2.4 Mounting the Same Tablespace Read-Only on Multiple Databases

You can use transportable tablespaces to mount a tablespace read-only on multiple databases.

In this way, separate databases can share the same data on disk instead of duplicating data on separate disks. The tablespace data files must be accessible by all databases. To avoid database corruption, the tablespace must remain read-only in all the databases mounting the tablespace, and the tablespace's data files must be read-only at the operating system level.

The following are two scenarios for mounting the same tablespace read-only on multiple databases:

- The tablespace originates in a database that is separate from the databases that will share the tablespace.

  You generate a transportable set in the source database, put the transportable set onto a disk that is accessible to all databases, and then import the metadata into each database on which you want to mount the tablespace.

- The tablespace already belongs to one of the databases that will share the tablespace.

  It is assumed that the data files are already on a shared disk. In the database where the tablespace already exists, you make the tablespace read-only, generate the transportable set, and then import the tablespace into the other databases, leaving the data files in the same location on the shared disk.

You can make a disk accessible by multiple computers in several ways. You can use either a cluster file system or raw disk. You can also use network file system (NFS), but be aware that if a user queries the shared tablespace while NFS is down, the database will stop responding until the NFS operation times out.

Later, you can drop the read-only tablespace in some of the databases. Doing so does not modify the data files for the tablespace. Thus, the drop operation does not corrupt the tablespace. Do not make the tablespace read/write unless only one database is mounting the tablespace.

13.1.2.2.5 Archiving Historical Data

When you use transportable tablespaces or transportable tables, the transported data is a self-contained set of files that can be imported into any Oracle database. Therefore, you can archive old or historical data in an enterprise data warehouse using the transportable tablespaces and transportable tables procedures.

> **See Also:**
>
> *Oracle Database Data Warehousing Guide* for more details

13.1.2.2.6 Using Transportable Tablespaces to Perform TSPITR

You can use transportable tablespaces to perform tablespace point-in-time recovery (TSPITR).

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide* for information about how to perform TSPITR using transportable tablespaces

13.1.2.2.7 Copying or Moving Individual Tables

You can use transportable tables to move a table or a set of tables from one database to another without transporting the entire tablespaces that contain the tables. You can also copy or move individual partitions and subpartitions from one database to another using transportable tables.

> **See Also:**
>
> "Transporting Tables, Partitions, or Subpartitions Between Databases"

13.1.3 Transporting Data Across Platforms

You can transport data across platforms.

The functionality of transporting data across platforms can be used to:

- Enable a database to be migrated from one platform to another.
- Provide an easier and more efficient means for content providers to publish structured data and distribute it to customers running Oracle Database on different platforms.
- Simplify the distribution of data from a data warehouse environment to data marts, which are often running on smaller platforms.
- Enable the sharing of read-only tablespaces between Oracle Database installations on different operating systems or platforms, assuming that your storage system is accessible from those platforms and the platforms all have the same endianness, as described in the sections that follow.

Many, but not all, platforms are supported for cross-platform data transport. You can query the `V$TRANSPORTABLE_PLATFORM` view to see the platforms that are supported, and to determine each platform's endian format (byte ordering). The following query displays the platforms that support cross-platform data transport:

```
COLUMN PLATFORM_NAME FORMAT A40
COLUMN ENDIAN_FORMAT A14
```

```
SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
  FROM V$TRANSPORTABLE_PLATFORM
  ORDER BY PLATFORM_ID;

PLATFORM_ID PLATFORM_NAME                              ENDIAN_FORMAT
----------- ---------------------------------------- --------------
          1 Solaris[tm] OE (32-bit)                  Big
          2 Solaris[tm] OE (64-bit)                  Big
          3 HP-UX (64-bit)                           Big
          4 HP-UX IA (64-bit)                        Big
          5 HP Tru64 UNIX                            Little
          6 AIX-Based Systems (64-bit)               Big
          7 Microsoft Windows IA (32-bit)            Little
          8 Microsoft Windows IA (64-bit)            Little
          9 IBM zSeries Based Linux                  Big
         10 Linux IA (32-bit)                        Little
         11 Linux IA (64-bit)                        Little
         12 Microsoft Windows x86 64-bit             Little
         13 Linux x86 64-bit                         Little
         15 HP Open VMS                              Little
         16 Apple Mac OS                             Big
         17 Solaris Operating System (x86)           Little
         18 IBM Power Based Linux                    Big
         19 HP IA Open VMS                           Little
         20 Solaris Operating System (x86-64)        Little
         21 Apple Mac OS (x86-64)                    Little
```

If the source platform and the target platform are of the same endianness, then the data is transported from the source platform to the target platform without any data conversion.

If the source platform and the target platform are of different endianness, then the data being transported must be converted to the target platform format. You can convert the data using one of the following methods:

- The `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package

  When you use one of these procedures to move data files between the source platform and the target platform, each block in each data file is converted to the target platform's endianness. The conversion occurs on the target platform.

- The RMAN `CONVERT` command

  Run the RMAN `CONVERT` command on the source or target platform. This command converts the data being transported to the target platform format.

> **Note:**
>
> Conversion of data files between different endian formats is not supported for data files having undo segments.

Before the data in a data file can be transported to a different platform, the data file header must identify the platform to which it belongs. To transport read-only tablespaces between Oracle Database installations on different platforms, make the data file read/write at least once.

> **See Also:**
>
> "Converting Data Between Platforms"

13.1.4 General Limitations on Transporting Data

There are general limitations on transporting data. There are also limitations that are specific to full transportable export/import, transportable tablespaces, or transportable tables.

Be aware of the following general limitations as you plan to transport data:

- The source and the target databases must use compatible database character sets. Specifically, one of the following must be true:
  - The database character sets of the source and the target databases are the same.
  - The source database character set is a strict (binary) subset of the target database character set, and the following three conditions are true:
    - The source database is Oracle Database 10g Release 1 (10.1.0.3) or later.
    - The tablespaces that you transport contain no table columns with character length semantics, or the maximum character width is the same in both the source and target database character sets.
    - The data that you transport contain no columns with the `CLOB` data type, or the source and the target database character sets are both single-byte or both multibyte.
  - The source database character set is a strict (binary) subset of the target database character set, and the following two conditions are true:
    - The source database is earlier than Oracle Database 10g Release 1 (10.1.0.3).
    - The maximum character width is the same in the source and target database character sets.

> **Note:**
>
> The subset-superset relationship between character sets recognized by Oracle Database is documented in *Oracle Database Globalization Support Guide*.

- The source and the target databases must use compatible national character sets. Specifically, one of the following must be true:
  - The national character sets of the source and target databases are the same.
  - The source database is Oracle Database 10g Release 1 (10.1.0.3) or later, and the tablespaces to be transported contain no columns with `NCHAR`, `NVARCHAR2`, or `NCLOB` data types.
- When running a transportable export operation, the following limitations apply:
  - The default tablespace of the user performing the export must not be one of the tablespaces being transported.
  - The default tablespace of the user performing the export must be writable.

- In a CDB, you cannot transport a tablespace to a target container that contains a tablespace of the same name. However, different containers can have tablespaces with the same name.

  You can use the `REMAP_TABLESPACE` import parameter to import the database objects into a different tablespace. Alternatively, before the transport operation, you can rename either the tablespace to be transported or the target tablespace.

- Transporting data with XMLTypes has the following limitations:
  - The target database must have XML DB installed.
  - Schemas referenced by XMLType tables cannot be the XML DB standard schemas.
  - If the schema for a transported XMLType table is not present in the target database, then it is imported and registered. If the schema already exists in the target database, then a message is displayed during import.
  - You must use only Oracle Data Pump to export and import the metadata for data that contains XMLTypes.

  The following query returns a list of tablespaces that contain XMLTypes:

  ```
  select distinct p.tablespace_name from dba_tablespaces p,
    dba_xml_tables x, dba_users u, all_all_tables t where
    t.table_name=x.table_name and t.tablespace_name=p.tablespace_name
    and x.owner=u.username;
  ```

  See *Oracle XML DB Developer's Guide* for information on XMLTypes.

- Types whose interpretation is application-specific and opaque to the database (such as `RAW`, `BFILE`, and AnyType can be transported, but they are not converted as part of the cross-platform transport operation. Their actual structure is known only to the application, so the application must address any endianness issues after these types are moved to the new platform. Types and objects that use these opaque types, either directly or indirectly, are also subject to this limitation.

- When you transport a tablespace containing tables with `TIMESTAMP WITH LOCAL TIME ZONE` (TSLTZ) data between databases with different time zones, the tables with the TSLTZ data are not transported. Error messages describe the tables that were not transported. However, tables in the tablespace that do not contain TSLTZ data are transported.

  You can determine the time zone of a database with the following query:

  ```
  SELECT DBTIMEZONE FROM DUAL;
  ```

  You can alter the time zone for a database with an `ALTER DATABASE` SQL statement.

  You can use Oracle Data Pump to perform a conventional export/import of tables with TSLTZ data after the transport operation completes.

- Analytic workspaces cannot be part of cross-platform transport operations. If the source platform and target platform are different, then use Data Pump export/import to export and import analytic workspaces. See *Oracle OLAP DML Reference* for more information about analytic workspaces.

> **Note:**
>
> Do not run the Oracle Data Pump export utility `expdp` or import utility `impdp` as `SYSDBA`, except at the request of Oracle technical support. `SYSDBA` is used internally and has specialized functions; its behavior is not the same as for general users.

Related Topics

- Limitations on Full Transportable Export/import
  There are limitations on full transportable export/import.
- Limitations on Transportable Tablespaces
  When you use transportable tablespaces, review the encryption, timezone, object, endianness, and other limitations that apply.
- Limitations on Transportable Tables
  There are limitations on transportable tables.
- Character Sets
- XMLType Data Type

13.1.5 Compatibility Considerations for Transporting Data

When transporting data, Oracle Database computes the lowest compatibility level at which the target database must run.

You can transport a tablespace or a table from a source database to a target database having the same or higher compatibility setting using transportable tablespaces, even if the target database is on the same or a different platform. The data transport operation fails if the compatibility level of the source database is higher than the compatibility level of the target database.

The following table shows the minimum compatibility requirements of the source and target databases in various scenarios. The source and target database need not have the same compatibility setting.

Table 13-2 Minimum Compatibility Requirements for Transport Scenarios

Transport Scenario	Source Minimum Database Compatibility	Target Minimum Database Compatibility
Transporting a database using full transportable export/import	12.0 (`COMPATIBLE` initialization parameter setting for an Oracle Database 12c or later database 12 (`VERSION` Oracle Data Pump export parameter setting for an 11.2.0.3 or later database)	12.0 (`COMPATIBLE` initialization parameter setting)
Transporting a tablespace between databases on the same platform using transportable tablespaces	8.0 (`COMPATIBLE` initialization parameter setting)	8.0 (`COMPATIBLE` initialization parameter setting)
Transporting a tablespace with different database block size than the target database using transportable tablespaces	9.0 (`COMPATIBLE` initialization parameter setting)	9.0 (`COMPATIBLE` initialization parameter setting)
Transporting a tablespace between databases on different platforms using transportable tablespaces	10.0 (`COMPATIBLE` initialization parameter setting)	10.0 (`COMPATIBLE` initialization parameter setting)
Transporting tables between databases	11.2.0 (`COMPATIBLE` initialization parameter setting for an Oracle Database 12c, or a later database	11.2.0 (`COMPATIBLE` initialization parameter setting)

Note:

- When you use full transportable export and import, the source database must be an Oracle Database 11g Release 2 (11.2.0.3) or later database, and the target database must be an Oracle Database 12c or later database.
- When transporting a database from Oracle Database 11g Release 2 (11.2.0.3) or later database to Oracle Database 12c or later database, you must set the Oracle Data Pump export parameter `VERSION` to `12` or higher.
- When transporting a database from an Oracle Database 19c database to an Oracle Database 19c database or later release, you must set the initialization parameter `COMPATIBLE` to `19.0.0` or higher.

13.2 Transporting Databases

You can transport a database to a new Oracle Database instance.

- Introduction to Full Transportable Export/Import
  You can use the full transportable export/import feature to copy an entire database from one Oracle Database instance to another.
- Limitations on Full Transportable Export/import
  There are limitations on full transportable export/import.
- Transporting a Database Using an Export Dump File
  You can transport a database using an export dump file.
- Transporting a Database Over the Network
  You can transport a database over the network.

13.2.1 Introduction to Full Transportable Export/Import

You can use the full transportable export/import feature to copy an entire database from one Oracle Database instance to another.

You can use Oracle Data Pump to produce an export dump file, transport the dump file to the target database if necessary, and then import the export dump file. Alternatively, you can use Oracle Data Pump to copy the database over the network.

The tablespaces in the database being transported can be either dictionary managed or locally managed. The tablespaces in the database are not required to be of the same block size as the target database standard block size.

Starting with Oracle Database Release 21c, you can use Oracle Data Pump to export databases to and import databases from the object store in Oracle Cloud. This simplifies migration to Oracle Cloud.

> **Note:**
>
> This method for transporting a database requires that you place the user-defined tablespaces in the database in read-only mode until you complete the export. If this is undesirable, then you can use the transportable tablespaces from backup feature described in *Oracle Database Backup and Recovery User's Guide*.

Related Topics

- Exporting Data from On Premises Databases to Oracle Autonomous Databases

> **See Also:**
>
> "About Transporting Data"

13.2.2 Limitations on Full Transportable Export/import

There are limitations on full transportable export/import.

Be aware of the following limitations on full transportable export/import:

- The general limitations described in "General Limitations on Transporting Data" apply to full transportable export/import.
- Full transportable export/import can export and import user-defined database objects in administrative tablespaces using conventional Data Pump export/import, such as direct path or external table. Administrative tablespaces are non-user tablespaces supplied with Oracle Database, such as the `SYSTEM` and `SYSAUX` tablespaces.
- Full transportable export/import cannot transport a database object that is defined in both an administrative tablespace (such as `SYSTEM` and `SYSAUX`) and a user-defined tablespace. For example, a partitioned table might be stored in both a user-defined tablespace and an administrative tablespace. If you have such database objects in your database, then you can redefine them before transporting them so that they are stored entirely in either an administrative tablespace or a user-defined tablespace. If the database objects cannot be redefined, then you can use conventional Data Pump export/import.
- When transporting a database over the network using full transportable export/import, auditing cannot be enabled for tables stored in an administrative tablespace (such as `SYSTEM` and `SYSAUX`) when the audit trail information itself is stored in a user-defined tablespace. See *Oracle Database Security Guide* for more information about auditing.
- Full transportable database import cannot be executed with Transaction Guard enabled. Disable Transaction Guard during a full database import.

Related Topics

- Introduction to Auditing

13.2.3 Transporting a Database Using an Export Dump File

You can transport a database using an export dump file.

The following list of tasks summarizes the process of transporting a database using an export dump file. Details for each task are provided in the subsequent example.

1. At the source database, configure each of the user-defined tablespaces in read-only mode and export the database.

   Ensure that the following parameters are set to the specified values:

   - `TRANSPORTABLE=ALWAYS`
   - `FULL=Y`

   If the source database is an Oracle Database 11g database (11.2.0.3 or later), then you must set the `VERSION` parameter to `12` or higher.

   If the source database contains any encrypted tablespaces or tablespaces containing tables with encrypted columns, then you must either specify `ENCRYPTION_PWD_PROMPT=YES`, or specify the `ENCRYPTION_PASSWORD` parameter.

   The export dump file includes the metadata for objects contained within the user-defined tablespaces and both the metadata and data for user-defined objects contained within the administrative tablespaces, such as `SYSTEM` and `SYSAUX`.

2. Transport the export dump file.

   Copy the export dump file to a place that is accessible to the target database.

3. Transport the data files for all of the user-defined tablespaces in the database.

   Copy the data files to a place that is accessible to the target database.

   If the source platform and target platform are different, then check the endian format of each platform by running the query on the `V$TRANSPORTABLE_PLATFORM` view.

   See "Transporting Data Across Platforms".

   If the source platform's endian format is different from the target platform's endian format, then use one of the following methods to convert the data files:

   - Use the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically.
   - Use the RMAN `CONVERT` command to convert the data files to the target platform's endian format.

     > **Note:**
     >
     > Conversion of data files between different endian formats is not supported for data files having undo segments.

   See "Converting Data Between Platforms" for more information.

4. (Optional) Restore the user-defined tablespaces to read/write mode on the source database.

5. At the target database, import the database.

When the import is complete, the user-defined tablespaces are in read/write mode.

Example

The tasks for transporting a database are illustrated in detail in this example. This example assumes that the source platform is Solaris and the target platform is Microsoft Windows.

It also assumes that the source platform has the following data files and tablespaces:

Tablespace	Type	Data File
sales	User-defined	/u01/app/oracle/oradata/mydb/sales01.dbf
customers	User-defined	/u01/app/oracle/oradata/mydb/cust01.dbf
employees	User-defined	/u01/app/oracle/oradata/mydb/emp01.dbf
SYSTEM	Administrative	/u01/app/oracle/oradata/mydb/system01.dbf
SYSAUX	Administrative	/u01/app/oracle/oradata/mydb/sysaux01.dbf

The following assumptions are made for this example:

- The target database is a new database that is being populated with the data from the source database. The name of the source database is `mydb`.
- Both the source database and the target database are Oracle Database 21c databases.

Complete the following tasks to transport the database using an export dump file:

Example 13-1 Task 1: Generate the Export Dump File

Generate the export dump file by completing the following steps:

1. Start SQL*Plus and connect to the database as an administrator or as a user who has either the `ALTER TABLESPACE` or `MANAGE TABLESPACE` system privilege.
2. Make all of the user-defined tablespaces in the database read-only.

```
ALTER TABLESPACE sales READ ONLY;

ALTER TABLESPACE customers READ ONLY;

ALTER TABLESPACE employees READ ONLY;
```

3. Start the Oracle Data Pump export utility as a user with `DATAPUMP_EXP_FULL_DATABASE` role, and specify the full transportable export/import options.

```
SQL> HOST

$ expdp user_name full=y dumpfile=expdat.dmp directory=data_pump_dir
        transportable=always logfile=export.log

Password: password
```

   You must always specify `TRANSPORTABLE=ALWAYS`, which determines whether the transportable option is used.

   This example specifies the following Oracle Data Pump parameters:

   - The `FULL` parameter specifies that the entire database is being exported.
   - The `DUMPFILE` parameter specifies the name of the structural information export dump file that will be created, `expdat.dmp`.

- The `DIRECTORY` parameter specifies the directory object that points to the operating system or Oracle Automatic Storage Management location of the dump file. You must create the `DIRECTORY` object before starting Oracle Data Pump, and you must grant the `READ` and `WRITE` object privileges on the directory to the user running the Export utility.

  See CREATE DIRECTORY in *Oracle Database SQL Language Reference* for information on the `CREATE DIRECTORY` command

  In a non-CDB, the directory object `DATA_PUMP_DIR` is created automatically. Read and write access to this directory is automatically granted to the `DBA` role, and thus to users `SYS` and `SYSTEM`.

  However, the directory object `DATA_PUMP_DIR` is not created automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB, and specify the directory object when you run Oracle Data Pump.

  > **See Also:**
  >
  > - Tracking Progress Within an Oracle Data Pump Job in *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
  > - Overview of Multitenant Administration in *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `LOGFILE` parameter specifies the file name of the log file that the export utility will write. In this example, the log file is written to the same directory as the dump file, but it can be written to a different location.

To perform a full transportable export on an Oracle Database 11g Release 2 (11.2.0.3) or later Oracle Database 11g database, use the `VERSION` parameter, as shown in the following example:

```
expdp user_name full=y dumpfile=expdat.dmp directory=data_pump_dir
      transportable=always version=12 logfile=export.log
```

Full transportable import is supported only for Oracle Database 12c and later databases.

> **Note:**
>
> In this example, the Oracle Data Pump utility is used to export only data dictionary structural information (metadata) for the user-defined tablespaces. Actual data is unloaded only for the administrative tablespaces (`SYSTEM` and `SYSAUX`), so this operation goes relatively quickly, even for large user-defined tablespaces.

4. Check the log file for errors, and take note of the dump file and data files that you must transport to the target database. `expdp` outputs the names and paths of these files in messages similar to these:

```
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /u01/app/oracle/admin/mydb/dpdump/expdat.dmp
******************************************************************************
Datafiles required for transportable tablespace SALES:
```

```
  /u01/app/oracle/oradata/mydb/sales01.dbf
Datafiles required for transportable tablespace CUSTOMERS:
  /u01/app/oracle/oradata/mydb/cust01.dbf
Datafiles required for transportable tablespace EMPLOYEES:
  /u01/app/oracle/oradata/mydb/emp01.dbf
```

5. When finished, exit back to SQL*Plus:

```
$ exit
```

> **See Also:**
>
> Oracle Data Pump Export in *Oracle Database Utilities*

Example 13-2 Task 2: Transport the Export Dump File

Transport the dump file to the directory pointed to by the `DATA_PUMP_DIR` directory object, or to any other directory of your choosing. The new location must be accessible to the target database.

At the target database, run the following query to determine the location of `DATA_PUMP_DIR`:

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';

OWNER      DIRECTORY_NAME   DIRECTORY_PATH
---------- ---------------- ---------------------------------
SYS        DATA_PUMP_DIR    C:\app\orauser\admin\orawin\dpdump\
```

Example 13-3 Task 3: Transport the Data Files for the User-Defined Tablespaces

Transport the data files of the user-defined tablespaces in the database to a place that is accessible to the target database.

In this example, transfer the following data files from the source database to the target database:

- `sales01.dbf`
- `cust01.dbf`
- `emp01.dbf`

If you are transporting the database to a platform different from the source platform, then determine if cross-platform database transport is supported for both the source and target platforms, and determine the endianness of each platform. If both platforms have the same endianness, then no conversion is necessary. Otherwise, you must do a conversion of each tablespace in the database, either at the source database or at the target database.

If you are transporting the database to a different platform, then you can run the following query on each platform. If the query returns a row, then the platform supports cross-platform tablespace transport.

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

The following is the query result from the source platform:

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Solaris[tm] OE (32-bit)          Big
```

The following is the query result from the target platform:

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Microsoft Windows IA (32-bit)    Little
```

In this example, you can see that the endian formats are different. Therefore, in this case, a conversion is necessary for transporting the database. Use either the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically. Transport the data files to the location of the existing data files of the target database. On the Unix and Linux platforms, this location is typically `/u01/app/oracle/oradata/`*`dbname`*`/`, or `+`*`DISKGROUP`*`/`*`dbname`*`/datafile/`. Alternatively, you can use the RMAN `CONVERT` command to convert the data files.

See "Converting Data Between Platforms" for more information.

> **Note:**
>
> If no endianness conversion of the tablespaces is needed, then you can transfer the files using any file transfer method.

Example 13-4 Task 4 (Optional) Restore Tablespaces to Read/Write Mode

Make the transported tablespaces read/write again at the source database, as follows:

```
ALTER TABLESPACE sales READ WRITE;
ALTER TABLESPACE customers READ WRITE;
ALTER TABLESPACE employees READ WRITE;
```

You can postpone this task to first ensure that the import process succeeds.

Start the Oracle Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role, and specify the full transportable export/import options.

```
impdp user_name full=Y dumpfile=expdat.dmp directory=data_pump_dir
   transport_datafiles=
      '/u01/app/oracle/oradata/mydb/sales01.dbf',
      '/u01/app/oracle/oradata/mydb/cust01.dbf',
      '/u01/app/oracle/oradata/mydb/emp01.dbf'
   logfile=import.log

Password: password
```

Example 13-5 Task 5: At the Target Database, Import the Database

This example specifies the following Oracle Data Pump parameters:

- The `FULL` parameter specifies that the entire database is being imported in `FULL` mode.
- The `DUMPFILE` parameter specifies the exported file containing the metadata for the user-defined tablespaces, and both the metadata and data for the administrative tablespaces that will be imported.

- The `DIRECTORY` parameter specifies the directory object that identifies the location of the export dump file. You must create the `DIRECTORY` object before starting Oracle Data Pump, and you must grant the `READ` and `WRITE` object privileges on the directory to the user running the Import utility.

  See CREATE DIRECTORY in *Oracle Database SQL Language Reference*.

  In a non-CDB, the directory object `DATA_PUMP_DIR` is created automatically. Read and write access to this directory is automatically granted to the `DBA` role, and thus to users `SYS` and `SYSTEM`.

  However, the directory object `DATA_PUMP_DIR` is not created automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB and specify the directory object when you run Data Pump.

  > **See Also:**
  >
  > - Tracking Progress Within an Oracle Data Pump Job in *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
  > - *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `TRANSPORT_DATAFILES` parameter identifies all of the data files that will be imported.

  If there are many files, then you can specify the `TRANSPORT_DATAFILES` parameter multiple times in a parameter file specified with the `PARFILE` parameter.

- The `LOGFILE` parameter specifies the file name of the log file that the import utility will write. In this example, the log file is written to the directory from which the dump file is read, but it can be written to a different location.

After this statement completes successfully, check the import log file to ensure that no unexpected error has occurred.

When dealing with a large number of data files, specifying the list of data file names in the statement line can be a laborious process. It can even exceed the statement line limit. In this situation, you can use an import parameter file. For example, you can start the Oracle Data Pump import utility as follows:

```
impdp user_name parfile='par.f'
```

For example, `par.f` might contain the following lines:

```
FULL=Y
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES=
'/u01/app/oracle/oradata/mydb/sales01.dbf',
'/u01/app/oracle/oradata/mydb/cust01.dbf',
'/u01/app/oracle/oradata/mydb/emp01.dbf'
LOGFILE=import.log
```

Note:

- During the import, user-defined tablespaces might be temporarily made read/write for metadata loading. Ensure that no user changes are made to the data during the import. At the successful completion of the import, all user-defined tablespaces are made read/write.
- When performing a network database import, the `TRANSPORTABLE` parameter must be set to `always`.
- When you are importing into a PDB in a CDB, specify the connect identifier for the PDB after the user name. For example, if the connect identifier for the PDB is `hrpdb`, then enter the following when you run the Oracle Data Pump Import utility:

  ```
  impdp user_name@hrpdb ...
  ```

See Also:

- Oracle Data Pump Import in *Oracle Database Utilities*
- *Oracle Multitenant Administrator's Guide*

13.2.4 Transporting a Database Over the Network

You can transport a database over the network.

To transport a database over the network, you perform an import using the `NETWORK_LINK` parameter, the import is performed using a database link, and there is no dump file involved.

The following list of tasks summarizes the process of transporting a database over the network. Details for each task are provided in the subsequent example.

1. Create a database link from the target database to the source database.

   The import operation must be performed by a user on the target database with `DATAPUMP_IMP_FULL_DATABASE` role, and the database link must connect to a user on the source database with `DATAPUMP_EXP_FULL_DATABASE` role. The user on the source database cannot be a user with `SYSDBA` administrative privilege. If the database link is a connected user database link, then the user on the target database cannot be a user with `SYSDBA` administrative privilege. See "Users of Database Links" for information about connected user database links.
2. In the source database, make the user-defined tablespaces in the database read-only.
3. Transport the data files for the all of the user-defined tablespaces in the database.

   Copy the data files to a place that is accessible to the target database.

   If the source platform and target platform are different, then check the endian format of each platform by running the query on the `V$TRANSPORTABLE_PLATFORM` view in "Transporting Data Across Platforms".

   If the source platform's endian format is different from the target platform's endian format, then use one of the following methods to convert the data files:

- Use the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically.
- Use the RMAN `CONVERT` command to convert the data files to the target platform's endian format.

> **Note:**
>
> Conversion of data files between different endian formats is not supported for data files having undo segments.

See "Converting Data Between Platforms" for more information.

4. At the target database, import the database.

   Invoke the Data Pump utility to import the metadata for the user-defined tablespaces and both the metadata and data for the administrative tablespaces.

   Ensure that the following parameters are set to the specified values:

   - `TRANSPORTABLE=ALWAYS`
   - `TRANSPORT_DATAFILES=`*`list_of_datafiles`*
   - `FULL=Y`
   - `NETWORK_LINK=`*`source_database_link`*

     Replace *`source_database_link`* with the name of the database link to the source database.
   - `VERSION=12`

     If the source database is an Oracle Database 11*g* Release 2 (11.2.0.3) or later Oracle Database 11*g* database, then the `VERSION` parameter is required and must be set to `12`. If the source database is an Oracle Database 12*c* or later database, then the `VERSION` parameter is not required.

   If the source database contains any encrypted tablespaces or tablespaces containing tables with encrypted columns, then you must either specify `ENCRYPTION_PWD_PROMPT=YES`, or specify the `ENCRYPTION_PASSWORD` parameter.

   The Data Pump network import copies the metadata for objects contained within the user-defined tablespaces and both the metadata and data for user-defined objects contained within the administrative tablespaces, such as `SYSTEM` and `SYSAUX`.

   When the import is complete, the user-defined tablespaces are in read/write mode.

5. (Optional) Restore the user-defined tablespaces to read/write mode on the source database.

Example

These tasks for transporting a database are illustrated more fully in the example that follows, where it is assumed the following data files and tablespaces exist:

Tablespace	Type	Data File
sales	User-defined	/u01/app/oracle/oradata/mydb/sales01.dbf
customers	User-defined	/u01/app/oracle/oradata/mydb/cust01.dbf

Tablespace	Type	Data File
employees	User-defined	/u01/app/oracle/oradata/mydb/emp01.dbf
SYSTEM	Administrative	/u01/app/oracle/oradata/mydb/system01.dbf
SYSAUX	Administrative	/u01/app/oracle/oradata/mydb/sysaux01.dbf

This example makes the following additional assumptions:

- The target database is a new database that is being populated with the data from the source database. The name of the source database is `sourcedb`.
- The source database and target database are running on the same platform with the same endianness.

  To check the endianness of a platform, run the following query:

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

- The `sales` tablespace is encrypted. The other tablespaces are not encrypted.
- The source database is an Oracle Database 11*g* Release 2 (11.2.0.3) database and the target database is an Oracle Database 19c database.

> **Note:**
>
> This example illustrates the tasks required to transport an Oracle Database 11*g* Release 2 (11.2.0.3) to a new Oracle Database 19c PDB inside a CDB.

> **See Also:**
>
> *Oracle Multitenant Administrator's Guide*

Complete the following tasks to transport the database over the network:

Task 1 Create a Database Link from the Target Database to the Source Database
Create a database link from the target database to the source database by completing the following steps:

1. Ensure that network connectivity is configured between the source database and the target database.

   See *Oracle Database Net Services Administrator's Guide* for instructions.
2. Start SQL*Plus and connect to the target database as the administrator who will transport the database with Data Pump import. This user must have `DATAPUMP_IMP_FULL_DATABASE` role to transport the database.

   See "Connecting to the Database with SQL*Plus" for instructions.
3. Create the database link:

```
CREATE PUBLIC DATABASE LINK sourcedb USING 'sourcedb';
```

Specify the service name for the source database in the using clause.

During the import operation, the database link must connect to a user on the source database with `DATAPUMP_EXP_FULL_DATABASE` role. The user on the source database cannot be a user with `SYSDBA` administrative privilege.

> **See Also:**
>
> - "Creating Database Links"
> - *Oracle Database SQL Language Reference*

Task 2 Make the User-Defined Tablespaces Read-Only
Complete the following steps:

1. Start SQL*Plus and connect to the source database as an administrator or as a user who has either the `ALTER TABLESPACE` or `MANAGE TABLESPACE` system privilege.

   See "Connecting to the Database with SQL*Plus" for instructions.

2. Make all of the user-defined tablespaces in the database read-only.

```
ALTER TABLESPACE sales READ ONLY;

ALTER TABLESPACE customers READ ONLY;

ALTER TABLESPACE employees READ ONLY;
```

Task 3 Transport the Data Files for the User-Defined Tablespaces
Transport the data files to the location of the existing data files of the target database.
On the UNIX and Linux platforms, this location is typically /u01/app/oracle/oradata/*dbname*/ or +*DISKGROUP*/*dbname*/datafile/.
In this example, transfer the following data files from the source database to the target database:

- `sales01.dbf`
- `cust01.dbf`
- `emp01.dbf`

> **See Also:**
>
> "Guidelines for Transferring Data Files"

Task 4 At the Target Database, Import the Database
Invoke the Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role and specify the full transportable export/import options.

```
impdp user_name full=Y network_link=sourcedb transportable=always
   transport_datafiles=
      '/u01/app/oracle/oradata/mydb/sales01.dbf',
      '/u01/app/oracle/oradata/mydb/cust01.dbf',
```

```
    '/u01/app/oracle/oradata/mydb/emp01.dbf'
  encryption_pwd_prompt=YES version=12 logfile=import.log

Password: password
```

This example specifies the following Data Pump parameters:

- The `FULL` parameter specifies that the entire database is being imported in `FULL` mode.
- The `NETWORK_LINK` parameter specifies the database link used for the network import.
- The `TRANSPORTABLE` parameter specifies that the import uses the transportable option.
- The `TRANSPORT_DATAFILES` parameter identifies all of the data files to be imported.

  You can specify the `TRANSPORT_DATAFILES` parameter multiple times in a parameter file specified with the `PARFILE` parameter if there are many data files.
- The `ENCRYPTION_PWD_PROMPT` parameter instructs Data Pump to prompt you for the encryption password, and Data Pump encrypts data and metadata sent over the network connection. Either the `ENCRYPTION_PWD_PROMPT` parameter or the `ENCRYPTION_PASSWORD` parameter is required when encrypted tablespaces or tables with encrypted columns are part of the import operation.
- The `VERSION` parameter is set to `12` because the source database is an Oracle Database 11*g* Release 2 (11.2.0.3) or later Oracle Database 11*g* database.
- The `LOGFILE` parameter specifies the file name of the log file to be written by the import utility.

After this statement executes successfully, check the import log file to ensure that no unexpected error has occurred.
When dealing with a large number of data files, specifying the list of data file names in the statement line can be a laborious process. It can even exceed the statement line limit. In this situation, you can use an import parameter file.
Use of an import parameter file is also recommended when encrypted tablespaces or tables with encrypted columns are part of the import operation. In this case, specify `ENCRYPTION_PWD_PROMPT=YES` in the import parameter file.
For example, you can invoke the Data Pump import utility as follows:

```
impdp user_name parfile='par.f'
```

For example, `par.f` might contain the following lines:

```
FULL=Y
NETWORK_LINK=sourcedb
TRANSPORTABLE=always
TRANSPORT_DATAFILES=
'/u01/app/oracle/oradata/mydb/sales01.dbf',
'/u01/app/oracle/oradata/mydb/cust01.dbf',
'/u01/app/oracle/oradata/mydb/emp01.dbf'
ENCRYPTION_PWD_PROMPT=YES
VERSION=12
LOGFILE=import.log
```

> **Note:**
>
> - During the import, user-defined tablespaces might be temporarily made read/write for metadata loading. Ensure that no user changes are made to the data during the import. At the successful completion of the import, all user-defined tablespaces are made read/write.
> - When you are importing into a PDB in a CDB, specify the connect identifier for the PDB after the user name. For example, if the connect identifier for the PDB is `hrpdb`, then enter the following when you run the Oracle Data Pump Import utility:
>
>   ```
>   impdp user_name@hrpdb ...
>   ```

> **See Also:**
>
> *Oracle Database Utilities* for information about using the import utility

Task 5 (Optional) Restore User-Defined Tablespaces to Read/Write Mode
Make the user-defined tablespaces read/write again at the source database, as follows:

```
ALTER TABLESPACE sales READ WRITE;

ALTER TABLESPACE customers READ WRITE;

ALTER TABLESPACE employees READ WRITE;
```

You can postpone this task to first ensure that the import process succeeds.

13.3 Transporting Tablespaces Between Databases

You can transport tablespaces between databases.

> **Note:**
>
> To import a transportable tablespace set into an Oracle database on a different platform, both databases must have compatibility set to at least 10.0.0. See "Compatibility Considerations for Transporting Data" for a discussion of database compatibility for transporting tablespaces across release levels.

- Introduction to Transportable Tablespaces
  You can use the transportable tablespaces feature to copy a set of tablespaces from one Oracle Database to another.
- Limitations on Transportable Tablespaces
  When you use transportable tablespaces, review the encryption, timezone, object, endianness, and other limitations that apply.
- Transporting Tablespaces Between Databases
  You can transport a tablespace or a set of tablespaces between databases.

13.3.1 Introduction to Transportable Tablespaces

You can use the transportable tablespaces feature to copy a set of tablespaces from one Oracle Database to another.

The tablespaces being transported can be either dictionary managed or locally managed. The transported tablespaces are not required to be of the same block size as the target database standard block size. These scenarios are discussed in "Transporting Data: Scenarios".

There are two ways to transport a tablespace:

- Manually, following the steps described in this section. This involves issuing commands to SQL*Plus and Data Pump.
- Using the Transport Tablespaces Wizard in Oracle Enterprise Manager Cloud Control

  To run the Transport Tablespaces Wizard:

  1. Log in to Cloud Control with a user that has the `DATAPUMP_EXP_FULL_DATABASE` role.
  2. Access the Database Home page.
  3. From the Schema menu, select **Database Export/Import**, then **Transport Tablespaces**.

> **Note:**
>
> - This method for transporting tablespaces requires that you place the tablespaces to be transported in read-only mode until you complete the transporting process. If this is undesirable, you can use the transportable tablespaces from backup feature, described in *Oracle Database Backup and Recovery User's Guide*.
> - You must use Data Pump for transportable tablespaces. The only circumstance under which you can use the original import utility, IMP, is for a backward migration of XMLType data to an Oracle Database 10*g* Release 2 (10.2) or earlier database. See *Oracle Database Utilities* for more information on these utilities and to *Oracle XML DB Developer's Guide* for more information on XMLTypes.

> **See Also:**
>
> - "About Transporting Data"
> - *Oracle Database Data Warehousing Guide* for information about using transportable tablespaces in a data warehousing environment

13.3.2 Limitations on Transportable Tablespaces

When you use transportable tablespaces, review the encryption, timezone, object, endianness, and other limitations that apply.

Be aware of the following limitations for transportable tablespaces:

- The general limitations described in "General Limitations on Transporting Data" apply to transportable tablespaces.
- When transporting a tablespace set, objects with underlying objects (such as materialized views) or contained objects (such as partitioned tables) are not transportable unless all of the underlying or contained objects are in the tablespace set.
- Transportable tablespaces cannot transport tables with `TIMESTAMP WITH TIMEZONE` (TSTZ) data across platforms with different time zone file versions. The transportable tablespace operation skips these tables. You can export and import these tables conventionally.
- You cannot include administrative tablespaces, such as `SYSTEM` and `SYSAUX` in a transportable tablespace set.
- Transportable tablespaces cannot contain any tables with columns that are encrypted using TDE column encryption
- If a tablespace is encrypted using TDE, then you can only transport this tablespace to a platform that uses the same endian format. If you need to transport the tablespace across endianness, then you must decrypt, transport, and re-encrypt the tablespace. Starting with Oracle Database Release 12.2, these operations can be performed online.
- Transportable tablespace sets include TDE policies only. Other security policies (such as redaction policies, masking policies, and so on) are not included; they must be recreated after the tablespace is imported into the new database.

Related Topics

- How Does Oracle Data Pump Handle Timestamp Data?

13.3.3 Transporting Tablespaces Between Databases

You can transport a tablespace or a set of tablespaces between databases.

Starting with Oracle Database Release 21c, transportable jobs can be restarted at or near the point of failure.

The following list of tasks summarizes the process of transporting a tablespace. Details for each task are provided in the subsequent example.

1. Pick a self-contained set of tablespaces.
2. At the source database, configure the set of tablespaces in read-only mode and generate a transportable tablespace set.

   A transportable tablespace set (or transportable set) consists of data files for the set of tablespaces being transported and an export dump file containing structural information (metadata) for the set of tablespaces. You use Oracle Data Pump to perform the export.
3. Transport the export dump file.

   Copy the export dump file to a place that is accessible to the target database.
4. Transport the tablespace set.

   Copy the data files to a directory that is accessible to the target database.

   If the source platform and target platform are different, then check the endian format of each platform by running the query on the `V$TRANSPORTABLE_PLATFORM` view.

   If the source platform's endian format is different from the target platform's endian format, then use one of the following methods to convert the data files:

- Use the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically.
- Use the RMAN `CONVERT` command to convert the data files to the target platform's endian format.

> **Note:**
>
> Conversion of data files between different endian formats is not supported for data files having undo segments.

5. (Optional) Restore tablespaces to read/write mode on the source database.
6. At the target database, import the tablespace set.

   Run the Oracle Data Pump utility to import the metadata for the tablespace set.

Example 13-6 Example

These tasks for transporting a tablespace are illustrated more fully in the example that follows, where it is assumed the following data files and tablespaces exist:

Tablespace	Data File
`sales_1`	`/u01/app/oracle/oradata/salesdb/sales_101.dbf`
`sales_2`	`/u01/app/oracle/oradata/salesdb/sales_201.dbf`

- Task 1: Pick a Self-Contained Set of Tablespaces
  See how to determine if the tablespaces you want to use are self-contained, or have dependencies.
- Task 2: Generate a Transportable Tablespace Set
  After ensuring that you have a self-contained set of tablespaces that you want to transport, generate a transportable tablespace set.
- Task 3: Transport the Export Dump File
  Transport the dump file to the directory pointed to by the `DATA_PUMP_DIR` directory object, or to any other directory of your choosing.
- Task 4: Transport the Tablespace Set
  Transport the data files of the tablespaces to a directory that is accessible to the target database.
- Task 5: (Optional) Restore Tablespaces to Read/Write Mode
  Make the transported tablespaces read/write again at the source database.
- Task 6: Import the Tablespace Set
  To complete the transportable tablespaces operation, import the tablespace set.

Related Topics

- Transporting Data Across Platforms
  You can transport data across platforms.
- Converting Data Between Platforms
  To transport tables across platforms, check platform endianness, and review other restrictions.

13.3.3.1 Task 1: Pick a Self-Contained Set of Tablespaces

See how to determine if the tablespaces you want to use are self-contained, or have dependencies.

There may be logical or physical dependencies between the database objects in the transportable set, and the database objects outside of the transportable set. You can only transport a tablespace set that is **self-contained**, that is, none of the database objects inside a tablespace set are dependent on any of the database objects outside of that tablespace set.

Some examples of self-contained tablespace violations are:

- An index inside the set of tablespaces is for a table outside of the set of tablespaces.

  > **Note:**
  >
  > It is not a violation if a corresponding index for a table is outside of the set of tablespaces.

- A partitioned table is partially contained in the set of tablespaces.

  The tablespace set that you want to copy must contain either all partitions of a partitioned table, or none of the partitions of a partitioned table. To transport a subset of a partition table, you must exchange the partitions into tables.

  See *Oracle Database VLDB and Partitioning Guide* for information about exchanging partitions.

- A referential integrity constraint points to a table across a set boundary.

  When transporting a set of tablespaces, you can choose to include referential integrity constraints. However, doing so can affect whether a set of tablespaces is self-contained. If you decide not to transport constraints, then the constraints are not considered as pointers.

- A table inside the set of tablespaces contains a `LOB` column that points to `LOB`s outside the set of tablespaces.

- An XML DB schema (`*.xsd`) that was registered by user A imports a global schema that was registered by user B, and the following is true: the default tablespace for user A is tablespace A, the default tablespace for user B is tablespace B, and only tablespace A is included in the set of tablespaces.

To determine whether a set of tablespaces is self-contained, run the `TRANSPORT_SET_CHECK` procedure in the Oracle-supplied package `DBMS_TTS`. To run this procedure, you must have been granted the `EXECUTE_CATALOG_ROLE` role (initially signed to `SYS`).

When you run the `DBMS_TTS.TRANSPORT_SET_CHECK` procedure, specify the list of tablespaces in the transportable set to be checked for self containment. You can optionally specify if constraints must be included. For strict or full containment, you must additionally set the `TTS_FULL_CHECK` parameter to `TRUE`.

The strict or full containment check is for cases that require capturing not only references going outside the transportable set, but also those coming into the set. Tablespace Point-in-Time Recovery (TSPITR) is one such case where dependent objects must be fully contained or fully outside the transportable set.

For example, it is a violation to perform TSPITR on a tablespace containing a table `t`, but not its index `i`, because the index and data will be inconsistent after the transport. A full

containment check ensures that there are no dependencies going outside or coming into the transportable set. See the example for TSPITR in the *Oracle Database Backup and Recovery User's Guide*.

> **Note:**
>
> The default for transportable tablespaces is to check for self-containment, rather than full containment.

Example 13-7 Determining Whether Tablespaces are Self-Contained

The following statement can be used to determine whether tablespaces `sales_1` and `sales_2` are self-contained, with referential integrity constraints taken into consideration (indicated by `TRUE`).

```
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK('sales_1,sales_2', TRUE);
```

After running the `DBMS_TTS.TRANSPORT_SET_CHECK` procedure, you can see all the violations by selecting from the `TRANSPORT_SET_VIOLATIONS` view. If the set of tablespaces is self-contained, then this view is empty. The following example illustrates a case where there are two violations: a foreign key constraint, `dept_fk`, across the tablespace set boundary, and a partitioned table, `jim.sales`, that is partially contained in the tablespace set.

```
SELECT * FROM TRANSPORT_SET_VIOLATIONS;

VIOLATIONS
---------------------------------------------------------------------------
Constraint DEPT_FK between table JIM.EMP in tablespace SALES_1 and table
JIM.DEPT in tablespace OTHER
Partitioned table JIM.SALES is partially contained in the transportable set
```

You must resolve these violations before `sales_1` and `sales_2` are transportable. One choice for bypassing the integrity constraint violation is to not to export the integrity constraints.

Related Topics

- DBMS_TTS in *Oracle Database PL/SQL Packages and Types Reference*
- Performing Fully Automated RMAN TSPITR in *Oracle Database Backup and Recovery User's Guide*

13.3.3.2 Task 2: Generate a Transportable Tablespace Set

After ensuring that you have a self-contained set of tablespaces that you want to transport, generate a transportable tablespace set.

To generate a transportable tablespace set:

1. Start SQL*Plus and connect to the database as an administrator or as a user who has either the `ALTER TABLESPACE` or `MANAGE TABLESPACE` system privilege.
2. Make all tablespaces in the set read-only.

   ```
   ALTER TABLESPACE sales_1 READ ONLY;

   ALTER TABLESPACE sales_2 READ ONLY;
   ```

3. Run the Oracle Data Pump export utility as a user with `DATAPUMP_EXP_FULL_DATABASE` role, and specify the tablespaces in the transportable set.

```
> expdp user_name PARFILE=my-export.par

Password: password
```

The contents of the `my-export.par` file are as follows:

```
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_TABLESPACES=sales_1,sales_2
LOGFILE=tts_export.log
EXCLUDE=TABLE_STATISTICS,INDEX_STATISTICS
```

> **Caution:**
>
> Never use `AS SYSDBA` When you are importing into Oracle Database 12c Release 1 (12.1) or higher, use `LOGTIME=ALL` and `METRICS=Y`.

To perform a transport tablespace operation with a strict containment check, use the `TRANSPORT_FULL_CHECK` parameter, as shown in the following example:

```
expdp user_name dumpfile=expdat.dmp directory=data_pump_dir
      transport_tablespaces=sales_1,sales_2 transport_full_check=y
      logfile=tts_export.log
```

In this case, the Oracle Data Pump export utility verifies that there are no dependencies between the objects inside the transportable set and objects outside the transportable set. If the tablespace set being transported is not self-contained, then the export fails and indicates that the transportable set is not self-contained. You must resolve these violations and then run this task again.

> **Note:**
>
> In this example, the Oracle Data Pump utility is used to export only data dictionary structural information (metadata) for the tablespaces. No actual data is unloaded, so this operation goes relatively quickly even for large tablespace sets.

4. The `expdp` utility displays the names and paths of the dump file and the data files on the command line as shown in the following example. These are the files that you need to transport to the target database. Also, check the log file for any errors.

```
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /u01/app/oracle/admin/salesdb/dpdump/expdat.dmp
******************************************************************************
Datafiles required for transportable tablespace SALES_1:
  /u01/app/oracle/oradata/salesdb/sales_101.dbf
Datafiles required for transportable tablespace SALES_2:
  /u01/app/oracle/oradata/salesdb/sales_201.dbf
```

5. When the Oracle Data Pump export operation is completed, exit the `expdp` utility to return to SQL*Plus:

```
$ EXIT
```

> **See Also:**
>
> - Tracking Progress Within an Oracle Data Pump Job in *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
> - Oracle Data Pump Export in *Oracle Database Utilities* for information about using the Oracle Data Pump utility

13.3.3.3 Task 3: Transport the Export Dump File

Transport the dump file to the directory pointed to by the `DATA_PUMP_DIR` directory object, or to any other directory of your choosing.

The new location must be accessible to the target database.

At the target database, run the following query to determine the location of `DATA_PUMP_DIR`:

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';

OWNER      DIRECTORY_NAME   DIRECTORY_PATH
---------- ---------------- ----------------------------------
SYS        DATA_PUMP_DIR    C:\app\orauser\admin\orawin\dpdump\
```

13.3.3.4 Task 4: Transport the Tablespace Set

Transport the data files of the tablespaces to a directory that is accessible to the target database.

In this example, transfer the following files from the source database to the target database:

- `sales_101.dbf`
- `sales_201.dbf`

If you are transporting the tablespace set to a platform different from the source platform, then determine if cross-platform tablespace transport is supported for both the source and target platforms, and determine the endianness of each platform. If both platforms have the same endianness, then no conversion is necessary. If the platform endianness is different, then you must convert the data, either at the source database or at the target database.

If you are transporting `sales_1` and `sales_2` to a different platform, then you can run the following query on each platform. If the query returns a row, then the platform supports cross-platform tablespace transport.

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

The following is the query result from the source platform:

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Solaris[tm] OE (32-bit)          Big
```

The following is the result from the target platform:

```
PLATFORM_NAME                    ENDIAN_FORMAT
-------------------------------- --------------
Microsoft Windows IA (32-bit)    Little
```

In this example, you can see that the source and target platform endian formats are different. Therefore, in this case, a conversion is necessary for transporting the database. To transfer the data files, use either the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package. These procedures convert the data files to the target platform's endian format automatically. Transport the data files to the location of the existing data files of the target database. On Unix and Linux platforms, this location is typically `/u01/app/oracle/oradata/`*`dbname`*`/` or `+`*`DISKGROUP`*`/`*`dbname`*`/datafile/`. Alternatively, you can use to convert the data files.

> **Note:**
>
> - If you use the RMAN `CONVERT` command, then conversion of data files between different endian formats is not supported for data files having undo segments.
> - If no endianness conversion of the tablespaces is needed, then you can transfer the files using any file transfer method.

Related Topics

- Converting Data Between Platforms
  To transport tables across platforms, check platform endianness, and review other restrictions.
- Guidelines for Transferring Data Files
  You should follow a set of guidelines when transferring the data files.

13.3.3.5 Task 5: (Optional) Restore Tablespaces to Read/Write Mode

Make the transported tablespaces read/write again at the source database.

The following statements make the `sales_1` and `sales_2` tablespaces read/write:

```
ALTER TABLESPACE sales_1 READ WRITE;
ALTER TABLESPACE sales_2 READ WRITE;
```

You can postpone this task to first ensure that the import process succeeds.

13.3.3.6 Task 6: Import the Tablespace Set

To complete the transportable tablespaces operation, import the tablespace set.

To import the tablespace set:

1. Run the Oracle Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role and import the tablespace metadata.

   ```
   impdp user_name dumpfile=expdat.dmp directory=data_pump_dir
      transport_datafiles=
   ```

```
    'c:\app\orauser\oradata\orawin\sales_101.dbf',
    'c:\app\orauser\oradata\orawin\sales_201.dbf'
    remap_schema=sales1:crm1  remap_schema=sales2:crm2
    logfile=tts_import.log

Password: password
```

This example specifies the following Data Pump parameters:

- The `DUMPFILE` parameter specifies the exported file containing the metadata for the tablespaces to be imported.
- The `DIRECTORY` parameter specifies the directory object that identifies the location of the export dump file. You must create the `DIRECTORY` object before running Data Pump, and you must grant the `READ` and `WRITE` object privileges on the directory to the user running the Import utility.

> **Caution:**
>
> Never use `AS SYSDBA` When you are importing into Oracle Database 12c Release 1 (12.1) or higher, use `LOGTIME=ALL` and `METRICS=Y`.

> **See Also:**
>
> CREATE DIRECTORY for information on the `CREATE DORECTORY` command

However, the database does not create the directory object `DATA_PUMP_DIR` automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB and specify the directory object when you run Oracle Data Pump.

> **See Also:**
>
> - Tracking Progress Within an Oracle Data Pump Job in *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
> - *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `TRANSPORT_DATAFILES` parameter identifies all of the data files containing the tablespaces that you want to import.

  If there are many data files, then you can specify the `TRANSPORT_DATAFILES` parameter multiple times in a parameter file specified with the `PARFILE` parameter.
- The `REMAP_SCHEMA` parameter changes the ownership of database objects. If you do not specify `REMAP_SCHEMA`, then all database objects (such as tables and indexes) are created in the same user schema as in the source database, and those users must already exist in the target database. If they do not exist, then the import utility returns an error. In this example, objects in the tablespace set owned by `sales1` in the source database will be owned by `crm1` in the target database after the tablespace set is

imported. Similarly, objects owned by `sales2` in the source database will be owned by `crm2` in the target database. In this case, the target database is not required to have users `sales1` and `sales2`, but must have users `crm1` and `crm2`.

Starting with Oracle Database 12c Release 2 (12.2), Recovery Manager (RMAN) `RECOVER` command can move tables to a different schema while remapping a table.

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide*

- The `LOGFILE` parameter specifies the file name of the log file to be written by the import utility. In this example, the log file is written to the directory from which the dump file is read, but it can be written to a different location.

After this statement runs successfully, all tablespaces in the set being copied remain in read-only mode. Check the import log file to ensure that no error has occurred.

When dealing with a large number of data files, specifying the list of data file names in the statement line can be a laborious process, as the data file list can even exceed the statement line limit. In this situation, you can use an import parameter file. For example, you can run the Oracle Data Pump import utility as follows:

```
impdp user_name parfile='par.f'
```

The `par.f` parameter file contains the following:

```
DUMPFILE=expdat.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES=
'C:\app\orauser\oradata\orawin\sales_101.dbf',
'C:\app\orauser\oradata\orawin\sales_201.dbf'
REMAP_SCHEMA=sales1:crm1  REMAP_SCHEMA=sales2:crm2
LOGFILE=tts_import.log
```

> **See Also:**
>
> Oracle Data Pump Import in *Oracle Database Utilities*

2. If required, put the tablespaces into read/write mode on the target database.

13.4 Transporting Tables, Partitions, or Subpartitions Between Databases

You can transport tables, partitions, and subpartitions between databases.

- Introduction to Transportable Tables
  You can use the transportable tables feature to copy a set of tables, partitions, or subpartitions from one Oracle Database to another. A transportable tables operation moves metadata for the specified tables, partitions, or subpartitions to the target database.
- Limitations on Transportable Tables
  There are limitations on transportable tables.

- Transporting Tables, Partitions, or Subpartitions Using an Export Dump File
  You can transport tables, partitions, or subpartitions between databases using an export file.
- Transporting Tables, Partitions, or Subpartitions Over the Network
  To transport tables over the network, you perform an import using the `NETWORK_LINK` parameter, the import is performed using a database link, and there is no dump file involved.

13.4.1 Introduction to Transportable Tables

You can use the transportable tables feature to copy a set of tables, partitions, or subpartitions from one Oracle Database to another. A transportable tables operation moves metadata for the specified tables, partitions, or subpartitions to the target database.

A transportable tables operation automatically identifies the tablespaces used by the specified tables. To move the data, you copy the data files for these tablespaces to the target database. The Data Pump import automatically frees the blocks in the data files occupied by tables, partitions, or subpartitions that were not part of the transportable tables operation. It also frees the blocks occupied by the dependent objects of the tables that were not part of the transportable tables operation.

You can transport the tables, partitions, and subpartitions in the following ways:

- Using an export dump file

  During the export, specify the `TABLES` parameter and set the `TRANSPORTABLE` parameter to `ALWAYS`. During import, do not specify the `TRANSPORTABLE` parameter. Data Pump import recognizes the transportable tables operation automatically.
- Over the network

  During the import, specify the `TABLES` parameter, set the `TRANSPORTABLE` parameter to `ALWAYS`, and specify the `NETWORK_LINK` parameter to identify the source database.

13.4.2 Limitations on Transportable Tables

There are limitations on transportable tables.

Be aware of the following limitations for transportable tables:

- The general limitations described in "General Limitations on Transporting Data" apply to transportable tables.
- You cannot transport a table to a target database that contains a table of the same name in the same schema. However, you can use the `REMAP_TABLE` import parameter to import the data into a different table. Alternatively, before the transport operation, you can rename either the table to be transported or the target table.

  Starting with Oracle Database 12*c* Release 2 (12.2), the Recovery Manager (RMAN) `RECOVER` command can move tables to a different schema while remapping a table. See *Oracle Database Backup and Recovery User's Guide* for more information.
- You cannot transport tables with `TIMESTAMP WITH TIMEZONE` (TSTZ) data across platforms with different time zone file versions.

  See *Oracle Database Utilities* for more information.

13.4.3 Transporting Tables, Partitions, or Subpartitions Using an Export Dump File

You can transport tables, partitions, or subpartitions between databases using an export file.

The following list of tasks summarizes the process of transporting tables between databases using an export dump file. Details for each task are provided in the subsequent example.

1. Pick a set of tables, partitions, or subpartitions.

   If you are transporting partitions, then you can specify partitions from only one table in a transportable tables operation, and no other tables can be transported in the same operation. Also, if only a subset of a table's partitions are exported in a transportable tables operation, then on import each partition becomes a non-partitioned table.

2. At the source database, place the tablespaces associated with the data files for the tables, partitions, or subpartitions in read-only mode.

   To view the tablespace for a table, query the `DBA_TABLES` view. To view the data file for a tablespace, query the `DBA_DATA_FILES` view.

3. Perform the Data Pump export.

4. Transport the export dump file.

   Copy the export dump file to a place that is accessible to the target database.

5. Transport the data files for the tables, partitions, or subpartitions.

   Copy the data files to a place that is accessible to the target database.

   If the source platform and target platform are different, then check the endian format of each platform by running the query on the `V$TRANSPORTABLE_PLATFORM` view in "Transporting Data Across Platforms".

   If the source platform's endian format is different from the target platform's endian format, then use one of the following methods to convert the data files:

   - Use the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically.
   - Use the RMAN `CONVERT` command to convert the data files to the target platform's endian format.

   > **Note:**
   >
   > Conversion of data files between different endian formats is not supported for data files having undo segments.

   See "Converting Data Between Platforms" for more information.

6. (Optional) Restore tablespaces to read/write mode on the source database.

7. At the target database, perform the import.

   Invoke the Data Pump utility to import the metadata for the tables.

Example

These tasks for transporting tables, partitions, and subpartitions using a Data Pump dump file are illustrated more fully in the example that follows, where it is assumed that the following partitions exist in the `sh.sales_prt` table:

- `sales_q1_2000`
- `sales_q2_2000`
- `sales_q3_2000`
- `sales_q4_2000`

This example transports two of these partitions to the target database.

The following SQL statements create the `sales_prt` table and its and partitions in the `sh` schema and the tablespace and data file for the table. The statements also insert data into the partitions by using data in the `sh` sample schemas.

```
CREATE TABLESPACE sales_prt_tbs
   DATAFILE 'sales_prt.dbf' SIZE 20M
   ONLINE;

CREATE TABLE sh.sales_prt
    (prod_id        NUMBER(6),
     cust_id        NUMBER,
     time_id        DATE,
     channel_id     CHAR(1),
     promo_id       NUMBER(6),
     quantity_sold  NUMBER(3),
     amount_sold    NUMBER(10,2))
        PARTITION BY RANGE (time_id)
         (PARTITION SALES_Q1_2000 VALUES LESS THAN
                  (TO_DATE('01-APR-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
          PARTITION SALES_Q2_2000 VALUES LESS THAN
                  (TO_DATE('01-JUL-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
          PARTITION SALES_Q3_2000 VALUES LESS THAN
                  (TO_DATE('01-OCT-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
          PARTITION SALES_Q4_2000 VALUES LESS THAN
                  (TO_DATE('01-JAN-2001','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')))
 TABLESPACE sales_prt_tbs;

INSERT INTO sh.sales_prt PARTITION(sales_q1_2000)
  SELECT * FROM sh.sales PARTITION(sales_q1_2000);

INSERT INTO sh.sales_prt PARTITION(sales_q2_2000)
  SELECT * FROM sh.sales PARTITION(sales_q2_2000);

INSERT INTO sh.sales_prt PARTITION(sales_q3_2000)
  SELECT * FROM sh.sales PARTITION(sales_q3_2000);

INSERT INTO sh.sales_prt PARTITION(sales_q4_2000)
  SELECT * FROM sh.sales PARTITION(sales_q4_2000);

COMMIT;
```

This example makes the following additional assumptions:

- The name of the source database is `sourcedb`.

- The source database and target database are running on the same platform with the same endianness. To check the endianness of a platform, run the following query:

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

- Only the `sales_q1_2000` and `sales_q2_2000` partitions are transported to the target database. The other two partitions are not transported.

Complete the following tasks to transport the partitions using an export dump file:

Task 1 Generate the Export Dump File

Generate the export dump file by completing the following steps:

1. Start SQL*Plus and connect to the source database as an administrator or as a user who has either the `ALTER TABLESPACE` or `MANAGE TABLESPACE` system privilege.

   See "Connecting to the Database with SQL*Plus" for instructions.

2. Make all of the tablespaces that contain the tables being transported read-only.

```
ALTER TABLESPACE sales_prt_tbs READ ONLY;
```

3. Invoke the Data Pump export utility as a user with `DATAPUMP_EXP_FULL_DATABASE` role and specify the transportable tables options.

```
SQL> HOST

expdp user_name dumpfile=sales_prt.dmp  directory=data_pump_dir
      tables=sh.sales_prt:sales_q1_2000,sh.sales_prt:sales_q2_2000
      transportable=always logfile=exp.log

Password: password
```

   You must always specify `TRANSPORTABLE=ALWAYS`, which specifies that the transportable option is used.

   This example specifies the following additional Data Pump parameters:

   - The `DUMPFILE` parameter specifies the name of the structural information export dump file to be created, `sales_prt.dmp`.
   - The `DIRECTORY` parameter specifies the directory object that points to the operating system or Oracle Automatic Storage Management location of the dump file. You must create the `DIRECTORY` object before invoking Data Pump, and you must grant the `READ` and `WRITE` object privileges on the directory to the user running the Export utility. See *Oracle Database SQL Language Reference* for information on the `CREATE DIRECTORY` command.

     However, the directory object `DATA_PUMP_DIR` is not created automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB and specify the directory object when you run Data Pump.

> **See Also:**
>
> – *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
>
> – *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `TABLES` parameter specifies the tables, partitions, or subpartitions being exported.
- The `LOGFILE` parameter specifies the file name of the log file to be written by the export utility. In this example, the log file is written to the same directory as the dump file, but it can be written to a different location.

4. Check the log file for unexpected errors, and take note of the dump file and data files that you must transport to the target database. `expdp` outputs the names and paths of these files in messages like these:

```
Processing object type TABLE_EXPORT/TABLE/PLUGTS_BLK
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/END_PLUGTS_BLK
Master table "SYSTEM"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/rdbms/log/sales_prt.dmp
******************************************************************************
Datafiles required for transportable tablespace SALES_PRT_TBS:
  /u01/app/oracle/oradata/sourcedb/sales_prt.dbf
Job "SYSTEM"."SYS_EXPORT_TABLE_01" successfully completed at 11:32:13
```

5. When finished, exit back to SQL*Plus:

```
$ exit
```

> **See Also:**
>
> *Oracle Database Utilities* for information about using the Data Pump utility

Task 2 Transport the Export Dump File

Transport the dump file to the directory pointed to by the `DATA_PUMP_DIR` directory object on the target database, or to any other directory of your choosing. The new location must be accessible to the target database.

In this example, transfer the `sales_prt.dmp` dump file from the source database to the target database.

At the target database, run the following query to determine the location of `DATA_PUMP_DIR`:

```
SELECT * FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME = 'DATA_PUMP_DIR';

OWNER      DIRECTORY_NAME   DIRECTORY_PATH
---------- ---------------- ----------------------------------
SYS        DATA_PUMP_DIR    /u01/app/oracle/rdbms/log/
```

Task 3 Transport the Data Files for the Tables

Transport the data files of the tablespaces containing the tables being transported to a place that is accessible to the target database.

Typically, you transport the data files to the location of the existing data files of the target database. On the UNIX and Linux platforms, this location is typically /u01/app/oracle/oradata/*dbname*/ or *+DISKGROUP*/*dbname*/datafile/.
In this example, transfer the `sales_prt.dbf` data file from the source database to the target database.

> **See Also:**
>
> "Guidelines for Transferring Data Files"

Task 4 (Optional) Restore Tablespaces to Read/Write Mode
Make the tablespaces that contain the tables being transported read/write again at the source database, as follows:

```
ALTER TABLESPACE sales_prt_tbs READ WRITE;
```

You can postpone this task to first ensure that the import process succeeds.

Task 5 At the Target Database, Import the Partitions
At the target database, invoke the Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role and specify the transportable tables options.

```
impdp user_name dumpfile=sales_prt.dmp directory=data_pump_dir
   transport_datafiles='/u01/app/oracle/oradata/targetdb/sales_prt.dbf'
   tables=sh.sales_prt:sales_q1_2000,sh.sales_prt:sales_q2_2000
   logfile=imp.log

Password: password
```

This example specifies the following Data Pump parameters:

- The `DUMPFILE` parameter specifies the exported file containing the metadata for the data to be imported.
- The `DIRECTORY` parameter specifies the directory object that identifies the location of the export dump file. You must create the `DIRECTORY` object before invoking Data Pump, and you must grant the `READ` and `WRITE` object privileges on the directory to the user running the Import utility. See *Oracle Database SQL Language Reference* for information on the `CREATE DIRECTORY` command.

  However, the directory object `DATA_PUMP_DIR` is not created automatically in a PDB. Therefore, when importing into a PDB, create a directory object in the PDB and specify the directory object when you run Data Pump.

  > **See Also:**
  >
  > – *Oracle Database Utilities* for information about the default directory when the `DIRECTORY` parameter is omitted
  > – *Oracle Multitenant Administrator's Guide* for more information about PDBs

- The `TRANSPORT_DATAFILES` parameter identifies all of the data files to be imported.

You can specify the `TRANSPORT_DATAFILES` parameter multiple times in a parameter file specified with the `PARFILE` parameter if there are many data files.

- The `TABLES` parameter specifies the tables, partitions, or subpartitions being imported.
- The `LOGFILE` parameter specifies the file name of the log file to be written by the import utility. In this example, the log file is written to the directory from which the dump file is read, but it can be written to a different location.

After this statement executes successfully, check the import log file to ensure that no unexpected error has occurred.
When dealing with a large number of data files, specifying the list of data file names in the statement line can be a laborious process. It can even exceed the statement line limit. In this situation, you can use an import parameter file. For example, you can invoke the Data Pump import utility as follows:

```
impdp user_name parfile='par.f'
```

For example, `par.f` might contain the following lines:

```
DUMPFILE=sales_prt.dmp
DIRECTORY=data_pump_dir
TRANSPORT_DATAFILES='/u01/app/oracle/oradata/targetdb/sales_prt.dbf'
TABLES=sh.sales_prt:sales_q1_2000,sh.sales_prt:sales_q2_2000
LOGFILE=imp.log
```

Note:

- The partitions are imported as separate tables in the target database because this example transports a subset of partitions.
- During the import, tablespaces might be temporarily made read/write for metadata loading. Ensure that no user changes are made to the data during the import. At the successful completion of the import, all user-defined tablespaces are made read/write.
- When performing a network database import, the `TRANSPORTABLE` parameter must be set to `always`.

See Also:

Oracle Database Utilities for information about using the import utility

13.4.4 Transporting Tables, Partitions, or Subpartitions Over the Network

To transport tables over the network, you perform an import using the `NETWORK_LINK` parameter, the import is performed using a database link, and there is no dump file involved.

The following list of tasks summarizes the process of transporting tables, partitions, and subpartitions between databases over the network. Details for each task are provided in the subsequent example.

1. Pick a set of tables, partitions, or subpartitions.

If you are transporting partitions, then you can specify partitions from only one table in a transportable tables operation, and no other tables can be transported in the same operation. Also, if only a subset of a table's partitions are exported in a transportable tables operation, then on import each partition becomes a non-partitioned table.

2. At the source database, place the tablespaces associated with the data files for the tables, partitions, or subpartitions in read-only mode.

   To view the tablespace for a table, query the `DBA_TABLES` view. To view the data file for a tablespace, query the `DBA_DATA_FILES` view.

3. Transport the data files for the tables, partitions, or subpartitions.

   Copy the data files to a place that is accessible to the target database.

   If the source platform and target platform are different, then check the endian format of each platform by running the query on the `V$TRANSPORTABLE_PLATFORM` view in "Transporting Data Across Platforms".

   If the source platform's endian format is different from the target platform's endian format, then use one of the following methods to convert the data files:

   - Use the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data files. These procedures convert the data files to the target platform's endian format automatically.
   - Use the RMAN `CONVERT` command to convert the data files to the target platform's endian format.

   > **Note:**
   >
   > Conversion of data files between different endian formats is not supported for data files having undo segments.

   See "Converting Data Between Platforms" for more information.

4. At the target database, perform the import.

   Invoke the Data Pump utility to import the metadata for the tables.

5. (Optional) Restore tablespaces to read/write mode on the source database.

Example

These tasks for transporting tables over the network are illustrated more fully in the example that follows, where it is assumed that the tables exist in the source database:

Table	Tablespace	Data File
`hr.emp_ttbs`	`emp_tsp`	/u01/app/oracle/oradata/sourcedb/emp.dbf
`oe.orders_ttbs`	`orders_tsp`	/u01/app/oracle/oradata/sourcedb/orders.dbf

This example transports these tables to the target database. To complete the example, these tables must exist on the source database.

The following SQL statements create the tables in the `hr` schema and the tablespaces and data files for the tables. The statements also insert data into the tables by using data in the `hr` and `oe` sample schemas.

```
CREATE TABLESPACE emp_tsp
   DATAFILE 'emp.dbf' SIZE 1M
   ONLINE;

CREATE TABLE hr.emp_ttbs(
   employee_id    NUMBER(6),
   first_name     VARCHAR2(20),
   last_name      VARCHAR2(25),
   email          VARCHAR2(25),
   phone_number   VARCHAR2(20),
   hire_date      DATE,
   job_id         VARCHAR2(10),
   salary         NUMBER(8,2),
   commission_pct NUMBER(2,2),
   manager_id     NUMBER(6),
   department_id  NUMBER(4))
 TABLESPACE emp_tsp;

INSERT INTO hr.emp_ttbs SELECT * FROM hr.employees;

CREATE TABLESPACE orders_tsp
   DATAFILE 'orders.dbf' SIZE 1M
   ONLINE;

CREATE TABLE oe.orders_ttbs(
   order_id      NUMBER(12),
   order_date    TIMESTAMP WITH LOCAL TIME ZONE,
   order_mode    VARCHAR2(8),
   customer_id   NUMBER(6),
   order_status  NUMBER(2),
   order_total   NUMBER(8,2),
   sales_rep_id  NUMBER(6),
   promotion_id  NUMBER(6))
 TABLESPACE orders_tsp;

INSERT INTO oe.orders_ttbs SELECT * FROM oe.orders;

COMMIT;
```

This example makes the following additional assumptions:

- The name of the source database is `sourcedb`.
- The source database and target database are running on the same platform with the same endianness. To check the endianness of a platform, run the following query:

```
SELECT d.PLATFORM_NAME, ENDIAN_FORMAT
     FROM V$TRANSPORTABLE_PLATFORM tp, V$DATABASE d
     WHERE tp.PLATFORM_NAME = d.PLATFORM_NAME;
```

Complete the following tasks to transport the tables over the network:

Task 1 Create a Database Link from the Target Database to the Source Database
Create a database link from the target database to the source database by completing the following steps:

1. Ensure that network connectivity is configured between the source database and the target database.

   See *Oracle Database Net Services Administrator's Guide* for instructions.

2. Start SQL*Plus and connect to the target database as the administrator who will transport the data with Data Pump import. This user must have `DATAPUMP_IMP_FULL_DATABASE` role to transport the data.

   See "Connecting to the Database with SQL*Plus" for instructions.

3. Create the database link:

   ```
   CREATE PUBLIC DATABASE LINK sourcedb USING 'sourcedb';
   ```

   Specify the service name for the source database in the using clause.

   During the import operation, the database link must connect to a user on the source database with `DATAPUMP_EXP_FULL_DATABASE` role. The user on the source database cannot be a user with `SYSDBA` administrative privilege.

> **See Also:**
>
> - "Creating Database Links"
> - *Oracle Database SQL Language Reference*

Task 2 Make the Tablespaces Containing the Tables Read-Only
At the source database, complete the following steps:

1. Start SQL*Plus and connect to the source database as an administrator or as a user who has either the `ALTER TABLESPACE` or `MANAGE TABLESPACE` system privilege.

   See "Connecting to the Database with SQL*Plus" for instructions.

2. Make all of the tablespaces that contain data to be transported read-only.

   ```
   ALTER TABLESPACE emp_tsp READ ONLY;
   ALTER TABLESPACE orders_tsp READ ONLY;
   ```

Task 3 Transport the Data Files for the Tables
Transport the data files of the tablespaces containing the tables being transported to a place that is accessible to the target database.
Typically, you transport the data files to the location of the existing data files of the target database. On the UNIX and Linux platforms, this location is typically /u01/app/oracle/oradata/*dbname*/ or *+DISKGROUP*/*dbname*/datafile/.
In this example, transfer the `emp.dbf` and `orders.dbf` data files from the source database to the target database.

> **See Also:**
>
> "Guidelines for Transferring Data Files"

Task 4 At the Target Database, Import the Database
Invoke the Data Pump import utility as a user with `DATAPUMP_IMP_FULL_DATABASE` role and specify the full transportable export/import options.

```
impdp user_name network_link=sourcedb transportable=always
   transport_datafiles=
      '/u01/app/oracle/oradata/targetdb/emp.dbf'
```

```
      '/u01/app/oracle/oradata/targetdb/orders.dbf'
   tables=hr.emp_ttbs,oe.orders_ttbs
   logfile=import.log

Password: password
```

This example specifies the following Data Pump parameters:

- The `NETWORK_LINK` parameter specifies the database link to the source database used for the network import.
- The `TRANSPORTABLE` parameter specifies that the import uses the transportable option.
- The `TRANSPORT_DATAFILES` parameter identifies all of the data files to be imported.

  You can specify the `TRANSPORT_DATAFILES` parameter multiple times in a parameter file specified with the `PARFILE` parameter if there are many data files.
- The `TABLES` parameter specifies the tables to be imported.
- The `LOGFILE` parameter specifies the file name of the log file to be written by the import utility.

After this statement executes successfully, check the import log file to ensure that no unexpected error has occurred.
When dealing with a large number of data files, specifying the list of data file names in the statement line can be a laborious process. It can even exceed the statement line limit. In this situation, you can use an import parameter file. For example, you can invoke the Data Pump import utility as follows:

```
impdp user_name parfile='par.f'
```

For example, `par.f` might contain the following lines:

```
NETWORK_LINK=sourcedb
TRANSPORTABLE=always
TRANSPORT_DATAFILES=
      '/u01/app/oracle/oradata/targetdb/emp.dbf'
      '/u01/app/oracle/oradata/targetdb/orders.dbf'
TABLES=hr.emp_ttbs,oe.orders_ttbs
LOGFILE=import.log
```

> **Note:**
>
> During the import, user-defined tablespaces might be temporarily made read/write for metadata loading. Ensure that no user changes are made to the data during the import. At the successful completion of the import, all user-defined tablespaces are made read/write.

> **See Also:**
>
> *Oracle Database Utilities* for information about using the import utility

Task 5 (Optional) Restore Tablespaces to Read/Write Mode
Make the tables that contain the tables being transported read/write again at the source database, as follows:

```
ALTER TABLESPACE emp_tsp READ WRITE;
ALTER TABLESPACE orders_tsp READ WRITE;
```

13.5 Converting Data Between Platforms

To transport tables across platforms, check platform endianness, and review other restrictions.

When you perform a transportable operation, and the source platform and the target platform are of different endianness, you must convert the data being transported to the target platform format. If the source platform and the target platform are of the same endianness, then data conversion is not necessary. You can use the `DBMS_FILE_TRANSFER` package or the RMAN `CONVERT` command to convert data.

> **Note:**
>
> Some limitations might apply that are not described in these sections. Refer to the following documentation for more information:
>
> - "Transporting Data Across Platforms" for information about checking the endianness of platforms
> - DBMS_FILE_TRANSFER in *Oracle Database PL/SQL Packages and Types Reference* for information about limitations related to the `DBMS_FILE_TRANSFER` package
> - CONVERT IN *Oracle Database Backup and Recovery Reference* for information about limitations related to the RMAN `CONVERT` command

- Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package
  You can use the `GET_FILE` or `PUT_FILE` procedure of the `DBMS_FILE_TRANSFER` package to convert data between platforms during a data file transfer.
- Converting Data Between Platforms Using RMAN
  To convert data with the RMAN convert command, be aware of restrictions and follow these guidelines.

13.5.1 Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package

You can use the `GET_FILE` or `PUT_FILE` procedure of the `DBMS_FILE_TRANSFER` package to convert data between platforms during a data file transfer.

When you use one of these procedures to move data files between the source platform and the target platform, each block in each data file is converted to the target platform's endianness.

This section uses an example to describe how to use the `DBMS_FILE_TRANSFER` package to convert a data file to a different platform. The example makes the following assumptions:

- The `GET_FILE` procedure will transfer the data file.
- The `mytable.342.123456789` data file is being transferred to a different platform.
- The endianness of the source platform is different from the endianness of the target platform.

- The global name of the source database is `dbsa.example.com`.
- Both the source database and the target database use Oracle Automatic Storage Management (Oracle ASM).

> **Note:**
>
> You can also use the `DBMS_FILE_TRANSFER` package to transfer data files between platforms with the same endianness.

Complete the following steps to convert the data file by transferring it with the `GET_FILE` procedure:

1. Use SQL*Plus to connect to the source database as an administrative user who can create directory objects.
2. Create a directory object to store the data files that you want to transfer to the target database.

   For example, to create a directory object named `sales_dir_source` for the `+data/dbsa/datafile` directory, execute the following SQL statement:

   ```
   CREATE OR REPLACE DIRECTORY sales_dir_source
      AS '+data/dbsa/datafile';
   ```

   The specified file system directory must exist when you create the directory object.
3. Use SQL*Plus to connect to the target database as an administrative user who can create database links, create directory objects, and run the procedures in the `DBMS_FILE_TRANSFER` package.
4. Create a database link from the target database to the source database.

   The connected user at the source database must have read privileges on the directory object that you created in Step 2.
5. Create a directory object to store the data files that you want to transfer from the source database.

   The user at the local database who will run the procedure in the `DBMS_FILE_TRANSFER` package must have write privileges on the directory object.

   For example, to create a directory object named `sales_dir_target` for the `+data/dbsb/datafile` directory, run the following SQL statement:

   ```
   CREATE OR REPLACE DIRECTORY sales_dir_target
     AS '+data/dbsb/datafile';
   ```
6. Run the `GET_FILE` procedure in the `DBMS_FILE_TRANSFER` package to transfer the data file.

   For example, run the following procedure to transfer the `mytable.342.123456789` data file from the source database to the target database using the database link you created in Step 4:

   ```
   BEGIN
     DBMS_FILE_TRANSFER.GET_FILE(
       source_directory_object      => 'sales_dir_source',
       source_file_name             => 'mytable.342.123456789',
       source_database              => 'dbsa.example.com',
       destination_directory_object => 'sales_dir_target',
       destination_file_name        => 'mytable');
   ```

```
END;
/
```

> **Note:**
>
> In this example, the destination data file name is `mytable`. Oracle ASM does not allow a fully qualified file name form in the `destination_file_name` parameter of the `GET_FILE` procedure.

Related Topics

- About Connecting to the Database with SQL*Plus
  Oracle Database includes the following components: the Oracle Database instance, which is a collection of processes and memory, and a set of disk files that contain user data and system data.
- Creating Database Links
  To support application access to the data and schema objects throughout a distributed database system, you must create all necessary database links.
- DBMS_FILE_TRANSFER in *Oracle Database PL/SQL Packages and Types Reference*
- Fully Qualified File Name Form in *Oracle Automatic Storage Management Administrator's Guide*
- CREATE DATABASE LINK in *Oracle Database SQL Language Reference*

13.5.2 Converting Data Between Platforms Using RMAN

To convert data with the RMAN convert command, be aware of restrictions and follow these guidelines.

When you use the RMAN `CONVERT` command to convert data, you can either convert the data on the source platform after running Oracle Data Pump export, or you can convert the data on the target platform before running Oracle Data Pump import. In either case, you must transfer the data files from the source system to the target system.

To convert data, you can use the following RMAN `CONVERT` commands:

- `CONVERT DATAFILE`
- `CONVERT TABLESPACE`
- `CONVERT DATABASE`

> **Note:**
>
> - Datatype restrictions apply to the RMAN `CONVERT` command.
> - RMAN `CONVERT` commands do not support conversion of data files between different endian formats for data files having undo segments.

- Converting Tablespaces on the Source System After Export
  You can use this example to see how to use the RMAN `CONVERT TABLESPACE` command to convert tablespaces to a different platform.

- Converting Data Files on the Target System Before Import
  An example illustrates how to use the RMAN `CONVERT DATAFILE` command to convert data files to a different platform.

Related Topics

- CONVERT in *Oracle Database Backup and Recovery Reference*
- Transporting Data Across Platforms in *Oracle Database Backup and Recovery User's Guide*

13.5.2.1 Converting Tablespaces on the Source System After Export

You can use this example to see how to use the RMAN `CONVERT TABLESPACE` command to convert tablespaces to a different platform.

The example makes the following assumptions:

- The `sales_1` and `sales_2` tablespaces are being transported to a different platform.
- The endianness of the source platform is different from the endianness of the target platform.
- You want to convert the data on the source system, before transporting the tablespace set to the target system.
- You have completed the Oracle Data Pump export on the source database.

To convert the tablespaces on the source system, complete the following steps:

1. At a command prompt, start RMAN, and connect to the source database:

```
$ RMAN TARGET /

Recovery Manager: Release 12.1.0.1.0 - Production

Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All rights reserved.

connected to target database: salesdb (DBID=3295731590)
```

2. Use the RMAN `CONVERT TABLESPACE` command to convert the data files into a temporary location on the source platform.

   In this example, assume that the temporary location, directory `/tmp`, has already been created. The converted data files are assigned names by the system.

```
RMAN> CONVERT TABLESPACE sales_1,sales_2
2> TO PLATFORM 'Microsoft Windows IA (32-bit)'
3> FORMAT '/tmp/%U';

Starting conversion at source at 30-SEP-08
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile conversion
input datafile file number=00007 name=/u01/app/oracle/oradata/salesdb/sales_101.dbf
converted datafile=/tmp/data_D-SALESDB_I-1192614013_TS-SALES_1_FNO-7_03jru08s
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:45
channel ORA_DISK_1: starting datafile conversion
input datafile file number=00008 name=/u01/app/oracle/oradata/salesdb/sales_201.dbf
converted datafile=/tmp/data_D-SALESDB_I-1192614013_TS-SALES_2_FNO-8_04jru0aa
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:25
Finished conversion at source at 30-SEP-08
```

> **See Also:**
>
> *Oracle Database Backup and Recovery Reference* for a description of the RMAN `CONVERT` command

3. Exit Recovery Manager:

```
RMAN> exit
Recovery Manager complete.
```

4. Transfer the data files to the target system.

Related Topics

- Guidelines for Transferring Data Files
  You should follow a set of guidelines when transferring the data files.

13.5.2.2 Converting Data Files on the Target System Before Import

An example illustrates how to use the RMAN `CONVERT DATAFILE` command to convert data files to a different platform.

During the conversion, you identify the data files by file name, not by tablespace name. Until the tablespace metadata is imported, the target instance has no way of knowing the desired tablespace names.

The example makes the following assumptions:

- You have not yet converted the data files for the tablespaces being transported.

  If you used the `DBMS_FILE_TRANSFER` package to transfer the data files to the target system, then the data files were converted automatically during the file transfer. See "Converting Data Between Platforms Using the DBMS_FILE_TRANSFER Package".
- The following data files are being transported to a different platform:
  - C:\Temp\sales_101.dbf
  - C:\Temp\sales_201.dbf
- The endianness of the source platform is different from the endianness of the target platform.
- You want to convert the data on the target system, before performing the Data Pump import.
- The converted data files are placed in C:\app\orauser\oradata\orawin\, which is the location of the existing data files for the target system:

Complete the following steps to convert the tablespaces on the target system:

1. If you are in SQL*Plus, then return to the host system:

```
SQL> HOST
```

2. Use the RMAN `CONVERT DATAFILE` command to convert the data files on the target platform:

```
C:\>RMAN TARGET /

Recovery Manager: Release 12.1.0.1.0 - Production

Copyright (c) 1982, 2012, Oracle and/or its affiliates.  All rights reserved.
```

```
connected to target database: ORAWIN (DBID=3462152886)

RMAN> CONVERT DATAFILE
2>'C:\Temp\sales_101.dbf',
3>'C:\Temp\sales_201.dbf'
4>TO PLATFORM="Microsoft Windows IA (32-bit)"
5>FROM PLATFORM="Solaris[tm] OE (32-bit)"
6>DB_FILE_NAME_CONVERT=
7>'C:\Temp\', 'C:\app\orauser\oradata\orawin\'
8> PARALLELISM=4;
```

If the source location, the target location, or both do not use Oracle Automatic Storage Management (Oracle ASM), then the source and target platforms are optional. RMAN determines the source platform by examining the data file, and the target platform defaults to the platform of the host running the conversion.

If both the source and target locations use Oracle ASM, then you must specify the source and target platforms in the `DB_FILE_NAME_CONVERT` clause.

3. Exit Recovery Manager:

```
RMAN> exit
Recovery Manager complete.
```

Related Topics

- CONVERT in *Oracle Database Backup and Recovery User's Guide*

13.6 Guidelines for Transferring Data Files

You should follow a set of guidelines when transferring the data files.

If both the source and target are file systems, then you can transport using:

- Any facility for copying flat files (for example, an operating system copy utility or ftp)
- The `DBMS_FILE_TRANSFER` package
- RMAN
- Any facility for publishing on CDs

If either the source or target is an Oracle Automatic Storage Management (Oracle ASM) disk group, then you can use:

- ftp to or from the `/sys/asm` virtual folder in the XML DB repository

  See *Oracle Automatic Storage Management Administrator's Guide* for more information.
- The `DBMS_FILE_TRANSFER` package
- RMAN

Do not transport the data files for the administrative tablespaces (such as `SYSTEM` and `SYSAUX`) or any undo or temporary tablespaces.

If you are transporting data of a different block size than the standard block size of the database receiving the data, then you must first have a `DB_`*`n`*`K_CACHE_SIZE` initialization parameter entry in the receiving database parameter file.

For example, if you are transporting data with an 8K block size into a database with a 4K standard block size, then you must include a `DB_8K_CACHE_SIZE` initialization parameter entry in

the parameter file. If it is not already included in the parameter file, then this parameter can be set using the `ALTER SYSTEM SET` statement.

See *Oracle Database Reference* for information about specifying values for the `DB_nK_CACHE_SIZE` initialization parameter.

Starting with Oracle Database 12*c*, the `GET_FILE` or `PUT_FILE` procedure in the `DBMS_FILE_TRANSFER` package can convert data between platforms during the data file transfer. See "Converting Data Between Platforms".

Starting with Oracle Database 12*c*, RMAN can transfer files using network-enabled restore. RMAN restores database files, over the network, from a remote database instance by using the `FROM SERVICE` clause of the `RESTORE` command. The primary advantage of network-enabled restore is that it eliminates the requirement for a restore of the backup to a staging area on disk and the need to transfer the copy. Therefore, network-enabled restore saves disk space and time. This technique can also provide the following advantages during file transfer: compression, encryption, and transfer of used data blocks only. See *Oracle Database Backup and Recovery User's Guide* for more information.

Note:

Exercise caution when using the UNIX `dd` utility to copy raw-device files between databases, and note that Oracle Database 12*c* and later do not support raw devices for database files. The `dd` utility can be used to copy an entire source raw-device file, or it can be invoked with options that instruct it to copy only a specific range of blocks from the source raw-device file.

It is difficult to ascertain actual data file size for a raw-device file because of hidden control information that is stored as part of the data file. If you must use the `dd` utility to operate on raw devices, then specify the entire source raw-device file contents. If you move database file content from a raw device to either ASM or a file system to adhere to the desupport of raw devices with Oracle Database 12*c* and later, then use an Oracle-provided tool such as RMAN.

See Also:

"Copying Files Using the Database Server" for information about using the `DBMS_FILE_TRANSFER` package to copy the files that are being transported and their metadata