87

DBMS_FILE_TRANSFER

The `DBMS_FILE_TRANSFER` package provides procedures to copy a binary file within a database or to transfer a binary file between databases.

This chapter contains the following topics:

See Also:

Oracle Database Administrator's Guide for instructions about using file transfer

DBMS_FILE_TRANSFER Overview

The `DBMS_FILE_TRANSFER` package provides procedures to copy a binary file within a database or to transfer a binary file between databases.

The destination database converts each block when it receives a file from a platform with different endianness. Datafiles can be imported after they are moved to the destination database as part of a transportable operation without RMAN conversion. Both `GET` and `PUT` operations will converted the file across platform difference at the destination. However, `COPY` is a local operation and therefore no conversion is required.

DBMS_FILE_TRANSFER Security Model

The `DBMS_FILE_TRANSFER` package must be created under `SYS` (`CONNECT INTERNAL`). Operations provided by this package are performed under the current calling user, not the package owner (`SYS`).

To use this interface the following users must have the following privileges:

- The current user at the local database must have `READ` privilege on the directory object specified in the `source_directory_object` parameter.
- The connected user at the destination database must have `WRITE` privilege to the directory object specified in the `destination_directory_object` parameter.

DBMS_FILE_TRANSFER Operational Notes

`DBMS_FILE_TRANSFER` supports online backup. You should therefore be careful in copying or transferring a file that is being modified by the database because this can result in an inconsistent file, and require recovery. To guarantee consistency, bring files offline when the database is in use.

If you want to use `DBMS_FILE_TRANSFER` for performing backups, note that you are implementing self-managed backups, and should therefore put the files in hot backup mode.

Summary of DBMS_FILE_TRANSFER Subprograms

This table lists the `DBMS_FILE_TRANSFER` subprograms and briefly describes them.

Table 87-1 DBMS_FILE_TRANSFER Package Subprograms

Subprogram	Description
COPY_FILE Procedure	Reads a file from a source directory and creates a copy of it in a destination directory. The source and destination directories can both be in a local file system, or both be in an Automatic Storage Management (ASM) disk group, or between local file system and ASM with copying in either direction.
GET_FILE Procedure	Contacts a remote database to read a remote file and then creates a copy of the file in the local file system or ASM
PUT_FILE Procedure	Reads a local file or ASM and contacts a remote database to create a copy of the file in the remote file system

COPY_FILE Procedure

This procedure reads a file from a source directory and creates a copy of it in a destination directory. The source and destination directories can both be in a local file system, or both be in an Automatic Storage Management (ASM) disk group, or between local file system and ASM with copying in either direction.

You can copy any type of file to and from a local file system. However, you can copy only database files (such as datafiles, tempfiles, controlfiles, and so on) to and from an ASM disk group.

The destination file is not closed until the procedure completes successfully.

Syntax

```
DBMS_FILE_TRANSFER.COPY_FILE(
   source_directory_object       IN  VARCHAR2,
   source_file_name              IN  VARCHAR2,
   destination_directory_object  IN  VARCHAR2,
   destination_file_name         IN  VARCHAR2);
```

Parameters

Table 87-2 *COPY_FILE Procedure Parameters*

Parameter	Description
`source_directory_object`	The directory object that designates the source directory. The directory object must already exist. (You create directory objects with the `CREATE DIRECTORY` command).

Table 87-2 (Cont.) *COPY_FILE Procedure Parameters*

Parameter	Description
`source_file_name`	The name of the file to copy. This file must exist in the source directory.
`destination_directory_object`	The directory object that designates the destination directory. The directory object must already exist. If the destination is ASM, the directory object must designate either a disk group name (for example, `+diskgroup1`) or a directory created for alias names. In the case of a directory, the full path to the directory must be specified (for example: `+diskgroup1/dbs/control`).
`destination_file_name`	The name to assign to the file in the destination directory. A file with the same name must not exist in the destination directory. If the destination is ASM: • The file is given a fully qualified ASM filename and created in the appropriate directory (depending on the database name and file type) • The file type tag assigned to the file is `COPY_FILE` • The value of the `destination_file_name` argument becomes the file's alias name in the designated destination directory The file name can be followed by an ASM template name in parentheses. The file is then given the attributes specified by the template.

Usage Notes

To run this procedure successfully, the current user must have the following privileges:

- `READ` privilege on the directory object specified in the `source_directory_object` parameter
- `WRITE` privilege on directory object specified in the `destination_directory_object` parameter

This procedure converts directory object parameters to uppercase unless they are surrounded by double quotation marks, but this procedure does not convert file names to uppercase.

Also, the copied file must meet the following requirements:

- The size of the copied file must be a multiple of 512 bytes.
- The size of the copied file must be less than or equal to two terabytes.

The `source_file_name` parameter must specify a file that is in the directory specified by the `source_directory_object` parameter before running the procedure, and the `destination_file_name` parameter must specify the new name of the file in the new location specified in the `destination_directory_object` parameter. Relative paths and symbolic links are not allowed in the directory objects for the `source_directory_object` and `destination_directory_object` parameters.

Transferring the file is not transactional. To monitor the progress of a long file copy, query the `V$SESSION_LONGOPS` dynamic performance view.

See Also:

*Oracle Automatic Storage Management Administrator's Guide*for instructions about using file transfer

Examples

```
SQL> create directory DGROUP as '+diskgroup1/dbs/backup';

Directory created.

SQL>  BEGIN
   2    DBMS_FILE_TRANSFER.COPY_FILE('SOURCEDIR','t_xdbtmp.f', 'DGROUP',
                                   't_xdbtmp.f');
   3  END;
   4  /

PL/SQL procedure successfully completed.

SQL> EXIT
$ASMCMD
ASMCMD> ls
DISKGROUP1/
ASMCMD> cd diskgroup1/dbs/backup
ASMCMD> ls
t_xdbtmp.f => +DISKGROUP1/ORCL/TEMPFILE/COPY_FILE.267.546546525
```

GET_FILE Procedure

This procedure contacts a remote database to read a remote file and then creates a copy of the file in the local file system or ASM. The file that is copied is the source file, and the new file that results from the copy is the destination file. The destination file is not closed until the procedure completes successfully.

Syntax

```
DBMS_FILE_TRANSFER.GET_FILE
   source_directory_object      IN  VARCHAR2,
   source_file_name             IN  VARCHAR2,
   source_database              IN  VARCHAR2,
   destination_directory_object IN  VARCHAR2,
   destination_file_name        IN  VARCHAR2);
```

Parameters

Table 87-3 GET_FILE Procedure Parameters

Parameter	Description
`source_directory_object`	The directory object from which the file is copied at the source site. This directory object must exist at the source site.
`source_file_name`	The name of the file that is copied in the remote file system. This file must exist in the remote file system in the directory associated with the source directory object.

Table 87-3 (Cont.) GET_FILE Procedure Parameters

Parameter	Description
`source_database`	The name of a database link to the remote database where the file is located.
`destination_directory_object`	The directory object into which the file is placed at the destination site. This directory object must exist in the local file system.
`destination_file_name`	The name of the file copied to the local file system. A file with the same name must not exist in the destination directory in the local file system.

Usage Notes

To run this procedure successfully, the following users must have the following privileges:

- The connected user at the source database must have read privilege on the directory object specified in the `source_directory_object` parameter.
- The current user at the local database must have write privilege on the directory object specified in the `destination_directory_object` parameter.

This procedure converts directory object parameters to uppercase unless they are surrounded by double quotation marks, but this procedure does not convert file names to uppercase.

Also, the copied file must meet the following requirements:

- The size of the copied file must be a multiple of 512 bytes.
- The size of the copied file must be less than or equal to two terabytes.

Transferring the file is not transactional. To monitor the progress of a long file transfer, query the `V$SESSION_LONGOPS` dynamic performance view.

Examples

```
CREATE OR REPLACE DIRECTORY df AS '+datafile' ;
GRANT WRITE ON DIRECTORY df TO "user";
CREATE DIRECTORY DSK_FILES AS ''^t_work^'';
GRANT WRITE ON DIRECTORY dsk_files TO "user";

-- asumes that dbs2 link has been created and we are connected to the instance.
-- dbs2 could be a loopback or point to another instance.

BEGIN
-- asm file to an os file
-- get an asm file from dbs1.asm/a1 to dbs2.^t_work^/oa5.dat
  DBMS_FILE_TRANSFER.GET_FILE ( 'df' , 'a1' , 'dbs1', 'dsk_files' , 'oa5.dat' );

-- os file to an os file
-- get an os file from dbs1.^t_work^/a2.dat to dbs2.^t_work^/a2back.dat
  DBMS_FILE_TRANSFER.GET_FILE ( 'dsk_files' , 'a2.dat' , 'dbs1', 'dsk_files' ,
'a2back.dat' );

END ;
/
```

PUT_FILE Procedure

This procedure reads a local file or ASM and contacts a remote database to create a copy of the file in the remote file system.

The file that is copied is the source file, and the new file that results from the copy is the destination file. The destination file is not closed until the procedure completes successfully.

Syntax

```
DBMS_FILE_TRANSFER.PUT_FILE(
   source_directory_object       IN  VARCHAR2,
   source_file_name              IN  VARCHAR2,
   destination_directory_object  IN  VARCHAR2,
   destination_file_name         IN  VARCHAR2,
   destination_database          IN  VARCHAR2);
```

Parameters

Table 87-4 PUT_FILE Procedure Parameters

Parameter	Description
`source_directory_object`	The directory object from which the file is copied at the local source site. This directory object must exist at the source site.
`source_file_name`	The name of the file that is copied from the local file system. This file must exist in the local file system in the directory associated with the source directory object.
`destination_directory_object`	The directory object into which the file is placed at the destination site. This directory object must exist in the remote file system.
`destination_file_name`	The name of the file placed in the remote file system. A file with the same name must not exist in the destination directory in the remote file system.
`destination_database`	The name of a database link to the remote database to which the file is copied.

Usage Notes

To run this procedure successfully, the following users must have the following privileges:

- The current user at the local database must have read privilege on the directory object specified in the `source_directory_object` parameter.
- The connected user at the destination database must have write privilege to the directory object specified in the `destination_directory_object` parameter.

This procedure converts directory object parameters to uppercase unless they are surrounded by double quotation marks, but this procedure does not convert file names to uppercase.

Also, the copied file must meet the following requirements:

- The size of the copied file must be a multiple of 512 bytes.
- The size of the copied file must be less than or equal to two terabytes.

Transferring the file is not transactional. To monitor the progress of a long file transfer, query the `V$SESSION_LONGOPS` dynamic performance view.

Examples

```
CREATE OR REPLACE DIRECTORY df AS '+datafile' ;
GRANT WRITE ON DIRECTORY df TO "user";
CREATE OR REPLACE DIRECTORY ft1 AS '+datafile/ft1' ;
GRANT READ,WRITE ON DIRECTORY ft1 TO "user";
CREATE OR REPLACE DIRECTORY ft1_1 AS '+datafile/ft1/ft1_1' ;

CONNECT user;
Enter password: password

-- - put a1.dat to a4.dat (using dbs2 dblink)
-- - level 2 sub dir to parent dir
-- - user has read privs on ft1_1 at dbs1 and write on df in dbs2
BEGIN
 DBMS_FILE_TRANSFER.PUT_FILE ( 'ft1_1' , 'a2.dat' , 'df' , 'a4.dat' ,
                               'dbs2' ) ;
END ;
```