21
Database File System Links

Database File System Links enable storing SecureFiles LOBs in a different location than usual.

- About Database File System Links
  DBFS Links allows storing SecureFiles LOBs transparently in a location separate from the segment where the LOB is normally stored. Instead, you store a link to the LOB in the segment.
- Ways to Create Database File System Links
  Database File System Links require the creation of a Database File System through the use of the DBFS Content package, `DBMS_DBFS_CONTENT`.
- Database File System Links Copy
  The API `DBMS_LOB.COPY_DBFS_LINK(DSTLOB, SRCLOB, FLAGS)` provides the ability to copy a linked SecureFiles LOB.
- The DBMS_LOB Package Used with DBFS
  The `DBMS_LOB` package provides subprograms to operate on, or access and manipulate specific parts of a LOB or complete LOBs.
- DBMS_LOB Constants Used with DBFS
  Certain constants support DBFS link interfaces.
- DBMS_LOB Subprograms Used with DBFS
  You should note that some changes have been made to the `DBMS_LOB` subprograms over time.
- Copying a Linked LOB Between Tables
  You can copy DBFS links from source tables to destination tables.
- Online Redefinition and DBFS Links
  Online redefinition copies any DBFS Links that are stored in any SecureFiles LOBs in the table being redefined.
- Transparent Read
  DBFS Links can read from a linked SecureFiles LOB even if the data is not cached in the database.

21.1 About Database File System Links

DBFS Links allows storing SecureFiles LOBs transparently in a location separate from the segment where the LOB is normally stored. Instead, you store a link to the LOB in the segment.

The link in the segment must reference a path that uses DBFS Content API to locate the LOB when accessed. This means that the LOB could be stored on another file system, on a tape system, in the cloud, or any other location that can be accessed using DBFS Content API.

When a user or application tries to access a SecureFiles LOB that has been stored outside the segment using a DBFS Link, the behavior can vary depending on the attempted operation and the characteristics of the DBFS store that holds the LOB:

- Read:

If the LOB is not already cached in a local area in the database, then it can be read directly from the DBFS content store that holds it, if the content store allows streaming access based on the setting of the `PROPNAME_STREAMABLE` parameter. If the content store does not allow streaming access, then the entire LOB will first be read into a local area in the database, where it will be stored for a period of time for future access.

- Write:

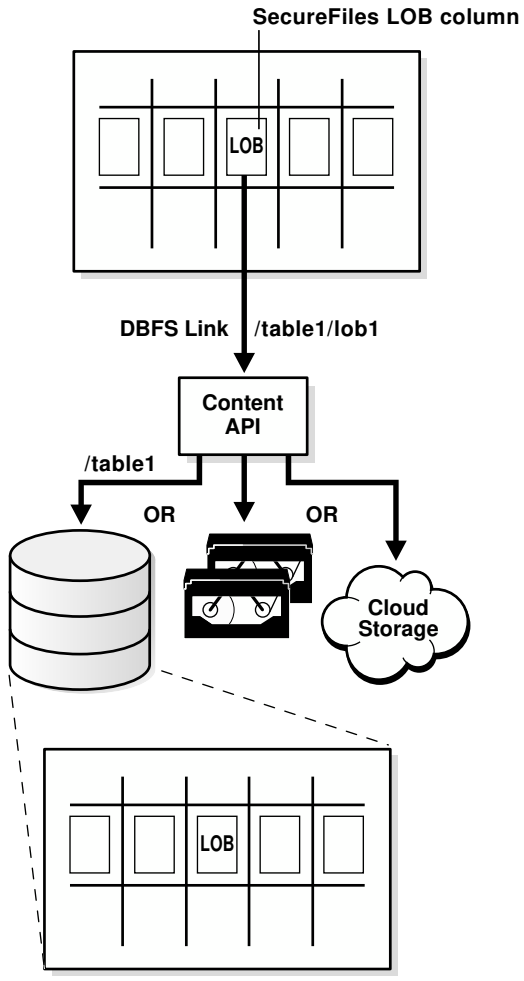
  If the LOB is not already cached in a local area in the database, then it will first be read into the database, modified as needed, and then written back to the DBFS content store defined in the DBFS Link for the LOB in question.

- Delete:

  When a SecureFiles LOB that is stored through a DBFS Link is deleted, the DBFS Link is deleted from the table, but the LOB itself is NOT deleted from the DBFS content store. Or it is more complex, based on the characteristics/settings, of the DBFS content store in question.

DBFS Links enable the use of SecureFiles LOBs to implement Hierarchical Storage Management (HSM) in conjunction with the DBFS Hierarchical Store (DBFS HS). HSM is a process by which the database moves rarely used or unused data from faster, more expensive, and smaller storage to slower, cheaper, and higher capacity storage.

Figure 21-1 Database File System Link



21.2 Ways to Create Database File System Links

Database File System Links require the creation of a Database File System through the use of the DBFS Content package, `DBMS_DBFS_CONTENT`.

Oracle provides several methods for creating a DBFS Link:

- Move SecureFiles LOB data into a specified DBFS pathname and store the reference to the new location in the LOB.

  Call `DBMS_LOB.MOVE_TO_DBFS_LINK()` with LOB and DBFS path name arguments, and the system creates the specified DBFS HSM Store if it does not exist, copies data from the SecureFiles LOB into the specified DBFS HSM Store, removes data from the SecureFiles LOB, and stores the file path name for subsequent access through this LOB.

- Copy or create a reference to an existing file.

  Call `DBMS_LOB.COPY_DBFS_LINK()` to copy a link from an existing DBFS Link. If there is any data in the destination SecureFiles LOB, the system removes this data and stores a copy of the reference to the link in the destination SecureFiles LOB.

- Call `DBMS_LOB.SET_DBFS_LINK()`, which assumes that the data for the link is stored in the specified DBFS path name.

  The system removes data in the specified SecureFiles LOB and stores the link to the DBFS path name.

Creating a DBFS Link impacts which operations may be performed and how. Any `DBMS_LOB` operations that modify the contents of a LOB will throw an exception if the underlying LOB has been moved into a DBFS Link. The application must explicitly replace the DBFS Link with a LOB by calling `DBMS_LOB.COPY_FROM_LINK()` before making these calls.

When it is completed, the application can move the updated LOB back to DBFS using `DBMS_LOB.MOVE_TO_DBFS_LINK()`, if needed. Other `DBMS_LOB` operations that existed before Oracle Database 11*g* Release 2 work transparently if the DBFS Link is in a file system that supports streaming. Note that these operations fail if streaming is either not supported or disabled.

If the DBFS Link file is modified through DBFS interfaces directly, the change is reflected in subsequent reads of the SecureFiles LOB. If the file is deleted through DBFS interfaces, then an exception occurs on subsequent reads.

For the database, it is also possible that a DBA may not want to store all of the data stored in a SecureFiles LOB HSM during export and import. Oracle has the ability to export and import only the Database File System Links. The links are fully qualified identifiers that provide access to the stored data, when entered into a SecureFiles LOB or registered on a SecureFiles LOB in a different database. This ability to export and import a link is similar to the common file system functionality of symbolic links.

The newly imported link is only available as long as the source, the stored data, is available, or until the first retrieval occurs on the imported system. The application is responsible for stored data retention. If the application system removes data from the store that still has a reference to it, the database throws an exception when the referencing SecureFiles LOB(s) attempt to access the data. Oracle also supports continuing to keep the data in the database after migration out to a DBFS store as a cached copy. It is up to the application to purge these copies in compliance with its retention policies.

21.3 Database File System Links Copy

The API `DBMS_LOB.COPY_DBFS_LINK(DSTLOB, SRCLOB, FLAGS)` provides the ability to copy a linked SecureFiles LOB.

sBy default, the LOB is not obtained from the DBFS HSM Store during this operation; this is a copy-by-reference operation that exports the DBFS path name (at source side) and imports it (at destination side). The `flags` argument can dictate that the destination has a local copy in the database and references the LOB data in the DBFS HSM Store.

21.4 The DBMS_LOB Package Used with DBFS

The `DBMS_LOB` package provides subprograms to operate on, or access and manipulate specific parts of a LOB or complete LOBs.

The `DBMS_LOB` package applies to both SecureFiles LOB and BasicFiles LOB.

DBMS_LOB Constants Used with SecureFiles LOBs and DBFS and DBMS_LOB Subprograms Used with SecureFiles LOBs and DBFS describe modifications made to the `DBMS_LOB` constants and subprograms with the addition of SecureFiles LOB and Database File System (DBFS).

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for more information about `DBMS_LOB` package
- Introducing the Database File System

21.5 DBMS_LOB Constants Used with DBFS

Certain constants support DBFS link interfaces.

Table 21-1 lists constants that support DBFS Link interfaces.

See Also:

Oracle Database PL/SQL Packages and Types Reference for complete information about constants used in the PL/SQL `DBMS_LOB` package

Table 21-1 DBMS_LOB Constants That Support DBFS Link Interfaces

Constant	Description
`DBFS_LINK_NEVER`	DBFS link state value
`DBFS_LINK_YES`	DBFS link state value

Table 21-1 (Cont.) DBMS_LOB Constants That Support DBFS Link Interfaces

Constant	Description
DBFS_LINK_NO	DBFS link state value
DBFS_LINK_CACHE	Flag used by COPY_DBFS_LINK() and MOVE_DBFS_LINK().
DBFS_LINK_NOCACHE	Flag used by COPY_DBFS_LINK() and MOVE_DBFS_LINK().
DBFS_LINK_PATH_MAX_SIZE	The maximum length of DBFS path names; 1024.
CONTENTTYPE_MAX_SIZE	The maximum 1-byte ASCII characters for content type; 128.

21.6 DBMS_LOB Subprograms Used with DBFS

You should note that some changes have been made to the DBMS_LOB subprograms over time.

Table 21-2 summarizes changes made to PL/SQL package DBMS_LOB subprograms.

Be aware that some of the DBMS_LOB operations that existed before Oracle Database 11*g* Release 2 throw an exception error if the LOB is a DBFS link. To remedy this problem, modify your applications to explicitly replace the DBFS link with a LOB by calling the DBMS_LOB.COPY_FROM_LINK procedure before they make these calls. When the call completes, then the application can move the updated LOB back to DBFS using the DBMS_LOB.MOVE_TO_DBFS_LINK procedure, if necessary.

Other DBMS_LOB operations that existed before Oracle Database 11*g* Release 2 work transparently if the DBFS Link is in a file system that supports streaming. Note that these operations fail if streaming is either not supported or disabled.

Table 21-2 DBMS_LOB Subprograms

Subprogram	Description
COPY_DBFS_LINK	Copies an existing DBFS link into a new LOB

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference*

Table 21-2 (Cont.) DBMS_LOB Subprograms

Subprogram	Description
COPY_FROM_DBFS_LINK	Copies the specified LOB data from DBFS HSM Store into the database **See Also:** *Oracle Database PL/SQL Packages and Types Reference*
DBFS_LINK_GENERATE_PATHNAME	Returns a unique file path name for creating a DBFS Link **See Also:** *Oracle Database PL/SQL Packages and Types Reference*
GET_DBFS_LINK	Returns the DBFS path name for a LOB **See Also:** *Oracle Database PL/SQL Packages and Types Reference*
GET_DBFS_LINK_STATE	Returns the linking state of a LOB **See Also:** *Oracle Database PL/SQL Packages and Types Reference*
MOVE_TO_DBFS_LINK	Moves the specified LOB data from the database into DBFS HSM Store **See Also:** *Oracle Database PL/SQL Packages and Types Reference*

Table 21-2 (Cont.) DBMS_LOB Subprograms

Subprogram	Description
SET_DBFS_LINK	Links a LOB with a DBFS path name

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference*

21.7 Copying a Linked LOB Between Tables

You can copy DBFS links from source tables to destination tables.

Use the following code to copy any DBFS Links that are stored in any SecureFiles LOBs in the source table to the destination table.

`CREATE TABLE ... AS SELECT (CTAS)` and `INSERT TABLE ... AS SELECT (ITAS)`

21.8 Online Redefinition and DBFS Links

Online redefinition copies any DBFS Links that are stored in any SecureFiles LOBs in the table being redefined.

21.9 Transparent Read

DBFS Links can read from a linked SecureFiles LOB even if the data is not cached in the database.

You can read data from the content store where the data is currently stored and stream that data back to the user application as if it were being read from the SecureFiles LOB segment. This allows seamless access to the DBFS Linked data without the prerequisite first call to `DBMS_LOB.COPY_FROM_DBFS_LINK()`.

Whether or not transparent read is available for a particular SecureFiles LOB is determined by the `DBFS_CONTENT` store where the data resides. This feature is always enabled for `DBFS_SFS` stores, and by default for `DBFS_HS` stores. To disable transparent read for `DBFS_HS` store, set the `PROPNAME_STREAMABLE` parameter to `FALSE`.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference*