20
DBFS Hierarchical Store

The DBFS Hierarchical Store and related store wallet management work together to store less frequently used data.

20.1 About the Hierarchical Store Package DBMS_DBFS_HS

The Oracle DBFS Hierarchical Store package (DBMS_DBFS_HS) is a store provider for DBMS_DBFS_CONTENT that supports hierarchical storage for DBFS content.

The package stores content in external storage devices like tape and Amazon S3 web service, and associated metadata (or properties) in the database. The DBFS HS may cache frequently accessed content in database tables to improve performance.

The DBMS_DBFS_HS package provides you the ability to use tape as a storage tier when implementing Information Lifecycle Management (ILM) for database tables or content. The data on tape or Amazon S3 is part of the Oracle Database and all standard APIs can access it, but only through the database.

DBMS_DBFS_HS has additional interfaces needed to manage the external storage device and the cache associated with each store.

To use the package DBMS_DBFS_HS, you must be granted the DBFS_ROLE role.

20.2 Setting up the Store

You can create, register, and mount a hierarchical Store.

20.2.1 Creating, Registering, and Mounting the Store

Setting up a hierarchical file system store requires creating, registering, and mounting the store.

Creating, registering, and mounting the store.

1. Call `CREATESTORE`.

> **See Also:**
>
> CREATESTORE Procedure for more information on `CREATESTORE` procedure.

> **Note:**
>
> You create a wallet with the credentials of the Amazon S3 accounts if Amazon S3 is used as the external storage.

2. Set mandatory and optional properties using `DBMS_DBFS_HS.SETSTOREPROPERTY`.

> **See Also:**
>
> SETSTOREPROPERTY Procedure for more information on `SETSTOREPROPERTY` procedure.

3. Register the store using `DBMS_DBFS_CONTENT.REGISTERSTORE`.

> **See Also:**
>
> REGISTERSTORE Procedure for more information on `REGISTERSTORE` procedure.

4. Mount the store using `DBMS_DBFS_CONTENT.MOUNTSTORE`.

> **See Also:**
>
> MOUNTSTORE Procedure for more information on `MOUNTSTORE` procedure.

20.3 Using the Hierarchical Store

You can use the Hierarchical Store as an independent file system or as an archive solution for SecureFile LOBs.

- Using Hierarchical Store as a File System
  Use the `DBMS_DBFS_CONTENT` package to create, update, read, and delete file system entries in the store.

- Using Hierarchical Store as an Archive Solution For SecureFiles LOBs
  Use the `DBMS_LOB` package to archive SecureFiles LOBs in a tape or an S3 store.
- Dropping a Hierarchical Store
  You can drop a hierarchical store.
- Compression to Use with the Hierarchical Store
  The DBFS hierarchical store can store its files in compressed forms.
- Program Example Using Tape
  This example program configures and uses a tape store.
- Program Example Using Amazon S3
  This example program configures and uses an Amazon S3 store.

20.3.1 Using Hierarchical Store as a File System

Use the `DBMS_DBFS_CONTENT` package to create, update, read, and delete file system entries in the store.

> **See Also:**
>
> DBFS Content API

20.3.2 Using Hierarchical Store as an Archive Solution For SecureFiles LOBs

Use the `DBMS_LOB` package to archive SecureFiles LOBs in a tape or an S3 store.

The `DBMS_LOB` package archives SecureFiles LOBs in a tape or an S3 store. Use the following method to free space in the cache or to force cache resident contents to be written to an external storage device:

```
DBMS_DBFS_HS.storePush(store_name);
```

20.3.3 Dropping a Hierarchical Store

You can drop a hierarchical store.

To drop a hierarchical store, call:

```
DBMS_DBFS_HS.dropStore(store_name, opt_flags);
```

20.3.4 Compression to Use with the Hierarchical Store

The DBFS hierarchical store can store its files in compressed forms.

The DBFS hierarchical store has the ability to store its files in compressed form using the `SETPROPERTY` method and the property `PROPNAME_COMPRESSLVL` to specify the compression level.

Valid values are:

- `PROPVAL_COMPLVL_NONE`: No compression

- PROPVAL_COMPLVL_LOW: LOW compression
- PROPVAL_COMPLVL_MEDIUM: MEDIUM compression
- PROPVAL_COMPLVL_HIGH: HIGH compression

Generally, the compression level LOW performs best and still provides a good compression ratio. Compression levels MEDIUM and HIGH provide significantly better compression ratios, but compression times can be correspondingly longer. Oracle recommends using NONE or LOW when write performance is critical, such as when files in the DBFS HS store are updated frequently. If space is critical and the best possible compression ratio is desired, use MEDIUM or HIGH.

Files are compressed as they are paged out of the cache into the staging area (before they are subsequently pushed into the back end tape or S3 storage). Therefore, compression also benefits by storing smaller files in the staging area and effectively increasing the total available capacity of the staging area.

20.3.5 Program Example Using Tape

This example program configures and uses a tape store.

In the example, you must substitute valid values in some places, as indicated by <...>, for the program to run successfully.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* DBMS_DBFS_HS documentation for complete details about the methods and their parameters

```
Rem Example to configure and use a Tape store.
Rem
Rem hsuser should be a valid database user who has been granted
Rem the role dbfs_role.

connect hsuser/hsuser

Rem The following block sets up a STORETYPE_TAPE store with
Rem DBMS_DBFS_HS acting as the store provider.

declare
storename varchar2(32) ;
tblname varchar2(30) ;
tbsname varchar2(30) ;
lob_cache_quota number := 0.8 ;
cachesz number ;
ots number ;
begin
cachesz := 50 * 1048576 ;
ots := 1048576 ;
storename := 'tapestore10' ;
tblname := 'tapetbl10' ;
tbsname := '<TBS_3>' ; -- Substitute a valid tablespace name

-- Create the store.
-- Here tbsname is the tablespace used for the store,
-- tblname is the table holding all the store entities,
```

```
-- cachesz is the space used by the store to cache content
--   in the tablespace,
-- lob_cache_quota is the fraction of cachesz allocated
--   to level-1 cache and
-- ots is minimum amount of content that is accumulated
--   in level-2 cache before being stored on tape
dbms_dbfs_hs.createStore(
  storename,
  dbms_dbfs_hs.STORETYPE_TAPE,
  tblname, tbsname, cachesz,
  lob_cache_quota, ots) ;

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_SBTLIBRARY,
  '<ORACLE_HOME/work/libobkuniq.so>') ;
  -- Substitute your ORACLE_HOME path

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_MEDIAPOOL,
  '<0>') ;  -- Substitute valid value

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_COMPRESSLEVEL,
  'NONE') ;

-- Please refer to DBMS_DBFS_CONTENT documentation
-- for details about this method
dbms_dbfs_content.registerstore(
  storename,
  'tapeprvder10',
  'dbms_dbfs_hs') ;

-- Please refer to DBMS_DBFS_CONTENT documentation
-- for details about this method
dbms_dbfs_content.mountstore(storename, 'tapemnt10') ;
end ;
/

Rem The following code block does file operations
Rem using DBMS_DBFS_CONTENT on the store configured
Rem in the previous code block

connect hsuser/hsuser

declare
  path varchar2(256) ;
  path_pre varchar2(256) ;
  mount_point varchar2(32) ;
  store_name varchar2(32) ;
  prop1 dbms_dbfs_content_properties_t ;
  prop2 dbms_dbfs_content_properties_t ;
  mycontent blob := empty_blob() ;
  buffer varchar2(1050) ;
  rawbuf raw(1050) ;
  outcontent blob := empty_blob() ;
  itemtype integer ;
  pflag integer ;
  filecnt integer ;
  iter integer ;
```

```
  offset integer ;
  rawlen integer ;
begin

  mount_point := '/tapemnt10' ;
  store_name := 'tapestore10' ;
  path_pre := mount_point ||'/file' ;


-- We create 10 empty files in the following loop
  filecnt := 0 ;
  loop
    exit when filecnt = 10 ;
    path := path_pre || to_char(filecnt) ;
    mycontent := empty_blob() ;
    prop1 := null ;

    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.createFile(
      path, prop1, mycontent) ; -- Create the file

    commit ;
    filecnt := filecnt + 1 ;
  end loop ;

  -- We populate the newly created files with content
  -- in the following loop
  pflag := dbms_dbfs_content.prop_data +
           dbms_dbfs_content.prop_std  +
           dbms_dbfs_content.prop_opt  ;

  buffer := 'Oracle provides an integrated management '  ||
            'solution for managing Oracle database with '||
            'a unique top-down application management '  ||
            'approach. With new self-managing '          ||
            'capabilities, Oracle eliminates time-'      ||
            'consuming, error-prone administrative '     ||
            'tasks, so database administrators can '     ||
            'focus on strategic business objectives '    ||
            'instead of performance and availability '   ||
            'fire drills. Oracle Management Packs for '  ||
            'Database provide signifiCant cost and time-'||
            'saving capabilities for managing Oracle '   ||
            'Databases. Independent studies demonstrate '||
            'that Oracle Database is 40 percent easier ' ||
            'to manage over DB2 and 38 percent over '    ||
            'SQL Server.';

  rawbuf := utl_raw.cast_to_raw(buffer) ;
  rawlen := utl_raw.length(rawbuf) ;
  offset := 1 ;
  filecnt := 0 ;
  loop
    exit when filecnt = 10 ;
    path := path_pre || to_char(filecnt) ;
    prop1 := null;

    -- Append buffer to file
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.putpath(
```

```
    path, prop1, rawlen,
    offset, rawbuf) ;

  commit ;
  filecnt := filecnt + 1 ;
end loop ;

-- Clear out level 1 cache
dbms_dbfs_hs.flushCache(store_name) ;
commit ;

-- Do write operation on even-numbered files.
-- Do read operation on odd-numbered files.
filecnt := 0 ;
loop
  exit when filecnt = 10;
  path := path_pre || to_char(filecnt) ;
  if mod(filecnt, 2) = 0 then
    -- Get writable file
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.getPath(
      path, prop2, outcontent, itemtype,
      pflag, null, true) ;

    buffer := 'Agile businesses want to be able to '    ||
              'quickly adopt new technologies, whether '||
              'operating systems, servers, or '         ||
              'software, to help them stay ahead of '   ||
              'the competition. However, change often ' ||
              'introduces a period of instability into '||
              'mission-critical IT systems. Oracle '    ||
              'Real Application Testing-with Oracle '   ||
              'Database 11g Enterprise Edition-allows ' ||
              'businesses to quickly adopt new '        ||
              'technologies while eliminating the '     ||
              'risks associated with change. Oracle '   ||
              'Real Application Testing combines a '    ||
              'workload capture and replay feature '    ||
              'with an SQL performance analyzer to '    ||
              'help you test changes against real-life '||
              'workloads, and then helps you fine-tune '||
              'the changes before putting them into'    ||
              'production. Oracle Real Application '    ||
              'Testing supports older versions of '     ||
              'Oracle Database, so customers running '  ||
              'Oracle Database 9i and Oracle Database ' ||
              '10g can use it to accelerate their '     ||
              'database upgrades. ';

    rawbuf := utl_raw.cast_to_raw(buffer) ;
    rawlen := utl_raw.length(rawbuf) ;

    -- Modify file content
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_lob.write(outcontent, rawlen, 10, rawbuf);
    commit ;
  else
    -- Read the file
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
```

```
      dbms_dbfs_content.getPath(
        path, prop2, outcontent, itemtype, pflag) ;
    end if ;
    filecnt := filecnt + 1 ;
  end loop ;

  -- Delete the first 2 files
  filecnt := 0;

  loop
    exit when filecnt = 2 ;
    path := path_pre || to_char(filecnt) ;
    -- Delete file
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.deleteFile(path) ;
    commit ;
    filecnt := filecnt + 1 ;
  end loop ;


  -- Move content staged in database to the tape store
  dbms_dbfs_hs.storePush(store_name) ;
  commit ;

end ;
/
```

20.3.6 Program Example Using Amazon S3

This example program configures and uses an Amazon S3 store.

Valid values must be substituted in some places, indicated by <...>, for the program to run successfully.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* `DBMS_DBFS_HS` documentation for complete details about the methods and their parameters

```
Rem Example to configure and use an Amazon S3 store.
Rem
Rem hsuser should be a valid database user who has been granted
Rem the role dbfs_role.

connect hsuser/hsuser

Rem The following block sets up a STORETYPE_AMAZONS3 store with
Rem DBMS_DBFS_HS acting as the store provider.

declare
storename varchar2(32) ;
tblname varchar2(30) ;
tbsname varchar2(30) ;
lob_cache_quota number := 0.8 ;
cachesz number ;
ots number ;
begin
```

```
cachesz := 50 * 1048576 ;
ots := 1048576 ;
storename := 's3store10' ;
tblname := 's3tbl10' ;
tbsname := '<TBS_3>' ; -- Substitute a valid tablespace name

-- Create the store.
-- Here tbsname is the tablespace used for the store,
-- tblname is the table holding all the store entities,
-- cachesz is the space used by the store to cache content
--   in the tablespace,
-- lob_cache_quota is the fraction of cachesz allocated
--   to level-1 cache and
-- ots is minimum amount of content that is accumulated
--   in level-2 cache before being stored in AmazonS3
dbms_dbfs_hs.createStore(
  storename,
  dbms_dbfs_hs.STORETYPE_AMAZONS3,
  tblname, tbsname, cachesz,
  lob_cache_quota, ots) ;

dbms_dbfs_hs.setstoreproperty(storename,
  dbms_dbfs_hs.PROPNAME_SBTLIBRARY,
  '<ORACLE_HOME/work/libosbws11.so>');
  -- Substitute your ORACLE_HOME path

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_S3HOST,
  's3.amazonaws.com') ;

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_BUCKET,
  'oras3bucket10') ;

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_WALLET,
  'LOCATION=file:<ORACLE_HOME>/work/wlt CREDENTIAL_ALIAS=a_key') ;
  -- Substitute your ORACLE_HOME path

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_LICENSEID,
  '<xxxxxxxxxxxxxxxx>') ; -- Substitute a valid SBT license id

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_HTTPPROXY,
  '<http://www-proxy.mycompany.com:80/>') ;
  -- Substitute valid value. If a proxy is not used,
  -- then this property need not be set.

dbms_dbfs_hs.setstoreproperty(
  storename,
  dbms_dbfs_hs.PROPNAME_COMPRESSLEVEL,
  'NONE') ;

dbms_dbfs_hs.createbucket(storename) ;

-- Please refer to DBMS_DBFS_CONTENT documentation
```

```
-- for details about this method
dbms_dbfs_content.registerstore(
  storename,
  's3prvder10',
  'dbms_dbfs_hs') ;

-- Please refer to DBMS_DBFS_CONTENT documentation
-- for details about this method
dbms_dbfs_content.mountstore(
  storename,
  's3mnt10') ;
end ;
/

Rem The following code block does file operations
Rem using DBMS_DBFS_CONTENT on the store configured
Rem in the previous code block

connect hsuser/hsuser

declare
path varchar2(256) ;
path_pre varchar2(256) ;
mount_point varchar2(32) ;
store_name varchar2(32) ;
prop1 dbms_dbfs_content_properties_t ;
prop2 dbms_dbfs_content_properties_t ;
mycontent blob := empty_blob() ;
buffer varchar2(1050) ;
rawbuf raw(1050) ;
outcontent blob := empty_blob() ;
itemtype integer ;
pflag integer ;
filecnt integer ;
iter integer ;
offset integer ;
rawlen integer ;
begin

  mount_point := '/s3mnt10' ;
  store_name := 's3store10' ;
  path_pre := mount_point ||'/file' ;

  -- We create 10 empty files in the following loop
  filecnt := 0 ;
  loop
    exit when filecnt = 10 ;
    path := path_pre || to_char(filecnt) ;
    mycontent := empty_blob() ;
    prop1 := null ;

    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.createFile(
      path, prop1, mycontent) ; -- Create the file

    commit ;
    filecnt := filecnt + 1 ;
  end loop ;

  -- We populate the newly created files with content
  -- in the following loop
```

```
pflag := dbms_dbfs_content.prop_data +
         dbms_dbfs_content.prop_std  +
         dbms_dbfs_content.prop_opt  ;

buffer := 'Oracle provides an integrated management '  ||
          'solution for managing Oracle database with '||
          'a unique top-down application management '  ||
          'approach. With new self-managing '          ||
          'capabilities, Oracle eliminates time-'      ||
          'consuming, error-prone administrative '     ||
          'tasks, so database administrators can '     ||
          'focus on strategic business objectives '    ||
          'instead of performance and availability '   ||
          'fire drills. Oracle Management Packs for '  ||
          'Database provide signifiCant cost and time-'||
          'saving capabilities for managing Oracle '   ||
          'Databases. Independent studies demonstrate '||
          'that Oracle Database is 40 percent easier ' ||
          'to manage over DB2 and 38 percent over '    ||
          'SQL Server.';

rawbuf := utl_raw.cast_to_raw(buffer) ;
rawlen := utl_raw.length(rawbuf) ;
offset := 1 ;
filecnt := 0 ;
loop
  exit when filecnt = 10 ;
  path := path_pre || to_char(filecnt) ;
  prop1 := null;

  -- Append buffer to file
  -- Please refer to DBMS_DBFS_CONTENT documentation
  -- for details about this method
  dbms_dbfs_content.putpath(
    path, prop1, rawlen,
    offset, rawbuf) ;

  commit ;
  filecnt := filecnt + 1 ;
end loop ;

-- Clear out level 1 cache
dbms_dbfs_hs.flushCache(store_name) ;
commit ;

-- Do write operation on even-numbered files.
-- Do read operation on odd-numbered files.
filecnt := 0 ;
loop
  exit when filecnt = 10;
  path := path_pre || to_char(filecnt) ;
  if mod(filecnt, 2) = 0 then
    -- Get writable file
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.getPath(
      path, prop2, outcontent, itemtype,
      pflag, null, true) ;

    buffer := 'Agile businesses want to be able to '    ||
              'quickly adopt new technologies, whether '||
              'operating systems, servers, or '         ||
```

```
                'software, to help them stay ahead of '   ||
                'the competition. However, change often ' ||
                'introduces a period of instability into '||
                'mission-critical IT systems. Oracle '    ||
                'Real Application Testing-with Oracle '   ||
                'Database 11g Enterprise Edition-allows ' ||
                'businesses to quickly adopt new '        ||
                'technologies while eliminating the '     ||
                'risks associated with change. Oracle '   ||
                'Real Application Testing combines a '    ||
                'workload capture and replay feature '    ||
                'with an SQL performance analyzer to '    ||
                'help you test changes against real-life '||
                'workloads, and then helps you fine-tune '||
                'the changes before putting them into'    ||
                'production. Oracle Real Application '    ||
                'Testing supports older versions of '     ||
                'Oracle Database, so customers running '  ||
                'Oracle Database 9i and Oracle Database ' ||
                '10g can use it to accelerate their '     ||
                'database upgrades. ';

      rawbuf := utl_raw.cast_to_raw(buffer) ;
      rawlen := utl_raw.length(rawbuf) ;

      -- Modify file content
      -- Please refer to DBMS_DBFS_CONTENT documentation
      -- for details about this method
      dbms_lob.write(outcontent, rawlen, 10, rawbuf);
      commit ;
    else
      -- Read the file
      -- Please refer to DBMS_DBFS_CONTENT documentation
      -- for details about this method
      dbms_dbfs_content.getPath(
        path, prop2, outcontent, itemtype, pflag) ;
    end if ;
    filecnt := filecnt + 1 ;
  end loop ;

  -- Delete the first 2 files
  filecnt := 0;

  loop
    exit when filecnt = 2 ;
    path := path_pre || to_char(filecnt) ;
    -- Delete file
    -- Please refer to DBMS_DBFS_CONTENT documentation
    -- for details about this method
    dbms_dbfs_content.deleteFile(path) ;
    commit ;
    filecnt := filecnt + 1 ;
  end loop ;


  -- Move content staged in database to Amazon S3 store
  dbms_dbfs_hs.storePush(store_name) ;
  commit ;

end ;
/
```

20.4 The DBMS_DBFS_HS Package

The `DBMS_DBFS_HS` package is a service provider that enables use of tape or Amazon S3 Web service as storage for data.

- Constants for DBMS_DBFS_HS Package
  The `DBMS_DBFS_HS` PL/SQL package constants are very detailed.
- Methods for DBMS_DBFS_HS Package
  There are many methods in the `DBMS_DBFS_HS`package.

20.4.1 Constants for DBMS_DBFS_HS Package

The `DBMS_DBFS_HS` PL/SQL package constants are very detailed.

> **See Also:**
>
> See *Oracle Database PL/SQL Packages and Types Reference* for details of constants used by `DBMS_DBFS_HS` PL/SQL package

20.4.2 Methods for DBMS_DBFS_HS Package

There are many methods in the `DBMS_DBFS_HS`package.

Table 20-1 summarizes the `DBMS_DBFS_HS` PL/SQL package methods.

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

Table 20-1 Methods of the DBMS_DBFS_HS PL/SQL Packages

Method	Description
`CLEANUPUNUSEDBACKUPFILES`	Removes files that are created on the external storage device if they have no current content. *Oracle Database PL/SQL Packages and Types Reference*
`CREATEBUCKET`	Creates an AWS bucket, for use with the `STORETYPE_AMAZON3` store. *Oracle Database PL/SQL Packages and Types Reference*
`CREATESTORE`	Creates a DBFS HS store. *Oracle Database PL/SQL Packages and Types Reference*
`DEREGSTORECOMMAND`	Removes a command (message) that was associated with a store. *Oracle Database PL/SQL Packages and Types Reference*
`DROPSTORE`	Deletes a previously created DBFS HS store. *Oracle Database PL/SQL Packages and Types Reference*

Table 20-1 (Cont.) Methods of the DBMS_DBFS_HS PL/SQL Packages

Method	Description
FLUSHCACHE	Flushes out level 1 cache to level 2 cache, increasing space in level 1. *Oracle Database PL/SQL Packages and Types Reference*
GETSTOREPROPERTY	Retrieves the values of a property of a store in the database. *Oracle Database PL/SQL Packages and Types Reference*
RECONFIGCACHE	Reconfigures the parameters of the database cache used by the store. *Oracle Database PL/SQL Packages and Types Reference*
REGISTERSTORECOMMAND	Registers commands (messages) for a store so they are sent to the Media Manager of an external storage device. *Oracle Database PL/SQL Packages and Types Reference* .
SENDCOMMAND	Sends a command (message) to the Media Manager of an external storage device. *Oracle Database PL/SQL Packages and Types Reference*
SETSTOREPROPERTY	Associates name/value properties with a registered Hierarchical Store. *Oracle Database PL/SQL Packages and Types Reference*
STOREPUSH	Pushes locally cached data to an archive store. *Oracle Database PL/SQL Packages and Types Reference*

20.5 Views for DBFS Hierarchical Store

The DBFS Hierarchical Stores have several types of views.

- DBA Views
  There are several views available for DBFS Hierarchical Store.
- User Views
  There are several views available for the DBFS Hierarchical Store.

> **See Also:**
>
> *Oracle Database Reference* for the columns and data types of these views

20.5.1 DBA Views

There are several views available for DBFS Hierarchical Store.

Following are the views available for DBFS Hierarchical Store:

- DBA_DBFS_HS

  This view shows all Database File System (DBFS) hierarchical stores
- DBA_DBFS_HS_PROPERTIES

  This view shows modifiable properties of all Database File System (DBFS) hierarchical stores.
- DBA_DBFS_HS_FIXED_PROPERTIES

This view shows non-modifiable properties of all Database File System (DBFS) hierarchical stores.

- `DBA_DBFS_HS_COMMANDS`

  This view shows all the registered store commands for all Database File System (DBFS) hierarchical stores.

20.5.2 User Views

There are several views available for the DBFS Hierarchical Store.

- `USER_DBFS_HS`

  This view shows all Database File System (DBFS) hierarchical stores owned by the current user.

- `USER_DBFS_HS_PROPERTIES`

  This view shows modifiable properties of all Database File System (DBFS) hierarchical stores owned by current user.

- `USER_DBFS_HS_FIXED_PROPERTIES`

  This view shows non-modifiable properties of all Database File System (DBFS) hierarchical stores owned by current user.

- `USER_DBFS_HS_COMMANDS`

  This view shows all the registered store commands for all Database File system (DBFS) hierarchical stores owned by current user.

- `USER_DBFS_HS_FILES`

  This view shows files in the Database File System (DBFS) hierarchical store owned by the current user and their location on the backend device.