Changes in Oracle Database

The following are the changes in *SecureFiles and Large Objects Developer's Guide* for Oracle Database 23ai.

- New Features in Release 23ai
- Deprecated Features in Release 23ai

New Features in Release 23ai

The following are the new features in the *SecureFiles and Large Objects Developer's Guide* for Oracle Database Release 23ai.

- Automatic SecureFiles Shrink
  Automatic SecureFiles Shrink automatically selects suitable SecureFiles LOB segments and shrinks the selected segment in the background.
- Distributed and Sharded Environments Support Additional Types of LOBs
  Earlier, persistent LOBs and temporary LOBs were supported in distributed and sharded environments.
- Estimate the Space Saved with Deduplication
  Before you enable deduplication, use the `GET_LOB_DEDUPLICATION_RATIO` function to estimate the space that you can save by enabling this feature for existing LOBs.
- Improved Performance of LOB Writes
  You can experience improved LOB read and write performance.
- Maximum Size of Inline LOBs is 8000
  Actual LOB values are stored either in the table row (inline) or outside of the table row (out-of-line).
- Migrate BasicFile LOBs Using the SecureFiles Migration Utility
  Use the SecureFiles Migration Utility to simplify the migration and compression of BasicFiles LOB segments to SecureFiles LOB segments.
- Rename LOB Segments
  From release 23ai onwards, you can use the `ALTER TABLE RENAME LOB` statement to rename LOB segments, and partitions, as well as subpartitions, in partitioned tables.
- Tune Performance for Parallel File System Operations
  Tune performance in environments that contain many PDBs and require multiple `DBMS_FS` requests to be processed in parallel.
- Value LOBs Optimize Reading LOB Values in a SQL Query
  Value LOBs, are a subset of Temporary LOBs, which are valid for a SQL fetch duration.

Automatic SecureFiles Shrink

Automatic SecureFiles Shrink automatically selects suitable SecureFiles LOB segments and shrinks the selected segment in the background.

With Automatic SecureFiles Shrink, the shrink operation happens transparently in small and gradual steps over time while allowing DDL and DML statements to execute concurrently. In the manual method, you must decide on which LOB segments to shrink using tools like

Segment Advisor, and use a DDL statement to execute the shrink operation. The manual method may not be feasible for very large LOB segments because it is time-consuming.

Automatic SecureFiles Shrink simplifies administrator duties and saves time due to the automation of this process. See Automatic SecureFiles Shrink.

Distributed and Sharded Environments Support Additional Types of LOBs

Earlier, persistent LOBs and temporary LOBs were supported in distributed and sharded environments.

You can now work with inline LOBs, value LOBs, and all temporary LOBs in distributed and sharded environments. Avail good performance, scalability, and garbage collection when you work with temporary LOBs. See Distributed LOBs.

Estimate the Space Saved with Deduplication

Before you enable deduplication, use the `GET_LOB_DEDUPLICATION_RATIO` function to estimate the space that you can save by enabling this feature for existing LOBs.

This enables you to take an informed decision to enable deduplication. See ALTER TABLE with Advanced LOB Deduplication.

Improved Performance of LOB Writes

You can experience improved LOB read and write performance.

The following enhancements improve the LOB read and write performance:

- Multiple LOBs in a single transaction are buffered simultaneously. This improves performance when you use switch between LOBs while writing within a single transaction.
- Various enhancements, such as acceleration of compressed LOB append and compression unit caching, improve the performance of reads and writes to compressed LOBs.
- The input-output buffer is resized based on the input data for large writes to LOBs with the NOCACHE option. This improves the performance for large direct writes, such as writes to file systems on DBFS and OFS.

Maximum Size of Inline LOBs is 8000

Actual LOB values are stored either in the table row (inline) or outside of the table row (out-of-line).

Now, the maximum size of the inline LOB is 8000. Earlier, the maximum size was 4000. This provides better input-output performance while processing LOB columns. You can experience the improved performance while running operations, such as full table scans, range scans, and DML. See Summary of CREATE TABLE LOB Storage Parameters for Securefile LOBs.

Migrate BasicFile LOBs Using the SecureFiles Migration Utility

Use the SecureFiles Migration Utility to simplify the migration and compression of BasicFiles LOB segments to SecureFiles LOB segments.

Earlier it was challenging to decide which BasicFile LOBs to migrate to SecureFile LOBs, and whether or not to compress the LOBs, especially considering that organizations often have

many databases, with a large numbers of schemas, tables, and segments. SecureFiles Migration Utility automates several steps that were earlier performed manually. It also generates several reports that help you decide which BasicFile LOBs you want to migrate and compress. This is the recommended method to migrate BasicFile LOB data to SecureFile storage. See Migrating LOBs with SecureFiles Migration Utility.

Rename LOB Segments

From release 23ai onwards, you can use the `ALTER TABLE RENAME LOB` statement to rename LOB segments, and partitions, as well as subpartitions, in partitioned tables.

Example 1 Command Syntax

```
ALTER TABLE RENAME LOB (<column_name>) <oldsegment_name> to <newsegment_name>;


ALTER TABLE RENAME LOB (<column_name>) partition <oldsegment_name> to
newsegment_name;


ALTER TABLE RENAME LOB (<column_name>) subpartition <oldsegment_name> to
newsegment_name;
```

Ensure that new name that you provide for the segment is unique within the database. In case of any conflicts, the database returns `ORA-64223` or `ORA-64223`. To check the names of the existing segments, you can query LOB views, such as `all_lobs`, `all_lobs_partition`, or `all_lob_subpartitions`.

See ALTER TABLE BNF.

Tune Performance for Parallel File System Operations

Tune performance in environments that contain many PDBs and require multiple `DBMS_FS` requests to be processed in parallel.

You can update the number `OFS_THREADS` to increase the number of `DBMS_FS` requests that are executed in parallel. This increases the number of worker threads executing the make, mount, unmount, and destroy operations on Oracle file systems in the Oracle database.

Increase in the value of `OFS_THREADS`, results in a significant reduction of time taken to execute parallel file system requests in environments that contain multiple PDBs. You can query the `V$OFS_THREADS` view to list all the running OFS threads and to retrieve details about the different OFS threads. See Views for OFS.

Value LOBs Optimize Reading LOB Values in a SQL Query

Value LOBs, are a subset of Temporary LOBs, which are valid for a SQL fetch duration.

Use Value LOBs to optimize reading LOB values in the context of a SQL query. Many applications use LOBs to store medium-sized objects, about a few mega-bytes in size, and just want to read the LOB value in the context of a SQL query.

Value LOBs are autonomous, read-only and more performant. A value LOB, in most instances, provides faster performance than a reference LOB. Oracle highly recommends that you use value LOBs if your application fetches a LOB for read purposes as part of a SQL query and consumes the LOB data before the next fetch is performed on the cursor.

A value LOB gets automatically freed when the next fetch for a cursor is performed. This prevents accumulation of temporary LOBs, which translates to better performance and scalability of the query. In the case of temporary LOBs, it has always been the user's responsibility to free the temporary LOB when their application is done processing it.

A SQL function for JSON can also return a value LOB. See Value LOBs.

Deprecated Features in Release 23ai

The following are the deprecated features in the *SecureFiles and Large Objects Developer's Guide* for Oracle Database Release 23ai.

- Deprecation of the mkstore Wallet Management Tool
- Deprecation of Enterprise User Security (EUS)
- Deprecation of Oracle JDBC Proprietary BLOB/CLOB Open and Close Methods

Deprecation of the mkstore Wallet Management Tool

The `mkstore` wallet management command line tool is deprecated with Oracle Database 23ai, and can be removed in a future release.

Deprecation of Enterprise User Security (EUS)

Enterprise User Security (EUS) is deprecated with Oracle Database 23ai.

Deprecation of Oracle JDBC Proprietary BLOB/CLOB Open and Close Methods

The Oracle JDBC methods `open()`, `close()`, and `isClosed()` in `OracleBlob`, `OracleClob`, and `OracleBfile` are deprecated for removal in Oracle Database 23ai.

Oracle is deprecating these methods, which are replaced with `openLob()`, `closeLob()` and `isClosedLob()`. The method `close()` conflicts with the interface type `java.lang.Autocloseable`. Removing the proprietary method `close()` makes it possible for `java.lang OracleBlob`, `java.lang.OracleClob`, and `java.lang.OracleBfile` to extend the Autocloseable interface at some future time. The `open()` and `isClosed()` methods will be removed and replaced in a future release to maintain rational names for these methods.