19c ORACLE Database

# Oracle SecureFiles

ORACLE WHITE PAPER | FEBRUARY 2019

ORACLE

## Table of Contents

## Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Introduction

The nature of critical business information is changing rapidly and is no longer limited to structured text or numeric data. Unstructured content – images, audio, video, PDF, Word documents etc – is becoming pervasive. Web-based applications are forcing organizations to capture and manage more unstructured data such as photos, videos, and news clips more than ever before. Regulatory changes like Sarbanes-Oxley and HIPAA are further fueling this trend.

Mainstream business applications have to integrate both relational or structured and unstructured content to provide users with a seamless and richer user experience. Traditionally, unstructured content has been stored in file systems due to their simplicity and performance. This choice forces organizations to compromise Oracle's security, scalability, transaction capability, read consistency and high availability for file or unstructured data. However, this tradeoff comes at a cost, often prohibitive.

Different storage mechanisms for structured and unstructured data introduces disjoint security/audit models, difficulty in searching across all related enterprise content and disparate IT management procedures for backup and recovery – which can ultimately lead to lower ROI. SecureFiles is a feature of Oracle Database specifically engineered to deliver high performance for file or unstructured data comparable to that of traditional file systems while retaining the advantages of the Oracle Database.

SecureFiles removes the need for compromise by offering the 'best-of-both-worlds' architecture for storing unstructured content. SecureFiles is designed as a superset of the ANSI standard for large objects (LOBs) and offers easy migration from old LOBs or BasicFiles. Applications can transparently take advantage of industry-standard encryption for added security and intelligent storage features like compression and deduplication, for increased space savings and faster performance.

With SecureFiles, organizations can now manage all relational data and associated file data in Oracle using a single security/audit model, a unified backup and recovery process, and perform seamless search across all information -- making Oracle Database the ideal solution for storing both structured, or relational data, and related unstructured data.

## Best Storage for File Data

Oracle Database provides SecureFiles to eliminate the distinction between structured and unstructured content storage. With SecureFiles, Oracle has perfected the use of the database for storage of all enterprise content.

SecureFiles represents the core infrastructure in the Oracle Database for unstructured content management – the SecureFiles architecture includes: New disk formats for LOB data blocks and meta-data blocks, network protocol, space management, redo and undo formats, buffer caching, and intelligent I/O subsystem. SecureFiles delivers substantially improved performance along with optimized storage for unstructured data inside the Oracle Database.

**Note:** While LOBs from Oracle Database 10g, and prior releases, are still supported using 'BasicFiles', this paper provides an overview of SecureFiles and some examples on how to use this feature.

## SecureFiles Overview

SecureFiles includes numerous architectural enhancements for greatly improved performance, scalability, efficient storage and easier manageability. Some of the innovations are listed below.

### Write-Gather Cache (WGC)

SecureFiles WGC buffers data typically up to 4MB during write operations before flushing or committing to the underlying storage layer. This buffering of in-flight data allows for large contiguous space allocation and large disk I/O. Write performance is greatly improved due to reduced disk seek costs.

The WGC is maintained on a per-transaction basis. Only one WGC is used for all SecureFiles in a transaction. Oracle automatically determines if the WGC needs to be flushed to disk before the 4MB limit is reached. A 'commit' by the user also causes flushing of the WGC to disk.

**Note:** The WGC size is 4MB and is not a user tunable parameter.

### Space Management

Reclamation of space is done in the background. Space management is greatly improved by separately managing the committed free space and uncommitted free space. Thus, SecureFiles do not suffer from high water mark contention during space reclamation, unlike older LOBs.

SecureFiles segments require tablespaces managed with automatic segment space management (ASSM). Space management for SecureFiles is optimized for storing file data inside the Oracle Database. An intelligent space manager provides fast allocation of large, contiguous disk blocks and efficient deletion by automatically reclaiming freed space. Space is also pre-allocated based on heuristics of recent demand and this ensures no stalling and consequently no impact on foreground performance. Space management is designed to be self-adaptable and uses statistics for efficient memory and space allocation.

### Reduced Fragmentation

SecureFiles, unlike BasicFiles or older LOBs, uses a dynamic CHUNK (one or more contiguous Oracle blocks) size setting to maximize contiguous allocations on disk and reduce fragmentation. The user-specified CHUNK value is used as a suggestion, along with other factors such as size of the SecureFile and the availability of space in the segment, to determine the optimal CHUNK size. SecureFiles CHUNK is completely an internal concept.

A fixed CHUNK size setting causes internal fragmentation if it is too large and poor write performance if it is too small. With a dynamic CHUNK size setting, large regions of disk can be allocated and deallocated in a single operation. SecureFiles is thus designed from the ground up to be a high performance and storage efficient solution for unstructured or file data of all sizes.

### Intelligent Pre-Fetching

Read performance is enhanced by read-ahead or pre-fetching data from disk while data is sent over the network. SecureFiles keeps track of access patterns and intelligently pre-fetches data before the request is actually made. Read latency is reduced by the overlap of the network roundtrip with the disk I/O for the pre-fetched data and throughput is greatly increased.

### Network Layer

SecureFiles client/server network layer allows for high-speed data transfer between the client and server. This allows for reading and writing bulk data directly from the network socket without the need for any temporary staging. This provides for significantly higher read/write performance.

### No LOB Index Contention

Older LOBs use a global per-segment B+ tree index to maintain the inodes, which are data structures that map the logical offsets to the physical disk blocks.  The global B+ tree structure is used for both access and navigation into the LOBs, and for free space management in LOB segment. This causes contention and significant performance degradation in highly concurrent environments.

Unlike older LOBs, SecureFiles do not use the LOB index to manage such meta-data. Instead, they use "private" meta-data blocks that are co-located with the data blocks in the LOB segment.  This design removes the LOB index contention that existed with BasicFile LOBs and greatly improves performance, especially under real world insert-delete-reclaim situations.

**Easier Manageability**

SecureFiles feature self-tuning and intelligent space and memory management algorithms and consequently require lesser number of user-tuned parameters. Specifically, FREEPOOLS, FREELISTS, FREELIST GROUPS & PCTVERSION no longer need to be specified and are ignored for SecureFiles.

Not only are some of these parameters difficult to tune for unpredictable spikes in workloads, but also cannot be changed online. SecureFiles maintains internal statistics to self-tune the space management algorithms and ensures high performance and scalability under diverse workloads.

## SecureFiles Storage Parameters

1. ENABLE STORAGE IN ROW (Default) -
   (Other value would be DISABLE STORAGE IN ROW)
2. NOLOGGING
3. CACHE
4. NOCACHE (Default)
5. CACHE READS
6. LOGGING (Default)
7. FILESYSTEM_LIKE_LOGGING

Parameters like CACHE/NOCACHE are irrelevant for inline LOBs since such LOBs are stored in heap table segment blocks, and are accessed via the regular mechanism just like other non-LOB data in heap blocks. These parameters have an impact only on out-of-line LOBS, i.e. LOBs stored in LOB segment.

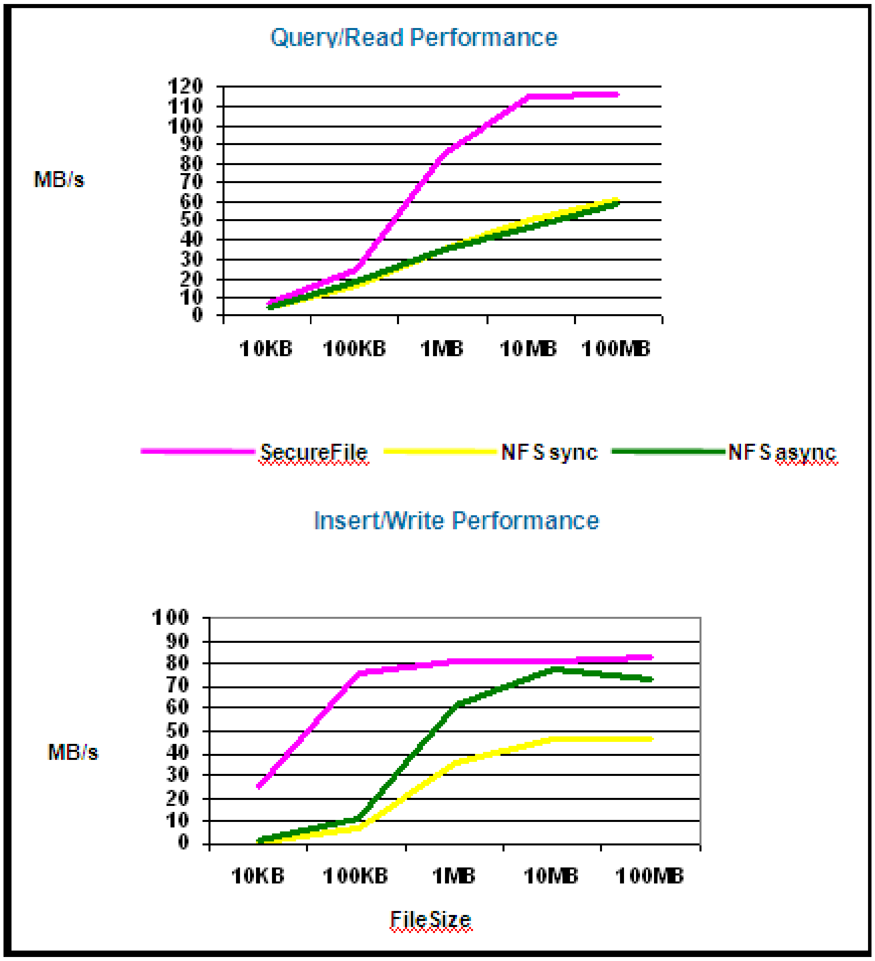
Similarly, LOGGING, NOLOGGING are irrelevant for inline LOBs. Changes to such LOBs are always accompanied by corresponding redo. These parameters have an impact only on out-of-line LOBS, i.e. LOBs stored in LOB segment.

For SecureFiles, NOLOGGING is implicitly converted to FILESYSTEM_LIKE_LOGGING, and this gives a behavior semantically equivalent to meta-data only journaling file systems.

## Superior Performance

The performance of SecureFiles under both queries and inserts was compared to that of Linux NFS/Ext3. The test application simulates a real world DICOM application consisting of patient metadata and DICOM images. Oracle compared the performance of SecureFiles to that of the NFSv3 file system. In both cases, metadata is stored in an Oracle Database. In the case of file system, the DICOM images are stored on file servers that are accessed from a client machine over NFSv3. In the case of SecureFiles, the metadata as well as the DICOM images are stored inside an Oracle Database.

In the tests, logging for SecureFiles is set to FILESYSTEM_LIKE_LOGGING mode to keep the logging level and functionality comparable to that of the file system. The tests compare the performance of NFSv3 in both the async and sync modes.



SecureFile outperformed the NFSv3 access for all sizes for both read and write operations. Gains for the smaller file sizes are also due to reduced roundtrips where metadata and data is accessed in one roundtrip unlike the NFSv3 case where metadata and file is accessed or written in separate roundtrips.

## Using SecureFiles

### Compatibility

SecureFiles features are available with the compatibility set to Oracle Database 11g Release 1 or higher. Starting with Oracle Database 12*c*, SecureFiles is the default storage mechanism for LOBs in Oracle. In Oracle Database 11g Release1, 'BASICFILE' is the default storage LOB storage type. The keyword 'SECUREFILE' is used for specifying SecureFiles. The default LOB storage can be changed using the db_securefile parameter.

### db_securefile parameter

This system parameter allows the database administrator to specify the SecureFile usage policy in the init.ora file. This parameter is dynamic and the scope is 'ALTER SYSTEM'

Allowable values for this parameter include:

PERMITTED – allow SecureFiles to be created (Default)

FORCE – create all LOBs as SecureFiles

ALWAYS – attempt to create SecureFiles, but fall back to BasicFiles

IGNORE – ignore attempts to create SecureFiles

NEVER – disallow new SecureFile creation

### Retention

The RETENTION keyword specifies the retention policy to be used by the database for managing consistent reads (CR) for SecureFiles. It is specified in the lob_storage_clause of the CREATE TABLE or ALTER TABLE statement. It is used to configure the SecureFile column to store old versions of the SecureFile data for a period of time. Allowable values for this parameter are:

**AUTO** (default) – system automatically manages space as efficiently as possible. The system retains UNDO sufficient for consistent read capability.

**MAX** – SecureFiles retains older versions of LOB data until the size of LOB segment hits the limit specified in MAXSIZE. If MAXSIZE is not specified, MAX behaves like AUTO.

> If the combination is used as:
>
> RETENTION MAX
>
> MAXSIZE UNLIMITED then the value of retention is ignored and the following combination takes effect.
>
> RETENTION AUTO
>
> MAXSIZE UNLIMITED

**MIN** – Specify UNDO retention duration in specified seconds.

**NONE** – Space freed is immediately reusable since space management layer will not retain any UNDO if retention is NONE.

## Advanced Features

The following features are available only for SecureFile and do not apply to BasicFiles. These features in SecureFiles - Deduplication, Compression and Encryption - can be setup independently or as a combination of one or more features. If all three features are turned on, Oracle will perform deduplication first and then compression followed by encryption.

### Deduplication

This feature eliminates multiple, redundant copies of SecureFiles data and is completely transparent to applications. Oracle automatically detects multiple, identical SecureFiles data and stores only one copy, thereby saving storage space. Deduplication not only simplifies storage management but also results in significantly better performance, especially for copy operations.

DEDUPLICATION is done on per LOB segment basis; duplicates are not detected across multiple LOB segments. The lob_storage_clause allows for specifying deduplication at a partition level.  However, duplicate detection does not span across partitions or subpartitions for partitioned SecureFiles columns.

The DEDUPLICATE keyword is used to enable *Advanced LOB Deduplication* checking for SecureFiles. Oracle uses a secure hash index to detect duplicate SecureFile data and stores a single copy for all identical content. Deduplication is transparent to applications, reduces storage and simplifies storage management. KEEP_DUPLICATES is used to turn off deduplication and retain duplicate copies of SecureFile data. Deduplication is turned off by default.

**Note:** *Advanced LOB Deduplication* (previously named SecureFiles LOB Deduplication with Oracle Database 11g) is a feature of Oracle Advanced Compression.

**Example:**

- Create a table with a SECUREFILE LOB column and LOB deduplication enabled on only one partition. Only LOBs that belong to partition p1 are deduplicated.

```
CREATE TABLE t1 ( REGION VARCHAR2(20), mydata BLOB)
     LOB(mydata) STORE AS SECUREFILE )
PARTITION BY LIST (REGION) (
     PARTITION p1 VALUES ('x', 'y')
          LOB(mydata) STORE AS SECUREFILE (
               DEDUPLICATE),
     PARTITION p2 VALUES (DEFAULT)
);
```

- Disable deduplication on a SECUREFILE LOB column.

```
ALTER TABLE t1 MODIFY
     LOB(mydata) (
         KEEP_DUPLICATES);
```

**Compression**

Oracle detects if SecureFile data is compressible and will compress using industry standard compression algorithms. If the compression does not yield any savings or if the data is already compressed, SecureFiles will turn off compression for such LOBs. Compression not only results in significant savings in storage but also improved performance by reducing I/O, buffer cache requirements, redo generation and encryption overhead. Random access reads and writes to compressed SecureFiles are achieved without the need to decompress the entire file. Only the sub portion (compression unit) of the compressed file, corresponding to the logical offset being read or written, needs to be decompressed thus saving CPU and I/O.

There are three levels of *Advanced LOB Compression*: LOW, MEDIUM, and HIGH. By default, Advanced LOB Compression uses the MEDIUM level, which typically provides good compression with a modest CPU overhead of 3-5%. Advanced LOB Compression LOW is optimized for high performance. Advanced LOB Compression LOW maintains about 80% of the compression achieved through MEDIUM, while utilizing 3x less CPU. Finally, Advanced LOB Compression HIGH achieves the highest storage savings but incurs the most CPU overhead.

The COMPRESS keyword is used to enable SecureFile Advanced LOB Compression. SecureFile compression is orthogonal to table or index compression. Setting table or index compression does not affect SecureFile compression or vice versa. For partitioned tables, the lob_storage_clause is used for specifying compression at a partition level.

**Note:** *Advanced LOB Compression* (previously named SecureFiles LOB Compression with Oracle Database 11g) is a feature of Oracle Advanced Compression.

**Example:**

- Create a table with a SECUREFILE LOB column and LOB compression enabled on only one partition. Only LOBs that belong to partition p1 are compressed.

```
CREATE TABLE t1 ( REGION VARCHAR2(20), mydata BLOB)
     LOB(mydata) STORE AS SECUREFILE )
     PARTITION BY LIST (REGION) (
          PARTITION p1 VALUES ('x', 'y')
               LOB(mydata) STORE AS SECUREFILE (
                   COMPRESS ),
          PARTITION p2 VALUES (DEFAULT)
     );
```

- Disable compression on SECUREFILE LOB.

```
ALTER TABLE t1 MODIFY
     LOB(mydata) (
           NOCOMPRESS
     );
```

### Encryption

SecureFiles supports the use of Transparent Data Encryption (TDE) syntax. The database supports automatic key management for all SecureFile columns within a table and transparently encrypts/decrypts data and backups. Applications require no changes and can take advantage of SecureFiles using TDE semantics. SecureFiles supports the following encryption algorithms:

**3DES168:** *Triple Data Encryption Standard* with a 168-bit key size

**AES128:** Advanced Encryption Standard with a 128 bit key size

**AES192:** *Advanced Encryption Standard* with a 192-bit key size (default)

**AES256:** Advanced Encryption Standard with a 256-bit key size

The ENCRYPT and DECRYPT keywords are used to turn on or turn off SecureFile encryption and optionally select the encryption algorithm to be used. Encryption is performed at the block level and all SecureFiles in the specified column are encrypted. Column-level and tablespace-level encryption for SecureFiles must not be turned on at the same time because encryption comes at a non-trivial CPU cost; and one of them is unnecessary in this case.

### Example:

- Create a table with SECUREFILE LOB column and LOB encryption enabled on only one partition. Only LOBs that belong to partition p1 are encrypted.

```
CREATE TABLE t1 ( REGION VARCHAR2(20), mydata BLOB)
     LOB(mydata) STORE AS SECUREFILE (
          CACHE
          ENCRYPT USING 'AES128' )
     PARTITION BY LIST (REGION) (
           PARTITION p1 VALUES ('x', 'y')
           PARTITION p2 VALUES (DEFAULT)
     );
```

- Enable LOB encryption using 3DES168.

```
ALTER TABLE t1 MODIFY
       (mydata BLOB ENCRYPT USING '3DES168');
```

## Migration

Online Table Redefinition is the recommended technique for migrating from BasicFiles to SecureFiles. Migrating tables with BasicFile LOB columns/partitions using online redefinition has the advantage of not requiring the table/partition be offline, but it requires additional storage equal to or slightly greater than the space used by the table/partition (in order to use online table redefinition at least twice the space being currently used needs to be preserved). If the destination table is partitioned, normal methods for parallel execution for partitioning apply and online redefinition is executed in parallel. Global indexes need to be rebuilt.

Generation of redo space could cause performance problems during the process of migrating BasicFile LOB columns. In such cases, Oracle recommends setting the NOLOGGING storage parameter for the destination SecureFile columns and turning on LOGGING once the migration is complete.

**Example:**

Upgrade table 'tab1' with a BasicFile CLOB column 'c'

**Step 1:** Create interim table that is a replica of the source table and specify SecureFile attributes for the CLOB column

```
CREATE TABLE tab1_tmp(id number not null, c clob)
   LOB(c) STORE AS SECUREFILE (tablespace tbs_1);
```

**Step 2:** Perform Online Redefinition

```
DECLARE
      error_count pls_integer := 0;
BEGIN
  DBMS_REDEFINITON.start_redef_table('scott', 'tab1', 'tab1_tmp', 'id
id, c c');
 DBMS_REDEFINITON.copy_table_dependents('scott', 'tab1', 'tab1_tmp', 1,
true, true, true, false, error_count);
  DBMS_OUTPUT.put_line('errors := ' || to_char(error_count));
 DBMS_REDEFINITON.finish_redef_table('scott', 'tab1', 'tab1_tmp');
end;
/
```

## Interfaces

SecureFiles supports standard client interfaces including SQL, PL/SQL, OCI, ODBC and JDBC.

## Conclusion

SecureFiles is a feature of Oracle Database that truly delivers the ability to manage all enterprise content, structured or unstructured, in one seamless, efficient and high-performing Oracle Database. With SecureFiles, 'all data' in your organization can be stored and managed in a database, not just relational or structured data.

SecureFiles is engineered to break the performance barrier keeping file and other unstructured data out of the database. It offers comparable performance and simplicity to popular file systems. SecureFiles can take advantage of several advanced database capabilities that are not possible with file systems.

Data in SecureFiles, and other relational content, can be managed together in a consistent and seamless manner. The ability to participate in Oracle Database transactions ensures atomicity, read consistency and recovery capabilities.

Using SecureFiles, all information can be managed using the same database tools and made available to all applications in an efficient, consistent and seamless manner.

**Oracle Corporation, World Headquarters**
500 Oracle Parkway
Redwood Shores, CA 94065, USA

**Worldwide Inquiries**
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

**Hardware and Software, Engineered to Work Together**