

# Oracle Database 11g: SecureFiles

*An Oracle White Paper*  
*June 2007*

## 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 2.0 and the collaborative nature of Web-based Internet applications are forcing organizations to capture and manage more unstructured data such as photos, videos, news clips etc than ever before. Regulatory changes like Sarbanes-Oxley, HIPAA are further fueling this trend. Mainstream business applications now 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 – ultimately leads to lower ROI.

SecureFiles is a new feature in Oracle Database 11g that is 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 & recovery process, and perform seamless search across all information. Oracle Database 11g is the best solution for storing both structured or relational data and related unstructured data.

## **BEST STORAGE FOR FILE DATA**

Oracle Database 11g introduces 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 upon which future content management strategies will be based on. SecureFiles is a new re-architecture featuring entirely new disk formats, 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. LOBs from Oracle Database 10g and prior releases are still supported and will now be referred to as 'BasicFiles'. This paper provides an overview of SecureFiles and some examples on how to use this feature.

## **WHAT'S NEW WITH SECUREFILES**

With SecureFiles, Oracle has introduced 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 uses a new cache that 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 allocated from the buffer cache and 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 buffer size is not a tunable parameter*

### **Space Management**

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 new in-memory statistics for efficiently memory and space allocation. SecureFiles segments require tablespaces managed with automatic segment space management (ASSM).

### **Reduced Fragmentation**

SecureFiles, unlike BasicFiles or older LOBs, uses a dynamic CHUNK (one or more 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, availability of space in the segment to determine the optimal CHUNK size. 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 keep 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.

### **New Network Layer**

SecureFiles have a new client/server network layer that allows for high-speed data transfer between the client and server. A new protocol allows for reading and writing bulk data directly from the network socket without the need for any temporary staging. This allows 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. This LOB index is used both for access navigation and for managing freed or deleted blocks. 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 has plagued BasicFile LOBs and greatly improves performance, especially under real world insert-delete-reclaim situations.

### **No High water mark contention**

Reclamation of space is done in the background and all the space belonging to a freed SecureFile is reclaimed at once. Thus SecureFiles do not suffer from the high water mark contention during space reclamation unlike older LOBs.

### **Easier Manageability**

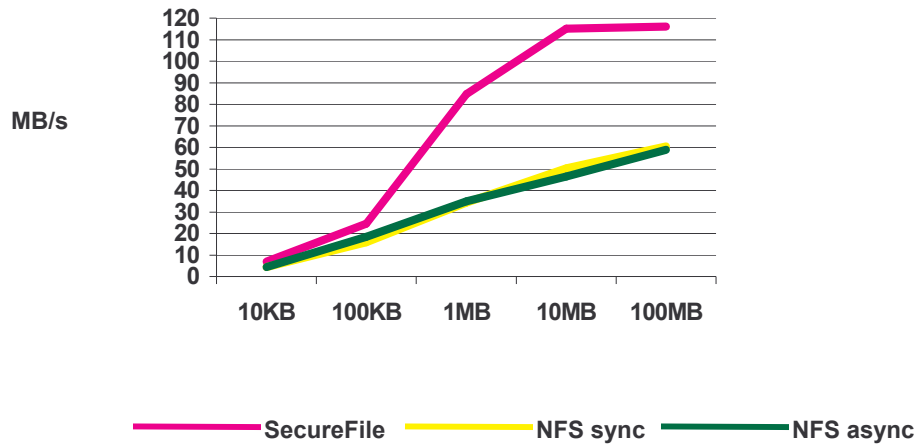
SecureFiles feature self-tuning and intelligent space and memory management algorithms and consequently require lesser number of user-tuned parameters. Specifically, FREEPOOLS, FREELISTS, FREELIST GROUPS & PCTVERSION no longer need to be specified and are ignored for SecureFiles. Not only are some of 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.

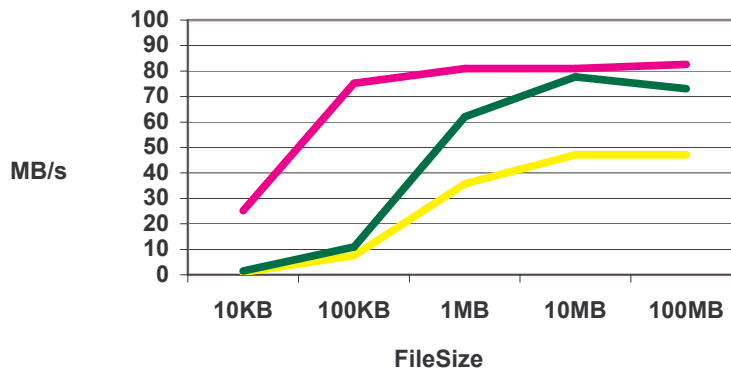
### SUPERIOR PERFORMANCE

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. We compare 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.

#### Query/Read Performance



#### Insert/Write Performance



SecureFile outperforms 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

SecureFile features are available with the compatibility set to 11.1 or higher. LOBs

from 10g and prior releases are still supported under 11.1 compatibility and a new keyword 'BASICFILE' is used to refer to such older LOBs. In 11gR1, 'BASICFILE' is the default storage LOB storage type. A new keyword 'SECUREFILE' is used for specifying SecureFiles. The default LOB storage can be changed using the db\_securefile parameter.

```
<LOB_storage_clause> ::=
  LOB
  { (LOB_item [, LOB_item ]...)
    STORE AS [ SECUREFILE | BASICFILE ]
  (LOB_storage_parameters)
  | (LOB_item)
    STORE AS [ SECUREFILE | BASICFILE ]
    { LOB_segname (LOB_storage_parameters)
    | LOB_segname
    | (LOB_storage_parameters)
    }
  }
```

**Note: SecureFiles requires tablespaces managed with ASSM.**

### db\_securefile parameter

This new 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

```
ALTER SYSTEM SET db_securefile = 'ALWAYS';
```

## 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 – Keep old versions until space used by segment has reached the size specified by the MAXSIZE parameter (mandatory when MAX is specified). The MAXSIZE parameter limits the amount of space in bytes that can be used by old versions of SecureFile data. If MAXSIZE is not specified, MAX behaves like AUTO

MIN – Specify UNDO retention duration in specified seconds.

NONE – No retention period, system reuses space as necessary

**Advanced Features** – The following features are available only for SecureFile and do not apply to BasicFiles. The new 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. Duplicate detection happens within a LOB segment. The lob\_storage\_clause allows for specifying deduplication at a partition level. However, duplicate detection does not span across partitions or subpartitions for partitioned SecureFile columns.

```
<LOB_sharing_clause> ::=  
{ DEDUPLICATE | KEEP_DUPLICATES }
```

The DEDUPLICATE keyword is used to turn on deduplicate 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.

**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), a BLOB)
                LOB(a) STORE AS SECUREFILE (
                    CACHE)
                PARTITION BY LIST (REGION) (
                    PARTITION p1 VALUES ('x', 'y')
                    LOB(a) STORE AS SECUREFILE (
                        DEDUPLICATE),
                    PARTITION p2 VALUES (DEFAULT)
                );
```

- Disable deduplication on a SECUREFILE LOB.

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

**Compression:** Oracle automatically 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 automatically 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. Compression is performed on the server-side and allows for random reads and writes to SecureFile data. SecureFile provides for varying degrees of compression: MEDIUM (default) and HIGH, which represent a tradeoff between storage savings and latency.

```
<LOB_compression_clause> ::=
{ COMPRESS [ HIGH | MEDIUM ]
| NOCOMPRESS }
```

The COMPRESS keyword is used to turn on server-side SecureFile 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.



### 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), a BLOB)
  LOB(a) STORE AS SECUREFILE (
    CACHE)
  PARTITION BY LIST (REGION) (
    PARTITION p1 VALUES ('x', 'y')
      LOB(a) STORE AS SECUREFILE (
        COMPRESS ),
    PARTITION p2 VALUES (DEFAULT)
  );
```

- Disable compression on **SECUREFILE** LOB.

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

**Encryption:** In 11g, Oracle has extended the encryption capability to SecureFiles and uses the Transparent Data Encryption (TDE) syntax. The database supports automatic key management for all SecureFile columns within a table and transparently encrypts/decrypts data, backups and redo log files. Applications require no changes and can take advantage of 11g 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

```
<LOB_encryption_clause> ::=
{ ENCRYPT [ USING 'encrypt_algorithm' ]
  [ IDENTIFIED BY password ]
| DECRYPT
}
```

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), a BLOB)
      LOB(a) 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
      ( a CLOB ENCRYPT USING '3DES168');
```

**Migration**

Online Table Redefinition is the recommended technique for migrating from BasicFiles to SecureFiles in 11g. 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. 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 tabl_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_REDEFINITION.start_redef_table('scott',
    'tabl', 'tabl_tmp', 'id id, c c');
    DBMS_REDEFINITION.copy_table_dependents('scott',
    'tabl', 'tabl_tmp', 1, true, true, false,
    error_count);
    DBMS_OUTPUT.put_line('errors := ' ||
    to_char(error_count));
    DBMS_REDEFINITION.finish_redef_table('scott',
    'tabl', 'tabl_tmp');
end;
/
```

## Interfaces

In Oracle Database 11g, SecureFiles supports extended LOB interfaces and are transparent to applications. SecureFiles support standard client interfaces including SQL, PL/SQL, OCI, ODBC and JDBC. In addition, Content DB can be used to access SecureFiles data using file system protocols such as WebDAV.

## CONCLUSION

SecureFiles is a new feature in Oracle Database 11g 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 database. It offers comparable performance and simplicity to popular file systems. SecureFiles can take advantage of several advanced DB 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. With Fine-grained auditing and Label security, organizations can now ensure a consistent authentication and access mechanism to all enterprise data, not just relational content. SecureFiles, combined with other Oracle technologies like RAC, ASM, partitioning and ILM, gives enterprises high availability, scalability and control over all content through its complete lifecycle. All information can be managed using the same database tools and made available to all applications in an efficient, consistent and seamless manner.

# ORACLE

SecureFiles

June 2007

Author: RAVI RAJAMANI

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

[oracle.com](http://oracle.com)

Copyright © 2007, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Other names may be trademarks of their respective owners.