

## Самые быстрые, самые надежные и самые маленькие LOB-ы (Faster, Safer, and Smaller LOBs, by Jonathan Gennick)

Джонатан Генник

Источник: журнал "Oracle Magazine", #5, 2007,

<http://www.oracle.com/technology/oramag/oracle/07-sep/057securefiles.html>

Коды скриптов и дополнительные материалы для этой статьи находятся в файле

<http://www.oracle.com/technology/oramag/oracle/07-sep/057securefiles.zip>

Механизм Oracle SecureFiles обеспечивает улучшенную производительность, повышенную безопасность доступа и уменьшенное использование диска.

Многие организации сегодня столкнулись с проблемой взрывного увеличения объема неструктурированных данных, часто находящихся в файлах данных, которыми также надо управлять наряду с более структурированными данными, которые, как правило, связаны с реляционными базами данных. Банковские приложения хранят образы зачетных чеков с подписями, приложения центров здравоохранения хранят цифровые изображения рентгеновских снимков и компьютерную томографию, гео-системы хранят снимки со спутников, системы промышленного планирования ресурсов хранят изображения счетов-фактур и т.д.

Oracle Database 11g прилагает большие усилия для решения проблемы адресации неструктурированных данных, разработав совершенно новую инфраструктуру хранения данных, которые в настоящее время часто остаются в файловых системах. Эта новая инфраструктура называется Oracle SecureFiles, которая полностью заменяет предыдущую инфраструктуру больших объектов Oracle (LOB) (которая теперь называется BasicFiles). Каждая грань Oracle SecureFiles LOB-архитектуры, от дискового формата до сетевого протокола и до алгоритмов redo и undo, были пересмотрены и реализованы заново с целью:

- Повышения производительности
- Повышения надежности
- Уменьшения использования диска

Предположим, выполняется миграция цифровых активов из баз данных и файловых систем и разрабатывается более совершенная система управления цифровыми активами, которая, например, может использоваться журнальным издательством. Такая система должна управлять заготовками статей и техпроцессом, включающим написание, редактирование, рецензирование и публикацию контента. Эта статья описывает, как установить и сконфигурировать Oracle SecureFiles и как применить демонстрационный

контент в подобной системе управления цифровыми активами.

### Конфигурирование экземпляра Oracle

После установки Database 11g и перед использованием Oracle SecureFiles, проверьте установленное значение нового в Oracle Database 11g параметра инициализации `db_securefile`. Его значение по умолчанию `db_securefile = permitted`

Это значение соответствует BasicFiles, а не SecureFiles. Оно означает способ хранения по умолчанию для любых новых создаваемых LOB-столбцов, и в то же время дает возможность явно указать SecureFiles там, где это необходимо. (BasicFiles – это LOB-архитектура в предыдущих версиях базы данных). Остальные значения параметра задают различные способы предопределенного использования того или иного типа памяти. Например, можно установить `db_securefile = force`, чтобы все LOB-ы создавались как SecureFiles, даже когда создающий пользователь явно указывает иное. Следует понимать, какое значение `db_securefile` действует в вашей базе данных.

### Создание табличных пространств

Любое табличное пространство, которое планируется использовать для Oracle SecureFiles, должно быть сконфигурировано с параметрами автоматического управления сегментами (automatic segment space management - ASSM). Для этого важно указать SEGMENT SPACE MANAGEMENT AUTO при создании табличного пространства для LOB. Следующая команда создает табличное пространство ARTICLE\_LOBS для этой статьи:

```
CREATE TABLESPACE
article_lobs
DATAFILE 'H:\APP\JONATHAN\ORADATA\ORCL\ARTICLE_LOBS.DBF'
SIZE 5M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;
```

Если вы повторяете действия, описываемые в

примере этой статьи, то выдайте пользователю необходимую квоту на табличное пространство ARTICLE\_LOBS. Также нужно правильно указать путь к соответствующему файлу в вашей системе.

### Настройки шифрования

Oracle SecureFiles дополнительно обеспечивает прозрачное (transparent) шифрование LOB-данных. Прозрачное шифрование данных защищает LOB-данные от неавторизованных пользователей, которые могут каким-либо способом получить доступ к файлу данных. Это шифрование совершенно незаметно конечному пользователю и приложениям.

Если вы планируете использовать шифрование для Oracle SecureFiles, необходимо создать кошелек прозрачного шифрования данных (transparent data encryption wallet) для хранения ключа шифрования. Сначала создайте директорию, в которой будет находиться кошелек. Эту директорию можно разместить в \$ORACLE\_HOME. Например:

```
mkdir H:\app\Jonathan\product\11.1.0\db_1\wallet
```

Затем добавьте в файл sqlnet.ora параметр ENCRYPTION\_WALLET\_LOCATION. Файл sqlnet.ora расположен в \$ORACLE\_HOME/network/admin. Следующий пример устанавливает соответствие с только что созданной директорией:

```
ENCRYPTION_WALLET_LOCATION=
(SOURCE=(METHOD=FILE)(METHOD_DATA=
(DIRECTORY= H:\app\Jonathan\product\
11.1.0\db_1\wallet)))
```

Убедитесь, что ENCRYPTION\_WALLET\_LOCATION в sqlnet.ora – это одна длинная строка. И ещё, если вы следуете нашему примеру, то необходимо назначить директории кошелька на какую-нибудь в собственной системе.

Наконец, установите ключ шифрования:

```
ALTER SYSTEM SET ENCRYPTION KEY
AUTHENTICATED BY "My secret key";
```

При такой настройке Oracle Database будет генерировать ключ шифрования, и размещать его в кошельке. Затем кошелек защищается паролем "My secret key".

### Создание SecureFile LOB

Создание Oracle защищенных LOB-файлов (SecureFiles LOB) может быть простым настолько, насколько просто указание STORE AS SECUREFILE в выражении для хранения LOB в предложении CREATE TABLE или ALTER TABLE. (Убедитесь, что ваши SecureFiles LOB связаны с ASSM-поддерживаемыми табличными пространствами). Листинг 1 создает таблицу article\_draft\_redef с единственным SecureFiles LOB-столбцом, названным article\_content. Данные для этого LOB

привязаны к табличному пространству ARTICLE\_LOBS (созданному ранее в этой статье). Каждая строка в этой таблице характеризует журнальную статью на заданной стадии цикла ее обработки, т.е. написания/редактирования/рецензирования.

*Листинг 1: Создание Oracle SecureFiles LOB*

```
CREATE TABLE article_draft_redef (
  article_id NUMBER,
  article_stage VARCHAR(10),
  article_content BLOB,

  CONSTRAINT article_stage_check_b
  CHECK (article_stage IN (
    '1st Draft', 'Edit Pass', '2nd Draft',
    'Copyedit', 'Review', 'Final')),

  CONSTRAINT article_draft_pk_b
  PRIMARY KEY (article_id)
)
LOB (article_content) STORE AS SECUREFILE
  article_content (
  TABLESPACE article_lobs
  RETENTION MIN 3600
  KEEP_DUPLICATES NOCOMPRESS DECRYPT
  CACHE READS);
```

Листинг 1 показывает некоторые опции, возможные при создании нового LOB.

**RETENTION MIN 3600** означает, что по крайней мере 1 час (60 минут x 60 секунд/минуту = 3,600) undo-данные будут сохраняться для LOB-столбца. Эта возможность указания минимального времени хранения полезно для запуска базы данных в режиме flashback. Значение по умолчанию RETENTION AUTO означает, что undo-данные будут храниться только в том случае, если есть свободный объем undo для удовлетворения целостности запросов на чтение, пока транзакция находится в работе. Вы также можете указать максимальный объем хранения в байтах или вообще ничего не хранить. Посмотрите Oracle SQL Reference, чтобы узнать подробнее синтаксис.

**KEEP\_DUPLICATES, NOCOMPRESS и DECRYPT** явно отключают устранение дублей, сжатие и шифрование. Я вернусь к этим опциям позже в этой статье. В реальной жизни вам, скорее всего, захочется оставить включенной одну или несколько из этих опций.

**CACHE READS** указывает LOB-данным разместиться в кэше (buffer cache) на время операции чтения, но не на время операции записи. Этим вы увеличите производительность чтения за счет потенциального уплотнения других данных кэша. NOCACHE отменяет поведение по умолчанию и означает никогда не помещать LOB-данные в кэш.

На Листинге 1 создается таблица article\_draft\_redef для миграции таблицы с устаревшими BasicFiles в новый метод

хранения SecureFiles. Чтобы проследить за процессом миграции, скачайте файл o57securefiles.zip, распакуйте его и следуйте инструкциям в readme.txt для создания исходной таблицы с названием article\_draft, которая будет мигрировать. Логическая структура столбца article\_draft точно такая же, как и в article\_draft\_redef.

Таблица article\_draft содержит существующие LOB-данные, хранимые как BasicFiles. Я предполагаю, что эти данные используются запущенным приложением, а их надо мигрировать из BasicFiles в SecureFiles. Таблица article\_draft\_redef со столбцом SecureFiles LOB является целевой для таблицы article\_draft.

### Планирование миграции

Так как Oracle SecureFiles представляет совершенно новый способ записи LOB-данных в базу данных, есть только один способ миграции LOB-данных из BasicFiles в SecureFiles – полностью переписать данные путем пересоздания или переопределения таблиц, содержащих столбцы BasicFiles. Если есть возможность получить данные в режиме offline, то можно просто извлечь данные из старой таблицы и вставить в новую. Начните с примерно с такого предложения:

```
INSERT INTO article_draft_redef
SELECT article_id,
       article_stage,
       article_content
FROM article_draft;
```

После INSERT выполните DROP и RENAME, чтобы новая таблица смогла заменить старую:

```
DROP TABLE article_draft;

RENAME article_draft_redef
TO article_draft;
```

Хотя на самом деле не обязательно использовать данные в режиме offline. Вместо этого обсудим переопределение таблиц в режиме online, что можно применять, начиная с Oracle9i Database Release 1. Листинг 2 показывает PL/SQL-блок, который мигрирует таблицу article\_draft в структуру, заданную таблицей article\_draft\_redef. Определение таблицы в режиме online сделать очень легко, и это делает миграцию прозрачной для пользователей и приложений, так как мигрируемая таблица все это время остается доступной.

*Листинг 2: Миграция через переопределение в режиме online.*

```
DECLARE
error_counter PLS_INTEGER;
BEGIN
```

```
--Начало процесса переопределения
DBMS_REDEFINITION.START_REDEF_TABLE (
'gennick', 'article_draft', 'article_draft_redef',
'article_id, article_stage, article_content');
```

```
--Окончание процесса переопределения
DBMS_REDEFINITION.FINISH_REDEF_TABLE (
```

```
'gennick', 'article_draft', 'article_draft_redef');
```

END;

Если LOB-таблица секционирована, есть еще один способ миграции – это обмен секциями. Указать параметры хранения LOB можно для секции, задав опцию миграции от BasicFiles к SecureFiles для каждой секции отдельно. (В статье Аруп Нанда (Arup Nanda) “Partition Decisions” (<http://www.oracle.com/technology/oramag/oracle/07-sep/o57partition.html>) находится пример, как делается обмен секциями.)

Убедитесь, что в табличном пространстве и на диске достаточно места для обеспечения выбранного метода миграции. Для переопределения таблицы в режиме online необходимо обеспечить пространство, достаточное для двух полных копий данных, существующих одновременно. Обмен секциями требует достаточный дисковый объем для размещения всего двух копий обмениваемых секций. Имейте в виду, что каждый обмен «старой» секции производится при том, что обмениваемая секция должна быть переведена в автономный режим (briefly taken offline). Наконец, необходимо оценить различные методы и выбрать тот метод, который лучше всего сработает в конкретной ситуации.

### Подтверждение миграции

Следует удостовериться в статусе SecureFiles заданного LOB-столбца проверкой подтипа сегмента с этим столбцом. Например, следующее предложение подтверждает, что article\_content теперь SecureFiles LOB:

```
SELECT segment_subtype
FROM user_segments
WHERE segment_name='ARTICLE_CONTENT';
```

Ожидаемый результат - это SECUREFILE, который означает хранение в виде SecureFiles. Если результат будет ASSM, хранение все еще производится в виде BasicFiles.

Устранение дублей

После миграции на Oracle SecureFiles вы можете включить режим устранения дублей, то есть опции Oracle SecureFiles, позволяющей серверу базы данных хранить только одну копию данного LOB в данном столбце в одной и той же секции. Два и более пользователя могут независимо хранить одинаковые данные в LOB-столбце — например, двойное хранение одного и того же варианта статьи — а база данных отслеживает эти дубли, храня только одну копию данных. Режим устранения дублей прозрачен. Пользователи убеждены, что они имеют каждый свою собственную копию данных и даже думают, что по-другому быть не может.

*Листинг 3: Проверка места, занимаемого LOB*

```
DECLARE
seg_blocks NUMBER;
seg_bytes NUMBER;
```

```
used_blocks NUMBER;
used_bytes NUMBER;
```



```
expired_blocks NUMBER;
expired_bytes NUMBER;
```

```
unexpired_blocks NUMBER;
unexpired_bytes NUMBER;
```

```
BEGIN
DBMS_SPACE.SPACE_USAGE (
'GENNICK', 'ARTICLE_CONTENT', 'LOB'
, seg_blocks, seg_bytes, used_blocks, used_bytes
, expired_blocks, expired_bytes, unexpired_blocks, unexpired_bytes);
```

```
DBMS_OUTPUT.PUT_LINE ('Bytes used = ' || to_
char(used_bytes));
END;
```

В процессе реальной миграции устранение дублей лучше проводить во время, а не после этого процесса. Хотя при демонстрации приятно увидеть как действительно производится удаление дубликатов. Выполните код Листинга 3, чтобы увидеть, сколько байтов занимает LOB-сегмент. (Убедитесь, что первый параметр процедуры ссылается на корректное название схемы в вашей системе). В моей системе результат следующий:

```
Bytes used = 450560
```

Чтобы устранить дубли в столбце `article_content` таблицы `article_draft`, выполните следующее предложение ALTER TABLE:

```
ALTER TABLE article_draft
MODIFY LOB(article_content)
(DEDUPLICATE LOB);
```

Существующие LOB-ы в столбце `article_content` будут просканированы, все дубли будут найдены и удалены. Все новые LOB-значения, записываемые в столбец, будут проверяться на наличие уже существующего такого же значения для предотвращения новых дублей.

Теперь выполните код Листинга 3 снова и посмотрите на результат устранения дублей. Количество байтов, используемых сегментом, должно уменьшиться. В моей системе я увидел следующее, уменьшенное значение:

```
Bytes used = 376832
```

[Разность - это] число байтов, занимаемых LOB-сегментом, было освобождено, так как документ #6 является дубль статьи, сохраненной в другой строке. Благодаря устранению дублей, эта статья не хранится дважды. Вместо этого, сохраняется указатель на LOB в другой строке. Когда вы считываете содержимое LOB для документа #6, этот указатель «прозрачно» трансформируется.

Определение дублей основано на контрольной сумме, вычисляемой по криптографическому хэш-алгоритму SHA1. Когда вы записываете новый LOB в LOB-сегмент,

контрольная сумма вычисляется по первым `n` байтам этого нового LOB-а. Если контрольная сумма не соответствует ни одному существующему LOB в сегменте, новый LOB записывается в базу данных. Контрольные суммы сохраняются для каждого LOB для устранения дублей в будущем.

Когда контрольная сумма нового LOB-а соответствует существующей контрольной сумме, есть большая вероятность, что эти два LOB-а идентичны. Экземпляр базы данных начинает побайтовое сравнение входящих LOB-данных с возможным дубликатом, который уже сохранен. Этот возможный дубликат назовем основным LOB. Процесс сравнения считывает основной LOB, но не записывает новых данных до тех пор, пока LOB-данные продолжают соответствовать. Если сравнение LOB-ов показало идентичность, записывается только указатель на основной LOB. Если сравнение неуспешно, данные основного LOB до этой точки используются для формирования первой части нового LOB, и оставшиеся входящие данные нового LOB записываются на диск.

### Сжатие

В режиме сжатия данные в LOB-е кодируются для уменьшения объема, требуемого для его хранения. Сжатие прозрачно и уменьшает количество памяти, необходимое для LOB-данных, позволяя сэкономить каждый доллар от организационных затрат на дисковую систему. Сжатие требует дополнительных мощностей CPU, так как сжатие выполняется перед вставкой в LOB-столбец. Например, изображения, такие как PNG-файлы и JPEG-файлы, часто сжимаются как часть применяемой схемы шифрования. Дальнейшее сжатие файлов изображений может привести к потере производительности CPU. Однако текстовые данные, такие как XML и текстовые документы, целесообразно сжимать, расходуя CPU, так как экономия дискового пространства в этом случае более существенна. Чтобы включить сжатие для демонстрационного LOB-столбца типа `SecureFiles`, выполните следующее предложение ALTER TABLE:

```
ALTER TABLE article_draft
MODIFY LOB(article_content)
(COMPRESS HIGH);
```

Это предложение активирует высокоуровневое сжатие данных столбца. Можно указать `MEDIUM` для отмены части этого сжатия для уменьшения затрат CPU. Варианта сжатия `LOW` нет, однако это ключевое слово зарезервировано для будущих реализаций.

Из кода Листинга 3 видно, что теперь задействовано еще меньше места для LOB-сегмента. Я получил следующий результат:

```
Bytes used = 90112
```

Помните, что сжатие имеет множество преимуществ. Требования к дисковому пространству меньше. Меньше

данных требуется перемещать вперед-назад, тем самым увеличивается производительность. Генерация redo уменьшается. Меньше используется извлечений и кэша, оставляя больше места в кэше для других объектов.

### Шифрование

Наконец мы добрались до прозрачного шифрования данных. Зашифрованные LOB-данные защищены, даже когда файлы данных или файлы резервных копий попадают в «грязные» руки. Зашифрованные LOB-данные защищены, даже когда они передаются по сети.

Чтобы зашифровать LOB-ы в Oracle SecureFiles, можно выбрать один из четырех алгоритмов шифрования: 3DES168, AES128, AES192 и AES256. Например, для выбора шифрования по AES256, которое представляет собой 256-битный алгоритм, выполните следующее предложение ALTER TABLE:

```
ALTER TABLE article_draft
MODIFY LOB(article_content)
(ENCRYPT USING 'AES256');
```

Помните, что шифрование данных защищает ваши данные от недоброжелателей, которые могут случайно заполучить файл данных или часть резервной копии. Также нужно управлять доступом к объектам базы данных внутри вашей базы данных, так как только авторизованные пользователи базы данных имеют доступ для выполнения запроса по LOB-данным, которые вы защитили.

### Заключение!

Oracle SecureFiles предлагает три отличные возможности, которые можно использовать сразу же без выполнения каких-либо изменений в ваших приложениях, использующих LOB:

- Устранение дублей
- Сжатие
- Шифрование

Предварительное тестирование показывает существенное повышение производительности, благодаря использованию Oracle SecureFiles. Внутреннее тестирование Oracle показывает от 200 до 900 процентов повышения производительности по записи в зависимости от размера LOB и места, в котором будет повторно использоваться существующий сегмент. Те клиенты, которые раньше всех начали использовать эти возможности, заявили о повышении производительности от 300 до 700 процентов по основным функциям приложения.

Так как Oracle SecureFiles легко обратно-совместимы с BasicFiles, даже до момента использования тех же названий типов данных, можно начать применять что-нибудь из того, что предложил Oracle SecureFiles, сразу же с момента перехода на Oracle Database 11g. Вам необходимо переписать LOB-данные в базу данных, и вы можете делать это в то время, как данные находятся в режиме online, и изменение совершенно прозрачно как по отношению к пользователям, так и по отношению к приложениям.

*Oracle SecureFiles* – это подарок. Взгляните на него. Получите сразу же улучшение производительности, сэкономьте денег на дисковое пространство, и защитите ваши данные от того, кто может вам навредить.