Содержание

Предисловие

Жизнь администраторов СУБД становится все более и более сложной, а напряженный режим работы быстро становится нормой. Администраторы баз данных постоянно сталкиваются с проблемами, которые при определенных условиях могут привести организации и прочие юридические лица к многомиллионным убыткам буквально за минуту или, при самом худшем развитии событий, к полной остановке работы инфраструктуры баз данных компании. Конечно, подобные ситуации маловероятны, но к потенциальной возможности их возникновения должен быть готов каждый администратор баз данных, чтобы вовремя устранить надвигающиеся проблемы.

Главная задача этой книги – показать способы максимально оперативного устранения в базах данных серьезных проблем, которые могут оказать воздействие на эксплуатационную среду в целом. Читателям демонстрируется пошаговое руководство, необходимое для решения конкретной проблемы, с разбором случаев из реальной практики, которые могут произойти (и происходят) в любой день, в любое время, в любой базе данных Oracle.

Чтобы не терять времени в попытках поиска решения проблемы, которая представляет опасность для вашей базы данных или уже нарушила ее работоспособность, вы можете сразу же обратиться к этой книге, предлагающей решения некоторых самых крупных проблем, вероятность возникновения которых наиболее велика. Даже если решение текущей проблемы не найдено здесь, вы узнаете, как нужно быстро искать в Интернете способы устранения любой конкретной проблемы.

Основная идея данной книги заключается в том, чтобы предложить вам реальную помощь при возникновении серьезных проблем с базами данных Oracle в различных эксплуатационных средах. Наряду с описанием самых эффективных обобщенных методик в книге исследуются некоторые наиболее важные проблемы, возникающие в базах данных Oracle, а также способы оперативного решения этих проблем, описанные в простой и понятной форме.

Ориентированная в основном на администраторов СУБД Oracle (DBA) и на администраторов аппаратно-программных комплексов (кластеров серверов) Oracle Database Machine (DMA) книга «Руководство по диагностике и устранению проблем в СУБД Oracle» может служить практическим техническим руководством для выполнения повседневных действий по диагностике и устранению неисправностей, по настройке и устранению проблем в ходе операций администрирования семейства программных продуктов Oracle Database Server.

Написанная известными во всем мире авторами с огромным опытом работы в группах Oracle ACE, в качестве руководителей ACE и независимых экспертов, эта книга должна стать настольным справочником по решению проблем, объединяющим практические примеры из реальной жизни и планы действий по диагностике и устранению неисправностей в сложных совокупностях баз данных Oracle. В данной книге подробно описаны методы решения следующих задач:

- выбор наикратчайшего пути для решения крупномасштабных проблем;
- наиболее эффективная организация рабочего дня с помощью надежных методик, основанных на практическом опыте экспертов этой области;

- создание собственного плана по устранению неисправностей и решению проблем;
- осуществление повседневного «упреждающего» сопровождения, обеспечивающего стабильность эксплуатационной среды;
- использование стандартных инструментальных средств и сценариев, наиболее пригодных для быстрых и эффективных решений проблем.

Авторы этой полезной, изобилующей разнообразными техническими подробностями книги при ее написании ориентировались на различные уровни (средний, опытный эксперт) пользователей семейства программных продуктов Oracle Database Server. В этой книге рассматриваются версии Oracle Database 11g и Oracle Database 12c и весь комплект соответствующего вспомогательного и дополнительного программного обеспечения для СУБД Oracle.

Глава 1

Диагностика и настройка производительности сегментов LOB

Тип данных LOB (или LOB-объект; Large OBject) позволяет хранить и обрабатывать неструктурированные или слабоструктурированные данные, такие как документы, графические изображения, фрагменты видео, звуковые файлы и XML-файлы. Пакет `DBMS_LOB` был специально разработан для обработки данных типа LOB. Начиная с версии Oracle Database 12c LOB-объекты могут хранить большие объемы данных с максимальным размером до 128 Тб в зависимости от размера блока, установленного для конкретной базы данных. Отдельная таблица может содержать один или несколько столбцов данных типа LOB, например бинарный LOB-объект (BLOB), символьный LOB-объект (CLOB), LOB-объект, содержащий национальные символы (NCLOB), и файл (BFILE). В этой главе описываются некоторые проблемы, которые могут возникать при наличии сегментов LOB в конкретной среде, а также способы устранения этих проблем.

Краткое описание типа данных LOB

При создании и проектировании LOB-объектов в своих базах данных помните о необходимости внимательного изучения руководства Oracle Database Secure Files and Large Objects Developer's Guide (Oracle, 2016), особенно главы 16 «Performance Guidelines» (Правила настройки производительности). Это руководство предоставит вам самые актуальные рекомендации по созданию LOB-объектов в зависимости от того, какие именно типы данных будут размещаться в этих столбцах таблицы.

Важно помнить, что при создании столбца для LOB-объекта в таблице в действительности создаются два различных сегмента: `LOBSEGMENT` и `LOBINDEX`. Сегмент `LOBINDEX` указывает на «фрагменты» LOB-объекта, которые хранятся в соответствующем сегменте `LOBSEGMENT`. В некоторых случаях LOB-объекты могут храниться как «встроенные» (inline), то есть прямо внутри сегмента таблицы, но такой способ хранения обычно используется для относительно небольших объемов данных в LOB-объекте (не более 4000 байтов) или для значения `NULL`. При соблюдении этих условий значения LOB-объектов хранятся непосредственно в сегменте таблицы.

Рассмотрим ситуацию, в которой база данных вынуждена сгенерировать событие маркера максимального уровня заполнения (high-watermark – HW) при включении в очередь. Когда в сеансе требуется доступ к ресурсу базы данных, типом блокировки, управляющей доступом к этому ресурсу, является включение в очередь (enqueue). Возникновение события при включении в очередь (тип enqueue) означает, что сеанс должен ждать, когда другой сеанс освободит данный ресурс. В сообщении о таком событии всегда указывается имя очереди в формате `enq: <enqueue_type >- <details>`, причем для каждого типа очереди подробности будут различными. Столбцы `P1`, `P2` и `P3` динамического вида (представления) `V$SESSION` и `V$SESSION_WAIT` также помогают узнать более подробно о том, в каком текущем состоянии находится ожидающий сеанс и что именно стало причиной блокировки. Эти значения могут иметь различный смысл в зависимости от типа очереди, как можно видеть в соответствующих столбцах `P1TEXT`, `P2TEXT` и `P3TEXT`. В листинге 1.1 показано лишь несколько из более чем 600 событий, возникших при включении в очередь.

Листинг 1.1 ❖ События при включении в очередь

```
select
  distinct name
from
  v$event_name
where
  name like '%enq%'
order by 1;
```

```
enq: AB = ABMR process initialized
enq: AB = ABMR process start/stop
enq: AC = acquiring partition id
enq: AD = allocate AU
enq: AD = deallocate AU
enq: AD = relocate AU
enq: AE = lock
…
```

Событие HW при включении в очередь является ответной реакцией на попытку распределения пространства за пределом наивысшего допустимого уровня для данного сегмента. Рекомендуемое действие по обработке непредвиденного события HW при включении в очередь – выделение дополнительных экстентов (extent) для данных сегментов LOB-объектов вручную. Но в последующих разделах мы будем рассматривать и профилактические методы, помогающие избежать событий HW при включении в очередь посредством реализации наиболее эффективных практических методик конфигурирования LOB-объектов.

Устранение проблемы с LOB-объектом: пример из практики

В приводимом ниже примере, взятом из реальной практики, база данных одной компании, занимающейся электронной коммерцией, полностью зависла. Следующие действия помогают быстро определить и устранить возникшую проблему:

1. Создание отчета по автоматическому репозиторию рабочей нагрузки (automatic workload repository – AWR), чтобы определить, не является ли событие `enq: HW` одним из ожидающих событий в пяти верхних строках отчета. За-

регистрируйтесь в своей базе данных как SYSDBA и выполните следующую команду. После появления промпта выберите два самых последних идентификатора снимков AWR, чтобы исследовать только два самых свежих снимка:

```
$> ?/rdbms/admin/awrrpt.sql
```

2. Изучение первых пяти ожидающих событий, приведенных в полученном отчете:

```
Event                          Waits         Time (s)     (ms)    Time   Wait Class
-----------------------------  ------------  -----------  ------  -----  ----------
enq:HW contention                  249,725        3,289      13   90.0  User I/O direct
path write                         168,486          103       1    2.8  User I/O DB CPU
PX qref latch                    6,392,581           40       0    1.1  Other
PX Deq: Slave Session Stats             18            1      51     .0  Other
```

3. Если искомое событие постоянно возникает в базе данных, просто выполните следующий запрос для определения сеансов, в которых имеет место событие HW (contention):

```
SELECT sid, event
  FROM gv$session_wait
  WHERE event LIKE '%contention%';

SID     EVENT
------  ------------------------
9426    enq: HW = contention
13050   enq: HW = contention
```

4. Выполните следующий запрос для выделения конкретного объекта, в котором возникло событие достижения наивысшего допустимого уровня HW:

```
SELECT
  DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(id2) file#
  DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(id2) block#
  FROM gv$lock
  WHERE type = 'HW';

FILE#       BLOCK#
----------  -----------
        19          195
```

5. Используя номер файла (19) и идентификатор блока (195), полученные из предыдущего запроса, найдите объект, заблокированный по данному событию, с помощью следующего запроса:

```
SELECT
  owner,
  segment_type,
  segment_name
 FROM
  dba_extents
 WHERE
  file_id = 19
 AND
  195 between block_id
 AND
  block_id + blocks = 1;
```

```
OWNER        SEGMENT_TYPE        SEGMENT_NAME
-----------  ------------------  ----------------------------
ORABPEL      LOBSEGMENT          SYS_LOB0000181226C00029$$
```

6. Далее необходимо определить имя таблицы, которая ссылается на этот LOB-сегмент, с помощью следующего запроса:

```
SELECT
  owner,
  table_name,
  segment_name
 FROM
  dba_lobs
 WHERE
  segment_name= 'SYS_LOB0000181226C00029$$';

OWNER            TABLE_NAME                    SEGMENT_NAME
---------------  ----------------------------  ----------------------------
ACOM_BPEL_AQ     INSERT_SITE_ORDER_BI_TBL      SYS_LOB0000181226C00029$$
```

7. После точного определения LOB-сегмента, из-за которого возникло событие переполнения очереди HW, следует частично устранить главную причину возникшей проблемы, вручную добавив новый экстент к этому сегменту. Важно знать, размещается ли объект в табличном пространстве, и если это так, то имеют ли его сегменты размер AUTO_ALLOCATED или UNIFORM. Если экстенты определены как UNIFORM, то нужно просто добавить новый экстент того же размера, в противном случае следует добавить новый экстент с размером, равным размеру самого большого экстента для этого сегмента:
   a. Определить наибольший размер экстента для данного объекта:

```
SELECT
  DISTINCT bytes
  FROM dba_extents
  WHERE segment_name = 'SYS_LOB0000181226C00029$$'
  AND owner = 'ORABPEL';

BYTES
---------
  1048576
  8388608
    65536
```

   b. Добавить экстенты такого же размера в данный LOB-сегмент:

```
ALTER TABLE orabpel.insert_site_order_bi_tbl
  MODIFY LOB ('SYS_LOB0000055018C00004$$')
  (ALLOCATE EXTENT (SIZE 8M));
```

Поздравляем, с помощью этой простой операции вы только что спасли свою компанию от миллионных убытков из-за существенных задержек или даже полной остановки обслуживания клиентов на веб-сайте электронной торговли.

Еще один пример из реальной практики: HW Resolution

Если база данных является репозиторием для таких приложений, как Oracle Transportation Management (OTM), Oracle Business Process Execution Language (BPEL) и Oracle E-Business Suite (EBS), то, вероятнее всего, в ней будут возникать

непредвиденные проблемы, связанные с LOB-сегментами. Например, недавно в OTM возникла подобная проблема в таблице I_TRANSMISSION при выполнении 150 000 000 команд INSERT и DELETE и 500 000 000 команд UPDATE в столбце XML_BLOB этой таблицы за один месяц. Ниже приведена последовательность действий по оперативному выявлению и устранению этой проблемы:

Примечание Описанный ниже подход также можно использовать в предыдущем примере enq: HW. Этот пример просто демонстрирует другой вариант выявления и решения данной проблемы.

1. Выполнить команду создания отчета по AWR и проверить раздел «Top 5 Timed Events». Если проблема с LOB-сегментами действительно возникла, то вы должны наблюдать в отчете нечто подобное:

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~
                                                    Avg
                                                    Wait    Total  Call Wait
Event                            Waits   Time (s)  (ms)    Time % Class
------------------------------ ------------ --------- ------- ------ -------------
db file sequential read          2,580,474    35,544   14      77.4 User I/O
SQL*Net more data from client      659,140     5,513    8 **   12.0 Network
CPU time                                       4,540  9.9
enq: HW - contention                88,910     2,890   33 *     6.3 Configuration
log file sync                      777,146     1,688    2       3.7 Commit
```

2. В том же отчете по AWR проверить раздел «Top Enqueue Activity» на наличие в нем содержимого, подобного выделенному фрагменту:

```
Enqueue Activity Snaps: 1234-1235
-> only enqueues with waits are shown
-> Enqueue stats gathered prior to 10g should not be compared with 10g data
-> ordered by Wait Time desc, Waits desc
                           Gets     Gets              Wait       Avg Wait
Enq Type            Reg    Succ     Failed   Waits    Time (s)   Time (ms)
------------------  -------  -------  -------  -------  ---------  -----------
HW-Segment
High Water Mark      93,860   93,862        0   88,226   2,961        33.56 *
TX-Transaction
(row lock contention)    272      272        0      209     570     2,729.44 **
TX-Transaction
(index contention)     4,564    4,564        0    4,144      34         8.16
TX-Transaction       793,989  794,042        0       97       0         4.08
```

3. Проверить раздел «Wait Events» того же отчета по AWR. В этом разделе должна выводиться информация, подобная следующей:

```
-> s -second
-> ms - millisecond - 1000th of a second
-> ordered by wait time desc, waits desc (idle events last)
                                                 % Time   Total Wait   Avg        Waits
Event                              Waits         -outs    Time (s)     Wait (ms)  /txn
---------------------------        ------------  -------  -----------  ---------  -----
db file sequential read                2,580,474    .0          35,544        14    3.3
SQL*Net more data from client  **        659,140    .0           5,513         8    0.9
enq: HW - contention                      88,910    .0           2,890        33*   0.1
log file sync                            777,146    .0           1,688         2    1.0
```

```
read by other session                  103,140    .0          929        9     0.1
SQL*Net break/reset to client          114,782    .0          813        7     0.1
enq: TX - row lock contention ***          380  43.4***      557     1466*    0.0
log file parallel write                552,663    .0          394        1     0.7
latch: cache buffers chains             55,203    .0          382        7     0.1
```

4. После сбора всей необходимой информации нужно найти объект, в котором было сгенерировано рассматриваемое событие ожидания `enq: HW`. Выполнить следующий запрос для выделения всех объектов, в которых встречается запись `enq: HW` между снимками AWR с интервалом, выбранным для создания отчетов по AWR:

```
SELECT
 sql_id,
 event,
 event_id,
 time_waited,
 current_obj#,
 current_file#,
 current_block#
FROM dba_hist_active_sess_history
WHERE snap_id BETWEEN 1072 AND 1076
AND event LIKE '%HW%CONTENTION%'
AND time_waited > 0
AND current_obj# <> -1
ORDER BY time_waited, event, sql_id;
```

5. Перехватить значения для `current_obj#` из предыдущего запроса и передать их в следующий запрос:

```
SELECT
 owner,
 object_name,
 object_type
FROM dba_objects
WHERE object_id = [current_obj# of query above];
```

Приведенный ниже альтернативный запрос выводит крайне важную информацию, необходимую для выделения имени объекта, его типа, и (что наиболее важно) SQL-идентификатор, соответствующий командам, на которые оказало воздействие событие `enq: HW`:

```
SQL> col object_name format a30
SQL> col program format a30
SQL> col event format a30
SQL> SELECT DISTINCT CURRENT_OBJ#,o.object_name,o.owner,o.
object_type,CURRENT_BLOCK#,SESSION_STATE,SQL_ID,EVENT
 from v$active_session_history a, dba_objects o
 where a.current_obj# = o.object_id
 and a.event like 'enq%HW%';
CUR_     OBJ_NAME         OWN   OBJECT_   CUR_      SESS     SQL_ID         EVENT
OBJ#                            TYPE      BLOCK#
-------  ---------------  ----  --------  --------  -------  ---------      ------
235738   MLOG$_ENI_OLTP_  ENI   TABLE     100747    WAITING  0ghshjjvf86bg  enq: HW -
         ITEM_STAR                                                         contention
612464   HIST_PEDIDOS     B2W   TABLE     394731    WAITING  9phv0npccjqa2  enq: HW -
                                                                           contention
```

6. Перед добавлением экстентов к данному объекту рекомендуется проверить размеры его существующих экстентов:

```
SELECT COUNT(*), bytes/1024/1024 "MB"
  FROM dba_extents
 WHERE segment_name = 'HIST_PEDIDOS' AND owner='B2W'
 GROUP BY bytes;
COUNT(*)   BYTES
---------- ----------
   1969801     131072
```

7. Теперь, когда известен наибольший размер экстента для данного объекта, вы просто добавляете несколько новых экстентов. Используйте язык SQL для компоновки необходимых команд `MODIFY` на основании полученной ранее информации:

```
SELECT
  'alter table '||table_owner||'.'||table_name||' modify partition '||partition_name||' lob
('||column_name ||') (allocate extent (size 131072));'
  FROM dba_lob_partitions
 WHERE table_name = 'HIST_PEDIDOS'
   AND partition_name like '%2014%';
```

Наш богатый практический опыт позволяет утверждать, что после добавления достаточного количества новых экстентов (рекомендуется добавить не менее 20) проблема `enq: HW` просто исчезнет из списка ожидающих обработки событий в этой базе данных, и приложения снова начнут работать с нормальной скоростью.

Добавление экстентов вручную вполне подходит для временного устранения конкуренции при распределении ресурсов, тем не менее следует отметить, что этот способ не является постоянным надежным решением. Поскольку событие `enq: HW` при включении в очередь возникает, когда необходимо увеличить предельное значение для LOB-объекта быстрее, чем фоновый процесс сможет захватить и отформатировать новые фрагменты LOB, мы просто предварительно выполняем эту работу для фоновых процессов, распределяя новые экстенты вручную. Но в зависимости от того, какой объем пространства необходимо выделить, чтобы справиться с увеличивающейся рабочей нагрузкой, 30 или даже 50 экстентов дадут приложению некоторое время для заполнения этого пространства, прежде чем снова возникнет проблема достижения предельного значения. Поэтому в зависимости от конкретной ситуации, от ожидаемого размера и продолжительности рабочей нагрузки для данного приложения распределение экстентов вручную будет лишь временным решением рассматриваемой проблемы.

Кроме того, важно отметить, что поскольку подобная ситуация типична почти для всех языков описания данных (data definition languages – DDL), размещение экстента вручную непременно потребует исключающей блокировки соответствующего сегмента в базе данных, следовательно, в зависимости от уровня распараллеливания вы не сможете работать с этим сегментом в течение достаточно длительного интервала времени. Таким образом, удачным решением является перехват результатов подобного запроса в скрипте, обрабатывающем аварийные ситуации, который должен отслеживать эти события в базе данных и выдавать соответствующее предупреждение при опасности возникновения рассматриваемой ситуации.

Проблемы BASICFILE LOB: более эффективное решение

Ниже приводится краткое описание возможных решений для постоянного увеличения количества фрагментов LOB, выделяемых для сегмента LOB:

- перемещение сегмента LOB в табличное пространство с увеличением размера сегмента UNIFORM. Этот метод зарекомендовал себя как наиболее эффективный, так как большие экстенты UNIFORM предоставляют больше фрагментов для каждой HW-операции;
- увеличение размера фрагмента LOB-объекта. Сначала определяется средний размер данных LOB-объекта с помощью процедур DBMS_LOB, GETLENGTH, затем устанавливается размер фрагмента LOB-объекта в диапазоне от 120 до 150 процентов от среднего размера данных LOB;
- увеличение размера фрагмента при корректировке посредством настройки события 44951 TRACE NAME CONTEXT FOREVER, LEVEL 1024. При достижении предельного значения для LOB-объекта будет предпринята попытка коррекции пространства внутри сегмента сначала при помощи удаления старых образов LOB с учетом значений параметров PCTVERSION и RETENTION. Настройка указанного события позволит увеличить количество фрагментов, которые Oracle будет корректировать в одной операции, то есть сделает ее более эффективной при обработке события enq: HW. Эта стратегия дополняет процедуру установки подходящего размера фрагмента LOB, описанную выше;
- необходимо убедиться в том, что был активирован механизм Automatic Segment Space Management (ASSM) для табличного пространства, в котором размещается проблемный LOB-объект. Кроме того, это является необходимым предварительным условием для LOB-объектов типа SECUREFILE.

В заключение важно отметить, что даже в СУБД Oracle Database Release 11.2.0.1 присутствовала весьма неприятная ошибка, которая могла повредить заголовок сегмента (то есть сделать весь сегмент некорректным) при прерывании ручной операции ALLOCATE EXTENT, особенно для сегментов LOB. Для получения более подробной информации об этой проблеме см. примечание My Oracle Support (MOS) Note 1229669.1 «Segment header corruption if extent allocation operation is interrupted».

Сравнение LOB-объектов BASICFILE с SECUREFILE LOB

В версии 11g СУБД Oracle появилась новая опция SECUREFILE для хранения данных типов LOB, предлагающая улучшенные функциональные возможности для управления типами данных LOB, включая устранение дублирования, шифрование и сжатие данных. Таким образом, настоятельно рекомендуется преобразовать LOB-объекты BASICFILE в LOB-объекты SECUREFILE при условии, что ваша СУБД обновлена с версии Oracle 9i или Oracle 10g до Oracle 11g. Сделав это, вы с большой вероятностью улучшите производительность и управляемость и в то же время устраните некоторые проблемы, описанные в предыдущем разделе. Но следует помнить, что переход к LOB-объектам SECUREFILE не обязательно должен быть всеобъемлющим, что демонстрируют приведенные ниже сценарии.

Одна вполне очевидная проблема может возникать, когда приложение в первый раз вставляет данные в новый LOB-объект, который был переведен в формат

SECUREFILE. Если у вас установлены версии 11.2.0.1 или 11.2.0.2, то могут возникать серьезные проблемы при вставке данных в поле LOB вне зависимости от того, установлен для него формат BASICFILE или SECUREFILE. (Отметим, что эта конкретная ошибка была исправлена в версии 11.2.0.3.)

Кроме того, важно определить, как параметры хранения LOB-объектов влияют на производительность. Ниже приведен краткий обзор наиболее важных параметров:

- параметр CHUNK определяет наименьшую единицу измерения LOBSEGMENT и всегда кратен значению параметра DB_BLOCK_SIZE. Если в базе данных размер DB_BLOCK_SIZE установлен равным 16 Кб, а размер данных, помещаемых в столбец LOB, составляет 2 Кб, то 14 Кб просто остаются неиспользованными. Поскольку максимальный размер фрагмента равен 32 Кб, важно определять параметр CHUNK так, чтобы пространство не расходовалось без пользы;
- директива CACHE сообщает Oracle о необходимости сохранять данные LOB в кэш-буфере базы данных, тогда как при установке значения NOCACHE данные LOB никогда не помещаются в кэш-буфер. В результате Oracle будет использовать прямые операции чтения/записи для некэшированных данных (характеризуется ожиданием события прямых операций чтения/записи) или выполнять чтение/запись кэш-буфера базы данных для кэшированных данных (характеризуется ожиданием события операции последовательного чтения из файла базы данных). Также отметим, что для данных LOB, которые только считываются, но никогда не записываются, существует третья директива CACHE READS, которая вносит данные LOB в кэш-буфер только при операциях чтения, но не при операциях записи;
- параметр LOGGING разрешает ведение записей об изменениях данных LOB в журналах отмены операций в режиме онлайн. Если нужно улучшить производительность языка обработки данных (data manipulation language – DML), работающего с данными LOB, и при этом известно, что для восстановления данных в ваших приложениях не требуется журналирование изменений данных LOB, то следует рассмотреть возможность установки параметра NOLOGGING для LOB. Отметим, что если определить параметр NOLOGGING для встроенных LOB-объектов, то любые изменения в данных этих объектов будут продолжать фиксироваться в журналах отмены операций в режиме онлайн.

Различия между новыми и старыми типами LOB

Для демонстрации различий между упомянутыми выше форматами LOB создадим две таблицы: одну с LOB-объектом BASICFILE, вторую с LOB-объектом SECUREFILE:

1. Создать таблицу test1, содержащую столбец с LOB-объектом BASICFILE. Помните, что если вы выполняете эти тесты в Oracle Database 12c, то необходимо определить параметр BASICFILE, чтобы использовать именно этот тип данных, так как в указанной версии (и в последующих) при создании LOB-объекта по умолчанию устанавливается формат SECUREFILE:

```
CREATE TABLE test1 (col1 CLOB,col2 number)
LOB(col1) STORE AS SECUREFILE(CACHE)
tablespace TS_GG_DATA;
```

После создания таблицы, содержащей сегмент LOB, в ней формируются два различных типа сегментов: LOBSEGMENT и LOBINDEX. По названиям можно понять, что LOBINDEX представляет собой индекс или указатель, используемый для доступа к страницам или фрагментам соответствующего LOBSEGMENT. Следующий запрос показывает характеристики сегментов:

```
SYS@ORCL AS SYSDBA> SELECT COUNT(*), segment_name, segment_type
FROM dba_extents
WHERE segment_name = 'TEST1'
GROUP BY segment_name, segment_type;
COUNT(*)   SEGMENT_NAME   SEGMENT_TYPE
---------- -------------- ------------
        24 TEST1          TABLE
```

2. Создать таблицу test2 с использованием формата SECUREFILE для LOB-объекта:

```
CREATE TABLE test2 (col1 CLOB,col2 number)
LOB(col1) STORE AS BASICFILE
tablespace TS_GG_DATA;
```

3. Выполнить следующий запрос для проверки таблиц и соответствующих типов данных LOB с использованием представления DBA_LOBS:

```
set lines 200
col column_name for a30
SELECT
 table_name,
 column_name,
 segment_name,
 securefile
FROM dba_lobs
WHERE table_name like 'TEST%';
TABLE_NAME                     COLUMN_NAME           SEGMENT_NAME              SECURE
------------------------------ --------------------- ------------------------- ------
TEST1                          COL                   SYS_LOB0000088862C00001$$ YES
TEST2                          COL1                  SYS_LOB0000088867C00001$$ NO
```

4. Использовать простой цикл в анонимном блоке Procedural Language/Structured Query Language (PL/SQL), запустить следующий тест с целью генерации различных произвольных значений с помощью операции конкатенации для вставки 1 миллиона строк в оба типа LOB-сегментов. Разумеется, перед выполнением этого теста необходимо убедиться в наличии достаточного места в табличных пространствах для размещения LOB-сегментов и соответствующей группы дисков, управляемой механизмом Automatic Storage Management (ASM):

```
SET TIME ON
SET TIMING ON
TRUNCATE TABLE test1;
TRUNCATE TABLE test2;
-- Load TEST1
TT@ORCL > BEGIN
   FOR v_Count_1 IN 1..1000000 LOOP
     INSERT INTO TEST1(col1) VALUES ('testInsertColLOBTest2'||v_count_1);
     commit;
```

```
   END LOOP;
  END;
  /
PL/SQL procedure successfully completed.
Elapsed: 00:03:00.40
-- Load TEST2
TT@ORCL > BEGIN
   FOR v_Count_1 IN 1..1000000 LOOP
     INSERT INTO TEST2(col1) VALUES ('testInsertColLOBTest2'||v_count_1);
     commit;
   END LOOP;
  END;
  /
22:00:55   2  22:00:55   3  22:00:55   4  22:00:55   5  22:00:55   6  22:00:55   7
PL/SQL procedure successfully completed.
Elapsed: 00:09:11.61
```

Проведенные тесты показывают, что вставка данных в SECUREFILE LOB-объект эффективнее более чем в три раза (180 с против 551 с), чем вставка тех же данных в SECUREFILE LOB-объект в базе данных Oracle 11g Release 2.

5. Следующий тест позволяет убедиться в отсутствии существенного различия в производительности при обновлении записей для LOB-объектов `BASICFILE` и `SECUREFILE`:

```
TT@ORCL > BEGIN
   FOR v_Count_2 IN 1..1000 LOOP
     update TEST2 set col1 = 'testInsertColLOBTest2'||v_count_2 where rownum <100;
     commit;
   END LOOP;
  END;
  /
PL/SQL procedure successfully completed.
Elapsed: 00:00:03.93
TT@ORCL > BEGIN
   FOR v_Count_2 IN 1..1000 LOOP
     update TEST1 set col1 = 'testInsertColLOBTest2'||v_count_2 where rownum <100;
     commit;
   END LOOP;
  END;
  /
PL/SQL procedure successfully completed.
Elapsed: 00:00:03.62
```

6. Тем не менее важно помнить, что после обновления и/или удаления данных сегмент LOB с большой вероятностью может стать сильно фрагментированным. Поэтому рекомендуется уплотнить (дефрагментировать) LOB с помощью следующей команды:

```
-- For Oracle Database 10.2 and above:
ALTER TABLE <table name>
MODIFY LOB (<lob column name>) (SHRINK SPACE [CASCADE]);

-- For Oracle Database 10.1 and below:
ALTER TABLE <table name>
MOVE LOB (<lob column name>) STORE AS (TABLESPACE <tablespace name>);
```

Преобразование BASICFILE LOB-объектов в SECUREFILE LOB-объекты

Преобразование BASICFILE LOB-объектов в SECUREFILE LOB-объекты можно выполнить с помощью одной простой команды:

```
TT@ORCL > ALTER TABLE test2 MOVE LOB (col1) STORE AS SECUREFILE (TABLESPACE users);

Table altered.

Elapsed: 00:00:23.54
```

Но у этого простого метода преобразования есть один недостаток – после завершения операции преобразования LOB-объект становится недоступным для любой операции языка обработки данных (DML). Другим способом преобразования LOB-объекта из формата BASICFILE в формат SECUREFILE в режиме ONLINE является использование пакета DBMS_REDEFINITION, демонстрируемое в следующем примере:

1. Если учетная запись пользователя, намеревающегося выполнить преобразование LOB-объекта, не обладает привилегиями SYSDBA, то необходимо выполнить следующую команду для временного предоставления данной учетной записи привилегий, достаточных для выполнения этого преобразования:

```
-- Create the migrating user (if it doesn't yet exist)...
grant dba to tt identified by tt123;

-- ... or grant the existing user account the necessary specific privileges
-- so it can use DBMS_REDEFINITION
grant execute on dbms_redefinition to tt;
grant alter any table to tt;
grant drop any table to tt;
grant lock any table to tt;
grant create any table to tt;
grant select any table to tt;
grant create session to tt;
```

2. Создать таблицу для данного примера. Необходимы таблица, которая будет преобразована (test3), и таблица, которая будет использоваться в процессе преобразования (test4), то есть временная или промежуточная таблица:

```
CREATE TABLE test3 (
col1  NUMBER PRIMARY KEY,
col2 CLOB
); 23:03:10   2  23:03:10   3  23:03:10   4

Table created.

Elapsed: 00:00:00.11
```

3. Для демонстрации процедуры преобразования вставить произвольные данные в таблицу test3:

```
TT@ORCL > BEGIN
   FOR v_Count_2 IN 1..10000 LOOP
     INSERT INTO TEST3(col1,col2) VALUES (v_count_2,'testInsertColLOBTest3'||v_count_2);
     commit;
   END LOOP;
```

```
END;
/
PL/SQL procedure successfully completed.

Elapsed: 00:00:01.82
```

4. Создать промежуточную таблицу (test4), которая будет работать как таблица временного хранения данных, вставляемых, удаляемых или обновляемых в процессе выполнения преобразования в режиме онлайн:

```
TT@ORCL > CREATE TABLE test4 (
col1 NUMBER NOT NULL,
col2 CLOB
) LOB(col2) STORE AS SECUREFILE (NOCACHE);

Table created.

Elapsed: 00:00:00.05
```

5. Начать переопределение таблицы test3, используя процедуру START_REDEF_TABLE из пакета DBMS_REDEFINITION:

```
23:07:09 TT@ORCL > DECLARE
   col_mapping VARCHAR2(1000);
BEGIN
   col_mapping := 'col1 col1, '|| 'col2 col2';
   DBMS_REDEFINITION.START_REDEF_TABLE('tt', 'test3', 'test4', col_mapping);
END;
/

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.97
```

6. Для уверенности в том, что все ограничивающие условия скопированы из исходной таблицы в промежуточную, нужно вызвать процедуру COPY_TABLE_DEPENDENTS из пакета DBMS_REDEFINITION, как показано ниже:

```
23:07:42 TT@ORCL > DECLARE
   error_count pls_integer := 0;
BEGIN
   DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
  uname=> 'tt',
  orig_table=> 'test3',
  int_table=>  'test4',
  copy_indexes=> DBMS_REDEFINITION.CONS_ORIG_PARAMS,
  copy_triggers=> TRUE,
  copy_constraints=> TRUE,
  copy_privileges=> TRUE,
  copy_statistics=> FALSE,
  num_errors=> error_count);
   DBMS_OUTPUT.PUT_LINE('errors := ' || TO_CHAR(error_count));
END;
/

PL/SQL procedure successfully completed.

Elapsed: 00:00:05.89
```

7. Завершить переопределение таблицы test3 в таблицу test4 с помощью процедуры FINISH_REDEF_TABLE из пакета DBMS_REDEFINITION:

```
23:08:10 TT@ORCL > EXEC DBMS_REDEFINITION.FINISH_REDEF_TABLE('tt', 'test3', 'test4');

PL/SQL procedure successfully completed.

Elapsed: 00:00:02.52
```

8. Убедиться в том, что столбец в таблице test3, который изначально являлся BASICFILE LOB-объектом, теперь имеет тип SECUREFILE LOB. Для этого выполнить запрос к таблице DBA_LOBS:

```
23:18:33 TT@ORCL > select owner,table_name,column_name,securefile
from dba_lobs where table_name='TEST3';

OWNER       TABLE_NAME            COLUMN_NAME           SEC
----------  --------------------  --------------------  --------------
TT          TEST3                 COL2                  YES

Elapsed: 00:00:00.05
23:18:53 TT@ORCL >
```

Другим способом выполнения массового эффективного преобразования в режиме онлайн LOB-объектов из формата BASICFILE в формат SECUREFILE является использование операции CTAS (create table as as select) в сочетании с инструментом Oracle GoldenGate для репликации данных в той же исходной и целевой базах данных. При этом имеется возможность синхронизации содержимого исходных и новых таблиц после операции CTAS, когда преобразование таблиц полностью завершено. После завершения преобразования приложение приостановит свою работу лишь на короткий интервал времени, в течение которого будут переименованы исходные и новые таблицы.

Воздействие параметра PCTFREE на LOB-объекты

Установка правильного значения для параметра PCTFREE чрезвычайно важна для того, чтобы избежать нерационального использования пространства при создании таблиц со столбцами, содержащими встроенные LOB-объекты. Параметр PCTFREE определяет минимальный процент свободного пространства в блоке, который СУБД Oracle резервирует при обновлении существующих строк в таблице, то есть главным образом для того, чтобы исключить ненужные перемещения строк, когда после обновления и увеличения длины фрагмента строки он не умещается в блоке.

По умолчанию для параметра PCTFREE устанавливается значение 10 процентов, но достаточно часто это значение увеличивают, чтобы предотвратить перемещения строк. Например, если для какой-либо таблицы установлено значение параметра PCTFREE, равное 30, то Oracle обязательно зарезервирует 30 процентов блока для обеспечения возможности увеличения длины строк при их обновлении. Если при вставке нового фрагмента строки в такой блок будет занято 70 процентов пространства, то этот блок помечается как заполненный, и следующая строка будет вставлена в другой блок с достаточным свободным пространством.

Но определение значения, большего, чем действительно необходимо, для параметра PCTFREE может привести к ситуации, в которой большие объемы свободного пространства просто никак не используются, когда таблица содержит столбцы со встроенными LOB-объектами. В следующем примере наглядно показано, как более рационально использовать табличное пространство в подобной ситуации:

1. Создать простую таблицу со столбцом, содержащим LOB-объект, и установить для параметра PCTFREE значение 40 процентов, вставить 100 000 строк в эту новую таблицу, затем проверить счетчик строк и собрать статистические данные для оптимизации:

```
SYS@ORCL AS SYSDBA> create table tt.test_pctfree (col1 clob) PCTFREE 40;

Table created.

SYS@ORCL AS SYSDBA> BEGIN
   FOR v_Count_2 IN 1..100000 LOOP
    INSERT INTO tt.test_pctfree
          VALUES ('testInsertColLOBTest2'||v_count_2);
    COMMIT;
   END LOOP;
END;
/

SYS@ORCL AS SYSDBA> select count(*) from tt.test_pctfree;

  COUNT(*)
----------
    100000

SYS@ORCL AS SYSDBA> EXEC DBMS_STATS.GATHER_TABLE_STATS ('TT', 'TEST_PCTFREE');

PL/SQL procedure successfully completed.
```

2. Проверить количество блоков и средний размер блока в этой таблице:

```
SYS@ORCL AS SYSDBA>
select
  blocks,
  avg_space,
  pct_free
from
  dba_tables
where
  table_name='TEST_PCTFREE';

BLOCKS     AVG_SPACE  PCT_FREE
---------- ---------- ----------
      1504       3362         40
```

3. Удалить все строки из таблицы и изменить для нее значение параметра PCTFREE на 0 (нуль):

```
SYS@ORCL AS SYSDBA> truncate table tt.test_pctfree;

Table truncated.

Elapsed: 00:00:00.07

SYS@ORCL AS SYSDBA> alter table tt.test_pctfree pctfree 0;

Table altered.

Elapsed: 00:00:00.01
```

4. Еще раз вставить то же количество строк и снова собрать статистические данные:

```
SYS@ORCL AS SYSDBA> BEGIN
   FOR v_Count_2 IN 1..100000 LOOP
     INSERT INTO tt.test_pctfree VALUES ('testInsertColLOBTest2'||v_count_2);
     COMMIT;
   END LOOP;
END;
/

PL/SQL procedure successfully completed.

Elapsed: 00:00:14.13

SYS@ORCL AS SYSDBA> EXEC DBMS_STATS.GATHER_TABLE_STATS ('TT', 'TEST_PCTFREE');

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.22
SYS@ORCL AS SYSDBA>
```

5. Проверить новые значения размеров блоков и размер пространства, которое в среднем занимает отдельный блок:

```
SYS@ORCL AS SYSDBA> select
blocks,
avg_space,
pct_free
from
dba_tables
where
table_name='TEST_PCTFREE';
  2    3    4    5    6    7    8
BLOCKS     AVG_SPACE  PCT_FREE
---------- ---------- ----------
       874        303          0

Elapsed: 00:00:00.00
```

Этот пример показывает, что количество занятых блоков существенно уменьшается: 874 при значении PCTFREE, равном 0 процентам, против 1504 при значении PCTFREE, равном 40 процентам, то есть эффективнее приблизительно на 41 процент. Использование больших значений для параметра PCTFREE в таблицах со столбцами LOB-объектов может приводить к бесполезной потере гигантского пространства, именно поэтому некоторые табличные пространства с LOB-объектами становятся непомерно огромными, что затрудняет их реорганизацию. Если в таблице, содержащей LOB-объекты, пространство пока еще не распределено, то администратор СУБД Oracle получает отличную возможность полностью контролировать запросы по выделению пространства для этой таблицы, так как в дальнейшем, вероятнее всего, практически невозможно будет найти время, достаточное для создания этой таблицы заново, даже при использовании пакета DBMS_REDEFINITION.

В особенности важно отметить, что процедура сокращения бесполезно расходуемого пространства в таблице с LOB-объектами применима только к встроенным LOB-объектам. В примере, приведенном выше, вставляемые данные имели относительно небольшой размер, и большинство этих данных сохранялось непосредственно в таблице, поэтому на общий размер таблицы оказывало существенное влияние значение параметра PCTFREE. Но для внешних LOB-объектов значение PCTFREE не является столь важным, потому что пространство LOB распределяется

и обслуживается с учетом фрагментов как единиц измерения (размер фрагментов составляет несколько блоков), а не на основе последовательного выделения блоков, как в сегменте таблицы.

Несмотря на то что типы данных LOB (BLOB, CLOB, NCLOB и BFILE) не используют параметра таблиц хранения PCTFREE и списков свободных блоков для управления свободным пространством, можно применить аналогичную стратегию для управления наращиванием пространства LOB-объекта, определяя размер фрагмента при работе с внешними LOB-объектами. Таким образом, следует установить значение размера фрагмента, превышающее средний размер данных LOB, если ожидаются обновления, увеличивающие объем данных. В противном случае возможно постоянное перемещение данных LOB, которое значительно снизит производительность приложения, поскольку для каждого запрошенного LOB-объекта операций извлечения будет выполнено вдвое больше, по сравнению с количеством фрагментов в этом LOB-объекте.

И последнее, но очень важное предупреждение, которое следует запомнить: размер фрагмента также влияет на размер данных в операции redo, генерируемых при обработке LOB-объектов средствами языка управления данными. Запись в операции redo обязательно должна быть сгенерирована для всего фрагмента в целом, поэтому размер фрагмента напрямую влияет на общий объем данных, генерируемых операцией redo, следовательно, определяет соответствующее воздействие на производительность языка управления данными. Таким образом, важно быть уверенным в том, что установленный размер фрагмента не превышает действительно необходимого значения.

Улучшение производительности операции вставки данных INSERT

Если в коде приложения неожиданно возникает проблема снижения производительности при выполнении операций вставки данных INSERT после преобразования LOB-объектов из формата BASICFILE в формат SECUREFILE в версии Oracle Database 11g Release 2, то следует знать, что с большой вероятностью это проявление серьезной ошибки в версии 11.2.0.1, которая описана в официальном документе MOS Note 1323933.1 «Securefiles performance appears slower than basicfiles LOB» («Производительность LOB-объектов в формате securefile снижается по сравнению с форматом basicfile»).

Чтобы проверить, что эта ошибка действительно является причиной снижения производительности операций INSERT (и быстро устранить эту проблему), выполните следующую команду, устанавливающую корректное значение параметра _kdli_sio_fileopen на уровне текущего сеанса работы перед продолжением выполнения команд INSERT в таблице, содержащей один или несколько столбцов с LOB-объектами в формате SECUREFILE:

```
SQL> alter session set "_kdli_sio_fileopen"='nodsync';
```

Документ MOS Note 1323933.1 настоятельно рекомендует применить соответствующее исправление для указанной версии СУБД, чтобы полностью устранить эту проблему. В заключение отметим, что в версии 11.2.0.3 эта ошибка исправлена.

Резюме

Из этой главы вы узнали, что чем больше таблиц со столбцами LOB в базе данных, тем выше вероятность возникновения проблем, связанных с производительностью при обработке и сопровождении LOB-объектов. Необходимо всегда выполнять следующую рекомендацию: при создании новой таблицы со столбцами LOB попытайтесь использовать оптимальные параметры хранения, которые полностью соответствуют поведению соответствующей таблицы:

- для таблиц, которые никогда не будут обновляться, без сомнений устанавливайте для параметра `PCTFREE` значение «нуль» (0);
- если использование типа LOB не продиктовано действительной необходимостью, то вместо него следует рассмотреть возможность использования типов `VARCHAR` или `RAW`.