Note 60233 - Oracle rollback segments, more information.

Note Language: English **Version: 32** **Validity:** Valid from 27.09.2004

Summary

Symptom
Background information and explanation of rollback segment problems.

More Terms
ORA-01555 ORA-01628 ORA-01650 FET PRS_1 PRS_2 PRS_3 PRS_4 PRS_5 PRS_6
PRS_7 PRS_8 PRS_9 PRS_10 RBS rollback segment max extents
snapshot PSAPROLL optimal

Cause and Prerequisites
This is an ADJUNCT to notes 16277 and 3807 not a replacement
for them, it is VITAL you read the applicable notes. This note is long
explanation of what rollback segments are and do, covering the background
needed to understand the resolution of rollback errors.
After the background information I have included a Question and Answer
section for common questions asked in regard to rollback segments.

***** After reading this note please feel free to send me any corrections**

***** or comments to "glenn.cadman@sap-ag.de" .**

***** Any questions to be included in the Q&A section are also welcome.**

***** Unfortunately I only speak English and a little Japanese so if**

***** possible can you mail me in either English or Japanese.**

Solution

Oracle uses rollback segments for 2 purposes :

Purpose 1 : Provision to undo incomplete database changes.

The first and most important use is to insure that in case of a problem,
update/insert/delete transactions can be undone safely, preventing logical
inconsistencies with the data.

To understand this lets look at a simple example :
Imagine a bank transfer of $50, from a company called PAS, to a company
called Elcaro. For this to be successful the following changes must be made
:
1. Debit $50 from the balance of PAS's account
2. Write a record of the debit to PAS's history table.
3. Credit $50 to the balance of Elcaro account.
4. Write a record of the credit to Elcaro's account history table.

In a normal situation the database will update PAS's account balance record, insert a debit record into PAS's history table, update Elcaro's balance record and finally insert a credit record into Elcaro's history table. Thus to perform a transfer at least 4 separate database changes must be made.

Imagine the database updated PAS's account with the $50 debit (1.) before the DBA killed the transfer program leaving steps (2.) (3.) and (4.) uncompleted. Obviously a debit does not have any meaning without a corresponding credit, hence a mechanism is needed to ensure that the debit is not permanently saved without the corresponding credit and history records also being permanently saved.

This concept is the basis of a TRANSACTION: a logical unit work that is either successfully completed and saved (committed), or it is unsuccessful (fails) and all changes made by the transaction are undone/removed. Undoing of database changes by failed transactions is know as a ROLLBACK.

Now the question is:

"What is the mechanism Oracle uses to enable the rollback (undoing) of changes made by unsuccessful transactions ?"
The answer is simple, oracle writes a COPY of the way the data was like BEFORE making changes. If a transaction fails or is canceled for any reason, then this copy used to OVERWRITE the changed database entries with the original database entries. The copies are know as ROLLBACK ENTRIES, and are stored within structures called ROLLBACK SEGMENTS. Rollback entries that form part of an uncompleted transaction are known as "active entries" as they form part of an "active" transaction. "Active entries" are not released or reused until the transaction is completed. When a transaction is completed the rollback entries are tagged as completed and the rollback segment space made available for reuse. The completed entries are known as "released entries".

Please note that even though rollback entries are "released", the entries are not NOT ERASED or DELETED until the rollback segment has completed a full cycle and ORACLE starts overwriting the "released entries" with new "active entries".

You can visualize a rollback segment as a circular entity. I like to think of rollback segments in terms of a circular railroad track. A train travels around the track never reaching the end, because as soon as it does, it is back to its starting point, the track behind the train was used the shortest time ago, the track ahead was used the longest time ago. With rollback segments, ORACLE has a pointer to the current position in the segment, behind the pointer lay the rollback entries just created, in front are the rollback entries written longest ago. Once ORACLE reaches the last extent of the rollback segment it loops back to the first extent overwriting the entries it wrote during the last cycle (unless there are still "active entries" remaining in the extent, more about that latter!).

When a transaction is begun, it is assigned to ONE rollback segment, ALL rollback entries generated by that transaction, are ONLY written that rollback segment. The rollback segment typically assigned is the one with the fewest active transactions (uncompleted updates / inserts / deletes). Using the train analogy, when a passenger books a ticket they are assigned to a particular train, once assigned a train the passenger cannot change trains or tracks until they have completed the trip.

Large update transactions can generate more rollback entries than can stored within the assigned rollback segment. If this happens, ORACLE tries to increase the size of the rollback segment, to allow more active rollback entries to saved. This is done by dynamically adding extents (chunks of free space) to the rollback segment requiring additional space. Using the train analogy, if the track ahead is in use, then new track is inserted to prevent the train stopping.

When ORACLE is unable to add addition extents the transaction will be canceled (rolled back) and one of the following messages returned to the program :

ORA-01650 unable to extend rollback segment xxx by nnn in tablespace PSAPROLL
ORACLE speak for:
"I am unable to allocate a chunk of space for a new rollback segment extent because I can find no contiguous chunk of storage of that size (nnn 8K blocks) in the PSAPROLL tablespace's datafiles, please free up space or add a new datafile to your PSAPROLL tablespace"

ORA-1628 max # of extents nnn reached for rollback segment xxx
ORACLE speak for:
"I am unable to allocate any further extents to this rollback segment because the maximum number of extents limitation defined in the storage parameters of the rollback segment has been reached"

See notes 16277 and 3807 for resolution and further explanation of the above errors.

It is important to note that only transactions that CHANGE data generate rollback entries. Transactions that do not change data are know as "read-only" transactions and by definition don't generate rollback entries. Thus it is impossible for read-only transaction to generate the above oracle errors.

Purpose 2: Maintaining a read consistent view of the database.

Using our example of a $50 PAS to Elcaro transfer, imagine that the banks monthly account statement generator/report is run at the SAME TIME as the transfer. As bank customers we expect that the credit and debits shown in our statement history will reconcile to our account balance. The statement generator may perform the following task list:
1. Get all account and account balances from account balance table.
2. For each of the accounts from 1. obtain all the credit /debit history records.
3. Get the name and address of the account holder.
4. Sort and format the above information and print statement.

If the account transfer program added history records between the statement report obtaining the account balances (1.) and selecting the debit/credit history records (2.) then the credit/debit list could find extra records which would be inconsistent with the account balance. (because those debit/credit records where added after the account balance was extracted)
Using our example the PAS's statement may show an addition $50 debit inconsistent with PAS's account balance.

Thus the database needs a mechanism to allow transactions to VIEW a CONSISTENT SET of DATA, unaffected by changes made by other transactions during the time the transaction is active.
This facility is known as READ CONSISTENCY.

One method to facilitate read consistency would be to wait until all the tables the transaction needs to access are "free" from any updates, then lock all them against any further changes until the transaction has finished running. An effective method, but this would cause processes to wait for a turn to lock required tables, seriously handicapping database performance.

However there is another possibility to maintain read consistency ! Remember, ORACLE has a copy of the old data stored in the rollback segments as rollback entries ! Even after a transaction is completed, and the rollback entries are released for reuse, they will remain accessible until being overwritten during the next cycle of the rollback segment. Thus by combining a database table read with a rollback segment entries read, ORACLE can build a view the data, consistent with the point in time at which the transaction was started. However, this view can only be maintained for a TEMPORARY period, because the rollback segment will cycle and the rollback entries needed to maintain "read consistency" will be overwritten. During a database read a timestamp is checked to determine if changes have been made since the transaction started, any changed data is ignored and the corresponding historical rollback entry is read from the rollback segment. NOTE that ORACLE uses rollback BOTH "active" and "released" rollback entries, as both may contain data needed to build read consistency. Before overwriting released rollback entries, ORACLE does not and cannot check if current read transactions will need or not need the data.

Using the example of the banks statement report, ORACLE will read the transaction history table entries, however a timestamp indicates that changes have been made since the transaction started, that data is then ignored, and the old data is retrieved from its corresponding rollback entry. Thus the read of the debit/credit history table is made consistent with the time at which the statement generator started running.

To achieve read consistency oracle MUST be able to read EVERY relevent rollback entry created since the transaction started. If ANY rollback segment completes a cycle, essential rollback entries could be overwritten, then ORACLE will unable to deliver read consistency (a SNAPSHOT), in which case ORACLE will fail the transaction and generate the following error:

ORA-1555 snapshot too old (rollback segment too small)
ORACLE speak for :
"I cannot build a read consistent view of the data you require because other processes have generated enough rollback entries to cause just one rollback segment to cycle, overwritting necessary data, during the period your transaction has been running. Because I cannot now create a read consistent view of the data you require, I terminated the transaction"

How to resolve ORA-1555 error ? , LOOK AT NOTE 3807!

*** This is to adjunct the information in note 3807 not a replacement of it.**

To understand the resolution in note 3807 (ORA-1555) lets go back to our analogy of a train traveling around a loop of track. Any solution involves increasing the period before the train completes a circuit and then reuses same piece of track. I can think of three ways to do this:

1. Slow down the train. Run the report when less update activity is happening e.g. over the weekend / at night etc. With less update activity happening, less rollback entries are being written, thus increasing the period before a rollback segment will cycle. Remember rollback entries are only created by transactions make changes to the database (insert, update or delete, known as DML statements, Data Manipulation Language), so separating all your reporting operations from your updating operations is very effective.

2. Lengthening the track. By increasing track length the train will take longer before completing a full circuit. Increasing the size of the rollback segments will increase the cycle time of the rollback segments. A 100MB rollback segment will hold 10 times the number of rollback entries, hence 10 times historical data as a 10MB segment. Thus a 10 fold increase in rollback segment size should give a 10 times increase in the time a read transaction can run before a rollback segment is cycled and the dreaded ORA-1555 is generated.

3. Increase the number of tracks, reducing the load on any individual track. Having more rollback segments (of the at least the same size) will spread load of rollback entries, increasing the cycle time of individual rollback segments. Having 100, 10MB rollback segments should give 10 fold increase in the period ORACLE can maintain a read consistent view of the database compared to 10, 10MB rollback segments

*** * * * * Questions and Answers regarding rollback segments * * * * * ***

Q. Why are my rollback segments not automatically shrinking back to the optimal size ?

A. After completion of a large update transaction, extents that have been dynamically added are not automatically deallocated, as this would reduce the period that ORACLE can maintain read consistency. Before shrinking rollbacks, ORACLE first must allocate another update transaction to the same rollback segment, then this transaction must generate enough rollback entries for ORACLE to advance the rollback pointer to the boundry of the current extent. The next extent is then deleted IF AND ONLY IF the follow 3 conditions are meet :
1.) The next extent does not contain ANY active rollback entries.
2.) The total size of the rollback segment is greater than optimal.
3.) Deleteing the next extent will not reduce the total size of the rollback segment to less than the optimal size
ONLY THEN will ORACLE release the next extent and hence reduce the size of the rollback segment. Thus you need update activity by additional transactions before rollbacks are shrunk (strange but true !). If the next extent DOES contain active rollback entries ie. 1.) is false, then ADDITIONAL exents will be added to the rollback segment and it will grow even bigger.

To force the shrink of a rollback segment you can use the command
(as of Oracle 7.3 and above inc Oracle8)

```
alter rollback segment <seg> shrink;
eg. SVRMGR> alter rollback segment RBS_1 shrink;
this will shrink rolback segment RBS_1 to its optimal size if there
extents free of "active transactions".
```

Q. To avoid getting ORA-1555 I increased the optimal size of my rollback segments, but I still got an ORA-1555 error, WHY ?

Why did increasing the optimal size of my rollbacks not cause them to grow bigger ?
A.

***** INCREASING OPTIMAL SIZE OF ROLLBACKS GENERALLY *****

***** DOES NOT HELP IN AVOIDING ORA-1555 ERRORS !!!! *****

Increasing the optimal size of rollback segment does not cause ORACLE to automatically increase there size unless a large transaction generates enough rollback entries to cause the extension of the rollback segments to that size. Setting the optimal size just sets a "high water mark", thus oracle will shrink a large rollback segment back to its optimal size, but not make a smaller rollback segment grow to its optimal size. To make a rollback segment bigger look at the following example using sqldba:

```
SQLDBA> alter rollback segment prs_01 offline;
SQLDBA> drop rollback segment prs_01;
SQLDBA> create rollback segment prs_01 tablespace PSAPROLL storage (
initial 1M next 1M optimal 100M minextents 100 maxextents 500 );
SQLDBA> alter rollback segment prs_01 online;
```

***** IT IS THE COMBINATION OF "next" SIZE *****

***** AND "minextents" THAT FORCES ORACLE *****

***** TO CREATE BIGGER ROLLBACK *****

Q. Why set all rollbacks the same size?
A. ALL (ALL means every one) rollback segments should be set to the same size (the exception is the "SYSTEM" rollback segment as this is used exclusively for internal oracle's processing). Remember that if you receive an ORA-1555 it means that JUST ONE rollback segment has been completely recycled during the run time of the read consistent transaction. Thus having nine 100MB rollback segments and a single 1MB rollback segment will be NO MORE as effective in preventing 1555's as JUST HAVING ten 1MB rollback segments.

Q. How can I check the sizing of the rollbacks?
A. You can look at rollback segment sizing using your unix ora<SID> login, starting sqldba, connect internal and then issuing the command:

```
SQLDBA> select a.name, b.optsize, b.hwmsize
         from  v$rollname a,
               v$rollstat b
        where a.usn = b.usn;
(See note 82634 )
Or you can use transaction st04 (30+)
Tools -> Admin -> Monitoring -> Performance -> Database -> Activity
```

Then -> Press "Detailed Analysis Menu" -> Press "Display V$tables" -> Choose rollback segments (V$rollstat)

Q. How can I check how long it is taking for a rollback segment to cycle so I can then estimate the how big to make my rollback segments ?
A. Good question! if you find a method let me know, as far as I can tell you can only guesstimate it by looking at the rollback write and "wrap" statistics from table v$rollstat.

Q. Why is having bigger rollback segments, more effective in prevention of 1555's, than having more rollbacks of the same size ?
A. The smaller a rollback segment is, the greater the chance that a large update process can generate enough rollback entries to cause a rollback segment to cycle. Hence from the point of view of preventing 1555's, increasing the SIZE of the rollbacks is MORE effective than increasing the NUMBER of equal sized rollback segments. I am not saying that increasing the number of same size rollback segments is not an effective method of resolving 1555's, it is just not quite as effective (especially when update transactions occur that generate enough rollback entries to require additional extents to your current rollback segments).

Q. I have increased the size of my rollbacks and I have increased the number of rollbacks segments and I still get 1555's what do I do now ?
A. Apart from running the report at a different time (when less update is happening) you just have to keep increasing the size and perhaps numbers of rollback segments until the 1555 is not generated. At one site I know they needed a total of 3.5 GB of rollbacks to prevent 1555's. If you are having serious problems with long running reports and rollback segments an Earlywatch session will help identify and hence recommend possible solutions for your site. (Tip, double check that all rollback segments are actually the same size AND are at least the SIZE of your OPTIMAL size setting. This is a major area of confusion, the problem is not HOW big the rollbacks can GROW TO, but HOW big the rollback segments ARE NOW !. So first check that all your rollback segments are already BIG! see above question on how to check each rollback segments ACTUAL size)

Q. I have an emergency situation! An internally developed ABAP report fails with an ORA-1555 error. It works in the testing environment, but in production it runs for several hours before terminating with an ORA-1555 error. Management NEEDS the report URGENTLY and I am responsible to make it work, PLEASE HELP ME SAP !
A. DONT PANIC! The real "long term" solution to this problem is to look at why the report is taking so long to run (SQL trace, indexing, use temp tables etc.) but of course this takes TIME and your management needs the report yesterday! So basically you need a temporary fix while your developer tunes the ABAP (better indexes, temporary tables, procedure flow, sub reports, closing and reopening cursors) to run faster reducing the chance of any rollback segments looping

TWO TEMPORARY WORKAROUNDS FOR ORA-1555's ERRORS.

Fix 1.) "DO IT WHEN NOBODY ELSE IS DOING IT"

Can the report be run during a period of little or no other SAP system activity ? eg. at night on the weekend. If you have a time window of lowsystem activity, then run the report then! EASY!.

If the rollback segment pointers are stopped or moving very slowly the R/3 server will have more "real" time in which to run the report to completion before one of the rollback segment completes a loop. Imagine that a rollback segment as a clock with a second hand, however unlike a normal clock the second hand can speed up or slow down depending on the volumn of modifications being made to the database. By running the report when the second hand is moving slowest, hence taking longer to complete a cycle, read consistency should be maintained until the report completes.

Fix 2.) "BIGGER IS BETTER".....for ORA-1555, but not for anything else.

If Fix 1 is impossible, then to prevent rollback segments looping bigger rollback segments are needed (allocating datafile blocks to rollback segments => longer read consistency period). Because the exact sizing of the rollback segments cannot be predetermined, I recommend you allocate any spare disk space available to temporarily increase the size of your rollback segments. The "bigger is better" method, basically means "Using spare disk space to create very big rollback segments". Using the clock analogy, the "ticking" of the second hand remains constant, but we add more seconds to the minute (eg. from 60 to 500 seconds to a minute). Again by increasing the time it takes before any rollback segment completes a cycle, the longer oracle can maintain read consistency, hence the longer the report can run before failing with an ORA-1555.

WARNING: The following is not in any way recommemend by SAP. This is a temporary workaround to avoid ORA-1555 errors. If you do not understand what is being done then DON'T TRY IT on your system ! If you are not confident using sapdba and sqldba, again don't try it !

The example below is based on a UNIX system using normal file systems. If your operating system is NT or your filesystems are raw disk partitions then the method is the same, however the commands being used will be different.

STEP 1. Determine how much free space is available.
Check the sapdata filesystems, can new sapdata file systems be created using spare disks or spare disk space from volumn groups ? Are there other non sapdata file systems with large chunks of free space ?

```
  EXAMPLE
   sapdata filesystem have the following :
     sapdata1 has 500M free
     sapdata3 has 300M free
     sapdata5 has 100M free
     sapdata6 has 400M free
   Examination of non sapdata file systems:
     /usr/sap/old has 1GB free
   Examination of free disks or disk volumns:
     show that a 2GB disk is free and available for temporary use.
   TOTAL space available for temporary use :
     500+300+100+400+1000+2000= 4300MB
```

STEP 2. Assign additional space from step 1. to new sapdata file systems. For non sapdata file systems on unix you can use symbolic links to create temporary sapdata directores

```
eg. ln -s /usr/sap/old /oracle/PRD/sapdata10
For spare disks etc. you will have to use the operating system utilities (
sam, smitty etc. see your installation guide for details ) to format and
mount new sapdata filesystems eg. /oracle/PRD/sapdata11.

STEP 3. Now add the additional space the PSAPROLL tablespace.
sapdba -> c) tablespace adminstration -> a) Tablespace , enter PSAPROLL ->
f) Alter tablespace PSAPROLL Add Datafile.
In the above example your will create 6 additional datafiles, 4 on the
existing sapdata filesystems, 1 in the symbolic linked directory
/usr/sap/old, 1 on the spare disk mounted as a new sapdata filesystem.

STEP 4. Now determine the exact size of your PSAPROLL tablespace :

Method 1: Operating System Method.
sapdba -> c) Tablespace adminstration -> c)  Free space and fragmentation
of all tablespaces  -> Find figure of "Total" column for the PSAPROLL
tablespace (in Kbytes)

Method 2: From SAP menu -> Adminstration -> Monitoring -> Performance ->
Database -> Tables/Indexes -> (Database system) Space statistics ->
Tablespaces Look at Total size for PSAPROLL tablespace.
 Scale: Day        Size (Kbyte)
   Tablespace      Total
PSAPROLL           4,856,000 (this is 4.8GB )

STEP 5. Find out how many rollback segments "PRS_x" you have in use ? (See
question "How can I check the sizing of the rollbacks" on how to see this)
eg. 10 rollback segments PRS_1, PRS_2, PRS_3, PRS_4, PRS_5, PRS_6, PRS_7
PRS_8, PRS_9, PRS_10

STEP 6. Calculate the maximum size you can make each rollback segment eg.
4,856 (MB) / 10 (because 10 rollback segments) = 480 MG each Thus we can
create 10, 480 MB rollback segments.

STEP 7. Create, and run a rollback segment size changing script. For each
rollback segment PRS_1 thru PRS_10 (or whatever number of rollback segments
you have) edit or vi a file eg. /tmp/changeroll.sql inserting the following
lines:
----------------------------------------------
spool changeroll.log
alter  rollback segment prs_1 offline;
drop   rollback segment prs_1;
create rollback segment prs_1 tablespace PSAPROLL
storage ( initial 1M
          next 1M                 <---- Each addition extent is 1MB
          minextents 480          <---- This line is the critical !
          maxextents 500
          optimal 480M   );       <---- Optimal is not so important
alter rollback segment prs_1 online;
alter rollback segment prs_2 offline;
drop rollback segment prs_2;
create rollback segment prs_2 tablespace PSAPROLL
...
...  etc etc upto prs_10 or however many rollback segments you have.
...
spool off;
-----------------------------------------------------
```

```
Note: I recomend you split the script into many sub scripts eg.
/tmp/changeroll_p1.sql, /tmp/changeroll_p2.sql each with a single set of
functions (eg. offline prs_1 thru 10, drop prs_1 thru 10, create prs_1 thru
10) in that way you can fix any problems before the next step is started (
not many people can get a 100 or so line script to run perfectly the first
time ).

Now start the script
sqldba mode=line
connect internal;
@changeroll.sql
...script runs
exit
review (vi changeroll.log) for errors

STEP 8. Rerun your long running report, and hope it works. If it
runs for many many hours, one example ran for 18 hours then crashed, then
there is not much more you can do! I generally cry, kick my PC, then go
down to the pub and drink lots of beer :-). But seriously, if all spare
space available has been added to your PSAPROLL tablespace and created the
bigest rollback segments possible, and the report still crashes with an
ORA-1555, then a serious look at what the report is doing and how to make
it run faster is the only remaining solution.

STEP 9. Return the system back to "normal" once a permanent solution is
found.  Thus you must resize the rollback segments back to the orginal size
and rebuild the PSAPROLL tablespace, this requires R/3 down time !
create a script eg. /tmp/rebuild.sql similar to the following example.

Note: Again I recomend you split the script into many sub scripts eg.
/tmp/rebuild_p1.sql, tmp/rebuild_p2.sql each with a single set of actions
(eg create roll_temp, offline prs_1..thru 10, etc. etc. ) in that way you
can fix any problems before the next step is started.
-----------------------------
set echo on
spool /tmp/rebuild.log
create rollback segment roll_temp tablespace SYSTEM;
alter rollback segment roll_temp online;
alter rollback segment prs_1 offline;
...repeat for prs_2 thru prs_10 (or however many rollback segment you have
in use)
drop rollback segment prs_1;
...repeat for prs_2 thru prs_10 ( or however many rollback segments you
have in use)
drop   tablespace PSAPROLL;
create tablespace PSAPROLL datafile
'/oracle/<yoursid>/sapdata/roll_1/roll.data1' reuse;
create rollback segment prs_1 tablespace psaproll
storage ( initial 1M next 1M minextents 16 maxextents 500 optimal 16M);
...repeat for prs_2 thru prs_10 ( or however many rollback segments you
have in use)
alter rollback segment prs_1 online;
...repeat for however each rollback segment created
alter rollback segment roll_temp offline;
drop rollback segment roll_temp;
spool off;
-------------------------------
sqldba mode=line
```

```
connect internal;
@/tmp/rebuild
exit;
review the /tmp/rebuild.log file for errors.
```

Q. There is plent of free space available in my PSAPROLL tablespace why then do I get ORA-1555's saying that my rollback segment is too small ?

A. If you read the above background in regard to rollback segments you will notice that read transactions do not generate rollback entries, hence it is impossible for such a transaction to cause your rollback segment to extended in size ! (using a chunk of freespace in your PSAPROLL tablespace). A rollback segment containing data data needed by ORACLE to generate a read consistent view was overwritten with new rollback entries generated by an update transaction. This can only occur when a rollback segment cycles during the run time of your failed (1555) transaction. By increasing the ACTUAL size of your rollback segments (minextents, optimal size) you can increase the cycle period of your rollback segments and hence increase the length time ORACLE can maintain read consistency.

Q. When setting the storage parameters for a rollback, why is defining a large number of smaller extents better than 2 large extents.

A. Imagine that a transaction begins, rollback entries start being writing at a point just before one of the rollback segments extent boundaries. While this transaction remains active, that WHOLE extent will be unavailable for reuse by ANY transaction. Thus a rollback segment of 2 equal sized extents in the worse case may have only 50% (1/2) of the total allocated space available to any further writing of rollback entries. However having 20 equal sized extents will mean that 95% (19/20) of the total rollback storage allocation is available to a transaction.

Q. I have a large data load to do, how do I prevent getting ORA-1562 and ORA-1628.

A. Have a look a note 3807 for the solution.

Q. What are the negative impacts of using huge rollback segments or large numbers of rollback segments? ie. having the total size of all rollback segments in the order of 100 megabytes or more (maybe a few gigabytes).

A. For optimal OLTP (online transaction processing) performance, many tiny (a few hundred KB at most) rollback segments work best! Why? Because this will mean that the entire rollback segment will be more likely to be cached/buffered in the SGA (physical memory) thus minimizing expensive reads/writes to the disk. Now before you start reducing all your rollback segments to 16K, remember the smaller the rollback segment, the shorter the time oracle can maintain read consistency, that means lots of ORA-1555's. Also larger updates will cause rollback segment extension, which is even more time consuming for disk reads/writes, locking e tc. I would also like to add that reducing rollback segments to sizes less than default (8 or 16MB) is not supported by SAP , if a SAP TCC consultant says "do it" then OK, but otherwise don't reduce your rollback segments to anything less than default!
Now what about the question increasing Oracle R/3 system's rollback segments well above the recommended default size! Is there a performance penalty to doing this? Or is it just an issue of disk space? For example imagine a site that is getting ORA-1555's and has if required, plenty of disk space to allocate to rollback segments, maybe the administrator is

thinking:
    "I don't care about allocating a couple of GB of disk for rollback
    segments, I just want to kill the ORA-1555 problem! I will make the
    rollback segments REALLY BIG",
This will certainly kill the ORA-1555 problem, but should the administrator worry about any performance degradation of his/her system? Remember the of inserts updates or deletes requested by your R/3 system is not changed by having bigger or smaller rollback segments! Only reducing the number of users, or what they do, will change that! Since going to your board of directors recommending that "limits" be placed on customers ordering stuff from your company or executives running financial reports, will be detrimental to your career, you have little option but to work with it! As the volume of rollback entries written to your rollback segments is independent of rollback segment size, the only remaining consideration is if the rollback segment will be completely cached in the buffer cache (Oracle SGA) or not? If the rollback segment is 100% cached then the updating process won't have to wait for a disk read to read the "tail" of the rollback segment from disk, thus oracle can just immediately write the new rollback segment entries. If the rollback segment is not 100% cached and the tail of the rollback segment must be read from disk anyway, then using very large rollback segments (eg. 500's MB) compared to using large (eg. 50's MB) will make zero difference.
Now my feeling on this subject is that if you must create bigger rollback segments to avoid ORA-1555 then just don't worry about any potential performance degradation resulting from using larger rollbacks. I am interested in anyone who believes or experience have shown them otherwise, send me mail?
If you are super concerned about performance, and if you know when the transactions that require a long period of read consistency are started, then it's feasible to consider createing some large rollback segments that are put "online" and the smaller "normal" rollback segments taken "offline" (alter rollback segment PRS_XX offline;) prior to starting the long running transaction. After the long running transaction is finished the smaller rollback segments are brought online (alter rollback segment PRS_XX online;) again and the large rollback segment(s) then taken offline, this could be done while system R/3 system is up and ru nning for user access. For example I may use this method to perform a large client copy on a QAS (quality assurance system) ! But I would be very cautious about doing this on a productive system!
Again I would like to stress the value of tuning your customizing!
ORA-1555's on production systems in most cases are a wake-up or a larm bell saying "Mr/Ms Administrator, I've got long running transactions on my R/3 system". Creating additional rollback segments and/or bigger rollback segments will prevent ORA-1555's, but I would then start asking some questions "Is what is being run necessary? Why does it take so long? Can the ABAP's be tuned better? Can it be run at a different time? Is there another way of doing this?" poorly tuned ABAP's not only run slowly, but drag down the overall performance of your system. Poorly tuned customizing can and does turn powerful servers into impotent pieces of junk.

Q. Can I run my system permanently with one BIG rollback segment, as per the temporary solution in note 3807 ?

A. It depends on your system and the level of update,insert and delete operations. For example, when a dialog process updates a table eg. VBLOG then rollback segment entries must be created, to do this oracle searches for free rollback segment buffers. If the buffers are being used by other update process then the dialog process must wait until buffers freed. Obviously having processes waiting around is BAD for performance. Adding more rollback segments, increases the number of rollback segment buffers, reduing the possiblity of rollback buffer waits. (issue is called "rollback segment buffer contention") Thus if your system has very few active processes coupled with some very large insert/update/delete tasks then it may be possible to run the system with one rollback segment. However in general I would see no reason to have less than 10 rollback segments online at any one time, by running less that 10 rollback segment could cause serious performance problems on your system. I would never use one rollback segment unless I needed to do some kind of mass data transfer or load as per note 3807.

Q. I have a system with low numbers of active transactions, ie. few uncommited transactions active at any point in time. However the init<SID>.ora parameter "transactions_per_rollback_segment" is set to the default of 20, which is much higher than the average number of transactions active at any point in time. Does this result in oracle assigning a disproportionately high number of transactions to just a few rollback segments ?. If this is true, then the rollback segments with the disproportionately high transaction volumn will cycle very quickly, reducing the period of time oracle can maintain read consistency hence increasing the chance of ORA-1555's errors occuring ?

e.g. consider a system with 20 rollback segments and an average of 43 active transactions and the init<SID>.ora parameter transactions_per_rollback_segment=20. Would this result in the first 20 active transactions to be assigned to RBS_1, with next 20 to RBS_2, and the last 3 to RBS_3, leaving and the other 17 rollback segments unused until at some point in time the number of transactions increased beyond 60 (3 rbs with 20 transactions each)

A. Well my answer is no ! Having a the default value of "transactions_per_rollback_segment" will not impact upon the occurence of ORA-1555. This is because oracle attempts to average out the transaction load to ALL the online rollback segments. Oracle assigns transaction to rollback segments via the following logic:

1) Always assign a transaction to the rollback segment with the least number of active transactions.
2) If two or more rollback segments have the same number of active transactions then choose the rollback that had a transaction allocated the longest time ago.

This strategy will on average keep rollback segment entries for the longest possible time hence lessening the chance of ORA-1555. The parameter transactions_per_rollback_segment I believe serves no purpose in a "normally" configured oracle R/3 system (except for oracle parallel server)

The purpose of the parameter "transactions_per_rollback_segment" is that when combined with the value of parameter "transactions", upon DB instance startup oracle will try to allocate the number of rollback segments needed to retain the the total number of transactions using the value of transactions_per_rollback_segment. e.g. if transactions=105 and transactions_per_rollback_segment=20, oracle will try to allocate 6 rollback segments (rounding up the calculation of 105/20 = 5.25 < 6). HOWEVER this is only in the case in installations that is use "PUBLIC" rollback segments which are not used with standard R/3 systems. A correctly configured SAP system will use the default typeof rollback segments known as "private rollback segments". When using private rollback segments another init<SID>.ora parameter "rollback_segments" explicity defines the

```
names of each of the rollback segment to attach (online) to the oracle
instance upon startup. If you look at your systems init<SID>.ora (again
excepting parrallel server) you may see something like:
rollback_segments = ( prs_1, prs_2, prs_3, prs_4, prs_5,
                      prs_6, prs_7, prs_8, prs_9, prs_10 )
In this case your system with use the 10 rollback segments prs_1 thru
prs_10 and oracle will try to average the transaction load evenly to those
10 rollback segments. The parameter transactions_per_rollback segment will
not have any impact on this allocation of rollback segments nor
transactions to rollback segments.
```

Header Data

Release Status:	Released for Customer
Released on:	07.06.2004 11:12:26
Priority:	Recommendations/additional info
Category:	Help for error analysis
Main Component	BC-DB-ORA Oracle

The note is not release-dependent.

Related Notes

Number	Short Text
489690	CC INFO: Copying large production clients
426021	Archiving object BC_DBLOGS: Termination with Oracle DB
383823	MP38: Restriction according to materials
360087	Update task termination: Rollback segment too small
338222	Balance carryfwd: DBIF_RSQL_ERROR: Snapshot too old
181275	Reporting: error ORA 1555 and COMMIT WORK
165247	Notes on creating rollback segments
125116	REGUP, REGUH and BKPF in 4.0B upgrade with IS F3.04
123793	SIS variant configuration:Termination S126 -> S127
118919	Problems RMCVISCP (SQL error 1555)
102034	R3trans: Control of the COMMIT intervals
96296	CC-ERROR: Database problem in client copy
88656	R3trans: Duplicate key composite SAP note (Release 4.0-4.6)
82634	PSAPROLL-Which rollback segments created, and how?
68896	R3trans: Performance problems and cluster
67205	CC-INFO: Copying large and productive clients

Number	Short Text
63639	Error in activating an operating concern
49111	FI-SL: conversion of the line items is too slow
35415	CC-TOPIC: Memory space analysis / Test run
16277	ORA1562 / ORA1628
13655	SAPDBA: Long runtimes: Exp./imp./reorganization
3807	Error messages regarding rollback and undo segments

Attributes

Attribute	Value
Database system	ORACLE