Avoiding ORA-01555 Snapshot too old: Rollback segment too small errors.

Discover the reasons for the "Snapshot too old" error, and a way of avoiding in future.

Andrew Deighton, Phaldor IT Solutions Limited

The "Snapshot too old" error can be produced whenever a long running query is set off against a database, no matter what the client software is, or the release of the database. The article is directed towards DBAs or operators who are responsible for ensuring that the database provides answers rather than error messages.

Introduction

Running the year-end financial report, using export to back up critical data before an upgrade, or running a large data integrity report each have certain common elements: they all take a long time; they all read the database without necessarily doing an update; and they are all susceptible to the "ORA-01555 Snapshot too old, rollback segment too small" error.

The ORA-01555 error can occur whenever a read-only transaction is run against a database, and the longer the transaction runs for, the higher the probability of encountering the error. This article describes the uses of Rollback Segments within the Oracle database, and how Oracle maintains and allocates them. Once their use is understood, resolving the "Rollback segment too small" problem can be tackled. A method for preventing the error is given, together with template code to help you implement a solution within your own company.

Background to Rollback Segment Use

Rollback segments are used in a number of ways by the Oracle database server. The idea behind rollback segments is fairly simple - whenever a change is written to the database, the data as it was before the change is written to a rollback segment. The *before images* that are stored in the rollback segments allow the server to provide functionality such as transaction rollback, crash recovery and, of interest to this article, *read consistency*.

In a multi -user environment, while one user is reading data, others may be updating the same data. In order to avoid confusion, Oracle guarantees that the data you retrieve for a query is consistent as it was at the time you issue the query, no matter how much it has changed while the query is running. The rollback segments are used to provide this functionality.

If your query needs to read some data that has been changed by another process, it can retrieve the original values by looking in the rollback segments. However, as rollback segments (like any other segment in the database) are of finite size, a time will come when they become full. Oracle then makes a decision whether to add more space to the rollback segment or to reuse space that is already allocated and overwrite data that is in the extent. When Oracle to reuses previously used storage, and your query needs the data that has been overwritten in order to provide a consistent view of the data, an ORA-01555 error is returned, as the required data is no longer available.

The key to preventing the error message therefore lies in understanding why Oracle reuses rollback segment space instead of allocating more, and then preventing this reuse. It also relies on sufficient space being available for the rollback segments to grow into.

Whenever a (read/write) transaction starts, Oracle assigns it to a rollback segment. The transaction stays assigned to that rollback segment until it terminates (commits or rolls back) at which time it releases its hold on the rollback segment. The data, although no longer held by an active transaction, is not deleted, but left in place to provide read consistency for other transactions.

Rollback segments are made up of multiple extents (in the same way as other segments are). Transactions log their changes into the active extent of the rollback segment until it is full. Oracle then examines the next extent in the segment (in a circular fashion, looping back to the beginning if the current extent is the last one in the segment). If this (next) extent contains no active (update) transactions, it is reused, and becomes the current extent. If, however, it does contain an active transaction, Oracle will attempt to add a new extent.

It should be clear, then, that in order to prevent Oracle from reusing the rollback segments' extents, what is needed is an active (update) transaction assigned to each rollback segment. In order to allow for a long query to run without the ORA-01555 error, it is necessary to first ensure that there is an active transaction in every online rollback segment in the database before the process starts, and that these transactions remain active for the duration of the process.

Fortunately, from release 7 onwards, it is possible to specifically assign a transaction to a rollback segment. This is accomplished by using the statement:
`SET TRANSACTION USE ROLLBACK SEGMENT <<NAME>>;`
as the first statement in the transaction.

Avoiding ORA-01555

Before you begin, it is necessary to get a list of the rollback segments in your database that need to be protected from reuse. This is generated by the command:

```
SQL> SELECT SEGMENT_NAME
2 FROM DBA_ROLLBACK_SEGS
3 WHERE STATUS = 'ONLINE'
4 AND SEGMENT_NAME != 'SYSTEM';
```

Two points of interest are raised in this statement. First, there are no USER_ROLLBACK_SEGS or ALL_ROLLBACK_SEGS views, so this can only be run by a user with appropriate privileges, and second, there is a SYSTEM rollback segment, which like the SYSTEM tablespace is not (or should not be) used OTHER THAN by the RDBMS itself.

Once all the necessary rollback segments have been identified, each can be held by first issuing the appropriate SET TRANSACTION command, and then performing some update. One way of achieving this is to have a table that logs the activities that have been carried out:

```
create table rbs_hold_log
(Message varchar2(200));
```

Each rollback segment can then be held from wrapping around by using the following script:

- Insert a message to show which segments are being held.

```
insert into rbs_hold_log (message)
values ('&1 locked at '||sysdate)
/
```

- Commit the data entry to:
- 1. Make the change known to any other session (e.g. For monitoring)
- 2. End the transaction and allow a new one to start.

```
commit
/
```

- Attach the new transaction to the required rollback segment

```
set transaction use rollback segment &1
/
```

- Remove the entry that was added earlier.

```
delete from rbs_hold_log
where message like '&1 locked%'
/
```

- Now wait until the hold can be released WITHOUT committing or rolling back.

Once the rollback segment has been attached, it is necessary for the session to suspend itself for the duration of the job that requires read consistency. This can be accomplished in a number of ways. One method is to use a "flag file". Here, a unique file name is chosen and the file is created on the computer. (This file would typically be used throughout the system to indicate that the process is running and would contain the process ID of the program.) As long as this file exists, the SQL sessions that are holding the rollback segments will sleep (using dbms_lock.sleep). When the process completes, it removes the flag file, and the SQL sessions can terminate (either committing or rolling back their transactions). This has been implemented through a PL/SQL function:

```
   CREATE OR REPLACE FUNCTION FLAG_OK (p_dir varchar, p_file varchar)
RETURN boolean
   IS
   l_FileHandle utl_file.File_Type;
   begin
   l_FileHandle:=utl_file.FOPEN(p_dir,p_file,'r');
   utl_file.FCLOSE(l_FileHandle);
   return true;
   EXCEPTION
   WHEN OTHERS THEN
   return false;
   END FLAG_OK;
```

The sessions that have locked the rollback segments can use this function through an anonymous PL/SQL block, as in:

```
   begin
   while flag_ok('/tmp','FlagFile')
   loop
   dbms_lock.sleep(60);
   end loop;
   end;
```

This entire process can be scripted and automated to provide an easy way of locking the rollback segments.

For a complete scripted solution using a UNIX shell script, look at www.phaldor-it.com/downloads/long_job.html.

Conclusion

Preventing the ORA-01555 error is not difficult once the workings of the Rollback segments within Oracle are understood. A simple script to automate the process can be produced and run whenever such a job is required. This will reduce the possibility of errors being generated.

Andrew Deighton is a database consultant at Phaldor IT Solutions Limited. He has been working with Oracle for 8 years. Andrew can be contacted at Andrew.Deighton@Phaldor-IT.Com.