Why do I have 30 million rows in the FND_LOGINS table?
OR
How much damage a can a simple check box cause?

A CASE REVIEW
By
G.Piper
www.PIPER-Rx.com

1 Case Presentation

The site in question presents with over 30 million rows in the ***fnd_logins*** table, growing at a rate of approximately 25,000 records per day.

Based on the site's application activity, the estimated number of rows that should be held in the ***fnd_logins*** table should be around 800,000 records (i.e. holding 32 days history on-line). It is expected that this number should reduce post concurrent manager activity review.

The site was running the concurrent program **Purge Signon Audit data** (FNDSCPRG) daily as part of their normal maintenance program to purge the sign-on audit data and was unaware of the high growth rate in the sign-on audit tables.

The site had not noticed this issue. I have to say this scenario does exist out there as this is not the first time I have encountered this issue.

2 Diagnosis

On review of the site's scheduled requests it was found that the **Purge Signon Audit data** was being run daily with the date argument set to 10-Oct-06, however, "**the increment date parameter each run"** check box had not been checked when the scheduled request was created. As such the data parameter has not been incrementing with each run, thus with each run the program has been purging all sign-on audit records prior to 10-Oct-06. As a result any record added after that date has not been being purged.

So, since 10-Sep-06 (1,221 days at the time of presentation) the program has been running but purging nothing.

At the time the case was presented:

- The ***fnd_logins*** table was holding over 30 million rows and growing at a rate of approximately 25,000 records per day
- The ***fnd_login_responsibilities*** table was holding 800,000 records and growing at a rate of approximately 650 per day
- The ***fnd_login_forms*** table was holding over 4 million records and growing at a rate of approximately 3,300 records per day

Why so many records in ***fnd_logins*** in relation to ***fnd_login_responsibilities*** and ***fnd_login_forms***?

The site is currently generating an average of 25,000 requests per day. Every concurrent request adds an entry on ***fnd_logins*** but not in the responsibilities and forms audit tables.

Warning: The simple immediate solution would appear to be to run a **Purge Signon Audit data** request with an appropriate date and setting the increment date feature on. If this were to be implemented the first run would attempt to delete approximately 29 million rows from the table and may cause issue with rollback segments impacting other application activity.

TIP: Do not just turn the application sign-on audit future off to stop the growth. There is a lot of valuable application usage profiling information collected by the application; who did what, where and when. You must fully understand the impact to application usage audit before turning this feature off.

2.1 Symptoms

This issue is not a business threatening issue requiring immediate attention.

The site has been living with the issue for 3 years 11 months and has not noticed; it could continue to live with the issue for some time without any additional adverse effects. The sign-on audit tables are written to but vary rarely read from, all they will do is continue to consume database space for both tables and their associated indexes.

The concurrent program **Purge Signon Audit data** (FNDSCPRG)[1] is designed to keep the sign-on audit tables in a manageable state by deleting data from the following application tables[2] collected by the applications sign-on audit feature:

- fnd_logins
- fnd_login_resonsibilities

- fnd_login_resp_forms
- fnd_unsuccessful_logins
- fnd_appl_sessions

Each of these table will be growing and continue to grow until the issue is resolved.

In addition, the sixteen[3] (16) indexes associated with these tables will continue to grow. A list of indexes can be found in **Appendix 1 – Tables and Indexes**

It was estimated that the application was holding 4-5 Gb of data and index space associated with this issue.

The biggest performance impact to the application would be when the concurrent program runs. The concurrent program **Purge Signon Audit data** (FNDSCPRG) uses an unindexed column ***start_time*** in its execution - (and ***attempt_time*** in the case of ***fnd_unsuccessful_logins***)

Where start_time < fnd_date.canonical_to_date('&1');

As such the purge program will execute a full scan on each of the target tables in order to determine the rows to delete. So for this site, that would include a daily full scan of a 30 million and a 4 million row table[4] to **not** find any rows to delete.

3 Pre Treatment

Task 1: Cancel the current scheduled purge job

3.1 Application Impact:

At this site cancelling the daily scheduled **Purge Signon Audit data** request will have no effect on the application as the request has been doing nothing anyway. What it will do is stop the full daily full scan of the 30 million and 4 million row tables, removing the associated performance overhead.

4 The Operation

Task 1: Controlled removal of sign-on audit history

At this site removing all but 32 days[5] of sign-on audit history in one pass would delete an estimated 29 million rows from the ***fnd_logins*** table and approximately 3.5 million rows from the ***fnd_login_forms*** table. This may cause issues with the application's rollback segments and potentially impact other application activity.

What we want to do instead is perform controlled deletions, more akin to what should have been happening as part of the normal purge process.

The aim here is to run several non-scheduled **Purge Signon Audit data** requests, with each request deleting one month's data at a time. In this way we will remove approximately 800,000 to one million rows per request.

This will require a large number of runs of the **Purge Signon Audit data** requests, incrementing the date parameter with each run.

Given that the date of the current run is 10-Sep-06:

- The first delete run should be 01-Oct-06
- The second delete run should be 01-Nov-06
- The third delete run should be 01-Dec-06
- The fourth delete run should be 01-Jan-07
- etc.....

...until you get to 32 days prior to the current date.

Note: For those technically minded, you could run the source SQL setting the new date each run, however this is **not** recommended procedure; you should always run the program that should have been run in the first place.

4.1 Application Impact:

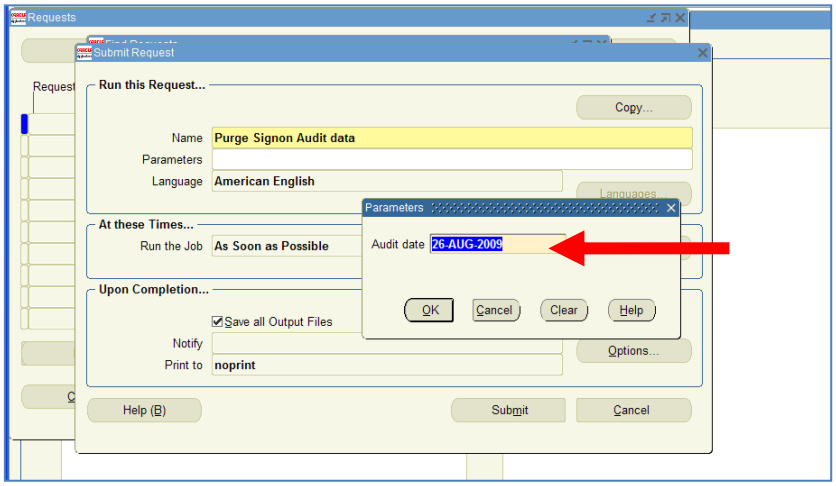
Providing the **Purge Signon Audit data** requests are not run during business hours the impact will be little to none.

5 Post-Operative treatment

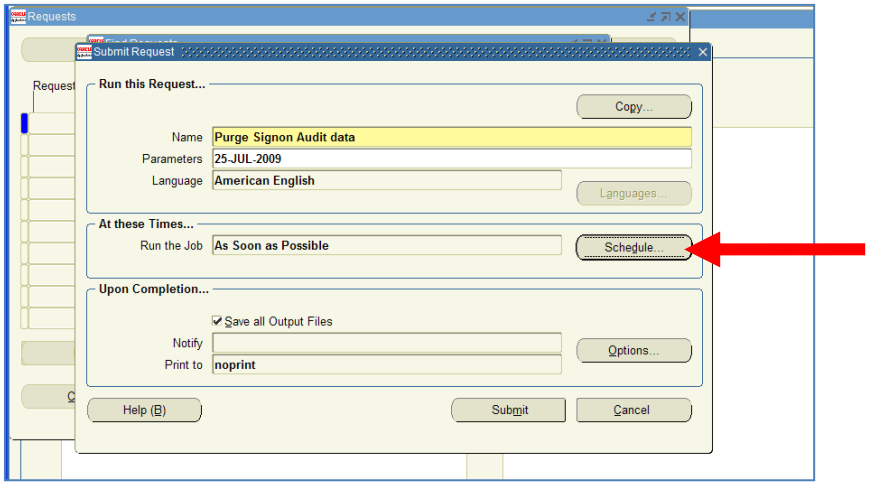
5.1 Task 1 – Create Scheduled Purge Request

Task 1: Create a scheduled **Purge Signon Audit data** request with the increment date future enabled

Once the ***Purge Signon Audit data*** program has been selected, set the "Audit Date" to the current date minus 32 days:



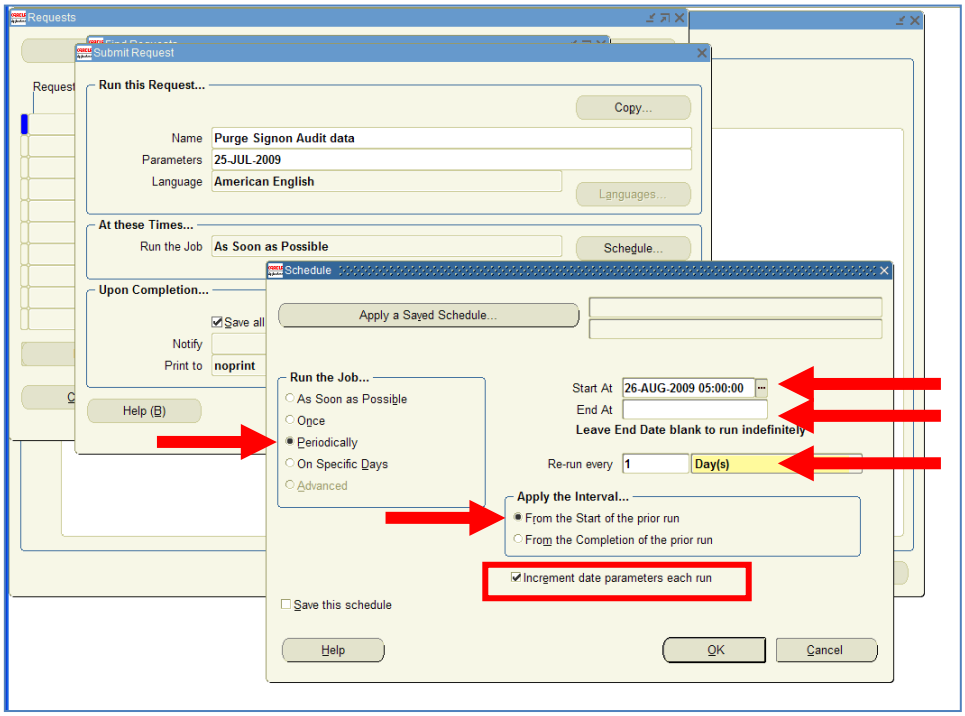
Next, setup the request as a scheduled reoccurring request by selecting "Schedule":



- On the Schedule screen select the ***"Periodically"*** option
- Set the ***"Start At"*** date to a date and time greater than the current time. Also set the hour of day to a suitable non business activity time e.g. 4:00am. This will ensure the purge runs when there is little to no user activity.
- Do not set a value for ***"End At",*** this will ensure the program runs indefinitely , or until you manually cancel the program

- Set the value for ***"Re-run every"*** value to 1 day(s), this will set the program to run daily at the time set in the ***"Start At"*** date
- Select the value for ***"Apply the "Interval..."*** to From the Start of the prior run. This will prevent he program from creeping forward in time by the run time of the program each time it runs
- The final step is to check the box ***"Increment date parameter each run".*** This will increment the parameter date each time the request is run ensuring you only keep 32 days history on-line
- Select OK and submit the request

Note: Not checking the ***"Increment date parameter each run"*** box was the original cause of the issue.



Impact – None Providing this is done once the controlled deletion has been completed there should not be any impact as this is what should have been happening anyway. Until the underlying table high water marks have been reset and the indexes rebuilt there may still be performance issues, but most likely less than has been occurring.

5.2 Task 2 – Rest High Watermarks:

At this site the base table ***fnd_logins*** has grown way beyond its normal level of 1 million rows (based on 32 days history at current activity levels) and will have a very high "high water mark" which will need to be addressed. This is a normal DBA activity and as such is not detailed in this case study. Make sure your DBA addresses this.

Given the high number of rows being deleted it is recommended that all affected tables be reviewed for high watermarks and be suitable adjusted. This may require the rebuild of one or more of the affected tables which will be required to be undertaken during a maintenance window when there is no application activity.

A potential faster, low impact method would be to truncate the affected tables but only when you have ensured there are no active user sessions and the concurrent managers are shutdown. With this method it would be as if sign-on audit had just been turned on – This again is **not a recommended process** but has been mentioned so as to aid your thought processes for alternate methods for resetting high watermarks and rebuilding indexes.

Impact – Very High Rebuilding tables is a high risk activity. You must make sure none of the synonyms are missing and that all invalid objects generated due to the table rebuild are recompiled.

5.3 Task 3 – Rebuild Indexes:

Given the high number of rows being deleted the indexes will be very "brown", "honeycombed" or imbalanced, as they are full of logically deleted rows. It is recommended that all indexes (sixteen)[3] on the affected tables be rebuilt.

Impact –High Rebuilding indexes requires that the sign-on audit tables are not being used. This will require all user session closed and concurrent managers to be shut down.

6 Ongoing Care

Given the sign-on audit tables are now being regularly purged, the indexes on these tables should be rebuilt on a regular basis; about every 6 months.

The **PIPER-Rx *PAM*** monitor (http://www.piper-rx.com/pages/pam_content.html) includes an index maintenance reminder letting you know when selected indexes have not been rebuilt in 6 months (default).

About once every 6 months you should check the actual start time for the ***Purge Signon Audit data*** scheduled program as the time will tend to creep forward. Where this is the case you may need to cancel and re-create the scheduled program if the number of days history on-line changes due to short and long months.

Footnotes:

1 The purge Signon Audit data source code is version dependant - please check your site's purge code $FND_TOP/sql/ FNDSCPRG.sql

2 The tables being purged and their associated indexes are version dependant, please check your sites purge code $FND_TOP/sql/ FNDSCPRG.sql

3 The number of indexes is dependent on the OEBS version

4 Row counts were not obtained for the remaining sign-on audit tables being purged by the purge program

5 I always advocate that at least one full business cycle's history be held on-line for activity profiling and capacity planning purposes

7 APPENDIX 1 – Tables and Indexes

The following table shows the sixteen (16)[3] indexes associated with the five (5)[2] sign-on audit tables being purged **Purge Signon Audit data** (FNDSCPRG)[1]:

Table Name	Index Name	Position	Column Name
FND_LOGINS	FND_LOGINS_N1	1	SPID
	FND_LOGINS_N2	1	USER_ID
	FND_LOGINS_U1	1	LOGIN_ID
FND_LOGIN_RESPONSIBILITIES	FND_LOGIN_RESPONSIBILITIES_N1	1	AUDSID
	FND_LOGIN_RESPONSIBILITIES_U1	1	LOGIN_ID
	FND_LOGIN_RESPONSIBILITIES_U1	2	LOGIN_RESP_ID
FND_LOGIN_RESP_FORMS	FND_LOGIN_RESP_FORMS_N1	1	LOGIN_ID
	FND_LOGIN_RESP_FORMS_N1	2	LOGIN_RESP_ID
	FND_LOGIN_RESP_FORMS_N2	1	AUDSID
	FND_LOGIN_RESP_FORMS_N3	1	FORM_APPL_ID
	FND_LOGIN_RESP_FORMS_N3	2	FORM_ID
FND_ UNSUCCESSFUL_LOGINS	FND_ UNSUCCESSFUL_LOGINS_N1	1	USER_ID
	FND_ UNSUCCESSFUL_LOGINS_N1	2	ATTEMPT_TIME
	FND_ UNSUCCESSFUL_LOGINS_N2	1	LOGIN_NAME
FND_APPL_SESSIONS	FND_APPL_SESSIONS_N1	1	AUDSID
	FND_APPL_SESSIONS_N2	1	LOGIN_ID

8 Want to know more?

There is loads more ***FREE*** information on this topic and all aspects of OEBS Application Administration at the **PIPER-Rx** website. After over 20+ years working with Oracle (the product, not the Company) and Oracle E-Business Suite (since Release 5) I have visited countless sites and pretty much seen it all when it comes to Applications Administration. Since the late 1990's I have spent more time sharing these learnings and the most popular papers and case studies I have presented are available at the **PIPER-Rx**.com website as well as a whole host of Tips and Reports I have used throughout my career.

All information at the **PIPER-Rx**.com website is ***FREE*** so why not check it out….I hope you find it useful! **– 30,000+ downloaders to date can't be wrong!**

9 Disclaimer

The material contained in this document is provided by the author "as is" and any express or implied warranties, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of any content or information, even if advised of the possibility of such damage. It is always recommended that you seek independent, professional advice before implementing any ideas or changes to ensure that they are appropriate.