Indexing Alternatives

Jerry Specht, May 6, 2012   Updated, April 2, 2017

1. **Regular Indexing in Actual Library.** Rather than defining a parallel library you just do the indexing in the actual library.

Pros:

a) Doesn't require defining a parallel library.
b) Requires slightly less Oracle table space (under parallel indexing you need to have space – at least temporarily -- for both the actual tables and the parallel tables).

Cons:

a) OPAC needs to either be down or pointed to the Test server – but if you have a method for refreshing the Test server, you may as well use the Test server for indexing, as described in alternative #2, below.
b) Though Circulation can continue while the index job is running, continued use of GUI Cataloging and Acquisitions requires unlocking the bib library being indexed. {This is described in section 7 of the "How To Run Index Jobs" (the current version of which is on the Tech Seminar CD). You toggle ue_01 "DOWN" (util w/5/10) before unlocking the library. Then, once the indexing job is done, you would toggle it back to "UP". Note that this stops the ue_01 process in *all* libraries and it would need to be restarted for all libraries once the job is done.} Even though Word and Browse indexing is stopped, updating of z11 (Direct), z13 (Short), and z00r indexes – which

are updated both immediately *and* by ue_01 – continues. {This is described in KB 8192-2329 ("Indexes updated immediately; indexes updated by ue_01"). } Thus, unlocking-the-library cannot work for p_manage_05 (Direct/z11) or p_manage_07 (Short/z13/z00r).

c) Like parallel indexing, it can affect the performance of the Production server, slowing down other processes running on the server.

Bottom Line: This method is generally only desirable/feasible when the index job can be performed during a window where the Aleph OPAC and GUI Search can be down. *The time required should be determined by running the job on the Test server first.*

2. Parallel Indexing (on Production server)

The **Pros** of parallel indexing vs. regular indexing in actual library are shown immediately above, in "Cons": OPAC can be up and Cataloging and Acq can be up.
But there really are no **Pros** vs. Indexing on Test Server (immediately below): any indexing (parallel or regular) will affect the performance of other functions on the Production server.
At the time parallel indexing was devised, transferring a z98 table of any size from one server to another was not practical, but the introduction of the Oracle Data Pump changed that completely.

Bottom Line: If you already have parallel indexing set up, continuing to use it makes sense, but, otherwise we suggest one of the other methods: if running the job on the Test server shows that it can fit in your window for system downtime, you might run it on the Prod server, otherwise, we suggest indexing on the Test server. (See below.) Indexing on the Test server and transferring the index tables generated to Production is definitely the *safest* option.

3. Indexing on Test Server

a. Do util e/5/1 to save z07 indexing request records as z07h's and create trigger on the Production server – as described in section 6 of the **Parallel Indexing.pdf** .
b. Refresh Test server. Export all of your Oracle data and the u-tree (Aleph tables) from Production to Test (being sure to preserve the Test aleph_start and license files). This could be done either via cloning or with the Data Pump. **Refreshing Test server data Using Oracle Data Pump**
c. Run index jobs on Test server as described in the "How To Run Index Jobs" document: **How To Run Index Jobs.doc** in **Additional How To Presentations from Support**
d. Test the new indexes on the Test server.
e. Rename the files created on Production in step b so that they are not overlaid in the following step f.
f. Export the rebuilt index tables from Test using the Oracle Data Pump command:

```
csh -f $aleph_proc/oracle_expdp_table
LAW04,z98 > & oracle_expdp_table.js.log &
```

This is the same as the Parallel Indexing, Step 9, Option 2 ("Oracle Import") described in section 10b (slide 17) of the TS2012_Aleph_Parallel_indexing.pptx Powerpoint, which we looked at earlier.

g. ftp the exported tables from Test to Production
h. Import the rebuilt index tables on Production using the Oracle Data Pump:

```
csh -f $aleph_proc/oracle_impdp_table
LAW04,z98 > & oracle_impdp_table.js.log &
```

i. Update the relevant util g/2 counters (last-word-number, last-acc-number, last-long-acc-number) in the Production xxx01. If you changed the xxx01 $data_tab files for indexing, copy those also from Test to Production.l
j. Test the new indexes on Production. If not working, load the versions saved in step e above.
k. Do util e/5/2 ("Restore - Dump Z07H to Z07 and Delete Oracle Trigger"), as described in section 12, slide 21, of the Powerpoint.

Pros:

- Simpler: doesn't require definition of a parallel library.
- Doesn't slow down Production server.
- Doesn't require additional Oracle space or workspace on Production server.
- Complete testing can be performed on the Test server to make sure the files are OK before moving/loading them to Prod.

Cons:

- Requires extra steps of exporting and importing the data (but you want to refresh the data on the Test server periodically anyway….)
- Test server is unavailable for other uses (except Circ) while indexing is occurring.

2.1 MNPALS Variation of Indexing on Test Server

The MNPALS variation on the preceding is described in the parallel_indexing_with_tt.pdf and related documents, available from Al Rykhus (alan.rykhus@MNSU.EDU).* It differs from the preceding in steps b, f, g, and h: the Data Pump is *not* used. The Production tablespaces are reconfigured so that the index tables are in separate "transportable" tablespaces, such as, the Z01Z02 tablespace, the Z95Z97Z98 tablespace, etc. The data is refreshed on the Test server by copying all the tablespaces from Production to Test (as in step 2.b above). After the indexing is done, these transportable tablespaces are copied from Test to Production .

Pros: Import on Prod is faster (less downtime) with transportable tablespaces than with the Data Pump

Cons: Requires reconfiguring of tablespaces and – though Al has detailed instructions – some familiarity with cloning is probably desirable.

*Al has moved to ODIN (alan.rykhus@ndus.edu) and is no longer using this method there.