Phone: (800) 621-2808
Outside the U.S.: (713) 621-2808
Fax: (713) 621-1639
On the Web: www.bradmark.com

Top Reasons for Performing Online Reorganizations with DBControl *Online*

1. Moving tables between tablespaces
Move hot (more active) tables to faster disks. Move tables around to spread the I/O load over disk drives. Move tables around when administrative guidelines change. Group tables according to extent sizes to avoid fragmentation (and thus wasting space.)

2. Eliminating fragmentation
Fragmentation occurs when pockets of space are too small to hold any new extents. The space in these pockets becomes unusable. Immediately reclaim space by performing an online reorg. Avoid future fragmentation by setting initial extent = next extent, setting
pctincrease = 0, and grouping tables by extent size in different tablespaces (i.e. TS1 contains tables with extent size 16K, TS2 has extent size 1MB, TS3 has extent size 100MB.)

3. Moving tables from dictionary-managed to locally-managed tablespaces
Get faster allocation and de-allocation of extents with locally-managed tablespaces. Realize benefits of uniform- and automatic-allocation LMTs. Uniform allocation helps avoid fragmentation (and thus wasting space), for example.

4. Partitioning or repartitioning tables
Partition large tables to ease management tasks. By performing a task on one (or a few) partitions at a time, large management tasks can be done in increments, enabling the tasks to be done in ever-shrinking downtime windows. Add hash partitioning to a table to spread I/O over several disks, to increase performance. Repartition a range-partitioned table when new subsets of data are needed or when existing subsets must be changed.

5. Eliminating row migrations
If PCTFREE is too low, updated rows may migrate from block to block, causing DML to traverse many blocks to get to the data. Too much migration can significantly reduce performance. Increasing PCTFREE and performing an online reorg can eliminate the existing migrations and increase performance.

6. Optimizing FREELISTS and FREELIST GROUPS for tables and indexes
Increasing freelists in a heavy update environment, such as an audit table, can greatly increase performance by reducing contention on the freelist by multiple sessions. In an OPS or RAC environment, increasing freelist groups can decrease contention on freelists between instances, thereby increasing scalability.

7. Optimizing PCTFREE for indexes
PCTFREE controls how much of blocks are reserved for new keys. If PCTFREE is too low, excessive block splitting can occur on update operations, slowing down the applications updating the table.

8. Optimizing INITRANS and MAXTRANS for tables and indexes
INITRANS and MAXTRANS control the minimum and maximum numbers of entries in a block's transaction table. Every transaction must get a slot in order to use a block. The transaction table can expand. If PCTFREE and INITRANS are both too low to handle the number of concurrent transactions on a block, its transaction table may not be able to expand. One session may have to wait on another. Setting INITRANS appropriately can increase scalability.

9. Optimizing PCTUSED and PCTFREE for tables
PCTUSED controls when blocks are added back to the freelist. PCTFREE controls how much space in each block is reserved for updates. Different scenarios require different settings to optimize allocation of blocks to and from the freelists.

10. Reducing the number of extents
In Oracle 8 and later, managing hundreds of extents will probably not affect performance. A table or index with several thousand extents, however, can severely impact performance; especially in dictionary-managed tablespaces (DMTs.) Allocation and deallocation of extents can suffer. In addition, the table can become unmanageable. Dropping or truncating such a table, in a DMT for example, can cause SMON to consume excessive CPU for long durations of time. The same thing can happen when coalescing free space in the DMT.

11. Changing (heap) tables to IOTs
Look-up tables can convert to IOTs to greatly increase performance, since row data can be stored in an index segment.

12. Compacting table and index segments
Maintenance tasks, such as archiving historical data, can leave large portions of the allocated table segment unused. For example, the high water mark of a table may lie significantly beyond the last piece of data for the table. Full table scans performed by report generators or exports can greatly benefit from a more compact table segment created by an online reorg. The high water mark in the more compact table segment is set as tight as possible against the last piece of data.