Craig S. Mullins

Return to Home Page

October 1998

Oracle update

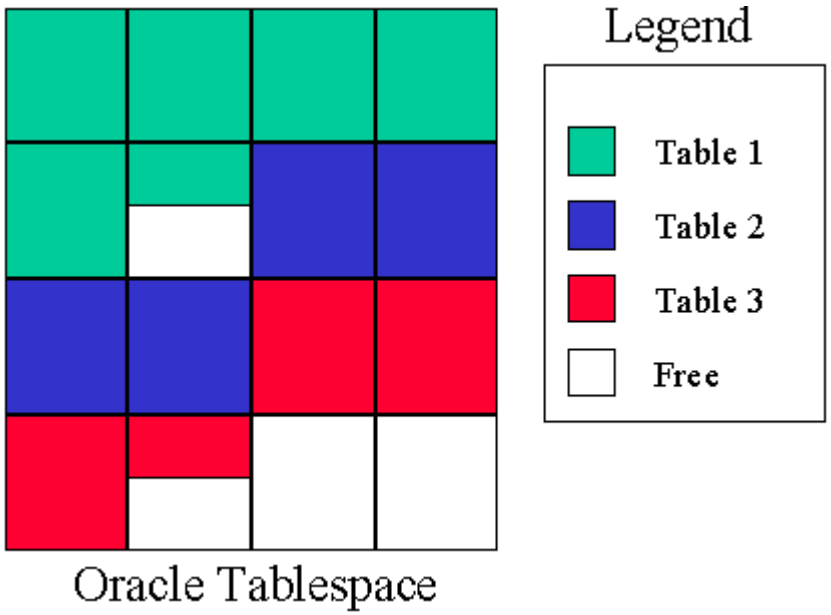
Dealing with Fragmentation and Disorganization

By Craig S. Mullins

Relational technology and SQL make data modification easy. Just issue an insert, update, or delete statement with the appropriate WHERE clause, and the RDBMS takes care of the actual data navigation and modification. In order to provide this level of abstraction, the RDBMS handles the physical placement and movement of data on disk. Theoretically, this makes everyone happy. The programmer's interface is simplified and the RDBMS takes care of the hard part — manipulating the actual placement of data. However, things are not quite that simple. The manner in which the RDBMS physically manages data can cause subsequent performance problems to arise.

Most of us have experienced the situation where an application slows down after it has been in production for awhile. But do you know why? Perhaps the number of transactions issued has increased and maybe the volume of data has expanded, but it doesn't seem that these factors should cause such a large performance degradation. The problems might be due to database disorganization. Database disorganization occurs when a database's logical and physical storage allocations contain many scattered areas of storage that are too small, not physically contiguous, or too disorganized to be used productively.

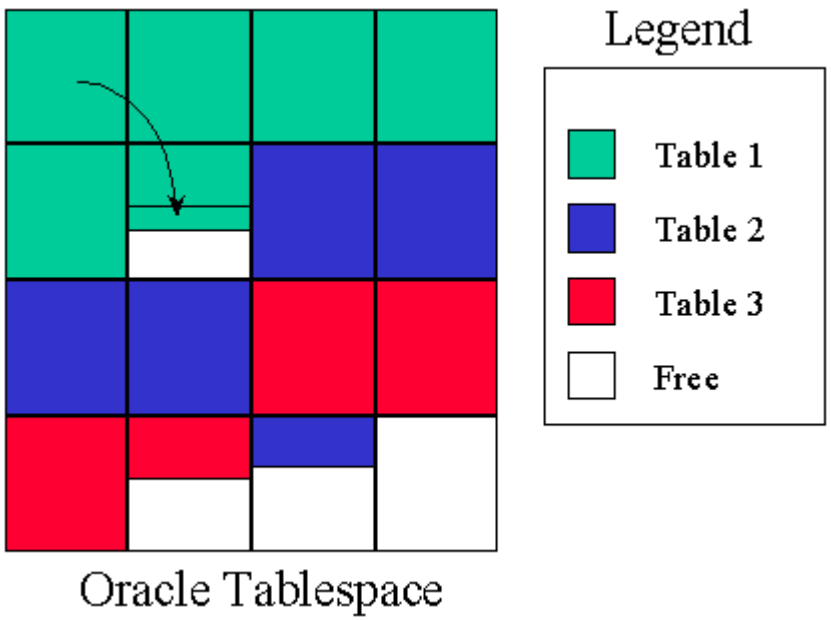
To understand how performance can be impacted by database disorganization, let's examine an Oracle table space as modifications are made to data. Assume that an Oracle table space exists that consists of three tables across multiple blocks, such as the table space and tables depicted in Figure 1.



Fragmentation and Row Chaining
The building block for Oracle storage is the data page.

Let's make two data changes to tables in the table space. First, let's add six rows to the second table. But no free space exists into which these new rows can be stored. How can the rows be added? Oracle requires that another extent is taken into which the new rows can be placed. For the second change, let's update a row in the first table to change a VARCHAR2 column; for example, let's change the LASTNAME column from "ROGERS" to "FILIPOWSKI". This update results in an expanded row size because the value for LASTNAME is longer in the new row: "FILIPOWSKI" contains 10 characters whereas "ROGERS" only consists of 6.

The resultant table space might now look like this:



Two potential problems are depicted in Figure 2: fragmentation and row chaining.

Fragmentation is a condition in which there are many scattered areas of storage in a database that are too small to be used productively. It results in wasted space, which can hinder performance and even cause database failure.

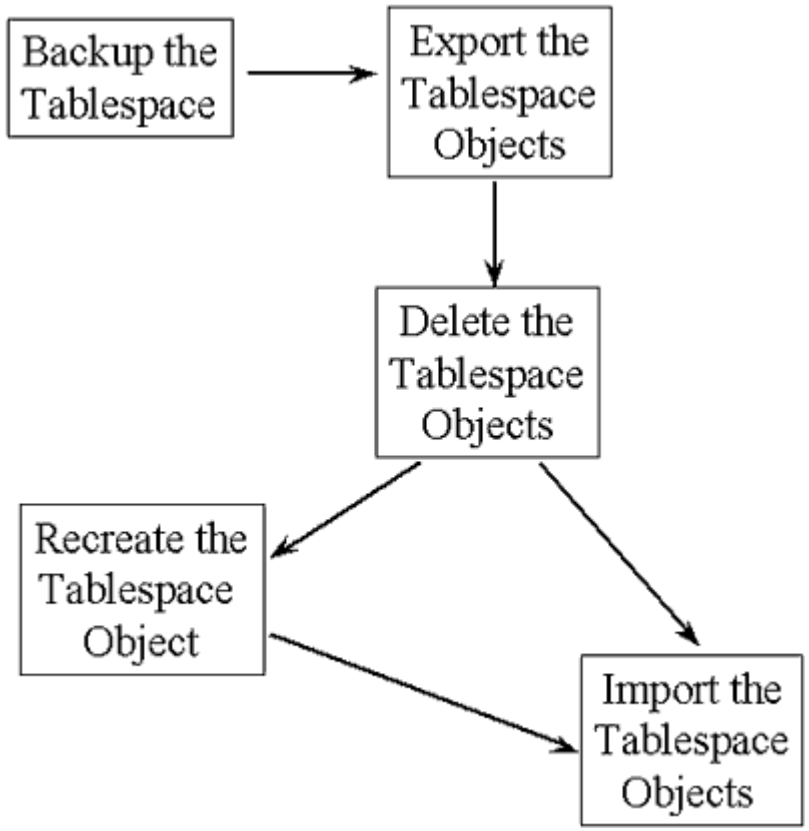
When updated data does not fit in the space it currently occupies Oracle will find space for the row using either row chaining or row migration. With row chaining Oracle will move a part of the new, larger row

to a location within the table space where free space exists. With row migrations the full row is placed elsewhere in the segment. In each case a block-resident pointer is used to locate either the rest of the row or the full row. Both row chaining and row migration will result in multiple I/Os being issued to read a single row. This will cause performance to suffer because, obviously multiple I/Os are more expensive than a single I/O.

Reorganizing Tablespaces

To minimize fragmentation and row chaining, database objects must be restructured on a regular basis. This process is also known as reorganization. The primary benefit is the resulting speed and efficiency of database functions because the data is organized in a more optimal fashion on disk. In short, table space reorganization maximizes availability and reliability for Oracle databases

Traditionally, DBAs have done this manually by completely rebuilding databases. But a reorganization requires a complex series of steps to accomplish. The diagram in Figure 3 depicts the reorganization process.



In order to accomplish this reorganization, the database must be down. The high cost of downtime creates pressures both to perform and to delay preventive maintenance — a double-bind familiar to all DBAs. Third party tools are available that automate the manual process of defragmentation and reorganization of tables, indexes, and entire tablespaces — eliminating the need for time- and resource-consuming complete database rebuilds. In addition to automation, these type of tools typically speed up the reorg process and analyze whether a reorganization is needed at all.

Synopsis

Reorganizations can be costly in terms of downtime and computing resources. And it can be difficult to determine when a reorganization will actually create performance gains. However, the performance gains that can be accrued are tremendous when fragmentation and disorganization exist. The wise DBA will plan for Oracle table space reorganization if the above types of disorganization are likely to occur in their systems.

From Oracle Update (Xephon), October 1998.

Home. Phone: 281-494-6153 Fax: 281-491-0637