SELECT® Online
The Journal of the International Oracle Users Group

Select Magazine - April 2001
Volume 7, No. 3

The Effectiveness of Global Temporary Tables

By Craig Shallahamer

> *If your application is generating a tremendous amount of I/O and your I/O subsystem just can't handle the requirements, a feature released in Oracle8 called Global Temporary Tables, or GTT for short, may provide the answer you've been looking for.*

Most applications use what are called interim tables. Interim tables, sometimes incorrectly called temporary tables, live in the middle between Oracle temporary segments and data segments (i.e., tables). Interim tables typically hold data only during a batch process and typically do not have to be recovered. Doesn't it seem silly then to create, buffer and write all that redo? Global Temporary Tables (GTT) answer this question by having specific characteristics of both temporary segments and data segments. For example, while a GTT contains rows and can have associated indexes, like temporary segments, the amount of redo generated is virtually zero.

When I first heard about GTTs, I was so intrigued by Oracle's claim that no redo was generated, I decided to perform some experiments. This paper documents the results of my experiment. The condensed research paper and the actual experimental results (in spreadsheet format) are available for free on OraPub's web site.

What are Global Temporary Tables?
Oracle 8.1 introduced a new feature called Global Temporary Tables or GTT for short. If you are familiar with the concept of an interim table, then think of a GTT as a true interim table. To summarize, GTTs are characterized by or can do the following:

- Have many temporary segment characteristics;
- Have many data segment characteristics; and
- Radically reduce redo creation - transactions can be rolled-back but not recovered, and they can be indexed.

Each will be discussed in more detail.
Data Segment Characteristics
GTTs have many data-segment characteristics. For example, a GTT is created by a session using a create table variant. Once created, the session can insert, update and delete rows just like a standard table. While rollback is possible, because redo is not generated, recovery is not possible. Just as with data tables, one of the most powerful GTT features is that indexes can be created on the GTT. When the object is dropped (implicitly or explicitly), so are the indexes.

Temporary Segment Characteristics
GTTs also have many temporary segment characteristics. For example, the GTT lives only as long as the session, with an option to live only until a commit is issued. So just as with a temporary object, when the session disconnects, the object is removed. A GTT lives in the user's temporary tablespace and, while you give it a name (for example GL_POSTING) internally, it is named just like a temporary segment (i.e., file#. block#). Another important characteristic is that a GTT is session-specific. This means, only the owner of the GTT has access to it. This also means that many sessions can have an active GTT named, GL_POSTING.

So in many respects, you get the best of both worlds: Table-like control and indexing along with automatic temporary segment management.

Redo Creation
GTTs radically reduce redo creation. Many applications create an enormous amount of redo that the DBA can do absolutely nothing about. The DBA basically has been handed the application I/O requirements and told to somehow make sure that performance meets expectations. But with GTTs, the DBA has a tool that will work at the Oracle server level avoiding the desire (if you can call it that) to re-design the application. For most DBAs, the radically reduced redo is the feature that makes GTTs so attractive.

While the Oracle documentation is not entirely clear, it does seem to indicate that GTTs do not generate any redo. While my experiments showed this to be totally false, even the worse case scenario achieved a remarkable 43 percent redo reduction. For example, with a row size of 750 bytes, redo was consistently reduced by 68.1 percent. And even more important, as the row size increased, so did the redo savings.

I would like to encourage you to perform your own experiments. Feel free to use the scripts I developed. While you should obviously review the experimental scripts yourself, you should be able to run them with only minor modifications. The scripts are built to be self-contained and to be easily and repeatedly run. There are three scripts:

1. GO.SH is the main driver script. It creates the experimental results table, sets the appropriate experimental parameters and then calls the main experimental script (DOTEST.SQL). You will need to modify GO.SH based upon your experimental desires and your Oracle username/password.
2. DOTEST.SQL is the script that actually performs the various experiments. You will most likely not need to modify this script.
3. RESULTS.SQL is run after some or all of the experiments are performed. It reports on all of the experiments run and displays the information to make spreadsheet entry very simple. Experimental Environment The experimental environment for any test is very important. Here is a summary of the environment I created:

- The experiments were conducted on an HP/UX (B.11.00 E 9000/800) machine, with 4GB of ram, and 16, 440 MHz CPUs.
- As the experimental code shows, I only tested insert statements and the inserts were performed using a simple PL/SQL loop with a commit every 1,000 rows.
- Two, three-column indexes were also created on the GTT table to make the test more realistic. GTTs have two options related to how long the GTT lives, both of which were tested. The two options are on commit preserve rows, which keeps the GTT and its data until the session disconnects and on commit delete rows, which drops the GTT once a commit is issued. The amount of redo generated during the insert operations was determined by querying v$sesstat and looking at redo bytes written. The row size was varied 15 times in 50-byte steps - from 50 bytes per row to 750 bytes per row. To statistically quantify chance, each row size and the two "on commit" options were run 12 times. So in total, there were 360 separate and independent runs performed (two commit options x 15 different row sizes x 12 runs).

To textually summarize the results, Global Temporary Table redo savings increases as row size increases. Regardless of the row size, inserting into the experimental GTT with the on commit preserve option generated an average of 528.38 redo bytes (standard deviation 0.42 redo bytes) and with the on commit delete rows option an average of 517.08 redo bytes (standard deviation 0.29 redo bytes) were generated. The smallest row size tested was 50 bytes resulting in an average 43.1 percent redo reduction (standard deviation 0.00) . The largest row size tested was 750 bytes, which resulted in a 68.1 percent, redo reduction (standard deviation 0.00). Full experimental results are available for free on OraPub's web site.

About the Author
Craig Shallahamer's 17-plus years of experience in the IT marketplace brings a unique balance of controlled creativity to any person, team or classroom. As the President of OraPub, Inc., his objective is to empower Oracle performance specialists and capacity planners. To find out more about OraPub and its free tools and technical papers, visit its web site athttp://www.orapub.com.

Advertisers' Index

call the numbers listed below. And be sure to let them know you saw their ad in Select!

Advertiser Phone Web Address Page Number
BMC +1.713.918.8800 www.bmc.com Inside Back Cover
DB Doctor +1.303.642.4252 www.dbdoctor.net 32
Hot SOS +1.817.329.3155 www.hotsos.com Back Cover
FiveNine Solutions +1.775.852.2995 www.fiveninesolutions.com 5
Pocket DBA Systems +1.800.404.4059 www.pocketdba.com 2
Quest Software +1.949.754.8000 www.quest.com 9
Revealnet +1.800.738.3254 www.revealnet.com 7
SQL Power Tools +1.702.950.3209 www.SQLPower.com 10
TUSC +1.630.960.2909 www.tusc.com Inside Front Cover
Veritas Software +1800.729.7894 www.veritas.com/oracle 1