

Tuning the Data Warehouse According to its Usage:

Usage Tracking in Decision-Support Systems

Do You Really Know What's Going On?

Tim Gorman

Principal - *Evergreen Database Technologies, Inc.*

Sweet smell of success!

It's finally up and running!

Starting as a gleam in a forward-thinker's eye, born of the persuasive arguments of an early guardian angel. All of the analysis, all of the design and data modeling, all of the sheer work involved in extracting precious data from the jealously-guarded mission-critical systems upon which the business depends is completed. The data warehouse is up and running.

It's being used! -- you can see from the performance monitors that the system is being pushed hard. Feedback from the user-community is positive -- everything is being delivered as promised. The CEO is happy, the CFO has found something important already, and the CIO looks at you with affection and awe.

The data warehouse is finally up and running. The business is analyzing its own day-to-day data, taking the long view, seeking new and interesting questions, answering strategic imperatives. Success is sweet!

Now, fast-forward four months...

The honeymoon is long over...

Load-processing has been interrupted by "out of space" errors every other day for the past two weeks. The DBAs are trying to find more space, but progress is slow and incremental.

A full third of the user-community is unhappy with response-time. You're in negotiations with the corporate sponsors on a *service level agreement*, and you're not happy with their unrealistic expectations. You're being asked to guarantee an average response time, across a user-community that stretches across 12 time zones.

You're being asked to commit to 24x7 availability for the system, so that corporate analysts can burn the midnight oil from Sydney to Stockholm, even on the weekends.

Of course, all this is on top of your newest projects, enthusiastically heaped upon your shoulders in the aftermath of the successful data warehouse project.

That data warehouse now sometimes seems to have turned into a monster -- growing without bound, consuming huge amounts of hardware in an ever-increasing spiral, crashing more frequently due to the hurried nature of hardware and software upgrades.

The Riddle

It sounds almost like a riddle.

Question: What's worse for your career than failing at a data warehouse project?

Answer: Succeeding at a data warehouse project.

By its nature, a data warehouse is something that can grow without bound, when successful.

Analyzing five years of data can be more illuminating than analyzing two years. As a result, the retention period of data might be increased by the corporate sponsors.

Of course, when the corporate sponsors change the requirements, they realize that they are going to have to pay extra. More insidious (and more common) is the situation where no archive/purge was implemented or even designed. In general, archival and purging of data is the last thing considered during traditional systems implementations, and this is also true during the compressed timeframes of data warehouse implementations. Additionally, since much of the analysis to be performed on a data warehouse cannot be projected in advance, it seems fruitless to even consider archival and purging until the usage patterns of the user-communities becomes clearer.

Add to this the fact that *relational on-line analytical processing* (ROLAP) utilities require huge numbers of indexes in order to access data from all the different dimensions. Instead of the **data:index** ratio of **2:1** for

Tuning the Data Warehouse According to its Usage

space consumption that everybody assumed from more familiar operational systems, the ratio now seems to look like **1:2** or **1:3**, meaning that more space is consumed with indexing than with the data! For such huge data volumes, this is no laughing matter.

Knowledge is power

The *operational* systems which run the business are generating data. While they have some reporting capabilities, they typically are not designed for, and generally cannot be optimized for, transforming that detailed transactional data into strategic *business information*.

Turning *data* into *information* was the justification for the data warehouse in the first place.

Now, the data warehouse itself is not providing information. The information that it is not providing is information about itself, such as:

- What data is being used? How often? By whom? When?

If we knew the answer to these, we would know where to concentrate our efforts in performance tuning, availability and redundancy, as well as redesign for the data model.

- What is not being used? Is it ever being used? Or, is it being used less often based its *age*? Or, some criteria other than *age*?

Answers to these questions would suggest optimal archiving and purging strategies.

- Who is using the data warehouse? Should they be there?

Suppose funding for the data warehouse could be based on who is using it? Suppose it already was, and someone from a non-funding department was using it?

- Who is not using the system who should be?

Users are incredibly resourceful about obtaining business data. If someone is reporting about data in the data warehouse but not using the data warehouse, where are they getting it? Maybe they know something you don't, or maybe they need to know about the data warehouse. Maybe they need to be persuaded...

- What are the most resource-intensive operations on the system?

Fix these, and you may not have to procure more hardware. Fix these, and formerly unhappy users become the data warehouse's biggest fans.

- What is the average response-time of queries across the system? For queries from department X? For queries from user Y and Z?

...now, you're one step closer to signing that service level agreement with the response-time requirements, because whatever you can measure you can meet or improve.

The aftermath of success

Asking these questions is the first step toward improving the data warehouse. It is the aftermath of a successful data warehouse implementation, and they are questions generally asked by people contemplating the *next-generation data warehouse*.

An understanding of the hierarchy of needs implies that more basic issues are dealt with while the data warehouse is being built. Issues such as the basic data model, the overall design, building and testing the extraction, transportation, and transformation processes, and simply procuring and configuring the hardware and software.

Therefore, it is only after the success of this initial phase of data warehousing that these questions have been asked, which then seems entirely natural. After all, why worry about what color your living room carpet should be when you are trying to complete your new shelter before winter?

In reality, providing answers to these questions is not quite as trivial as that, and should receive a higher priority than most people (with the probable exception of Martha Stewart) would allow.

Providing answers to these questions must be part of the *infrastructure* of the data warehouse, and not a separate project. Therefore, it makes sense to consider these questions as part of the up-front analysis of the data warehouse implementation project, as everybody knows that it is easier to build something into the infrastructure from the beginning rather than try to retrofit it in later.

Two approaches

There are two basic approaches to getting these usage questions answered. Each approach has its own assumptions, tools, and methods.

The first approach is to impose traditional operational controls on the data warehouse environment, using proven methods such as restricting or eliminating *ad-hoc* accesses, job control systems, job prioritization, and full documentation of all processes, accompanied by change management procedures.

The second approach is less intrusive. It advocates a more *reactive* approach to the problem. Rather than *proactively* impose centralized controls upon the data warehouse, the second approach focuses on *monitoring* and *recording* usage patterns in the data warehouse for later *analysis*. Following analysis, active measures can then be taken, reactively. No effort is made to change the ways that users access the data warehouse across the board, unless analysis suggests this as the best course of action (which is rare). More often than not, analysis will pinpoint specific optimizations that are transparent to the user-community.

1. Centralizing control

This approach is familiar to organizations and people with experience in more traditional operational *on-line transaction processing* (OLTP) systems, such as *order entry*, *inventory*, *customer service*, etc. In these environments, centralized control is clearly advantageous, as the way that end-users access the system is tightly constrained by forms, canned reports, and batch programs. Each of the access methods are developed by IT professionals and designed to optimize the performance of specific, well-documented tasks. There is generally little (or no) *ad-hoc* access to these systems.

Long-running operations are typically executed in *batch*, using job control systems. Multiple *job queues* are employed by the job control systems to throttle and redistribute the consumption of resources. The user communities of these systems are typically large numbers data entry operators.

2. Monitor, record, analyze, and correct

As this approach is more *reactive* than *proactive*, it seems to suffer from a credibility gap from the beginning. Certainly, a *proactive* approach is superior to a *reactive* one?

Well, this would be true, provided you had accurate information on which to act. The *monitoring* and *recording* of usage data, followed by *analysis*, provides that intelligence. Then, specific *corrective action* can be taken to provide accurate fixes.

One of the goals of a decision-support environment is often to decentralize control of the data, to eliminate the middle-person from the process of turning data into information.

Imposing centralized controls can violate this basic assumption.

Analysis implies a train-of-thought, not a disconnected series of actions without continuity. Analysis has a *momentum*, with the answer to one question leading to another question. As humans, a quick turnaround time for answers to questions is a requirement to keep that momentum. If it takes hours or days to answer a question, then this momentum can be lost.

If the IT department takes the simplistic approach of filtering all queries through a series of job queues, it is possible that a particular query can be delayed for hours, breaking the analyst's train of thought.

However, if analysis reveals that the majority of queries can be made more efficient with a small addition to the data model, then it may not be necessary to queue the queries. As Yogi Berra said notably, "*You can observe a lot by looking.*"

Applying data warehouse techniques to data warehouse management

OLTP systems are implemented from a large set of specific requirements. Decision-support systems (DSS) are implemented to perform foreseen as well as unforeseen analysis.

Managing OLTP systems lends itself well to centralized control. Managing DSS systems, with its rapidly changing and largely *ad-hoc* workload, requires analysis before action. As a result, administrators of the data warehouse need to change their viewpoints, much the same as their counterparts in the development and business organizations had to shift theirs in building the data warehouse.

Usage tracking

The basic intent behind the strategy to *monitor*, *record*, *analyze*, and *correct* is also known as *usage tracking*.

Tuning the Data Warehouse According to its Usage

There are three general mechanisms for usage tracking:

1. Purchase or build end-user tools which record usage and present it for analysis
2. Sample usage information from the database's internal views
3. Intercept, record, and analyze (all or selected) network traffic

Purchase or build end-user tools which record usage and present it for analysis

Oracle's *Discoverer* product is a good example of a product which is designed to track its own usage and conduct self-improvement.

The *Discoverer* product has two modules: *user* and *administrator*. All access to the target data is controlled through the *end-user layer* (EUL), which is a series of tables and procedures in an Oracle database. End-users use the *user* modules to access the target data through the metadata provided in the EUL. System administrators set up and maintain that metadata in the EUL using the *administrator* modules.

The EUL also captures and records all SQL statements issued by the *user* modules, storing these statements for analysis. This analysis can be viewed through the *administrator* modules.

Based on *Discoverer's* analysis of the SQL statements issued through the *user* modules, the *Discoverer administrator* can take several actions. One major feature of the *Discoverer* product is *summary management*, where repeated large queries are distilled into a maintained *summary table*. Following the creation of the summary table using the *Discoverer administrator* module, subsequent queries issued by the *Discoverer user* modules will be automatically redirected against the more-efficient summary table, instead of the originally-requested detailed data tables.

There are other products with similar functionality, but Oracle *Discoverer* is an advanced example of a single product providing advanced usage tracking capabilities.

Check with your Oracle sales representatives for a more comprehensive description of *Discoverer's* capabilities.

Sample usage information from the database's internal views

Over the years, I've been tuning all kinds of Oracle-based applications using the information found in the dynamic performance (or V\$) views in the database.

In particular, the V\$SQLAREA view provides an excellent snapshot of all the recently executed SQL statements across the entire database instance. V\$SQLAREA is a *real-time* glimpse into the Shared SQL Area of the Shared Pool in the Oracle *system global area* (SGA). The Shared SQL Area is a memory-based cache of SQL *cursors* (memory structures used to execute SQL statements), and this cache is managed using a *least-recently-used* (LRU) algorithm. Essentially, frequently used cursors will persist in the Shared SQL Area cache, while cursors that are not used frequently may *age out*. Of course, the size of the cache has an impact on the retention period in the cache.

Some useful columns in V\$SQLAREA include:

<u>Column-name</u>	<u>Comments</u>
SQL_TEXT	SQL statement text
BUFFER_GETS	cumulative logical I/Os by the SQL statement
DISK_READS	cumulative physical I/Os by the SQL statement
USERS_EXECUTING	Number of executions of the SQL statement
HASH_VALUE	Identifier value derived by hashing the text of the SQL statement
ADDRESS	An binary-address value representing the physical address of the cursor in the Shared SQL Area in memory

Since V\$SQLAREA is a *real-time* view into the Shared SQL Area cache, it is an excellent tool for analyzing *recent* SQL activity.

Additionally, there is another view, V\$SESSION, which has a relationship to the V\$SQLAREA view via the foreign-key columns SQL_HASH_VALUE and SQL_ADDRESS, related to the V\$SQLAREA primary-key columns HASH_VALUE and ADDRESS. The V\$SESSION view contains information about the

Tuning the Data Warehouse According to its Usage

client-side of an Oracle client-server connection. V\$SESSION only shows active, connected sessions.

<u>Column-name</u>	<u>Comments</u>
SID	Session ID, an identifier for an active client-side session
SQL_HASH_VALUE	Foreign-key to the HASH_VALUE volume on V\$SQLAREA
SQL_ADDRESS	Foreign-key to the ADDRESS column on V\$SQLAREA
USERNAME	Oracle account for the session
OSUSER	Operating-system account on the client-side machine for the session
MACHINE	The network hostname or IP address of the client-side machine for the session
PROGRAM	The name of the program module on the client-side for the session

When a session is actively processing a SQL statement, there will be a relationship between the V\$SQLAREA and V\$SESSION views. Any query which joins these two views will display actively running SQL statements, and some basic information about who is running them.

However, all Oracle V\$ views are reinitialized every time the Oracle database instance is started. This means that you cannot analyze information across database instance shutdowns.

What is needed is a cumulative repository of all information every displayed in the join between the V\$SQLAREA and V\$SESSION views. Unfortunately, this does not exist.

One ideal solution to creating a persistent cumulative repository is to create a database trigger to log all changes to the V\$SESSION and V\$SQLAREA views to a permanent table. Unfortunately, Oracle does not allow database triggers to be created on the V\$ views or their underlying X\$ virtual tables.

The next best option for trapping the information in V\$SQLAREA and V\$SESSION is to *poll* it periodically, which is why I wrote the USAGE package in PL/SQL.

The USAGE package has three procedures:

POLL	Queries V\$SQLAREA and V\$SESSION, and inserts the information into permanent USAGE tables
EXPLAIN	Performs an EXPLAIN PLAN on SQL statements stored in the permanent USAGE tables
PURGE	Deletes information from the permanent USAGE tables, based on specified retention periods

The DBMS_JOB package in Oracle is used to execute the USAGE.POLL procedure every so often, say every 15 minutes. DBMS_JOB is also used to execute the USAGE.EXPLAIN procedure less frequently, say every 12 hours. The USAGE.PURGE procedure should be then be executed far less frequently, say once per day or once per week, to keep the permanent USAGE tables from growing too large.

Check your standard Oracle *Server Application Development Guide* for more information on enabling and using the DBMS_JOB submission facility.

The USAGE package has the advantage of being absolutely free; it can be downloaded from the Evergreen Database Technologies web-site at <http://www.evergreen-database.com/>. However, as free software, there is absolutely no warranty, and it's installation, configuration, and maintenance is the sole and complete responsibility of the user.

The USAGE package comes with a small number of SQL*Plus reports included. Some of the analysis that can be done is straight-forward: top 10 worst SQL statements, SQL statements by Oracle account, etc. Others are more involved, such as querying the EXPLAIN PLAN information to determine which indexes are being used.

Care should be taken to remember the *polling* nature of the USAGE package. It is not catching everything. In fact, given a polling frequency of 15 minutes, it will be only a matter of luck if SQL statements that complete more quickly will ever be recorded. Thus, it

Tuning the Data Warehouse According to its Usage

must be remembered that the USAGE package will only trap SQL statements that take longer than the polling frequency (i.e. 15 minutes) to complete.

Intercept, record, and analyze network traffic

Pine Cone Systems, Inc., founded by data warehousing guru Bill Inmon, has a product called *Usage Tracker*, which is designed to *sniff* network traffic between the *client-side* and *server-side* processes in an Oracle client-server connection.

This is accomplished using a technique called *spoofing*, in which a Pine Cone process *impersonates* an Oracle SQL*Net listener process or an Oracle server process. The Pine Cone process, called a *SQL Trap*, does not replace the Oracle processes. It merely *sniffs* the contents of the network traffic as it passes. So, in plumbing terms, the Pine Cone *Usage Tracker* product becomes a *tap* or *T-fitting* on the *pipe* that is the SQL*Net connection between the client- and server-processes. Oracle network traffic continues to pass back and forth unchanged as before, but there is an extra process in the mix. Now the data is also shunted off to another destination: the Pine Cone *Usage Tracker* data mart.

Usage Tracker is therefore very non-intrusive. All products which access an Oracle database instance from the network must pass through SQL*Net and Net8 (i.e. the Oracle8 version of SQL*Net). No exceptions.

Because of this, *Usage Tracker* is ideally suited to trap everything that is executed against the Oracle database instance. Furthermore, for simple client-server connections, this means that *Usage Tracker* has the closest end-user perspective to be found on the database server. Pine Cone becomes a SQL statements point-of-entry to the database server, and it also becomes the point-of-exit for all results returned by the SQL statement. Therefore, *Usage Tracker* is able to effectively measure response-time end-to-end on the database server.

Like the USAGE package, *Usage Tracker* traps information about *who* executed the SQL, *when* it was executed, and from *where* on the network. But *Usage Tracker*, as a fully-supported licensed product, has much superior analysis and reporting functionality built-in. In fact, I originally developed the USAGE package as a *poor man's alternative* to the *Usage Tracker* product. It was never intended to replace

Usage Tracker, but rather it is intended to better display the need for a more complete SQL monitoring solution.

Contact Pine Cone Systems, Inc. ("<http://www.pine-cone.com>") for more information about the *Usage Tracker* product and its companion products:

- *Cost Tracker*
- *Content Tracker*
- *Refreshment Tracker*
- *MetaExchange* and *MetaView*

Pine Cone Systems is not the only vendor in this product space. Other vendors with similar products include:

- CYRANO ("<http://www.cyrano.com/>")
- Teleran Technologies ("<http://www.teleran.com/>")

There are only the companies of which I am aware; if I missed anyone, it is an omission based solely on my ignorance, and please (gently) let me know more about you...

Tradeoffs

Each of the three *monitoring, recording, analysis, and correction* options have their pro's and con's.

Front-end tools such as Oracle's Discoverer include some usage tracking administrative functionality, in addition to all of its other attractive features. There are a lot of good reasons to use Discoverer and similar products; usage tracking is just another.

However, tools such as Discoverer are no good at tracking SQL statements issued by other tools and program modules, so that can be considered a significant drawback.

The USAGE package has the sole advantage of being free and relatively simple. Additionally, it has the advantage of trapping SQL activity regardless of the point of origin, although its polling nature makes it almost certain to miss any SQL statements that execute quickly. There are many drawbacks due to its status a bit of software developed over the course of about 2 weeks by a programming hack. However, it could prove useful as a stopgap or further justification to a more complete solution such as Pine Cone's *Usage Tracker* product.

Tuning the Data Warehouse According to its Usage

Despite its status as the most complete tool set I've seen, the *Usage Tracker* product does have some drawbacks.

To prevent performance problems, *Usage Tracker* does not query information from the database in real-time. It only samples data from the network traffic stream. Therefore, it only *sees* the text of a SQL statement, but has no insight into whether the objects being referenced by the statement are *tables* or *views*. Additionally, unless a *fully-qualified* table name is used (i.e. both the table's owner as well as the table name is specified), *Usage Tracker* has no idea exactly which table or view is being referenced, if there is more than one object owned by different schemas. Last, Pine Cone cannot penetrate into PL/SQL procedures or functions, so that applications which make heavy use of PL/SQL may prove impervious to analysis by *Usage Tracker*. This is a situation where the USAGE package may have an advantage!

Last, *Usage Tracker* was intended for simple 2-tier client-server connections to data warehouses based on Oracle, Sybase, Informix, DB2, and Teradata. SQL activity trapped from a 3-tier architecture (i.e. a transaction-processing monitor or a web server or an object broker) may also prove impervious to analysis by *Usage Tracker*. This would be due to the potential stripping-away of client-side information by the middle-tier of the 3-tier architecture.

Still, *Usage Tracker* is designed from the ground up specifically for data warehouse management, non-intrusively, in an enterprise-wide, heterogeneous relational data warehouse environment. It is a new product, and it is being improved rapidly, ahead of the curve in the relatively new area of data warehouse management.

Summary

More than anything else, the need to answer the questions posed at the beginning of this paper points a new role, rarely identified at the start of a data warehouse project. That role is the *data warehouse administrator* (DWA).

Most project leads understand that a System Administrator (SysAdmin) role is required, to administer the hardware and the operating-system. Also, the role of the Database Administrator (DBA) is widely recognized, to administer the database upon which the data warehouse is based. The roles of Network Administrator and Customer Support are also widely recognized.

Often, one or several of these roles is also expected to take on the additional responsibilities of DWA. But the prospect of managing the data warehouse, indeed the responsibility of installing, configuring, and using a usage tracking tool such as Pine Cone's *Usage Tracker*, flushes out the fact that someone with global responsibility for data warehouse administration is necessary. Each of the other roles listed here (i.e. SysAdmin, DBA, NA, or CS) may have several responsibilities beyond the data warehouse that make them unsuitable as a DWA. During the development phase, the technical project lead may fill this role, but after the database transitions to production status and the project lead moves onto his/her next project, this responsibility goes unfilled.

The responsibility for improving and enhancing the data warehouse belongs to the role of data warehouse administrator. Whether this is a separate full-time position or an additional part-time responsibility of another role depends on the scale and scope of the data warehouse, as well as the depth of desire by the organization that the data warehouse continue to succeed.

Additional information on the Web

Oracle Corporation

<http://www.oracle.com/>

<http://technet.oracle.com/>

<http://education.oracle.com/>

<http://support.oracle.com/>

Evergreen Database Technologies, Inc.

<http://www.evergreen-database.com/>

Pine Cone Systems, Inc.

<http://www.pine-cone.com/>

Teleran Technologies, Inc.

<http://www.teleran.com/>

CYRANO

<http://www.cyrano.com/>

The Data Warehouse Institute

<http://www.dw-institute.com/>