Summer 2013

Evaluation of Sql Performance Tuning Features in Oracle Database Software

Katarzyna Marta Dobies
Regis University

EVALUATION OF SQL PERFORMANCE TUNING FEATURES

IN ORACLE DATABASE SOFTWARE

A THESIS

SUBMITTED ON 23 OF AUGUST, 2013

TO THE DEPARTMENT OF INFORMATION SYSTEMS

OF THE SCHOOL OF COMPUTER & INFORMATION SCIENCES

OF REGIS UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF MASTER OF SCIENCE

IN SOFTWARE ENGINEERING AND DATABASE TECHNOLOGIES

BY

Katarzyna Dobies

Katarzyna Marta Dobies

APPROVALS

Darl Kuhn

Darl Kuhn, Thesis Advisor

Donald J. Ina

Don Ina, Faculty of Record

Nancy Birkenheuer

Nancy Birkenheuer, Program Coordinator

Abstract

Timely access to data is one of the most important requirements of database management system. Having access to data in acceptable time is crucial for efficient decision making. Tuning inefficient SQL is one of the most important elements of enhancing performance of databases. With growing repositories and complexity of underlying data management systems maintaining decent levels of performance and tuning has become a complicated task. DBMS providers acknowledge this tendency and developed tools and features that simplify the process. DBAs and developers have to make use of these tools in the attempt to provide their companies with stable and efficient systems. Performance tuning functions differ from platform to platform. Oracle is the main DBMS provider in the world and this study focuses on the tools provided in all releases of their software. A thorough literature analysis is performed in order to gain understanding of the functionality and assessment of each tool is performed. It also provides insight into factual utilization of tools by gathering responses through the use of online survey and analysing the results.

Acknowledgements

To my fellow students,

Thank you for sharing your knowledge with me. It has been a pleasure…

To Darl,

One of the greatest achievements of my life would be to become your equal in a discussion,

It has been a privilege…

To Marcin,

For some only exist thanks to the silent presence by their side…

To Mom,

You've given me strength and determination that made me who I AM…

To Dad,

You're my STAR, shining on me now…

To Robert,

If I could pick from all the boys in the world, I would pick you,

Always…

Table of Contents

List of Figures

List of Tables

Chapter 1 – Introduction

Globalization greatly defined dynamics of nowadays business and considerably altered the decision making process. Lack of geographical ties made around-the-clock service a necessity. Ubiquitous computing broadened the client range, its heterogeneity and dispersion. Social media enhanced access to information immensely. Society is changing – the Generation Y and, coming of age, Generation Z – are technology savvy, highly dynamic and demanding superb quality of interaction with suppliers, while being considerably less brand loyal and increasingly flexible.

Recent economic turmoil and its aftermath proved disastrous to plenty of respected and seemingly stable businesses and put enormous pressure on the management. Paramount is the importance of making deliberate steps and well-informed decisions which often determines the success or failure in the market.

In the world where people are more than ever reliant on technological enhancements and the time slots of the day-to-day operations are decreasing, businesses are under profound pressure to react promptly to changes in their environment. Ability to adapt to these dynamics is crucial. Access to important data to derive information and predict future trends is an indispensable pillar of the decision making processes.

Companies support their operational and strategic needs by investing in software that supports data management and retrieval. Presently, the most popular method used for data storage is the relational approach, which is based on relations between tables of data. This system creates an intermediary entity (database management system) between data and client that is responsible for managing complexities of data storage and retrieval and relieves the client from tasks of maintaining integrity of data.

Oracle Database is one of the available solutions on the market. Investments in such systems usually consume a considerable percentage of a company's budget. Given that, a return is expected in terms of productivity of software. Such package is to provide the site with a comprehensive set of tools for data management to facilitate efficient data retrieval. Timely access to data is one of the most important requirements of database management system. Having access to data in acceptable time is crucial for efficient decision making. Any bottlenecks in processing data need to be addressed.

Performance tuning is a process set to analyse these bottlenecks, reduce data access times, increase throughput and improve the overall functionality of database management software. It concentrates mostly on areas such as SQL tuning, memory tuning and I/O tuning.

Oracle Database software introduced a set of features aimed to support DBAs in performance tuning process. DBAs have to make use of those tools in order to maximize investment return and provide their companies with efficient data environment.

Performance tuning functions differ from platform to platform. This thesis provides intensive analysis of literature on SQL tuning features and how to maximize the use of them in Oracle Database software.

Thesis statement

Oracle introduced SQL performance tuning features in its RDBMS software. DBAs and developers must take advantage of these tools to maximize a company's investment in the Oracle RDBMS. This thesis explores Oracle Database SQL tuning features and how to maximize the benefits of these tools.

Scope

Database performance tuning is a vast area thus the scope of this work is limited only to SQL tuning features introduced through Oracle RDBMS releases. At the time of writing, the Oracle Database 12*c* has only been released, therefore tools specific to this release are only presented for reference. Once the 12*c* version is widely adopted within the industry, research on additions in the area of SQL tuning would be a great supplement to this study.

The use of indexes is sometimes analysed in conjunction with other performance related issues. The scope of this study already includes a broad variety of tools, thus a decision was made to exclude index-related tools.

There are third-party SQL tuning tools available on the market (such as Toad from Qwest Software or DB Optimizer from Embarcadero Technologies); they will not be included in this work.

Research Methodology

Critical analysis of available literature in the subject of Oracle database SQL tuning will be performed. Available tools and features will be categorized and assessed in terms of the benefits of their usage and their successfulness.

Also a questionnaire will be used to provide information on utilizing the features in real environments. The contents of the questionnaire are presented in Appendix A.

Significance of the Study

An investment in packages such as Oracle DBMS is usually of considerable significance to a company's budget. Being able to utilize available enhancements in hired software is crucial to effective data management, thus yielding higher investment return.

Effective data management and retrieval is often a crucial factor contributing to company success on the market. SQL tuning is an excellent tool ensuring response times and throughput are of accepted levels.

With growing heterogeneity of customer base, their dispersion and the supporting technology, Database Management Systems have grown to immense sizes, challenging abilities and knowledge of not only novice DBAs. Staying on top of new technology being implemented with each software package release is of great benefit to the companies and knowledge in the area should be strived for by the professionals in order to stay competitive.

Success Criteria

The paper is to provide an overview of performance tuning issues based SQL tuning functionality available in Oracle DBMS. It will present the importance of SQL tuning along with its benefits to the database system. It is also to explore the benefits of utilizing Oracle SQL tools and how in reality they are implemented by various DBAs.

Structure

The remainder of this thesis is structured as follows:

- Chapter 2 – Review of Literature provides a review of available literature concerning database performance tuning. It presents the theoretical base for the subject, touching on categories of performance tuning with the emphasis on SQL tuning and its benefits.
- Chapter 3 – Research Methodology provides description of methodologies used within the study.

- Chapter 4 – Results analyses the discoveries of the literature and the questionnaire. It describes SQL tuning tools in Oracle Database and categorizes them into subsections based on their characteristics and functionality. This chapter also presents analysis of responses to the survey questions and their implications.
- Chapter 5 – Discussions summarizes findings of literature and survey and draws conclusions. It evaluates the beneficence of each tool in performance tuning strategy and describes the circumstances in which they should be used.
- Chapter 6 – Conclusions presents the findings of the research and conclusions. It will present the summary of discoveries and propose further research.

Chapter 2 – Review of Literature

The primary objective of database systems is to retrieve and present requested data from the underlying physical structure. In the advent of the relational databases, access to data was extremely complicated by procedural approach. In order to fetch data, the system had to be told exactly where required data was stored. This in turn asked for profound knowledge of the data system architecture and only people possessing programming skills could perform even the easiest queries (Rob, P., Coronel, C., Crockett, C., 2008).

Codd (1970) set ground for an innovative solution for data storage and management which revolutionized the database industry. The emergence of relational concept diverted the courtesy of directing access and pre-organization of data from the user to the database management system (DBMS). The introduction of SQL and its adaptation in Oracle version 2 released in 1979 by Oracle Corporation set a standard of communication between the user and system (Greenwald, R., Stackowiak, R., Stern, J., 2007).

Structure Query Language is a non-procedural language which typically does not provide the system with guidelines on how to access data. It hides complexities of data retrieval from the user but puts extra pressure on the database management system. It is the role of the DBMS to act as an intermediary between user and data and to decide how to access and pre-organize data for presentation. Connolly and Begg (2010) enlist some of the most important DBMS roles as:

- data storage, retrieval and update – core function of data management system;
- managing data dictionary – storing data about data, or in other words – metadata;
- transaction support – transaction is a logical unit of work that a database performs; either all elements are completed or the whole set of activities (a whole transaction) is dropped so that the database is left in consistent state; this function

relates to ACID (Atomicity, Consistency, Isolation and Durability) qualities of transaction;

- concurrency management – enabling multiple database connections while still preserving consistency of database;
- managing data integrity and consistency – data are accurate by enforcing constraints, such as primary, foreign or unique keys;
- backup and recovery functions – allowing database to return to consistent state if the system crashes;
- authorisation – ensuring data is accessible by users of the right privileges, sensitive information is guarded against illegal access;
- communication facilities, i.e. for remote database access – distributed databases and other systems require some sort of communication link which has to be enabled so that remote users can access the database; this function is growing in importance with cloud computing gaining wide acclaim in the information systems arena;
- separation of data from the middle tier applications – this function allows better portability and changes to the business logic;
- other system utilities – support database administration by monitoring, maintaining statistics, garbage collection etc.

The list of DBMS duties has grown over years. These responsibilities are extremely time consuming and have a profound impact on overall performance of the system. Complex

algorithms guide and support these functions, so that the negative impact is minimized and benefits outweigh the costs.

What is database performance?

Database operation relies heavily on its performance levels. Database performance is related to the ability to retrieve and present requested data from the file system and is defined as "the activity making a database system run faster" (Shasha, 1996). TPC.org (n.d.) describes performance in terms of tasks the database is able to perform in a unit of time, i.e. per second, per minute. Rob, Coronel and Crockett (2008) define it as "a set of activities and procedures designed to reduce the response time of the database system".

Aside from *time* Millsap (2010) depicts *throughput* as a measure of performance, which is an amount of task executions completed within a predefined time slot. Therefore its goal is to minimize the response time and maximize throughput. Response time and throughput are not reciprocal measures and efforts should be undertaken to assess both in order to evaluate database performance health.

Performance is in direct correlation with how the system resources are allocated (Dageville B. G., 2005). These are typically CPU, memory and I/O services, which configuration may either boost performance or hinder it. The complexities of relationships that interlink the resources make performance tuning a time consuming and highly difficult task.

Millsap (2010) stresses that performance is a feature destined to be carefully designed and built, rather than just happening, yet still its levels can only be accurately assessed when under production workload. Runtime workload volumes define greatly what performance levels are achieved by database systems.

Categories of database performance tuning

Database performance tuning is a vast area, spread over many diverse issues within and beyond database environment. Good level of performance starts with a careful design and testing and its goal is to execute queries as fast as possible.

Depending on the initiator, database tuning takes the form of (Oracle, Oracle Database 2 Day + Performance Tuning Guide 12c Release 1, 2013):

- proactive database tuning – an ideal system, in which database administrator initiates tuning activities on a daily basis, minimizing probability of unexpected performance crisis or even outage; the proactive approach requires constant system monitoring and tuning;
- reactive database tuning – is usually initialized by end users complaining about system's performance level; this approach requires analysis of symptoms in order to find an appropriate fix before the system's performance reaches crisis levels; it is a post-facto effort and its occurrence should be minimized by employing proactive tuning.

Database tuning typically demands knowledge spanned across multiple areas within database system environment. Rob, Coronel and Crockett (2008) acknowledge the complexity of performance tuning activities and variety of entities under impact and present guidelines for achieving better performance. As shown in Table 1, they divide actions on database tuning into client and server as well as hardware and software side. Whether on the client or server side, the resources have to be configured so that best response times and throughput are possible. Each of the resources has its parameters which contribute to the overall systems performance. Their amount and quality usually is constrained by budgets and other regulations within company.

Given such restrictions, very often the only solution to ascertain acceptable performance is by monitoring the system and regular tuning.

Table 1: Performance improvement areas. Adapted from "Database Systems: Design, Implementation and Management" by P. Rob, C. Coronel and K. Crockett. Copyright 2008 by Cengage Learning.

	System Resources	Client	Server
Hardware	CPU	The fastest possible	Multiple processors The fastest possible (For example 2 * Quad Core Intel 2.66 GHz)
	RAM	The maximum possible	The maximum possible (e.g. 64GB)
	Hard Disk	Fast IDE hard disk with sufficient free hard disk space	Multiple high-speed, high-capacity (e.g. 750GB) hard disks in RAID configuration
	Network	High-speed connection	High-speed connection
Software	Operating System	Fine-tuned for best client application performance	Fine-tuned for best server application performance
	Network	Fine-tuned for best throughput	Fine-tuned for best throughput
	Application	Optimize SQL in client application	Optimize DBMS for best performance

Note. IDE = Integrated Drive Electronics is a standard used in hard drives where the controller is integrated into the device.

Harrison (2001) describes a performance tuning process spanning all areas of database system implementation. Figure 1 depicts components of performance tuning strategy and their order in the process.

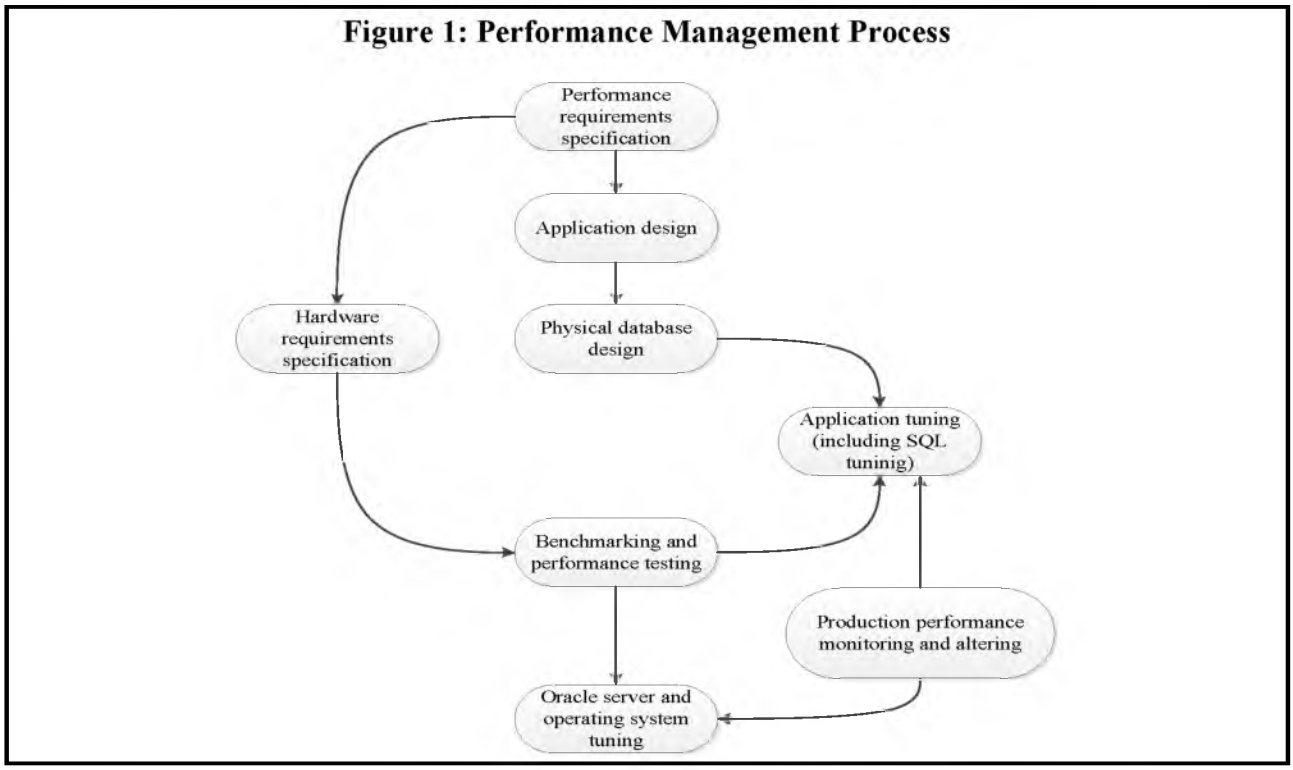

Figure 1: Database performance management process. Redrawn from "Oracle SQL High-Performance Tuning" by G. Harrison. Copyright 2001 by Prentice Hall.

Areas of concern and importance of each of the steps is described below:

- Performance requirements specification – these actions set target performance results used to verify effectiveness of performance tuning efforts;
- Application design – the architecture of application layer determines how the system will deliver service to the users and what performance requirements are; this stage incorporates also logical database design;
- Physical database design – this step is extremely important in achieving performance goals. Database physical layout describes how the data is physically stored which can benefit performance or intensely complicate DBMS ability to satisfy end user needs;

- Hardware requirements specification – as noted before, considerable budget constraints usually bound hardware selection; available configuration options should be explored to select the best and most suitable arrangement;
- Application tuning (excluding SQL) – properly designed and coded applications are efficient and achieve synergy with underlying database construction;
- Application tuning (SQL) – SQL tuning is often number one performance improvement contributor;
- Benchmark and performance testing – allows to assess what performance levels are achieved by configuration;
- Oracle server tuning – it is another important contributor to overall system performance; changing configuration alters the underlying DBMS processes;
- Operating system tuning – along with Oracle server tuning, operating system adjustments can change how DBMS performs its actions and benefit performance;
- Hardware upgrades – a costly option and not necessarily proportionately profitable in terms of performance gained especially when tuning poorly scalable systems.

Incentives for database tuning

Society is growing ever more dependent on information technology increasing the complexity of the supporting IT system. With growing heterogeneity of clients and interrelations between system components, performance is degrading. Database tuning results in lower response times and higher throughput, increasing support for business operation and end users comfort of system interaction. Service Level Agreements legally bind performance levels that

implemented systems have to achieve under any conditions to customer satisfaction. Performance tuning ascertains these conditions are met (Wiese, D., Rabinovitch, G., Reichert, M., Arenswald, S., 2008).

Good level of performance starts with system design and testing, and continues through the deployment of application tier and database level. The earlier in the process of system development life cycle (SDLC) the efforts are undertaken, the least costly it is to make changes and fastest results are achievable (Harrison, 2001).

Increasing database load, that is competition for system resources created by running tasks, degrade response times. *Utilization*, that is relation of resource usage to capacity for a specific time, is a measure of load. With increasing utilization the response time rises (Millsap, 2010). Figure 2 presents the implications of utilization levels compared to response times.

Figure 2: M/M/8 Utilization Model

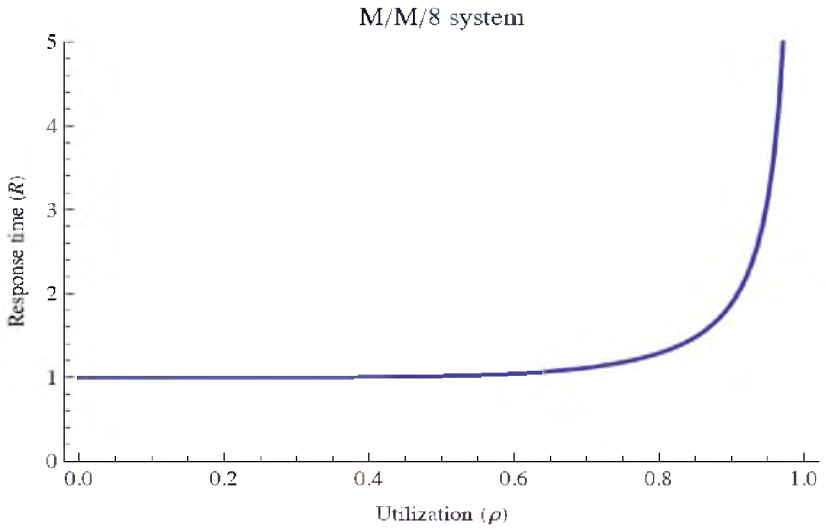

Figure 2: Relation between resource utilization and response time in a system with 8 service channels. Adapted from "Thinking Clearly about Performance" by C. Millsap. Copyright 2010 by System R Corporation.

This model is true under utopian condition of perfect linear scalability which is the ability for the system to adapt to additional load. Realistically it is impossible to achieve perfect scalability in nowadays information system environments. Author corrects that in less linear systems the curve flattens and response times degrade even faster.

In a typical system, resources are limited by budgets and, over time and development of system, become scarce. Database workload competes to acquire service from those resources, which in turn become system's bottlenecks. This causes processes to queue, leading to delays and disappointments of database users.

Workload volume determines performance levels profoundly. Expanding businesses require more complex data systems that grow in complexity and processing requirements. Millsap (2003) claims that 50% or more of average system's workload could be omitted without losing any business functionality. He calls this part of load *waste* and emphasizes how important eliminating those processes would be to the overall performance of the system. Freed resources could be utilized for other operations, helping both end users reach their level of system satisfaction and DBAs in their daily routines, enabling the system to grow even faster.

Vilfredo Pareto, a notable Italian economist, constructed a theory widely incorporated into many of life's disciplines, called 20/80 principle. His theory states that 20% of causes produces 80% of effects. If we adapt the principle to performance tuning arena, we can estimate that 20% of efforts would produce 80% of results. We could extend his theory to workload as well, predicting 80% of load being caused by 20% or processes/users/applications. Therefore in theory, by extending our tuning efforts on 20% of causes we should be able to achieve 80% rise in performance levels (response time decrease or throughput increase).

SQL Performance Tuning

SQL (Structured Query Language, commonly pronounced Sequel) is a non-procedural language used as a means of communication between end user and database management system. It describes data that the user is seeking, but does not typically provide details of how this data should be accessed. It is a portable language, guarded by ANSI and ISO standards (Casteel, 2007).

SQL performance tuning is one of the most important aspects of the overall database performance tuning activities. It is a vast and complex area of database interaction, requiring the tuner to possess knowledge and understanding of domains beyond mere concepts of SQL language.

Tuning SQL is an action aimed at poorly-performing execution plan generated by the query optimizer. System changes such as new statistics, configuration parameters tweaking, software and hardware upgrades can alter the proposed execution plan for SQL. The efficacy of the tuning strategy very often depends on the administrator's experience and skills (Herodotou H., Babu Sh., 2009).

Harrison (2001) points that the earlier in the process SQL is tuned, the least costly it proves. According to his estimates, the proportion of time spent on tuning SQL in development versus production stage of the SDLC is 20 to 1. This stresses the importance of understanding SQL principles, as well as how the DBMS processes the queries and adapting the knowledge from the early stages of system development. He also claims that for established systems SQL tuning might be the only viable option, as some parameters are difficult to change when already in the production stage. SQL tuning is said to be the second best in terms of benefits provided in the overall performance tuning opportunities (see Figure 3 below).

Figure 3: Areas of Benefit of Performance Tuning

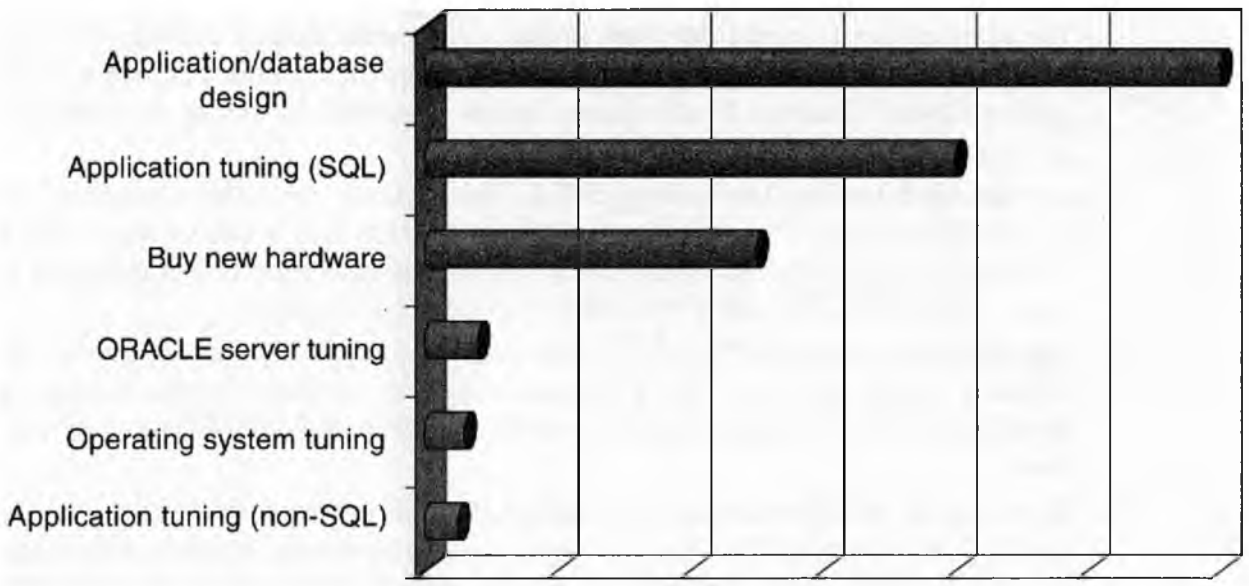

Figure 3: SQL performance tuning effects in the overall performance tuning process. From "Oracle SQL High-Performance Tuning" by G. Harrison. Copyright 2001 by Prentice Hall.

SQL Tuning Benefits

Millsap (2010) puts emphasis on application tuning, advising even off-the-shelf packages tuning as opposed to adjusting to inherently inefficient systems. Furthermore, he persuades that it is best to minimize risk of damage to efficient applications by beginning the process with local changes, rather than global adjustments.

Millsap (1999) points that even CPU upgrades many intuitively believe to boost performance could be risky to performance levels. Millsap and Holt (2003) continue convincing very often hardware upgrades in general can not only cause little or no benefit to overall system's performance, but sometimes even result in performance degradation.

These conflicting opinions prove that database performance tuning and any subset of activities within is a demanding field, often causing disarray between strategies proposed by

various professionals. Naturally, it is difficult to assess and treat each problem using the same measure and very often seemingly similar issues might yield very different solutions.

Harrison (2001) assures that tuning SQL can bring sensational results. The following are according to him reasons why, despite often being an extremely difficult process, tuning SQL should be incorporated into DBAs everyday activities:

- SQL tuning radically improves response times. Response time is one of the most important indicators of system performance health. Rewriting SQL or advising alternative execution paths helps decrease time the DBMS spends on retrieving requested data.
- Support and enhance efficiency of batch processing – these operations process thousands or even millions of rows of data. It is required to allocate time slots for these operations that will not interfere with other system processes. Very often
- Increase systems scalability – the level of acceptance of load growth is very often a cause of major disruptions to the information systems operation. As shown in Figure 2, the more workload a system is presented with the worse performance it generates.
- Reduce workload – as cited in Millsap (2003), 50% of a typical database system workload could easily be omitted without losing functionality required by the end users. Additionally, these activities will most likely free resources that may come of use in other areas.
- Avoid hardware upgrades – while often recommended to improve performance, this method of enhancing system capabilities is rather futile in case of poorly

> scalable implementations that will quickly deplete limited budgets. Millsap (2003) support the notion of their doubtful beneficence, claiming that in some cases hardware upgrades might even degrade performance further, rather than heal it.

It should be highlighted that the best approach to factually improve performance and achieve long term results is to act at the core of the problem, rather than mask it. Taking the easy way that does not resolve the root cause of performance problems will result in recurring issues and growing end user dissatisfaction.

Current Tendencies in Database Tuning

Over time databases do not become less complicated – on the contrary, the abundance of technology in everyday life pressurizes companies to adapt to highly heterogeneous environment. Their supporting information systems are growing in complexity, increasing difficulty of database management and administration.

Performance tuning, spread across diverse and broad areas within and beyond database domain, pose extreme challenge not only to inexperienced DBAs. Very often reliance on experiment driven (or in other words – trial-and-error) methodologies are their only solution (Herodotou H., Babu Sh., 2009). Chaudhuri and Weikum (2005) go further, suggesting that sometimes is resembles more traits of black art, rather than principled engineering.

Millsap and Holt (2003) bluntly state that Oracle performance tuning so difficult that it brings specialists to convene and more often than not argue on the root cause of the problem. They suggest that the traditional Oracle performance tuning methodology is flawed and tools are unreliable and inefficient.

There is a pronounced debate on the need for methodology or a set of rules that could be used in tuning databases. Human error is said to be the biggest reason for performance problems

and system outages (Chaudhuri, S., Weikum, G., 2005). Furthermore, some state that tuning is done manually and reactively, with emphasis on effort put into fixing symptoms rather than their causes (Morton, 2008), (Wiese, D., Rabinovitch, G., Reichert, M., Arenswald, S., 2008).

One of the solutions widely discussed in the recent years is the automation of tuning efforts through incorporation of the functionality in DBMS own processes, transparent to the administrator and end users, yet highlighting issues if any encountered. The emphasis is on self-managing, self-monitoring and self-tuning technologies, hiding complexities of tasks associated with deployment and usage of databases, thus relieving the administrator from manually running many monitoring services. It allows DBAs to concentrate on more strategic actions and increases value of the software suite by simplifying system maintenance tasks.

Wiese et al. (2008) claim there is no formalized standard for best-practice database tuning procedures in the industry. Existence of such commodity could minimize administrators' participation and allow autonomic tuning. They propose an initiative to create community-based SOP-like (Standard Operation Procedure) set of tuning outlines for automating typical tuning tasks. According to their concept, events are occurrences of previously encountered conditions that trigger tuning procedures. These procedures contain tuning actions that the system will undertake should the triggering event happen. The idea simplifies administrators tuning tasks but its drawback is set by limited ability to adapt to events not specified in the triggers listing.

Herodotou and Babu (2009) introduce their *zTuned* tool, which helps automate the process of SQL tuning. They assess that a considerable part of tuning SQL is experiment based and requires a lot of steps which are then verified for success conditions. *zTuned* performs analysis of execution plans generated by the optimizer and present alternatives that are less costly. Their tests show improvement of even 86% in time saved by selecting *zTuned* execution

plan versus the plan generated by PostreSQL own query optimizer. All the trial-and-error steps that the DBA has to perform in order to find this plan are performed by the tool.

Accurate statistics are critical to cost-based optimizers function. The query optimizer uses gathered statistics to generate the best possible access plan. Generation of statistics may be costly in terms of the overhead created in the overall system performance. Therefore determination of required statistics that will contribute to the execution plan selection is a difficult task. Automated creation of ad-hoc query statistics for single column histograms was first used in Microsoft SQL Server 7.0. Some also propose generating statistics based on views that will reflect direct relations between tables and columns [(Chaudhuri, S., Narasayya, V., 2007) as cited in (Bruno, N., Chaudhuri, S., 2002) and (Galindo-Legaria, C., Joshi, M., Waas, F., Wu, M., 2003)].

Chaudhuri and Narasayya (2007) also present other autonomic tuning concepts:

- self-tuning histograms, which are able to update their structure by usage feedback, reflecting the frequently queried data in more detail;
- he points that in terms of monitoring infrastructure there is still a place for improvement by researching the following:
- query progress estimation and ad-hoc monitoring and diagnosis)

Morton (2008) on the other hand shows effects of overreliance on automatic tools might be misleading and cause false belief in good performance condition of the system. She presents her observations of Oracle's ADDM, SQL Tuning Advisor and Autotrace facilities (described in more detail in later chapters). None of these tools were able to point to solution that would meet client's requests of improved performance and only the human intervention, skills and experience led to a satisfying solution. Morton emphasizes that relying solely on indicators of

automated tools we become less knowledgeable in the area and often unable to act without the help of automated tools.

Oracle Performance Tuning Method

Over the years of Oracle Database operation, the suite implemented features that facilitate and in many cases automate tuning efforts. Oracle also proposed a tuning method incorporating these tools in tuning strategy. It is an iterative process and involves the following areas (Oracle, Oracle Database 2 Day + Performance Tuning Guide 12c Release 1, 2013):

- pre-tuning activities – gathering user feedback to specify the scope and goal of tuning efforts; assessment of resource utilization in periods of time that are under concern and enabling automatic tools, such as Automatic Workload Repository and Automatic Database Diagnostic Monitor;
- proactive database tuning – requires monitoring on a daily basis with review of ADDM and other reporting tools; monitoring real-time performance issues with Oracle Enterprise Manager (both Desktop Control and Cloud Control); OEM is an interface built to group all supporting tools for database management that also highlights performance issues; analysis alerts and any negative changes, as well as users satisfaction;
- reactive database tuning – ADDM and AWR report analysis in order to gain understanding of reasons for users complaints; comparison of good and bad performance times and other historical data; ASH (Active Session History) report analysis for short lived performance problems;

- high-load SQL optimizations – identifying inefficient SQL using Top SQL; SQL Tuning Advisor makes suggestions on SQL improvements; SQL Access Advisor and SQL Performance Analyzer are other tools useful for tuning SQL statements.

SQL Tuning Methodology

SQL-specific methodology was also proposed by Oracle. It includes advice related to both designing and deploying new applications (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013).

Design Stage

In the design stage it is important to properly model data structures. If time constraints exist, it is advisable to concentrate greatest effort on the entities most frequently accessed by the system, so that the most sensitive areas of operation are designed with best possible accuracy.

It is also crucial that developers understand SQL processing efficiencies when coding their applications. Following factors should be considered:

- database connection – an expensive operation therefore designing to minimize concurrent connections is recommended; the use of connection pool supports more complex installations;
- cursor management – cursors are reused so that hard parsing can be omitted and only soft parsing occurs for a given SQL statement; hard parsing occurs always when the SQL is submitted to the optimizer for the first time and there is no possibility to find corresponding cursor in the shared memory pool; application should be designed so that a hard parse is performed only once and further executions of the SQL reuse the cursor;

- bind variables usage – sharing cursors requires that previously and currently executed SQL statements are identical; bind variables allow specifying literal values as an argument of the SQL statement, enabling the use of different values for subsequent executions, while still sharing the same cursor retrieved from the shared pool.

Careful design that incorporates above suggestions is likely to produce good performance levels, as well as well scalable application.

Deployment Stage

In the deployment stage Oracle distinguishes between test deployment and rollout into the production system. While testing, Oracle recommends the following:

- ADDM and SQL Tuning Advisor reports monitoring and validation;
- ensuring realistic data volumes and distributions are maintained – fully populated tables and data that is representative to the production environment; structural elements, i.e. indexes and materialized views, have to be created to reflect the realistic data;
- use the same mode for the optimizer – this allows to remain closest to the realities of the production system; if the optimizer uses different settings in testing and production environments the test might not reflect factual performance characteristics;
- single user testing – idle or lightly used database testing for a single user should be performed;

- execution plan documentation and validation – monitoring optimizer choices for all SQL statements involved in deployment allows to verify the most optimal access paths, joins and sorts are chosen and alter the plans if necessary;
- multiuser testing – should be performed as accurately as possible to simulate behaviour of the production system; it allows to verify if serialization or locking issues arise;
- maintaining correct hardware configuration – inefficiencies and shortages of hardware can be addressed if properly tested in a system closely resembling the production environment;
- maintaining steady state performance – benchmark testing should be performed under steady conditions.

The method of introduction of the new functionality into the system also makes difference to performance management. Rollout strategies are divided into:

- Big Bang approach – all users are introduced to the new application at once and the old is simply switched off; this method's advantage over the trickle approach is that only single version of the application is maintained and there is less need for data migration management, yet it requires careful testing in workload conditions closely resembling the production system;
- Trickle approach – introduces users or groups of users to the new application gradually; performance needs and shortages are addressed with time and affect only the users that already migrated to the new solutions; on the other hand, this method poses greater need for managing data synchronization and conversions.

Oracle's presence in relational database market

Oracle is a primary RDBMS provider in the world. Oracle.com (n.d.) presented results for its market share in 2012 (as cited in Garnter, 2013). According to the report, Oracle Corporation holds the lead position in the RDBMS market by 29% compared to the next supplier and its revenue exceeds next four competitors revenues combined. This is an extraordinary result.

Oracle's journey to the top began in 1977. Oracle began work on Codd's relational project. Its first release was a prototype written in asembly language, but the second version was the first in history commercially available database using SQL language. Since then many database enhancements were introduced allowing Oracle Database assertain domination in the market and a solid position for years to come. Table 2 presents the history of releases and notable enhancements of each version.

Table 2: History of Oracle Database releases. Redrawn from "Oracle Essentials: Oracle Database 11*g*" by R. Greenwald, R. Stackowiak and J. Stern. Copyright 2008 by O'Reilly.

Year	Feature
1977	Software Development Laboratories founded by Larry Ellison, Bob Miner, Ed Oates
1979	Oracle version 2: first commercially available relational database to use SQL
1983	Oracle version 3: single code base for Oracle across multiple platforms
1984	Oracle version 4: with portable toolset, read consistency
1986	Oracle version 5 generally available: client/server Oracle relational database
1987	CASE and 4GL toolset
1988	Oracle Financial Applications built on relational database
1989	Oracle6 generally available: row-level locking and hot backups
1991	Oracle Parallel Server on massively parallel platforms
1993	Oracle7: with cost-based optimizer

Year	Feature
1994	Oracle version 7.1 generally available: parallel operations including query, load and create index
1996	Universal database with extended SQL via cartridges, thin client and application server
1997	Oracle8 generally available: object-relational and Very Large Database (VLDB) features
1999	Oracle8*i* generally available: Java Virtual Machine (JVM) in the database
2000	Oracle9*i* Application Server generally available: Oracle tools integrated in the middle tier
2001	Oracle9*i* Database Server generally available: Real Application Clusters, OLAP and data mining in the database
2003	Oracle Database 10*g* and Oracle Application Server 10*g*: "grid computing enabled; Oracle Database 10*g* automates key management tasks
2005	Oracle completes PeopleSoft acquisition and announces Siebel acquisition, thus growing ERP and CRM applications and business intelligence offerings
2007	Oracle Database 11*g*: extension of self-managing capabilities and end-to-end database change environment; Hyperion acquisition adds database-independent OLAP and Financial Performance Management applications

In addition to the above 2013 is the year of long awaited new release that is already known to utilize “pluggable” database concept and is said to maximize the cloud related technology. It is their latest release marked as Oracle Database 12*c* Release 1.

Holding leadership is a responsible function. Current position in the market is very likely a result of many years of efforts to accommodate changing IT environments and adapting to customers needs – both those pronounced, as well as unpronounced. Oracle supports its users with a broad set of tools that enhance utilization of the DBMS.

Summary

This chapter concentrated on providing theoretical context of SQL tuning. It defines performance and SQL tuning and enlists their benefits. Tuining methodologies in the industry are investigated and Oracle’s presence in the database market is assessed.

In the next chapters available SQL tuning functionality will be described along with the benefits they convey. Additionally, the results of a study amongst database industry professionals in relation to their factual knowledge and utilization of SQL tuning tools available in the Oracle's RDBMS Suite is peformed. Next chapter describes the methodology used to derive the conclusions.

Chapter 3 – Research Methodology

In today's competitive, fast-paced business environments it is imperative that access to data is stable and efficient. Ensuring SQL effectively performs queries and delivers results promptly supports informed decisions. Previous chapter presented the theoretical context to the subject of performance and SQL tuning in relational databases. This section of the study will present the methodology used to gain understanding and enable the researcher to draw conclusions on benefits and utilization of SQL tuning tools and features in Oracle Database.

The thesis is divided into two parts and uses a hybrid methodology. For a thorough understanding of available functionality of SQL tuning tools and features literature review is performed. In order to assess factual utilization of named tools within the industry professionals a survey was conducted.

Literature Analysis

In this part a review of the literature is presented. Sources like ACM Digital Library, IEEE, periodical articles, conference proceedings, published books and white papers from vendors will be used. Thorough review of the literature will help the researcher understand the functionality of the tools. They will be assessed in terms of the benefits they bring to the overall performance tuning efforts and what the best environment for their utilization is. Based on the derived information tools will be divided into categories in order to simplify evaluation process.

Questionnaire Analysis

The questionnaire will be available online with the use of Survey Monkey. Responses will be recorded anonymously.

First, a background check is to be performed in order to present profile of a typical respondent. This part is to assess the level of proficiency in relation to database technologies. Further questions will be related to specific tools and functions that Oracle makes available to the administrators and developers in its packages.

The summary of information derived from the survey related to each tool is presented as a part of their assessment. The source of this input in the tools evaluation is clearly marked as survey-derived.

Chapter 4 – Results

Oracle Database, as a dominating RDBMS in the market, must be doing something right if it is able to sustain its position with such distance to the competitors. It addresses the complexity of managing data repositories in nowadays competitive economy and shows support to the database administrators by presenting them with a broad package of tools.

Everything comes at a price though, and investment in Oracle, while enabling a comprehensive set of features, usually also consumes a considerable part of a typical company's budget. Having the market leader's support is both a privilege and a responsibility. At one end, such thorough support yields less trouble with managing data repositories. Utilization of the functionality available in the software can benefit all involved in operation within the data system and contribute to successful strategic and vision goals. On the other side, having the possibilities at hand and not taking advantage of them is a serious waste.

This chapter focuses on SQL tuning features that Oracle Database incorporates for administrator's assistance. It presents the tools that will be scrutinized further. Furthermore, it presents questionnaire results and a profile of a typical survey respondent.

SQL Tuning Tools

With growing sizes of data repositories, DBAs and developers face greater challenges in their everyday duties. Oracle is a complex package and incorporates a bundle of functionality implemented with its tools. A following comprehensive list of tools was selected for this study.

EXPLAIN PLAN

EXPLAIN PLAN is a statement allowing presentation of the steps included in the execution plan for a given query. The syntax is as follows:

```
EXPLAIN PLAN FOR
      [SET STATEMENT_ID = 'statement_id']
      [INTO table_name]
      FOR sql_statement
```

Defining `statement_id` allows storing multiple execution plans in the `PLAN_TABLE` (used as default) or the table specified by the `INTO` clause. This non-default table has to adhere to the structure of the `PLAN_TABLE`.

Below a sample EXPLAIN PLAN output along with the SQL query is presented (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013):

```
EXPLAIN PLAN FOR
SELECT e.employee_id, j.job_title, e.salary,
d.department_name
FROM employees e, jobs j, departments d
WHERE e.employee_id < 103
AND e.job_id = j.job_id
AND e.department_id = d.department_id;
```

By querying the PLAN_TABLE we can view the execution plan:

```
--------------------------------------------------------------------------------
| Id | Operation                     | Name        | Rows | Bytes | Cost (%CPU)|
--------------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |             | 3    |   189 |     10 (10)|
|  1 |  NESTED LOOPS                 |             | 3    |   189 |     10 (10)|
|  2 |   NESTED LOOPS                |             | 3    |   141 |      7 (15)|
|* 3 |    TABLE ACCESS FULL          | EMPLOYEES   | 3    |    60 |      4 (25)|
|  4 |    TABLE ACCESS BY INDEX ROWID| JOBS        | 19   |   513 |      2 (50)|
|* 5 |     INDEX UNIQUE SCAN         | JOB_ID_PK   | 1    |       |            |
|  6 |    TABLE ACCESS BY INDEX ROWID| DEPARTMENTS | 27   |   432 |      2 (50)|
|* 7 |     INDEX UNIQUE SCAN         | DEPT_ID_PK  | 1    |       |            |
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
3 - filter("E"."EMPLOYEE_ID"<103)
5 - access("E"."JOB_ID"="J"."JOB_ID")
7 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID"
```

The execution plan is presented as a hierarchy of actions with basic statistics associated to each step. The order of processing each step usually conforms to the depth of indentation – the more indented the step, the earlier in the execution it takes place.

In case of the above query three tables are accessed: `employees`, `departments` and `jobs`. Tables `jobs` and `departments` are accessed by an index, whereas `employees` table uses full scan access method. Because the index scan is a means of accessing the table it is associated with the table and along with it is treated as a single step. All tables have the same level of indentation. This means they will be executed sequentially starting from the topmost element.

The `PLAN_TABLE` (or the user defined table for storing execution plans) contains multiple columns that shed light on how the optimizer aims to process the query. Using this tool is a start for monitoring and enhancing performance, yet it requires the ability to read the plan and knowledge of one's own database design.

In the occasions when a single statement execution plan is required, the use of EXPLAIN PLAN statement is the best tool (Greenwald, R., Stackowiak, R., Stern, J., 2007). When monitoring multiple statements the exercise of issuing EXPLAIN PLAN for each could become a cumbersome task. In such occasions other tools should be selected, such as SQL Trace or Autotrace with EXPLAIN option specified. On the other hand, EXPLAIN PLAN does not execute the query, only generates the plan and stores it in the defined table. Autotrace mode needs to specify not return the result set in order to only generate the explain plan and not to add to the overhead of processing the query.

The use of EXPLAIN PLAN for statements including bind variables might not produce valid execution plans. Because the bind variables are implicitly declared with `VARCHAR2` type, optimizer has to perform datatype conversions. Another problem is that there is no bind variable peeking (Antognini, 2008). This drawback should be considered when monitoring the execution plan in such cases.

DBMS_XPLAN

DBMS_XPLAN, introduced in Oracle Database 9*i* Release 2, is one of the easiest tools used to format and present the explain plan for the following (Oracle, Oracle Database PL/SQL Packages and Types Reference 12c Release 1, 2013):

- `EXPLAIN PLAN` command (extract data from `PLAN_TABLE` or any user defined table for storing execution plans);
- Automatic Workload Repository (AWR);
- SQL Tuning Set;
- SQL Plan Baseline;
- Fixed views, i.e. `V$SQL_PLAN` or `V$SQL_PLAN_STATISTICS_ALL`.

The DBMS_XPLAN package also allows specifying the level of detail for the generated report. Table 3 enlists the options along with their output consequences.

Table 3: Formatting options for DBMS_XPLAN output. From "Oracle Database 11*g* Performance Tuning Receipts: A Problem-Solution Approach" by S. R. Alapati, D. Kuhn and B. Padfield. Copyright 2011 by Apress.

Format Option	BASIC	TYPICAL	SERIAL	ALL	Description
Basic (ID, Operation, Object Name)	X	X	X	X	
ALIAS (Section)				X	Information on object aliases and query block information
BYTES (Column)		X	X	X	Estimated bytes
COST (Column)		X	X	X	Displays optimizer cost

Format Option	BASIC	TYPICAL	SERIAL	ALL	Description
NOTE (Section)		X	X	X	Shows NOTE section of the explain plan
PARALLEL (Detail within plan)		X		X	Shows parallelism information related to the explain plan
PARTITION (Columns)		X	X	X	Displays partition pruning information
PREDICATE (Section)		X	X	X	Shows PREDICATE section of the explain plan
PROJECTION (Section)				X	Shows PROJECTION section of the explain plan
REMOTE (Detail within plan)				X	Shows information for distributed queries
ROWS (Column)		X	X	X	Shows estimated number of rows

DBMS_XPLAN package enables viewing of the execution plans from variety of sources in a few available formats in a quick and easy way. It is a flexible tool that allows defining the level of detail required thus specifying the format. It also enables adding or removing elements of the defined format in a way that suits the user (Alapati, S. R., Kuhn, D., Padfield, B., 2011).

Its flexibility is increased by the functions allowing to present execution plans from multiple sources, such AWR, SQL Plan Baselines or SQL Tuning Sets. It makes the DBMS_XPLAN a tool that could be adapted to various needs in an extremely helpful way. Additionally, it adapts also to specific query, formatting output differently for parallel execution, partition or execution statistics if they are available. Below a sample output with 'advanced' formatting (Antognini, 2008):

```
SQL> SELECT * FROM table(dbms_xplan.display(NULL,NULL,'advanced'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 2966233522
```

```
-------------------------------------------------------------------------------
| Id | Operation          | Name |  Rows |  Bytes | Cost (%CPU)  | Time     |
-------------------------------------------------------------------------------
|  0 | SELECT STATEMENT   |      |     1 |     13 |       3 (0)  | 00:00:01 |
|  1 | SORT AGGREGATE     |      |     1 |     13 |              |          |
|* 2 | TABLE ACCESS FULL| T    |    14 |    182 |       3 (0)  | 00:00:01 |
-------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

       1 - SEL$1
       2 - SEL$1 / T@SEL$1

Outline Data
-------------
   /*+
       BEGIN_OUTLINE_DATA
       FULL(@"SEL$1" "T"@"SEL$1")
       OUTLINE_LEAF(@"SEL$1")
       ALL_ROWS
       OPTIMIZER_FEATURES_ENABLE('10.2.0.3')
       IGNORE_OPTIM_EMBEDDED_HINTS
       END_OUTLINE_DATA
   */

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("N">=6 AND "N"<=19)

Column Projection Information (identified by operation id):
-----------------------------------------------------------

   1 - (#keys=0) COUNT(*)[22]

Note
-----
   - dynamic sampling used for this statement
```

SQLTXPLAIN

SQLTXPLAIN derives its name from SQL Tuning and Explain Plan and is commonly referred to as SQLT. It is a set of packages and scripts created to produce a summary on the condition of the whole installation in relation to SQL performance. It was first developed with the Oracle 9*i* Release 2, and is also available through 10*g* and 11*g* releases.

This tool gathers information from as many sources as possible and formats them into a user friendly, drill-down HTML report that presents a snapshot view of the vital system elements. As it was initially developed to support diagnosis and highlight underperforming

indicators in a week-long period of time, it concentrates on the areas that a typical DBA will consult when dealing with troublesome SQL.

SQLT enables analysing and tuning SQL statements, by simply running the statement and relevant report – with or without the statement execution. Omitting SQL execution is beneficial in cases when the SQL in question already runs for a long time, thus causing it to execute for diagnostics would not be feasible. The report will then contain less accurate statistics, yet it will still shed some light on the causes of the response time values. SQLXTRACT is the report option without execution extracting information from other sources, whereas SQLXECUTE includes the run-time statistics.

Within the SQLT reports information such as the following can be found (Charalambides, 2013):

- global information, such as CBO environment and statistics, DBMS_STATS setup, initialization parameters;
- cursor sharing and bind peeking information;
- SQL Tuning Advisor reports;
- execution plans and statistics, including the history of changes;
- stored outlines, SQL Plan Baselines and SQL Profiles data;
- SQL execution data, including ASH and AWR, as well as parallel processing statistics;
- table statistics and history of changes, as well as related information, such as column, constraint, indexes, histograms and partition data;

- other object information, including metadata and tablespace information.

Figure 4: Sample SQTX Execution Plan

ID	Exec Ord	Operation	Go To	More	Cost[2]	Estim Card	Work Area
0	18	SELECT STATEMENT			256	1	
1	17	SORT AGGREGATE		[+]	256	1	
2	16	. VIEW DBA_OBJECTS			256	68503	
3	15	.. UNION-ALL			260		
4	11	... FILTER		[+]	259		
5	5	.... HASH JOIN		[+]	255	73572	[+]
6	1	....+ INDEX FULL SCAN I_USER2	[+]	[+]	1	93	
7	4	....+ HASH JOIN		[+]	253	73572	[+]
8	2	....+. INDEX FULL SCAN I_USER2	[+]	[+]	1	93	
9	3	....+. TABLE ACCESS FULL OBJ$	[-]	[+]	251	73572	
			Table Columns				
			Col Statistics				
			Stats Versions				
			Column Usage				
			Col Properties				
			Histograms				
			Table				
			Constraints				
			Indexed Cols				
			Indexes				
			Partitions				
			Metadata				
10	7	.... TABLE ACCESS BY INDEX ROWID IND$		[+]	2	1	
11	6	....+ INDEX UNIQUE SCAN I_IND1	[+]	[+]	1	1	
12	10	.... NESTED LOOPS			2	1	
13	8	....+ INDEX FULL SCAN I_USER2	[+]	[+]	1	1	
14	9	....+ INDEX RANGE SCAN I_OBJ4	[+]	[+]	1	1	

Figure 4: Sample execution plan using SQLTXPLAIN scripts. From "Oracle SQL Tuning with Oracle SQLTXPLAIN" by S. Charalambides. Copyright 2013 by Apress.

Figure 4 presents sample execution plan generated by SQLT with a presentation of its interlinks with different areas for deriving information. One of the biggest advantages of the tool is its comprehensiveness and no license requirement. It is simply a script that a talented developer-tuner created to automate his routine checks and was adapted by many. This tool gathers information from many database areas that a proficient DBA or developer would know to

check, yet it performs much quicker and provides a cross-section view of the most important parameters.

On the other hand, with each release, scripts have to be adapted. With the recent Oracle Database 12*c* release, there is currently no SQLT runnable version. As any piece of software, it may also contain bugs.

Given that other tools offering the same level of comprehensiveness require costly investments, SQLTXPLAIN may be a great solution. All in all, it provides a glance on all vital factors that contribute to performance troubles in relation to poorly executing SQL.

Some of the issues that SQLTXPLAIN enables addressing are (Charalambides, 2013):

- identification of tale system statistics and non-default initialization parameters;
- under- and over-estimates of cardinality;
- bind variables, skewed data and histograms relevance and accuracy.

SQL Trace

SQL tracing generates a trace file that encloses performance statistics for executed SQL statements. Each statement is analysed separately and the statistics enclosed in the trace reflect this level of detail. It is also possible to generate execution plans for the selected SQL statements. The following information can be determined by using SQL Trace (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013):

- parse, execute and fetch operations;
- CPU and elapsed time;
- physical and logical reads;
- rows processed count;

- library cache misses;
- user name;
- commit and rollback information;
- wait events;
- execution plan (for closed cursors);
- row count, consistent reads, physical reads and writes, elapsed time (for closed cursors).

This covers a great amount of indicators that allow a holistic view of the process in question. SQL Trace allows either session or instance detail level. It is recommended not to enable tracing at system level, as this can have a disruptive impact on performance (Allen, G., Bryla, B., Kuhn, D., 2009).

Enabling SQL tracing is a straight forward process. All the recording takes place in the background and the only input the user is requested of is to issue enable/disable commands and specify the level of detail (session or instance). The resulting raw trace file is a rich information source of the processes and resource usage and it is a great facility for monitoring and debugging, yet it is difficult to read and requires certain level of expertise. Sample SQL Trace output is presented below (Antognini, 2008):

```
PARSING IN CURSOR #1 len=142 dep=1 uid=28 oct=3 lid=28 tim=1156387084566620
hv=1624534809 ad='6f8a7620'
SELECT CUST_ID, EXTRACT(YEAR FROM TIME_ID), SUM(AMOUNT_SOLD) FROM SH.SALES
WHERE CHANNEL_ID = :B1 GROUP BY CUST_ID, EXTRACT(YEAR FROM TIME_ID)
END OF STMT
PARSE #1:c=0,e=93,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1156387084566617
BINDS #1:
kkscoacd
Bind#0
oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
oacflg=03 fl2=1206001 frm=00 csi=00 siz=24 off=0
kxsbbbfp=2a9721f070 bln=22 avl=02 flg=05
```

```
value=3
EXEC #1:c=1000,e=217,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=1156387084566889
WAIT #1: nam='db file sequential read' ela= 19333 file#=4 block#=211 blocks=1
obj#=10293 tim=1156387084610301
WAIT #1: nam='db file sequential read' ela= 2962 file#=4 block#=219 blocks=1
obj#=10294 tim=1156387084613517
...
...
WAIT #2: nam='SQL*Net message from client' ela= 978 driver id=1413697536
#bytes=1
p3=0 obj#=10320 tim=1156387086475763
STAT #1 id=1 cnt=16348 pid=0 pos=1 obj=0 op='HASH GROUP BY (cr=1720 pr=2588
pw=941
time=1830257 us)'
STAT #1 id=2 cnt=540328 pid=1 pos=1 obj=0 op='PARTITION RANGE ALL PARTITION:
1 28
(cr=1720 pr=1647 pw=0 time=1129471 us)'
STAT #1 id=3 cnt=540328 pid=2 pos=1 obj=10292 op='TABLE ACCESS FULL SALES
PARTITION:
1 28 (cr=1720 pr=1647 pw=0 time=635959 us)'
WAIT #0: nam='SQL*Net message to client' ela= 1 driver id=1413697536 #bytes=1
p3=0
obj#=10320 tim=1156387086475975
```

As with all tracing functionality, caution has to be taken and monitoring of open traces performed as generating trace files may contribute to severe performance problems, especially in relation to CPU and disk space usage (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013).

Some point to more discerning problems, such as identifying location of the trace file and analysing (Harrison, 2001) and (Millsap, 2011). The documentation suggest tagging one's trace files by including statements specific to the program analysed or simply defining a `trace_identifier` variable (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013).

Millsap and Holt (2003) undertook an evaluation of SQL Tracing. Table 4 presents the results. The grading is based on marks in scale of 1 to 10 for each of the assessed categories. SQL trace achieved a score of 61 out of 80. They also convince raw trace files convey great amount of information and should be viewed independently of any formatted output. The

superiority of the trace files over system views (V$views) is pointed out, as the trace file allows greater specialization of the statistics (distinctive for each executed SQL statement), whereas the views present aggregates, which may not be beneficial in tuning at statement level.

Table 4: Evaluation of SQL Trace functionality. Redrawn and adapted from "Optimizing Oracle Performance" by C. Millsap and J. Holt. Copyright 2003 by O'Reilly.

SQL Trace	Points
Ease of getting results now	8
Ease of storing the retrieved data	10
Ease of parsing the retrieved data	7
Minimal invasiveness upon Oracle kernel	7
Minimal invasiveness upon other resources	7
Capacity for historical drill-down analysis	7
Cost to develop tools to assist in analysis	6
Diagnostic reliability	9
Total	61

SQL Trace is a tool that performs a tremendous amount of data recording, which is expensive in terms of used resources, but invaluable when diagnosing performance problem without access to application code, thus simpler tools such as AUTOTRACE cannot be used. In such cases SQL Trace will record all database and OS calls and by filtering the contents of the trace file (i.e. by utilizing *trcsess*) DBA or developer is able to identify executed SQL and perform further diagnosis (Fiorillo, 2012).

trcsess

In multitier, shared server configurations or when dealing with parallel SQL, trace files will most likely be scattered across a few machines, as the client's requests might be dealt with

by multiple servers. Locating and analysing those files could be a cumbersome task. *trcsess* utility was implemented to facilitate these actions. Introduced in Oracle Database 10*g*, *trcsess* utility aids in identification of relevant trace records and allows concatenating them by defining one or more of the following attributes (Alapati, S. R., Kuhn, D., Padfield, B., 2011):

- session ID,
- client ID,
- service,
- action,
- module.

Any combination of the above variables is feasible, thus more detailed reporting is available. It is possible to consolidate multiple trace files without concatenation by omitting the above attributes from the syntax and specifying only the trace files names. *tkprof* can then generate a report using the trace files consolidated by *trcsess* utility.

trcsess is a straight-forward, easy to use command line utility. For simple systems there is not much need for trace file consolidation as the required information should be easily located. The benefits of this utility are best visible in multi-tier, shared server, parallel SQL and multi-session monitoring. *trcsess* consolidates required trace files dependent on the conditions such as Session ID, Client ID, Service, Action, and Module, extracting the important data from multiple trace files and discarding the statistics that the administrator does not need to analyse. This saves a tremendous amount of DBA time and focuses his attention on vital information.

Tkprof (Trace Kernel PROfiler)

Raw trace files are difficult to read without editing and require a level of expertise that DBAs gather through years of practice. The tool that Oracle proposes to ease the process is *tkprof* that is Oracle's most popular profiler. Not only does it present the trace file in a more user friendly format, it is also capable of creating a SQL script that stores statistics of the trace in the database and generating execution plans for enclosed SQL statements (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013).

The syntax for *tkprof* is as follows:

```
tkprof filename1 filename2
[waits=yes|no]
[sort=option]
[print=n]
[aggregate=yes|no]
[insert=filename3]
[sys=yes|no]
[table=schema.table]
[explain=user/password]
[record=filename4]
[width=n]
```

The only required arguments are the input file name (trace file) and output file name. Other options are not obligatory, yet they help extract the required information from the raw trace file. The abundance of options enables to specify narrow area of interest and limit the data to distinctive elements.

Since Oracle 9*i* *tkprof* processes wait events and includes them by default. Wait events can contribute considerably to response times (Millsap, C., Holt, J., 2003). The `SORT` argument can adopt multiple values and arrange the trace information accordingly.

Table 5 presents available options for formatting trace file using *tkprof* Those options take a prefix from among `prs` (parse), `exe` (execute) and `fch` (fetch – only for `SELECT`

statements) and a suffix from the second part of the table. It is also possible to arrange the report on `USERID` sort condition.

Table 5: Sort options of *tkprof* utility. Redrawn from "Oracle SQL High-Performance Tuning" by G. Harrison. Copyright 2001 by the Prentice Hall.

FIRST PART	Description	SECOND HALF	Description
prs	Sort on values during parse calls	cnt	Sort on number of calls
exe	Sort on values during execute calls (equivalent to open cursor for a query)	cpu	Sort on CPU consumption
fch	Sort on values during fetch calls (queries only)	ela	Sort on elapsed time
		dsk	Sort on disk reads
		qry	Sort on consistent reads
		cu	Sort on current reads
		mis	Sort on library cache misses
		row	Sort on rows processed

The `PRINT` option allows shortening the report to the first *n* results. `AGGREGATE` orders whether to separate the values by users of the SQL text. `INSERT` generates the SQL script that stores the trace in the database, whereas `SYS` instructs *tkprof* whether to include the recursive SQL in the report (it is included by default if the script is generated). `TABLE` and `EXPLAN` specify the schema and the name of the table that will be used to temporarily sore the execution plans, as well as credentials that allowing connection to the database in order to generate them.

Tkprof generates a user friendly report of statistics gathered by SQL Trace. A range of parameters define the outcome and enables different perspective of the monitored data,

enhancing DBAs ability to diagnose and resolve potential problems. Sample *tkprof* output is presented below (Antognini, 2008):

```
SELECT CUST_ID, EXTRACT(YEAR FROM TIME_ID), SUM(AMOUNT_SOLD)
FROM SH.SALES
WHERE CHANNEL_ID = :B1
GROUP BY CUST_ID, EXTRACT(YEAR FROM TIME_ID)

call     count       cpu    elapsed       disk      query    current        rows
------- ------ -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      164      1.12       1.90       2588       1720          0       16348
------- ------ -------- ---------- ---------- ---------- ----------  ----------
total      166      1.13       1.90       2588       1720          0       16348

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 28 (SH)  (recursive depth: 1)

Rows     Row Source Operation
------- ---------------------------------------------------
  16348 HASH GROUP BY
 540328  PARTITION RANGE ALL PARTITION: 1 28
 540328    TABLE ACCESS FULL SALES PARTITION: 1 28

Elapsed times include waiting on following events:
Event waited on                             Times  Max. Wait  Total Waited
----------------------------------------   Waited  ----------  ------------
db file sequential read                        30        0.01          0.07
db file scattered read                        225        0.02          0.64
direct path write temp                        941        0.00          0.00
direct path read temp                         941        0.01          0.05
```

Millsap and Holt (2003) express an opinion that *tkprof* reports are sometimes erroneous, especially in the `STAT` line, responsible for presenting vital information about the selected execution plan steps. They also suggest an effort should be made by professionals to be able to read raw trace files, rather than the output generated by *tkprof* Additionally, Antognini (2008) enlists lack of relationship when sorting is used, aggregation and hidden bind variables as the shortages of *tkprof*

Overall, *tkprof* is a tool available in all Oracle releases and enables a decent level of versatility in terms of presented output. Taking its shortages under consideration will make one

aware of any discrepancies or inaccuracies that may occur and allow benefiting from the report regardless.

Autotrace

Autotrace is a SQL*Plus tracing tool. It is a simple feature producing instant results. As with other tools, Autotrace also allow a certain level of differentiation between its reports. The following table enlists available settings enabling Autotrace in selected mode:

Table 6: AUTOTRACE output options. Redrawn from "Oracle Database 11*g* Performance Tuning Receipts: A Problem-Solution Approach" by S. R. Alapati, D. Kuhn, B. Padfield. Copyright 2011 by Apress.

AUTOTRACE Option	Execution Plan Shown	Statistics Shown	Query Executed
AUTOT[RACE] OFF	No	No	Yes
AUTOT[RACE] ON	Yes	Yes	Yes
AUTOT[RACE] ON EXP[LAIN]	Yes	No	Yes
AUTOT[RACE] ON STAT[ISTICS]	No	Yes	Yes
AUTOT[RACE] TRACE[ONLY]	Yes	Yes	Yes, but query output is suppressed
AUTOT[RACE] TRACE[ONLY] EXP[LAIN]	Yes	No	No

The advantage of Autotrace is related mostly to the simplicity of use and interpretation of its output (Kuhn, 2010). It does not require as many steps before viewing the statistics as SQL Trace, which makes it a preferable tool in situations demanding prompt decisions. Additionally, Autotrace presents a set of statistics as described in Table 7.

Its agility allows making tuning decisions and later verifying with more thorough tools, such as the SQL Trace/*tkprof* combination. As Autotrace and SQL Trace/*tkprof* are features presenting similar information, Harrison (2001) performed a comparison between the two. As he points out that no row counts are presented for execution plan steps when using Autotrace. This is a crucial piece of information, as it can easily point to inefficient access paths or join methods.

Table 7: Statistics definitions for Autotrace. Redrawn from "Oracle 10*g*: SQL" by J. Casteel. Copyright 2007 by Cengage Learning.

Statistics	Description
Recursive calls	Number of recursive calls generated at both the user and system level. Oracle maintains tables used for internal processing. When Oracle needs to make a change to these tables, it internally generates a SQL statement, which, in turn, generates a recursive call.
Db block gets	Number of times a CURRENT block was requested.
Consistent gets	Number of times a consistent read was requested for a block.
Physical reads	Total number of data blocks read from disk. This number equals the value of "physical reads direct" plus all reads into buffer cache.
Redo size	Total amount of redo generated in bytes.
Bytes sent via SQL*Net to client	Total number of bytes sent to the client from the foreground to client processes.
Bytes received via SQL*Net from client	Total number of bytes received from the client over Oracle Net.
SQL*Net roundtrips to/from client	Total number of Oracle Net messages sent to and received from the client.
Sorts (memory)	Number of sort operations that were performed completely in memory and did not require any disk writes.
Sorts (disk)	Number of sort operations that required at least on disk write.
Rows processed	Number of rows processed during operation.

An inherent trait of Autotrace is that it only works from within SQL*Plus and it also does not compute CPU and elapsed times and distinguish between parse, execution and fetch operations.

Harrison (2001) enlists simplicity of use and fast results as its strength. When using SQL Trace/*tkprof* the user has to locate the files (as noted above – not always an easy task) and then usually format them for a user friendly output. Autotrace brings the output straight away in a few available formats and is a quick and easy way to monitor changes to SQL statement. He also point that in some system's configuration developers will not have access to trace files located on the servers and Autotrace is a tool that can be used with little privileges.

STATSPACK

STATSPACK is a set of scripts implemented in Oracle 8*i* Release 1 (8.1.6) that supersede similar functionality of `BSTAT/ESTAT` scripts. It does not require any extra license and is available in all editions of Oracle.

It requires manual configuration and installation by which is a simple process initiated by running the *spcreate.sql* script. The script creates `PERFSTAT` user which will own all PL/SQL packages and objects that STATSPACK uses. Later the following scripts are run: *spcusr.sql* creates user and grants privileges, *spctab.sql* creates tables and *spcpkg.sql* creates package (Oracle, Oracle 9i Database Performance Tuning Guide and Reference Release 2, 2002).

The main goal of STATSPACK is to create snapshot of important database statistics in order to monitor and diagnose performance issues. It stores the statistics in the database tables under a unique ID and allows comparing any given snapshots identified by the generated ID, provided that the instance had not been shut down between the start and end times, as the data source for STATSPACK are the dynamic performance views which are emptied at instance

shutdown (Fiorillo, 2012). Similarly to Automatic Workload Repository (AWR) – its successor – STATSPACK calculates delta values for the given metrics.

A *spauto.sql* script allows defining hourly snapshot generation and `DBMS_JOBS` package can be used to specify other time intervals. In order to export all gathered data *spuexp.sql* script is used, *sppurge.sql* deletes specified snapshots, *sptrunc.sql* is used to empty all `PERFSTAT` user's tables and *spdrop.sql* uninstalls STATSPACK (Fiorillo, 2012).

Snapshots are not be automatically purged, therefore monitoring database using STATSPACK requires ensuring sufficient space in its tablespace exists at all times, otherwise the tools stops working. Specifying snapshot level and thresholds alters the amount of data captured. Thresholds define the tolerance of variation from predefined values and only statements that exceed them are captured.

Specifying snapshot level defines the volume, thus system overhead caused by the capture. Each higher level includes all lower level statistics as well as information specific to its own level. Table 8 depicts the levels of snapshots and their descriptions.

Table 8: STATSPACK levels. Redrawn from "Oracle Database 11*g* R2 Performance Tuning Cookbook" by C. Fiorillo. Copyright 2012 by Packt Publishing.

Level	Description
0	General performance statistics
5	Additional data: High resource SQL statements
6	Additional data: SQL plans and SQL plan usage information for high resource usage SQL statements
7	Additional data: Segment level statistics including logical and physical reads, row locks
10	Additional data: Parent and Child latches

STATSPACK includes two text reports – at instance level generated by *spreport.sql* and at SQL level by *sprepsql.sql* script. The former depicts general health of the instance performance and may be used to identify hash value (numerical representation of the SQL text) that can be used to generate the latter which enlists statistics related to specific SQL statement (Powell, 2007).

Sample output is presented below (Allen, G., Bryla, B., Kuhn, D., 2009):

```
SQL ordered by CPU DB/Inst: DW11/DW11 Snaps: 11-14
-> Total DB CPU (s): 107
-> Captured SQL accounts for 246.0% of Total DB CPU
-> SQL reported below exceeded 1.0% of Total DB CPU

    CPU                  CPU per             Elapsd                     Old
  Time (s)  Executions   Exec (s) %Total   Time (s)     Buffer Gets Hash Value
---------- ---------- ---------- ------ ---------- --------------- ----------
    254.95          4      63.74  238.1     249.74          12,811 2873951798
Module: SQL*Plus
select count(*) from dba_indexes, dba_tables
```

STATSPACK gathers a considerable amount of statistics which are best used for general system monitoring and analysis (Powell, 2007). Some of the information that can be found in STATSPACK reporting is (Fiorillo, 2012):

- SQL ordered by specific criteria, i.e. elapsed time, buffer gets or parse calls;
- top 5 wait elements;
- CPU load;
- memory statistics
- initialisation parameters.

SQL Tuning Advisor (STA)

SQL Tuning Advisor, introduced with Oracle Database 10*g* Release 1, is a part of Oracle Tuning Pack which requires purchase of a license, as well as Oracle Diagnostic Pack and is available only in Enterprise Edition. It recommends enhancements to analysed SQL statements, such as statistics gathering, index creation or rewording SQL (Cao, W., Shasha, D., 2013). Additionally, STA may suggest creating SQL Profiles or SQL Plan Baselines (described further).

Figure 5: SQL Tuning Advisor

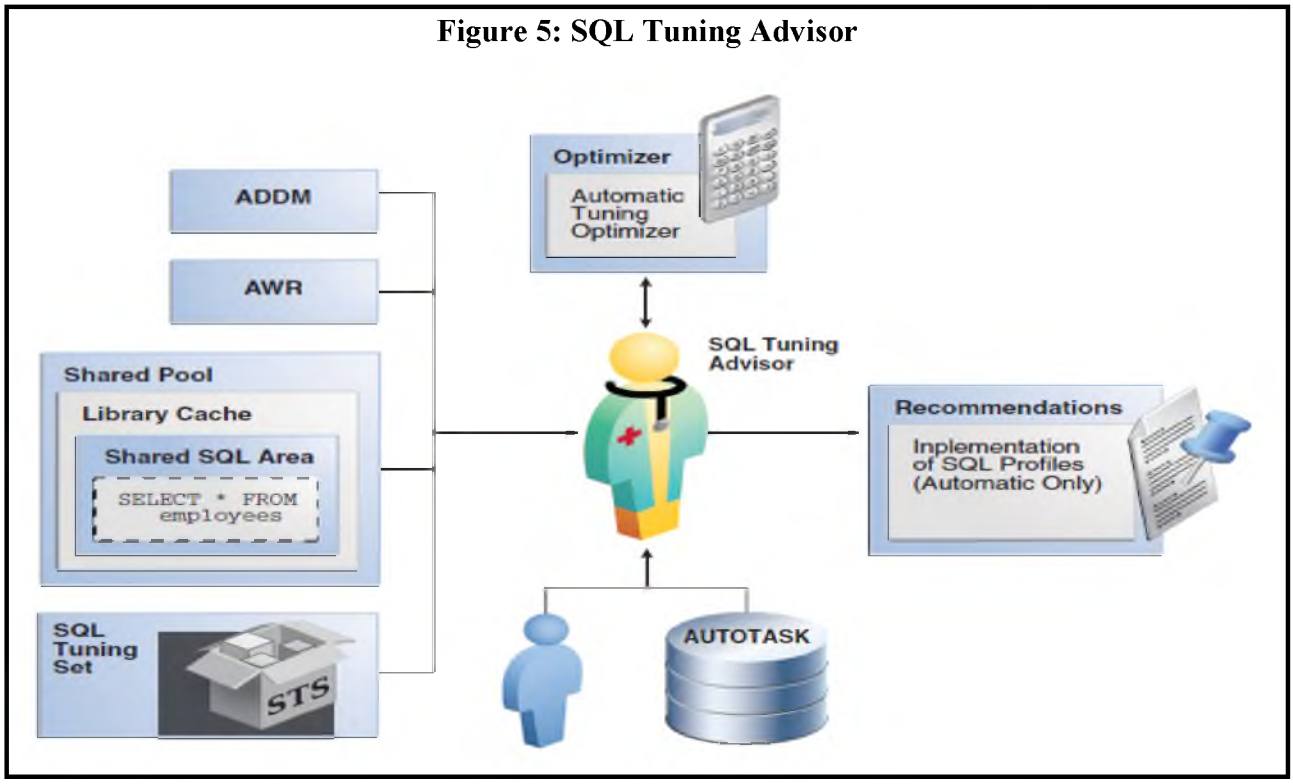

Figure 5: SQL Tuning Advisor architecture. From "Oracle Database SQL Tuning Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

As presented in Figure 5, STA accepts SQL statements from multiple sources, such as ADDM, AWR, shared pool and STS. It can be invoked automatically by maintenance task (`AUTOTASK`) in the maintenance windows or manually when needed, either through Oracle

Enterprise Manager and command line interface with the use of `DBMS_ADVISOR` procedural language package (Dageville B. D., 2004).

Automatic Tuning Optimizer (ATO) performs the analysis and recommendations along with rationale and benefits are presented to the user. Its main benefit is that the ATO is the same optimizer that selects execution plans at run-time but when invoked by the STA it is allowed to perform the calculations for a longer period of time, making its assumptions more accurate (Dageville B. D., 2004).

The Tuning Optimizer will perform the following actions (Oracle, Oracle Tuning Pack for Oracle Database, 2013):

- analyse statistics – check if they are stale or missing;
- verify if creation of SQL profile is beneficial for future statement executions – depending on the setting parameters of the optimizer it can implement the SQL profile without user intervention;
- analyse access paths and determine if different method is beneficial and verify the suggestion by invoking SQL Access Advisor;
- verify structural composition of the statement, to include semantic, syntactic and design construction;
- evaluate the degree of parallelism;
- propose alternative execution plan.

SQL Tuning Advisor, especially when automated and used within maintenance windows, can greatly simplify DBAs tuning efforts. It is also a great diagnostic tool for the pre-implementation stages that will definitely enhance developers' abilities to produce efficient SQL.

Figure 6 presents sample report produced by SQL Tuning Advisor. By clicking highlighted "View" option, details of the selected SQL appear and automatic implementation is possible.

Figure 6: SQL Tuning Advisor Report

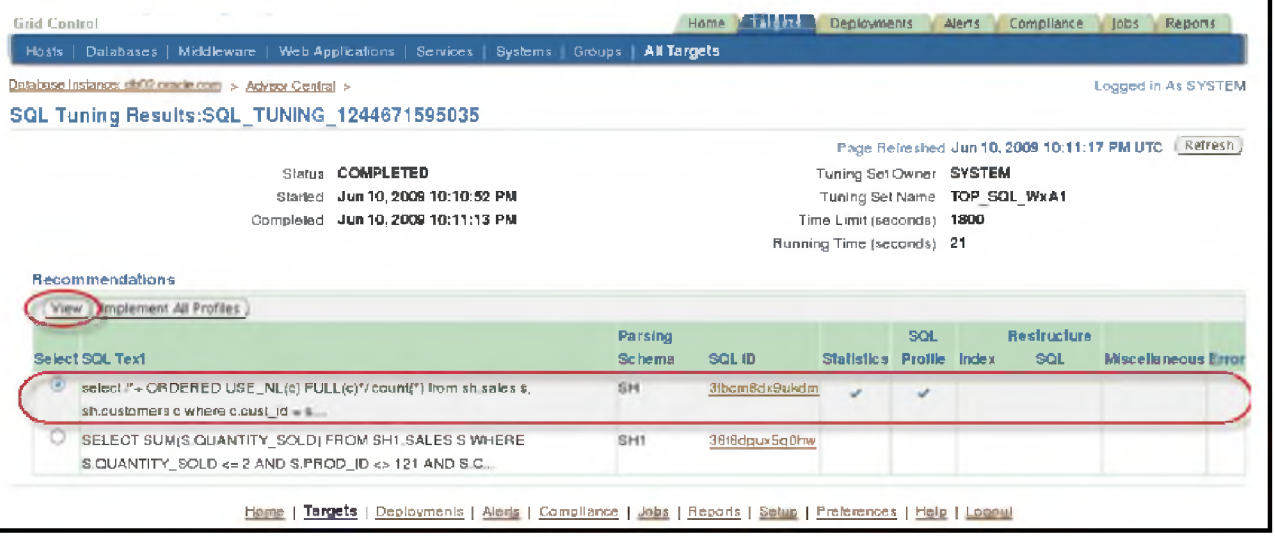

Figure 6: Sample report from SQL Tuning Advisor. From "Oracle Database Performance Diagnostics Tuning Lab". Copyright by Oracle Corporation.

It can assess both single statements as well as SQL sets and other groups of statements, making comprehensive recommendations and – in the automated mode – implement them without users intervention. It makes recommendations for mitigating recognized problems, along with their rationale and the benefit in database time (Dageville, B., Dias, K., 2006).

By specifying an Automated SQL Tuning Task, DBA or developer will limit the amount of SQL Profiles that can be adapted at statement and database level, as well as any relevant time constraints. The SQL Profiles will be implemented for at least threefold performance improvement (Oracle, Oracle Tuning Pack for Oracle Database, 2013), which is a considerable benefit. STA will then produce a report that can be verified by the DBA.

All actions undertaken by the STA can free considerable amount of valuable DBA time, leaving them available to other necessary operations. Dageville et al. (2004) present the impressive tuning benefits achieved by running Automatic SQL Tuning, which are presented in . Table 9: Response time benefits using SQL Tuning Advisor. Redrawn from "Automatic SQL Tuning in Oracle 10*g*" by Dageville et al. Copyright 2004 by VLDB.

	Average Response Time	Maximum Response Time	Cumulative Response Time
No Tuning	817s	5,751s	58,821s
Manual Tuning	30s	275s	2,131s
Auto Tuning	13s	59s	929s

Additionally, it is easy to use and can provide valuable information (Alapati, S. R., Kuhn, D., Padfield, B., 2011). It does not affect the production system as the workload can be defined and moved to an independent machine and analysed by SQL Tuning Advisor. It also stores the workloads and recommendations in the database, allowing to monitor and review the history of change and emerging patterns (Hobbs, n.d.).

SQL Access Advisor

SQL Access Advisor is a part of Oracle Tuning Pack introduced in Oracle Database 10*g* Release 1 version, that analyses database schema in order to produce recommendations on any possible deficiencies in the area of access structures. It requires Enterprise Edition environment and purchase of a license for both the Tuning and Diagnostic Packs. The typically used API is Oracle Enterprise Manager (if available) and a PL/SQL package called `DBMS_ADVISOR`.

As depicted in Figure 7, SQL Access Advisor receives a workload with its context as input from SQL Tuning Set, shared pool or derives hypothetical representation from the schema. It allows applying filter, limiting the workload according to requirements.

Figure 7: SQL Access Advisor

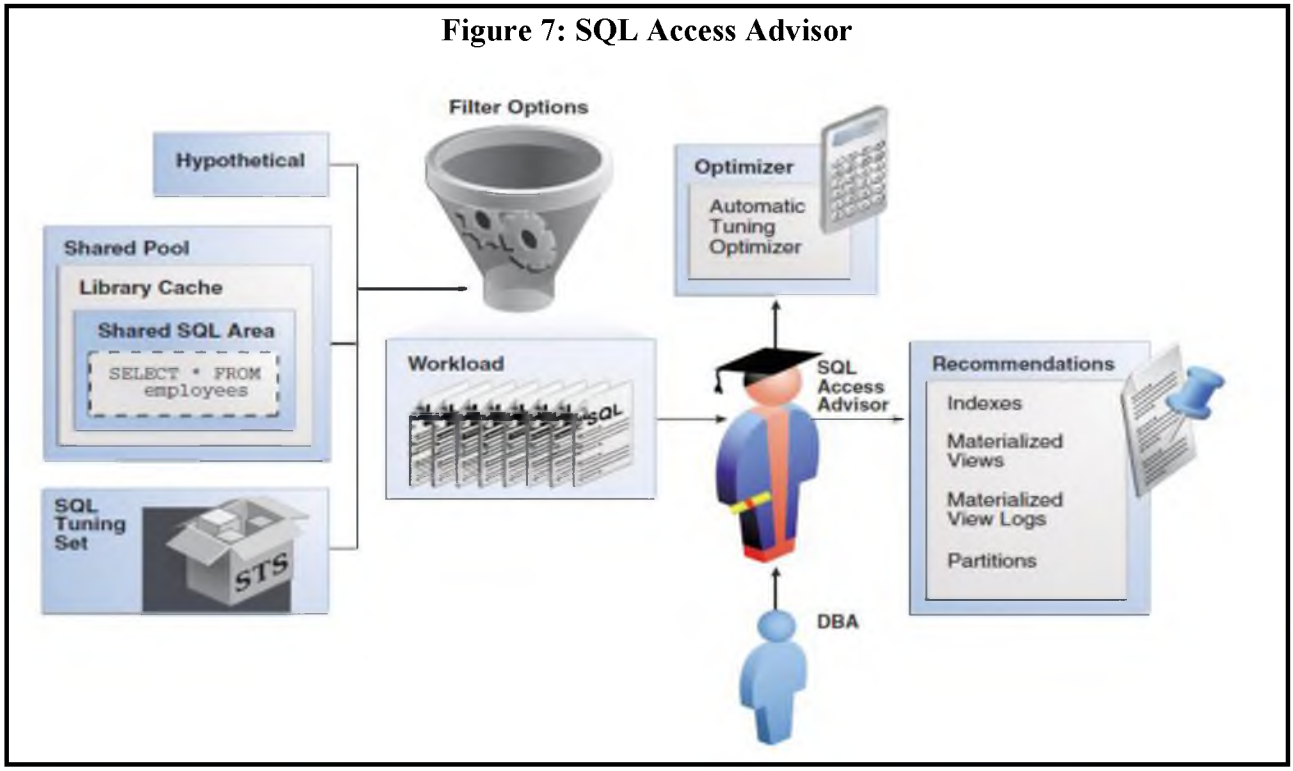

Figure 7: SQL Access Advisor architecture. From "Oracle Database SQL Tuning Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

Subsequently Automatic Tuning Optimizer is invoked and statistics and other context relevant information are analysed so that the findings and recommendations related to the following elements can be presented:

- indexes;
- materialized views;
- materialized view logs;
- partitions.

SQL Access Advisor can, and often will unless otherwise configured, make recommendations related to more than one action, i.e. creating index and partitioning. When revised and accepted, it is important to execute the instructions in their entirety as they are interrelated and omitting a part of the implementation script might not lead to desired outcomes. Therefore even though SQL Access Advisor enables alterations or even ignoring the recommendations, it has to be considered wisely.

Figure 8: SQL Access Advisor Recommendations

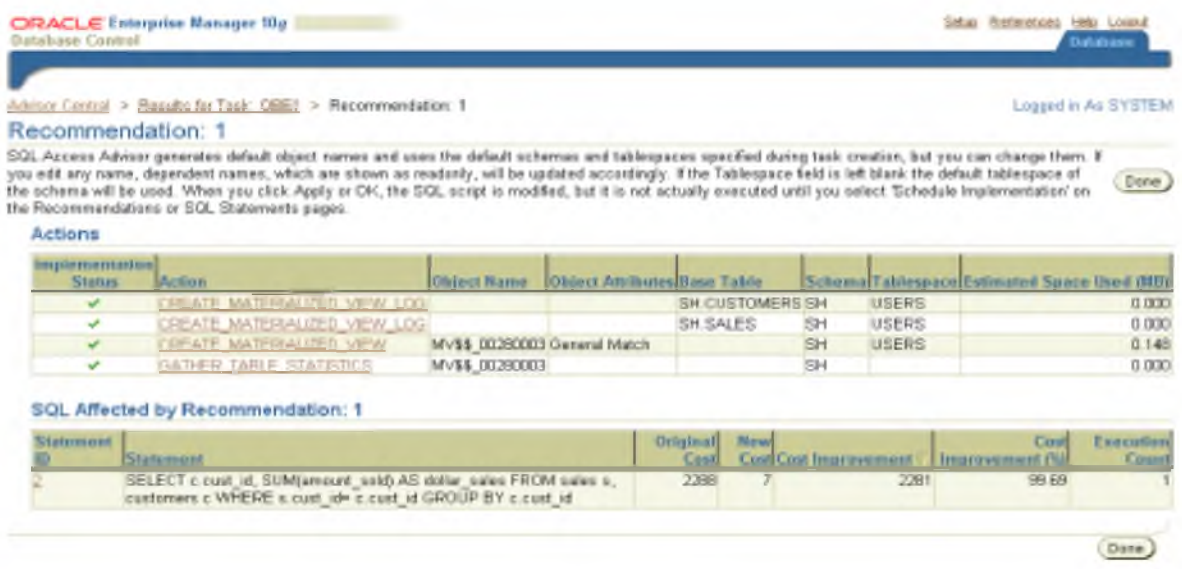

Figure 8: SQL Access Advisor sample recommendations report. From "Performance Tuning using the SQL Access Advisor". Copyright 2006 by Oracle Corporation.

Figure 8 presents the Recommendation section of Oracle Enterprise Manager output for SQL Access Advisor and its related options. Templates are a feature of SQL Access Advisor allowing to determine and save a set of parameters for the task execution and use it in further testing. This is a great facility when workload testing is required under the same set of conditions but at time intervals.

SQL Performance Analyzer (SPA)

SQL Performance Analyzer, introduced in Oracle Database 11*g* Release 1, is a part of Oracle Real Application Testing which requires extra license and Enterprise Edition environment. The typical API used for managing SPA functionality is either Oracle Enterprise Manager (if available) or a dedicated PL/SQL package called `DMBS_SQLPA`.

SPA allows evaluating captured workload on a test system, analysis of benefits and applying changes to the production system. Figure 9 depicts the workflow of the SQL Performance Analyzer.

Figure 9: SQL Performance Analyzer Workflow

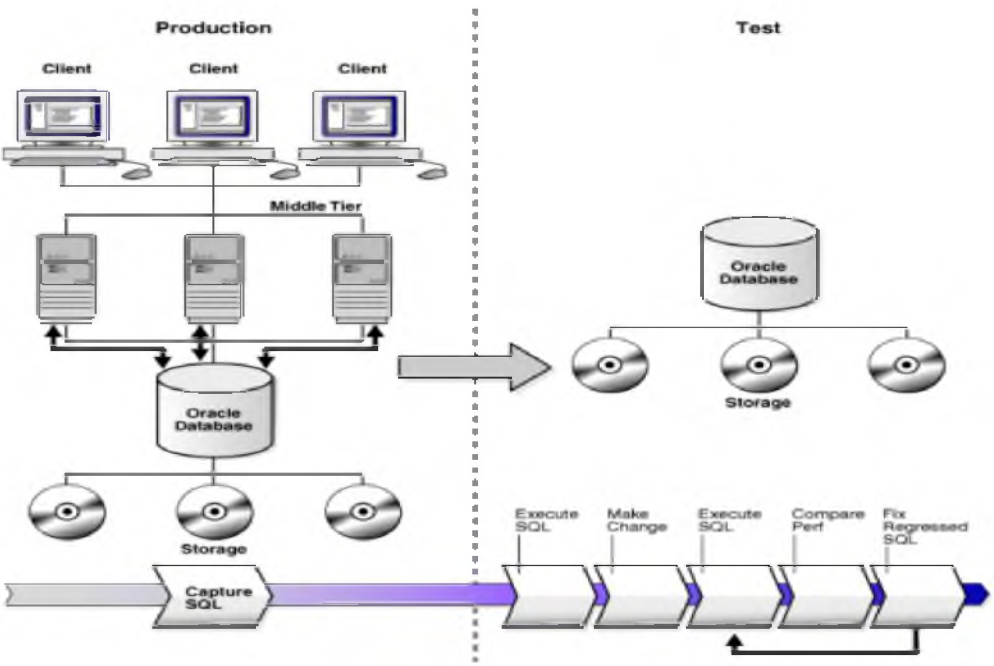

Figure 9: SQL Performance Analyzer Workflow. From "Oracle Database Testing Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

The first element is to capture the workload and, if desired, configure the test environment closely resembling the production system. Next the workload is executed and pre-change SQL Trial data is collected. Then changes are applied, which may relate to platform

upgrade, implementing recommendations from other advisors or new access structures. SQL is rerun as a post-change SQL Trial in order to assess performance impact of the alterations. Further data of the before and after states is compared and the process can be repeated until the accepted performance levels are achieved.

Additionally, it is worth noting that SPA can execute the SQL Trial by invoking the optimizer to generate execution plans only or running the SQL workload which will generate statistics as well (Yagoub, 2008). In the former mode the only comparison metric available is elapsed time, whereas the latter mode provides data sufficient to compare the following statistics:

- CPU time;
- user I/O time;
- buffer gets;
- physical I/O;
- optimizer cost;
- I/O interconnect bytes.

SPA executes each statement in seclusion, one at a time, without ensuring concurrency or particular order, yet it accounts for the number of executions. This establishes a test that can be repeated and each statement is guaranteed statistics related only to its execution. Performance impact can be then assessed at particular SQL statement level and aggregated into total workload interference.

The results of SQL Trials are stored within the database and available for comparison with any other trial. This is a very important feature, as it allows historically comparing executions of the given SQL workload and performing the analysis iteratively.

SQL Performance Analyzer can also be invoked remotely on a foreign system, which is a feature that can support administrators with operations as complex as database upgrades. Figure 10 presents the workflow of such transition.

Figure 10: Oracle 9*i* to Oracle Database 10*g* Release 2 Upgrade Workflow

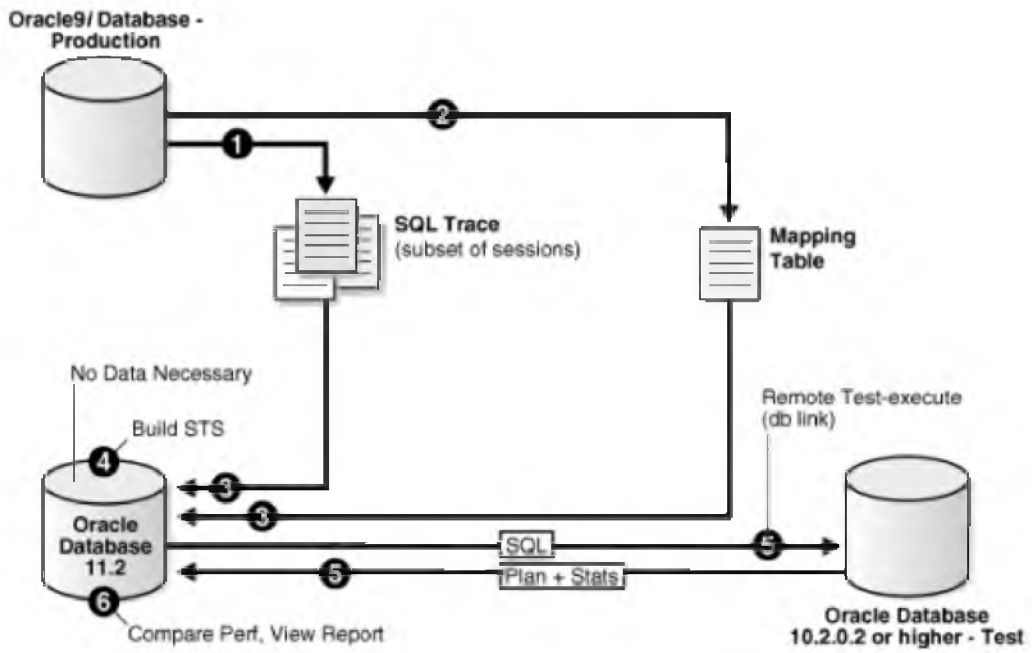

Figure 10: Database upgrade workflow using SQL Performance Analyzer. From "Oracle Database Testing Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

SQL Performance Analyzer's report compares the pre- and post-change executions and statistics and presents the results by highlighting the changes to performance, whether positive or negative. It indicates performance change experienced by SQL statements, contributors to overall workload execution efficiency, as well as impact of a particular SQL statement performance change on the overall workload performance. This comprehensive analysis is presented in a user friendly format, using graphs and presenting the user with recommendations for solutions, i.e. running SQL Tuning Advisor or creating SQL Plan Baseline (Yagoub, 2008).

The report is divided into Summary and Details sections, both of which provide information about performance change impact in different granularity with drill-down capabilities of the graphs and detailed section for SQL statement statistics (Belknap, 2008). Sample SQL Performance Analyzer report is presented in Figure 11.

Figure 11: SQL Performance Analyzer Report

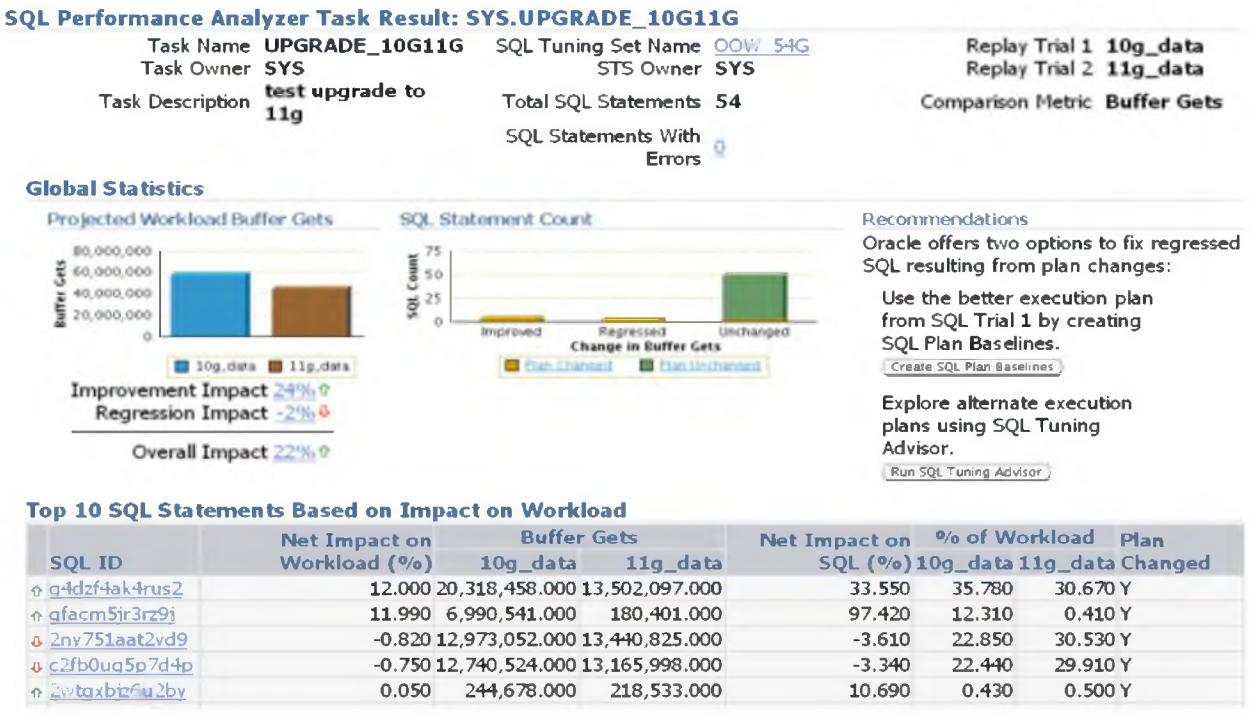

Figure 11: Sample SQL Performance Analyzer report (for transition from 10*g* to 11*g* release). From "SQL Performance Analyzer". Copyright 2007 by Oracle Corporation.

Automatic Workload Repository (AWR)

Automatic Workload Repository is a part of Oracle Diagnostic Pack and the main Oracle Database statistics gathering tool. It was introduced in Oracle Database 10*g* Release 1 and requires Enterprise Edition setting. The APIs used for AWR are Oracle Enterprise Manager or PL/SQL package `DBMS_WORKLOAD_REPOSITORY`.

Automatic Workload Repository creates snapshot representations critical metrics at regular time intervals and compares them to their previous values (Figure 12). Information that AWR reports contain include:

- object statistics;
- time model statistics;
- system and session statistics;
- SQL workload statistics;
- Active Session History (ASH) statistics.

Figure 12: Snapshot Generation with Automatic Workload Repository

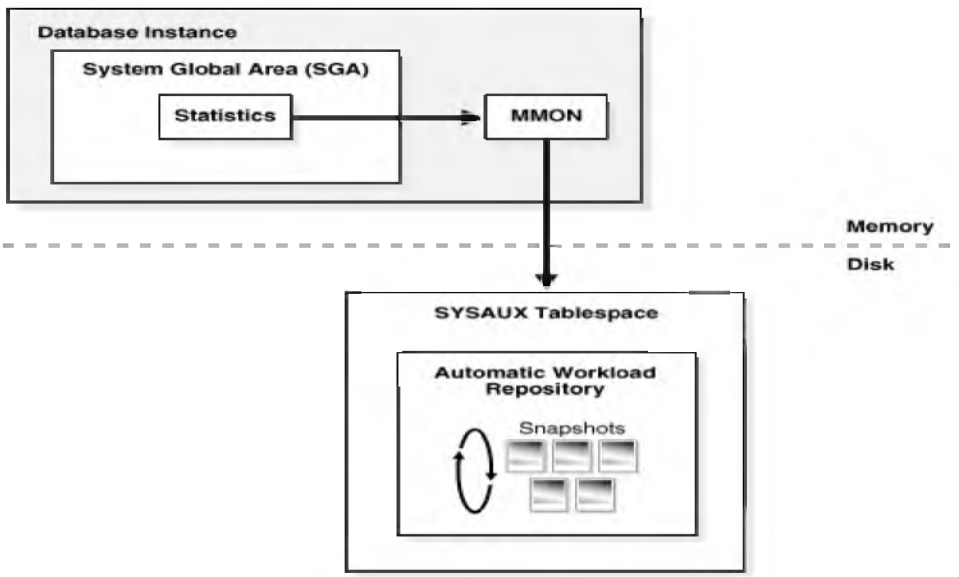

Figure 12: Automatic Workload Repository (AWR) snapshot generation process. From "Oracle Database Concepts 12*c* Release 1". Copyright 2013 by Oracle Corporation.

The default interval for AWR snapshot is 60 minutes and their retention time is 8 days. Some of the snapshots that present good performance levels can be stored indefinitely. These

snapshots are called baselines and used in comparisons with statistics captured at times of poor performance. There are fixed, moving window and template baselines (Fiorillo, 2012).

Automatic Workload Repository is capable of comparing periods of time. The Compare Period Report analyses four snapshots (a set of start and end snapshots bounding each of the periods) and presents the findings in order to identify main contributors of performance variation. For example, Top SQL section of the report presents the main contributors from among the SQL workload depending on metrics, such as CPU time or I/O operations.

Adaptive thresholds are the means of detecting performance problems by highlighting changes to vital metrics over the accepted levels. Since Oracle 11*g* they are able to adjust their values depending on the usage trends calculated in the moving window baseline, minimizing user intervention and enhancing self-tuning abilities of the database (Fiorillo, 2012).

Figure 13: Automatic Workload Repository Sample Report

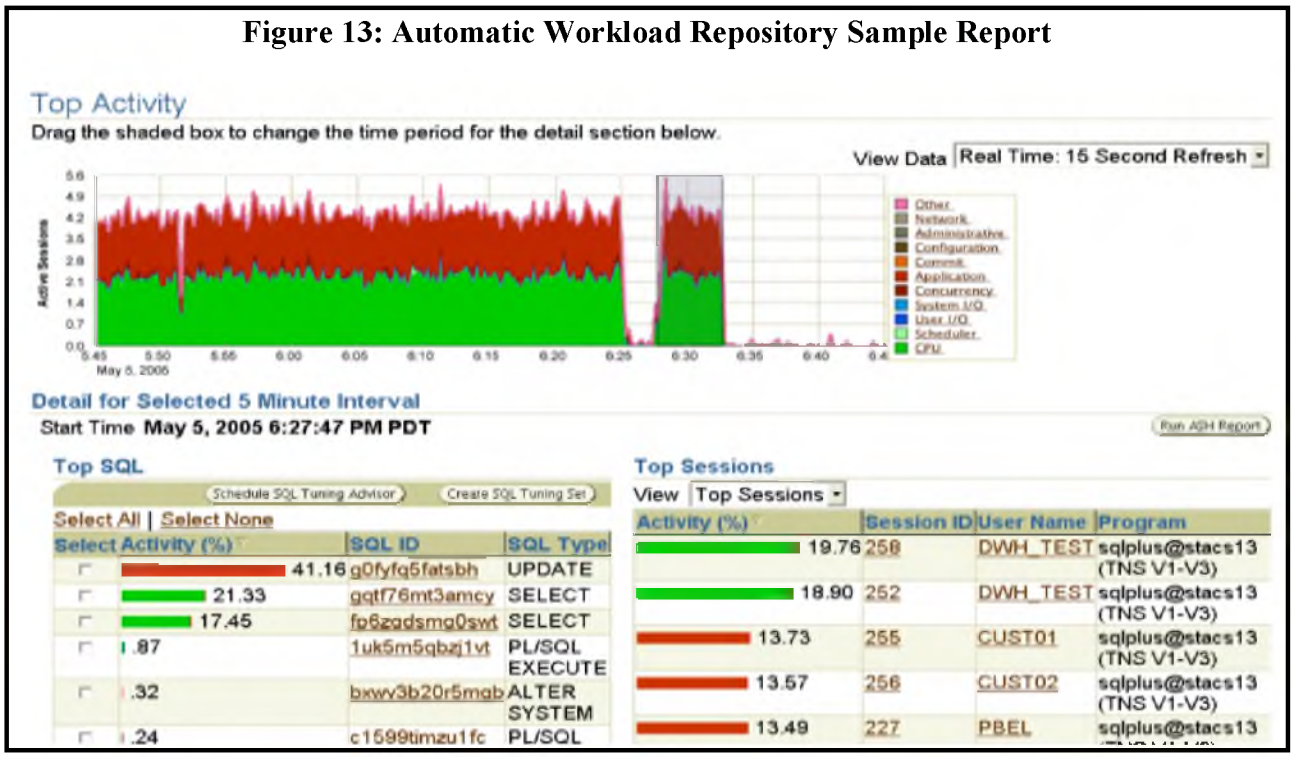

Figure 13: Sample historical analysis using AWR report. From "Oracle Diagnostic Pack – Oracle Data Sheet". Copyright 2008 by Oracle Corporation.

Automatic Workload Repository acts as a central observation and monitoring Oracle feature which records delta values and stores them as chronological snapshots. It is a part of the self-tuning database framework and presents multiple performance indicators over time (Figure 13). (Dageville, B., Dias, K., 2006).

Automatic Database Diagnostic Monitor (ADDM)

One of the tools that rely on AWR monitoring capabilities is the Automatic Database Diagnostic Monitor (ADDM). It was first implemented in Oracle Database 10*g* Release 1 and requires purchase of the Oracle Diagnostics Pack which is available in the Enterprise Edition. The APIs available for ADDM are either Oracle Enterprise Manager or the dedicated PL/SQL package called `DBMS_ADDM`.

It performs analysis of AWR snapshots each hour or at any other user defined interval for AWR activity. The primary objective of ADDM is to monitor the instance operation, notify the administrator of any issues and recommend possible corrective actions. ADDM investigates the symptoms and aims to identify their cause in order to minimize the time database spends processing user request. By introducing this common denominator it simplifies the process of tuning by unifying the impacted measure enabling comparisons between several areas of interest (Dias, 2005). A sample output of ADDM viewed by Oracle Enterprise Manager is presented in Figure 14. It allows drilling down the important information in order to gather more information and understanding of arisen issues.

ADDM also proposes a set of actions that aim to fix discovered issues. Those actions may be alternative paths leading to alleviating encountered performance problems and not necessarily have to be implemented in their entirety (as in SQL Access Advisor). Among others, ADDM will analyse the following areas of database operation:

- CPU bottlenecks;
- I/O issues;
- memory deficiencies;
- high load SQL.

Figure 14: Automatic Database Diagnostic Monitor Graphical Interface

Figure 14: Main ADDM reporting screen in Oracle Enterprise Manager. From "Oracle Database Performance Diagnostics and Tuning Lab". Copyright by Oracle Corporation.

In basic Oracle installations, ADDM will monitor the database at instance level. In RAC (Real Application Clusters) ADDM performs monitoring at database (all instances), single instance or subset of instances levels. In Oracle Database 12*c* Real-Time ADDM was introduced, which helps resolve issues with irresponsive databases without the need to restart the instance.

ADDM acts as a central advisor, invoking other specific advisors thus coordinating tuning activities. Its resulting reports are retained for a month which makes monitoring of

historical performance levels an easier task. Its concentrated efforts aim to minimize the time database spends processing user requests (dbtime). Other objectives include (Dageville, B., Dias, K., 2006):

- providing holistic system view including interactions between components;
- separating symptoms from causes;
- highlighting performance problems as early as their first occurrence;
- adapting to changing technologies and new implementations.

ADDM analyses statistics related to both the time spent processing user request as well as time using various resources, which are the two dimensions represented in Figure 15. ADDM will poll each of the nodes in search for excessive dbtime sources, discarding the elements that do not point to any performance problems, thus the cost of running ADDM is not dependent on the workload, but on the number of issues.

Figure 15: Automatic Database Diagnostic Monitor Nodes of Assessment

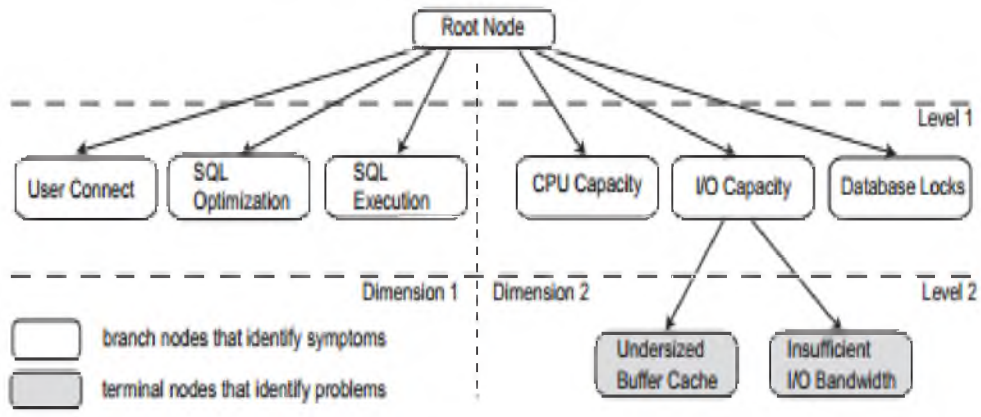

Figure 15: Areas of interest for ADDM. From "Oracle's Self-Tuning Architecture and Solutions" by B. Dageville and K. Dias. Copyright 2006 by IEEE.

Active Session History (ASH)

Active Session History was first implemented in Oracle 10*g* Release 1 and requires Enterprise Edition environment, as well as Oracle Diagnostic Pack license. Its API includes scripts, `V$ACTIVE_SESSION_HISTORY` dynamic view and Oracle Enterprise Manager graphical interface.

Active Session History gathers information about every non-idle session on the database by sampling the activities at a second interval and is always enabled by default. It stores the statistics in the circular buffer of System Global Area. The amount of data that can be collected is limited by the available memory and depends on the database activity, thus in busy environments only a portion of the activity is recorded.

Because some of the performance problems are short-lived they may not be caught by AWR hourly snapshot and therefore will not be included in ADDM report. For those transient problems ASH reports might be of assistance. ASH second-interval samples will have a greater chance to record these processes, signalling to ADDM of possible transient problem (Dageville, B., Dias, K., 2006)

The ASH reports contain the following sections (Oracle, Oracle Database Performance Tuning Guide 12c Release 1, 2013):

- top wait events by origin;
- load profile, i.e. service/module/client originator;
- top SQL, i.e. generating most wait events or row sources;
- top PL/SQL;
- top Java programs;

- top sessions, including blocking sessions and parallel queries;
- top object/files/latches;
- activity over time.

Millsap (2011) points to gaps in recording where the circular buffer runs out of space. This could happen when the workload is intense preventing from capturing the whole problem in the Active Session History recording, leading to incomplete information.

SQL Tuning Sets

SQL Tuning Sets is a database object used to store multiple SQL statements. STS functionality is available Oracle Database 10*g* Release 2 and requires purchase of either Oracle Tuning Pack or Oracle Real Application Testing Pack. It provides PL/SQL API with the use of DBMS_SQLTUNE package or alternatively by using Oracle Enterprise Manager.

Figure 16: SQL Tuning Sets Originators and Utilizers

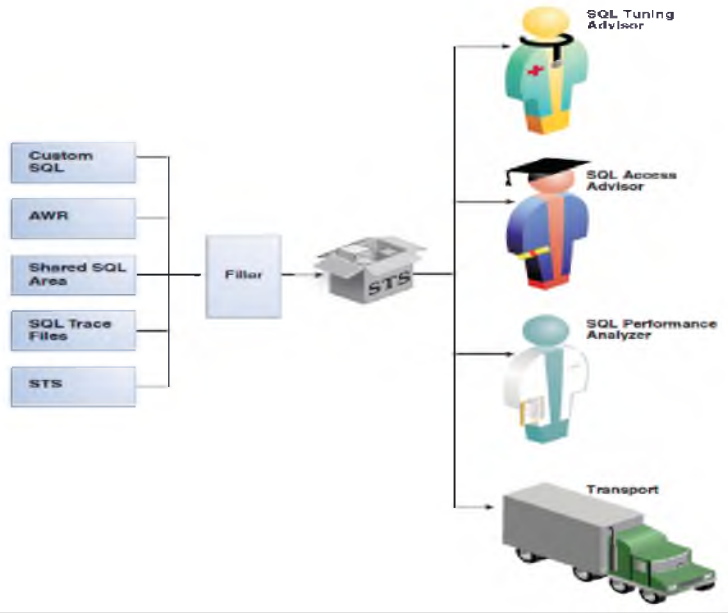

Figure 16: SQL Tuning Sets: generation and utilizers. From "Oracle Database SQL Tuning Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

As presented in Figure 16, the source of the SQL statements include AWR, SGA, trace files, another STS or user defined SQL statements. STS stores the statements with their execution context, statistics and execution plans.

STS can be filtered by module and/or action name, as well as any statistics specified by user, limiting the workload to relevant statements. SQL Tuning Sets are often sourced into Oracle Advisors or used as a convenient way to export workload from one database to another, i.e. from production to test systems.

SQL Profiles

SQL Profiles, introduced in Oracle Database 10*g* Release 1 and they require Enterprise Edition environment, along with the license for both Oracle Diagnostic and Tuning Packs.

SQL Profiles are created by the SQL Tuning Advisor and represent statement specific statistics that enhance optimizer choices. The usual interface for SQL Profiles creation is the Oracle Enterprise Manager and subparts of PL/SQL package called `DBMS_SQLTUNE` can be used as command-line interface.

Typically, statistics are gathered at an object level – table, index, column etc. SQL Profiles analyse the query as a whole and sample all involved elements. Therefore the profile can be adapted and shared by related statements. SQL profiling helps optimizer make informed decisions and produce better cardinality estimates, thus select more optimal execution plans. Three main areas are related to the generation of SQL Profile (Dageville, B., Dias, K., 2006):

- Statistics analysis – check for missing and/or stale statistics is performed;
- Estimates analysis – cardinality and selectivity estimates are validated and, if necessary, compensatory auxiliary information is included in the SQL Profile;

- Optimizer parameter settings analysis – a historical check of previous executions is performed in order to determine the most efficient optimizer settings.

The generation of a SQL Profile does not tie the optimizer to a specific execution plan. It simply provides more data for inspection in order to help determine the best course of action for selected SQL statement. Figure 17 presents profiling role in generating execution plans and possibly contributing to creation of SQL Plan Baselines.

Figure 17: SQL Profiles Environment

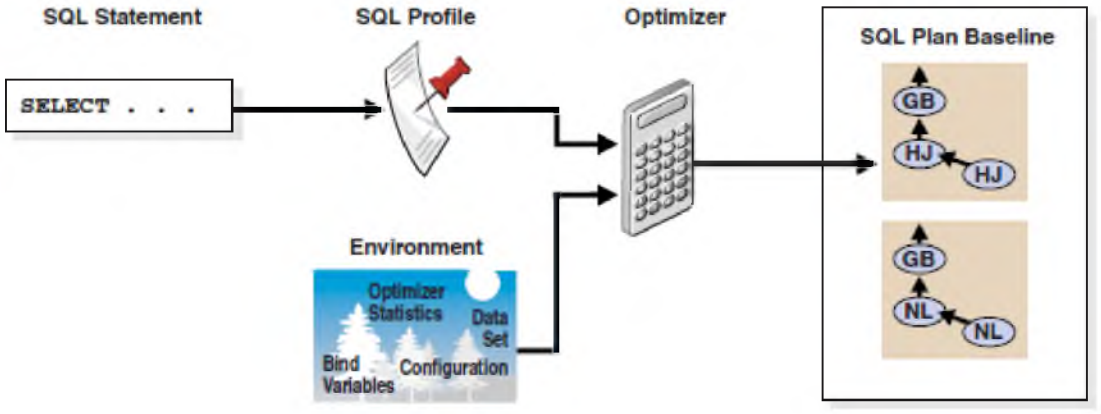

Figure 17: SQL Profile environment. From "Oracle Database SQL Tuning Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

SQL Profiles allow tuning SQL statements without the need of altering the code. This is a huge advantage, as it is a transparent method of improving the choices of the optimizer. (Dageville, B., Dias, K., 2006). Conversely, by being completely transparent to the application code, it is easy to ignore its influence (Antognini, 2008).

Performance benefits may be profound in some cases and hiring Automatic SQL Tuning tool can perform the implementation actions with close to none user intervention (Dageville B. D., 2004). Profiles, as any type of statistics repositories, become stale with changing data. This will lead to less accurate estimates, yet as a part of adaptive database framework the change in

the system will cause the Automatic SQL Tuning Advisor to eventually recognize the need to refresh the stale SQL Profile. It also is a resource intensive solutions, thus its suggested usage should concentrate on crucial and underperforming statements (Dageville B. D., 2004).

SQL Profiles will not be automatically purged when the objects they reference are dropped, which is beneficial in case when updates are performed on those objects. Furthermore, when two SQL statements share signature they might be using the same SQL Profile (Antognini, 2008).

SQL Plan Baselines

In times of radical data changes generated execution plans could change, leading to suboptimal performance. Database upgrade or new application implementation could lead to regressed performance due to new schema objects and statistics. In order to minimize the negative impact of such changes Oracle Database 11*g* Release 1 introduced SQL Plan Baselines (superseding Stored Outlines). It requires Enterprise Edition setting and a purchase of a separate license called Oracle Plan Management. Oracle Enterprise Manager can be used for SQL Plan Baselines management, as well as a dedicated PL/SQL package (`DBMS_SPM`).

SQL Plan Baseline is a set of verified and accepted execution plans for a given SQL statement. It ensures the optimizer does not select a plan other than accepted as a part of the baseline.

SQL Plan Baselines can be generated automatically by SQL Tuning Advisor or loaded manually from SQL Tuning Sets, shared pool, stored outlines or staging table (imported from other database, i.e. test system). Figure 18 shows their place in the process of SQL Plan Baselines creation.

Figure 18: SQL Plan Baselines Origins

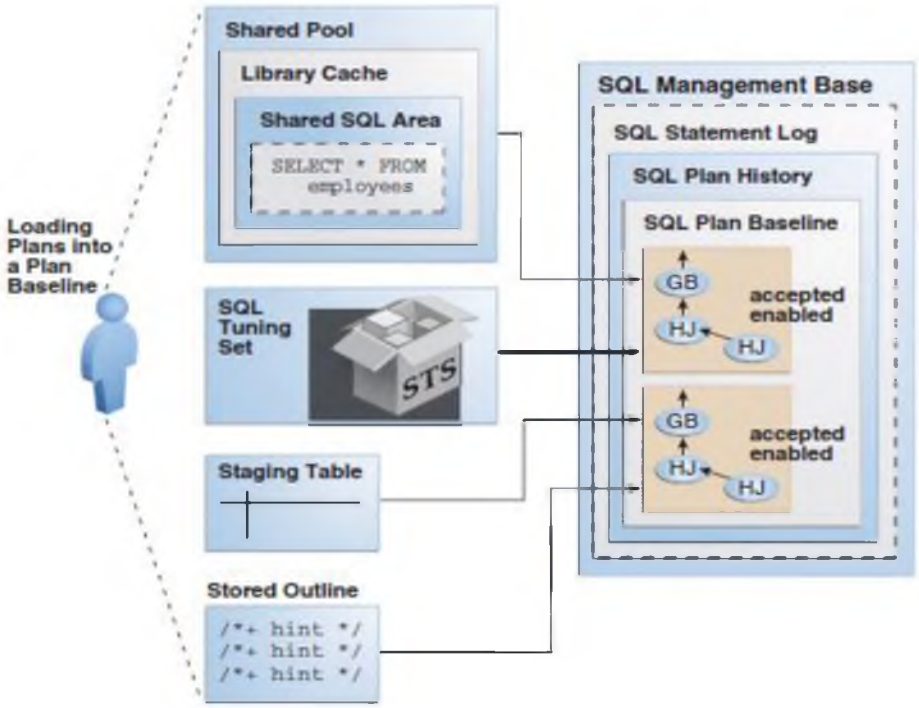

Figure 18: SQL Plan Baseline creation. From "Oracle Database SQL Tuning Guide 12*c* Release 1". Copyright 2013 by Oracle Corporation.

SQL Plan Baselines can evolve over time. When optimizer discovers an execution plan which does not belong to the baseline, it is added to the plan history and marked unaccepted. Subsequently the new plan is verified upon which it is either adopted into the baseline or rejected when not meeting the performance threshold requirements. Execution plans uploaded manually are accepted by default.

One important trait of SQL Plan Baselines is its ability to mitigate inefficient hints. If there is no possibility to alter the application code which contains hint causing optimizer to choose suboptimal execution plan, SQL Plan Baselines will force it to follow one of the accepted plans rather than continue respecting the hint. Additionally, it is important to remember that SQL Plan Baselines are not automatically purged when the objects they refer to are dropped. This is

beneficial for updating and other forms of reorganization as they will not have to be recreated. Also, some SQL statements may share signatures which will cause the optimizer to share the plan baselines between them as well. (Antognini, 2008).

Hints

Hints, introduced as early as Oracle 7, are instructions that are passed through to the optimizer as a SQL statement comments. The syntax for all Hints begins with `/*+` and ends with `*/` or alternatively `--+` although the latter delimiter requires the comment in single line. The hint comment should directly follow the `SELECT`, `UPDATE`, `INSERT`, `DELETE`, or `MERGE` keyword. There are many hint keywords which can be categorized as follows:

- optimizer mode;
- access path;
- join order and method;
- parallel execution hints;
- online application upgrade;
- query transformation hints;
- XML hints.

Hints can also reference single or multiple tables and relate to statement as a whole or differentiate at query block level.

Using hint combinations can enhance the optimizer's decisions yet it can also be dangerous to performance when changes are applied to the system, i.e. when a referenced index

is dropped and other hints are in place the optimizer will be forced to follow any valid suggestions which might be catastrophic (Ziauddin, 2008).

Hints are often inherited with legacy systems and it is difficult to remove them, as usually DBAs and developers will be afraid to make changes in fear of causing regressions. When tuning statements that use Hints, it is recommended to remove all and allow the optimizer to make an independent decision in order to verify if the system changes did not alter the optimizer's choices. This way when data changes we might hope that the optimizer will adapt and select a corrected course of action, rather than follow a hint that was suitable at the moment of reviewing the query's performance, but may not be beneficial in different circumstances. (Charalambides, 2013).

Hints provide a way of inputting data into optimizer's process of developing execution plan. They are needed because other information fed to the optimizer is wrong which cause the optimizer to select suboptimal execution plans. In such cases, experience of the DBA or developer replaces the optimizer's logic pointing to better solutions. It is also useful when exploring alternatives, yet they should not be used as a long-term solution as they deprive the optimizer of its ability to adapt (Antognini, 2008).

Oracle Database 12c Release 1 New Features

Oracle Database 12*c*R1 was introduced while this research was in its completion stage. Since the new features were implemented there is very little that can be found in the literature, thus the new features will only be listed here for reader's reference (Oracle, Oracle Database SQL Tuning Guide 12c Release 1, 2013):

- Adaptive SQL Plan Management: runs automatic evolve task in the default maintenance window in order to verify and accept new execution plans;

- Adaptive Query Optimization – run-time optimizers adjustments; includes:
    - Adaptive plans – optimizer makes a choice of the final plan based on the progress of the SQL execution;
    - Automatic reoptimization – optimizer record a new set of statistics and uses them for future executions of the SQL;
    - SQL plan directives – are stored persistently in the `SYSAUX` tablespace and uses them to obtain more information and chose better performing execution plans;
    - Dynamic statistics – the optimizer decides whether to use dynamic statistics and at what level, as opposed to previously using this feature when an object did not possess its own statistics;
- New Histograms – top frequency and hybrid histograms; if the number of distinct values exceeds 254 (maximum number of buckets for histograms) top frequency option ignores statistically insignificant values and produces more accurate estimates for the most frequent values; hybrid option is an enhanced height-based histogram that ensures separate buckets for a value;
- Statistics – concurrent statistics gathering; `DBMS_STATS` reporting mode and past statistics gathering report; automatic column group creation; session-specific dynamic statistics; bulk load automatic statistics gathering; table-level synopses gathering; stale or locked partition statistics automatic update.

Tools Categories

In order to simplify the process of tools assessment and evaluation categorized. Following is a list of the categories and corresponding tools.

Execution plans

Over the years Oracle adopted the cost based optimizer as its predominant decision maker that replaced the rule based optimizer very fast. Complex algorithms govern optimizer's choices, yet the knowledge, skills and experience of a human mind often lead to more beneficial judgement calls and account for circumstances that the algorithms do not consider.

Execution plans enlist the steps undertaken by the optimizer in order to process the query. They are very informative and DBAs and developers will monitor the decisions of the optimizer in order to ensure performance levels are met and spot areas for improvement. The following elements are the most important in relation to performance:

- order in which the referenced tables are accessed;
- table access paths, that is a decision on how to access data in the most beneficial way for a given SQL query;
- table join methods, that is a decision on what method should be used for joining tables in queries containing more than one table referenced;
- data sort, filter and other presentation and selection related choices.

The following are the explain plan tools described herein:

- EXPLAIN PLAN,
- DBMS_XPLAN,

- SQLTXPLAIN,
- AUTOTRACE,
- SQL Trace/*tkprof*/*trcsess*.

Tracing

In order to fix a problem one must first be able to find it. Tracing allows recording of database and operating system calls and reviewing it afterwards in order to identify the source of excessive workload, such as SQL statement consuming large amounts of CPU time and I/O operations. It is mostly related to the following features:

- SQL Tracing/*trcsess*/*tkprof*
- Autotrace

Decision Support

Oracle facilitates DBAs and developers with a set of automated tools, capable of not only highlighting possible issues, but also suggesting a corrective action. Oracle Tuning Pack was introduced in Oracle Database 10*g* release as a package of new "advisors", automating many daily DBAs duties. The functionality of some of those related to SQL tuning will be described herein:

- SQL Access Advisor;
- SQL Tuning Advisor (STA);
- SQL Performance Analyzer (SPA);
- Automatic Database Diagnostic Monitor (ADDM).

Stability Features

With the growing data repository, changes take place in the system which may interfere with how optimizer executes statements. System and hardware upgrades, new applications being implemented, refreshed statistics or changed initialization parameters are among risk factors. Although these alterations are made in order to improve the system, performance may be temporarily impacted until the changes are fully accepted within the system and necessary adjustments take place. Stability is crucial in these moments of transformation. Oracle proposes some features that will secure the performance levels and help the optimizer chose more efficient execution plans.

- SQL Profiles
- SQL Plan Baselines
- Hints

Diagnostic Features

Oracle Tuning Pack was introduced in Oracle Database 10*g* release as a package of new tools, capable of highlighting an issue with the system and point to the root cause, automating many daily DBAs duties. It is a part of the Oracle's self-tuning and self-managing architecture. The following tools are included in the study:

- Automatic Database Diagnostic Monitor (ADDM);
- Automatic Workload Repository (AWR);
- Active Session History (ASH).

Survey Analysis

The survey was available online for a period of 2 weeks and 42 respondents answered the enclosed questions. This section begins with presenting a typical questionnaire respondent. The level of proficiency in database area is going to be assessed.

Further, evaluation of the SQL tuning tools presented above is undertaken. This part concentrates on calculating analysing the questionnaire's responses by question (top 3 and bottom 3 tools will be presented there) and by tool itself (overall statistics).

Survey Respondent Profile

The first questions of the survey intended to reveal the experience of the respondents in the field of database technologies. The respondents were asked to select the number of years they are professionally dealing with databases. Figure 19 presents the percentile distributions of responses.

Figure 19: Respondents Working Experience with Databases

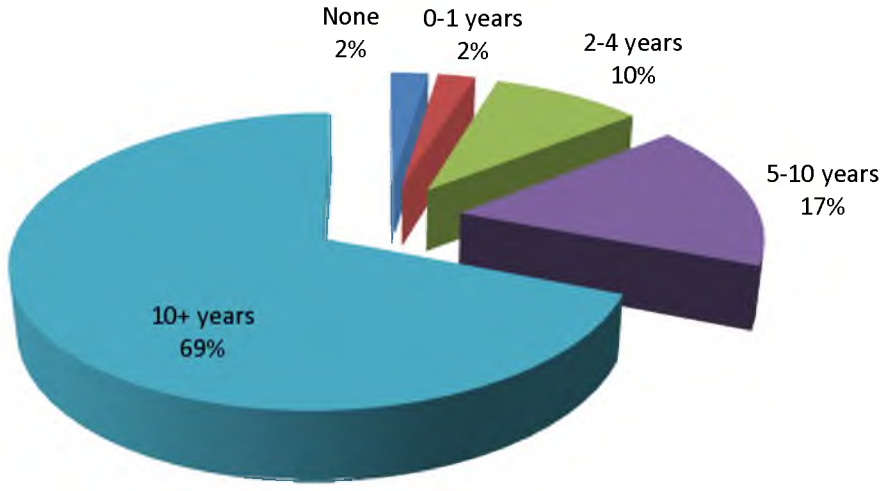

Figure 19: Respondents experience in database industry. Source: researcher's survey.

Further, respondents were to assess their experience administering and using Oracle Database. The distribution of responses is depicted in Figure 20 below.

What comes to the forefront is the considerable experience both in database industry in general, as well as specifically in Oracle. The vast majority of respondents' (69% and 62% of general database and Oracle-specific respectively) experience is more than 10 years. Only 6 of the 42 respondents have less than 5 years' experience in database technology and 28% in Oracle-specific.

Figure 20: Respondents Working Experience with Oracle

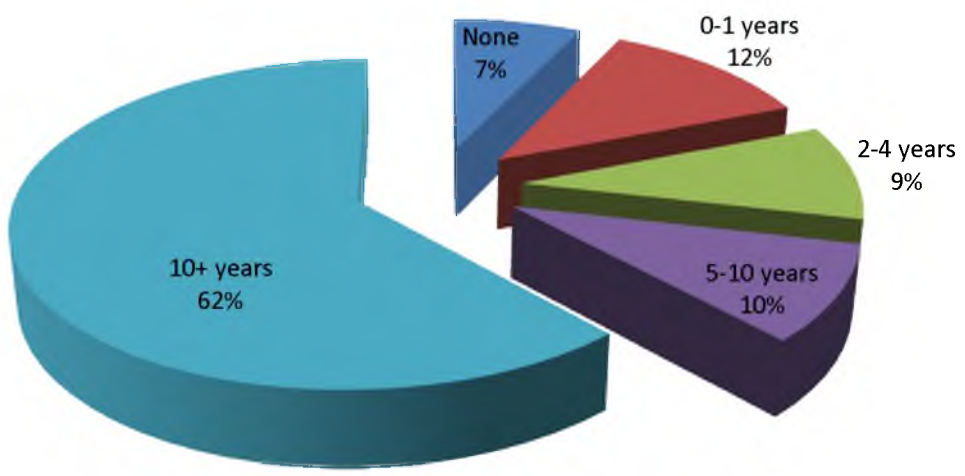

Figure 20: Respondents experience with Oracle Database. Source: researcher's survey.

The respondents were also familiar with other database management systems. Among the most popular were Microsoft SQL Server with 69% of respondents' familiarity, 55% for Microsoft Access and 24% for Sybase. None of the survey participants ever used Ingress. Other DBMSs mentioned in the study where: MySQL, IBM DB2, PostgreSQL, IBM IMS, Synergy/DE and dBASE.

In terms of the size of the database that the respondents were responsible for two measures were used: number of concurrent users and physical database size. When asked about the former, 38% of the survey participants estimated the size of their databases falling into the biggest interval of above 1000 users and the next highest picked response with 21% was the smallest interval of below 50 concurrent users. 19% assessed the size at 201 – 500 users and 14% at 50 to 200 users. The smallest group of only 3 respondents was the middle section with 501 to 1000 users.

In relation to the physical storage of the database, the biggest group of respondents fell into 1 – 10TB interval. Figure 21 present the distribution of other answers.

Figure 21: Respondents Biggest Database Size

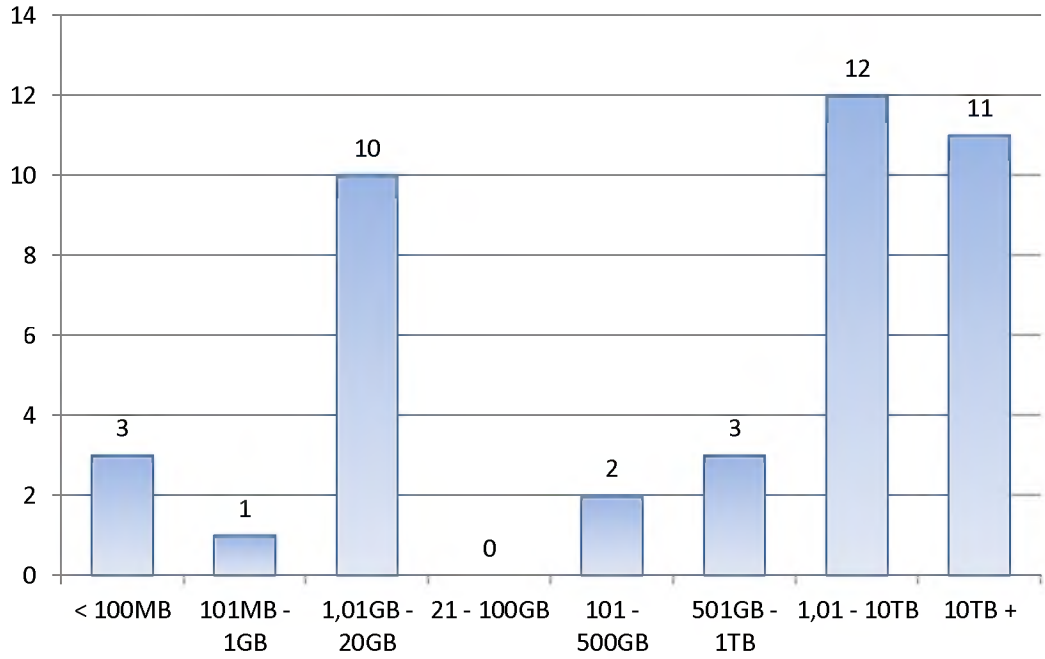

Figure 21: Size of the biggest database survey respondents are responsible for by the physical storage. Source: researcher's survey.

By far the most popular type of database the respondents were administering was OnLine Transaction Processing database with 86% respondents. 57% used Decision Support Database and 45% selected OnLine Analytical Processing type.

This question concludes the general assessment part. As presented above, the sampled population is highly experienced majority of the group assessing their career length at above 10 years in both in general database area and Oracle specific. The participants understanding and knowledge of the field is supported by their familiarity with many other systems. They administer big and medium sized databases, both in terms of number of concurrent users as well as physical storage size. They are also familiar with transaction database model, as well as decision support and analytical types.

Tools Assessment

Questions included in this part of the survey were related to factual knowledge of the Oracle's SQL tuning tools. The following aspects were aimed to be discovered:

- familiarity with the tool;
- factual usage;
- individual score (on a scale of 1 to 10);
- overall benefits in relation to other tools (a total of 100 points to be assigned among tools).

Figure 22 presents the hierarchy in terms of usage (combined regular and occasional). The most widely used tools were EXPLAIN PLAN, SQL Trace/*tkprof* and Hints. Only 9% of the respondents claim to be using SQL Tuning Sets, SQL Profiles and SQL Access Advisor

regularly. 58% of the respondents said they were using SQL Profiles occasionally, which placed this tool in the middle of the scoreboard.

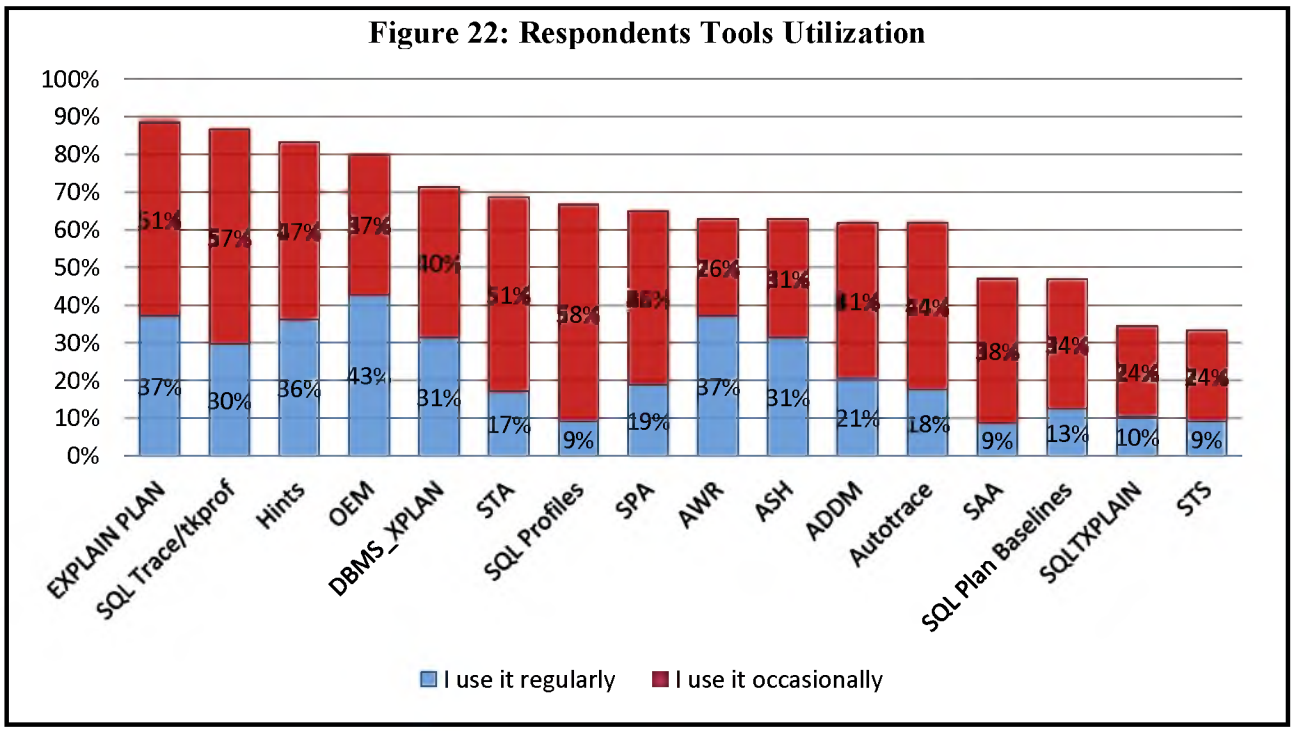

Figure 22: Factual usage of each tool by the survey respondents. Source: researcher's survey.

Figure 23 presents the hierarchy of tools that respondents assessed in terms of individual score on a scale of 1 to 10. This score excluded respondents who did not use the tool. Oracle Enterprise Manager achieved the lead position, just ahead of the SQL Trace/*tkprof* combination. EXPLAIN PLAN and Hints were also assessed at an average of above 6 out of 10. With an average score below 4.0/10 SQL Access Advisor, SQL Plan Baselines, SQL Tuning Sets and SQLTXPLAIN were placed at the bottom of the scoreboard.

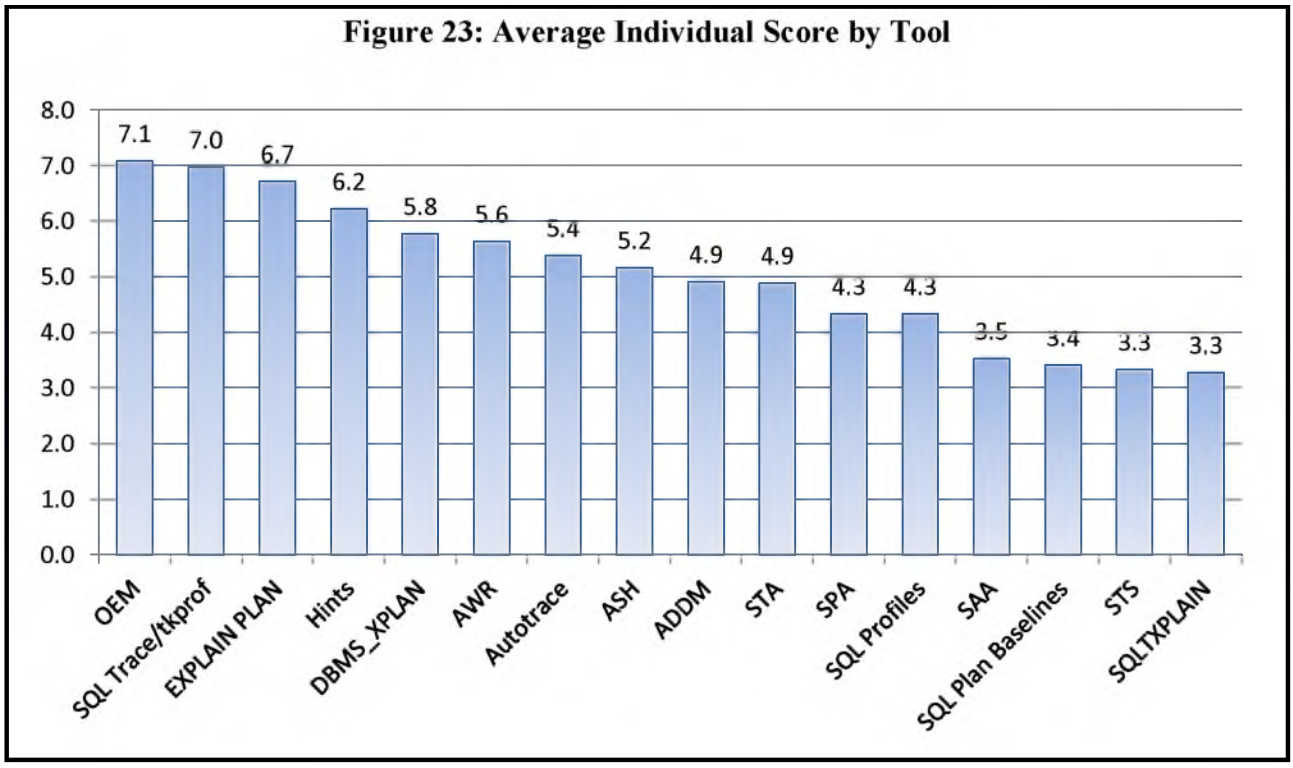

Figure 23: Individual tools score (on a scale of 1 to 10). Source: researcher's survey.

Individual Tools Evaluation

This section of analysis summarizes the survey findings in relation to each tool under assessment. An analysis of each question's results for every tool and aggregate and average results will be presented.

SQL Trace/tkprof

38 out of 42 respondents were familiar with SQL Trace/tkprof, making it the most widely recognized tool. Only 14% do not use this tool, 3% of which never heard of it. With other 86% of respondents using the tool either occasionally or regularly it is second in the category.

Also the average score of 7.0/10 makes it a second best scored. This assessment is supported by the highest percentage of points accumulated with 15% of total awarded and the most respondents (25) deciding to score this particular tool.

Autotrace

More than half (55%) of the respondents recognize the tool. A third of the participants do not use the tool and the average score assigned was 5.4/10. In terms of overall beneficence in tuning Autotrace gained only 4,4%, placing it closer to the bottom of the list.

EXPLAIN PLAN

EXPLAIN PLAN was a statement recognized by 3/4 of the respondents and almost 90% used it in their duties. The average score it acclaimed was 6.7/10 and it managed to accumulate 13,1% of the total bundle of points, making it the third in the category.

DBMS_XPLAN

This PL/SQL package was accounted for exactly 6% of the points awarded by the respondents and was on average scored 5.8 in the 10 point scale. Roughly 7 in 10 respondents use it either regularly or occasionally. DBMS_XPLAN accumulated 6,4% of total points.

SQLTXPLAIN

Only 40% of the survey participants were familiar with the tool, which made it the second least recognized in the group. A third utilized the tool in their tuning tasks with only 3 using it regularly. It was scored at 3.3/10 and accounted for 1,6% of total points, giving it a status of one of the worst assessed tools.

SQL Tuning Sets

The least respondents were aware of STS existence and least used it. It was given a score of 3.3/10 placing it at the bottom of the list *ex aequo* with SQLT. It also accumulated the least amount of points in the overall assessment (1,6%).

SQL Tuning Advisor

STA is known by 7 in 10 respondents and used by almost as many. It got awarded an individual score of 4.9/10 and accumulated 5,6% of total points.

SQL Access Advisor

SAA is one of the least known tools with 43% of recognition in the respondent base, yet 38% use it occasionally. The individual score is set at 3.5/10 and the participation in the points distribution places the tool at the last position with 1,6%.

Automatic Database Diagnostic Monitor

62% of the respondents know ADDM and the same percentage uses it both occasionally and regularly. Its individual scoring of 4.9/10 places it in the middle of the scoreboard. It points participation is set at 4,6%.

Active Session History

Slightly more than half of the respondents recognized ASH. Its individual scoring is set at 5.2/10 and 4,9% of the total point were awarded to ASH. The tools was used by 62% of the respondents, with equal share of those using it regularly and occasionally.

Automatic Workload Repository

AWR is used regularly by 37% of the respondents, which makes it one of the most routinely utilized tools in the bundle, with the same percentage not using the tool at all. Additionally, 1 in 5 participants claim to have never heard of the tool. Yet it individual score of 5.6/10 places it just outside the top-5 and 7,1% of points accumulation makes it a fifth tool in that category.

SQL Performance Analyzer

SPA is one of the more recognized tools with 71%. Almost half of the respondents use it occasionally, but it individual score of 4,3/10 places it at the bottom of the scoreboard. SPA accounts for 6,3% of the points distributed for contributions to tuning strategy.

SQL Profiles

SQL Profiles are familiar to 2/3 of the respondents and the same percentage claims to use them, either regularly or occasionally. This tool's average individual score is equal to SPA's score of 4.3/10 but the total percentage of awarded points is only 2,8%, which is less by 3,5% comparing to SPA.

SQL Plan Baselines

SQL Plan Baselines are placed in the bottom three by familiarity to the respondent group with 43% participants having heard of them. It is one of the lower scoring with 3.4/10 average and 2,9% of total distributed points.

Hints

Hints are used by 83% of the respondents, which places them in the top 3 of the most used tools. Their average score of 6.2/10 and total 7,4% points accumulation places it just outside of top 3 in these categories.

Oracle Enterprise Manager

With 88% recognition and 8 in 10 respondents' utilization, Oracle Enterprise Manager is one of the most popular tools in the set. Its average individual score of 7.1 out of 10 makes it the leader on the scoreboard and in terms of overall points accumulation with 14,7% it is second best assessed tool.

Top and Bottom Scoring Tools

In relation to familiarity with the tools, the top 3 tools are SQL trace/*tkprof* combination known to 90% of the respondents, Oracle Enterprise Manager to 88% and Hints picked by 79%. The least known were SQL Tuning Sets with 38%, SQLT(XPLAIN) with 40% and SQL Access Advisor along with SQL Plan Baselines that only 43% respondents knew.

45% of the respondents never heard of SQLT and never used it. SQL Tuning Sets were never used by 39% of the participants, but they knew of their existence. 58% of respondents used SQL Profiles and 57% used SQL Trace/*tkprof* occasionally. Oracle Enterprise Manager was picked by 43% as used regularly, which made it the top scorer in the category. It also achieved the best individual average score of 7.1/10 from all the tools, followed closely by SQL Trace/*tkprof* combination with 7.0/10 points. SQL Tuning Sets and SQLT were the worst scored on average with 3.3/10 points.

In their last assessment respondents were asked to distribute a total of 100 points among the tools of their choice, based on contribution to their overall tuning strategy. SQL Trace/*tkprof* accounted for exactly 15%, OEM for 14,7% and EXPLAIN PLAN statement for 13,1% of the whole point bundle. With the score below 2% of the points bundle SQLT, SQL Access Advisor and SQL Tuning Sets were the least beneficial in the respondents tuning strategies.

Summary

Chapter 4 – Results presented the tools under assessment, described their functionality and scrutinized available literature. Tools were categorized based on their functionality and area of interest. Furthermore survey was analysed and discoveries were presented in aggregates and averages, along with descriptive analysis of responses. Chapter 5 – Discussions will discuss benefits and disadvantages and will aim to provide a comprehensive overview of each tool.

Chapter 5 – Discussions

There is an abundance of tools and features available with Oracle Database software. With each release new functionality is added, some is replaced – the system evolves. DBAs and developers face more complex and bigger data repositories, along with greater pressure to provide the end users with stable environment and sustainable performance levels.

Tuning SQL is one of the main areas that ensure these requirements are met. Oracle supports the administrators and developers with a subset of its features dedicated to resolving SQL-related issues. This chapter presents the final evaluation of the tools, categorized as previously enlisted for simplicity of assessment.

Execution plans

In order to hide the structure underneath data storage and retrieval from the user, optimizer generates a series of steps that lead to desired result. This poses serious responsibility on the optimizer. Understanding how the DBMS processes queries is a priority. Execution plan is a series of steps that the DBMS undertakes in order to present the user with requested result set or complete action and is one of the first elements inspected when performance problems are encountered.

Table 10 presents the summary of findings and comparison of the tools facilitating generation and interpretation of execution plans. As presented in the summary table, all features are available throughout all versions of Oracle and from quite early releases. They are all free to use and do not require extra license purchased. In terms of ease of use, they range from simple to difficult, with SQL trace being the most complicated to extract execution plans.

Table 10: Summary of execution plan tools in Oracle Database.

	Execution plans				
	Edition (XE/SE/EE)	Version	Ease of Use	Output Readability	Benefits/ Disadvantages
EXPLAIN PLAN	XE/SE/EE	7 and older	simple and straight-forward	needs querying output table	simple; easiest use with DBMS_XPLAN
DBMS_XPLAN	XE/SE/EE	9*i* upwards	simple and straight-forward	good; allows formatting of the output, adding and removing elements; flexible	allows displaying plans from many sources, i.e. AWR, SGA and plan_table; adapts to the specifics of a query
SQLTXPLAIN	XE/SE/EE	9.2 upwards, no 12c developed yet	initially medium difficulty; requires familiarity with the tool	good; interactive, drill down output	allows drilling down each step in order to extract more detailed information; includes history of plan changes; requires installation
AUTOTRACE	XE/SE/EE	8*i* upwards	simple and straight-forward	good; fast output generation; no post processing required	needs to be enabled before SQL execution; easy to interpret
SQL Trace	XE/SE/EE	7 upwards	medium to difficult; enabling the trace is straight-forward but requires post processing or good level of expertise reading raw trace files	poor; requires post processing (i.e. trcsess and tkprof); otherwise it is difficult to read the explain plan from trace files	needs to be enabled before the execution; difficult analysis without formatting; very thorough, detailed, continuous recording; no need to have access to application code

Note. XE = Express Edition, SE – Standard Edition, EE – Enterprise Edition

For execution plans extraction DBMS_XPLAN is the most flexible tool as it allows generating reports from many sources, such as cursors in SGA, AWR report, SQL Tuning Set SQL Plan Baseline or V$ views. `DBMS_XPLAN` has a set of defined formatting that extracts execution plans from `PLAN_TABLE`. It also adapts to the specifics of the query and adds or

removes columns depending on their applicability or upon request, making its output more relevant depending on user's requirements.

SQL Trace requires locating the files and either experience in reading raw trace files or post-formatting. This complexity of the file is both its hindrance and strength. If detailed analysis is needed, i.e. for situations where attempts to identify the root cause of performance problems are made, SQL Trace facility proves an abundant information source. When only basic advice is needed, the identification, concatenation and post-formatting of trace files and the cost associated to their creation may prove not to be the best approach. SQL Trace is the only tool from the group that does not require the administrator to have access to application code as it records all database calls and by using *trcsess* utility allows concatenation of unnecessary content. It makes this tool the only option should such situation occur.

In the cases where fast results are needed and the statements do not need to be executed Autotrace is a very useful tool. Its output does not need any formatting and some settings are available, i.e. for only producing execution plans rather than executing the statements or to show or hide statistics. Autotrace is a great tool when different options are available for the statement and the DBA or developer wants to determine what the optimizer's choices are for each variation. Tuning with Autotrace is often an initial step supported by more thorough monitoring with SQL trace.

One tool that stands out from the group in terms of the ease of use, intuitiveness and the cross-section reporting ability is the SQLTXPLAIN. Its drill-down functionality implemented in HTML report by hyperlinks allows viewing related information from multiple sources, which would otherwise be available as well, yet extracting them as a SQLT report saves a lot of precious DBA's or developer's time. It provides comprehensive view of SQL-related statistics

extracted and grouped in an interactive report that to some extent resembles OEM interface. It is also a free tool available since Oracle9*i* Release 2, yet it is not developed for the most recent Oracle's release – 12*c*. It is not supported by Oracle as part of the tuning and diagnostic infrastructure and its existence is owed solely to a talented developer who wanted to simplify and maximize his own diagnostic efforts in a short time.

Surprisingly, industry professionals are not that impressed with the tool. In their assessment SQLT was one of the least recognized tools and achieved lowest score of 3.3 out of 10. This might be related to the fact that SQLT scripts are not officially supported by Oracle and thus are not as popular as other tools. This is also reflected in the survey – 45% of the respondents never heard of the tools and further 21% were aware of its existence but never used it.

On the other hand, SQL trace/*tkprof* combination was the most popular tool within the respondent group, but the survey questions did not distinguish between categories of use of the tools, which may lead to belief that SQL Trace achieved such scoring thanks to its other attributes, like tracing functionality. Autotrace survey assessment placed it in the middle but only 18% of the respondents claim to have been using it regularly.

Execution plans are the first step each DBAs and developers will consult when they encounter problems with SQL execution. Oracle provides them with many was of viewing them, each with its own advantages and disadvantages. With its drill-down abilities the unpopular SQLT stands out in terms of its agility. Fast results are guaranteed by the use of Autotrace, whereas SQL Trace provides thorough context in the form of extensive statistics. `DBMS_XPLAN` presents execution plans from many sources in a set of predefined and adjustable formats. Each

of the tools has its use in DBAs and developers efforts to produce effective SQL and will be picked upon another depending on the circumstances and severity of addressed situation.

Tracing

Thorough understanding of one's system and its pertaining processes is crucial to maintaining stability and acceptable levels of performance. The biggest advantage of familiarity with tracing facilities is access to linear sequential information that is crucial to effective performance tuning. Tracing statistics will vary, depending on the tools selected, offering different degrees of detail. Tracing with Oracle can thus be adapted to the situation requirements and administrator's needs and preferences. In any case, a presentation of the processes and their resource usage is a start in identifying areas for improvement and addressing root causes of a problem. Below a summary of tracing tools is presented (Table 11).

Table 11: Summary of tracing tools in Oracle Database.

	Trace				
	Edition (XE/SE/EE)	Version	Output Readability	Output Usage	Benefits/ Disadvantages
SQL Trace	XE/SE/EE	8*i* upwards (extended)	poor; requires post processing or good level of expertise	comprehensive, detailed report; good for multiple statement assessment	thorough, detailed; does not require source code; sometimes requires permissions to access trace files
Autotrace	XE/SE/EE	8*i* upwards	very good; instant view	simple assessments and diagnostics; trials before exposing statement to SQL Trace	basic but succinct; requires knowledge of the SQL executed; does not require many privileges and permissions

Note. XE = Express Edition, SE – Standard Edition, EE – Enterprise Edition

A downside of tracing is that it is resource expensive and in busy system it is not possible to have the recording enabled constantly, thus it has to be used sparingly when needed. It is said that tracing can add 5 – 10% to the burden of the database (Burleson, 2011). That points to another disadvantage of the functionality – the administrator has to be able to predict problematic

areas, i.e. newly implemented application, and enable the trace recording accordingly. Another perspective of the problem is related to intermittent performance issues. If it is impossible to assess any specific time or context of the problem, it leads either to guess work in order to pin point the issue or to having the resource expensive trace recording constantly enabled until the issue shows itself (Millsap, C., Holt, J., 2003).

Autotrace traces the execution of SQL and provides instant information on execution plan and statistics. It allows specifying that no execution should take place, which is beneficial in situations when diagnosing statements that execute for a long time. Its greatest strength is the easily readable output generated in a short time. On the other hand its statistics are basic and do not provide as comprehensive view of the kernel's actions as does SQL Trace.

SQL Trace allows a very thorough recording, with all database and operating systems calls. It even allows monitoring of performance of SQL without the knowledge of particular SQL executed. The recording separates each statement as well as different execution actions, i.e. parsing, execution and fetch. When used with *trcsess* utility necessary trimming of data can be performed as well as integrating multiple trace files into a single file so that only crucial information can be extracted.

SQL Trace produces files that are difficult to read, which is the biggest disadvantage of the tool. It requires practice of the DBA or developer to be able to perform the analysis of the raw trace file, but the information contained is often invaluable. Profilers were created to simplify the interpretation but they sometimes produce inaccurate results, i.e. *tkprof* `STAT` line. Efforts have been made to extract information one deems important, thus new profiling tools were developed, i.e. TVD$XTAT, Hotsos Profiles or OraSRP (Antognini, 2008).

Despite the cost and difficulties of use and interpretation, the survey participants assessed SQL Trace/*tkprof* as more popular tool than Autotrace. Its impressive 7.0 out of 10 individual score and 15% of total points assigned among all tools as an award for beneficence to overall tuning strategy place it as a distinctive leader among all other tools. In comparison Autotrace scored 5.4/10 and 4,4% of the points total.

SQL Trace is the best tool for thorough understanding of underlying processes and is the most popular among surveyed population. Clearly the ability to interpret trace files is an invaluable benefit to DBA or developer skillset. The use of Autotrace should not be discouraged as its agility and succinct reporting capabilities can prove to be sufficient or at least a good starting point in simpler situations, producing relevant information in less complicated way. It may also be the only option available when the DBA or developer does not have access to the servers with the trace files.

Decision Support

Evolution is based on one's ability to change and adapt to environment transitions. With today's data environments growing in complexity, it is difficult to manage the repository, tune it and maintain sustainable performance. Oracle facilitates DBAs and developers daily duties by implementing new features that will automate a lot of their duties and adhere to the concept of self-managing database. With the release of Oracle Database 10*g* a set of 'advisors' was implemented that fulfil these tasks. They are enlisted below along with their summary in Table 12.

All of the features require the Enterprise Edition of Oracle and at least 10*g* release, with the exception of SQL Performance Analyzer which was only introduced in Oracle Database 11*g*.

All of the tools in the group have their dedicated PL/SQL package interface as well as they are accessible with the use of Oracle Enterprise Manager.

Table 12: Summary of decision support features in Oracle Database.

Decision support					
	Edition (XE/SE/EE)	Version	Pack/License	API	Benefits/ Disadvantages
SQL Tuning Advisor	EE	10gR1 upwards	Oracle Tuning Pack (requires Diagnostics Pack as a prerequisite)	SQL_ADVISOR + OEM GUI	recommends creation of SQL Profiles and/or SQL Plan Baselines and simplified their implementation; invokes SQL Access Advisor; auto or manual mode; extra license required
SQL Access Advisor	EE	10gR1 upwards	Oracle Tuning Pack (requires Diagnostics Pack as a prerequisite)	SQL_ADVISOR + OEM GUI	recommends creation of new access structures; extra license required
SQL Performance Analyzer	EE	11gR1 upwards	Oracle Diagnostic Pack	DBMS_SQLPA + OEM GUI	allows managing changes; analyses SQL in separation providing accurate statistics but with no concern for the realities of run-time environment; extra license required
Automatic Database Diagnostic Monitor (ADDM)	EE	10gR1 upwards	Oracle Diagnostic Pack	DBMS_ADDM + OEM GUI	points to root cause of a problem; primary objective is to minimize database time; extra license required

Note. XE = Express Edition, SE – Standard Edition, EE – Enterprise Edition, API – Application Programming Interface, OEM GUI – Oracle Enterprise Manager Graphical User Interface

SQL Tuning Advisor analyses given SQL or SQL workload and presents recommendations that lead to more beneficial executions by invoking the same optimizer that is responsible for generating execution plans at run-time but allowing a much longer analysis time. It will recommend statistics gathering, creation of access paths (by invoking SQL Access

Advisor), implementation of SQL Plan Baselines and/or SQL Profiles and analyse if partitioning or parallel execution would be beneficial. When run in the maintenance windows in its automatic mode it can perform system alterations with minimal user intervention.

SQL Access Advisor recommends alterations to the access path structure of the schema. It is most important to either follow its recommendations in their entirety or ignore in whole, as they are interrelated and only partial implementation could likely lead to unexpected performance changes.

SQL Performance Analyzer introduces the concept of implementing changes into system that are well tested and do not pose risk to the system's stability. By executing each statement separately SPA is able to gather all relevant information related to particular piece of SQL, although it deprives its analysis of the run time environment realities. SPA compares different states of the system and highlights any possible issues, making it easier and faster to spot the problems that arise with planned changes and apply corrective actions before it impacts the production system. It can further suggest running SQL Tuning Advisor or adapting SQL Plan Baselines, and in conjunction with Database Replay provides an end-to-end solution to accurate testing (Wang, 2009).

Automatic Database Diagnostic Monitor is reliant on another tool – Automatic Workload Repository – which generates the input for ADDM's analysis. Its main objective is to minimize the time spent processing user requests. It is able to point towards the root cause of a problem with no action required from the DBA and recommend corrective actions. If necessary it will also point to other advisors in order to progress further in the tuning efforts.

All of the decision support tools automate tuning efforts and minimize the time the DBAs and developers need to spend performing cumbersome manual and often time-consuming tuning

actions. They come at a price of the license and in some cases that will prevent the professionals from taking advantage of them. SQL Performance Analyzer and ADDM require only Diagnostic Pack license, whereas SQL Tuning Advisor and SQL Access Advisor need additional Tuning Pack purchased.

It is unfeasible to point towards one over another, as they all relate to different areas of SQL tuning and have their specific benefits of use, as well as disadvantages. It is important to understand the scope of interest for each of these tools whether they are available within one's professional setting or not, as sometimes investment in related license could be well worth it and easily repay in system's stability and users' satisfaction levels.

The survey indicates that the most widely used tool within decision support group was SQL Tuning Advisor with 69% respondents using it, although 5 out of 10 only used it occasionally. Roughly 1 in 5 participants used both SQL Performance Analyzer and ADDM regularly. SQL Access Advisor was the tools that least amount of respondents (47%) used, whether regularly (9%) or only occasionally (38%). The vast majority (around 70%) of the survey participants know all the tools apart from SQL Access Advisor, which was only recognized by 43% of the respondents. The poor assessment of SQL Access Advisor continues in the individual score category, where it is lowest assessed tool in the group with 3.5/10 average. Other tools with score below 5.0/10 also take places in the bottom half of the scoreboard, which is surprising when compared with the functionality they provide.

Despite relieving DBAs and developers from the burden of manual tuning and automating many of their daily tuning tasks, the decision support tools did not impress in terms of their assessment by the polled group of respondents. The requirement of additional licensing may be the factor limiting access to the tools in many cases, although the scoring did not include

the part of the respondents who did not use the tool. A tale of monkey and astronaut by Morton (2008) comes to mind, where the ability to make independent decisions, whether or not the automatic features suggest any action, may be the factor differentiating between professional and independent thinking from simply following sometimes false indicators of good performance.

Stability Features

Nowadays data environments are subject to constant change initiated as demands from company customers or other external pressures leading to internal alterations – schema and parameters changes, system and hardware upgrades. DBAs and developers face pressures to provide users with an efficient system that is going to perform requested actions without any delay. In order to achieve best performance levels they will have to adapt and alter the system. Stability is of crucial importance in those times of transition and sometimes the fact that optimizer is able to adapt to modifications might be a hindrance.

In those times Oracle supports its users with SQL Profiles, SQL Plan Baselines and Hints. All of these tools will influence optimizer's decisions, yet all have different traits and can be used as a remedy for different situations. Summary for all is presented in Table 13.

Table 13: Summary of stability features in Oracle Database.

	Stability					
	Edition (XE/SE/EE)	Version	Pack/License	API	Ease of Use	Benefits/ Disadvantages
SQL Profiles	EE	10gR1 upwards (shared from R2)	Oracle Tuning Pack (requires Diagnostics)	DBMS _SQLTUNE + OEM GUI	automatically generated; acceptance required or can be adapted automatically	resource intensive; become stale; code independent

	Edition (XE/SE/EE)	Version	Pack/License	API	Ease of Use	Benefits/ Disadvantages
SQL Plan Baselines	EE	11gR1 upwards	Oracle Plan Management	DBMS_SPM + OEM GUI	creation and verification can be performed automatically	limit optimizers adaptability to adapt; can mitigate inefficient hints; code independent
Hints	XE/SE/EE	all	no required	N/A	medium; many variations; need monitoring so that they do not cause regressions	use combinations is possible; if one hint is invalid (i.e. index dropped) the other will be respected; requires hard-coding; inflexible

Note. XE = Express Edition, SE – Standard Edition, EE – Enterprise Edition, API – Application Programming Interface, OEM GUI – Oracle Enterprise Manager Graphical User Interface

SQL Plan Baselines functionality is twofold – for one, it prevents from suboptimal plan selection, and two – allows adaptation of new plans when their performance is verified. This is a very beneficial attribute when the user wants to secure the most vital parts of the system. It groups all accepted executions plans for a particular SQL, thus limits the optimizer choices to the accepted subset and prevents from selection of any other plan, even if it is a better performing plan, until it is included in the baseline.

Between SQL Profiles and SQL Plan Baselines flexibility is the trait that distinguished their best usage scenarios. With profiling, optimizer is not forced to perform a particular action. It simply receives additional information that may benefit plan selection, i.e. by impacting cardinality estimates. SQL Profiles can also be shared among related statements if the elements they refer to are also shared.

Neither of the tools requires changing SQL, leaving the code for application logic unchanged. Hints, on the other hand, require coding alterations and, alike SQL Plan Baselines, force the optimizer to make a certain choice, yet Hints leave no room for adjustments. While

SQL Plan Baselines will continue to adapt to changes by verifying and accepting better performing plans, hints have the disadvantage of forcing specific action without leaving optimizer any manoeuvre around it. The only way to change the decision is therefore by changing or removing hints, which inherently causes code alterations.

Availability of the tools in specific releases and versions may sometimes lead to difficult choices and determine solution selection, whether or not entirely beneficial. Hints are the only tool in the group available throughout all releases, and even though they require hard-coding they may be the only available option. SQL Profiles were only implemented in Oracle10g and can be shared among statements from the second release, whereas SQL Plan Baselines were implemented with 11*g* release. They both require Enterprise Edition and extra licensing. None of this is needed to use hints.

The flexibility of these tools determines the benefits and circumstances in which each should be selected. When it is critical to have control over optimizer's decisions, i.e. in times of great transitions such as system upgrades, SQL Plan Baselines should be used. SQL Profiles will come to aid when optimizer generally performs well, making efficient plan selections, but some estimates are not entirely accurate and contribute to slight system performance downgrading. Hints should be used sparingly, only when necessary, as they have the biggest long term impact on the optimizer and are least flexible allowing no adaptation to change for optimizer.

Conversely, the discoveries of the survey prove that Hints are among the more popular tools used in the industry. Their individual score of 6.2/10 placed them in fourth position. They are used, whether regularly or occasionally, by a vast majority of respondents (83% with third position). SQL Plan Baselines are used regularly by only 13% of respondents and SQL Profiles 9%. SQL Plan Baselines are in the bottom of the recognition table with 43% of respondents

being aware of their existence. Individual scoring placed neither SQL Plan Baselines nor SQL Profiles among the best assessed tools with 3.4 and 4.3 out of 10 averages.

Diagnostic Features

The tendency of automatic database actions is also clearly visible in the diagnostic area. With complex data structures and growing pressures identification of the root causes of problems becomes an increasingly difficult task. With the release of Oracle Database 10*g* the Oracle Diagnostic Pack was introduced which requires the Enterprise Edition setting. It contains the following tools: Automatic Database Diagnostic Monitor, Automatic Workload Repository and Active Session History. They are complementary tools and their summary is presented in Table 14.

All the tools are accessible with the use of Oracle Enterprise Manager. ADDM and AWR make the use of PL/SQL dedicated packages and ASH is available with the use of scripts and a V$ view. Because they are treated as a package and their functionality is dependent on one another, they will also be assessed herein as parts of an interrelated toolset created as a comprehensive solution for diagnosis and monitoring.

Table 14: Summary of diagnostic features in Oracle.

	Diagnostics				
	Edition (XE/SE/EE)	Version	Pack/License	API	Benefits/ Disadvantages
ADDM	EE	10gR1 upwards	Oracle Diagnostics Pack	DBMS_ADDM + OEM GUI	combines and analyses the effects of other two tools work; separates symptoms from causes; points to root cause of a problem; points to relieving action or invokes advisor tools; extra license required

	Edition (XE/SE/EE)	Version	Pack/License	API	Benefits/ Disadvantages
AWR	EE	10gR1 upwards	Oracle Diagnostics Pack	DBMS_WORKLOAD_REPOSITORY + OEM GUI	captured snapshots can become baselines for performance; adaptive thresholds differentiate performance requirements depending on usage trends; able to compare periods of time; extra license required
ASH	EE	10gR1 upwards	Oracle Diagnostics Pack	scripts, V$ACTIVE_SESSION_HISTORY + OEM GUI	greater time granularity than AWR recording; able to catch transient problems; identifies top sources by category; limited by memory, gaps in recording; extra license required

Note. XE = Express Edition, SE – Standard Edition, EE – Enterprise Edition, API – Application Programming Interface, OEM GUI – Oracle Enterprise Manager Graphical User Interface

AWR is the central monitoring tool in Oracle Database. It creates snapshot views of vital performance metrics every hour as delta values and stores them chronologically for ADDM's analysis. Creating baselines by selecting reports of good performance for certain time points the system to the anticipated performance levels. Adaptive thresholds allow more flexibility and adaptation to the changing workload, making the AWR reporting a well-deserved part of self-tuning and self-monitoring database architecture.

Snapshot recording takes place every 60 minutes by default and although it is possible to change the intervals and retention periods, Active Session History report were implemented to provide additional input to ADDM at a different level of granularity. ASH gathers statistics with exclusion of idle sessions and stores them in a part of SGA, which is limited by the physical memory available. This may cause incomplete data view yet it provides a more detailed input for ADDM's analysis, which in turn may point to areas that should be further investigated.

Even though SQL Trace is not included in the diagnostic group assessment, comparing its functionality and ASH reports is beneficial in understanding the best setting for utilization of both tools. There are two important functional differences. SQL Trace records a whole period of

kernel's actions (practically 100%), whereas ASH only stores information at a second interval. This is not as thorough information source as the trace file, but it is accessible through viewing reports, which interpret the recording and highlight the most important areas. Trace files require much more instance analysis by the DBA or developer in order to extract vital information.

ASH has also the advantage of being constantly enabled by default. SQL Trace is manually enabled and it is recommended against unnecessary tracing. In situations where a problem occurs, SQL Trace requires reproducing the issue, whereas ASH analysis will usually give enough information to point to problematic areas. Although ASH produces highly usable information, it is only available in the Enterprise Edition and requires additional licensing as opposed to SQL Trace. It is worth noting that STATSPACK, with functionality similar to that of AWR and ASH, is a free tool that was implemented as early as Oracle 8.1.6 and has been available in all releases and versions since and is often used in place of the licensed monitoring tools.

Automatic Database Diagnostic Monitor is the tool that consolidates the reports generated by AWR and ASH. It performs analysis by filtering the areas of database operation that generated most dbtime. The use of a single measure allows comparing different areas of interest. After identifying the root cause ADDM suggests corrective action or invokes specific advisors.

The diagnostics group was slightly better judged by the survey respondents. On average, all the tools were used (whether regularly or occasionally) by 2 in 3 of participating in the survey study. AWR was one of the most widely tools used among the professionals with 37% claiming to have been using it regularly. In comparison ASH were used regularly by 31% and ADDM by 21% of the respondents.

As a group ADDM, AWR and ASH accumulated 16,6% of the total points for the benefits they bring to tuning strategies among the survey respondents. Both AWR and ASH had their average individual score higher than 5 (5.6 and 5.2 respectively) and ADDM at 4.9 out of 10.

System's ability to self-monitor and self-diagnose will grow in importance proportionally to the complexity of the data repository. DBAs and developers will need more assistance in managing database. Oracle Diagnostic Pack provides a solid and comprehensive solution that automates and simplifies the process of database diagnostics.

Summary

Above consolidation provided an intense analysis and comparison of SQL tuning functionality available in Oracle Database through its many releases. The abundance of features required grouping them into categories for easier assessment. Each of the tool strengths and weaknesses were analysed so that the best setting and circumstances for use could be identified. The survey results were also taken into consideration in order to provide more realistic view of the tools.

Chapter 6 – Conclusions

Growing complexities of the data structures caused management of the data repositories to become a difficult and time consuming task. Self-managing and self-tuning capabilities of DBMSs help DBAs and developers fulfil their daily duties.

Although there is a great tendency in the industry towards automated tuning and self-managing database, relying solely on automatic features cannot be the only action towards better performance. Ability to understand, verify and alter functioning of the tools available within the performance support packages is of paramount importance to optimal and sustainable performance levels.

With the abundance of new and established functionality education in the field and staying on top of new features implementations is a must. This study provides an overview and assessment of SQL tuning functionality of Oracle Database releases.

Contributions

This study presented a set of SQL tuning tools available with Oracle Database releases. It described the available tools, including literature critics, and assessed their beneficence by polling a sample of DBAs and developers, who as professionals evaluated each tool in terms of their contribution to their tuning strategy. Tools were categorized and grouped for assessment and circumstances for utilizing each were identified.

This research provides a valuable assessment of SQL tuning functionality with industry leader DBMS – Oracle. It is a point of reference for DBAs and developers, both new, only introduced to the tuning field, as well as those well established.

Future Research

One interesting development that could be monitored further is the incorporation of the new Oracle Database 12*c* features and their benefits in the tuning strategies. Additionally, the survey could be conducted on the bigger sample to present more statistically relevant data, as well as possibly include the new Oracle Database 12*c* features.

The population sample for the survey study contained rather experienced Oracle users, with mostly big databases under their care. It would be interesting to find out what less experienced users with knowledge of smaller databases think about different Oracle tools and how they deem them important and beneficial in their tuning strategy.

When asked about the tools used in their day-to-day tuning tasks, many respondents mentioned third party tuning tools. This might be indicative of the tendencies towards supplementing the functionality within Oracle with other software. Third party tuning tools were out of scope but research of this subject might be complementary to this study.

Additionally, the survey respondents are familiar with other DBMSs. Comparison of SQL tuning tools other DBMS provide could shed more light on what the level of support Oracle presents in contrast to its competitors.

References

Alapati, S. R., Kuhn, D., Padfield, B. (2011). *Oracle Database 11g Performance Tuning Recipes: A Problem-Solution Approach.* NY: Apress.

Allen, G., Bryla, B., Kuhn, D. (2009). *Oracle SQL Recipes: A Problem-Solution Approach.* NY: Apress.

Antognini, C. (2008). *Troubleshooting Oracle Performance.* NY: Apress.

Belknap, P. B. (2008, June 13). Oracle Real Application Testing. *Proceedings of the 1st International Workshop on Testing Database Systems.* NY: ACM.

Bruno, N., Chaudhuri, S. (2002). Exploiting Statistics on Query Expressions for Optimization. *Proceedings of ACM SIGMOD Conference 2002.* ACM.

Burleson. (2011, September 28). *trcsess Tips*. Retrieved August 5, 2013, from Burleson Consulting: http://www.dba-oracle.com/t_trcsess_tips.htm

Cao, W., Shasha, D. (2013). Tuning in Action. *Proceedings of the 16th International Conference on Extending Database Technology* (pp. 737 - 740). NY: ACM.

Casteel, J. (2007). *Oracle 10g SQL.* Boston, MA: Cengage Learning.

Charalambides, S. (2013). *Oracle SQL Tuning with Oracle SQLTXPLAIN.* NY: Apress.

Chaudhuri, S., Narasayya, V. (2007). Self-Tuning Database Systems: A Decade of Progress. *Proceedings of the 33rd International Conference on Very Large Databases*, (pp. 3 - 14).

Chaudhuri, S., Weikum, G. (2005). Foundations of Automated Database Tuning. *Proceedings of the 2005 ACM SIGMOD Interntional Conference on Management of Data* (pp. 964 - 965). NY: ACM.

Codd, E. (1970, June 6). A Relational Model of Data for Large Shared Data Banks. 377-387. NY, USA. Retrieved July 3, 2013, from dl.acm.org: DOI: 10.1145/362384.362685

Connolly, T. M., Begg, C. E. (2010). *Database Systems. A practical Approach to Design, Implementation and Management* (5th ed.). Boston, MA, USA: Pearson.

Dageville, B. D. (2004). Automatic SQL Tuning in Oracle 10g. *Proceedings of the 30th VLDB Conference.* Toronto: VLDB.

Dageville, B. G. (2005). Automatic Diagnosis of Performance Problems in Database Management Systems. *Proceedings of the Second International Conference on Automatic Computing (ICAC)* (pp. 326-327). Washington, DC: IEEE Computing Society.

Dageville, B., Dias, K. (2006). Oracle's Self-Tuning Architecture and Solutions. *Bulleting of the IEEE Computer Society Technical Committee on Data Engineering.* IEEE.

Dias, K. R. (2005). Automatic Performance Diagnosis and Tuning in Oracle. *Proceedings of the 2005 CIDR Conference.* VLDB Endowment.

Fiorillo, C. (2012). *Oracle Database 11g R2 Performance Tuning Cookbook.* Birmingham, UK: Packt Publishing.

Galindo-Legaria, C., Joshi, M., Waas, F., Wu, M. (2003). Statistics on Views. *Proceedings of VLDB 2003.*

Graham, C., Correia, J., Coyle, D., Biscotti, F., Cheung, M., Contu, R., Dharmasthira, Y., Eid, T., Eschinger, Ch., Granetto, B., Hong Swinehart, H., Mertz, Sh., Pang, Ch., Raina, A., Sommer, D., Sood, B., D'Aquila, M, Wurster, L., Zhang, J. (2013). *Market Share: All Software Markets Worldwide 2012.* Gartner, Inc.

Greenwald, R., Stackowiak, R., Stern, J. (2007). *Oracle Essentials. Oracle Database 11g.* Sebastopol, CA: O'Reilly.

Harrison, G. (2001). *Oracle SQL High-Performance Tuning* (2nd. ed.). NJ: Prentice Hall.

Herodotou H., Babu Sh. (2009). Automated SQL Tuning Through Trial and (Sometimes) Error. *Proceedings of the Second International Workshop on Testing Database Systems.* NY: ACM.

Hobbs, L. (n.d.). *Turbocharge Your Database: Use the Oracle Database 10g SQL Access Advisor.* Retrieved August 10, 2013, from http://download.oracle.com/owsf_2003/40150_Hobbs.doc

Kuhn, D. (2010). *Pro Oracle Database 11g Administration.* NY: Apress.

Millsap, C. (2010, November 5). *Thinking Clearly about Performance.* Retrieved July 7, 2013, from http://method-r.com/downloads/cat_view/38-papers

Millsap, C. (2010, September). *Thinking Clearly About Performance.* Retrieved June 27, 2013, from dl.acm.org: DOI: 10.1145/1854039.1854041

Millsap, C. (2011, February 23). Mastering Performance with Oracle Extended SQL Trace. Retrieved July 6, 2013, from http://method-r.com/downloads/cat_view/38-papers

Millsap, C., Holt, J. (2003). *Optimizing Oracle Performance.* Sebastopol, CA, USA: O'Reilly.

Morton, K. (2008, December 19). The Oracle Advisors from a Different Perspective: Are You a Monkey or an Astronaut? Retrieved July 4, 2013, from Method R Corporation: http://method-r.com/downloads/cat_view/38-papers

Oracle.com. (n.d.). Retrieved June 29, 2013, from Oracle.com: http://www.oracle.com/us/corporate/features/number-one-database/index.html

TPC.org. (n.d.). Retrieved July 02, 2013, from http://www.tpc.org/information/about/abouttpc.asp

Oracle. (2002, October). *Oracle 9i Database Performance Tuning Guide and Reference Release 2.* Retrieved August 22, 2013, from http://docs.oracle.com/cd/B10501_01/server.920/a96533.pdf

Oracle. (2006, June). *Performance Tuning Using the SQL Tuning Advisor.* Retrieved July 4, 2013, from http://www.oracle.com/technetwork/database/manageability/twp-manage-tuning-using.pdf

Oracle. (2007). *SQL Performance Analyzer.* Retrieved August 4, 2013, from http://www.oracle.com/technetwork/database/performance/spa-white-paper-ow07-132047.pdf

Oracle. (2010). *Oracle Diagnostic Pack - Oracle Data Sheet.* Retrieved July 4, 2013, from http://www.oracle.com/us/products/enterprise-manager/diagnostic-pack-11g-ds-068465.pdf

Oracle. (2013, March). Oracle Database 2 Day + Performance Tuning Guide 12c Release 1. Retrieved July 4, 2013, from http://www.oracle.com/pls/db121/homepage

Oracle. (2013, June). Oracle Database Concepts 12c Release 1. Retrieved July 8, 2013, from http://www.oracle.com/pls/db121/homepage

Oracle. (2013, June). Oracle Database Performance Tuning Guide 12c Release 1. Retrieved July 8, 2013, from http://www.oracle.com/pls/db121/homepage

Oracle. (2013, June). Oracle Database PL/SQL Packages and Types Reference 12c Release 1. Retrieved August 6, 2013, from http://www.oracle.com/pls/db121/homepage

Oracle. (2013, May). Oracle Database SQL Tuning Guide 12c Release 1. Retrieved July 4, 2013, from http://www.oracle.com/pls/db121/homepage

Oracle. (2013, June). Oracle Database Testing Guide 12c Release 1. Retrieved August 8, 2013, from http://www.oracle.com/pls/db121/homepage

Oracle. (2013). *Oracle Tuning Pack for Oracle Database.* Retrieved August 8, 2013, from http://www.oracle.com/technetwork/database/manageability/ds-tuning-pack-db12c-1964661.pdf

Oracle. (2013, April). SQL*Plus User's Guide and Reference 12c Release 1. Retrieved August 6, 2013, from http://www.oracle.com/pls/db121/homepage

Powell, G. (2007). *Oracle Performance Tuning for 10gR2 (2nd Ed.).* NY, USA: Digital Press.

Rob, P., Coronel, C., Crockett, C. (2008). *Database Systems. Design, Implementation & Management.* Hampshire, UK: Cengage Learning.

Shasha, D. (1996, March). Tuning Databases for High Performance. *28*, 113 - 115. Retrieved July 6, 2013, from dl.acm.org: DOI: 10.1145/234313.234363

Wang, Y. B. (2009). Real Application Testing with Database Replay. *Proceedings of the Second International Workshop on Testing Database Systems.* NY: ACM.

Wiese, D., Rabinovitch, G., Reichert, M., Arenswald, S. (2008). Autonomic Tuning Expert - A Framework for Best-Practice Oriented Autonomic Database Tuning. *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds.* NY: ACM.

Wiese, D., Rabinovitch, G., Reichert, M., Arenswald, S. (2008). Autonomic Tuning Expert - A Framework for Best-Practice Oriented Autonomic Database Tuning. *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds* (pp. 1 - 15). NY: ACM.

Yagoub, K. B. (2008). *Oracle's SQL Performance Analyzer.* Retrieved July 15, 2013, from http://ftp.research.microsoft.com/pub/debull/a08mar/yagoub.pdf

Ziauddin, M. D. (2008). Optimizer Plan Change Management: Improved Stability and Performance in Oracle 11g. *Proceedings of the VLDB Endowment* (pp. 1346 - 1355). Auckland, New Zealand: ACM.

Appendix A - Survey

SQL Performance Tuning Features in Oracle Database Software - Questionnaire

You are being asked to participate in a research study of SQL performance tuning tools in Oracle Database software. It aims to discover how different tools are utilized by the industry professionals. This study is conducted by Katarzyna Dobies of Regis University in Denver, Colorado.

Your experience in the field is highly valued and answers to the following questions are appreciated. The results will be presented within my thesis and aggregates will be analysed to support a discussion in order to broaden understanding of how useful SQL tuning tools are within the industry.

The survey requires you to answer a total of 9 questions, which will take approximately 5-10 minutes.

Participation in this project is completely voluntary. You may change your mind and withdraw at any time with no consequences.

By clicking on the button below you indicate your voluntary agreement to participate in this online survey.

Thank you.

1. What is your working experience with databases?
    a. None
    b. 0-1 years
    c. 2-4 years
    d. 5-10 years

e. 10+ years

2. What is your experience with Oracle?
    a. None
    b. 0-1 years
    c. 2-4 years
    d. 5-10 years
    e. 10+ years

3. What other database management systems are you familiar with?
    a. Microsoft SQL Server
    b. Microsoft Access
    c. Sybase
    d. Ingress
    e. Informix
    f. Other:

4. What is the size of the biggest database you have been responsible for:
    a. By physical disk storage required
        i. <100MB
        ii. 101MB – 1GB
        iii. 1,01GB – 20GB
        iv. 21GB – 100GB

v. 101GB – 500GB

vi. 501GB – 1TB

vii. 1,01TB – 10TB

viii. >10TB

b. By number of concurrent users

i. <50

ii. 50-200

iii. 201-500

iv. 501-1000

v. 1001+

5. Are you administering a database of the following characteristics:

a. Transactional (OLTP)

b. Data Warehouse (DSS)

c. Online Analytical Processing (OLAP/BI)

6. Have you heard of any of the following tools:

a. SQL trace/tkprof

b. Autotrace

c. EXPLAIN PLAN FOR statement

d. DBMS_XPLAN package

e. SQLTXPLAIN

f. SQL Tuning Sets (STS)

g. SQL Tuning Advisor

h. SQL Access Advisor

i. Automatic Database Diagnostic Monitor (ADDM)

j. Active Session History (ASH)

k. Automatic Workload Repository (AWR)

l. SQL Performance Analyzer

m. SQL Profiles

n. SQL Plan Baselines

o. Hints

p. Oracle Enterprise Manager

q. Other:

7. How would you describe your familiarity with the following tools (matrix question)?

a. SQL trace/tkprof

b. Autotrace

c. EXPLAIN PLAN FOR statement

d. DBMS_XPLAN package

e. SQLTXPLAIN

f. SQL Tuning Sets (STS)

g. SQL Tuning Advisor

h. SQL Access Advisor

i. Automatic Database Diagnostic Monitor (ADDM)

j. Active Session History (ASH)

k. Automatic Workload Repository (AWR)

l. SQL Performance Analyzer

m. SQL Profiles

n. SQL Plan Baselines

o. Hints

p. Oracle Enterprise Manager

For each tool choose one of the following answers:

i. I use it regularly

ii. I use it occasionally

iii. I heard of it, but never used it

iv. I never heard of it

8. On a scale of 1 to 10 (with 1 being extremely unbeneficial and 10 being extremely beneficial) how do you assess each of the following tools in terms of improvement offered in performance:

a. SQL trace/tkprof

b. Autotrace

c. EXPLAIN PLAN FOR statement

d. DBMS_XPLAN package

e. SQLTXPLAIN

f. SQL Tuning Sets (STS)

g. SQL Tuning Advisor

h. SQL Access Advisor

i. Automatic Database Diagnostic Monitor (ADDM)

j. Active Session History (ASH)

k. Automatic Workload Repository (AWR)

l. SQL Performance Analyzer

m. SQL Profiles

n. SQL Plan Baselines

o. Hints

p. Oracle Enterprise Manager

9. Distribute a 100 points among the following tools depending on its helpfulness in your overall performance tuning strategy (you have a **total** of 100 points to assign):

a. SQL trace/tkprof

b. Autotrace

c. EXPLAIN PLAN FOR statement

d. DBMS_XPLAN package

e. SQLTXPLAIN

f. SQL Tuning Sets (STS)

g. SQL Tuning Advisor

h. SQL Access Advisor

i. Automatic Database Diagnostic Monitor (ADDM)

j. Active Session History (ASH)

k. Automatic Workload Repository (AWR)

l. SQL Performance Analyzer

m. SQL Profiles

n. SQL Plan Baselines

o. Hints

p. Oracle Enterprise Manager