MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN
SULEYMAN DEMIREL UNIVERSITY

UDK 004.658

KALIDOLDA YERKULAN

Development and research of the data processing system in the automated enterprise management system.

Specialty: "6M070400 – Computing systems and software"

Academic Degree: Master of Computer Sciences

"Admitted to defense":
Dean of Faculty
Assist. Prof., PhD Zhaparov M.
«___»_____20__ г.

Head of Department — Assist. Prof.,PhD. Alimanova M.U.

Scientific advisor — Prof., PhD. Aitchanov B.H.

Kaskelen, 2018

АҢДАТПА

Бұл жұмыстың негізгі зерттеута қырыбы Oracle деректер қорының құрылымы және архитектурасы, соның ішінде деректер қорын басқару жүйесінің (ДҚБЖ) өнімділігі туралы ақпаратты сақтау және алудың тиімді жолдарын зерттеу болып табылады.

Магистрлік диссертацияның негізгі міндеті - SGA (System Global Area) негізгі жадынан деректер қорының өнімділік көрсеткіштері туралы ақпаратты алып Oracle ДҚБЖ үшін төменгі деңгейлі мониторинг құралын құрастырып шығу болып табылады.

Мониторинг жүйесі деректер қорындағы көптеген нысандарды бақылауға мүмкіндік береді. Бұл жұмыс Oracle ДҚБЖ негізгі объектілеріне мониторинг жүргізу мен қатар осы нысандар туралы ақпарат алудың әртүрлі әдістерін қарастыратын болады. Атап айтқанда, пайдаланушы сеанстарына бақылау жасау және пайдаланушы сеанс идентификаторлары, орындалатын сұраулар және күту процесінің идентификаторлары сияқты метрикаларды алумүмкіндігі қарастырылады. Сонымен қатар Oracle ДҚБЖ стандарттық мониторинг құралына талдау жасап оның артықшылығы мен кемшілігін зерттейтін боламыз.

Жұмыстың ғылыми жаңалығы – қазіргі таңда SGA жадынан деректер қоры құрылымын динамикалық түрде анықтап осы ақпарат негізінде ДҚБЖ – нің дәл қазіргі сәттегі өнімділік көрсеткішін анықтай алатын бірде – бір құрал болмауында.

АННОТАЦИЯ

Объектом изучения этой работы является архитектура и структура экземпляра базы данных Oracle, в то время как предметом исследования является поиск и принцип хранения показателей производительности экземпляра базы данных для получения информации о производительности СУБД.

Основная задача магистерской диссертации - разработать метод низкоуровневого мониторинга СУБД Oracle на основе методов получения показателей производительности из общей памяти SGA(System Global Area) экземпляра базы данных Oracle и получения структуры хранения для этих показателей с помощью дальнейшее внедрение программного продукта или модуля в существующее программное обеспечение для мониторинга.

Системы мониторинга позволяют отслеживать огромное количество объектов базы данных. В этой работе мы обсудим основные объекты мониторинга, а также различные способы получения данных об этих объектах в СУБД Oracle. В частности, мы рассмотрим возможность мониторинга пользовательских сеансов и получения показателей, таких как идентификаторы процессов, запускаемых пользователем, идентификаторы сеанса, исполняемые запросы и ожидания. Мы рассмотрим основные стандартные системы мониторинга для СУБД Oracle, определим их преимущества и недостатки.

Научная новизна работы заключается в том, что на сегодняшний день нет единого внедренного метода для считывания показателей производительности, который может динамически определять структуру хранения разделяемой памяти SGA и на основе этих данных получать показатели для текущей производительности СУБД.

ABSTRACT

The object of the study of this work is the architecture and structure of the Oracle database instance, while the subject of the study is the finding and the principle of storing the performance metrics of the database instance to obtain information about the performance of the DBMS.

The main goal of the master's thesis will be to develop a method for low-level monitoring of the Oracle DBMS based on the techniques for obtaining performance metrics from the SGA (System Global Area) shared memory of an Oracle database instance and obtaining the storage structure for these metrics, with further implementation of the software product or module to the existing monitoring software.

Monitoring systems allow you to track a huge number of database objects. In this work we will discuss the main monitoring objects, as well as various ways of obtaining data on these objects in the Oracle DBMS. In particular, we will consider the possibility of monitoring user sessions and obtaining metrics such as the identifiers of the processes started by the user, session identifiers, executable queries and waiting events. We will review the basic standard monitoring systems for Oracle DBMS, identify their advantages and disadvantages.

The scientific novelty of the work is that, to date, there is not one implemented method for reading the performance metrics, which can dynamically determine the storage structure of the SGA shared memory and on the basis of this data, obtain metrics for the current performance of the DBMS.

CONTENTS

INTRODUCTION

Today, there are quite a few large firms and manufacturing enterprises operating in different fields of business, science, training, and the provision of services that use databases in their work and production. At a certain point in time, enterprise development, databases become one of the key elements of the enterprise infrastructure. The databases focus on key information, the availability of which affects the successful operation of most units that constantly access databases to obtain the required information. And the faster the database will be able to provide the necessary information, the more valuable this information will be for solving the set business problem.

For the enterprise, the performance and fault tolerance of the DBMS are very important. To get good performance of the database, we need make a preliminary analysis of how we are going to use our database. The performance of the database is determined by a number of indicators (metrics). The admissible and optimal values of these metrics for each database will be are individual (each database has its own baseline), they will be depend on the mode of operation of this database.

The monitoring system is extremely necessary when dealing with large databases used in business. The introduction of this system can reduce the company's losses caused by reduced productivity, downtime or inefficient use of resources. Using monitoring tools, we can:

- Assess the workload generated by user activity and internal processes;
- Define the basic performance line, the deviation from which will reveal anomalies;
- Timely identify performance problems and their sources.

For monitoring it is very important to determine those key metrics, monitoring of which will allow us to quickly draw conclusions about the "health" of our system. The collection of statistics on these metrics during the normal operation of the system will allow us to form that basic performance line, the deviations from which will signal the presence of anomalies in the operation of our system. To ensure the collection of key statistical information and build a basic performance database, the most important metrics are the response time and bandwidth of the database. The availability of statistical information will allow us to analyze in detail the life cycle of

the system, plan its modernization, and evaluate the changes that are made to the code of the software that interacts with the database. After setting up and commissioning, the monitoring system helps to detect faults as quickly as possible and to investigate the causes of their occurrence. It becomes an indispensable tool in the maintenance of the system, allowing us to quickly and accurately collect the required metrics and performance indicators.

Today, quite a few different databases are used in the work, such as Firebird, MySQL, Interbase, IBM DB2, Informix, MS SQL Server, Sibas Adaptive Server Enterprise, PostgreSQL, Red, etc. One of the most used at the moment can be considered Oracle DBMS. As of the beginning of 2018, Oracle occupies a leading position in the market for commercial DBMS [1]. Installations with this DBMS are found in most large enterprises around the world. Since this DBMS is the most common, we will try to take a deeper look at aspects of the system's work under heavy loads, how to obtain database performance statistics and how monitoring can affect the operation of the database itself.

The practical significance of the work is that based on this method it is possible to implement a software module that will be able to obtain data in an abnormal situation of the DBMS operation, and also will enable to increase the speed of polling performance data without creating overhead in DBMS performance and exclude the Heisenberg uncertainty effect, when monitoring itself can affect the reliability of the data received from the monitoring object.

To achieve this goal of the work, it will be necessary to solve the specified tasks:

- Review existing monitoring solutions and analyze existing methods used in monitoring;
- Examine the storage structure for performance metrics within an Oracle instance;
- Get access to metrics by avoiding using an instance of Oracle;
- Compare the load on the DBMS and the performance of reading metrics through standard tools and through the method of reading data from the SGA;
- Justify the advantages of this method;
- Implementation of the program / module for reading data from the SGA.

1. METHODS OF OBTAINING PERFORMANCE METRICS IN DATABASE

To get operational information about database performance, we have to query it, generating queries that generate additional load on the database (processing queue and runtime). If we have a "powerful" system that has free processor and disk resources, these one-off delays in processing the monitoring request can be neglected. What if we do not have these free resources, and statistics need to be received in real time, and with a high degree of intensity, what should we do then? Will not we have the situation when all the capacities of the database server will be spent on monitoring itself? It is possible, because every poll of a database instance is the generation of a SQL query that creates a load on the system. But at the same time, the information we are interested in is already present in the shared SGA RAM. So why do not we throw out all the additional overheads and do not receive the information that interests us directly?

When working with databases, to which numerous parallel client connections are made, we always have a struggle for the use of common resources. In a situation where all the capacities are allocated for the execution of a resource-intensive query, all subsequent requests can fall into the waiting queue. The processing of the following requests will not start until the previous query is completed. When working with any database, there are problems of competition that stem from queuing problems. "Systems in which, on the one hand, there are massive requests (requirements) for the performance of some types of services, and on the other hand, these requests are satisfied, are called queuing systems (QMS)." [2]. In all computer applications there are initiators of requests, something requiring, and information providers, providing a response to such requests. All Oracle performance analysis is dedicated to the relationship between suppliers and consumers, especially in the face of intense competition for shared resources.

The queuing model is visually represented using a sequence diagram. To illustrate the sequence diagram in the queuing model, we will consider an example with the dependence of the response time of the database on the user's request, in which the disk subsystem is used to read data from the database and the processor, to process the request. In the real situation, the number of factors that affect the response time can be much greater.

A sequence diagram is a form of representing the structure of the time consuming of a user operation traveling through different levels of the technological stack.

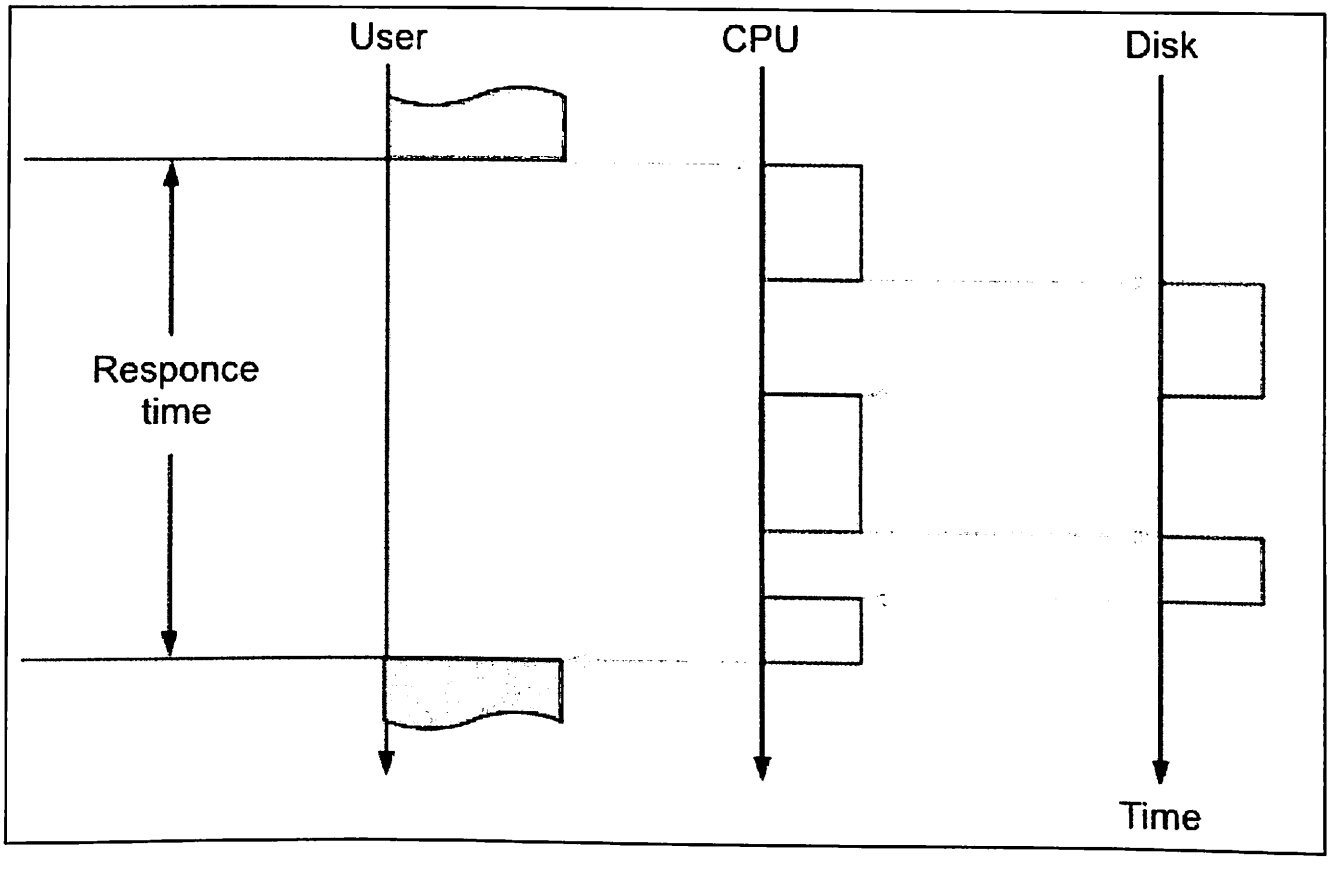

Figure 1.1 - Sequence diagram, time consumed by the user request.

A sequence diagram helps you understand how competition for shared resources affects a multitasking system. It reflects the reason why an increase in load can lead to a decrease in productivity. In an unloaded system, requests for processor usage are executed without delay. When we increase the load, some requests to the processor are forced to wait, because the processor at the time of their receipt is busy with another job. Each subsequent request in a heavily loaded system is forced to wait for additional time, compared to an unloaded system, which is characterized by the response time of the system. As a result, we get a situation, when a certain loading threshold is reached, each additional query to the database causes the load to grow exponentially.

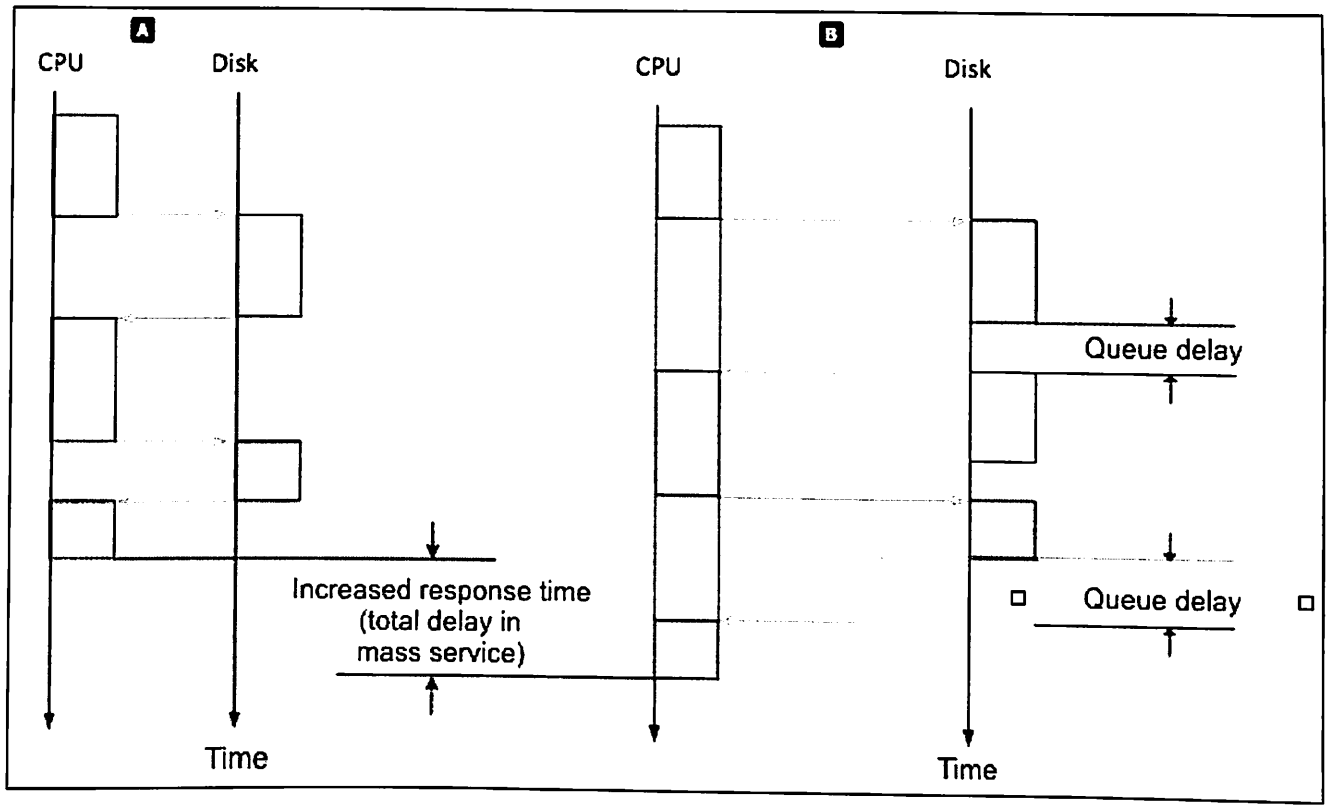

Figure 1.2 - Comparison of the response time of an unloaded system (A) with a highly loaded system (B).

In our case, this model demonstrates to us that if we use the mechanism of querying database performance representations for monitoring, then we may find ourselves in a situation where for execution of the query we will have to wait for some time until all previously arrived queries . And also that the monitoring request itself creates some delay in the execution queue. Thus, by frequent requests to the database, we will affect the performance of the system as a whole.

Performance modeling is a complex subject. The response time is essentially the only parameter that disturbs the end user. For the end user, the response time is the time between sending the request and returning the first byte of the response to this request. Because the response time is equal to the amount of service time and delay in the queue, we can reduce the response time by decreasing one of the components: the service time or the delay in the queue.

In loaded systems, the response time deteriorates due to the appearance of queues. Fighting them can reduce the load or reduce maintenance time. Finding ways to reduce workload is a necessary operation.

«The queuing model is well researched. It allows you to predict the performance of a system in which the time between the arrival of requests and the speed of service are distributed exponentially. This law corresponds to Oracle

DBMS. One of the main advantages of the queuing model is the illustration of a clear mathematical relationship between load parameters, service speed and waiting time. Also, the model allows you to focus attention on the fact that in order to optimize the system, it is necessary to consider the values of all parameters that can be discussed. The most cost-effective way to increase system performance is to get rid of unnecessary workload. There are two main ways to improve the efficiency of the system: clearing of unnecessary business functions and shortening the function code.»[3].

To determine the optimal performance of the database server, you can use the analysis of the amount of I / O operations used for process utilization, the system response time, and the transaction execution time. In each implementation, the level of performance will be determined by the set of software and hardware and the way they are used.

«Response time is the interval of time for which the database returns the first row of the result from the sent query to the database, i.e. this is the moment when the user receives a visual confirmation that his request has begun to be executed. ».

«The bandwidth determines how many requests a server can execute at a fixed time interval. Determines how many rows of the query result and what size is returned to the client per unit of time.».

«The increase in the number of active users and processes creates additional competition for system resources. The result of this load is an increase in the response time of the system, as well as a decrease in overall throughput. Also, the logical and physical integrity of the data with which it performs operations has a great effect on DBMS performance.» [4].

1.1 Overview of Oracle Database Architecture

The Oracle kernel has a large number of configuration options, the change of which can have a significant impact on system performance. To understand the processing of any query to the database, we should consider the architecture of the Oracle DBMS in more detail. The general scheme is quite clearly presented in the official documentation of Oracle, which is presented in Figure 1.3.

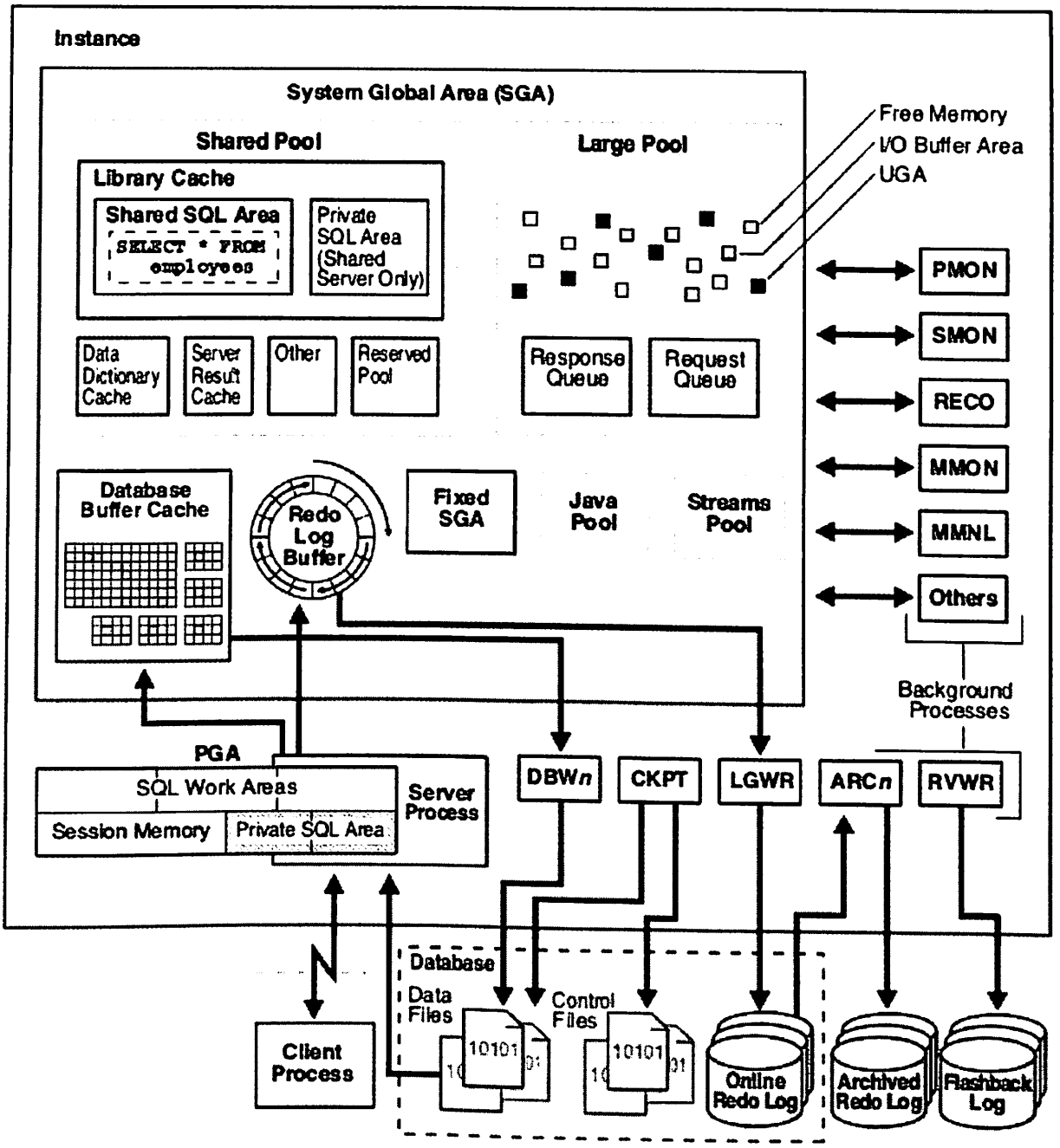

Figure 1.3 - The structure of the architecture of Oracle Database.

In this diagram (Figure 1.3), it is very clearly shown from which components the architecture of the Oracle database is composed and how they are related to each other. Let's take a closer look at this architecture in order to have an idea of where the information received through the queries comes from, and how to get access to it bypassing the database itself.

1.1.1 Oracle Structure. The database consists of three main elements:

1. Database files;
2. Processes (Server Process, Background process + Client process);

3. Areas of RAM (SGA, PGA).

The redo log files, or they are also called transaction logs, carry the following roles:

- Contain information about the execution of transactions;
- Used to restore database transactions to in the event of a database failure;
- Saving the replay log information is external to the relation to the data files;
- Provide Oracle with a way to write data to disk.

Control Files:

- Contain management information about all database files;
- Maintain the internal integrity of the database;
- Manage recovery operations;
- Usually stored on different disks to minimize them possible damage if the disk fails.

Tracing files (Flashback log):

- There is every background process that takes place in the DBMS
- Contain information about significant events that accompany the execution of the background process;
- Most useful when clarifying the causes of a serious malfunction.

Alert log:

- The commands and their results are written for the main events during the operation of the database;
- An important source of information for day-to-day management of the database;
- The journal entries will provide information about any problems that have occurred during the execution of operations in the database.

The file is created automatically if it is missing. It is used to link the log to a certain day. The file is renamed at the end of the day - the name contains the current date. The archive file of the warning message is obtained. The sequence of logs is important in order to understand the causes that lead to abnormal operation of the database.

1.1.2 Oracle processes. System:

- PMON is a process that monitors (processes monitor);
- SMON - system monitor;
- DBWR - the process of writing data to the database;
- LGWR is a process that records information in a transaction log or a replay log (LogWriter);
- CKPT is the process responsible for synchronizing the headers of the data files and the control file when the checkpoint passes.

Custom (to connect to the database):

- Server code + tool part.

The tool part is the code of some software tool (utility with which the user connects to the database, for example, SQL * Plus, SQL Navigator).

Oracle processes use files whose aggregate is a physical representation of the database:

- Data files;
- Control files;
- Log files.

Control files and log files support the functioning of the server. Must be present in the database, be open and accessible to the server.

1.1.3 The memory used by Oracle. "System memory for the entire database - SGA (System Global Area) - System Global Area. SGA in Oracle Database is part of the RAM shared by all processes of a single database instance. The MUH contains all the necessary information for the operations of the instance. " [5].

Used in common by different processes. The last requested information is stored in certain parts of the SGA. SGA parts:

- Fixed section (Fixed SGA) - User data, database statistics, table X$DUAL, etc.;
- Database Pool Buffer Pool (Large Pool);
- Shared SQL pool (Library cache / Shared pool);
- The Java pool;
- Redo log buffer;
- Database buffer cache.

Size SGA is formed mainly from the sum of 4 sections - Shared Pool, Large Pool, Redo Log, Fixed SGA. These parts can total up to 95% of the SGA.

For the process (user or system), the server memory allocates PGA (Process Global Area):

- Contains data and control information of one process;
- Between processes is not divided.

An instance of a database (Instance) is a set of memory structures and background processes that access a group of database files.

The configuration file is a plain text file, can be changed by the database administrator for the purpose of server settings. The file parameters determine the amount of resources associated with a specific database. A running instance gets a unique identifier - SID (Server Identifier).

Parameters that determine the size and composition of an instance:

- Initialization parameter file (init.ora);
- Server settings file (spfile.ora).

The initialization parameter file is read when the instance is loaded and can be changed by the DBA. All changes made take effect after the instance is restarted.

1.1.4 Data dictionaries. Logically, a database is a set of schemes, each of which is identified by its own name, unique in this database. Information about the structure of database objects, their location, access rights, etc. stored in the data dictionary (metadata base).

The information of the data dictionary is stored in the form of tables, over which numerous representations are created.

After the database is created and started, it is required, using system accounts SYS or SYSTEM, to enter the DBMS to create accounts for other users.

A user account is not a physical structure, but is associated with important relationships with database objects, because users own objects.

User SYS - owns data dictionary tables that contain information about other database structures. The SYSTEM user has views referring to these data dictionary tables.

Schema - a set of objects belonging to a user account.

Data dictionary views (prefixes):

- DBA - information about all database objects;

- ALL - information only about those objects that are available to the user;
- USER - information about all database objects owned by the user.

For example, USER_TABLES is information about tables that are in the current user's schema.

1.1.5 Dynamic Views. View current server activity - provide special views with the prefix "V$". These views are called dynamic. For example, V$SESSION - information about current sessions.

Dynamic performance views provide access to information about the changing state of the instance memory structures [6]. The Oracle database also supports a dynamic set of data about the performance and performance of the database instance. Dynamic performance views include information about:

- Sessions;
- File states;
- Progress of tasks and tasks;
- Locks;
- Backup status;
- Use memory and memory allocations;
- System and session settings;
- Executing SQL;
- Statistics and metrics.

In the version of Oracle 11g, there are more than 590 such views. All V$ views and all dictionary views are fully documented in the Oracle Database Reference.

1.1.6 Fixed tables. In fact, when we make a read from the SGA, we read from fixed X$ tables. In fact, X$ tables are memory areas that are located in the fixed-area area (Fixed Area). The description of fixed tables X$ is available via the V$FIXED_TABLE view. Fixed tables X$ are basic tables for dynamic representations of V$. Storage structures in fixed X$ tables are not documented in Oracle and can vary from version to version. However, the very structure of X$ tables is stored in the database and you can see this description through a query to V$FIXED_VIEW_DEFINITION.

"The names of X$ tables look unreadable at first glance, but some logic is embedded in these names. Oracle kernel consists of several levels. For each level there are their memory areas (which are visible to us as X$ tables), containing information about the statuses and statistics of the functions of this level. So, the first two letters in the name X$ tables indicate to what level it belongs. "[7].
List of kernel levels:

- Compilation layer (KK). The level is responsible for compiling PL / SQL objects and generating the explain plan based on the data from the dictionary.
- Execution layer (KX). Executes the compiled code from the previous level and makes the binding of SQL and PL / SQL objects. Also responsible for recursive calls to the dictionary.
- Distributed Transaction layer (K2). Provides two-phase commits in distributed transactions.
- Security layer (KZ). Manages roles and system privileges. Participates in the compilation and execution phase.
- Query layer (KQ). Caches the data from the dictionary. Compilation (KK) and Security (KZ) levels use this level data at compile time.
- Access layer (KA). Responsible for access to segments and provides information to the above levels.
- Data layer (KD). Responsible for writing and reading physical data to / from segments. Configures segments to store tables or indexes.
- Transaction layer (KT). Works with rollback segments. Manages freelist, ITL (Intrested Transaction List) within the data blocks, row-level locks during transactions and generation of undo.
- Cache layer (KC). In close contact with the operating system controls buffer cache'em and shared memory. Also responsible for creating redo-information and writing redolog files.
- Service layer (KS). Works for other levels. Tracks the initialization parameters for sessions and the system as a whole. Latches and locks in the case of a one-instance instance are also his concern. Also responsible for watt events and system statistics.
- Lock Management layer (KJ). Manages locks and resource allocation when running Oracle in a cluster (RAC).

- Generic layer (KG). Provides control for common data structures that are used by higher layers, such as linked lists.

1.1.7 Use information from the SGA area. Information in the SGA is accumulated during the operation of the database instance. Part of the information is filled in automatically by internal Oracle processes, the other part of the information is populated and cached during execution of user queries to the database.

Automatic filling:

- Caching the data block;
- Log buffer
- SQL cache
- Update system and user statistics.

Custom queries:

- Information about users from V$SESSION;
- Information about the system from V$PARAMETER;
- Performance statistics from V$SYSSTAT, V$SYSTEM_EVENT;
- Cache buffer headers from X$BH;

1.1.8 Location of the SGA in RAM. Oracle stores SGA in RAM. In order for each process to have free access to stored information and quickly interact with other processes, this section is placed in the shared memory of the operating system (Shared Memory). Shared memory is the fastest means of exchanging data between processes.

The shared memory technology allows information to be exchanged through a common memory segment for processes without using system kernel calls. The segment of shared memory is connected to the free part of the virtual process address space. Thus, two different processes can have different addresses of the same cell of the connected shared memory.

Since there are restrictions on the length of the segment, when you place SGA in shared memory, Oracle divides it into several segments. This gives more flexibility in allocating additional address space, as well as some logical separation in the stored information. For each segment, the operating system generates a random segment identifier - SHMID (Shared Memory ID), through which the process of connecting to

the segment. The operating system allows you to assign a label - KEY - when creating a segment. Key is not a required attribute, but allows you to associate a segment with a label in a more convenient form.

In one segment, Oracle stores all information about all existing SGA memory segments in the system, which is the table of contents of the stored information in the database instance. In order for each process to accurately locate this segment, Oracle generates to it a KEY based on the instance's SID and the ORACLE_HOME environment variable. Unfortunately, Oracle does not publish in public sources what function is used when generating this identifier.

1.2 Solving performance problems.

1.2.1 Factors affecting productivity. The performance of the Oracle database can be affected by a range of different factors. Thomas Kite in his book mentions the main problems encountered in the operation of the Oracle DBMS:

1. «Inefficient SQL operators can significantly affect the performance of queries. The more specifically the operator is defined, the faster we can get the result.
2. Inexperienced database configuration negatively affects performance. The specific initialization and configuration parameters of the database affect not only the launch configuration, but also the processing of requests.
3. Inadequate memory allocation of the system global area (System Global Area - SGA) negatively affects performance. The system global area is a group of common memory structures that contain data and control information for one instance of the Oracle database. Users who simultaneously connect to the same instance share data in the instance's SGA. Therefore, SGA is sometimes called a common global area. To ensure efficient database performance, you need to determine the optimal SGA memory size and configure it accordingly. Too little memory can significantly reduce the performance of the database.»[8].

In order that you can work on increasing the performance of the database, you must first find bottlenecks in its work. To do this, we need to analyze the Oracle waiting events. Analyzing event data will help us understand how we need to adjust database parameters to reduce these costs. The waiting events greatly affect the performance drop of the database. They occur during the processing of requests. A read / write request to the Oracle database causes several processes. These processes may encounter hardware, software, operating system or configuration problems, which will lead to slower processing or complete stopping of the database. Such

blocking factors are called waiting events. You should analyze all the events of expectations in the database and try to reduce them. However, it is impossible to completely eliminate all waiting events.

«Oracle Database tracks more than 1000 wait events, which can lead to delays in processing the request.» [9]. These waiting events are divided into the following groups:

- Cluster;
- Network;
- Administration;
- Configuration;
- Commit;
- Application (Application);
- Concurrency;
- System I / O (System I / O);
- User input / output (User I / O);
- Processor (CPU).

Analysis of internal processes of the database can be obtained through requests to performance views. These views contain all the necessary data so that you can evaluate the indicators that affect the response time and system performance. For the analysis, we need to have both current performance indicators and statistically accumulated data during the operation of the database.

1.2.2 SQL query processing in Oracle. «When executing each database query, the Oracle kernel tries to ensure the integrity of the data. This means that by generating any request through an SQL query, we generate a chain of actions that the internal Oracle engine performs before we start to access the data blocks directly.» [10].

The preliminary step to begin processing requests to the database will be to start the session itself to the database instance. At the stage of creating the session, Oracle starts the server process, which is generated by the client process. Next, the Oracle server process creates a software global area (PGA) for each user when the user session is opened. This area contains data and control information for the dedicated server process, such as temporary storage of variables, SQL parsing, etc.,

which Oracle creates for each individual user. PGA is designed for exclusive use by each server process and can not be shared among several processes. All these processes are performed in the private area of the PGA and can not be used globally between all processes, similar to SGA.

Next, a request is generated in the client session. He understands, the system variables are attached. After that, the request is executed. When executed, the database instance itself ensures the integrity and consistency of the data. To do this, it checks the access rights to each object it is querying, creates an audit for the requested object, checks the syntax of the supplied query, commits a data lock to prevent simultaneous data changes from different places, checks all existing columns in the request for the fact that they exist in the database. Next, the optimizer determines the execution plan for the query. After that, Oracle places the compiled SQL query in Shared Pool so that it does not have to be compiled next time and could be reused [11].

1.3 Comparison and analysis of monitoring systems used to track storage systems. To ensure maximum performance and fault tolerance of the system, it is necessary to continuously monitor it. Monitoring is a mandatory process in the operation of a large data storage system. The result of the monitoring is a set of measured values of the parameters obtained in inextricably adjacent intervals of time, during which the values of the parameters do not change significantly. From these values, conclusions are drawn about the current state and functioning of the system behind which monitoring is performed.

At present, there are quite a few tools for monitoring the health status of key information systems. For the monitoring of databases, two types of monitoring systems can be provisionally identified. The main difference will be in what means and what performance metrics will be used.

The first class can be described as external controls and monitoring system. They are characterized by the use of mechanisms provided by the database itself or the operating system on which it is installed. Typically, such indicators can be statistics on the loading of hardware components of the system (CPU load, the amount of RAM used, disk I / O parameters, network load, etc.), as well as the performance of the database itself, which return to the tool monitoring through database queries or mechanisms that are supported by the database to broadcast this information (SCR queries, return metrics via SNMP queries, WMI poll, HTTP

requests). Such monitoring tools include a rather wide range of products for monitoring computer networks, which are "universal combines" with written extensions and modules for various tasks. These tools are good enough, but often do not provide optimization of performance, speed of processing requests, often have overhead expenses for the statistics cathedral. Since such monitoring tools are oriented to a wide range of tasks and systems that they should monitor, it is likely that they will not provide comprehensive statistics on the monitoring object, but will rely on the baseline indicators.

In addition to universal monitoring tools, you can use specialized software products that are developed for specific DBMS. These are narrowly focused products that deal with detailed diagnostics of the state of work, and also often provide DBMS management tools. Such products are a powerful tool based on a single management interface, providing detailed control of all aspects of behavior and management of the DBMS. Such systems are completed with agents that are put on database host machines, which allows you to analyze in more depth and in detail the processes occurring within the operating system and database. Typically, manufacturers try to supply such software products with all kinds of analysis, monitoring and management mechanisms that provide exhaustive data on the operation of the DBMS. But precisely because of this, they have a serious minus - these systems turn into "monsteroidal" complexes, which require large resources for both work and maintenance. Sometimes working with such a system comes to an absurd situation, when a specialized monitoring complex requires a means of monitoring itself. So it can be cumbersome and depend on the work of a very large number of components on which such a database management and monitoring tool relies.

To the second class of monitoring tools, it is possible to include the built-in diagnostic systems. These systems are executed in the form of program modules built into the database engine. They perform diagnostics and control functions for only one device, and this is their main difference from centralized control systems. A DBMS can provide access to database information stored in its resident performance tables, often referred to simply as system tables or system databases, depending on which DBMS is selected. These tables store all system-level data for the database instance. These include dynamic performance views that are populated by the database engine at runtime and allow you to obtain statistics on the current state through specialized queries to these views. Such representations are used by the database itself, and

information from them is transferred to a higher level of work with the database to external monitoring systems, individual requests or samples sent to the database.

When choosing a database monitoring tool, first of all the goal is to obtain data for:

- Server performance checks under load testing;
- Checks database architecture;
- Indications of changes in productivity over time;

For each goal, the main objects are defined, from which the monitoring tool will receive metrics. The main objects of database monitoring can be:

- Auditing user activity - checking current user sessions, identifying the user ID, the host from which the session is started, determining the application module through which the database is accessed, identifying the processes that are loading the database, analyzing the events of expectations;
- Audit allocated resources for Oracle memory structures, such as SGA, PGA and buffer cache;
- SQL query auditing - checking for SQL queries that consume too much system resources;
- Analysis of the lack of resources for memory structures. Is the resources allocated enough?

Next, you need to evaluate what metrics you need to track to solve the task. After the requirements have been clarified, it is possible to proceed with the selection of the system or groups of monitoring systems that will help to cope with the task. For the correct choice of the monitoring system, it is necessary to take into account all existing and potential problems of the database and its environment.

Having accumulated statistical data on the metrics of interest to us through the monitoring system, we can proceed to an analysis of the operation of our system in the following areas:

- Analysis of the need to upgrade the server hardware configuration;
- Analysis of processes and components that require optimization;-
- Diagnosing problems that affect system performance;
- Analysis of recovery and backup plans;
- Analysis of the service schedule;
- Formation of the main criteria for assessing performance.

These data will help determine the vector of required work to optimize and improve system performance. With such use, monitoring will be the main tool in dealing with problems in the operation of the monitored system.

To select a monitoring tool, you also need to understand what tools the monitoring system uses to get data from the database, which queries are used in the poll. Depending on this, the effectiveness of the executed query can change significantly, the additional load on the database can be changed, and the relevance and reliability of the information returned to the monitoring system can vary. It is very important to understand and know how the selected monitoring tool works. Understanding this, you can optimize monitoring for the tasks set in the production, while not spending the resources of the database servers.

1.3.1 Internal monitoring tools Oracle. Oracle already includes a prepared set of tools and tools for monitoring databases. Perhaps both a joint and a separate use of these tools, depending on the task. The simplest and most obvious way to monitor Oracle is to use SQL queries. This method is convenient in those situations where we know exactly what metrics we need to get to solve a particular problem. In this case, this approach will be the most rapid and does not require additional configuration of monitoring tools. In cases where monitoring should be carried out on an ongoing basis over a long period of time and the number of metrics received is too large, such a method becomes uncomfortable due to limitations in the field of automation.

Oracle Enterprise Manager (OEM) - a set of tools for the centralized management of systems based on Oracle products, including databases, application servers, HTTP servers, Internet applications, etc. Oracle Enterprise Manager allows us to use two basic tools: Oracle Enterprise Grid Control and Oracle Enterprise Database Control. Oracle Enterprise Grid Control is a single tool for managing all Oracle databases in the enterprise. Oracle Enterprise Database Control is an integrated tool for managing a single database for a local server, with a greatly simplified functionality and clipping of unnecessary functions for clusters, agent management, standby, etc. Both of these products are essentially the same tool, with only a different set of functionalities used, and are Oracle Enterprise Manager's solution.

OEM includes three components: central consoles, followed by database administrators, management servers, implementing all the logic of the OEM

operation and intelligent agents (Intelligent Agents) running on the nodes hosting the database and performing tasks on behalf of the managers servers. The management server has its own repository, where it stores the information necessary for work on database users, nodes, privileges, etc. The repository is stored in the Oracle database. The console performs the functions of the interface.

OEM allows you to execute not only the commands that are executed by the Instance of Oracle, but also the commands of the operating system, start and stop the database. Therefore, on each managed node, a service that is not tied to the state of the database must work. This role is performed by an intelligent agent. It can execute scripts, start the database, execute operating system commands, monitor the occurrence of OEM-ordered events. Moreover, the execution of these works can occur at pre-specified times or with a certain periodicity, and the result will be transferred to the management server when it will have communication with the agent.

Together with the OEM, the user can install his Web version. It does not require additional configuration. The user simply runs an installed simplified version of the application server on the computer with the OEM and works with the console through the internet/intranet from any computer where there is a Web browser. Through the Web-interface all functions of OEM and additional modules are available.

OEM graphically displays the state of the database during processing. It also provides a detailed analytical report in which you can go to each event and find the corresponding SQL statements (Figure 1.5). The legend on the right shows the types of waiting events.

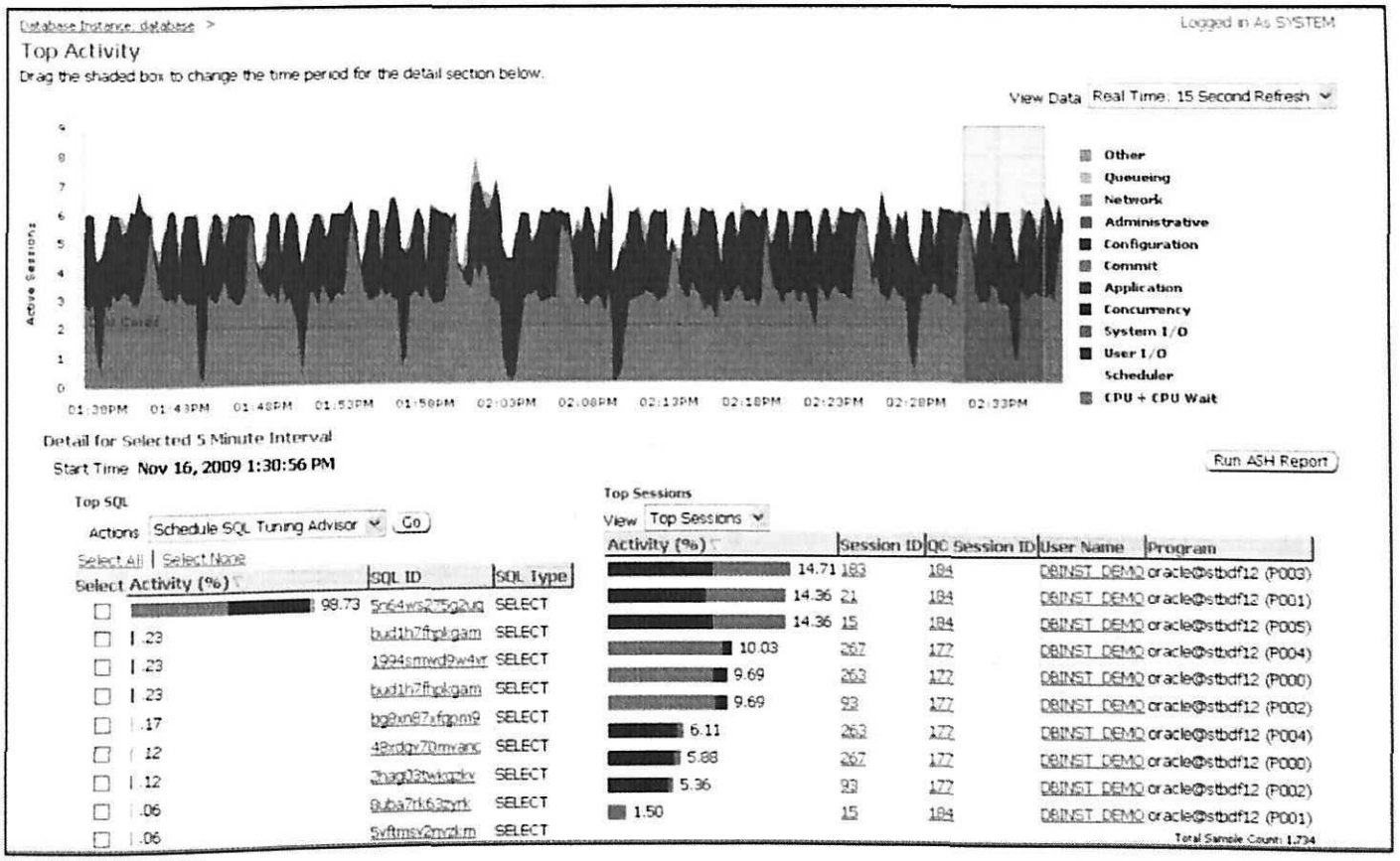

Figure 1.5 - Oracle Enterprise Manager. Web console activities in Oracle.

We can proceed to a more detailed display of the data. Figure 1.6 shows a graph of the User I/O waiting for the data.

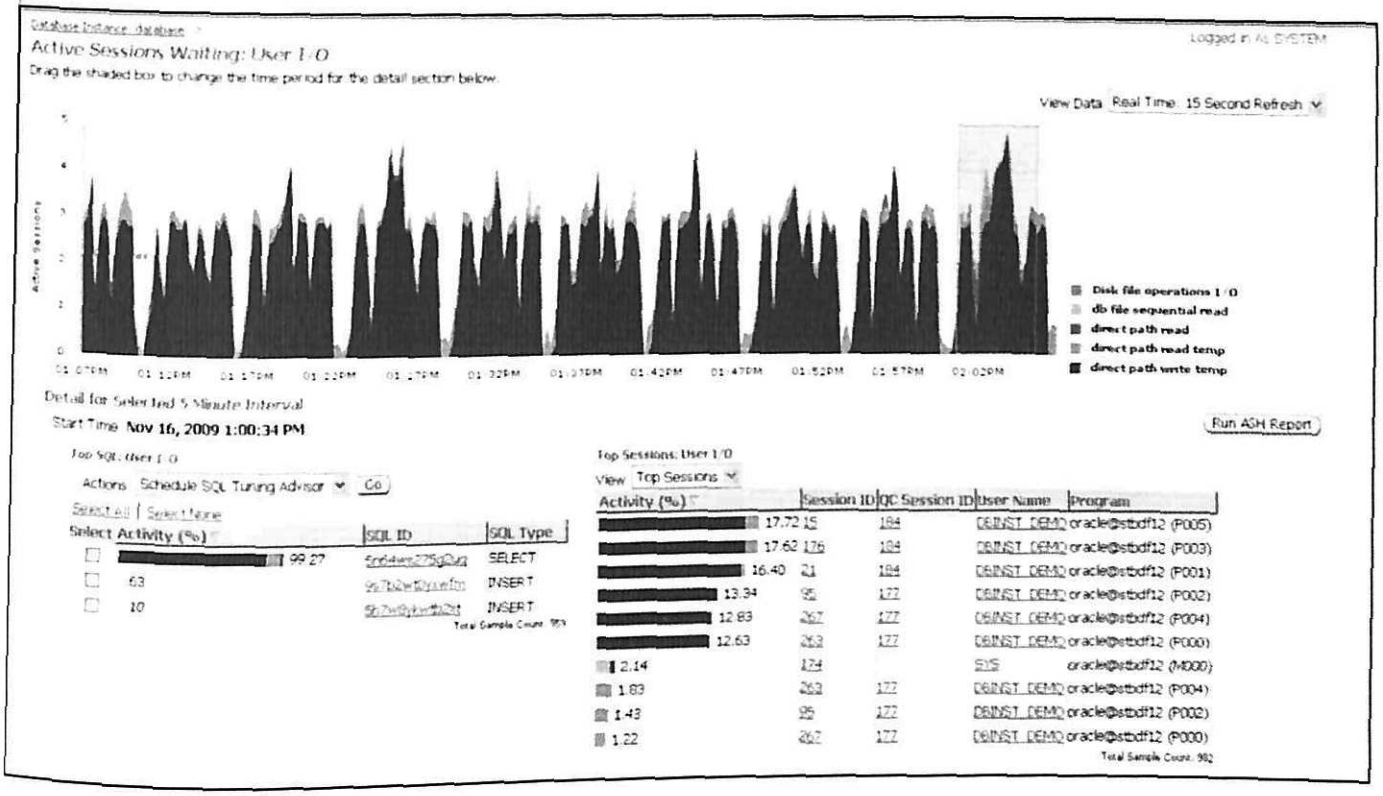

Figure 1.6 - Waiting for active sessions.

In addition, you can collect statistics on waiting events of an Oracle database without using tools, using a shell script and built-in system utilities.

1.3.2 Internal monitoring tools Oracle. ADD and AWR. Automated database monitor (ADDM) allows Oracle to diagnose itself, to identify potential problems and solutions. ADDM is performed automatically after each statistics collection in the Automatic Workload Repository (AWR), generating the actual performance diagnostics data. Because data in the AWR is collected regularly, a similar automatic start mechanism provides reliable diagnostics of database performance and identification of sources of problems. Here is a list of what ADDM checks or considers as problems:

- CPU bottlenecks. Checks Oracle applications and third-party applications to misuse all CPU resources;
- I/O bandwidth problems. Analyzes the state of the I/O subsystem;
- High execution and compilation load of PL/SQL, as well as high load of Java usage;

- Specific RAC problems (Real Application Clusters). What are the most used blocks and common cache objects? Identifies whether there are delays related to the operation of the logic and internal connections of the cluster;
- The non-optimal use of Oracle by the application. Looks for problems in poor connection management, with excessive structural analysis or locks at the application level;
- Database configuration problems. Checks for signs of incorrect logging, archiving problems, excessive control points or suboptimal settings:
- Concurrent access problems. Analyzes requests for the frequency of the "buffer busy" response;
- Hot spots and top SQL for problems in different areas.

ADDM and AWR reports are a very rich tool for identifying performance problems and a good starting point for
optimization of the database.

1.3.3 Internal monitoring tools Oracle. Emergency Monitoring. In Enterprise Manager 12c, the function of Emergency Monitoring appeared. The function of Emergency Monitoring 12c Enterprise Manager Cloud Control is an evolution of the memory access mode inherited from Grid Control 11g, which allows access to blocks stored in shared memory. This tool allows you to automate the method of accessing memory without having to manually enable the "memory access mode". This tool will provide information that will help analyze performance problems, even when the database is "hung" and there is no way to get performance statistics through standard tools. This mechanism is proprietary, but the main functionality relies on the ability to download and analyze a dump of RAM from the server - the utility "Oracledebug Hang Analyze". This allows you to analyze different views, such as V$SESS_TIME_MODEL, V$SESSTAT, V$SQLSTATS, V$ACTIVE_SESSION_HISTORY, DBA_BLOCKERS, DBA_WAITERS, etc., after which the OEM will show the supposed blockers that can affect the database and allow them to be eliminated through the graphical interface (Figure 1.7).

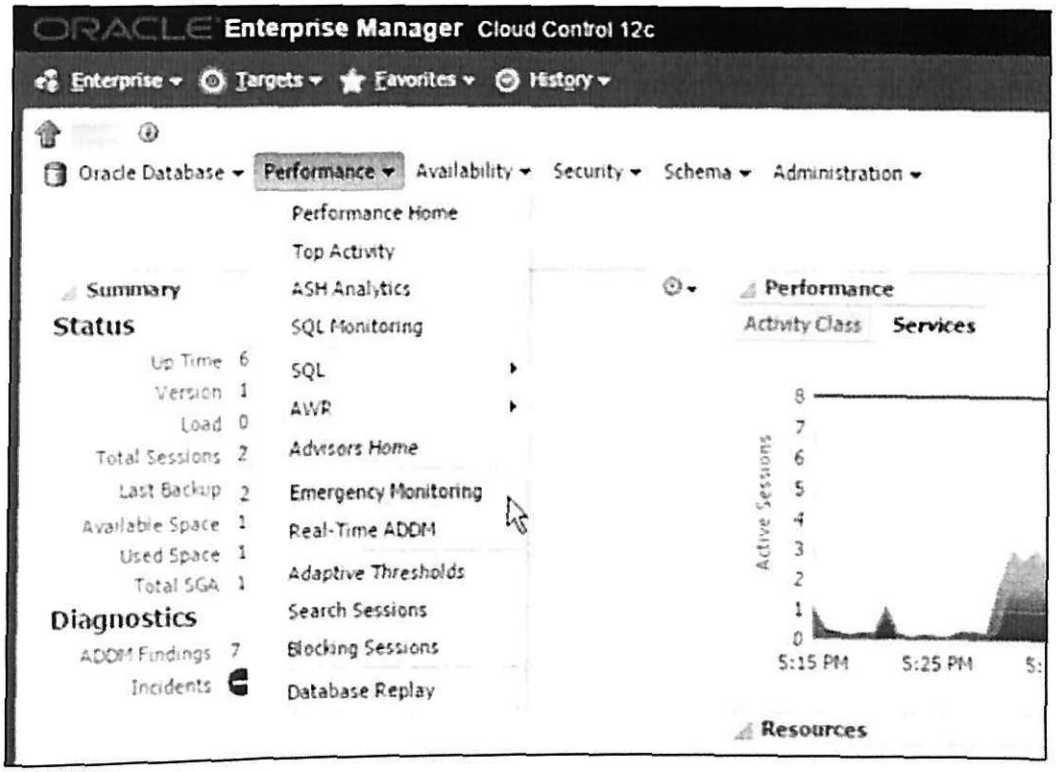

Figure 1.7 – Activation of Emergency Monitoring.

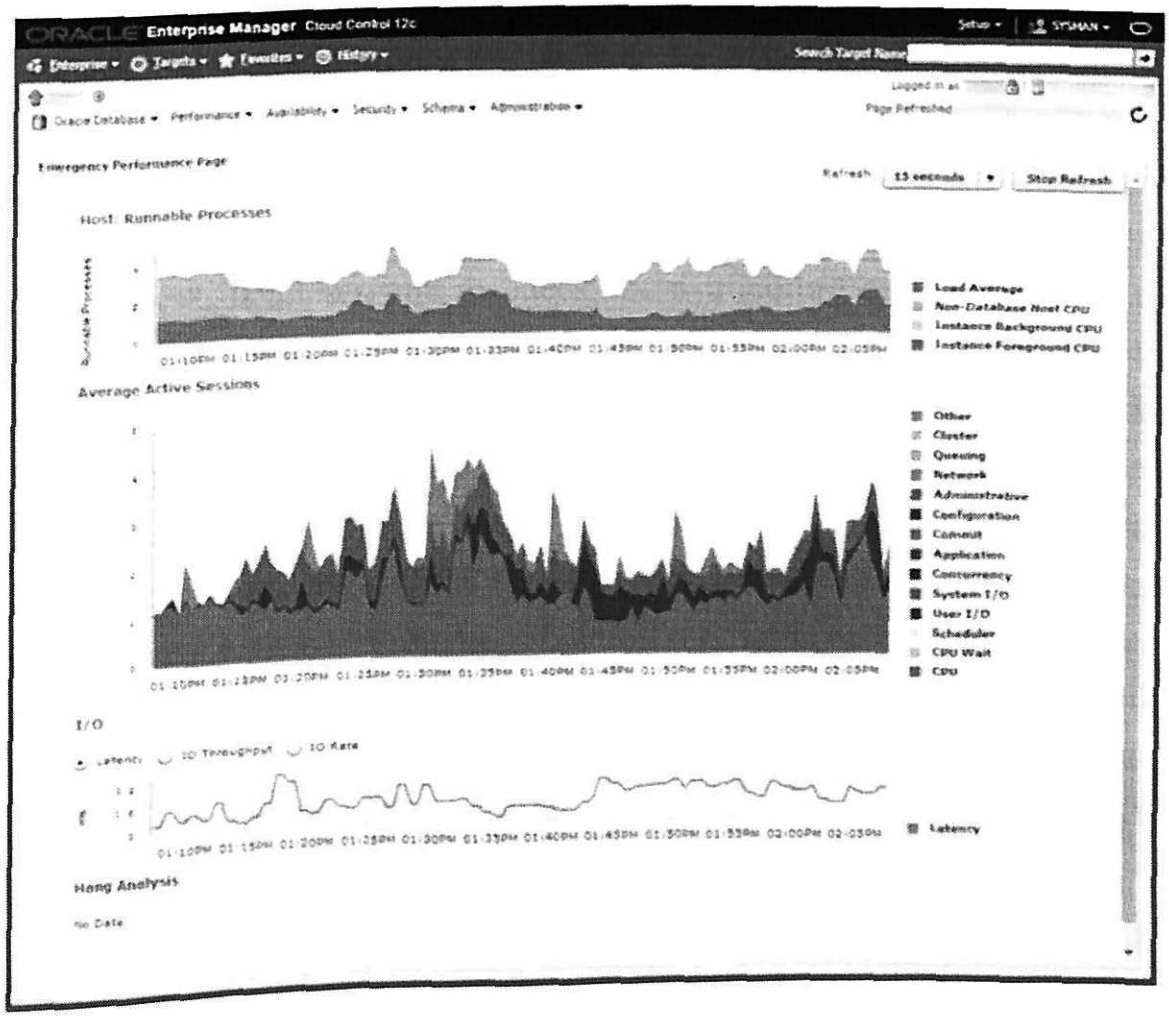

Figure 1.8 - Resulting screen Emergency Monitoring.

Although this tool is aimed at working with data directly from the SGA and has console access (Figure 1.8), its use is not suitable for monitoring on an ongoing basis, because the mechanism does not work directly with objects in RAM, but through the collection of a memory dump, which leads to large time delays before obtaining statistics and overheads to the disk subsystem for storing traces and a memory dump [12].

1.3.4 Third-party monitoring tools for Oracle databases. Third-party external monitoring tools include a number of products aimed at monitoring computer networks. These tools usually use database monitoring modules, which include various sensors, which in turn return values for different metrics of the database under investigation. In principle, all these monitoring tools are based on the same mechanisms of information collection as standard ones. The main conceptual difference is that all information on all enterprise information systems will be consolidated in a single management interface.

A common multifunctional monitoring tool is the open-source Zabbix system. This monitoring system serves to monitor the status of various services of the computer network, network equipment, servers and operating systems. Zabbix supports several models of obtaining statistics from the observed object. The first method is a method of obtaining statistics without the use of monitoring agents "Simple checks". It allows you to check the availability and speed of response of standard services, such as HTTP, SMTP, ODBC, WMI, SNMP. The second method is to install Zabbix Agent on the host. The agent can be installed on Windows and UNIX-like hosts to obtain data on processor load, network usage, disk space and other statistics. There is also a method called "External check". It allows you to execute external programs and the result of their work Zabbix uses both monitoring metrics. In essence, this is a remote execution on the server of shell scripts or binary programs that return a certain result to Zabbix. Thanks to its good extensibility, this monitoring system makes it possible to monitor virtually any metrics of the system. Some drawbacks include laboriousness in tuning, if necessary, to monitor non-standard parameters of the system [13].

To monitor the hardware of the server, you can use the monitoring system Cacti. This is an Open-Source web application that builds performance graphs, through a set of utilities for working with RRD (Round-robin Database, ring database). Monitoring the hardware of the server is necessary, because a hardware

failure will lead to a decrease in performance or total failure of the database itself. This is a proactive measure to protect the hardware of the database server. This software uses statistical data on the processor load, allocation of RAM, the number of running processes, the use of network traffic. This monitoring system is useful for determining the cause-and-effect relationship between the operation of the database, analyzing network traffic and loading the server hardware parts [24].

Considering the tools for monitoring Oracle databases, in fact we are considering various interfaces to the basic monitoring tools of this database. In the end, it all comes down to using all the same SQL queries sent to the database. Therefore, analyzing the work of Oracle monitoring tools, first of all you need to explore the built-in functionality.

1.3.5 Alternative ways of obtaining performance metrics. In his book, Carrie Milsap, «Oracle. Optimization of performance» [15] investigated the issue of obtaining data through queries to fixed views when diagnosing performance problems of Oracle databases. He investigated the effect of the meter in questioning the data of representations. Based on his research, he comes to the following conclusion: «Interrogating fixed Oracle representations with SQL help creates an extremely strong effect of the meter on the system. It is simply impossible to obtain detailed real-time performance statistics. Example 8.1 illustrates this problem. On our eight-hundred-megahertz server under Linux, the typical speed does not exceed 50 executions per second for a 50-line request to V$SESSION» [16], which means that the standard monitoring methods used may not be suitable for very frequent database polling for statistics on user sessions and executable procedures that were launched within these sessions.

In one of his books «Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g Programming» [17], one of the leading Oracle engineers - Thomas Kite, indicates the possibility of accessing dynamic views directly in the main memory allocated for SGA . These thoughts lead us to the idea that we can implement monitoring through accesses to memory directly. Given that none of the current Oracle monitoring tools is using this approach when accessing metrics, I conducted a study of whether there was an attempt to implement a program to read data directly from the SGA. The work MiladinModracovic, in which the idea of direct access to SGA through the launched program is developed.

The program implemented in this project allows you to read from the SGA, but in order for it to earn, it must first enter the input parameters of the current database. To do this, enter the Shmid values, the starting address of the memory segment, and manually fill the indentation table for information about the sessions. Given that the first two values change each time the database instance is restarted, and the indent values change with each new Oracle database release, this tool is not suitable for work because of its non-universality. This program will work in a single case, and the time spent on finding out the preliminary parameters will be unacceptably large.
Based on these ideas, you can implement a method for monitoring the database, which will be universal (independent of the release of Oracle version) and the most customizable (not requiring manual input parameters). And based on this method, implement a monitoring program that will have no problems creating an additional load on the performance of the Oracle database, and allows you to receive data from "hung" databases. Further, these developments can be used to write the module to existing monitoring systems.

1.4 Description and reasoning of its decision. As we can see, there are various tools for querying the state of database components. On the market there are both internal and external software products, which are sharpened to monitor all possible parameters of the system. And although these funds are quite large, they all rely on approximately one approach for obtaining state metrics - queries into the database itself. And with this approach of obtaining metrics, all the monitoring tools discussed have the same problem - they create a load on the database, causing the database to process requests. Basically, these are SQL queries that are directed to the dynamic views of Oracle.

In order to implement proactive monitoring of the DBMS, we need a different mechanism that will allow us to directly obtain statistics from dynamic views, bypassing the processing stage of the query, which, on the one hand, will not create additional load on the database, and on the other hand, receive data even when the database is not responds.

Oracle receives performance data not from standard tables, but through calls to shared SGA memory. Thus, all performance views are based on virtual tables that are created from memory structures on the database server and are located in the public area of the SGA memory.

When direct reading of data from SGA to the Oracle server does not place additional burden. It does not need to create latches and buffer locks (latches, buffer locks), SQL parsing, care for consistent data. When the server becomes overloaded, this feature becomes very important. Also from SGA, you can read data very quickly and often. Much faster than Oracle reads itself, which follows from the previous statement. And also you can join SGA even when the Oracle server does not respond, for example, if the maximum number of sessions is exceeded or when the place under archive logs has run out.

As mentioned earlier in the "Solving performance problems" chapter, each query in the database generates a whole chain of actions that the database produces before it begins to provide the requested data. In the case of reading the data on system performance (monitoring), and in fact these are queries to fixed X$ tables - all these things are not particularly meaningful. This is a complete absence of logical (read from the cache) and physical reading from disks, no calls to the buffer cache are created, there is no need to lock the data, which occurs when working with SQL-query to regular structures (tables, views).

Because SGA is located in the shared memory of the operating system (Shared memory) and access to it can get any process, then you can read the contents of the fixed X$ tables even if the database is not open, but simply raised. Given the fact that the structure of fixed values is unchanged during the operation of the database instance, knowing the "map" of information storage in these tables, we can obtain the required data without access to the database itself.

1.4.1 Advantages of using direct access to SGA. When accessing SGA, we get the following additional benefits:

- Access to hidden information;
- Access to data at the time when the database “hung";
- Fast access speed;
- Small overhead.

1. Access to hidden information.

With direct access to SGA, you can read the contents of data blocks, such as the names of headers or values from fixed X$ tables.

2. Suspending or slowing down the database.

On large installations, where it is not possible to work easily, there is often a situation that access to the database becomes limited. There are common causes that can cause this behavior of the database:

- ORA 4031 shared pool errors
- Suspending the archiving process or switching the retry log;
- Problems with the "hang" of the base;
- Latch cache library competition.
- The use of free buffers in the DEFAULT buffer pool

In some cases, direct access to the SGA may be the only way to obtain up-to-date information from a running instance. For example, if we had a mutual lock during the execution of a heavy SQL query, we may lose access through Enterprise Manager and not see what caused this behavior. As a result, we will either have to wait for all requests to be executed, or stop the database instance. In the case of reading data from the SGA, we can see what request is being executed at the current moment and kill only that session that affects the operation of the instance, rather than stopping the whole instance.

3. Fast access speed

Access via SGA allows you to get values much faster than through SQL-query. A permanent survey of the view allows us to have statistical data about who and what requests are sent to the database for execution. Using frequent SQL statistics and user samples that use this statistics, you can calculate how often a specific user accesses a specific SQL value. Having accumulated data, you can get statistics on the following areas:

- Selecting user expectations, tracking transaction (ester) data;
- Statistics of custom SQL queries;
- Statistics of the instance operation.

Constant and frequent polling of views through a normal SQL query will result in degradation of database performance.

4. Small overheads

When using direct access to the SGA area, we do not spend the resources allocated to the operation of the database instance. And access to the segment reading from RAM in the operating system requires significantly less time and resources than processing a request through the DBMS to access information stored in RAM as well. This gives us the following statements:

- Monitoring can significantly affect the operation of the database;
- The Heisenberg uncertainty effect - a small load affects to a lesser extent how monitoring itself affects the performance of the database that it monitors.

1.4.2 Disadvantages of using direct access to SGA. And although the method of direct access to SGA has many advantages, there are a number of undoubted shortcomings.

First, this is a very troublesome method of obtaining information, because to connect to SGA you need to know the shmid shared memory segments of the operating system in which the information we need is stored, and each time the database instance is rebooted, shmid changes and it needs to be recognized again.

The second disadvantage is that when reading from the SGA, the consistency of reading is not observed. Formally, when reading data, say from V$SESSION, we can get a picture of user sessions that was not at any one time. And in the blocks of memory reserved for storing information about sessions, residual information or "garbage" can be stored. This is due to the fact that some of the data is populated at the time of the request or through triggers of Oracle's internal mechanisms. Oracle does not care about the preliminary and subsequent cleaning of data after obsolescence of information.

And to the main disadvantages of using SGA is the fact that not all data can be read directly. In the X$ tables, there are fields that Oracle calculates dynamically, at the moment when a request is made to read these fields. As a result, instead of information in a readable form, we will get some numerical values that will require further processing (decoding).

2. TESTING

«Interrogating Oracle fixed views with SQL creates an extremely powerfu effect of the meter's impact on the system and it is simply impossible to obtai detailed statistics for real-time execution.» [15].

The purpose of testing is to confirm the inefficiency of using on-going monitoring tools and to test the effectiveness of direct access to shared SGA memory that does not affect the operation of the database.

2.1 Test environment. To confirm this assumption, we deploy a test environment, including: a separate test database Oracle 11g, Oracle Enterprise Manager 13c, pre-making the distribution of the database management agent . In the test environment, configure the database monitoring tools. To perform a test load generated by users, we will use a separate host, running various scripts through Shell and the SQL Plus and PL / SQL system utility. To collect statistics on the operation of the database, we will use the Oracle Enterprise Manager Web Console.

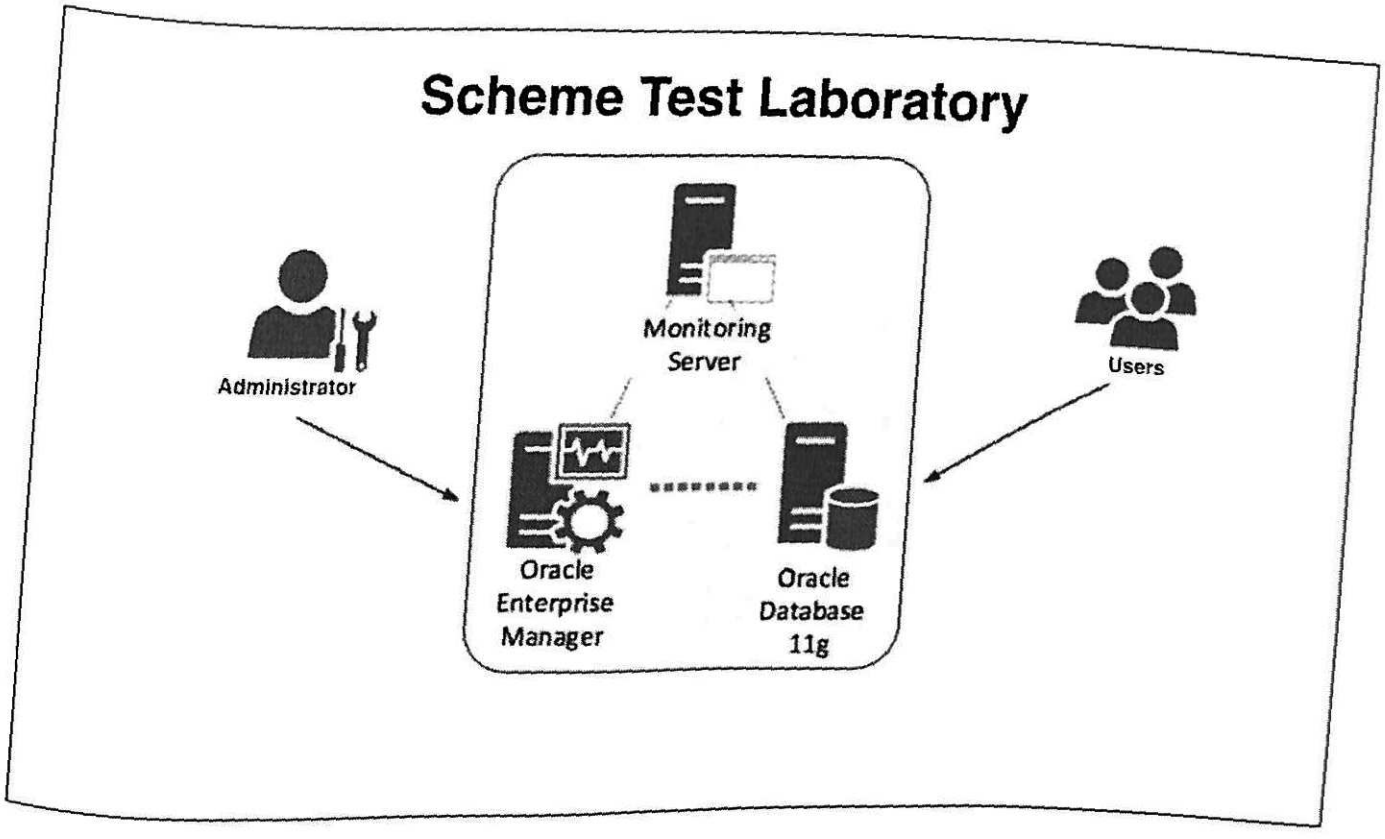

Figure 2.1 - Scheme of the test stand.

To implement the test bench, I use the Virtual Box virtual environment of the atest version hosted by the host under the Linux Mint 13 management. Within the

virtual infrastructure, two servers have been raised for the Oracle 11g database and Oracle Enterprise Manager 13c. Oracle Database is installed on Oracle Linux 7.2, Oracle Enterprise Manager is installed on Oracle Linux 6.0. Between the host machine and the servers, complete network connectivity is configured. All load tests are started from the host machine, so as not to create additional load on the virtual environment (Figure 2.1).

For each virtual machine resources are allocated, in accordance with Oracle's minimum requirements for Oracle Database and Enterprise Manager:

1. Host - 4 CPU, 16Gb Ram, 500Gb SSD;
2. VM Oracle Database 11g - 2 CPU, 4Gb Ram, 30Gb SSD (Figure 2.2);
3. VM Oracle Enterprise Manager 13c - 2CPU, 8Gb Ram, 40Gb SSD (Figure 2.3).

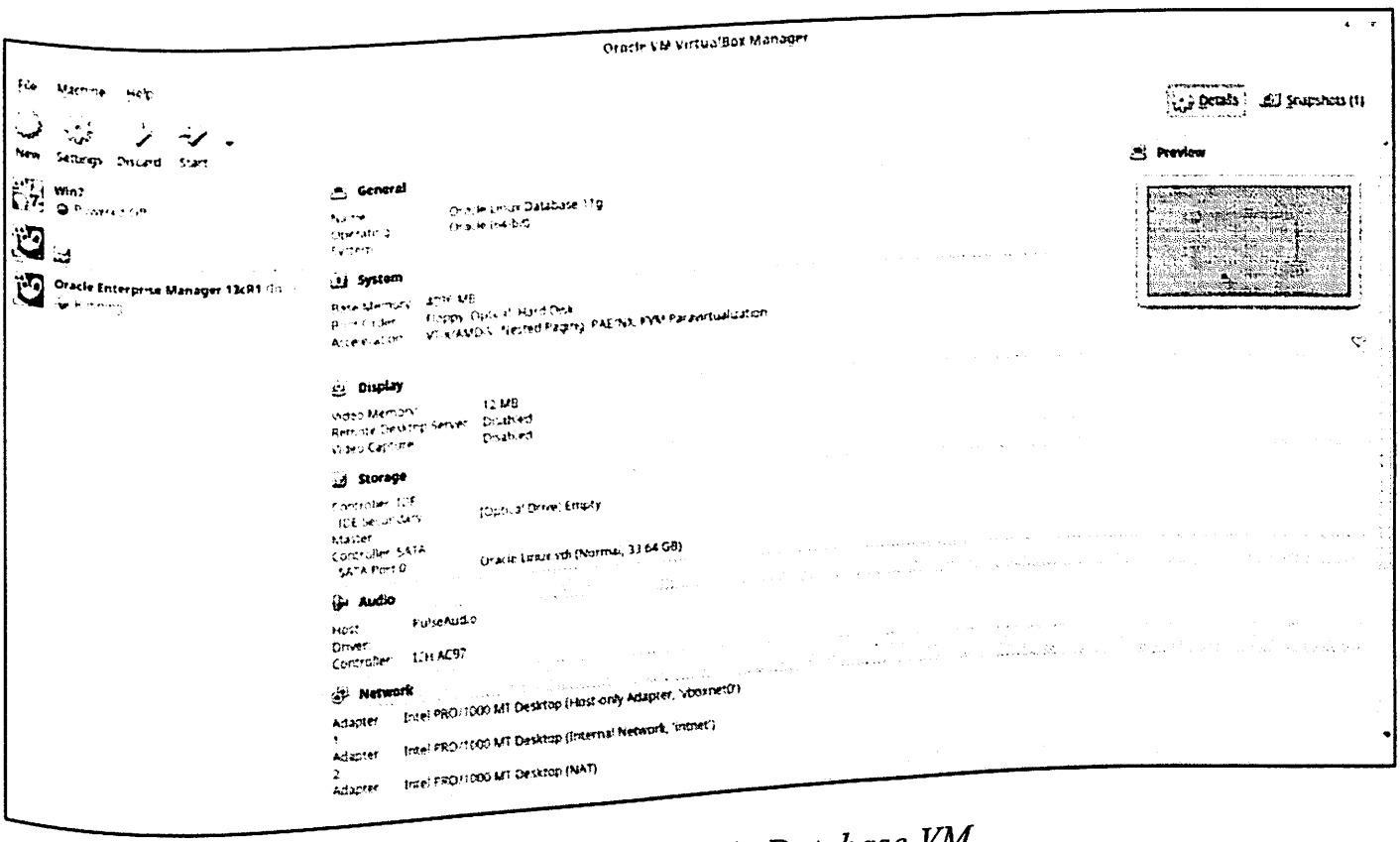

Figure 2.2 - Oracle Database VM.

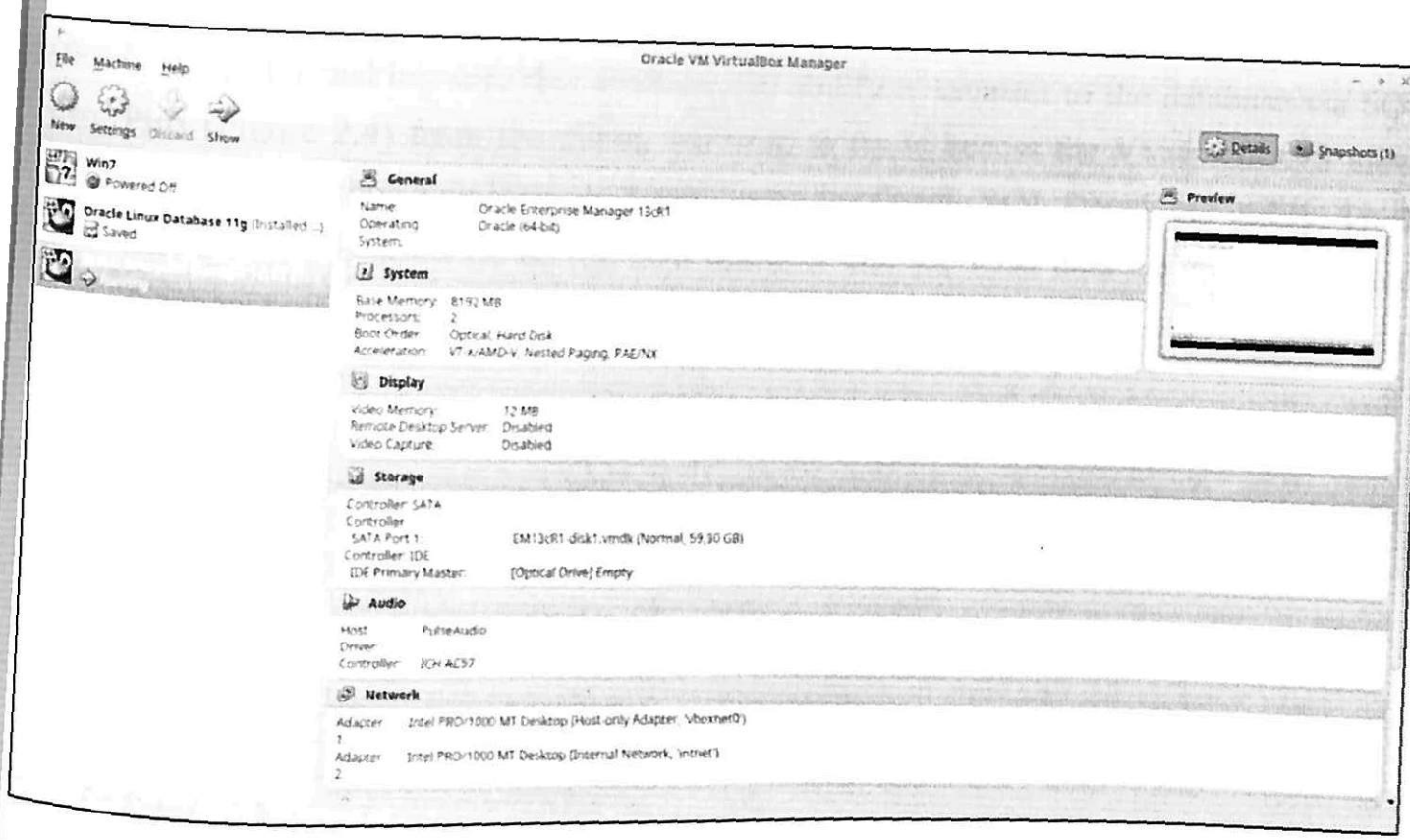

Figure 2.3 - Oracle Enterprise Manager VM.

2.2. Testing of existing monitoring tool. After starting and configuring all database servers, we check that the database is up and running, we can make requests to it and that the system does not issue any warnings, and we also check the basic download rates via Oracle Enterprise Manager.

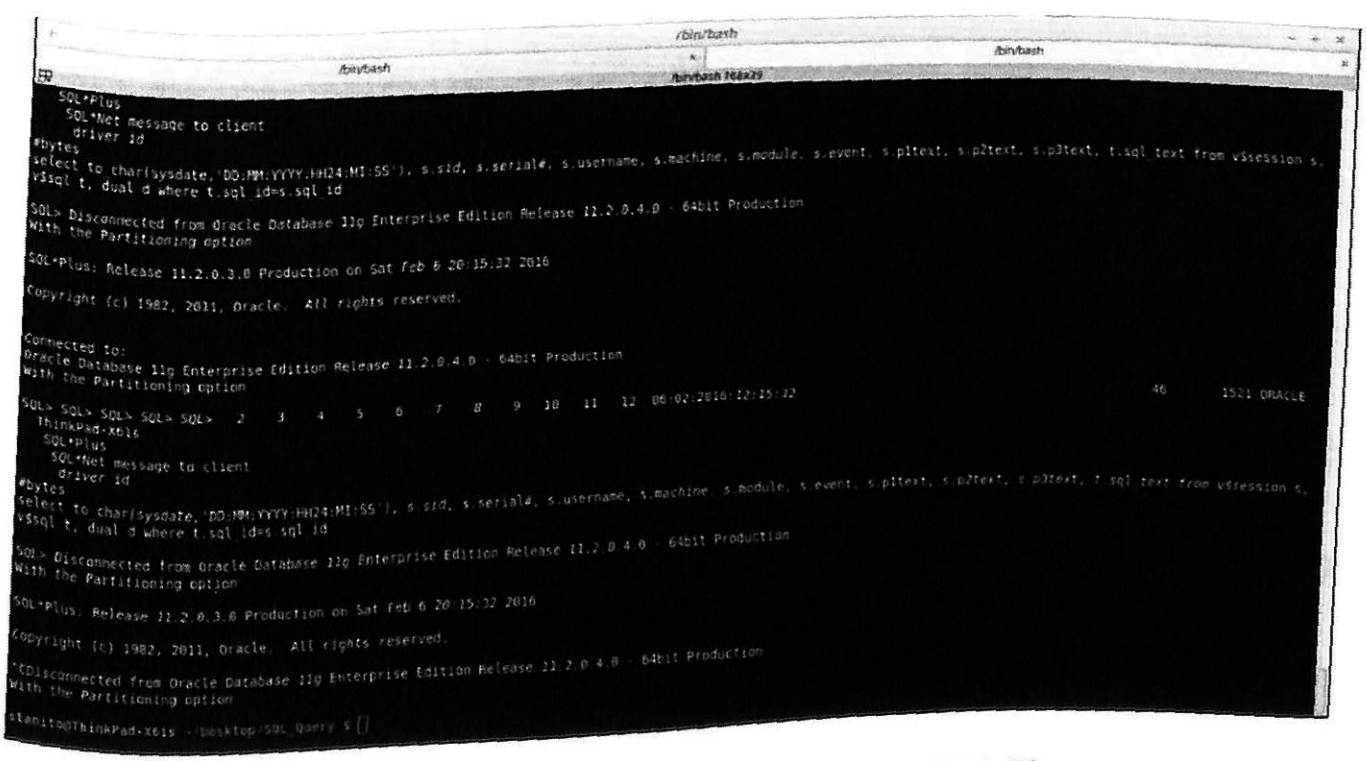

Figure 2.4 - Connect to the database via SQL Plus.

After making sure that we have the ability to connect to the database via SQL Plus (Figure 2.4) from the client, we need to try to access the V$SESSIONS view. The most obvious way is to get data through the Oracle SQL Developer utility. To do this, install this software on the host machine and see what data we can get through this view.

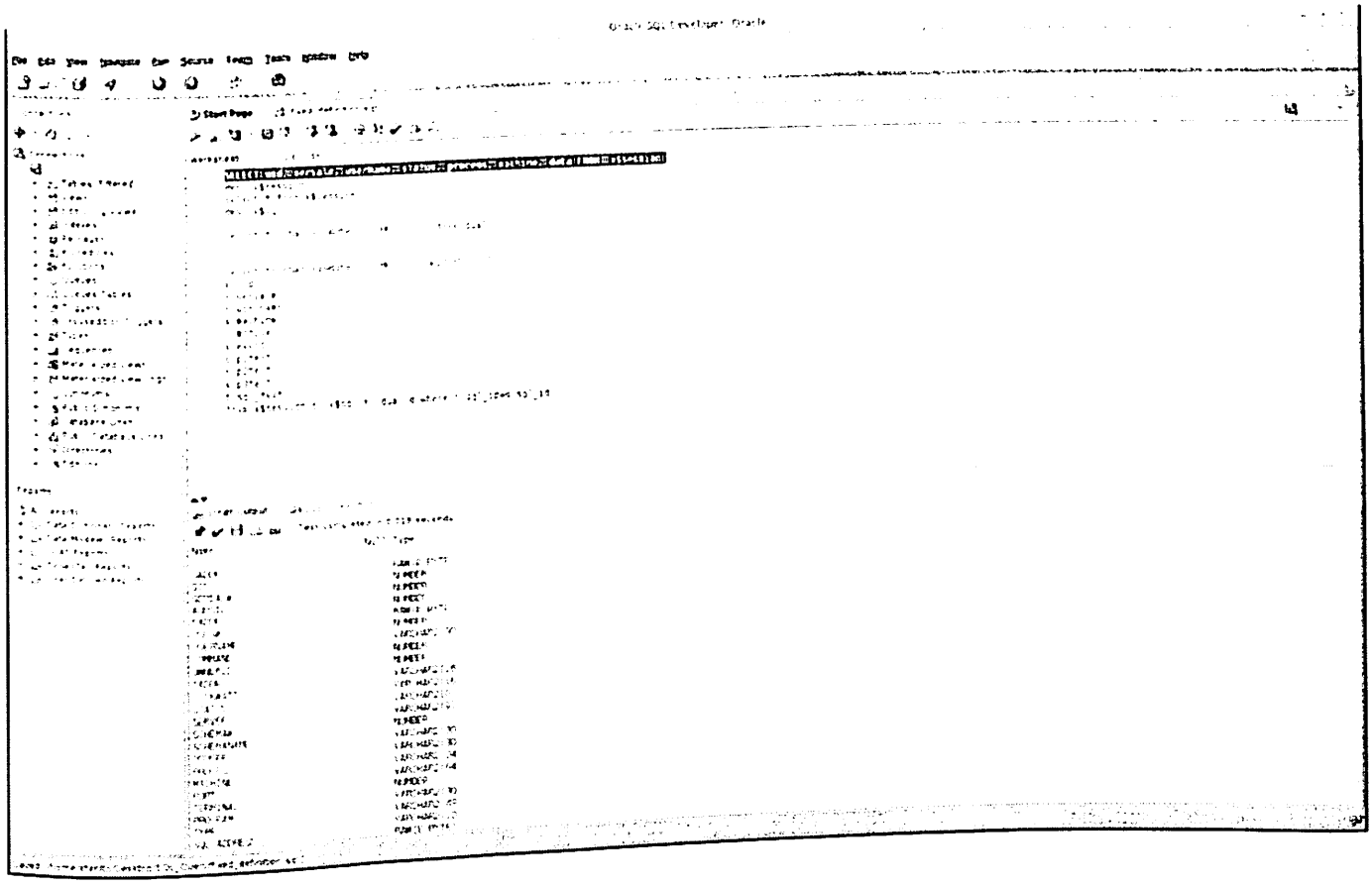

Figure 2.5 - Oracle SQL Developer

In the SQL query interpreter we will create several queries to the views and try to execute them.

- Request descv$session; we get the basic definitions of the columns in this table and the type of data that is stored in them;
- Select * from v$session; Returns to us all the data that is stored in this view since the instance of the database was started;

Next, we need to determine what information we want to get through the SQL query to collect statistics on the monitoring of the instance. For this, we turn to a detailed description in the Oracle documentation of all the information on this view.

2.3 A description of the dynamic representation of V$SESSION. This view displays information about each connected session to the database instance. Next in Figure 6 is a detailed description of each field, taken from the official documentation of Oracle 11g.

«The V$SESSION view enumerates numeric values that correspond to commands that can run during the current session. These values can appear in the column V$SESSION_COMMAND. They are also displayed in the SYS.AUDIT_ACTIONS panel of the data dictionary.»[4].

After analyzing the data available to us, we will try to form a query that will be used for monitoring. In principle, to obtain the basic statistics, it will be sufficient to use the following data:

- Sysdate - The time of the operation. Because time is stored in a system format, for convenient work with it we will convert it into a human-readable format;
- Sid - Process ID in the database;
- Serial # - Session number;
- Username - Username;
- Machine - The name of the host from which the operation is started;
- Module - The module through which the database accesses;
- Event - A resource or event that the process is waiting for;
- P1text, P2text, P3text - Expectations parameters.

After analyzing the operability of the query, create a shell script (Figure 2.6), which will be launched in an infinite loop, querying the database through a SQL query to provide these values for the operation of the database. This script is written in Bash and uses SQL Plus as a tool to transfer the query to the database.

```
#!/bin/bash
while :
do
sqlplus oracle/beer4Admin@TEST1 << EOF




EOF
done
```

Figure 2.6 - The text of the shell script for querying the database.

On the host machine, we start monitoring Enterprise Manager. We will make sure that the database is in a "quiet" state, that there is no load on it. This will be seen from the workload schedule, which should be in the state of zero. After that on the host machine we will launch our written shell script and see how the database reacts to such a frequent poll. After 10 minutes of work, stop the execution of our script. We must see that the load on the database has returned to its original state.

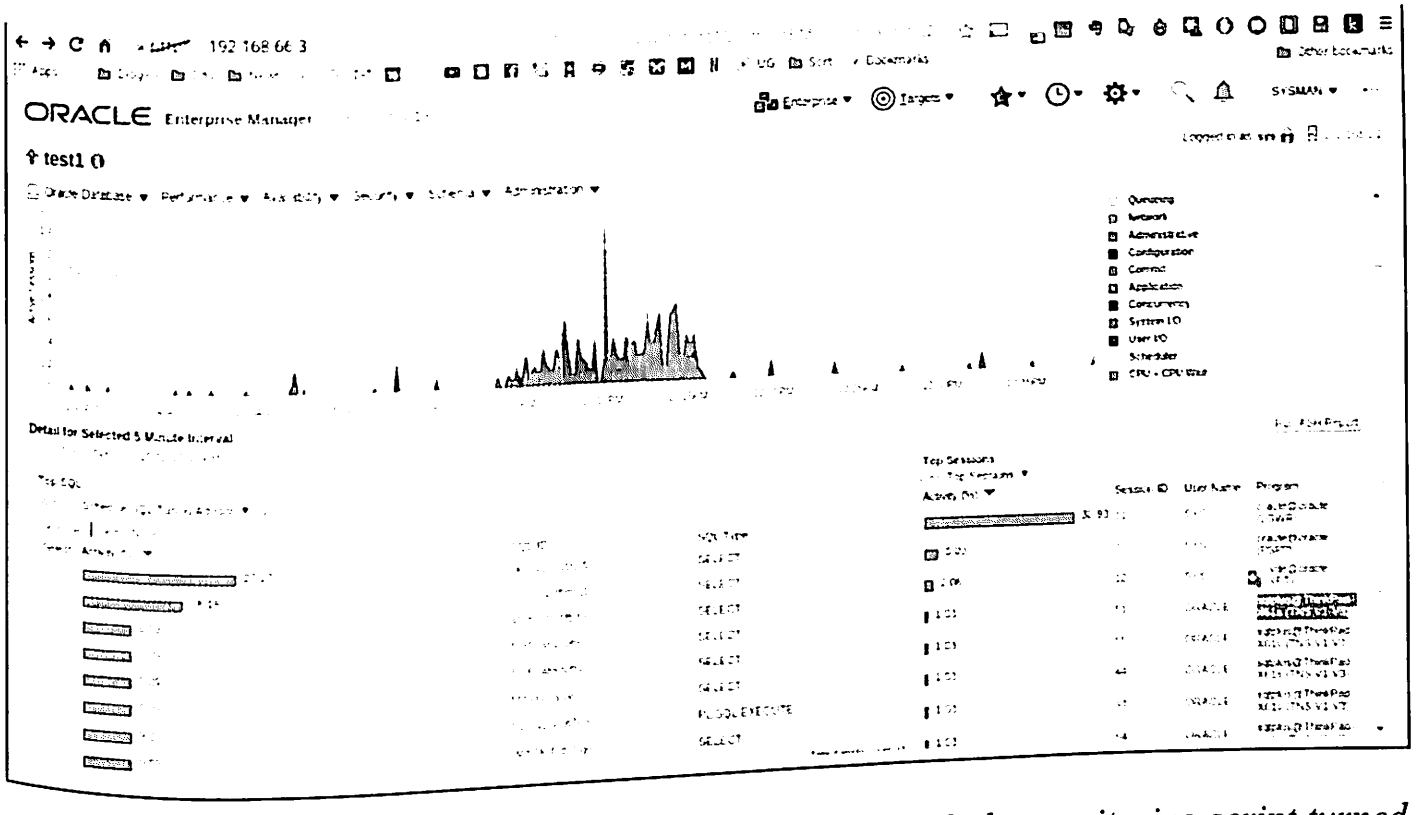

Figure 2.7 - The schedule for loading the database with the monitoring script turned off and on.

2.4 Results of testing standard monitoring tools. An example of polling a V$SESSION view (Figure 2.7) with a high intensity shows how a completely unloaded database is almost inoperative. In peaks, the load is about 80%. After disconnection, the load returns to zero again. From the legend of Enterprise Manager, we can see from which machine the requests were, what the query text was and what the database was using its precious resources.

On our 3.2 GHz Linux host, we could not get the speed of 11-line query execution more than 50 times per second to the V$SESSION view, provided that the database was not loaded and did not contain a lot of data. This example in practice demonstrates the inapplicability of using dynamic presentation data for frequent polling due to the ineffectiveness of processing SQL queries against these views.

Thus, we proved that SQL is not suitable for polling even small V$ views at a rate of 100 times per second. Our plain script demonstrates the fundamental limitation of the polling method with SQL. It is also important to pay attention to the fact that we do not do read operations from physical media, and parsing is performed once, and not a row-by-row selection, but a selection by an array.

3. SOLUTION AND TESTING OF THE ACCESS METHOD

The content of shared memory becomes understandable through access to fixed tables X$.

The basic procedure for obtaining information from the SGA can be divided into the following steps:

1. We choose the representation V$ from which we need to obtain data;
2. Find the base table X$ for the selected representation V$;
3. We establish the correspondence of the fields between the base table X$ and the representation V$;
4. Get the starting address of the base table X$ in the SGA segment;
5. Get the size of each record in the base table X$;
6. We get the number of records in the base table X$;
7. We get the displacement of each required field in the base table X$;
8. Get the starting address of the SGA segment.

With all this data, we can connect to the shared memory segment of the operating system and start reading metrics from fixed X$ tables.

3.1 The choice of the view of V$ and the base table for it in X$. Having studied the V$SESSION view, we learn that we can obtain the following information, which will allow:

- Get all user sessions;
- Get the status of query execution;
- Access to short-lived data;
- A special tool for diagnosing bottlenecks in performance;
- Ability to make frequent polls.

This will allow us to use this information as a special tool for diagnosing bottlenecks in performance. In doing so, we get the opportunity to make frequent polls.

The V$SESSION view, as well as the V$SESSION_WAIT table, will be very useful for obtaining information from the SGA. The description of all submissions should be sought in the official Oracle documentation for the required version of the product, due to the fact that from version to version of the view can be modified.

Knowing the fact that dynamic representations of V$ are constructed on the basis of fixed tables X$, we need to find the source text of the V$SESSION we are interested in.

In the V$FIXED_TABLE view, there is a TYPE field that contains information about the type of the object (View or Table). For example, V$PROCESS is described as a view and X$KSUPR as a table.

All V$ representations that are described in V$FIXED_TABLE are built on fixed X$ tables. To get a description of each view, you need to query V$FIXED_VIEW_DEFINITON.

In the query, must specify the name of view (Figure 3.1)

```
SQL> select view_definition from v$fixed_view_definition where
view_name='V$SESSION';
```

Figure 3.1 - Request to get the source text of the V $ SESSION representation.

As a result, we get the definition of fields from $FIXED_VIEW_DEFINITIONS for V$SESSION (Figure 3.2).

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
select
SADDR , SID , SERIAL# , USER# , USERNAME , COMMAND , OW
NERID, TADDR , LOCKWAIT , STATUS , SERVER , SCHEMA# , SCHEMANAME ,OSUSER , PROCE
SS , MACHINE , PORT , TERMINAL , PROGRAM , TYPE , SQL_ADDRESS , SQL_HASH_VALUE,
 SQL_ID, SQL_CHILD_NUMBER , SQL_EXEC_START, SQL_EXEC_ID, PREV_SQL_ADDR , PREV_HA
SH_VALUE , PREV_SQL_ID, PREV_CHILD_NUMBER , PREV_EXEC_START , PREV_EXEC_ID , PLS
QL_ENTRY_OBJECT_ID, PLSQL_ENTRY_SUBPROGRAM_ID, PLSQL_OBJECT_ID, PLSQL_SUBPROGRAM
_ID, MODULE , MODULE_HASH , ACTION , ACTION_HASH , CLIENT_INFO , FIXED_TABLE_SEQ
UENCE , ROW_WAIT_OBJ# , ROW_WAIT_FILE# , ROW_WAIT_BLOCK# , ROW_WAIT_ROW# , TOP_L
EVEL_CALL#, LOGON_TIME , LAST_CALL_ET , PDML_ENABLED , FAILOVER_TYPE , FAILOVE
R_METHOD , FAILED_OVER, RESOURCE_CONSUMER_GROUP, PDML_STATUS, PDDL_STATUS, PQ_ST
ATUS, CURRENT_QUEUE_DURATION, CLIENT_IDENTIFIER, BLOCKING_SESSION_STATUS,
BLOCKING_SESSION, FINAL_BLOCKING_SESSION_STATUS, FINAL_BLOCKING_INS
TANCE, FINAL_BLOCKING_SESSION, SEQ#, EVENT#,EVENT,P1TEXT,P1,P1RAW,P2TEXT,P2,P2RA
W, P3TEXT,P3,P3RAW,WAIT_CLASS_ID, WAIT_CLASS#,WAIT_CLASS,WAIT_TIME, SECONDS_IN_W
AIT,STATE,WAIT_TIME_MICRO,TIME_REMAINING_MICRO, TIME_SINCE_LAST_WAIT_MICRO,SERVI
CE_NAME, SQL_TRACE, SQL_TRACE_WAITS, SQL_TRACE_BINDS, SQL_TRACE_PLAN_STATS, SESS
ION_EDITION_ID, CREATOR_ADDR, CREATOR_SERIAL#, ECID
from
GV$SESSION
where
inst_id = USERENV('Instance')
```

Figure 3.2 - The result of the query to get the source text of the V$SESSION view.

From the data obtained, we see that V$SESSION is based on the GV$SESSION table.

Let's try to get similar data by looking in the dictionary view on the contents of the table GV$SESSION (Figure 3.3).

```
SQL> select view_definition from v$fixed_view_definition
where view_name='GV$SESSION';
```

Figure 3.3 - Request to get the source text of the GV$SESSION representation.

This will allow us to use this information as a special tool for diagnosing bottlenecks in performance. In doing so, we get the opportunity to make frequent polls.

The V$SESSION view, as well as the V$SESSION_WAIT table, will be very useful for obtaining information from the SGA. The description of all submissions should be sought in the official Oracle documentation for the required version of the product, due to the fact that from version to version of the view can be modified.
The result of the request to get the source text of the GV$SESSION representation (Appendix 1).

The double-based representation of the representations X$ → GV$ → V$ is due to the fact that data is written into the GV$ representation without decoding the values into a convenient form for the person, but convenient for Oracle's work. Further values are decoded by the dictionary and written to the representation V$.

Thus, we proved that SQL is not suitable for polling even small V$ views at a rate of 100 times per second. Our plain script demonstrates the fundamental limitation of the polling method with SQL. It is also important to pay attention to the fact that we do not do read operations from physical media, and parsing is performed once, and not a row-by-row selection, but a selection by an array.

As a result, we get the source text to get the data in V $. It will not be formatted. For greater clarity, we get the data of the definition of each field through the queries for determining each of the representations. Here is an abridged version, only with the fields of interest to us for the V$SESSION representation (Figure 3.5)

```
SQL> desc v$session

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SADDR                                              RAW(8)
 SID                                                NUMBER
 SERIAL#                                            NUMBER
 PADDR                                              RAW(8)
 USER#                                              NUMBER
 USERNAME                                           VARCHAR2(30)
 COMMAND                                            NUMBER
 LOCKWAIT                                           VARCHAR2(19)
 STATUS                                             VARCHAR2(8)
 SERVER                                             VARCHAR2(9)
 OSUSER                                             VARCHAR2(30)
 PROCESS                                            VARCHAR2(24)
 MACHINE                                            VARCHAR2(64)
 PORT                                               NUMBER
 TERMINAL                                           VARCHAR2(30)
 PROGRAM                                            VARCHAR2(48)
 TYPE                                               VARCHAR2(10)
 SQL_ADDRESS                                        RAW(8)
 MODULE                                             VARCHAR2(64)
 ACTION                                             VARCHAR2(64)
 CLIENT_INFO                                        VARCHAR2(64)
 LOGON_TIME                                         DATE
 SEQ#                                               NUMBER
 EVENT#                                             NUMBER
 EVENT                                              VARCHAR2(64)
 P1TEXT                                             VARCHAR2(64)
 P1                                                 NUMBER
 P2                                                 NUMBER
 P3                                                 NUMBER
 WAIT_CLASS                                         VARCHAR2(64)
 WAIT_TIME                                          NUMBER
 SECONDS_IN_WAIT                                    NUMBER
 STATE                                              VARCHAR2(19)
```

Figure 3.5 - Result of the request to receive the description of the V$SESSION.

As we can see, the dynamic table V$SESSION is based on another dynamic table GV$SESSION, which in turn takes data from the fixed representations X$KSUSE, X$KSLED and X$KSLWT.

The double-based representation of the representations X$ → GV$ → V$ is due to the fact that data is written into the GV$ representation without decoding the values into a convenient form for the person, but convenient for Oracle's work. Further values are decoded by the dictionary and written to the representation V$.

3.2 The ratio of the fields between the base table X$ and the view V$. Next, we need to relate the fields of the V$SESSION representation to the fields X$KSUSE, X$KSLED, X$KSLWT. Given that this information is not published by Oracle and the search on the Internet did not provide a ready solution, this operation had to be done manually, creating two identical requests to the view and a fixed table, combining fields with the same values. Because I knew for sure that in my test environment therewas only one session from the ThinkPad-x61s computer, then I used the machine name as a unique value in both queries (Table 3.1).

V$SESSION	X$ Field	X$ Table
#(decode)	ksspaflg	X$KSUSE
#(decode)	ksspaflg	X$KSUSE
SID	INDX	X$KSUSE
SERIAL	KSUSESER	X$KSUSE
USERNAME	KSUUDLNA	X$KSUSE
STATUSID	KSUSEIDL	X$KSUSE
STATUS (ksspaflg, ksspaflg)	#(decode)	X$KSUSE
SADDR	ADDR	X$KSUSE
PADDR	KSSPAOWN	X$KSUSE
OSUSER	KSUSEUNM	X$KSUSE
MACHINE	KSUSEMNM	X$KSUSE
PROGRAM	KSUSEAPP	X$KSUSE
MODULE	KSUSEPNM	X$KSUSE
LOGON_TIME	KSUSELTM	X$KSUSE
SQL_EXEC_START	KSUSESESTA	X$KSUSE
SERVICE_NAME	KSUSESVC	X$KSUSE
EVENT	KSLEDNAM	X$KSLED
P1	KSUSSP1	X$SUSECST
P2	KSUSSP2	X$SUSECST
P3	KSUSSP3	X$SUSECST
SEQ#	KSUSESEQ	X$SUSECST

Table 3.1.The relationship between the fields of the dynamic representation of V$SESSION and the base tables X$.

Combining these data with the previously obtained description of the V $ SESSION representation, we end up with a fairly clear table, which will describe which tables and fields X $ the V $ SESSION consists of (Appendix 2). The source text of the GV$SESSION representation with the correlations of the field names in the base tables X$.

Corresponding fields, I check that requests to X $ and V $ return the same values (Figure 3.7).

```
SQL> select STATUS, STATUS, OSUSER, MACHINE, MODULE, PROGRAM, LOGON_TIME,
SERVICE_NAME
from v$session
where MACHINE like 'ThinkPad-X61s';
--------------------------------------------------------------------------
INACTIVE INACTIVE stanito ThinkPad-X61s SQL Developer SQL Developer 05-APR-16
SYS$USERS


SQL> select KSSPAFLG, KSUSEFLG, KSUSEUNM, KSUSEMNM, KSUSEPNM, KSUSEAPP, KSUSELTM,
KSUSESVC
from x$ksuse
where KSUSEMNM like 'ThinkPad-X61s';
--------------------------------------------------------------------------
   stanito ThinkPad-X61s SQL Developer SQL Developer 05-APR-16 SYS$USERS
```

Figure 3.7 - Comparison of queries to V$SESSION and X$KSUSE.

As a result, we see that to get username, we have to read the field kssuudlna from X$KSUSE. And to get the status field, we need to read the fieldsksuseidl, ksuseflg and then do some small calculations.

After we got a list of the correspondences of the fields V$ and X$, we know which fields in X$KSUSE we need to get information. Next, we'll try to find out exactly where in the RAM are the data fields.

3.3 Getting the starting address of the base table X$ in the SGA segment.

3.3.1 Storage of information in the SGA

- Most of the data from the SGA is not readable;
- Table X$KSMMEM is a value storage card in the SGA area;
- Access to information occurs through access to tables X $;
- Useful and necessary information is exported;
- The exported information becomes available through the V $ views.

3.3.2 Getting the initial SGA address. You can find out the basic address of the SGA via a request to the fixed table X$KSMMEM (Kernel Service Memory Management SGA memory). This table is essentially the entire SGA as a whole, so choosing addr for the first line, we get the desired value for the starting address of the SGA as a whole (Figure 3.8).

```
SQL> select addr from x$ksmmem where rownum=1;

ADDR
--------
20000000
```

Figure 3.8 - Request to get the start address of the segment.

3.3.3 Location of SGA data in RAM. Essentially, the basic tables are programs written in C that are run with the database instance. Oracle takes the C-structure of the table X$KSMMEM, which describes the SGA, and stores it into the shared memory of the segment (Figure 3.9).

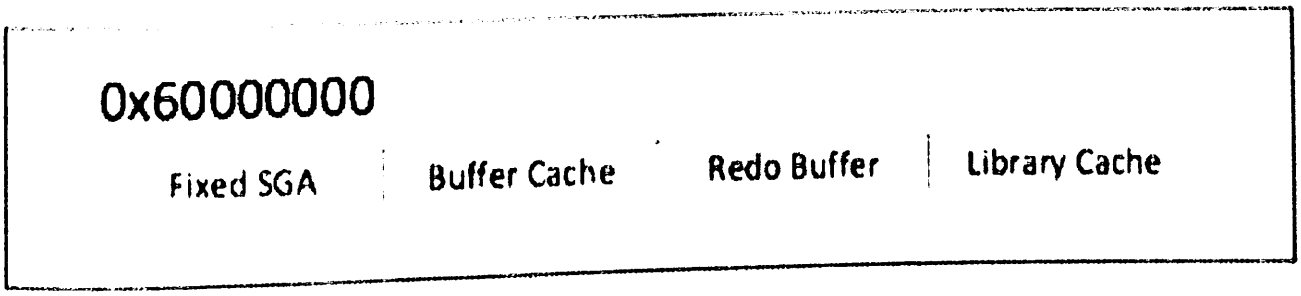

Figure 3.9 - Addressing memory in the SGA area.

3.3.4 The initial address of the tables X$ and their correlation with the view V$. The initial address of the X$ table is recognized by selecting the minimum value of addr in the table. Due to the fact that the starting address of the SGA can be shifted in RAM, the start address information of each of the X$ tables is not constant and they need to be clarified after each instance of the database is started.

In fact, the initial address of each table can be obtained by analogy with how we got the initial address of the SGA itself. To do this, we choose the minimal addr for each of the tables X$ of interest to us (Figure 3.10).

```
SQL> select min(addr) from x$ksuse;
```

Figure 3.10 - Getting the starting address of the table X $ KSUSE.

Next, I'll try to find out the initial addresses of the base tables for my Oracle test database. These addresses will be relevant only until the instance is in the active state.

All addresses will be displayed in hexadecimal notation (Table 3.2).

Начальный Адрес	X$	V$
9A034020	**X$KSURU**	
9A034020	**X$KSUSE**	V$SESSION
9A034020	**X$KSUSECST**	V$SESSION_WAIT
7F4B1439B5E8	**X$KSUSESTA**	V$SESSION_STAT

Table 3.2.Example of obtaining start addresses of base tables X$.

From the example, we see that different tables have the same starting address in memory. The reason lies in the fact that the fields of each of the tables in the RAM

are stored not strictly one after the other, but are placed in a mixed order. And initial fields with a zero indent are computed fields, which leads to the fact that some tables have the same starting address. For precise positioning, Oracle stores in its database the initial memory address of each of the fields in the base table.

3.3.5 Getting the size of each record (line) in the base table X$. In the structure of a fixed table, all rows are arranged sequentially. Therefore, knowing the starting address of two adjacent rows, you can find the size of one line by making the subtraction of the smaller address smaller. The size of the string will be bytes

```
SQL> create or replace function to_dec
( p_str in varchar2,
 p_from_base in number default 16 ) return number
is
 l_num   number default 0;
 l_hex   varchar2(16) default '0123456789ABCDEF';
begin
 if ( p_str is null or p_from_base is null )
 then
  return null;
 end if;
 for i in 1 .. length(p_str) loop
  l_num := l_num * p_from_base + instr(l_hex,upper(substr(p_str,i,1)))-
1;
 end loop;
 return l_num;
end to_dec;
/
show errors

-- find size of each record of session in KSUSE (Decimel)
SQL> select ((to_dec(f.addr)-to_dec(e.addr))) row_size
from (select addr from x$ksuse where rownum < 2)f,
(select min(addr) addr from x$ksuse where rownum < 3)e;

ROW_SIZE
----------
     1512
```

Figure 3.11 - Query and result of the query for obtaining the length of one line in the base table X$KSUSE.

3.4 Getting the number of records in the base table X$. Each base table consists of a fixed number of records (rows) that correspond to the number of valid sessions in the database instance. This number is set to the Oracle configuration file and the value can not be changed until the database instance is restarted. It turns out, after counting the exact number of sessions, we will know exactly how many rows will be located in the SGA. To do this, execute the SQL query:

```
SQL> select count(addr) from sys.x$ksuse;
```

Figure 3.12 - Query and result of the query for obtaining the number of rows in the base table X$KSUSE.

3.5 Obtaining the displacement of the required fields in the base table X$. The offset of the fields is considered from the initial address of the required table X$. All data stored in fields with an Offset of 0 are computed fields and are essentially stored outside the SGA area. You can get this data from special tables X$KQFCO and X$KQFTA, which contains information about the indentation and size of the field, respectively. The following is an example of the query and the result of the execution for getting the indentation of the fields of the table X$KSUSE. Here is a shortened version with fields of interest to us, the full result of the query is in Figure 3.13.

FIELD_NAME	OFFSET	SIZE
ADDR	0	8
INDX	8	4
KSUSEACT	0	4
KSUSEAPH	0	64
KSUSEAPP	124	4
KSUUDUID	192	30
KSUUDNAM	168	30
KSUUDLNA	204	4
KSUUDFLG	256	4
KSUUDSID	260	30
KSUUDSNA	5824	2
KSUSESEQ	5826	2
KSUSEOPC	5832	8
KSUSEP1	5840	8
KSUSEP2	5848	6
KSUSEP3	5922	2
KSUSESER	5936	4
KSUSEFLG	5952	8
KSQPSWAT	6072	6
KSUSESQL	6104	4
KSUSESEID	6160	7
KSUSEPESTA	6200	32
KSUSEUNM	6240	64
KSUSEMNM	6312	4
KSUSEMNP	6656	2
KSUSEQCSID	6958	4
KSUSEFLG2		

Figure 3.13 - Query and result of an indentation request for each field in the base table X $ KSUSE.

As we see from the previous query in SGA between indent 1 and 6000+ (the end of the Fixed Area segment) there is an unoccupied space. In this address space there can be:

- Other calculated tables, such as V $ SESSION_WAIT, V $ SESSION_STAT, etc .;
- Unallocated space for future Oracle database implementations.

3.6 Getting the start address of the SGA segment. At the same time, it is possible to obtain intermixed SGA segments. The segments are stored in ascending numbers of shmid identifiers. This is done in order to be able to create large shared memory, in systems where for some reason the size is limited, and also to be able to work and increase the dynamic areas of the SGA during the operation of the instance. This change was introduced in Oracle 11g. You can not change the partition size during the instance operation, the size is static.

To get the key and shared memory identifier, you need to use the ipcs operating system call, which will list all the segments contained in the RAM:

```
[oracle@oracle ~]$ ipcs -a

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 18055168   oracle     600        524288     2          dest
0x00000000 18087937   oracle     600        4194304    2          dest
0x00000000 22872066   oracle     600        524288     2          dest
0x00000000 22511619   oracle     600        31248      2          dest
0x00000000 18481156   oracle     640        12582912   37
0x00000000 18513925   oracle     640        977272832  37
0xaa5ccb7c 18546694   oracle     600        2097152    37
0x00000000 18579463   oracle     600        2097152    2          dest
0x00000000 18710536   oracle     600        4194304    2          dest
```

Figure 3.14 - Getting the list of shared memory segments in the operating system.

Also, Oracle has a sysresv utility that can show which segments are used by the running instance of the database on the local computer. Data about the name of the instance this utility takes through the variable environment of the operating system $ {ORACLE_SID} (Figure 3.15).

```
[oracle@oracle ~]$ sysresv

IPC Resources for ORACLE_SID "test1" :
Shared Memory:
ID              KEY
18481156        0x00000000
18513925        0x00000000
18546694        0xaa5ccb7c
Semaphores:
ID              KEY
622592          0x1c190dd0
Oracle Instance alive for sid "test1"
```

Figure 3.15 - Getting the list of shared memory segments used by the Oracle instance, through the Oracle "sysresv" utility.

Knowing the address of the initial SGA segment and its size, you can calculate the start address of the next section, which will have the next shmid ascending.

To do this, we translate the segment size to 16 number system and add the starting address and segment size:

Start address = 0x600000000

Segment size = 12582912 = 0xC00000

600000000 + 0xC00000 = 60c00000.

Accordingly, the initial address of the 2 segment with SHMID = 18513925 will be 0x60c00000

By the same algorithm, subsequent segments addresses are calculated.

3.7 Example of location of stored information in shared memory SGA. The location of the SGA segments in memory is usually consistent. But if several instances start simultaneously, there is a possibility that the segments from one instance will not be strictly located one after the other (Figure 3.16).

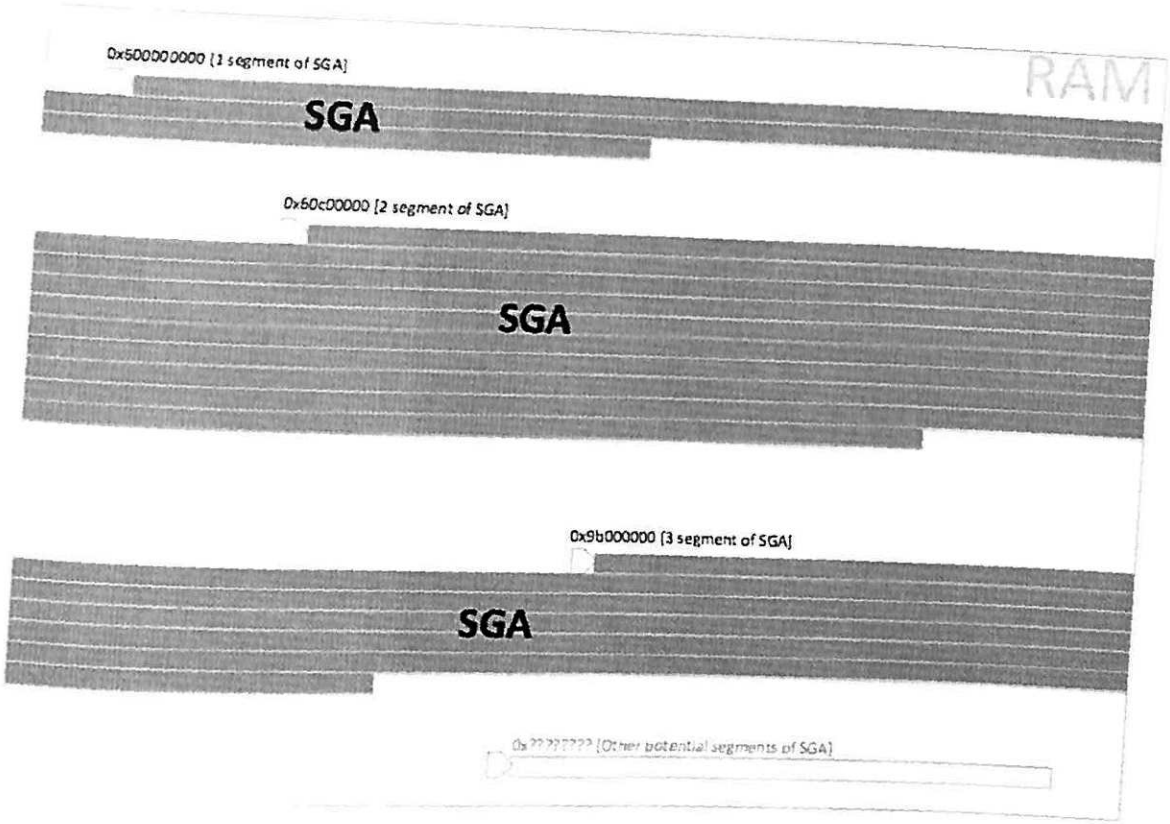

Figure 3.16 - Using RAM to store SGA.

SGA memory sections are arranged in the specified order in RAM: Fixed Area → Buffer Cache → Shared Pool → Log Buffer (Figure 3.17).

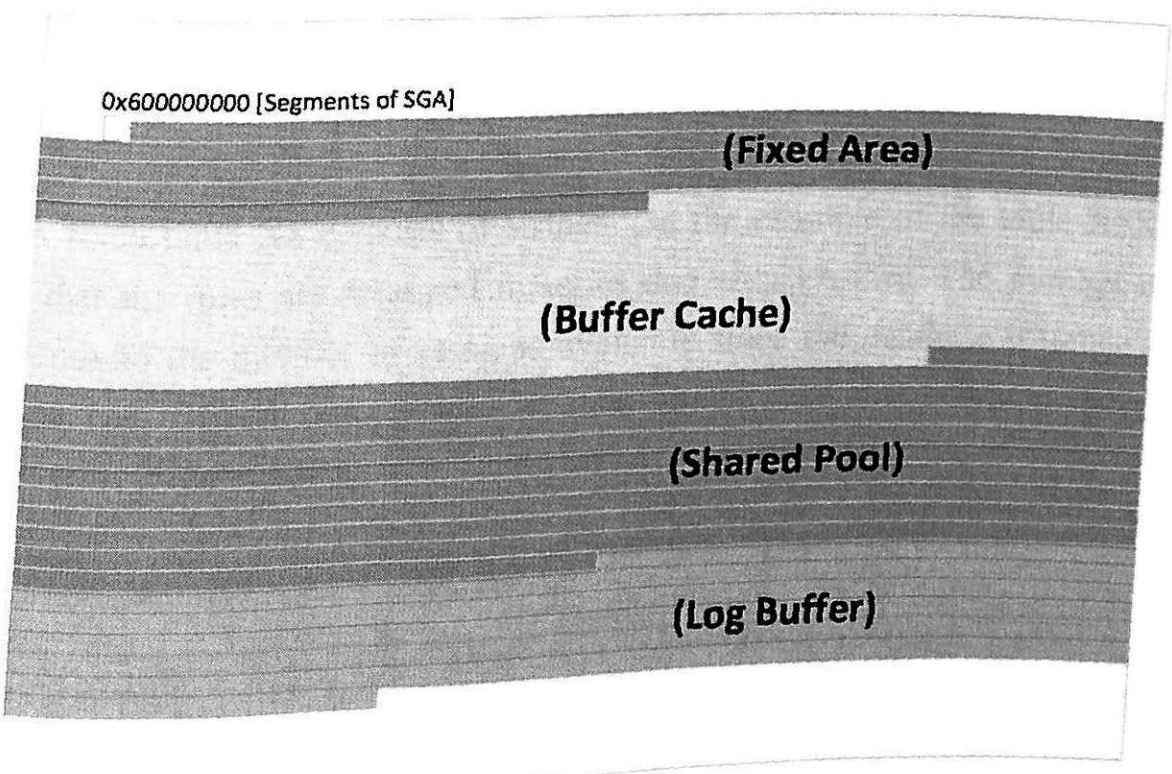

Figure 3.17 - Graphical location of memory regions in SGA.

Fixed tables are stored in a fixed section of SGA memory. This section contains a lot of X$ tables, each of which has its own start address, which is found when the database instance is started (Figure 3.18).

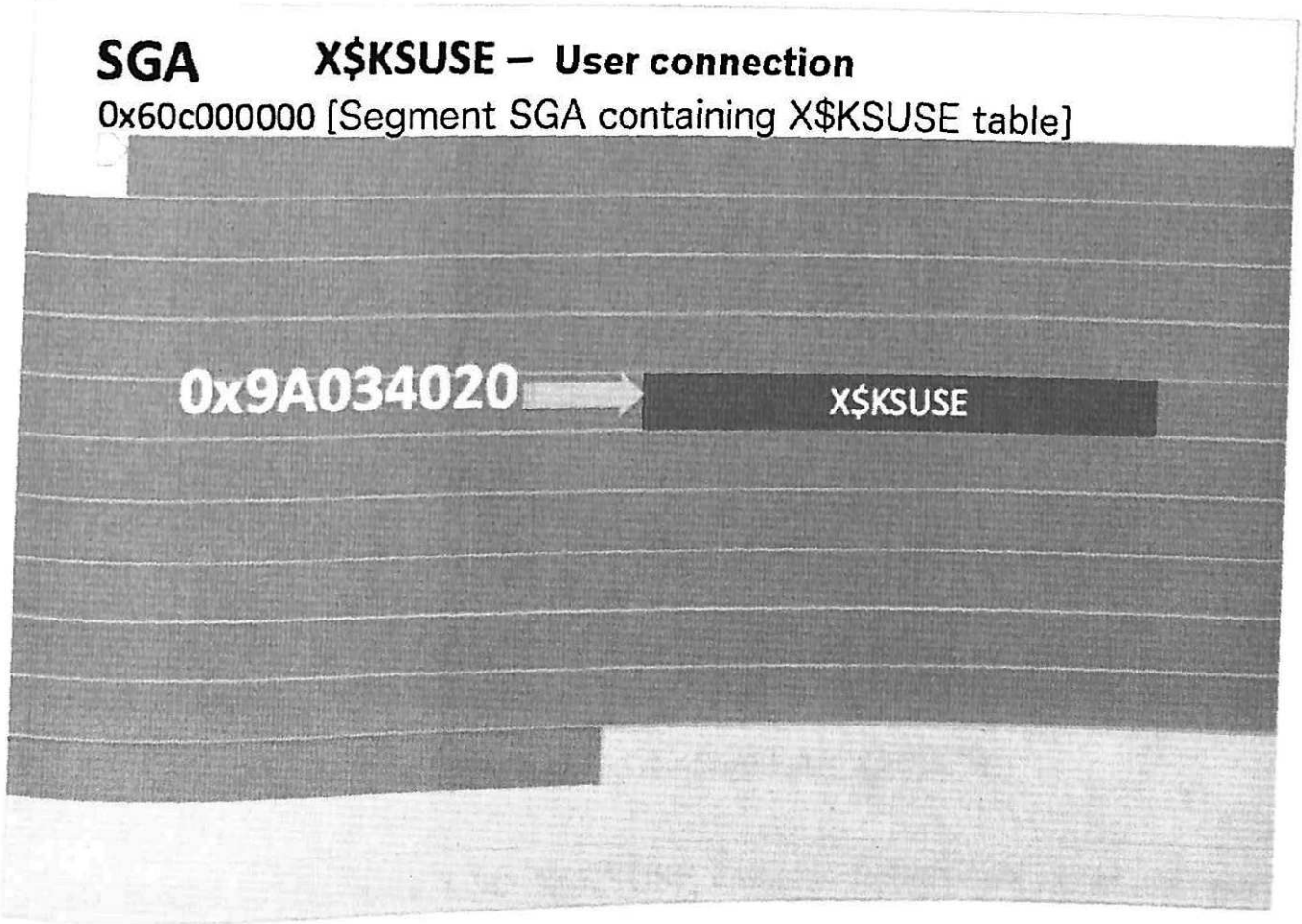

Figure 3.18 - Fixed memory partition.

A fixed table X$ consists of strings. On the example of the table X$KSUSE, we see that the rows are arranged in series one after another. The number of rows corresponds to the number of sessions available since the database instance started. Each line has a fixed size, which depends on the version of Oracle that we are working with (Figure 3.19).

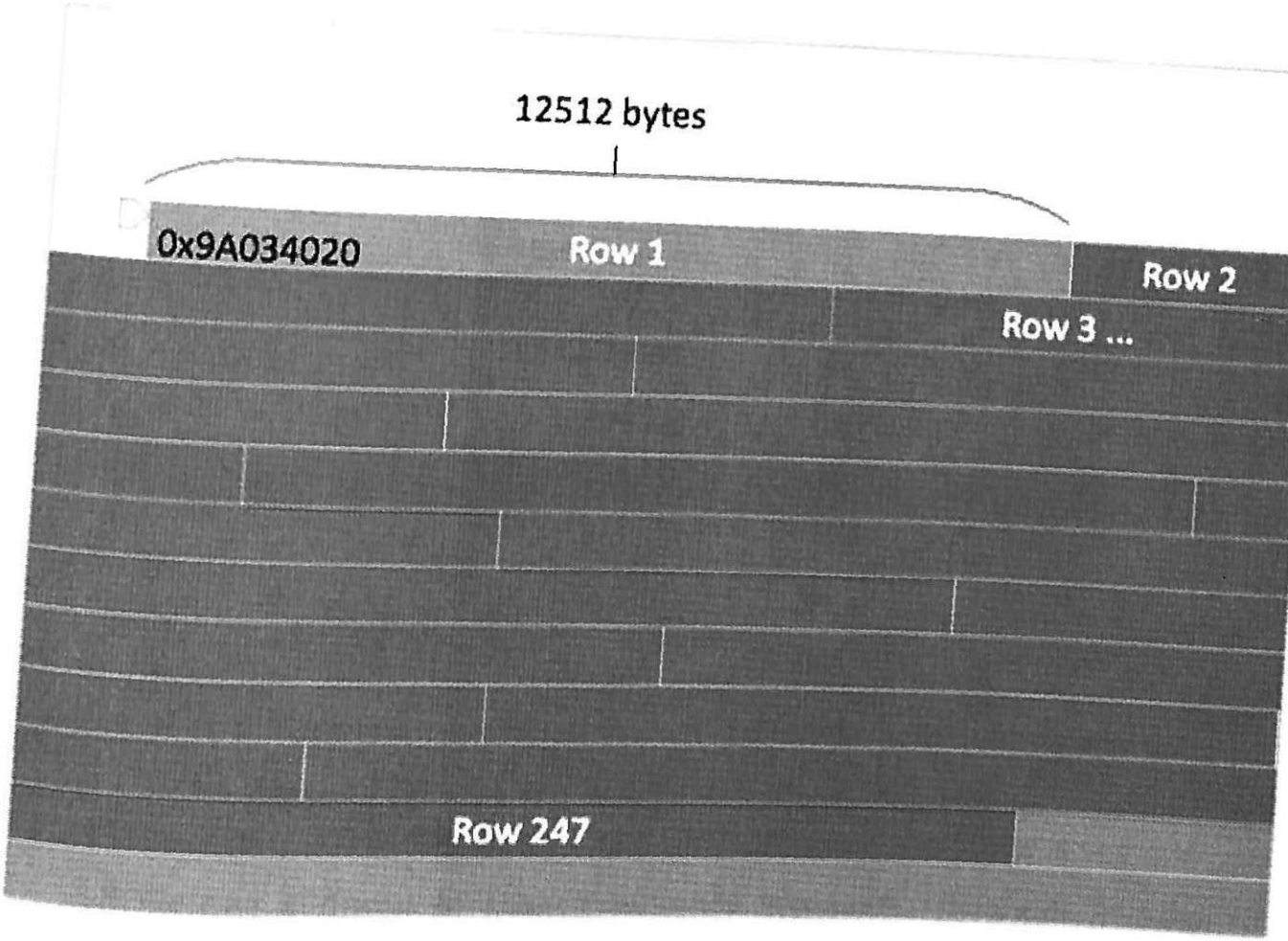

Figure 3.19 - Structure of the base table X$KSUSE.

Each line of the fixed table X$KSUSE contains information about the user session. Each record has a fixed size and location in the line, which is calculated by indenting the starting address of the string (Figure 3.20).

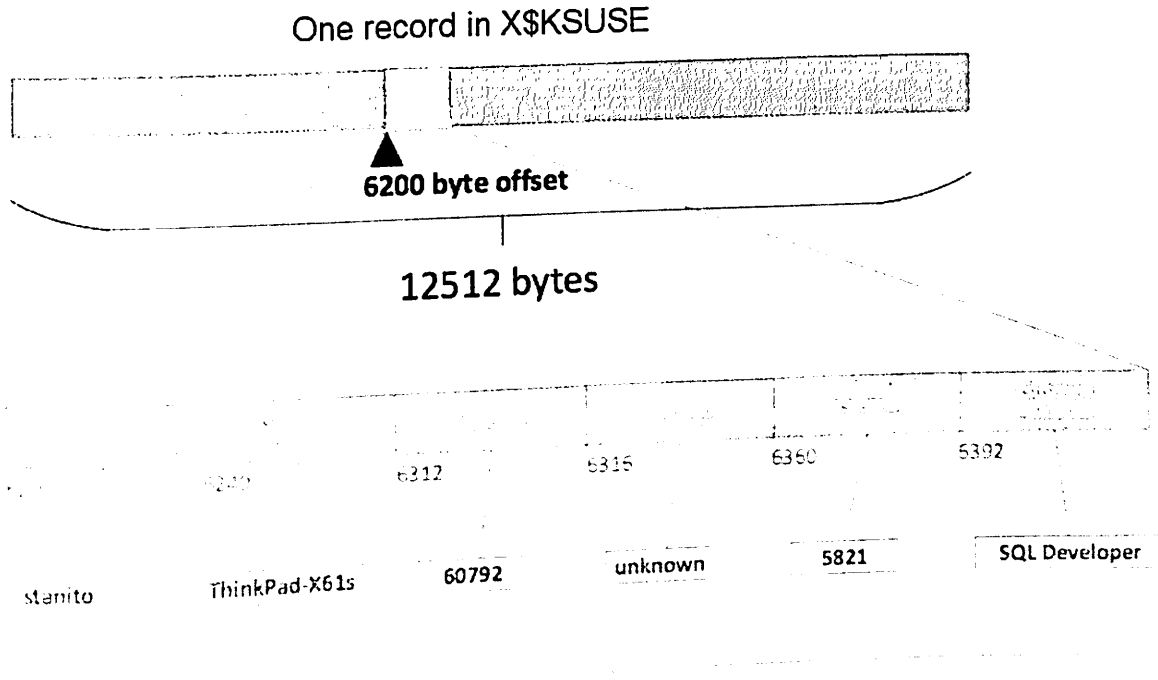

Figure 3.20 - Storing data in the record X$KSUSE.

3.8 Getting a dump of shared memory segments. To obtain a dump of segments, you should use the undocumented Oracle diagnostic utility - Oradebug. This diagnostic tool is used by Oracle Support to diagnose problems with remote support. But on the Internet there is information about some of its functionality, available on the link: [http://www.orafaq.com/papers/oradebug.pdf]

The process of getting the SGA memory dump from the current running instance will look like this:

4. IMPLEMENTATION OF THE PROGRAM

4.1 Repository

Link to the actual version of the program, hosted on GitHub: https://github.com/yerkulan/ORACLE_MONITOR/blob/master/oracle_monitor_sga.py

4.2 Work algorithm.

1. At startup, the program creates a connection to the Oracle database via the Python cx_Oracle module (Figure 4.1). The method of connection with OSAUTH is used. This allows you to connect to the local host as SYSDBA without using a password. The instance name is taken through the existing variables in ORACLE_SID and ORACLE_HOME (Figure 4.2).

2. Queries are made to the database to obtain information about (Figure 4.3):

- Address of the KSUSE table;
- Starting address of SGA
- Number of rows in the table KSUSE
- Length of each line
- Indent for the field Ksspaflg
- Indent for the field Ksuseflg
- Indent for the Serial field
- Indent for the Username field
- Indent for the Machinename field
- Indent for the Statusid field
- Start addresses of all lines
- Getting the sequence numbers of the lines
- Indent for the Sequence field
- Indent for the Event field
- Indent for field P1
- Indent for the P2 field
- Indent for field P3

3. Getting through the utility of the operating system ipcs all segments of shared memory, with the information about the identifiers SHMID, the starting address of the segment and its size.

4. Search for a segment whose address space includes the address of the KSUSE table

5. We make a reading from the found segment in the loop. We use the previously obtained data on the initial address of the line and indentation to obtain data from the fields of interest.

6. We combine data (decoding) from different tables. We substitute the dictionary values in the fields that contain the key fields of other tables.

7. Write information to a text file

8. At the end of the cycle, it catches the completion signals (SIGKILL, SEGFAULT ...)

9. End the program.

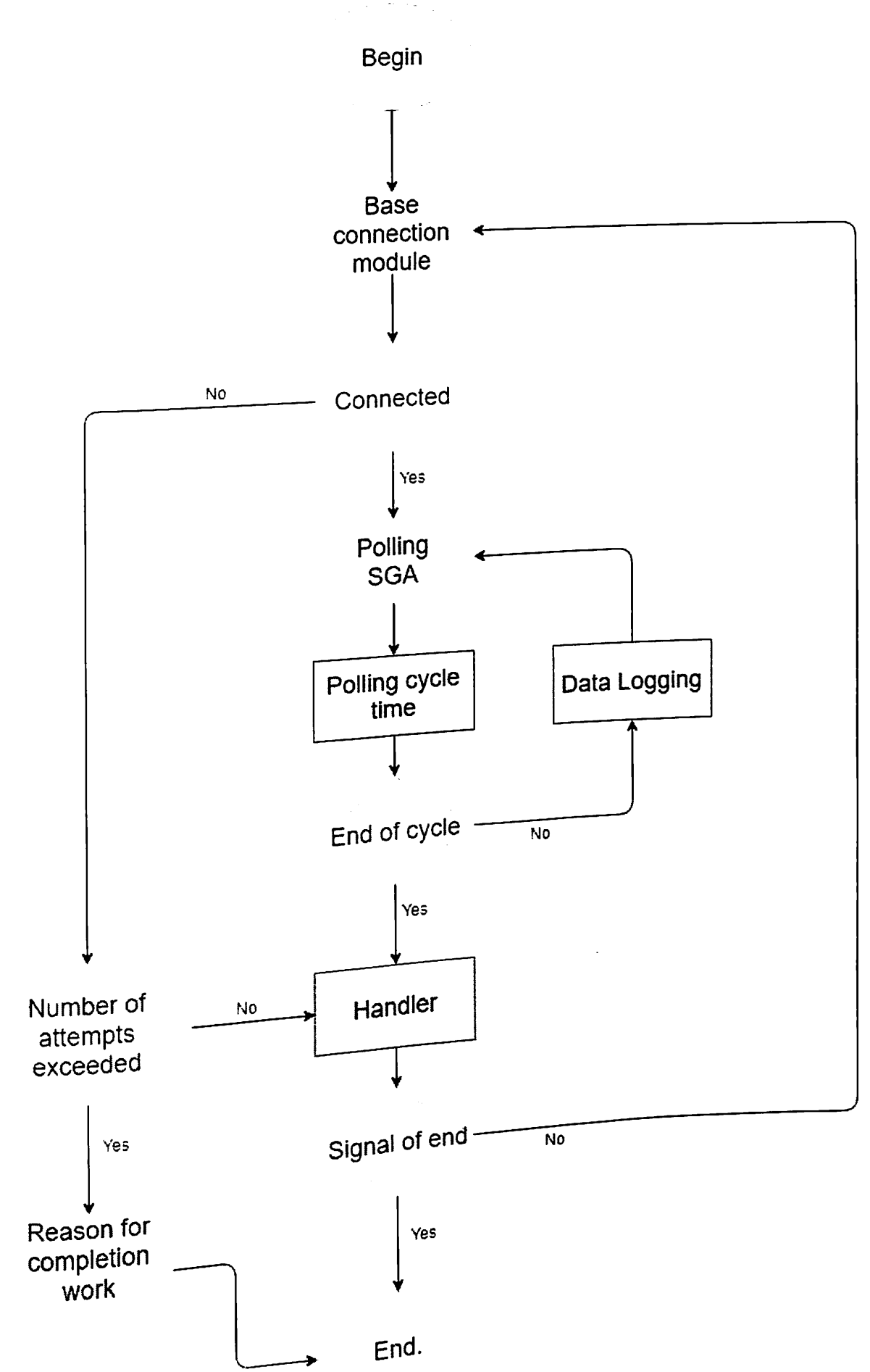

Figure 4.1 - The algorithm of the program reading data from the SGA.

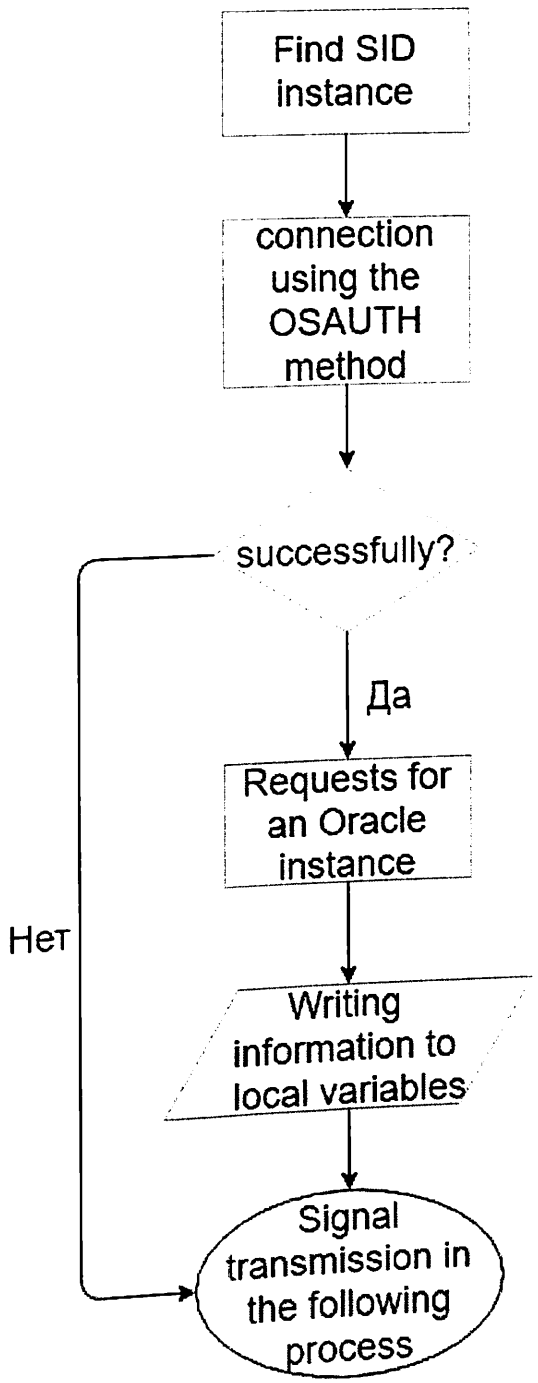

Figure 4.2 - The algorithm of the program "Base connection module"

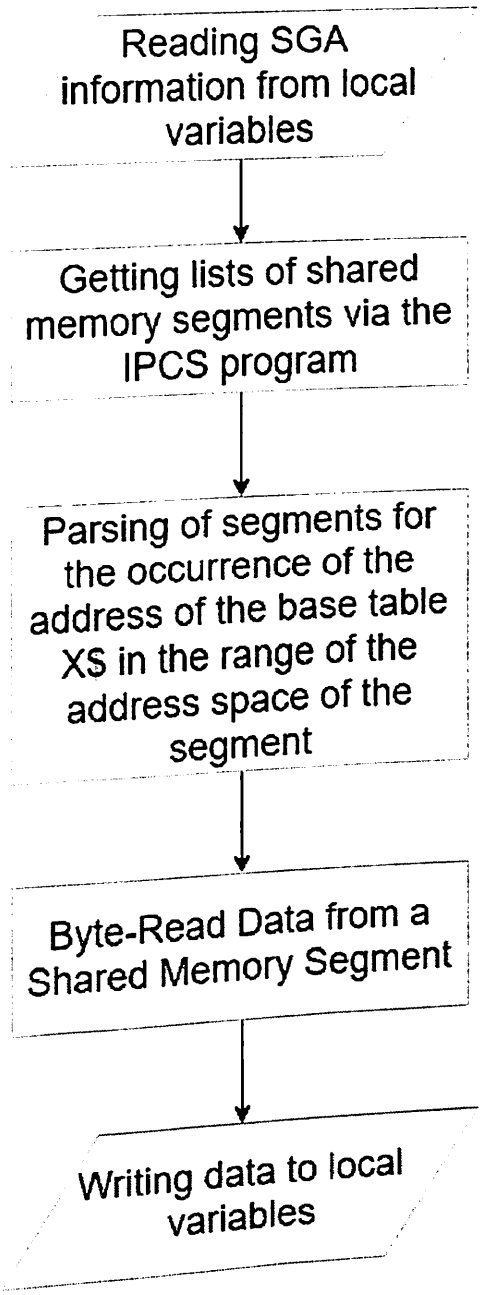

Figure 4.3 - The algorithm of the program "Polling SGA"

CONCLUSION

The purpose of this work was to develop a method for low-level monitoring of the Oracle DBMS, which can be used in an abnormal situation in the operation of a database instance. We can say that the goals and objectives set in this paper have been achieved.

In the course of the work, existing software tools and methods for obtaining performance metrics from the Oracle DBMS were analyzed and tested. The reason for the unusability of the use of existing Oracle database monitoring software products was proved for abnormal operation of the database instance.

A method of low-level interrogation of performance metrics from memory was developed, which was tested in practice and confirmed its suitability for use. Also, a positive property of this method was revealed, that it does not create a parasitic load on the operation of the DBMS and to a lesser extent affects the reliability of the information obtained.

In the course of the study, the existing techniques for obtaining performance metrics from the shared memory of an Oracle instance and obtaining a data storage structure from the database itself were studied and adapted for use in the method.

To create the method, the tasks were studying the structure of data storage in the Oracle architecture, gaining access to shared memory through the operating system mechanisms, bypassing the use of the database instance, and dynamically obtaining the storage structure of information in shared SGA memory. Based on this method, a prototype program was implemented to access performance metrics, the results of which were compared with the results of polling the instance through a standard access method that uses SQL queries. Summarizing, we can say that this method can find its application in the implementation of the module to existing monitoring systems, which can be used for non-routine behavior of the DBMS, which will allow administrators to respond quickly to the current situation and have up-to-date information for analyzing the operation of the Oracle database.

REFERENCES

1. Petrelevich Sergey, Review of the logical structure of the database: [electronic document] - (http://www.orahome.ru/ora-arch/36).
2. Tarakanov OV, Modified model of query processing in distributed databases: [electronic document] - (http://www.swsys.ru/index.php?page=article&id=3761)
3. Introduction to Oracle SQL [Text] // Vladimir Przhiyalkovsky - Binom. Laboratory of Knowledge, Internet University of Information Technology, 2011. - 320 c .;
4. Oracle. Optimizing performance. [Text] // Millsap K., Holt D. - St. Petersburg: Symbol Plus, 2006. - 464 p .;
5. Wikipedia electronic library, Oracle System Global Area article: [electronic document] - (https://en.wikipedia.org/wiki/System Global Region).
6. Dynamic Oracle Performance Views: [electronic document] - (http://oracle-admin.ru/dinamicheskie-predstavleniya-proizvoditelnosti.html).
7. Egor N., Definition of fixed representations .: [electronic document] - (http://www.oracledba.ru)
8. Expert One-on-One Oracle, 1st Edition [Text] // Thomas Kyte - Wrox Press Ltd, 2002. - 1427 p .;
9. Sergey Zhilin, Oracle's expectations events: [electronic document] - (http://alldba.ru/index.php/novosti/38-subd/oracle-database/417-sobytiya-ozhidaniya-oracle-a-e)
10. Oracle memory structures. Article: [electronic document] - (http://oracle-dba.ru/docs/architecture/memory/)
11. Oracle data types: [electronic document] - (http://www.interface.ru/fset.asp?Url=/oracle/ora7/ora7a04a.htm)
12. Emergency monitoring in Oracle Enterprise Manager Cloud Control 12c: [electronic document] - (https://oracle-base.com/articles/12c/emergency-monitoring-em12c)
13. Zabbix 1.8 Network Monitoring [Text] // RihardsOlups - PACKT publishing, 2010. - 428 p .;

14. Description of the commercial product of the Cacti system monitoring: [electronic document]. - (http://www.cacti.net/features.php)

15. Optimizing Oracle Performance [Text] // Cary Millsap, Jeff Holt - O'Reilly, 2003. - 461 c .;

16. Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide [Text] // Mark Bauer, Richard Strohm - Oracle, 2009. - 25 c .;

17. Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g Programming Techniques and Solutions [Text] // Thomas Kyte - Apress, 2010. - 841 p .;

18. Administration in information systems [Text] // M.N. Belenkaya, S.T. Malinovsky, N.V. Yakovenko - M .: Hot line - Telecom, 2011. - 400 p .;

19. Introduction to operating systems: a textbook for students of higher educational institutions [Text] // DV Irtegov - SPb .: BHV @ Petersburg, 2008. - 1040 p .;

20. Designing database structures [Text] // Tiori T., Frei J. - M .: Mir, 1985. - 287 p .;

21. Expert Oracle Database 11g Administration [Text] // Sam R. Alapati - Apress, 2010. - 1440 c .;

22. Oracle Database Concepts 11g Release 2 (11.2) E40540-04 [Text] // Lance Ashdown, Tom Kyte - Oracle, 2015. - 472 c .;

23. Oracle Database Reference 11g Release2 (11.2) E40402-18 [Text] // Bert Rich - Oracle, 2015. - 1306 c .;

24. Oracle Database 2 Day DBA, 10g Release 2 [Text] // Colin McGregor - Oracle, 2012. - 210 sec .;

25. RFC 3411 - An Architecture for Describing Simple Network Management Protocol Frameworks [Text] // D. Harrington, R. Presuhn, R. Presuhn // Standard, 2002. - 5 p .;

26. Korogodova AO, Belenkaya MN, Problems of performance of Oracle DBMS running Windows network OS: [electronic document] - (http://cyberleninka.ru/article/n/problemy-proizvoditelnosti-subd-oracle- pod-upravleniem-setevoy-os-windows).

27. Mehul Joshi, Performance monitoring and analysis of waiting events when using Oracle's Unix database: [electronic document]. - (https://www.ibm.com/developerworks/ru/library/r-performance-oracle-unix/).

28. Monitoring of SQL server: [electronic document] - (http://www.sql.ru/subscribe/70028/09.shtml/).
29. Processing of SQL and PL \ SQL statements: [electronic document] - (http://www.interface.ru/oracle/ora7/ora7a04a.htm).
30. Description of the commercial product of Zabbix system monitoring: [electronic document]. - (http://www.zabbix.com/ru/product.php).
31. Rating database popularity portal: [electronic document]. - (http://db-engines.com/en/ranking).
32. Khutorov VS, Main problems and objectives of database monitoring by means of Oracle DBMS: [electronic document] - (http://cyberleninka.ru/article/n/osnovnye-problemy-i-tseli-monitoringa-bazy-dannyh -sredstvami-subd-oracle).
33. Wikipedia electronic library, article about the SGA System Global Area: [electronic document] - (https://ru.wikipedia.org/wiki/System GlobalSpace).

Appendix 1

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
select
S.INST_ID,S.ADDR,S.INDX,S.KSUSESER,S.KSUUDSES,S.KSUSEPRO,S.KSUUDLUI,S.KSUUDLNA,S
.KSUUDOCT,S.KSUSESOW,DECODE(S.KSUSETRN,HEXTORAW('00'),NULL,S.KSUSETRN),DECODE(S.
KSQPSWAT,HEXTORAW('00'),NULL,S.KSQPSWAT),DECODE(BITAND(S.KSUSEIDL,11),1,'ACTIVE'
,2,DECODE(BITAND(S.KSUSEFLG,4096),0,'INACTIVE','CACHED'),2,'SNIPED',3,'SNIPED','
KILLED'),DECODE(S.KSSPATYP,1,'DEDICATED',2,'SHARED',3,'PSEUDO',4,'POOLED','NONE'
),S.KSUUDSID,S.KSUUDSNA,S.KSUSEUNM,S.KSUSEPID,S.KSUSEMNM,S.KSUSEMNP,S.KSUSETID,S
.KSUSEPNM,DECODE(BITAND(S.KSUSEFLG,19),17,'BACKGROUND',1,'USER',2,'RECURSIVE','?
 ),S.KSUSESQL,S.KSUSESQH,S.KSUSESQI,DECODE(S.KSUSESCH,65535,TO_NUMBER(NULL),S.KS
USESCH),S.KSUSESESTA,DECODE(S.KSUSESEID,0,TO_NUMBER(NULL),S.KSUSESEID),S.KSUSEPS
Q,S.KSUSEPHA,S.KSUSEPSI,DECODE(S.KSUSEPCH,65535,TO_NUMBER(NULL),S.KSUSEPCH),S.KS
USEPESTA,DECODE(S.KSUSEPEID,0,TO_NUMBER(NULL),S.KSUSEPEID),DECODE(S.KSUSEPEO,0,T
O_NUMBER(NULL),S.KSUSEPEO),DECODE(S.KSUSEPEO,0,TO_NUMBER(NULL),S.KSUSEPES),DECOD
E(S.KSUSEPCO,0,TO_NUMBER(NULL),DECODE(BITAND(S.KSUSSTMBV,POWER(2,11)),POWER(2,11
),S.KSUSEPCO,TO_NUMBER(NULL))),DECODE(S.KSUSEPCS,0,TO_NUMBER(NULL),DECODE(BITAND
(S.KSUSSTMBV,POWER(2,11)),POWER(2,11),S.KSUSEPCS,TO_NUMBER(NULL))),S.KSUSEAPP,S.
KSUSEAPH,S.KSUSEACT,S.KSUSEACH,S.KSUSECLI,S.KSUSEFIX,S.KSUSEOBJ,S.KSUSEFIL,S.KSU
SEBLK,S.KSUSESLT,S.KSUSEORAFN,S.KSUSELTM,S.KSUSECTM,DECODE(BITAND(S.KSUSEPXOPT,1
),1,'NO','YES'),DECODE(S.KSUSEFT,1,'SESSION',4,'SELECT',8,'TRANSACTIONAL','NONE
'),DECODE(S.KSUSEFM,1,'BASIC',2,'PRECONNECT',4,'PREPARSE','NONE'),DECODE(S.KSUSE
FS,1,'YES','NO'),S.KSUSEGRP,DECODE(BITAND(S.KSUSEPXOPT,4),4,'ENABLED',DECODE(BIT
AND(S.KSUSEPXOPT,8),8,'FORCED','DISABLED')),DECODE(BITAND(S.KSUSEPXOPT,2),2,'FOR
CED',DECODE(BITAND(S.KSUSEPXOPT,1),1,'DISABLED','ENABLED')),DECODE(BITAND(S.KSUS
EPXOPT,16),16,'DISABLED','ENABLED')),DECODE(S.KSUSECQD,S.KSUSECLID,DECODE(S.KSUSEBLOCKER,4294967295,'UNKNOWN',4294967294,'UN
KNOWN',4294967293,'UNKNOWN',4294967292,'NOHOLDER',4294967291,'NOTINWAIT','VALID'
),DECODE(S.KSUSEBLOCKER,4294967295,TO_NUMBER(NULL),4294967294,TO_NUMBER(NULL),42
94967293,TO_NUMBER(NULL),4294967292,TO_NUMBER(NULL),4294967291,TO_NUMBER(NULL),B
ITAND(S.KSUSEBLOCKER,2147418112)/65536),DECODE(S.KSUSEBLOCKER,4294967295,TO_NUMB
ER(NULL),4294967294,TO_NUMBER(NULL),4294967293,TO_NUMBER(NULL),4294967292,TO_NUM
BER(NULL),4294967291,TO_NUMBER(NULL),BITAND(S.KSUSEBLOCKER,65535)),DECODE(S.KSUS
EFBLOCKER,4294967295,'UNKNOWN',4294967294,'UNKNOWN',4294967293,'UNKNOWN',4294967
292,'NOHOLDER',4294967291,'NOTINWAIT','VALID'),DECODE(S.KSUSEFBLOCKER,4294967295
,TO_NUMBER(NULL),4294967294,TO_NUMBER(NULL),4294967293,TO_NUMBER(NULL),4294967292
,TO_NUMBER(NULL),4294967291,TO_NUMBER(NULL),BITAND(S.KSUSEFBLOCKER,2147418112)/
65536),DECODE(S.KSUSEFBLOCKER,4294967295,TO_NUMBER(NULL),4294967294,TO_NUMBER(NU
LL),4294967293,TO_NUMBER(NULL),4294967292,TO_NUMBER(NULL),4294967291,TO_NUMBER(N
ULL),BITAND(S.KSUSEFBLOCKER,65535)),W.KSLWTSEQ,W.KSLWTEVT,E.KSLEDNAM,E.KSLEDP1,W
.KSLWTP1,W.KSLWTP1R,E.KSLEDP2,W.KSLWTP2,W.KSLWTP2R,E.KSLEDP3,W.KSLWTP3,W.KSLWTP3
R,E.KSLEDCLASSID,E.KSLEDCLASS#,E.KSLEDCLASS,DECODE(W.KSLWTINWAIT,0,DECODE(BITAND
(W.KSLWTFLAGS,256),0,-2,DECODE(ROUND(W.KSLWTSTIME/10000),0,-1,ROUND(W.KSLWTSTIME
/10000))),0),DECODE(W.KSLWTINWAIT,0,ROUND((W.KSLWTSTIME+W.KSLWTLTIME)/1000000),R
OUND(W.KSLWTSTIME/1000000)),DECODE(W.KSLWTINWAIT,1,'WAITING',DECODE(BITAND(W.KSL
WTFLAGS,256),0,'WAITEDSHORTTIME',DECODE(ROUND(W.KSLWTINWAIT,0,TO_NUMBER(NULL)
/10000),0,'WAITEDSHORTTIME',DECODE(W.KSLWTINWAIT,0,'WAITEDUNKNOWNTIME',DECODE(ROUND(W.KSLWTSTIME/10000),0,'WAITEDSHO
RTTIME','WAITEDKNOWNTIME'))),W.KSLWTSTIME,DECODE(W.KSLWTINWAIT,0,TO_NUMBER(NULL)
,DECODE(BITAND(W.KSLWTFLAGS,64),64,0,W.KSLWTTREM)),W.KSLWTLTIME,S.KSUSESVC,DECOD
E(BITAND(S.KSUSEFLG2,32),32,'ENABLED','DISABLED'),DECODE(BITAND(S.KSUSEFLG2,64),
64,'TRUE','FALSE'),DECODE(BIT
AND(S.KSUSEFLG2,65536)+BITAND(S.KSUSEFLG2,131072),65536,'ALLEXEC',131072,'NEVER'
,'TRUE','FALSE'),DECODE(BITAND(S.KSUSEFLG2,128),128,'TRUE','FALSE'),DECODE(BIT
AND(S.KSUSEFLG2,65536),65536,'FIRSTEXEC'),S.KSUUDSAE,S.KSUSECRE,S.KSUSECSN,S.KSUSEECID
from
X$KSUSE S,
X$KSLED E,
X$KSLWT W
where
BITAND(S.KSSPAFLG,1)!=0 AND BITAND(S.KSUSEFLG,1)!=0 AND S.INDX=W.KSLWTSID AND W.
KSLWTEVT=E.INDX
```

```
SELECT
  s.inst_id,                                    -- INST_ID NUMBER
  s.addr,                                       -- SADDR RAW(8)
  s.indx,                                       -- SID NUMBER
  s.ksuseser,                                   -- SERIAL# NUMBER
  s.ksuudses,                                   -- AUDSID NUMBER
  s.ksusepro,                                   -- PADDR RAW(8)
  s.ksuudlui,                                   -- USER# NUMBER
  s.ksuudlna,                                   -- USERNAME VARCHAR2(30)
  s.ksuudoct,                                   -- COMMAND NUMBER
  s.ksusesow,                                   -- OWNERID NUMBER
  decode(                                       -- TADDR VARCHAR2(16)
    s.ksusetrn,hextoraw('00'),null,s.ksusetrn),
  decode(                                       -- LOCKWAIT VARCHAR2(16)
    s.ksqpswat,hextoraw('00'),null,s.ksqpswat),
  decode(                                       -- STATUS VARCHAR2(8)
bitand(s.ksuseidl,11),1,'ACTIVE',0,decode(bitand(s.ksuseflg,4096),0,'INACTIVE','CACHED'
),2,'SNIPED',3,'KILLED'),
  decode(                                       -- SERVER VARCHAR2(9)
    s.ksspatyp,1,'DEDICATED',2,'SHARED',3,'PSEUDO',4,'POOLED','NONE'),
  s.ksuudsid,                                   -- SCHEMA# NUMBER
  s.ksuudsna,                                   -- SCHEMANAME VARCHAR2(30)
  s.ksuseunm,                                   -- OSUSER VARCHAR2(30)
  s.ksusepid,                                   -- PROCESS VARCHAR2(24)
  s.ksusemnm,                                   -- MACHINE VARCHAR2(64)
  s.ksusemnp,                                   -- PORT NUMBER
  s.ksusetid,                                   -- TERMINAL VARCHAR2(16)
  s.ksusepnm,                                   -- PROGRAM VARCHAR2(64)
  decode(                                       -- TYPE VARCHAR2(10)
    bitand(s.ksuseflg,19),17,'BACKGROUND',1,'USER',2,'RECURSIVE','?'),
  s.ksusesql,                                   -- SQL_ADDRESS RAW(8)
  s.ksusesqh,                                   -- SQL_HASH_VALUE NUMBER
  s.ksusesqi,                                   -- SQL_ID VARCHAR2(13)
  decode(                                       -- SQL_CHILD_NUMBER NUMBER
    s.ksusesch,65535, to_number(null), s.ksusesch),
  s.ksusesesta,                                 -- SQL_EXEC_START DATE
  decode(                                       -- SQL_EXEC_ID NUMBER
    s.ksuseseid,0, to_number(null), s.ksuseseid),
  s.ksusepsq,                                   -- PREV_SQL_ADDR RAW(8)
  s.ksusepha,                                   -- PREV_HASH_VALUE NUMBER
  s.ksusepsi,                                   -- PREV_SQL_ID VARCHAR2(13)
  decode(                                       -- PREV_CHILD_NUMBER NUMBER
    s.ksusepch,65535, to_number(null), s.ksusepch),
  s.ksusepesta,                                 -- PREV_EXEC_START DATE
  decode(                                       -- PREV_EXEC_ID NUMBER
    s.ksusepeid,0, to_number(null), s.ksusepeid),
  decode(                                       -- PLSQL_ENTRY_OBJECT_ID NUMBER
    s.ksusepeo,0,to_number(null),s.ksusepeo),
  decode(                                       -- PLSQL_ENTRY_SUBPROGRAM_ID NUMBER
    s.ksusepeo,0,to_number(null),s.ksusepes),
  decode(                                       -- PLSQL_OBJECT_ID NUMBER
    s.ksusepco,0,to_number(null), decode(bitand(s.ksusstmbv, power(2,11)),
power(2,11), s.ksusepco, to_number(null))),
  decode(                                       -- PLSQL_SUBPROGRAM_ID NUMBER
    s.ksusepcs,0,to_number(null), decode(bitand(s.ksusstmbv, power(2,11)),
```

```
power(  ,  ), s.ksusepcs, to_number(null))),
  s.ksuseapp,                                   -- MODULE VARCHAR2(48)
  s.ksuseaph,                                   -- MODULE_HASH NUMBER
  s.ksuseact,                                   -- ACTION VARCHAR2(32)
  s.ksuseach,                                   -- ACTION_HASH NUMBER
  s.ksusecli,                                   -- CLIENT_INFO VARCHAR2(64)
  s.ksusefix,                                   -- FIXED_TABLE_SEQUENCE NUMBER
  s.ksuseobj,                                   -- ROW_WAIT_OBJ# NUMBER
  s.ksusefil,                                   -- ROW_WAIT_FILE# NUMBER
  s.ksuseblk,                                   -- ROW_WAIT_BLOCK# NUMBER
  s.ksuseslt,                                   -- ROW_WAIT_ROW# NUMBER
  s.ksuseorafn,                                 -- TOP_LEVEL_CALL# NUMBER
  s.ksuseltm,                                   -- LOGON_TIME DATE
  s.ksusectm,                                   -- LAST_CALL_ET NUMBER
  decode(                                       -- PDML_ENABLED VARCHAR2(3)
    bitand(s.ksusepxopt,  ),  ,'NO','YES'),
  decode(                                       -- FAILOVER_TYPE VARCHAR2(13)
    s.ksuseft,  , 'SESSION', 4, 'SELECT',  , 'TRANSACTIONAL','NONE'),
  decode(                                       -- FAILOVER_METHOD VARCHAR2(10)
```

```
    w.kslwtinwait,  ,round((w.kslwtstime+w.kslwtltime)/  ),
round(w.kslwtstime/  )),
  decode(                                       -- STATE VARCHAR2(19)
    w.kslwtinwait,  , 'WAITING', decode(bitand(w.kslwtflags,  ),  , 'WAITED UNKNOWN
  , decode(round(w.kslwtstime/  ),  , 'WAITED SHORT TIME', 'WAITED KNOWN
  ))),
  w.kslwtstime,                                 -- WAIT_TIME_MICRO NUMBER
  decode(                                       -- TIME_REMAINING_MICRO NUMBER
    w.kslwtinwait,  ,to_number(null),
decode(bitand(w.kslwtflags,  ),  ,  ,w.kslwttrem)),
  w.kslwtltime,                                 -- TIME_SINCE_LAST_WAIT_MICRO NUMBER
  s.ksusesvc,                                   -- SERVICE_NAME VARCHAR2(64)
  decode(                                       -- SQL_TRACE VARCHAR2(8)
    bitand(s.ksuseflg2,  ),  ,'ENABLED','DISABLED'),
  decode(                                       -- SQL_TRACE_WAITS VARCHAR2(5)
    bitand(s.ksuseflg2,  ),  ,'TRUE','FALSE'),
  decode(                                       -- SQL_TRACE_BINDS VARCHAR2(5)
    bitand(s.ksuseflg2,  ),  ,'TRUE','FALSE'),
  decode(                                       -- SQL_TRACE_PLAN_STATS VARCHAR2(10)
    bitand(s.ksuseflg2,  ),  ,'TRUE','FALSE'),
  decode(                                       -- SESSION_EDITION_ID NUMBER
    bitand(s.ksuseflg,  ) + bitand(s.ksuseflg2,  ),  ,'ALL
EXEC',  ,'NEVER',  ,'FIRST EXEC'),
  s.ksuudsae,                                   -- CREATOR_ADDR RAW(8)
  s.ksusecre,                                   -- CREATOR_SERIAL# NUMBER
  s.ksusecsn,                                   -- ECID VARCHAR2(64)
  s.ksuseecid
from
  x$ksuse s,
  x$ksled e,
  x$kslwt w
where
  bitand(s.ksspaflg,  )!=  and bitand(s.ksuseflg,  )!=  and s.indx=w.kslwtsid and
w.kslwtevt=e.indx
```