***Performance tuning***

We tune the database regularly for optimal performance. The following are the tuning areas, where we are concentrating to tune the database

1. **Memory Tuning**
2. **SQL Tuning**

The following are the two ways to monitor and tune the database

1. ***Reactive Performance tuning:*** In this way we monitor and tune the database after a management or performance issues arisen.
2. ***Proactive Performance tuning:*** In this way we monitor and tune the database before management or performance issues arisen.

There is no exact tool that is used to tune the database. To tune the database it is always been a trial and error method. Either way you go, the following are the systematic way to tune the database for optimal performance.

***Memory Tuning***

Retrieving data from memory is generally faster than retrieving it from disk. Oracle uses two kinds of memory structure:-

1. **PGA ( Process Global Area)**
2. **SGA( System Global Area)**

**PGA (Process Global Area)**

It is a Process specific memory (allocated in RAM) to a single process or thread that is not accessible from other processes or threads. It is allocated when a dedicated server process is established and deallocated when it is disconnected .In PGA oracle typically stores session’s information, cursor space, sorting area etc.

In 10g, PGA does not require to tune any parameter because PGA memory is allocated dynamically based upon requirement.

***SGA (System Global Area)***

It is an important component in instance. It is a common area (memory) which is shared by all server processes and Background processes. The main purpose of SGA is to speed up query performance and to enable concurrent database activity.

Because processing in SGA (memory) is much faster than disk I/O, the size of the SGA is one of the important configuration issues when you are tuning the database for optimal performance.

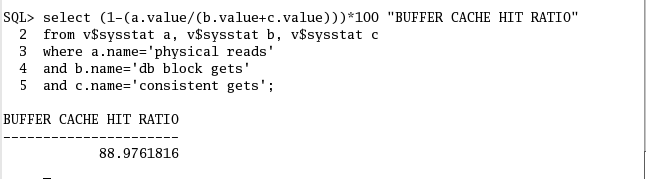
When you start an instance, the instance takes a certain amount of memory from operating system’s RAM. The amount is based on the size of the SGA component in the initialization parameter. When the instance is shutdown, the memory used by the SGA, goes back to the host systems.

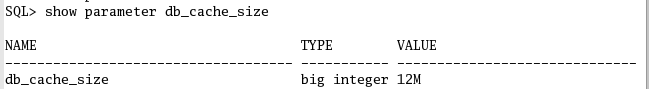
The SGA is not a homogenous entity rather it’s a combination of several memory structure. The following are the main components of SGA.

***DATABASE BUFFER CACHE –*** It is a part of SGA. It holds copies of data blocks so they can be accessed quicker by oracle. The main purpose of buffer cache is to minimize physical I/O. Reading a block from the buffer cache is less costly (in terms of time) than reading it from the disk. For optimal performance, buffer cache tuning is very important. Oracle keeps track of all logical and physical reads in the v$sysstat performance view.

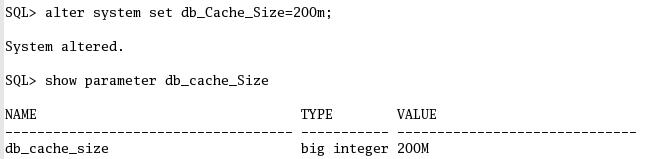
To check the efficiency of buffer cache we measure the hit ratio. Hit ratio indicates the percentage of the data block accesses without requiring a physical read from the disk. Make sure it is more than 80% for an OLTP environment and 99 is the best value. If hit ratio is below the range , you may need to increase the value of DB\_CACHE\_SIZE

***How to calculate Buffer Cache Hit Ratio***

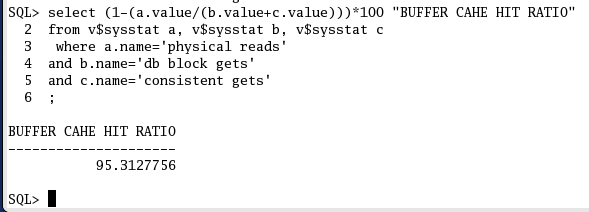




***You may need to increase to the value of DB\_CACHE\_SIZE***



**After increasing yo do fetch data from database from different -2 sessions. And then check the hit ratio.**



**Tuning the SHARED POOL**

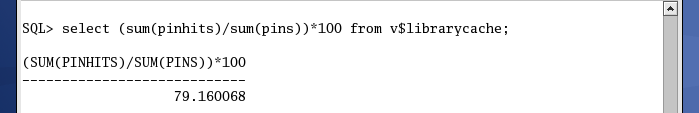
It is an important part of SGA. Unlike database buffer cache, which holds actual data blocks. The shared pool holds parsed and executable version of SQL and PL/SQL code as well as information regarding the data dictionary tables.

Parameter ***SHARED\_POOL\_SIZE*** controls the size of the shared pool itself. The size of the shared pool can impact performance. If it is too small then it is likely that sharable information will be flushed from the pool and then later need to be reloaded (rebuilt). The shared pool consists of two major areas. Library cache and Data dictionary cache

***Tuning the Library cache***

To hold the parsed and executable version of code. Library cache hit ratio should be >95%. If hit ratio is below 95%, you may need to increase the value of SHARED\_POOL\_SIZE.

***How to calculate Library Cache Hit Ratio***

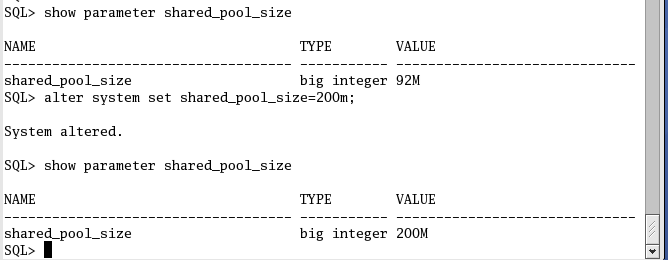


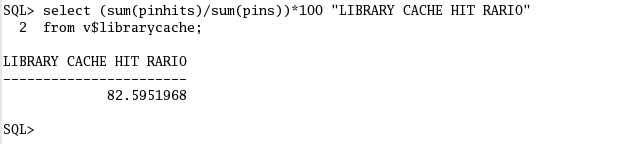
Where

**PINHITS**- It reports on the number of times Oracle found parsed and executable version of SQL statements in memory.

**PINS**- It reports on the number of times Oracle requests for objects in cache in order to access them.

***You may need to increase to the value of SHARED\_POOL\_SIZE***

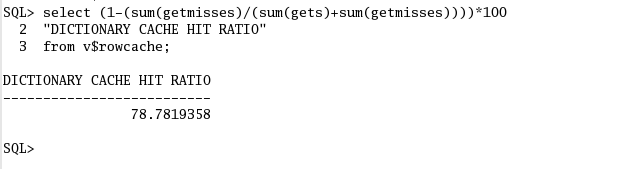




***Tuning the Data dictionary cache***

To hold data dictionary tables, that contains metadata about the tables, users, privileges, roles etc... It indicates the percentage of time the data dictionary objects are read from memory rather than on disk. The value should be greater than 95%. If less then increase the shared pool (SHARED\_POOL\_SIZE init parameter).

***How to calculate Data dictionary Cache Hit Ratio***



Where

Gets: Total number of requests for information on the data object

Getmisses: Number of data requests resulting in cache misses

***You may need to increase to the value of SHARED\_POOL\_SIZE***

***Tuning the log buffer***

The log buffer is NOT tunable anymore in 10gR2

***Tuning the JAVA POOL***

It is optional memory. It does not require tuning. This is memory space allocated for database-driven Java code.

***Tuning the LARGE POOL***

It is optional memory. It does not require tuning .The purpose of large pool memory is to reduce the strain of the shared pool. If you create a large pool

1. Oracle uses it to store shared server session information
2. Rman uses it to for caching buffers during backup and recovery.

***If you do not create a large pool your instance will still function but perhaps not as efficiently.***

***Tuning the STREAMS POOL***

It is optional memory. It does not require tuning .It supports oracle streams feature

======================================

***Automatic Shared Memory Management (ASMM)***

In Oracle10g memory can be managed automatically by setting parameter sga\_target. Oracle will then manage parameters like

1. ***Default pool (db\_cache\_Size)***
2. ***Shared pool(shared\_pool\_size)***
3. ***Large pool( large\_pool\_size)***
4. ***Java pool(java\_pool\_size)***
5. ***Streams pool(streams\_pool\_size)***

ASMM does not automatically manage the size of

1. ***Log buffer***
2. ***Keep pool, recycle pool,***
3. ***Non default buffer cache***
4. ***Fixed size***

When SGA\_TARGET is set(non zero value) , the total size of the manual SGA size parameter is subtracted from the SGA\_TARGET value and balance is available for auto tuned SGA components.

SGA\_TARGET can be dynamically set with alter system command. It can be increased upto the value of SGA\_MAX\_SIZE. It can be reduced until any one auto tuned component(either user specified or an internally determine minimum). If you increase or decrease the value in SGA\_TARGET that affects only the sizes of auto tuned components.

ASMM uses new background process MMAN(Memory Manager) that co-ordinates the size of memory components.MMAN observes the system and workload in order the to determine the ideal size of the memory.

Example :

First of all you set the value of SGA\_MAX\_SIZE

And then you set the value of SGA\_TARGET in the parameter file. Don’t set the value of any auto tuned parameter , let it oracle will allocate as per the workload.  
**-----------------------------------------------------------------------------------------------**

~dbs]$ vi initprd.ora

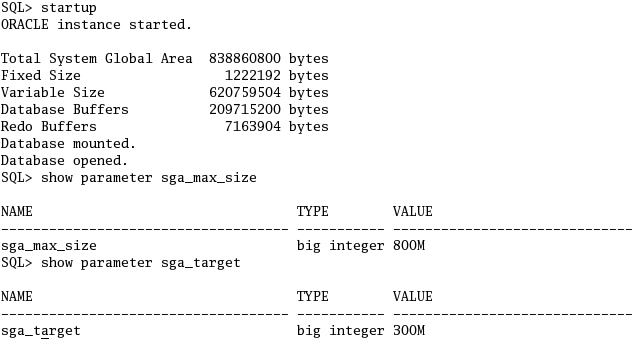
***SGA\_MAX\_SIZE=800M***

***SGA\_TARGET=300M***

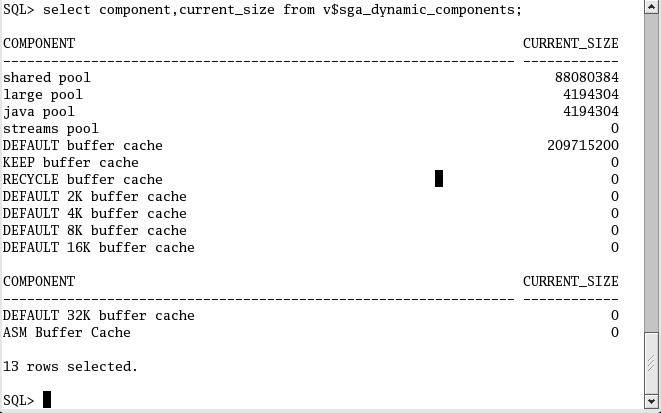
:wq

~dbs]$ sqlplus ‘/as sysdba’

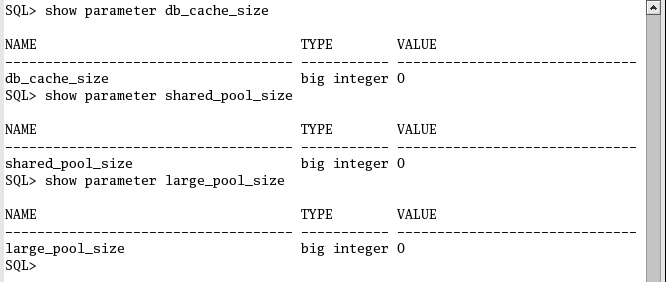
**-----------------------------------------------------------------------------------------------**



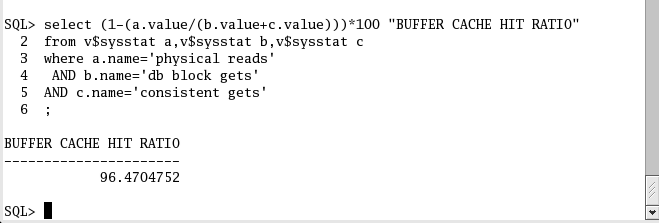
***When you use sga\_target , then if you want to know the current value of auto tuned parameter , use the following command***



***You can not see the auto tuned parameter value by using this command***



***Now you do some fetching operation from the database(repeat it 2-3 times) and then check the hit ratio***

***Now check the Buffer cache hit ratio***  
 

==========================================

***SQL TUNING***

SQL tuning is the process of ensuring that the SQL statements that an application will issue will run in the fastest possible time.

A user’s SQL statement goes through parsing, optimization and execution stages.

***Parsing:*** In this stage oracle check syntax and semantics (metadata of the database tables, users, roles etc) of the SQL statement.

***Optimization***: In this stage oracle (optimizer) chooses the optimal execution plan.

***Execution:*** In this oracle executes the executable version of code which is done by optimizer

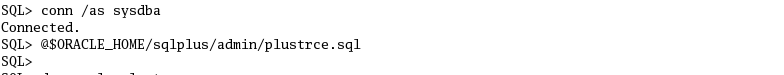
The following are the various ways are used to tune the SQL

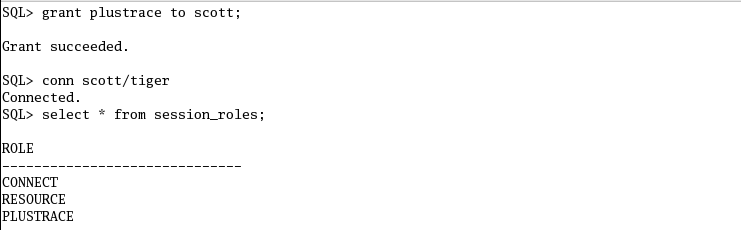
1. ***QUERY OPTIMIZATION USING PARALLEL QUERY***

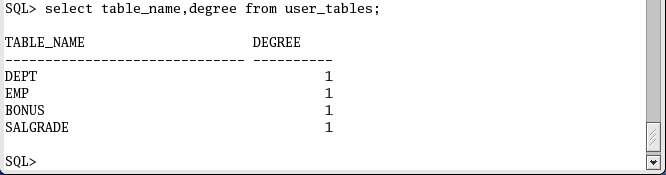
When you enable parallel query option for a table, oracle partition an SQL query into sub-queries and dedicate separate processors to each one. At this time, parallel query is useful only for queries that perform full-table scans on large tables.

***LARGE QUERY WITHOUT USING PARALLEL QUERY OPTION***

First of all you create plustrace role and grant it to user, that will help to trace the execution plan of the query.

****



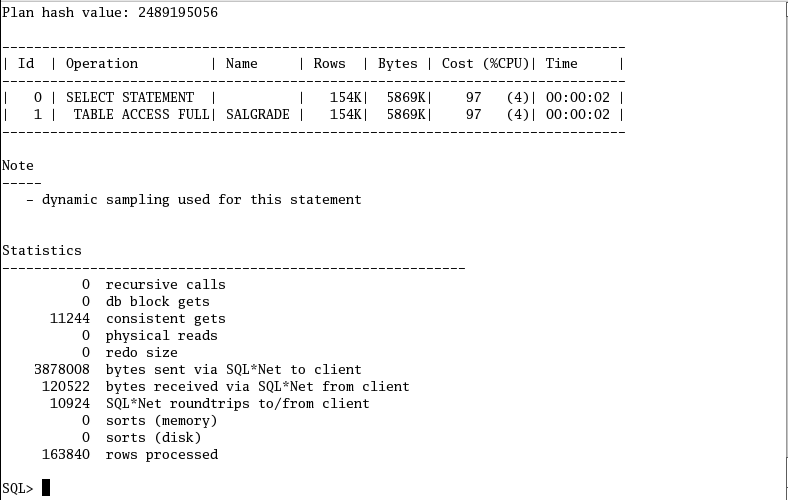




This salgrade table contains thousands of records

After execution of this query it will display report along with it execution plan. Why because this session is enabled autotrace on

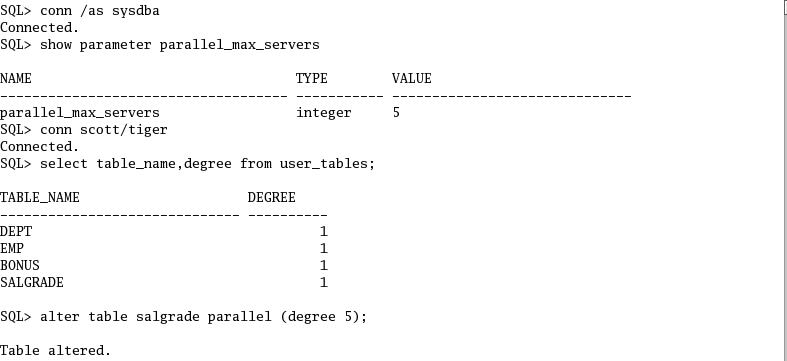
This is execution plan

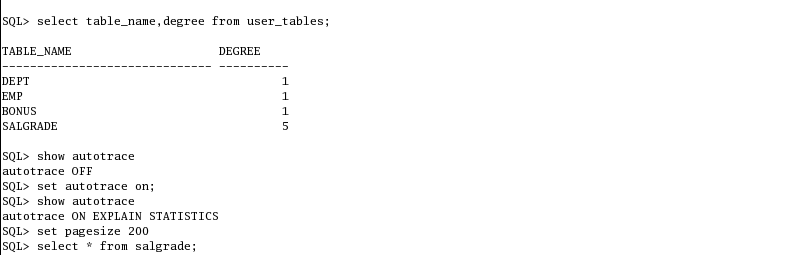


Note: Note down cost of cpu took to execute this query without using parallel query option.

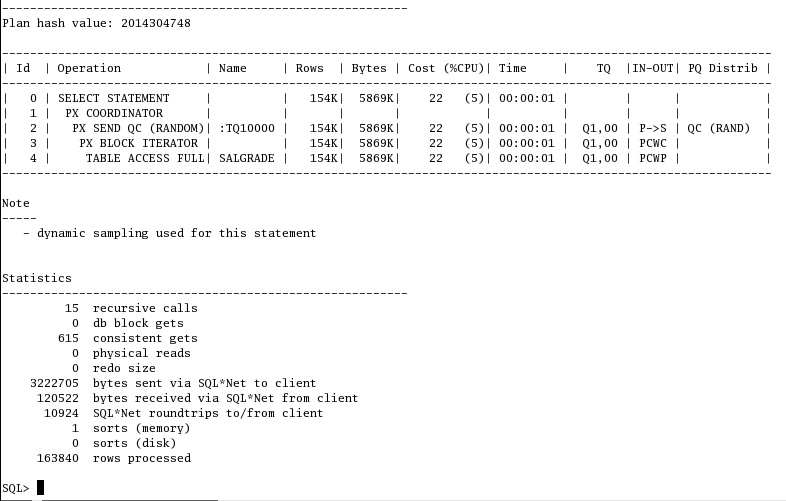
***Large Query with using parallel query option***

Use the following steps to enable parallel query option





After execution of this query it will display report along with it execution plan. Compare the two execution plan before and after , you will see cost of cpu is drastically decreased by using parallel query option.



Yes, cost of the cpu is reduced from (97 to 22).

1. ***Query optimization by using caching tables in memory***

When a SQL statement requests a row from a table, Oracle first checks the internal memory structures to see if the data is already in a data buffer. If the requested data is there, it’s returned, saving a physical IO operation. With the very large SGAs in some 64-bit releases of Oracle, small databases can be entirely cached. For very large databases, however, the RAM data buffers cannot hold all of the database blocks.

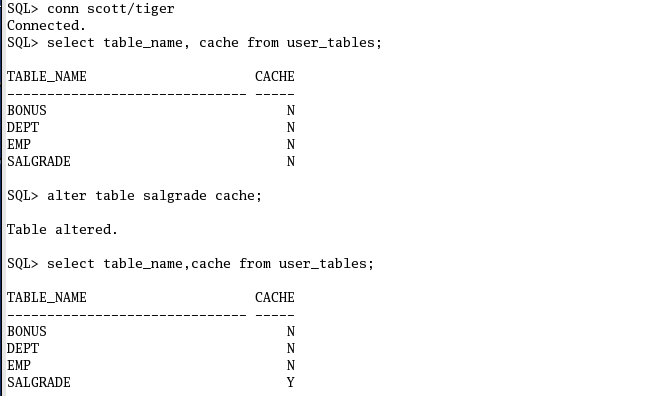
Oracle has a scheme for keeping frequently used blocks in RAM. When there isn't enough room in the data buffer for the whole database, Oracle utilizes a least-recently-used algorithm to determine which database pages are to be flushed from memory. Oracle keeps an in-memory control structure for each block in the data bufferData blocks that aren't frequently referenced will wind up at the end of the data buffer where they will eventually be erased to make room for a new data block

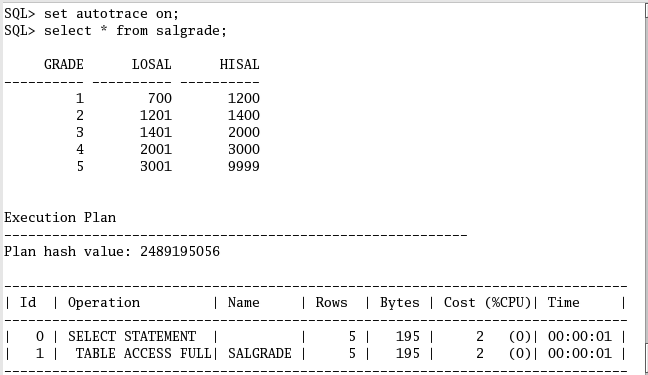
One of the most important areas of Oracle tuning is the management of the RAM data buffers. Oracle performance will be dramatically improved if you can avoid a physical disk I/O by storing a data block inside the RAM memory of the data buffer caches.

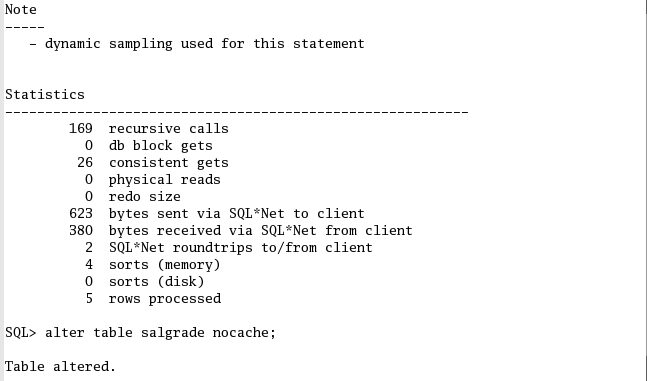
The main purpose of the caching a table is to allow small tables that are always read from front to back to remain in the data buffer

1. ***Caching table in default pool (db\_cache\_Size)***

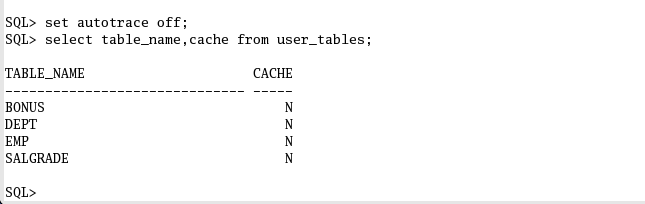
Frequently accessed table you can keep in default pool for optimize the query(To minimize disk I/O and maximize performance). If you cache a table in buffer it will stay all the time it won’t aged out from the buffer.







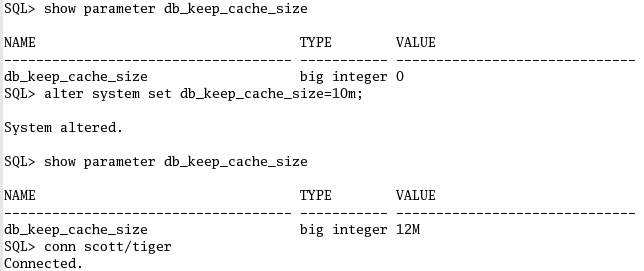
Since table salgrade is cached in the buffer and it is present in the buffer , hence physical reads is 0. Means data block of salgrade table is still present in default pool.



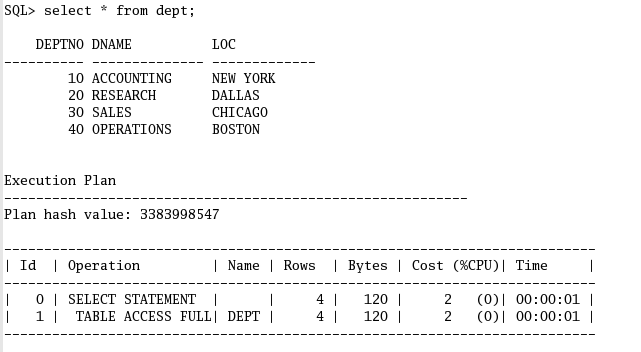
1. ***Oracle keep pool caching(db\_keep\_cache\_Size)***

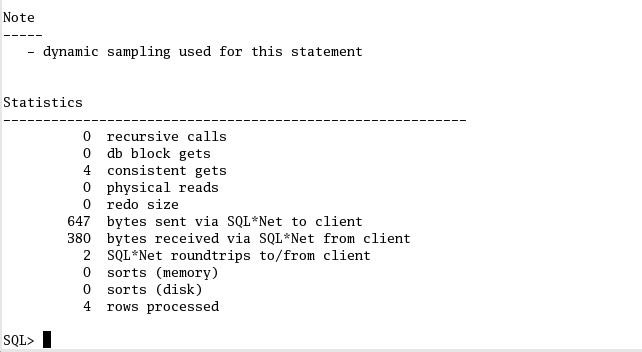
Data which is frequently accessed should be kept in Keep buffer pool. Keep buffer pool retains data in the memory. So that next request for same data can be entertained from memory. This avoids disk read and increases performance. Usually small objects should be kept in Keep buffer. **DB\_KEEP\_CACHE\_SIZE** initialization parameter is used to create Keep buffer Pool. If DB\_KEEP\_CACHE\_SIZE is not used then no Keep buffer is created. Use following syntax to create a Keep buffer pool of 10 MB.

**DB\_KEEP\_CACHE\_SIZE=10M**

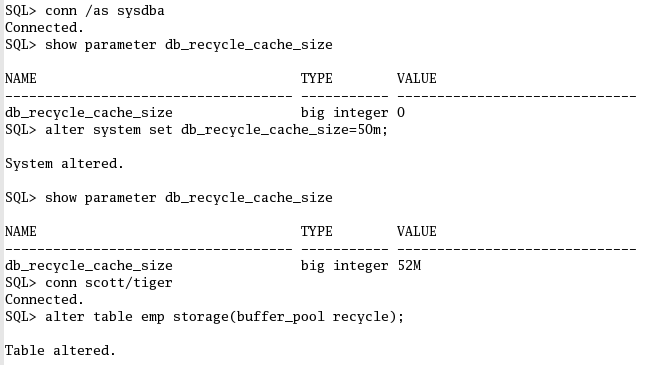


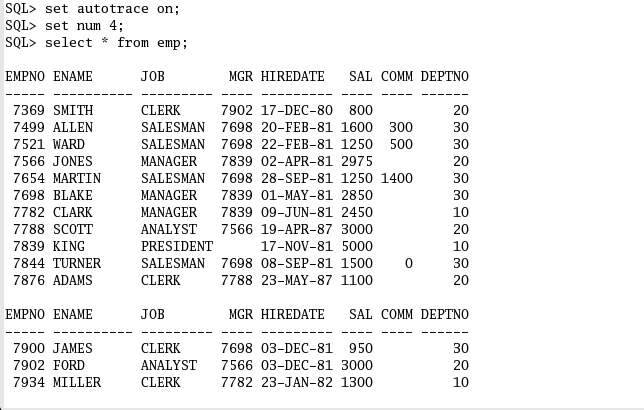


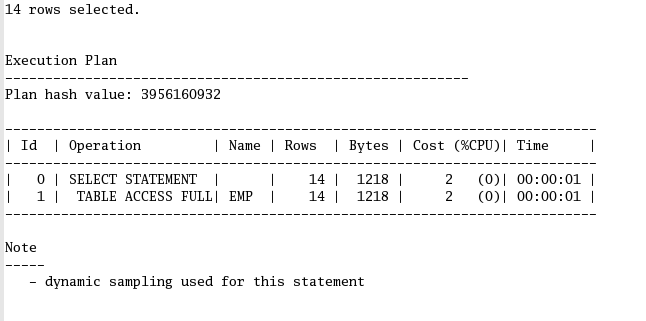


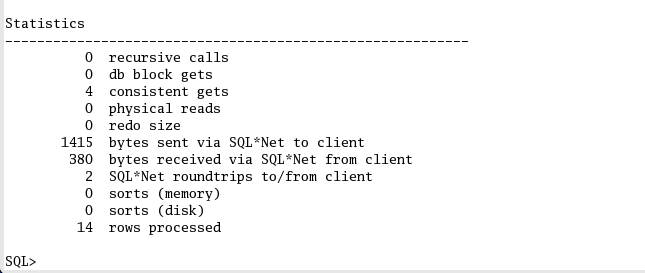


1. ***Caching table in recycle pool***









====================================

***SQL Tuning tools***

Q. What is execution plan?

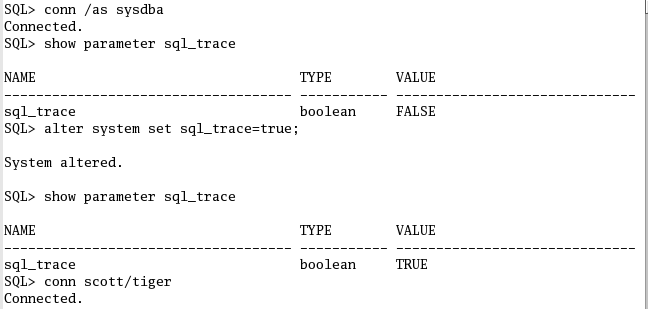
* Ans. An execution plan is a list of steps that Oracle will follow in order to execute a SQL statement. Each step is one of a finite number of basic operations known to the database server. Even the most complex SQL statement can be broken down into a series of basic operations.

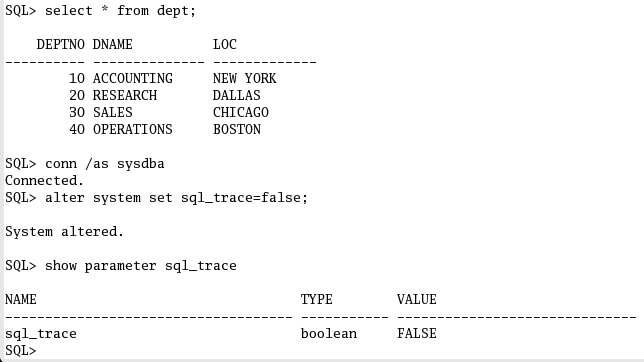
The following are the tools is used to get the execution plan of the SQL statement. Based on the execution plan , we try to find the bottlenecks and resolve it.

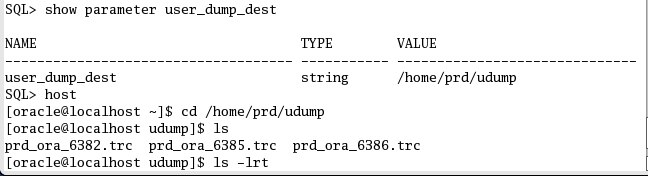
1. **SQL trace**
2. **Tkprof**
3. **Explain plan**
4. ***SQL trace***

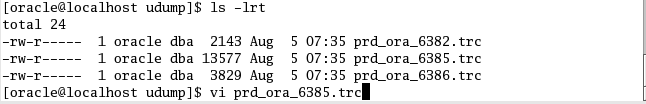
It gives the actual execution plan of the entered query. It is generated once the execution of a query is completed. To get the execution plan of the sql statement. You must enable sql\_trace=true. When you enable sql\_trace =true. Oracle generates execution plan and send it to user\_dump\_dest location, there you can get the trace file of the sql statement. But it is not readable .

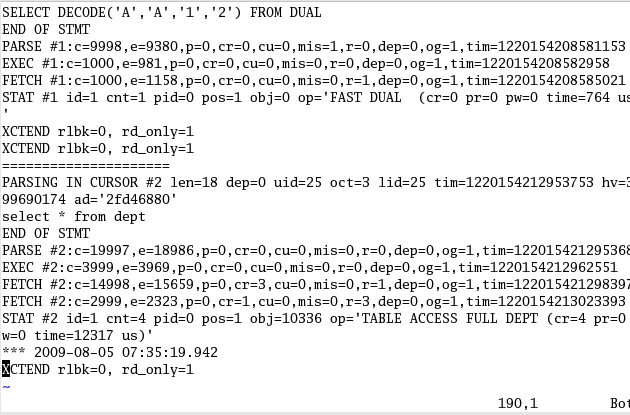
***How to enbable sql\_trace***

****

****

****

****

****

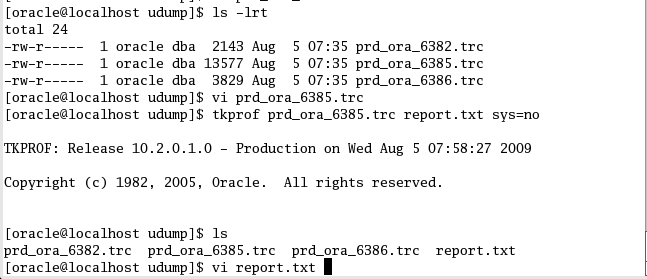
This is the trace of sql statement but is it not understandable format. For this use tkprof.

**Tkprof**

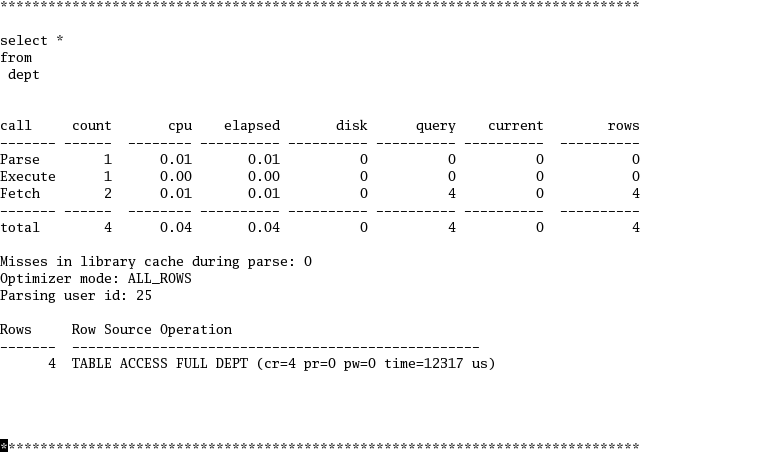
TKPROF is a utility provided by Oracle that formats SQL trace files into very helpful and readable reports. TKPROF is installed automatically when the database server software is installed. You invoke TKPROF from the operating system command line; there is no graphical interface for TKPROF.

***Formatting a Trace File with TKPROF***

***Invoke TKPROF from the operating system prompt like this***

****

Where sys=n Omit “recursive SQL” performed by the SYS user

****

By using this report we identify the cpu time, physical reads, number of records which are executed a particular sql statemen.

|  |  |
| --- | --- |
| **Column Value** | **Meaning** |
| PARSE | Translates the SQL statement into an execution plan, including checks for proper security authorization and checks for the existence of tables, columns, and other referenced objects. |
| EXECUTE | Actual execution of the statement by Oracle. For INSERT, UPDATE, and DELETE statements, this modifies the data. For SELECT statements, this identifies the selected rows. |
| FETCH | Retrieves rows returned by a query. Fetches are only performed for SELECT statements. |
| COUNT | Number of times a statement was parsed, executed, or fetched. |
| CPU | Total CPU time in seconds for all parse, execute, or fetch calls for the statement. This value is zero (0) if TIMED\_STATISTICS is not turned on. |
| ELAPSED | Total elapsed time in seconds for all parse, execute, or fetch calls for the statement. This value is zero (0) if TIMED\_STATISTICS is not turned on. |
| DISK | Total number of data blocks physically read from the datafiles on disk for all parse, execute, or fetch calls. |
| QUERY | Total number of buffers retrieved in consistent mode for all parse, execute, or fetch calls. Usually, buffers are retrieved in consistent mode for queries. |
| CURRENT | Total number of buffers retrieved in current mode. Buffers are retrieved in current mode for statements such as INSERT, UPDATE, and DELETE. |
| ROWS | Total number of rows processed by the SQL statement. This total does not include rows processed by subqueries of the SQL statement. |

Note: Query + Current = Logical Reads (total number of buffers accessed)

**Disadvantage with tkprof**

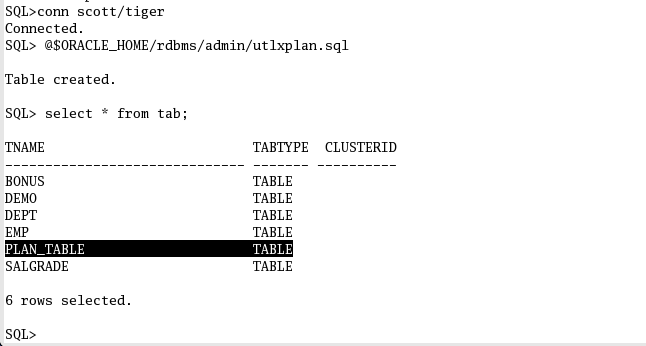
Tkprof will lets us to know once the complete query fired-in and displays the output, then we can come to know that whether the optimizer went for full table scan or index.

***Explain Plan***

EXPLAIN PLAN is a statement that allows you to have Oracle generate the execution plan for any SQL statement without actually executing it. You will be able to examine the execution plan by querying the plan table.

***The Plan Table***

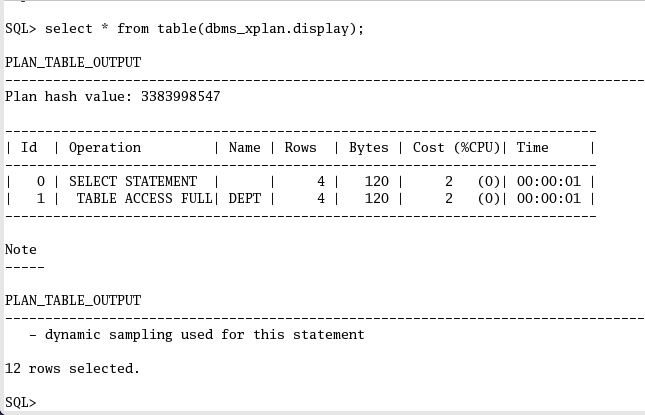
A plan table holds execution plans generated by the EXPLAIN PLAN statement.The typical name for a plan table is plan\_table.Create the plan table by running utlxplan.sql, located in $ORACLE\_HOME/rdbms/admin

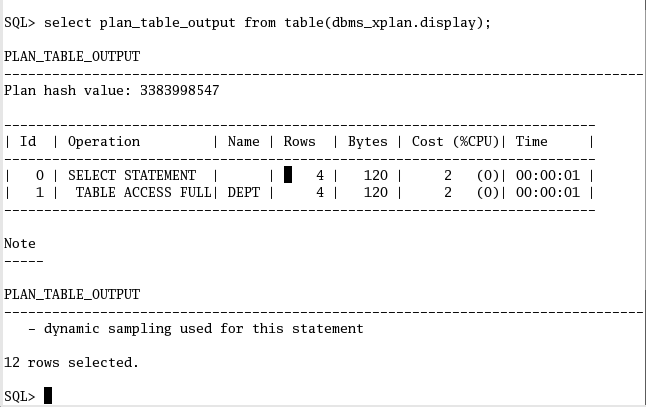




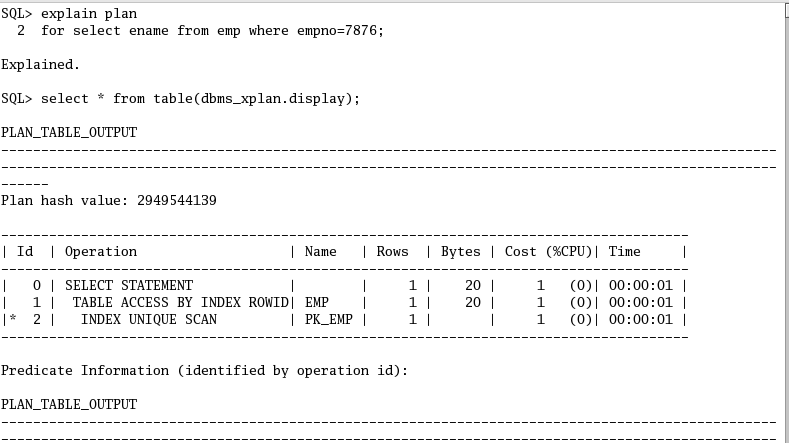
**To display the result of the last EXPLAIN PLAN command stored in the plan table:**

SELECT \* FROM table(DBMS\_XPLAN.DISPLAY);





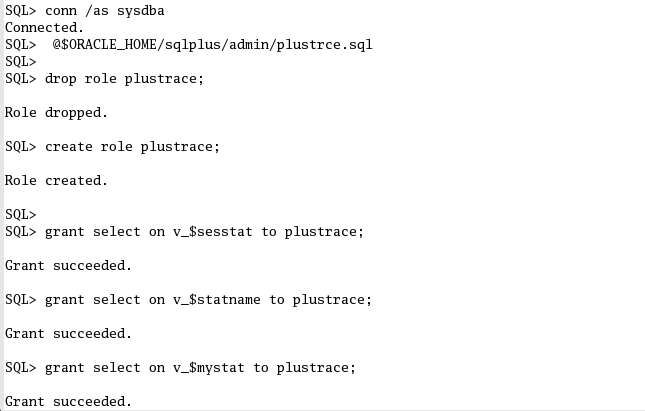
Another example

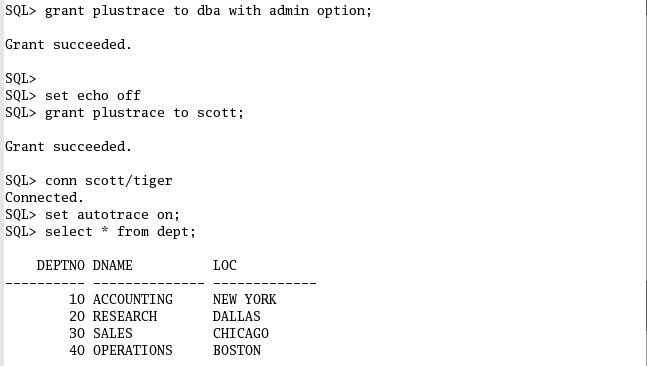
****

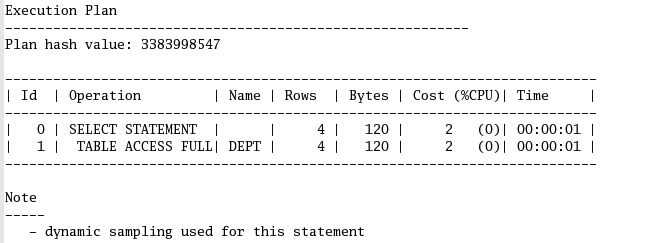
***SQL Autotrace***

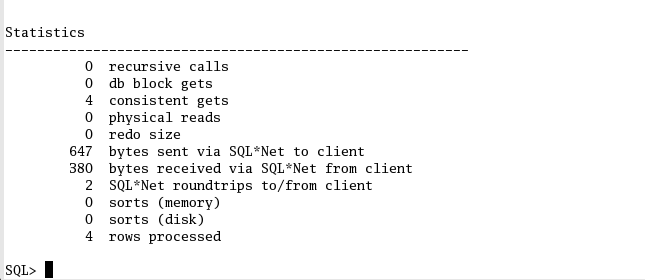
SQL autotrace automatically will display output along with the report on the execution path used by the SQL optimizer and the statement execution statistics along with . It is useful for monitoring and tuning the performance of the sql statements.

To use this feature, you must have the PLUSTRACE role granted to you.









***Database performance tuning tools***

In order to tune the database, oracle provides a diagnostic tool by which you can find out the health of your database.

The following are the performance monitoring and reporting tools are used to get the complete database level performance report. Based on this as a DBA we tune the database.

1. **Statspack**
2. **Awr**
3. **Addm**

**STATSPACK**

**Q. What is Statspack?**

**Ans-** Statspack is a set of performance monitoring and reporting utilities provided by Oracle starting from Oracle 8i .It is still in oracle 10g

**How to configure statspack Performance diagnostic tool**

**1. Installation**

 Installing the statspack is an easy step. The only two things that need to be done is to execute the statspack creation script that is located in $ORACLE\_HOME/rdbms/admin. The script is named as “spcreate.sql” for creating the statspack objects and “spdrop.sql” to drop the statspack user and its associated objects. The second thing though not necessary that you can do is to dedicate a seperate tablespace for statspack.

In oder to install the statspack, use the following comand replacing the location where the script is located:

**SQL> @?/oracle/rdbms/admin/spcreate.sql;**

During the execution of the script, it will ask three things from the user (1) the password for the username of statspack which is “perfstat”.  (2) Default tablespace for statspack and (3) Temporary tablespace for use.

***2.  Generating Snapshots***

In order to generate statspack report, first a snapshot of the database needs to be taken so that the analysis can be done. The report is always generated between any two snapshots. To generate a snapshot use the following command:

**SQL> execute statspack.snap;**

***3.  Generating Statspack Report***

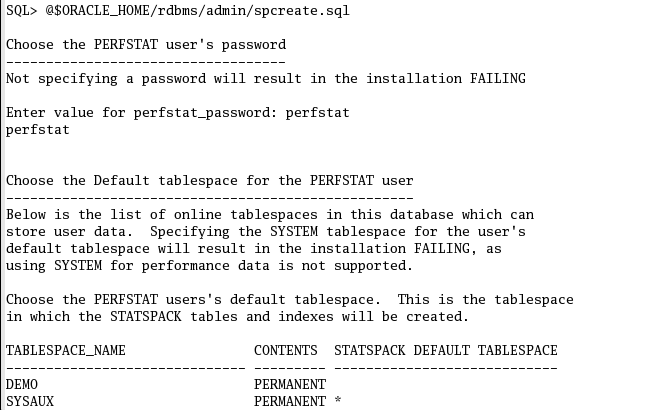
In order to generate statspack report, we only have to execute another script supplied by oracle located in rdbms\admin folder named as “spreport.sql”. To generate a report, it is necessary that you must have atleast two snapshots of the database otherwise lots of portions of the report would not be analysed or calculated by the statspack tool. To generate a report, use the following command :

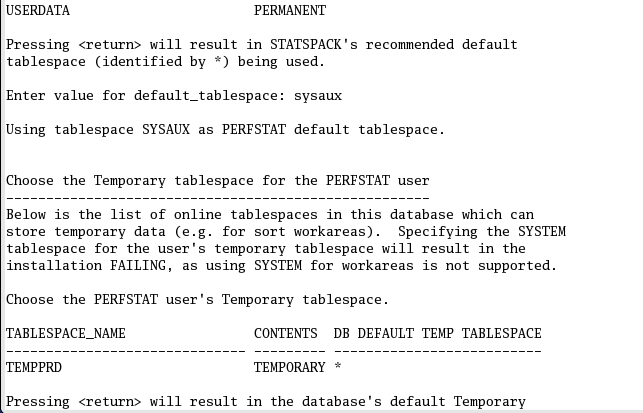
**SQL> @?/oracle/rdbms/admin/spreport.sql;**

At the start of the report, it will show the list of snapshots currently present and would onwards ask for the beginning and ending snapshot ID to compare and analyse.

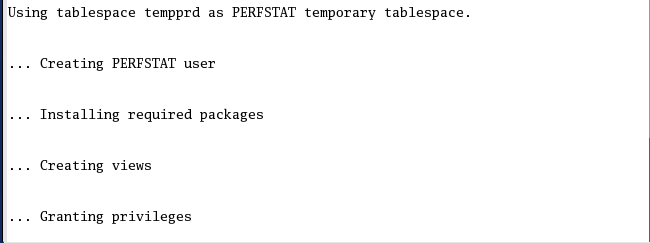
***SCREEN SHOT***

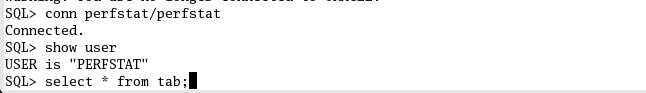
***To execute the spcreate.sql script , first connect as sys user***



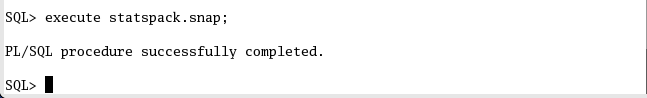




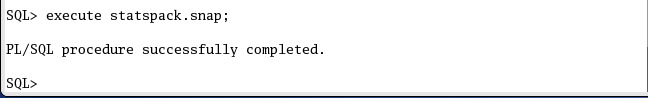




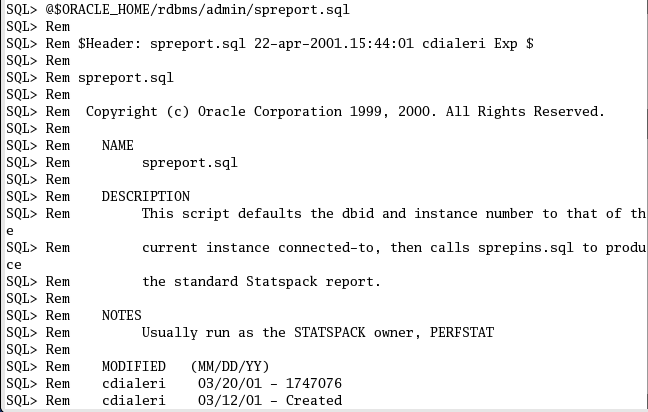
***Generate Snapshots At frequent intervals***

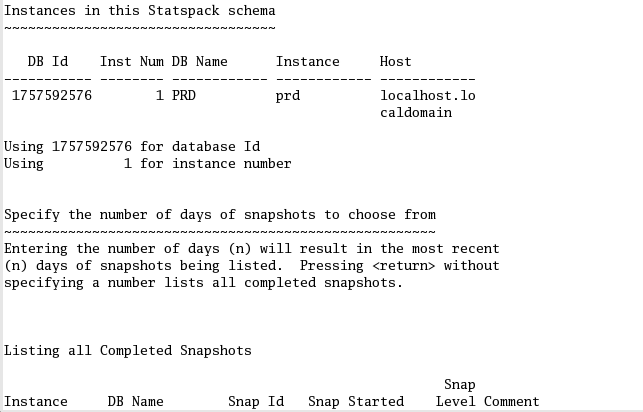
****

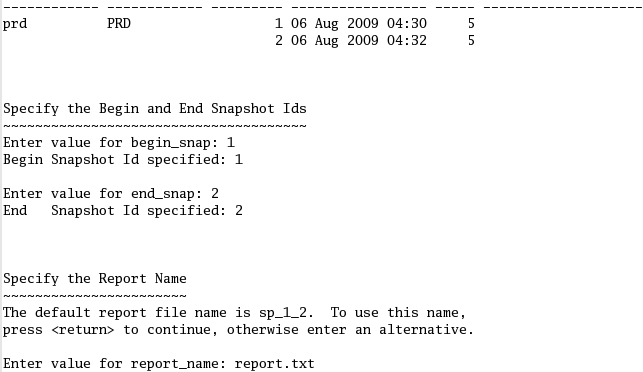
***After one hour take another snapshot***

****

***Minimum two snapshot is required to take a statspack report***

****

****

****

***Go to os level open the report and analysis it.***

***AUTOMATIC WORKLOAD REPOSITORY***

***(AWR)***

**What is the Automatic Workload Repository (AWR)?**

Automatic workload repository is a collection of tables that stores persistent system performance statistics which is owned by user sys. It resides in SYSAUX tablespace. By default snapshot are generated once every 60 min and maintained for 7 days by default. Two new background processes such as Memory Monitor (MMON) and Memory Monitor Light (MMNL), work together and collect performance statistics directly from the System Global Area (SGA). The major part of the collection work is done by MMON that wakes up every sixty minutes by default and gathers statistical information from the data dictionary views and dynamic performance views and then stores this information in the database.

***Creating SNAPSHOT Manually:***   
===========================   
  
BEGIN   
  DBMS\_WORKLOAD\_REPOSITORY.create\_snapshot();   
END;   
/

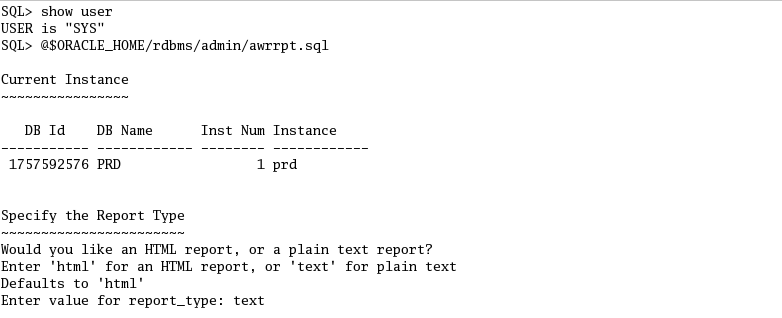
***How to Modify the AWR SNAP SHOT SETTINGS:***  
=====================================   
  
BEGIN   
  DBMS\_WORKLOAD\_REPOSITORY.modify\_snapshot\_settings(   
    retention => 43200,        -- Minutes (= 30 Days). Current value retained if NULL.   
    interval  => 30);          -- Minutes. Current value retained if NULL.   
END;   
/

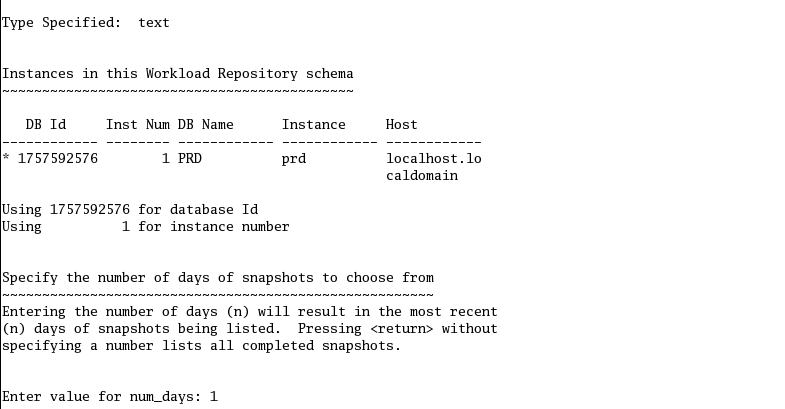
***Workload Repository Views:***   
==========================  
   
DBA\_HIST\_SNAPSHOT - Displays snapshot information.

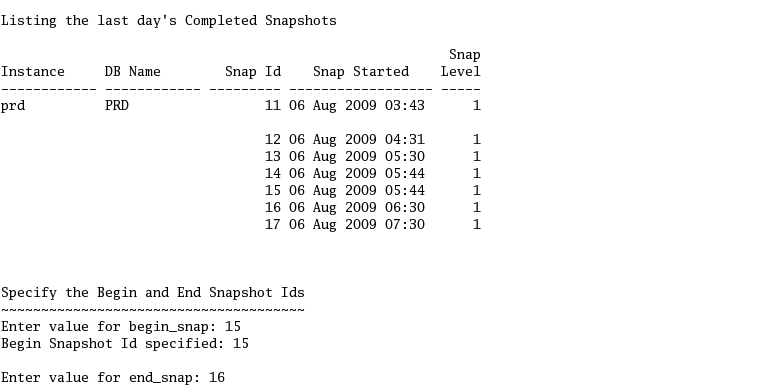
DBA\_HIST\_WR\_CONTROL - Displays AWR settings.

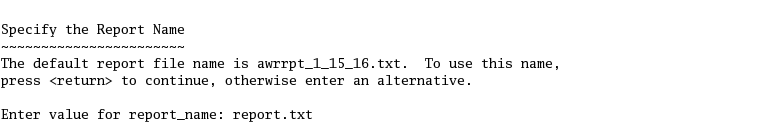
***How to genereate AWR report:***   
=================================

The script awrrpt.sql is used to generate awr report









***Go to os level open the report and analyze it.***

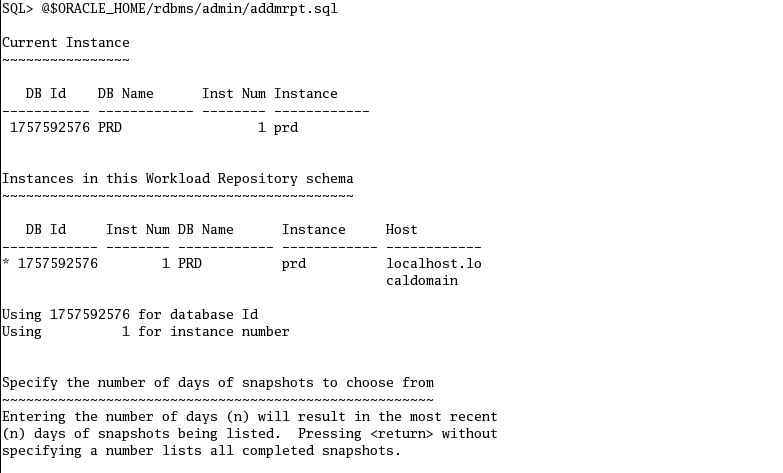
***Automatic Database Diagnostic Monitor***

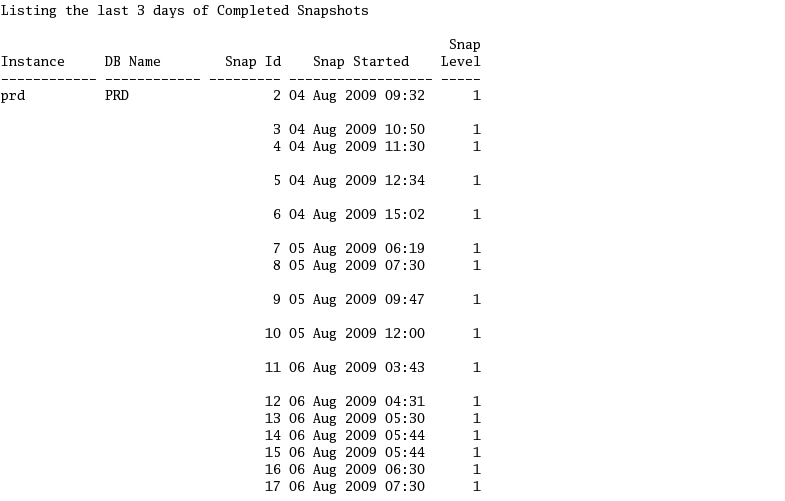
***(ADDM)***

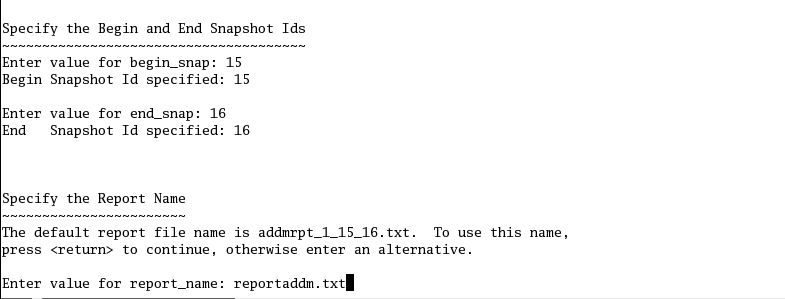
The Automatic Database Diagnostic Monitor (ADDM) analyzes data in the Automatic Workload Repository (AWR) to identify potential performance bottlenecks. For each of the identified issues it locates the root cause and provides recommendations for correcting the problem.

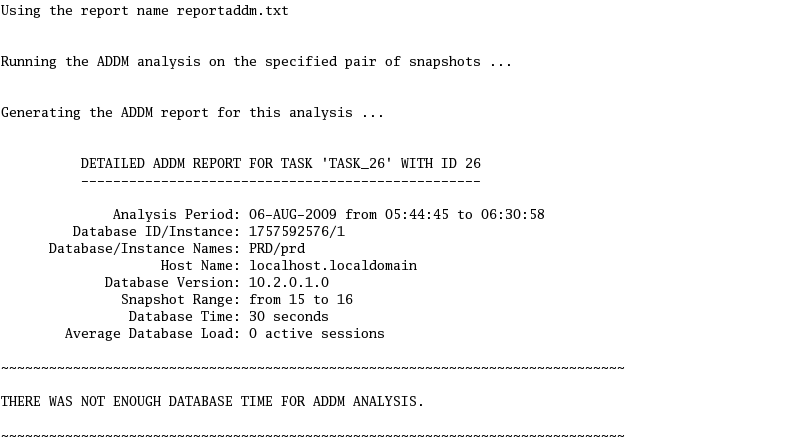
ADDM can be described as the database’s doctor. It allows an Oracle database to diagnose itself and determine how potential problems could be resolved. ADDM runs automatically after each AWR statistics capture, making the performance diagnostic data readily available.

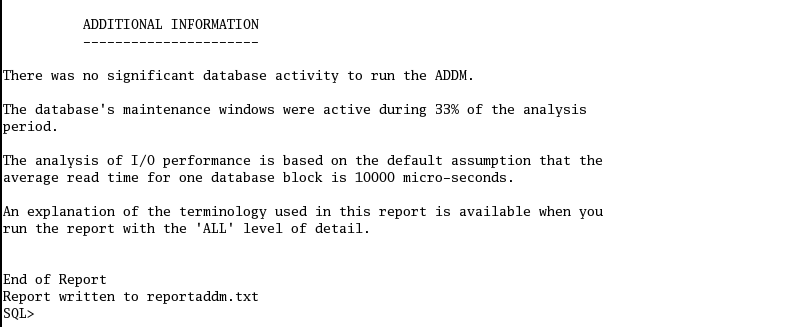
***How to generate ADDM report***

****

****

****

****

****

**Go to os level read the report and analyze it.**