Oracle findings

This section includes the following topics:

About the Oracle Findings view

The Findings view displays recommendations that can be used to create a better execution plan and improve the performance of a statement. The Findings vary according to the type of operation in a statement, the Precise product and the technology.

Precise for Oracle is capable of providing additional insight into a problem by helping you focus on the item or object that is causing the problem. When you click on certain findings in the SQL tab or the Object tab, and you launch to the tab that can provide additional information on the problem, the row representing the item or object that is most likely causing the problem is highlighted. This provides a further indication as to where you should focus your analysis.

About the Highlights area

The Highlights area displays a brief description of the findings for the selected type of operation.

The What to do next area displays one or more recommended steps to identify the cause of the problem. Carefully review all data for the finding before continuing.

About the Advice area

The Advice area displays one or more recommended options to resolve or reduce the problem for the selected finding. Carefully review all data for the finding and then perform the advice that best suits your needs.

How to investigate findings

When you start investigating the findings, it is good practice to start with the finding that has the highest severity ranking in the Findings table.

To investigate a finding

1. Identify the finding with the highest severity ranking in the Findings table.
2. Select the finding type to view additional information on the selected type of operation.
3. Read the **Highlights** and **What to do next** areas for the finding.
4. After you have studied all of the information provided, read the **Advice** area, and then perform the recommendation that best suits your needs.
5. Follow up on performance to verify that the problem was resolved.

About Instance findings

Several Dashboard tab findings exist in the table to help the user. The Dashboard tab for an instance has the following findings:

- High CPU Wait
- High Other Host Wait
- High Memory Wait
- High Shared Pool Wait
- High Rollback Segment Wait
- High Redo Log Buffer Wait
- High Log Switch and Clear Wait
- High RAC/OPS Wait
- High Other Lock Wait
- High Background Process Wait
- High Parallel Query Server Wait
- High Buffer Wait
- High Other Wait
- High Remote Query Wait
- High Client Communication Wait
- High Resource Manager Wait
- High MTS Wait
- Heavy Statement
- Frequently Executed Statement
- Heavily Accessed Object
- Locked Object
- High Sorts on Disk
- High Undo Activity
- Heavily Accessed Cluster
- Locked Cluster

- Storage Contention on Device (Clariion)
- Storage Contention on Device (Symmetrix Thick)
- Storage Contention on Device (Symmetrix Thin)
- Storage Contention on Device (Symmetrix F.A.S.T. VP)
- Storage Contention on Redo Logs and DB Files
- Storage Contention on Temporary Objects
- Heavy Storage Device Holding Undo Objects
- Unbalanced Storage Devices Activity
- Heavy J2EE Caller Service
- High SQL Executions for J2EE Caller Service

High CPU Wait

Your instance has spent much of its In Oracle time waiting for CPU.

Table 1 High CPU Wait

	Description
What to do next	Perform one of the following options: • Examine high wait for CPU statements in the Activity tab. • Examine high CPU statements usage in the Activity tab. • Examine CPU utilization in the Statistics tab. • Try to identify the system processes consuming CPU resources using the Insight Savvy for OS tool.
Advice	Perform one of the following options: • Identify other processes in the system. • The instance has spent much of its In Oracle time waiting for CPU; every process running in your system affects the available CPU resources. Effort spent tuning non-Oracle factors can improve Oracle performance. • Identify heavy statements using CPU or Waiting for CPU and try to tune them.

High Other Host Wait

Your instance has spent much of its In Oracle time in Other Host Wait.

Table 2 High Other Host Wait

	Description
What to do next	Perform one of the following options: • Examine heavy wait for Other Host Wait statements in the Activity tab. • Try to identify the system processes consuming OS resources using the Insight Savvy for OS tool.
Advice	Other Host Wait can result from any of the following causes: asynchronous I/O, gateways, or the use of NFS and TP monitors. Check the statements and programs suffering from this state and check whether the above resources are being utilized efficiently.

High Memory Wait

Your instance has spent much of its In Oracle time waiting for memory.

Table 3 High Memory Wait

	Description
What to do next	Perform one of the following options: • Examine high Memory Wait statements in the Activity tab. • Try to identify the system processes consuming memory using the Insight Savvy for OS tool.

Advice	Perform one of the following options: • Identify other processes in the system. The instance has spent much of its In Oracle time waiting for memory; every process running in your system affects the available memory. Effort spent tuning non-Oracle factors can improve Oracle performance. For example: the result of setting a high number of MAX_PARALLEL_SERVERS when using a parallel query option. • Identify heavy statements using High Memory Wait and try to tune them.

High Shared Pool Wait

Your instance has spent much of its In Oracle time waiting for the group event Shared Pool Wait.

Table 4 High Shared Pool Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped into the Wait for Shared Pool in the Statistics tab. Determine the dominant Oracle event and follow the tuning scenario set by this event. • Examine high Shared Pool Wait statements in the Activity tab.
Advice	Common scenarios for this wait occur when the shared pool is either too small or too big. Verify that your shared pool is sized according to the type of application being used (cursor sharing, literals usage, and so on.)

High Rollback Segment Wait

Your instance has spent much of its In Oracle time waiting for the group event Rollback Segment Wait.

Table 5 High Rollback Segment Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped into the Wait for Rollback segment. Determine the dominant Oracle event and follow the tuning scenario set by this event. • Examine the statements or objects with the highest values of Rollback Segment Wait and determine which applications are creating this wait.
Advice	To reduce contention on the rollback segment, consider one of the following solutions: • Add rollback segments, moving them into a less busy tablespace. • Change the application flow or change the rollback policy (using no logging on specific objects).

High Redo Log Buffer Wait

Your instance has spent much of its In Oracle time waiting for the Redo Log.

Table 6 High Redo Log Buffer Wait

	Description
What to do next	Perform one of the following options: • Examine the related Oracle events (lower area), Redo Activity (upper area), to determine the problem type in the Statistics tab. • Examine high Redo Log Buffer Wait statements in the Activity tab.

Advice	Use any one of the typical problem scenarios described below. • If the log buffer size is too small, this usually results in long waits for the Log Buffer Space event. Consider increasing the Log_buffer parameter. • If the log buffer size is too big, this usually results in a low number of user commits, high redo wastage statistics, and long waits for the Log File Sync event. Consider decreasing the Log_buffer parameter and/or the hidden LOG_I/O_SIZE parameter. • If there are too many commits, this usually results in long waits for the Log File Sync event and the number of user commits is very high. Consider changing the application flow and logic (by decreasing the commit frequency or using bulk commits [resulting in larger transactions]). • If the LGWR is too slow, check whether Log File Sync is still the dominant event. This may be due to high values for Log File Parallel Write, or because there are not many commits. This may mean that the LGWR is underperforming. Consider moving the log file to a faster, dedicated device. Whenever the Log Buffer Space and Log File Sync events occur together, consider changing the hidden LOG_I/O_SIZE parameter.

High Log Switch and Clear Wait

Your instance has spent much of its In Oracle time waiting for the group event Log Switch and Clear.

Table 7 High Log Switch and Clear Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped into the Wait for Log Switch and Clear. Determine the dominant Oracle event and follow the tuning scenario set by this event in the Statistics tab. • Examine the statements with the highest values for this wait and determine which applications are creating this wait in the Activity tab.
Advice	If the related Oracle events show too many log switches, try and reduce them by one of the following options: • Increase the Redo log size (when the Log File Switch (checkpoint incomplete) and/or Log File Switch Completion event is dominant). • Change the application flow or logging policy (by changing the commit frequency or using No Logging on specific objects). There can be other reasons for a high Log Switch and Clear wait, such as an LGWR delay where the files cannot be switched until ARCH archiving is completed. This is usually caused by the Log File Switch (archiving needed) event.

High RAC/OPS Wait

Your instance has spent much of its In Oracle time waiting for RAC or OPS.

Table 8 High RAC/OPS Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped in the Oracle RAC/OPS Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event in the Statistics tab. • Examine Load balancing between RAC instances in the Activity tab. • Examine Heavy objects suffering from RAC Waits in the Activity tab.
Advice	There are two typical scenarios relevant to RAC Waits. Launch to the Dashboard RAC Database view. This view compares the selected instance with other instances in the same database. From the RAC Database view, examine each of the following issues: • If there is a balancing instances issue, launch to the Activity tab and identify the root cause for this unbalanced issue. • If there is a RAC Wait event, compare them to other events in the database in the Statistics tab. • If there are Objects suffering from RAC Waits, launch to the Activity tab and identify their use across Database instances.

High Other Lock Wait

Your instance has spent much of its In Oracle time waiting for a latch.

Table 9 High Other Lock Wait

	Description

What to do next	Perform one of the following options: • Examine Latching view overtime and the Oracle latches that are grouped in the Other Lock Wait. Then determine the dominant Oracle latch or enqueue and follow the tuning scenario set by this latch in the Statistics tab. • Examine high Other Lock Wait statements in the Activity tab.
Advice	Examine the Oracle latches that are grouped into the Other Lock Wait. Determine the dominant Oracle latch or enqueue and follow the tuning scenario set by this event.

High Background Process Wait

Your instance has spent much of its In Oracle time waiting for the group event Background Process Wait.

Table 10 High Background Process Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped in the Background Process Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event in the Statistics tab. • Examine high Background Process Wait statements in the Activity tab.
Advice	Examine the Oracle events that are grouped into the Background Processes Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event.

High Parallel Query Server Wait

Your instance has spent much of its In Oracle time waiting for the group event Parallel Query Server Wait.

Table 11 High Parallel Query Server Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped in the Parallel Query Server Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event in the Statistics tab. • Examine high Parallel Query Server Wait statements in the Activity tab.
Advice	Examine the Oracle events that are grouped into the Parallel Query Server Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event.

High Other Wait

Your instance has spent much of its In Oracle time waiting for the group event Other Wait.

Table 12 High Other Wait

	Description
What to do next	Perform one of the following options: • Examine the Oracle events that are grouped as Other Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event in the Statistics tab. • Examine high Other Wait statements in the Activity tab.
Advice	In the Statistics tab, examine the Oracle events that are grouped as Other Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event.

High Buffer Wait

Your instance has spent much of its In Oracle time waiting for database buffers.

Table 13 High Buffer Wait

	Description

What to do next	Perform one of the following options: • Examine the Oracle events that are grouped in the Buffer Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event in the Statistics tab. • If the dominant Oracle event is buffer busy, launch into the Activity tab and locate the objects with the highest buffer wait.
Advice	There are two typical scenarios relevant to buffer wait that are determined by the dominant Oracle event: • Buffer Busy wait event There is high contention on specific table blocks. To reduce contention, increase the Freelists or the Pctfree for the table. • Free Buffer Wait event There are no available buffers in the buffer cache. This is usually an I/O wait related problem. Do the following: ◦ Try and tune the heaviest statements or objects. ◦ If this event persists after statement tuning, try and increase DBWR throughput by adding more DBWR processes or DBWR_I/O_SLAVES. Increase the buffer cache size.

High Remote Query Wait

Your instance has spent much of its In Oracle time waiting for remote queries to complete.

Table 14 High Remote Query Wait

	Description
What to do next	Perform one of the following solutions: • Examine DBLink relevant statistics in the Statistics tab. • Try to identify the statement suffering from a high wait for Oracle Comm Wait in the Activity tab.
Advice	To reduce remote access wait time, consider the following options: • Launch Precise for Oracle on the remote instance, locate the remote query, and tune the statement. • Control the driving instance executing the statement by using the DRIVING_SITE hint. • Tune SQL*Net throughput by checking SDU and TDU settings. • Include TCP.NODELAY=yes parameter in the SQLNET.ORA configuration file. • Specify the parameters TDU and SDU in the connection description on the application client. For example: in the TNSNAMES.ORA configuration file and in the LISTENER.ORA configuration file on the Oracle database server.

High Client Communication Wait

Your instance has spent much of its In Oracle time waiting for data from the Oracle server.

Table 15 High Client Communication Wait

	Description
What to do next	Perform one of the following options: • Examine SQL*Net relevant statistics in the Statistics tab. • Identify top statements that suffer from the high Oracle Client Comm Wait in the Activity tab.
Advice	To reduce the amount of data transferred using SQL*Net, consider the following solutions: • Change the SQL statements so that only needed data is transferred to the client. For example, select only columns that are actually used in the client application and only retrieve required rows. • Check that you are not suffering implicit delays in your TCP/IP network by specifying TCP.NODELAY=yes in the SQLNET.ORA files for the Oracle listener and for the application client. • Specify the parameters TDU and SDU in the connection description on the application client. For example: in the TNSNAMES.ORA configuration file and in the LISTENER.ORA configuration file on the Oracle database server.

High Resource Manager Wait

Your instance has spent much of its In Oracle time waiting for the group event Resource Manager Wait.

Table 16 High Resource Manager Wait

	Description
What to do next	Perform one of the following options: • Examine the top statements that suffer from high Resource Manager Wait, and the influence of the wait on the instance in the Activity tab. • Try to identify the most dominant events related to Resource Manager Wait in the Statistics tab.
Advice	To reduce the high Resource Manager Wait, consider the following solutions: • Examine the Overtime graph for the instance, to measure the severity of the Resource Manager Wait. Also check the top statements that suffer from Resource Manager Wait. Checking the statements can reveal whether the problem relates to a specific statement. • In the Statistics tab, examine the Oracle events that are grouped to Resource Manager Wait. Determine the dominant Oracle event and follow the tuning scenario set by this event.

High MTS Wait

Your instance has spent much of its In Oracle time waiting for MTS Wait.

Table 17 High MTS Wait

	Description
What to do next	Perform one of the following options: • Examine high MTS wait by programs in the Activity tab. • Try to identify the most dominant events related to MTS Wait in the Statistics tab.
Advice	To reduce high MTS wait, consider the following solutions: • Examine the Overtime graph for the instance, to measure the severity of the MTS Wait. Look for top programs suffering from MTS Wait. • Examine the memory usage of MTS connections by querying v$sesstat and v$sessions. Measure the maximum amount of UGA memory used at any given moment and divide that amount by the number of current user sessions. This determines the average amount of memory each connection allocates. • If your application is not suited to MTS, use "dedicated" connections which create a separate server process for each user connection.

Heavy Statement

The statement is a major consumer of Oracle resources. By tuning the statement, you may free resources needed by other statements and processes.

Table 18 Heavy Statement

	Description
What to do next	Try to determine what is causing the statement's high resource consumption. In the SQL tab, examine the text of the relevant statement, and its findings, execution plan, change data and statistics.
Advice	For resource consumption, these are the possible scenarios: • Object wait on statement objects. Use SQL findings to identify the heaviest waiting object. • High CPU consumption without object wait. Check the number of executions (to identify possible infinite loops) or intensive statements (with a low average time but a high aggregated CPU time). • Instance-related wait (such as: internal lock wait, shared pool wait, and redo log wait). In this case, switch to the Statistics tab and examine the breakdown of this state in Oracle events. • A change in the execution plan. If this is responsible for the statement's performance degradation, check the History tab and the All Plans view to identify the actual change that caused the slowdown. • Check the Binds tab for possible offensive values resulting in differing execution plans and performance.

Frequently Executed Statement

The statement is a major consumer of Oracle resources. This statement is frequently executed with a low In Oracle time average.

Table 19 Frequently Executed Statement

	Description

What to do next	Go to the Activity tab and examine the statement executors (programs and users).
Advice	Examine the Activity tab for statement exaggerated usage patterns. Try tuning scenarios for resource consumption from the following list: • Object wait on the statement objects. Use SQL findings to identify the heaviest waiting object. • High CPU consumption without object wait. Check the number of executions (to identify possible infinite loops) or intensive statements (with a low average time but a high aggregated CPU time). • Instance related wait (such as: Internal lock wait, Shared pool wait, Redo log wait). In this case, switch to the Statistics tab and examine the breakdown of this state in Oracle events. • A change in the execution plan. This is responsible for the statements performance degradation. Use history and all plans to identify the actual change that caused the slowdown. • Check the binds tab for possible offensive values resulting in differing execution plans and performance.

Heavily Accessed Object

Much of the instance In Oracle time was spent on waits (lock, I/O, Buffer, and so on) for the object.

Table 20 Heavily Accessed Object

	Description
What to do next	Perform one of the following options: • To tune the object, go to the Tune Object tab. • Examine the waits for the object in the Activity tab.
Advice	To reduce the Waits for the object, follow the findings instructions in the Object tab. In the Object tab, you can examine the following object data: • Access path pattern to the object (full scans, range scans, and so on) • Read-Write access patterns • Partition usage (such as whether there are extreme cases in use) • Object changes versus performance changes

Locked Object

Much of the instance In Oracle time is spent waiting for a lock on the table.

Table 21 Locked Object

	Description
What to do next	Perform one of the following options: • To extensively tune the object, go to the Tune Object tab. • Examine the Lock for the statement in the Activity tab.
Advice	To reduce the lock wait for the table, consider the following solutions: • Check if the lock appears in the Current tab. If so, examine the lock chain to discover which statement is holding the lock. • Try to identify the locking statement in the Activity tab, using narrow time frames that match the lock periods. Focus on the locked table and its associated statements. The immediate suspect is the DML statements (and update queries) that are not waiting for locks.

High Sorts on Disk

The result table for a sort operation could not be completed in memory and was performed on a temporary tablespace.

Table 22 High Sorts on Disk

	Description

What to do next	In the Activity tab, examine temporary tablespace overtime I/O consumption, and the statements using temporary tablespace I/O.
Advice	To reduce the I/O consumption for the sort operation, consider the following solutions: • Change the SORT_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid sort and hash area size problems. You can either change the values for a specific session, using the `Alter Session` command, or for the entire instance. • Try to identify statement-originated sorts. If there are many sorts located in a few statements, try to solve the problem at the statement level by doing one of the following: ◦ Add an index to prevent a sort. If your statement has an Order by clause that has columns for a single table, check whether you can add an index. In some cases, you may get an index recommendation that prevents a sort. If you are considering adding an index, check the effect of adding that index in the What-If tab. ◦ Identify the heavy sort or hash consumer step. Run a statement with statistics_level=all. Click the Run and Compare tab. Examine LAST_TEMPSEG_SIZE and MAX_TEMPSEG_SIZE in the extended section of the run results. Change the SORT_AREA_SIZE to a higher value.

High Undo Activity

Much of the instance I/O is spent waiting for the Undo object.

Table 23 High Undo Activity

	Description
What to do next	Examine Undo activity over time and the statement accessing it, in the Activity tab.
Advice	Examine Undo behavior over time, identify the statement accessing it, and try to tune them.

Heavily Accessed Cluster

Much of the instance I/O is spent waiting for the cluster.

Table 24 Heavy Cluster Activity

	Description
What to do next	Examine Cluster activity over time and the statement accessing it, in the Activity tab.
Advice	Examine Cluster behavior over time, identify the statement accessing it, and try to tune them.

Locked Cluster

Much of the instance time is spent waiting for a lock on the cluster.

Table 25 Cluster Locks

	Description
What to do next	Examine the lock for the statement in the Activity tab.
Advice	To reduce the lock wait for the table, consider the following solutions: • Check if the lock appears in the Current tab. If so, examine the lock chain to discover the statement holding the lock. • Try to identify the locking statement in the Activity tab, using narrow time frames that match the lock periods. Focus on the locked table and associated statements. The immediate suspect is the DML statements (and update queries) that are not waiting for locks.

Storage Contention on Device (Clariion)

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 26 Storage Contention On Device

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.
Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Device (Symmetrix Thick)

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 27 Storage Contention On Device

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.
Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Device (Symmetrix Thin)

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 28 Storage Contention On Device

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.

Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Device (Symmetrix F.A.S.T. VP)

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 29 Storage Contention On Device

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.
Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Redo Logs and DB Files

Redo/Transaction Log files are frequently accessed by the database. The majority of the operations performed are writing commands, which cause a heavy load on the underlying disks.

As these files are considered heavy I/O consumers, it is highly recommended to place them on a separate disk without other database files. Separating the Redo/Transaction Logs files by placing them on different volumes (e.g. E:/ and F:/) may not be enough, as the storage devices (LUNs) and physical disks may be shared between several file systems and volumes.

Table 30 Storage Contention on Redo Logs and DB Files

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.
Advice	It has been detected that the Redo/Transaction Log files share the storage devices (LUNs) with other database files. Consult the storage administrator about provisioning the storage devices (LUNs) better to avoid this.

Storage Contention on Temporary Objects

Temporary tablespace files are frequently accessed by the database. The majority of the operation performed are writing commands, which cause a heavy load on the underlying disks.

As these files are considered heavy I/O consumers, it is highly recommended to place them on a separate disk without other database files. Separating the temporary tablespace files by placing them on different volumes (e.g. E:/ and F:/) may not be enough, as the storage devices (LUNs) and physical disks may be shared between several file systems and volumes.

Table 31 Storage Device on Temporary Objects

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.
Advice	It has been detected that the temporary tablespace files share the storage devices (LUNs) with other database files. Consult the storage administrator about provisioning the storage devices (LUNs) better to avoid this.

Heavy Storage Device Holding Undo Objects

Undo tablespace files are frequently accessed by the database. The majority of the operation performed are writing commands, which cause a heavy load on the underlying disks.

As these files are considered heavy I/O consumers, it is highly recommended to place them on a separate disk without other database files. Separating the undo tablespace files by placing them on different volumes (e.g. E:/ and F:/) may not be enough, as the storage devices (LUNs) and physical disks may be shared between several file systems and volumes.

Table 32 Heavy Storage Device Holding Undo Objects

	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage device statistics and contention on the Raid Group and Physical Disks.
Advice	It has been detected that the undo tablespace files share the storage devices (LUNs) with other database files. Consult the storage administrator about provisioning the storage devices (LUNs) better to avoid this.

Unbalanced Storage Devices Activity

There are several storage devices (LUNs) allocated to the instance. However, the I/O activity is not spread evenly across these storage devices. The contention on the heavy storage devices increases the response time for the activities run on them. Such a situation can be caused by imbalanced internal database activity, contention on the storage device by other applications or an inefficient RAID policy.

Table 33 Unbalanced Storage Devices Activity

	Description
What to do next	• Compare the storage devices activity over time. • Examine the storage devices statistics.
Advice	• In the Activity tab, check which database files are the most I/O consuming and spread them evenly across the storage devices. • Consult with the storage administrator and check for other applications using the same storage devices or their underlying physical disks. • Consult with the storage administrator about the RAID policy. A different striping may spread the I/O load across the storage devices.

Heavy J2EE Caller Service

The J2EE Caller Service is a major consumer of Oracle resources. By tuning it, you may free resources needed by other Caller Services and processes.

Table 34 Heavy J2EE Caller Service

	Description
What to do next	• Try to determine what is causing the J2EE Caller Service's high resource consumption. • In the Activity tab, examine its findings, relevant statements, objects, and the Oracle resources

Advice	For resource consumption, these are the possible scenarios: • **Object bottleneck**. Examine the resource consumption of objects in context of this Caller Service in the Activity tab. • **Statement bottleneck**. Examine the resource consumption of SQL statements in context of this Caller Service in the Activity tab. • **Instance-related wait (such as: internal lock wait, shared pool wait, and redo log wait)**. In this case, switch to the Statistics tab and examine the breakdown of this state in Oracle events.

High SQL Executions for J2EE Caller Service

The J2EE Caller Service is a major consumer of Oracle resources, issuing and exceptionally high number of SQL Statements executions. By tuning it and reducing the number of executions, you may free resources needed by other Caller Services and processes.

Table 35 High SQL Executions for J2EE Caller Service

	Description
What to do next	• Try to determine what is causing the J2EE Caller Service's high resource consumption and high number of executions. • In the J2EE application Expert view examine the entry point logic to detect redundant or infinite loop executions. • In the Activity tab, examine its findings, relevant statements, objects, and the Oracle resources.
Advice	For resource consumption, these are the possible scenarios: • **Object bottleneck**. Examine the resource consumption of objects in context of this Caller Service in the Activity tab. • **Statement bottleneck**. Examine the resource consumption of SQL statements in context of this Caller Service in the Activity tab. • **Instance-related wait (such as: internal lock wait, shared pool wait, and redo log wait)**. In this case, switch to the Statistics tab and examine the breakdown of this state in Oracle events.

About Activity tab findings

Several Activity tab findings exist in the table to help the user. These findings are available in Service Caller and Web Transactions entities.

The Activity tab has the following findings:

- Slow Statement
- Heavy Full Scan
- High Redo Log Buffer Wait
- High Buffer Wait
- High Temporary I/O
- Heavy Statement

Slow Statement

The statement is identified with a high average time in the chosen context. By tuning the statement, you may improve the application response time.

Table 36 Slow Statement

	Description
What to do next	Try to determine what is causing the statement's high average time. In the SQL tab, examine the text of the relevant statement, and its findings, execution plan, change data and statistics. The Recommend is executed as explain in the Recommend tab.
Advice	For resource consumption, these are the possible scenarios: • Object wait on statement objects. Use SQL findings to identify the heaviest waiting object. • High CPU consumption without object wait. Check the number of executions (to identify possible infinite loops) or intensive statements (with a low average time but a high aggregated CPU time). • Instance-related wait (such as: internal lock wait, shared pool wait, and redo log wait). In this case, switch to the Statistics tab and examine the breakdown of this state in Oracle events. • A change in the execution plan. If this is responsible for the statement's performance degradation, check the History tab and the All Plans view to identify the actual change that caused the slowdown. • Check the Binds tab for possible offensive values resulting in differing execution plans and performance.

Heavy Full Scan

The J2EE Caller Service in the chosen context, spent a high amount of its resources on performing disk related full scans on the referenced object.

Table 37 Heavy Full Scan

	Description
What to do next	Try to determine what is causing the J2EE Caller Service to spent a high amount of its resources on performing disk related full scans on the referenced objects: • Get index recommendations for the SQL statements of the Caller Service. When activating the Recommend, Precise generates a filtered statements workload in a context of the time frame and caller service/Web transaction. Oracle advisor provides enhanced recommendations based on the provided workload. The recommendation can be evaluated by the What-if analysis. • Examine the Scattered and Direct I/O, translated into full scan access, over time and related objects. • Examine the Scattered and Direct I/O, translated into full scan access, over time and related devices.
Advice	To reduce the I/O consumption for the J2EE Caller Service, consider the following solutions: • Create an index matching the statement's predicates. • Change the DB_FILE_MULTIBLOCK_READ_COUNT parameter in INIT.ORA to a higher value. • Partition the objects according to the best predicates existing in the J2EE Caller Service's statements. • Use a parallel query option. • Move the objects to another tablespace with a higher block size.

High Redo Log Buffer Wait

The J2EE Caller Service in the chosen context, spent much of its In Oracle time waiting for the Redo Log events.

Table 38 High Redo Log Buffer Wait

	Description
What to do next	Perform one of the following options: • Examine the top statements waiting for the Redo Log Buffer events. • Examine the Redo Log Buffer events for the entire instance.
Advice	Use any one of the typical problem scenarios described below. • If the log buffer size is too small, this usually results in long waits for the Log Buffer Space event. Consider increasing the Log_buffer parameter. • If the log buffer size is too big, this usually results in a low number of user commits, high redo wastage statistics, and long waits for the Log File Sync event. Consider decreasing the Log_buffer parameter and/or the hidden LOG_I/O_SIZE parameter. • If there are too many commits, this usually results in long waits for the Log File Sync event and the number of user commits is very high. Consider changing the application flow and logic (by decreasing the commit frequency or using bulk commits [resulting in larger transactions]). • If the LGWR is too slow, check whether Log File Sync is still the dominant event. This may be due to high values for Log File Parallel Write, or because there are not many commits. This may mean that the LGWR is underperforming. Consider moving the log file to a faster, dedicated device. Whenever the Log Buffer Space and Log File Sync events occur together, consider changing the hidden LOG_I/O_SIZE parameter.

High Buffer Wait

The J2EE Caller Service in the chosen context, spent much of its In Oracle time waiting for database buffer events.

Table 39 High Buffer Wait

	Description
What to do next	Perform one of the following options: • Examine the top objects waiting for the Buffer events. • Examine the Buffer events for the entire instance.

Advice	There are two typical scenarios relevant to buffer wait that are determined by the dominant Oracle event: • Buffer Busy wait event There is high contention on specific table blocks. To reduce contention, increase the Freelists or the Pctfree for the table. • Free Buffer Wait event There are no available buffers in the buffer cache. This is usually an I/O wait related problem. Do the following: ◦ Try and tune the heaviest statements or objects. ◦ If this event persists after statement tuning, try and increase DBWR throughput by adding more DBWR processes or DBWR_I/O_SLAVES. ◦ Increase the buffer cache size.

High Temporary I/O

The J2EE Caller Service in the chosen context, spent much of its In Oracle time waiting for the waiting on sort or hash operations performed on a temporary tablespace.

Table 40 High Temporary I/O

	Description
What to do next	Perform the following: • Examine the temporary I/O activity over time and related statements
Advice	To reduce the I/O consumption for the sort operation, consider the following solutions: • Change the SORT_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid sort and hash area size problems. You can either change the values for a specific session, using the `Alter Session` command, or for the entire instance. • Try to identify statement-originated sorts. If there are many sorts located in a few statements, try to solve the problem at the statement level by doing one of the following: ◦ Add an index to prevent a sort. If your statement has an Order by clause that has columns for a single table, check whether you can add an index. In some cases, you may get an index recommendation that prevents a sort. If you are considering adding an index, check the effect of adding that index in the What-If tab. ◦ Identify the heavy sort or hash consumer step. Run a statement with statistics_level=all. Click the Run and Compare tab. Examine LAST_TEMPSEG_SIZE and MAX_TEMPSEG_SIZE in the extended section of the run results. Change the SORT_AREA_SIZE to a higher value.

Heavy Statement

The statement is identified as a major consumer of the selected J2EE Caller's Oracle resources. By tuning the statement, you may improve the application response time.

Table 41 Heavy Statement

	Description
What to do next	Perform one of the following options: • Tune the statement using SQL Tune methodology • Examine the statement activity over time • Tune the caller service using J2EE application expert views
Advice	For resource consumption, these are the possible scenarios: • Object wait on statement objects. Use SQL findings to identify the heaviest waiting object. • High CPU consumption without object wait. Check the number of executions (to identify possible infinite loops) or intensive statements (with a low average time but a high aggregated CPU time). • Instance-related wait (such as: internal lock wait, shared pool wait, and redo log wait). In this case, switch to the Statistics tab and examine the breakdown of this state in Oracle events. • A change in the execution plan. If this is responsible for the statement's performance degradation, check the History tab and the All Plans view to identify the actual change that caused the slowdown. • Check the Binds tab for possible offensive values resulting in differing execution plans and performance.

About SQL tab findings

Several SQL tab findings exist to help the user. The SQL tab has the following findings:

Sorts Performed on Disk

The result table for a sort operation could not be completed in memory and was performed on a temporary tablespace.

Table 42 Sorts Performed on Disk

	Description
What to do next	In the Activity tab, examine temporary tablespace overtime I/O consumption for the statement, and the programs activating the statement.
Advice	To reduce the I/O consumption for the sort operation, consider the following solutions: • Change the SORT_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid sort and hash area size problems. You can either change the values for a specific session using the `Alter Session` command, or for the entire instance. • Try to identify statement originated sorts. If there are many sorts located in a few statements, try to solve the problem at the statement level by one of the following options: ○ Add an index to prevent a sort. If your statement has an Order by clause that has columns for a single table, check whether you can add an index. In some cases, you may get an index recommendation that prevents a sort. If you are considering adding an index, you must check the effect of that index. To analyze that effect, launch to the What-If tab. ○ Identify the heavy sort or hash consumer step. Run a statement with statistics_level=all. Click the Run & Compare tab. Examine the LAST_TEMPSEG_SIZE and MAX_TEMPSEG_SIZE in the extended section of the run results, and set parameters according to the first advice.

No Parallel Processes Available

Some of the executions for the statement were not run in parallel; they worked serially. Oracle has reached the threshold of the MAX_PARALLEL_SERVERS and was not able to allocate parallel processes for the statement.

Table 43 No Parallel Processes Available

	Description
What to do next	Select the finding type to see the minimum and maximum parallel processes available for the statement in the Activity tab.
Advice	To prevent this problem, consider one of the following solutions: • Increase the MAX_PARALLEL_SERVERS parameter in INIT.ORA while closely monitoring the effect on wait for memory. This will lower the chances for a serial execution, but may cause a high memory consumption. • Set PARALLEL_AUTOMATIC_TUNING parameter in INIT.ORA to TRUE. This will make Oracle use more sophisticated algorithms in determining the number of PQ processes for each session.

Bottleneck in Remote Access

Your statement has spent much of its In Oracle time waiting for a remote query to complete.

Table 44 Bottleneck in Remote Access

	Description
What to do next	Perform one of the following options: • Select the findings type to see the SQL text sent to the remote database on the Statistics tab, Other column. • Examine DBLink relevant statistics in the Statistics tab.
Advice	To reduce remote access wait time, consider the following options: • Launch Precise for Oracle on the remote instance, locate the remote query, and tune the statement. • Control the driving instance executing the statement by using the DRIVING_SITE hint. • Tune SQL*Net throughput by checking SDU and TDU settings. • Include TCP.NODELAY=yes parameter in the SQLNET.ORA configuration file. • Specify the parameters TDU and SDU in the connection description on the application client. For example: the TNSNAMES.ORA configuration file and in the LISTENER.ORA configuration file in the Oracle database server.

Heavy Scattered I/O on Index

Statement I/O is spent on scattered I/O (usually representing a full scan) on the index specified in the Object column.

Table 45 Heavy Scattered I/O on Index

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the index consumption for the statement in the Activity tab.
Advice	To reduce the index I/O consumption, consider the following solutions: • If the index is not used to prevent a sort operation, consider adding the INDEX_FFS hint and change DB_FILE_MULTIBLOCK_READ_COUNT parameter to a higher value. • Switch to another index or add a new index. • Use a parallel query option. • Move the index to another tablespace with a higher block size. Findings refer to the whole statement - not to a specific execution plan. If a step does not exist in the selected execution plan (unless this is due to an index overhead), switch to another plan and locate the relevant step.

Heavy Sequential I/O on Index

Statement I/O is spent on sequential I/O (usually representing a range scan) on the index specified in the Object column. If the statement is DML and the index is not used in the execution plans, then the I/O represents the index maintenance overhead, caused by fetching the index blocks for update to memory.

Table 46 Heavy Sequential I/O on Index

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to see if the index is used in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the index overtime consumption for the statement in the Activity tab.
Advice	If this is the result of a range scan, consider one of the following solutions: • If the index is not used to prevent a sort operation, consider adding the INDEX_FFS hint and change DB_FILE_MULTIBLOCK_READ_COUNT (now standing on X) to a higher value. • Switch to another index, add a new index, or change the index structure. • Add columns to the index to enable index only access. • Switch to a full table scan • Partition the table or the index • Create a cluster with just one table in it and a cluster key used as the index key. If this is due to an index overhead, consider reducing the index overhead by deleting unused indices, or unused columns in used indices. Findings refer to the whole statement - not to a specific execution plan. If a step doesn't exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Heavy Scattered I/O on Table

Statement I/O is spent on scattered I/O (usually representing a full scan) on the table specified in the Object column.

Table 47 Heavy Scattered I/O on Table

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the table consumption for the statement in the Activity tab
Advice	To reduce the I/O consumption for the table, consider the following solutions: • Create an index matching the statement's predicates. • Change the DB_FILE_MULTIBLOCK_READ_COUNT parameter in INIT.ORA to a higher value. • Partition the table according to the best predicates existing in the statement. • Use a parallel query option. • Move the table to another tablespace with a higher block size. Findings refer to the whole statement - not to a specific execution plan. If a step doesn't exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Heavy Sequential I/O on Table

Statement I/O is spent on sequential I/O (usually representing table access by rowid following an index range scan) on the table specified in the Object column.

Table 48 Heavy Sequential I/O on Table

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the table consumption for the statement in the Activity tab.

Advice	To reduce the I/O consumption for the table, consider the following solutions: • Change the index structure by adding a better filtering column, or change the column sequence to improve the matching and screening performed in the index tree. • Add columns to the index to enable index only access. • Switch to a full table scan. • Partition the table or the index. • Create a cluster with just one table in it and a cluster key used as the index key. Findings refer to the whole statement - not to a specific execution plan. If a step doesn't exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Heavy I/O Due to Direct Access

Statement I/O is spent on direct I/O (usually representing the SQL Loader in the direct path), on the object specified in the Object column.

Table 49 Heavy I/O Due to Direct Access

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the object (table/index) consumption for the statement in the Activity tab.
Advice	To reduce the I/O consumption for the object, consider the following solutions: • If this is the result of an SQL loader utility in a direct path, verify that the temporary segments for the index updates are on a fast device. • Use the SINGLEROW option. Findings refer to the whole statement - not to a specific execution plan. If a step doesn't exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Heavy I/O Due to Other Access

Statement I/O is spent on another I/O on the object specified in the Object column.

Table 50 Heavy I/O Due to Other Access

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the object (table/index) consumption for the statement in the Activity tab. Findings refer to the whole statement - not to a specific execution plan. If a step doesn't exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Statement State Row Lock

Much of the statement I/O is spent on waiting for a lock on the table specified in the Object column.

Table 51 Statement State Row Lock

	Description
What to do next	Select the findings type to examine lock for the statement in the Activity tab.

Advice	To reduce the lock wait for the table, consider the following solutions: • Check to see if the lock appears in the Current tab. If so, examine the lock chain to identify the statement holding the lock. • Try to identify the locking statement in the Activity tab using the narrow time frames matching the lock periods. Now focus on the locked table and associated statements. The DML statement (and update queries) that are not waiting for locks are the immediate suspects. Findings refer to the whole statement - not to a specific execution plan. If a step doesn't exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Buffer Wait Contention

Your statement has spent much of its In Oracle time on Buffer Wait. This usually occurs because one of two possible scenarios:

- Contention on a table buffer in Insert statements (Buffer Busy Wait), or
- Lack of free buffers when trying to load blocks from a disk (Free Buffer Wait)

Table 52 Buffer Wait Contention

	Description
What to do next	Perform one of the following options: • Select the findings type to identify the tables suffering from Buffer Wait in the Objects tab, or • Examine Buffer Wait sub-state events in the Statistics tab.
Advice	When Buffer Busy Wait is the more dominant Oracle event, consider the following options: • Increase the free lists for the table to reduce the chances for contention. • Increase the PCTFREE parameter or decrease the block size for the table to spread the data among many blocks and reduce the chances for contention. When Free Buffer Wait is the more dominant Oracle event, consider the following options: • Tune the object's access, so as to reduce the number of blocks fetched. • If this is a global instance problem, increase the number of DBWR processes or I/O slaves.

Redo Log Activity

Your statement has spent much of its In Oracle time waiting for Redo Log Wait.

Table 53 Redo Log Activity

	Description
What to do next	Perform one of the following options: • Examine Redo Log Wait overtime I/O consumption for the statement in the Activity tab, or • Examine Redo Log Wait relevant statistics in the Statistics tab.
Advice	To reduce Redo Log Wait, consider the following solution: • Check Dashboard findings and overtime graphs to see the overall Redo Log activity for the instance.

Undo Activity

Your statement has spent much of its In Oracle time waiting for undo.

Table 54 Undo Activity

	Description
What to do next	Perform one of the following options: • Examine Undo Wait overtime I/O consumption for the statement in the Activity tab, or • Examine Undo Wait relevant statistics in the Statistics tab.

Advice	To reduce Undo Wait time, consider the following solutions: • Check Dashboard findings and overtime graphs to see the overall undo activity for the instance.

RAC Wait

Your instance has spent much of its In Oracle time waiting for a RAC activity to complete on the object specified in the Object column.

Table 55 RAC Wait

	Description
What to do next	Select the findings type to see the instances consuming object RAC Wait in the Activity tab.
Advice	The object is suffering from a RAC Wait because several instances are using it simultaneously. To solve the problem, identify all programs currently accessing the object and try to avoid accessing it concurrently.

Bind Variables Were Collected

Bind sets were collected for the statement.

Table 56 Bind Variables Were Collected

	Description
What to do next	Perform one of the following options: • Select the findings type to examine the bind values in the Bind Variables tab. • Use the Estimate Cost button to check whether the Oracle optimizer identified a better execution plan for a provided set of binds.
Advice	If the Oracle optimizer finds a better execution plan, try to evaluate one plan in relation to another for a set of captured binds and try to stabilize it by using outlines.

More Than One Real Plan Was Detected

More than one real plan was collected for the statement during the selected time frame.

Table 57 More Than One Real Plan Was Detected

	Description
What to do next	Select the findings type and use the All Plans tab to examine the different captured execution plans and their real resource consumption.
Advice	Try to identify the source for the different execution plans. To resolve the multiple plans, consider the following solutions: • Go to the History tab to see whether there were any changes (such as schema or statistics changes), or • Go to the Activity tab to see whether it is being run by different programs.

Costs Have Changed Over the Last Month

The cost for the statement has changed over the last month.

Table 58 Costs Have Changed Over the Last Month

	Description
What to do next	Select the findings type and use the History tab to examine the different costs caught for the statement.
Advice	Try to identify the source for the different execution plan costs. Go to the History tab to check whether there were any changes (such as schema or statistics changes).

Frequently Executed Statement

Much of the statements In Oracle time is spent on CPU usage, even though the average In Oracle time is low (indicating executions run many times).

Table 59 Frequently Executed Statement

	Description
What to do next	Perform one of the following options: • Launch to the Activity tab to examine the scalability of the statement. • Examine the activity of the program enabling the statement.
Advice	Try to determine if the large number of executions is valid or if it is derived from redundant executions as a result of inefficient program design (e.g. if the statement is enabled in an inefficient loop).

The Average Execution Uses CPU Heavily

Much of the stats In Oracle time is spent on CPU usage, and the average In Oracle time is high.

Table 60 The Average Execution Uses CPU Heavily

	Description
What to do next	Run the statement with statistics_level=ALL.
Advice	Perform one of the following: • Examine run results in the Extended Statistics tab. Analyze problematic data (such as many logical reads, table scans and more). • Explain the statement and use the SQL tab tools to improve performance.

Heavy Index Overhead

Most I/O wait on indexes is due to fetching index pages from the disk, reflecting changes made by the DML statement. The indexes do not appear in the execution plan.

Table 61 Heavy Index Overhead

	Description
What to do next	Perform one of the following options: • In the Objects tab, observe the I/O breakdown between the different indexes. Focus on the heaviest index and launch to the Tune Object tab. • Check if this index is being used in other access types other than Index Overhead. If the only access type is Index Overhead, this may indicate that the index is not being used to filter table rows. • In the Statements tab, examine the different statements using the index, to learn about index usage patterns. • In the Read/Write Operations tab, in the In Oracle graph, try to identify index update patterns (such as, day or night).
Advice	Perform one of the following options: • When inserts are part of a load, batch, or night activity, consider dropping the index before performing the activity, and recreating it afterwards. • If the index is not used in execution plans, consider dropping the index or unused columns from the index to reduce index overhead. If the index is used in execution plans, launch to the What-If tab to see which statements may be effected by this change. ⓘ The Statements tab shows the activity and execution plans that were detected during the selected time frame and may not reflect the activity of all the statements and execution plans executed during this time frame. Proceed with caution when determining whether to drop an index or delete a column from an index.
Example	**Table**: INSERTED_TABLE (C1 number,C2 date, C3 varchar2(128), C4 number) Indexes on table: IX1 (C1,C2) IX2(C4,C3) **Statement**: Insert into INSERTED_TABLE values (:h1,:h2,:h3,:h4) In this example, Oracle fetches the relevant index blocks of the two indexes, for the new rows, even though the indexes do not appear in the execution plan. The I/O wait accumulated while fetching these index blocks is considered to be an index update.

Preferable Plan Detected by Oracle

This finding can only appear after running the "Get Best Plan" command in the Bind Variables tab.

It indicates that some bind values may lead Oracle to choose a different execution plan than others. This does not mean that when those bind values are used for the statement that their relevant plan will be used. This depends on which version of Oracle is being used, whether the statement's plan exists in memory, and whether the init.ora parameter "_optim_peek_user_binds" is set to TRUE or FALSE.

Table 62 Preferable Plan Detected by Oracle

	Description
What to do next	Perform one of the following options: • Check the PLAN_HASH_VALUE field that appears in the left pane in the Bind Variables tab and examine the difference between plans of different bind sets. • Consider running the statement using a different bind set leading to a different plan, and compare their run time.
Advice	If different execution plans result in a fluctuation in run time consider the following: • Fixing an optimal plan using outlines or profiles. • Disabling bind peeking by setting the "_optim_peek_user_binds" parameter to FALSE • Consider using literals instead of bind variables. Setting "_optim_peek_user_binds" to FALSE will influence all statements running in the selected instance.

Low End of Fetch Count

The query activator doesn't fully read the query result set.

Consider speeding up the first set of rows fetch time. This is usually done by using the "FIRST_ROWS" hint.

Verify that the application is not performing a huge data scan for no reason. Unnecessarily large scans are expensive (in terms of CPU and I/O time).

The end of fetch count is identified by the number of times the specified cursor was fully executed since the cursor was brought into the library cache. Its value is not incremented when the cursor is partially executed, either because it failed during execution or because only the first few rows produced by this cursor were fetched before the cursor was closed or re-executed.

Table 63 Low End of Fetch Count

	Description
What to do next	• Examine the statement's execution plan and verify that unnecessary scans are not being performed. • Observe which program is calling the statement in the Activity Tab, and check your Application to verify that there is a true need to scan all data and fetch only part of it.
Advice	• Consider tuning the statement so that it scans fewer rows, by adding a "FIRST_ROWS" hint. Compare the result execution plan to the original, and run both statements to evaluate the influence of the change. • Consider changing the application's logic to improve filtering. This will lead to smaller scans.

Major Difference Between Plans

This finding may appear when the difference between the best and worse plans, within a selected time frame, is significant in terms of in Oracle time. This indicates that some plans were used, and the best plan consumed significantly less resources of Oracle than the worse plan.

Table 64 Significant Differentiation between Best and Worse Plans

	Description
What to do next	Perform one of the following options: • Compare the execution plan of the best plan with the execution plan of the worse plan (also includes a comparison of costs). In the Bind variables tab, check if the difference can be attributed to using a different bind set. If not, check if the significant difference can be derived from a change of one of the objects. • If you find that the difference between the best and worse plans is derived from a different bind set, consider running the statement using a different bind set and compare their run times.

Advice	If the significant difference between the best and worse plans was derived from using a different bind set consider: • Fixing an optimal plan using outlines or profiles. • Disabling bind peeking by setting the "_optim_peek_user_binds" parameter to FALSE. • Using literals instead of bind variables. If the significant difference between the best and worse plans were derived from a change of one of the objects within the plan: • Explore the objects that are being accessed inefficiently in the worst plan, in the Objects tab. Check if the table grew significantly or if one of the indexes was dropped and can be rebuilt. • Consider using outlines or profiles for checking improvements to the worst plan. Notice that setting the "_optim_peek_user_binds" parameter to FALSE will influence all statements running in this instance.

Newer Execution Plan Exists

A newer execution plan than the one you are viewing was collected for the statement during the selected time frame, or during a later time frame.

Table 65 Newer Execution Plan Exists

	Description
What to do next	Perform one of the following options: • Expand the time frame you are viewing, using the time frame list box, to identify the new execution plan. Expand the time frame until it includes the current time so that the newer execution plan is loaded. • Examine the different execution plans that were captured, and their real resource consumption.
Advice	Try to identify the source of the different execution plans. To resolve the multiple plans, consider the following solutions: • In the All Changes section view, observe whether any changes took place (such as schema or statistics changes). • In the Activity tab, observe whether the execution plan is being run by different programs.

Cartesian Join Used

The use of a merge join Cartesian is very expensive for Oracle. Cartesian joins can be caused by a missing Table Join condition to the WHERE clause.

A Cartesian Join is used when one or more of the tables does not have any join conditions to any other tables in the statement. The optimizer joins every row from one data source with every row from the other data source, creating the Cartesian product of the two sets.

Table 66 Cartesian Join Used

	Description
What to do next	Examine the predicates in the statement's WHERE clause.
Advice	To avoid a Cartesian Join verify that you have provided the proper join conditions in the statement's WHERE clause.

CPU Used for Sorts

I/O found on temporary tablespace may indicate sort operations are consuming CPU time of the statement.

Table 67 CPU Used for Sorts

	Description
What to do next	In the Activity tab, examine temporary tablespace overtime I/O consumption for the statement and for the programs activating the statement.

Advice	To reduce the I/O consumption for the sort operation, consider the following solutions: • Change the SORT_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid sort and hash area size problems. You can either change the values for a specific session using the `Alter Session` command, or for the entire instance. • Try to solve the problem at the statement level by one of the following options: Add an index to prevent a sort. If your statement has an Order by clause that has columns for a single table, check whether you can add an index. In some cases, you may get an index recommendation that prevents a sort. If you are considering adding an index, you must check the effect of that index. To analyze that effect, launch to the What-If tab. Identify the heavy sort or hash consumer step. Run a statement with statistics_level=all. Click the Run & Compare tab. Examine the LAST_TEMPSEG_SIZE and MAX_TEMPSEG_SIZE in the extended

CPU Used for Index Scattered Read

Statement 's I/O may indicate Index Scattered read operation (often full scan) on the index specified in the Object column.

Table 68 CPU Used for Index Scattered Read

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the index consumption for the statement in the Activity tab.
Advice	To reduce the CPU consumption, consider the following solutions: • If the index is not used to prevent a sort operation, consider adding the INDEX_FFS hint and change DB_FILE_MULTIBLOCK_READ_COUNT parameter to a higher value. • Switch to another index or add a new index. • Use a parallel query option. • Move the index to another tablespace with a higher block size. Findings refer to the whole statement - not to a specific execution plan. If a step does not exist in the selected execution plan (unless this is due to an index overhead), switch to another plan and locate the relevant step.

CPU Used for Index Sequential Read

Statement 's I/O may indicate Index Sequential read operation (often range scan) on the index specified in the Object column. If the statement is DML and the index is not used in the execution plans, then the I/O represents the index maintenance overhead, caused by fetching the index blocks for update to memory.

Table 69 CPU Used for Index Sequential Read

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to see if the index is used in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the index overtime consumption for the statement in the Activity tab.
Advice	If this is the result of a range scan, consider one of the following solutions: • Switch to another index, add a new index, or change the index structure. • Add columns to the index to enable index only access. • Switch to a full table scan. • Partition the table or the index. • Create a cluster with just one table in it and a cluster key used as the index key. If this is due to an index overhead, consider reducing the index overhead by deleting unused indices, or unused columns in used indices. Findings refer to the whole statement - not to a specific execution plan. If a step does not exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

CPU Used for Table Scattered Read

Statement 's I/O may indicate table scattered read operation (often full scan) on the table specified in the Object column.

Table 70 CPU Used for Table Scattered Read

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the table consumption for the statement in the Activity tab.
Advice	To reduce the CPU consumption for the table, consider the following solutions: • Create an index matching the statement's predicates. • Change the DB_FILE_MULTIBLOCK_READ_COUNT parameter in INIT.ORA to a higher value. • Partition the table according to the best predicates existing in the statement. • Use a parallel query option. • Move the table to another tablespace with a higher block size. Findings refer to the whole statement - not to a specific execution plan. If a step does not exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

CPU Used for Table Sequential Read

Statement 's I/O may indicate a Table Sequential read operation (often ROWID access) on the table specified in the Object column.

Table 71 CPU Used for Table Sequential Read

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to find the relevant step in the execution plan. • Select the findings type to investigate the objects used and their structure in the Objects tab. • Select the Activity tab, locate the statement associated with objects, and drill to the table consumption for the statement in the Activity tab.
Advice	To reduce the CPU consumption for the table, consider the following solutions: • Change the index structure by adding a better filtering column, or change the column sequence to improve the matching and screening performed in the index tree. • Add columns to the index to enable index only access. • Switch to a full table scan. • Partition the table or the index. • Create a cluster with just one table in it and a cluster key used as the index key. Findings refer to the whole statement - not to a specific execution plan. If a step does not exist in the selected execution plan (unless this is the result of an index overhead), switch to another plan and locate the relevant step.

Heavy Table Full Scan

Statement resources are spent performing Full table scans on the table specified in the Object column.

Table 72 Heavy Table Full Scan

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Get Index Recommendations for the statement. • Select the finding type to investigate the objects used and their structure in the Objects tab. • Select the finding type to investigate the step statistics, step resource consumption and step vs. plan over time.

Advice	To reduce the Full Scan resource consumption consider: • Creating an index matching the statement's predicates. • If full scan is using I/O heavily consider changing the DB_FILE_MULTIBLOCK_READ_COUNT init.ora parameter to a higher value. • Partition the table according to the best predicates existing in the statement. • Use a parallel query option. • Move the table to another tablespace with a higher block size.

Heavy Step

Statement resources are spent performing the specified step.

Table 73 Heavy step

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.

Heavy Sort

Statement resources are spent performing sort operation.

Table 74 Heavy Sort

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	Try to solve the problem at the statement level by one of the following options: • Add an index to prevent a sort. • If your statement has an Order by clause that has columns for a single table, check whether you can add an index. In some cases, you may get an index recommendation that prevents a sort. If you are considering adding an index, you must check the effect of that index. To analyze that effect, launch to the What-If tab. • If the sort involves I/O on the temporary tablespace, you can reduce the I/O consumption for the sort operation by changing the SORT_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid sort area size problems. You can either change the values for a specific session using the `Alter Session` command, or for the entire instance.

Heavy Hash

Statement resources are spent performing hash operation.

Table 75 Heavy Hash

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	Try to solve the problem at the statement level by one of the following options: • If the hash operation involves I/O against temporary tablespace, you can reduce the I/O consumption for the hash operation by changing the HASH_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid hash area size problems. You can either change the values for a specific session using the `Alter Session` command, or for the entire instance.

Heavy Merge

Statement resources are spent performing merge operation.

Table 76 Heavy Merge

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	Try to solve the problem at the statement level by one of the following options: • If the hash operation involves I/O against temporary tablespace, you can reduce the I/O consumption for the hash operation by changing the SORT_AREA_SIZE to a higher value. If you are using PGA settings, you can change the PGA_Aggregated_Target, so as to avoid sort area size problems. You can either change the values for a specific session using the `Alter Session` command, or for the entire instance.

Heavy Index Full Scan

Statement resources are spent performing Full index scans on the index specified in the Object column.

Table 77 Heavy Index Full Scan

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate the objects used and their structure in the Objects tab. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	To reduce the Full Scan resource consumption consider: • Creating an index matching the statement's predicates. • If the full scan is using I/O heavily and the step name is a fast full scan, consider changing the DB_FILE_MULTIBLOCK_READ_COUNT init.ora parameter to a higher value. • Partition the table according to the best predicates existing in the statement. • Use a parallel query option. • Move the table to another tablespace with a higher block size.

Heavy Index Range Scan

Statement resources are spent performing Range index scans on the index specified in the Object column.

Table 78 Heavy Index Range Scan

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate the objects used and their structure in the Objects tab. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	To reduce the Range Scan resource consumption consider: • Checking the matching level of the Index to discover if there is a mismatch between existing Where predicates and operators with column order in the Index. • Changing column order in the Index for better matching level or adding a new index. Check the global effect of this change by launching to WhatIF tab.

Heavy Index Skip Scan

Statement resources are spent performing Range index skip scans on the index specified in the Object column.

Table 79 Heavy Index Skip Scan

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate the objects used and their structure in the Objects tab. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	To reduce the Range Skip Scan resource consumption consider: • Changing column order in the Index for better matching level or adding a new index. Check the global effect of this change by launching to WhatIF tab.

Heavy Cartesian Join

Statement resources are spent performing Cartesian join.

Table 80 Heavy Cartesian Join

	Description
What to do next	Perform one of the following options: • Click the **Locate** icon to focus on the relevant step in the execution plan. • Select the finding type to investigate step statistics and resource consumption in the Statistics tab.
Advice	To avoid a Cartesian Join verify that you have provided the proper Join conditions in the statement's WHERE clause.

About Object Findings

Several Object tab findings exist to help the user. The Objects tab includes the following findings:

- Heavy Index Overhead
- Extensive Full Table Scan Access
- Full Scan Reading Deleted Blocks
- Index Clustering Factor Very High
- Buffer Wait Contention
- Object or Row Lock Contention
- Bottleneck in RAC Wait
- Many Chained Rows
- Statistics Not Updated on Object
- Changes Detected in Object Structure
- Table Grew Considerable
- Partition Is Accessed Extensively
- Segment Hit Ratio Very Low
- Extensive Activity on Non-explained Statements
- Extensive Index Range Scan Access
- Extensive Full Index Scan Access
- Extensive Fast Full Index Scan Access
- Extensive Index Skip Scan Access

Heavy Index Overhead

Most of the I/O wait on indexes is due to the fetching of index pages from disk that reflect changes made by INSERT, DELETE, UPDATE, or MERGE statements. The index does not appear in the execution plan.

Table 81 Heavy Index Overhead

	Description
What to do next	Perform one of the following options: • Examine the DML statements causing index updates in the Statements tab (see example below). Focus on Index Update access types in the Access Types table. • Check if the index is being used in execution plans, in the Access Types table, in the Statements tab. If the only access type is Index Update, this may indicate that the index is not being used. • Try to identify index update patterns (such as, day or night) in the In Oracle graph, in the Read/Write Operations tab.

Advice	Perform one of the following options: • When inserts are part of load, batch, or night activity, consider dropping the index before performing the activity, and recreating it afterwards. • If the index is not used in execution plans consider dropping the index or unused columns from the index to reduce index overhead. If the index is used in execution plans, launch to the What-If tab to see which statements may be effected by this change. The Statements tab shows the activity and execution plans that were detected during the selected time frame and may not reflect the activity of all the statements and execution plans executed during this time frame. Proceed with caution when determining whether to drop an index or delete a column from an index.
Example	**Table**: INSERTED_TABLE (C1 number,C2 date, C3 varchar2(128), C4 number) **Indexes on table**: IX1 (C1,C2) IX2(C4,C3) **Statement**: Insert into INSERTED_TABLE values (:h1,:h2,:h3,:h4) In this example, Oracle fetches the relevant index blocks of the two indexes, for the new rows, even though the indexes do not appear in the execution plan. The I/O wait accumulated while fetching these index blocks is considered to be an index update.

Extensive Full Table Scan Access

Table extensively accessed using full table scans.

Table 82 Extensive Full Table Scan Access

	Description
What to do next	Perform the following options: • Examine associated statements in the Statements tab. Focus on Full Table Scan access types in the Access Types table. • Examine column usage for each statement in the Columns table. • Get Index Recommendations for the object's statements.

Advice	Perform one of the following options:
	• Eliminate Full Table Scan by:
	  ◦ Trying to identify common high selectivity columns for the top statements. Create an index matching the statements predicates.
	  ◦ Partitioning the table according to the best predicates existing in the statements.
	• Improve Full Table Scan by:
	  ◦ Moving the table to another tablespace with a higher block size.
	  ◦ Increasing the value of the DB_FILE_MULTIBLOCK_READ_COUNT init.ora parameter.

Full Scan Reading Deleted Blocks

Table containing many deleted blocks is extensively accessed using full table scans.

Table 83 Full Scan Reading Deleted Blocks

	Description
What to do next	Perform the following options: • Examine associated statements in the Statements tab. Focus on Full Table Scan access types in the Access Types table. • Examine column usage for each statement in the Columns table.

Advice	Perform one or more of the following options:
	• Eliminate full table scan by:
	  ◦ Creating an index matching the statements predicates.
	  ◦ Partitioning the table according to the best predicates existing in the statements.
	• Treat deleted blocks:
	  ◦ Consider exporting the table data, truncating the table, and then reloading the data.
	  ◦ If the dirty blocks problem is repeated, the reorganization solution may be less relevant and the user can use an index using range scan or even full index scan to reduce access to the table.
	• Improve full table scan by:
	  ◦ Moving the table to another tablespace with a higher block size.
	  ◦ Increasing the value of the DB_FILE_MULTIBLOCK_READ_COUNT init.ora parameter.

Index Clustering Factor Very High

Intensive I/O wait activity on table due to a range scan carried out by an index with a bad clustering factor (mismatch between physical order of rows in table and order of ROWIDs from the index range scan leads to re-reading of table blocks).

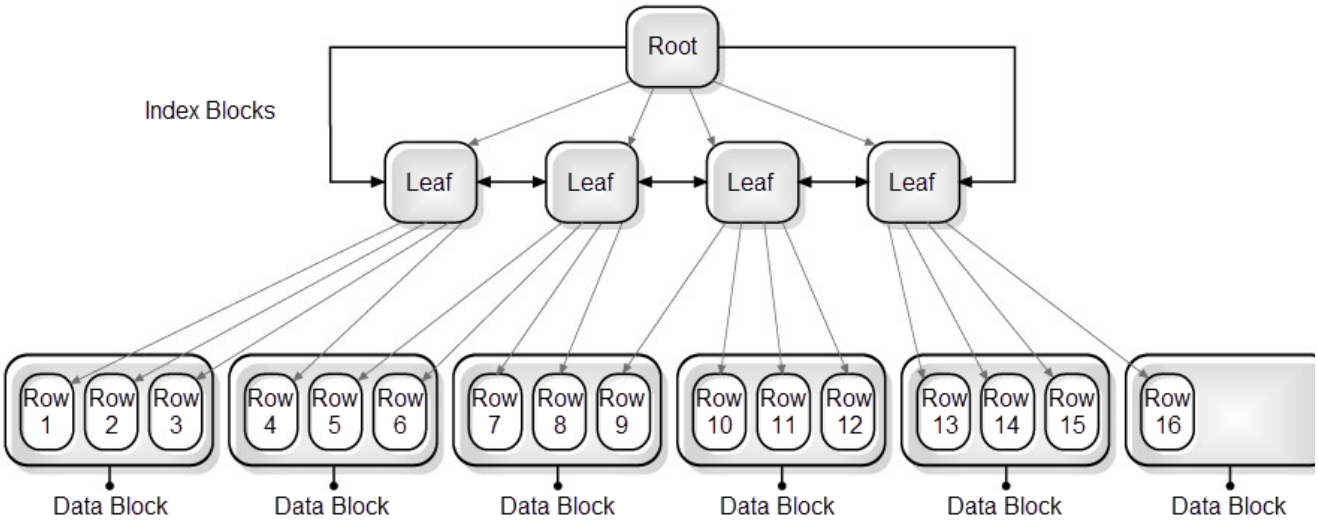
Table 84 Index Clustering Factor Very High

	Description
What to do next	Perform the following options: • Examine associated statements in the Statements tab. Focus on Index Range Scan access types in the Access Types table. • Examine column usage for each statement in the Columns table.
Advice	Perform one of the following options: • Enhance filtering of the table data blocks by adding columns to the index or ensure that index only accesses top statements. • If no primary key exists and the index is the most used or essential index for the table, consider sorting the table data according to the index key.

The following example shows the effect that a bad clustered index can have on performance when an index is scanned:

The figure below shows an example of an index with a good clustering factor. In this example, the root is read first, followed by the first leaf page. Then the first data block that serves the first three keys matching the three rows in the data block is fetched. In this way the keys and data blocks that follow are read. The I/O operations required by this scan include five index blocks and six data blocks, which is the equivalence of 11 I/O operations.

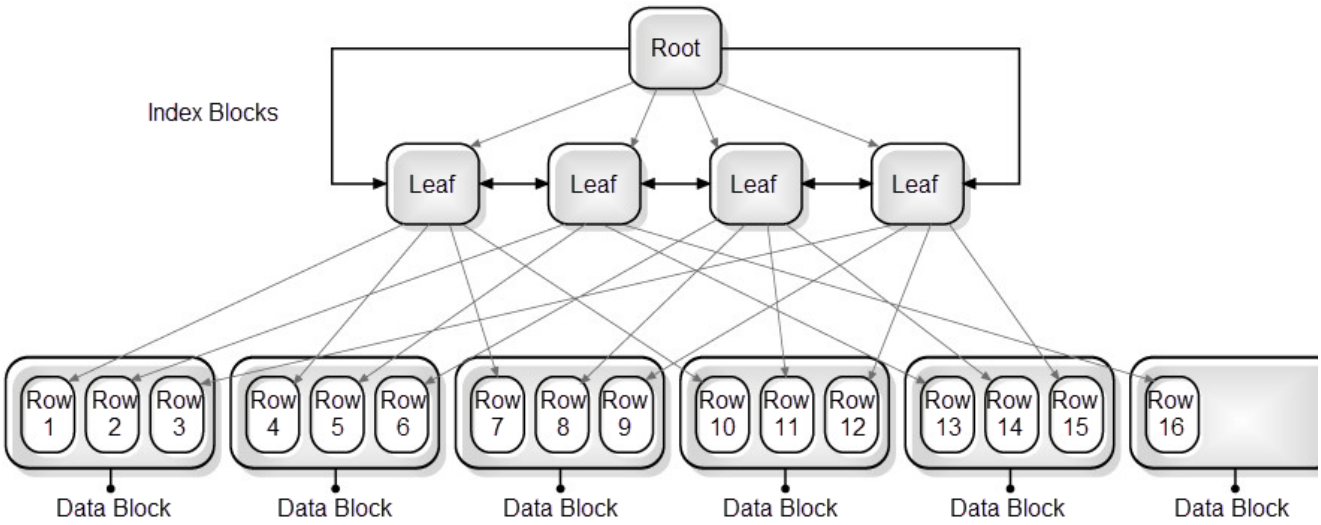
Figure 1 Index with good clustering factor (low = number of table blocks)



The following figure shows an example of an index with a bad clustering factor.

The index with the bad clustering factor starts in the same way as the index with the good clustering factor. However, when the second key in the index is read, the row for the second key in the first data block has not yet been fetched, so another block must be fetched. By the time Oracle accesses the index key matching the second row in the first table block, it has already been swapped out of memory and needs to be re-read. In the worse case scenario, I/O for the table blocks will be required for every index key. The I/O operations required by this scan include five index blocks and 16 table blocks, which is equivalence of 21 I/O operations. When the difference between the number of blocks and number of rows is great, performance can be greatly impacted.

Figure 2 Index with bad clustering factor (high = number of rows)



Buffer Wait Contention

Object (table and indexes) spent much of its In Oracle time on Buffer wait. This usually occurs as a result of one of the following:

- Contention on a table or index buffer in Insert statements (Buffer Busy wait).
- Lack of free buffer space when trying to load blocks from a disk (Free Buffer wait).

Table 85 Buffer Wait Contention

	Description
What to do next	Perform the following options: • Examine buffer wait over time, in the Activity Tab. • Examine buffer wait substate events in the Statistics tab.

Advice	Perform one of the following options:
	• If Buffer Busy wait is the more dominant Oracle event, consider doing the following:
	  ◦ Increase the free lists for the table to reduce the chances of contention.3
	  ◦ Increase the PCTFREE parameter or decrease the block size for the table, to distribute data among many blocks and reduce the chances for contention.
	• If Free Buffer wait is the more dominant Oracle event, consider doing the following:
	  ◦ Tune access to the object, to reduce the number of blocks fetched.
	  ◦ If the problem is a global instance problem, increase the number of DBWR processes or I/O slaves.

Object or Row Lock Contention

Much of the objects (table and indexes) I/O time is spent waiting for a lock on the object specified in the Object column.

Table 86 Object Or Row Lock Contention

	Description
What to do next	Examine the statement in the Activity tab.
Advice	To reduce the lock wait for the object, perform one of the following: • Check to see if the lock appears in the Current tab. If so, examine the lock chain to identify which statement is holding the lock. • Try to identify the locking statement in the Activity tab using smaller time frames that match the lock periods. Focus on the locked table and associated statements. The DML statements (and update queries) that are NOT waiting for locks should be the immediate suspects.

Bottleneck in RAC Wait

The object (table and indexes) spent much of its In Oracle time waiting for a RAC activity to complete on the object specified in the Object column.

Table 87 Bottleneck in RAC Wait

	Description
What to do next	Examine the statement in the Activity tab.
Advice	The object is suffering RAC wait because several instances are using it simultaneously. To solve this problem, identify all programs currently accessing the object and try to avoid accessing it concurrently.

Many Chained Rows

Access to table deteriorated as a result of chained rows.

Table 88 Many Chained Rows

	Description
What to do next	Examine table dictionary information.
Advice	The object is suffering because it is accessing an object that suffers from chained rows. Chained rows are typically caused by the Insert operation. To solve the problem, perform one of the following: • Increase the PCTFREE parameter (can be by alter or move table). • Reorganize table (export/import or manually reorganize table). • Move table to tablespace with higher block size.

Statistics Not Updated on Object

A significant block change occurred since the last time the object was analyzed, for at least one of the objects related to table.

Table 89 Statistics Not Updated on Object

	Description
What to do next	Examine the dictionary details of the object in the Details section. Look at the Read/Write Operation Tab to quantify the magnitude of the block change.
Advice	To reduce potential access type problems resulting from statistics that are not up-to-date, consider analyzing the table and checking it periodically.

Changes Detected in Object Structure

Changes were made to the table or index structure. Possible changes include:

- Index was added or dropped.
- Partitions or subpartitions were added or dropped.
- Table was altered (columns were added).

Table 90 Changes Detected in Object Structure

	Description
What to do next	Examine the changes in the Changes graph in the Read/Write Operations tab.
Advice	Perform one of the following options: • Try to determine whether the changes in object structure and changes in performance are related by comparing the In Oracle graph and the Changes graph in the Read/Write Operations tab. If it seems that performance deteriorated as a result of the change in table or index structure, consider rolling back the change. • Consider adjusting index structure and execution plans.

Table Grew Considerably

The table is considerably larger than it was at the start of the time frame.

Table 91 Table Grew Considerably

	Description
What to do next	Examine if there is a correlation between table growth and performance degradation in the Read/Write Operations tab.
Advice	Verify that full scans are not widely used for the table, that the existing indexes correlate with the table growth, and that no new indexes are required. You can also check Materialized Views usage.

Partition Is Accessed Extensively

A large percentage of In Oracle time for the object is spent accessing one partition.

Table 92 Partition Is Accessed Extensively

	Description
What to do next	Perform the following options: • Examine the In Oracle activity of the partition, in the Partitions tab. Check if the massive activity spent accessing the partition is abnormal. • Examine the activity of statements accessing the partition in the Activity tab.

Advice	Perform one of the following options:
	• If the partition is a table partition:
	  ◦ Create local or global indexes for statements that access the partition.
	  ◦ Subpartition the partitions.
	  ◦ If the partitions are not balanced well, consider building the partitioned table with new partition keys.
	• If the partition is an index partition:
	  ◦ Subpartition the partitions.
	  ◦ Add more columns to the index to improve filtering.

Segment Hit Ratio Very Low

The hit ratio, for at least one of the objects related to the table, is very low.

Table 93 Segment Hit Ratio Very Low

	Description
What to do next	Perform the following options: • Examine associated statements in the Statements tab. Focus on the All access type in the Access Types table. • Examine buffer cache usage in the Statistics tab. Check if there is an overall wait on the Free Buffer event.
Advice	If there are no outstanding contentions on the buffer cache consider moving the object into Keep or Recycle buffer cache pools.

Extensive Activity on Non-explained Statements

Extensive activity on statements that were not explained.

Table 94 Extensive Activity on Non-explained Statements

	Description
What to do next	Examine associated statements in the Statements tab. Focus on non-explained statements in the Access Types table.
Advice	Perform an explain on the non-explained statements.

Extensive Index Range Scan Access

Extensive I/O wait was experienced, as a result of range scans on the index. Although this may be normal, it can often indicate a matching level problem, indicating that the structure of the index can be improved.

Table 95 Extensive "Index Range Scan" Access

	Description
What to do next	Examine statements using the index in the Statements tab. Examine column usage, selectivity and matching level (see example) for the top statements, in the Columns table to assess the efficiency of the index.
Advice	If the index structure does not fit the Where predicates of the top statements consider doing one of the following: • Add columns to the index in the right sequence (columns with high selectivity and equal predicates should be first) to improve the matching level. This leads to a better filtering of leaf pages. • Change the sequence of columns in the index to the optimal sequence (columns with high selectivity and equal predicates should be first) to improve the matching level. This leads to a better filtering of leaf pages.
Example	**Table**: TAB1 (C1 number, C2 number, C3 number, C10 Date) **Index**: IX1 (C1,C2,C5) **Statement**: select * from TAB1 where C1=:h and C5=10; Execution plan uses IX1 in Index Range Scan In this statement the matching level of the index is 1. This means that Oracle uses only C1 to filter index leaf pages, it cant match C5=10 against the index tree because of the absence of a C2= predicate. Because C1 is not selective, many irrelevant index leaf pages can be read. Oracle will apply the C5=10 predicate on the index keys to screen irrelevant table ROWIDs. An index on C1 followed by C5 would be more efficient for the query.

Extensive Full Index Scan Access

Index is extensively accessed using full index scans. This is sometimes done to avoid sorts, when the sort order matches the leading portion of the index key, or to avoid accessing table blocks, when all the columns required by the query exist in the index key.

Table 96 Extensive “Full Index Scan” Access

	Description
What to do next	Perform the following: • Examine associated statements in the Statements tab. • Focus on the Full Index Scan access type in the Access Types table. • Examine column usage, for each statement in the Columns table.

Advice	Perform one of the following options:
	• Full index scan, can be eliminated by:
	  ◦ Identifying common high selectivity columns for the top statements. Create an index matching the statements predicates.
	  ◦ Partitioning the table according to the best predicates existing in the statements.
	• Full index scan, can be improved by switching to Fast Full Index Scan when the index is not used to save sorts (a fast full scan retrieves the rows according to the index key). Perform one of the following options to enable this access path:
	  ◦ Ensure that the Fast_Full_Scan_Enabled parameter = yes.
	  ◦ Use the "Index_ffs" hint for major statements that use the full index scan, to check if the fast full scan improves performance. When the full index scan doesn't save sort results, performance may improve considerably.

Example	**Table**: TAB1 (C1 number, C2 number, C3 number, C10 Date) **Index**: IX1 (C1) **Statement**: select C1 from TAB1 Order by C1; Execution plan uses IX1 in Full Index Scan In this case a full index scan is the best option. Because there are no filtering predicates, there is no need to access the table blocks and the sort operation is avoided.

Extensive Fast Full Index Scan Access

Index is extensively accessed using fast full index scans.

Table 97 Extensive "Fast Full Index Scan" Access

	Description

What to do next	Perform the following: • Examine associated statements in the Statements tab. • Focus on the Fast Full Index Scan access type in the Access Types table. • Examine column usage for each statement in the Columns table.

Advice	Perform one of the following options:
	• Fast Full index scan can be eliminated by:
	  ○ Identifying common high selectivity columns for the top statements. Create an index matching the statements predicates.
	  ○ Partitioning the table according to the best predicates existing in the statements.
	• Fast Full Index scan can be improved by:
	  ○ Moving the index to another space with a higher block size.
	  ○ Increasing the DB_FILE_MULTIBLOCK_READ_COUNT init.ora parameter.

Extensive Index Skip Scan Access

Index extensively accessed using index skip scans which often means that the index structure does not fit the query in the best possible way, and leads Oracle to perform heavy activity against the index

Table 98 Extensive "Index Skip Scan" Access

	Description
What to do next	Perform the following: • Examine associated statements in the Statements tab. • Focus on the Index Skip Scan access type in the Access Types table. • Examine column usage for each statement in the Columns table.
Advice	Perform one of the following options: • Change the columns sequence within the index that use the index skip scan. Check the effect of this change in the What-If tab. • Identify common high selectivity columns for top statements. Create an index matching the statements predicates. • If you cannot create a new index, use hints, such as, "full" and "index_ffs" to determine if by using these access types you can achieve better performance results.
Example	**Table**: TAB1 (C1 number, C2 number, C3 number, C10 Date) **Index**: IX1 (C1,C2) (C1 has two distinct values Yes and No) **Statement**: select * from TAB1 where C2=10; Execution plan uses Index Skip Scan on IX1. In this case Oracle has to perform two range scans on the index—one with a key of (Yes,10) and another with a key of (No,10)and then unite the results. The more distinct values defined for C1, the more index scanning required. Defining a new index on (C2), or changing the column sequence in IX1 to be (C2,C1), enables a more efficient access path for the Index Range Scan.