**SQL Tunning**

**ARQUITECTURA**

* **Razones para el rendimiento ineficiente de SQL**
* Estadísticas de optimizador obsoletas o faltantes.
* Estructuras de acceso faltantes.
* Selección de plan de ejecución subóptima.
* SQL mal construido.

Las declaraciones SQL pueden funcionar mal por una variedad de razones:

• Estadísticas del optimizador obsoleto: el optimizador basado en costos **(CBO)** genera los planes de ejecución de SQL. Para que CBO elija efectivamente el plan más eficiente, necesita información precisa sobre el volumen de datos y la distribución de tablas e índices a los que se hace referencia en las consultas. Sin estadísticas precisas del optimizador, la CBO puede fácilmente confundirse y generar planes de ejecución subóptimos.

• Estructuras de acceso faltantes: la ausencia de estructuras de acceso, como índices, vistas materializadas y particiones, es una razón común para el bajo rendimiento de SQL. El conjunto correcto de estructuras de acceso puede mejorar el rendimiento de SQL en varios órdenes de magnitud.

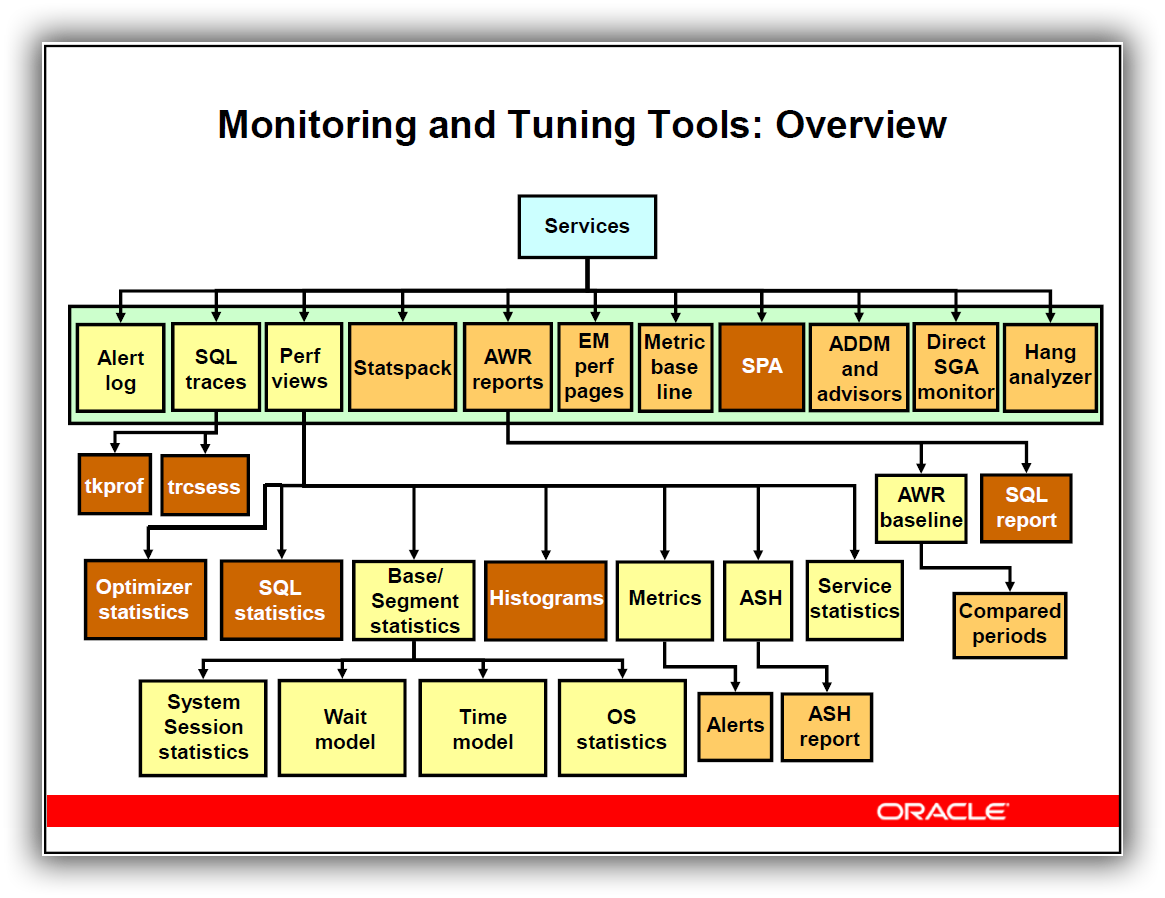
• Selección del plan de ejecución subóptimo: la CBO a veces puede seleccionar un plan de ejecución subóptimo para una instrucción SQL. Esto sucede en su mayor parte debido a estimaciones incorrectas de algunos atributos de esa declaración SQL, como su costo, cardinalidad o selectividad de predicado.

• SQL mal construido: si la instrucción SQL está mal diseñada, no hay mucho que el optimizador pueda hacer para mejorar su rendimiento. Falta una condición de unión que conduzca a un producto cartesiano, o el uso de construcciones SQL más costosas como UNION en lugar de UNION ALL, son solo un par de ejemplos de diseño SQL ineficiente

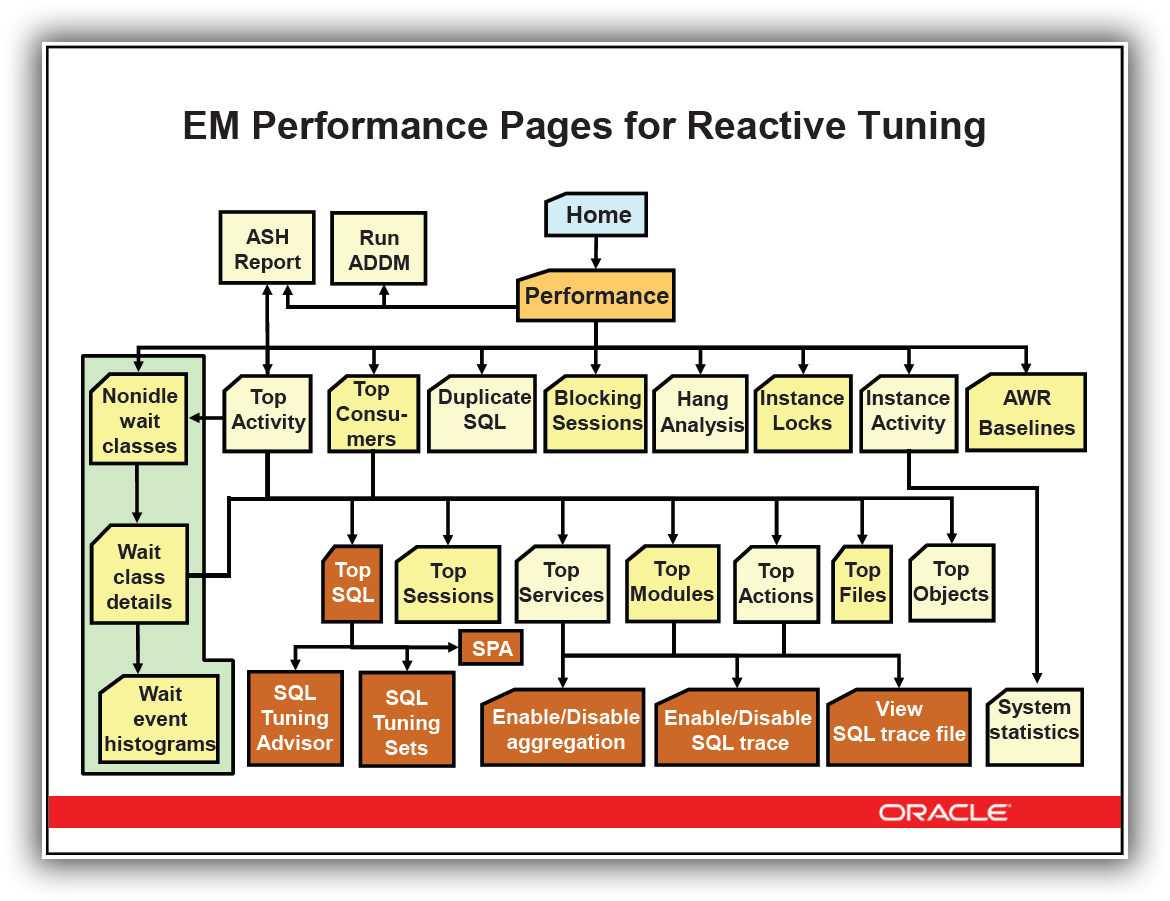
Estas cuatro causas principales de una pobre optimización de SQL pueden tener un impacto drástico en el rendimiento.

**Nota:** motivos adicionales para un bajo rendimiento pueden estar relacionados con problemas relacionados con el hardware, como memoria, E / S, CPU, etc.

* **Herramientas de monitoreo y ajuste: Overview**



* **Páginas de rendimiento de EM para ajuste reactivo**

****

* **Herramientas de ajuste: Descripción general**
* Automatic Database Diagnostic Monitor (ADDM)
* SQL Tuning Advisor
* SQL Tuning Sets
* SQL Access Advisor
* SQL Performance Analyzer
* SQL Monitoring
* SQL Plan Management

**Monitor de diagnóstico automático de la base de datos (ADDM):**

Analiza continuamente los datos de rendimiento que se recopilan de la instancia de la base de datos.

**SQL Tuning Advisor:**

Analiza las declaraciones SQL que se han identificado como problemáticas, en un esfuerzo por volver a ajustarlas. Por defecto, esta es una tarea automatizada. También puede, en cualquier momento, ejecutar el Asesor de ajuste SQL en una carga de trabajo SQL específica para buscar formas de mejorar el rendimiento.

**SQL Tuning Sets:**

Sirven como repositorio para conjuntos de sentencias SQL. Por ejemplo, el Asesor de ajuste de SQL puede ejecutarse contra una carga de trabajo representada por un Conjunto de ajuste de SQL. Incluso se pueden transportar de una base de datos a otra para realizar análisis en diferentes máquinas.

**SQL Access Advisor:**

Analiza una instrucción SQL y proporciona consejos sobre vistas materializadas, índices, registros de vistas materializadas y particiones

**SQL Performance Analyzer:**

Automatiza el proceso de evaluar el efecto general de un cambio, como actualizar una base de datos o agregar nuevos índices, en la carga de trabajo completa de SQL mediante la identificación de divergencia de rendimiento para cada declaración.

**SQL Monitoring:**

Le permite supervisar el rendimiento de las declaraciones de SQL mientras ejecutan

SQL Plan Management (SPM):

Se puede utilizar para controlar la evolución del plan de ejecución. Al crear una línea base SQL, SPM permitirá que solo se usen planes de ejecución aprobados. Otros planes descubiertos por el optimizador se almacenarán en el historial del plan SQL, pero no se utilizarán hasta que se aprueben.

* **Tareas de ajuste de SQL: Descripción general**

Muchas tareas de ajuste de SQL se deben realizar de forma regular. Es posible que vea una forma de reescribir una cláusula WHERE, pero puede depender de la creación de un nuevo índice. Esta lista de tareas le brinda un antecedente de algunas tareas importantes que deben realizarse y le da una idea de las dependencias que puede tener al ajustar SQL:

• Identificar las sentencias SQL de alta carga es una de las tareas más importantes que debe realizar. El ADDM es la herramienta ideal para esta tarea en particular.

• De manera predeterminada, la base de datos Oracle recopila estadísticas del optimizador automáticamente. Para esto, un trabajo está programado para ejecutarse en las ventanas de mantenimiento.

• Las estadísticas del sistema operativo proporcionan información sobre el uso y el rendimiento de los principales componentes de hardware, así como el rendimiento del propio sistema operativo.

• A menudo, hay un impacto beneficioso en el rendimiento al reconstruir los índices. Por ejemplo, eliminar índices no selectivos para acelerar el lenguaje de manipulación de datos (DML) o agregar columnas al índice para mejorar la selectividad.

• Puede mantener el plan de ejecución existente de las sentencias SQL a lo largo del tiempo utilizando estadísticas almacenadas o líneas base del plan SQL.

* **Dimensiones de ajuste de CPU y tiempo de espera**

Cuando sintoniza su sistema, es importante que compare el tiempo de CPU con el tiempo de espera de su sistema. Al comparar el tiempo de CPU con el tiempo de espera, puede determinar cuánto tiempo de respuesta se gasta en trabajo útil y cuánto espera los recursos que otros procesos pueden tener.

Como regla general, los sistemas donde el tiempo de CPU es dominante generalmente necesitan menos ajustes que aquellos en los que el tiempo de espera es dominante. Por otro lado, el alto uso de la CPU puede ser causado por declaraciones SQL mal escritas.

Aunque la proporción de tiempo de CPU a tiempo de espera siempre tiende a disminuir a medida que aumenta la carga en el sistema, los aumentos abruptos en el tiempo de espera son un signo de contención y deben abordarse para una buena escalabilidad.

Agregar más CPU a un nodo, o nodos a un clúster, proporcionaría un beneficio muy limitado bajo contención. Por el contrario, un sistema donde la proporción de tiempo de CPU no disminuye significativamente a medida que aumenta la carga puede escalar mejor, y lo más probable es que se beneficie al agregar

Instancias de CPU o Real Application Clusters (RAC) si es necesario.

Nota: los informes de AWR muestran el tiempo de CPU junto con el tiempo de espera en la sección de 5 eventos cronometrados principales, si la porción de tiempo de CPU se encuentra entre los cinco eventos principales.

* **Escalabilidad con diseño, implementación y configuración de aplicaciones**

El diseño, la implementación y la configuración deficientes de la aplicación tienen un impacto significativo en la escalabilidad. Esto resulta en:

• Mal diseño de SQL e índice, lo que resulta en un mayor número de entradas / salidas lógicas (I/O) para el mismo número de filas devueltas

• Disponibilidad reducida porque los objetos de la base de datos tardan más en mantenerse. Sin embargo, el diseño no es el único problema. La implementación física de la aplicación puede ser el eslabón débil, como en los siguientes ejemplos:

• Los sistemas pueden pasar a entornos de producción con SQL mal escrito que causa una alta I/O.

• Los COMPROMISOS o ROLLBACK de transacciones poco frecuentes pueden causar bloqueos largos en los recursos.

• El entorno de producción puede usar planes de ejecución diferentes a los generados en las pruebas.

• Las aplicaciones de uso intensivo de memoria que asignan una gran cantidad de memoria sin pensar demasiado en liberarla pueden causar una fragmentación excesiva de la memoria.

• El uso ineficiente de la memoria ejerce una gran presión sobre el subsistema operativo de memoria virtual.

Esto afecta el rendimiento y la disponibilidad.

* **Diseño de índice**

El diseño de índice también es un proceso en gran medida iterativo basado en el SQL generado por los diseñadores de aplicaciones. Sin embargo, es posible comenzar de manera sensata creando índices que impongan restricciones de clave externa (para reducir el tiempo de respuesta en las uniones entre tablas de clave principal y tablas de clave externa) y creando índices sobre datos a los que se accede con frecuencia, como el nombre de una persona. Las claves primarias y las claves únicas se indexan automáticamente, excepto por las restricciones DISABLE VALIDATE y DISABLE NOVALIDATE RELY. A medida que la aplicación evoluciona y las pruebas se realizan en tamaños realistas de datos, ciertas consultas necesitan mejoras de rendimiento, por lo que construir un mejor índice es una buena solución.

Las siguientes ideas de diseño de indexación deben considerarse al crear un nuevo índice.

Agregar columnas a un índice o usar tablas organizadas por índice Una de las formas más fáciles de acelerar una consulta es reducir el número de E / S lógicas eliminando un escaneo de tabla del plan de ejecución. Esto se puede hacer creando un índice sobre todas las columnas de la tabla a las que hace referencia la consulta. Estas columnas son las columnas de la lista de selección, las columnas de la cláusula WHERE y cualquier columna de unión u ordenación requerida. Esta técnica es particularmente útil para acelerar los tiempos de respuesta de una aplicación en línea cuando se reducen las E / S que consumen mucho tiempo. Esto se aplica mejor cuando se prueba la aplicación con datos de tamaño adecuado por primera vez. La forma más agresiva de esta técnica es construir una tabla organizada por índices (IOT).

* **Usar vistas**

Las vistas pueden acelerar y simplificar el diseño de la aplicación. Una definición de vista simple puede enmascarar la complejidad del modelo de datos de los programadores cuyas prioridades son recuperar, mostrar, recopilar y almacenar datos. Las vistas se utilizan a menudo para proporcionar restricciones de acceso simples a nivel de fila y columna.

Sin embargo, aunque las vistas proporcionan interfaces de programación limpias, pueden generar consultas subóptimas que requieren muchos recursos cuando se anidan demasiado. El peor tipo de uso de vista es crear uniones en vistas que hacen referencia a otras vistas, que a su vez hacen referencia a otras vistas. En muchos casos, los desarrolladores pueden satisfacer la consulta directamente desde la tabla sin usar una vista. Debido a sus propiedades inherentes, las vistas generalmente dificultan que el optimizador genere el plan de ejecución óptimo.

* **Eficiencia de ejecución de SQL**

Una aplicación diseñada para la eficiencia de ejecución de SQL debe admitir las siguientes características:

Buena gestión de la conexión de la base de datos: conectarse a la base de datos es una operación costosa que no es escalable. Por lo tanto, el número de conexiones simultáneas a la base de datos debe minimizarse tanto como sea posible. Un sistema simple, donde un usuario se conecta en la inicialización de la aplicación, es ideal. Sin embargo, en una aplicación basada en web o de varios niveles, donde los servidores de aplicaciones se utilizan para multiplexar las conexiones de la base de datos a los usuarios, esto puede ser difícil.

Con este tipo de aplicaciones, los esfuerzos de diseño deben garantizar que las conexiones de la base de datos se agrupen y no se restablezcan para cada solicitud de usuario.

Buen uso y administración del cursor: mantener las conexiones de los usuarios es igualmente importante para minimizar la actividad de análisis en el sistema. El análisis es el proceso de interpretar una instrucción SQL y crear un plan de ejecución para ella. Este proceso tiene muchas fases, que incluyen comprobación de sintaxis, comprobación de seguridad, generación de planes de ejecución y carga de estructuras compartidas en el grupo compartido.

Hay dos tipos de operaciones de análisis:

* **Hard parsing:** se envía una instrucción SQL por primera vez y no se encuentra ninguna coincidencia en el grupo compartido. Los análisis duros son los que requieren más recursos y no son escalables porque realizan todas las operaciones involucradas en un análisis.
* **Soft parsing:** se envía una instrucción SQL por primera vez y se encuentra una coincidencia en el grupo compartido. La coincidencia puede ser el resultado de la ejecución previa de otro usuario. La instrucción SQL se comparte, lo cual es bueno para el rendimiento. Sin embargo, los análisis suaves no son ideales porque aún requieren sintaxis y verificación de seguridad, que consumen recursos del sistema.

Debido a que el análisis debe minimizarse tanto como sea posible, los desarrolladores de aplicaciones deben diseñar sus aplicaciones para analizar las instrucciones SQL una vez y ejecutarlas muchas veces. Esto se hace a través de cursores. Los programadores de SQL con experiencia deben estar familiarizados con el concepto de abrir y volver a ejecutar cursores.

Los desarrolladores de aplicaciones también deben asegurarse de que las declaraciones SQL se compartan dentro del grupo compartido. Para hacer esto, vincule las variables para representar las partes de la consulta que cambian de ejecución en ejecución. Si esto no se hace, es probable que la instrucción SQL se analice una vez y nunca reutilizado por otros usuarios. Para asegurarse de que se comparte SQL, use variables de enlace y no use literales de cadena con instrucciones SQL.

* **Escribir SQL para compartir cursores**

Las aplicaciones pueden compartir cursores cuando el código se escribe de la misma manera (lo que permite que el sistema reconozca que dos declaraciones son iguales y, por lo tanto, se pueden compartir), incluso si utiliza algunos parámetros especiales de inicialización, como CURSOR\_SHARING, que se analiza más adelante en Lección titulada "Uso de variables de enlace". Debe desarrollar convenciones de codificación para sentencias SQL en consultas ad hoc, scripts SQL y llamadas de Oracle Call Interface (OCI).

Use código genérico compartido:

• Escribir y almacenar procedimientos que se puedan compartir entre aplicaciones.

• Usar desencadenantes de bases de datos.

• Escriba disparadores y procedimientos referenciados cuando use herramientas de desarrollo de aplicaciones.

• Escribir rutinas y procedimientos de la biblioteca en otros entornos.

Escriba a los estándares de formato:

• Desarrollar estándares de formato para todas las declaraciones, incluidas las del código PL / SQL.

• Desarrollar reglas para el uso de mayúsculas y minúsculas.

• Desarrollar reglas para el uso de espacios en blanco (espacios, tabulaciones, devoluciones).

• Desarrolle reglas para el uso de comentarios (preferiblemente manteniéndolos fuera de las propias declaraciones SQL).

• Utilice los mismos nombres para referirse a objetos de base de datos idénticos. Si es posible, prefije cada objeto con un nombre de esquema.

* **CHECKLIST verificación de rendimiento**

• Establecer el número mínimo de parámetros de inicialización. Idealmente, la mayoría de los parámetros de inicialización deben dejarse por defecto. Si hay más ajustes para realizar, esto aparece cuando el sistema está bajo carga. Establezca opciones de almacenamiento para tablas e índices en espacios de tablas apropiados.

• Verifique que todas las declaraciones SQL sean óptimas y comprenda su uso de recursos.

• Valide que el middleware y los programas que se conectan a la base de datos son eficientes en la administración de su conexión y no inician y cierran sesión repetidamente.

• Valide que las declaraciones SQL usen cursores de manera eficiente. Cada instrucción SQL debe analizarse una vez y luego ejecutarse varias veces. Esto sucede principalmente cuando las variables de enlace no se usan correctamente y los predicados de la cláusula WHERE se envían como literales de cadena.

• Valide que todos los objetos de esquema se migren correctamente del entorno de desarrollo a la base de datos de producción. Esto incluye tablas, índices, secuencias, disparadores, paquetes, procedimientos, funciones, objetos Java, sinónimos, concesiones y vistas. Asegúrese de que las modificaciones realizadas en las pruebas se realicen en el sistema de producción.

• Tan pronto como se implemente el sistema, establezca un conjunto básico de estadísticas de la base de datos y el sistema operativo. Este primer conjunto de estadísticas valida o corrige cualquier suposición hecha en el proceso de diseño y despliegue.

**Introducción al Optimizador**

* **Structured Query Language**

Todos los programas y usuarios acceden a los datos en una base de datos Oracle con el lenguaje SQL. Las herramientas Oracle y los programas de aplicación a menudo permiten a los usuarios acceder a la base de datos sin usar SQL directamente, pero estas aplicaciones deben usar SQL al ejecutar las solicitudes de los usuarios. Oracle Corp se esfuerza por cumplir con los estándares aceptados por la industria y participa en comités de estándares SQL (ANSI e ISO). Puede clasificar las declaraciones SQL en seis conjuntos principales:

• Las declaraciones del lenguaje de manipulación de datos (DML) manipulan o consultan datos en objetos de esquema existentes.

• Las declaraciones del lenguaje de definición de datos (DDL) definen, alteran la estructura de los objetos de esquema y los sueltan.

• Las declaraciones de control de transacciones (TCS) gestionan los cambios realizados por las declaraciones DML y agrupan las declaraciones DML en transacciones.

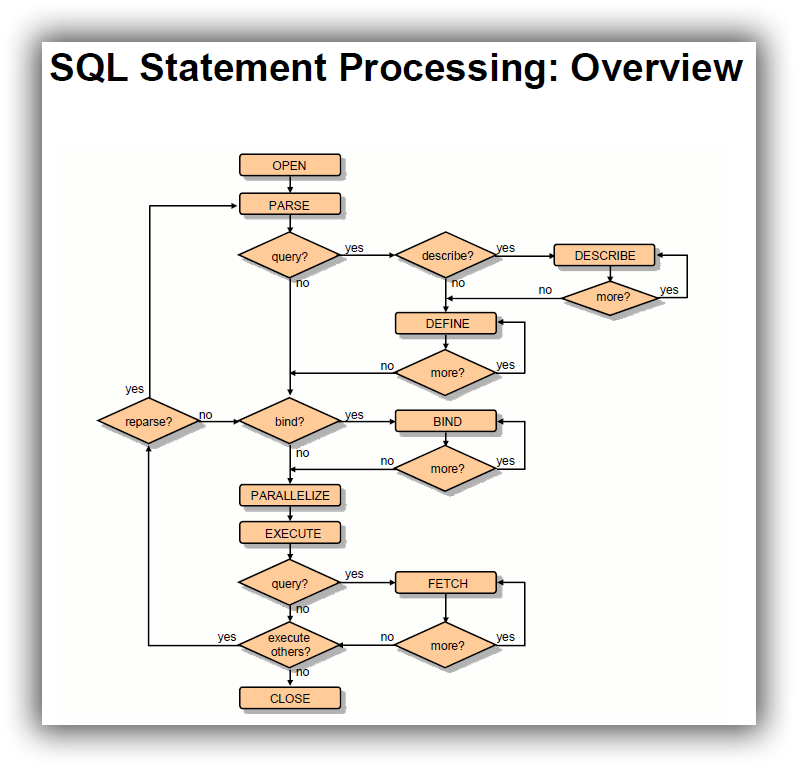
• Las declaraciones de control del sistema cambian las propiedades de la instancia de Oracle Database.

• Las declaraciones de control de sesión administran las propiedades de una sesión de usuario en particular.

• Las sentencias de SQL incorporado incorporan DDL, DML y TCS dentro de un programa de lenguaje de procedimiento, como PL / SQL y precompiladores de Oracle. Esta incorporación se realiza utilizando las declaraciones enumeradas en la diapositiva bajo la categoría ESS.

**Nota:** las declaraciones SELECT son las declaraciones más utilizadas. Si bien su curso se centra principalmente en consultas, es importante tener en cuenta que cualquier tipo de instrucción SQL está sujeta a optimización.

* **SQL Statement Processing: Overview**



1. Create a cursor.

2. Parse the statement.

3. Describe query results.

4. Define query output.

5. Bind variables.

6. Parallelize the statement.

7. Execute the statement.

8. Fetch rows of a query.

9. Close the cursor.

Tenga en cuenta que no todas las declaraciones requieren todos estos pasos. Por ejemplo, las declaraciones DDL no paralelas se requieren en solo dos pasos: Crear y Analizar.

Paralelizar la declaración implica decidir que se puede paralelizar en lugar de construir estructuras de ejecución paralelas.

* **Paso 1: crea un cursor**

• Un cursor es un identificador o nombre para un área SQL privada.

• Contiene información para el procesamiento de estados de cuenta.

• Se crea mediante una llamada a la interfaz del programa en espera de una instrucción SQL.

• La estructura del cursor es independiente de la instrucción SQL que contiene.

Un cursor puede considerarse como una asociación entre un área de datos del cursor en un programa cliente y las estructuras de datos del servidor Oracle. La mayoría de las herramientas de Oracle ocultan gran parte del manejo del cursor del usuario, pero los programas de Oracle Call Interface (OCI) necesitan la flexibilidad para poder procesar cada parte de la ejecución de consultas por separado. Por lo tanto, los precompiladores permiten la declaración explícita del cursor.

La mayor parte de esto también se puede hacer usando el paquete DBMS\_SQL también.

Un asa es similar al asa de una taza. Cuando mantiene presionado el controlador, tiene presionado el cursor. Es un identificador único para un cursor en particular que solo puede ser obtenido por un proceso a la vez.

Los programas deben tener un cursor abierto para procesar una instrucción SQL. El cursor contiene un puntero a la fila actual. El puntero se mueve a medida que se recuperan las filas hasta que no quedan más filas para procesar.

Las siguientes diapositivas usan el paquete DBMS\_SQL para ilustrar la gestión del cursor. Esto puede ser confuso para las personas que no están familiarizadas con él; sin embargo, es más amigable que PRO \* C u OCI. Es un poco problemático porque realiza FETCH y EXECUTE juntos, por lo que la fase de ejecución no se puede identificar por separado en el rastreo.

* **Paso 2: analizar la declaración**

• Declaración pasada del proceso de usuario a la instancia de Oracle

• Representación analizada de SQL creada y movida al área de SQL compartido si no hay un SQL idéntico en el área de SQL compartido

• Se puede reutilizar si existe un SQL idéntico

Durante el análisis, la instrucción SQL se pasa del proceso del usuario a la instancia de Oracle, y una representación analizada de la instrucción SQL se carga en un área SQL compartida.

La traducción y la verificación implican verificar si la declaración ya existe en el caché de la biblioteca.

Para declaraciones distribuidas, verifique la existencia de enlaces a bases de datos.

Por lo general, la fase de análisis se representa como la etapa donde se genera el plan de consulta.

El software del cliente puede diferir el paso de análisis para reducir el tráfico de red. Lo que esto significa es que el PARSE está incluido con el EJECUTAR, por lo que hay menos viajes de ida y vuelta al servidor.

**Nota:** Al verificar si las declaraciones son idénticas, deben ser idénticas en todos los aspectos, incluyendo mayúsculas y minúsculas.

* **Pasos 3 y 4: describir y definir**

• El paso de descripción proporciona información sobre los elementos de la lista de selección; es relevante al ingresar consultas dinámicas a través de una aplicación OCI.

• El paso de definición define la ubicación, el tamaño y la información del tipo de datos necesarios para almacenar valores recuperados en variables.

Paso 3: describe

La etapa de descripción es necesaria solo si no se conocen las características del resultado de una consulta, por ejemplo, cuando un usuario ingresa una consulta interactivamente. En este caso, la etapa de descripción determina las características (tipos de datos, longitudes y nombres) del resultado de una consulta. Describe le dice a la aplicación qué elementos de la lista de selección se requieren. Si, por ejemplo, ingresa una consulta como:

SQL> seleccionar \* de empleados ;,

Se requiere información sobre las columnas en la tabla de empleados.

Paso 4: definir

En la etapa de definición, especifique la ubicación, el tamaño y el tipo de datos de las variables definidas para recibir cada valor obtenido. Estas variables se llaman definir variables. Oracle Database realiza la conversión del tipo de datos, si es necesario.

Estos dos pasos generalmente están ocultos para los usuarios en herramientas como SQL \* Plus. Sin embargo, con DBMS\_SQL u OCI, es necesario decirle al cliente cuáles son los datos de salida y cuáles son las áreas de configuración.

* **Pasos 5 y 6: vincular y paralelizar**

• Vincular los valores de vinculación:

- Permite la dirección de memoria para almacenar valores de datos

- Permite compartir SQL aunque los valores de enlace pueden cambiar

• Paralelo a la declaración:

- SELECT

- INSERT

- UPDATE

- MERGE

- DELETE

- CREATE

- ALTER

**Paso 5: enlace**

En este punto, Oracle Database conoce el significado de la declaración SQL, pero aún no tiene suficiente información para ejecutar la declaración. Oracle Database necesita valores para cualquier variable listada en la declaración. El proceso de obtención de estos valores se denomina variables de enlace.

**Paso 6: Paralelo**

Oracle Database puede paralelizar la ejecución de sentencias SQL (como SELECT, INSERT, UPDATE, MERGE, DELETE) y algunas operaciones DDL, como la creación de índices, la creación de una tabla con una subconsulta y las operaciones en particiones. La paralelización hace que múltiples procesos del servidor realicen el trabajo de la instrucción SQL, por lo que puede completarse más rápido.

La paralelización implica dividir el trabajo de una declaración entre varios procesos esclavos.

El análisis ya ha identificado si una declaración se puede paralelizar o no y ha creado el plan paralelo apropiado. En el momento de la ejecución, este plan se implementa si hay suficientes recursos disponibles.

* **Pasos 7 a 9**

• Ejecutar:

- Controla la instrucción SQL para producir los resultados deseados.

• Obtener filas:

- En variables de salida definidas

- Resultados de la consulta devueltos en formato de tabla

- Mecanismo de búsqueda de matriz

• Cerrar el cursor.

**Pasos 7 a 9**

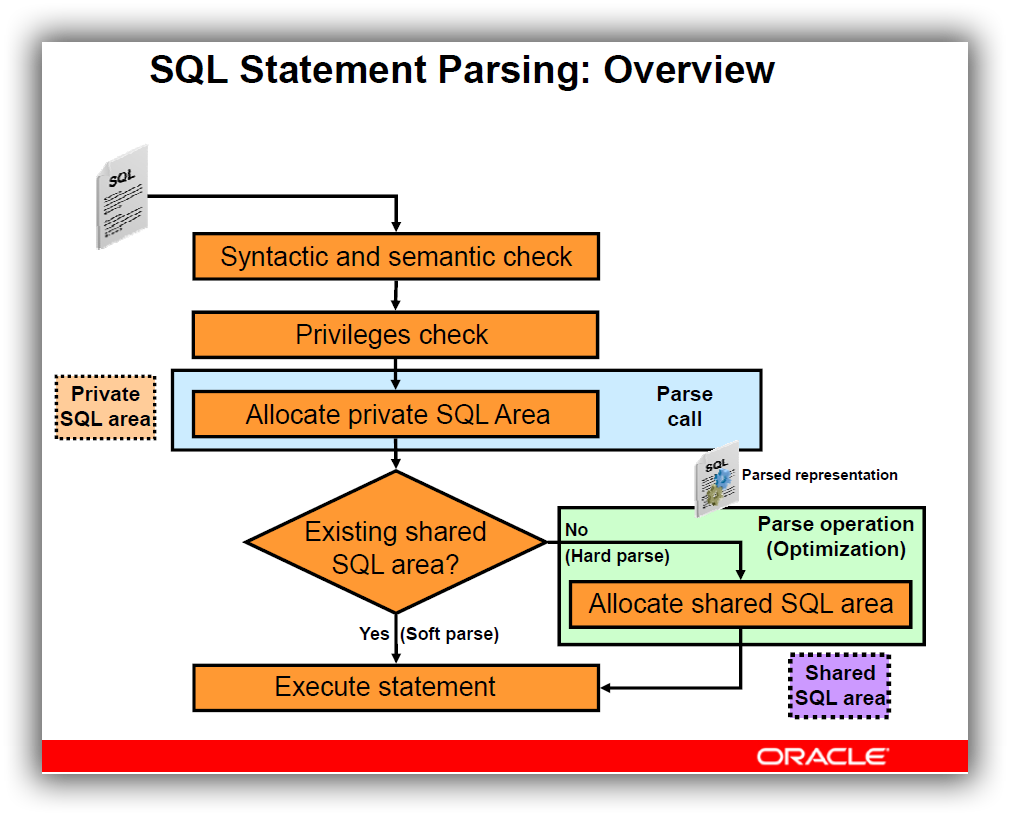
En este punto, Oracle Database tiene toda la información y los recursos necesarios, por lo que se ejecuta la declaración. Si la declaración es una consulta (sin la cláusula FOR UPDATE), no es necesario bloquear las filas porque no se modifican los datos. Sin embargo, si la declaración es una declaración UPDATE o DELETE, todas las filas a las que afecta la declaración se bloquean hasta el próximo COMMIT, ROLLBACK o SAVEPOINT para la transacción. Esto asegura la integridad de los datos.

Para algunas declaraciones, puede especificar una serie de ejecuciones a realizar. Esto se llama procesamiento de matriz. Dado n número de ejecuciones, se supone que las ubicaciones de enlace y definición son el comienzo de una matriz de tamaño n.

En la etapa de recuperación, las filas se seleccionan y ordenan (si así lo solicita la consulta), y cada recuperación sucesiva recupera otra fila del resultado hasta que se haya recuperado la última fila.

La etapa final del procesamiento de una declaración SQL es cerrar el cursor.

* **SQL Statement Parsing: Overview**



El análisis es una etapa en el procesamiento de una declaración SQL. Cuando una aplicación emite una declaración SQL, la aplicación realiza una llamada de análisis a Oracle Database. Durante la llamada de análisis, Oracle Database realiza las siguientes acciones:

• Verifica la declaración de validez sintáctica y semántica

• Determina si el proceso que emite la declaración tiene los privilegios para ejecutarla.

• Asigna un área SQL privada para la declaración

• Determina si existe o no un área SQL compartida que contenga la representación analizada de la declaración en la caché de la biblioteca. Si es así, el proceso de usuario utiliza esta representación analizada y ejecuta la declaración de inmediato. De lo contrario, Oracle Database genera la representación analizada de la declaración, y el proceso del usuario asigna un área SQL compartida para la declaración en el caché de la biblioteca y almacena allí su representación analizada.

Tenga en cuenta la diferencia entre una aplicación que realiza una llamada de análisis para una instrucción SQL y Oracle Database realmente analiza la instrucción.

• Una llamada de análisis de la aplicación asocia una instrucción SQL con un área privada de SQL. Después de que una declaración se haya asociado con un área SQL privada, se puede ejecutar repetidamente sin que su aplicación realice una llamada de análisis.

• Una operación de análisis de Oracle Database asigna un área SQL compartida para una instrucción SQL. Después de que se haya asignado un área SQL compartida para una instrucción, se puede ejecutar repetidamente sin volver a analizarla.

Tanto las llamadas de análisis como el análisis pueden ser costosos en relación con la ejecución, por lo tanto, realícelos tan raramente como sea posible.

Nota: Aunque el análisis de una declaración SQL valida esa declaración, el análisis solo identifica errores que se pueden encontrar antes de la ejecución de la declaración. Por lo tanto, algunos errores no pueden detectarse mediante el análisis. Por ejemplo, los errores en la conversión de datos o los errores en los datos (como un intento de ingresar valores duplicados en una clave primaria) y los puntos muertos son todos errores o situaciones que pueden encontrarse e informarse solo durante la etapa de ejecución.

* **Optimización durante la operación Hard Parse**

El optimizador crea el plan de ejecución para una instrucción SQL.

Las consultas SQL enviadas al sistema se ejecutan primero a través del analizador, que verifica la sintaxis y analiza la semántica. El resultado de esta fase se denomina representación analizada de la declaración y está constituido por un conjunto de bloques de consulta. Un bloque de consulta es un DML autónomo contra una tabla. Un bloque de consulta puede ser un DML de nivel superior o una subconsulta. Esta representación analizada se envía al optimizador, que maneja tres funcionalidades principales:

Transformación, estimación y generación de planes de ejecución.

Antes de realizar cualquier cálculo de costo, el sistema puede transformar su estado de cuenta en un estado de cuenta equivalente y calcular el costo del estado de cuenta equivalente. Dependiendo de la versión de Oracle Database, hay transformaciones que no se pueden hacer, algunas que siempre se hacen y otras que se hacen, cuestan y descartan.

La entrada al transformador de consulta es una consulta analizada, que está representada por un conjunto de bloques de consulta interrelacionados. El objetivo principal del transformador de consultas es determinar si es ventajoso cambiar la estructura de la consulta para permitir la generación de un mejor plan de consulta. El transformador de consultas emplea varias técnicas de transformación de consultas, como la transitividad, la fusión de vistas, la inserción de predicados, la anulación de subconsultas, la reescritura de consultas, la transformación en estrella y la expansión OR.

* **Optimizador basado en costos (CBO)**

• Pieza de código:

- Estimador

- Generador de planos

• El estimador determina el costo de las sugerencias de optimización realizadas por el generador del plan:

- Costo: la mejor estimación del optimizador del número de E / S estandarizadas realizadas para ejecutar una optimización de declaración particular

• Generador de planes:

- Prueba diferentes técnicas de optimización de sentencias

- Utiliza el estimador para costar cada sugerencia de optimización

- Elige la mejor sugerencia de optimización basada en el costo

- Genera un plan de ejecución para la mejor optimización.

La combinación del estimador y el código del generador de plan se denominan comúnmente optimizador basado en costos (CBO).

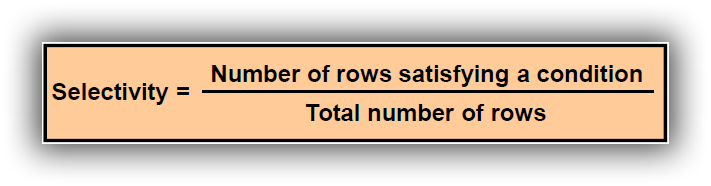
El estimador genera tres tipos de medidas: selectividad, cardinalidad y costo. Estas medidas están relacionadas entre sí. La cardinalidad se deriva de la selectividad y, a menudo, el costo depende de la cardinalidad. El objetivo final del estimador es estimar el costo total de un plan dado. Si hay estadísticas disponibles, el estimador las utiliza para mejorar el grado de precisión al calcular las medidas.

La función principal del generador de planes es probar diferentes planes posibles para una consulta determinada y elegir el que tenga el costo más bajo. Son posibles muchos planes diferentes debido a las diversas combinaciones de diferentes rutas de acceso, métodos de unión y órdenes de unión que pueden usarse para acceder y procesar datos de diferentes maneras y producir el mismo resultado. El número de planes posibles para un bloque de consulta es proporcional al número de elementos de unión en la cláusula FROM.

Este número aumenta exponencialmente con el número de elementos de unión.

El optimizador utiliza varios elementos de información para determinar la mejor ruta: cláusula WHERE, estadísticas, parámetros de inicialización, sugerencias proporcionadas e información de esquema.

* **Estimator: Selectivity**



• La selectividad es la proporción estimada de un conjunto de filas recuperado por un predicado particular o una combinación de predicados.

• Se expresa como un valor entre 0.0 y 1.0:

- Alta selectividad: pequeña proporción de filas

- Baja selectividad: gran proporción de filas

• Cálculo de selectividad:

- Si no hay estadísticas: use muestreo dinámico

- Si no hay histogramas: suponga una distribución uniforme de filas

• Información estadística:

- DBA\_TABLES y DBA\_TAB\_STATISTICS (NUM\_ROWS)

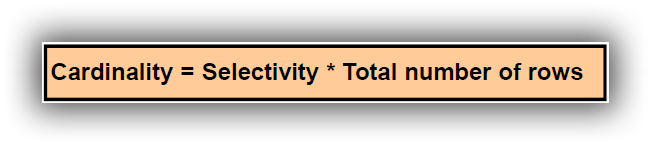
- DBA\_TAB\_COL\_STATISTICS (NUM\_DISTINCT, DENSITY, HIGH / LOW\_VALUE, ...)

La selectividad representa una fracción de filas de un conjunto de filas. El conjunto de filas puede ser una tabla base, una vista o el resultado de una unión o un operador GROUP BY. La selectividad está vinculada a un predicado de consulta, como last\_name = 'Smith', o una combinación de predicados, como last\_name = 'Smith' AND job\_type = 'Clerk'. Un predicado actúa como un filtro que filtra un cierto número de filas de un conjunto de filas. Por lo tanto, la selectividad de un predicado indica el porcentaje de filas de un conjunto de filas que pasa la prueba de predicado. La selectividad se encuentra en un rango de valores de 0.0 a 1.0. Una selectividad de 0.0 significa que no se seleccionan filas de un conjunto de filas, y una selectividad de 1.0 significa que se seleccionan todas las filas.

Si no hay estadísticas disponibles, el optimizador utiliza muestreo dinámico o un valor predeterminado interno, según el valor del parámetro de inicialización OPTIMIZER\_DYNAMIC\_SAMPLING. Cuando hay estadísticas disponibles, el estimador las usa para estimar la selectividad. Por ejemplo, para un predicado de igualdad (last\_name = 'Smith'), la selectividad se establece en el recíproco del número n de valores distintos de LAST\_NAME porque la consulta selecciona filas que contienen uno de n valores distintos. Por lo tanto, se supone una distribución uniforme. Si hay un histograma disponible en la columna LAST\_NAME, el estimador lo usa en lugar del número de valores distintos. El histograma captura la distribución de diferentes valores en una columna, por lo que produce mejores estimaciones de selectividad.

Nota: Es importante tener histogramas en columnas que contengan valores con grandes variaciones en el número de duplicados (sesgo de datos).

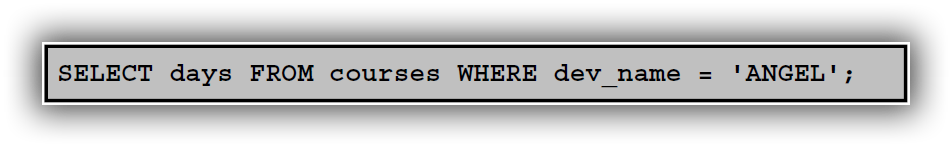
* **Estimador: cardinalidad**

****

• Número esperado de filas recuperadas por una operación particular en el plan de ejecución

• Figura vital para determinar costos de unión, filtros y clasificación.

• Ejemplo simple:



- El número de valores distintos en DEV\_NAME es 203.

- El número de filas en CURSOS (cardinalidad original) es 1018.

- Selectividad = 1/203 = 4.926 \* e-03

- Cardinalidad = (1/203) \* 1018 = 5.01 (redondeado a 6)

La cardinalidad de una operación en particular en el plan de ejecución de una consulta representa el número estimado de filas recuperadas por esa operación en particular. La mayoría de las veces, el origen de la fila puede ser una tabla base, una vista o el resultado de una unión o un operador GROUP BY.

Al calcular el costo de una operación de unión, es importante conocer la cardinalidad del origen de la fila de conducción.

Con la unión de bucles anidados, por ejemplo, el origen de la fila de conducción define con qué frecuencia el sistema sondea el origen de la fila interior.

Debido a que los costos de clasificación dependen del tamaño y el número de filas que se ordenarán, las cifras de cardinalidad también son vitales para el costo de clasificación.

En el ejemplo de la diapositiva, basado en estadísticas asumidas, el optimizador sabe que hay 203 valores diferentes en la columna DEV\_NAME y que el número total de filas en la tabla CURSOS es 1018. En base a esta suposición, el optimizador deduce que la selectividad del predicado DEV\_NAME = 'ANGEL' es 1/203 (suponiendo que no haya histogramas), y también deduce que la cardinalidad de la consulta es (1/203) \* 1018. Este número se redondea al entero más cercano, 6.

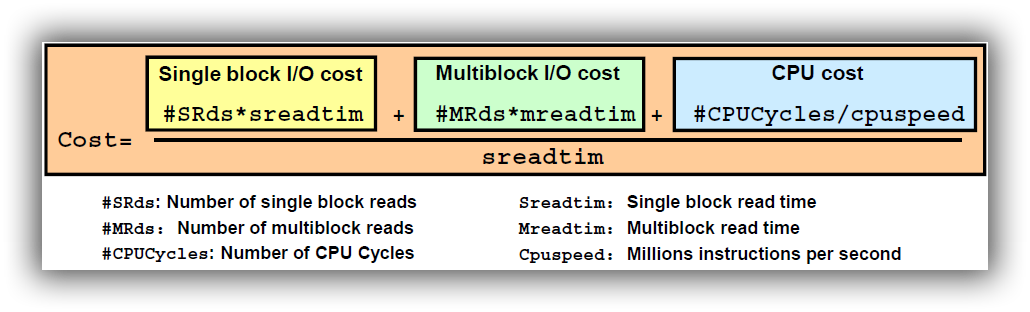
* **Estimador: costo**

• El costo es la mejor estimación del optimizador del número de E / S estandarizadas que se necesitan para ejecutar una declaración en particular.

• La unidad de costo es una lectura aleatoria estandarizada de un solo bloque:

- 1 unidad de costo = 1 SRds

• La fórmula de costos combina tres unidades de costos diferentes en unidades de costos estándar.



El costo de una declaración representa la mejor estimación del optimizador del número de entradas / salidas estandarizadas (E / S) que se necesita para ejecutar esa declaración. Básicamente, el costo es un valor normalizado en términos de un número de lecturas aleatorias de un solo bloque. La métrica de costo estándar medida por el optimizador es en términos del número de lecturas aleatorias de un solo bloque, por lo que una unidad de costo corresponde a una lectura aleatoria de un solo bloque. La fórmula que se muestra en la diapositiva combina tres unidades de costo diferentes:

• Tiempo estimado para hacer todas las lecturas aleatorias de bloque único

• Tiempo estimado para hacer todas las lecturas multibloque

• Tiempo estimado para que la CPU procese el estado de cuenta en una unidad de costo estándar

El modelo incluye el costo de la CPU porque en la mayoría de los casos la utilización de la CPU es tan importante como la E / S; a menudo es la única contribución al costo (en casos de clasificación en memoria, hash, evaluación de predicados y E / S en caché).

Este modelo es sencillo para la ejecución en serie. Para la ejecución paralela, se realizan los ajustes necesarios mientras se calculan las estimaciones para #SRds, #MRds y #CPUCycles.

Nota: #CPUCycles incluye el costo de procesamiento de consultas de la CPU (costo de CPU puro) y el costo de recuperación de datos de la CPU (costo de CPU de la memoria caché del búfer).

* **Controlando el comportamiento del optimizador**

• CURSOR\_SHARING: SIMILAR, EXACT, FORCE

• DB\_FILE\_MULTIBLOCK\_READ\_COUNT

• PGA\_AGGREGATE\_TARGET

• STAR\_TRANSFORMATION\_ENABLED

• RESULT\_CACHE\_MODE: MANUAL, FORCE

• RESULT\_CACHE\_MAX\_SIZE

• RESULT\_CACHE\_MAX\_RESULT

• RESULT\_CACHE\_REMOTE\_EXPIRATION

Estos parámetros controlan el comportamiento del optimizador:

• **CURSOR\_SHARING** determina qué tipo de instrucciones SQL pueden compartir los mismos cursores:

- FORCE: fuerza declaraciones que pueden diferir en algunos literales, pero que son idénticas, para compartir un cursor, a menos que los literales afecten el significado de la declaración

- SIMILAR: hace que las declaraciones que pueden diferir en algunos literales, pero que son idénticas, compartan un cursor, a menos que los literales afecten el significado de la declaración o el grado en que se optimiza el plan. Forzar el uso compartido del cursor entre sentencias similares (pero no idénticas) puede tener resultados inesperados en algunas aplicaciones del sistema de soporte de decisiones (DSS) o en aplicaciones que utilizan esquemas almacenados.

- EXACTO: solo permite que las declaraciones con texto idéntico compartan el mismo cursor. Este es el valor predeterminado.

• **DB\_FILE\_MULTIBLOCK\_READ\_COUNT** es uno de los parámetros que puede usar para minimizar las E / S durante los escaneos de tablas o indexar el escaneo completo rápido. Especifica el número máximo de bloques leídos en una operación de E / S durante una exploración secuencial. El número total de E / S necesarias para realizar una exploración de tabla siendo utilizado para la operación. A partir de Oracle Database 10g, Versión 2, el valor predeterminado de este parámetro es un valor que corresponde al tamaño máximo de E / S que se puede realizar de manera eficiente. Este valor depende de la plataforma y se calcula al inicio de la instancia para la mayoría de las plataformas.

Debido a que el parámetro se expresa en bloques, automáticamente calcula un valor que es igual al tamaño máximo de E / S que se puede realizar de manera eficiente dividido por el tamaño de bloque estándar. Tenga en cuenta que si el número de sesiones es extremadamente grande, el valor del recuento de lectura multibloque disminuye para evitar que el caché del búfer se inunde con demasiados búferes de escaneo de tablas. Aunque el valor predeterminado puede ser un valor grande, el optimizador no favorece los planes grandes si no configura este parámetro. Solo lo haría si establece explícitamente este parámetro en un valor grande. Básicamente, si este parámetro no se establece explícitamente (o si se establece es 0), el optimizador utiliza un valor predeterminado de 8 al calcular el costo de los escaneos de tablas completas e indexar los escaneos completos rápidos. El procesamiento de transacciones en línea (OLTP) y los entornos por lotes suelen tener valores en el rango de 4 a 16 para este parámetro. Los entornos de almacenamiento de datos y DSS tienden a beneficiarse más al maximizar el valor de este parámetro. Es más probable que el optimizador seleccione una exploración de tabla completa sobre un índice, si el valor de este parámetro es alto.

• **PGA\_AGGREGATE\_TARGET** especifica la memoria PGA agregada de destino disponible para todos los procesos del servidor conectados a la instancia. Establecer PGA\_AGGREGATE\_TARGET en un valor distinto de cero tiene el efecto de establecer automáticamente el parámetro WORKAREA\_SIZE\_POLICY en AUTO. Esto significa que las áreas de trabajo de SQL utilizadas por los operadores de SQL con uso intensivo de memoria (como ordenar, agrupar por, combinación de hash, fusión de mapa de bits y creación de mapa de bits) se dimensionan automáticamente. El valor predeterminado para este parámetro es cero porque, a menos que especifique lo contrario, el sistema lo establece al 20% de la SGA o 10 MB, lo que sea mayor. Establecer PGA\_AGGREGATE\_TARGET en 0 establece automáticamente el parámetro WORKAREA\_SIZE\_POLICY en MANUAL. Esto significa que las áreas de trabajo de SQL se dimensionan utilizando los parámetros \* \_AREA\_SIZE. El sistema intenta mantener la cantidad de memoria privada por debajo del objetivo especificado por este parámetro adaptando el tamaño de las áreas de trabajo a la memoria privada. Al aumentar el valor de este parámetro, indirectamente aumenta la memoria asignada a las áreas de trabajo. En consecuencia, las operaciones más intensivas en memoria pueden ejecutarse completamente en la memoria y un número menor de ellas se abren camino hacia el disco. Al configurar este parámetro, debe examinar la memoria total en su sistema que está disponible para la instancia de Oracle y restar el SGA.

Puede asignar la memoria restante a PGA\_AGGREGATE\_TARGET.

• **STAR\_TRANSFORMATION\_ENABLED** determina si una transformación de consulta basada en el costo se aplica a las consultas en estrella. Esta optimización se explica en la lección titulada "Estudio de caso: transformación estelar".

• El optimizador de consultas gestiona el mecanismo de caché de resultados en función de la configuración del parámetro RESULT\_CACHE\_MODE en el archivo de parámetros de inicialización. Puede usar este parámetro para determinar si el optimizador envía o no automáticamente los resultados de las consultas al caché de resultados. Los posibles valores de los parámetros son MANUAL y FORCE:

- Cuando se establece en MANUAL (el valor predeterminado), debe especificar, utilizando la sugerencia RESULT\_CACHE, que un resultado particular se almacenará en la memoria caché.

- Cuando se establece en FORCE, todos los resultados se almacenan en la memoria caché. Para la configuración FORCE, si la instrucción contiene una sugerencia [NO\_] RESULT\_CACHE, la sugerencia tiene prioridad sobre la configuración del parámetro.

• El tamaño de la memoria asignada a la caché de resultados depende del tamaño de la memoria del SGA, así como del sistema de administración de memoria. Puede cambiar la memoria asignada a la caché de resultados configurando el parámetro RESULT\_CACHE\_MAX\_SIZE. El caché de resultados completa o una exploración completa rápida de índice depende de factores, como el tamaño del segmento, el recuento de lectura multibloque y la ejecución paralela se deshabilita si establece su valor en 0. El valor de este parámetro se redondea al múltiplo más grande de 32 KB que no sea mayor que el valor especificado. Si el valor redondeado es 0, la función está deshabilitada.

• Use el parámetro **RESULT\_CACHE\_MAX\_RESULT** para especificar la cantidad máxima de memoria caché que puede usar cualquier resultado individual. El valor predeterminado es 5%, pero puede especificar cualquier valor de porcentaje entre 1 y 100.

• Utilice el parámetro **RESULT\_CACHE\_REMOTE\_EXPIRATION** para especificar el tiempo (en número de minutos) durante el cual un resultado que depende de los objetos de la base de datos remota sigue siendo válido. El valor predeterminado es 0, lo que implica que los resultados que usan objetos remotos no deben almacenarse en caché. Establecer este parámetro en un valor distinto de cero puede producir respuestas obsoletas, por ejemplo, si la tabla remota utilizada por un resultado se modifica en la base de datos remota.

• OPTIMIZER\_INDEX\_CACHING

• OPTIMIZER\_INDEX\_COST\_ADJ

• OPTIMIZER\_FEATURES\_ENABLED

• OPTIMIZER\_MODE: ALL\_ROWS, FIRST\_ROWS, FIRST\_ROWS\_n

• OPTIMIZER\_CAPTURE\_SQL\_PLAN\_BASELINES

• OPTIMIZER\_USE\_SQL\_PLAN\_BASELINES

• OPTIMIZER\_DYNAMIC\_SAMPLING

• OPTIMIZER\_USE\_INVISIBLE\_INDEXES

• OPTIMIZER\_USE\_PENDING\_STATISTICS

• **OPTIMIZER\_INDEX\_CACHING:** este parámetro controla el costeo de una sonda de índice junto con un bucle anidado o un iterador en lista. El rango de valores de 0 a 100 para OPTIMIZER\_INDEX\_CACHING indica el porcentaje de bloques de índice en la memoria caché del búfer, que modifica las suposiciones del optimizador sobre el almacenamiento en caché de índice para bucles anidados e iteradores en lista. Un valor de 100 infiere que es probable que el 100% de los bloques de índice se encuentren en la memoria caché del búfer y el optimizador ajusta el costo de una sonda de índice o un bucle anidado en consecuencia. El valor predeterminado para este parámetro es 0, lo que da como resultado un comportamiento de optimizador predeterminado. Tenga cuidado al usar este parámetro porque los planes de ejecución pueden cambiar a favor del almacenamiento en caché de índice.

• **OPTIMIZER\_INDEX\_COST\_ADJ** le permite ajustar el comportamiento del optimizador para que la selección de la ruta de acceso sea más o menos amigable con el índice, es decir, para que el optimizador sea más o menos propenso a seleccionar una ruta de acceso al índice en un escaneo completo de la tabla. El rango de valores es de 1 a 10000. El valor predeterminado para este parámetro es el 100 por ciento, en el cual el optimizador evalúa las rutas de acceso de índice al costo regular. Cualquier otro valor hace que el optimizador evalúe la ruta de acceso a ese porcentaje del costo regular. Por ejemplo, una configuración de 50 hace que la ruta de acceso al índice parezca la mitad de cara de lo normal.

• **OPTIMIZER\_FEATURES\_ENABLED** actúa como un parámetro general para habilitar una serie de funciones optimizadoras basadas en un número de versión de Oracle.

Por ejemplo, si actualiza su base de datos de la versión 10.1 a la versión 11.1, pero desea mantener el comportamiento del optimizador de la versión 10.1, puede hacerlo estableciendo este parámetro en 10.1.0. En un momento posterior, puede probar las mejoras introducidas en las versiones hasta la versión 11.1 inclusive, estableciendo el parámetro en 11.1.0.6. Sin embargo, no se recomienda establecer explícitamente el parámetro OPTIMIZER\_FEATURES\_ENABLE en una versión anterior. Para evitar una posible regresión del rendimiento de SQL que pueda resultar de cambios en el plan de ejecución, considere usar la administración del plan de SQL.

• **OPTIMIZER\_MODE** establece el comportamiento predeterminado para seleccionar un enfoque de optimización para la instancia o su sesión. Los valores posibles son:

- ALL\_ROWS: el optimizador utiliza un enfoque basado en el costo para todas las declaraciones SQL en la sesión, independientemente de la presencia de estadísticas y se optimiza con el objetivo del mejor rendimiento (uso mínimo de recursos para completar la declaración completa). Este es el valor predeterminado.

- FIRST\_ROWS\_n: el optimizador utiliza un enfoque basado en el costo, independientemente de la presencia de estadísticas, y optimiza con el objetivo del mejor tiempo de respuesta para devolver el primer n número de filas; n puede ser igual a 1, 10, 100 o 1000.

- FIRST\_ROWS: el optimizador utiliza una combinación de costo y heurística para encontrar el mejor plan para la entrega rápida de las primeras filas. El uso de la heurística a veces lleva al optimizador de consultas a generar un plan con un costo que es significativamente mayor que el costo de un plan sin aplicar la heurística. FIRST\_ROWS está disponible para compatibilidad con versiones anteriores y estabilidad del plan; use FIRST\_ROWS\_n en su lugar.

• **OPTIMIZER\_CAPTURE\_SQL\_PLAN\_BASELINES** habilita o deshabilita el reconocimiento automático de sentencias SQL repetibles, así como la generación de líneas base del plan SQL para dichas sentencias.

• **OPTIMIZER\_USE\_SQL\_PLAN\_BASELINES** habilita o deshabilita el conjunto de líneas base del plan SQL almacenadas en SQL Management Base. Cuando está habilitado, el optimizador busca una línea base del plan SQL para la declaración SQL que se está compilando. Si se encuentra uno en SQL Management Base, el optimizador le cuesta a cada uno de los planes de referencia y elige uno con el costo más bajo.

• **OPTIMIZER\_DYNAMIC\_SAMPLING** controla el nivel de muestreo dinámico realizado por el optimizador. Si OPTIMIZER\_FEATURES\_ENABLE se establece en:

- 10.0.0 o posterior, el valor predeterminado es 2

- 9.2.0, el valor predeterminado es 1

- 9.0.1 o anterior, el valor predeterminado es 0

• **OPTIMIZER\_USE\_INVISIBLE\_INDEXES** habilita o deshabilita el uso de índices invisibles.

• **OPTIMIZER\_USE\_PENDING\_STATISTICS** especifica si el optimizador utiliza o no estadísticas pendientes al compilar sentencias SQL.

**Nota:** los índices invisibles, las estadísticas pendientes y el muestreo dinámico se analizan más adelante en este curso.

**Interpretando Planes de Ejecución**

* **¿Qué es un plan de ejecución?**

Un plan de ejecución es la salida del optimizador y se presenta al motor de ejecución para su implementación. Instruye al motor de ejecución sobre las operaciones que debe realizar para recuperar los datos requeridos por una consulta de la manera más eficiente.

La instrucción EXPLAIN PLAN reúne los planes de ejecución elegidos por el optimizador de Oracle para las instrucciones SELECT, UPDATE, INSERT y DELETE. Los pasos del plan de ejecución no se realizan en el orden en que están numerados. Hay una relación padre-hijo entre los pasos. El árbol fuente de la fila es el núcleo del plan de ejecución.

Muestra la siguiente información:

• Un orden de las tablas a las que hace referencia la declaración

• Un método de acceso para cada tabla mencionada en la declaración

• Un método de unión para tablas afectadas por operaciones de unión en la instrucción

• Operaciones de datos, como filtro, clasificación o agregación.

Además del árbol de origen de fila (o árbol de flujo de datos para operaciones paralelas), la tabla de plan contiene información sobre lo siguiente:

• Optimización, como el costo y la cardinalidad de cada operación.

• Particionamiento, como el conjunto de particiones a las que se accede

• Ejecución paralela, como el método de distribución de entradas de unión.

Los resultados de EXPLAIN PLAN lo ayudan a determinar si el optimizador selecciona un plan de ejecución particular, como la unión de bucles anidados.

* **¿Dónde encontrar planes de ejecución?**

• PLAN\_TABLE (SQL Developer or SQL\*Plus)

• V$SQL\_PLAN (Library Cache)

• V$SQL\_PLAN\_MONITOR (11g)

• DBA\_HIST\_SQL\_PLAN (AWR)

• STATS$SQL\_PLAN (Statspack)

• SQL management base (SQL plan baselines)

• SQL tuning set

• Trace files generated by DBMS\_MONITOR

• Event 10053 Trace File, se utiliza para volcar los cálculos del (CBO) puede incluir un plan.

• Process state dump trace file since 10gR2

* **Viewing Execution Plans**

• The EXPLAIN PLAN command followed by:

– SELECT from PLAN\_TABLE

– DBMS\_XPLAN.DISPLAY()

• SQL\*Plus Autotrace: SET AUTOTRACE ON

• DBMS\_XPLAN.DISPLAY\_CURSOR()

• DBMS\_XPLAN.DISPLAY\_AWR()

• DBMS\_XPLAN.DISPLAY\_SQLSET()

• DBMS\_XPLAN.DISPLAY\_SQL\_PLAN\_BASELINE()

**Nota:** Puede crear su propia PLAN\_TABLE usando el script $ORACLE\_HOME/rdbms/admin/utlxplan.sql si desea mantener la información del plan de ejecución a largo plazo.

**Nota:** Alternativamente, puede ejecutar el script utlxpls.sql (o utlxplp.sql para consultas paralelas) (ubicado en el directorio $ORACLE\_HOME/rdbms/admin/) para mostrar el plan de ejecución almacenado en PLAN\_TABLE para la última instrucción explicada. Este script usa la función de tabla DISPLAY del paquete DBMS\_XPLAN.

* **Displaying from PLAN\_TABLE: ALL**

• ROWS: si es relevante, muestra el número de filas estimadas por el optimizador

• BYTES: si es relevante, muestra el número de bytes estimados por el optimizador

• COST: si es relevante, muestra información del costo del optimizador

• PARTITION: si es relevante, muestra información de poda de partición

• PARALLEL: si es relevante, muestra información PX (método de distribución e información de la cola de la tabla)

• PREDICATE: si es relevante, muestra la sección de predicado

• PROJECTION: si es relevante, muestra la sección de proyección

• ALIAS: si es relevante, muestra la sección "Nombre de bloque de consulta / Alias de objeto"

• REMOTE: si es relevante, muestra la información de la consulta distribuida (por ejemplo, remota desde la distribución en serie y SQL remoto)

• NOTA: si es relevante, muestra la sección de notas del plan de explicación

Si la tabla del plan de destino también almacena columnas de estadísticas del plan (por ejemplo, es una tabla utilizada para capturar el contenido de la vista fija **V$SQL\_PLAN\_STATISTICS\_ALL**), se pueden usar palabras clave de formato adicionales para especificar qué clase de estadísticas mostrar cuando se usa DISPLAY función. Estas palabras clave de formato adicionales son IOSTATS, MEMSTATS, ALLSTATS y LAST.

**Nota:** Las palabras clave de formato pueden tener como prefijo el signo "-" para excluir la información especificada.

Por ejemplo, "-PROYECCIÓN" excluye la información de proyección.

**Nota:** Cuando el formato AVANZADO se usa con **V$SQL\_PLAN**, hay una sección más llamada Enlaces Peeked (identificados por posición).

* **AUTOTRACE**

Para usar esta función, debe tener un PLAN\_TABLE disponible en su esquema y luego tener el rol PLUSTRACE otorgado. Los privilegios de administrador de la base de datos (DBA) son necesarios para otorgar el rol PLUSTRACE. El rol PLUSTRACE se crea y se otorga al rol DBA ejecutando el script $ORACLE\_HOME/sqlplus/admin/plustrce.sql suministrado.

El rol PLUSTRACE contiene el privilegio de selección en tres vistas V$. Estos privilegios son necesarios para generar estadísticas de AUTOTRACE.

AUTOTRACE es una excelente herramienta de diagnóstico para el ajuste de sentencias SQL. Debido a que es puramente declarativo, es más fácil de usar que EXPLAIN PLAN.

**Nota:** El sistema no es compatible con EXPLAIN PLAN para declaraciones que realizan conversión de tipo implícito de variables de enlace de fecha. Con las variables de enlace en general, la salida EXPLAIN PLAN podría no representar el plan de ejecución real.

**Nota:** El sistema no es compatible con EXPLAIN PLAN para declaraciones que realizan conversión de tipo implícito de variables de enlace de fecha. Con las variables de enlace en general, la salida EXPLAIN PLAN podría no representar el plan de ejecución real.

***AUTOTRACE: ejemplos***

• Para iniciar declaraciones de rastreo utilizando AUTOTRACE:

SQL> set autotrace on

• Para mostrar el plan de ejecución solo sin ejecución:

SQL> set autotrace traceonly explain

• Para mostrar filas y estadísticas:

SQL> set autotrace on statistics

• Para obtener solo el plan y las estadísticas (suprimir filas):

SQL> set autotrace traceonly

***AUTOTRACE: Estadísticas***

|  |
| --- |
| SQL> show autotrace  autotrace OFF  SQL> set autotrace traceonly statistics  SQL> SELECT \* FROM oe.products;  288 rows selected.  Statistics  --------------------------------------------------------  1334 recursive calls  0 db block gets  686 consistent gets  394 physical reads  0 redo size  103919 bytes sent via SQL\*Net to client  629 bytes received via SQL\*Net from client  21 SQL\*Net roundtrips to/from client  22 sorts (memory)  0 sorts (disk)  288 rows processed |

El servidor registra las estadísticas cuando se ejecuta su declaración e indica los recursos del sistema necesarios para ejecutar su declaración. Los resultados incluyen las siguientes estadísticas:

• recursive calls es el número de llamadas recursivas generadas tanto a nivel de usuario como de sistema. Oracle Database mantiene las tablas utilizadas para el procesamiento interno. Cuando Oracle Database necesita hacer un cambio en estas tablas, genera internamente una instrucción SQL interna, que a su vez genera una llamada recursiva.

• db block gets es el número de veces que se solicitó un bloque CURRENT.

• consistent gets es el número de veces que se solicitó una lectura consistente para un bloque.

• physical reads es el número total de bloques de datos leídos del disco. Este número es igual al valor de "lecturas físicas directas" más todas las lecturas en la memoria caché del búfer.

• redo size es la cantidad total de rehacer generada en bytes.

• bytes sent via SQL\*Net to client es el número total de bytes enviados al cliente desde los procesos en primer plano.

• bytes received via SQL\*Net from client es el número total de bytes recibidos del cliente a través de Oracle Net.

• SQL\*Net roundtrips to/from client son el número total de mensajes de Oracle Net enviados y recibidos del cliente.

Nota: Las estadísticas impresas por AUTOTRACE se recuperan de V$SESSTAT.

• sorts (memory) es el número de operaciones de clasificación que se realizaron por completo en la memoria y no requirieron ninguna escritura en el disco.

• sorts (disk) es el número de operaciones de clasificación que requieren al menos una escritura en disco.

• rows processed es el número de filas procesadas durante la operación.

**Nota:** db block gets indica lecturas del bloque actual de la base de datos. los consistent gets son lecturas de bloques que deben satisfacer un número de cambio de sistema (SCN) en particular, physical reads indican lecturas de bloques del disco. db block gets y consistent gets son las dos estadísticas que generalmente se monitorean. Estos deben ser bajos en comparación con el número de filas recuperadas. Las clasificaciones deben realizarse en la memoria en lugar de en el disco.

* **Uso de la vista V $ SQL\_PLAN**

• V$SQL\_PLAN proporciona una forma de examinar el plan de ejecución de los cursores que todavía están en la caché de la biblioteca.

• V$SQL\_PLAN es muy similar a PLAN\_TABLE:

- PLAN\_TABLE muestra un plan teórico que se puede usar si esta declaración se ejecutara.

- V$SQL\_PLAN contiene el plan real utilizado.

• Contiene el plan de ejecución de cada cursor en el caché de la biblioteca (incluido el hijo).

• Enlace a V $ SQL:

- ADDRESS, HASH\_VALUE y CHILD\_NUMBER

**Nota:** Otra vista útil es **V$SQL\_PLAN\_STATISTICS**, que proporciona las estadísticas de ejecución de cada operación en el plan de ejecución para cada cursor en caché. Además, la vista V$SQL\_PLAN\_STATISTICS\_ALL concatena información de V$SQL\_PLAN con estadísticas de ejecución de V$SQL\_PLAN\_STATISTICS y V$SQL\_WORKAREA.

**Nota:** desde Oracle Database 10g, SQL\_HASH\_VALUE en V$SESSION se ha complementado con SQL\_ID, que puede recuperar en muchas otras vistas de (V$) SQL\_HASH\_VALUE es un valor de 32 bits y no es lo suficientemente exclusivo para grandes repositorios de datos AWR. SQL\_ID es un valor hash de 64 bits, que es más exclusivo, cuyos 32 bits inferiores son SQL\_HASH\_VALUE. Normalmente se representa como una cadena de caracteres para hacerlo más manejable.

* **The V$SQL\_PLAN\_STATISTICS View**

• V$SQL\_PLAN\_STATISTICS proporciona ejecución real

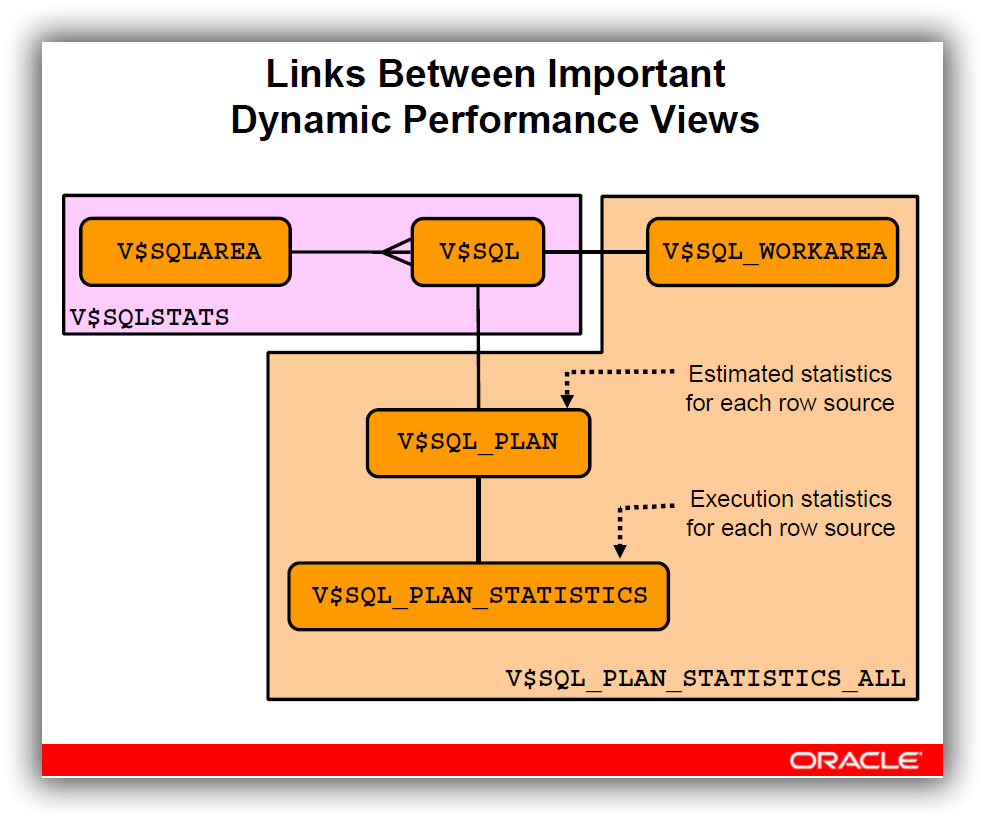
Estadísticas:

- STATISTICS\_LEVEL establecido en ALL

- La sugerencia GATHER\_PLAN\_STATISTICS

• V$SQL\_PLAN\_STATISTICS\_ALL permite comparaciones en paralelo de las estimaciones del optimizador con las estadísticas de ejecución reales.

* **Links Between Important Dynamic Performance Views**



V$SQLAREA muestra estadísticas en áreas SQL compartidas y contiene una fila por cadena SQL. Proporciona estadísticas sobre sentencias SQL que están en la memoria, analizadas y listas para su ejecución:

• SQL\_ID es el identificador de SQL del cursor principal en el caché de la biblioteca.

• VERSION\_COUNT es el número de cursores secundarios que están presentes en la memoria caché debajo de este elemento primario.

V$SQL enumera estadísticas sobre áreas SQL compartidas y contiene una fila para cada elemento secundario del texto SQL original ingresado:

• DIRECCIÓN representa la dirección del identificador para el padre para este cursor.

• HASH\_VALUE es el valor de la declaración principal en el caché de la biblioteca.

• SQL\_ID es el identificador de SQL del cursor principal en el caché de la biblioteca.

• PLAN\_HASH\_VALUE es una representación numérica del plan SQL para este cursor. Al comparar un PLAN\_HASH\_VALUE con otro, puede identificar fácilmente si los dos planes son iguales o no (en lugar de comparar los dos planes línea por línea).

• CHILD\_NUMBER es el número de este cursor secundario.

Las estadísticas que se muestran en V $ SQL normalmente se actualizan al final de la ejecución de la consulta. Sin embargo, para consultas de larga duración, se actualizan cada cinco segundos. Esto facilita ver el impacto de las declaraciones SQL de larga ejecución mientras aún están en progreso.

V$SQL\_PLAN contiene la información del plan de ejecución para cada cursor secundario cargado en el caché de la biblioteca. Las columnas ADDRESS, HASH\_VALUE y CHILD\_NUMBER se pueden usar para unirse con V $ SQL para agregar la información específica del cursor secundario.

V$SQL\_PLAN\_STATISTICS proporciona estadísticas de ejecución en el nivel de origen de fila para cada cursor secundario. Las columnas ADDRESS y HASH\_VALUE se pueden usar para unir con V$SQLAREA para ubicar el cursor principal. Las columnas ADDRESS, HASH\_VALUE y CHILD\_NUMBER se pueden usar para unir con V $ SQL para localizar el cursor secundario usando esta área.

V$SQL\_PLAN\_STATISTICS\_ALL contiene estadísticas de uso de memoria para orígenes de fila que usan memoria SQL (clasificación o combinación hash). Esta vista concatena información en V$SQL\_PLAN con estadísticas de ejecución de V$SQL\_PLAN\_STATISTICS y V$SQL\_WORKAREA.

V$SQL\_WORKAREA muestra información sobre las áreas de trabajo utilizadas por los cursores SQL. Cada instrucción SQL almacenada en el grupo compartido tiene uno o más cursores secundarios que se enumeran en la vista V$SQL. V$SQL\_WORKAREA enumera todas las áreas de trabajo que necesitan estos cursores secundarios.

V$SQL\_WORKAREA se puede unir con V$SQLAREA en (ADDRESS, HASH\_VALUE) y con V$SQL en (ADDRESS, HASH\_VALUE, CHILD\_NUMBER).

Puede usar esta vista para encontrar respuestas a las siguientes preguntas:

• ¿Cuáles son las 10 áreas de trabajo principales que requieren más área de caché?

• Para las áreas de trabajo asignadas en el modo AUTO, ¿qué porcentaje de áreas de trabajo se ejecutan con la memoria máxima?

V$SQLSTATS muestra estadísticas básicas de rendimiento para cursores SQL, y cada fila representa los datos para una combinación única de texto SQL y plan optimizador (es decir, combinación única de SQL\_ID y PLAN\_HASH\_VALUE). Las definiciones de columna para columnas en V$SQLSTATS son idénticas a las de las vistas V$SQL y V$SQLAREA. Sin embargo, la vista V$SQLSTATS difiere de V$SQL y V$SQLAREA en que es más rápida, más escalable y tiene una mayor retención de datos (las estadísticas aún pueden aparecer en esta vista, incluso después de que el cursor haya caducado La piscina compartida). Tenga en cuenta que V$SQLSTATS contiene un subconjunto de columnas que aparecen en V$SQL y V$ SQLAREA.

**Querying V$SQL\_PLAN**

Puede consultar V $ SQL\_PLAN utilizando la función DBMS\_XPLAN.DISPLAY\_CURSOR () para mostrar la declaración actual o la última ejecutada (como se muestra en el ejemplo). Puede pasar el valor de SQL\_ID para la declaración como un parámetro para obtener el plan de ejecución para una declaración dada. SQL\_ID es el SQL\_ID de la instrucción SQL en la caché del cursor. Puede recuperar el valor apropiado consultando la columna SQL\_ID en V $ SQL o V $ SQLAREA. Alternativamente, puede seleccionar la columna PREV\_SQL\_ID para una sesión específica de V $ SESSION. El valor predeterminado de este parámetro es nulo, en cuyo caso se muestra el plan del último cursor ejecutado por la sesión. Para obtener SQL\_ID, ejecute la siguiente consulta:

SELECT e.last\_name, d.department\_name

FROM hr.employees e, hr.departments d

WHERE e.department\_id =d.department\_id;

SELECT SQL\_ID, SQL\_TEXT FROM V$SQL

WHERE SQL\_TEXT LIKE '%SELECT e.last\_name,%' ;

13saxr0mmz1s3 select SQL\_id, sql\_text from v$SQL …

47ju6102uvq5q SELECT e.last\_name, d.department\_name …

CHILD\_NUMBER es el número secundario del cursor para mostrar. Si no se proporciona, se muestra el plan de ejecución de todos los cursores que coinciden con el parámetro SQL\_ID proporcionado. CHILD\_NUMBER solo se puede especificar si se especifica SQL\_ID.

El parámetro FORMAT controla el nivel de detalle del plan. Además de los valores estándar (BÁSICO, TÍPICO, SERIAL, TODO y AVANZADO), hay valores adicionales admitidos para mostrar estadísticas de tiempo de ejecución para el cursor:

• IOSTATS: suponiendo que las estadísticas básicas del plan se recopilan cuando se ejecutan sentencias SQL (ya sea utilizando la sugerencia GATHER\_PLAN\_STATISTICS o estableciendo el parámetro stats\_level en ALL), este formato muestra las estadísticas de E / S para ALL (o solo para ÚLTIMO) ejecuciones del cursor.

• MEMSTATS: suponiendo que la administración de memoria del Área global del programa (PGA) esté habilitada (es decir, el parámetro pga\_aggregate\_target está configurado en un valor distinto de cero), este formato permite mostrar estadísticas de administración de memoria (por ejemplo, modo de ejecución del operador, cómo se usó mucha memoria, número de bytes derramados en el disco, etc.) Estas estadísticas solo se aplican a las operaciones que requieren mucha memoria, como las combinaciones hash, la ordenación o algunos operadores de mapas de bits.

• ALLSTATS: un atajo para 'IOSTATS MEMSTATS'

• ÚLTIMO: Por defecto, las estadísticas del plan se muestran para todas las ejecuciones del cursor. La palabra clave LAST puede especificarse para ver solo las estadísticas de la última ejecución.

* **Automatic Workload Repository (AWR)**

• Recopila, procesa y mantiene estadísticas de rendimiento para fines de detección de problemas y autoajuste

• Las estadísticas incluyen:

- Estadísticas de objetos

- Estadísticas del modelo de tiempo

- Algunas estadísticas del sistema y de la sesión

- Estadísticas de historial de sesión activa (ASH)

• Genera automáticamente instantáneas de los datos de rendimiento.

**Nota:** Al usar paquetes PL / SQL, como DBMS\_WORKLOAD\_REPOSITORY u Oracle Enterprise Manager, puede administrar la frecuencia y el período de retención de SQL que se almacena en el AWR. **DBA\_HIST\_SNAPSHOT, DBA\_HIST\_WR\_CONTROL**

• Creating snapshots:

SQL> exec DBMS\_WORKLOAD\_REPOSITORY.CREATE\_SNAPSHOT ('ALL');

• Dropping snapshots:

SQL> exec DBMS\_WORKLOAD\_REPOSITORY.DROP\_SNAPSHOT\_RANGE –

(low\_snap\_id => 22, high\_snap\_id => 32, dbid => 3310949047);

• Managing snapshot settings:

SQL> exec DBMS\_WORKLOAD\_REPOSITORY.MODIFY\_SNAPSHOT\_SETTINGS –

(retention => 43200, interval => 30, dbid => 3310949047);

**Important AWR Views**

• V$ACTIVE\_SESSION\_HISTORY: Esta vista muestra la actividad activa de la sesión de la base de datos, muestreada una vez por segundo.

• V$ metric views: Proporcionar datos métricos para rastrear el rendimiento del sistema. Las vistas métricas se organizan en varios grupos, como evento, clase de evento, sistema, sesión, servicio, archivo y métricas de espacio de tabla. Estos grupos se identifican en la vista V $ METRICGROUP.

• DBA\_HIST views:

- DBA\_HIST\_ACTIVE\_SESS\_HISTORY muestra el historial de los contenidos del historial de sesiones activas en memoria muestreadas para la actividad reciente del sistema.

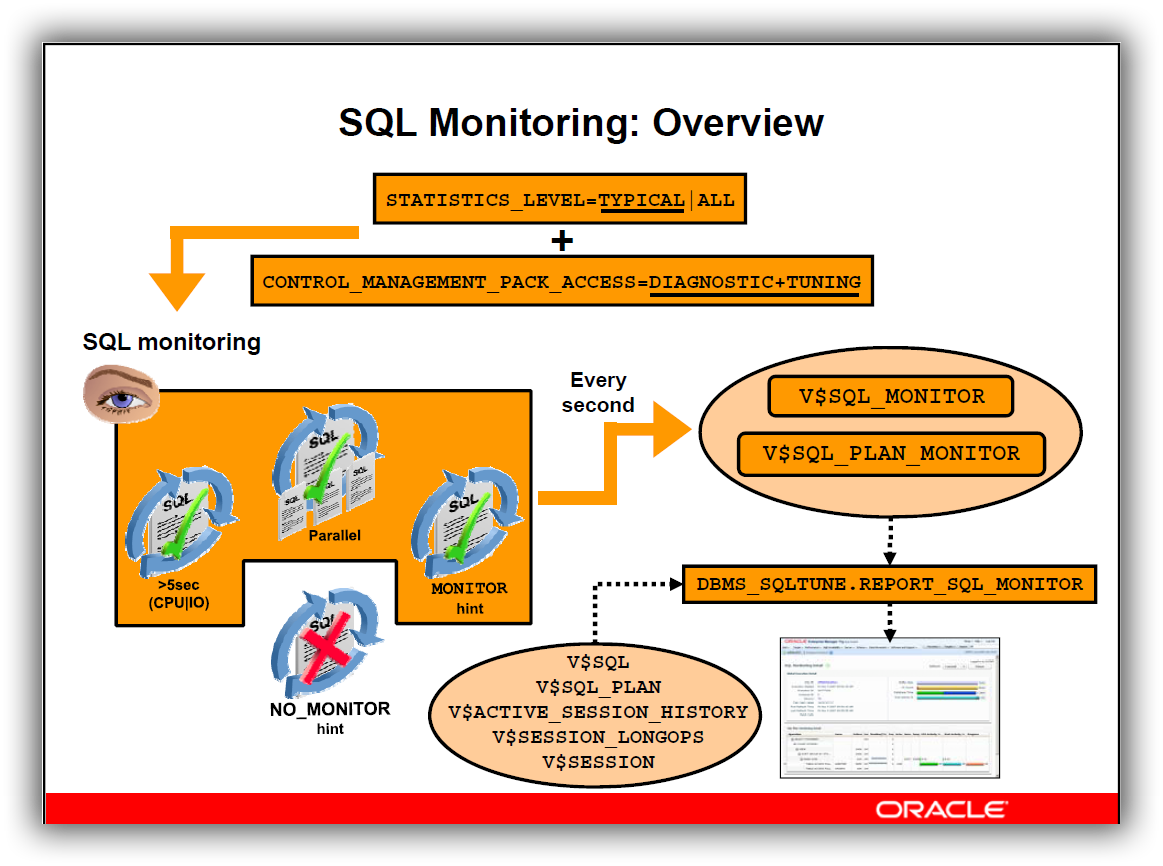
- DBA\_HIST\_BASELINE muestra información sobre las líneas de base capturadas en el sistema.

- DBA\_HIST\_DATABASE\_INSTANCE muestra información sobre el entorno de la base de datos.

- DBA\_HIST\_SNAPSHOT muestra información sobre instantáneas en el sistema.

- DBA\_HIST\_SQL\_PLAN muestra planes de ejecución de SQL.

- DBA\_HIST\_WR\_CONTROL muestra la configuración para controlar AWR.



La función de supervisión de SQL está habilitada de manera predeterminada cuando el parámetro de inicialización STATISTICS\_LEVEL está establecido en ALL o TYPICAL (el valor predeterminado).

Además, el parámetro CONTROL\_MANAGEMENT\_PACK\_ACCESS debe establecerse en DIAGNOSTIC + TUNING (el valor predeterminado) porque la supervisión de SQL es una característica del paquete de ajuste de la base de datos Oracle.

De manera predeterminada, la supervisión de SQL se inicia automáticamente cuando una instrucción SQL se ejecuta en paralelo, o cuando ha consumido al menos cinco segundos del tiempo de CPU o E / S en una sola ejecución.

Como se mencionó, la supervisión de SQL está activa por defecto. Sin embargo, hay dos sugerencias de nivel de instrucción disponibles para forzar o evitar que se supervise una instrucción SQL. Para forzar la supervisión de SQL, use la sugerencia MONITOR. Para evitar que se supervise la instrucción SQL insinuada, use la sugerencia NO\_MONITOR.

Puede supervisar las estadísticas para la ejecución de instrucciones SQL utilizando las vistas V$SQL\_MONITOR y V$SQL\_PLAN\_MONITOR.

Después de iniciar la supervisión, se agrega una entrada a la vista de rendimiento dinámico V$SQL\_MONITOR. Esta entrada rastrea las métricas de rendimiento clave recopiladas para la ejecución, incluido el tiempo transcurrido, el tiempo de CPU, el número de lecturas y escrituras, el tiempo de espera de E / S y otros tiempos de espera. Estas estadísticas se actualizan casi en tiempo real a medida que se ejecuta la instrucción, generalmente una vez por segundo.

Una vez que finaliza la ejecución, la información de monitoreo no se elimina de inmediato, sino que se mantiene en la vista V$SQL\_MONITOR durante al menos un minuto. La entrada finalmente se elimina para que su espacio pueda recuperarse a medida que se supervisan las nuevas declaraciones.

Las vistas V$SQL\_MONITOR y V$SQL\_PLAN\_MONITOR se pueden usar junto con las siguientes vistas para obtener información adicional sobre la ejecución que se supervisa:

V$SQL, V$SQL\_PLAN, V$ACTIVE\_SESSION\_HISTORY, V$SESSION\_LONGOPS y V$SESSION En su lugar, puede usar el informe de supervisión de SQL para ver los datos de supervisión de SQL.

El informe de supervisión de SQL también está disponible en una versión GUI a través de Enterprise Manager y SQL Developer.

**Nota:** No se muestra en este informe en particular, las columnas Memoria y Temp indican la cantidad de memoria y espacio temporal consumido por la operación correspondiente del plan de ejecución.

* **Looking Beyond Execution Plans**

Mirando más allá de los planes de ejecución

• Un plan de ejecución solo no puede decirle si un plan es bueno o no.

• Puede necesitar pruebas y ajustes adicionales:

- Asesor de ajuste SQL

- Asesor de acceso SQL

- Analizador de rendimiento SQL

- Monitoreo SQL

- Rastreo

**Application Tracing (Rastreo de aplicaciones)**

* **Seguimiento de aplicaciones de extremo a extremo**

***End-to-End Application Tracing***

• Simplifica el proceso de diagnóstico de problemas de rendimiento en entornos de varios niveles al permitir que las cargas de trabajo de las aplicaciones sean vistas por:

• Client identifier: especifica un usuario final basado en la ID de inicio de sesión, como HR

• Service: especifica un grupo de aplicaciones con atributos comunes, umbrales de nivel de servicio y prioridades; o una sola aplicación

• Module: especifica un bloque funcional dentro de una aplicación

• Action: especifica una acción, como una operación INSERT o UPDATE, en un módulo

• Session: especifica una sesión basada en un identificador de sesión de base de datos (SID) dado

• End-to-end application tracing tools:

– Enterprise Manager

– DBMS\_APPICATION\_INFO, DBMS\_SERVICE,DBMS\_MONITOR, DBMS\_SESSION

– SQL Trace and trcsess utility

– tkprof

**Nota:** La principal diferencia entre un rastreo y un volcado es que un rastreo es una salida continua, como cuando se activa el rastreo de SQL, y un volcado es una salida única en respuesta a un evento, como un incidente. Además, un volcado de núcleo es un volcado de memoria binaria que es específico del puerto.

***¿Qué es un servicio?***

• Es un medio de agrupar sesiones que realizan el mismo tipo de trabajo.

• Proporciona una imagen de sistema único en lugar de una imagen de instancias múltiples

• Forma parte de las tareas de administración regulares que proporcionan una asignación dinámica de servicio a instancia

• Es la base para una alta disponibilidad de conexiones.

• Proporciona una dimensión de ajuste de rendimiento

• Es un identificador para capturar información de rastreo.

Los servicios se pueden configurar, administrar, habilitar, deshabilitar y medir como una sola entidad utilizando interfaces estándar, Enterprise Manager y SRVCTL.

Los servicios brindan disponibilidad. Después de las interrupciones, un servicio se recupera rápida y automáticamente en las instancias sobrevivientes.

***Using Services with Client Applications***

ERP=(DESCRIPTION= (ADDRESS=(PROTOCOL=TCP)(HOST=mynode)(PORT=1521))(CONNECT\_DATA=(SERVICE\_NAME=ERP)))

url="jdbc:oracle:oci:@ERP"

url="jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=mynode)(PORT=1521))(CONNECT\_DATA=(SERVICE\_NAME=ERP)))"

Cualquier cliente que se conecte al servidor de bases de datos utiliza un nombre de servicio. Ese nombre de servicio se aplica automáticamente a las acciones del cliente. Las aplicaciones se pueden agrupar por servicios simplemente usando un nombre de servicio diferente para cada aplicación para conectarse.

Las aplicaciones y los grupos de conexiones de nivel medio seleccionan un servicio mediante el descriptor de conexión de Substrato de red transparente (TNS).

El servicio seleccionado debe coincidir con el servicio que se ha creado.

El primer ejemplo en la diapositiva muestra el descriptor de conexión TNS que se puede utilizar para acceder al servicio ERP.

El segundo ejemplo muestra la descripción de la conexión de Java Database Connectivity (JDBC) utilizando el descriptor de conexión TNS previamente definido.

El tercer ejemplo muestra la descripción de la conexión JDBC delgada utilizando el mismo descriptor de conexión TNS.

* **Tracing Services**

• Applications using services can be further qualified by:

– MODULE

– ACTION

– CLIENT\_IDENTIFIER

• Set using the following PL/SQL packages:

– DBMS\_APPLICATION\_INFO

– DBMS\_SESSION

• Tracing can be done at all levels:

– CLIENT\_IDENTIFIER

– SESSION\_ID

– SERVICE\_NAMES

– MODULE

– ACTION

– Combination of SERVICE\_NAME, MODULE, ACTION

**Nota:** DBA\_ENABLED\_TRACES **muestra** información sobre los rastreos habilitados.

* **Service Tracing: Example**

• Trace on service, module, and action:

exec DBMS\_MONITOR.SERV\_MOD\_ACT\_TRACE\_ENABLE('AP');

exec BMS\_MONITOR.SERV\_MOD\_ACT\_TRACE\_ENABLE('AP','PAYMENTS','QUERY\_DELINQUENT');

• Trace a particular client identifier:

exec DBMS\_MONITOR.CLIENT\_ID\_TRACE\_ENABLE(client\_id=>'C4', waits => TRUE, binds => FALSE);

En el primer cuadro de código, se rastrean todas las sesiones que inician sesión con el servicio AP. Se crea un archivo de rastreo para cada sesión que utiliza el servicio, independientemente del módulo y la acción. También puede habilitar el rastreo para tareas específicas dentro de un servicio. Esto se ilustra en el segundo ejemplo, donde se rastrean todas las sesiones del servicio AP que ejecutan la acción QUERY\_DELINQUENT dentro del módulo PAGOS.

El rastreo por servicio, módulo y acción le permite concentrar sus esfuerzos de ajuste en SQL específico, en lugar de examinar archivos de rastreo con SQL desde diferentes programas. Solo las declaraciones SQL que se identifican con este MÓDULO y ACCIÓN se registran en el archivo de seguimiento. Con esta característica, se pueden identificar eventos de espera relevantes para una acción específica.

También puede comenzar a rastrear un identificador de cliente particular como se muestra en el tercer ejemplo. En este ejemplo, C4 es el identificador del cliente para el que se debe habilitar el seguimiento de SQL. El argumento VERDADERO especifica que la información de espera está presente en el archivo de rastreo. El argumento FALSE especifica que la información de enlace no está presente en el archivo de rastreo.

Aunque no se muestra en la diapositiva, puede utilizar el procedimiento CLIENT\_ID\_TRACE\_DISABLE para deshabilitar el seguimiento global de la base de datos para un identificador de cliente determinado. Para deshabilitar el seguimiento, para el ejemplo anterior, ejecute el siguiente comando:

EJECUTAR DBMS\_MONITOR.CLIENT\_ID\_TRACE\_DISABLE (client\_id => 'C4');

**Nota:** CLIENT\_IDENTIFIER se puede configurar mediante el procedimiento BMS\_SESSION.SET\_IDENTIFIER.

**Session Level Tracing: Example**

• For all sessions in the database:

EXEC dbms\_monitor.DATABASE\_TRACE\_ENABLE(TRUE,TRUE);

EXEC dbms\_monitor.DATABASE\_TRACE\_DISABLE();

• For a particular session:

EXEC dbms\_monitor.SESSION\_TRACE\_ENABLE(session\_id=>27, serial\_num=>60, waits=>TRUE, binds=>FALSE);

EXEC dbms\_monitor.SESSION\_TRACE\_DISABLE(session\_id=>27, serial\_num=>60);

Puede usar el rastreo para depurar problemas de rendimiento. Los procedimientos de habilitación de rastreo se han implementado como parte del paquete DBMS\_MONITOR. Estos procedimientos permiten el rastreo global de una base de datos.

Puede usar el procedimiento DATABASE\_TRACE\_ENABLE para habilitar el seguimiento de SQL de nivel de sesión en toda la instancia. El procedimiento tiene los siguientes parámetros:

• WAITS: especifica si se debe rastrear la información de espera

• BINDS: especifica si se debe rastrear la información de enlace

• INSTANCE\_NAME: especifica la instancia para la cual se habilitará el rastreo. Omitiendo INSTANCE\_NAME significa que el seguimiento a nivel de sesión está habilitado para toda la base de datos.

Use el procedimiento DATABASE\_TRACE\_DISABLE para deshabilitar el seguimiento de SQL para toda la base de datos o una instancia específica.

De manera similar, puede usar el procedimiento SESSION\_TRACE\_ENABLE para habilitar el seguimiento de un identificador de sesión de base de datos dado en la instancia local. La información SID y SERIAL # puede ser encontrado de V$SESSION.

Utilice el procedimiento SESSION\_TRACE\_DISABLE para deshabilitar el rastreo para un identificador de sesión de base de datos y un número de serie determinados.

Nota: el seguimiento de SQL implica cierta sobrecarga, por lo que generalmente no desea habilitar el seguimiento de SQL en el nivel de instancia.

**Trace Your Own Session**

• Enabling trace:

EXEC DBMS\_SESSION.SESSION\_TRACE\_DISABLE();

• Disabling trace:

EXEC DBMS\_SESSION.SESSION\_TRACE\_ENABLE(waits =>TRUE, binds => FALSE);

• Easily identifying your trace files:

alter session set

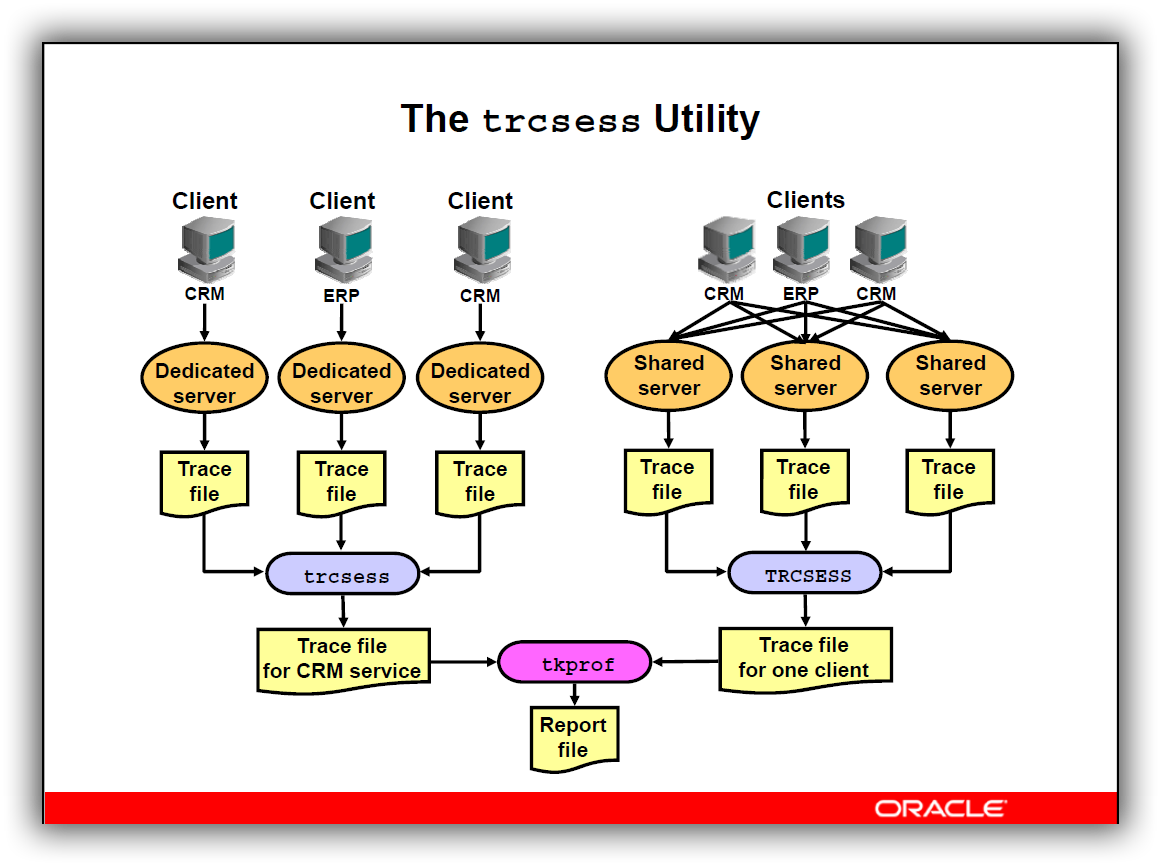
tracefile\_identifier='mytraceid';

Aunque el paquete DBMS\_MONITOR solo puede ser invocado por un usuario con el rol DBA, cualquier usuario puede habilitar el seguimiento de SQL para su propia sesión utilizando el paquete DBMS\_SESSION. El procedimiento SESSION\_TRACE\_ENABLE puede ser invocado por cualquier usuario para habilitar el seguimiento de SQL a nivel de sesión para su propia sesión. Un ejemplo se muestra en la diapositiva.

Luego puede usar el procedimiento DBMS\_SESSION.SESSION\_TRACE\_DISABLE para detener el volcado a su archivo de rastreo.

El parámetro de inicialización TRACEFILE\_IDENTIFIER especifica un identificador personalizado que se convierte en parte del nombre del archivo de rastreo de Oracle. Puede usar dicho identificador personalizado para identificar un archivo de seguimiento simplemente desde su nombre y sin abrirlo ni ver su contenido. Cada vez que este parámetro se modifique dinámicamente en el nivel de sesión, el siguiente volcado de rastreo escrito en un archivo de rastreo tendrá el nuevo valor del parámetro incrustado en su nombre. Este parámetro solo se puede usar para cambiar el nombre del archivo de rastreo del proceso en primer plano; los procesos en segundo plano continúan teniendo sus archivos de rastreo nombrados en el formato normal. Para procesos en primer plano, la columna TRACEID de la vista V $ PROCESS contiene el valor actual de este parámetro. Cuando se establece este valor de parámetro, el nombre del archivo de rastreo tiene el siguiente formato: sid\_ora\_pid\_traceid.trc.

* **The trcsess Utility**



La utilidad trcsess consolida la salida de rastreo de los archivos de rastreo seleccionados sobre la base de varios criterios: ID de sesión, identificador de cliente, nombre de servicio, nombre de acción y nombre de módulo. Después de que trcsess combina la información de rastreo en un solo archivo de salida, tkprof puede procesar el archivo de salida.

Cuando se utiliza el procedimiento DBMS\_MONITOR.SERV\_MOD\_ACT\_TRACE\_ENABLE, la información de rastreo está presente en varios archivos de rastreo y debe usar la herramienta trcsess para recopilarla en un solo archivo.

La utilidad trcsess es útil para consolidar el seguimiento de una sesión o servicio en particular con fines de rendimiento o depuración.

El seguimiento de una sesión específica generalmente no es un problema en el modelo de servidor dedicado porque un solo proceso dedicado sirve una sesión durante su vida útil. Toda la información de rastreo para la sesión se puede ver desde el archivo de rastreo que pertenece al servidor dedicado que lo sirve.

Sin embargo, el seguimiento de un servicio puede convertirse en una tarea compleja incluso en el modelo de servidor dedicado.

Además, en una configuración de servidor compartido, una sesión de usuario es atendida por diferentes procesos de vez en cuando. La traza perteneciente a la sesión del usuario está dispersa en diferentes archivos de traza que pertenecen a diferentes procesos. Esto hace que sea difícil obtener una imagen completa del ciclo de vida de una sesión.

trcsess [output=output\_file\_name]

[session=session\_id]

[clientid=client\_identifier]

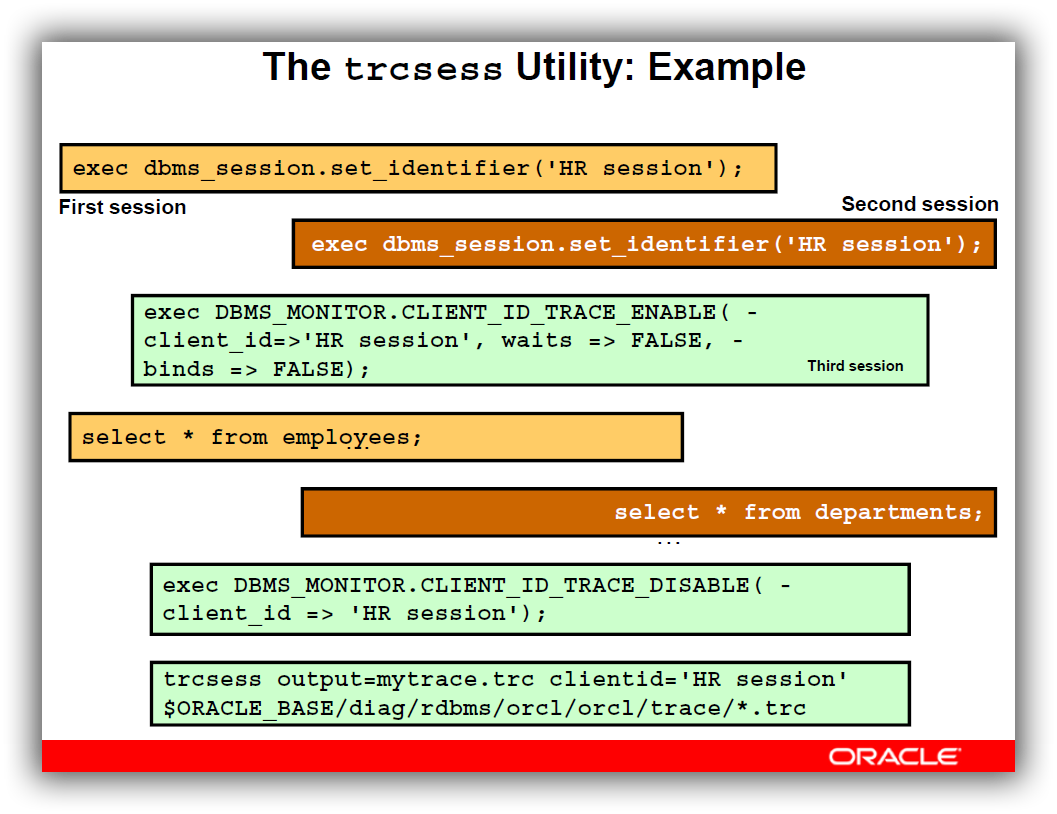
[service=service\_name]

[action=action\_name]

[module=module\_name]

[<trace file names>]

**Nota:** Debe especificarse una de las opciones de sesión, cliente, servicio, acción o módulo. Si se especifica más de una opción, los archivos de rastreo, que satisfacen todos los criterios especificados, se consolidan en el archivo de salida.



* **Contenido del archivo de rastreo SQL**

Como ya se vio, un archivo de rastreo SQL proporciona información de rendimiento en sentencias SQL individuales. Genera las siguientes estadísticas para cada declaración:

• Analizar, ejecutar y buscar recuentos

• CPU y tiempos transcurridos

• Lecturas físicas y lecturas lógicas.

• Número de filas procesadas

• Falta en el caché de la biblioteca

• Nombre de usuario bajo el cual ocurrió cada análisis

• Cada commit y rollback

• Espere los datos del evento para cada instrucción SQL y un resumen para cada archivo de rastreo Si el cursor para la instrucción SQL está cerrado, SQL Trace también proporciona información de origen de fila que incluye:

• Operaciones de fila que muestran el plan de ejecución real de cada instrucción SQL

• Número de filas, número de lecturas consistentes, número de lecturas físicas, número de escrituras físicas y tiempo transcurrido para cada operación. Esto solo es posible cuando el parámetro de inicialización STATISTICS\_LEVEL está establecido en ALL.

**Nota:** El uso de la función de seguimiento de SQL puede tener un impacto severo en el rendimiento y puede resultar en un aumento de la sobrecarga del sistema, un uso excesivo de la CPU y un espacio en disco inadecuado.

* **SQL Trace File Contents: Example**

Oracle Database puede generar múltiples tipos de archivos de rastreo. El que se menciona en esta lección generalmente se llama un archivo de rastreo SQL. La diapositiva muestra una salida de muestra del archivo de rastreo SQL mytrace.trc generado por el ejemplo anterior.

En este tipo de archivo de rastreo, puede encontrar (para cada declaración que se rastreó) la declaración en sí, con algunos detalles correspondientes del cursor. Puede ver detalles estadísticos para cada fase de la ejecución de la declaración: PARSE, EXEC y FETCH. Como puede ver, puede tener múltiples FETCH para un EXEC dependiendo del número de filas devueltas por su consulta.

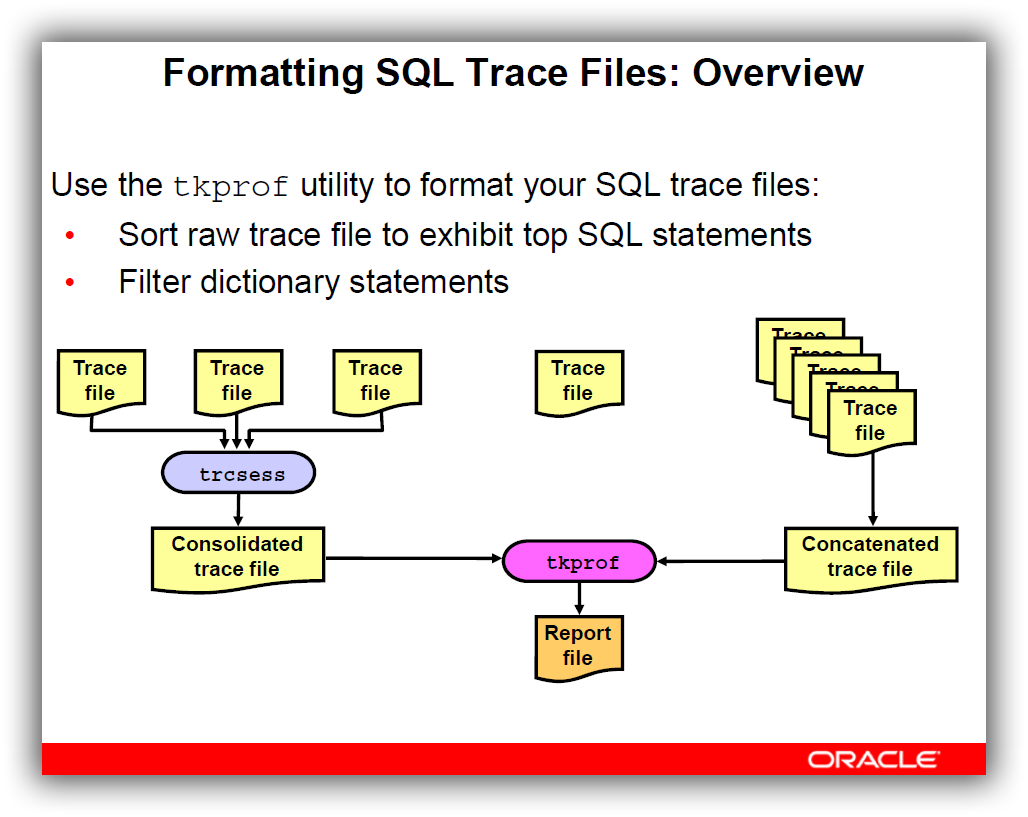
La última parte del seguimiento es el plan de ejecución con algunas estadísticas acumuladas para cada origen de fila.

Dependiendo de la forma en que habilitó el rastreo, también puede obtener información sobre eventos de espera y variables de enlace en los archivos de rastreo generados.

Generalmente, no intenta interpretar el archivo de rastreo en sí. Esto se debe a que no tiene una idea general de lo que hicieron sus sesiones. Por ejemplo, una sesión podría haber ejecutado la misma declaración varias veces en diferentes momentos. Las trazas correspondientes se dispersan por todo el archivo de trazas, lo que hace que sea difícil de encontrar.

En su lugar, utiliza otra herramienta, como tkprof, para interpretar el contenido de la información de rastreo sin procesar.

* **Formatting SQL Trace Files: Overview**



La utilidad tkprof analiza los archivos de rastreo SQL para producir una salida más legible. Recuerde que toda la información en tkprof está disponible desde el archivo de rastreo sin procesar. Hay una gran cantidad de opciones de clasificación que puede invocar con tkprof en el símbolo del sistema. Un punto de partida útil es la opción de ordenación fchela, que ordena la salida por recuperación de tiempo transcurrido. El archivo resultante contiene la instrucción SQL más lenta al comienzo del archivo. Otro parámetro útil es SYS=NO. Esto se puede utilizar para evitar que se ejecuten sentencias SQL mientras se muestra el usuario SYS. Esto puede hacer que el archivo de salida sea mucho más corto y fácil de administrar.

Después de que se hayan generado varios archivos de rastreo SQL, puede realizar cualquiera de las siguientes acciones:

• Ejecute tkprof en cada archivo de rastreo individual, produciendo una cantidad de archivos de salida formateados, uno para cada sesión.

• Concatene los archivos de rastreo y luego ejecute tkprof en el resultado para producir un archivo de salida formateado para toda la instancia.

• Ejecute la utilidad de línea de comandos trcsess para consolidar la información de rastreo de varios archivos de rastreo, luego ejecute tkprof en el resultado.

tkprof no informa los COMPROMISOS y ROLLBACK que se registran en el archivo de seguimiento.

**Nota:** Establezca el parámetro TIMED\_STATISTICS en TRUE cuando rastree sesiones porque no se pueden hacer comparaciones basadas en el tiempo sin esto. VERDADERO es el valor predeterminado con Oracle Database 11g.

* **Invoking the tkprof Utility**

tkprof inputfile outputfile [waits=yes|no]

[sort=option]

[print=n]

[aggregate=yes|no]

[insert=sqlscriptfile]

[sys=yes|no]

[table=schema.table]

[explain=user/password]

[record=statementfile]

[width=n]

Cuando ingresa el comando tkprof sin ningún argumento, genera un mensaje de uso junto con una descripción de todas las opciones de tkprof. Los diversos argumentos se muestran en la diapositiva:

• inputfile: especifica el archivo de entrada de rastreo de SQL

• outputfile: especifica el archivo en el que tkprof escribe su salida formateada

• waits: especifica si se registrará el resumen de los eventos de espera encontrados en el archivo de seguimiento. Los valores son SÍ o NO. El valor predeterminado es SÍ.

• sorts: ordena las sentencias SQL rastreadas en el orden descendente de la opción de ordenación especificada antes de enumerarlas en el archivo de salida. Si se especifica más de una opción, la salida se ordena en orden descendente por la suma de los valores especificados en las opciones de clasificación. Si omite este parámetro, tkprof enumera las declaraciones en el archivo de salida en el orden de primer uso.

• print: enumera solo las primeras declaraciones SQL ordenadas de enteros del archivo de salida. Si omite este parámetro, tkprof enumera todas las instrucciones SQL rastreadas. Este parámetro no afecta el script SQL opcional. El script SQL siempre genera datos de inserción para todas las declaraciones SQL rastreadas.

• aggregate: si se establece en NO, tkprof no agrega múltiples usuarios del mismo texto SQL.

• insert: crea un script SQL para almacenar las estadísticas del archivo de rastreo en la base de datos. tkprof crea este script con el nombre que especifique para sqlscriptfile. Este script crea un tabla e inserta una fila de estadísticas para cada instrucción SQL rastreada en la tabla.

• sys: activa y desactiva la lista de sentencias SQL emitidas por el usuario SYS, o sentencias SQL recursivas, en el archivo de salida. El valor predeterminado de YES hace que tkprof enumere estas declaraciones. El valor de NO hace que tkprof los omita. Este parámetro no afecta el script SQL opcional. El script SQL siempre inserta estadísticas para todas las declaraciones SQL rastreadas, incluidas las declaraciones SQL recursivas.

• table: especifica el esquema y el nombre de la tabla en la que tkprof coloca temporalmente los planes de ejecución antes de escribirlos en el archivo de salida. Si la tabla especificada ya existe, tkprof elimina todas las filas de la tabla, la usa para la instrucción EXPLAIN PLAN (que escribe más filas en la tabla) y luego las elimina. Si esta tabla no existe, tkprof la crea, la usa y luego la descarta. El usuario especificado debe poder emitir instrucciones INSERT, SELECT y DELETE en la tabla. Si la tabla aún no existe, el usuario también debe poder emitir las instrucciones CREATE TABLE y DROP TABLE. Esta opción permite que varias personas ejecuten tkprof simultáneamente con el mismo usuario en el valor EXPLAIN. Estas personas pueden especificar diferentes valores de TABLE y evitar interferir destructivamente con el procesamiento de los demás en la tabla de plan temporal. Si usa el parámetro EXPLAIN sin el parámetro TABLE, tkprof usa la tabla PROF $ PLAN\_TABLE en el esquema del usuario especificado por el parámetro EXPLAIN. Si usa el parámetro TABLE sin el parámetro EXPLAIN, tkprof ignora el parámetro TABLE. Si no existe una tabla de plan, tkprof crea la tabla PROF $ PLAN\_TABLE y luego la descarta al final.

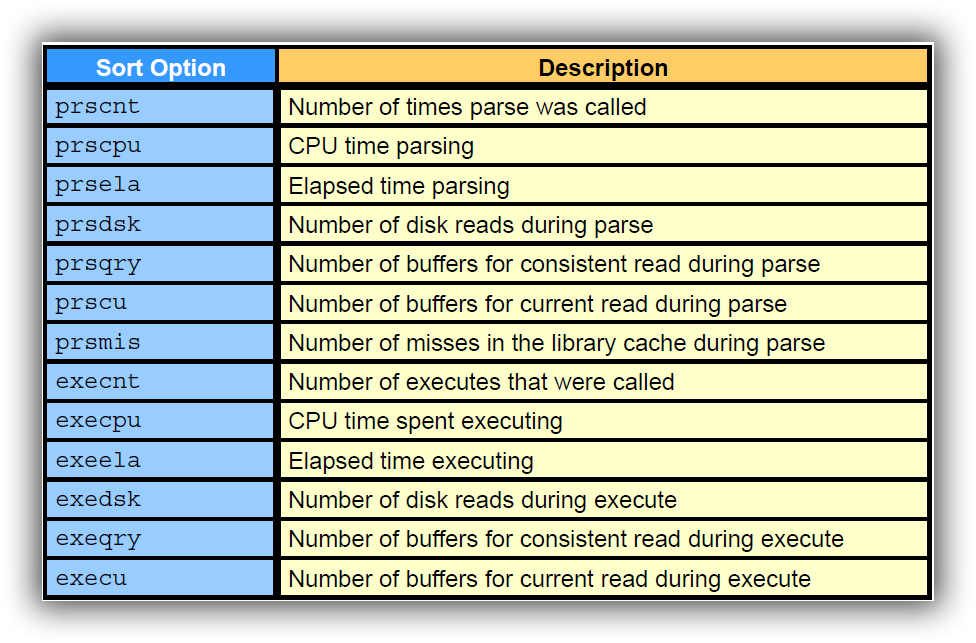
• explain: determina el plan de ejecución para cada instrucción SQL en el archivo de rastreo y escribe estos planes de ejecución en el archivo de salida. tkprof determina los planes de ejecución emitiendo la instrucción EXPLAIN PLAN después de conectarse al sistema con el usuario y la contraseña especificados en este parámetro. El usuario especificado debe tener privilegios del sistema CREATE SESSION. tkprof tarda más en procesar un archivo de rastreo grande si se usa la opción EXPLAIN.

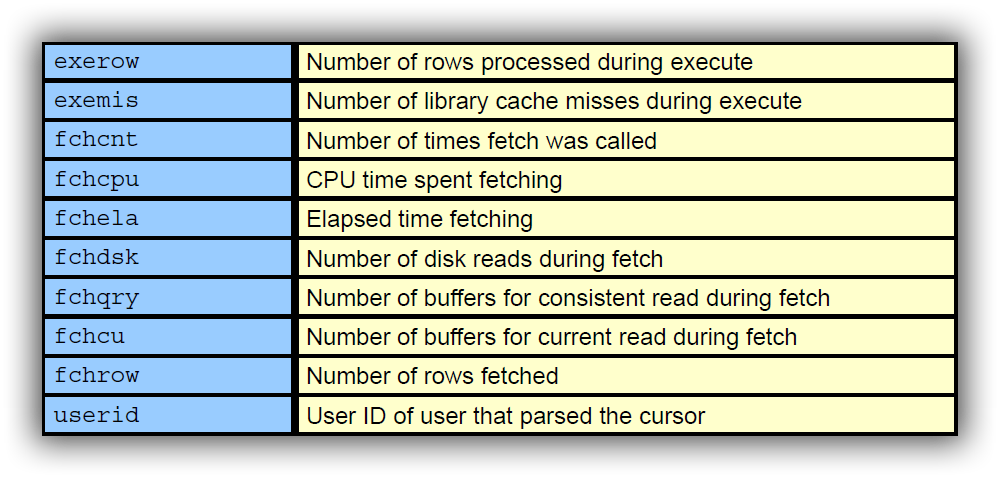
• record: crea un script SQL con el archivo de declaración del nombre de archivo especificado con todas las declaraciones SQL no recursivas en el archivo de rastreo. Esto se puede usar para reproducir los eventos del usuario desde el archivo de rastreo.

• width: un entero que controla el ancho de la línea de salida de algunos resultados de tkprof, como el plan de explicación. Este parámetro es útil para el procesamiento posterior de la salida tkprof.

Los archivos de entrada y salida son los únicos argumentos requeridos.

* **tkprof Sorting Options**





* **Salida del comando tkprof**

El archivo de salida tkprof enumera las estadísticas para una declaración SQL por el paso de procesamiento SQL.

El paso para cada fila que contiene estadísticas se identifica por el valor de la columna de llamada.

**PARSE** Este paso traduce la instrucción SQL en un plan de ejecución e incluye comprobaciones de la autorización de seguridad adecuada y comprobaciones de la existencia de tablas, columnas y otros objetos referenciados.

**EXECUTE** Este paso es la ejecución real de la declaración por parte del servidor Oracle. Para las instrucciones INSERT, UPDATE y DELETE, este paso modifica los datos (incluidos los tipos cuando sea necesario). Para las instrucciones SELECT, este paso identifica las filas seleccionadas.

**FETCH** Este paso recupera las filas devueltas por una consulta y las ordena cuando es necesario.

Las recuperaciones se realizan solo para las instrucciones SELECT.

**Nota:** El valor PARSE incluye tanto los análisis duros como los blandos. Un análisis estricto se refiere al desarrollo del plan de ejecución (incluida la optimización); posteriormente se almacena en el caché de la biblioteca Un análisis suave significa que se envía una instrucción SQL para analizarla a la base de datos, pero la base de datos la encuentra en la caché de la biblioteca y solo necesita verificar cosas, como los derechos de acceso. Los análisis duros pueden ser costosos, particularmente debido a la optimización. Un análisis suave es mayormente costoso en términos de actividad de caché de la biblioteca 25

La salida de muestra es la siguiente:

**call count cpu elapsed disk query current rows**

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

**Parse** 1 0.03 0.06 0 0 0 0

**Execute** 1 0.06 0.30 1 3 0 0

**Fetch** 2 0.00 0.46 0 0 0 1

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

**total** 4 0.09 0.83 1 3 0 1

Junto a la columna CALL, tkprof muestra las siguientes estadísticas para cada declaración:

• **Count**: número de veces que se analizó, ejecutó o extrajo una instrucción (verifique en esta columna los valores mayores que 1 antes de interpretar las estadísticas en las otras columnas. A menos que se use la opción AGREGAR = NO, tkprof agrega ejecuciones de instrucciones idénticas en una tabla de resumen).

• **CPU**: tiempo total de CPU en segundos para todas las llamadas de análisis, ejecución o recuperación

• **Elapsed**: tiempo total transcurrido en segundos para todas las llamadas de análisis, ejecución o recuperación

• **Disk**: número total de bloques de datos leídos físicamente de los archivos de datos en el disco para todas las llamadas de análisis, ejecución o recuperación

• **Query**: número total de buffers recuperados en modo consistente para todas las llamadas de análisis, ejecución o recuperación (los buffers generalmente se recuperan en modo consistente para consultas).

• **Current**: número total de buffers recuperados en modo actual (los buffers generalmente se recuperan en modo actual para declaraciones de lenguaje de manipulación de datos. Sin embargo, los bloques de encabezado de segmento siempre se recuperan en modo actual).

• **Rows**: número total de filas procesadas por la instrucción SQL (este total no incluye filas procesadas por subconsultas de la instrucción SQL. Para las instrucciones SELECT, el número de filas devueltas aparece para el paso de búsqueda. Para ACTUALIZAR, ELIMINAR e INSERTAR , el número de filas procesadas aparece para el paso de ejecución).

**Nota:**

• DISK es equivalente a lecturas físicas de v$sysstat o AUTOTRACE.

• QUERY es equivalente a obtener consistentemente de v$sysstat o AUTOTRACE.

• CORRIENTE es equivalente al bloque db que se obtiene de v$sysstat o AUTOTRACE.

* **Output of the tkprof Command**

The tkprof output also includes the following:

• Recursive SQL statements

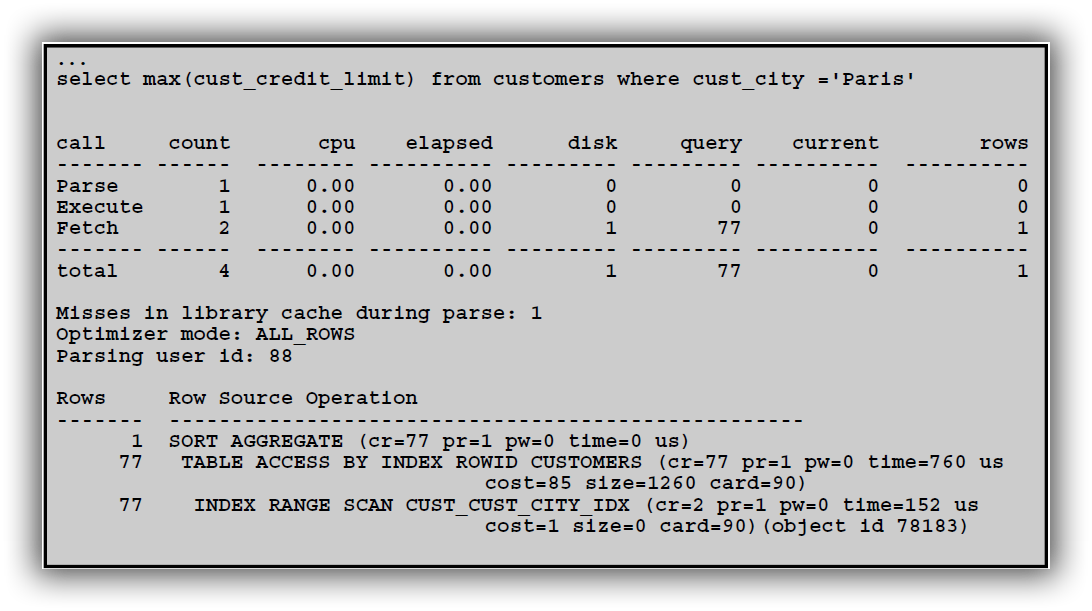
• Library cache misses

• Parsing user ID

• Execution plan

• Optimizer mode or hint

• Row source operation

****

**Recursive Calls**

Para ejecutar una declaración SQL emitida por un usuario, el servidor de Oracle ocasionalmente debe emitir declaraciones adicionales. Tales declaraciones se llaman declaraciones SQL recursivas. Por ejemplo, si inserta una fila en una tabla que no tiene suficiente espacio para contener esa fila, el servidor Oracle realiza llamadas recursivas para asignar el espacio dinámicamente. Las llamadas recursivas también se generan cuando la información del diccionario de datos no está disponible en la memoria caché del diccionario de datos y debe recuperarse del disco.

Si se producen llamadas recursivas mientras el recurso de Rastreo SQL está habilitado, tkprof las marca claramente como sentencias SQL recursivas en el archivo de salida. Puede suprimir la lista de llamadas recursivas en el archivo de salida configurando el parámetro de línea de comandos SYS=NO. Tenga en cuenta que las estadísticas de las sentencias SQL recursivas siempre se incluyen en el listado de la sentencia SQL que provocó la llamada recursiva.

**Library Cache Misses**

tkprof también enumera el número de errores de caché de la biblioteca resultantes de los pasos de análisis y ejecución para cada instrucción SQL. Estas estadísticas aparecen en líneas separadas siguiendo las estadísticas tabulares.

**Row Source Operations**

Proporcionan el número de filas procesadas para cada operación ejecutada en las filas e información adicional sobre el origen de las filas, como lecturas y escrituras físicas; cr = lecturas consistentes, w = escrituras físicas, r = lecturas físicas, tiempo = tiempo (en microsegundos).

**Parsing User ID**

Este es el ID del último usuario que analizó la declaración.

**Row Source Operation**

La operación de origen de fila muestra los orígenes de datos para la ejecución de la instrucción SQL. Esto se incluye solo si el cursor se ha cerrado durante el rastreo. Si la operación de origen de fila no aparece en el archivo de rastreo, es posible que desee ver la salida del EXPLAIN PLAN.

**Execution Plan**

Si especifica el parámetro EXPLAIN en la línea de comando tkprof, tkprof usa el comando EXPLAIN PLAN para generar el plan de ejecución de cada instrucción SQL rastreada. tkprof también muestra el número de filas procesadas por cada paso del plan de ejecución.

**Nota:** Tenga en cuenta que el plan de ejecución se genera en el momento en que se ejecuta el comando tkprof y no en el momento en que se produjo el archivo de rastreo. Esto podría marcar la diferencia si, por ejemplo, se ha creado o eliminado un índice desde el seguimiento de las declaraciones.

**Optimizer Mode or Hint**

Esto indica la sugerencia de optimizador que se utiliza durante la ejecución de la declaración. Si no hay ninguna pista, muestra el modo optimizador que se utiliza.

**Nota:** Si la CPU o los valores transcurridos son 0, timed\_statistics no se establece.

**Nota:** Los índices no deben construirse a menos que sea necesario. Los índices ralentizan el procesamiento de los comandos INSERT, UPDATE y DELETE porque las referencias a las filas deben agregarse, modificarse o eliminarse. Los índices no utilizados deben eliminarse. Sin embargo, en lugar de procesar todo el SQL de la aplicación a través de EXPLAIN PLAN, puede usar la supervisión de índice para identificar y eliminar cualquier índice que no se use.

**Optimizer Operators 245**