TEMA 9
RENDIMIENTO Y AFINACIÓN

9.1. INTRODUCCIÓN.

Las principales áreas de estudio para realizar performance y tuning de la base de datos son:

- Planeación para obtener un buen desempeño
- Afinamiento de la instancia (Instance tuning)
- Afinamiento de sentencias SQL (SQL tuning)

9.1.1. Instance tuning

- Como primer elemento, es fundamental considerar y revisar el correcto diseño inicial de la base de datos. Lo anterior con la finalidad de evitar cuellos de botella que pudieran originar problemas serios de desempeño.
- Otra actividad importante es la correcta asignación de memoria a las diferentes estructuras de la base de datos.
- Determinar correctamente los requerimientos en cuanto a la estructura física de la base de datos (configuración de los archivos de la base de datos).
- Afinamiento del sistema operativo para obtener el mejor desempeño de la base de datos.

9.1.2. Principios de desempeño

- Realizar afinación y rendimiento de la base de datos requiere de una metodología la cual es *diferente* a la metodología empleada para realizar la configuración inicial de la base de datos y de su entorno.
- Afinación y rendimiento está dirigido a la identificación de los principales *cuellos de botella* y a la aplicación de ciertos cambios que permitirán reducir este efecto.
- Típicamente esta es una actividad reactiva ya sea en sistemas pre- productivo o en sistemas ya en producción.

9.1.3. Obtención de datos iniciales.

- Como actividad fundamental requerida para obtener una adecuada medida del desempeño de una base de datos, es la obtención de métricas o datos iniciales que servirán como punto de partida para realizar comparaciones contra datos obtenidos en diversos escenarios y así detectar posibles problemas.
- Es importante identificar ***patrones*** de comportamiento del sistema. Por ejemplo, períodos ***pico*** o de máxima carga, periodos de nula o prácticamente nula actividad, existencia de periodos en los que la base de datos puede estar detenida.
- Es de vital importancia identificar estos periodos pico de máxima carga así como la instalación de una ***herramienta*** que permita realizar la recolección de datos bajo estos periodos de carga máxima.
- Esta recolección de datos debe incluir las siguientes categorías:
    - Estadísticas de la aplicación (tiempos de respuesta, volumen de transacciones realizadas)
    - Estadísticas de la base de datos.

  - Estadísticas del sistema operativo.
  - Estadísticas del uso de dispositivos I/O
  - Estadísticas del uso de una red de datos.
- Como se comentó al inicio del curso la herramienta que ofrece este manejador se llama ***AWR: Automatic Workload repository*** encargado de capturar estos datos cada cierto periodo de tiempo llamados ***snapshots*** que son empleados para realizar comparaciones entre medidas futuras y poder así detectar posibles problemas de afinamiento o desempeño.

9.1.4. Síntomas y problemas.

- Una de las principales fallas es la identificación errónea de problemas a partir de un conjunto de síntomas.
- Las estadísticas indican síntomas, sin embargo su identificación no es suficiente para proporcionar una solución. Las causas pueden ser diversas.

Ejemplos:

Lentitud al realizar operaciones I/O

- Causada por malas configuraciones aunque también podría ser que instrucciones SQL mal diseñadas o escritas provocan una innecesaria cantidad de operaciones I/O.

Excesivo uso del CPU

- Puede ser causado por una subestimación de recursos de hardware respecto al tamaño del sistema, por sentencias SQL mal afinadas.

9.1.5. ¿Cuándo se debe hacer tuning?

Existen 2 estrategias:
- Monitoreo proactivo
- Eliminación de cuellos de botella (reactivo)

A nivel general el objetivo general de realizar *tuning* es la reducción de uso de recursos o la reducción de los tiempos de respuesta obtenidos para completar una tarea. Para cualquiera de los 2 casos, se desea hacer *efectivo* el uso de un recurso.

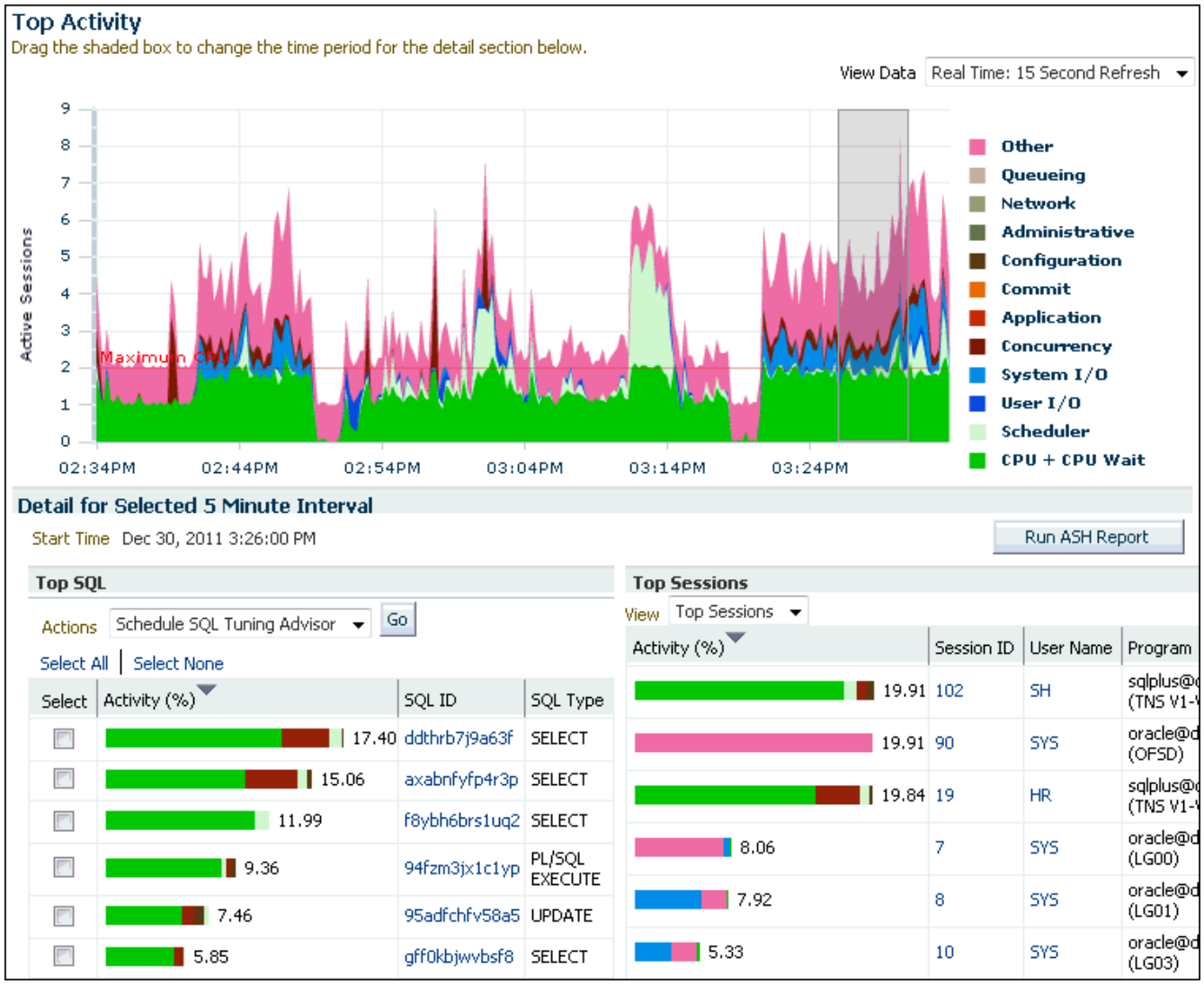
- La ***contención*** representa uno de los síntomas comunes que ocurren al momento de realizar *tuning.* Estos síntomas pueden resolverse de diversas formas
  - Cambios a nivel de la aplicación
  - Cambios en la base de datos
  - Cambios en la configuración del hardware.

9.1.6. SQL tuning

- Herramientas que generan SQL generalmente producen sentencias ineficientes.
- El correcto entendimiento de la forma en la que trabaja el mecanismo empleado para ***procesar sentencias SQL*** es necesario para escribir sentencias SQL óptimas. Esto en especial para sistemas OLTP que producen grandes cantidades de transacciones.

9.1.6.1. Optimizador y planes de ejecución.

- Cuando una sentencia SQL es ejecutada, el ***optimizador*** de consultas determina el ***plan de ejecución*** más eficiente posterior a haber considerado una serie de factores y condiciones que permitan obtener el menor costo posible.
- Durante el proceso de evaluación de la sentencia, el optimizador revisa las estadísticas recolectadas por el sistema para determinar la mejor estrategia para acceder a los datos solicitados.
- Las decisiones realizadas por el optimizador pueden ser sobrescritas por el usuario a través del uso de ***hints.***



9.1.7. Principales herramientas para realizar tuning.

- Con base a lo mencionado anteriormente, realizar una eficiente recolección de estadísticas es esencial para identificar y corregir problemas de desempeño.
- Cada manejador ofrece diversas herramientas que permiten realizar y automatizar esta tarea.
- Adicional a estas herramientas existen otras para realizar monitoreo del desempeño, realizar diagnósticos, y herramienta que permiten resolver el problema.
- Tanto la recolección como el monitoreo son actividades automáticas que son realizadas por procesos de background.

- El parámetro `statistics_level` debe tener el valor `typical` o `all`.

Algunas de las herramientas que realizan estas actividades se muestran en la siguiente tabla:

Herramienta	Descripción
Automatic Workload Repository (AWR)	Recolecta, procesa y mantiene un estadísticas de desempeño para realizar detección de problemas y habilitar la ***auto corrección***
Automatic Database Diagnostic Monitor (ADDM)	Analiza la información recolectada por el AWR para detectar posibles problemas de desempeño originados al interior de la base de datos.
SQL Tuning Advisor	Proporciona técnicas rápidas para realizar optimizaciones de sentencias SQL sin la necesidad de modificarlas. Ofrece soporte para diversos objetos. Tablas, índices, vistas, vistas materializadas, etc.
End to End Application tracing	Identifica excesivas cargas de trabajo en el sistema a través de la identificación del usuario, servicio o aplicación que esta generando esta problemática.
Server-generated alerts	De forma automática generan alertas o notificaciones cuando ocurren problemas que impidan realizar alguna tarea.
Additional Advisors	Pueden ser iniciados manualmente para realizar alguna tarea asociada con la detección de problemas de desempeño: memory advisors, mean time to recover advisor, etc.
Vistas del diccionario `v$`	Las vistas del diccionario de datos que inician con `v$` son conocidas como performance views ya que contienen datos que son empleadas por las herramientas antes mencionadas. Las vistas `v$` residen en memoria, son inicializadas durante el proceso de `startup` y son mantenidas mientras la instancia está iniciada.

9.1.8. Consideraciones de diseño de una aplicación.

Respecto diseño y desarrollo de software, los siguientes puntos representan aspectos muy relevantes que pueden impactar al desempeño y correcto funcionamiento de la base de datos

- El diseño lógico (modelo relacional) de la base de datos debe ser diseñado de la forma más simple y entendible posible, con los niveles de normalización adecuados, haciendo uso de las llamadas ***buenas prácticas.***
- Hacer uso de herramientas case empleadas para modelado apegadas lo más posible a los estándares de modelado, notaciones.
- Diseño físico de tablas e índices. Crucial definir un esquema de indexado como parte del diseño lógico considerando todos los tipos de índices disponibles (B-tree, BitMap, compuesto, basado en funciones, particionados), ventajas y desventajas. Respecto a las tablas, existe una gran cantidad de configuraciones que pueden ser utilizadas para implementar el diseño plasmado en los modelos relacionales. Algunas de estas opciones son: *pre –joined tables, derived columns, cluster tables, index organized tables, vistas materializadas, particionamiento, etc.*
- Uso de vistas. Permiten simplificar y agilizar el diseño de aplicaciones, principalmente al ocultar la consulta que las define, reduciendo considerablemente la complejidad. Sin embargo, su uso podría ser sub-optimo en especial para cuestiones de procesamiento de consultas. Hacer joins entre vistas o vista- tabla puede provocar la ejecución de sentencias SQL que requieran un uso excesivo e innecesario de recursos. Este tipo de sentencias evitan que el optimizador genere planes de ejecución óptimos. Siempre es preferible resolver las consultas sin el uso de vistas.

- Evitar a medida de lo posible el uso de bloques PL/SQL desde la aplicación, por ejemplo, invocar bloques PL/SQL desde Java, etc.
- Hacer uso de un cache del lado del software (middle tier) para almacenar consultas repetitivas hacia la base de datos que producen los mismos valores o que cambian con muy baja frecuencia. Asegurarse que este mecanismo genera un costo menor al de acceder repetidamente a la base de datos.
- Siempre hacer uso de FKs para garantizar referencias en lugar de implementarlas en la aplicación.
- Ojo con el uso de ORMs. (Object Relational Mapping). El uso de estas herramientas son muy populares hoy en día principalmente por la facilidad de interactuar con la base de datos sin tener que escribir SQL. Sin embargo, el código SQL que generan estas herramientas no siempre resulta en código óptimo provocando en algunas situaciones problemas serios de desempeño. El uso de ORMs es buena opción siempre y cuando exista una correspondencia entre el diseño lógico, el diseño orientado a objetos del software y la organización física de la base de datos.
- Otros aspectos importantes se especifican en el siguiente checklist.

9.1.9. Estimación del tamaño de la base de datos.

- Como parte el diseño de la base de datos es fundamental considerar o estimar lo más real posible el crecimiento de cada uno de los objetos de la base de datos. En especial tablas e índices. Este análisis permitirá realizar ajustes necesarios al diseño con la finalidad de mejorar desempeño. Esto incluye el uso de índices, posibles denormalizaciones e inclusive particionamiento.
- Una vez que la base de datos está en producción, algunos de estos objetos pueden tener un comportamiento de crecimiento incierto o difícil de estimar. En estos casos es importante que el DBA realice un monitoreo constante y comprobar que se cuenta con el espacio suficiente para soportar volúmenes grandes de datos. Esto incluye data files, tablespaces temporales, segmentos de rollback (undo).

9.1.10. Testing para verificar desempeño.

Las siguientes acciones pueden ser aplicadas para verificar si las recomendaciones hechas hasta el momento producen buenos resultados en cuanto a desempeño o si se requiere aplicar algún ajuste. Estas actividades deben ser documentadas correctamente para referencias futuras.

- Usar el ADDM y el SQL advisor para validar el diseño de la aplicación respecto a la interacción con la base de datos.
- Realizar pruebas con volúmenes de datos realistas y representativos. Las tablas deben contar con cardinalidades similares a las de producción, así como la creación de índices y recolección de estadísticas.
- Configurar los modos del optimizador que serán empleados en producción.
- Realizar pruebas de desempeño de forma inicial con un solo usuario conectado. Si estas pruebas no son exitosas, mucho menos serán las pruebas con usuarios concurrentes.
- Obtener y documentar ***todos*** o la gran mayoría de los planes de ejecución que produce el optimizador para las consultas. Asegurarse que el plan generado tenga el menor costo posible, y realizar acciones correctivas en caso de detectar planes de ejecución no óptimos. Marcar a aquellos que presentan un mayor uso de recursos ya que serán el punto de atención y monitoreo
- Intentar realizar pruebas con concurrencia de usuarios. Generalmente este tipo de pruebas son complicadas pero por lo menos tratar de ejecutar concurrencia de sentencias DML para descartar posibles conflictos de bloqueos.

- Probar con hardware y configuraciones similares al hardware de producción. Esto permitirá evitar problemas de desempeño en cuanto uso de recursos, configuraciones de red, etc.

9.1.11. Checklist inicial para desempeño

De manera simple e inicial se puede considerar el siguiente checklist como punto de partida para iniciar con un proceso adecuado de afinamiento una vez que la base de datos comience a operar.

- Configurar los siguientes parámetros con un valor mayor al estimado de tal forma que permitan su crecimiento en caso de no contar con un estimado preciso del crecimiento de la base de datos. `maxinstances`, `maxdatafiles`, `maxlogfiles`, `maxlogmembers`, `maxloghistory`. Esto provocará mayor espacio en disco, pero evitan la necesidad de aplicar operaciones de expansiones urgentes posterior a la ocurrencia de una falla.
- Si el proceso de testing antes de poner a producción la nueva base de datos fue realizado en sistemas similares a producción con cargas de trabajo similares, es posible exportar las estadísticas del ambiente de pruebas, así como el tamaño de bloque de datos empleado hacia la base de datos productiva.
- Configurar la cantidad de parámetros ***mínima*** indispensable. Lo más recomendable es iniciar con los valores por default. Si algún parámetro requiere ser ajustado, este se deberá ajustar cuando se presente la necesidad de realizar tuning, por ejemplo, al aumentar la carga de trabajo.
- Objetos que experimenten fuertes cargas de operaciones DML insert, update, delete, como son índices o tablas, deben configurar sus parámetros de almacenamiento a *automatic segment space management.* De forma adicional, emplear *automatic undo management.* Estas configuraciones permitirán principalmente evitar ***contención*** de bloques de datos.
- Todas las sentencias SQL deberán ser verificadas y optimizadas para hacer uso óptimo de los recursos de la base de datos. Un ejemplo importante es el uso de ***soft parsing*** empleando sentencias parametrizadas o preparadas.
- Verificar que las aplicaciones que se conectan a la base de datos hagan uso de pool de conexiones para evitar abrir y cerrar conexiones de forma repetitiva.
- Verificar que sentencias SQL hagan uso de cursores de forma eficiente. La base de datos debería parsear la sentencia una sola vez y ejecutarla N veces. Esto no ocurre cuando se escriben sentencias SQL con valores fijos en lugar del uso de ***bind variables,*** conocidas también como sentencias parametrizadas o sentencias preparadas.

Ejemplo:

Incorrecto:

```
select * from empleado
where ap_paterno like 'king';
```

Correcto:

```
select * from empleado
where ap_paterno like :1;
```

- Establecer un punto de referencia inicial o punto de partida respecto a la recolección y cálculo de estadísticas en todos los rubros antes mencionados. Esta medida inicial permitirá realizar comparaciones y adquirir conocimiento sobre el desempeño de la base de datos en posteriores recolecciones una vez que la base de datos se encuentre en un ambiente productivo.

- Prepararse para el primer cuello de botella el cual ocurrirá inevitablemente, a través del establecimiento de una metodología bien definida para atacar estas situaciones. Se recomienda segur las metodologías recomendadas por la propia base de datos.

9.2. CATEGORÍAS DE RENDIMIENTO Y AFINACIÓN

- Afinamiento de la instancia
  - Memoria
  - Afinamiento de la SGA
  - Afinamiento del DB Buffer cache
  - Afinamiento de Shared Pool y el Large Pool
  - Afinamiento de Query Result cache
  - Afinamiento de la PGA
- Herramientas empleadas en afinación y rendimiento.
  - Vistas del diccionario de datos asociadas con rendimiento
- Diseño y desarrollo de aplicaciones para obtener el mejor rendimiento
  - Escalabilidad
  - Arquitectura de la aplicación
  - Simplificación y optimización de diseño de bases de datos.
  - Diseño de tablas e índices
    - Particionamiento
  - Uso de vistas
  - Escritura de sentencias SQL desde una aplicación
    - Sentencias preparadas.
- Configuraciones de la base de datos para desempeño.
  - Espacio UNDO
  - Espacio REDO
  - Tablespaces
- Métricas de desempeño
  - Estadísticas
  - Recolección de estadísticas

Afinamiento SQL

- Optimizador de sentencias SQL
  - Transformación de sentencias SQL
    - OR expantion
    - View merging
    - Predicate pushing
    - Subquery Unnesting
    - Query rewrite using materialized views
    - Start transformation
  - Planes de ejecución
  - Métodos de acceso
    - Accesos a tablas
    - Accesos a índices
      - Full Index Scan
      - Range Index Scan
      - Unique index scan

        - Skip index scan
    - Joins
        - Nested Loop Joins
        - Hash Joins
        - Sort Merge Joins
        - Join factorization
- Monitoreo del desempeño de sentencias SQL