Servicio Andaluz de Salud
CONSEJERÍA DE SALUD

Oficina Técnica para la Gestión y Supervisión de Servicios TIC
Subdirección de Tecnologías de la Información

Best practices para Dataware Housing sobre Oracle RDBMS 12cR2

Referencia documento: InfV5_JASAS_DWH_BestPractices_v920.doc
Fecha: 16/11/2018
Versión: 9.2.0

Registro de Cambios

Fecha	Autor	Versión	Notas
14 de Abril de 2011	Oracle ACS	2.2.0	Versión inicial
14 de Julio de 2011	Oracle ACS	2.3.0	Versión Julio de 2011
13 de Octubre de 2011	Oracle ACS	2.4.0	Versión Octubre de 2011
12 de Enero de 2012	Oracle ACS	3.1.0	Versión Enero de 2012
14 de Marzo de 2013	Oracle ACS	4.1	Revisión de Marzo de 2013, contrato 2012-2014
13 de Junio de 2013	Oracle ACS	4.2	Revisión de Junio de 2013, contrato 2012-2014
17 de Octubre de 2013	Oracle ACS	4.3	Revisión de Octubre de 2013, contrato 2012-2014
16 de Julio de 2015	Enrique Ramiro	6.1	Revisión de Julio de 2015, contrato 2014-2016
16 de Diciembre de 2015	Enrique Ramiro	6.2	Revisión de Diciembre de 2015, contrato 2014-2016
16 de Junio de 2016	Enrique Ramiro	7.1	Revisión de Junio de 2016, contrato 2014-2016
16 de Noviembre de 2016	Enrique Ramiro	7.2	Revisión de Noviembre de 2016, contrato 2014-2016
16 de Junio de 2017	Enrique Ramiro	8.1	Revisión de Junio de 2017, contrato 2016-2018
16 de Noviembre de 2.017	Enrique Ramiro	8.2	Revisión de Noviembre de 2017, contrato 2016-2018
16 de Junio de 2.018	Enrique Ramiro	9.1	Revisión de Junio de 2017, contrato 2016-2018
16 de Noviembre de 2.018	Enrique Ramiro	9.2	Revisión de Noviembre de 2018, contrato 2016-2018

Revisiones

Nombre	Role
Jonathan Ortiz	Advanced Support Engineer
Gregorio Adame	Advanced Support Engineer
José María Gómez	Technical Account Manager

Distribución

Copia	Nombre	Empresa
1	Subdirección de Tecnologías de la Información	Servicio Andaluz de Salud, Junta de Andalucía
2	Dirección General de Política Digital	Consejería de Hacienda y Administración Pública, Junta de Andalucía

Índice de Contenidos

Control de cambios

Cambio	Descripción	Página
1	No se realizan cambios en esta versión	N/A

Introducción

Este documento recoge una serie de recomendaciones de Oracle Soporte planteadas como buenas prácticas de desarrollo para aplicaciones que hagan uso de Oracle RDBMS 12cR2 en entornos de Data Warehousing.

Estas recomendaciones están encaminadas a minimizar los posibles problemas de rendimiento en sistema de cualquier tamaño y en la gran mayoría de los casos se basan en la experiencia de casos reales gestionados por Oracle Soporte.

Finalmente, este documento también recoge una serie de conceptos de componentes, módulos y tecnologías relacionadas con Oracle RDBMS 12cR2 para Data Warehousing, que a juicio de Oracle Soporte, deberían tenerse claros para asegurar la aplicación de las recomendaciones recogidas en este documento, y de manera general, entender los productos Oracle sobre los que se sustentan los sistemas y aplicaciones.

Objetivos de este documento

A lo largo de los puntos de este documento se irá definiendo una guía de buenas prácticas para el desarrollo de aplicaciones sobre bases de datos de tipo Data Warehousing a través de Oracle RDBMS 12cR2.

Esta guía contendrá tanto prácticas recomendadas como prácticas a evitar y se apoyará en ejemplos y en información que permita analizar las recomendaciones en cada uno de los entornos de desarrollo y preproducción.

Este documento se centra principalmente en la versión Oracle RDBMS 12cR2, aunque la mayoría de las recomendaciones son igualmente aplicables a versiones anteriores; hasta la 9iR2.

Best Practices para Data Warehousing

Introducción

Las arquitecturas empleadas en sistemas de información hoy en día, son mucho más dinámicas que hace unos pocos años. Ahora, los negocios exigen más información, que se entregue más rápido y ofrecen servicios analíticos desde sus Enterprise DatawareHousing (EDW) a un conjunto de usuarios y aplicaciones cada vez más amplio. Con el fin de mantenerse al día con este incremento de la demanda el EDW debe trabajar casi en tiempo real y en alta disponibilidad.

Este documento proporciona un conjunto de buenas prácticas y ejemplos "how-to" para implementar un Data Warehouse (DWH) en Oracle Database 12c aprovechando todas sus funcionalidades. Para ello, este documento se divide en cuatro secciones:

- La primera sección trata los aspectos claves para la elección de la configuración de su plataforma hardware para asegurar un rendimiento óptimo.
- La segunda describe brevemente los dos modelos lógicos principales utilizados para los almacenes de bases de datos.
- La tercera describe cómo implementar el modelo físico para esos modelos lógicos de la manera más óptima en una base de datos Oracle.
- Por último, la cuarta sección abarca técnicas del sistema de gestión, incluyendo la gestión de la carga de trabajo y la configuración de base de datos.

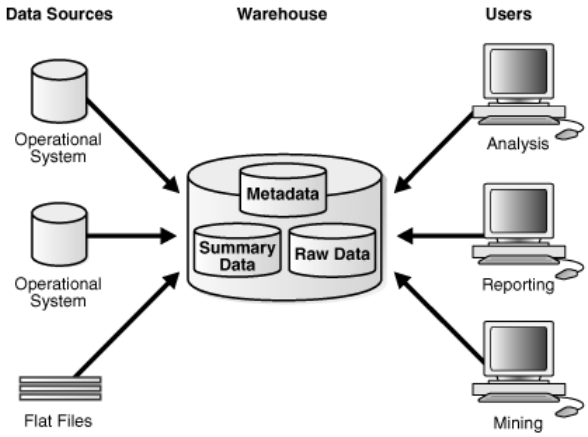
Este documento no es en absoluto una guía completa para Data Warehousing con Oracle. Debería consultar la documentación de la base de datos de Oracle, especialmente Oracle Data Warehouse, para obtener información detallada sobre todas las funciones de almacenamiento de Oracle.

Arquitectura de DWH

Los Data Warehouses y sus arquitecturas varían según las necesidades de una organización. Las tres arquitecturas más comunes de Data Warehouse son:

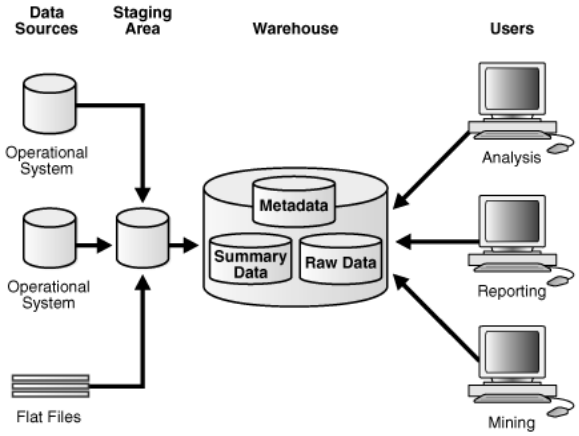
- Básica
- Con Área de Staging
- Con Área de Staging y Data Marts

Arquitectura Básica: Los usuarios finales acceden directamente a los datos derivados de varios sistemas fuente a través del DWH. Ejemplo:

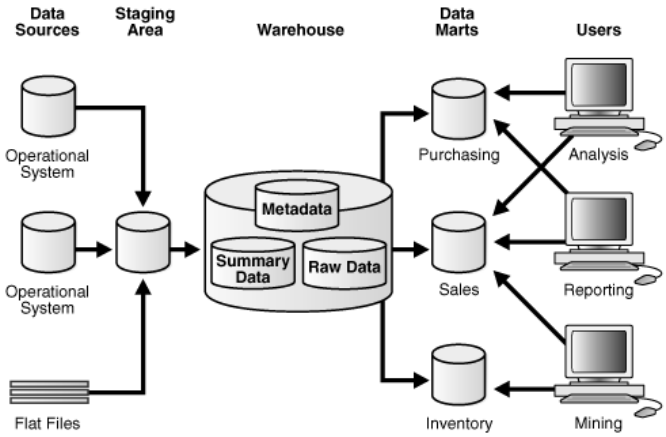


El almacenamiento consolidado de los datos como centro de una arquitectura data warehousing es lo que se conoce como Enterprise Data Warehouse (EDW). Un EDW ofrece una vista de 360 grados del negocio de una organización para mantener toda la información comercial relevante en el formato más detallado.

Arquitectura Con Área de Staging: Cuando se deben limpiar y procesar los datos operacionales antes de colocarlos en el warehouse. Ejemplo:



Arquitectura Con Área de Staging y Data Marts: Es bastante común cuando se desea personalizar la arquitectura para diferentes grupos dentro de su organización. Puede hacer esto agregando Data Marts, que son sistemas diseñados para una determinada línea de negocio. Ejemplo:



Los Data Marts se pueden implementar de forma física o puramente lógica a través de vistas. Pueden ubicarse junto con el EDW o construirse como sistemas separados. No es objetivo de este documento profundizar sobre Data Marts.

Configuración Balanceada

Sin importar el diseño o la implementación de un Data Warehouse, la principal clave para obtener un buen rendimiento se encuentra en la configuración del hardware utilizado. Esto se ha hecho más evidente que con el reciente aumento del número de aplicaciones Data Warehouse en el mercado. Muchas operaciones de tipo Data Warehouse se basan en pesados accesos a tablas (*Large Table Scans*) y otras operaciones de entrada/salida (E/S) masivas. Con el objetivo de lograr un rendimiento óptimo, la configuración hardware debe dimensionarse correctamente de extremo a extremo para soportar este nivel de rendimiento o throughput. Este tipo de configuración hardware se denomina sistema balanceado. En un sistema balanceado todos los componentes - desde la CPU a los discos - se organizan para trabajar en conjunto y así garantizar el máximo throughput de E/S posible.

Pero, ¿cómo clasificar el tamaño de un sistema? En primer lugar, debe tener en cuenta la capacidad de procesamiento que necesita su sistema y cuánto throughput puede manejar cada CPU individual o core en su configuración. Esta información puede determinarse a partir de un sistema existente.

Sin embargo, si los valores específicos no están disponibles, un valor aproximado de 200MB/seg de throughput de E/S por núcleo es una buena estimación para el diseño de un sistema balanceado. Todos los demás componentes críticos en el/los path/s de E/S - los adaptadores de Host Bus, las conexiones de fibra, el switch, controlador, y los discos - tienen que clasificarse en tamaño apropiadamente.

Con el uso de nuestra heurística de calibración, con 200MB/seg por núcleo, cada nodo, de 2 cores, puede manejar 400 MB/seg de E/S. En consecuencia, cada nodo necesita dos adaptadores de HBAs de 2Gbit: los dos adaptadores HBA por nodo pueden mantener un throughput de alrededor de 400 MB/s (200MB/s cada uno).

Para soportar un *full table scan* en los 4 nodos de una base de datos basada en Real Application Cluster, el throughput requerido para cada uno de los switch del canal de fibra es de aproximadamente 4 x 200 MB/s = 800 MB/s. Es importante tener en cuenta que sólo hay un HBA de cada nodo conectado a cada switch.

Por otro lado, cada switch atiende 4 arrays de discos. Cada conexión de canal de fibra a la matriz de disco es capaz de ofrecer 200 MB/s, por lo tanto el máximo throughput desde la matriz de discos hasta el switch también es 800MB/s por switch.

Cuando se dimensiona la matriz de disco se debe garantizar que los discos y los controladores puedan mantener un throughput de procesamiento total de 200 MB/s. Esto es muy importante para las unidades de disco de hoy en día ya que siguen aumentando la capacidad de almacenamiento sin el requisito del aumento de la velocidad. Es importante asegurar que el almacenamiento se dimensiona para el throughput y no sólo para las operaciones de E/S por segundo (IOPS) o capacidad (TB).

Oracle Real Application Clusters (RAC)

La tecnología RAC permite a las bases de datos de Oracle escalar horizontalmente en varias máquinas que tienen acceso compartido a una infraestructura de almacenamiento compartido. RAC se adapta de maravilla a EDW porque se puede combinar con el procesamiento paralelo de Oracle para escalar las cargas de trabajo en un grupo de servidores. RAC requiere una interconexión de datos (vía interconnect) entre todos los servidores del clúster. Para maximizar la escalabilidad y flexibilidad del clúster, la interconnect de RAC debe diseñarse para mantener un rendimiento similar al del subsistema de disco. Si se sigue esta recomendación de diseño, las consultas escalarán bien cuando se ejecuten en paralelo en todo el clúster.

RAC es una tecnología clave y se trata con más detalle en otros documentos de esta Asesoría Técnica. Entre ellos, hay uno de *Best practices para Dataware Housing sobre Real Application Cluster* (InfV5_JASAS_DWH_vs_RAC_v820.pdf).

La ejecución en paralelo también se trata con más detalle en otro documento de esta Asesoría Técnica: *Best practices de Parallel Execution para Dataware Housing* (InfV5_JASAS_DWH_vs_PEX_v820.pdf).

Interconnect

Cuando un Data Warehouse corre sobre una base de datos en Real Application Cluster (RAC), es importante dimensionar la interconnect del clúster con el mismo cuidado y precaución con que se dimensionó el throughput del subsistema de E/S.

La regla de oro para el dimensionamiento de la interconnect es mantenerlo a 200 MB/s por núcleo. ¿Por qué tanto? La razón es simple: una vez que los datos se leen de los discos para una determinada consulta se ubicarán en la memoria de proceso en uno de los nodos del clúster, en caso de que otro proceso en un nodo diferente requiera todos o algunos de esos datos para completar esta consulta, pedirá los datos que pasan a través de la interconnect en lugar de leer de nuevo de disco.

Si el ancho de banda de interconnect no es igual al ancho de banda de E/S de disco, esto dará lugar a un importante cuello de botella para la consulta. Dependiendo del número de núcleos por nodo que tenga, y el número total de nodos, es posible que necesite usar InfiniBand en lugar de múltiples tarjetas Ethernet Gibabit para la interconnect para garantizar que pueda mantener el throughput deseado. InfiniBand ofrece la mejor solución para sistemas de gran escala, ya que consume menos CPU por mensaje enviado o recibido.

Veamos un ejemplo, para un pequeño sistema de 4 nodos, la interconnect tiene que ser capaz de soportar 3.2GB/sec (unos 800 MB/s por nodo) a escala lineal para las operaciones de ejecución paralela entre nodos. Una sola tarjeta InfiniBand sería capaz de soportar este throughput. Sin embargo, si la interconnect se construyó usando tarjetas Ethernet de 4 Gb se necesitarán al menos cuatro tarjetas por nodo para mantener el throughput.

Diseño de la configuración y disposición de los discos

Una vez que la configuración del hardware ha sido establecida para funcionar como un sistema balanceado que pueda mantener el throughput requerido es necesario

centrarse en la distribución del disco. Uno de los problemas clave que podemos observar en las actuales implementaciones de Data Warehouse es un pobre diseño de la disposición de los discos.

En muchas ocasiones vamos a ver un gran Data Warehouse alojado en la misma matriz de discos que otras aplicaciones. Esto se hace a menudo porque el Data Warehouse no genera un número de IOPS que pueda saturar la matriz de discos. Lo que se tiene en cuenta en estas situaciones es el hecho de que el Data Warehouse hará menos E/S pero más grandes, que superarán fácilmente la capacidad de throughput de las matrices de discos en términos de gigabytes por segundo. Lo ideal sería que quisiera que su Data Warehouse residiera en su propia matriz/matrices de almacenamiento.

La configuración del subsistema de almacenamiento para un Data Warehouse debería ser simple, eficiente, de alta disponibilidad y muy escalable. Esto último es fundamenteal y no debería ser complicado o difícil de escalar. Una de las maneras más sencillas de lograrlo es aplicar la metodología Stripe and Mirror Everything (SAME). SAME se pueden implementar a nivel de hardware o mediante Automatic Storage Management (ASM) o mediante una combinación de ambos. Este documento sólo se ocupará de la implementación de SAME utilizando una combinación de características de hardware y ASM.

Desde el punto de vista hardware, se puede conseguir redundancia con la implementación de mirroring a nivel de subsistema de almacenamiento con RAID 1. Una vez que los grupos RAID han sido creados se puede volcar a ASM. ASM proporciona un sistema de archivos y capacidades del gestor de volúmenes, construidos en el núcleo de Oracle Database. Para utilizar ASM para almacenamiento de bases de datos, primero debe crear una instancia ASM. Una vez que la instancia de ASM se inicia, puede crear grupos de discos ASM. Un grupo de discos es un conjunto de dispositivos de disco, que ASM maneja como una sola unidad.

Un grupo de discos es comparable a un grupo de volúmenes LVM (Logical Volume Manager) o un grupo de almacenamiento. Cada disco del grupo de discos debe tener las mismas características físicas incluyendo el tamaño y la velocidad, ASM distribuye los datos de forma homogénea en todos los dispositivos del grupo de discos para optimizar el rendimiento y el uso. Si los dispositivos de un grupo de discos tienen características físicas diferentes, pueden crearse puntos críticos o cuellos de botella, así que es importante utilizar siempre dispositivos similares en un grupo de discos. ASM utiliza un tamaño de stripe de 1 MB por defecto.

ASM aumenta la disponibilidad de la base de datos, proporcionando la capacidad de agregar o quitar dispositivos de disco de los grupos de discos sin necesidad de parar la instancia de ASM o la base de datos que los utiliza. ASM automáticamente rebalancea los datos a través del grupo de discos después de que los discos hayan sido añadidos o eliminados. Esta capacidad permite a su Data Warehouse escalar sin problemas.

Para un entorno Data warehouse se recomiendan un mínimo de dos grupos de discos ASM, uno para datos (DATA) y otro para el Fast Recover Area (FRA).

Una vez que los grupos RAID se han establecido el grupo de discos ASM DATA puede crearse. El grupo de discos ASM contendrá los ocho pares RAID 1. Cuando se crea un tablespace, ASM asignará espacio en el grupo de discos en trozos o unidades de asignación de 1 MB en forma de round robin entre cada uno de los pares RAID 1.

ASM inicia cada asignación de espacio en un disco diferente y utiliza un orden aleatorio diferente cada vez para asegurar que los datos se distribuyen uniformemente entre todos los discos.

Modelo Lógico

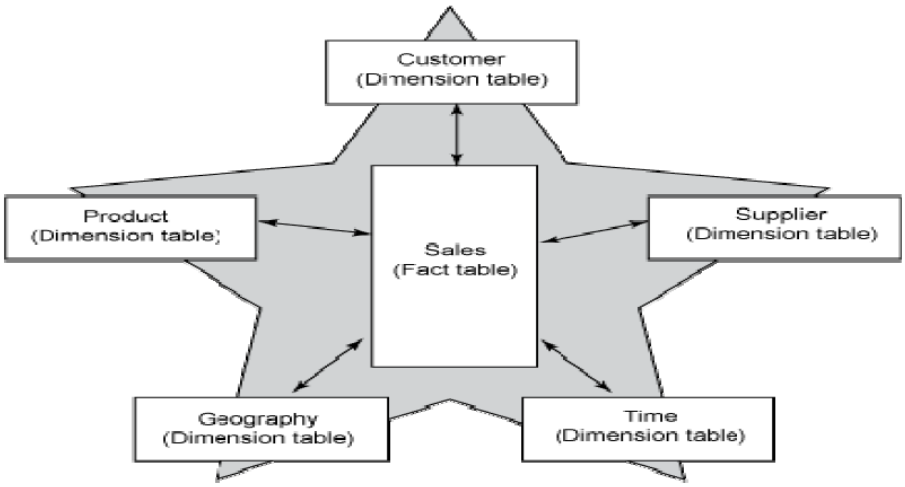
La distinción entre un modelo lógico y un modelo físico a veces es confuso. El modelo lógico de un Data Warehouse será tratado más como un modelo conceptual o abstracto, una visión más ideológica de lo que debería ser el Data Warehouse. El modelo físico describirá cómo se construye el Data Warehouse actualmente en una base de datos Oracle.

El modelo lógico es una parte esencial del proceso de desarrollo de un Data Warehouse. Permite definir los tipos de información que se necesitan para responder a las preguntas de negocio y las relaciones lógicas entre las diferentes partes de la información. Debería ser simple, fácil de entender y no tener en cuenta la base de datos física, el hardware que se utilizará para ejecutar el sistema o las herramientas que los usuarios finales utilizarán para acceder a él.

Hay dos modelos clásicos utilizados para Data Warehouse, la tercera forma normal y el dimensional o de esquema en estrella.

- Tercera Forma Normal (3FN) es una técnica clásica de modelado de bases de datos relacionales que minimiza la redundancia de datos a través de la normalización. Un esquema 3FN es un esquema de diseño neutral independiente de cualquier aplicación, que por lo general tiene un gran número de tablas. Conserva un registro detallado de cada transacción sin ningún tipo de redundancia de datos y da la oportunidad de codificar atributos y todas las relaciones entre elementos de datos. Normalmente, los usuarios necesitan un conocimiento sólido de los datos para navegar por la estructura más elaborada de forma fiable.

- El esquema de estrella se llama así porque el esquema se asemeja a una estrella, con puntos que irradian desde un centro. El centro de la estrella consiste de una o más tablas de hechos o fact tables y los puntos de la estrella son tablas de dimensiones o dimension tables. Las fact tables son tablas de gran tamaño que almacenan medidas de negocios y suelen tener claves externas a las dimension tables. Las dimension tables, también se conocen como buscadores o tablas de referencia, contienen los datos relativamente estáticos o descriptivos del Data Warehouse. Las fronteras del esquema de estrella en un modelo físico, así como rutas de navegación, la jerarquía y el perfil de consulta están incorporados en el propio modelo de datos en lugar de en los datos. Esto, en parte al menos, es lo que hace que la navegación por el modelo sea tan sencilla para los usuarios finales.



Existe una clara discusión con respecto a cuál es el "mejor" enfoque de modelado para un determinado Data Warehouse, se puede escoger entre 3FN clásico o dimensional cada uno con sus propias fortalezas y debilidades.

La mayoría de los EDW modernos deben abarcar los beneficios de cada tipo de modelo en lugar de depender de solo uno. Este es el enfoque que adopta Oracle en su arquitectura de referencia de gestión de la información (*Information Management Reference Architecture*): la mayoría de las implementaciones de EDW utilizan una combinación de ambos modelos. Lo más importante es diseñar su modelo de acuerdo a sus necesidades específicas de negocio.

Modelo Físico - Implementando el Modelo Lógico

El punto de partida para el modelo físico es el modelo lógico. El modelo físico debería reflejar el modelo lógico en medida de lo posible, a pesar de que sean necesarios algunos cambios en la estructura de las tablas y/o columnas. Además, el modelo físico incluirá staging tables o maintenance tables que normalmente no estarán incluidas en el modelo lógico.

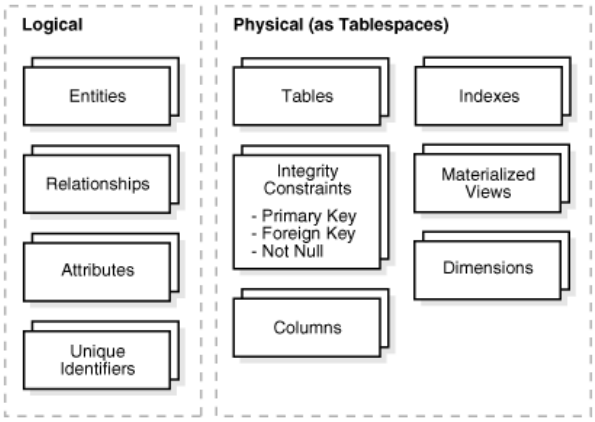
Aunque su entorno puede que no tenga estas capas claramente definidas, debería tener algunos aspectos de cada capa de la base de datos para asegurarse de que seguirá escalando a medida que aumente en tamaño y complejidad.

Para convertir su diseño lógico en un diseño físico, debe crear algunas o todas las siguientes estructuras: tablespaces, tablas, particiones de tablas o tablas index-organized, índices que incluyen índices particionados, vistas, restricciones de integridad, vistas materializadas y dimensiones.

Durante el proceso de diseño físico, se traducen los esquemas esperados a estructuras reales de base de datos. En este momento, se debe mapear:

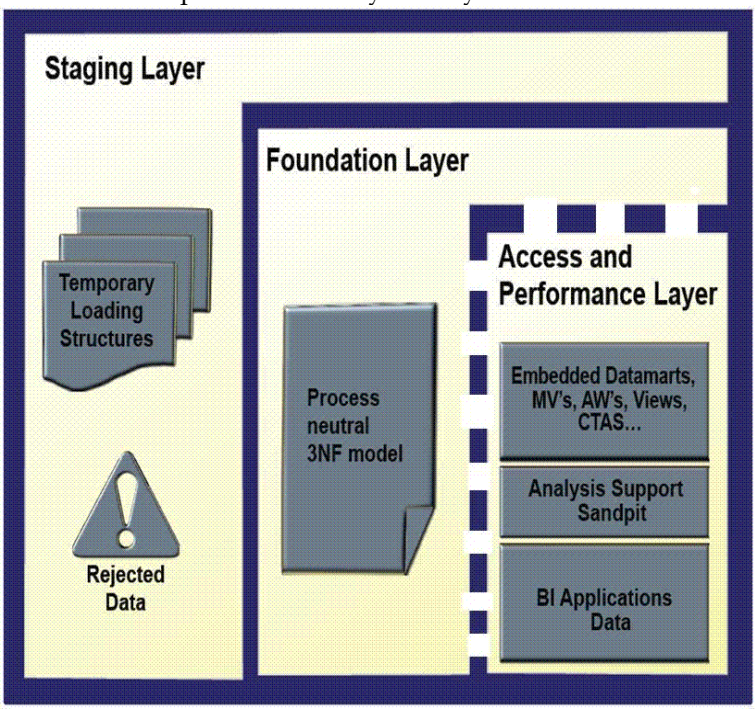
- Entidades con tablas
- Relaciones con restricciones de clave externa
- Atributos con columnas
- Identificadores únicos primarios con restricciones de clave primaria
- Identificadores únicos con restricciones de clave única

Logical Design Compared with Physical Design



La herramienta de interfaz gráfica de usuario, Oracle SQL Developer es útil para el proceso del diseño físico. Además, hay utilidades dentro del Oracle Tuning Pack conocidas como "advisors" (SQL Tuning Advisor y SQL Access Advisor) que se pueden utilizar de forma continua para identificar dónde las optimizaciones serán útiles.

Estas son las capas físicas o "Physical layers" de un Data Warehouse:



Muy comúnmente, las definiciones de las capas foundation y access se combinan en un único esquema.

Capa de Staging

La capa de staging o staging layer permite una rápida extracción, transformación y carga (ETL) de datos desde ficheros al Data Warehouse sin la distribución final de éstos. Es en esta capa donde se produce la mayor parte de la transformación de

datos complejos y el tratamiento de la calidad de los datos. Las tablas de la capa de staging, normalmente, van a mantenerse separadas del resto de los datos del Data Warehouse.

El enfoque más básico para la staging layer es que sea un esquema idéntico al que existe en el/los sistema/s operacional/es origen, pero con algunos cambios estructurales en las tablas, como el partitioning por rango. También es posible que algunas implementaciones de esta capa no sean necesarias, ya que todo el proceso de transformación de datos se hará "al vuelo" porque los datos se extraerán del sistema origen antes de ser insertados directamente en la capa final y todavía tendrán que cargar datos en el almacén.

Carga eficiente de datos

Tanto si van a cargarse en una staging layer o directamente en la capa final, la meta es obtener los datos del almacén de la manera más conveniente. Con el fin de conseguir un buen rendimiento durante la carga, necesita empezar centrándose en dónde se encuentran los datos que van a ser cargados y cómo cargarlos en la base de datos. Por ejemplo, para mover grandes volúmenes de datos no debería usar un único enlace en serie a la base de datos o una sola conexión JDBC. Los fichero planos o flat files son el medio más común y preferido para grandes volúmenes de datos.

Área de Staging

El área donde se almacenan los flat files antes de ser cargados en la staging layer de un sistema Data Warehouse se conoce comúnmente como área de staging o staging área. La velocidad total de la carga será determinada por (a) la rapidez con los datos no tratados o en bruto que pueden leerse de la staging layer y (b) la rapidez con que pueden procesarse e insertarse en la base de datos. Es muy recomendable que la etapa de datos en bruto se haga a través de tantos discos físicos como sea posible para asegurar que la lectura no da lugar a un cuello de botella durante la carga.

Preparación para los archivos de datos en bruto

Con el fin de paralelizar la carga de datos, Oracle necesita ser capaz de partir lógicamente los archivos de datos en bruto en trozos, conocidos como gránulos o granule. Para asegurar el procesamiento en paralelo balanceado, el número de gránulos suele ser mucho mayor que el número de procesos paralelos. En cualquier momento, a un proceso paralelo se le asigna un gránulo para trabajar en él, una vez que el proceso paralelo termina el trabajo, se le asignará otro gránulo hasta que todos los gránulos se hayan procesado y los datos se hayan cargado.

Para crear varios gránulos en un único fichero, Oracle tiene que ser capaz de mirar dentro del fichero de datos en crudo, sin procesar y determinar dónde empieza y termina cada fila de datos. Esto sólo es posible si cada fila ha sido claramente delimitada por un carácter como el de nueva línea o el de punto y coma.

Si un archivo no tiene esta característica, no es position-able o seek-able, por ejemplo, se trata de un archivo comprimido zip, éste no puede dividirse en gránulos y todo el archivo se trata como un único gránulo. En este caso, un sólo proceso paralelo podrá trabajar en el archivo completo.

Para paralelizar la carga de archivos de datos comprimidos necesita utilizar varios archivos de datos comprimidos. El número de archivos de datos comprimidos utilizados determinará el grado máximo de paralelismo usado en la carga.

Al cargar varios archivos de datos (comprimidos o sin comprimir) con una sola tabla externa se recomienda que los archivos sean similares en tamaño y que ese tamaño sea múltiplo de 10 MB. Si se utilizan archivos de diferentes tamaños se recomienda que los archivos estén ordenados de mayor a menor.

Por defecto, Oracle asume que el flat file tiene el mismo conjunto de caracteres que la base de datos. Si este no es el caso, usted deberá especificar el conjunto de caracteres del flat file en la definición de la tabla externa para garantizar la conversión al conjunto de caracteres adecuado.

Tablas externas

Oracle ofrece distintas opciones de carga de datos:

- Tabla externa o SQL * Loader.
- Oracle Data Pump, importación y exportación.
- Mecanismos de Change Data Capture y Trickle feed, por ejemplo, Oracle GoldenGate.
- Oracle Database Gateways para sistemas abiertos y mainframes.
- Conectividad Genérica(ODBC y JDBC)

¿Qué enfoque se debe tomar? Obviamente, esto dependerá del origen y el formato de los datos que se reciban. Como se mencionó anteriormente los flat files son el mecanismo más común para la carga de grandes volúmenes de datos, por lo que este documento sólo se centrará en la carga de datos desde flat files.

Si va a cargar archivos en Oracle tiene dos opciones, SQL * Loader o tablas externas. se recomienda encarecidamente que la carga se haga utilizando tablas externas en lugar de SQL * Loader:

- A diferencia de SQL * Loader, las tablas externas permiten la paralelización transparente dentro de la base de datos.
- Puede evitar datos de staging y la aplicación de transformaciones directamente en el archivo de datos mediante el uso de construcciones SQL o PL/SQL cuando acceda a tablas externas. SQL Loader requiere que la carga de los datos sea tal y como se hace en la base de datos.
- La paralelización de cargas con tablas externas permite una gestión más eficiente del espacio en comparación con SQL * Loader, donde cada cargador paralelo individual es una sesión de la base de datos independiente con su propia transacción. Para tablas muy particionadas esto podría dar lugar a una gran cantidad de espacio desaprovechado.
- Uso completo de capacidades SQL directamente sobre los datos
- Posibilidad de uso de DataPump y pre-procesamiento

Una tabla externa se crea usando la sintaxis estándar CREATE TABLE, excepto que hay que añadir información adicional de los flat files que están fuera de la base de datos.

```
SH@DB1> CREATE TABLE ext_tab_for_sales_data (
      2     Price       NUMBER(6),
```

```
 3     Quantity    NUMBER(6),
 4     Time_id     DATE)
 5  ORGANIZATION EXTERNAL (
 6  type ORACLE_LOADER
 7  default directory data_dir
 8  access parameter
 9  (  RECORDS DELIMITED BY newline
10     PREPROCESSOR execdir:'uncompress.sh'
11     BADFILE 'ulcase1.bad'
12     LOGFILE 'ulcase1.log'
13     FIELDS TERMINATED BY ",")
14     LOCATION ('sales_data_for_january.dat'))
15  parallel
16  REJECT LIMIT UNLIMITED;
```

El método más común para cargar datos desde una tabla externa es hacer una sentencia CREATE TABLE AS SELECT (CTAS) o una sentencia INSERT AS SELECT (IAS) en una tabla existente.

```
SH@DB1> insert into sales partition(p2) select * from
ext_tab_for_sales_data;
```

Batch Loading

La clave para obtener un buen rendimiento en la carga es utilizar direct path load siempre que sea posible. Un direct path load analiza los datos de entrada de acuerdo a la descripción dada en la definición de la tabla externa, convierte los datos de cada campo de entrada en su correspondiente tipo de datos de Oracle y a continuación, crea una estructura de array de columnas para los datos. Estas estructuras se utilizan para dar formato a bloques de datos de Oracle y para crear claves de índices.

Los bloques de base de datos recién formateados se escriben directamente en la base de datos, sin pasar por el motor de procesamiento SQL estándar ni por el buffer caché de la base de datos.

El mecanismo CTAS siempre utilizará direct path load pero la sentencia IAS no. Para lograr direct path load con una sentencia IAS, debe agregar APPEND al comando.

```
SH@DB1> insert /*+ APPEND */ into sales partition(p2)
select * from ext_tab_for_sales_data;
```

Las operaciones de tipo direct path load también pueden ejecutarse en paralelo. Puede establecerse el grado de paralelismo, bien mediante la adición de PARALLEL a la declaración CTAS o IAS o mediante el establecimiento de la cláusula PARALLEL tanto en la tabla externa como en la tabla en la que los datos van a ser cargados. Una vez que el grado de paralelismo se ha establecido en CTAS, automáticamente se hará el direct path load en paralelo, pero una IAS no. Para que las IAS hagan direct path load en paralelo debe modificarse el período de sesiones para habilitar DML en paralelo.

```
SH@DB1> alter session enable parallel dml;

Session altered.
```

```
SH@DB1> insert /*+ APPEND */ into sales partition(p2)
select * from ext_tab_for_sales_data;
```

Cargas por partitioning exchange

Se recomienda encarecidamente que las tablas o fact tables más grandes en un data warehouse se particionen. Uno de los beneficios del partitioning es la capacidad de carga de datos rápida y fácil con un mínimo impacto en los usuarios de negocio mediante el comando exchange partition. El comando exchange partition permite intercambiar los datos de una tabla sin particionar a una partición particular de una tabla particionada. El comando no mueve los datos físicamente, sino que actualiza el diccionario de datos a intercambiar con un puntero desde la partición de la tabla a la tabla y viceversa. Como no hay ningún movimiento físico de datos, el intercambio no genera UNDO ni REDO, por lo que es una operación de menos de un segundo y tiene menos probabilidad de impactar en el rendimiento que cualquier otro método tradicional de movimiento de datos como INSERT.

Supongamos que tenemos una tabla grande llamada Sales, que está particionada por rango de día. Al final de cada jornada, los datos del sistema de ventas online se cargan en la tabla Sales del almacén de datos. Los cinco pasos que se muestran a continuación aseguran que los datos diarios se cargan en la partición correcta con un impacto mínimo en los usuarios de negocio del sistema Data Warehouse y a una velocidad óptima.

Pasos para carga con partitioning exchange

- Crear una tabla externa para los flat files de datos que proceden del sistema online
- Utilizar una sentencia CTAS, crear una tabla sin particionar llamada tmp_sales que tenga la misma estructura de columnas que la tabla Sales
- Generar los mismos índices que están en la tabla Sales en la tabla tmp_sales
- Emitir el comando partition exchange

```
SH@DB1> alter tables sales exchange partition p2 with
  2   table tmp_sales including indexex without validation;
```

El comando exchange partition en el cuarto paso, intercambia las definiciones de dicha partición con la tabla tmp_sales, por lo que los datos instantáneamente están en el lugar correcto en la tabla particionada. Por otra parte, con la inclusión de las dos cláusulas extra opcionales, las definiciones de índices se intercambiarán y Oracle no comprobará si los datos pertenecen realmente a la partición - por lo que el intercambio es muy rápido. Se supone que la integridad de los datos fue verificada en el momento de la extracción. Si no está seguro de la integridad de los datos, no utilice la cláusula WITHOUT VALIDATION; la base de datos se encargará de verificar la validez de los datos.

Optimización de las Transformaciones de datos

Cualquier tipo de procesamiento de datos basado en conjuntos "*set-based*" es el método más rápido de procesar datos.

Los procesos de carga pueden incluir transformaciones de datos, o las transformaciones pueden seguir pasos de carga de datos. En muchos casos, las herramientas de transformación como Oracle Data Integrator (ODI) se utilizarán

para lograr esto, pero vale la pena tener en cuenta algunos principios generales para procesar grandes volúmenes de datos de manera oportuna.

La transformación de datos utilizando operaciones masivas "bulk" como CREATE TABLE AS SELECT (CTAS), MERGE and INSERT AS SELECT (IAS) son muy eficientes. Es muy fácil escalar estas operaciones usando consultas paralelas, DDL y DML en caso de que sea necesario acortar los tiempos transcurridos. Las operaciones masivas como estas generalmente se conocen como basadas en conjuntos "set-based". CTAS e IAS se recomiendan para el movimiento de datos entre las diferentes capas del EDW, especialmente si la cantidad de transformación requerida es mínima.

La alternativa al procesamiento basado en conjuntos es el procesamiento de fila por fila "row-by-row". Esto generalmente implica ejecutar un loop para una gran cantidad de iteraciones, cada una de las cuales agrega o cambia una pequeña cantidad de datos usando el conventional path DML. Este enfoque a menudo se considera más intuitivo que el procesamiento basado en conjuntos cuando se aplica la lógica de negocios, pero casi siempre es menos eficiente. Esto en sí mismo puede no ser un problema, pero escalar una tarea que incorpora un loop de un solo proceso es a menudo difícil. Si se propone el procesamiento fila a fila, se debe establecer un método para escalar a través de múltiples flujos de procesamiento al principio del proceso de diseño. En este caso, Oracle Partitioning es muy útil para subdividir conjuntos de datos en unidades de trabajo separadas que serán tratadas por múltiples hilos de procesamiento.

Compresión de datos

Otra decisión fundamental que hay que tomar durante la fase de carga es si comprimir o no los datos. El uso de comprensión de tablas, obviamente, reduce el uso del disco y la memoria, dando como resultado un mejor rendimiento de escalado para las operaciones de sólo lectura. La compresión de tablas también puede acelerar la ejecución de la consulta minimizando el número de rotaciones necesarias para recuperar los datos de los discos. La compresión de datos sin embargo, impone una reducción del rendimiento en la velocidad de la carga de los datos. Normalmente, el aumento del rendimiento global es mayor que el costo de la compresión.

Oracle ofrece tres tipos de compresión: 1) compresión básica, 2) la compresión de OLTP ahora llamada *Advanced Row Compression*, que es una funcionalidad de la opción con coste adicional *Advanced Compression*, y 3) Hybrid Columnar Compression (HCC) que sólo está disponible en Oracle Exadata Database Machine y por lo tanto no será tratada aquí.

Con la compresión básica o estándar, Oracle comprime los datos en un bloque de base de datos mediante la eliminación de valores repetidos. La compresión estándar sólo funciona para operaciones direct path, CTAS o IAS, como se explicó anteriormente. Si los datos son modificados por cualquier tipo de operación DML convencional, por ejemplo update, los datos dentro de ese bloque de base de datos se descomprimirán al realizar las modificaciones y serán grabados en disco sin comprimir.

Con la compresión OLTP ó Advanced Row Compression, al igual que en la compresión estándar, Oracle comprime los datos en un bloque de base de datos mediante la eliminación de valores repetidos. Pero a diferencia de la compresión

estándar, la compresión de datos OLTP permite que los datos sigan estando comprimidos durante cualquier tipo de operación de manipulación de datos, incluyendo DML convencionales tales como INSERT y UPDATE. Además, minimiza la sobrecarga de las operaciones de escritura en datos comprimidos, por lo que es adecuado tanto para entornos transaccionales/OLTP como para Data Warehouses, extendiendo los beneficios de la compresión a toda la carga de trabajo de las aplicaciones.

Si decide utilizar compresión, considere ordenar los datos antes de cargarlos para lograr la mejor tasa de compresión posible. La forma más sencilla de ordenar los datos de entrada es cargar con una cláusula ORDER BY en cualquiera de las sentencias CTAS o IAS. Debe hacer ORDER BY en una columna NOT NULL, lo ideal es que no tenga valor numérico que tenga un gran número de valores distintos, por ejemplo, de 1.000 a 10.000.

Tercera Forma Normal (3FN)

Desde la capa de staging, los datos harán la transición a la capa final a través de otro conjunto de procesos ETL. En este momento, los datos comienzan a tomar forma y no es raro que algunas de las aplicaciones de usuario final accedan a estos datos desde esta capa especialmente, sobre todo si son sensibles al tiempo, ya que estos datos estarán disponibles aquí antes de que sean transformados en la capa dimension/performance. Tradicionalmente esta capa se implementa en tercera forma normal (3FN).

Optimizando el esquema 3FN

La optimización de un esquema 3FN en Oracle requiere las tres Ps: Power, Partitioning y Paralell Execution. Power significa que la configuración del hardware debe ser balanceada. Las tablas más grandes deben particionarse mediante partitioning compuesto, por ejemplo, rango-hash o list-hash. Hay tres razones para ello:

- Mayor facilidad de administración de terabytes de datos
- Mayor accesibilidad a datos necesarios
- Joins de tablas eficientes y potentes

Por último la ejecución en paralelo permite que una tarea de base de datos pueda paralelizarse o dividirse en unidades más pequeñas de trabajo, lo que permite que varios procesos trabajen concurrentemente. Mediante el uso de paralelismo, un terabyte de datos puede ser analizado y procesado en cuestión de minutos o menos, no en horas ni días. La ejecución paralela se discutirá con más detalle a continuación en la sección de administración del sistema.

Partitioning

Partitioning permite dividir una tabla, index o index-organized table (IOT) en componentes más pequeños. Cada pieza de un objeto de base de datos se llama partición. Cada partición tiene su propio nombre y, opcionalmente, puede tener sus propias características de almacenamiento. Desde la perspectiva de un administrador de bases de datos, un objeto particionado tiene múltiples componentes que pueden ser gestionados de forma colectiva o individual.

Esto le da bastante flexibilidad al administrador en la gestión de objetos particionados. Sin embargo, desde la perspectiva de la aplicación, una tabla particionada es idéntica a una tabla no particionada; cuando se accede a una tabla particionada usando comandos SQL, no son necesarias modificaciones.

Partitioning puede proporcionar enormes beneficios a una amplia variedad de aplicaciones mejorando la gestión, la disponibilidad y el rendimiento.

Partitioning y manejo de objetos

El partitioning por rango ayudará a mejorar la gestión y la disponibilidad de grandes volúmenes de datos. Considere un caso en el que se almacenan los datos de ventas de dos años o 100 terabytes (TB) en una tabla. Al final de cada día un nuevo lote de datos tiene que cargarse en la tabla y los valores de los días más antiguos tienen que ser eliminados. Si la tabla de ventas está particionada por rango de día los nuevos datos pueden ser cargados con una carga de tipo partition exchange como se describió anteriormente. Esta es una operación de menos de un segundo y debe tener poco o ningún impacto para consultas de usuarios finales. Para eliminar el día más antiguo de los datos, simplemente ejecute el siguiente comando:

```
SH@DB1> alter table sales drop partition sales_q4_2011;
```

Partitioning y el acceso de datos

El partitioning por rango también garantizará que sólo se recorren los datos necesarios para responder a una consulta. Supongamos que los usuarios de negocio acceden predominantemente a los datos de ventas sobre una base semanal, por ejemplo, las ventas totales por semana, entonces particionando por rango de día esta tabla se asegura que los datos van a ser accedidos de la manera más eficiente, ya que sólo se necesita recorrer la partición 7 para responder a la consulta de usuarios de negocio en lugar de recorrer toda la tabla. La capacidad para evitar el recorrido de particiones irrelevantes se conoce como partitioning prunning.

Partitioning y el rendimiento de joins

El sub-partitioning por hash se utiliza principalmente por motivos de rendimiento. Oracle utiliza un algoritmo de hashing lineal para crear sub-particiones. Para garantizar que los datos se distribuyan uniformemente entre las particiones hash es muy recomendable que el número de particiones hash sea una potencia de 2 (por ejemplo, 2, 4, 8, etc).

Una partición hash debe tener al menos 16 MB de tamaño. Con algo menos no se obtendrán tasas de recorrrido eficientes con consultas en paralelo. Una de las principales ventajas de rendimiento en hash partitioning es el uso de partition-wise joins. Los Partition-wise joins reducen el tiempo de respuesta de la consulta, minimizando la cantidad de datos intercambiados entre los procesos de ejecución paralela cuando se ejecutan joins en paralelo. Esto reduce significativamente el tiempo de respuesta y mejora tanto el uso de la CPU como el uso de los recursos de memoria. En un data warehouse en clúster, se reducen significativamente los tiempos de respuesta mediante la limitación del tráfico de datos a través de interconnect (IPC), que es la clave para lograr una buena escalabilidad para

operaciones de join masivas. El Partition-wise join puede ser total o parcial, dependiendo del esquema de partición de las tablas a las que se va a hacer join.

Un full partition-wise join divide un join entre dos tablas de gran tamaño en varios joins más pequeños. Cada join pequeño, realiza un join entre un par de particiones, una de cada una de las tablas sobre las que se hace el join. Para el optimizador elija el método full partition-wise join, ambas tablas deben ser equi-particionadas por sus claves de join. Es decir, tienen que estar particionadas por la misma columna y con el mismo método de partitioning.

La ejecución paralela de un full partition-wise join es similar a su ejecución secuencial, excepto que en lugar de hacer join a un par de particiones a la vez, se hace join a varios pares de particiones en paralelo por varios procesos de consultas paralelas. El número de particiones a las que se les hace join al mismo tiempo está determinado por el grado de paralelismo (DOP).

Para asegurarse obtención de un rendimiento óptimo al ejecutar una partition-wise join en paralelo, el número de particiones en cada una de las tablas debe ser mayor que el grado de paralelismo usado en el join. Si hay más particiones que procesos paralelos, a cada proceso paralelo se le dará un par de particiones para hacerles el join, cuando el proceso paralelo termine ese join, pedirá otro par de particiones para hacerles join. Este proceso se repite hasta que todas las parejas se hayan procesado. Este método permite que la carga sea balanceada dinámicamente (por ejemplo, 128 particiones con un grado de paralelismo de los 32).

¿Qué pasa si sólo una de las tablas sobre las que se va a hacer join está particionada? En este caso, el optimizador puede elegir partial partition-wise join. A diferencia de full partition-wise joins, partial partition-wise join pueden aplicarse si una sola tabla sobre las que se va a hacer join está particionada. Por lo tanto, los partial partition-wise join son más comunes que los full partition-wise joins. Para ejecutar un partial partition-wise join, Oracle dinámicamente particiona la otra tabla en base a la estrategia de partitioning de la tabla particionada. Una vez que la otra tabla está particionada, la ejecución es similar a un full partition-wise join. La operación de distribución consiste en el intercambio de las filas entre los procesos de ejecución paralela. Esta operación da lugar a tráfico de interconnect en entornos RAC, ya que los datos necesitan particionarse través de los nodos.

Capa de acceso – Esquema en estrella

La capa de acceso representa los datos de forma que la mayoría de los usuarios y las aplicaciones puedan entender. En esta capa es muy probable ver un esquema en estrella.

Una consulta típica en la capa de acceso será un join entre la fact table y un cierto número de dimension tables, esta consulta suele conocerse como consulta en estrella. En una consulta en estrella a cada dimensión table se le hará join con la fact table usando la clave principal y la clave externa. Normalmente, a las dimension tables no se les hace join entre sí. Una cuestión de negocio que podría pedirse al esquema en estrella sería "¿Cuál fue el número total de paraguas vendidos en Boston durante el mes de mayo de 2011?"

```
SH@DB1> select SUM(quantity_sold) total_umbrellas_sold
  2   from sales s, customers c, products p, times t
  3   where s.cust_id = c.cust_id
```

```
4   and s.prod_id = p.prod_id
5   and s.time_id = t.time_id
6   and c.cust_city = 'BOSTON'
7   and p.product = 'UMBRELLA'
8   and t.month = 'MAY'
9   and t.year = 2011;
```

Como se puede ver todos los predicados de la cláusula where se hacen sobre las dimension tables y a la fact table (Sales) se le hace join con cada una de las dimension tables utilizando su clave externa, relacionada con la clave principal. Pero, ¿cómo abordar la optimización de este tipo de consultas?

Optimización de consultas en estrella

La optimización de una consulta en estrella es muy sencilla. Los dos criterios más importantes son:

- Crear un bitmap index en cada una de las columnas de clave externa en la fact table/s.
- Inicializar el parámetro de STAR_TRANSFORMATION_ENABLED a TRUE. Esto habilitará la función del optimizador para consultas en estrella que está desactivada por defecto para la compatibilidad con versiones anteriores.

Si su entorno cumple estos dos criterios sus consultas en estrella deben utilizar una técnica de optimización que reescriba o transforme su SQL. Esta técnica se conoce como transformación en estrella. La transformación en estrella ejecuta la consulta en dos fases, la primera fase recupera las filas necesarias de la fact table (conjunto de filas), mientras que la segunda fase hace join a este conjunto de filas con las dimension tables. Las filas de la fact table se recuperan mediante el uso de joins de bitmap entre los bitmap index y todas las columnas de clave externa. El usuario final no necesita saber nada de los detalles de STAR_TRANSFORMATION, ya que el optimizador seleccionará automáticamente STAR_TRANSFORMATION cuando sea necesario.

Pero, ¿cómo se efectuará o se reescribirá STAR_TRANSFORMATION nuestra consulta en estrella?. Como se mencionó anteriormente, la consulta será procesada en dos fases. En la primera fase, Oracle transformará o reescribirá nuestra consulta para que cada uno de los joins en la fact table se reescriban como sub-consultas.

Al volver a escribir la consulta de esta forma, podemos aprovechar la fortaleza de los bitmap index. Los bitmap index proporcionan un procesamiento basado en conjuntos dentro de la base de datos, lo que nos permite utilizar métodos fact para operaciones sobre conjuntos como AND, OR, MINUS y COUNT. Por lo tanto, vamos a utilizar el bitmap index sobre TIME_ID para identificar el conjunto de filas de la fact table correspondientes a las ventas de mayo de 2008.

En el bitmap el conjunto de filas en realidad se representa como una cadena de 1s y 0s. Un bitmap similar recupera las filas de la fact table que corresponden a la venta de paraguas y otro accede a las ventas realizadas en Boston. En este momento contamos con tres bitmaps, cada uno representando un conjunto de filas de la fact table que satisfacen una restricción individual. Los tres bitmap se combinan con un bitmap de operación AND y es este último bitmap el que se utiliza para extraer las filas de la fact table necesarias para evaluar la consulta.

La segunda fase es hacer join a las filas de la fact table con las dimesion tables. Para los join a las dimension tables se suelen hacer uso de hash join, aunque el optimizador de Oracle seleccionará el método de join más eficiente dependiendo del tamaño de las dimension tables.

```
SH@DB1> select SUM(quantity_sold) total_umbrellas_sold
  2   from sales s
  3   where s.cust_id IN
  4   (select c.cust_id from customers c where
  5      c.cust_city='BOSTON')
  6   and s.prod_id IN
  7   (select p.prod_id from products p where
  8      p.product='UMBRELLA')
  9   and s.time_id IN
 10   (select t.time_id from times t where
 11      t.month='MAY' and t.year='2011')
```

La figura 14 muestra el plan de ejecución típico de una consulta en estrellas donde se ha habilitado STAR_TRANSFORMATION. El plan de ejecución puede no ser exactamente como lo imaginaba. Puede haber notado que no se hace join a la tabla customer una vez que las filas se han recuperado con éxito de la tabla sales. Si se fija bien en la lista del select en realidad no se selecciona nada de la tabla Customers por lo que el optimizador no se molesta en hacer el join a la dimension table. También puede observar que en algunas consultas, incluso si STAR_TRANFORMATION se ha inicializado pueden no usarse todos los bitmap index en la fact table. El optimizador decide cuántos bitmap index se necesitan para recuperar las filas de la fact table. Si uno los bitmap index adicionales no va a mejorar, el optimizador no lo usará. La única vez que podrá ver la dimension table que se corresponde con el bitmap excluido en el plan de ejecución será durante la segunda fase o fase de join back.

Attribute Clustering y Zone Maps

Attribute Clustering y Zone Maps son dos nuevas funcionalidades de Oracle Database 12.1.0.2. Hay que considerar su uso para optimizar los esquemas de estrella. Los mapas de zonas y los clústeres de atributos son funciones que se pueden usar para reducir la E/S en aplicaciones de base de datos que de otra manera escanearían grandes volúmenes de datos. Estas dos características se pueden usar individualmente, pero están diseñadas para funcionar juntas.

Sin embargo, la funcionalidad de Zone Maps únicamente está disponible para los sistemas de ingeniería conjunta de Oracle "*Engineered Systems*" como Oracle Exadata y Oracle SuperCluster, de ahí que no se aborde en este documento en profundidad.

Los attribute clustering son una directiva a nivel de tabla que se utiliza para "ordenar" filas para que se almacenen con proximidad física en base a los valores de columna. El orden de las filas se puede basar en los valores de columna unicamente de la tabla con el atributo clusterizado, pero también en los valores de columna de tablas que se unen con la tabla que tiene el atributo clusterizado.

Un *zone map* o mapa de zona es una estructura de datos que conceptualmente divide una tabla en regiones contiguas de bloques llamadas zonas. Para cada zona, el mapa de zona registra los valores mínimo y máximo para las columnas especificadas. Las consultas que filtran en columnas del mapa de zona tienen el potencial de ser optimizadas; es posible evitar el escaneo de zonas que contienen rangos de valores

de columna que se sabe están fuera del rango definido por los predicados de consultas. Los mapas de zona, por lo tanto, permiten hacer "pruning" de zonas (y potencialmente particiones completas) en función de los predicados de columna y reducen significativamente la cantidad de datos que deben escanearse y por tanto I/O.

Ejemplo: El attribute clustering se puede usar en la tabla VENTAS para mantener juntas todas las filas que corresponden al nombre de la región "SEVILLA", dentro de un número mínimo de zonas. De esta forma, cualquier consulta que se filtre en "SEVILLA" necesitará escanear un número mínimo de zonas para encontrar las filas de "SEVILLA" coincidentes.

Los mapas de zonas se pueden usar en lugar de índices en fact tables y ofrecen una alternativa a la optimización del esquema en estrella con índices bitmap que se ha tratado en el sub-apartado anterior. Sin embargo, difieren de los índices en la forma en que se sincronizan con los cambios realizados en los datos de la tabla. Los índices siempre se sincronizan con los datos que se guardan en las tablas, pero los mapas de zona son similares a las vistas materializadas (que se ven a continuación); deben actualizarse para que se sincronicen con los datos de la tabla.

Vistas Materializadas

Hay que considerar el uso de vistas materializadas para consultas costosas repetitivas, optimizando así el tiempo de ejecución y el consumo de recursos.

La agrupación/recopilación y resumen de los datos es una característica importante de muchos data warehouses. Normalmente, las consultas analizarán un subconjunto o agrupación de los datos detallados, por lo que un mecanismo para pre-resumir y pre-agrupar datos para su uso directo mediante consultas, ofrece la posibilidad de mejorar significativamente el rendimiento de las mismas. Las consultas que acceden a datos de resumen en lugar de a los datos detallados utilizarán menos recursos del sistema, por lo que existe la posibilidad de mejorar también la escalabilidad general del sistema. Idealmente, los resúmenes y agrupaciones deberían ser transparentes para la capa de aplicación para que puedan optimizarse y evolucionar a lo largo del tiempo sin tener que realizar ningún cambio en la aplicación en sí.

Los resúmenes y agrupación/recopilación a los que se hace referencia en la literatura de data warehousing se crean en Oracle Database utilizando un objeto de esquema denominado Vista Materializada Materialized View (MV). Una característica llamada query rewrite reescribe automáticamente las consultas SQL para acceder a las summary tables cuando sea apropiado para que las vistas materializadas permanezcan transparentes para la aplicación.

La vista materializada no es más que una vista, definida con una sentencia SQL de Oracle, de la que además de almacenar su definición, se almacenan los datos que retorna, realizando una carga inicial y después cada cierto tiempo un refresco de los mismos.

Las vistas materializadas se encuentran comúnmente en la capa de acceso y rendimiento. Debido a que las MVs deben actualizarse y sincronizarse con los datos de la tabla base, son más apropiados para entornos o esquemas donde los datos no son muy volátiles o donde los datos se anexan continuamente. Las MVs también son apropiadas en los EDW que actualizan las capas de acceso y rendimiento utilizando una reconstrucción de esquema completa.

El diseño de vista materializada es una actividad que a menudo ocurrirá durante el proceso de mapeo lógico a físico. Además, es beneficioso evaluar continuamente la efectividad de las vistas materializadas porque es probable que los requisitos del negocio evolucionen con el tiempo, cambiando el tipo de consultas que el data warehouse deberá soportar. Oracle Database 12c nos proporciona un mecanismo para simplificar este proceso con el Summary Management.

Componentes del Summary Management:

- Mecanismos para definir vistas materializadas y dimensiones.
- Un mecanismo de actualización "refresh" para garantizar que todas las vistas materializadas contengan los datos más recientes.
- Una capacidad de reescritura de consultas "query rewrite" para reescribir de manera transparente una consulta y que use una vista materializada.
- El SQL Access Advisor, que recomienda vistas materializadas, particiones e índices a crear. Se requiere de licencia de Oracle Tuning Pack.
- El paquete TUNE_MVIEW, que le muestra cómo hacer que su vista materializada sea rápidamente actualizable y se utilice query rewrite. Se requiere de licencia de Oracle Tuning Pack.

Oracle Database In-Memory option

Hay que considerar aprovechar el almacenamiento de columnas en memoria (In-Memory column store) para optimizar el rendimiento en el nivel de acceso y rendimiento.

La opción Oracle Database In-Memory se introdujo en Oracle Database 12c Release 12.1.0.2 y requiere la licencia de la opción de base de datos "Database In-Memory".

Esta opción permite almacenar tablas o particiones en memoria en un formato columnar. El formato columnar permite que los escaneos se realicen mucho más rápido que los formatos tradicionales en disco para las consultas de tipo analítico.

Este almacén de columnas se ha integrado en la base de datos, por lo que las aplicaciones pueden usar esta característica de forma transparente sin ningún cambio. Un DBA simplemente tiene que asignar memoria al In-Memory column store. El optimizador conoce el In-Memory column store, de modo que cada vez que una consulta accede a objetos que residen en el almacén de columnas, el optimizador se percata que la consulta se podría beneficiar de su formato de columna y la envía allí directamente.

El In-Memory Transaction Manager (IM transaction manager) mantiene los datos columnares consistentes con la memoria buffer cache para que las aplicaciones no se vean afectadas por la adición del IM column store.

El IM column store se puede utilizar para simplificar la implementación física del data warehouse porque comúnmente reduce la necesidad de crear y mantener vistas materializadas o usar índices para la optimización del esquema en estrella; también hace que el procesamiento de datos sea más permisivo para diseños de esquemas subóptimos. Esta simplificación reduce los gastos generales de recursos asociados con el mantenimiento de índices y la administración de resúmenes. Menos índices y resúmenes dejarán más espacio para almacenar datos.

Gestión del sistema

Gestión de la carga de trabajo - Ejecución Paralela

Independientemente del propósito de su data warehouse el reto es siempre el mismo, acceso y proceso de grandes volúmenes de datos en una cantidad de tiempo extremadamente corto. La clave para conseguir un buen rendimiento en su data warehouse es aprovechar todos los recursos hardware disponibles: varias CPUs, varios canales de E/S, varias unidades de disco y arrays de almacenamiento, y grandes volúmenes de memoria. La ejecución en paralelo es una de las características clave, ya que le permitirá utilizar plenamente su sistema además debería utilizarse independientemente del modelo de datos que vaya a implementar. La ejecución en paralelo debe aprovecharse para todas las operaciones que hacen uso intensivo de recursos, incluyendo:

- Consultas complejas que acceden a grandes cantidades de datos
- Creación de índices en tablas grandes
- Recopilación de las estadísticas del optimizador
- Carga o manipulación de grandes volúmenes de datos
- Backups de la Base de datos

Este documento sólo se centrará en la ejecución en paralelo de SQL para consultas de gran tamaño. La ejecución de SQL en paralelo en Oracle Database se basa en los principios de un coordinador, a menudo llamado el Coordinador de Consulta - QC, para abreviar, y procesos paralelos.

El QC es el proceso que inicia la ejecución SQL en paralelo y los procesos paralelos son los procesos individuales que trabajan en paralelo. El QC distribuye el trabajo entre los procesos paralelos y puede tener que realizar una mínima parte del trabajo, sobre todo logístico, que no puede ejecutarse en paralelo. Por ejemplo, una consulta en paralelo con una operación SUM(), en la que se requiere la adición de los distintos sub-totales calculados por cada proceso paralelo.

El QC se identifica fácilmente en la ejecución paralela como “PX COORDINADOR”. El proceso que actúa como QC de una operación SQL paralela es el proceso de sesión de usuario en sí. Los procesos en paralelo se toman de un pool de procesos paralelos disponibles globalmente y se asignan a una determinada operación. Los procesos paralelos hacen todo el trabajo mostrado en un plan paralelo por debajo del QC.

Por defecto, Oracle Database está configurada para soportar la ejecución en paralelo out-of-the-box y se controla por dos parámetros de inicialización parallel_max_servers y parallel_min_servers. Si bien la ejecución en paralelo proporciona un marco muy potente y escalable para acelerar las operaciones SQL, no hay que olvidar utilizar algunas reglas de sentido común. Mientras que la ejecución en paralelo podría ganar con un incremento adicional de rendimiento, requiere más recursos y también podría tener efectos secundarios sobre otros usuarios u operaciones en el mismo sistema. En tablas/índices pequeños (hasta unos miles de registros; de hasta 10 bloques de datos) nunca debe estar habilitada la ejecución en paralelo. Las operaciones que sólo atacan a pequeñas tablas no se beneficiarán mucho de la ejecución en paralelo, pero utilizarán procesos paralelos que tendrán que estar disponibles para las operaciones de acceso a tablas de gran tamaño. Recuerde también que una vez que una operación se inicia con un cierto

grado de paralelismo (DOP), no hay manera de reducir su DOP durante la ejecución. Las reglas generales para la determinación del DOP apropiado para un objeto son:

- Los objetos más pequeños que 200 MB no deben usar ningún paralelismo
- Los objetos de entre 200 MB y 5 GB deben utilizar un DOP de 4
- Los objetos de más de 5GB deben usar un DOP de 32

No es necesario decir que la configuración óptima puede variar en su sistema, ya sea en el rango de tamaño o DOP - y muy dependientemente de la carga de trabajo, los requisitos de negocio, y su configuración de hardware.

Cuándo usar o no ejecución paralela entre instancias en RAC

Por defecto, Oracle Database permite la ejecución paralela entre nodos (ejecución en paralelo de una sola sentencia en la que participa más de un nodo). Como se mencionó en la sección de configuración balanceada, la interconnect en un entorno RAC debe tener el tamaño apropiado ya que la ejecución paralela entre nodos puede dar lugar a una gran cantidad de tráfico de interconnect. Si está utilizando un interconnect que soporta poca carga en comparación con el ancho de banda de E/S del servidor para el subsistema de almacenamiento, es posible que sea mejor limitar la ejecución en paralelo a un solo nodo o un número limitado de nodos. La ejecución paralela entre nodos no escala con un tamaño inferior que la interconnect. Desde Oracle Database 11gR1 en adelante, se recomienda utilizar los servicios de RAC para controlar la ejecución paralela en un clúster.

Monitorización de la carga de trabajo

Con el fin de tener una visión global de lo que está sucediendo en el sistema y establecer un baseline en el rendimiento esperado se deben tomar informes de estadísticas de base de datos de forma periódica (Statspacks y/o AWRs si éste último se encuentra licenciado). Sin embargo, cuando se trata de sistema de monitorización en tiempo real, lo mejor es empezar por comprobar si el sistema está utilizando una gran cantidad de recursos de la CPU o si está a la espera de un recurso en particular y en caso afirmativo averiguar cuál es ese recurso.

Se puede encontrar esta información utilizando las vistas de rendimiento V$, tales como V$sesion o mirando la pantalla principal de rendimiento de Oracle Enterprise Manager Database Control o Grid Control, que muestra un gráfico con los eventos espera que duran más tiempo. Si una parte significativa de la carga de trabajo consiste en instrucciones SQL que se ejecutan en paralelo, será usual ver un uso elevado de CPU y/o significativas esperas de usuario de E/S.

Si se revisaran los informes de Statspacks y/o AWR, es probable que se vieran eventos de espera de tipo PX en la parte superior o cerca de la parte superior de la lista de eventos de espera. Los eventos PX más comunes son de intercambio de mensajes (de datos) entre procesos paralelos y con el coordinador de la consulta. Es muy probable ver eventos como PX Deq Credit: send blkd, éste se debe a que un conjunto de procesos paralelos (productores o lectores de datos) están esperando a que los consumidores (otro conjunto de procesos paralelo) acepten datos.

Igualmente se pueden observar eventos de tipo PX Deq Credit: need buffer, que es causado por consumidores que están esperando a que los productores produzcan

datos. Los eventos de espera PX son inevitables, pero realmente no perjudican el rendimiento ya que puede considerar inactividad. Por lo general, en la ejecución paralela los eventos de espera causantes de un rendimiento lento del sistema no son muy característicos, sino más bien esperas introducidas por la carga de trabajo en paralelo, como la espera de E/S, o la utilización de la CPU. Un aumento en el número de eventos PX activos suele considerarse un síntoma de un problema de rendimiento en lugar de la causa. Por ejemplo, un aumento de las esperas PX Deq de crédito: need buffer (consumidores esperando a que los productores produzcan datos) es probable que indique un cuello de botella de E/S o un problema de rendimiento, ya que las operaciones de productores tienden a involucrar E/S de disco (por ejemplo, una full table scan paralela).

Si operaciones SQL paralelas forman cuello de botella de E/S por lo general será debido a que se ha alcanzado el throughput máximo (MB/s) en lugar del máximo de operaciones de E/S por segundo (IOPS).

Desde Oracle Database 11gR1 se ha introducido una nueva vista dinámica GV$SQL_MONITOR. Esta vista permite la monitorización en tiempo real de sentencias SQL de larga duración y de todas las sentencias SQL en paralelo sin ningún tipo de sobrecarga. A partir de 11.1.0.7, en Oracle Enterprise Manager Database Console (ahora en 12c reemplazada por la Oracle Enterprise Manager Database Express) también hay una interfaz gráfica para GV$SQL_MONITOR. La pantalla de SQL Monitoring muestra el plan de ejecución de una sentencia de larga duración o una sentencia que se ejecuta en paralelo, en tiempo casi real (el ciclo de actualización por defecto es de 5 segundos). Puede monitorizar lo que está pasando en el plan de ejecución que está siendo ejecutado así como si hay esperas.

Administrador de recursos

El Resource Manager de la base de datos Oracle (DBRM) le permite priorizar el trabajo dentro de una base de datos Oracle. Es muy recomendable utilizarlo si un sistema tiene límite de CPU, ya que protegerá a los usuarios o tareas con alta prioridad de ser afectados por trabajo de menor prioridad. Se proporciona esta protección mediante la asignación de tiempo de CPU a las diferentes tareas en función de su prioridad. Para utilizar DBRM tendrá que crear grupos de consumidores, que son grupos de usuarios formados en función de unas características dadas, por ejemplo username o rol.

A continuación, debe crear un plan de recursos donde se especifica cómo se distribuirán los recursos entre los distintos grupos de consumidores. Los recursos incluyen los porcentajes de tiempo de CPU, número de sesiones activas, y la cantidad de espacio disponible en el tablespace undo. También puede restringir la ejecución paralela a los usuarios que estén dentro de un grupo de consumidores. DBRM es el último factor decisivo para determinar el máximo grado de paralelismo, y ningún usuario en un grupo de consumidores (mediante un plan de recursos específico) nunca será capaz de ejecutar con un DOP mayor que el máximo del grupo de recursos. Por ejemplo, si su plan de recursos tiene la política de utilizar un DOP máximo de 4 y usted solicitara un DOP de 16, el SQL se ejecutará con un DOP de 4.

En un ejemplo de su uso, se puede restringir la ejecución en paralelo con un DOP de 4 de un plan de recursos denominado 'DW_USERS'. Como se mencionó anteriormente DBRM puede controlar el número máximo de sesiones activas para un grupo de recursos determinado. En este plan de recursos, 'DW_USERS ' el grupo

de consumidores tiene un límite máximo de 4 sesiones activas. Esto significa que es posible que "DW_USERS" tenga un consumo de recursos máximo de 4 (sesiones) x 4 (DOP) x 2 (conjuntos de esclavos) = 32 procesos paralelos.

Gestión de las estadísticas del optimizador

Saber cuándo y cómo recopilar estadísticas del optimizador se ha convertido en algo difícil de determinar. Sobre todo en un entorno data warehouse donde el mantenimiento de estadísticas puede obstaculizarse por el hecho de que, como el conjunto de datos aumenta el tiempo necesario para recopilar estadísticas también aumentará. Por defecto los paquetes DBMS_STATS se recopilarán globalmente (a nivel de tabla), a nivel de partición, y las estadísticas de sub-partición para cada una de las tablas de la base de datos. La única excepción es si existen sub-particiones hash. Las sub-particiones hash no necesitan estadísticas, ya que el optimizador puede derivar con exactitud las estadísticas necesarias de las estadísticas a nivel de partición, porque las particiones hash son aproximadamente del mismo tamaño, debido al algoritmo de hashing lineal.

Como se mencionó anteriormente la cantidad de tiempo que se tarda en recopilar estadísticas crecerá proporcionalmente al conjunto de datos, por lo que ahora se estará preguntando si el optimizador realmente necesita las estadísticas para todos los niveles de una tabla o si el tiempo se podría mejorar saltando uno o más niveles La respuesta es "no" ya que el optimizador utiliza las estadísticas de uno o más niveles en diferentes situaciones.

El optimizador utiliza las estadísticas de nivel global o de tabla si una o más de sus consultas se componen de dos o más particiones. El optimizador utilizará las estadísticas de nivel partición si sus consultas hacen eliminación de partición, de tal manera que sólo una partición es necesaria para responder a cada consulta. Si sus consultas se componen de dos o más particiones, el optimizador utilizará una combinación de estadísticas de nivel global y de partición.

Las estadísticas globales son, con mucho, las estadísticas más importantes, pero también necesitan más tiempo para recopilar porque requieren una full table scan. Sin embargo, desde Oracle Database 11g esta cuestión se ha abordado con la introducción de estadísticas incrementales globales. Normalmente con tablas particionadas, las nuevas particiones se agregan y los datos se cargan en estas nuevas particiones. Una vez que la partición se ha cargado completamente, las estadísticas de nivel de partición deben recogerse y las estadísticas globales necesitan ser actualizadas para reflejar los nuevos datos. Si el valor de INCREMENTAL de la tabla de particiones se establece en TRUE, y el parámetro DBMS_STATS GRANULARTY se establece a AUTO, Oracle recopilará estadísticas de la nueva partición y actualizará las estadísticas de la tabla global recorriendo sólo las particiones que se han modificado y no toda la tabla. A continuación se presentan los pasos necesarios para hacer uso de las estadísticas incrementales globales:

```
SQL> exec dbms_stats.set_table_prefs('SH', 'SALES',
'INCREMENTAL','TRUE');

SQL> exec dbms_stats.gather_table_stats( Owner=>'SH',
Tabname=>'SALES', Partname=>'23_MAY_2008', Granularity=>'AUTO');
```

Las Estadísticas Incrementales Globales funcionan almacenando una sinopsis de cada partición en la tabla. Una sinopsis son metadatos estadísticos de la partición y

de las columnas de la partición. Cada sinopsis se almacena en el tablespace SYSAUX y ocupa aproximadamente 10KB. Las estadísticas globales se generan mediante la agregación de la sinopsis de cada partición, eliminando así la necesidad de full table scan. Cuando una nueva partición se añade a la tabla sólo tendrá que recopilar las estadísticas para la nueva partición. Las estadísticas globales se actualizarán automáticamente mediante la agregación de la sinopsis de las particiones nuevas a las sinopsis de las particiones existentes.

Pero ¿qué pasa si no está utilizando Oracle Database 11g o superior y no puede permitirse el lujo de recopilar estadísticas a nivel de partición (por no hablar de estadísticas globales) después de la carga de los datos? En Oracle Database 10g (10.2.0.4), puede utilizar el procedimiento DBMS_STATS.COPY_TABLE_STATS. Este procedimiento le permite copiar las estadísticas de una [sub] partición existente a la nueva [sub] partición y modificar las estadísticas para tener en cuenta la partición de datos adicionales (por ejemplo, el número de bloques, el número de filas). Se establece un límite alto de partitioning para la nueva partición con el valor máximo de la primera columna de partición y un límite alto de partitioning de la partición anterior como el valor mínimo de la primera columna de partición de una tabla de rango con particiones. Para una tabla particionada por lista encontrará el máximo y mínimo de la lista de valores.

Frecuencia de la recopilación de estadísticas

Si se utiliza la tarea de estadísticas automática o dbms_stats.gather_schema_stats con la opción "GATHER AUTO", Oracle sólo recogerá estadísticas a nivel global si la tabla ha cambiado más de un 10% o si las estadísticas globales no han sido recogidas. Las estadísticas a nivel de partición siempre se recogerán si se han perdido. Para la mayoría de las tablas esta frecuencia es buena.

Sin embargo, un entorno data warehouse es un escenario en el que este caso no se da. Si a una partición de una tabla se le están añadiendo constantemente nuevas particiones y luego los datos se cargan en una nueva partición y los usuarios comienzan inmediatamente consultar los nuevos datos, entonces es posible obtener una situación en la que una consulta de usuario final da un valor en uno de los predicados de la cláusula where que está fuera del rango de la columna [min, max] de acuerdo a las estadísticas del optimizador. Para los valores predicado fuera del rango de estadísticas [min, max] el optimizador prorratea la selectividad de ese predicado basándose en la distancia entre el valor y el máximo (suponiendo que el valor es superior al máximo). Esto significa que, cuanto más lejano esté el valor del valor máximo, menor será la selectividad, lo que puede dar lugar a planes de ejecución menos óptimos.

Se puede evitar esta situación de "fuera de rango", mediante el uso de nuevas estadísticas incrementales globales o el procedimiento de estadísticas de copia de tabla.

Conclusión

Con el fin de garantizar la obtención de un rendimiento óptimo en su data warehouse y asegurar que escala tal y como aumenta el conjunto de datos, necesita tener en cuenta los siguientes fundamentos:

- Hardware:
  - La configuración hardware debe estar balanceada debidamente.
  - Se debe lograr la velocidad de procesamiento CPU y el throughput de E/S necesarios y requeridos para soportar la carga máxima del sistema.
  - Se debe lograr la disponibilidad y rendimiento deseados.
  - Hay que considerar las implicaciones del escalado futuro.

- Gestionar y almacenar grandes volúmenes de datos de manera efectiva:
  - Usar *partitioning*.
  - Usar *compression*.

- El modelo de datos, si es 3FN deberían conseguirse con partition-wise joins o si es un esquema en estrella debería usarse star transformation

- Carga y transformación de datos de forma eficiente:
  - Usar tablas externas para carga de conjuntos de datos
  - Utilizar el procesamiento de datos basado en conjuntos "*set-based*" siempre que sea posible.
  - Diseñar y desarrollar la capacidad de escalado utilizando el paralelismo.
  - Considerar qué es lo mejor en su situación: equilibrando las necesidades de rendimiento, la visibilidad de los datos y los niveles de servicio, los patrones de llegada de datos y la facilidad de uso general.

- Optimización del rendimiento:
  - Usar *partitioning*.
  - Usar paralelismo.
  - Optimizar el esquema 3FN con las tres Ps: Power, Partitioning y Paralell Execution.
  - Optimizar el esquema en estrella usando optimizaciones físicas tales como la optimización del esquema en estrella, star transformation, In-Memory column stores, Attribute Clustering, zone maps y OLAP.
  - Usar vistas materializadas para consultas costosas repetitivas

- Gestión del sistema:
  - Usar la ejecución paralela
  - Utilizar Resource Manager (DBRM)
  - Monitorización del sistema
  - Realizar una buena gestión de las estadísticas del optimizador