Unidad 5

Monitorización y optimización

Administración de sistemas gestores de bases de datos

5

Índice

Introducción

En esta fase, con el sistema instalado, configurado y con las medidas de seguridad necesarias, es el momento de comenzar con la explotación del sistema. Las tareas habituales que tiene que realizar el DBA en fase de explotación del sistema son:

Arrancar o parar el sistema, monitorizarlo, hacer copias de seguridad periódicas y custodiarlas y optimizar el acceso y la descarga de los objetos y datos más consultados. Vamos a definir algunos de los conceptos necesarios para gestionar las tareas relacionadas con la monitorización y optimización del sistema.

- **Plan de ejecución:** Ruta que va a seguir el SGBD para acceder a los resultados de una consulta.
- **Caché de consultas:** Almacén donde se guardan los planes de ejecución óptimos de las consultas que se ejecutan con más frecuencia.
- **Tabla particionada:** Una tabla que ha sido dividida en diferentes fragmentos que el sistema interpreta como tablas individuales con el objetivo de acceder más rápido a la información.
- **Fragmentación:** Se produce cuando aparecen espacios vacíos en los soportes de almacenamiento de la información en las operaciones realizadas con los datos. Se ocupa más espacio del necesario y ralentizan las búsquedas de información.
- **Índice balanceado:** Es un índice con estructura de árbol. Está balanceado si el número hijos de cada nodo es similar.
- **Log:** Archivo que contiene una secuencia de eventos registrados en el sistema.
- **Optimizador:** Herramienta del SGBD que calcula los costes de ejecución de las consultas, seleccionado el plan de ejecución de menor coste.

Al finalizar esta unidad

- Aprenderemos a establecer los mecanismos de seguridad del SGBD que nos van a permitir garantizar la integridad y confidencialidad de los datos.
- Conoceremos las diferentes herramientas de los SGBD que permiten garantizar la seguridad del sistema.
- Seremos conscientes de lo importante que es definir y mantener unas políticas de copias de seguridad y restauración.
- Comprenderemos la utilidad de las herramientas incluidas en el SGBD y que contribuyen a garantizar las medidas impuestas por la normativa de protección de datos.

5.1. Monitorización

Es conveniente ser selectivo a la hora de hacer uso de las herramientas de monitorización del SGBD. Hay que tener en cuenta que este tipo de herramientas cargan al sistema de trabajo, por lo que habrá que filtrar qué es lo que se necesita monitorizar, y en qué nivel de automatización es posible.

Para realizar las tareas de monitorización se utilizan tres herramientas: monitor de rendimiento, logs de ejecución y diccionario de datos.

5.1.1. Monitor de rendimiento

Permite consultar el uso de recurso que hace el sistema (CPU, memoria, red, discos, etc.) y prevenir posibles problemas. El monitor de rendimiento puede informar sobre diferentes aspectos como:

Métricas de rendimiento

El rendimiento se puede evaluar en base a diferentes métricas que se pueden visualizar en el monitor de rendimiento. Estas métricas se pueden definir. Algunas de las más comunes son:

- **Transacciones por unidad de tiempo:** mide la carga de trabajo de la base de datos.
- **Tiempo de respuesta:** tiempo mínimo y promedio en la ejecución de una tarea o conjunto de tareas.
- **Escalabilidad:** mide lo preparado que está el sistema para recibir picos de carga.
- **Concurrencia:** carga del servidor en momentos de máxima actividad.

Bloqueos

A través del monitor de rendimiento, es posible comprobar si se han dado situaciones de interbloqueos u otro tipo de situaciones que bloqueen los recursos del sistema. Se puede registrar si ocurren con frecuencia y en qué tipo de objetos.

Detección de procesos de alto consumo

Permite detectar si hay algún tipo de proceso que esté haciendo un uso excesivo de los recursos disponibles, comprometiendo el rendimiento del sistema. En caso necesario, puede detener el proceso.

Consumo de recursos

Sirve para comprobar qué tipo de consultas tienen un alto consumo de recursos, de forma que se puede medir el rendimiento si se aplican cambios con el optimizador.

Alertas

Permiten generar eventos que se activen cuando ocurre una condición de activación, por ejemplo, si falta espacio en disco o si un proceso está consumiendo más recursos de los esperados. Se podría notificar al DBA

5.1.2. Registro de errores

Este tipo de logs son diferentes a los que se registran en el cuaderno de bitácora. La actividad que se registra en este caso está relacionada con el funcionamiento del servidor:

- **Paradas/arranques**
- **Operaciones relevantes**
- **Errores o avisos**
- **Registro de consultas pesadas**

El nivel de detalle de los logs es configurable. En modo "debug", se guarda la información detallada de las operaciones del SGBD, no es recomendable, salvo para casos puntuales, pues puede consumir recursos en exceso. Los otros dos modos son "warning" y "error" que guardan registros de eventos relevantes o errores en la aplicación respectivamente.

5.1.3. Diccionario de datos

El registro que genera la monitorización guarda información en el diccionario de datos. Si esta información es demasiado detallada o frecuenta, se corre el riesgo de que el rendimiento del sistema quede afectado.

PARA TENER EN CUENTA...

Las herramientas de monitorización que hacen uso de gráficas para mostrar información en tiempo real van a consumir más recursos del sistema debido a que lanzan múltiples peticiones al sistema en forma de consultas al diccionario de datos.

Aunque elaborar las consultas al diccionario de datos no es sencillo, se pueden reutilizar y elaborar informes cuando sea necesario.

5.2. Monitorización en Oracle

Con la instalación del gestor de Oracle no viene una herramienta de monitorización, aunque existen diferentes alternativas que nos permiten hacerlo.

5.2.1. Monitor de rendimiento

La herramienta más utilizada en Oracle como monitor de rendimiento es Oracle EM, aunque también es posible utilizar el diccionario de datos mediante SQL Developer.

Enterprise Manager

Está pensada para versiones de pago de Oracle. El Enterprise Manager de Oracle ofrece una amplia gama de funciones de monitorización de la actividad de la base de datos. Algunas de estas funciones son:

1. **Monitor de rendimiento:** permite examinar el uso que el sistema hace de los recursos.
2. **Gestionar incidencias:** picos de consumo de recursos, errores, registrar incidencias.
3. **Tareas programadas:** Se pueden programar tareas, así como registrar para analizar las ejecución de dichas tareas.
4. **Definir y consultar informes** con información sobre el rendimiento y el estado de la bbdd.
5. **Definir y generar alertas o eventos** y definir acciones correctivas sobre las mismas.

La versión express solamente permite monitorizar el uso de recursos y el rendimiento de las consultas. Para ejecutar EM, primero se ha de configurar el servidor web. Desde sqlplus, hay que verificar que está disponible el puerto 5500:

```
SQL> SELECT DBMS_XDB_CONFIG.gethttpport FROM dual;

GETHTTPPORT
-----------
          0

SQL> SELECT DBMS_XDB_CONFIG.gethttpsport FROM dual;

GETHTTPSPORT
------------
        5500

SQL> EXEC DBMS_XDB_CONFIG.sethttpsport(5500);

Procedimiento PL/SQL terminado correctamente.

SQL>
```

Imagen 1. Preparando el servidor web

Una vez preparado el servidor web, podremos conectarnos a EM, escribiendo la dirección en el navegador: https://localhost:5500/em.

Posiblemente, el navegador requiera autorización o añadir excepciones al sitio por el certificado de seguridad.

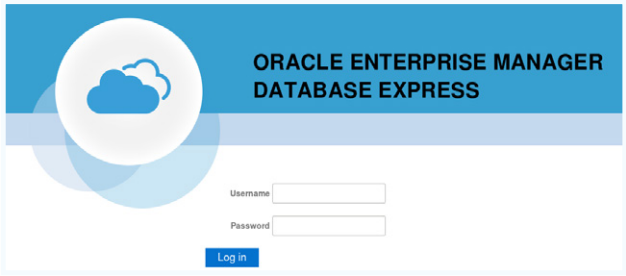

Imagen 2. Login en Oracle Enterprise Manager Database Express (EM)

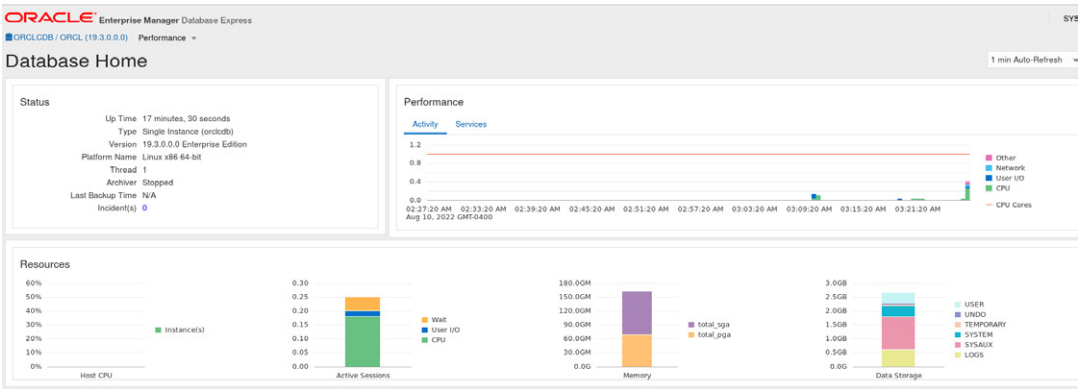

Imagen 3. Monitor de rendimiento a través de la web EM

Diccionario de datos

La información del diccionario de datos relativa al rendimiento se encuentra en vistas dinámicas de rendimiento a las que se puede hacer referencia con el prefijo "v$".

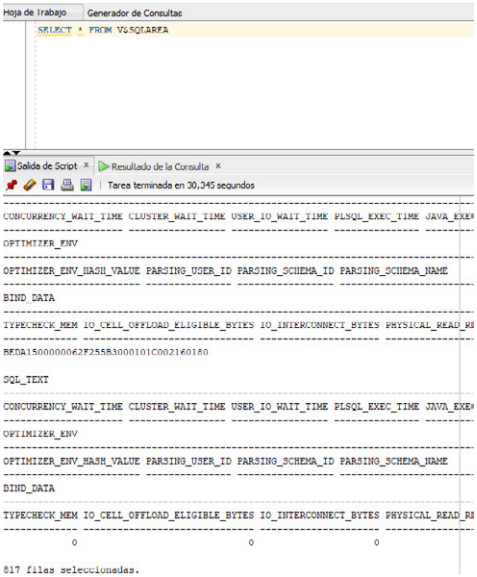

Imagen 4. Consulta a la vista V$SQLAREA

SQL Developer

Permite consultar información de monitorización del uso del SGBD, consultar el histórico de consultas, tiene informes pre-definidos para desarrollo, gestión y seguridad.

5.2.2. Registro de errores

Oracle almacena la información de logs del sistema en el repositorio automático de diagnóstico (ARD). Consiste en una estructura de directorios almacenada en el directorio \diag para poder diagnosticar el estado de la base de datos incluso cuando está parada.

```
c:\app\Administrador\product\21c\diag>dir /b/s *.log
c:\app\Administrador\product\21c\diag\clients\user_OracleServiceXE\host_1737181618_110\trace\sqlnet.log
c:\app\Administrador\product\21c\diag\rdbms\globaldb\globaldb\trace\alert_globaldb.log
c:\app\Administrador\product\21c\diag\rdbms\globaldb\globaldb\trace\attention_globaldb.log
c:\app\Administrador\product\21c\diag\rdbms\orcl\orcl\trace\alert_orcl.log
c:\app\Administrador\product\21c\diag\rdbms\orcl\orcl\trace\attention_orcl.log
c:\app\Administrador\product\21c\diag\rdbms\xe\xe\log\hcs_xe.log
c:\app\Administrador\product\21c\diag\rdbms\xe\xe\trace\alert_xe.log
c:\app\Administrador\product\21c\diag\rdbms\xe\xe\trace\attention_xe.log
c:\app\Administrador\product\21c\diag\tnslsnr\WIN-TIPCBEBVF20\listener\trace\listener.log
c:\app\Administrador\product\21c\diag>dir /b/s *.xml
c:\app\Administrador\product\21c\diag\clients\user_OracleServiceXE\host_1737181618_110\alert\log.xml
c:\app\Administrador\product\21c\diag\rdbms\globaldb\globaldb\alert\log.xml
c:\app\Administrador\product\21c\diag\rdbms\orcl\orcl\alert\log.xml
c:\app\Administrador\product\21c\diag\rdbms\xe\xe\alert\log.xml
c:\app\Administrador\product\21c\diag\rdbms\xe\xe\log\hcs\log.xml
c:\app\Administrador\product\21c\diag\tnslsnr\WIN-TIPCBEBVF20\listener\alert\log.xml

c:\app\Administrador\product\21c\diag>
```

Imagen 5. Ejemplos de registros en .log y .xml

La vista v$DIAG_INFO nos va a indicar la ubicación del repositorio.

```
SELECT * FROM V$DIAG_INFO vdi
```

1000 Table (read only)

INST_ID NUMBER	NAME VARCHAR2(64)	VALUE VARCHAR2(512)
1	Diag Enabled	TRUE
1	ADR Base	C:\APP\ADMINISTRADOR\PRODUCT\21C
1	ADR Home	C:\APP\ADMINISTRADOR\PRODUCT\21C\diag\rdbms\xe\xe
1	Diag Trace	C:\APP\ADMINISTRADOR\PRODUCT\21C\diag\rdbms\xe\xe\trace
1	Diag Alert	C:\APP\ADMINISTRADOR\PRODUCT\21C\diag\rdbms\xe\xe\alert
1	Diag Incident	C:\APP\ADMINISTRADOR\PRODUCT\21C\diag\rdbms\xe\xe\incident
1	Diag Cdump	C:\app\Administrador\product\21c\diag\rdbms\xe\xe\cdump
1	Health Monitor	C:\APP\ADMINISTRADOR\PRODUCT\21C\diag\rdbms\xe\xe\hm
1	Default Trace File	C:\APP\ADMINISTRADOR\PRODUCT\21C\diag\rdbms\xe\xe\trace\xe_ora_6684.trc
1	Active Problem Count	0
1	Active Incident Count	0
1	ORACLE_HOME	C:\app\Administrador\product\21c\dbhomeXE

Imagen 6. Vista

Los principales archivos y directorios donde se ubican los ficheros de registros de logs de error son:

- `alert.log:` archivo en formato xml que registra por orden cronológico las secuencias de errores derivados de la operativa diaria del SGBD. Hay un archivo por cada instancia alertXE.log, aunque tienen uno común alert.log
- `cdmump:` en este directorio se registran las trazas que genera el núcleo del sistema.
- `Trace:` directorio que registra las trazas de los procesos en segundo plano del servidor, del listener y otros programas del usuario.
- `Incident:` en este directorio se genera un subdirectorio por cada incidente.
- `Others:` Puede haber más subdirectorios donde se almacenen incidencias, informes de monitorización, alertas u otro tipo de información.

5.3. Optimización

La optimización del servidor donde esté instalado el SGBD, requiere realizar tareas en varios niveles (red, equipo, etc.). Es necesario tener privilegios de administrador de sistemas para poder modificar algunos parámetros.

5.3.1. Optimización del entorno

Para poder realizar optimizaciones en el SGBD, es conveniente cerciorarse en primer lugar de que el entorno del servidor es apropiado. Los aspectos a tener en cuenta son fundamentalmente el sistema operativo y la conexión de red.

Sistema operativo

Desde los sistemas operativos se puede hacer uso de herramientas de monitorización de los recursos del equipo. Algunas están incluidas en el propio sistema y otras es necesario instalarlas de forma externa. Estas herramientas pueden servir para analizar algunas métricas de rendimiento del SGBD.

Conectividad

Si el servidor tiene múltiples tarjetas de red y núcleos, es posible definir qué tipo de paquetes se procesas por cada núcleo. De esta forma se pueden evitar bloqueos producidos por compartir recursos de CPU.

5.3.2. Optimización del SGBD

Dentro del gestor, las operaciones que podremos ejecutar para optimizar el sistema están orientadas al dimensionamiento físico de los componentes del sistema (almacenamiento interno). Tendremos en cuenta el tamaño de los bloques de datos, tamaño y ubicación de los archivos de datos, tener controlados los procesos en segundo plano y gestionar el almacén temporal.

Bloques de datos

Aunque la información en las tablas se almacena lógicamente en filas, físicamente lo hacen en bloques de datos. Mantener un apropiado equilibrio entre el tamaño de esos bloques y el número de archivos es crucial. Por ejemplo, si una tabla tiene un tamaño horizonal grande (muchas columnas), y el tamaño de los bloques es pequeño, va a requerir más de un bloque (fragmentación), por lo tanto, cuando se acceda a la información de la tabla, se va a requerir más de una operación de acceso a disco, penalizando en rendimiento. Por lo tanto, las tablas con un número de columnas elevado tendrán mayor tamaño de bloque.

Ficheros de datos

Una operación de acceso a disco es un proceso que hace uso de un recurso físico del servidor y suele ser una operación "lenta" y puede suponer un cuello de botella. Si toda la base de datos se almacena en el mismo disco físico, los accesos son secuenciales y no se pueden paralelizar.

Cuando una tabla tiene un tamaño considerable y ha de ser particionada, se recomienda separar los datos en varios discos, así como los índices para poder acceder en paralelo a los datos.

Un SGBD en entornos de producción, suelen utilizar almacenamiento RAID o NAS, por lo que suelen tener más de un disco. Una recomendación genérica sería utilizar tantos discos como CPU tenga el servidor, de forma que se puedan ejecutar las operaciones en paralelo en distintos discos.

Procesos en segundo plano

Algunos SGBD realizan tareas de autogestión del tamaño de los ficheros como el incremento de tamaño o compactación. Si estos procesos se ejecutan en momentos de alta carga del servidor, puede suponer un problema de rendimiento. En ese caso, es muy recomendable controlar estos procesos manualmente y ejecutarlos en horarios de baja carga.

Almacenamiento temporal

Las transacciones que se van ejecutando en el SGBD, necesitan un espacio de almacenamiento durante el proceso de inicio hasta su finalización. Si el espacio se agota, el SGBD no permitirá la ejecución de más transacciones hasta que no se libere. Debemos reservan una cantidad de espacio apropiada. Por ejemplo, los datawarehouse realizan pocas transacciones, pero ocupan muchos recursos, por lo que el espacio temporal ha de ser grande para almacenar consultas parciales. En entornos operacionales (ERP, CRM, ...), las transacciones ocupan poco espacio, pero es importante que no se produzcan bloqueos.

5.3.3. Optimización de la base de datos

Hay varias estrategias que debemos adoptar a la hora de realizar un diseño físico apropiado de nuestro modelo de datos:

Tipos de datos

Cuando se diseñan las tablas, debe elegirse un tipo de datos apropiado para cada campo, sin que se produzca desbordamiento, pero sin utilizar tipos que reservan una gran cantidad de espacio que no se va a llenar. Además del tamaño apropiado de los tipos, debe evitarse en la medida de lo posible las operaciones de conversión de tipos, que afectarán al rendimiento del sistema.

Campos calculados

Debe evitarse el uso de campos calculados en situaciones en las que el cálculo no será modificado posteriormente con la introducción de nuevas tablas.

Desnormalización

Hay consultas que se realizan con frecuencia y que pueden ser combinaciones de join muy pesadas con varias tablas complementarias. Una estrategia común, especialmente en diseño de datawarehouse, es romper esa normalización e introducir información redundante en una tabla destino con información sobre la consulta. Esto aumenta la velocidad de las consultas, pero ocupa más espacio en disco.

Particionamiento

Cuando una tabla adquiere cierto tamaño, es conveniente dividirla en varios segmentos (particiones). Al acceder a una sola partición, no es necesario procesar toda la tabla, se permite que una tabla ocupe mayor espacio y además se pueden realizar accesos a la tabla en paralelo. En el caso de eliminar o archivar información antigua, solamente sería necesario bloquear una partición durante la operación.

Desfragmentación

Cuando se ha movido o eliminado información de una tabla, es posible que se hayan dejado huecos en los bloques, que desperdician espacio en disco. La operación de compactar la tabla para eliminar estos huecos, se llama desfragmentación.

Balanceo de índices

Cuando la estructura de árbol del índice se empieza a desnivelar, por causas como borrado masivo de filas, el índice podría quedar desbalanceado de forma que pierde su eficacia. Al reconstruir un índice, éste quedaría desfragmentado.

5.3.4. Índices

Los índices permiten acelerar las consultas que seleccionan una proporción pequeña de los registros de una tabla. Si bien esta solución es muy efectiva, también tiene sus desventajas. Por ejemplo, si se realizan muchas operaciones de inserción sobre una tabla, el índice de esta debe ser actualizado, consumiendo recursos extra. Por lo tanto, un índice lleva cierto mantenimiento para que no pierda su efectividad ni penalice al sistema.

Columnas

A la hora de seleccionar las columnas sobre las que crear los índices, hay que tener en cuenta ciertos criterios.

Una buena estrategia, en general suele ser crear índices de columnas que habitualmente se usan como filtros en SELECT o WHERE. Si estas columnas tienen valores poco repetidos en las diferentes filas, también es un buen criterio. Las claves primarias y foráneas suelen llevar índices creados automáticamente.

Por el contrario, debemos evitar crear índices en tablas pequeñas (caben en memoria), columnas con datos nulos o cuyos valores se modifican con mucha frecuencia. Si se crean índices sobre muchas columnas de forma simultánea, aumenta el coste de mantenimiento, su espacio e incluso pierden su eficacia.

El orden del índice debe escogerse cuidadosamente, pues el primer campo determina el orden de las entradas del índice. Las búsquedas por el segundo campo del índice requieren recorrer el índice completo.

5.3.5. Tipos de índices

Cada sistema implementa de forma diferente los índices. La decisión sobre el tipo de índice a crear dependerá del tipo de sistema gestor que estemos utilizando. Se pueden clasificar por su organización o por su estructura.

Organización

- **Agrupados:** Las tablas se ordenan por los campos definidos en el índice. No ocupan espacio adicional y no se puede definir más de un índice de este tipo para cada tabla. Es recomendable para aplicarlo sobre campos que no varían, pues de lo contrario se requiere reordenar la tabla, con el consiguiente consumo de recursos. Los sistemas gestores lo usan para las claves primarias.
- **No agrupados:** se crean como una estructura con punteros a los datos de la tabla y ocupan espacio adicional. Se puede crear más de uno por tabla y son recomendados para acceder a parte de una tabla, enlazar con join o consultas agregadas. También funcionan bien en consultas que usan where o cuando las columnas no tengan muchos valores únicos.

Estructura

- **Agrupados:** Un índice agrupado es un tipo de índice de tipo único en el que los valores de la columna indexada se almacenan en orden. Los índices agrupados se usan a menudo para mejorar el rendimiento de las consultas en las que se hacen referencia a los valores de la columna indexada.
- **B-Tree:** es una estructura de datos en la que cada nodo contiene una clave y un puntero a un hijo. Cada nodo del índice B-tree contiene un valor de clave único. Las claves se almacenan en orden, de tal forma que las claves menores se encuentran en los nodos hijos más a la izquierda, mientras que las claves mayores se encuentran en los nodos hijos más a la derecha. Los índices B-tree se pueden usar para implementar índices de tipo único y de tipo multiple**.**
- **Bitmap:** Se implementan con vectores de valores binarios. Para cada valor del campo, se crea un vector que va a tener un elemento por cada fila de la tabla. Si en una fila el valor del campo coincide con el del vector, se almacena 1, sino un 0. Son ideales para columnas con pocos cambios y poca variedad de valores (datawarehouse).
- **Hash:** Un índice hash es una estructura de datos que permite almacenar y recuperar valores de una tabla de forma más eficiente. Un índice hash se usa a menudo para mejorar el rendimiento de las consultas en las que se hacen referencia a los valores de la columna indexada. Las ventajas de usar un índice hash incluyen la eficiencia y la flexibilidad. Los inconvenientes de usar un índice hash incluyen la complexidad y el costo.

5.3.6. Tipos de índices: ejemplos

```
SELECT * FROM HR.EMPLEADOS;
```

EMPLOYEE_ID NUMBER(6)	FIRST_NAME VARCHAR2(20)	SALARY NUMBER(8, 2)	DEPARTMENT_ID NUMBER(4)
100	Steven	24000	90
101	Neena	17000	90
102	Lex	17000	90
103	Alexander	9000	60
104	Bruce	6000	60
105	David	4800	60
106	Valli	4800	60
107	Diana	4200	60
108	Nancy	12008	100
109	Daniel	9000	100
110	John	8200	100
111	Ismael	7700	100
112	Jose Manuel	7800	100
113	Luis	6900	100

Record 1 of 14

Imagen 7. Tabla de ejemplo EMPLEADOS

Partiendo de la tabla empleados (extraída de los datos de ejemplo del esquema HR de Oracle), un índice agrupado sería la clave primaria (EMPLOYEE_ID), define el orden de los datos y no ocupa espacio.

Para optimizar la consulta SELECT * FROM EMPLEADOS WHERE FIRST_NAME = 'Valli', se puede crear un índice de tipo B-tree sobre la columna FIRST_NAME.

En este tipo de índices, se intenta evitar que una consulta tenga que recorrer secuencialmente todos los datos cuando se consulta, un valor extremo.

FIRST_NAME VARCHAR2(20)	SALARY NUMBER(8, 2)
Alexander	9000
Bruce	6000
Daniel	9000
David	4800
Diana	4200
Ismael	7700
John	8200
Jose Manuel	7800
Lex	17000
Luis	6900
Nancy	12008
Neena	17000
Steven	24000
Valli	4800

Imagen 8. Índice B-Tree (first_name)

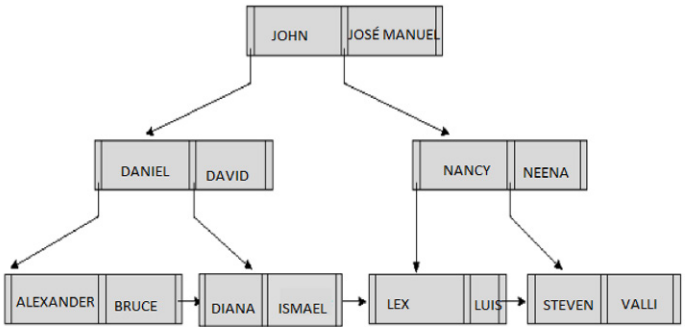

Imagen 9. Índice B-tree (nodos first_name)

Para consultar 'Valli', se accedería al nodo central, como es superior iría de derecha hacia abajo en tres saltos solamente llegaría al dato, mientras que si tiene que recorrer secuencialmente toda la tabla daría 14 saltos.

Siguiendo con el ejemplo anterior, crearemos un índice bitmap para las columnas GENDER y DEPARTMENT_ID

					índice género		índice dpto		
EMPLOYEE_ID	FIRST_NAME	SALARY	DEPART_ID	GENDER	male	female	60	90	100
100	Steven	24000	90	male	1	0	0	1	0
101	Neena	17000	90	female	0	1	0	1	0
102	Lex	17000	90	male	1	0	0	1	0
103	Alexander	9000	60	male	1	0	1	0	0
104	Bruce	6000	60	male	1	0	1	0	0
105	David	4800	60	male	1	0	1	0	0
106	Valli	4800	60	female	0	1	1	0	0
107	Diana	4200	60	female	0	1	1	0	0
108	Nancy	12008	100	female	0	1	0	0	1
109	Daniel	9000	100	male	1	0	0	0	1
110	John	8200	100	male	1	0	0	0	1
111	Ismael	7700	100	male	1	0	0	0	1
112	Jose Manuel	7800	100	male	1	0	0	0	1
113	Luis	6900	100	male	1	0	0	0	1

Vemos como en la tabla anterior, las columnas de la derecha (índice) forman un vector de ceros y unos. Para localizar empleado, mujer del dpto. 90, bastaría con realizar una operación AND entre las columnas del índice female y "90".

5.3.7. Optimización de consultas

Cuando estamos realizando una consulta mediante el lenguaje SQL, le indicamos al sistema qué queremos conseguir, pero es el sistema el que decide cómo hacerlo.

El flujo de ejecución es el siguiente:

1. El sistema analiza sintácticamente el código de la sentencia SQL
2. Se comprueba que el cruce de tablas es correcto (binding)
3. Se busca el plan de ejecución óptimo. Para ello, hará uso de los índices, restricciones, estadísticas, etc. El plan con menor coste se almacenará en la caché de consultas por si es necesario volver a recalcularlo.
4. Ejecución del plan con menor coste.

Estadísticas

Las estadísticas de las tablas son guardadas en el diccionario de datos. Inicialmente, la información que se almacena es la de las características de la tabla (índices, columnas, etc.). La información sobre bloques vacíos, tamaño promedio, distribución de los datos, etc. no se creará hasta que no se haga uso de las estadísticas de la tabla. Si no se actualizan las estadísticas, el plan de ejecución puede que no sea óptimo.

PARA TENER EN CUENTA...

Es recomendable antes de optimizar una consulta, actualizar las estadísticas de la tabla, pues es un proceso menos drástico que la modificación o creación de índices, tablas temporales, etc.

Reescritura de consultas

Aunque la sintaxis de nuestra consulta sea correcta, antes de proceder a modificar la estructura de la información, debemos asegurarnos de que nuestra consulta está optimizada en cuanto a código y si no es así, proceder a reescribir la consulta.

Cuando reescribimos una consulta, buscamos un consumo de recursos menor, y una mayor velocidad de respuesta. Para ello evitaremos recorrer toda la tabla siempre que sea posible. Simplificaremos, reduciendo el tamaño de las tablas que intervengan en la consulta, que se crucen con otras tablas, de forma que se filtre antes del cruce para obtener el mismo resultado.

Algunos ejemplos de optimización:

- **Sustituir OR por UNION:** Si se realiza una consulta sobre dos tablas con dos condiciones, una de cada tabla, el optimizador primero une las tablas, guarda los datos en una tabla temporal y aplica los filtros. Los índices definidos sobre las columnas que han sido filtradas no los utilizará. Si disponemos de índices en las tablas cuya condición queremos filtrar, es preferible aplicar primero los filtros para aprovechar los índices y luego realizar la unión:

```
SELECT
E.FIRST_NAME, D.DEPARTMENT_NAME
 FROM EMPLOYEES e, DEPARTMENTS d
 WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
 AND (E.FIRST_NAME = 'Valli' OR D.DEPARTMENT_NAME = 'IT')
```

```
SELECT
E.FIRST_NAME, D.DEPARTMENT_NAME
 FROM EMPLOYEES e, DEPARTMENTS d
 WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
 AND (E.FIRST_NAME = 'Valli')
 UNION
 SELECT
E.FIRST_NAME, D.DEPARTMENT_NAME
 FROM EMPLOYEES e, DEPARTMENTS d
 WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
AND D.DEPARTMENT_NAME = 'IT'
```

Imagen 10. La segunda consulta estaría optimizada si las columnas FIRST_NAME y DEPARTMENT_NAME tienen sus índices respectivos.

- **IN-ANY-ALL vs Exists:** En la instrucción SQL, IN y EXISTS tienen la misma función, pero cuando se utilizan con una cláusula de subconsulta, IN y EXISTS tienen un rendimiento diferente. IN se evalúa antes de ejecutar la subconsulta, mientras que EXISTS se evalúa después de ejecutar la subconsulta. Esto significa que, cuando se usa con una cláusula de subconsulta, IN es más lento que EXISTS. Sin embargo, IN tiene una ventaja sobre EXISTS en que puede usarse con un índice de columna, lo que mejora el rendimiento. Por lo tanto, es mejor evitar el uso de IN con una cláusula de subconsulta. En su lugar, debe usar EXISTS.

```
SELECT * FROM EMPLOYEES e
WHERE E.DEPARTMENT_ID IN
(SELECT DEPARTMENT_ID FROM DEPARTMENTS d WHERE D.DEPARTMENT_NAME = 'Sales');

SELECT * FROM EMPLOYEES e
WHERE EXISTS(
  SELECT 1 FROM DEPARTMENTS d WHERE e.DEPARTMENT_ID = d.DEPARTMENT_ID
  AND D.DEPARTMENT_NAME = 'Sales')
```

Imagen 11. Exists vs IN-ANY-ALL. La segunda consulta estaría optimizada.

- **IN vs BETWEEN:** Cuando el filtro a aplicar es un rango, es más eficiente usar BETWEEN que usar IN.

```
SELECT
E.FIRST_NAME, E.LAST_NAME, E.SALARY
FROM EMPLOYEES E INNER JOIN DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE E.EMPLOYEE_ID IN (101,102,103,104,105)

SELECT
E.FIRST_NAME, E.LAST_NAME, E.SALARY
FROM EMPLOYEES E INNER JOIN DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE E.EMPLOYEE_ID BETWEEN 101 AND 105
```

Imagen 12. IN vs BETWEEN. La segunda sería óptima

- **Operadores de comparación:** Las columnas que tienen índices, pero incluyen valores nulos en sus registros, no utilizan el índice para resolver la consulta al emplear IS NULL o IS NOT NULL. En algunos SGBD, el operador "distinto" (! o <>) tampoco usa los índices, por lo que la consulta puede reescribirse como en el siguiente ejemplo:

```
SELECT * FROM HR.DEPARTMENTS
WHERE MANAGER_ID IS NOT NULL;

SELECT * FROM HR.DEPARTMENTS
WHERE MANAGER_ID > 0
```

Imagen 13. Optimizando la consulta de una columna indexada con valores nulos.

- **Tablas derivadas, subconsultas y joins:** Un SELECT dentro de otro SELECT es una subconsulta. Si el segundo SELECT está dentro del primer FROM, Se considera una tabla derivada. La subconsulta se ejecuta una vez por cada registro de la consulta principal, mientras que la derivada se utiliza almacenamiento temporal para las tablas al final de la jerarquía y se van enlazando mediante join con la principal.

```
SELECT
  E1.FIRST_NAME,
  E1.LAST_NAME,
  E1.DEPARTMENT_ID,
  E1.SALARY,
  E2.AVG_SALARY
FROM
EMPLOYEES E1,
(
  SELECT ROUND(AVG(E.SALARY)) AVG_SALARY,
  E.DEPARTMENT_ID
  FROM EMPLOYEES E
  GROUP BY E.DEPARTMENT_ID
) E2
WHERE E2.DEPARTMENT_ID = E1.DEPARTMENT_ID
```

Imagen 14. Consulta con tabla derivada

```
SELECT
E1.FIRST_NAME,
E1.LAST_NAME,
E1.DEPARTMENT_ID,
E1.SALARY,
(SELECT ROUND(AVG(E2.SALARY))
  FROM EMPLOYEES E2
  WHERE E2.DEPARTMENT_ID = E1.DEPARTMENT_ID
  GROUP BY E1.DEPARTMENT_ID) AVG_SALARY
FROM EMPLOYEES E1 INNER JOIN DEPARTMENTS D
ON E1.DEPARTMENT_ID = D.DEPARTMENT_ID;
```

Imagen 15. Consulta que incluye una subconsulta antes del FROM (más ineficiente)

- **GROUP BY:** Cuando una consulta incluye varias tablas, normalmente el optimizador elige a qué tablas acceder primero. Si la consulta incluye group by, no tiene en cuenta el costo de agrupar y ordenar la información en memoria. El group by en vistas empeora el rendimiento de las consultas.

```
CREATE VIEW v_AVG_SALARY_DPT AS
SELECT E.DEPARTMENT_ID, ROUND(AVG(E.SALARY)) AVG_SALARY
  FROM EMPLOYEES E
  GROUP BY E.DEPARTMENT_ID

SELECT * FROM v_AVG_SALARY_DPT
WHERE DEPARTMENT_ID = 10

SELECT E.DEPARTMENT_ID, ROUND(AVG(E.SALARY)) AVG_SALARY
  FROM EMPLOYEES E
  GROUP BY E.DEPARTMENT_ID HAVING E.DEPARTMENT_ID = 10
```

Imagen 16. Consulta sobre una vista y su equivalente

- **Cursores y funciones:** No todos los sistemas gestores son eficientes con el uso de cursores. Cuando se usan cursores o funciones, el sistema gestor no puede optimizar el uso de la caché, pues al tratar con variables, los planes de ejecución se tienen que rehacer continuamente, añadiendo carga adicional, por lo que solamente deberíamos emplear cursores y funciones si es estrictamente necesario para la consulta. El uso de funciones o cursores, en algunos SGBD anulan la funcionalidad de los índices. Por ejemplo, si tenemos un índice sobre la columna FIRST_NAME, de la tabla EMPLOYEES, podríamos evitar el uso de la función length, y reemplazarla por el operador like:

```
SELECT * FROM EMPLOYEES
WHERE LENGTH(FIRST_NAME) > 10;

SELECT * FROM EMPLOYEES
WHERE FIRST_NAME LIKE '_________%'
```

Imagen 17. Ambas consultas devuelven lo mismo, pero si hay un índice en FIRST_NAME, la segunda es más rápida

5.3.8. Herramientas de optimización

El SGBD hace uso de las herramientas de monitorización para optimizar el rendimiento. El monitor de rendimiento informa sobre el uso excesivo de recursos de determinadas consultas, en el diccionario también podremos consultar información de rendimiento (estadísticas) y a través del registro de errores, podremos monitorizar fallos o cuellos de botella del sistema.

Plan de ejecución

Es esencial para optimizar las consultas. Ayuda a detectar ineficiencias y decide cómo se realizan las consultas.

Para poder optimizar una consulta, revisaremos en primer lugar su plan de ejecución, que es el modo en el que se acceden a los datos para devolver la información de la consulta (uso de índices, orden de acceso a las tablas, agrupaciones, etc.). El plan de ejecución incluye el coste de ejecución de la consulta (tiempo. Recursos, etc.). Además del coste total, también informa sobre el coste de los pasos intermedios. La información del plan de ejecución varia de unos SGBDs a otros, pero en general suele ser:

- **Orden de consulta de tablas**
- **Índices utilizados**
- **Tipo de acceso a cada tabla (de más a menos eficiente)**
  - Rango (solo una parte de la tabla)
  - Índice completo y acceso localizado a tabla
  - Recorrido completo
- **Tipo de cruces de tablas**
  - Nested loop: filtrar en la primera tabla y buscar coincidencias con la segunda. Ideal para las tablas posteriores pequeñas.
  - Merge join: Se ordenan todas las tablas por el valor de unión y se mezclan ambas. Ideal para relaciones entre tablas uno a uno.
  - Hash map join: Se aplica una función hash a todos los campos de búsqueda, después se ordena la información y se extraen las coincidencias. Funciona bien en conjuntos grandes y desordenados.

Diccionario de datos

El uso del diccionario de datos para monitorización y optimización del sistema depende en gran medida del SGBD. Las opciones de utilidad más comunes que podremos encontrar para optimizar consultas son:

- **Detección de las consultas más lentas**
- **Recursos que emplean las consultas, bloqueos, cantidad de accesos a disco**
- **Fragmentación de objetos**
- **Uso apropiado de los índices**
- **Propuestas realizadas por las herramientas de optimización**

Herramientas específicas

Las herramientas específicas que ayudan a optimizar el rendimiento de nuestro SGBD, pueden ayudar con propuestas y mejoras como:

- **Ejecución de estadísticas sobre las tablas**
- **Asistente para la creación de índices**
- **Reescritura de consultas**
- **Desfragmentación**
- **Gestión de perfiles de usuario (limitación de recursos)**

5.4. Optimización en Oracle

Oracle ofrece una variedad de sistemas de supervisión y optimización que proporcionan supervisión y alertas en tiempo real, asistencia en el diseño de la base de datos, recomendaciones para optimizar el rendimiento y ahorro de costes mediante la automatización.

5.4.1. Optimización del sistema gestor

Bloques de datos

El tamaño de los bloques de la base de datos viene determinado por el parámetro DB_BLOCK_SIZE. Podemos consultarlo en el diccionario de datos, o abriendo el fichero SPFILE.

```
*.control_files='C:\app\Admin
*.db_block_size=8192
*.db_name='XE'
*.diagnostic_dest='C:\app\Adm
```

Imagen 18. Tamaño del bloque, consultando SPFILE.ORA

```
SELECT DISTINCT US.BYTES/US.BLOCKS tam_bloques
 FROM USER_SEGMENTS us
```

10000

TAM_BLOQUES NUMBER
8192

Imagen 19. Tamaño del bloque consultando el diccionario de datos

Ficheros de datos

La opción de configuración de Oracle DATA_FILES especifica los distintos archivos de datos de una base de datos. Puedes utilizar esta opción para optimizar y monitorizar el rendimiento de estos archivos. Este capítulo contiene información sobre la creación de consultas contra las tablas del diccionario de datos DBA_DATA_FILES y USER_DATA_FILES.

```
SELECT
FILE_NAME,
TABLESPACE_NAME,
BYTES,
BLOCKS,
MAXBYTES
 FROM DBA_DATA_FILES
```

10000 Table (read only)

FILE_NAME VARCHAR2(513)	TABLESPACE_NAME VARCHAR2(30)	BYTES NUMBER	BLOCKS NUMBER	MAXBYTES NUMBER
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\SYSTEM01.DBF	SYSTEM	1405091840	171520	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\SYSAUX01.DBF	SYSAUX	671088640	81920	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\UNDOTBS01.DBF	UNDOTBS1	120586240	14720	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\USERS01.DBF	USERS	5242880	640	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\XEPDB1\TBASIR02.DBF	TBASIR	5242880	640	0

Imagen 20. Fichero de datos (diccionario de datos SYS Contenedor root

```
SELECT
FILE_NAME,
TABLESPACE_NAME,
BYTES,
BLOCKS,
MAXBYTES
 FROM DBA_DATA_FILES
```

FILE_NAME VARCHAR2(513)	TABLESPACE_NAME VARCHAR2(30)	BYTES NUMBER	BLOCKS NUMBER	MAXBYTES NUMBER
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\XEPDB1\SYSTEM01.DBF	SYSTEM	293601280	35840	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\XEPDB1\SYSAUX01.DBF	SYSAUX	440401920	53760	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\XEPDB1\UNDOTBS01.DBF	UNDOTBS1	104857600	12800	34359721984
C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\XEPDB1\USERS01.DBF	USERS	5242880	640	34359721984

Imagen 21. Fichero de datos (diccionario de datos SYS@xepdb1

Deshabilitar procesos ocultos

El incremento automático de cada datafile puede ser modificado y consultado en el diccionario de datos.

```
SELECT
DBF.AUTOEXTENSIBLE,
DBF.INCREMENT_BY,
DBF.MAXBYTES
 FROM DBA_DATA_FILES dbf
```

AUTOEXTENSIBLE VARCHAR2(3)	INCREMENT_BY NUMBER	MAXBYTES NUMBER
YES	1280	34359721984
YES	1280	34359721984
YES	640	34359721984
YES	160	34359721984

Imagen 22. Parámetros de los datafiles relativos al incremento automático

```
ALTER DATABASE DATAFILE 'C:\APP\ADMINISTRADOR\PRODUCT\21C\ORADATA\XE\XEPDB1\TBASIR02.DBF' AUTOEXTEND ON;
COMMIT;

SELECT
DDF.FILE_NAME, DDF.TABLESPACE_NAME,
DDF.AUTOEXTENSIBLE
 FROM DBA_DATA_FILES ddf
```

Imagen 23. Modificando el parámetro autoextend

Cuando el parámetro autoextend lo cambiamos a OFF, los procesos ocultos quedan deshabilitados. Esto evita que los procesos automáticos se puedan ejecutar en momentos de máxima carga, aunque hay que tener especial precaución en caso de que se necesite más espacio. Esta medida debe ir acompañada de una alerta para revisar si hay algún tablespace que se pueda quedar sin espacio.

Almacenamiento temporal

Si se desea incrementar su tamaño, se pueden añadir tantos datafiles como sean necesarios.

5.4.2. Optimización de los objetos

Los problemas de optimización que pueden generar los objetos suelen ser por algún error o ineficiencia que se crea debido a la forma de almacenar los datos.

Fragmentación de tablas

Ocurre cuando hay muchas operaciones de modificación o borrado, dejando espacios vacíos en los bloques de datos. Los bloques de datos tienen un tamaño constante, por lo que cuando se borran registros de una tabla, se crean espacios vacíos dentro de los bloques. Esto provoca que se pierda capacidad de almacenamiento y además que las consultas tengan que recorrer más bloques para obtener los datos.

Con el comando MOVE, se eliminan los huecos de los bloques, moviendo las tablas a un tablespace diferente o compactándolas si no se especifica el tablespace. Con la opción COMPRESS, la tabla ocupa menos espacio, al comprimirse, aunque las operaciones de acceso son más costosas. Solo se recomienda comprimirlas si son tablas de acceso poco frecuentes.

```
ALTER TABLE <nombre_tabla> MOVE [COMPRESS];
```

Índices

En Oracle podemos crear índices b-tree ó bitmap (para datawarehouse). Los b-tree, guardan los valores de los campos de forma ordenada, pero si se realizan muchas modificaciones, pueden quedar desbalanceados y perder su eficacia (ver ejemplo 5.3.4).

Cuando el índice ha quedado desbalanceado, se puede reindexar, con REBUILD.

Los comandos para crear y reconstruir índices son:

```
CREATE [BITMAP] INDEX <nombre_indice> ON <nombre_tabla>(columna1, columna2, …);
```

```
ALTER INDEX <nombre_indice> REBUILD;
```

Actualización de estadísticas

El optimizador recurre a las estadísticas que están almacenadas en el diccionario de datos, si éstas no están actualizadas, los costes calculados no serán correctos, y por lo tanto, el plan de ejecución no será apropiado.

Mediante el paquete DBMS_STATS, podremos acceder a los procedimientos disponibles para este fin. La lista de procedimientos es extensa y variada. Se pueden recalcular estadísticas de una tabla, base de datos completa, esquema, etc.

Particionamiento

En Oracle no se puede particionar una tabla existente. Se debe crear una nueva tabla particionada e insertar los datos en ella. Tampoco se puede eliminar la partición de una tabla ya particionada.

```
CREATE TABLE employees (
  id INT NOT NULL,
  hire_date DATE NOT NULL,
  PRIMARY KEY (id)
)
PARTITION BY RANGE ( hire_date ) (
    PARTITION p0 VALUES LESS THAN ('1981-01-01'),
    PARTITION p1 VALUES LESS THAN ('1990-01-01'),
    PARTITION p2 VALUES LESS THAN ('2000-01-01'),
    PARTITION p3 VALUES LESS THAN ('2010-01-01'),
    PARTITION p4 VALUES LESS THAN ('2020-01-01')
);
```

Imagen 24. Creación de una tabla particionada por antigüedad de empleados

5.4.3. Optimización de consultas

El optimizador de consultas de Oracle realiza las siguientes comprobaciones y operaciones en el proceso de ejecución:

- **Evaluar las expresiones y condiciones de la consulta:** Si hay expresiones con operaciones intermedias, las resuelve y transforma antes de ejecutar la consulta. Sustituye determinados operadores e instrucciones para hacerlas más eficientes, por ejemplo LIKE por "=" si no hay wildcards, IN,ANY por between/union según el caso. Siempre que sea posible, se reemplaza una variable con valor conocido por el valor para aprovechar los índices.
- **Simplificar sentencias:** detecta subconsultas que pueden ser fácilmente transformadas en joins.
- **Conversión de vistas en consultas**
- **Optimizar los cruces de tablas:** se evalúan las consultas que implican varias combinaciones de tablas, evaluando la posición correcta del acceso a las mismas.

```
SELECT
D.DEPARTMENT_NAME,
E.FIRST_NAME,
E.SALARY
FROM
EMPLOYEES E
INNER JOIN DEPARTMENTS D
 ON E.DEPARTMENT_ID = D.DEPARTMENT_ID;

SELECT
D.DEPARTMENT_NAME,
E.FIRST_NAME,
E.SALARY
FROM
EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID;
```

Imagen 25. En versiones antiguas de Oracle, la primera sentencia era más eficiente que la segunda, pero en las actuales, el optimizador de consultas gestiona automáticamente los cruces.

5.4.4. Herramientas de optimización

Plan de ejecución

Los permisos SELECT_CATALOG_ROLE y SELECT_ANY_DICTIONARY son necesarios para que un usuario puede acceder a la revisión de los planes de ejecución de sus consultas.

```
GRANT SELECT_CATALOG_ROLE TO <usuario_nombre>;
```

Para ver el plan de ejecución de una consulta, con SQL Developer, se puede hacer de forma gráfica, pulsando F10 o el icono:

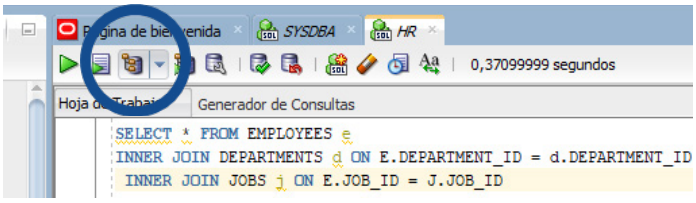

Imagen 26. Ejemplo de consulta examinando el plan de ejecución con SQL Developer

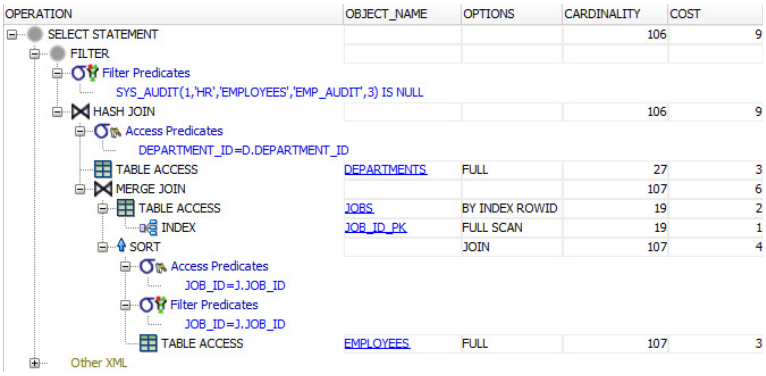

Imagen 27. Plan de ejecución de una consulta simple

SQL Tunning Advisor

Es específica para los SGBD de Oracle. Recopila consultas ejecutadas durante un tiempo, las más frecuentes y las que consumen más recursos, proponiendo posibles mejoras que las pueden optimizar. Se ejecuta en segundo plano si está activo.

Se puede activar con EM o mediante el paquete DBMS_AUTO_TASK_ADMIN (funciones enable y disable). El siguiente comando lo habilita:

```
EXECUTE DBMS_AUTO_TASK_ADMIN.ENABLE ( client_name =>
'sql tuning advisor', operation => NULL, window_name
=> NULL);
```

Si queremos consultar las advertencias generadas automáticamente:

```
SELECT * FROM DBA_ADVISOR_ACTIONS
```

Monitor de operaciones

Podemos consultar el monitor de operaciones de Oracle, con el paquete DBMS_SQL_MONITOR.

Si activamos la monitorización, los resultados se pueden consultar con el procedimiento report_sql_monitor del paquete DBMS_SQL_MONITOR, o mediante la vista V$SQL_MONITOR

Operaciones particulares de Oracle

Oracle dispone de varias herramientas específicas para optimizar tareas habituales, más allá de las propias consultas.

- **Inserciones masivas:** permite insertar datos por bloques en lugar de hacerlo fila a fila. En Oracle se hace con SQL Loader. Otros gestores implementan algo similar, como BulkCopy de SQL Server.
- **Insert append:** cuando se realiza una inserción con append, se rellenan los espacios libres que queden en los bloques de la tabla. Si no se hace, los datos se insertan al final, dejando huecos sin completar. Es más rápida, pero fragmenta la tabla.
- **Nologging:** se evita dejar registro de cada operación, pero no se guardan las transacciones que se vayan ejecutando en este modo.
- **Intercambio de particiones:** se permite intercambiar una tabla por la partición de otra tabla. En caso de update masivo, es más rápido realizar una inserción que una operación de actualización, aunque se requiera una tabla temporal, se mejora el rendimiento.
- **Vistas materializadas:** Es una vista que guarda también los datos además de la consulta. Contiene datos con cálculos y agrupaciones precalculados, que no se actualizan con frecuencia, normalmente se hace solamente en el proceso de carga (ETL) de un datawarehouse. En este tipo de tablas, habitualmente se usan motores de almacenamiento de carga en memoria
- **Merge:** Instrucción que combina update e insert sobre tablas que tienen una clave. Si la clave existe, se ejecuta update, sino insert.
- **Hints:** Los hints son instrucciones que se pueden agregar a una consulta para indicarle a Oracle cómo debe procesarla. Los hints le permiten al usuario "sugerir" a Oracle cómo debe optimizar la consulta.

www.universae.com