Índice

1. INTRODUCCIÓN

Existen dos formas fundamentales de mejorar las prestaciones de un sistema ORACLE:

- indexación
- agrupamientos

Los dos métodos anteriores no afectan la sintaxis de los comandos SQL, por lo que cualquier consulta y/o actualización de datos se realizará de la misma forma.

Sin embargo, los tiempos de respuesta mejorarán considerablemente.

2. INDICES

2.1 MEJORAS DE LOS INDICES

Los índices se utilizan principalmente por dos razones:

- aumentar la velocidad de acceso a columnas indexadas
- garantizar la unicidad de datos

Los índices localizan las filas que han sido seleccionadas en una consulta y reducen de esta manera el número de accesos a disco.

La utilización de índices es muy recomendable para mejorar el rendimiento en la unión de múltiples tablas.

Los índices pueden garantizar que los datos de una columna o de un conjunto de columnas tengan un valor único para cada fila de la tabla.

Esta última propiedad nos permite simular las restricción de entidad (clave principal) del modelo de datos relacional.

2.2 CREACION DE INDICES

El propietario de una tabla o los usuarios que tengan privilegio de indexación sobre la misma pueden indexar sus columnas.

```
CREATE [ UNIQUE ] INDEX  < nombre_del_índice >
ON  < nombre_de_la_tabla >
(< nombre_de_la_columna1 > [, < nombre_de_la_columna2 >,...]);
```

Ejemplos:

```
SQL>CREATE INDEX título_libro
2       ON libros ( título );

Index created.
```

```
SQL>CREATE UNIQUE INDEX numero_de_empleado
2       ON emp ( empno);

Index created.
```

2.3 BORRADO DE INDICES

Los índices, como cualquier otro objeto de la base de datos, se borran con el comando de definición de datos DROP:

```
DROP INDEX  < nombre_del_índice >;
```

2.4 UTILIZACION DE INDICES

El optimizador ORACLE analizará la sentencia SQL para determinar el camino de acceso más rápido a los datos. Si las columnas que aparecen en la cláusula de condición "where" están indexadas, ORACLE utilizará los índices para acceder a los datos.

La sintaxis de los comandos SQL no varía aunque una columna esté indexada.
Para que se utilice un índice definido sobre una columna, ésta no debe ser modificada por ninguna operación.

Ejemplo:

```
SQL>CREATE INDEX  precio_libro
2       ON libros (precio);
```

La siguiente sentencia utilizará el índice:

```
SQL>SELECT * FROM libros
2       WHERE precio = 1000;
```

La siguiente sentencia no utilizará el índice:

```
SQL>SELECT * FROM libros
2       WHERE precio * 0.5 = 1000;
```

2.5 ALMACENAMIENTO DE INDICES

Los índices se **almacenan separadamente de los datos**, por ello se pueden crear y borrar en cualquier momento sin afectar a los datos, sólo se verá afectado el rendimiento.

Los índices son actualizados automáticamente por ORACLE cada vez que se valida una sentencia de actualización de datos.

No existe un límite para la creación de índices.

Para conocer los índices que ha creado un usuario se puede consultar la vista del diccionario de datos **INDEXES.**

2.6 RECOMENDACIONES PARA INDEXAR

Se deben indexar las columnas que forman parte del predicado de unión de una combinación.

No se deben utilizar índices cuando la columna indexada pertenece a una tabla pequeña.

Para cada índice habrá que evaluar si la mejora de prestaciones no se ve superada por los costes de mantenimiento y almacenamiento.

Si los datos son estáticos se reducirán los costes de mantenimiento y es más recomendable crear índices.

Para reducir los costes de mantenimiento se deben **cargar primero los datos y después crear el índice.**

2.7 INDICES CONCATENADOS

Con el comando CREATE INDEX se pueden crear índices concatenados, nombrando **las columnas** de la tabla que deben ser claves, en el orden que deben aparecer.

Para Crear:

```
SQL>CREATE INDEX departamento_empleado
2       ON emp (deptno, job);
```

Para borrar:

```
SQL>DROP INDEX departamento_empleado;
```

La clave de un índice concatenado puede estar formada por un **máximo de 16 columnas o de 240 caracteres**.

Las columnas de la clave de un índice concatenado no tienen por que ser consecutivas ni del mismo tipo de datos.

Para que el índice sea utilizado se deberá referenciar en la cláusula de condición la primera columna del índice.

Ejemplo:

```
SQL>  SELECT * FROM emp
2       WHERE deptno = 10
3       AND job = 'MANAGER';
```

3. AGRUPAMIENTOS

Los agrupamientos son un método alternativo para almacenar los datos en la base datos ORACLE que no tienen ninguna influencia en la sintaxis de los comandos SQL.

Las operaciones de combinaciones de tablas se pueden mejorar utilizando agrupamientos. Un agrupamiento o cluster es un objeto de base de datos que almacena varias tablas que tienen una o más columnas en común en una misma área del disco. Con esto se consigue que las operaciones de combinación de tablas ganen rapidez. Las filas de las tablas con los mismos valores en las columnas de combinación se almacenan físicamente juntas; de esta forma, las consultas serán más rápidas.

3.1 MEJORA DE LOS AGRUPAMIENTOS

Los agrupamientos son totalmente transparentes a los usuarios: no hay que modificar ni las consultas ni las aplicaciones, las consultas son iguales tanto si las tablas están agrupadas como si no.

Los datos de una o más tablas (hasta 32) **se almacenan en un mismo bloque físico** para mejorar el rendimiento en el acceso a los datos de esas tablas.

Se suelen agrupar tablas que tienen **columnas comunes y que son utilizadas en combinaciones de tablas (joins).**

Los agrupamientos no pueden ser consultados como tales, se consultarán las tablas que están incluidas en él.

Cuando se crea un agrupamiento, **se tiene que crear un índice sobre la clave del agrupamiento antes de poder empezar a trabajar con él.**

3.2 CREACIÓN DE AGRUPAMIENTOS

La sintaxis del comando SQL para crear agrupamientos es la siguiente:

```
CREATE CLUSTER  < nombre_del_agrupamiento >
( < columna_clave1 >  < tipo_de_datos1 >,
  < columna_clave2 >  < tipo_de_datos2 >,
  ...
  < columna_clave16 >  < tipo_de_datos16 > );
```

Ejemplo:

```
SQL>CREATE CLUSTER emp_dept
2       ( deptno  NUMBER(2));
```

Especificación de las tablas que se deben agrupar

Se utiliza el comando CREATE TABLE en cualquiera de sus dos formatos para especificar las tablas que se deben agrupar (al menos una columna de la clave debe ser no nula):

Con el primer formato:

```
CREATE TABLE < nombre_de_la_nueva_tabla >
    ( < columna1 >  < tipo_de_datos1 >,
      < columna2 >  < tipo_de_datos2 >,
      ...
      < columnaN >  < tipo_de_datosN > )
    CLUSTER < nombre_del_agrupamiento >
                ( < nombre_de_las_columnas > );
```

Con el segundo copiamos los datos de una tabla ya existente en los bloques del agrupamiento:

```
CREATE TABLE < nombre_de_la_nueva_tabla >
    CLUSTER < nombre_del_agrupamiento >
                ( < nombre_de_las_columnas > )
    AS SELECT * FROM < tabla_existente >;
```

Después habrá que borrar la tabla de la que se han obtenido los datos, puesto que si no tendremos dos copias de los datos.

También habrá que cambiar de nombre a la nueva tabla para que los usuarios y las aplicaciones puedan seguir refiriéndose a ella con el mismo nombre.

```
DROP TABLE < tabla_existente >;
RENAME < nombre_de_la_nueva_tabla > TO < tabla_existente >;
```

Ejemplo:

```
SQL>CREATE TABLE emp2
2       CLUSTER emp_dept ( deptno )
3       AS SELECT * FROM emp;

SQL>CREATE TABLE dept2
2       CLUSTER emp_dept ( deptno )
3       AS SELECT * FROM dept;


SQL>DROP TABLE dept;
SQL>RENAME dept2 TO dept;

SQL>DROP TABLE emp;
SQL>RENAME emp2 TO emp;
```

3.3 BORRAR AGRUPAMIENTOS

Antes de borrar un agrupamiento habrá que borrar las tablas que están incluidas en él a menos que se indique la cláusula INCLUDING CONTENTS.

```
DROP CLUSTER <nombre_del_agrupamiento> [ INCLUDING CONTENTS ];
```

```
SQL>DROP TABLE emp;
SQL>DROP TABLE dept;
SQL>DROP CLUSTER emp_dept;
```

o bien:

```
SQL>DROP CLUSTER emp_dept INCLUDING CONTENT;
```

4. EJERCICIOS PROPUESTOS

1º. Selecciona a todos los "Smith" de la tabla empleados y comprueba el tiempo que tarda en ejecutarse la consulta.

2º. Crea una índice llamado **IDX_APELLIDO** de la tabla empleados y repite la consulta.

3º. Crea una índice llamado **IDX_NOM_APE** de la tabla empleados.

4º. Realiza una consulta que visualice el codigo_pedido y la fecha de pedido de la tabla **SALES_ORDER.**

5º. Crea una índice llamado **IDX_FECH_PEDIDOS** de la tabla **SALES_ORDER**, para que se puedan encontrar mas rápidamente los pedidos por fecha.

6º.Realiza la consulta del ejercicio 4 para ver que tarda menos en ejecutarse.

7º.Mostrar de cada depto su código, su nombre y el nº de empleados que tiene, pero si no tiene empleados que salga un 0.

8º. Crea un indice llamado **IDX_COD_DEP_NOM** de la tabla **DEPARTMENT** para que se puedan encontrar mas rapidamente los empleados por departamento.

9º. Realiza la consulta del ejercicio 7 para ver que tarda menos en ejecutarse.

10º. Visualizar los índices que tenemos en las tablas **EMPLOYEE**, **DEPARTMENT** y **SALES_ORDER**.

11º. Borra los índices creados anteriormente,

12º.- Agrupa las tablas **ARTICULOS**, **PEDIDOS** y **NUEVAS_VENTAS** creadas en los ejercicios propuestos en el tema 10. Para ello crea un cluster por todos los campos comunes.

13º. Borrar los cluster creados.