

Optimización DE CONSULTAS

Sistemas de Bases de Datos 1



Sección B

OPTIMIZACION DE CONSULTAS

- El lenguaje SQL permite escribir una consulta de maneras distintas, obteniendo el mismo resultado. Por tanto, se puede decir que existen maneras mas optimas que otras de escribir una consulta, de tal forma que se maximiza el rendimiento de la misma.

FORMAS DE OPTIMIZAR

- Colocar el nombre de las columnas en lugar de '*'.
 - Aunque se devuelvan todos los campos, no usar '*'.
 - Solo devolver los campos necesarios.
- Ejemplo:

Select * from Estudiante	
Select carnet, nombre, fecha_nacimiento From Estudiante	

FORMAS DE OPTIMIZAR

- Reducir el numero de sub consultas en la consulta principal.

- Ejemplo:

Select nombre from Empleados where salario=
(SELECT MAX(salario) FROM DETALLE_EMPLEADO)
AND edad = (SELECT MAX(edad) FROM
DETALLE_EMPLEADO) and departamento =
'Electronicos';



Select nombre from Empleados where
(salario, edad) = (SELECT MAX(salario), MAX(age) from
DETALLE_EMPLEADO) and departamento =
'Electronicos';



MOVER CONSULTAS A PROCEDIMIENTOS

- Los procedimientos almacenados usualmente son mas rápidos por las siguientes razones:
 - Los procedimientos son compilados, el código SQL no, por lo mismo es mas rápido.
 - Los procedimientos no utilizan tanto ancho de banda porque se pueden hacer varias consultas en un mismo procedimiento.
 - Un procedimiento almacenado crea un plan de ejecución que se utiliza en posteriores ejecuciones.

UTILIZAR EXISTS, IN correctamente

- Usualmente IN es menos eficiente.
 - IN es eficiente cuando la mayoría de los criterios de filtro se encuentran en la sub consulta.
 - EXISTS es eficiente cuando la mayoría de los criterios de filtro se encuentran en la consulta principal.
 - Ejemplo:

Select * from producto p where id_producto
IN (select id_item from item_orden o where
o.id_producto = p.id_producto)



Select nombre from producto p where EXISTS
(select * from item_orden o where o.id_producto
= p.id_producto)

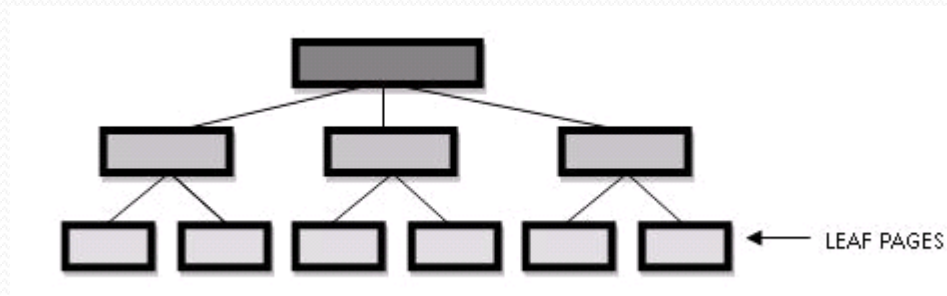


INDICES

- Los objetos son objetos de la base de datos, cuya función es optimizar el acceso a datos.
- En general mejoran el rendimiento de las sentencias `SELECT` y empeoran (mínimamente) el rendimiento de los `INSERT` y los `DELETE`.

INDICES

- Un índice normal es una estructura de árbol, que cuenta con una pagina principal y luego este con paginas hijas, que a su vez tiene mas paginas hijas hasta llegar al final del índice (leaf level).
- La clave del índice esta repartida en las paginas del índice, de modo tal que la búsqueda se haga leyendo la menor cantidad de datos posibles.



INDICES

- Tipos de Índices (SQL Server)
 - Clustered Index: Se crea automáticamente al declarar una llave primaria.
 - Non-Clustered Index: Se declara un Índice para un cierto campo. Se recomienda usar índices para campos que sean:
 - Usados Frecuentemente en criterios de búsqueda.
 - Usados en un JOIN con otras tablas.
 - Usado como llave foránea.
 - Usados en una clausula ORDER BY

INDICES

- Sintaxis:
 - Create INDEX nombre_index ON TABLA(columna)
- Ejemplo:
 - *create index IND_CNTRY ON Mundo(Country)*
 - *create index IND_CNTNT ON Mundo(Continent)*

FORMAS DE OPTIMIZAR

- Evitar el uso de condiciones LIKE, debido a que estos utilizan los índices incorrectamente (si el campo tiene uno).
- Si se debe utilizar este tipo de condiciones, evaluar si puede evitarse el carácter ‘%’ al inicio de la condición.

Select nombre from Estudiante where nombre LIKE
‘%Miguel%’



Select nombre from Estudiante where nombre LIKE
‘Miguel%’



FORMAS DE OPTIMIZAR

- Evitar condiciones de diferencia (!=) o negaciones (NOT). Las condiciones de este tipo hacen que el motor de base de datos realice barridos a los índices en lugar de búsquedas.
- Antes de crear condiciones de este tipo, asegurarse que la condición no puede ser cambiada para utilizar eficientemente los índices.

FORMAS DE OPTIMIZAR

- Para aquellas consultas en las que el numero de datos a devolver sea muy grande, se debe considerar limitar el numero de resultados devueltos si no se requieren todos.
- Se puede utilizar la columna ROWNUM de ORACLE o TOP para SQL Server.
- Ejemplo:
 - *Select top 10 cancion from Música → SQL SERVER*
 - *Select cancion from Musica where*
 - *rownum <= 10 → ORACLE*

TABLAS TEMPORALES

- Las tablas temporales son visibles solamente en la sección actual. Estas se eliminan automáticamente al acabar la sesión o la función o procedimiento almacenado en el cual fueron definidas.
- Si se utiliza un set de datos para varias consultas, por ejemplo utilizando sub consultas, se recomienda guardar el set de datos en una tabla temporal para reutilizar los datos.

TABLAS TEMPORALES

Ejemplo (SQL Server):

-- Creación de tabla

```
create table #Temporal(  
    codigo int  
);
```

-- Inserción en tabla

```
insert into #Temporal(codigo) values (1);
```

-- Select de tabla

```
select * from #Temporal;
```

TAREA # 5

- Investigar la sentencia EXPLAIN en MySQL.
- Investigar el Execution Plan de SQL Server.
- Realizar una consulta en donde se realice un JOIN de dos o mas tablas (tablas a su elección).
- Por medio de EXPLAIN de MySQL, obtener los detalles de la consulta y explicar los resultados.
- Obtener el Execution Plan de la consulta en SQL Server y explicar los resultados obtenidos.

TAREA # 5

- Fecha de Entrega: Lunes 12 de Octubre hasta las 11:59 pm por correo electrónico.
- Entregables:
 - Archivo PDF.
 - Nombre archivo: Tarea5_carnet.pdf
- Asunto Correo: [BD1] Tarea5