“Iteración 4 – Diseño físico y optimización de consultas”

Daniel Perilla, Juan C. Corrales

Universidad de los Andes, Bogotá, Colombia

{d.perilla11, jc.corrales[}@uniandes.edu.co](mailto:}@uniandes.edu.co)

Fecha de presentación: Noviembre 21 de 2017

Tabla de contenido

[1 Análisis 1](#_Toc363135282)

[2 Diseño físico 1](#_Toc363135283)

[3 Documentación escenario de pruebas 6](#_Toc363135284)

1. Análisis: (1%)

RFC11: Clase RegistroVentas que tiene de atributos producto más consumido, producto menos consumido restaurante más frecuentado, restaurante menos frecuentado y día de la semana que además va a ser una llave.

1. Diseño físico:

* (1%) Dentro del diagrama UML y en la base de datos es necesario crear dos nuevas clases llamada REGISTRO\_VENTAS que tiene de atributos producto más consumido, producto menos consumido restaurante más frecuentado, restaurante menos frecuentado y día de la semana que además va a ser una llave lo que significa además que no va a ser nulo. La segunda clase recibe el nombre de TIPO\_CLIENTE que contiene el ID de un cliente junto con su respectivo tipo. En esta tabla van a haber dos llaves privadas de manera que exista la posibilidad de combinaciones de los distintos tipos con las distintas tuplas.
* (19%) Documente su diseño físico

RFC9: Índice denso secundario en RESTAURANTE

RAZÓN: Debido a que se está buscando un restaurante determinado se utiliza un índice en primario en lo que es idRestaurante para llegar en un paso a los productos ofrecidos por ese restaurante. A pesar de que se almacenaría una mayor cantidad de datos comparado con un índice disperso, se contrasta con el hecho de que no son muchos los restaurantes y de que la velocidad de búsqueda compensa lo que cuesta el almacenamiento.

Índice disperso secundario en FECHA

RAZÓN: Como se busca un valor dentro de un rango de fechas entonces se espera un índice disperso que cubre un rango de datos, lo hace de manera secuencial y ocupa menos espacio. De igual manera se tie

ne en cuenta que la eficiencia y el costo de almacenamiento relacionado con este esquema es determinado por el rango entre el cual se designan las llaves.

RFC9

Índice en la tabla PEDIDO en ID\_RESTAURANTE; TIPO: denso primario

Índice en la tabla PEDIDO en FECHA; TIPO: disperso secundario

RFC10: Índice disperso secundario en FECHA

RAZÓN: Debido a que se está buscando un restaurante determinado se utiliza un índice en primario en lo que es idRestaurante para llegar en un paso a los productos ofrecidos por ese restaurante. A pesar de que se almacenaría una mayor cantidad de datos comparado con un índice disperso, se contrasta con el hecho de que no son muchos los restaurantes y de que la velocidad de búsqueda compensa lo que cuesta el almacenamiento.

Índice disperso primario en Restaurante

RAZÓN: A pesar de que se buscan aquellos clientes que no han consumido productos de un restaurante dado, es igual importante confirma que no lo hagan hecho, debido a esto toca tener un índice para poder encontrar los restaurantes en primera instancia para luego recorrer su respectiva lista de productos para confirmar que no hayan comprado nada.

RFC10

Índice en la tabla PEDIDO en ID\_RESTAURANTE; TIPO: denso primario

Índice en la tabla PEDIDO en FECHA; TIPO: disperso secundario

RFC11: Índice denso primario en REGISTROVENTAS.DIA

RAZÓN: Creamos una nueva clase para manejar este registro que recibe el nombre de RegistroVentas, este contiene un atributo día que contiene toda la información buscada para un día específico, debido a esto solo es necesario crear un índice denso primario en lo que es día para para que este vaya directamente a la información. Además no nos costará mucho espacio este índice pues solo hay 7 días en la semana.

RFC11

Índice en la tabla REGISTROVENTAS en DIA; TIPO: denso secundario

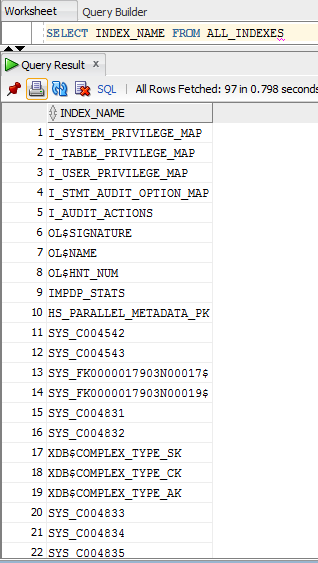
RFC12: Índice denso secundario

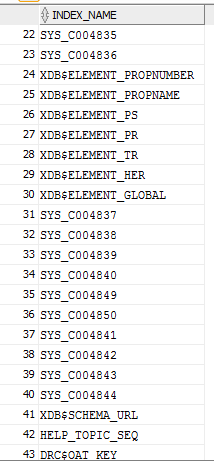
Razón: Esta consulta la decidimos hacer construyendo tres sentencias SQL separadas, estas por su cuenta recogían la información dependiendo del tipo de cliente que se estaba buscando, que se sabía en un if dentro del método en Java. En el primero de los SQL propuestos utilizaba una comparación entre dos fechas de la misma tabla, lo que se puede mejorar si existe un índice secundario denso en este atributo. En el segundo se hacía una comparación con una valor predeterminado, pero debido a que no tiene importancia en este caso crear un índice no lo hacemos. En último lugar se crea un índice secundario denso el la tabla MENUS\_PEDIDOS en el atributo IDORDEN pues se busca que no existen ordenes que tengan menú que es más eficiente saberlo si existe un índice en el idOrden de menusPedidos.

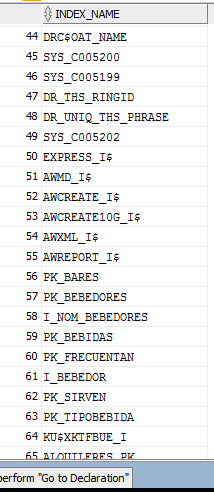
RFC12

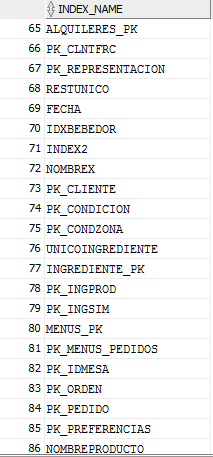
Índice en la tabla MENUS\_PEDIDOS en IDORDEN; TIPO: denso secundario

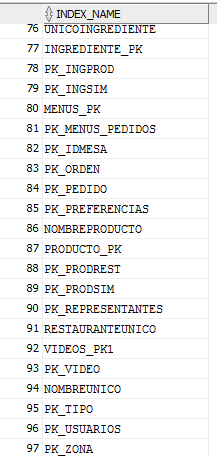
Índice en la tabla ORDENES en FECHA; TIPO: denso secundario











Utilizando el comando SELECT INDEX\_NAME FROM ALL\_INDEXES, encontramos los valores de todos los índices en el sistema. Los índices relacionados con los requerimientos son aquellos creados como índices primarios densos derivados de la creación de la llave primaria en una tabla en la base de datos. Ayudan en los requerimientos funcionales debido a que gracias a que son primarios y densos la persona consultando no tiene que esperar mucho tiempo para que el sistema encuentre la información pues hay un relación intrínseca entre el índice y un atributo de una tupla buscada.

* (44%) Documente plenamente el análisis realizado

Documentación del escenario de pruebas:

Sentencias SQL:

RFC9:

* SELECT CLIENTES.ID, CLIENTES.NAME, CLIENTES.IDMESA FROM (SELECT \* FROM CLIENTES, (SELECT \* FROM PEDIDOS, ORDENES WHERE PEDIDOS.ID\_ORDEN = ORDENES.ID) WHERE CLIENTES.ID = ORDENES.ID\_CLIENTE), (SELECT \* FROM PRODUCTO\_RESTAURANTE, (SELECT \* FROM RESTAURANTES WHERE RESTAURANTES.ID =" + idRestaurante + ") WHERE RESTAURANTES.ID = PRODUCTO\_RESTAURANTE.ID\_REST) WHERE PEDIDOS.ID\_PRODUCTO = PRODUCTO\_RESTAURANTE.ID\_PROD AND ORDENES.FECHA BETWEEN" + fecha1 +"AND" +fecha2+ " ORDER BY"+orderBy+" GROUP BY"+groupBy+";"

RFC10:

* SELECT CLIENTES.ID, CLIENTES.NAME, CLIENTES.IDMESA FROM (SELECT \* FROM CLIENTES, (SELECT \* FROM PEDIDOS, ORDENES WHERE PEDIDOS.ID\_ORDEN = ORDENES.ID) WHERE CLIENTES.ID = ORDENES.ID\_CLIENTE), (SELECT \* FROM PRODUCTO\_RESTAURANTE, (SELECT \* FROM RESTAURANTES WHERE RESTAURANTES.ID =" + idRestaurante + ") WHERE RESTAURANTES.ID <> PRODUCTO\_RESTAURANTE.ID\_REST) WHERE PEDIDOS.ID\_PRODUCTO = PRODUCTO\_RESTAURANTE.ID\_PROD AND ORDENES.FECHA BETWEEN" + fecha1 +"AND" +fecha2+ " ORDER BY"+orderBy+" GROUP BY"+groupBy+";"

RFC11:

* SELECT \* FROM REGISTROVENTAS

RFC12:

* SELECT \* FROM ORDENES, MENUS\_PEDIDOS, CLIENTES WHERE ORDENES.ID\_CLIENTE = CLIENTES.ID AND NOT (ORDENES.ID = MENUS\_PEDIDOS.IDORDEN);
* SELECT \* FROM CLIENTES, ORDENES WHERE CLIENTES.ID = ORDENES.ID\_CLIENTE AND COSTOTOTAL > 11065755;
* SELECT \* FROM (SELECT CLIENTES.ID AS ID1, NAME AS NAME1, IDMESA AS IDMESA1, ORDENES.FECHA AS FECHA1 FROM CLIENTES, ORDENES WHERE CLIENTES.ID = ORDENES.ID\_CLIENTE) T1 WHERE NOT EXISTS (SELECT \* FROM (SELECT CLIENTES.ID AS ID2, NAME AS NAME2, IDMESA AS IDMESA2, ORDENES.FECHA AS FECHA2 FROM CLIENTES, ORDENES WHERE CLIENTES.ID = ORDENES.ID\_CLIENTE) T2 WHERE T1.FECHA1 <= T2.FECHA2 + 7);

PLANES DE EJECUCIÓN

RFC9:

RFC12:

