Unidad 3

Fecha: 21/04/2016

Optimización en BD

1° Parte

Una empresa de venta de electrodomésticos con varias sucursales en todo el país requiere implementar un nuevo sistema de facturación.

El nuevo sistema es una aplicación Cliente/Servidor de tres capas:

- Browser: posee la presentación al usuario final mediante formularios en HTML.
- Servidor de aplicación: posee toda la lógica de la aplicación y del negocio.
- Servidor de base de datos: contiene el modelo de datos implementado en SQL Server 2005.

Tanto el servidor de base de datos como el de aplicación se encuentran centralizados en el Data Center en Buenos Aires, y las sucursales ingresan vía http a través de una red WAN privada.

Situación Inicial:

El sistema luego de su instalación inicial, funcionaba de manera óptima en cuanto a performance, pero a medida que fue pasando el tiempo se empezó a notar lentitudes en algunos puntos de la aplicación.

El modelo de datos cuenta con las siguientes tablas (entre muchas otras):

```
CREATE TABLE Cliente
        (Id_Cliente      INT NOT NULL,
        Nombre_Cliente CHAR(255),
        Dirección_Cliente CHAR(512))

CREATE TABLE Factura
        (Nro_Factura INT NOT NULL,
        Id_Cliente INT,
        Fecha_Factura DATETIME)

ALTER TABLE Cliente ADD CONSTRAINT PK_Cliente PRIMARY KEY (Id_Cliente)

ALTER TABLE Factura ADD CONSTRAINT PK_Factura PRIMARY KEY (Nro_Factura)
```

ALTER TABLE Factura ADD CONSTRAINT FK_Cliente FOREIGN KEY (Id_Cliente) REFERENCES Cliente (Id_Cliente)

Luego de 1 año, la tabla de clientes tiene 1 millón de registros, y la de facturas 10 millones.

Situación 1:

Uno de los puntos en donde se puso lento el sistema es en la búsqueda de clientes. El sistema tiene la funcionalidad de buscar por id o por nombre. La consulta por Id es y siempre fue rápida, en cambio la por nombre sufrió una degradación importante:

```
SELECT * FROM Cliente WHERE Id_Cliente = 1234

SELECT * FROM Cliente WHERE Nombre_Cliente LIKE 'Perez%'
```

Mirando el plan de ejecución:

1. ¿Cuál es la diferencia en el plan de ejecución que causa tal diferencia en el comportamiento?

Una de las medidas que se intentaron para remediar la situación es crear un índice por el campo "Nombre" ya que no había ninguno. El índice por Id ya estaba creado porque SQL Server lo crea automáticamente cuando se define una clave primaria:

```
CREATE INDEX IDXNombre ON Cliente (Nombre_Cliente)
```

El usuario dice que la situación mejoró para algunos casos, pero no en todos. Cuando se hace una búsqueda por un apellido poco común, el sistema responde rápidamente, en cambio cuando lo hace por un apellido para el cual hay muchos clientes vuelve a ser muy lento (ej.: Perez).

Mirando nuevamente el plan de ejecución:

2. ¿Hubo alguna diferencia en los procedimientos de bajo nivel usados antes y después de la creación del índice?

Se llegó a la conclusión de que el problema estaba que al tratarse de un índice NONCLUSTERED de SQL Server, el optimizador detectaba previamente consultando su catálogo que el resultado de la búsqueda por nombre puede llegar a tener muchos resultados (lo que involucraría mucha utilización de punteros) por lo que calculaba que iba a ser más rápido hacer un SCAN sobre la tabla entera que usar el índice.

Como segunda medida, se plantea transformar el índice por Nombre en CLUSTERED aprovechando las ventajas del orden físico. El problema es que ese tipo de índice ya está usado para el Id_Cliente y solamente puede haber uno solo por tabla.

Por eso se llevó a cabo una tarea de mantenimiento durante el fin de semana para transformar el índice de Id_Cliente en NONCLUSTERED y el de Nombre en CLUSTERED.

ALTER TABLE Factura DROP CONSTRAINT FK_Cliente

ALTER TABLE Cliente DROP CONSTRAINT PK_Cliente

ALTER TABLE Cliente ADD CONSTRAINT PK_Cliente PRIMARY KEY NONCLUSTERED (Id_Cliente)

ALTER TABLE Factura ADD CONSTRAINT FK_Cliente FOREIGN KEY (Id_Cliente) REFERENCES Cliente (Id_Cliente)

DROP INDEX Cliente.IDXNombre

CREATE CLUSTERED INDEX IDXNombre ON Cliente (Nombre_Cliente)

El usuario ahora si dice que ambas consultas de cliente les funciona bien.

Mirando nuevamente los planes de ejecución:

3. ¿Qué diferencia se encuentra a cómo se venía resolviendo antes?
4. ¿Por qué el cambio de tipo de índice NONCLUSTERED => CLUSTERED mejoró mucho la consulta por Nombre, pero el mismo cambio a la inversa no empeoró prácticamente en nada la búsqueda por ID? (Analizar la densidad de cada campo en cuanto a valores distintos que puede tomar).

Situación 2:

El usuario indica que el mismo problema de lentitud lo tiene en la consulta de facturas de un cliente en particular.

```
SELECT Nro_Factura, Nombre_Cliente, Fecha_Factura

FROM Factura F JOIN Cliente C ON  F.Id_Cliente = C.Id_Cliente

WHERE F.Id_Cliente = 1234
```

Mirando el plan de ejecución y pensando en que un cliente puede tener muchísimas facturas cargadas.

1. ¿Cuál puede ser el problema?
2. ¿Cómo se puede mejorar la situación? (SQL Server NO crea automáticamente un índice al definir una clave extranjera/ajena)

Parte 2: Análisis de Costos

Situación 3:

Si se conoce la siguiente situación:

Tabla cliente: 9.500 registros.

Atributo Tipo_Cliente de la tabla cliente: tiene 5 valores distintos (Monotributista, Responsable Inscripto, Exento, Consumidor Fina).

Tabla Producto: 25.000 registros

Atributo Familia_producto de la tabla producto: tiene 20 valores distintos.

Tabla Venta_producto: 120.000 registros. Esta tabla tiene definida una clave foránea en su atributo cliente con la clave primaria de la tabla Cliente, y una clave foránea en su atributo producto sobre la clave primaria de la tala Producto.

Atributo Cantidad de la tabla venta_producto toma valores enteros entre 1 a 100.

Calcule las estimaciones de costos que se realizarán para cada una de las siguientes consultas:

1. **SELECT id_cliente, apellido, nombres**

   FROM cliente

   WHERE tipo_cliente = 'Monotributista';

2. **SELECT id_producto, descripción, costo_final**

   FROM producto

   WHERE familia_producto = 12;

3. **SELECT c.apellido, p.descripcion, v.cantidad**

   FROM venta_producto v, cliente c, producto p

   WHERE v.cantidad > 10 AND

   v..cliente = c.id_cliente AND

   v.producto = p.id_producto;

Situación 4:

Dadas las siguientes tablas de una BD Relacional:

CUENTA(nombre-sucursal,numero-cuenta,nombre-cliente,saldo)

PRESTAMO(nombre-sucursal,numero-prestamo,nombre-cliente,monto)

SUCURSAL(nombre-sucursal,activo ciudad)

CLIENTE(nombre-cliente,calle,ciudad-cliente)

Donde se encuentran subrayados los campos que componen la clave primaria de cada tabla (los campos de clave primaria siempre tienen un índice generado). Las tablas sucursal y préstamo tienen menos tuplas que cuenta y cliente respectivamente. Dadas las siguientes consultas SQL que intentan responder los clientes que tienen una cuenta y un préstamo en la sucursal "Casa Central".

i.
```
(select nombre-cliente
from cuenta
where nombre-sucursal = "Casa Central")
intersect
(select nombre-cliente
from préstamo
where nombre-sucursal = "Casa Central")
```

ii.
```
select nombre-cliente
from préstamo
where nombre-sucursal = "Casa Central"
and nombre-cliente in (select nombre-cliente
from cuenta
where nombre-sucursal ="Casa Central")
```

iii.
```
(select nombre-cliente
from prestamo p, cuenta c
where p.nombre-cliente=c.nombre-cliente
and nombre-sucursal = "Casa Central")
```

1. Haga un análisis de los costos en cada una de las sentencias (en base a los tipos de operaciones que observa en cada una de ellas, los atributos que se quieran mostrar, los índices si existen, etc.), y de acuerdo con su análisis, haga el ranking de las mismas de acuerdo con su costo.

Situación 5:

Sea una empresa de construcciones que está ejecutando varios proyectos en el país y suponga que tenemos las siguientes tablas en una Base de Datos Relacional:

Proveedor (prov, nombre-prov, calle, ciudad, fax)
Partes (parte, nombre-parte, descripcion)
Proyecto (proyecto, nombre-proyecto, ciudad)
PPP (prov, parte, proyecto, cantidad) prov, parte y proyecto son claves foráneas.

Escriba las sentencias de SQL y del álgebra relacional SQL para las siguientes consultas, genere el árbol canónico y luego optimice el árbol:

1. Muestre la clave de las partes y el nombre de la parte que el proveedor "P5" proveyó en una cantidad cuyo rango va de 350 a 750 inclusive.

   SELECT partes.parte, partes.nombre-parte
   FROM partes, ppp
   WHERE partes.parte=ppp.parte **AND**
   ppp.prov= "P5" **AND**
   ppp.cantidad **between** 350 **AND** 750

2. Muestre el número de las partes, su descripción, de aquellas partes suministradas por los proveedores de "Córdoba" a los proyectos localizados en "Rosario".
3. Para las partes que son suministradas a un proyecto, muestre el número de parte, el nombre del proyecto, el nombre del proveedor, y la cantidad total correspondiente.
4. Seleccione los números de las partes que son suministradas por un proveedor situado en "Rosario" o a un proyecto situado en "Rosario".¿Cómo se explicaría la situación detectada en la auditoría?