Propuesta de esquema de fragmentación

Consultas

2)

```
SELECT SalesPersonID, COUNT(Sales.SalesOrderHeader.TerritoryID)
FROM Sales.SalesOrderHeader INNER JOIN Sales.SalesPerson
ON Sales.SalesOrderHeader.SalesPersonID = Sales.SalesPerson.BusinessEntityID
GROUP BY SalesPersonID ORDER BY COUNT(*) DESC
```

3)

```
SELECT TOP 1 Sales.Customer.CustomerID, COUNT(SalesOrderID) AS NumberOfOrders
FROM Sales.SalesOrderHeader INNER JOIN Sales.Customer
ON Sales.SalesOrderHeader.CustomerID = Sales.Customer.CustomerID
WHERE Sales.SalesOrderHeader.TerritoryID IN (1,2,3,4,5,6)
GROUP BY Sales.Customer.CustomerID ORDER BY NumberOfOrders DESC
```

4)

```
SELECT * FROM Sales.Customer C INNER JOIN Sales.SalesTerritory ST
ON C.TerritoryID = ST.TerritoryID WHERE [ST].[Group] = 'North America';
```

5)

```
SELECT * FROM Sales.SpecialOffer WHERE SpecialOfferID IN (
SELECT SpecialOfferID FROM Sales.SpecialOfferProduct SOP
INNER JOIN Production.Product P ON SOP.ProductID = P.ProductID
INNER JOIN Production.ProductSubcategory PS ON P.ProductSubcategoryID =
PS.ProductSubcategoryID
INNER JOIN Production.ProductCategory PC ON PS.ProductCategoryID =
PC.ProductCategoryID WHERE
[PC].[Name] = 'Bikes');
```

6)

```
select top 3 so.ProductID, COUNT(*) as Cantidad_Productos
from EuropePacific.sales.SalesOrderDetail so
inner join EuropePacific.sales.SalesOrderHeader sh
on sh.SalesOrderID = so.SalesOrderID
where sh.TerritoryID = 9
group by so.ProductID
order by 2 asc
```

7)

```
create or alter procedure sp_ActualizarSubcategor?a(
	@productId int,
	@subcategoriaID int
)
as
	IF exists( select * from AdventureWorks2019.Production.Product where
ProductID = @productId )
	BEGIN
		IF exists( select * from
AdventureWorks2019.Production.ProductSubcategory where ProductSubcategoryID =
@subcategoriaID )
```

```
                    update AdventureWorks2019.Production.Product
                    set [ProductSubcategoryID] = @subcategoriaID, ModifiedDate =
GETDATE()
                    where ProductID = @productId
             ELSE
                    SELECT -1 as resultado
      END
      ELSE
             SELECT 0 as resultado
```

8)

```
SELECT * FROM Production.Product WHERE SellEndDate is not null
```

9)

```
SELECT CustomerID FROM Sales.Customer WHERE (TerritoryID = 1 or TerritoryID = 4) AND
PersonID IS NULL
```

10)

```
select * from NorthAmerica.sales.SalesOrderHeader
where TerritoryID = 1 and CustomerID in ( select CustomerID from
NorthAmerica.sales.SalesOrderHeader
where TerritoryID between 2 and 6 )
union all
select * from NorthAmerica.sales.SalesOrderHeader
where TerritoryID = 1 and CustomerID in ( select CustomerID from
EuropePacific.sales.SalesOrderHeader
where TerritoryID between 7 and 10 )
```

Decidiendo cómo fragmentar

El primer paso para saber cómo fragmentar nuestro esquema es observar los enunciados que el profesor enlistó. Analizando estos enunciados nos daremos cuenta de que los enunciados del 1 al 6, el enunciado 9, el 10 y el 14 acceden a la información de clientes y ventas a través de territorio, por lo cual esto nos dará una idea de cómo segmentar.

Si vamos a nuestro gestor de base de datos y consultamos la tabla SalesTerritory nos daremos cuenta de que contamos con 10 territorios. Si decidimos tener un fragmento por territorio podría no ser tan factible, recordemos que para decidir qué fragmentos utilizaremos, primero debemos determinar la selectividad a partir de los predicados, es decir, a cuántas tuplas accedo en cada una de las consultas. Aquellas consultas que son las que más accedo son las que me convendrían segmentar.

Sabiendo esto, ahora surge la siguiente pregunta: ¿Sobre qué tablas nos conviene segmentar? Observemos de nuevo nuestra lista de enunciados y veamos cuáles son las tablas que están siendo más accedas. Haciendo un análisis rápido nos daremos cuenta de que las tablas Customer, SalesOrderHeader, SalesOrderDetail y Product son las tablas en las que nos deberemos enfocar, por lo que para poder realizar su fragmentación deberemos generar el conjunto de predicados y con base en estos conjuntos determinaremos los segmentos. Un punto importante para tomar en consideración es que la tabla Customer está por arriba de SalesOrderHeader en orden jerárquico, debido a que Customer tiene una llave foránea que se exparse a SalesOrderHeader y lo mismo pasa

de SalesOrderHeader hacia SalesOrderDetail, por lo cual, si deseamos fragmentar estas tablas mencionadas, deberemos realizarlo a partir de la fragmentación horizontal de la tabla Customer.

Predicados simples de Customer

Una vez teniendo identificadas las tablas y atributo en los que nos deberemos enfocar para realizar los predicados, es momento de recordar la definición de predicado simple. Un predicado simple es una expresión condicional que evalúa un atributo asociado a una tabla a través de un operador relacional para ver si satisface un valor específico que se asocia a la expresión. En nuestro ejemplo de fragmentación, a nosotros nos interesa realizar los predicados simples sobre el atributo de territorio, por las razones que ya mencionamos previamente, así que nuestro conjunto de predicados simples se verá de la siguiente manera:

PrCustomer = {

P1: TerritoryID = 1

…

P10: TerritoryID = 10

}

Naturalmente, podríamos pensar que tendríamos 10 fragmentos debido a nuestros 10 predicados simples, sin embargo, el enunciado 1 nos pide agrupar a los clientes por región. Si vamos a la tabla SalesTerritory y observamos la cantidad de regiones nos daremos cuenta de que tenemos 3 en total, por lo que tendremos 3 fragmentos.

Predicados minitérminos de Customer

Una vez teniendo nuestros predicados simples es momento de generar el conjunto de predicados minitérminos. Los predicados minitérminos son la conjunción de las posibilidades que nos da combinar los predicados simples en su forma natural y negada. Cada predicado minitérmino representa un posible fragmento, sin embargo, hay predicados minitérminos que se van a contraponer con otros, en cuestión de la evaluación lógica, recordemos que para que un esquema de fragmentación sea válido deberemos comprobar las 3 reglas de un esquema correcto: completitud, reconstrucción y disjunción.

Con la fórmula $\Pr = 2^{10}$ veremos que en total tendríamos 1024 predicados minitérminos por lo que nos conviene ir a la tabla SalesTerritory y realizar una consulta para ver qué territorios se relacionan con qué regiones y así generar los 3 predicados minitérminos que a su vez se traducen a 3 segmentos. Nuestros predicados minitérminos de la tabla Customer se verán así:

MCustomer = {

M1: P1 ^ P2 ^ P3 ^ P4 ^ P5 ^ P6 ^ not(P7) ^ not(P8) ^ not(P9) ^ not(P10)

M2: not(P1) ^ not(P2) ^ not(P3) ^ not(P4) ^ not(P5) ^ not(P6) ^ P7 ^ P8 ^ P9 ^ not(P10)

M3: not(P1) ^ not(P2) ^ not(P3) ^ not(P4) ^ not(P5) ^ not(P6) ^ not(P7) ^ not(P8) ^ not(P9) ^ P10

M4: not(P1) ^ not(P2) ^ not(P3) ^ not(P4) ^ not(P5) ^ not(P6) ^ not(P7) ^ not(P8) ^ not(P9) ^ not(P10)

…

}

Selectividad de los minitérminos

Finalmente, deberemos calcular la selectividad, la cual nos indicará el número de tuplos de la relación que serán accesadas por una consulta de usuario especificada de acuerdo con un predicado minitérmino dado. Por cada predicado minitérmino tendremos una consulta como se muestra a continuación:

SEL(M1): SELECT COUNT(*) FROM Sales.Customer WHERE TerritoryID BETWEEN 1 AND 6

SEL(M2): SELECT COUNT(*) FROM Sales.Customer WHERE TerritoryID BETWEEN 7 AND 9

SEL(M3): SELECT COUNT(*) FROM Sales.Customer WHERE TerritoryID = 10

Fragmentación horizontal derivada de SalesOrderHeader

A partir de esta fragmentación horizontal primaria que realizamos sobre la tabla Customer, deberemos realizar nuestra fragmentación horizontal secundaria para SalesOrderHeader y SalesOrderDetail. Según el libro de texto proporcionado por el profesor una fragmentación horizontal secundaria o derivada se consigue de la siguiente manera:

"La fragmentación derivada horizontal se define partiendo de una fragmentación horizontal. En esta operación se requiere de Semi-junta (Semi-Join) el cual nos sirve para derivar las tuplas o registros de dos relaciones. Una fragmentación horizontal derivada se define en la relación miembro de una liga de acuerdo con la operación de selección especificada en la relación propietaria. La liga entre las relaciones propietaria y miembro se define como una equi-junta. Una equi-junta se puede implementar por semi-juntas. Esto es importante, ya que se quiere particionar una relación miembro de acuerdo con la fragmentación de su propietario, pero se quiere que los fragmentos resultantes queden definidos únicamente en los atributos de la relación miembro".

Nuestras operaciones de semi-join para la fragmentación horizontal derivada se verán de la siguiente forma:

$FSOH_1$ = SalesOrderHeader $\ltimes FC_1$

SELECT * FROM Sales.SalesOrderHeader WHERE CustomerID IN

(SELECT CustomerID FROM Sales.Customer WHERE TerritoryID BETWEEN 1 AND 6)

$FSOH_2$ = SalesOrderHeader $\ltimes FC_2$

SELECT * FROM Sales.SalesOrderHeader WHERE CustomerID IN

(SELECT CustomerID FROM Sales.Customer WHERE TerritoryID BETWEEN 7 AND 9)

$FSOH_3$ = SalesOrderHeader $\ltimes FC_3$

SELECT * FROM Sales.SalesOrderHeader WHERE CustomerID IN

(SELECT CustomerID FROM Sales.Customer WHERE TerritoryID = 10)

Fragmentación horizontal derivada de SalesOrderHeader

Después de esto solo resta realizar las operaciones de semijoin de los fragmentos de SalesOrderHeader con la tabla SalesOrderDetail por medio del atributo que tienen en común.

$FSOD_1$ = SalesOrderHeader $\ltimes FSOH_1$

SELECT * FROM Sales.SalesOrderHeader WHERE CustomerID IN

(SELECT CustomerID FROM Sales.Customer WHERE TerritoryID BETWEEN 1 AND 6)

$FSOD_2$ = SalesOrderHeader $\ltimes FSOH_2$

SELECT * FROM Sales.SalesOrderHeader WHERE CustomerID IN

(SELECT CustomerID FROM Sales.Customer WHERE TerritoryID BETWEEN 7 AND 9)

$FSOD_3$ = SalesOrderHeader $\ltimes FSOH_3$

SELECT * FROM Sales.SalesOrderHeader WHERE CustomerID IN

(SELECT CustomerID FROM Sales.Customer WHERE TerritoryID = 10)