Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Bases de datos distribuidas

Fragmentación de una base de datos

Integrantes: Lisardo René Morgado Resendiz
López Navarrete Sergio Hidekel
Moreno Galicia Jesús Antonio

Grupo: 3TM3 - Profesor: De la Cruz Sosa Carlos

Fecha: 9/Marzo/2022

Índice

Objetivo

Realizar una fragmentación horizontal con base en 10 enunciados a una base de datos con el fin de cumplir con las tres reglas de la fragmentación.

Consideraciones teóricas

- *Base de datos*

Una base de datos tiene diversas funciones las cuáles no sólo se limitan a almacenar datos, si no que también de conectarlos entre sí con una unidad lógica. En términos generales, una base de datos es un conjunto de datos estructurados que pertenecen a un mismo contexto y, en cuanto a su función, se utiliza para administrar de forma electrónica grandes cantidades de información.

- *Database Management System*

Los programas denominados sistemas gestores de bases de datos, abreviado SGBD (del inglés Database Management System o DBMS), permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

- *SQL*

SQL es un acrónimo en inglés para Structured Query Language. Un Lenguaje de Consulta Estructurado. Un tipo de lenguaje de programación que te permite manipular y descargar datos de una base de datos. Tiene capacidad de hacer cálculos avanzados y álgebra. Es utilizado en la mayoría de empresas que almacenan datos en una base de datos. Ha sido y sigue siendo el lenguaje de programación más usado para bases de datos relacionales.

- *Bases de datos distribuidas*

Las bases de datos distribuidas o Distributed Database Management System (DDBMS) se caracterizan por almacenar la información en varias computadoras conectadas entre sí, a las cuáles el usuario puede acceder desde cualquier sitio como si se tratara de una red local.

- *Fragmentación de una base de datos distribuida*

Los conceptos que necesitamos conocer para entender la arquitectura de referencia son:

Esquema global: Define todos los datos que están contenidos en la BDD como si la DB no fuese distribuida. Por esta razón, el esquema global puede ser definido exactamente de la misma manera que una DB no distribuida. Refiriéndose a un modelo relacional, se tendrían relaciones globales.

Fragmentos: Cada relación global puede ser dividida en porciones que no se solapen, llamados fragmentos. El mapa resultante se denomina esquema de fragmentación. Una relación global puede dividirse en n fragmentes y un fragmento solo puede pertenecer a una relación global. Los fragmentos se referencian por un nombre de relación global y un subíndice.

Los fragmentos son porciones lógicas de relaciones globales que pueden estar físicamente ubicadas en uno o varios nodos de la red. El esquema de ubicación define en qué nodos va a ser almacenado un fragmento. El tipo de mapa definido en el esquema de ubicación determina si la BDD es redundante o no. Todos los fragmentos que corresponden a la misma relación global R y están ubicados en el mismo nodo constituyen la imagen física de una relación global R en el nodo. Las imágenes físicas se pueden referenciar por el nombre de la relación global R y un superíndice.

Por último, se denomina copia de fragmento a la información de un fragmento en un nodo determinado, y se denota usando el nombre de la relación global, un subíndice y un superíndice.

En un nivel más bajo, es necesario construir un mapa que relacione las imágenes físicas con los objetos que son manipulados por los gestores locales. Este mapa se llama esquema de mapas locales y depende del tipo de SGBD local. Por lo tanto, en un sistema heterogéneo se tienen diferentes tipos de mapas locales en los distintos nodos.

Fragmentación: Los principales problemas de la fragmentación se pueden resumir básicamente en:

- Encontrar la unidad apropiada de distribución, es decir, definir qué contiene un fragmento.
- El rendimiento se afecta cuando existen aplicaciones que necesitan tener una vista completa de un objeto o entidad (relación en el modelo relacional) y está descompuesta en fragmentos ubicados físicamente en sitios distintos. Esta recuperación requiere la ejecución de operaciones de unión y combinación.
- Se pierde el significado semántico del objeto o entidad al tenerse el concepto subdivido en fragmentos ubicados en diferentes sitios. Esto puede provocar, por ejemplo, complicaciones en la interpretación y verificación del modelo conceptual que representa a los requerimientos.

A pesar de estos inconvenientes, la fragmentación facilita el proceso concurrente de las transacciones y, por lo tanto, la recuperación de información. Con el fin de realizar una fragmentación adecuada es necesario proporcionar información que ayude a realizarla. Esta información normalmente debe ser proporcionada por el usuario y tiene que ver con cuatro tipos:

1. Información sobre el significado de los datos.
2. Información sobre las aplicaciones que los usan.
3. Información acerca de la red de comunicaciones.
4. Información acerca de los sistemas de cómputo.

Existen tres tipos de fragmentación: 1. Fragmentación horizontal; 2. Fragmentación vertical; 3. Fragmentación mixta o hibrida. Dado que una relación se corresponde esencialmente con una tabla y la cuestión consiste en dividirla en fragmentos menores, inmediatamente surgen dos alternativas lógicas para llevar a cabo el proceso: la división horizontal y la división vertical.

Estos dos tipos de partición podrían considerarse los fundamentales y básicos. Sin embargo, existen otras alternativas.

Fundamentalmente, se habla de fragmentación mixta o híbrida cuando el proceso de partición hace uso de los dos tipos anteriores. A continuación, se enuncian las tres reglas que se han de cumplir durante el proceso de fragmentación, las cuales aseguraran la ausencia de cambios semánticos en la base de datos durante el proceso.

Condición de completitud: La descomposición de una relación R en los fragmentos R1, R2, …, R es completa sí y solamente sí cada elemento de datos en R se encuentra en uno o varios fragmentos R. Esta propiedad asegura que los datos de la relación global se proyecten sobre los fragmentos sin pérdida alguna.

Condición de reconstrucción: Si la relación R se descompone en los fragmentos R1, R2, …, R, entonces debe existir algún operador relacional Delta, tal que R = Delta. La reconstrucción de la relación a partir de sus fragmentos asegura la preservación de las restricciones definidas sobre los datos en forma de dependencias.

Condición de fragmentos disjuntos: Esta condición asegura que los fragmentos horizontales sean disjuntos. Si una relación R se descompone verticalmente sus atributos primarios clave normalmente se repiten en todos sus fragmentos

- *Fragmentación horizontal*

La fragmentación horizontal se refiere a la división de una relación en subconjunto (fragmentos) de tuplas (filas); cada fragmento se guarda en un nodo diferente y cada uno de ellos tiene filas únicas; sin embargo, todas las filas únicas tienen los mismos atributos (columnas). En suma, cada fragmento equivale a una sentencia SELECT, con la cláusula WHERE en un solo atributo.

Existen dos variantes de la fragmentación horizontal: la primaria y la derivada. La fragmentación horizontal primaria de una relación se desarrolla empleando los predicados definidos en esa relación. Por el contrario, la fragmentación horizontal derivada consiste en dividir una relación partiendo de los predicados definidos sobre alguna otra.

- *Fragmentación vertical*

La fragmentación vertical se refiere a la división de una relación en subconjuntos de atributos (columna); cada subconjunto (fragmento) se guarda en un nodo diferente y cada fragmento tiene columnas únicas, con la excepción de la columna clave, la cual es común a todos los fragmentos. Esto es el equivalente de la sentencia SELECT columna1, columna2 INTO Nueva_Tabla FROM Tabla.

- *Fragmentación mixta*

La fragmentación mezclada se refiere a una combinación de estrategias horizontales y verticales. En otras palabras, una tabla puede dividirse en varios subconjuntos horizontales (filas), y cada una tiene un subconjunto de los atributos (columnas)

Descripción de la práctica

1. Descripción de la fragmentación horizontal basada en los 10 enunciados aplicados a la BD AdventureWorks2019 en SQL Server. En este punto se debe de argumentar de acuerdo con los enunciados, la propuesta de fragmentación horizontal primaria y la fragmentación horizontal derivada. Se debe incluir los predicados miniterminos que generan los fragmentos primarios y las consultas SQL que generan los fragmentos derivados a partir de los fragmentos primarios. Argumentar porque la propuesta de fragmentación cumple las tres reglas de la fragmentación.

2. Descripción de la asignación de los fragmentos y tablas no fragmentadas en las instancias SQL Server/MySQL a considerar para la distribución de la BD. En este punto, se debe describir la configuración de las instancias, servidores vinculados, scripts de las BD en cada instancia. El punto 1 y 2 de sebe entregar un archivo con formato PDF.

3. Implementación del patrón DAO en la aplicación. Se debe de observar los elementos básicos del patrón DAO.

4. Implementación de las 10 operaciones mediante la aplicación o en SQL Server según sea la solución seleccionada.

5. Explicación de la implementación por un integrante del equipo.

6. Respuesta a preguntas individuales relacionadas con el ejercicio.

7. Código documentado. Comentarios en el código, indispensable en las secciones importantes de la solución.

8. Código en el repositorio del equipo. Se almacena el código de la aplicación y BD en el repositorio de GitHub.

9. Archivo .TXT con nombre aplicacionFragmentación.txt donde se indique ruta dentro del repositorio y nombre de los archivos que integran el ejercicio (scripts de las BD, código de la aplicación, scripts en SQL en caso de programar consultas a nivel del servidor). Asimismo, incluir al principio del archivo .TXT el nombre de cada integrante del equipo. El archivo .TXT debe ubicarse en la raíz del repositorio.

Datos de la práctica

Rutas

Aplicación node ---------- ./Fragmentacion_Horizontal/Fragmentacion/

Consultas ---------- ./Fragmentacion_Horizontal/FragmentacionHorizontal.sql

Archivos de la aplicación

Index.js: Archivo que inicia el servidor

App.js: Archivo principal de la aplicación

Config.js: Archivo con las configuraciones básicas para correr el servidor, como credenciales de la base de datos

Routes/products.routes.js: Archvio que guarda la ruta dentro de la api para cada controlador

Database/connection.js: Archivo que se encarga de realizar la conexión con la base de datos

Database/index.js: Archivo principal en el que se exporta la conexión y los querys

Database/querys.js: Archivo que guarda las querys en lenguaje sql a ejecutar

Controllers/products.controller.js: Archivo con Código js que gestiona las conexiones conforme a la ruta consultada

Consultas

Listar datos del empleado que atendió más ordenes por territorio.

```
--2. Listar datos del empleado que atendi@ m@s ordenes por territorio.
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
```

Listar los datos del cliente con más ordenes solicitadas en la región “North America”.

```
--3. Listar los datos del cliente con m@s ordenes solicitadas en la regi@n @North America@.
SELECT *FROM OPENQUERY(MYSQL, 'SELECT  adventureworks2019.customer.CustomerID, count(adventureworks2019.sles.SalesOrderID) AS NumberOfOrders
FROM adventureworks2019.sles
INNER JOIN adventureworks2019.customer ON adventureworks2019.sles.CustomerID = adventureworks2019.customer.CustomerID
GROUP BY adventureworks2019.customer.CustomerID ORDER BY NumberOfOrders DESC LIMIT 1 ');
```

Listar el producto más solicitado en la región “Europe”.

```
--4. Listar el producto m@s solicitado en la regi@n @Europe@.
SELECT TOP 1 ProductID, COUNT(ProductID) AS TOTAL FROM Sales.SalesOrderDetail SSA
INNER JOIN Sales.SalesOrderHeader SSO ON SSA.SalesOrderID = SSO.SalesOrderID WHERE TerritoryID IN(7,8,10)  GROUP BY ProductID
```

Listar las ofertas que tienen los productos de la categoría “Bikes”

```
--5. Listar las ofertas que tienen los productos de la categor@a @Bikes@
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');
```

Listar los 3 productos menos solicitados en la región “Pacific”

```
--6. Listar los 3 productos menos solicitados en la regi@n @Pacific@
SELECT TOP 3 ProductID, COUNT(ProductID) AS TOTAL FROM Sales.SalesOrderDetail SSA
INNER JOIN Sales.SalesOrderHeader SSO ON SSA.SalesOrderID = SSO.SalesOrderID WHERE TerritoryID IN(9)  GROUP BY ProductID
```

Actualizar la subcategoría de los productos con productId del 1 al 4 a la subcategoría valida para el tipo de producto.

```
--7. Actualizar la subcategor@a de los productos con productId del 1 al 4 a la subcategor@a valida para el tipo de producto.
CREATE PROCEDURE sp_ActualizarSubcatego
@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

EXEC sp_ActualizarSubcatego 1,2


SELECT *FROM AdventureWorks2019.Production.Product
SELECT *FROM Production.ProductSubcategory
```

Listar los productos que no estén disponibles a la venta.

```
--8. Listar los productos que no est@n disponibles a la venta.
SELECT *FROM OPENQUERY(MYSQL, 'SELECT *FROM Production.Product WHERE SellEndDate is not null;');
```

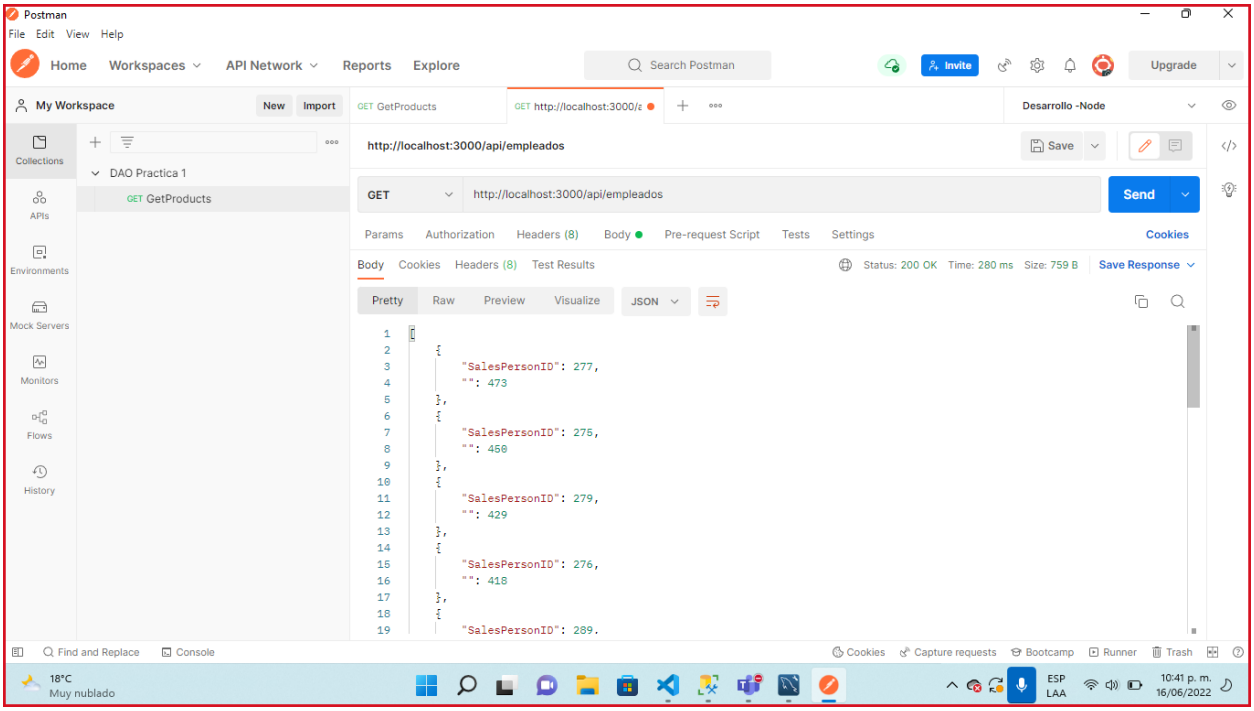
Listar los clientes del territorio 1 y 4 que no tengan asociado un valor en personId

```
--9. Listar los clientes del territorio 1 y 4 que no tengan asociado un valor en personId
SELECT CustomerID FROM Sales.Customer WHERE (TerritoryID = 1 or TerritoryID = 4) AND PersonID IS NULL
```

Es importante mencionar que esta base de datos se fragmentó en dos partes, una en mysql y dos en sql server

Respuestas en postman de la API de node

Ruta /empleados



Ruta /ordenes

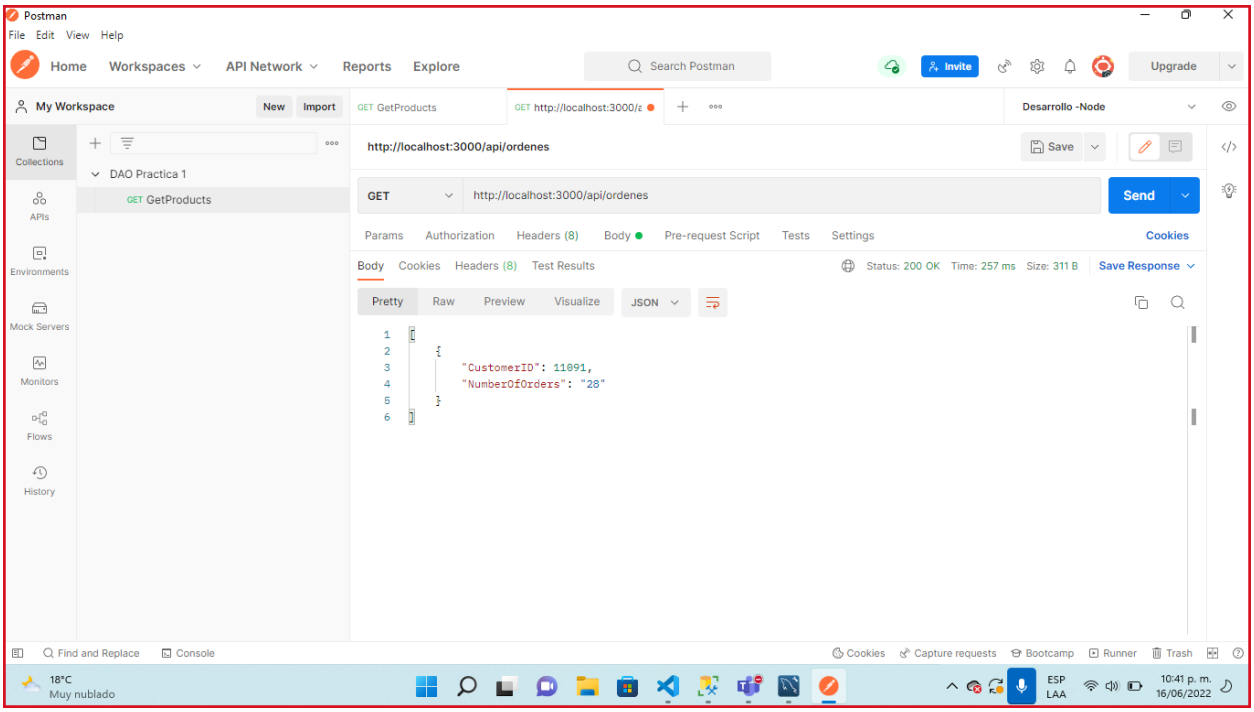


Tabla de evaluación

	Indicador	Cumple (SI/NO/ PARCIALMENTE)	Puntos	Observaciones
1	Describe la fragmentación horizontal desarrollada. (10% máximo)			
2	Describe la asignación de fragmentos. (10% máximo)			
3	Implementa el patrón DAO. (20% máximo)			
4	Implementa las operaciones sobre la BD Fragmentada. (20% máximo)			
5	Explica el desarrollo del ejercicio. (15% máximo)			
6	Responde a preguntas dirigidas a cada integrante. (10% máximo)			
7	Entrega código documentado. (5% máximo)			
8	Comparte el código en el repositorio de GitHub. (5% máximo)			
9	Entrega archivo TXT. (5% máximo)			

Bibliografía

[1] Elmasri, R., & Navathe, S. B. (2016). *Fundamentos de los sistemas de bases de datos, edición global* (7.ª ed.). Pearson.

[2] LeMahieu, W., Broucke, V. S., & Baesens, B. (2018). *Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data* (1.ª ed.). Cambridge University Press.