UNIVERSIDAD DE BUENOS AIRES

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Bases de Datos

Trabajo Práctico 2

1er Cuatrimestre 2016

Integrante	LU	Correo electrónico
Almansi, Emilio	**674/12**	ealmansi@gmail.com
Fixman, Martín	**391/11**	martinfixman@gmail.com
Gunski, María Celeste	**899/03**	celestegunski@gmail.com
Maurizio, Miguel	**635/11**	miguelmaurizio_92@hotmail.com

Índice

1. Introducción

Las bases de datos relacionales SQL cuentan con muchas ventajas pero su utilización puede ser imposible en el caso de tener grandes volúmenes de datos. En cambio, las bases de datos NoSQL tienen la principal ventaja de manejar mucha más inforcación debido a la posibilidad de escalar en forma horizontal, lo cual no es posible con las bases de datos tradicionales. Es decir, para aumentar la cantidad de datos que procesa una base de datos relacional debemos tener una computadora mejor.

Por otro lado, al estar utilizando una base NoSQL se puede incrementar la capacidad mediante una mayor cantidad de computadoras. Esto implica que en contextos donde la cantidad de datos aumenta rápidamente debemos utilizar una arquitectura distribuida, forzándonos a usar una base de datos NoSQL.

En este trabajo estudiamos la bases de datos NoSQL. Particularmente aquellas de la familia orientada a documentos, utilizando la tecnología $MongoDB$.

En la primera parte, utilizando el problema del trabajo práctico 1, estudiamos y analizamos el diseño de los documentos a utilizar con el propósito de responder queries específicas. No solo creamos la base SQL, sino que también implementamos la migración de SQL a NoSQL.

En la segunda parte, utilizamos el esquema $Map\ Reduce$ para realizar consultas sobre los datos, permitiendo parelelismo en su ejecución.

Por último, exploramos e investigamos la técnica de *sharding*.

2. Ejercicio 1

- Los documentos correspondientes a las facturas enviadas tendrán el siguiente diseño:

  ```
  {"id","usuario","suscripcionRubi","monto","fecha"}
  ```

  Los datos para llenarlos se pueden obtener ejecutando la siguiente consulta en la base de SQL:

  ```
  SELECT f.idFactura AS id, f.idUsuario AS usuario, 1 AS suscripcionRubi, f.monto,
  f.periodo AS fecha
  FROM factura f
  INNER JOIN suscripcionRubiOriente sr ON f.idUsuario = sr.idUsuario
  WHERE f.periodo = sr.periodo
  AND f.periodo < date('now','-1 month')
  UNION
  SELECT f.idFactura AS id, f.idUsuario AS usuario, 0 AS suscripcionRubi, f.monto,
  f.periodo AS fecha
  FROM factura f
  WHERE f.idUsuario NOT IN
  (SELECT idUsuario FROM suscripcionRubiOriente WHERE periodo = f.periodo)
  AND f.periodo < date('now','-1 month')
  ```

- Los documentos correspondientes a las ventas realizadas tendrán el siguiente diseño:

  ```
  {"id","fecha","idComprador","idVendedor","calificacionComprador",
    "calificacionVendedor","comision","tipoPublicacion"}
  ```

  Los datos para llenarlos se pueden obtener ejecutando la siguiente consulta en la base de SQL:

  ```
  SELECT c.idCompra AS id, c.fecha AS fecha, c.idUsuario AS idComprador,
  p.idUsuario AS idVendedor, cal.valoracionComprador AS calificacionComprador,
  cal.valoracionVendedor AS calificacionVendedor, (tp.comision * p.precio) AS comision,
  "servicio" AS tipoDePublicacion
  FROM compra c, calificacion cal, publicacion p, publicacionFinalizada pf,
  tipoPublicacion tp, item it
  WHERE c.idPublicacionFinalizada = pf.idPublicacion AND pf.idPublicacion = p.idPublicacion
  AND p.nombreTipoPublicacion = tp.nombre
  AND it.idPublicacion = c.idPublicacionFinalizada AND it.tipo = 1
  AND c.idCalificacion = cal.idCalificacion
  AND c.fecha < date('now','-1 month')
  UNION
  SELECT c.idCompra AS id, c.fecha AS fecha, c.idUsuario AS idComprador,
  p.idUsuario AS idVendedor, cal.valoracionComprador AS calificacionComprador,
  cal.valoracionVendedor AS calificacionVendedor,
  (tp.comision * p.precio) AS comision, "producto" AS tipoDePublicacion
  FROM compra c, calificacion cal, publicacion p, publicacionFinalizada pf,
  tipoPublicacion tp, item it
  WHERE c.idPublicacionFinalizada = pf.idPublicacion AND pf.idPublicacion = p.idPublicacion
  AND p.nombreTipoPublicacion = tp.nombre
  AND it.idPublicacion = c.idPublicacionFinalizada AND it.tipo = 0
  AND c.idCalificacion = cal.idCalificacion
  AND c.fecha < date('now','-1 month')
  ```

```
UNION
SELECT c.idCompra AS id, c.fecha AS fecha, c.idUsuario AS idComprador,
p.idUsuario AS idVendedor, cal.valoracionComprador AS calificacionComprador,
cal.valoracionVendedor AS calificacionVendedor,
(tp.comision * p.precio) AS comision, "mixto" AS tipoDePublicacion
FROM compra c, calificacion cal, publicacion p, publicacionFinalizada pf,
tipoPublicacion tp, item it
WHERE c.idPublicacionFinalizada = pf.idPublicacion AND pf.idPublicacion = p.idPublicacion
AND p.nombreTipoPublicacion = tp.nombre
AND it.idPublicacion = c.idPublicacionFinalizada AND it.tipo <> 0 AND it.tipo <> 1
AND c.idCalificacion = cal.idCalificacion
AND c.fecha < date('now','-1 month')
```

3. Ejercicio 2

En la sección previo desnormalizamos la base de datos SQL. En esta parte vamos a realizar diversas consultas utilizando *Map Reduce*.

3.1. Importe total de ventas por usuario

Por cada factura registrada, vemos el usuario y el importe, luego calculamos el total paralelizando por usuario:

```
var map1 = function(){
    emit(this["id"],this["monto"])
}

var reduce1 = function(key,values){
    return Array.sum(values)
}

db.facturas.mapReduce(map1,reduce1,{query : {}, out : "map_res1"})
```

3.2. reputación histórica de cada usuario

Por cada venta devolvemos tanto el puntaje del comprador como del vendedor. Luego juntamos todas las calificaciones en un mismo nodo.

Entendemos que la *reputacion historica* se refiere al promedio de las calificaciones. Si fuera otra la función, no sería un problema dado que tenemos el historial completo de cada usuario en cada reduce y podríamos implementar otra diferente.

```
var map2 = function(){
    emit(this["idComprador"],this["calificacionComprador"])
    emit(this["idVendedor"],this["calificacionVendedor"])
}

var reduce2 = function(key,values){
    return Array.sum(values) / values.length
}

db.ventas.mapReduce(map2,reduce2,{query : {}, out : "map_res2"})
```

3.3. Operaciones con comisión más alta

En este caso debemos saber cuál es el valor de la comisión más alta, la forma que encontramos para resolver es enviar todas las operaciones (en realidad solo el id y la comisión) a un solo nodo y ahi calcular las oeraciones con comisión máxima.

```
var map3 = function(){
    var pair = {"comision": this["comision"], "id": this["id"]};
    emit(0, pair)
}
```

```
var reduce3 = function(key,values){
    var max = 0;
    for(var i =0; i < values.length; i++){
        max = Math.max(max, values[i]["comision"]);
    }
    res = [];
    for(var i =0; i < values.length; i++){
        if(values[i]["comision"] === max) {
            res.push(values[i]["id"])
        }
    }
    return {"res": res};
}

db.ventas.mapReduce(map3,reduce3,{query : {}, out : "map_res3"})
```

3.4. Monto total facturado por año

Revisamos las facturas, separamos por año y luego sumamos cada monto. Notemos que el reduce es solamente sumar, que es lo mismo que usamos en el reduce de la consulta 1.

```
var map4 = function(){
    var s = new Date(
                this["fecha"].replace( /(\d{2})\/(\d{2})\/(\d{4})/, "$2/$1/$3")
            ).getFullYear();
    emit(s, this["monto"]);
}

var reduce4 = reduce1

db.facturas.mapReduce(map4,reduce4,{query : {}, out : "map_res4"})
```

3.5. Monto total facturado por año con suscripción Rubí

Esta consulta es idéntica a la consulta 4, con la diferencia que antes de emitir en el map debemos chequear que en la factura tenga suscripción a Rubí del oriente.

```
var map5 = function(){
    if(this["suscripcionRubi"] === "1"){
        var s = new Date(
                this["fecha"].replace( /(\d{2})\/(\d{2})\/(\d{4})/, "$2/$1/$3")
            ).getFullYear();
        emit(s, this["monto"])
    }
}

var reduce5 = reduce1

db.facturas.mapReduce(map5,reduce5,{query : {}, out : "map_res5"})
```

3.6. Total publicaciones por tipo

Revisamos las ventas y emitimos el tipo, luego contamos cuantas aparecen de cada una.

```
var map6 = function(){
    emit(this["tipoPublicacion"], 1)
}

var reduce6 = reduce1

db.ventas.mapReduce(map6,reduce6,{query : {}, out : "map_res6"})
```

4. Ejercicio 3

5. Conclusión