**PEP 2 Taller de Bases de Datos**

**Semestre 2-2021 - Departamento de Ingeniería Informática**

**Benjamín Jorquera 19182719-8, Bryan Salas 19316410-2**

**1.** Considere las siguientes tablas de una base de datos relacional.

PRODUCTOS( codprod, nombre, precio\_lista )

VENTAS( codventa, codprod, precio, fecha )

CONTIENE( codins, codprod, proporción )

INSUMOS( codins, nombre, descripción )

Se pide formular el equivalente BSP SQL a las siguientes consultas SQL secuenciales. Las tuplas están distribuidas en función del primer campo de cada tabla. En las consultas paralelas se asume la existencia del pseudo-lenguaje visto en clases, el cual permite generar tablas temporales y leer las tuplas de tablas una por una.

**1.a** Productos que contengan Almendras como insumo y que tengan un precio promedio de venta superior a 5000.

select P.nombre

from PRODUCTOS P, CONTIENE C, INSUMOS I, VENTAS V

where P.codprod = C.codprod and

C.codins = I.codins and

I.nombre = “Almendras” and

V.codprod = P.codprod

group by P.codprod

having average(precio)> 5000

order by nombre

Solución BSP SQL:

*Procesador maestro hace lo siguiente* ::

for( int i= 0; i<P; i++ )

bsp\_send( i, new message( "consulta 1.a" ) );

resultado = recibe\_resultados( P );

*fin procesador maestro*

*Cada procesador esclavo en paralelo hace lo siguiente* ::

msg = bsp\_get\_message();

-- Dado que las tuplas están distribuidas por código de insumo, un procesador

-- tendrá todas las tuplas que contienen el mismo código de insumo.

-- Es decir, el procesador que contenga la tupla con código de insumo

-- perteneciente a Almendras tendrá todas las tuplas de la tabla CONTIENE

-- que contiene el código de insumo de las Almendras.

------------------------------------------------------------------------------

-- Solución 1:

-- Se seleccionan los códigos de productos de las tablas CONTIENE e INSUMOS

-- donde coincida el código de insumo y que este último sean almendras.

-- Luego se envían a todos los procesadores los códigos de estos productos.

-- Se distribuyen a los procesadores los valores de las tablas de VENTAS por -- el código del producto.

-- Se seleccionan los códigos de los productos con un precio de venta promedio -- mayor a 5000.

-- Se envían los productos al procesador maestro.

T1 = select codprod from CONTIENE c, INSUMOS i

where c.codins = i.codins and i.nombre = "Almendras";

T2 = envia\_a\_todos(T2);

T3 = distribuye\_tabla(VENTAS, codprod);

T4 = select codprod from T3 where codprod in T2 group by codprod

having average(precio) > 5000;

Tr = select nombre from PRODUCTOS where codprod in T4;

bsp\_send\_table( maestro, Tr );

*fin procesador esclavo*

-------------------------------------------------------------------------------- Solución 2:

-- Se seleccionan los códigos de productos de las tablas CONTIENE e INSUMOS

-- donde coincida el código de insumo y que este último sean almendras.

-- Luego se envían a todos los procesadores los códigos de estos productos, y

-- se seleccionan estos mismos códigos y su precio de venta de la tabla

-- VENTAS.

-- Se envían todos los procesadores estas tuplas y se seleccionan los códigos

-- de los productos con un precio de venta promedio mayor a 5000.

-- Se envían los productos al procesador maestro.

T1 = select codprod from CONTIENE c, INSUMOS i

where c.codins = i.codins and i.nombre = "Almendras";

T2 = envia\_a\_todos(T1);

T3 = select codprod, precio from VENTAS where codprod in T2;

T4 = envia\_a\_todos(T3);

T5 = select codprod from T4 group by codprod

having average(precio) > 5000;

Tr = select nombre from PRODUCTOS where codprod in T5;

bsp\_send\_table( maestro, Tr );

*fin procesador esclavo*

**1.b** Se requiere saber los nombres de los productos que tienen la mayor cantidad de insumos.

create view TotalInsumos( codprod, cant\_insumos )

select codprod, count(codins) as cant\_insumos

from CONTIENE

group by codprod;

select P.nombre

from PRODUCTOS P, TotalInsumos T

where P.codprod = T.codprod and

T.cant\_insumos IN (

select max( cant\_insumos )

from TotalInsumos

);

Solución BSP SQL:

*Procesador maestro hace lo siguiente* ::

for( int i= 0; i<P; i++ )

bsp\_send( i, new message( "consulta 1.b" ) );

resultado = recibe\_resultados( P );

*fin procesador maestro*

*Cada procesador esclavo en paralelo hace lo siguiente* ::

msg = bsp\_get\_message();

-- Se distribuyen las tablas de CONTIENE a los procesadores por el código del

-- producto.

-- Se seleccionan los códigos de producto, contando la cantidad de insumos

-- de cada producto de la tabla T1 y agrupando por código de producto.

-- Se envían a todos los procesadores los valores de T2.

-- Se seleccionan los nombres de los productos con mayor cantidad

-- de insumos y se envían al procesador maestro.

T1 = distribuye\_tabla(CONTIENE, codprod);

T2 = select codprod, count(codins) as cant\_insumos from T1

group by codprod;

T3 = envia\_a\_todos(T2);

Tr = select P.nombre from PRODUCTOS P, T3 where P.codprod = T3.codprod and

T3.cant\_insumos IN (select max(cant\_insumos) from T3);

bsp\_send\_table( maestro, Tr );

*fin procesador esclavo*

**2.** Cálculo Map-Reduce de la respuesta para la consulta **1.b**. Se supone que las tablas CONTIENE y PRODUCTOS están distribuidas como CSV en un archivo de texto en cualquier orden. Asuma que cuenta con funciones de extracción de valores de cada campo en cada línea de texto similares al ejercicio de clase.

// Archivo: Producto - Contiene

// - P, codprod, nombre, precio\_lista

// - C, codins, codprod, proporción

---------------------------------------------------------------------

// - P, 1, chocolate, 1000,

// - P, 2, leche, 1500

// - C, 1, 1, 0.1

// - C, 2, 1, 0.9

Hiperstep 1::

//

mapper( linea\_texto ) {

string code = get\_campo( 1, linea\_texto);

// campo1 = codprod if P, codins if C

string campo1 = get\_campo( 2, linea\_texto);

// campo2 = nombre if P, codprod if C

string campo2 = get\_campo( 3, linea\_texto);

if(code == "P") {

emit( campo1, pair({nombre: campo2 , valor: 0}) );

} else {

emit( campo2, pair({nombre: null, valor: 1}) );

}

}

//

reducer( llave, valores ) {

int total = 0;

string nombre = "";

foreach par in valores do {

total = total + par.valor;

if(par.nombre != null) {

nombre = par.nombre;

}

}

emit( nombre, total );

}

Hiperstep 2::

//

mapper( linea\_texto ) {

string nombre = get\_campo( 1, linea\_texto);

int total = get\_campo( 2, linea\_texto);

emit( "all", pair(nombre, total) );

}

//

reducer( llave, valores ) {

int max = -1;

foreach par in valores do {

if(par.total > max){

max = par.total

}

foreach par in valores do {

if(par.total == max) {

emit( par.nombre );

}

}

}