Bases de Datos II **Primer Parcial**

TEMA 02

Alumno: Freixes Hutchings Matias Ezequiel L.U. 55100 23/09/24

1) Considerando como criterio de optimización a la minimización del transporte de información entre los nodos distribuidos (minimizar el tráfico de red).
Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los abonados de una empresa de telefonía celular:

Nodo 1: Abonados

Nro. abonado	Apellido y Nombres	Dirección	Tipo de servicio
8 bytes	40 bytes	30 bytes	3 bytes

Contiene: 15700 registros
Longitud del registro: 81 bytes

Nodo 2: Servicios

Tipo de servicio	Denominación del servicio	Costo del servicio
3 bytes	40 bytes	7 bytes

Contiene: 4800 registros
Longitud del registro: 50 bytes

a) Determinar el tamaño de las relaciones.
b) Se desea obtener la siguiente consulta:
Para cada abonado, obtener su Apellido y Nombres y la Denominación del servicio contratado.

i. Teniendo en cuenta el **Nodo 3**, como nodo resultado (nodo en el cual hay que entregar el resultado de la consulta), obtener las alternativas de solución a la consulta de tipo reunión (inner join) y optar por aquella solución que resulte más adecuada de acuerdo al criterio considerado ("*minimizar el transporte de bytes, entre los distintos nodos distribuidos*").
ii. Si ahora consideramos al **Nodo 2**, como nodo resultado, obtener las alternativas de solución a la consulta de tipo reunión (inner join) y optar por aquella que resulte más óptima.
iii. Para la consulta planteada, en el punto anterior (**Nodo 2 como resultado**), procesar la consulta distribuida usando el tipo de semireunión (semijoin), obtener las posibles soluciones a la consulta.

2) Determinar las instrucciones SQL necesarias, para la definición de las restricciones de integridad, reglas de negocio y triggers requeridos.

a- Crear la tabla **Clientes** de cajas de ahorros en pesos de una entidad bancaria, con los siguientes atributos: Nro. de cuenta de caja de ahorros, CUIL, Apellido y Nombres, código de localidad, situación crediticia y saldo, teniendo en cuenta lo siguiente:
- Nro. de cuenta de caja de ahorros: Clave primaria
- Débitos: puede contener los siguientes valores de 3, 5, 7 ó 9
- Situación crediticia: puede contener los siguientes valores 'A', 'M', 'B'
- Nro. de CUIL: tiene una integridad referencial con la tabla Afip(cuil)
- Cód. de localidad: tiene una integridad referencial con la tabla Provincia(localidad)

b- Crear la tabla **Créditos**, donde se registrarán aquellos importes que se acreditarán en los saldos de las cajas de ahorros en pesos, sus atributos son: Nro. de cuenta de caja de ahorros, denominación, código de empresa e importe de crédito, teniendo en cuenta lo siguiente:
- Nro. de cuenta de caja de ahorros: Clave primaria
- Código de empresa: tiene una integridad referencial con la Empresa(cód_empresa)
- Contemplar una restricción de integridad (**Constraint**) de nombre "Control_Cuenta" de tal manera que nro. de cuenta de caja de ahorros, tenga una referencia externa a la tabla Clientes(nro_caja_ahorro)

c- Crear la tabla **Nuevos_clientes**, donde se registrarán aquellos clientes que han sido dados de alta, con los siguientes atributos: Nro. de cuenta de caja de ahorros, cuil, usuario y fecha de alta.

d- Crear un **trigger** de nombre **Alta_clientesA**, que permita insertar en la tabla **Nuevos_clientes**, los datos de aquel cliente dado de alta y cuya situación crediticia sea igual a 'A', donde las columnas usuario y fecha de alta de esta tabla, se grabarán con las variables del sistema USER y SYSDATE.

El evento que dispara al trigger es **INSERT** en la tabla Clientes, y su tiempo de acción es **AFTER**.

e- Crear un **trigger** de nombre **Nuevo_saldo**, que permita ir actualizando el saldo del Nro. de cuenta de caja de ahorros de la tabla Clientes con el importe de crédito de la tabla Créditos (debería de incrementarse el saldo de la caja de ahorros de la tabla Clientes con el importe de crédito proveniente de la tabla Créditos).

El evento que dispara al trigger es **INSERT** de la tabla Créditos, y su tiempo de acción es **BEFORE**.

Resoluciones

Punto 1

a- Longitud de la relación

Longitud de registros * cantidad de registros

Nodo abonados

15700 registros/tuplas * 81 bytes = 1271700 bytes

Nodo servicios

4800 registros/tuplas *50 bytes = 240000 bytes

b-

i- Nodo 3 como resultado

(apellidos_y_nombres + denominación_servicio_contratado) * cant_abonados

(40 + 40) * 15700 = 80 * 15700 = 1256000 bytes

Opción 1

Transferencia de la relación al nodo 3 y ahí realizar el join

Relación de abonados + Relacion de servicios

1271600 bytes + 240000 bytes = **1511600** bytes transferidos

Opción 2

Transferencia de la relación abonados a nodo servicios y ahí realizar la unión y después enviarla al nodo 3

(15700 registros/tuplas de abonados + 1256000 bytes de la consulta) = **1271700** bytes transferidos

Opción 3

Transferencia de la relación servicios al nodo abonados y ahí realizar el join y después enviarla al nodo 3

(4800 registros/tuplas de servicios + 1256000 bytes de la consulta) = **1260800** bytes transferidos

De estas 3 opciones la mas recomendable y que minimizara la transferencia de bytes al mínimo es la opción 3, transferir servicios a abonados, realizar el join y pasar al nodo 3

ii- Nodo 2 como resultado

Opción 1

Se transfiere Abonados a Servicios

15700 registros/tuplas * 81 bytes = **1271700 bytes transferidos**

Opción 2

Se transfiere de Servicios a Abonados, se realiza el join y luego se envia el resultado al nodo 2

240000 bytes de servicios + 1256000 bytes de la consulta = **1496000 bytes transferidos**

De estas 2 opciones evaluadas, la mas conveniente es la opción 1, pasar abonados a servicios y ahí realizar el join

iii- Consulta con semijoin

1- **3 bytes longitud tipo servicio * 4800 tuplas/registros = 14400 bytes transferidos**

2- (40 bytes apellidos y nombres + 40 denominacion servicio contratado) * 15700 registros/tuplas = 1256000 bytes transferidos
3- 14400 bytes + 1256000 bytes = 1270400 bytes transferidos

Esta opción es mas recomendable que las anteriores

Punto 2

Codigo hecho en MySql

```
--Creacion tabla clientes
CREATE TABLE Clientes )
 nroCuenta INT NOT NULL,
 CUIL INT NOT NULL,
 apeYNom VARCHAR(200) NOT NULL,
 codLocalidad INT NOT NULL,
 sitCrediticia CHAR NOT NULL,
 saldo FLOAT NOT NULL,
 debitos INT NOT NULL,
 --Constraint clave primaria
 CONSTRAINT PK_Clientes_nroCuenta PRIMARY KEY (nroCuenta),
 --Constraint check DEBITOS
 CONSTRAINT CK_DEBITOS CHECK (debitos in (3, 5, 7, 9)),
 --Constraint situacion crediticia
 CONSTRAINT CK_controlSituacion CHECK (sitCrediticia IN ('A', 'M', 'B')),
 --CONSTRAINT FOREIGN KEY AFIP
 CONSTRAINT FK_Clientes_afip_cuil FOREIGN KEY (CUIL) REFERENCES Afip(cuil),
 --CONSTRAINT FOREIGN KEY PROVINCIA
 CONSTRAINT FK_Clientes_Provincia_codLocalidad FOREIGN KEY (codLocalidad)
REFERENCES PROVINCIA(localidad),
)

--Creacion tabla Creditos
CREATE TABLE Creditos (
 nroDeCuenta INT NOT NULL,
 denominacion VARCHAR(100) NOT NULL,
 codigoEmpresa INT NOT NULL,
 importeCredito FLOAT NOT NULL,
 --CONSTRAINT CLAVE PRIMARIA
 CONSTRAINT PK_Creditos_nroCuenta PRIMARY KEY (nroDeCuenta),
 --Constraint clave foranea
 CONSTRAINT FK_Creditos_Empresa FOREIGN KEY (codigoEmpresa) REFERENCES
Empresa(cod_Empresa),
 --Constraint
 CONSTRAINT Control_cuenta FOREIGN KEY (nroDeCuenta) REFERENCES Clientes
(nroCuenta),
)

--Creacion de la tabla nuevos Nuevos_clientes
CREATE TABLE Nuevos_clientes (
 nroCuentaCajaAhorro INT NOT NULL,
 CUIL INT NOT NULL,
 usuario VARCHAR(200) NOT NULL,
 fecha_de_alta timestamp NOT NULL,
)

--Creacion del Trigger Alta_clientesA
CREATE TRIGGER Alta_clientesA
AFTER INSERT ON Clientes
FOR EACH ROW
BEGIN
 --Se verifica si la situación crediticia es 'A'
 IF NEW.sitCrediticia = 'A' THEN
  --Si es se insertan los datos en la tabla Nuevos_clientes
  INSERT INTO Nuevos_clientes (nroCuentaCajaAhorro, CUIL, usuario, fecha_de_alta)
  VALUES (NEW.nroCuenta, NEW.CUIL, USER(), SYSDATE());
 END IF;
END

--Creacion del trigger Nuevo_saldo
CREATE TRIGGER Nuevo_saldo
BEFORE INSERT ON Creditos
FOR EACH ROW
BEGIN
 --Se actualiza el saldo del cliente con el importe del crédito
 UPDATE Clientes
 SET saldo = saldo + NEW.importeCredito
```

```
  WHERE nroCuenta = NEW.nroDeCuenta;
END
```