Facultad de Ciencias Exactas y Naturales y Agrimensura

TRABAJO PRACTICO 1, 2, 3 Y 4

BASE DE DATOS 2

INTEGRANTES GRUPO N° 8:

- Acevedo Casere, Nahuel Agustín - DNI: 42.450.133
- Cáceres Braun, Juan Gabriel - DNI: 41015736
- Gómez, Pablo Emanuel - DNI: 40.123.404
- Sandoval, Liz Dorilem - DNI: 41335142

2023

TP 1

Bases de datos 2

1) CREATE TABLE Empleados(
dni int Primary Key,
legajo Char(6) Unique,
ape_y_nom Varchar(60),
cargo Int Check(cargo between 50 and 120),
sueldo Dec(8,2) Check(sueldo <= 90000)
);

- CREATE DOMAIN Documento AS int CHECK (dni > 0);
  ALTER TABLE Empleados ALTER COLUMN dni SET DATA TYPE Documento;

- Create or replace Trigger Control_sueldo
  Before Update of sueldo on Empleados
  FOR EACH ROW
  Begin
  If :new.sueldo > (:old.sueldo * 1.20)
  Then “Error-Debe ir sentencia que cancela la actualización”
  End if
  End;

- Create Trigger Adicional_cargo
  After Insert or Update of cargo on Empleados
  For each row
  Begin
  If :new.cargo = 90
  Then :new.sueldo = :new.sueldo + (Select importe_basico from Importes_basicos where cargo = :new.cargo) * 0.15
  End if
  End;

2) Create table Seguros_autos(
Nro_poliza Char(8) Primary Key,
Dni Char(6) references Clientes(Dni),
Marca_auto Char(7) Check(Marca_auto In ('Ford','Renault','Fiat','Peugeot','VW', 'Toyota', 'Nissan')),
Anio_modelo Char(4) default ('2019'),
Nro_patente Char(10),
Nro_motor Varchar(30),
Periodo_pago Char (6) default ('202001'),
Imp_pagar Dec(10,2) not null,
Foreign Key(marca,anio_modelo) references Vehiculos(marca,anio_modelo)
);

- Create Trigger Facturas_mensuales
  After Insert on Seguros_autos
  For each row
    Declare importe_cuota Dec(10,2)
    Declare nro_cuota integer
    Declare comprobante Char (16)
    Declare fecha_pago_aux Char (8)
    Begin
      importe_cuota= :new.imp_pagar / 3
      nro_cuota = 1
      While nro_cuota < 4
      Loop
      nro_comprobante = :new.Nro_poliza + :new.Periodo_pago
    +convert(char(2), nro_cuota))
      fecha_pago = ´15' + convert(char(2), nro_cuota)) + substring
    (:new.Periodo_pago,3,2)

      Insert into Cuotas_seguros values (nro_comprobante,
    importe_cuota, convert(fecha_pago, date))
      nro_cuota = nro_cuota + 1
      End Loop
    End;

3) Create table Pedidos(
  Nro_orden Char(6) not null unique,
  Fecha_pedido Date default today,
  Cuit_cliente Char(11) not null,
  Cuit_proveedor Char(11) not null,
  Cuit_cliente references Clientes(Cuit_cliente),
  Primary key(Cuit_cliente, nro_orden),
  Nro_producto int,
  Tipo_pedido Char(1) Check(Tipo_pedido In ('M','C','G','H','N')),
  Cant_pedida int Check(Cant_pedida > 0)
  );

- Create Trigger Control_pedido
  Before Insert on Pedidos
  For each row
    Begin
      If :new.cant_pedida > (Select stock_actual from Stock_productos
    where producto = :new.nro_producto)
        Then RollBack
      Else
        Update Stock_productos
          Set stock_actual = stock_actual -
:new.cant_pedida
          where producto= :new.nro_producto
      End if;
    End;

4)

- Alter table Pedidos add Constraint Clave_Producto Foreign Key (nro_producto,tipo_pedido) references Productos(nro_producto,tipo_pedido) Constraint CantidadPermitida check(cant_pedida between 10 and 2000)

  Alter table Pedidos with no check add Constraint ControlCuit check (Cuit_cliente<>Cuit_proveedor)

- Alter table Stock_productos add Constraint ControlStock check (stock_actual < stock_minimo)

5) Create Domain CodigoCliente Int(9) check (CodigoCliente> 0);

```
Create table Clientes(
Cod_cliente CodigoCliente primary key,
Nombre_cli Varchar(40) not null,
Direccion Varchar(35) not null,
Cod_postal Char(5),
Foreign key (cod_postal) references CodPostales(cod_postal),
Importe_base Dec(8,2) not null
);

Create table Facturas(
Nro_factura Char(8) primary key,
Fecha_fact Date,
Cliente_factura CodigoCliente,
Foreign key (cliente_factura) references Clientes(cliente_factura),
Tipo_descuento int(2) check(tipo_descuento between 5 and 20),
Iva int(2) check((iva=15) or (iva=21)),
Importe_factura Dec(10,2) not null
);

Create Trigger Calcular_importe_factura
After Insert or Update importe_base on Clientes
For each row
    Declare resultado Dec(10,2)
    Begin
            resultado = :new.importe_base – ((:new.tipo_descuento *
:new.importe_base)/100)
            resultado = resultado + ((:new.iva * resultado)/100)
            Update Facturas
            Set importe_factura = resultado
            Where cliente_factura = :new.cod_cliente
    End;
```

6)
```
Create table Empleado_baja(
Dni Char(8),
Legajo Char(6),
Cargo number,
Usuario Varchar(15),
Fecha Date
```

```
);

Create trigger Empleado_eliminado
After delete on Empleados
For each row
    Begin
        Insert into Empleados_baja values (:old.Dni, :old.Legajo, :old.Cargo,
User, Sysdate)
    End;

Create trigger Baja_usuario
After delete on Empleados
    Begin
        Delete from Usuarios where usuarios.dni= :old.Dni;
    End;
```

7)
```
Replace trigger Baja_usuario
After delete on Empleados
    Begin
        Delete from Usuarios where usuarios.legajo= :old.legajo;
    End;

Alter trigger Calcular_importe disable;
Alter table Consumos disable all triggers;
```

8)
```
Alter table Facturas Add fecha_vto date;
Alter table Facturas Add fecha_pago date;
Alter table Facturas Add intereses int;

Create Trigger Recargo_factura
After Update fecha_pago on Facturas
For each row
    Begin
        If fecha_pago > fecha_vto then
            Update Facturas
            Set intereses = 5
            Where cliente_factura= :new.cod_cliente
        End if
    End;
```

9)
```
Create Trigger Importe_recargo
After Update intereses on Facturas
For each row
    Begin
        Update Facturas
        Set importe_factura = importe_factura + ((importe_factura *
intereses)/100)
        Where cliente_factura= :new.cod_cliente
    End;
```

10) Create table cursadas(
dni int primary key,
cod_carrera int,
cod_materia int,
anio_cursado int not null,
condicion char(1) (check condicion in ('R', 'P', 'D')),
nota_final int (check between 1 and 10),
constraint control_materia foreign key (cod_carrera, cod_materia)
references plan_carreras (cod_carrera, cod_materia)
);

```
Create trigger condicion_alumno
after insert on cursadas or update nota_final on cursadas
for each row
    begin
        if new.nota_final = 6 then
            update cursadas
            set condicion = ‘R’
            where dni = new.dni
        else if new.nota_final >= 7 then
            update cursadas
            set condicion = ‘P’
            where dni = new.dni
        else
            update cursadas
            set condicion = ‘D’
            where dni = new.dni
        End if
    end;
```

TP 2

Bases de datos 2

1)

Nodo 1 : Clientes

Nro. Cliente	Apellido y Nombres	Dirección	Correo electrónico	Fecha de Nacimiento	Nro. de Sucursal
6 bytes	30 bytes	35 bytes	40 bytes	8 bytes	4 bytes

Contiene: 5000 registros
Longitud del registro: 123 bytes

Nodo 2 : Sucursales

Nro. Sucursal	Nombre de la sucursal	Dirección	Código de Provincia
4 bytes	30 bytes	35 bytes	2 bytes

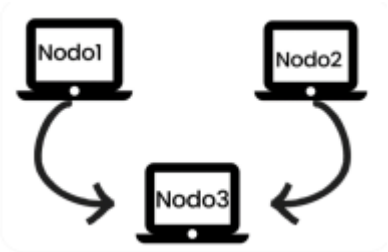
Contiene: 100 registros
Longitud del registro: 71 bytes

a) El tamaño de la relación Clientes (Nodo 1): 5000 tuplas * 123 B= 615000B.
El tamaño de la relación Sucursales (Nodo 2): 100 tuplas * 71 B= 7100B.

b) *Select C.ApellidosNombre, S.NombreSucursal*
From Clientes C Inner join Sucursales S
On C.nroSucursal = S.nroSucursal

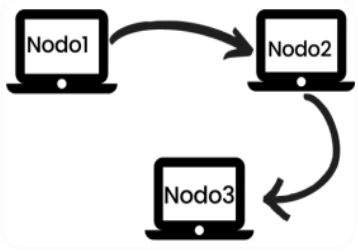
El tamaño de la consulta es: (30B + 30B) * 5000 tuplas = 300000B

i. Nodo 3 (como nodo resultado):
1ra. Opción: Transferimos el Nodo 1 y el Nodo 2 al Nodo 3:



615000B + 7100B = 622100B. transferidos.

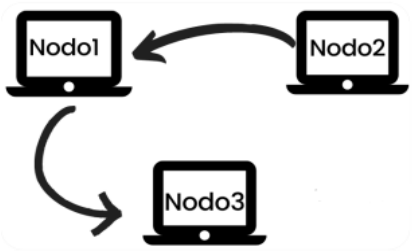
2da. Opción: Se transfiere el Nodo 1 al Nodo 2, donde se ejecuta la sentencia de la consulta y transferimos el resultado al Nodo 3 para su visualización:



615000B + 300000B = 915000B transferidos.

3ra.Opción: Se transfiere la información del Nodo 2 al Nodo 1, donde se ejecuta la

sentencia de la consulta y transferimos el resultado al Nodo 3 para la visualización:



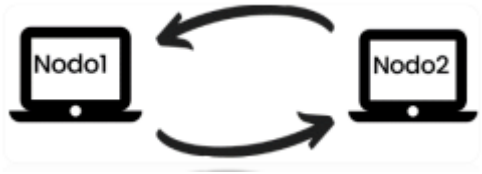
7100B + 300000B = 307100B transferidos.

❖ Luego de analizar las tres opciones, podemos observar que la 3a opción es la estrategia más adecuada, dado que implica menos transferencia de datos.

ii. Nodo 2 (como nodo resultado):
1ra. Opción: Se transfiere el Nodo 1 al Nodo 2, donde se ejecuta la sentencia de la consulta y se visualizará el resultado:

En esta opción el tamaño de bytes transferidos es 615000B, es decir, el tamaño de la relación Clientes.

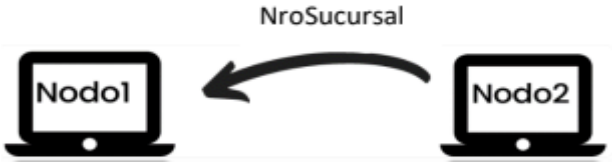
2da. Opción: Se transfiere el Nodo 2 al Nodo 1, y de este mismo se transfiere el resultado de la consulta al Nodo 2 en el cual se visualizará el resultado:



7100B + 300000B = 307100B transferidos.

❖ Siendo esta última opción la estrategia más óptima, dado que implica menor transferencia de los datos.

iii. semi reunión (semi join):



4B * 100 tuplas = 400B transferidos.

(4B + 30B) * 5000 tuplas = 170000B transferidos.

❖ Como podemos observar en semi join transferimos solamente aquellos atributos necesarios para ejecutar nuestra consulta, para obtener el total de datos transferidos debemos sumar ambos resultados:
400B + 170000B = 170400B transferidos.

Resultó más óptima que lo analizado en el punto anterior (ii.).

2)

Nodo 1 : Profesores

DNI	Apellido y Nombres	Correo electrónico	Fecha de Nacimiento	Cód. de Facultad
8 bytes	30 bytes	40 bytes	8 bytes	2 bytes

Contiene: 7000 registros
Longitud del registro: 88 bytes

Nodo 2 : Facultades

Cód. Facultad	Denominación	Dirección	Decano
2 bytes	32 bytes	35 bytes	8 bytes

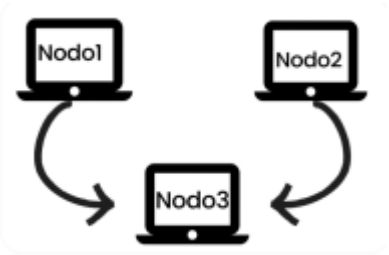
Contiene: 200 registros
Longitud del registro: 77 bytes

a) El tamaño de la relación Profesores (Nodo 1): 7000 tuplas * 88B = 616000B.
El tamaño de la relación Facultades (Nodo 2): 200 tuplas * 77B = 15400B.

b) *Select P.ApellidosNombres, F.Denominacion*
From Profesores P Inner join Facultades F
On P.codFacultad = F.codFacultad

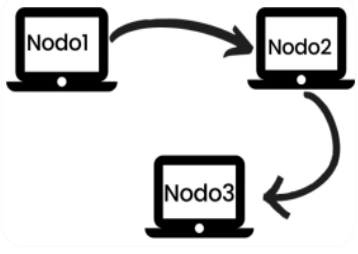
El tamaño de la consulta es: (30B + 32B) * 7000 tuplas = 434000B.

i. Nodo 3 (como nodo resultado):
1ra. Opción: Transferir tanto el Nodo 1 como el Nodo 2 al nodo resultado (Nodo 3):



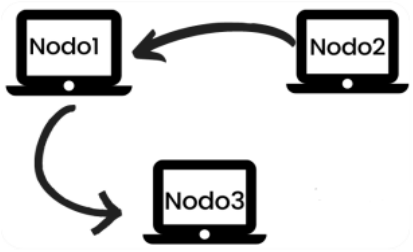
616000B + 15400B = 631400B. transferidos.

2da. Opción: Se transferir el Nodo 1 al Nodo 2, donde se ejecuta la sentencia de la consulta y transferimos el resultado al Nodo 3 para su visualización:



616000B + 434000B = 1050000B transferidos.

3ra.Opción: Transferir el Nodo 2 al Nodo1 , donde se ejecuta la sentencia de la consulta, transfiriendo el resultado al Nodo 3 para la visualización:



15400B + 434000B = 449400B transferidos.

❖ Como podemos observar de las tres opciones, la 3a opción es la estrategia más

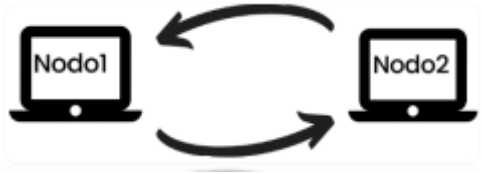
óptima de acuerdo al criterio considerado, siendo esta la que implica menor transferencia de datos.

ii Nodo 2 (como nodo resultado):

1ra. Opción: Se transfiere el Nodo 1 al Nodo 2, donde se ejecuta la sentencia de la consulta y se visualizará el resultado:

En esta opción el tamaño de bytes transferidos es 616000B, es decir, el tamaño de la relación Profesores.

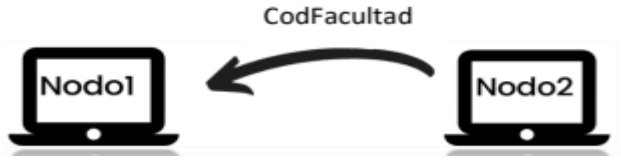
2da. Opción: Se transfiere el Nodo2 al Nodo1, el cual transfiere el resultado de la consulta al Nodo2 para su visualización:



15400B + 434000B = 449400B transferidos.

❖ Como podemos observar esta opción resulta la más óptima, dado que implica menor transferencia de los datos.

iii semi reunión (semi join):



2B * 200 tuplas = 400B transferidos.

(30B + 2B) * 7000 tuplas = 224000B transferidos.

❖ Como podemos observar en semi join transferimos sólo aquellos atributos necesarios para ejecutar nuestra consulta, obteniendo el total de datos transferidos sumando ambos resultados:
400B + 224000B = 224400B transferidos.
Resultando la más óptima incluso que lo obtenido en el punto anterior (2_b_ii.).

3)

Nodo 1 : Remedios

Producto	Nombre Comercial	Precio	Acción terapéutica	Laboratorio
6 bytes	20 bytes	10 bytes	40 bytes	5 bytes

Contiene: 100000 registros Longitud del registro: 81 bytes

Nodo 2 : Laboratorios

Laboratorio	Nombre	Dirección	Sitio web	Gerente	Teléfono
5 bytes	40 bytes	35 bytes	40 bytes	45 bytes	12 bytes

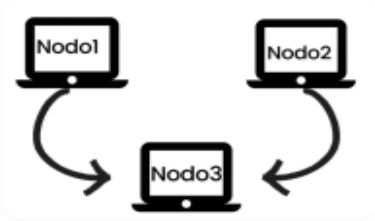
Contiene: 63800 registros Longitud del registro: 177 bytes

a) El tamaño de la relación Remedios (Nodo 1): 100000 tuplas * 81B= 8100000B.
El tamaño de la relación Laboratorios (Nodo 2): 63800 tuplas * 177B= 11292600B.

b) Select R.nombreComercial, L.Nombre
From Remedios R Inner join Laboratorios L
On R.Laboratorio = L.Laboratorio

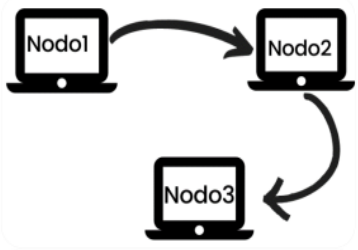
El tamaño de la consulta es: (20B + 40B) * 100000tuplas = 6000000B.

ii. Nodo 3 (como nodo resultado):
1ra. Opción: Transferir tanto el Nodo 1 como el Nodo 2 al nodo resultado (Nodo 3):



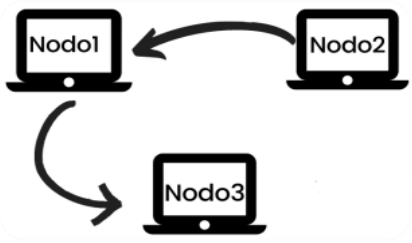
8100000B + 11292600B = 19392600B. transferidos.

2da. Opción: Transferir el Nodo 1 al Nodo 2, donde se ejecuta la consulta y se transfiere el resultado al Nodo 3 para la visualización:



8100000B + 6000000B = 14100000B transferidos.

3ra.Opción: Transferir el Nodo 2 al Nodo 1, donde se ejecuta la consulta, transfiriendo el resultado al Nodo 3 para la visualización:



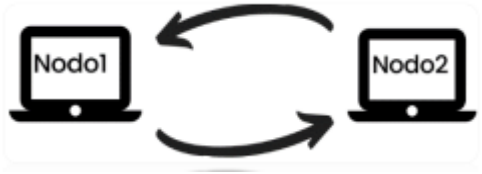
11292600B + 6000000B = 17292600B transferidos.

❖ Como podemos observar de las tres opciones, la 2da opción es la más óptima, teniendo en cuenta el criterio de “minimizar el transporte de bytes, entre los distintos nodos distribuidos”.

ii Nodo2 (como nodo resultado):
1ra. Opción: Se transfiere el Nodo 1 al Nodo 2, donde se ejecutará la consulta y se visualizará el resultado:

El tamaño de bytes transferidos es igual al tamaño de la relación Remedios: 8100000B.

2da. Opción: Se transfiere el Nodo 2 al Nodo 1, el cual transfiere el resultado de la consulta al Nodo 2 para su visualización:



11292600B + 6000000B = 17292600B transferidos.

❖ De las dos opciones, la 1a opción resulta la más óptima, ya que implica menor transferencia de datos.

4)

Nodo 1 : Empleados

DNI	Nombre y Apellido	Fecha Ingreso	Código Departamento	Categoría	Sueldo
8 bytes	25 bytes	8 bytes	3 bytes	2 bytes	10 bytes

Contiene: 3200 registros Longitud del registro: 56 bytes

Nodo 2 : Departamentos

Código	Nombre	Dni Jefe	Sector	Área	Mail
3 bytes	40 bytes	8 bytes	4 bytes	35 bytes	40 bytes

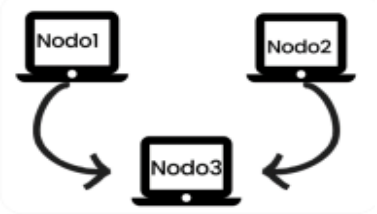
Contiene: 1500 registros Longitud del registro: 130 bytes

a) El tamaño de la relación Empleados (Nodo 1): 3200 tuplas * 56B= 179200B.
El tamaño de la relación Departamentos (Nodo 2): 1500 tuplas * 130B= 195000B.

b) Select NombreApellido, Nombre
From Empleados Inner join Departamentos
On codDepartamento = codigo

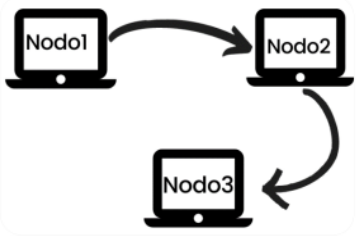
El tamaño de la consulta es: (25B + 40B) * 3200 tuplas = 208000B.

ii Nodo 3 (como nodo resultado):
1ra. Opción: Transferir tanto el Nodo 1 como el Nodo 2 al nodo resultado (Nodo 3):



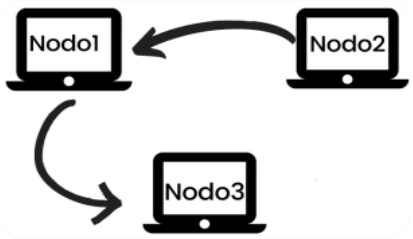
179200B + 195000B = 374200B. transferidos.

2da. Opción: Transferir el Nodo 1 al Nodo 2, donde se ejecuta la consulta y se transfiere el resultado al Nodo 3 para la visualización:



179200B + 208000B = 387200B transferidos.

3ra.Opción: Transferir el Nodo 2 al Nodo 1, donde se ejecuta la consulta, transfiriendo el resultado al Nodo 3 para la visualización:



195000B + 208000B = 403000B transferidos.

❖ Como podemos observar de las tres opciones, la 1a opción es la más óptima, teniendo en cuenta el criterio de “minimizar el transporte de bytes, entre los distintos nodos distribuidos”.

ii Nodo2 (como nodo resultado):
1ra. Opción: Se transfiere el Nodo 1 al Nodo 2, donde se ejecutará la consulta y se visualizará el resultado:

El tamaño de bytes transferidos es igual al tamaño de la relación Empleados: 179200B.

2da. Opción: Se transfiere el Nodo2 al Nodo 1, el cual transfiere el resultado de la consulta al Nodo 2 para su visualización:

195000B + 208000B = 403000B transferidos.

❖ De las dos opciones, la 1a opción resulta la más óptima, ya que implica menor transferencia de datos.

iii semi reunión (semi join):

3B * 1500 tuplas = 4500B transferidos.

(25B + 3B) * 3200 tuplas = 89600B transferidos.

❖ Como podemos observar en semi join transferimos sólo aquellos atributos necesarios para ejecutar la consulta, obteniendo el total de datos transferidos sumando ambos resultados:
4500B + 89600B = 94100B transferidos.
Resultando la más óptima incluso que lo obtenido en el punto anterior (4_b_ii.).

5)

Nodo 1 : Vehículos

Nro. Patente	Nombre y Apellido del Titular	DNI	Dirección	Código de Modelo	Registro Seccional
8 bytes	25 bytes	8 bytes	25 bytes	5 bytes	4 bytes

Contiene: 1300000 registros Longitud del registro: 73 bytes

Nodo 2 : Modelos (de los vehículos)

Código	Nombre Modelo	Año	País de origen	Precio
5 bytes	20 bytes	4 bytes	20 bytes	13 bytes

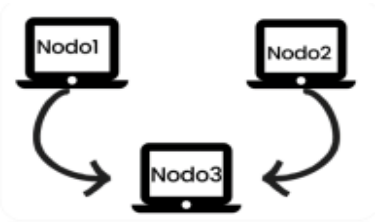
Contiene: 15300 registros Longitud del registro: 62

a) El tamaño de la relación Vehículos (Nodo 1): 1300000 tuplas * 73B= 94900000B.
El tamaño de la relación Modelos (Nodo 2): 15300 tuplas * 62B= 948600B.

b) Select V.NomApeTitular, M.NombreModelo, M.Año
From Vehículos V Inner join Modelos M
On V.codModelo = M.codigo

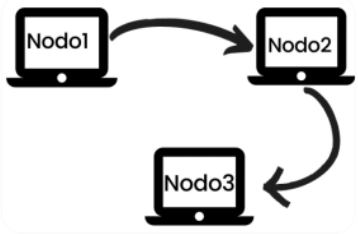
El tamaño de la consulta es: (25B + 20B + 4B) * 1300000 tuplas = 63700000B.

ii Nodo 3 (como nodo resultado):
1ra. Opción: Transferir tanto el Nodo 1 como el Nodo 2 al nodo resultado (Nodo 3):



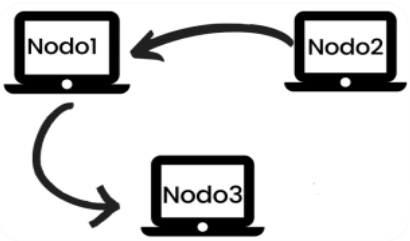
94900000B + 948600B = 95848600B. transferidos.

2da. Opción: Transferir el Nodo 1 al Nodo 2, donde se ejecutará la consulta, transfiriendo el resultado al Nodo 3 para la visualización:



94900000B + 63700000B = 158600000B transferidos.

3ra.Opción: Transferir el Nodo 2 al Nodo 1, donde se ejecuta la consulta y se transfiere el resultado al Nodo 3 para la visualización:



948600B + 63700000B = 64648600B transferidos.

❖ Como podemos observar de las tres opciones, la 3a opción es la más óptima, teniendo en cuenta el criterio de “minimizar el transporte de bytes, entre los distintos nodos distribuidos”.

ii Nodo 2 (como nodo resultado):
1ra. Opción: Se transfiere el Nodo 1 al Nodo 2, donde se ejecutará la consulta y se visualizará el resultado:

El tamaño de bytes transferidos es igual al tamaño de la relación Vehículos: 94900000B.

2da. Opción: Se transfiere el Nodo 2 al Nodo 1, el cual ejecutará la consulta y transferirá
el resultado al Nodo 2 para la visualización:

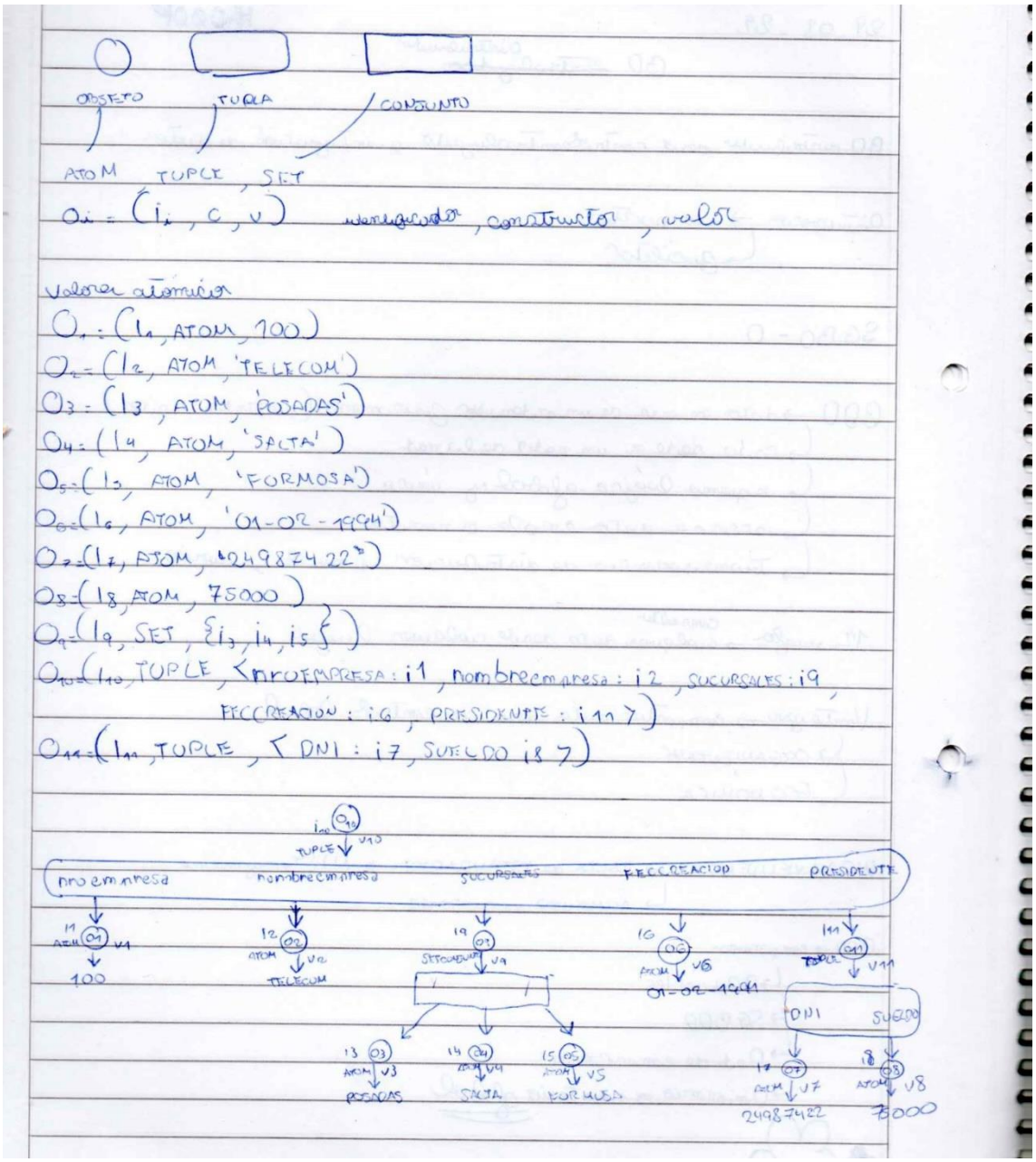
948600B + 63700000B = 64648600B transferidos.

❖ De las dos opciones, la 2a opción resulta la más óptima, ya que implica menor transferencia de datos.

TP 3

Bases de datos 2

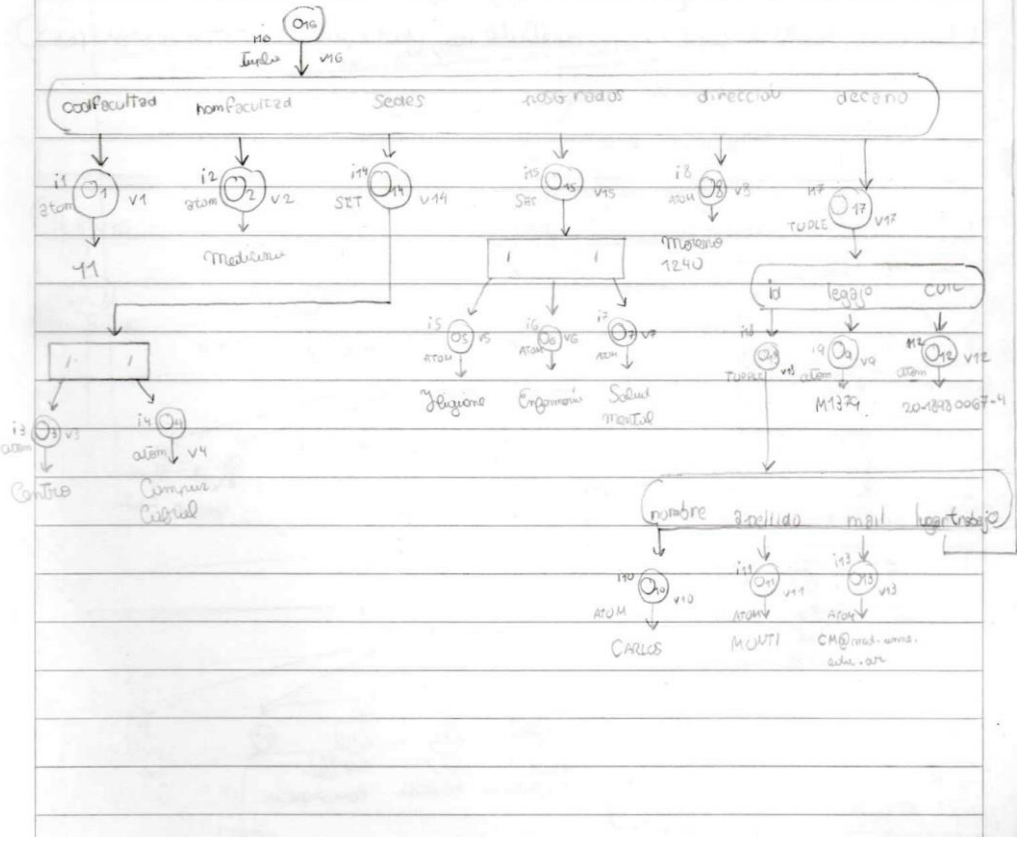
PUNTO 1:



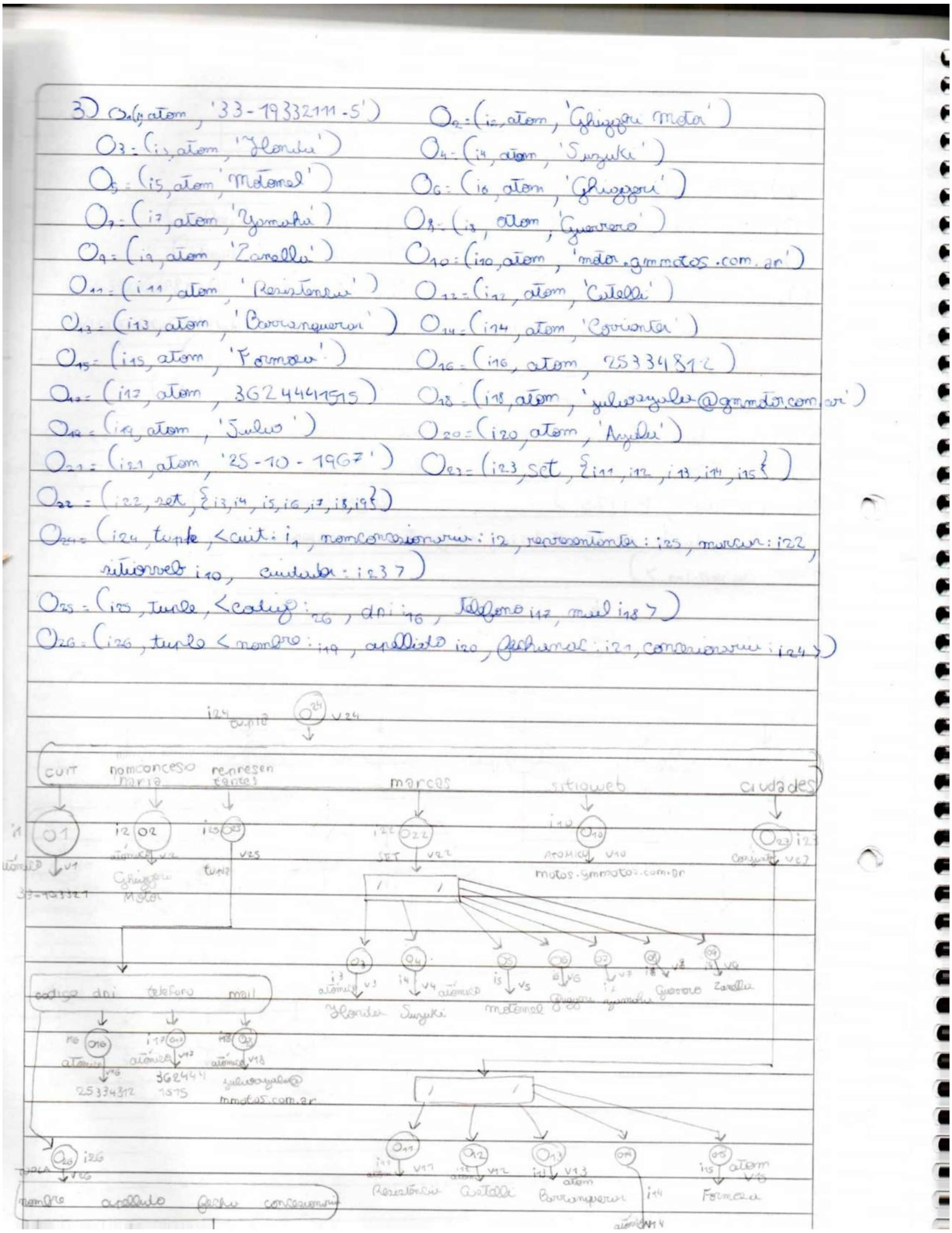
PUNTO 2:

2) Valores atómicos

$O_1 = (i_1, atom, 11)$
$O_2 = (i_2, atom, 'Medicina')$
$O_3 = (i_3, atom, 'Cantro')$
$O_4 = (i_4, atom, 'Campus Central')$
$O_5 = (i_5, atom, 'Higiene')$
$O_6 = (i_6, atom, 'Enfermería')$
$O_7 = (i_7, atom, 'Salud Mental')$
$O_8 = (i_8, atom, 'Moreno 1240')$
$O_9 = (i_9, atom, 'M1378')$
$O_{10} = (i_{10}, atom, 'Carlos')$
$O_{11} = (i_{11}, atom, 'Monti')$
$O_{12} = (i_{12}, atom, '20-18980067-4')$
$O_{13} = (i_{13}, atom, 'cm@med.unne.edu.ar')$
$O_{14} = (i_{14}, set, \{i_3, i_4\})$
$O_{15} = (i_{15}, set, \{i_5, i_6, i_7\})$
$O_{16} = (i_{16}, Tuple, <codfacultad: i_1, nomfacultad: i_2, sedes: i_{14}, autoridades: i_{15}, direccion: i_8, decano: i_{17}>)$
$O_{17} = (i_{17}, Tuple, <id: i_{18}, legajo: i_9, cuil: i_{12}>)$
$O_{18} = (i_{18}, Tuple, <nombre: i_{10}, apellido: i_{11}, mail: i_{13}, lugarTrabajo: i_{16}>)$



PUNTO 3:



3) O₁ (i₁, atom, '33-19332111-5')

O_2: (i2, atom, 'Ghizzoni Motor')

O_3: (i3, atom, 'Honda')

O_4: (i4, atom, 'Suzuki')

O_5: (i5, atom, 'Motomel')

O_6: (i6, atom, 'Ghizzoni')

O_7: (i7, atom, 'Yamaha')

O_8: (i8, atom, 'Guerrero')

O_9: (i9, atom, 'Zanella')

O_{10}: (i10, atom, 'motor.gmmotos.com.ar')

O_{11}: (i11, atom, 'Resistencia')

O_{12}: (i12, atom, 'Castelli')

O_{13}: (i13, atom, 'Barranqueras')

O_{14}: (i14, atom, 'Corrientes')

O_{15}: (i15, atom, 'Formosa')

O_{16}: (i16, atom, 25334812)

O_{17}: (i17, atom, 3624441515)

O_{18}: (i18, atom, 'julioayala@gmmotor.com.ar')

O_{19}: (i19, atom, 'Julio')

O_{20}: (i20, atom, 'Ayala')

O_{21}: (i21, atom, '25-10-1967')

O_{23}: (i23, set, {i11, i12, i13, i14, i15})

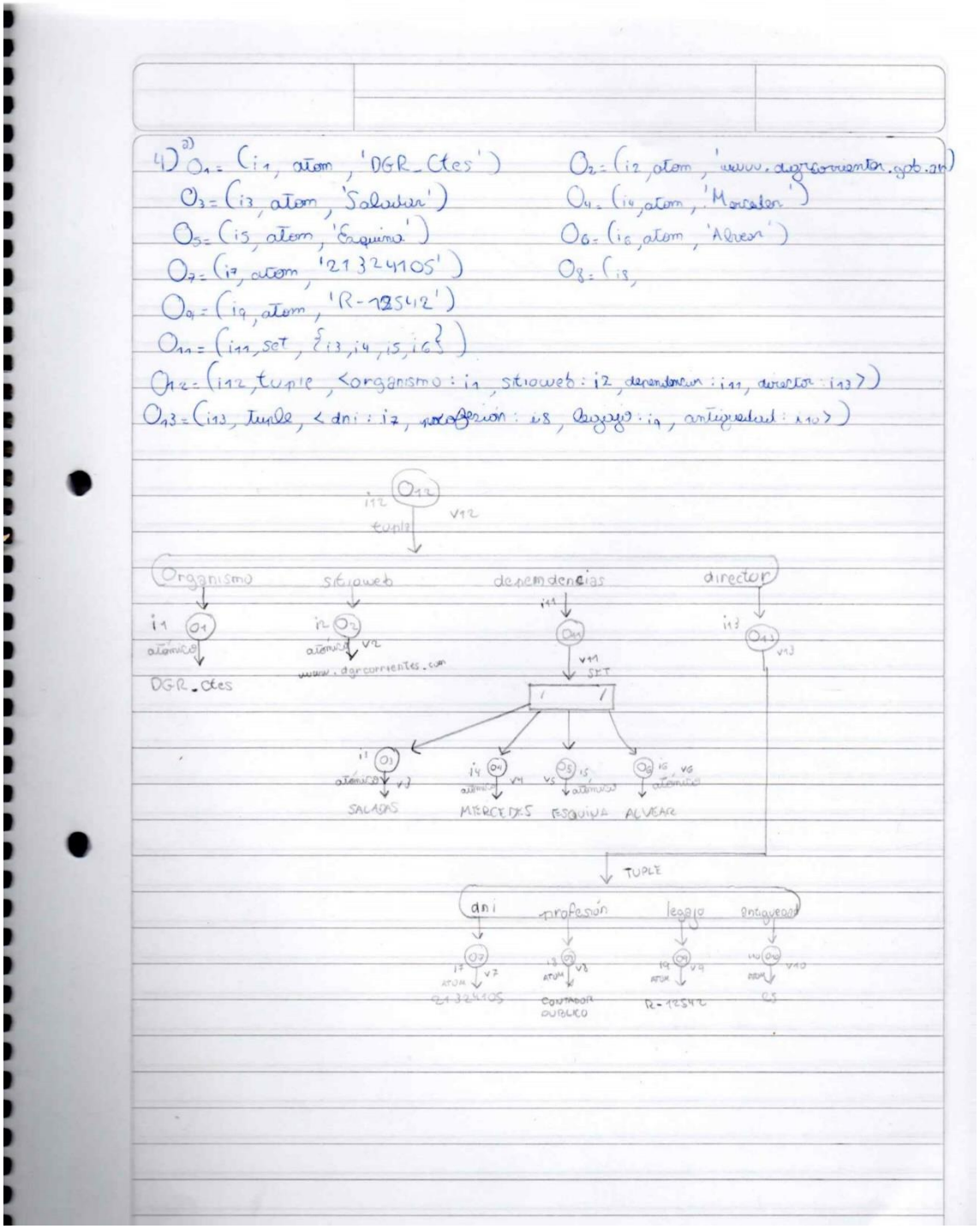
O_{22} = (i22, set, {i3, i4, i5, i6, i7, i8, i9})

O_{24}: (i24, tuple, <cuit: i1, nomconcesionaria: i2, representante: i25, marcas: i22, sitioweb i10, ciudades: i23>)

O_{25}: (i25, tuple, <codigo: i26, dni i16, telefono i17, mail i18>)

O_{26}: (i26, tuple <nombre: i19, apellido i20, fechanac: i21, concesionaria: i24>)

PUNTO 4:


a)
O1 = (i1, atom, 'DGR_Ctes')
O2 = (i2, atom, 'www.dgrcorrientes.gob.ar')
O3 = (i3, atom, 'Saladas')
O4 = (i4, atom, 'Mercedes')
O5 = (i5, atom, 'Esquina')
O6 = (i6, atom, 'Alvear')
O7 = (i7, atom, '21324105')
O8 = (i8,
O9 = (i9, atom, 'R-12542')
O11 = (i11, set, {i3, i4, i5, i6})
O12 = (i12, tuple, <organismo : i1, sitioweb : i2, dependencias : i11, director : i13>)
O13 = (i13, tuple, <dni : i7, profesión : i8, legajo : i9, antigüedad : i10>)
tupla
organismo
sitioweb
dependencias
director
DGR_Ctes
www.dgrcorrientes.com
SET
SALADAS
MERCEDES
ESQUINA
ALVEAR
TUPLE
dni
profesión
legajo
antigüedad
21324105
CONTADOR PUBLICO
R-12542
05

PUNTO 5:

```
class Persona (extent personas key DNI) {
        attribute string DNI;
        attribute struct nombrePersona {
                string nombre1,
                string nombre2,
                string apellido1,
                string apellido2
         } nombre;
         attribute date fechanac;
         attribute struct direPersona {
                 string calle,
                 integer numero,
                 string codpostal
        } direccion;
}


class Profesor extent Persona (extent profesores) {
        attribute integer categoria;
        attribute string despacho;
        attribute float salario;
        attribute string telefono;

        relationship Departamento trabaja_en
                 inverse Departamento::tiene_profesores;
}


class Estudiante extent Persona (extent estudiantes) {
        attribute integer LU;

        relationship set <Calificacion>materias_cursadas
                 inverse Calificacion::estudiante;
}


class Departamento (extent departamentos key nombre){
        attribute string nombre;
        attribute string despacho;
        attribute string telefono;
        attribute Profesor director;
        attribute string area;

        relationship set tiene_profesores
                inverse Profesor::trabaja_en;
        relationship set oferta
                 inverse Curso::ofertado_por;
}
```

```
class Curso (extent cursos key numero) {
attribute string nombre;
        attribute integer numero;
        attribute string descripcion;

        relationship set de_dictado
                inverse Dictado::tiene_cursos;
        relationship Departamento ofertado_por
                inverse Departamento::oferta;
}


class Dictado (extent dictados) {
attribute short numero;
attribute integer año;

relationship set estudiante_de
        inverse Calificacion::cursa;
relationship set tiene_cursos
        inverse Curso::de_dictado;
}


class Calificacion (extent calificaciones) {
        attribute float nota;

        relationship Dictado cursa
                inverse Dictado::estudiante_de;
         relationship Estudiante estudiante
                inverse Estudiante::materias_cursadas;
}
```

PUNTO 6:

```
class Libro (key ISBN) {
         attribute string ISBN;
         attribute string titulo;
         attribute integer anio;
         attribute float precio;
         relationship Editorial publicadoPor
                inverse Editorial::publica;
        relationship Autor escritoPor
                inverse Autor::escribe;
};
```

```
class Editorial (key nombre) {
        attribute string nombre;
        attribute string sitioWeb;
        relationship Libro publica
                inverse Libro::publicadoPor;
};


class Autor (key nombreApellido) {
        attribute Struct nombreApellido (
                string nombre,
                string apellido
        );
        attribute string email;
        attribute Struct direccion(
                string calle,
                integer numero,
                integer codPostal
        );
        relationship Libro escribe
                inverse Libro::escritoPor;
};
```

PUNTO 7:

```
class Facultad (key nombre) {
        attribute string nombre;
         attribute string sitioWeb;
         attribute string direccion;
         attribute string telefono;

         relationship Universidad administradoPor
                inverse Universidad::depende;
         relationship Alumno tieneAlumnos
                inverse Alumno::selnscribe;
        relationship Decano dirigidoPor
                inverse Decano::dirige;
};


class Universidad (key nombre) {
        attribute string nombre;
        attribute string sitioWeb;
        attribute string direccion;
        attribute string telefono;

        relationship Facultad depende
                inverse Facultad::administradoPor;
};
```

```
class Alumno (key LU) {
        attribute integer LU;
         attribute string nombre;
         attribute string direccion;
         attribute string telefono;
         attribute string mail;
         attribute string carrera;
         attribute Enum genero ('M', 'F');

         relationship Facultad selnscribe
                  inverse Facultad::tieneAlumnos;
};


class Decano (key dni) {
        attribute integer dni;
         attribute integer cuil;
         attribute string nombre;
         relationship Facultad dirige inverse Facultad::dirigidoPor;
};
```

TP 4

Bases de datos 2

1) Determinar las instrucciones Sql3 necesarias para:

a) Crear la tabla Libros, con los atributos: Código de libro, Título, Autor, Año de edición y Precio.

Contemplar en la definición, la inclusión de una columna identidad para código de libro, con valor de inicio/mínimo en 1, incremento en 1, sin ciclo y hasta el máximo valor de 5000.

Solución:

```
CREATE TABLE Libros (
        código_libro INTEGER GENERATED ALWAYS AS
IDENTITY (
                START WITH 1
                INCREMENT BY 1
                MINVALUE 1
                MAXVALUE 5000
                NO CYCLE
                ),
        título VARCHAR (100),
        autor VARCHAR (100),
        año_edicion INTEGER,
        precio DECIMAL (9,2)
);
```

b) Crear la tabla Editorial, con los atributos: Código de libro, Título de la obra, Descripción, Autor, Foto de la portada del libro, Vídeo de presentación y Año edición.

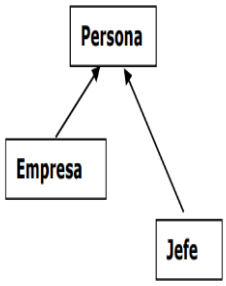
Considerar en la definición, a los siguientes atributos como objetos grandes, de longitud variable, con los siguientes tipos y tamaños máximos:

- Descripción – carácter – 40K
- Foto de la portada – binario – 2M
- Video de presentación – binario – 1G

Solución:

```
CREATE TABLE Editorial (
        código_libro INTEGER,
        título VARCHAR (100),
        descripción CLOB (40K),
        autor VARCHAR (100),
        foto_portada BLOB (2M),
        video_presentación BLOB (1G),
        año_edicion INTEGER
);
```

2) Determinar las instrucciones Sql3 necesarias para crear las jerarquías de tipos de múltiples niveles y tipos predefinidos:



-Definir los siguientes, como tipos predefinidos:

Tipo Persona:

DNI, Apellido y Nombres, Dirección, Ciudad, Fecha nacimiento

Tipo Empresa:

Empleado referencia a tipo persona, Mail, Código Postal, sitio web

Tipo Jefe:

Director referencia a tipo persona, Mail, Área, Sueldo, Antigüedad

-Definir la tabla Empleados según el tipo Empresa.

Solución:

```
CREATE TABLE empleados OF empresa;

CREATE TYPE persona AS (
        dni INTEGER,
        apellidos_y_nombres VARCHAR (100),
        dirección VARCHAR (100),
        ciudad VARCHAR (100),
        fecha_nacimiento DATE
) NOT INSTANTIABLE NOT FINAL;

CREATE TYPE empresa AS (
        empleado REF (persona),
        mail VARCHAR (100),
        código_postal INTEGER,
        sitio_web VARCHAR (200)
) INSTANTIABLE FINAL;

CREATE TYPE jefe AS (
        director REF (persona),
        mail VARCHAR (100),
        área VARCHAR (100),
        sueldo DECIMAL (9,2),
        antigüedad INTEGER
) INSTANTIABLE FINAL;
```

3) Determinar las instrucciones Sql3 necesarias para definir tipos construidos, colecciones:

a) Crear la tabla Universidad, con los atributos: Nombre, Rector, Dirección, Facultades, Cód. postal y sitio web.

- Definir Facultades, como carácter de longitud 15 y de tipo vector de 9 elementos.
- Variante: definir Facultades como tipo multiconjunto.

Solución:

```
CREATE TABLE universidad (
        nombre VARCHAR (100),
        rector VARCHAR (100),
        dirección VARCHAR (100),
        facultades VARCHAR (15) ARRAY [9],
        código_postal INTEGER,
        sitio_web VARCHAR (200)
);
```

Variante:

```
CREATE TABLE universidad (
        nombre VARCHAR (100), rector
        VARCHAR (100), dirección VARCHAR
        (100), facultades VARCHAR (15) MULTISET,
        código_postal INTEGER,
        sitio_web VARCHAR (200)
);
```

b)Especificar la sentencia necesaria, para insertar en la tabla Universidad los siguientes valores para cada atributo:

Nombre: Universidad Nacional del Nordeste
Rector: María Veiravé
Dirección: 25 de Mayo 868
Facultades (como vector): Medicina, Humanidades, Exactas, Veterinaria, Económicas, Agrarias, Arquitectura, Derecho, Odontología
Cód. postal: 3400
Sitio web: www.unne.edu.ar

• Variante: indicar que cambios hay que realizar a la sentencia anterior, si insertamos los valores en Facultades como multiconjunto.

Solución:

```
INSERT INTO universidad (nombre, rector, dirección, facultades,
código_postal, sitio_web) VALUES (“Universidad Nacional del
Nordeste”, “María Veraivé”, “25 de Mayo 868”, [“Medicina”,
“Humanidades”, “Exactas”, “Veterinaria”, “Economicas”,
“Agrarias”, “Arquitectura”, “Derecho”, “Odontologia”], 3400,
“www.unne.edu.ar”)
```

Variante:

INSERT INTO universidad (nombre, rector, dirección, facultades, código_postal, sitio_web) VALUES (“Universidad Nacional del Nordeste”, “María Veraivé”, “25 de Mayo 868”, Multiset [“Medicina”, “Humanidades”, “Exactas”, “Veterinaria”, “Economicas”, “Agrarias”, “Arquitectura”, “Derecho”, “Odontologia”], 3400, “www.unne.edu.ar”)

c) Especificar la sentencia necesaria, para obtener una consulta a la tabla Universidad, de los atributos: Nombre, Rector y de las 2 primeras facultades almacenadas.

Solución:

SELECT nombre, rector, facultades [0], facultades [1] FROM universidad

4) Determinar las instrucciones Sql3 necesarias para definir tipos construidos, filas:

- Crear la tabla Banco, con los atributos: Identificación, Razón social, Presidente, Dirección, Teléfono y sitio web.

Definir los siguientes atributos de tipo fila, considerando para cada uno de ellos lo siguiente:

- Identificación
  - Número de banco ante el BCRA
  - CUIT
- Razón social
  - Nombre comercial
  - Tipo de empresa
  - Condición tributaria ante la AFIP
- Presidente
  - Nombres
  - Apellido
  - DNI
  - Mail
- Dirección
  - Calle
  - Número (altura)
  - Ciudad
  - Código postal
- Teléfono
  - Prefijo
  - Número

Solución:

```
CREATE TABLE banco (
        identificación ROW (
                numero_bcra INTEGER,
                cuit INTEGER
                ),
        razón_social ROW (
                nombre_comercial VARCHAR (100),
                tipo_empresa VARCHAR (30),
                condición_afip VARCHAR (30)
                ),
        presidente ROW (
                nombres VARCHAR (50),
                apellidos VARCHAR (50),
                dni INTEGER,
                mail VARCHAR (50)
                ),
        dirección ROW(
                calle VARCHAR (50),
                altura INTEGER,
                ciudad VARCHAR (50),
                codigo_postal INTEGER
                ),
        telefono ROW(
                prefijo INTEGER,
                numero_telefono INTEGER
                ),
        );
```

5) Determinar las instrucciones Sql3 necesarias para definir el tipo y métodos requeridos:

- Crear el tipo Empleado, con los atributos: DNI, Apellido y Nombre, Antigüedad laboral, Dirección, Cargo y Sueldo.

Solución:

```
CREATE TYPE empleado AS (
        dni INTEGER,
        apellidos_y_nombres VARCHAR (100),
        METHOD antigüedad() RETURNS
    INTEGER,
        dirección VARCHAR (100),
        cargo VARCHAR (50),
        METHOD sueldo() RETURNS DECIMAL
    (9,2)
) INSTANTIABLE FINAL;
```

- Representar las sentencias necesarias para definir los métodos de Antigüedad y Sueldo, donde:

Antigüedad laboral se obtiene como: Año actual – Año ingreso
Sueldo: Básico + Adicional por título + Escolaridad – Aporte Jubilatorio

Solución:

```
CREATE METHOD antigüedad() FOR empleado
BEGIN
    RETURN (año_actual-año_ingreso)
END;

CREATE METHOD sueldo() FOR empleado
BEGIN
    RETURN (basico+adicional_titulo+escolaridad-
aporte_jubilatorio)
END;
```

6) Determinar las instrucciones Sql3 necesarias para definir el tipo y métodos requeridos:

- Crear el tipo Producto, con los atributos: Código producto, Denominación, Stock actual, Stock mínimo, precio de fábrica y precio al consumidor.

Solución:

```
CREATE TYPE producto AS (
        codigo_producto INTEGER,
        denominación VARCHAR (100),
        stock_actual INTEGER,
        stock_minimo INTEGER,
        precio_fabrica DECIMAL (9,2),
    METHOD precio_consumidor() RETURNS DECIMAL
(9,2)
        ) INSTANTIABLE FINAL;
```

- Representar las sentencias necesarias para definir el método de Precios, donde:
Precio al consumidor: Precio de fábrica + 15% sobre el precio de fábrica

Solución:

```
CREATE METHOD precio_consumidor() FOR empleado
 BEGIN
        RETURN (precio_fabrica*1.15)
 END;
```

7) Determinar las instrucciones Sql3 necesarias para definir el tipo estructurado definido por el usuario:

- Crear el tipo direccion_postal, con los atributos: calle, número, provincia y código postal.

Solución;

```
CREATE TYPE dirección_postal AS (
        calle VARCHAR(50),
        numero INTEGER,
        provincia  VARCHAR(70),
        codigo_postal INTEGER
);
```

- Crear la tabla personas, con los atributos: dni (clave primaria), nombre, apellidos, fecha de nacimiento, teléfonos y dirección.

El atributo dirección se define como el tipo direccion_postal.
El atributo teléfonos como de tipo colección text[], como variante de los tipos colecciones array o multiset.

Solución:

```
CREATE TABLE personas (
        dni INTEGER PRIMARY KEY,
        nombres VARCHAR(100),
        apellidos VARCHAR(100),
        fecha_nacimiento DATE,
        telefonos TEXT[],
        dirección dirección_postal
);
```