Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Cátedra: Base de Datos II

Año: 2020

Trabajos Prácticos 1ra. Parte (1,2,3,4)

Grupo 7

Alumno: Fernández Paulina Belén
L.U. N°: 51417

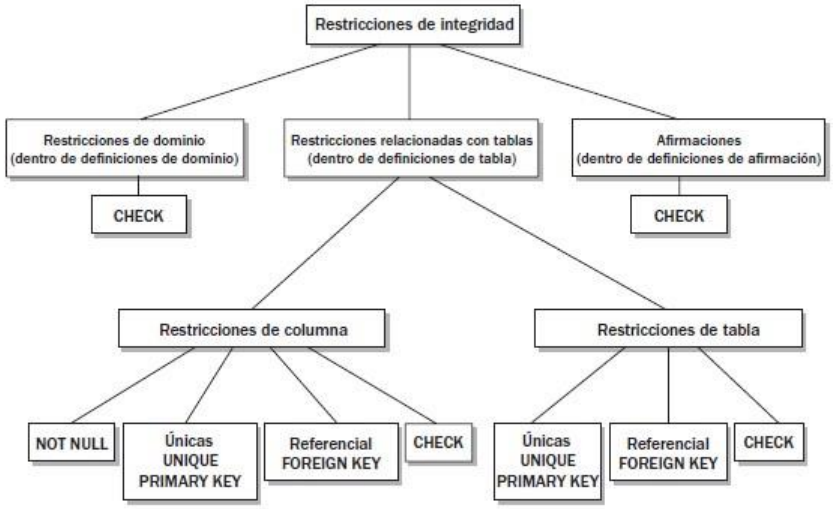
Alumno: Lucia del Valle Ledezma
L.U. N°: 43293

Alumno: Pruyas Araceli de los Angeles Itati
L.U. N°: 52323

Alumno: Giuliano Vallejos Santajuliana
L.U. N°: 52643

1.1 Integridad de Datos

Existen cinco diferentes tipos de restricciones: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY Y CHECK. En SQL, las restricciones UNIQUE y PRIMARY KEY se consideran restricciones únicas, y las restricciones FOREIGN KEY se consideran restricciones referenciales. El siguiente mapa sintetiza la utilización de estos tipos de restricciones.



1.2 Base de Datos Activas / Triggers– Activadores

```
CREATE TRIGGER <nombre del activador>
{ BEFORE | AFTER }
{ INSERT | DELETE | UPDATE [ OF <lista de la columna> ] }
ON <nombre de la tabla> [ REFERENCING <opciones para alias> ]
[ FOR EACH { ROW | STATEMENT } ]
[ WHEN ( <condición de búsqueda> ) ]
<instrucciones SQL activadas>
```

Las bases de datos activas son una evolución de las bases de datos pasivas(tradicionales) en donde en lugar de tener las reglas distribuidas en N programas, lo que hace es definirlas en el esquema de la Base de Datos como un conjunto de reglas con una serie de acciones que se ejecutan automáticamente cuando se producen ciertos eventos, llamamos a este tipo de reglas como **disparadores (Trigger).** Logrando así, hacer respetar reglas de integridad, generar datos derivados, controlar la seguridad o implementar reglas de negocio.

Estos triggers están basados en el modelo **evento-condición-acción (modelo ECA)**: cada regla reacciona ante un determinado evento, evalúa una condición y, si esta es cierta, ejecuta una acción.

Determinar las instrucciones SQL necesarias, para la definición de las restricciones de integridad, reglas de negocio y triggers requeridas en cada ejercicio.

1) Crear la tabla Empleados, con los siguientes atributos: DNI Empleado, Legajo, Apellido y Nombres, Cargo y Sueldo, tener en cuenta lo siguiente para cada uno de ellos:

- DNI: Clave primaria
- Legajo: cumpla con la restricción de unicidad
- Cargo: este comprendido entre los valores 50-120
- Sueldo: no sea mayor a 90000

```
Create Table Empleados (
        DNI int Primary Key,
        Legajo int UNIQUE,
          Cargo int CHECK cargo (BETWEEN 50 AND 120),
        Sueldo CHECK (sueldo ≤ 90000)
)
```

Alternativa: crear un dominio de valor para el atributo DNI, como entero de 8 dígitos, verificando que sólo acepte valores mayores a 0.

- Crear un trigger de nombre Control_sueldo, que impida que se incremente el sueldo en más de un 20%, cada vez que se modifique dicho atributo. El evento que dispara el trigger es UPDATE (sueldo), y su tiempo de acción es BEFORE.

```
Create Trigger Control_sueldo BEFORE UPDATE sueldo ON Empleados
        Begin
                If :new.cargo >= ( :old.sueldo * 1,20) then
                        :new.sueldo = :old.sueldo;
          end if
        End
```

- Crear un trigger de nombre Adicional_cargo, para el caso en que el atributo cargo tome el valor 90, entonces deberá de incrementarse el atributo sueldo de la tabla Empleados en un 15% respecto del importe básico para ese cargo, este último (importe básico) se halla almacenado en la tabla Importes_basicos, cuya clave primaria es el atributo cargo. El evento que dispara el trigger es INSERT/UPDATE del atributo cargo, y su tiempo de acción es BEFORE, de la tabla Empleados.

```
Create Trigger Adicional_cargo BEFORE UPDATE ON INSERT cargo ON
Empleados
        For each row
        Begin
                If :new.cargo ≤ 90 then
                        :new.sueldo = :new.sueldo + (Select importe_basico from
```

```
Importes_basicos WHERE Importes_basicos.Cargo = :new.Cargo) * 0.15
                    End If
                End
```

2) Crear la tabla Seguros_autos, referido a los seguros de automotores, con los siguientes atributos: Nro.Póliza, DNI Cliente, Marca del vehículo, Año modelo del vehículo, Nro. de patente, Nro. de motor, Periodo de pago e Importe a pagar (periodo trimestral), considerar:

- Nro. póliza: Clave primaria
- DNI Cliente: tiene una integridad referencial con la tabla Clientes (DNI)
- Año modelo del vehículo: tenga como valor predeterminado '2019´
- Marca: sólo pueda contener como dato uno de los siguientes valores 'Ford, Renault, Fiat, Peugeot, VW, Toyota, Nissan´
- Periodo: 202001 (Consideraremos el año 2020 y el 1er trimestre 01)
- Importe a pagar: no puede ser nulo
- Constituir clave externa con los atributos marca y año modelo, con referencia a la tabla Vehículos(marca,año_modelo)

```
Create Table Seguros_autos(
        nro. Poliza int Primary Key,
        DNI_Cliente int REFERENCE Clientes(DNI)
        Año_modelo int clientes DEFAULT '2019'
        Marca Varchar (20) CHECK marca IN
        ('Ford','Renault','Fiat','Nissan','Peugeot','Kw','Toyota'),
        periodo varchar (6) DEFAULT 202001; Importe_pagar
        Decimal (5,2) NOT NULL,
        FOREIGN KEY(Marca, Año, Modelo) REFERENCES Vehiculo (Marca,
        Año_Modelo )
)
```

- Crear un trigger de nombre Facturas_mensuales, en el cual se registrarán las cuotas que se deben de abonar en cada trimestre del año 2020, para ello se insertaran en la tabla Cuotas_Seguros los comprobantes respectivos a cada trimestre, considerado para este caso práctico las cuotas del primer trimestre del año 2020 (serían 3 nuevos registros a incorporar, para este trigger en particular).
  Los atributos de la tabla Cuotas_Seguros (tabla ya existente), se incorporarán de la siguiente manera: ○ Nro_comprobante: se compone del Nro. póliza, periodo y número de cuota (01, 02 o 03).
  ○ Monto facturado de cada cuota (corresponde a 1/3 del importe a pagar contenido en la tabla Seguros_autos). ○ Fecha de pago: corresponde al día 15 de cada mes (15/01/20, 15/02/20, 15/03/20), consideraremos estas tres fechas para representar este trigger en particular, es decir los meses del primer trimestre del año 2020.

El evento que dispara el trigger es INSERT de la tabla Seguros_autos y su tiempo de acción es AFTER.

```
Create Trigger Facturas_mensuales AFTER INSERT ON Seguros_Autos
        @declare nro_comprobante varchar (20),
        @declare monto decimal (5,2)
        @declare fecha_pago date,
        @declare nro_cuota int,

        For each row
        Begin
                monto = :new.importe_Pagar / 3
        nro_cuota = 1

                while nro_cuota <=3

                  nro_comprobante = concat (new_nroPoliza.substring, ( :new.periodo, 4, 2)
                                    convert (string.nro_cuota)),


                fecha_pago = DateFromPart (convert (int, substring(newperiodo,1,4 ) ),
                                (3*((convert (int,substring (new,periodo,4,2) ) -1) +
                                nro_cuota), 15),

                INSERT INTO Cuotas_Seguros VALUES (nro_comprobante, monto,
                fecha_Pago),
                nro_cuota = nro_cuota+1,
        End

        End
End
```

3) Crear la tabla Pedidos, con los siguientes atributos: Nro. de orden de pedido, fecha de la orden de pedido, CUIT del cliente, CUIT del proveedor, nro. de producto, tipo de pedido y cantidad pedida del producto, considerar:

- CUIT del cliente y Nro. de orden de pedido: Clave primaria
- Nro. de orden pedido: cumpla con la restricción de unicidad
- Fecha de la orden de pedido: valor por defecto la fecha actual del sistema
- CUIT del cliente: integridad referencial con la tabla Clientes(Cuit)
- Tipo de pedido: puede contener únicamente M, C, G, H, N
- Cantidad pedida: valores mayores a 0

```
Create Table Pedidos (
        nroProducto int REFERENCES Stock_productos (nro_Producto),
        nroOrden int UNIQUE,        fecha date getdate (),        cuitPro int,      cuitCli
int,    tipo_pedido vachar (1) CHECK (tipo_pedido IN (('M','C','G','H','N')),
          cantidad int CHECK (cantidad > 0),
          PRIMARY KEY (nroOrden, cuit),
)
```

- Crear un trigger de nombre Control_pedido, que impida dar de alta una orden de pedido, cuando la cantidad pedida del producto sea superior al atributo stock_actual de la tabla Stock_productos cuya clave primaria es el atributo número de producto, caso contrario actualizar el stock_actual, que resulta luego de restarle la cantidad del actual pedido. El evento que dispara el trigger es INSERT, y su tiempo de acción es BEFORE, respecto de la tabla Pedidos.

```
Create Trigger Control_pedido BEFORE INSERT ON Pedidos
        For each row
        Begin
                IF :new.cantidad > (Select stock_actual FROM stock_productos
                                        WHERE nroProducto = :new.nroProducto)
                UPDATE stock_productos SET stock_actual = stock_actual -
                :new_cantidad WHERE nroProducto = :new.nroProducto)
                ELSE  raiseerror ('no hay stock suficiente'),
                        rollback,
                END IF
        End End
```

4) Respecto a las tablas Pedidos y Stock_productos del ejercicio 3), agregue las siguientes restricciones:

Para la tabla Pedidos:

- "ClaveProducto" de tal manera que nro. de producto y tipo de pedido tenga una referencia externa a la tabla Productos(nro_producto,tipo_pedido)
- "CantidadPermitida" donde el atributo Cantidad pedida, sólo acepte valores comprendidos entre 10 y 2000.
- "ControlCUIT" donde el atributo CUIT del cliente no deberá de coincidir con el CUIT del proveedor, previendo deshabilitar esta comprobación en los datos ya existentes.

Tabla Pedidos:

```
ALTER TABLE Pedidos ADD CONSTRAINT claveProducto FOREIGN
KEY(nro_Producto,tipopedido) REFERENCES
```

```
Productos(nro_producto,tipo_pedido)

ALTER TABLE Pedidos ADD CONSTRAINT CantidadPermitida
CHECK (cantidad BETWEEN 10 AND 2000)

ALTER TABLE Pedidos ADD CONSTRAINT ControlCuit
CHECK (cuitPro <> cuitCli)
```

Para la tabla Stock_productos:

- "ControlStock" a fin de verificar que el stock actual de un determinado producto no sea inferior al stock mínimo del mismo, siendo los respectivos atributos stock_minimo y stock_actual.

```
ALTER TABLE Stock_producto ADD CONSTRAINT ControlStock CHECK
(stock_actual ≥ stock_minimo)
```

---

5) Crear la tabla Clientes, con los atributos: código de cliente, nombre, dirección, código postal e importe base.

Crear la tabla Facturas, con los atributos: Nro. de factura, fecha de la factura, código de cliente de la factura, tipo de descuento, valor de IVA e importe de la factura.

Contemplar:

- Crear un valor de dominio de nombre CodigoCliente para el atributo código de cliente, tipo entero de 9 dígitos, que sea mayor a 0

```
Create Domain Codigocliente check(CodigoCliente >0 and
CodigoCliente<10000000000);
```

Tabla Clientes:

- Código cliente: referenciar al valor dominio creado anteriormente, siendo clave primaria
- Nombre y dirección: no nulos
- Código postal: integridad referencial con la tabla CodPostales(cod_postal)
- Importe base no nulo

```
Create table Clientes ( codigo_cliente CodigoCliente Primary Key , nombre varchar (30) not null,
dirección varchar (50) not null,
```

```
código_postal int REFERENCES CodPostales (cod_postal),
importe_base decimal (3,2) not null,
)
```

Tabla Facturas:

- Nro. Factura: clave primaria
- Código de cliente de la factura: referenciar al valor dominio creado e integridad referencial a la tabla Clientes(cliente_factura)
- Tipo de descuento: sólo puede contener valores comprendidos entre 5 y 20 respectivamente (estos valores correspondes a porcentajes %)
- IVA: sólo puede contener alguno de estos valores 15 o 21 (corresponden a porcentajes %).
- Importe de la factura, no nulo.

```
Create Table Facturas (
        nroFactura int Primary Key,
        fecha date,
        código_cliente CodigoCliente REFERENCES Clientes(cod_cliente),
        tipo_desc Decimal (2,2) CHECK (tipo_desc BETWEEN 5 AND 20)
        iva int CHECK (iva IN (15,21),  importe Decimal (8,2) NOT
NULL,
)
```

- Crear un trigger de nombre Calcular_importe_factura, que permita actualizar el importe de la factura que debe de abonar cada cliente de la tabla Facturas, el cual se obtiene sobre el importe base de la tabla Clientes, descontando el porcentaje contenido en el atributo tipo de descuento, y sobre este resultado aplicar el porcentaje del IVA (15 o 21%), para obtener el valor definitivo correspondiente al importe de la factura, ambas tablas se relacionan en base al atributo código de cliente.

  El evento que dispara al trigger es INSERT de una tupla o UPDATE respecto al atributo importe base de la tabla Clientes y su tiempo de acción es AFTER.

```
Create Trigger Calcular_importe_factura AFTER INSERT OR UPDATE
                                        importe_base ON Clientes
        For each row
        Begin
            UPDATE Facturas SET importe = (:new.importe_base * 1-
    tipo_desc / 100 ) * (1+ IVA/100) ) WHERE cod_cliente = :new
                cod_cliente
        End
End
```

6) Crear la tabla Empleados_baja, con los atributos DNI Empleado, Legajo, Cargo, Usuario y Fecha.

```
Create Table Empleados_baja (
        DNI int PRIMARY KEY,
        legajo int NOT NULL UNIQUE,
        cargo varchar (20),
 usuario varchar (20),  fecha date,
)
```

Luego crear el trigger de nombre Empleado_eliminado, que permita insertar en la tabla Empleados_baja, los datos de aquel empleado que se elimina de la tabla Empleados (del ejercicio 1), donde las columnas usuario y fecha se grabarán con las variables del sistema USER y SYSDATE o DATE, asociados al identificador del usuario que opera el sistema en ese momento y la fecha actual en que se lleva a cabo este proceso de baja de un empleado.

```
Create Trigger Empleado_eliminado AFTER DELETE ON Empleados
        For each row
    INSERT INTO Empleados_baja VALUES (old.DNI, old.legajo, old.cargo,
    user, SYSDATE)
        End
End
```

Crear otro trigger de nombre Baja_usuario, que permita eliminar de la tabla Usuarios, aquella tupla cuyo DNI de esta última tabla, sea igual al DNI del empleado eliminado en el paso anterior.
El evento que dispara ambos triggers es DELETE, y su tiempo de acción es AFTER respecto a la tabla Empleados.

```
Create Trigger Baja_Usuario AFTER DELETE ON Empleados
        For Each now
                DELETE FROM Usuarios WHERE DNI  = :old.DNI
        End
End
```

7) Modificar el trigger Baja_usuario creado en el ejercicio anterior, de tal manera que, para eliminar de la tabla de Usuarios, en lugar de hacerlo con el atributo DNI, demos la baja del empleado mediante su número de Legajo, utilizar el comando Replace trigger para registrar este cambio en el disparador.

Indicar los comandos Sql necesarios para desactivar el trigger Calcular_importe y también todos los triggers que estén asociados a la tabla Consumos.

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) Respecto de la tabla Facturas, creada en el ejercicio 5), incorporar a la tabla las columnas de Fecha_vto y Fecha_pago de tipo fecha (date) y la columna intereses de tipo entero.

```
ALTER TABLE Facturas ADD Fecha_Vto date,

ALTER TABLE Facturas ADD Fecha_pago date,

ALTER TABLE Facturas ADD intereses int,
```

Crear un trigger de nombre Recargo_factura, que detecte aquellos pagos fuera de término, para lo cual, si la fecha de pago es superior a la fecha de vencimiento, asignar el valor 5 al atributo intereses previamente incorporado, este recargo es él porcentaje que se vería reflejado en el cobro de la próxima factura del cliente, esta actualización sobre la tabla Facturas la accedemos teniendo en cuenta el código de cliente de las tablas del punto 5).

El evento que dispara al trigger es UPDATE respecto al atributo fecha de pago de la tabla Facturas y su tiempo de acción es AFTER.

```
Create Trigger recargos_factura AFTER UPDATE Fecha_pago ON Facturas
        Begin
        IF (:new.Fecha_pago) > :new.Fecha_Vto)
                UPDATE Facturas SET intereses = 5 WHERE nro_factura =
                :new.nro_factura
        End
End
```

---

9) Crear un nuevo trigger de nombre Importe_recargo respecto de la tabla Facturas, que refleje el recargo (porcentaje) que corresponde pagar a aquellos clientes que abonaron la factura fuera de término, este se obtiene de aplicar el valor del atributo intereses respecto del importe de la factura, por lo que el nuevo monto de la factura se verá

incrementado con el importe resultante del recargo aplicado (Importe de la factura= Importe de la factura + (Importe de la factura con recargo).

El evento que dispara al trigger es UPDATE respecto al atributo intereses de la tabla Facturas y su tiempo de acción es AFTER.

```
Create Trigger Importe_recargo AFTER UPDATE intereses ON Facturas
        Begin
                UPDATE Facturas SET importe = importe * (1+ intereses/100)
                WHERE nroFactura = new.nroFactura
        End
End
```

Serie de Ejercicios Prácticos Nº2

Bases de Datos Distribuidas Algoritmos de optimización de consultas

---

1. Bases de Datos Distribuidas

Las bases de Datos distribuidas surgen a causa de que el entorno se vuelve cada vez más globalizado en donde nos encontramos por ejemplo empresas distribuidas geográficamente, que no se encuentran en el mismo lugar físico, pero necesitamos tener una visión global de toda la información de manera consolidada.

Es decir, acceder como si fuera un todo teniendo una distribución física *(Integración + Distribución)*

2. Procesamiento y optimización de consultas distribuidas

Aparecen las *consultas globales* donde se tiene que elegir qué sede es la que solicita la subconsulta y recoger los datos devueltos por todas las fuentes.

Por lo tanto, debemos tener alguna forma de poder optimizar estas consultas ya que pueden producir una carga importante en la red y esto puede impactar en los tiempos de respuestas, ancho de banda etc. Se busca minimizar el tráfico con el objetivo de reducir la cantidad de transferencia de datos por la red aplicando una *estrategia local de optimización.*

2.1 Algoritmo de Optimización de Consulta

1) <u>Tamaño de la relación</u>

Calculamos el tamaño de la relación de la siguiente forma

Tamaño de la relación = Cantidad de tuplas * Longitud del registro

2) <u>Resultado de la consulta</u>

Debemos calcular cuánto nos costaría transferir el resultado de una consulta determinada, para esto tenemos en cuenta la longitud de las columnas necesarios para realizar la consulta y multiplicar por la cantidad de tuplas

Resultado de la Consulta =
(Sumatoria de las longitudes de columnas necesarios) * cantidad de tuplas

3) <u>Alternativas para la realización de la consulta distribuida</u>

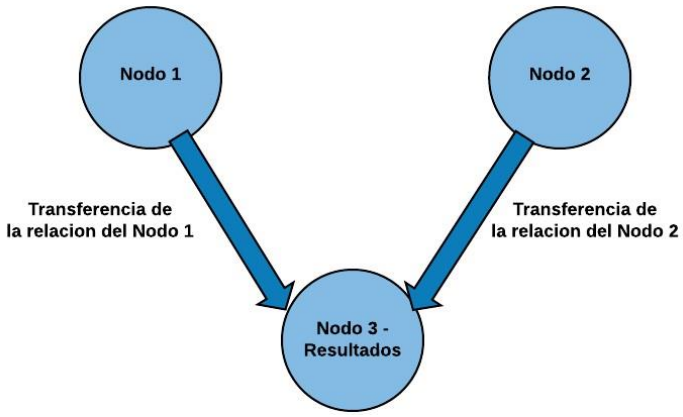
Dada una consulta distribuida determinada, evaluamos las diferentes alternativas de transferencia que se pueden realizar para así determinar cuál de los costes de transferencias total es más favorable para reducir el tráfico en la red.

Durante la resolución de todo el practico se presentó las siguientes alternativas

Alternativa 1: Nodo 3 como resultado

En este caso, denominados a un Nodo 3 como "Resultado" dado que es el lugar en donde precisan los resultados de la consulta y no tiene datos implicados en la consulta. Podemos transferir el resultado de la consulta distribuida mediante tres formas dependiendo de que nodo utilizamos para la ejecución de la consulta, cada una con un coste de transferencia distinta:

Ejecución de la Consulta en el Nodo 3



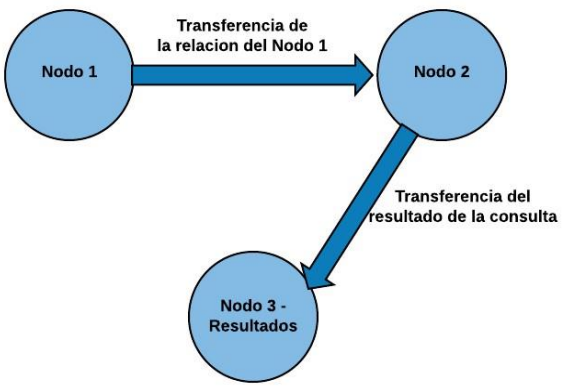
En este caso, transferimos la relación correspondiente de cada nodo, calculada previamente en el paso 1. La consulta se realiza en el Nodo 3 y el resultado se obtiene allí que es donde lo precisa, por lo tanto, no es necesario realizar ninguna transferencia más. El coste de transferencia total (CTT) es el siguiente

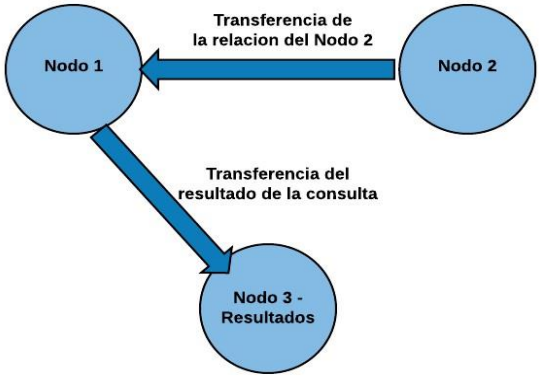
CTT = Transferencia de la relación del Nodo 1 + Transferencia de la relación del Nodo 2

Ejecución de la Consulta fuera del Nodo Resultado

Ya sea que hagamos la consulta en el Nodo 1 o Nodo 2, el procedimiento es similar, primero debemos transferir la relación del Nodo i donde puede ser el Nodo 1 o Nodo 2, al Nodo donde ejecutaremos la consulta. Transferimos el resultado obtenido al Nodo 3, lo cual el costo lo calculamos en el punto 2.

CTT = Transferencia de la Relación del Nodo i + Transferencia del Resultado de la consulta





Alternativa 2: Nodo 2 como Nodo resultado

En este caso el resultado de la consulta es requerida por el Nodo 2, por lo que tenemos dos caminos para realizar la transferencia del mismo

Ejecución de la Consulta en el Nodo 2

Este caso es sencillo, ya que solo se realiza la transferencia de la relación del Nodo 1, la consulta se realiza en el Nodo 2 por lo tanto los resultados quedan allí y no es necesario ninguna transferencia más.

CTT = Transferencia de la relación del Nodo 1

Ejecución de la Consulta en el Nodo 1

Otra opción es ejecutar la consulta en el Nodo 1 y luego transferir el resultado al Nodo 2.

CTT = Transferencia de la relación del Nodo 2 + Transferencia del resultado de la consulta

Alternativa 3: Consulta distribuida usando Semi-Join (Semi-Reunión)

Este tipo de consulta distribuida haciendo uso del Semi-Join, es una muy buena alternativa cuando debemos trabajar con un rango determinado, ya que permite una reducción en el número de tuplas antes de ser transferidas a otro nodo. Para lograr este tipo de consulta debemos hacer lo siguiente:

Caso 1: proyección de la columna de concatenación

Buscamos la columna que sirve de concatenación entre ambas relaciones de los nodos y la transferimos al nodo donde realizaremos la consulta

*Costo de transferencia: columna de concatenación(longitud) * tuplas registros*

Caso 2: proyección de las columnas necesarias

Vamos a obtener un conjunto que va a estar compuesto por todas las filas que se encuentran dentro del rango establecido para esa columna de concatenación. Transferimos solo los atributos necesarios al nodo inicial para realizar la consulta

*Costo de transferencia= columna de concatenación + (columnas necesarias para la consulta) * cantidad de tuplas*

1) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los clientes de una editorial para sus respectivas sucursales:

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

Se solicita:

a) Determinar el tamaño de cada relación.

Tamaño de la relación Nodo1: Clientes
T.C = 5000 tuplas * 123 bytes = **61,500 bytes**

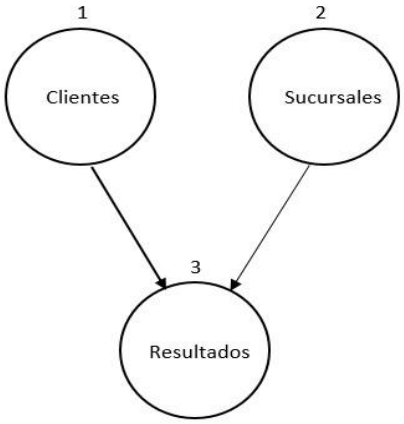
Tamaño de la relación Nodo2: Sucursales
T.S = 100 tuplas * 71 bytes = **7,100 bytes**

b) Se desea obtener la siguiente consulta:

Para cada cliente, obtener su Apellido y Nombres y el Nombre de la Sucursal en la cual adquiere los libros.

1) Resultado de la consulta:
Apellidos Y Nombres * Tuplas Cliente
(30 bytes + 30 bytes) * 5000 tuplas = 60 * 5000 tuplas = **300000 bytes**

I. ALTERNATIVA 1: NODO 3 COMO RESULTADO

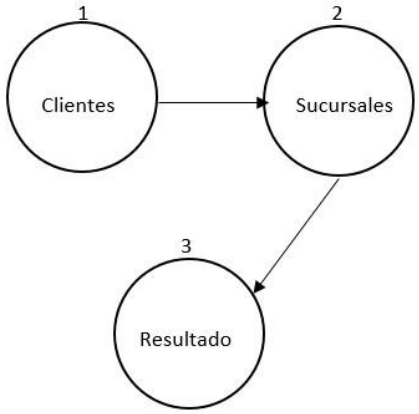


Opción 1: Ejecución de la consulta en Nodo 3

C.T.T = Relación Nodo Cliente + Relación Nodo Sucursales

C.T.T = 615000 bytes + 7100 bytes = 622,100 bytes transferidos

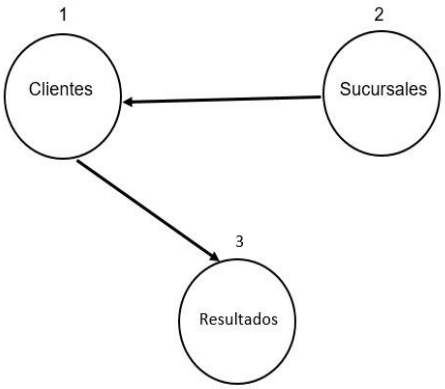
Opción 2: Ejecución de la consulta en Nodo 2



C.T.T = Relación Nodo Cliente + Resultado de la consulta

C.T.T = 615000 bytes + 300000 bytes = 915000 bytes transferidos

Opción 3: Ejecución de la consulta en Nodo 1

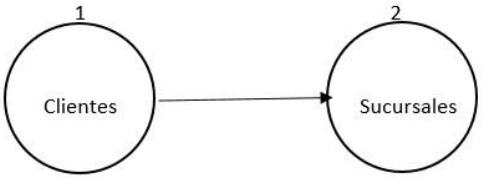


C.T.T =Relación Nodo Sucursales + Resultado de la consulta

C.T.T = 7100 bytes + 300000 bytes = 307100 bytes transferidos

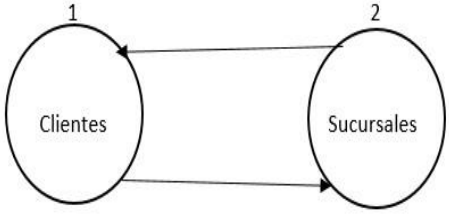
Respuesta: Para minimizar el transporte de bytes entre los distintos nodos, la mejor opción es el caso 3.

ALTERNATIVA 2: NODO 2 COMO RESULTADO



Opción 1: Ejecución de la consulta en el Nodo 2

C.T.T = Relación Nodo 1 Clientes
C.T.T = 615000 bytes transferidos



Opción 2: Ejecución de la consulta en el Nodo 1

C.T.T = Relación Nodo 2 Sucursales + Resultado Consulta

C.T.T = 7100 bytes + 300000 bytes = 307100 bytes transferidos

Respuesta: Para los dos casos planteados, la opción 2 es la estrategia más adecuada.

ALTERNATIVA 3: CONSULTA DISTRIBUIDA USANDO SEMI JOIN RESPECTO AL ITEM II

Paso 1: proyección de la columna de concatenación

Transferimos el atributo de concatenación entre ambas relaciones. En este caso desde el Nodo 2 al Nodo 1

Nro. Sucursal * Cantidad tuplas Sucursal
(4 bytes) * 100 tuplas = 400 bytes transferidos

Paso 2: proyección de las columnas necesarias

Luego, devolvemos al Nodo 2 el atributo de concatenación más los atributos necesarios para realizar la consulta, en este caso pedía los Apellidos y Nombres, por lo tanto:

(Nro. Sucursal + Apellidos Y Nombres) * Cantidad tuplas Cliente
(4 bytes + 30 bytes) * 5000 tuplas = **170000 bytes transferidos**

Como se realizaron dos transferencias, sumamos para saber el costo total

C.T.T = 400 bytes + 170000 bytes
170400 bytes transferidos

---

2) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los profesores y facultades de nuestra universidad:

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

Se solicita:

a) Determinar el tamaño de cada relación.

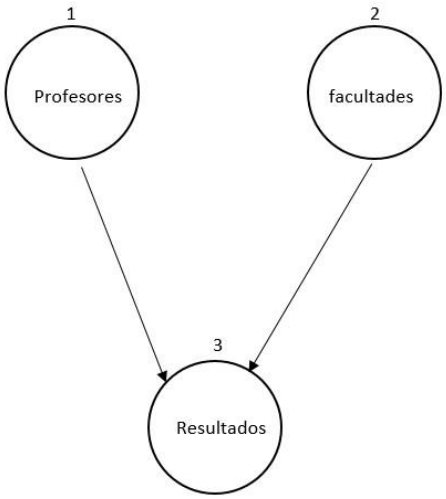
Nodo 1 Profesores
7000 tuplas * 88 bytes = 616000 bytes
Nodo 2 Facultades
200 tuplas * 77 bytes = 15400 bytes

b) Se desea obtener la siguiente consulta:

Por cada profesor, obtener su Apellido y Nombres y la Denominación de la facultad en la cual presta sus servicios.

1) Resultado de la consulta:
(Apellido Y Nombres del profesor + Denominación de la facultad) * tuplas relación Profesores
(30 bytes + 32 bytes) * 7000 tuplas = 434000 bytes

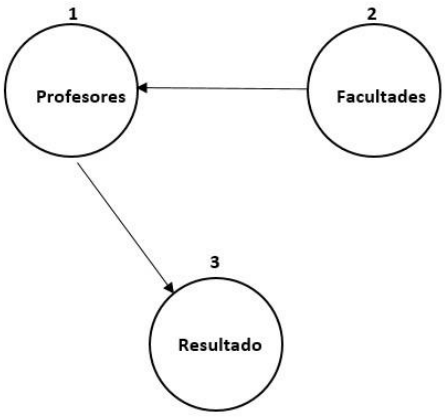
ALTERNATIVA 1: NODO 3 COMO RESULTADO



<u>Opción 1: Ejecución de la consulta en Nodo 3</u>

Costo Transferencia Total = Relación Nodo Profesores + Relación Nodo Facultades

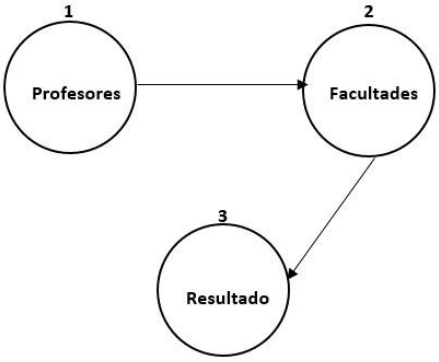
= 616000 bytes + 15400 bytes = 631400 bytes transferidos



Opción 2: Ejecución de la consulta en Nodo 2

C.T.T = Relación Nodo Profesores + Resultado de la consulta

= 616000 bytes + 434000 bytes = 1050000 bytes transferidos



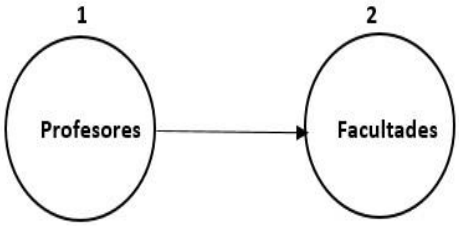
Opción 3: Ejecución de la consulta en Nodo 1

C.T.T = Relación Nodo Facultades + Resultado de la consulta

= 15400 bytes + 434000 bytes = 449400 bytes transferidos

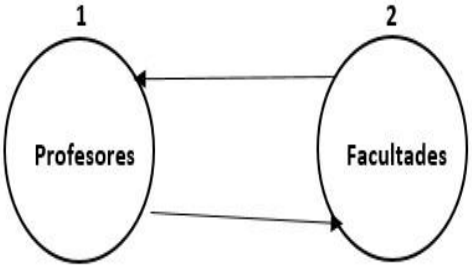
Respuesta: La opción 3 es la más adecuada ya que tiene un costo menos de transferencia de datos por la red. Siendo este nuestro criterio de optimización

ALTERNATIVA 2: NODO 2 COMO RESULTADO



Opción 1: Ejecución de la consulta en el Nodo 2

C.T.T = Relación Nodo 1 Profesores = 615000 bytes transferidos



Opción 2: Ejecución de la consulta en el Nodo 1

C.T.T = Relación Nodo 2 Facultades + Resultado Consulta

= 15400 bytes + 434000 bytes = 449400 bytes transferidos

Respuesta: De las opciones consideradas, la opción 2 es la más optima

ALTERNATIVA 3: CONSULTA DISTRIBUIDA USANDO SEMI JOIN RESPECTO AL ITEM II

Paso 1: proyección de la columna de concatenación

Transferimos el atributo de concatenación entre ambas relaciones. En este caso desde el Nodo 2 al Nodo 1

Código Facultad * Cantidad tuplas Facultades
(2 byte) * 200 tuplas = 400 bytes transferidos

Paso 2: proyección de las columnas necesarias

Luego, devolvemos al Nodo 2 el atributo de concatenación más los atributos necesarios para realizar la consulta, en este caso pedía los Apellidos y Nombres, por lo tanto:

(Codigo Facultad + NomyApe) * Tuplas Profesores
(2 bytes + 30 bytes)* 7000 tuplas = 224000 bytes transferidos

Paso 3: Costo total de transferencia

400 bytes + 224400 bytes = 224400 bytes transferidos

---

3) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los remedios y sus correspondientes laboratorios:

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

Se solicita:

a) Determinar el tamaño de cada relación.

Tamaño de la relación Nodo1: Remedios
100000 tuplas * 81 bytes = **8100000 bytes**

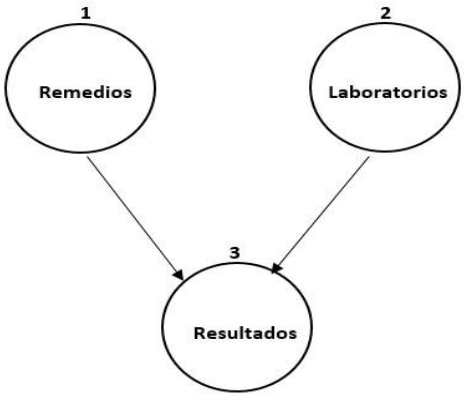
Tamaño de la relación Nodo2: Laboratorios
638000 tuplas * 177 bytes = **11292600 bytes**

b) Se desea obtener la siguiente consulta:

Por cada remedio, obtener su Nombre Comercial y Nombre del laboratorio que lo produce.

Resultado de consulta:
(Nombre Comercial de Remedio + Nombre del laboratorio) * tuplas de Remedios (20 bytes + 40 bytes) * 100000 tuplas = 6000000 bytes

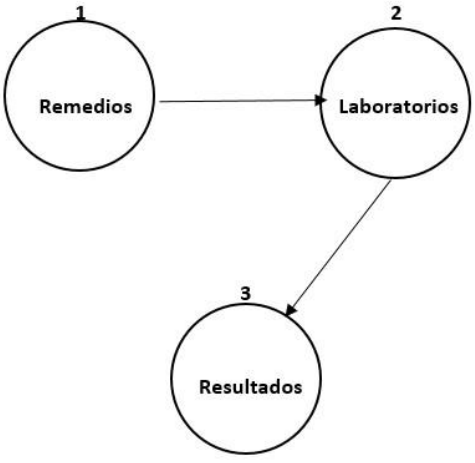
ALTERNATIVA 1: NODO 3 COMO RESULTADO



Opción 1: Ejecución de la consulta en Nodo 3

Costo Transferencia Total = Relación Nodo Remedios + Relación Nodo Laboratorios

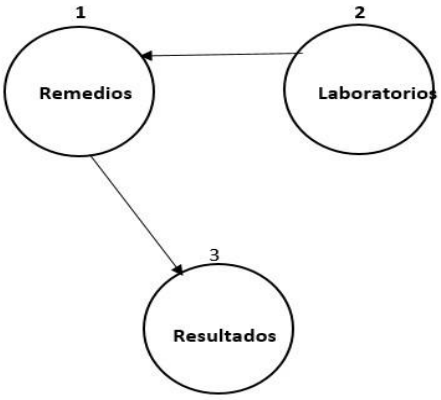
= 8100000 bytes + 11292600 bytes = 19392600 bytes transferidos



Opción 2: Ejecución de la consulta en Nodo 2

Costo Transferencia Total = Relación Nodo Remedios + Resultado de la consulta

= 8100000 bytes + 6000000 bytes = 14100000 bytes transferidos



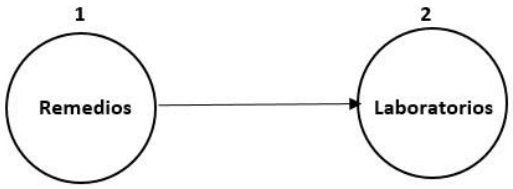
Opción 3: Ejecución de la consulta en Nodo 1

Costo Transferencia Total = Relación Nodo Laboratorios + Resultado de la consulta

=11292600 bytes + 6000000 bytes = 17292600 bytes transferidos

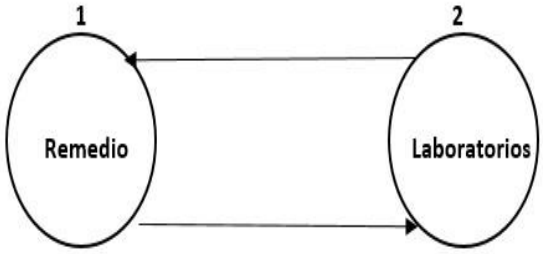
Respuesta: La opción 2 es la más adecuada ya que tiene un costo menos de transferencia de datos por la red. Siendo este nuestro criterio de optimización.

ALTERNATIVA 2: NODO 2 COMO RESULTADO



Opción 1: Ejecución de la consulta en el Nodo 2

Costo Transferencia Total = Relación Nodo 1 + Remedios = 8100000 bytes transferidos



Opción 2: Ejecución de la consulta en el Nodo 1

Costo Transferencia Total= Relación Nodo 2 Laboratorios + Resultado Consulta

= 11292600 bytes + 6000000 bytes = 17292600 bytes transferidos

Respuesta: De las opciones consideradas, la opción 1 es la más optima.

4) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los empleados de una empresa y los respectivos departamentos laborales.

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

Se solicita:

a) Determinar el tamaño de cada relación.

Tamaño de la relación Nodo1: Empleados

3200 tuplas * 56 bytes = **179200 bytes**

Tamaño de la relación Nodo2: Departamentos

1500 tuplas * 130 bytes = **195000 bytes**

b) Se desea obtener la siguiente consulta:

Por cada empleado, obtener su Nombre y Apellido y Nombre del departamento al cual pertenece dentro de una empresa multinacional.

Resultado de la consulta: (25 bytes + 40 bytes) * 3200 =
208000 bytes transferidos

ALTERNATIVA 1: NODO 3 COMO RESULTADO

Opción 1: Ejecución de la consulta en Nodo 3

C.T.T = Relación Nodo Empleados + Relación Nodo Departamentos

= 179200 bytes + 195000 bytes = 374200 bytes transferidos

Opción 2: Ejecución de la consulta en Nodo 2

C.T.T = Relación Nodo Empleados + Resultado de la consulta

= 179200 bytes + 208000 bytes = 387200 bytes transferidos

Opción 3: Ejecución de la consulta en Nodo 1

C.T.T = Relación Nodo Departamentos + Resultado de la consulta

= 195000 bytes + 208000 bytes = 403000 bytes transferidos

Respuesta: La opción 1 es la más adecuada ya que tiene un costo menos de transferencia. Siendo este nuestro criterio de optimización

ALTERNATIVA 2: NODO 2 COMO RESULTADO

Opción 1: Ejecución de la consulta en el Nodo 2

C.T.T = Relación Nodo 1 Empleados =
179200 bytes transferidos

Opción 2: Ejecución de la consulta en el Nodo 1

C.T.T = 195000 bytes + 208000 bytes = 403000 bytes

Respuesta: La opción 1 es la más adecuada ya que tiene un costo menos de transferencia. Siendo este nuestro criterio de optimización

ALTERNATIVA 3: CONSULTA DISTRIBUIDA USANDO SEMI JOIN RESPECTO AL ITEM II

Opción 1: proyección de la columna de concatenación
Código de departamento (longitud) = 3 bytes * 1500 tuplas = 4500 bytes transferidos

Opción 2: proyección de las columnas necesarias

Código de departamento + Nombre y Apellido del empleado.
(3 bytes + 25 bytes) * 3200 tuplas = 89600 bytes transferidos

C.T.T = 4500 bytes + 89600 bytes = 94100 bytes transferidos

Serie de Ejercicios Prácticos Nº3
Bases de Datos Objetos

---

1. Introducción:

Las bases de datos relacionales eran las más utilizadas, sin embargo, existen casos de uso en lo que se requiere gestionar datos muy complejos o no convencionales tales como imágenes, videos, documentos para los que las estructuras relacionales resultan muy complejas e ineficientes.

El modelo de Bases de Datos Orientados a Objetos intenta brindar una solución a esta problemática con la representación de estos **objetos complejos.** Compuestos por los siguientes elementos:

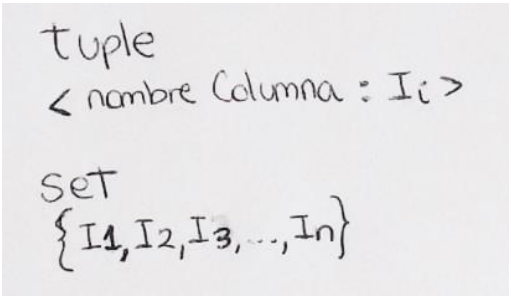
- **OID (OBJECT IDENTIFIER)**

  Representa la identidad de cada objeto, el cual es único para este. Es asignado por el sistema en el momento en que el objeto es creado y con el que se puede hacer referencias a este objeto.

- **CONSTRUCTORES**

  Son los encargados de definir los tipos de datos y los tipos de objetos necesarios para construir los objetos de una aplicación determinada, con las palabras reservadas *tuple, set y list.*

  tuple: permiten definir datos que son tuplas. Convenientes para agrupar varios valores y hacer como si fueran un único valor. Tienen varios componentes llamados atributos con un nombre y un tipo. set: representa un conjunto de cadenas, es una lista enumerada de valores permitidos que se especifican al definir el campo. Con la particularidad que son un grupo desordenado de objetos del mismo tipo. atom (tipos atómicos), elementos de los dominios básicos como enteros, reales, cadenas, etc.



A efectos prácticos definimos estos constructores como se observa en la imagen.

Algunas aclaraciones:

atom: simplemente se utiliza la palabra reservada

tuple: nombreColumna corresponde al nombre del campo de la tabla que referencia el OID(I) señalado alado.

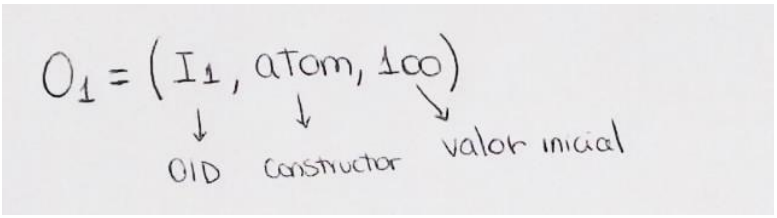
set: a diferencia de tuple, para el definir los sets solamente tenemos en cuenta los Ii, es decir los OID de cada objeto que queremos incluir en el conjunto.

Tanto i como n pertenecientes al dominio de los Naturales {0,1, 2, …, n}

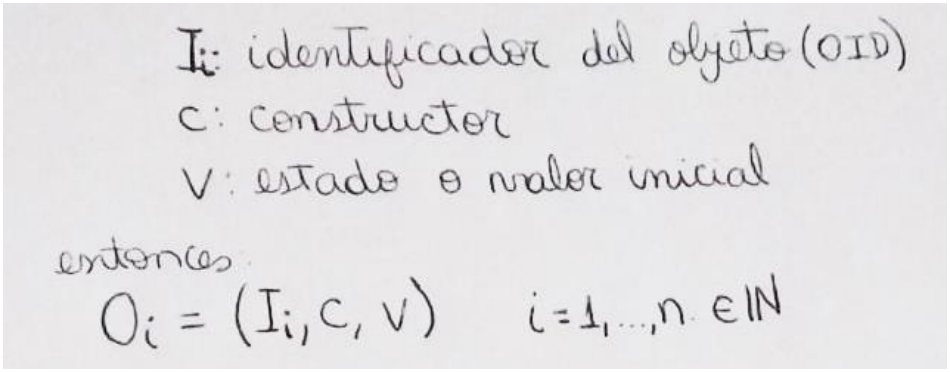
- **ESTADO**

  Es el valor que tiene asignado dicho objeto en un momento determinado. Estos se construyen a partir de los átomos (valores reales, enteros, cadenas, etc.)

Por ejemplo, un objeto con estos tres componentes nos quedaría de la siguiente manera:



En conclusión, un objeto complejo lo podemos definir de la siguiente forma:



1) Sean los datos de una empresa de telefonía celular:

Valores atómicos

1) Nro. Empresa: 100
2) Nombre Empresa: Telecom
3) Sucursal1: Posadas
4) Sucursal2: Salta
5) Sucursal3: Formosa
6) Fecha creación: 01-02-1994
7) DNI: 24987422
8) Sueldo: 35000

Conjuntos

9) Sucursales= {Sucursal1, Sucursal2, Sucursal3}

Tuplas

10) Empresa (objeto complejo)

Nro. Empresa	Nombre Empresa	Sucursales(9)	Fecha Creación	Presidente(11)
		Tipo set		Tipo Tuple

11) Presidente

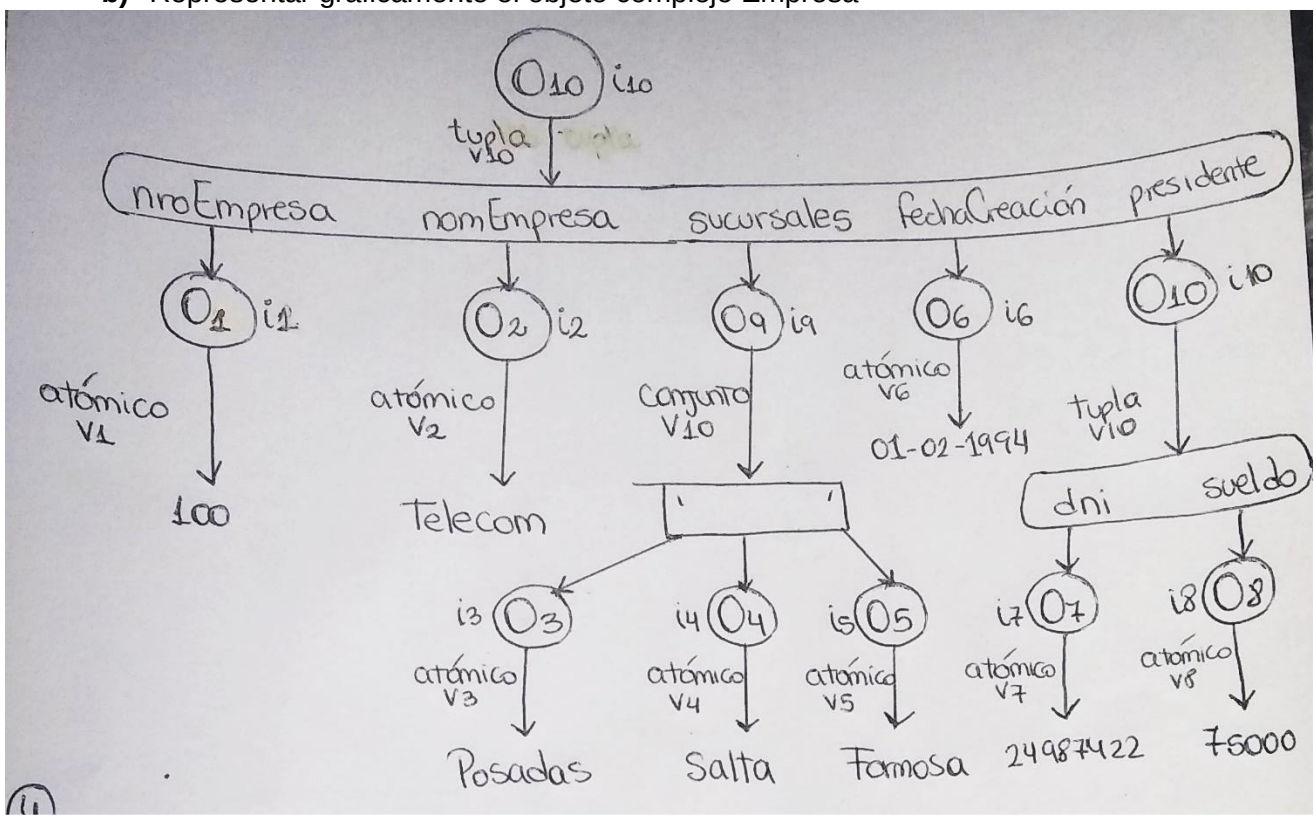
Dni	Sueldo

Resolver:

a) Definir los objetos, teniendo en cuenta los valores y tipos dados

o_1= (i_1, atom, 100) **o_2**= (i_2, atom, 'Telecom') **o_3**= (i_3, atom, 'Posadas') **o_4**= (i_4, atom, 'Salta') **o_5**= (i_5, atom, 'Formosa') **o_6**= (i_6, atom, '01-02-1994') **o_7**= (i_7, atom, 24987422)
o_8= (i_8, atom, 75000)
o_9= (i_9, set,{i_3, i_4, i_5})
o_{10}= (i_{10}, tuple, <nroEmpresa:i_1, nomEmpresa:i_2, sucursales:i_9, fechaCreacion:i_6, presidente:i_{11}>) **o_{11}**= (i_{11}, tuple, <dni:i_7, sueldo:i_8>)

b) Representar gráficamente el objeto complejo Empresa



2) Sean los datos de una facultad:

Valores atómicos

1) Código Facultad: 11
2) Nombre Facultad: Medicina
3) Sede1: Centro
4) Sede2: Campus Cabral
5) Posgrado1: Higiene
6) Posgrado2: Enfermería
7) Posgrado3: Salud Mental
8) Dirección: Moreno 1240
9) Legajo: M1378
10) Nombre: Carlos
11) Apellido: Monti
12) CUIL: 20-18980067-4
13) Mail: cm@med.unne.edu.ar

Conjuntos

14) Sedes = {Sede1, Sede2}
15) Posgrados = {Posgrado1, Posgrado2, Posgrado3}

Tuplas

16) Facultad (objeto complejo)

Código Facultad	Nombre Facultad	Sedes (14)	Posgrados (15)	Dirección	Decano (17)
		Tipo set	Tipo set		Tipo tuple

17) Decanos

Id (18)	Legajo	CUIL
Tipo tuple		

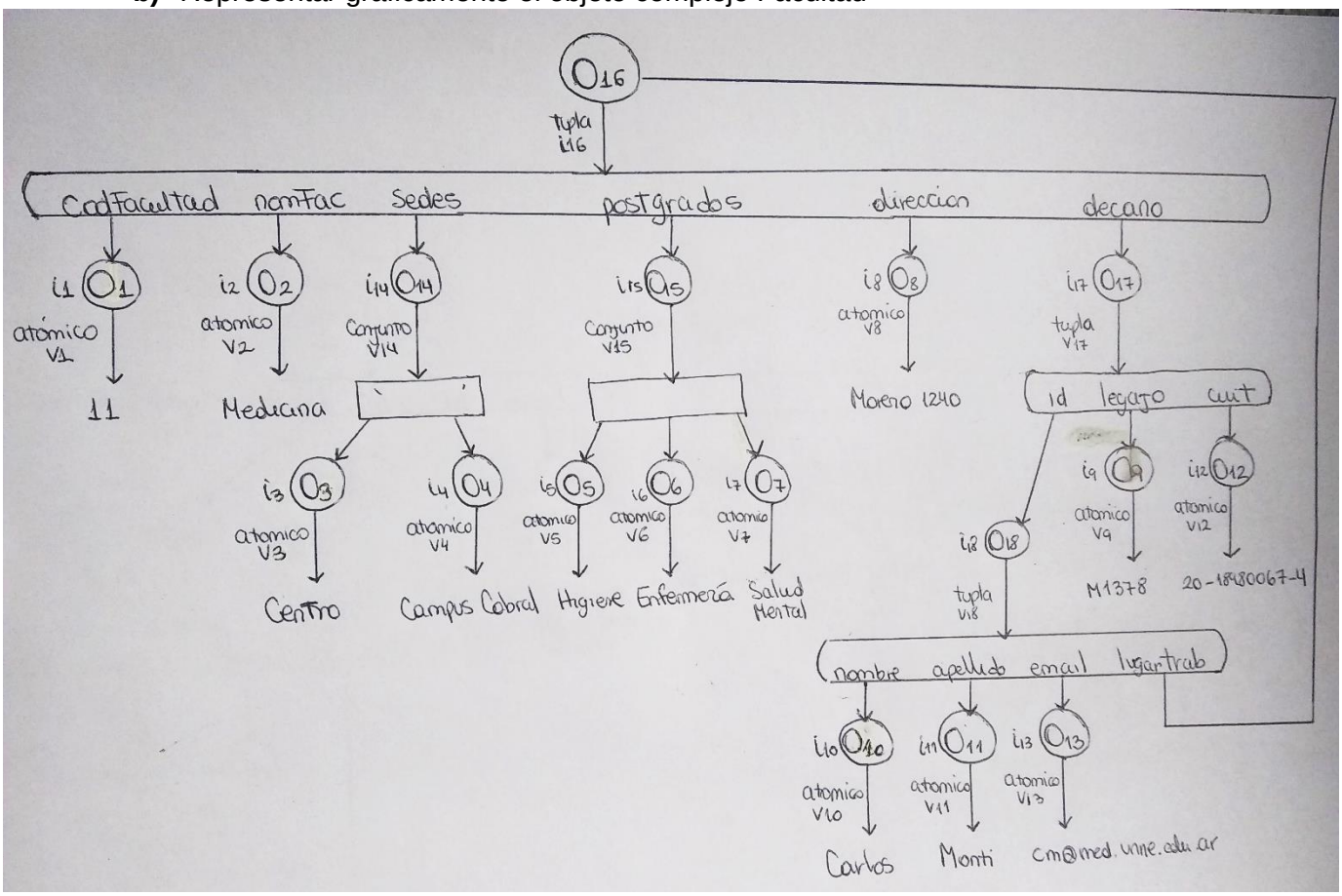
18) Empleados

Nombre	Apellido	email	Lugar trabajo (16)
			Tipo tuple

a) Definir los objetos, teniendo en cuenta los valores y tipos dados

o_1= (i_1, atom, 11) **o_2**= (i_2, atom, 'Medicina') **o_3**= (i_3, atom, 'Centro') **o_4**= (i_4, atom, 'Campus Cabral')
o_5= (i_5, atom, 'Higiene') **o_6**= (i_6, atom, 'Enfermería') **o_7**= (i_7, atom, 'Salud Social') **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, nomFac:i_2, sedes:i_{14}, posgrados: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}, lugartrab:i_{16}>)

b) Representar gráficamente el objeto complejo Facultad



3) Sean los datos de una concesionaria de motos:

Valores atómicos

1) CUIT: 33-19332111-5
2) Nombre Concesionaria: Ghiggeri Motos
3) Marca1: Honda
4) Marca2: Suzuki
5) Marca3: Motomel
6) Marca4: Ghiggeri
7) Marca5: Yamaha
8) Marca6: Guerrero
9) Marca7: Zanella
10) Sitio web: motos.gmmotos.com.ar
11) Ciudad1: Resistencia
12) Ciudad2: Castelli
13) Ciudad3: Barranqueras
14) Ciudad4: Corrientes
15) Ciudad5: Formosa
16) DNI: 25334812
17) Teléfono: 3624441515
18) Mail: julioayala@gmmmotos.com.ar
19) Nombre: Julio
20) Apellido: Ayala
21) Fecha Nacimiento: 25-10-1967

Conjuntos

22) Marcas = {Marca1, Marca2, Marca3, Marca4, Marca5, Marca6, Marca7}
23) Ciudades = {Ciudad1, Ciudad2, Ciudad3, Ciudad4, Ciudad5}

Tuplas

24) Concesionaria (objeto complejo)

CUIT	Nombre Concesionaria	Representantes (25)	Marcas (22)	Sitio web	Ciudades (23)
		Tipo tuple	Tipo set		Tipo set

25) Representantes

Código (26)	dni	Teléfono	Mail
Tipo tuple			

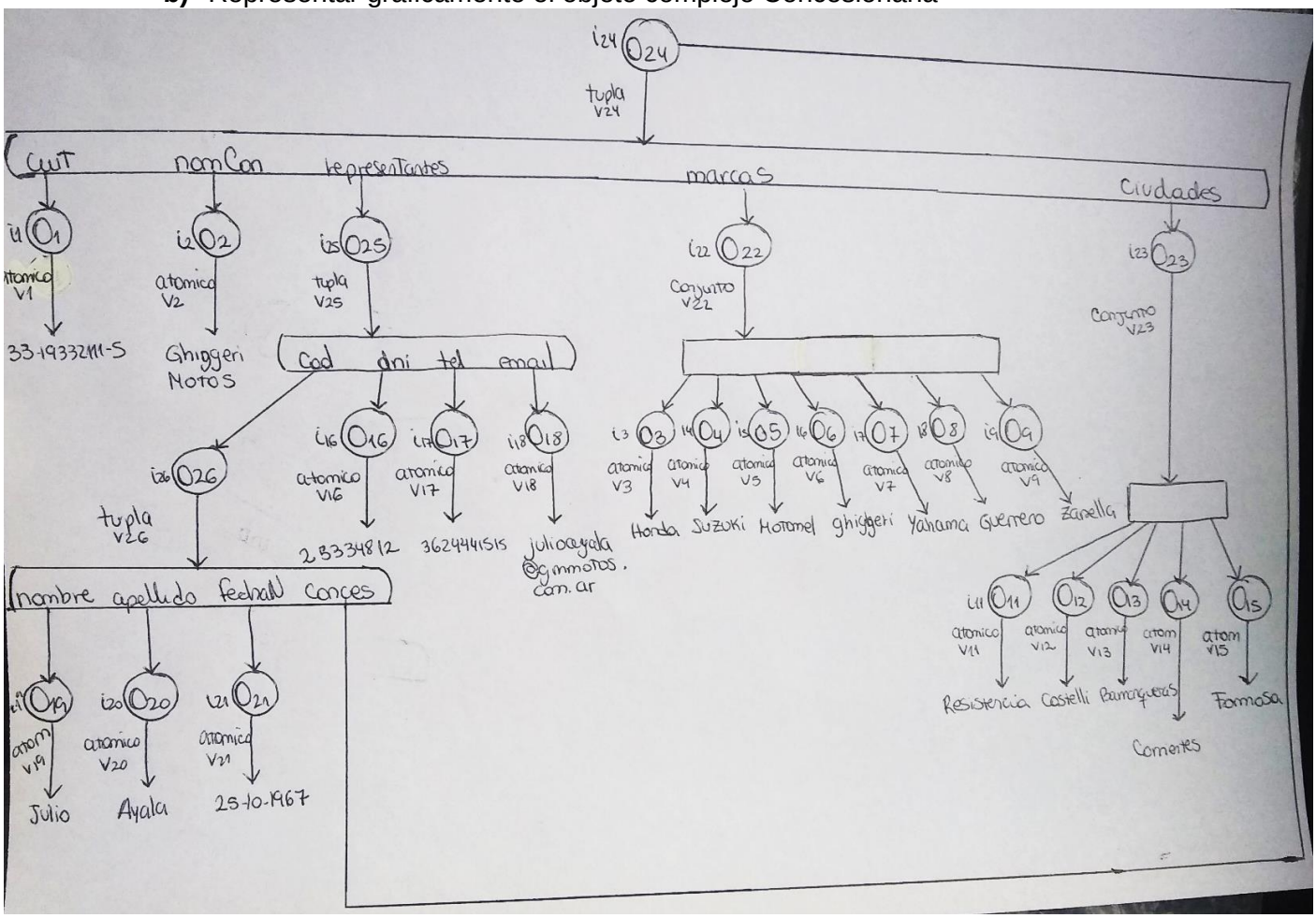
26) Personal

Nombre	Apellido	Fecha Nacimiento	Concesionaria (24)
			Tipo tuple

a) Definir los objetos, teniendo en cuenta los valores y tipos dados

o_1= (i_1, atom, '33-19332111-5') **o_2**= (i_2, atom, 'Ghiggeri Motos') **o_3**= (i_3, atom, 'Honda') **o_4**= (i_4, atom, 'Suzuki') **o_5**= (i_5, atom, 'Motomel') **o_6**= (i_6, atom, 'Ghiggeri') **o_7**= (i_7, atom, 'Yamaha') **o_8**= (i_8, atom, 'Guerrero') **o_9**= (i_9, atom, 'Zanella') **o_{10}**= (i_{10}, atom, 'motos.gmmotos.com.ar') **o_{11}**= (i_{11}, atom, 'Resistencia') **o_{12}**= (i_{12}, atom, 'Castelli') **o_{13}**= (i_{13}, atom, 'Barranqueras') **o_{14}**= (i_{14}, atom, 'Corrientes') **o_{15}**= (i_{15}, atom, 'Formosa') **o_{16}**= (i_{16}, atom, 25334812) **o_{17}**= (i_{17}, atom, 3624441515) **o_{18}**= (i_{18}, atom, 'julioayala@gmmotos.com.ar')
o_{19}= (i_{19}, atom, 'Julio') **o_{20}**= (i_{20}, atom, 'Ayala') **o_{21}**= (i_{21}, atom, '25-10-1967') **o_{22}**= (i_{22}, set,{i_3, i_4, i_5, i_6, i_7, i_8, i_9}) **o_{23}**= (i_{23}, set,{i_{11}, i_{12}, i_{13}, i_{14}, i_{15}}) **o_{24}**= (i_{24}, tuple, <cuit:i_1, nomCon:i_2, representantes:i_{25}, marcas:i_{22}, sitioweb:i_{10}, ciudades:i_{23}>)
o_{25}= (i_{25}, tuple, <codigo:$_{26}$, dni:$_{16}$, telefono:i_{17}, mail:i_{18}>) **o_{26}**= (i_{26}, tuple, <nombre:i_{19}, apellido:i_{20}, fechaN:i_{21}, conces:i_{24}>)

b) Representar gráficamente el objeto complejo Concesionaria



4) Sean los datos del organismo, Dirección de Rentas de la provincia de Corrientes:

Valores atómicos

1) Nombre organismo: DGR_Ctes
2) Sitio web: www.dgrcorrientes.gob.ar
3) Dependencia1: Saladas
4) Dependencia2: Mercedes
5) Dependencia3: Esquina
6) Dependencia4: Alvear
7) Dni: 21324105
8) Profesión: Contador Público
9) Legajo: R-12542
10) Antigüedad: 25

Conjuntos

11) Dependencias= {Dependencia1, Dependencia2, Dependencia3, Dependencia4}

Tuplas

12) Organismo (objeto complejo)

Nombre organismo	Sitio web	Dependencias (11)	Director (13)
		Tipo set	Tipo tuple

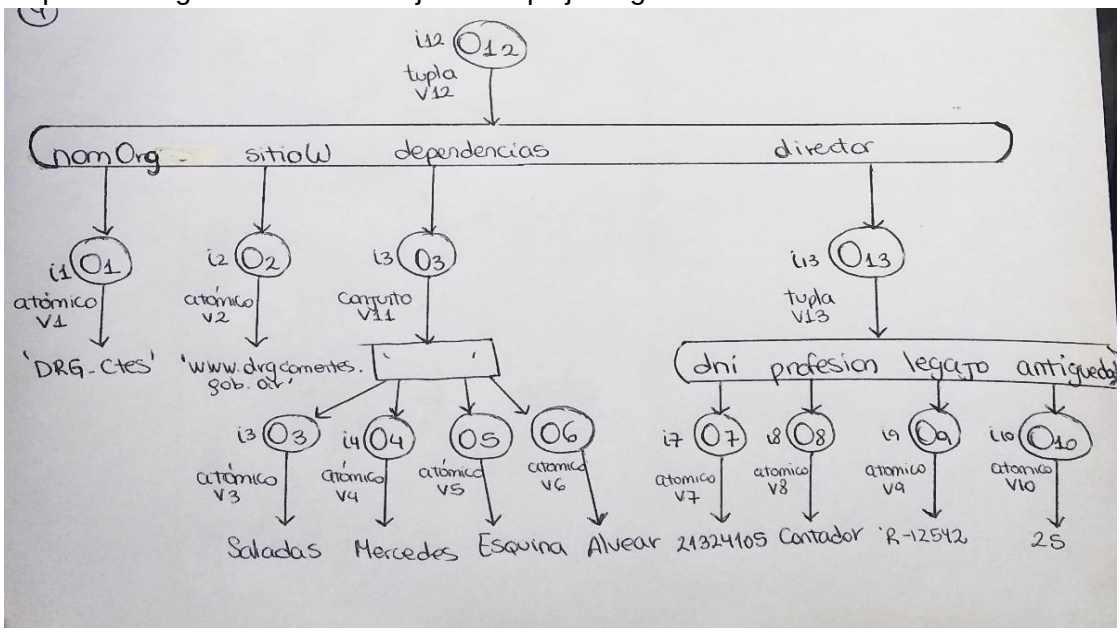
13) Director

Dni	Profesión	Legajo	Antigüedad

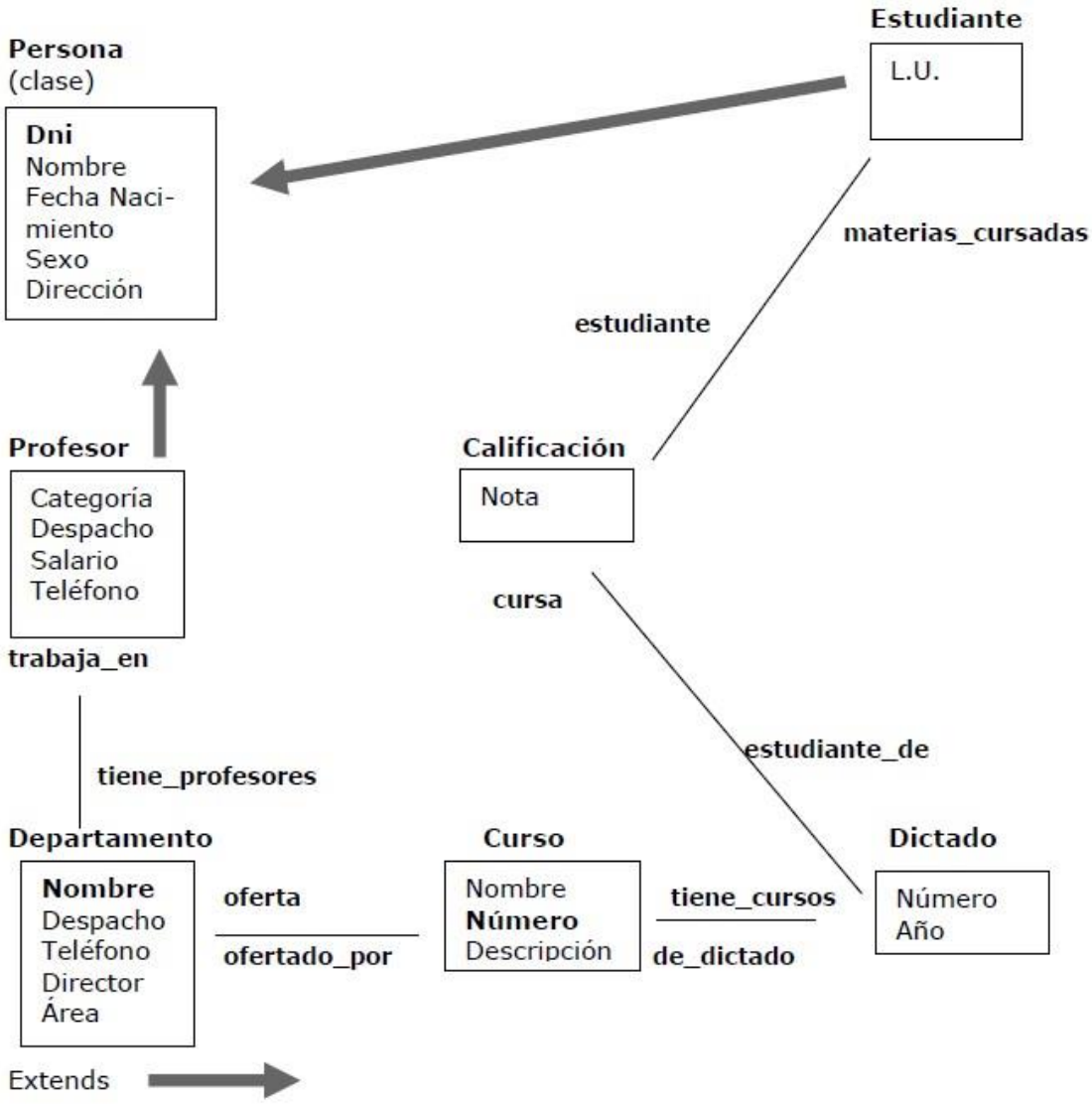
a. Definir los objetos, teniendo en cuenta los valores y tipos dados

$\mathbf{o_1}$= (i_1, atom, 'DGR_Ctes') $\mathbf{o_2}$= (i_2, atom, 'www.dgrcorrientes.gob.ar')
$\mathbf{o_3}$= (i_3, atom, 'Saladas') $\mathbf{o_4}$= (i_4, atom, 'Mercedes') $\mathbf{o_5}$= (i_5, atom, 'Esquina') $\mathbf{o_6}$= (i_6, atom, 'Alvear') $\mathbf{o_7}$= (i_7, atom, '21324105')
$\mathbf{o_8}$= (i_8, atom, 'Contador Público') $\mathbf{o_9}$= (i_9, atom, 'R-12542')
$\mathbf{o_{10}}$= (i_{10}, atom, 25) $\mathbf{o_{11}}$= (i_{11}, set, {i_3, i_4, i_5, i_6})
$\mathbf{o_{12}}$= (i_{12}, tuple, <nomOrg:i_1, sitioW:i_2, dependencias:i_{11}, director:i_{13}>)
$\mathbf{o_{13}}$= (i_{13}, tuple, <dni:i_7, profesion:i_8, legajo:i_9, antiguedad:i_{10}>)

b. Representar gráficamente el objeto complejo Organismo



5) Dada las siguientes relaciones:



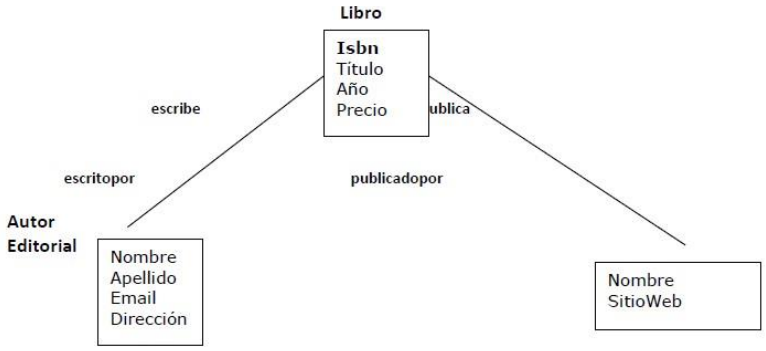
Clase Persona:

```
class Persona (extent personas key DNI)
        {
/*          Definición de atributos          */
        attribute string DNI;
          attribute struct nombrePersona {string nombre1, string nombre2, string apellido1,
                                          string apellido2} nombre;
          attribute date fechanac;
          attribute enum genero {F,M} sexo;
          attribute struct direPersona {string calle, integer numero, string codpostal} direccion;
}

class Profesor extent Persona (extent profesores)
        {
/*          Definición de atributos          */

        attribute integer categoria;
          attribute string despacho;
          attribute float salario;
          attribute string telefono;

/*          Definición de relaciones          */

          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 stringarea;

            relationship set<Profesor> tiene_profesores
                    inverse Profesor::trabaja_en;

            relationship set<Curso> oferta
                    inverse Curso::ofertado_por;
}

class Curso (extent cursos key numero)
```

```
        {
        attribute string nombre;
          attribute integer numero;
          attribute string descripcion;

          relationship set<Dictado> 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<Calificacion> estudiante_de
                inverse Calificacion::cursa;

          relationship set<Curso> 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;
}
```

---

6) Dada las siguientes relaciones:



a. Determine las sentencias necesarias con el lenguaje de definición de objetos ODL, para definir las clases, atributos, relaciones y campos claves (en negrita) de este caso. Considerar a los atributos nombre (nombre pila y 2 apellidos) y dirección (calle, número, código postal) de tipo estructurado.

Clase Libro:

```
class Libro (key ISBN)
        {attribute string ISBN; attribute
          string titulo; attribute integer
          anio; attribute float precio;

          relationship Editorial publicadoPor inverse Editorial::publica; relationship
          Autor escritoPorinverse Autor::escribe;
        };
```

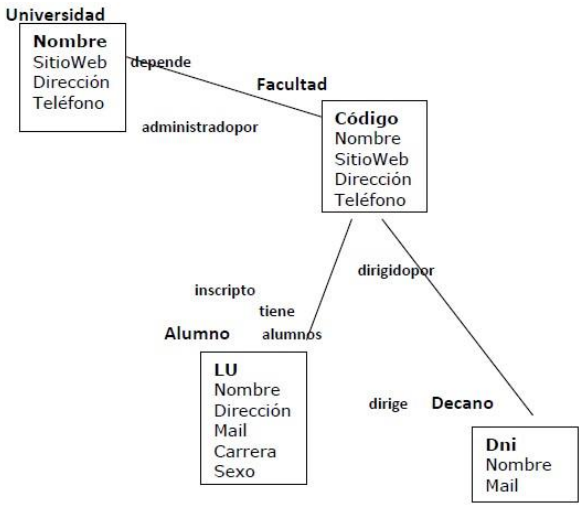
Clase Editorial:

```
class Editorial (key nombre) {attribute
        string nombre;   attribute string
        sitioWeb;

          relationship  Libro  publica  inverse
                  Libro::publicadoPor;
        };
```

Clase Autor:

```
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;
        );
```

7) Dada las siguientes relaciones:



a. Determine las sentencias necesarias con el lenguaje de definición de objetos ODL, para definir las clases, atributos, relaciones y campos claves (en negrita) de este caso. Considerar al atributo Genero de tipo enumerado (M o F). **Clase Facultad:**

```
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;
        };
```

Clase Universidad:

```
class Universidad (key nombre)
        {
          attribute string nombre;
          attribute string sitioWeb;
          attribute string direccion;
          attribute string telefono;
          relationship Facultad depende
          inverse
          Facultad::administradoPor;
        };
```

Clase Alumno: 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;
};
```

Clase Decano:

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

Serie de Ejercicios Prácticos Nº4

Bases de Datos Objeto Relacionales

El termino se usa para describir a una Base de Datos que ha evolucionado desde el Modelo Relacional hasta una Base de Datos Hibrida que contiene ambas tecnologías: Relacional y Objetos.

Algunos conceptos que nos encontramos durante la resolución del practico fueron los siguientes:

Tipo Identidad de Objetos: Cada columna identidad tiene los siguientes valores:

- valor inicial, incremento, valor máximo, valor mínimo y opción de ciclo.

Tipos Datos

1. **Predefinidos** (predefined)
   Son los numéricos, caracteres, booleano, DateTime, intervalos.

2. **Construidos (Constructed)**
   - *Atómicos*
   - *Tipo referencia (reference type)*
   - *Compuestos*
   - *Colecciones:*
     Vector(Array): Colección ordenada de valores, cuyos elementos se referencian por su posición ordinal.
     Multiconjunto (multiset): Colección no ordenada de valores no necesariamente distintos - *Tipo fila (row type):*
     Es una secuencia de uno o más campos

3. **Definidos por el usuario**
   - Distinto(Distinct)
   - Estructurado (structured types)
     Un tipo estructurado puede heredar (ser definido a partir de) otro tipo estructurado como un Subtipo o Supertipo.
     Al final debemos especificar con las palabras NOT FINAL para indicar que es parte de una jerarquía y que tiene más dependencias. Y FINAL indicando que debajo de su nivel, no existen más dependencias.

Objetos Grandes(LOB)

Los tipos de atributos Objet Large permiten que se pueda almacenar mayor capacidad de datos. Pueden ser de tipo Carácter o Binarios con diferentes capacidades, por ejemplo, 40K,2M,1G. Entonces:

BLOB(binarios): Audios, imagen, video
CLOB(carácter): textos, como ser alguna descripción, resúmenes, introducciones.

Método (Method)

Es una función SQL asociada a un Tipo definido por el Usuario, donde la signatura se define en el esquema dentro de la definición del Tipo definido por el Usuario (CREATE TYPE… METHOD …). Sin embargo, el cuerpo del método se define aparte ya sea en el esquema directamente o en un módulo.

Se invocan utilizando la notación punto.

---

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.

```
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 INT, precio
DECIMAL (9,2)
)
```

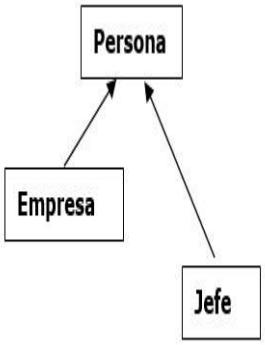
b) Crear la tabla Editorial, con los atributos: Código de libro, Título de la obra, Descripción, Autor, Foto de la portada del libro, Video de presentación y Año edición.

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

```
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 INT
)
```

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.

```
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),
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
INT
) 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.

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

    Variante:
    CREATE TABLE universidad (
        nombre VARCHAR (100),    rector
    VARCHAR (100),    dirección VARCHAR
    (100),  facultades VARCHAR (15) MULTISET,
        código_postal INT,
        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.

```
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”,
```

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.

```
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

```
CREATE TABLE banco (
        identifiación 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)
),
```

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.

```
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

```
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.

```
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

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

Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Cátedra: Bases de Datos II

Año: 2020

Trabajos Prácticos 1ra. Parte

Alumnos

- Diaz Edith Nancy Adela Lu: 49581,
- Gómez Mario Andrés. Lu: 42291.

Práctico 1- Base de Datos Activas

1) Crear la tabla Empleados, con los siguientes atributos: DNI Empleado, Legajo, Apellido y Nombres, Cargo y Sueldo, tener en cuenta lo siguiente para cada uno de ellos:

- DNI: clave primaria
- Legajo: cumpla con la restricción de unicidad
- Cargo: este comprendido entre los valores 50-120
- Sueldo: no sea mayor a 30000

```
1  Create table Empleados
2  (
3      Dni      Char(8) Primary Key,
4      Legajo       Char(6) Unique,
5      Ape_Nom Varchar2(50),
6      Cargo        Number Check(cargo between 50 and 120),
7      Sueldo       Dec(8,2) Check(sueldo <= 30000)
8  );
```

Alternativa: crear un dominio de valor para el atributo DNI, como entero de 8 dígitos, verificando que sólo acepte valores mayores a 0.

```
1  Create Domain Documento Int(8) Check(Documento > 0);
2
3  Create table Empleados
4  (
5      Dni      Documento Primary Key,
6      Legajo       Char(6) Unique,
7      Ape_Nom Varchar2(50),
8      Cargo        Number Check(cargo between 50 and 120),
9      Sueldo       Dec(8,2) Check(sueldo <= 30000)
10 );
```

Crear un trigger de nombre Control_sueldo, que impida que se incremente el sueldo en más de un 20%, cada vez que se modifique dicho atributo. El evento que dispara el trigger es UPDATE (sueldo), y su tiempo de acción es BEFORE.

```
13 Create Trigger Control_sueldo
14
15     /*
16     * Cuando se actualiza el atributo sueldo de la tabla Empleados
17     * se activa este disparador, controlando que el sueldo no se
18     * incremente en un más 20% del valor del sueldo anterior
19     */
20
21     Before Update of sueldo on Empleados
22         For each row
23         Begin
24             If :new.sueldo> (:old.sueldo * 1.20)
25                 Then   "Error-Debe ir sentencia que cancela la actualización"
26             End if:
27         End;
```

Crear un trigger de nombre Adicional_cargo, para el caso en que el cargo sea igual a 120, deberá de incrementarse el sueldo en un 10% del importe básico para ese cargo, residente en la tabla Importes_basicos (clave primaria: cargo). El evento que dispara el trigger es INSERT/UPDATE (cargo), y su tiempo de acción es AFTER.

```
30  Create Trigger Adicional_cargo
31      Alter Insert  or Update of cargo on Empleados
32      For each row
33          Begin
34          If  :new.cargo = 120 then
35              :new.sueldo = :new.sueldo +
36                  (Select básico from Importes_basicos where cargo = :new.cargo) * 0.10
37          End if:
38      End;
```

2) Crear la tabla Seguros de automotores (Seguros_autos), con los siguientes atributos: Nro. Póliza, DNI Cliente, Marca, Modelo, Patente, Nro. de motor, Trimestre e Importe a pagar (trimestral), considerar:

- Nro. póliza: Clave primaria
- DNI Cliente: tiene una integridad referencial con la tabla Clientes (DNI)
- Modelo: tenga como valor predeterminado '2017'
- Marca: sólo pueda contener como dato uno de los siguientes valores ´Ford, Renault, Fiat, Peugeot, VW, Toyota´
- Periodo: 201801 (Año 2018 y 1er trimestre)
- Importe a pagar:no puede ser nulo
- Constituir clave externa con los atributos marca y modelo, con referencia a la tabla Vehículos(marca,modelo)

```
1   Create table Seguros_autos
2   (
3       Nro_poliza   Char(8) Primary Key,
4       Dni          Char(6) references Clientes(Dni),
5       Marca        Char(7)
6                    Check(marca In ('Ford','Renault','Fiat','Peugeot','VW', 'Toyota')),
7       Modelo       Char(4) default ('2017'),
8       Patente      Char(10),
9       Nro_motor    Varchar2(30),
10      Periodo      Char (6) default ('201801')
11      Imp_pagar    Dec(10,2) notnull,
12
13      ForeignKey(marca,modelo) referencesVehiculos(marca,modelo)
14  );
```

Crear un trigger de nombre Facturas_mensuales, en el cual se registrarán las cuotas que se deben de abonar en cada trimestre, para ello se insertaran en la tabla Cuotas_Seguros, los comprobantes respectivos a cada trimestre considerado, para este caso son 3 nuevos registros a incorporar.

Los atributos de la tabla Cuota_Seguros (tabla ya existente), se actualizarán de la siguiente manera:

- Nro_comprobante: se compone del Nro. póliza, periodo y cuota (01, 02 o 03)
- Monto facturado de cada cuota (corresponde a 1/3 del importe a pagar de la tabla Seguros).
- Fecha de pago: el día 15 de cada mes (15/01/18, 15/02/18, 15/03/18)

El evento que dispara el trigger es INSERT (Tabla Seguros_autos), y su tiempo de acción es AFTER.

```
Create Trigger Facturas_mensuales
    After Insert on Seguros_autos
    For each row
        Declare  importe_cuota Dec(10,2)    /* var aux
        Declare  nro_cuota integer
        Declare comprobante Char (16)
        Declare fecha_pago_aux Char (8)
        Begin

              /*  Insertamos las filas en la tabla Facturas_mensuales
                  de la respectiva póliza de seguro  */
            importe_cuota= :new.imp_pagar / 3
            nro_cuota = 1

            While nro_cuota< 4
            Loop
            comprobante = :new.Nro_poliza + :new.Periodo +convert(char(2), nro_cuota))

              /*   fecha pago:  15 fecha de vencimiento + nro_cuota es el mes  +
                      substring (:new.periodo,3,2) obtengo el año*/

            fecha_pago = ´15' + convert(char(2), nro_cuota)) + substring (:new.periodo,3,2)

              /*  todo esto lo concatenamos y luego al insertar lo
                      convertimos a tipo fecha (date)  */

            Insert into Cuotas seguros values ( comprobante, importe_cuota,
                convert(fecha_pago, date))

                nro_cuota = nro_cuota + 1
            End Loop
    End;
```

3) Crear la tabla Pedidos, con los siguientes atributos: Nro. de orden de pedido, fecha de la orden de pedido, CUIT del cliente, CUIT del proveedor, nro. de producto, tipo de pedido y cantidad pedida del producto, considerar:

- CUIT del cliente y Nro. de orden de pedido: Clave primaria
- Nro. de orden pedido: cumpla con la restricción de unicidad
- Fecha de la orden de pedido: valor por defecto la fecha actual del sistema
- CUIT del cliente: integridad referencial con la tabla Clientes (Cuit)
- Tipo de pedido: puede contener únicamente M, C, G, H, N
- Cantidad pedida: valores mayores a 0

```
Create table Pedidos
(
    Nro_orden       Char(6) not null unique,
    Fecha_pedido    Date default today,
    Cuit            Char(11) not null,
    Cuit_proveedor  Char(11) not null,
    Nro_producto    Number,
    Tipo_pedido     Char(1)
                    Check (Tipo_pedido In ('M','C','G','H','N')),
    Cant_pedida     Int Check(Cant_pedidad>0)

    Cuit references Clientes(Cuit),
    Primary key (Cuit,nro_orden),
);
```

Crear un trigger de nombre Control_pedido, que impida dar de alta una orden de pedido, cuando la cantidad pedida del producto sea superior al stock_actual (atributo) de la tabla Stock_productos, caso contrario actualizar el stock_actual, que resulta luego de restarle la cantidad del pedido.
El evento que dispara el trigger es INSERT, y su tiempo de acción es BEFORE.

```
Create Trigger Control_pedido
    Before Insert on Pedidos
    For each row
        Begin

            /* evalúa que la nueva cantidad pedida no sea mayor al valor del stock actual

            If :new.cant_pedida> (Select stock_actual from Stock_productos
                                  where producto = :new.nro_producto) then
            "Error-Cantidad pedida superior al stock disponible"
            Else
                Update Stock_productos
                    Set stock_actual = stock_actual - :new.cant_pedida
                    where producto = :new.nro_producto
            End if;
        End;
```

4) Respecto a la tabla Pedidos, del ejercicio 3), agregue las siguientes restricciones:

- "ClaveProducto" de tal manera que nro. de producto y tipo de pedido tenga una referencia externa a la tabla Productos(Id_producto, id_tipo)
- "CantidadPermitida" donde el atributo Cantidad pedida, sólo acepte valores comprendidos entre 10 y 2000.
- "ControlCUIT" donde el atributo CUIT del cliente no deberá de coincidir con el CUIT del proveedor, previendo deshabilitar esta comprobación en los datos ya existentes.
- "ControlStock" a fin de verificar que la Cantidad pedida, no sea mayor al stock actual del producto requerido, atributo stock_actual de la tabla Productos.

```
Constraint Clave Producto Foreign Key (nro_producto, tipo_pedido)
            references Productos (Id_producto,Id_tipo)

Constraint CantidadPermitida check(cant_pedidabetween 10 and 2000)

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

5) Crear la tabla Clientes, con los atributos: código de cliente, nombre, dirección, código postal e importe base.

Crear la tabla Facturas, con los atributos: Nro. de factura, fecha de la factura, código de cliente, tipo de descuento, valor de IVA e importe de la factura.

Contemplar:

- Crear un valor de dominio para código de cliente, entero de 7 dígitos, mayor a 0

Tabla Clientes:

- Código cliente: referenciar al valor dominio creado anteriormente, clave primaria
- Nombre y dirección: no nulos
- Código postal: integridad referencial con la tabla CodPostales(cp)

Tabla Facturas:

- Nro. Factura: clave primaria
- Código de cliente: referenciar al valor dominio creado e integridad referencial a la tabla Clientes(codcliente)
- Tipo de descuento: sólo puede contener valores comprendidos entre 5y20 (porcentajes %)respectivamente
- IVA: sólo puede contener alguno de los valores 15 o 21 (porcentajes).
- Importe de la factura, no nulo.

```
Create Domain Codigo Cliente Int(7) check (CodigoCliente> 0);

CreatetableClientes
(
    Cod_cliente CodigoCliente primarykey,
    Nombre  Varchar2(40)  not null,
    Direccion   Varchar2(35)  not null,
    Cod_postal  Char(5),
    Importe_base    Dec(8,2)

    Foreign key (cod_postal) references CodPostales(cp),
);

Create table Facturas
(
    Nro_factura     Char(8) primarykey,
    Fecha_fact      Date,
    Cliente_factura CodigoCliente,
    Tipo_descuento  int(2) check(tipo_descuento between 5 and 20),
    Iva             int(2) check((iva= '15') or (iva = '21')),
    Importe         Dec(10,2) not null

    Foreign key(Cliente_factura) references Clientes(codcliente),
);
```

Crear un trigger de nombre Calcular_importe, que permita obtener el importe de la factura de cada cliente, el cual se obtiene sobre el importe base, descontando el porcentaje del tipo de descuento, y sobre este resultado aplicar el porcentaje del IVA (15 o 21%), para obtener el valor definitivo.
El evento que dispara al trigger es UPDATE respecto al atributo importe base de la tabla Facturas y su tiempo de acción es BEFORE.

```
Create Trigger Calcular_importe
    Before Update importe_base on Clientes
    For each row
    Declare  resultado Dec(10,2)    /* var aux
        Begin

            /* Descontamos del importe_base el Importe segun el descuento

            resultado = :new.importe_base - ((:new.tipo_descuento * :new.importe_base)/100)

            /* Incrementamos la variable segun el porcentaje del IVA

            resultado = resultado + ((:new.iva * resultado)/100)

            Update Facturas
                Set importe = resultado
                Where cliente_factura= :new.cod_cliente
        End;
```

6) Crear la tabla Empleados_baja, con los atributos DNI Empleado, Legajo, Cargo, Usuario y Fecha.

```
Create table Empleado_baja
(
    Dni     Char(8),
    Legajo  Char(6),
    Cargo   number,
    Usuario Varchar2(15),
    Fecha   Date
);
```

Luego crear el trigger de nombre Empleado_eliminado, que permita insertar en la tabla Empleados_baja, los datos de aquel empleado que se elimina de la tabla Empleados (del ejercicio 1), donde las columnas usuario y fecha se grabarán con las variables delsistema USER y SYSDATE.

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

Crear otro trigger de nombre Baja_usuario, que permita eliminar de la tabla Usuarios, aquella tupla cuyo DNI de esta última tabla, sea igual al DNI del empleado eliminado en el paso anterior.

El evento que dispara ambos triggers es DELETE, y su tiempo de acción es AFTER.

```
49  Create trigger Baja_usuario
50  After delete on Empleados
51          Begin
52                  */  Eliminamos el usuario del empleado, con el cual accedía al sistema
53                      de gestión de la empresa a la cual pertenecía  */
54
55              Delete from Usuarios where usuarios.dni= :old.Dni;
56          End;
```

7) Modificar el trigger Baja_usuario creado en el ejercicio anterior, de tal manera que, para eliminar de la tabla de Usuarios, en lugar de hacerlo con el atributo DNI, demos la baja del empleado mediante su número de Legajo, utilizar el comando Replace trigger.

```
1  Replace trigger Baja_usuario
2  After delete on Empleados
3          Begin
4              Delete from Usuarios where usuarios.legajo= :old.legajo;
5          End;
6
```

Indicar los comandos Sql necesarios para desactivar el trigger Calcular_importe y también todos los triggers que estén asociados a la tabla Consumos.

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

8) Respecto de la tabla Facturas, creada en el ejercicio 5), incorporar a la tabla las columnas de Fecha_vto y Fecha_Pago de tipo fecha (date) y la columna intereses de tipo entero.

```
1      Alter table Facturas Add fecha_vtodate;
2      Alter table Facturas Add fecha_pagodate;
3      Alter table Facturas Add intereses int;
```

Crear un trigger de nombre Recargo_factura, que detecte aquellos pagos fuera de término, por lo que si la fecha de pago es superior a la fecha de vencimiento, asignar el valor 1 al

atributo intereses, este recargo se vería reflejado en el cobro de la próxima factura del cliente.

El evento que dispara al trigger es UPDATE respecto al atributo fecha de pago de la tabla Facturas y su tiempo de acción es AFTER.

```
Create Trigger Recargo_factura
    After Update fecha_pago on Facturas
    For each row
        Begin
            If fecha_pago>fecha_vto then
            Update Facturas
                Set intereses = 1
                Where cliente_factura = :new.cod_cliente
        End;
```

9) Insertar un nuevo trigger que aplique el recargo que corresponde a los clientes que pagaron fuera de término su factura, en los casos en que el atributo intereses sea igual a 1 se aplicara un recargo del 2,5% respecto del monto total de factura, por lo que el nuevo monto a pagar se verá incrementado con el importe resultante del recargo aplicado (Importe de la factura = Importe de la factura + Importe recargo).

El evento que dispara al trigger es UPDATE respecto al atributo intereses de la tabla Facturas y su tiempo de acción es AFTER.

```
Create Trigger Recargo
After UPDATE of Intereses on Factura
For each row
    Begin
        declare aux dec(10,2)

        If :old.intereses = 1 then
            aux = :old.importe_factura + [(:old.importe_factura*2,5)/100]

        UPDATE Factura
            :new.importe_factura = aux
            where Cliente_factura = Cod_cliente
    end;
```

Práctico 2-Base de Datos Distribuidas.Algoritmos de Optimización de Consultas

1) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los clientes de una editorial para sus respectivas sucursales:

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

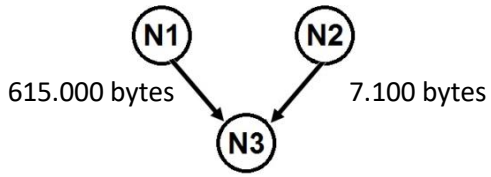
a) Tamaño de la relación:

Nodo 1: 5000 registros x 123 bytes = 615.000 bytes
Nodo 2: 100 registros x 71 bytes = 7100 bytes

b) Resultado de la consulta:
5000 tuplas (por cliente) x (30 bytes + 30 bytes) = 300.000 bytes

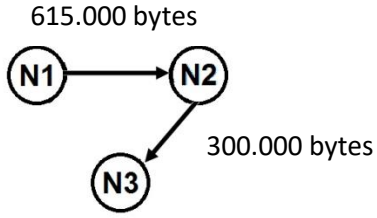
- **Nodo 3 como nodo resultado (INNER JOIN).**

Opción 1



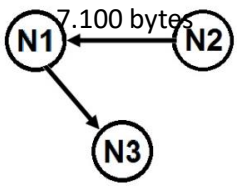
Total de la Transacción: 615.000 bytes + 7.100 bytes = 622.100 bytes

Opción 2



Total de la Transacción: 615.000 bytes + 300.000 bytes = 915.000 bytes

Opción 3



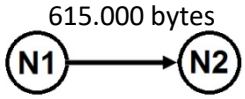
300.000 bytes

Total de la Transacción: 7.100 bytes + 300.000 bytes = 307.000 bytes

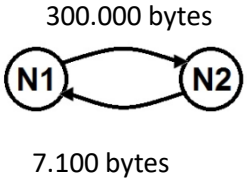
Respuesta.: La opción 3 es la mejor opción para minimizar la transferencia.

- **Nodo 2 como nodo resultado (INER JOIN).**

Opción 1



Opción 2



Total de la Transacción: 7.100 bytes + 300.000 bytes = 307.000 bytes

- **Nodo 2 como nodo resultado (SEMI JOIN)**

Paso 1 4 bytes x 100 sucursales = 400 bytes

Paso 2 (4 bytes + 30 bytes) x 5.000 registros = 170.000 bytes

Total de la transacción: 400 bytes + 170.000 bytes = 170.400 bytes

2) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los profesores y facultades de nuestra universidad:

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

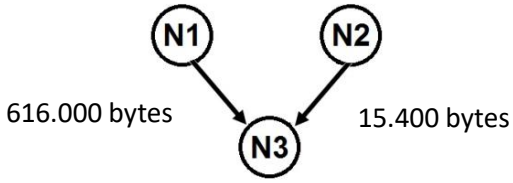
2 bytes 32 bytes 35 bytes 8 bytes
Contiene: 200 registros Longitud del registro: 77 bytes

a) Tamaño de la relación:
Nodo 1: 88 bytes x 7000 registros = 616.000 bytes
Nodo 2: 77 bytes x 200 registros = 15.400 bytes

b) Resultado de la Consulta:
7000 tuplas (por Profesor) x (30 bytes + 32 bytes) = 434.000 bytes

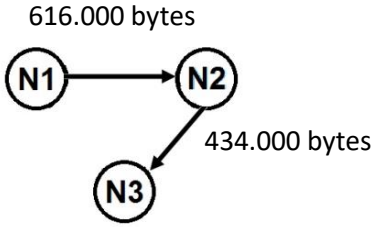
- **Nodo 3 como nodo resultado (INNER JOIN).**

Opción 1



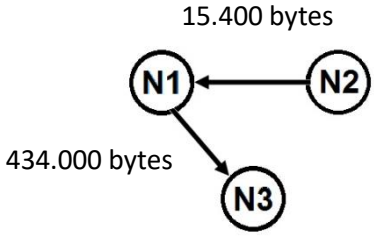
Total de la Transmisión: 616.000 + 15.400 = 631.400 bytes

Opción 2



Total de la Transmisión: 616.000 + 434.000 = 1.050.000 bytes

Opción 3

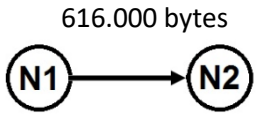


Total de la Transmisión: 15.400 + 434.000 = 449.400 bytes

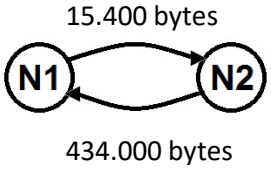
Respuesta: La opción 3 es la mejor opción para minimizar el transporte de la información.

Nodo 2 como nodo resultado (INNER JOIN).

Opción 1



Opción 2



Total de la Transmisión: 15.000 + 434.000 = 449.000 bytes.

Respuesta: La opción 2 es la más óptima.

- **Nodo 2 como nodo resultado (SEMI JOIN)**

Paso 1 | 2 bytes x 200 sucursales = 400 bytes

Paso 2 | (2 bytes + 30 bytes) x 7.000 registros = 224.000 bytes

Total de la transferencia de información: 400 + 224.000 = 224.400 bytes

3) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los remedios y sus correspondientes laboratorios:

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
5 bytes	20 bytes	25 bytes	40 bytes

Contiene: 100 registros Longitud del registro: 90 bytes

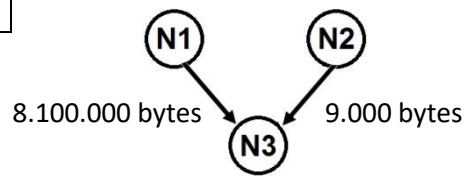
a) Tamaño de la relación:

Nodo 1: 100.000 registros x 81 bytes = 8.100.000 bytes
Nodo 2: 100 registros x 90 bytes = 9.000 bytes

b) Resultado de la consulta:
100.000 registros x (20 bytes + 20 bytes) = 4.000.000 bytes

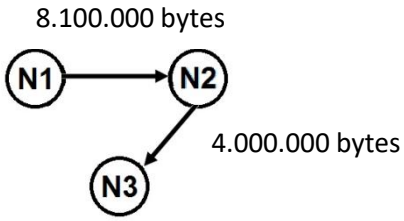
- **Nodo 3 como nodo resultado (INNER JOIN).**

Opción 1



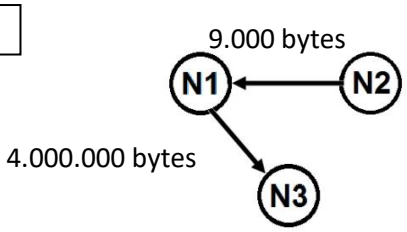
Total de la Transmisión de información: 8.100.000 bytes + 9.000 bytes = 8.109.000 bytes.

Opción 2



Total de la Transmisión de información: 8.100.000 bytes + 4.000.000 bytes = 12.100.000 bytes

Opción 3

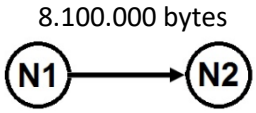


Total de la Transmisión de información: 9.000 bytes + 4.000.000 bytes = 4.009.000 bytes

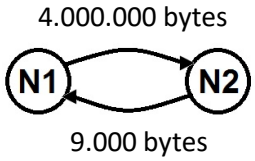
Respuesta: La opción 3 es la mejor opción para minimizar la transferencia.

- **Nodo 2 como nodo resultado (INNER JOIN).**

Opción 1



Opción 2



Total de la transmisión: 9.000 bytes + 4.000.000 bytes = 4.009.000 bytes

Respuesta: La opción 2 es la más óptima.

- **Nodo como nodo resultado (SEMI JOIN)**

Paso 1	5 bytes x 100 sucursales = 500 bytes

Paso 2	(5 bytes + 20 bytes) x 100.000 registros = 2.500.000 bytes

Total de la transacción: 500 + 2.500.000 = 2.500.500 bytes

4) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los empleados de una empresa y los respectivos departamentos laborales.

Nodo 1 : Empleados

DNI	Nombre y Apellido	Dirección	Mail	Departamento	Categoría	Sueldo
8 bytes	25 bytes	25 bytes	25 bytes	3 bytes	2 bytes	12bytes

Contiene: 10000 registros Longitud del registro: 100 bytes

Nodo 2 : Departamentos

Código	Nombre	Dni Jefe	Sector
3 bytes	20 bytes	8 bytes	4 bytes

Contiene: 100 registros Longitud del registro: 35 bytes

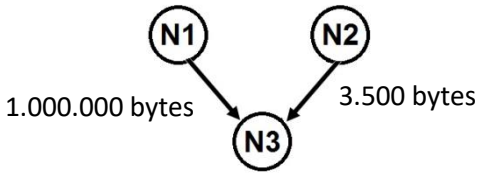
a) Tamaño de la relación:

Nodo 1: 10.000 registros x 100 bytes = 1.000.000 bytes
Nodo 2: 100 registros x 35 bytes = 3.500 bytes

b) Resultado de la consulta:
100.000 registros x (20 bytes + 20 bytes) = 450.000 bytes

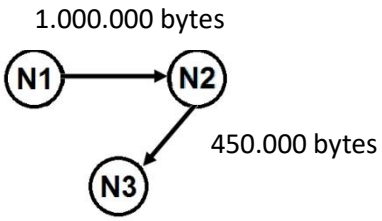
- **Nodo 3 como nodo resultado (INNER JOIN).**

Opción 1



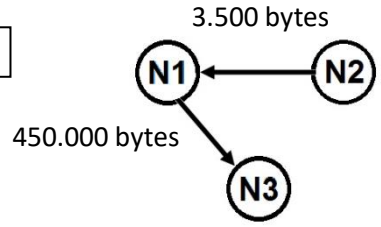
Total de la transmisión de la información: 1.000.000 + 3.500 = 1.003.500 bytes

Opción 2



Total de la transmisión de la información: 1.000.000 + 4500.000 = 1.450.000 bytes

Opción 3

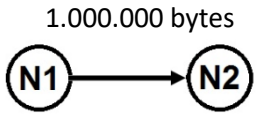


Total de la transferencia: 3.500 + 450.000 = 453.500 bytes

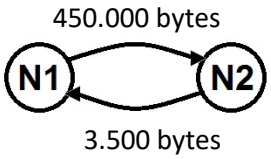
Respuesta: La opción 3 es la mejor opción para minimizar la transferencia.

- **Nodo 2 como nodo resultado (INNER JOIN).**

Opción 1



Opción 2



Total de transferencia de información: 450.000 + 3.500 = 453.500 bytes

Respuesta: La opción 2 es la más óptima.

- **Nodo 2 como nodo resultado (SEMI JOIN)**

Paso 1 3 bytes x 100 registros = 300 bytes

Paso 2 (3 bytes + 25 bytes) x 10.000 registros = 280.000 bytes

Total de transferencia de información: 300 + 280.000 = 280.300 bytes

5) Sean las siguientes tablas de bases de datos distribuidas en varios nodos, pertenecientes a los vehículos automotores que se encuentran registrados en el país, de los registros seccionales y cuáles son sus marcas y modelos.

Nodo 1: Vehículos

Nro. Patente	Nombre y Apellido del Titular	DNI	Dirección	Código de Marca	Código de Modelo	Registro Seccional
8 bytes	25 byte	8 bytes	25 bytes	3 bytes	4 bytes	25 bytes

Contiene: 13000000 registros Longitud del registro: 97 bytes

Nodo 2: Marcas

Código	Nombre Marca	País origen
3 bytes	20 bytes	25 bytes

Contiene: 100 registros Longitud del registro: 48 bytes

Nodo 3: Modelos

Código	Nombre Modelo	Precio
4 bytes	20 bytes	7 bytes

Contiene: 3000 registros Longitud del registro: 31

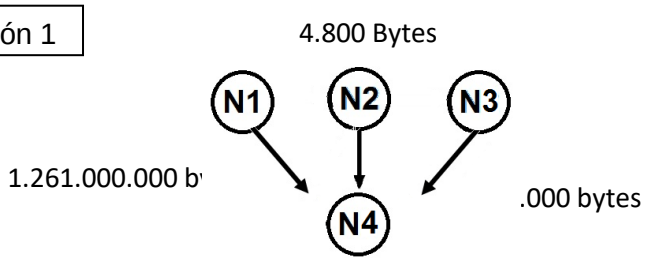
a) Tamaño de la relación

Nodo 1: 13.000.000 registros x 97 bytes = 1.261.000.000 bytes
Nodo 2: 100 registros x 48 bytes = 4.800 bytes
Nodo 3: 3.000 registros x 31 bytes

b) Resultado de la consulta
13.000.000 tuplas (por patente) x (25 bytes + 25 bytes + 20 bytes + 20 bytes) =
= 1.170.000.000 bytes

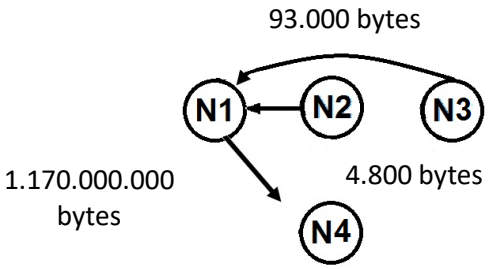
- **Nodo 4 como resultado (INNER JOIN).**

Opción 1



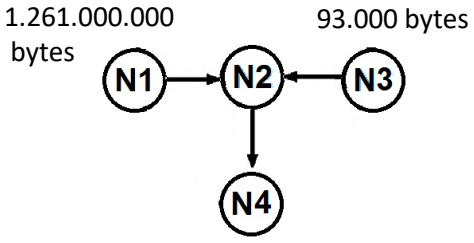
Total de la transmisión de información: 1.261.000.000 + 4.800 + 93.000 = 1.261.097.800 bytes

Opción 2



Total de la transmisión de información: 1.170.000.000 bytes + 93.000 + 4.800 bytes = 1.170.097.800 bytes

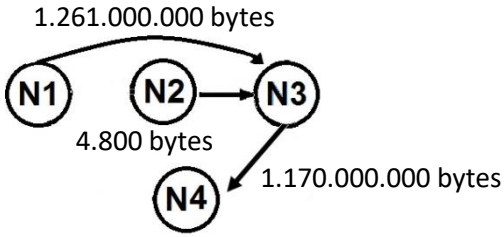
Opción 3



1.170.000.000 bytes

Total de la transmisión de información: 1.261.000.000 bytes + 93.000 bytes +
1.170.000.000 bytes = 2.431.093.000 bytes

Opción 4

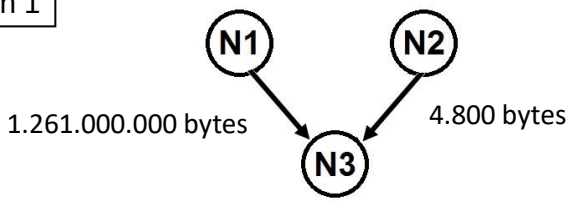


Total de la transmisión: 1.261.000.000 + 4.800 + 1.170.000.000 = 2.431.004.800 bytes

Respuesta: La opción 2 es la mejor opción para minimizar la transferencia.

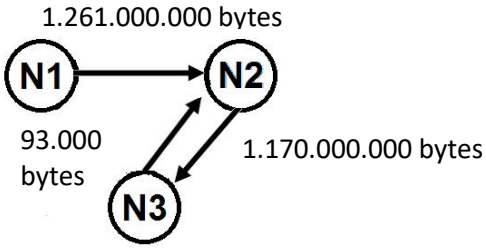
Nodo 3 como resultado (INNER JOIN)

Opción 1



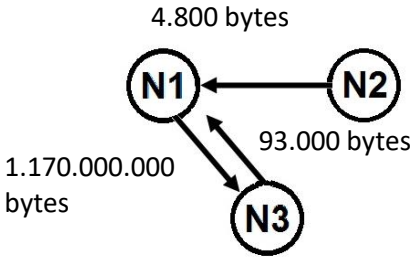
Total de la transmisión: 1.261.000.000 + 4.800 = 1.261.004.800 bytes

Opción 2



Total de la transmisión: 1.170.000.000 bytes + 4.800 bytes = 1.170.004.800 bytes.

Opción 3



Total de la transmisión: 1.261.000.000 + 1.170.000.000 = 2.431.000.000 bytes.

Respuesta: La opción 2 es la mejor opción para minimizar la transferencia.

Práctico 3- Base de Datos Orientado a Objetos Objetos Complejos

Considerar los siguientes datos, para definir los objetos mediante la forma (i,c,v), donde i=identificador del objeto, c=constructor, v=estado o valor actual, contemplando los siguientes constructores: atom, set y tuple.

1) Sean los datos de una empresa de telefonía celular:

Valores atómicos

1) Nro. Empresa: 100 2) Nombre Empresa: Telecom
3) Sucursal1: Posadas 4) Sucursal2: Salta 5) Sucursal3: Formosa
6) Fecha creación: 01-02-1994 7) Dni: 24987422 8) Sueldo: 35000

Conjuntos

9) Sucursales = {Sucursal1, Sucursal2, Sucursal3}

Tuplas

10) Empresa (objeto complejo)

Nro. Empresa	Nombre Empresa	Sucursales**(9)**	Fecha Creación	Presidente**(11)**
		tipo set		tipo tuple

11) Presidente

Dni	Sueldo

a) Objeto Complejo: Empresa.

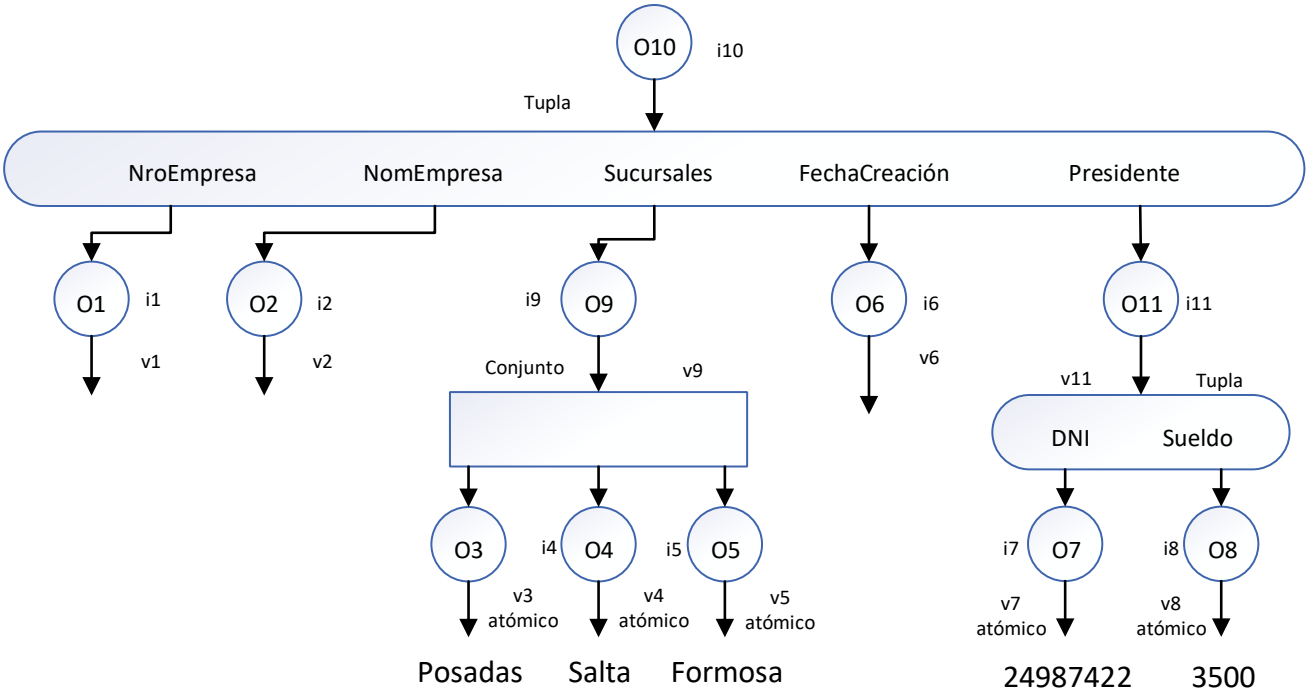
O_1 = (I_1, ATOM, 100)
O_2 = (I_2, ATOM, ”Telecom”)
O_3 = (I_3, ATOM, ”Posadas”)
O_4 = (I_4,ATOM, “Salta”)
O_5 = (I_5,ATOM, “Formosa”)
O_6 = (I_6,ATOM, “01-02-1994”)
O_7 = (I_7,ATOM, “24987422”)
O_8 = (I_8,ATOM, “35000”)
O_9 = (I_9, SET, {I_3 , I_4 , I_5 })
O_{10} = (I_{10}, TUPLE, <Nro. Empresa: I_1 , Nombre Empresa: I_2, Sucursales: I_9 , Fecha Creación: I_6 , Presidente: I_{11} >)
O_{11} = (I_{11}, TUPLE, <Dni: I_7 , Sueldo: I_8>)

b) Representación gráfica del objeto complejo Empresa

Átomos

Conjuntos

Tuplas



2) Sean los datos de una facultad:

Valores atómicos

1) Código Facultad: 11
2) Nombre Facultad: Medicina
3) Sede1: Centro
4) Sede2: Campus Cabral
5) Posgrado1: Higiene
6) Posgrado2: Enfermería
7) Posgrado3: Salud Social
8) Dirección: Moreno 1240
9) Legajo: M1378
10) Nombre: Carlos
11) Apellido: Monti
12) CUIL: 20-18980067-4
13) Mail: cm@med.unne.edu.ar

Conjuntos

14) Sedes = {Sede1, Sede2}
15) Posgrados = {Posgrado1, Posgrado2, Posgrado3}

Tuplas

16) Facultad (objeto complejo)

Código Facultad	Nombre Facultad	Sedes **(14)**	Posgrados **(15)**	Dirección	Decano **(17)**
		tipo set	tipo set		tipo tuple

17) Decanos

Id**(18)**	Legajo	CUIL

Tipo tuple

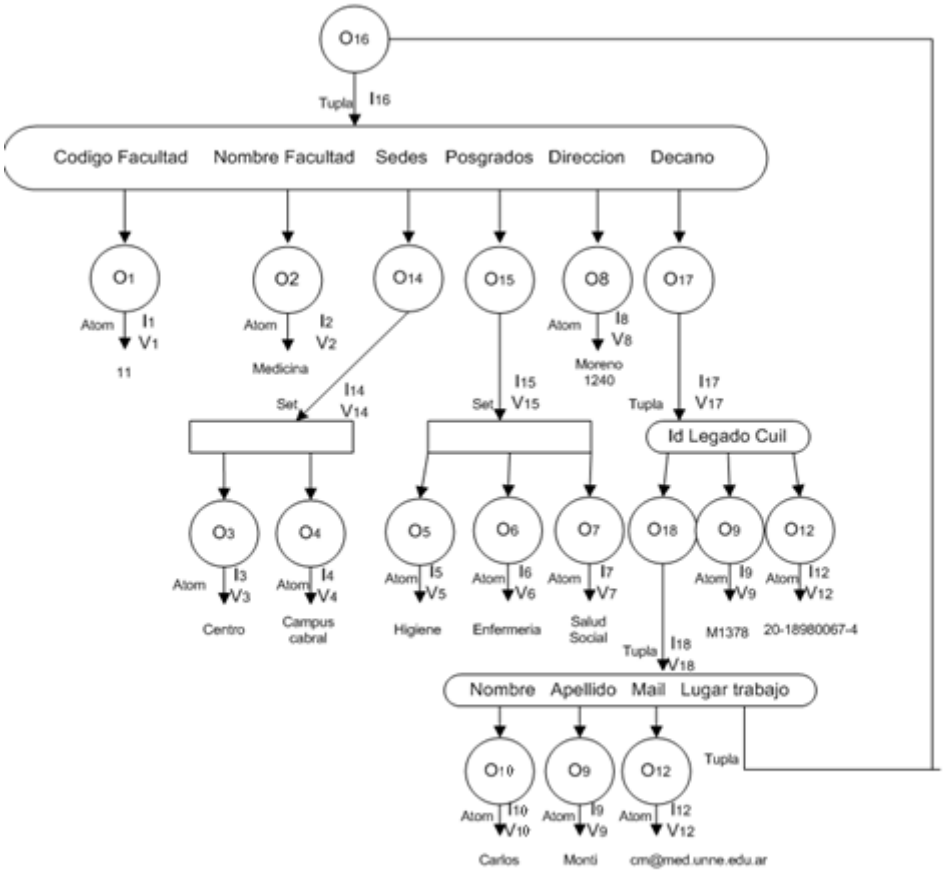
18) Empleados

Nombre	Apellido	Mail	Lugar trabajo **(16)**
			tipo tuple

a) Objeto Complejo: Facultad.

O_1 = (I_1, ATOM, 11)
O_2 = (I_2, ATOM, ”Medicina”)
O_3 = (I_3, ATOM, ”Centro”)
O_4 = (I_4, ATOM, “Campus Cabral”)
O_5 = (I_5, ATOM, “Higiene”)
O_6 = (I_6, ATOM, “Enfermería”)
O_7 = (I_7, ATOM, “Salud Social”)
O_8 = (I_8, ATOM, “Moreno 1240”)
O_9 = (I_9, ATOM, ”M1378”)
O_{10} = (I_{10}, ATOM, “Cabral”)
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, <Codigo Facultad: I_1 , Nombre Facultad: I_2, Sedes: I_8 , Posgrados: I_{15}, Dirección: I_8, Decano: I_{17}>)
O_{17} = ($I_{17,}$ TUPLE, <Id: I_{18} , Legado: I_9, Cuil: I_{12}>)
O_{18} = ($I_{18,}$ TUPLE, <Nombre: I_{10} , Apellido: I_{11}, Mail: I_{13} , Lugar trabajo: I_{16}>)

b) Representación gráfica del objeto complejo Facultad



3) Sean los datos de una concesionaria de motos:

Valores atómicos

1) CUIT: 33-19332111-5 2) Nombre Concesionaria: Ghiggeri Motos
3) Marca1: Honda 4) Marca2: Suzuki 5) Marca3: Motomel
6) Marca4: Ghiggeri 7) Marca5: Yamaha 8) Marca6: Guerrero
9) Marca7: Zanella 10) Sitio web: motos.gmmmotos.com.ar
11) Ciudad1: Resistencia 12) Ciudad2: Castelli
13) Ciudad3: Barranqueras 14) Ciudad4: Corrientes 15) Ciudad5: Formosa
16) DNI: 25334812 17) Teléfono: 3624441515
18) Mail: julioayala@gmmmotos.com.ar19) Nombre: Julio
20) Apellido: Ayala 21) Fecha Nacimiento: 25-10-1967

Conjuntos

22) Marcas = {Marca1, Marca2, Marca3, Marca4, Marca5, Marca6, Marca7}
23) Ciudades = {Ciudad1, Ciudad2, Ciudad3, Ciudad4, Ciudad5}

Tuplas

24) Concesionaria (objeto complejo)

CUIT	Nombre Concesionaria	Representante**(25)**	Marcas **(22)**	Sitio web	Ciudades**(23)**
		tipo tuple	tipo set		tipo set

25) Representantes

DNI **(26)**	Teléfono	Mail

tipo tuple

26) Personal

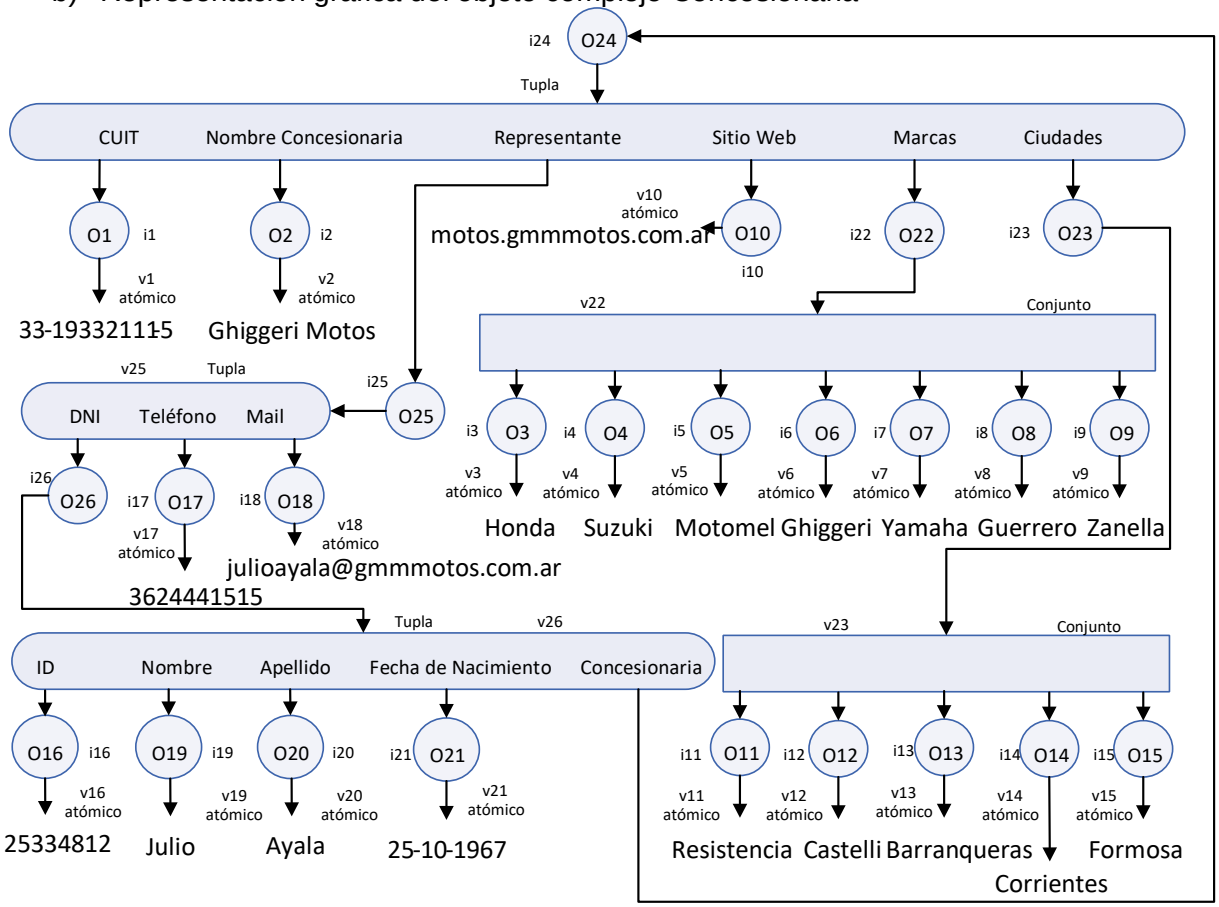
Nombre	Apellido	Fecha Nacimiento	Concesionaria**(24)**
			tipo tuple

a) Objeto complejo: Concesionaria.

O_1 = (I_1, ATOM, “33-19332111-5”)
O_2 = (I_2, ATOM, ”Ghiggeri Motos”)
O_3 = (I_3, ATOM, ”Honda”)
O_4 = (I_4, ATOM, “Suzuki”)
O_5 = (I_5, ATOM, “Motomel”)
O_6 = (I_6, ATOM, “Ghiggeri”)
O_7 = (I_7, ATOM, “Yamaha”)
O_8 = (I_8, ATOM, “Guerrero”)
O_9 = (I_9, ATOM, ”Zanella”)
O_{10} = (I_{10}, ATOM, “motos.gmmmotos.com.ar”)
O_{11} = (I_{11}, ATOM, “Resistencia”)
O_{12} = (I_{12}, ATOM, “Castelli”)
O_{13} = (I_{13}, ATOM, “Barranqueras”)
O_{14} = (I_{14}, ATOM, “Corrientes”)
O_{15} = (I_{15}, ATOM, “Formosa”)
O_{16} = (I_{16}, ATOM, 25334812)
O_{17} = (I_{17}, ATOM, 36244441315)
O_{18} = (I_{18}, ATOM, “julioayala@gmmmotos.com.ar”)
O_{19} = (I_{19}, ATOM, “Julio”)
O_{20} = (I_{20}, ATOM, “Ayala”)
O_{21} = (I_{21}, ATOM, “25-10-1967”)
O_{22} = (I_{22}, SET , { $I_3, I_4, I_5, I_6, I_7, I_8, I_9$ })
O_{23} = (I_{23}, SET , { $I_{11}, I_{12}, I_{13}, I_{14}, I_{15},$ })

O_{24} = (I_{24}, TUPLA, <Cuit: I_1 , Nombre Concesionaria: I_2, Representante: I_{25} , Marcas: I_{22}, Sitio Web: I_{10} , Ciudades:I_{23}>)
O_{25} = (I_{25}, TUPLA, <Dni:I_{16} , Teléfono:I_{17} , Mail: I_{18}>)
O_{26} = (I_{26}, TUPLA, <Nombre:I_{19} , Apellido: I_{20} , Fecha Nacimiento: I_{21} , Concesionaria: I_{24}>)

b) Representación gráfica del objeto complejo Concesionaria



4) Sean los datos del organismo, Dirección de Rentas de la provincia de Corrientes:

Valores atómicos

1) Nombre organismo: DGR_Ctes
2) Sitio web: www.dgrcorrientes.gov.ar
3) Dependencia1: Saladas
4) Dependencia2: Mercedes
5) Dependencia3: Esquina
6) Dependencia4: Alvear
7) Dni: 21324105
8) Profesión: Contador Público
9) Legajo: R-12542
10) Antigüedad: 25

Conjuntos

11)Dependencias={Dependencia1,Dependencia2,Dependencia3, Dependencia4}

Tuplas

12) Organismo (objeto complejo)

Nombre organismo	Sitio web	Dependencias **(11)**	Dirección	Director **(13)**
		tipo set		tipo tuple

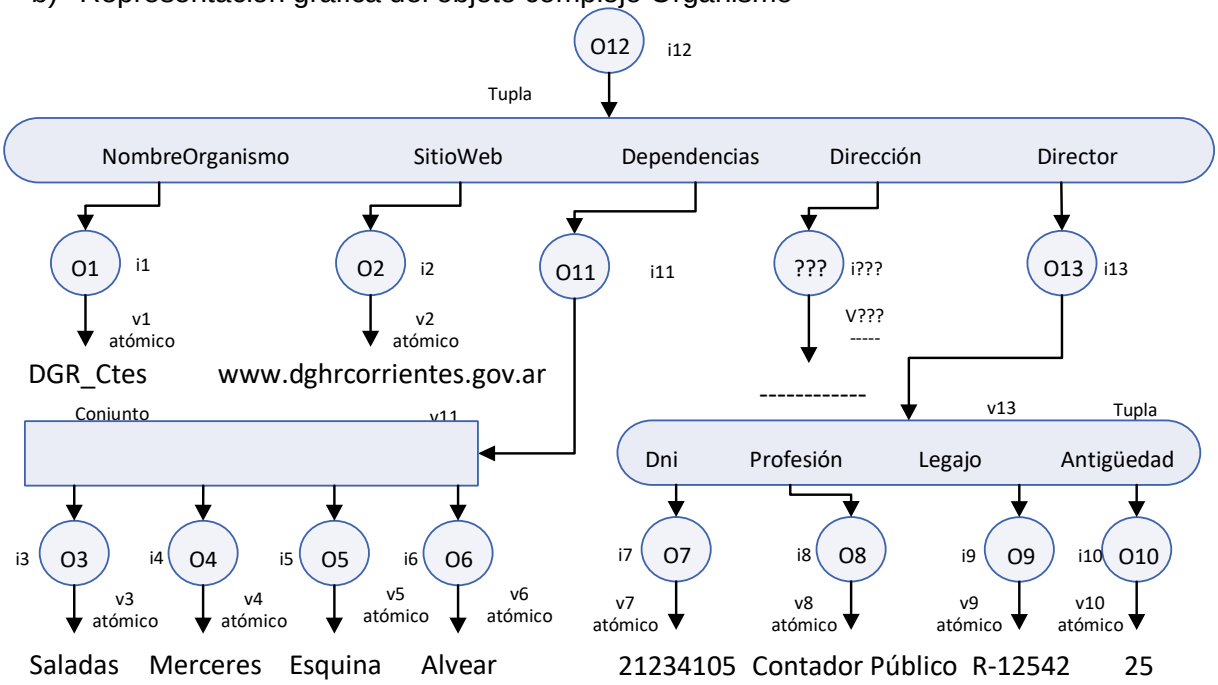
13) Director

Dni	Profesión	Legajo	Antigüedad

a) Objeto complejo: Organismo.

O_1 = (I_1, ATOM, “DGR_Cts”)
O_2 = (I_2, ATOM, ”www.dgrcorrientes.gov.ar”)
O_3 = (I_3, ATOM, ”Saladas”)
O_4 = (I_4, ATOM, “Mercedes”)
O_5 = (I_5, ATOM, “Esquina”)
O_6 = (I_6, ATOM, “Alvear”)
O_7 = (I_7, ATOM, “21324105”)
O_8 = (I_8, ATOM, “Contador Público”)
O_9 = (I_9, ATOM, ”R-12542”)
O_{10} = (I_{10}, ATOM, “25”)
O_{11} = (I_{10}, ATOM, “Av. Gdor. Pujol 2330”)
O_{12} = (I_{12}, SET, { I_3, I_4, I_5, I_6})
O_{13} = (I_{13}, TUPLA, <Nombre organismo: I_1, Sitio Web:I_2, Dependencias:I_{12}, Dirección: I_{11}, Director: I_{14} >)
O_{14} = (I_{14}, TUPLA, <Dni: I_7, Profesión: I_8, Legajo: I_9, Antigüedad: I_{10} >)

b) Representación gráfica del objeto complejo Organismo



Práctico 4- Base de datos Objeto-Relacional.

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.

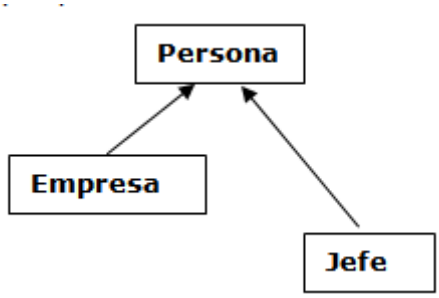
```
/*Creacion de tabla libos */
Create table Libros (
    /*Campo identidad o columna identidad Similar a OID */
    Cod_libro integer  generated always as identify
    (
        /*partes de la columna identidad*/
        start with 1 /*Valor de inicio*/
        increment by 1 /* incrementar de 1 en 1*/
        minvalue 1/*minimo valor*/
        maxvalue 5000 /*maximo valor*/
        no cycle /* indica si al alcanzar el valor maximo se
        debe re iniciar la secuencia */
    ),
    Titulo  varchar(40),
    Autor   varchar(35),
    Año char(4),
    Precio  decimal(10,2)
);
```

b) Crear la tabla Editorial, con los atributos: Código de libro, Título de la obra, Descripción, Autor, Foto de la portada del libro, Video 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

```
Create table Editorial (
    Cod_libro   integer  /*attr identidad de la tabla Libro*/
    Titulo      varchar(50), /*attr titulo de la obra*/
    Descripcion clob(40k), /*OBJETO Grandes DE CARACTERES*/
    Autor       varchar(40),
    Foto_portada    blob(2m), /*Binario Grandes*/
    Videopresen blob(1G) /*Binario Grandes 1 Gigabyte*/
);
```

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.

```
/*UDT DISTINTO persona*/
Create type persona as
(
    /*Def. de attrs del modelo*/
    Dni         char(8),
    Apeynom varchar2(50),
    Direccion   varchar2(40),
    Ciudad      char(25),
    Fechanac    date
) not final; /*puede ser derivado*/
```

Tipo Empresa: Empleado referencia a tipo persona, Mail, Código Postal, sitio web.

```
/*UDT Estructurado  empresa*/
Create type empresa as
(
    /*Def. de attrs del modelo*/
    Empleado    persona, /*Relación de
    agregacion con el tipo persona*/
    Mail        varchar2(50),
    Codpostal   integer
) not final; /*puede ser derivado*/
```

Tipo Jefe: Director referencia a tipo persona, Mail, Área, Sueldo, Antigüedad.

Definir la tabla Empleados según el tipo Empresa.

```
/*UDT Estructurado jefe*/
Create type jefe as
(
    /*Def. de attrs del modelo*/
    Director    persona, /* Relacion de
    agregacion con el UDT tipo 'persona'.
    UDT (como columna)*/
    Mail        varchar2(50),
    Area        char(2),
    Sueldo      decimal(8,2)
) final; /*no puede ser derivado*/

/* Tabla de UDTs. UDT como FILA */
Create table empleados of empresa;
```

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.

```
/*Tabla No 1FN*/
Create table Universidad (
    Nombre  varchar2(100),
    Rector      varchar2(50),
    Direccion   varchar(40),
    /*VECTOR (NO 1FN) Coleccion Ordenada
    con elementos distintos*/
    Facultdades varchar(15) array[9],
    Codpostal   integer,
    Sitio_web   varchar2(80)
);
```

Variante: definir Facultades como tipo multiconjunto.

```
/*Tabla No 1FN*/
Create table Universidad (
    Nombre  varchar2(100),
    Rector      varchar2(50),
    Direccion   varchar(40),
    /*Coleccion NO Ordenada, con
    elementos no necesariamente distintos*/
    Facultdades varchar(15) MULTISET,
    Codpostal   integer,
    Sitio_web   varchar2(80)
);
```

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

```
INSERT into universidad
(/*Attributos*/
    nombre, rector, direccion, facultades,
    codpostal, sitio_web
)
VALUES
(/*Valores*/
    'Universidad Nacional del Nordeste',
    'Maria Veirave',
    '25 de Mayo 868',
    [/*Columna multivaluada NO 1FN*/
        'Medicina', 'Humanidades',
        'Exactas', 'Veterinaria',
        'Económicas', 'Agrarias',
        'Arquitectura', 'Derecho',
        'Odontología'
    ],
    3400, 'www.unne.edu.ar'
);
```

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

```
INSERT IN TO universidad
    (/*Atributos*/
        nombre, rector, direccion, facultades,
        codpostal, sitio_web
    )
VALUES
    (/*Valores*/
        'Universidad Nacional del Nordeste',
        'Maria Veirave',
        '25 de Mayo 868',
        Multiset /*Columna multivaluada NO 1FN*/
        [
            'Medicina', 'Humanidades', 'Exactas',
            'Veterinaria', 'Económicas',
            'Agrarias', 'Arquitectura',
            'Derecho', 'Odontología'
        ],
        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.

```
SELECT
    nombre, rector,
    facultades[1], facultades[2]
FROM facultades
```

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

```
Create table Banco (
        Identificación ROW(
            NumBCRA integer,
            CUIT varchar(11)
        ),
        Razón social ROW(
            NombreComercial varchar2(100),
            TipoEmpresa integer,
            CondiciónTributaria varchar(50)
        ),
        Presidente ROW(
            Nombres varchar(50),
            Apellido varchar(50),
            DNI integer,
            Mail varchar(320),
            Dirección varchar(250),
            Calle varchar(100),
            Número integer,
            Ciudadvarchar(100),
            CódigoPostal integer
        ),
        Teléfono ROW(
            Prefijo integer,
            Número 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.

```
Create type Empleado as (
      Dni char(8),
      Apeynom varchar(50),
      Antigüedad integer,
      Direccion varchar(40),
      Cargo varchar(15),
      Sueldo decimal (10,2)
);
```

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

Antigüedad laboral se obtiene como: Año actual – Año ingreso

```
Create method AntiLaboral(…argumentos…) for empleado
Begin
        Return(año_actual - año_ingreso);
    End;
```

Sueldo: Básico + Adicional por título + Escolaridad – Aporte Jubilatorio

```
Create method Sueldo(…argumentos…) for empleado
    Begin
     Return((básico+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

```
Create type Producto as (
      CodProducto integer,
      Denominación varchar(100),
      StockActual integer,
      StockMinimo integer,
      PrecioFabrica decimal(10,2),
      PrecioConsumidor decimal(10,2)
);
```

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

```
Create method PreciosConsumidor(…argumentos…) for Producto
    Begin
        Return(PrecioFabrica+(PrecioFabrica/100*15));
End;
```