Resumen Ejercicios:

Bases de Datos Avanzadas

Calcular el numero de accesos para resolver la siguiente consulta

```
SELECT VENDEDORES.nombre, CLIENTES.nombre
FROM CLIENTES, VENDEDORES, FACTURAS, VENTAS, PRODUCTOS
WHERE CLIENTES.rut = FACTURAS.clientes
 AND VENDEDORES.rut = FACTURAS.vendedor
 AND FACTURAS.numero = VENTAS.factura
 AND VENTAS.producto = PRODUCTOS.codigo
 AND PRODUCTOS.precio = (SELECT MAX (precio) FROM PRODUCTOS)
```

Indices: FACTURAS.numero, CLIENTES.rut, CLIENTES.ciudad,
VENDEDORES.sucursal, PRODUCTOS.codigo, PRODUCTOS.nombre,
VENTAS.(producto, factura)

* 50.000 Clientes, 10.000 Vendedores, 1.000.000 Facturas, 1.000 Productos, 5.000.000 Ventas
* Los productos tienen distinto precio y su venta sigue una distribución uniforme
* En promedio, a los clientes se les emite cantidades similares de facturas.
* En promedio, los vendedores realizan números similares de ventas
* Orden del árbol = 100, filas por bloque = 500

* * *

1. Dado el siguiente modelo de datos,

```
LIBROS: Codigo, Nombre, Autor, Editorial, Valor_unitario
BIBLIOTECAS: Cod_biblio, Cod_libro, Copias, Ciudad
PRESTAMOS: Rut_lector, Cod_libro, Cod_biblio, Fecha_prestamo, Fecha_devol
LECTORES: Rut, Nombre, Ciudad

Supuestos:
- Una ciudad puede tener más de una biblioteca
- Un lector no puede pedir más de un libro diario
- En promedio, un libro es solicitado por 5 lectores
- Para el 1% de los libros hay una sola copia disponible
- Los libros están disponibles en 3 bibliotecas en promedio
```

calcular el número de accesos requeridos para responder a la siguiente consulta

```
SELECT Lectores.ciudad, Libros,nombre
FROM Lectores, Prestamos, Bibliotecas, Libros
WHERE Bibliotecas.copias = 1
 AND  Bibliotecas.cod_biblio = Prestamos.cod_biblio
 AND  Prestamos.rut_lector = Lectores.rut
 AND  Bibliotecas.cod_libro = Libros.codigo
```

* Indices: LIBROS.codigo, BIBLIOTECAS.(cod_biblio, cod_libro),
PRESTAMOS.(cod_biblio, cod_libro, rut_lector), LECTORES.rut
* 200.000 Libros, 2.000.000 Préstamos, 20.000 Lectores
* Orden del árbol: 10 Filas por bloque: 50

EJEMPLOS CONSULTAS:

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

1. Que espectadores se encuentran observando la banda Imagine Dragons.

```
SELECT espectadores.nombre
FROM espectadores, escenarios, bandas
WHERE espectadores.nombre_escenario = escenarios.nombre
AND escenarios.id = bandas.id_escenario
AND banda = 'Imagine Dragons';
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

2. Nombre y edad de artistas que pertenezcan a la banda Imagine Dragons.

```
SELECT artistas.nombre, artistas.edad
FROM artistas, bandas
WHERE bandas.nombre=artistas.nombre_banda
AND bandas.nombre = 'Imagine Dragons';
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

3. Nombre y género de las bandas favoritas de Daniela. (por consecuencia)

```
SELECT bandas.nombre, bandas.genero
FROM bandas, artistas, favorito,espectadores
WHERE bandas.nombre = artistas.nombre_banda
AND artistas.nombre = favorito.nombre_artista
AND favorito.rut_espectador = espectadores.rut
AND espectadores.nombre = 'Daniela';
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

4. Nombre de las bandas que están tocando en el escenario Poca Luz.

```
SELECT bandas.nombre
FROM bandas,escenarios
WHERE bandas.nombre_escenario=escenarios.nombre
AND escenarios.nombre = 'Poca Luz';
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

1. Nombre de los espectadores cuya edad sea la mayor entre todos los espectadores.

```
SELECT nombre
FROM espectadores
WHERE edad=(SELECT MAX(edad) FROM  espectadores);
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

2. Obtener la lista de los nombres de los escenarios en orden alfabético.

```
SELECT escenarios.nombre
FROM escenarios
ORDER BY escenarios.nombre;
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción
ARTISTAS: nombre, edad, pais, residencia, nombre_banda
BANDAS:nombre, genero, ciudad_origen, id_escenario.
ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.
FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

3. Promedio de las edades de los artistas pertenecientes a Italia.

```
SELECT AVG(edad)
FROM artistas
WHERE pais='Italia';
```

Ejercicio: Lollapelusa

ESCENARIOS : id, nombre, cantidad_participantes, descripción

ARTISTAS: nombre, edad, pais, residencia, nombre_banda

BANDAS:nombre, genero, ciudad_origen, id_escenario.

ESPECTADORES: rut, nombre, edad, mail, nombre_escenario.

FAVORITO: rut_espectador, nombre_artista.

Consultas a realizar:

4.Cuántos integrantes conforman cada banda inscrita en el festival.

```
SELECT artistas.nombre_banda,COUNT(*) AS Num_Integrantes
FROM artistas
GROUP BY artista.nombre_banda;
```

Ejercicio: Tienda

EMPLEADOS : rut, nombre, cargo, rut_jefe, sueldo, comisión, numdep
DEPTOS: numdep, nombre, ciudad
GRADOS: grado, sueldo_inf, sueldo_sup
PRODUCTO: codigo, nombre,precio.
CLIENTES: rut, nombre, comuna
VENTAS: num_venta, monto, fecha, rut_vende, rut_cliente
VENTAS_DETALLE: num_venta, cod_producto, cantidad

Consultas a realizar:

14. Total pagado por Pepe por *todas las compras* que ha hecho

```
SELECT SUM(ventas.monto)
FROM clientes, ventas
WHERE ventas.rut_cliente=clientes.rut
AND clientes.nombre='Pepe';
```

Ejercicio: Tienda

EMPLEADOS : rut, nombre, cargo, rut_jefe, sueldo, comisión, numdep
DEPTOS: numdep, nombre, ciudad
GRADOS: grado, sueldo_inf, sueldo_sup
PRODUCTO: codigo, nombre,precio.
CLIENTES: rut, nombre, comuna
VENTAS: num_venta, monto, fecha, rut_vende, rut_cliente
VENTAS_DETALLE: num_venta, cod_producto, cantidad

Consultas a realizar:

15. MONTO total de comisiones ganadas por ALLEN.

```
SELECT SUM(ventas.monto*(empleados.comision/100)) FROM empleados,ventas WHERE
ventas.rut_vende=empleados.rut AND empleados.nombre='ALLEN';
```

EJERCICIOS BDDA:

1:

Calcular el tiempo requerido para resolver la siguiente consulta

```
SELECT PRODUCTOS.cod, PRODUCTOS.nombre FROM
PRODUCTOS, CLIENTES, VENTAS WHERE
CLIENTES.nombre = "Juan Carlos" AND
CLIENTES.rut = VENTAS.rut_cliente AND
VENTAS.producto = PRODUCTOS.cod
```

Supuestos:
- Indices: CLIENTES.rut, CLIENTES.ciudad, VENDEDORES.sucursal, PRODUCTOS.cod, VENTAS.(producto, factura)
- 50.000 Clientes, 10.000 Vendedores, 1.000 Productos, 5.000.000 Ventas
- Todos los productos tienen distinto precio y su venta sigue una distribución uniforme.
- En promedio, a los clientes se les emite cantidades similares de compras.
- Los clientes no pueden tener el mismo nombre.
- Orden del árbol = 100, fb = 600.
- Considere 12 ms de tiempo de acceso.

PLAN DE EJECUCIÓN:

1. Buscar en la tabla clientes el nombre "Juan Carlos" y obtener el rut.

2. Para cada rut encontrado en (1), buscar en ventas ese cliente y obtener el producto

3. Para cada producto encontrado en (2), buscar en producto ese codigo y obtener su nombre.

CALCULO DE ACCESOS:

N1 = 50.000 / 600 = 84 accesos.

N2 = 5.000.000 / 600 = 8.334 accesos.

$N3 = 100 \cdot (\log_{100} 1.000 + 1) = 300$ accesos

NT = 84 · 8334 · 300 = 8718 accesos.

TT = 8718 · 12 ms = 104616 ms.

2.

Calcular el tiempo requerido para resolver la siguiente consulta

```
SELECT CLIENTES.nombre, DESPACHOS.comuna
FROM CLIENTES, DESPACHOS, FACTURAS, VENTAS,
PRODUCTOS WHERE CLIENTES.rut = FACTURAS.clientes AND
DESPACHOS.id = FACTURAS.despacho AND
FACTURAS.numero = VENTAS.factura AND
VENTAS.producto = PRODUCTOS.codigo AND
PRODUCTOS.precio = (SELECT MIN (precio) FROM
PRODUCTOS)
```

Supuestos:

- Indices: FACTURAS.numero, CLIENTES.rut, CLIENTES.ciudad, VENDEDORES.sucursal, PRODUCTOS.codigo, PRODUCTOS.nombre, VENTAS.(producto, factura)
- 50.000 Clientes, 10.000 Vendedores, 1.000.000 Facturas, 1.000 Productos, 5.000.000 Ventas, 4.000.000 Despachos
- Todos los productos tienen distinto precio y su venta sigue una distribución uniforme.
- En promedio, a los clientes se les emite cantidades similares de facturas.
- En promedio, los vendedores realizan números similares de ventas.
- Orden del árbol = 100, fb = 500
- El tiempo de acceso es de 12 ms

PLAN DE EJECUCIÓN:

1. Buscar en productos, el producto con el precio mínimo. ⟶ 1 producto.

2. Para cada precio obtenido buscar en productos ese precio y obtener el codigo del producto. ⟶ 1 codigo producto.

3. Para cada codigo obtenido en (2), buscar en ventas ese producto y obtener factura ⟶ 1 · (5.000.000/1000) = 5.000

4. Para cada factura obtenida en (3), buscar en facturas ese numero y obtener despacho y clientes. ⟶ 5.000 · 1 = 5000

5. Para cada despacho obtenido en (4) buscar el despacho en despachos y obtener comuna. ⟶ 5000 · (1) = 5.000

6. Para cada cliente encontrado en (4), buscar en cliente y obtener nombre y nit. ⟶ 5000 · 1 = 5000

CANTIDAD DE ACCESOS:

N1 = 1000/500 = 2 accesos

N2 = 1 · (1000/500) = 2 accesos

N3 = 1 · ($\log_{100}$ 1.000.000 + 4999/100 + 0)
N3 = 54 accesos

N4 = 5000 · ($\log_{100}$ 1.000.000 + 1)
N4 = 20.000 accesos

N5 = 4.000.000/500 = 8.000 acces

N6 = 5000 ($\log_{100}$ 50.000 + 1)
N6 = 20.000 accesos.

NT = 2 + 2 + 54 + 20.000 + 8.000 + 20.000
NT = 48.058 accesos
TT = 48.058 · 12 ms = 9,6 s.

3.

1. *(20 puntos)* Calcule el total de accesos requeridos para resolver la siguiente consulta.

```
SELECT Clientes.nombre, Productos.nombre
FROM Productos, Clientes, Ventas
WHERE Clientes.ciudad = 'Arica'
  AND  Ventas.cod_producto = Productos.codigo
  AND  Ventas.rut_cliente = Clientes.rut

Volumen: 10.000 clientes, 150 productos, 6.000.000 ventas
Indices: clientes.rut, productos.codigo, ventas.rut_cliente
Filas por bloque: 50
Orden del árbol:  100

* Las ventas se distribuyen uniformemente entre todos los clientes
* En cada venta se vende un solo producto hasta un máximo de 5 unidades
* Los clientes están distribuidos uniformemente entre las 100 ciudades en las que atiende la empresa
* Los precios de los productos están uniformemente distribuidos de $1.000 en $1.000 en el rango $1.000 a $15.000
```

PLAN DE EJECUCIÓN:

1. Buscar en clientes la ciudad "Arica" y obtener el rut y el nombre.
   ⟶ 10.000 / 100 = 100

2. Con el rut de (1), buscar en ventas el rut_cliente y obtener cod_producto.
   ⟶ 100 · (6000000 / 10000) = 60000
   100 · 600

3. Con el codigo obtenido en (2), buscar en productos el codigo y obtener el nombre del producto.
   ⟶ 60.000 · 1 = 60.000

CANTIDAD DE ACCESOS:

duplicados: número obtenido − 1 + puede orden del árbol

N1 = 10000 / 50 = 200 accesos
N2 = 100 · ($\log_{100}$ 6000000 + 599/100 + 600) = 61.000
N3 = 60.000 · ($\log_{100}$ 150 + 1) = 180.000
NT = (200 + 61.000 + 180.000 = 241.200 accesos
TT = 10 ms · 241.200 = 2.412.000 ms.

4.

CALCULAR EL TIEMPO REQUERIDO PARA RESOLVER LA SIGUIENTE CONSULTA

Clientes: rut, nombre, dirección, rut_vendedor

Vendedores: rut, nombre, departamento

```
SELECT Vendedores.nombre FROM
Clientes, Vendedores WHERE
Clientes.nombre = "Jose Luis Artigas" AND
Clientes.rut_vendedor = Vendedores.rut
```

Supuestos:

Índices: Clientes.rut, Vendedores.rut

Hay 600.000 Clientes y 15.000 vendedores

Fb = 20, orden del árbol = 10

Tiempo promedio de Acceso: 12 ms

Puede haber hasta 4 clientes con el mismo nombre

PLAN DE EJECUCIÓN:

1. Buscar en clientes el nombre "JLA" y obtener el rut_vendedor.

   ⟶ 1 * 4 = 4 clientes

2. Para cada uno de los ruts_vendedor obtenidos en (1), buscar el rut en vendedores y obtener el nombre del vendedor.

   ⟶ 4 · 1 = 4 ruts

N1 = 600.000 / 20 = 30.000 accesos

N2 = 4 · ($\log_{10}$ 15.000 + 1) = 21 accesos

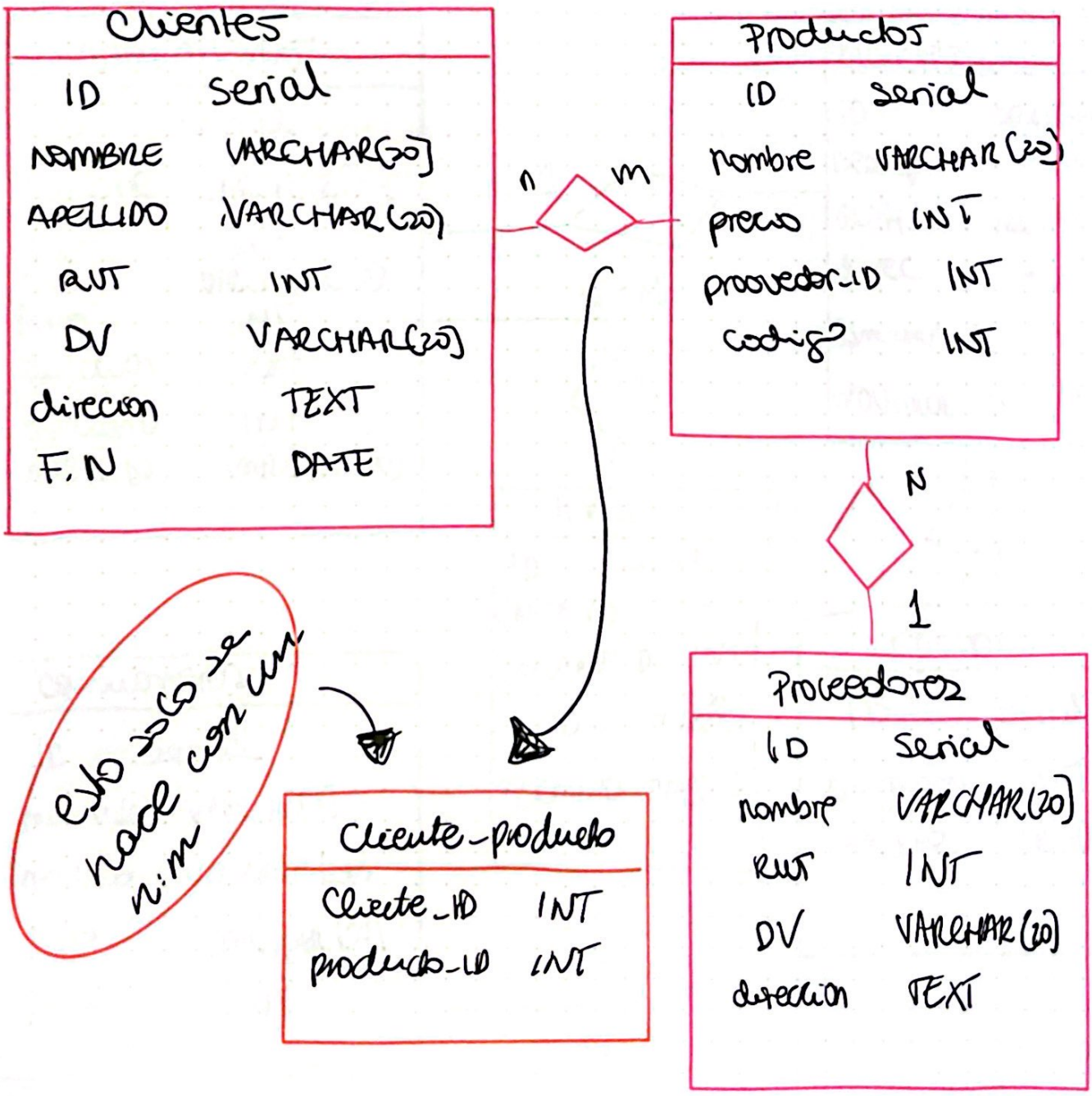
NT = 30.000 + 21 = 30.021 accesos

TT = 30.021 · 0,012 s = 360 s.

Ejercicio

A partir del siguiente enunciado se desea realiza el modelo entidad-relación.
"Una empresa vende productos a varios clientes. Se necesita conocer los datos personales de los clientes (nombre, apellidos, rut, dirección y fecha de nacimiento). Cada producto tiene un nombre y un código, así como un precio unitario. Un cliente puede comprar varios productos a la empresa, y un mismo producto puede ser comprado por varios clientes.

Los productos son suministrados por diferentes proveedores. Se debe tener en cuenta que un producto sólo puede ser suministrado por un proveedor, y que un proveedor puede suministrar diferentes productos. De cada proveedor se desea conocer el rut, nombre y dirección".

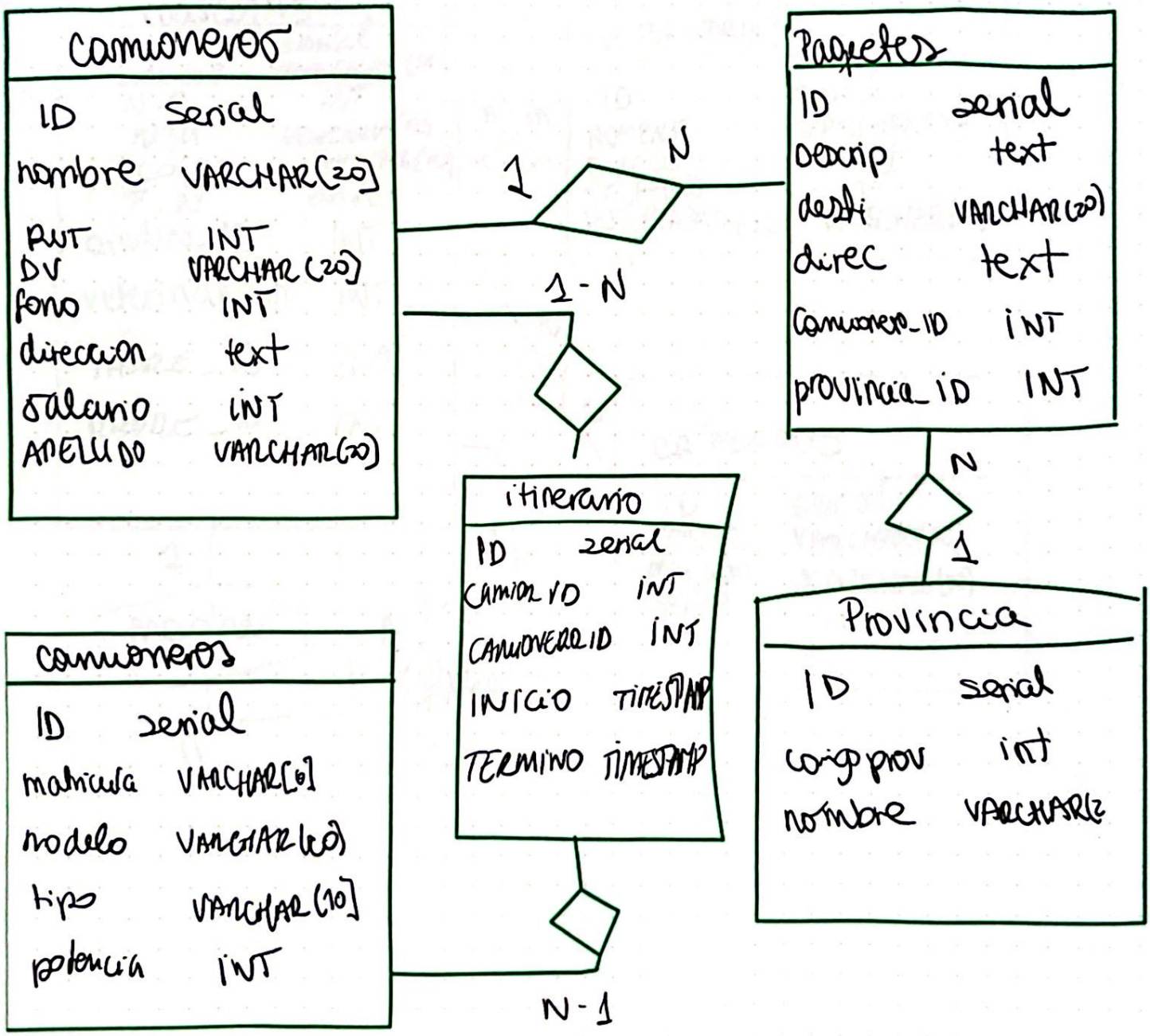


Ejercicio

A partir del siguiente enunciado se desea realizar el modelo entidad-relación. "Se desea informatizar la gestión de una empresa de transportes que reparte paquetes por toda Chile. Los encargados de llevar los paquetes son los camioneros, de los que se quiere guardar el rut, nombre, teléfono, dirección, salario. De los paquetes transportados interesa conocer el código de paquete, descripción, destinatario y dirección del destinatario. Un camionero distribuye muchos paquetes, y un paquete sólo puede ser distribuido por un camionero.

De las provincias a las que llegan los paquetes interesa guardar el código de provincia y el nombre. Un paquete sólo puede llegar a una provincia. Sin embargo, a una provincia pueden llegar varios paquetes.

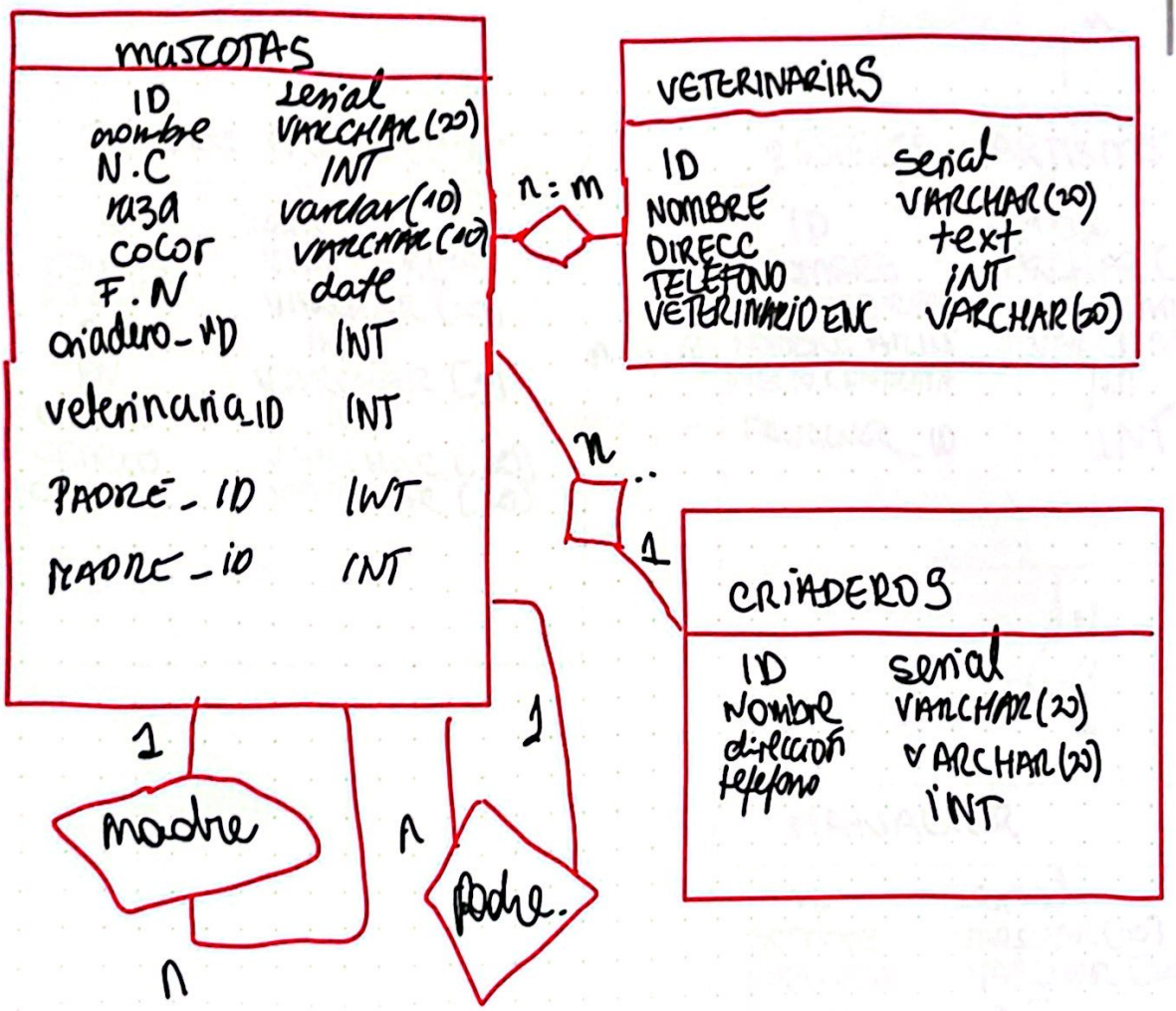
De los camiones que llevan los camioneros, interesa conocer la matrícula, modelo, tipo y potencia. Un camionero puede conducir diferentes camiones en fechas diferentes, y un camión puede ser conducido por varios camioneros".



Ejercicio

A partir del siguiente enunciado se desea realizar el modelo entidad relación. "Con la llegada de la ley cholito se necesita saber de una mascota(Perros) de que criadero proviene, en que veterinaria se atiende y quienes son sus familiares (Padre, Madre,Hermanos,Hermanas,Abuelos, Abuelas, Hijos y Hijas)

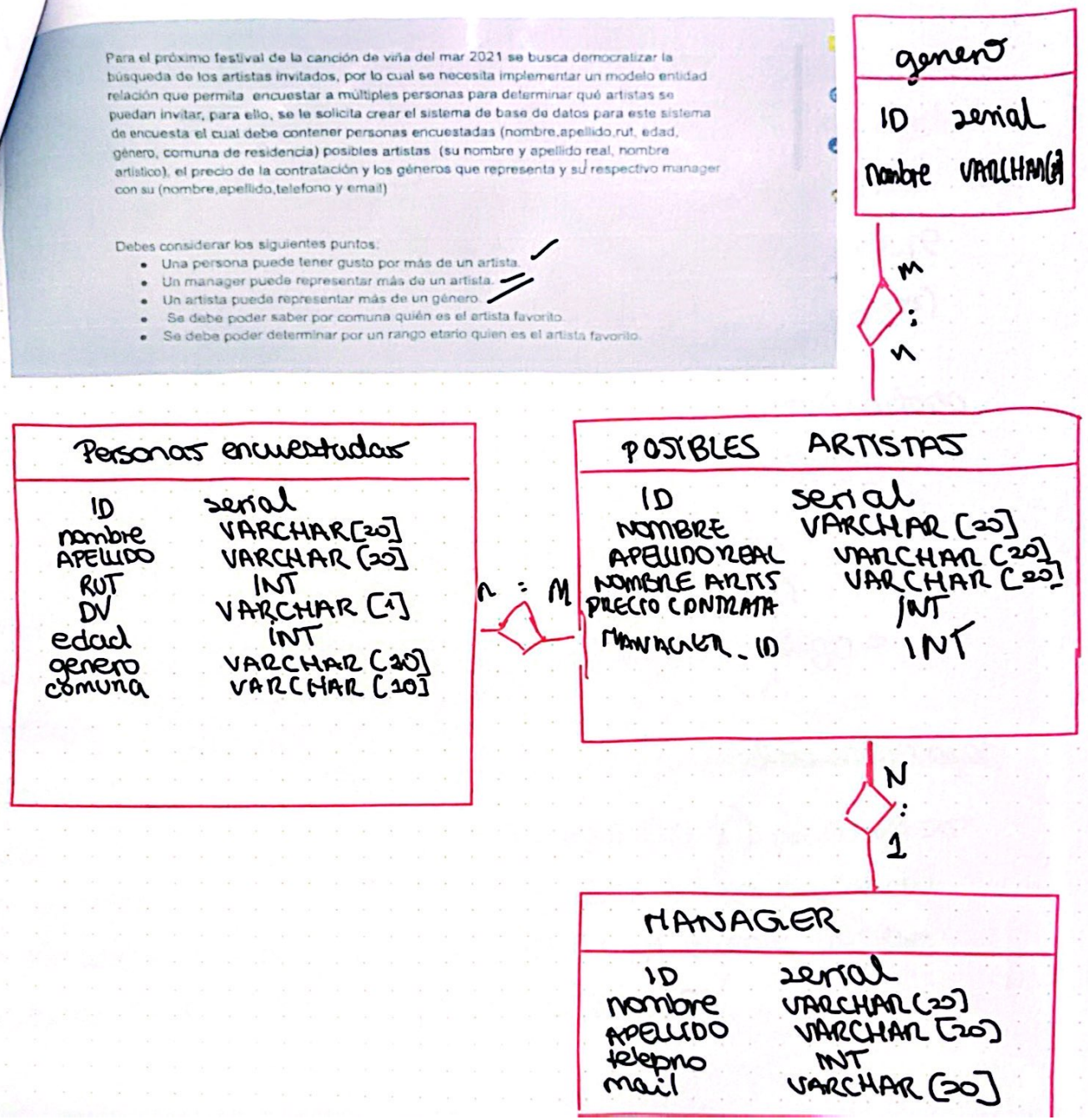
Considere que una mascota puede ser atendida por más de una veterinaria, una veterinaria tiene Nombre, Dirección, Teléfono,Veterinario Encargado, un criadero contiene Nombre, Direccion, Telefono y por último una mascota contiene los atributos nombre, numero de chip, raza, color y fecha de nacimiento."

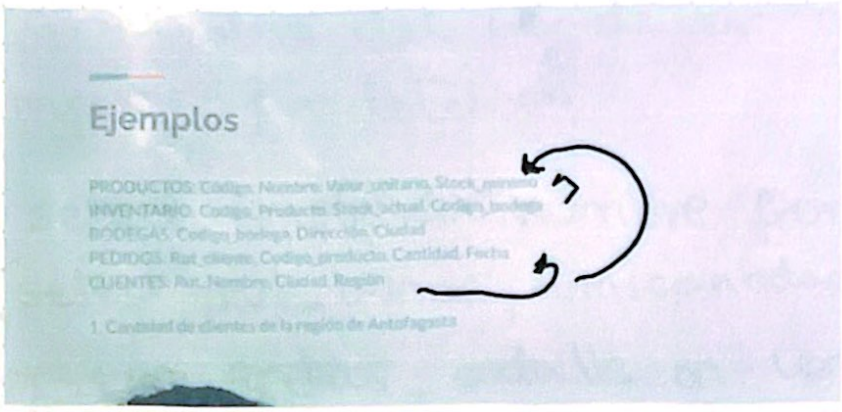


Para el próximo festival de la canción de viña del mar 2021 se busca democratizar la búsqueda de los artistas invitados, por lo cual se necesita implementar un modelo entidad relación que permita encuestar a múltiples personas para determinar qué artistas se puedan invitar, para ello, se le solicita crear el sistema de base de datos para este sistema de encuesta el cual debe contener personas encuestadas (nombre,apellido,rut, edad, género, comuna de residencia) posibles artistas (su nombre y apellido real, nombre artístico), el precio de la contratación y los géneros que representa y su respectivo manager con su (nombre,apellido,telefono y email)

Debes considerar los siguientes puntos:

- Una persona puede tener gusto por más de un artista.
- Un manager puede representar más de un artista.
- Un artista puede representar más de un género.
- Se debe poder saber por comuna quién es el artista favorito.
- Se debe poder determinar por un rango etario quien es el artista favorito.





1. Cantidad de clientes de la region de antofagasta.

R: select count(*) from clientes where region='Antofagasta'

2- cantidad promedio de los pedidos del 21/08/2017

R: select avg (cantidad, pedidos) from pedidos where fecha = '21/08/2017'

3. nombre del producto mas barato.

R: Select nombre from productos where valor_unitario (unidades)
= (select min valor_unitario) from productos.

4. valorizacion total del inventario.

R: select sum(valor unitario * stock actual) from productos join inventario on productos.codigo = inventario.codigo_producto

5. Monto total comprado por el cliente pedro soto mayor.

select sum (valor unitario * cantidad) productos join pedidos on producto.codigo = pedido.codigo_producto join clientes on clientes.rut = pedido.rut_cliente where clientes.nombre = 'pedro soto mayor'

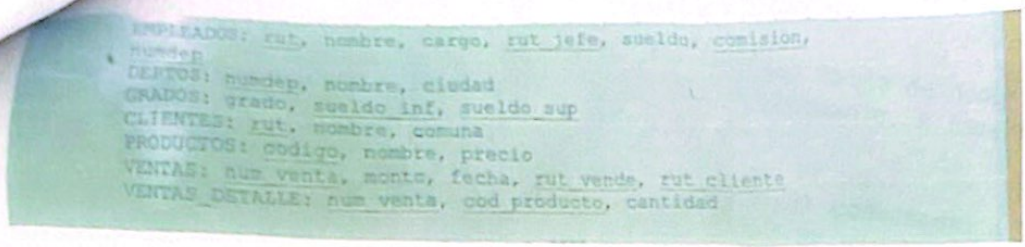
6= Nombre del vendedor que ha vendido mas productos.

```
select empleados.nombre from empleados join
(select nt_vende, sum(cantidad) from ventas
join ventas_detalle on ventas_detalle.num_venta
= VENTAS.num_venta group by nt_vende) as
empleados_cantidad.nt_vende = empleados.nt
join (                    → retorna un max
select max(sum) from (
select nt_vende, sum(cantidad) from ventas
join ventas_detalle on ventas_detalle.num_venta
= VENTAS.num_venta group by nt_vende) as tab )
as maximo on maximo.max = empleados_cantidad.sum.
```

7. Nombre del vendedor que ha vendido menos productos.

```
select empleados.nombre from empleados join
(select nt_vende, sum(cantidad) from ventas
join ventas_detalle on ventas_detalle.num_venta
= VENTAS.num_venta group by nt_vende) as
empleados_cantidad.nt_vende = empleados.nt
join (
select min(sum) from (
select nt_vende, sum(cantidad) from ventas
join ventas_detalle on ventas_detalle.num_venta
= VENTAS.num_venta group by nt_vende) as tab )
as maximo on maximo.min = empleados_cantidad.sum.
```

problemas típicos que debemos



1. numero de productos que valen mas de 200.

select count(*) from productos where productos.precio > 200

2. monto total recaudado por las ventas hechas.

select sum(monto * cantidad) from ventas join

3. Nombre del jefe de SCOTT.

select nombre from empleados where rut = (select rut_jefe from empleados where nombre = 'SCOTT');

4. Nombre y precio del producto mas caro.

select nombre, precio from productos where precio = (select max(precio) from productos where precio < 400).

5. nombre y sueldo del empleado de NEW YORK que tiene el peor sueldo.

Select empleados.nombre, sueldo, ciudad from empleados join deptos on deptos_numdep = empleados_numdep where ciudad = 'NEW YORK' and sueldo = (select min(sueldo) from empleados join deptos on deptos.numdep = empleados.numdep where ciudad = 'NEW YORK')

CONCURRENCIA:

lo de los problemas típicos que debemos enfrentar en bases de datos de gran volumen es el acceso ncurrente de múltiples usuarios a la misma información almacenada en ella. Ello produce contención el acceso a los datos, lo que se traduce en tiempos de respuesta cada vez mas largos. En esta tividad revisaremos los problemas que pueden ocurrir al haber acceso concurrente a la base de tos.

ra preparar el escenario, se debe crear una tabla de nombre DATOS que tenga dos atributos lentificados como CODIGO y NOMBRE, respectivamente, e insertar 10 filas con CODIGOs del 1 al 10 y MBREs cualesquiera a vuestra elección.

licionalmente, debe tener 2 consolas (C1 y C2) conectadas a la BdD. Ambas consolas deben estar sibles en todo momento para que pueda observar simultáneamente lo que esta ocurriendo en ellas.

TA: En esta actividad es de vital importancia que siga los pasos tal como están indicados. alquier modificación a la secuencia indicada no va a producir los resultados esperados. Ademas, be estar observando la actividad de ambas consolas cuando haga alguna operación en una de ellas.

- En C1, actualizar el NOMBRE de la fila de CODIGO 5.
- En C2, actualizar a un NOMBRE diferente al anterior el NOMBRE de la fila de CODIGO 5.
- En C1, consultar el NOMBRE de la fila de CODIGO 5.
- En C2, consultar el NOMBRE de la fila de CODIGO 5.

Registrar cuidadosamente lo ocurrido en ambas consolas.

- En C1, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 3. A ntinuación, consultar el NOMBRE de la fila de CODIGO 3.
- En C2, abrir una transacción (BEGIN) y consultar el NOMBRE de la fila de CODIGO 3.
- En C1, cerrar la transacción con COMMIT.
- En C2, volver a consultar el NOMBRE de la fila de CODIGO 3 y cerrar la transacción con COMMIT.

Registrar cuidadosamente lo ocurrido en ambas consolas.

- En C1, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 7.
- En C2, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 7 a un NOMBRE ferente.
- En C1, volver a actualizar el NOMBRE de la fila de CODIGO 7 a un NOMBRE diferente a los anteriore cerrar la transacción con COMMIT.
- En C2, cerrar la transacción con COMMIT.
- En C1, consultar el NOMBRE de la fila de CODIGO 7.
- En C2, consultar el NOMBRE de la fila de CODIGO 7.

Registrar cuidadosamente lo ocurrido en ambas consolas.

- En C1, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 4.
- En C2, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 6.
- En C1, consultar por el NOMBRE de la fila de CODIGO 6.
- En C2, consultar por el NOMBRE de la fila de CODIGO 4.
- En C1, cerrar la transacción con COMMIT.
- En C2, cerrar la transacción con COMMIT.
- En C1, consultar por el NOMBRE de la fila de CODIGO 6.
- En C2, consultar por el NOMBRE de la fila de CODIGO 4.

Registrar cuidadosamente lo ocurrido en ambas consolas.

- En C1, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 2.
- En C2, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 8.
- En C1, actualizar el NOMBRE de la fila de CODIGO 8 a un NOMBRE diferente a los anteriores.
- En C2, actualizar el NOMBRE de la fila de CODIGO 2 a un NOMBRE diferente a los anteriores.
- En C1, cerrar la transacción con COMMIT.

.1, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 1.
, C2, abrir una transacción (BEGIN) y actualizar el NOMBRE de la fila de CODIGO 1 a un NOMBRE
erente.
- En C1, actualizar el NOMBRE de la fila de CODIGO 9 a un NOMBRE diferente a los anteriores.
- En C2, actualizar el NOMBRE de la fila de CODIGO 9 a un NOMBRE diferente a los anteriores.
- En C1, cerrar la transacción con COMMIT.
- En C2, cerrar la transacción con COMMIT.
- En C1, consultar por el NOMBRE de la fila de CODIGO 9.
- En C2, consultar por el NOMBRE de la fila de CODIGO 1.

Registrar cuidadosamente lo ocurrido en ambas consolas.

(X no se hizo)

- En C1, abrir una transacción (BEGIN) y consultar el NOMBRE de la fila de CODIGO 10 usando la
ción de actualización
(SELECT ... FOR UPDATE).
- En C2, consultar normalmente por el NOMBRE de la fila de CODIGO 10 y actualizarlo a un NOMBRE
.ferente.
- En C1, consultar normalmente por el NOMBRE de la fila de CODIGO 10 y actualizarlo a un NOMBRE
.ferente a los anteriores.
- En C2, consultar normalmente por el NOMBRE de la fila de CODIGO 10.
- En C1, cerrar la transacción con COMMIT.
- En C1, consultar normalmente por el NOMBRE de la fila de CODIGO 10.
- En C2, consultar normalmente por el NOMBRE de la fila de CODIGO 10.

Registrar cuidadosamente lo ocurrido en ambas consolas.

icer un análisis critico de lo observado en cada uno de los pasos de esta actividad, señalando lo
ie se pretendía lograr en cada una de ellas, que fue lo que se observo y explicando por que ocurrió
) que ocurrió.

1:
- Al realizar el paso uno se perdio, ya que hubo otro usuario que actualizo lo mismo en otra ventana.
por ende, primer error de concurrencia se pierden actualizaciones.

2:
- cuando se abre una transaccion se aisla lo que el usuario hizo, por ende, en la base de datos se ve lo que es efectivo.
entonces si se realiza un commit se actualiza el dato recien ya que la transacaion se cerro mediante un COMMIT., abrir transacciones para no afectar lo sucedido mientras se actualiza o borra algo de la BBD.

3:
- si se realizan dos transacciones, es decir, dos usuarios que hacen exactamente lo mismo, el sistema bloquea la fila que quiere mientras se termine la primera transaccion.
Es decir dos transacciones, en donde se realiza lo mismo, una se bloquea hasta que la primera termine la transacción. Cuando se cierra la transaccion actualiza la segunda transaccion quedando como valor la de la transaccion cerrada de ultima.

5:
- deadlock ya que se necesitan que ambos avancen y uno suelte el siguiente para que el otro pueda soltar el otro, por ende es un bloqueo mutuo. matando a quien produce el bloqueo.
- ademas mantiene bloqueo hasta que la persona se de cuenta y cierre la transaccion.

7-

• select for update es avisar que se hara una actualizacion en una transaccion haciendo que se bloquee para la otra persona.
si se hace fuera de una transaccion no bloquea la situacion.
si se levanta el bloqueo la otra actualizacion funcionara correctamente.

• se produce cuando colas si existen muchos requerimientos de usuario.

• Cuando existen muchos usuarios intentando acceder, se esfuerza la situacion para poder contenerla.

1- Se tiene en funcionamiento un sistema de base de datos con 20 tablas y sus correspondientes llave primarias. Debido a una alta actividad de inserción de datos, la base de datos va ocupando mas y mas espacio de almacenamiento en el disco en el que reside. Según las proyecciones que hace el administrador de la base de datos, en un corto plazo se consumirá todo el espacio de almacenamiento disponible. Se pide analizar dos soluciones que posibilitan resolver esta situación.

* incrementar el espacio disponible para la BdD (disco mas grande o borrar archivos no BdD en el disco) => escalabilidad vertical
* adicionar nuevo disco, definir tablespace y mover llaves primarias e indices al tablespace => escalabilidad horizontal

2- Describa dos problemas que se pueden producir en una base de datos con muy alta actividad y analice las soluciones para cada uno de ellos.

- actualización perdida: lock a filas a modificar (select for update)
- bloqueos de acceso (deadlock): minimizar el tiempo de lock

3 - Analice el siguiente párrafo y justifique si usted está de acuerdo o en desacuerdo con lo que se expresa, justificando claramente su elección.

"Uno de los objetivos que se busca al definir vistas es mantener, de cara al usuario, el mismo model de datos inicialmente publicado. Con ello, se está favoreciendo la confidencialidad de los datos, dado que la vista filtra los datos que pueden ser entregados al usuario. Sin embargo, este beneficio tiene un costo debido a que se aumenta el tamaño de la base de datos y, por ende, aumenta también el número de accesos requeridos para satisfacer una consulta de datos."

Frase 1: de acuerdo.
Frase 2: puede ayudar a la confidencialidad al no mostrar ciertos atributos de carácter reservado, Aparo si esos atributos ya estaban en el modelo inicial, solo podría filtrar los datos a mostrar.
Frase 3: las vistas no aumentan el tamaño de la BdD, pero puede aumentar los accesos al tener que asociar tablas.

4- Analice el siguiente párrafo y justifique si usted está de acuerdo o en desacuerdo con lo que se expresa, justificando claramente su elección.

"Un indice nos permite disminuir los accesos a la base de datos, lo que implica un correspondiente ahorro de tiempo para resolver una instrucción SQL. Por este motivo, mientras mas indices tenga una tabla, mayor va a ser la reducción del tiempo de la consulta. En términos generales, lo que se debe considerar para lograr la máxima eficiencia de la base de datos es tener definido un indice por cada uno de los atributos de selección usados en las consultas."

Frase 1: depende del tipo de consulta y de las filas involucradas.
Frase 2: no. Mas indices => lentitud de insert/delete/update
Frase 3: si, pero solo favorece a select específicos

* * *

5- Analice los siguientes párrafos e indique si está de acuerdo con lo que exponen, justificando sus comentarios.

"Las transacciones de base de datos permiten aislar el procesamiento de lás sentencias SQL que estén en ella. Esto implica que la ejecución de dichas sentencias no se verá afectada por otras sentencias que estén ocurriendo simultáneamente. Así mismo, el resultado de dicha ejecución solo se podrá conocer una vez que haya finalizado la transacción."

R5: No es correcto decir que las sentencias SQL que estan en una transaccion de base de datos no se vean afectadas por la ejecucion de sentencias fuera de ella. De hecho, los cambios a la base de datos que ocurran mientras la transaccion esta ejecutando son visibles para las sentencias dentro de la transaccion.

Lo que invisibiliza la transaccion al "mundo exterior" son los cambios que sus sentencias estan haciendo y solo eso se conocera si la transaccion finaliza con commit.

Algunos de los problemas del uso de vistas es que ellas ocupan mucha memoria, tanto en su definición, como en su uso. Ademas, la vista se materializa cuando es referenciada en alguna sentencia SQL y desaparece una vez que ha finalizado el proceso que la haya utilizado. Una siguiente referencia a la misma vista implica una nueva materialización, volviendo a repetirse el proceso de generación de ella, con el consiguiente consumo de recursos del sistema."

La definición de una vista no ocupa espacio en la memoria del computador, sino que la usa cuando una sentencia SQL hace referencia a ella.

En este caso, se ejecuta la sentencia que define a la vista y, como es obvio, se debe traer a memori los bloques necesarios para obtener los datos solicitados.

La vista nace (no se materializa, pues una vista materializada es otra cosa), impacta el plan de ejecución de la sentencia que la referencia y es descartada (desaparece) cuando la ejecución de la sentencia SQL finaliza.

Este proceso se repite si hay otra sentencia SQL que también requiere de la vista para obtener los datos solicitados.

Resumiendo, la frase 1 es incorrecta y las frases 2 y 3 serian correctas si se asume que materializa se refiere a ejecutar la sentencia SQL que define a la vista.

3. "A pesar de lo expuesto en el párrafo anterior en cuanto al rendimiento, las vistas tienen un rol fundamental en garantizar la confidencialidad de los datos. En efecto, una vista puede ocultar los datos que no se quiere que sean conocidos por los usuarios. De esta forma, al consultar la vista, esta no va a entregar los datos definidos como confidenciales. Ademas, es posible tener varias vista sobre las mismas tablas, de manera de tener distintas visiones sobre los mismos datos."

En efecto, las vistas hacen un gran aporte a la confidencialidad de los datos, ya que una vista pued enmascarar (no mostrar) parte de los datos de una o varias tablas.

Sin embargo, la vista no oculta los datos, sino que simplemente muestra los datos que son solicitado en la definición de ella.

En una vista no existe el concepto de datos confidenciales, pues todos los atributos definidos en la vista van a ser entregados a la sentencia solicitante.

Y, finalmente, es posible definir varias vistas para la misma tabla, ya que la definición de una vista es independiente de otras vistas que existan.

En concreto, las frases 1 y 4 son correctas, mientras que las frases 2 y 3 no lo son.

4. "Teniendo como objetivo disminuir el tiempo de respuesta ante accesos al subsistema de almacenamiento, es recomendable utilizar un esquema de almacenamiento raid, ya que con ello se logra la recuperación en paralelo de los bloques requeridos. En efecto, en un esquema raid 5 hay cinco discos que van a estar leyendo simultáneamente los bloques, lo cual implica una reducción del tiempo total de lectura a aproximadamente un quinto del tiempo que implicaría leer los bloques desde un sol disco."

R4: El esquema de almacenamiento raid no asegura una reducción del tiempo de respuesta, pues el objetivo de raid es evitar la perdida de datos ante fallas de un disco, implementando mecanismos de redundancia de lo que se grabe.

En particular, en raid 5 un bloque se divide en tantas partes como sea la cantidad de discos que lo integran menos uno. No hay bloques que esten obteniendo en paralelo, pues todos los discos del raid estan aportando una parte de uno de los bloques requeridos.

Por lo tanto, no hay una reducción de tiempos.

Todo el parrafo es incorrecto.

Un deadlock se produce cuando dos o mas consultas solicitan exactamente el mimo dato (por ejemplo, el saldo disponible en una tarjeta). Con el fin de evitar el problema de la 'actualización perdida', el sistema procede a bloquear el acceso a ese dato y, como al menos hay dos procesos solicitando lo mismo, estos procesos quedan en una espera infinita. Sin embargo, el sistema es capaz de detectar esta situación y procede a 'matar' uno de los procesos para que el otro pueda continuar su ejecución y finalice exitosamente."

No, el deadlock no se produce por accesos simultáneos al mismo dato, sino que por dos transacciones haciendo intentos de actualizar un dato que está bloqueado por la otra transacción.

Si una transacción necesita el acceso a un dato bloqueado por otra, queda en espera hasta que se libere el bloqueo.

Y ese bloqueo puede liberarse cuando la transacción bloqueadora haya finalizado o porque se ha detectado un deadlock y el sistema procede a matar al proceso causante del deadlock.

En resumen, las frases 1 y 2 son incorrectas, mientras que la frase 3 es correcta.

* * *