UNIVERSIDAD ARGENTINA DE LA EMPRESA

Departamento de Tecnología Informática

BASES DE DATOS - BASE DE DATOS I - INGENIERÍA DE DATOS I

Profesor: Ing. Hernán Puelman

Trabajo Práctico

FECHA: 23-10-2024

Grupo de trabajo

Legajo	Apellido y Nombre
. . . . 1195949	. . . . . .Cejas, Katerina Andrea. . . . .
. . . . .	. . . . .
. . . . .	. . . . .

NOTA: LEA ATENTAMENTE.

- Responda claramente cada ejercicio. Escribiendo las respuestas en el documento y detallando su estrategia de resolución en caso de ser necesario.
- Sea claro, prolijo y ordenado en el desarrollo de las respuestas.
- Cada ejercicio resuelto correctamente vale 1,25 puntos.
- **Aprobación del TP: Con el 60% de los ejercicios resueltos correctamente.**
- **Plazo de entrega: 23/10/2024 11:59hs (AM)**

1. Dada una tabla PERSONAS que contenga la siguiente estructura, seleccione las filas de la tabla tal que los valores del atributo candidato a ser Primary Key estén repetidos más de una vez.

Id (Clave candidata)
Nombre varchar(20)
Apellido varchar(20)

La query realizada es:

```
select Id, Nombre, Apellido
from Personas
where Id in
       (select Id
        from Personas
        group by Id
        having count(Id) > 1)
```

Pude verificar la consulta generando la tabla Personas e insertándole algunos registros como ejemplo, teniendo en cuenta que el Id debía duplicarse, y que tanto el Nombre como el Apellido podían ser o no duplicados.

```
create table Personas(
       id int NOT NULL,
       Nombre varchar(20),
       Apellido varchar(20)
)

insert Personas (Id, Nombre, Apellido) values (1,'Kate','Cejas')
insert Personas (Id, Nombre, Apellido) values (1,'Kate','Cejas')
insert Personas (Id, Nombre, Apellido) values (1,'Chispi','Cejas')
insert Personas (Id, Nombre, Apellido) values (1,'Chispi','Cejas')
insert Personas (Id, Nombre, Apellido) values (1,'Atun','Cejas')
insert Personas (Id, Nombre, Apellido) values (1,'Atun','Cejas')
insert Personas (Id, Nombre, Apellido) values (2,'Nacho','Norte')
insert Personas (Id, Nombre, Apellido) values (3,'Uma','Norte')
insert Personas (Id, Nombre, Apellido) values (3,'Michu','Norte')
insert Personas (Id, Nombre, Apellido) values (3,'Michu','Norte')
```

El resultado devuelto por la query es el siguiente: (el registro con Id=2 no es devuelto)

Resultados | Mensajes

	Id	Nombre	Apellido
1	1	Kate	Cejas
2	1	Chispi	Cejas
3	1	Atun	Cejas
4	3	Uma	Norte
5	3	Michu	Norte
6	3	Michu	Norte
7	1	Kate	Cejas
8	1	Chispi	Cejas
9	1	Atun	Cejas

2. Realice una sentencia DELETE de forma tal que borre las filas de la tabla PERSONAS dejando sólo una fila con cada valor de la clave candidata.

Continuando con la tabla y los registros insertados en el ejercicio anterior, la sentencia delete es la siguiente:

```
with duplicados as (
    select Id, Nombre, Apellido,
           row_number() over (partition by Id order by Id) as nro_fila
    from Personas
)
delete from duplicados
where nro_fila > 1
```

El resultado luego del delete:

Resultados | Mensajes

	Id	Nombre	Apellido
1	1	Kate	Cejas
2	2	Nacho	Norte
3	3	Uma	Norte

3. Realizar una consulta que devuelva los totales comprados por todos los clientes en **cada** provincia de los fabricantes.

```
select fab.provincia_cod, coalesce(sum(det.cantidad * det.precio_unit), 0) as total_comprado
from fabricantes fab
       left join productos prod
              on prod.fabricante_cod = fab.fabricante_cod
       left join facturas_det det
              on prod.producto_cod = det.producto_cod
group by fab.provincia_cod
order by fab.provincia_cod
```

El resultado de la consulta:

Resultados | Mensajes | Esta

	provincia_cod	total_comprado
1	BA	33839.00
2	CF	45705.00
3	CH	0.00
4	CO	32210.00
5	ER	0.00
6	JU	6850.00
7	LP	7000.00
8	SA	3200.00

4. Sea una tabla que tiene un campo numérico secuencial, realizar una consulta (Select) que obtenga el valor del primer espacio libre de ese campo. Por ej.

1	1er valor disponible **4**
2	
3	
6	
7	

4	1er valor disponible **7**
5	
6	
10	
11	

	1er valor disponible 1

4	1er valor disponible **9**
5	
6	
7	
8	

La consulta es:

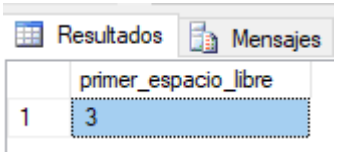
```
select coalesce(min(numero + 1), 1) as primer_espacio_libre
from numeros_secuenciales
where numero + 1 not in (select numero from numeros_secuenciales)
```

Pude probarla creando la tabla números_secuenciales e insertándole registros.

```
create table numeros_secuenciales(
	numero int null
)

insert numeros_secuenciales(numero) values (1)
insert numeros_secuenciales(numero) values (2)
insert numeros_secuenciales(numero) values (456)
insert numeros_secuenciales(numero) values (8)
```

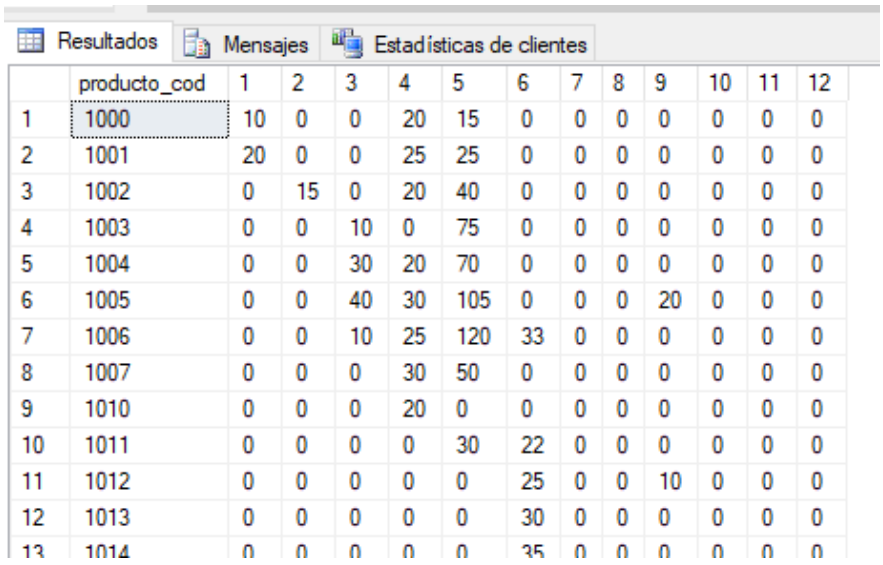
El resultado de la consulta con esos registros de ejemplo es:



5. Mostrar por cada Producto (producto_cod) l**as cantidades totales mensuales vendidas** (según la fecha de emisión de la factura). No importa si hay meses de diferentes años, se suman como si fueran del mismo año. La información se deberá mostrar en forma tabular como se muestra en la figura ordenada por código de producto.

Producto	1	2	3	4	5	6	7	8	9	10	11	12
1000	200	0	0	0	300	0	0	0	350	0	0	100
1001	0	200	0	300	0	400	0	0	0	0	0	0
1002	100	200	0	0	0	300	150	0	900	0	200	0
1003	100	50	100	50	0	0	0	100	200	150	0	100
1004	300	400	150	200	100	100	100	200	100	100	100	200
1005	0	0	0	0	100	0	0	0	100	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...

```
select det.producto_cod,
    sum(case when month(fact.fecha_emision) = 1 then det.cantidad else 0 end) as '1',
    sum(case when month(fact.fecha_emision) = 2 then det.cantidad else 0 end) as '2',
    sum(case when month(fact.fecha_emision) = 3 then det.cantidad else 0 end) as '3',
    sum(case when month(fact.fecha_emision) = 4 then det.cantidad else 0 end) as '4',
    sum(case when month(fact.fecha_emision) = 5 then det.cantidad else 0 end) as '5',
    sum(case when month(fact.fecha_emision) = 6 then det.cantidad else 0 end) as '6',
    sum(case when month(fact.fecha_emision) = 7 then det.cantidad else 0 end) as '7',
    sum(case when month(fact.fecha_emision) = 8 then det.cantidad else 0 end) as '8',
    sum(case when month(fact.fecha_emision) = 9 then  det.cantidad else 0 end) as '9',
    sum(case when month(fact.fecha_emision) = 10 then det.cantidad else 0 end) as '10',
    sum(case when month(fact.fecha_emision) = 11 then det.cantidad else 0 end) as '11',
    sum(case when month(fact.fecha_emision) = 12 then det.cantidad else 0 end) as '12'
from facturas_det det
       left join facturas fact
              on det.factura_num = fact.factura_num
group by det.producto_cod
```



Resultados | Mensajes | Estadísticas de clientes

	producto_cod	1	2	3	4	5	6	7	8	9	10	11	12
1	1000	10	0	0	20	15	0	0	0	0	0	0	0
2	1001	20	0	0	25	25	0	0	0	0	0	0	0
3	1002	0	15	0	20	40	0	0	0	0	0	0	0
4	1003	0	0	10	0	75	0	0	0	0	0	0	0
5	1004	0	0	30	20	70	0	0	0	0	0	0	0
6	1005	0	0	40	30	105	0	0	0	20	0	0	0
7	1006	0	0	10	25	120	33	0	0	0	0	0	0
8	1007	0	0	0	30	50	0	0	0	0	0	0	0
9	1010	0	0	0	20	0	0	0	0	0	0	0	0
10	1011	0	0	0	0	30	22	0	0	0	0	0	0
11	1012	0	0	0	0	0	25	0	0	10	0	0	0
12	1013	0	0	0	0	0	30	0	0	0	0	0	0
13	1014	0	0	0	0	0	35	0	0	0	0	0	0

6. Realizar un query que **asegure** que dos tablas con la misma estructura tengan exactamente la misma cantidad de filas y exactamente los mismos datos en cada columna. Es decir, que las tablas sean **exactamente** iguales.

```
create procedure asegurarMismaEstructura as
begin
   -- valido que tengan la misma cantidad de registros
   if (select count(*) from tabla1) != (select count(*) from tabla2)
       begin
              print 'Nro. Error: ' + cast(error_number() as varchar);
              print 'mensaje: ' + error_message();
              print 'estado: ' + cast(error_state() as varchar);
              throw 50000, 'Las tablas tienen distinta cantidad de registros', 1
       end

   -- en caso de que la cantidad de registros sea la misma, se valida que el resto sea
   exactamente igual.
   -- se hace una validacion de la cantidad previo a esta validacion porque el except no
   distingue entre duplicados,
   -- y si hay duplicados entonces podría variar la cantidad de registros por mas que los datos
   sean los mismos.

   declare @cantidad_registros_distintos int
   select @cantidad_registros_distintos = count(*) from
( select * from tabla1 except select * from tabla2
   union all
   select * from tabla2 except select * from tabla1 ) as diferencias_entre_tablas

   if @cantidad_registros_distintos > 0
       begin
              print 'Nro. Error: ' + cast(error_number() as varchar);
              print 'mensaje: ' + error_message();
              print 'estado: ' + cast(error_state() as varchar);
              throw 50000, 'Las tablas no son exactamente iguales', 1
       end

   -- en caso de no haberse lanzado ningun eror:
   print 'Las tablas son exactamente iguales'
end


exec asegurarMismaEstructura
```

7. Realizar una consulta que tenga el siguiente formato (los datos son solo de ejemplo) que muestre para cada cliente **que tenga referente** los distintos niveles de referentes que posee. Cada referente se deberá mostrar **en una columna diferente y debe mostrar hasta el tercer nivel si es que existe.**

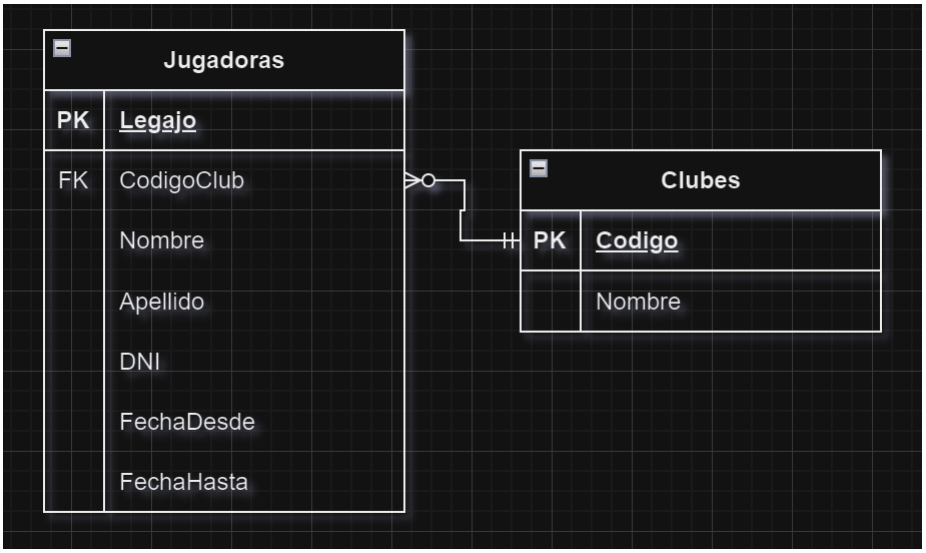
Nro cliente	Referente	Referente2	Referente3
108	107	106	103
106	103	101	
…	…	…	…
104	103	101	…

```sql
select c1.cliente_num,
       c1.cliente_ref as 'Referente',
       c2.cliente_ref as 'Referente2',
       c3.cliente_ref as 'Referente3'
from clientes c1
       left join clientes c2
              on c1.cliente_ref = c2.cliente_num
       left join clientes c3
              on c2.cliente_ref = c3.cliente_num
where c1.cliente_ref is not null
```

Resultados | Mensajes | Estadísticas de clientes

	cliente_num	Referente	Referente2	Referente3
1	102	101	NULL	NULL
2	103	101	NULL	NULL
3	104	103	101	NULL
4	105	103	101	NULL
5	106	103	101	NULL
6	107	106	103	101
7	108	107	106	103
8	109	107	106	103
9	112	111	NULL	NULL
10	113	111	NULL	NULL
11	114	111	NULL	NULL

8. Dada el siguiente modelo, desarrolle e implemente un mecanismo que controle y evite que una jugadora esté fichada para dos equipos al mismo tiempo (inclusive el mismo). Es decir, que dos registros no se superpongan dentro de un periodo para un mismo legajo.



```
create trigger evitarFichajeAlMismoTiempo
on jugadoras
instead of insert as
begin
-- validar la existencia de la jugadora
if exists (select top 1 legajo from jugadoras where legajo = (select legajo from inserted))
	begin
	-- se declara el cursor
	declare cursorJugadoras cursor for
	select legajo, codigoClub, nombre, apellido, dni, fechaDesde, fechaHasta
	from jugadoras
	where legajo = (select legajo from inserted)

	-- se declaran las variables que almacenaran cada iteracion del cursor
	declare @legajo int, @codigoClub int, @nombre varchar(30), @apellido varchar(30),
	@dni int, @fechaDesde date, @fechaHasta date

	--abrir cursor
	OPEN cursorJugadoras

	-- primera iteracion, guardo la primer fila en las variables
	fetch cursorJugadoras into @legajo, @codigoClub, @nombre, @apellido,
	@dni, @fechaDesde, @fechaHasta

	While (@@FETCH_STATUS=0) --mientras haya filas por leer de la misma jugadora
	begin
	-- validar que las fechas Desde y Hasta no esté ninguna de las dos dentro del periodo en
la jugadora ya existente
		if((select fechaDesde from inserted) between @fechaDesde and @fechaHasta)
		    or
		  ((select fechaHasta from inserted) between @fechaDesde and @fechaHasta)

		begin -- en caso de que los periodos no sean validos, se lanza error
			print 'Nro. Error: ' + cast(error_number() as varchar);
			print 'mensaje: ' + error_message();
			print 'estado: ' + cast(error_state() as varchar);
			throw 50000, 'Se intentó insertar a una jugadora en un periodo donde ya
tiene un club asignado.', 1
		end
		-- busca la fila siguiente
		fetch cursorJugadoras into @legajo, @codigoClub, @nombre, @apellido,
		@dni, @fechaDesde, @fechaHasta
	end

	--cerrar y liberar cursor
	close cursorJugadoras
	DEALLOCATE cursorJugadoras
	end

	-- en caso de no existir la jugadora, o que los periodos sean válidos, inserto en la
tabla
	insert into jugadoras (legajo, codigoClub, nombre, apellido, dni, fechaDesde,
fechaHasta)
	select legajo, codigoClub, nombre, apellido, dni, fechaDesde, fechaHasta
	from inserted
end
```

Para poder verificar el funcionamiento del trigger, cree las dos tablas y fui insertando registros de prueba.

```
create table clubes(
	codigo int primary key,
	nombre varchar(30) not null,
)

create table jugadoras(
	legajo int not null,
	codigoClub int references clubes(codigo) not null,
	nombre varchar(30) not null,
	apellido varchar(30) not null,
	dni int not null,
	fechaDesde date not null,
	fechaHasta date not null,
	primary key (legajo, fechaDesde, fechaHasta)
)
```

Si bien el modelo dado en la consigna indica que la única primary key de la tabla jugadoras es el legajo, yo decidí agregar a las 2 fechas como parte de la primary key y hacer una clave primaria compuesta, porque en la consigna pide textualmente “que dos registros no se superpongan dentro de un periodo para un mismo legajo”, por lo que el legajo se puede repetir y eso no lo haría una PK. Es por eso que modifiqué la tabla para poder tener en cuenta las fechas y armar el trigger.

```
insert into clubes (codigo, nombre) values (1, 'unClub')
insert into clubes (codigo, nombre) values (2, 'otroClub')
```

El primer registro que inserto en la tabla jugadoras es el siguiente, el cual se inserta correctamente porque la jugadora con ese periodo no existe en la tabla.

```
insert into jugadoras (legajo, codigoClub, nombre, apellido, dni, fechaDesde, fechaHasta)
values(100, 1, 'kate', 'cejas', 12345,'2024-01-01','2024-12-31')
```

Luego ingreso este otro registro que también se inserta en la tabla porque si bien la jugadora con ese legajo existe y ya está en un club, ahora esta en otro club en otro periodo de tiempo

```
insert into jugadoras (legajo, codigoClub, nombre, apellido, dni, fechaDesde, fechaHasta)
values(100, 2,'kate', 'cejas', 12345,'2025-01-01','2025-12-25')
```

Por último, este registro no se puede insertar porque durante ese periodo la jugadora ya está en el club 1.

```
insert into jugadoras (legajo, codigoClub, nombre, apellido, dni, fechaDesde, fechaHasta)
values(100, 2,'kate', 'cejas', 12345,'2024-09-03','2024-12-31')
```


```
Mens 50000, Nivel 16, Estado 1, Procedimiento evitarFichajeAlMismoTiempo, Línea 37
Se intentó insertar a una jugadora en un periodo donde ya tiene un club asignado.
```