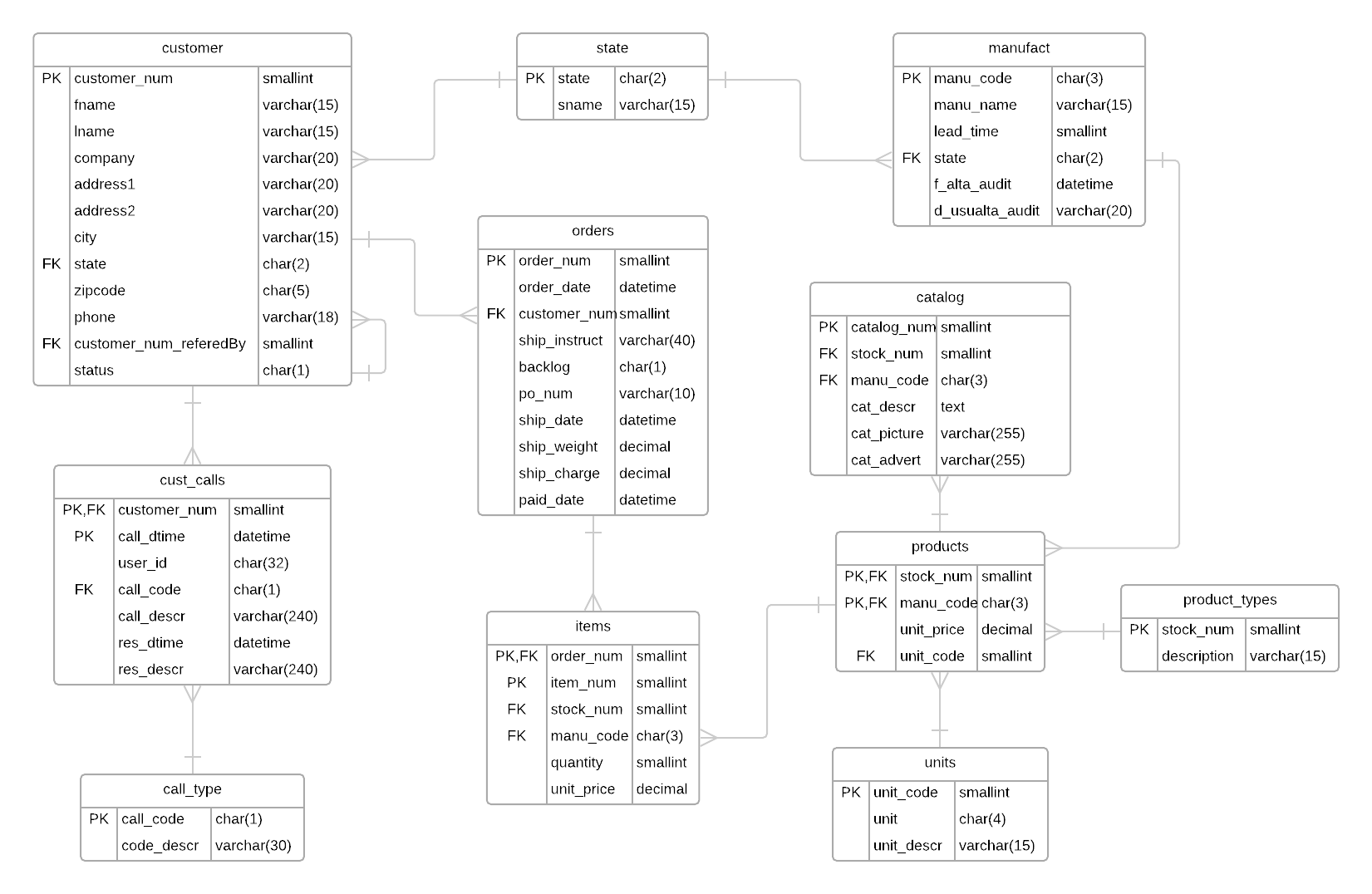
**PARTE 1 – Teoría y SQL**

* 1. Explique en menos de 15 renglones qué es Dominio y las diferentes formas de implementarlo en una BD.
  2. En una carilla explique Índices: Qué son, para qué sirven, tipos, ventajas, desventajas, y su relación con la funcionalidad de integridad.
  3. SQL

****

Obtener los Tipos de Productos, los montos totales vendidos por cliente y por sus referidos. Mostrar: descripción del Tipo de Producto, Nombre y apellido del cliente, monto total comprado de ese tipo de producto, Nombre y apellido de su cliente referido y el monto total comprado de su referido. Ordenado por Descripción y Nombre del cliente (Padre).

**Nota:** Si el Cliente no tiene referidos o sus referidos no compraron el mismo producto, mostrar ´--´ como nombre y apellido del referido y 0 (cero) en la cantidad vendida.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Description** | **Apellido** | **Nombre** | **Total comprado** | **Apellido**  **Referido** | **Nombre**  **Referido** | **Total comprado** |
| 3 golf balls | Putnum | Chris | 1248 | -- | -- | 0 |
| golf shoes | Jewell | Fred | 600 | Putnum | Chris | 825 |
| golf shoes | Putnum | Chris | 825 | -- | -- | 0 |
| metal Woods | Jewell | Fred | 230 | Putnum | Chris | 2070 |
| metal Woods | Jewell | Fred | 230 | Zuarez | Robert | 210 |
| metal Woods | Putnum | Chris | 2070 | -- | -- | 0 |
| tennis ball | Grant | Alfred | 84 | Parmelee | Jean | 84 |
| tennis ball | Higgins | Anthony | 84 | -- | -- | 0 |

select tp.description, c.lname, c.fname, sum(i.unit\_price\*i.quantity) totalPadre

,coalesce(r.lname, '--') lNameHijo, coalesce(r.fname, '--') fnamehijo,

coalesce(r.totalRef,0) totalHijo

from product\_types tp join items i on tp.stock\_num = i.stock\_num

join orders o on o.order\_num = i.order\_num

join customer c on o.customer\_num = c.customer\_num

left join (select c2.customer\_num, c2.customer\_num\_referedBy,

c2.lname, c2.fname, i2.stock\_num,

sum(i2.unit\_price\*i2.quantity) totalRef

from customer c2 join orders o2

on o2.customer\_num = c2.customer\_num

join items i2

on i2.order\_num = o2.order\_num

group by c2.customer\_num, c2.customer\_num\_referedBy,

c2.lname, c2.fname, i2.stock\_num) r

on c.customer\_num = r.customer\_num\_referedBy and

tp.stock\_num = r.stock\_num

group by tp.description, c.lname, c.fname, r.lname, r.fname, r.totalRef

order by 1, 2

**PARTE 2 –**

1. **Crear un procedimiento actualizaPrecios que reciba como parámetro una fecha a partir de la cual procesar los registros de una tabla Novedades que contiene los nuevos precios de Productos con la siguiente estructura/información.**

**FechaAlta, Manu\_code, Stock\_num, descTipoProducto, Unit\_price, Unit\_code**

**Por cada fila de la tabla Novedades**

**Si no existe el Fabricante, devolver un error de Fabricante inexistente y descartar la**

**novedad.**

**Si no existe el stock\_num (pero existe el Manu\_code) darlo de alta en la tabla**

**Product\_types e insertar el nuevo par (stock\_num, manu\_code) en la tabla Products.**

**Si existe el Producto actualizar su precio**

**Si no existe, Insertarlo en la tabla de productos.**

**Nota: Manejar una transaccion por novedad y errores no contemplados.**

**Solucion**

DROP table ##Novedades;

create table ##Novedades (

Stock\_num smallint,

Manu\_code char(3),

descTipoProducto varchar(15),

unit\_price decimal(6,2),

unit\_code smallint,

fechaAlta datetime);

create procedure ActualizaPrecios @fechaProceso date as

begin

-- declaro variables y cursor

declare @stock\_num smallint,

@Manu\_code char(3),

@descTipoProducto varchar(15),

@unit\_price decimal(6,2),

@unit\_code smallint

--

declare cNovedades CURSOR FOR

Select stock\_num, Manu\_code, descTipoProducto, unit\_price

from ##Novedades

where fechaAlta > @fechaProceso;

--

open cNovedades;

fetch NEXT from cNovedades into @stock\_num, @Manu\_code, @descTipoProducto,

@unit\_price, @unit\_code;

WHILE @@FETCH\_STATUS = 0

Begin

Begin try

Begin TRAN

if not exists (select 1 from manufact where manu\_code = @Manu\_code)

throw 50000, 'Fabricante erroneo', 2

if not exists (select 1 from product\_types where stock\_num = @stock\_num)

insert into product\_types values (@stock\_num, @descTipoProducto);

if not exists (select 1 from products where stock\_num = @stock\_num and

manu\_code = @Manu\_code)

insert into products values (@stock\_num, @Manu\_code, @unit\_price, @unit\_code);

else

update products set unit\_price = @unit\_price

where stock\_num = @stock\_num and manu\_code = @Manu\_code;

commit;

end try

begin catch

rollback

end catch

fetch NEXT from cNovedades into @stock\_num, @Manu\_code, @descTipoProducto,

@unit\_price, @unit\_code;

End

--

close cNovedades

Deallocate cNovedades

--

end;

1. **Triggers**

**Se desea llevar en tiempo real la cantidad de llamadas/reclamos (Cust\_calls) de los Clientes (Customers) que se producen por cada mes del año y por cada tipo (Call\_code).**

**Ante este requerimiento, se solicita realizar un trigger que cada vez que se produzca un Alta o Modificación en la tabla Cust\_calls, se actualice una tabla donde se lleva la cuenta por Año, Mes y Tipo de llamada.**

**Ejemplo. Si se da de alta una llamada, se debe sumar 1 a la cuenta de ese Año, Mes y Tipo de llamada. En caso de ser una modificación y se modifica el tipo de llamada (por ejemplo por una mala clasificación del operador), se deberá restar 1 al tipo anterior y sumarle 1 al tipo nuevo. Si no se modifica el tipo de llamada no se deberá hacer nada.**

**Tabla ResumenLLamadas**

**Anio decimal(4) PK,**

**Mes decimal(2) PK,**

**Call\_code char(1) PK,**

**Cantidad int**

**Nota: No se modifica la PK de la tabla de llamadas. Tener en cuenta altas y modificaciones múltiples.**

**create trigger custCallsTr ON Cust\_calls**

**AFTER Insert, update as**

**begin**

**--**

**declare @anio decimal(4)**

**declare @mes decimal(2)**

**declare @tipo char(1)**

**--**

**declare curIns CURSOR FOR**

**select year(call\_time), month(call\_time), call\_code**

**from inserted**

**declare curDel CURSOR FOR**

**select year(call\_time), month(call\_time), call\_code**

**from deleted**

**open curIns**

**FETCH NEXT FROM curIns into @anio, @mes, @tipo**

**while @@FETCH NEXT FROM\_status = 0**

**begin**

**update resumenLlamadas**

**set cantidad += 1**

**where anio = @anio**

**and mes = @mes**

**and call\_code = @tipo**

**--**

**FETCH NEXT FROM curIns into @anio, @mes, @tipo**

**end**

**close curIns**

**Deallocate curIns**

**open curDel**

**FETCH NEXT FROM curDel into @anio, @mes, @tipo**

**while @@FETCH NEXT FROM\_status = 0**

**begin**

**update resumenLlamadas**

**set cantidad = cantidad - 1**

**where anio = @anio**

**and mes = @mes**

**and call\_code = @tipo**

**--**

**FETCH NEXT FROM curDel into @anio, @mes, @tipo**

**end**

**close curDel**

**Deallocate curDel**

**--**

**COMMIT**

**--**

**end**