**Para levantar la BD: Windows + R: services.msc  
  
Clase 1**

RDBMS = Relational Data Base Management Systems (motores de base de datos)

SQL

-DML Data Manipulation Language  
 -DDL Data Definition Language

Concepto de E F Codd 1970:  
-Estructura: Recopilación de objetos o relaciones  
-Manipulación de datos: Operaciones sobre las relaciones para producir otras  
-Integridad: Para obtener precisión y consistencia (se definen reglas)

Relación: estructura bidimensional (tabla) con cabecera y cuerpo  
Cabecera: nombres de atributos  
Cuerpo: tuplas de datos con tantas entradas como atributos tiene la tabla  
Cardinalidad: cantidad de tuplas  
Grado: cantidad de atributos

Tupla tiene pares atributo-valor

Dominio: Conjunto de valores posibles que puede tomar un atributo.  
 -el nulo no es un valor posible  
 -los valores de dominio no son semánticamente divisibles  
Clave foránea: conjunto de atributos de R1 (una relación) cuyos valores no nulos deben coincidir con los valores de la clave primaria de la relación R2 (otra relación)

**Reglas de integridad**   
Regla de integridad de las entidades: Ningún componente de la clave primaria de una relación base puede aceptar nulos

Regla de integridad referencial: La base de datos NO debe contener valores no nulos de clave foránea para los cuales no exista un valor concordante de clave primaria en la relación referenciada.

Se reducen las tablas con select o lo que fuera antes de joinear.

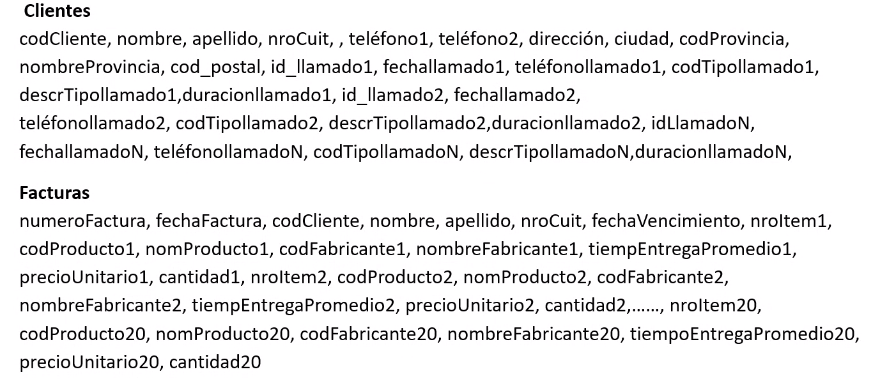
En caso que no sea posible se joinea con las tablas más chicas primero

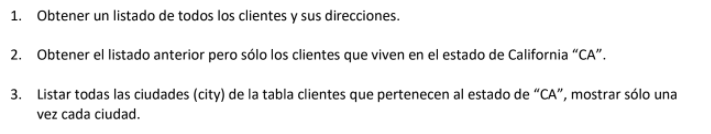
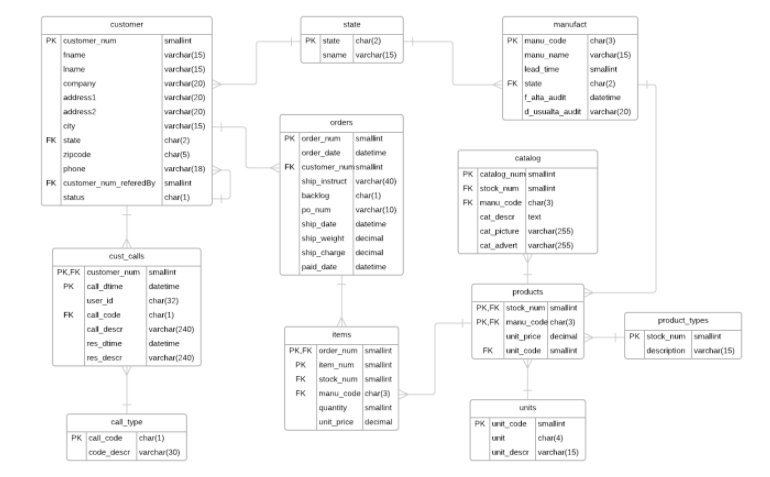
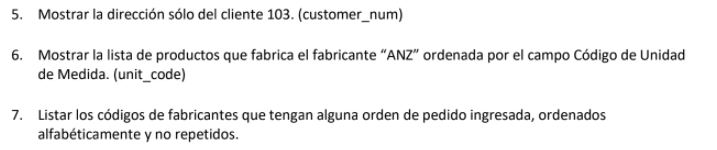
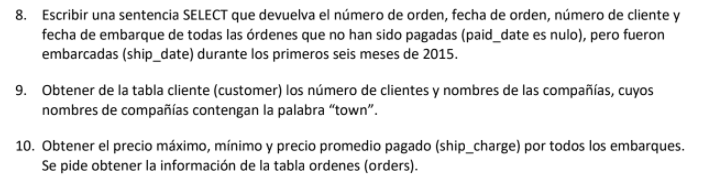
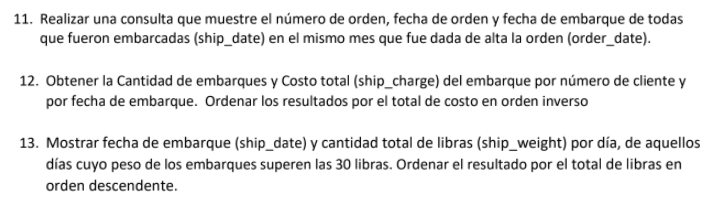
NO SE TOMA ÁLGEBRA RELACIONAL (ni en parcial ni en final)

Parcial práctico:  
2 preguntas teóricas (1 larga(1 página), 1 corta(10 renglones))  
1 Query  
1 StoreProcedure  
1 Trigger  
Y quizás una vista

**Clase 2:**

DDL: Crea, modifica, dropea, etc. tablas.

Las tablas en general se denominan en plural  
  
**Usamos hasta la 3ra forma normal.  
**

**--Facturas-**numeroFactura(PK)  
codCliente(FK)  
fechaFactura  
fechaVencimiento  
  
  
**--Items-**nroFactura(PK)(FK)  
nroItem(PK)  
codProducto(FK)  
cantidad  
  
  
  
**--Productos-**codProducto(PK)  
nomProducto  
codFabricante(FK)  
precioUnitario  
  
  
  
**--Fabricantes-**  
codFabricante(PK)  
nombreFabricante  
tiempoEntregaPromedio  
 **--Clientes-**codCliente (PK)  
codProvincia(FK)  
cod\_postal  
nombre  
apellido  
nroCuit  
telefono1  
telefono2  
dirección  
ciudad  
  
**--Provincias-**codProvincia(PK)  
nombreProvincia  
 **--Llamados-**  
id\_llamado (PK)  
codCliente(FK)  
codTipoLlamado (FK)  
duracionLlamado  
fecha\_llamado  
teléfonollamado  
  
 **--TiposDeLlamado-**codTipollamado(PK)  
descrTipoLlamado  
  
  
  
CREATE TABLE facturas (  
numeroFactura BIGINT PRIMARY KEY,  
codCliente INTEGER NOT NULL REFERENCES clientes,  
fechaFactura DATE CHECK (fechaFactura <= CURRENT\_DATE) DEFAULT CURRENT\_DATE,  
fechaVencimiento DATE  
);  
  
CREATE TABLE clientes (  
codCliente INTEGER PRIMARY KEY,  
codProvincia INTEGER REFERENC ES provincias,  
cod\_postal INTEGER,  
nombre VARCHAR (50) NOT NULL,  
apellido VARCHAR(50) NOT NULL,  
nroCuit NUMERIC(11,0) UNIQUE NOT NULL,  
telefono1 INTEGER,  
telefono1 INTEGER,  
dirección VARCHAR(100),  
ciudad VARCHAR(50)  
);  
  
CREATE TABLE provincias (  
codProvincia INTEGER PRIMARY KEY,  
nombreProvincia VARCHAR(50)  
);  
  
CREATE TABLE productos (  
codProducto INTEGER PRIMARY KEY,  
nomProducto NOT NULL VARCHAR(50),  
codFabricante INTEGER REFERENCES fabricantes,  
precioUnitario DOUBLE (7,2) NOT NULL CHECK (precioUnitario>0)  
);   
  
CREATE TABLE fabricantes (  
codFabricante INTEGER PRIMARY KEY,  
nombreFabricante VARCHAR(50) NOT NULL,  
tiempoEntregaPromedio TIME  
);  
  
  
  
  
  
  
CREATE TABLE items (  
nroFactura INTEGER REFERENCES facturas,  
nroItem INTEGER,  
codProducto INTEGER NOT NULL REFERENCES productos,  
cantidad INTEGER NOT NULL CHECK (cantidad > 0),  
PRIMARY KEY (nroFactura, nroItem)  
);  
CREATE TABLE llamados (  
id\_llamado INTEGER PRIMARY KEY,  
codCliente INTEGER NOT NULL REFERENCES clientes,  
codTipoLlamado INTEGER REFERENCES tipos\_de\_llamado,  
duracionLlamado TIME,  
fecha\_llamado DATE DEFAULT CURRENT\_DATE,  
teléfono\_llamado VARCHAR(20) NOT NULL  
);  
  
CREATE TABLE tipos\_de\_llamado(  
codTipoLlamado INTEGER PRIMARY KEY,  
descrTipoLlamado VARCHAR (200)  
);  
  
**Clase 3**El dominio en una base de datos se implementa/restringe a través de los nombres de los atributos, tipos de datos y foreign keys.  
  
  
  
1. SELECT customer\_num, lname, fname, address1, address2 FROM customer;  
2. SELECT customer\_num, lname, fname, address1, address2 FROM customer  
WHERE state = ‘CA’;  
3. SELECT DISTINCT city FROM customer WHERE state = ’CA’;  
4. SELECT DISTINCT city FROM customer WHERE state = ’CA’ ORDER BY city;  
5. SELECT address1, address2 FROM customer WHERE customer\_num = 103;  
6. SELECT stock\_num, manu\_code, unit\_code FROM products WHERE manu\_code = ‘ANZ’ ORDERED BY unit\_code;  
7. SELECT DISTINCT manu\_code FROM items ORDER BY manu\_code;  
  
  
8. SELECT order\_num, order\_date, customer\_num, ship\_date FROM orders WHERE paid\_date IS null AND YEAR(ship\_date) = 2015 AND (MONTH(ship\_date) BETWEEN 1 AND 6);  
  
// 8 alternativa//  
select o.order\_num, o.order\_date, o.customer\_num, o.ship\_date from orders o where o.paid\_date IS NULL and o.ship\_date >= '2015-01-01' and o.ship\_date < '2015-07-01' order by 1  
/////  
  
9. SELECT customer\_num, company FROM customer WHERE company LIKE ‘%town%’;  
10. SELECT MAX(ship\_charge) AS precioMaximo, MIN(ship\_charge) AS precioMinimo, AVG(ship\_charge) AS promedio FROM orders;  
  
11. SELECT order\_num, order\_date, ship\_date FROM orders WHERE YEAR(ship\_date) = YEAR(order\_date) AND MONTH(ship\_date) = MONTH(order\_date);

12. SELECT customer\_num, ship\_date, COUNT(\*) AS cantidadEmbarques, SUM(ship\_charge) AS costoTotal FROM orders GROUP BY customer\_num, ship\_date ORDER BY 4 DESC;  
  
13. SELECT ship\_date, SUM(ship\_weight) AS totalDeLibras FROM orders GROUP BY ship\_date HAVING SUM(ship\_weight) > 30 ORDER BY totalDeLibras DESC;  
  
  
**Clase 4**  
Nombre de tabla de sesion temporal empieza con “#” y una table global temporal se crea con “##” Adelante del nombre.  
  
  
Ejercicios Insert, Update, Delete  
1.  
SELECT \*

INTO #clientes

FROM customer;

2.

INSERT INTO #clientes

(Customer\_num, Fname, Lname,Company,State, City)

VALUES (144, 'Agustín','Creevy','Jaguares SA', 'CA','Los Angeles');

3.

SELECT \* INTO #clientesCalifornia

FROM customer

WHERE State='CA';

4.

INSERT INTO #clientes

(Customer\_num, Fname, Lname,Company,State, City)

SELECT 155, Fname, Lname,Company,State, City

FROM customer

WHERE customer\_num = 103;

5.

DELETE FROM #clientes

WHERE (zipcode between 94000 and 94050) AND (city like 'M%');

6.  
DELETE FROM #clientes

WHERE customer\_num NOT IN (SELECT DISTINCT customer\_num FROM orders);

Clase 8  
Repaso de Triggers, aspectos a tener en cuenta:  
-Sobre qué objeto se dispara el trigger (Tabla o vista)  
-Cuándo se dispara el trigger? Before, After, Instead Of (En SQL server, InsteadOf o After)  
-De qué tipo? Insert, Update o Delete (Evento ante el cual se dispara)  
  
Excepciones  
Throw se usa número de 50000 inclusive en adelante  
Mensaje de excepción  
State de la excepción Entre 1 y 16?  
  
Throw número(>50000), mensaje(string), state(int)  
  
 ó  
  
raiseerror(mensaje de error(string), nivel de error (16), state(int))  
  
  
Formas de manejar aislamiento  
  
-Read uncommited  
-Read commited (default para el select)  
-Repeatable read  
  
-Serializable  
  
  
  
Dirty Read  
 Datos actualizados en una transacción que luego se deshacen por un rollback (pueden estar consistentes o no)  
Phantom Records/Read  
 Entre dos selec t iguales dentro de una transacción, una segunda transacción inserta un nuevo registro que el select contempla  
  
Ejemplo:  
Begin Transaction  
SELECT bla bla (devuelve x registros)  
  
(en este momento del tiempo se inserta un nuevo registro)  
  
SELECT bla bla (devuelve x+1 registros)  
  
Repeatable reads  
 Dos select iguales en diferentes momentos dentro de la transacción devuelven los mismos datos  
  
Deadlock: El sistema lo resuelve matando una de las dos transacciones para desbloquear la otra  
  
cómo se cuándo hace falta subquery y cuándo no  
qué pasó con lo del apunte/resumen de sintaxis y herramientas