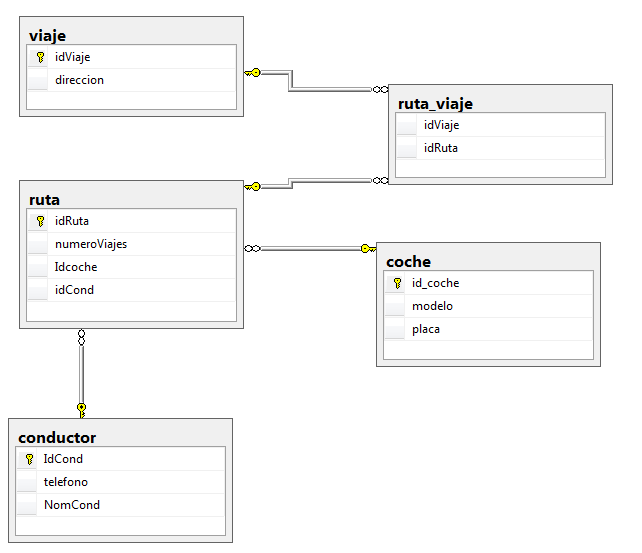
**Ejercicio UBER**

A continuación, se describen las características principales del sistema de bases de datos de UBER.

La siguiente figura muestra el modelo conceptual del sistema de base de datos.



Paso 1:

Es necesario crear los archivos **CreacionBDUber.txt** y **InsercionTuplasUber.txt** que contienen las instrucciones SQL (script) necesarias para definir la estructura de la base de datos del sistema de UBER y para insertar las tuplas que determinan el estado inicial de la base de datos.

**CreacionBDUber.txt**

-- DEFINICIÓN DE TABLAS concepto UBER

create table coche

(id\_coche smallint not null primary key,

modelo char(30),

placa char(8));

create table conductor

(IdCond smallint primary key,

telefono char(30),

NomCond char(30));

create table ruta

(idRuta smallint primary key,

numeroViajes smallint,

Idcoche smallint references coche,

idCond smallint references conductor);

create table viaje

(idViaje smallint primary key,

direccion char(30));

create table ruta\_viaje

(idViaje smallint references viaje,

idRuta smallint references ruta,

primary key (idViaje,idRuta));

**InsercionTuplasUber.txt**

-- Tuplas de la tabla conductor.

INSERT INTO conductor VALUES(501,'55509009','PEPE');

INSERT INTO conductor VALUES(601,'65509809','PACO');

INSERT INTO conductor VALUES(701,'55509779','LUIS');

INSERT INTO conductor VALUES(801,'55509280','ANA');

INSERT INTO conductor VALUES(901,'55409009','JOEL');

-- Tuplas de la tabla coche

INSERT INTO coche VALUES(1,'nissan', '1234ABC');

INSERT INTO coche VALUES(2,'bmw', '1234BCD');

INSERT INTO coche VALUES(3,'tesla', '1234CDE');

INSERT INTO coche VALUES(4,'tesla', '4321ABC');

INSERT INTO coche VALUES(5,'chevrolet', '4567ABC');

-- Tuplas de la tabla Ruta

INSERT INTO ruta VALUES(11,10,1,601);

INSERT INTO ruta VALUES(21,2,2,701);

INSERT INTO ruta VALUES(31,30,4,501);

INSERT INTO ruta VALUES(41,15,1,801);

INSERT INTO ruta VALUES(51,2,3,901);

INSERT INTO ruta VALUES(61,1,1,601);

-- Tuplas de la tabla viaje

INSERT INTO viaje VALUES(11,'sanAngel');

INSERT INTO viaje VALUES(22,'perisur');

INSERT INTO viaje VALUES(33,'barranca');

INSERT INTO viaje VALUES(44,'lomas');

INSERT INTO viaje VALUES(55,'reforma');

INSERT INTO viaje VALUES(66,'insurgentes');

INSERT INTO viaje VALUES(77,'chapultepec');

-- Tuplas de la tabla ruta\_viaje

INSERT INTO ruta\_viaje VALUES(11, 11);

INSERT INTO ruta\_viaje VALUES(22, 21);

INSERT INTO ruta\_viaje VALUES(33, 21);

INSERT INTO ruta\_viaje VALUES(44, 61);

INSERT INTO ruta\_viaje VALUES(55, 61);

INSERT INTO ruta\_viaje VALUES(66, 51);

INSERT INTO ruta\_viaje VALUES(77, 51);

Paso 2:

Ingresa al DBMS SQLServer, crea una nueva base de datos llamada **UBER**, ejecuta las sentencias SQL incluidas en los dos archivos mencionados, y posteriormente responde las siguientes preguntas elaborando las consultas que correspondan.

Después de resolver cada consulta, copia y pega la sentencias *de SQL*  y la tabla de resultados después de cada pregunta.

1. Mostrar los nombres (ordenados Z .. A) de los conductores sin repeticiones y la columna se debe llamar nombreCond.
2. Obtener el id del conductor, su nombre y los id de las rutas en las que trabaja.
3. Listar el (o los) id del conductor (es) que trabaja (n) en Perisur.
4. Los Ids de las rutas con el mayor número de viajes.
5. Obtener el promedio de viajes del conductor PEPE.
6. Listar el modelo de los autos que tengan viajes en Chapultepec o reforma, pero no en lomas.
7. Mostrar el id de las rutas que tienen más de un viaje registrado.
8. Obtener el nombre de los conductores que tienen viajes tanto a Perisur como a barranca (a ambos, no sólo a uno u otro).