Álgebra relacional:

DUEÑO (id_dueño, nombre, teléfono, dirección, dni)

CHOFER (id_chofer, nombre, tel, dir, f_licencia_desde, f_licencia_hasta, dni)

AUTO (patente, id_dueño, id_chofer, marca, modelo, año)

- Hallar el DNI, nombre y teléfono de todos los dueños -que no sean choferes- y que solamente poseen vehículos marca "FIAT".

D_CHOFER <- π (id_dueño, nombre, teléfono, dirección, dni) DUEÑO |X| CHOFER

D_NO_CHOFER <- D_CHOFER - DUEÑO

FIAT <- σ (marca = “FIAT”) (AUTO)

DISTINTO_A_FIAT <- σ (marca != “FIAT”) (AUTO)

D_FIAT <- D_NO_CHOFER |X| FIAT

D_OTROS <- D_NO_CHOFER |X| DISTINTO_A_FIAT

D_SOLO_FIAT <- D_FIAT - D_OTROS

OJO, no hacer con AUTO directamente porque voy a terminan borrando todo.

MySQL:

DUEÑO (id_dueño, nombre, teléfono, dirección, dni)

CHOFER (id_chofer, nombre, tel, dir, f_licencia_desde, f_licencia_hasta, dni)

AUTO (patente, id_dueño, id_chofer, marca, modelo, año)

- Utilizando el esquema presentado en el ejercicio de Álgebra Relacional, escriba un Store Procedure en MySQL llamado "nuevo_chofer". Este debe declarar como parámetro un dni y 2 fechas (licencia_desde y licencia_hasta), y debe generar, dentro de una transacción, un nuevo chofer con los datos recuperados del dueño con dni igual al recibido como parámetro. Asuma que este dni ya existe en DUEÑO (esquema del ej .de AR).

```
DELIMITER //
CREATE PROCEDURE agregarAppointment
        (IN new_dni INT, IN f_l_d DATE, IN f_l_h DATE)
BEGIN
   START TRANSACTION;

   SELECT nombre, tel, dir
   INTO new_nombre, new_tel, new_dir
   FROM dueño
   WHERE dni = new_dni

   INSERT INTO chofer (nombre, tel, dir, f_licencia_desde, f_licencia_hasta, dni)
        VALUES (new_nombre, new_tel, new_dir, f_l_d, f_l_h, new_dni);

   COMMIT;
END;
// DELIMITER ;
```

Id_chofer no se pone al ser autoincremental