UNIVERSIDAD DE BUENOS AIRES

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Bases de Datos

Trabajo Práctico 1

9 de agosto de 2015

Integrante	LU	Correo electrónico
Maurizio, Miguel Sebastián	**635/11**	miguelmaurizio.92@gmail.com
Prillo, Sebastián	**616/11**	sebastianprillo@gmail.com
Tagliavini Ponce, Guido	**783/11**	guido.tag@gmail.com

Índice

1. Introducción

En este trabajo práctico diseñamos e implementamos una base de datos para el Registro Único de Accidentes de Tránsito (RUAT), sistema que está que está preparando el Gobierno Nacional, para registrar y analizar información sobre accidentes e infracciones de tránsito ocurridos en el país.

Por un lado, el sistema registra todos los datos relacionados con siniestros de tránsito, lo cual abarca:

- vehículos involucrados;
- conductores involucrados;
- testigos;
- localización;
- modalidad del siniestro (atropello, vuelco, etc.);
- tipo de colisión;
- denuncia radicada por el hecho;
- estudios y peritajes hechos.

Por otro lado, registra infracciones de tránsito, y más específicamente:

- vehículo involucrado;
- conductor involucrado;
- localización;
- tipo de infracción;

Además, el sistema registra datos sobre los vehículos, personas y las vías nacionales. Sobre los vehículos, almacena:

- categoría de coche (gama media, gama alta, etc.);
- tipo de vehículo (auto, camión, moto, etc.);
- seguro automotor y su tipo;

Sobre las personas, almacena:

- datos personales;
- autos de los cuales es dueña;
- cedulas (verdes y azules) que posee;
- licencias de conducir que posee;
- antecedentes penales;

Finalmente, sobre las vías nacionales, el sistema almacena:

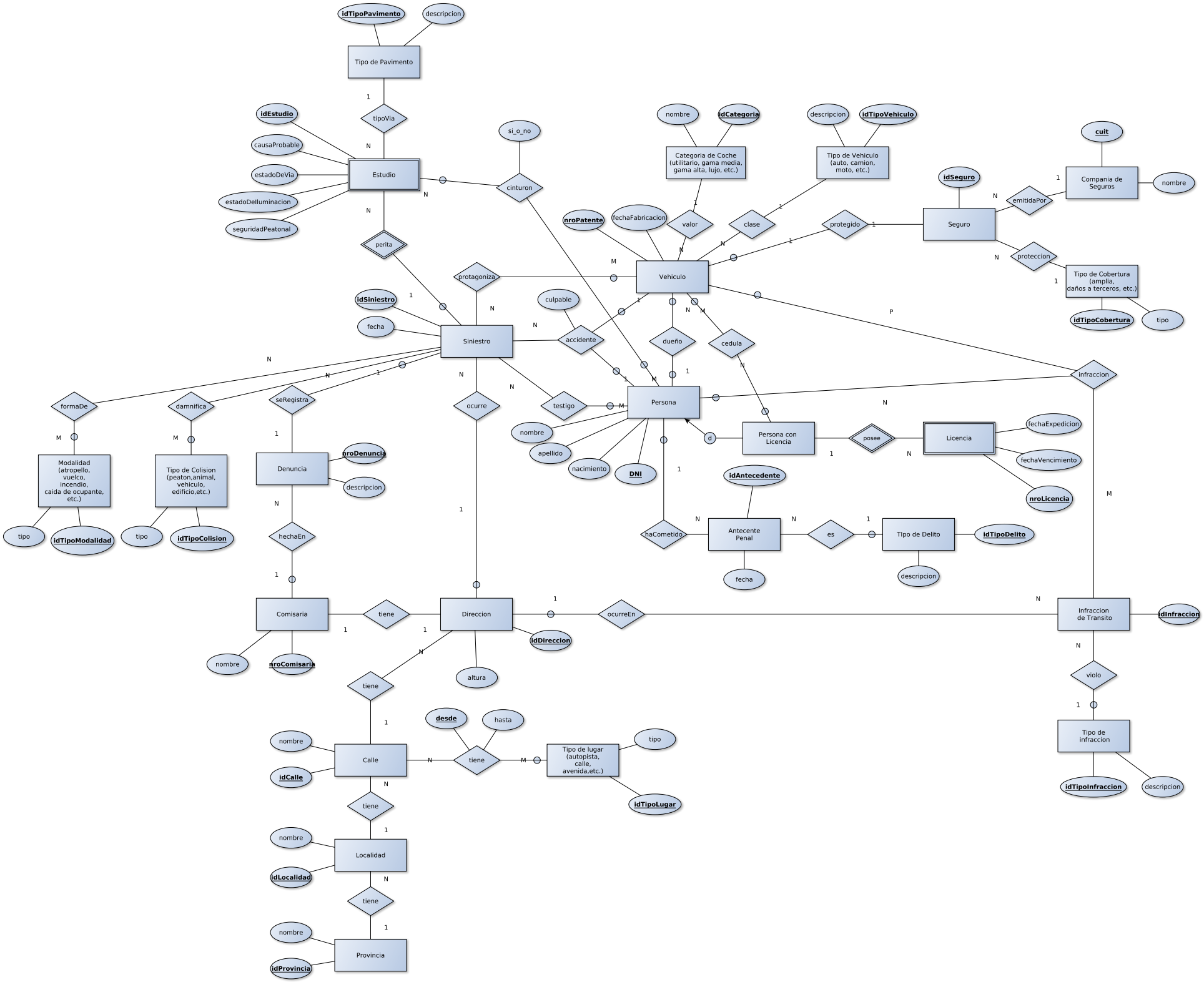
- tipo de vía según el tramo (calle, avenida, autopista, etc.);
- extensión del tramo;

La elicitación de estos requerimientos proviene, principalmente, de la lectura del enunciado del trabajo práctico, que contiene toda esta información.

En las subsiguientes secciones presentamos cada una de las etapas del diseño y la implementación: el DER, su transformación al MR, y la implementación de queries y triggers. Para la implementación, decidimos utilizar el motor *SQLite*.

2. Diagrama Entidad Relación

A partir de los requerimientos recabados, elaboramos un diagrama de entidad-relación, que presentamos a continuación.


idTipoPavimento
descripcion
Tipo de Pavimento
tipoVia
idEstudio
causaProbable
estadoDeVia
estadoDelluminacion
seguridadPeatonal
Estudio
perita
si_o_no
cinturon
nombre
idCategoria
Categoria de Coche (utilitario, gama media, gama alta, lujo, etc.)
descripcion
idTipoVehiculo
Tipo de Vehiculo (auto, camion, moto, etc.)
nroPatente
fechaFabricacion
valor
clase
protegido
idSeguro
Seguro
cuit
Compania de Seguros
nombre
emitidaPor
proteccion
Tipo de Cobertura (amplia, daños a terceros, etc.)
idTipoCobertura
tipo
protagoniza
Vehiculo
idSiniestro
fecha
Siniestro
culpable
accidente
dueño
cedula
infraccion
formaDe
damnifica
seRegistra
ocurre
testigo
Persona
nombre
apellido
nacimiento
DNI
Persona con Licencia
posee
Licencia
fechaExpedicion
fechaVencimiento
nroLicencia
Modalidad (atropello, vuelco, incendio, caida de ocupante, etc.)
tipo
idTipoModalidad
Tipo de Colision (peaton,animal, vehiculo, edificio,etc.)
tipo
idTipoColision
Denuncia
nroDenuncia
descripcion
hechaEn
haCometido
Antecente Penal
idAntecedente
fecha
es
TIpo de Delito
idTipoDelito
descripcion
Comisaria
nombre
nroComisaria
tiene
Direccion
idDireccion
altura
ocurreEn
Infraccion de Transito
idInfraccion
violo
Tipo de infraccion
idTipoInfraccion
descripcion
tiene
Calle
nombre
idCalle
desde
hasta
tiene
Tipo de lugar (autopista, calle, avenida,etc.)
tipo
idTipoLugar
tiene
Localidad
nombre
idLocalidad
tiene
Provincia
nombre
idProvincia

2.1. Restricciones en lenguaje natural

Las restricciones que el DER no puede capturar son las siguientes:

1. Las personas que aparecen relacionadas con el estudio en la relacion *cinturon*, deben ser personas que participaron del siniestro correspondiente al estudio, conduciendo uno de los vehículos de ese siniestro.
2. Si un vehículo sin conductor forma parte de un siniestro (es decir, está relacionado con un siniestro vía la relación binaria *protagoniza*), entones no era conducido por nadie en ese siniestro (es decir, no aparece en la relación ternaria *accidente* con ese siniestro).
3. No hay solapamiento entre los rangos de alturas de las calles. Por ejemplo, no puede ser que Monroe del 0 al 3000 sea calle y del 2500 al 4000 sea avenida.

3. Modelo Relacional

A partir del DER, derivamos el siguiente modelo relacional. Lo presentamos directamente en sintaxis SQL.

Algunas de las relaciones N-M del DER tenían nombres poco ilustrativos al ser sacados de contexto. Por ejemplo, la relación binaria entre *Siniestro* y *Tipo de Colisión* fue llamada *damnifica*, nombre que no tiene mucho sentido aisladamente. Esto representaba un problema, dado que en, el MR, esas relaciones pasan a ser tablas. En estos casos, decidimos transformar el nombre de estas relaciones, usando los nombres de las entidades relacionadas, del siguiente modo. Una relación de nombre R entre dos entidades de nombres X e Y, se pasa a llamar X_R_Y. Por ejemplo, en el caso de la relación *damnifica*, el nombre de la tabla asociada pasó a ser *siniestro_damnifica_tipo_de_colision*. En algunos casos, los nombres eran suficientemente autoexplicativos, y no fue necesario hacer este cambio. Incluimos, arriba de cada sentencia `CREATE TABLE`, la entidad o relación correspondiente a la tabla creada.

```
-- entidad Provincia
CREATE TABLE provincia (
 idProvincia INTEGER NOT NULL,
 nombre VARCHAR(255) NOT NULL,
 PRIMARY KEY(idProvincia)
);

-- entidad Localidad
CREATE TABLE localidad (
 idLocalidad INTEGER NOT NULL,
 nombre VARCHAR(255) DEFAULT NULL,
 idProvincia INTEGER NOT NULL,
 PRIMARY KEY(idLocalidad),
 FOREIGN KEY(idProvincia) REFERENCES provincia(idProvincia)
);

-- entidad Calle
CREATE TABLE calle (
 idCalle INTEGER NOT NULL,
 nombre VARCHAR(255) DEFAULT NULL,
 idLocalidad INTEGER NOT NULL,
 PRIMARY KEY(idCalle),
 FOREIGN KEY(idLocalidad) REFERENCES localidad(idLocalidad)
);
```

```
-- entidad Direccion
CREATE TABLE direccion (
 idDireccion INTEGER NOT NULL,
 altura INTEGER NOT NULL,
 idCalle INTEGER NOT NULL,
 PRIMARY KEY(idDireccion),
 FOREIGN KEY(idCalle) REFERENCES calle(idCalle)
);

-- entidad Tipo de Lugar
CREATE TABLE tipo_de_lugar (
 idTipoLugar INTEGER NOT NULL,
 tipo VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(idTipoLugar)
);

-- relacion tiene, entre Calle y Tipo de Lugar
CREATE TABLE calle_tiene_tipo_de_lugar (
 idCalle INTEGER NOT NULL,
 idTipoLugar INTEGER NOT NULL,
 desde INTEGER NOT NULL,
 hasta INTEGER NOT NULL,
 PRIMARY KEY(idCalle, idTipoLugar, desde),
 FOREIGN KEY(idCalle) REFERENCES calle(idCalle),
 FOREIGN KEY(idTipoLugar) REFERENCES tipo_de_lugar(idTipoLugar)
);

-- entidad Comisaria
CREATE TABLE comisaria (
 nroComisaria INTEGER NOT NULL,
 nombre VARCHAR(255) DEFAULT NULL,
 idDireccion INTEGER NOT NULL,
 PRIMARY KEY(nroComisaria),
 FOREIGN KEY(idDireccion) REFERENCES direccion(idDireccion)
);

-- entidad Denuncia
CREATE TABLE denuncia (
 nroDenuncia INTEGER NOT NULL,
 descripcion VARCHAR(255) DEFAULT NULL,
 nroComisaria INTEGER NOT NULL,
 PRIMARY KEY(nroDenuncia),
 FOREIGN KEY(nroComisaria) REFERENCES comisaria(nroComisaria)
);

-- entidad Tipo de Colision
CREATE TABLE tipo_de_colision (
 idTipoColision INTEGER NOT NULL,
 tipo VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(idTipoColision)
```

```
);

-- entidad Modalidad
CREATE TABLE modalidad (
 idTipoModalidad INTEGER NOT NULL,
 tipo VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(idTipoModalidad)
);

-- entidad Siniestro
CREATE TABLE siniestro (
 idSiniestro INTEGER NOT NULL,
 fecha DATETIME DEFAULT NULL,
 nroDenuncia INTEGER NOT NULL,
 idDireccion INTEGER NOT NULL,
 PRIMARY KEY(idSiniestro),
 FOREIGN KEY(nroDenuncia) REFERENCES denuncia(nroDenuncia),
 FOREIGN KEY(idDireccion) REFERENCES direccion(idDireccion)
);

-- relacion damnifica, entre Siniestro y Tipo de Colision
CREATE TABLE siniestro_damnifica_tipo_de_colision (
 idSiniestro INTEGER NOT NULL,
 idTipoColision INTEGER NOT NULL,
 PRIMARY KEY(idSiniestro, idTipoColision),
 FOREIGN KEY(idSiniestro) REFERENCES siniestro(idSiniestro),
 FOREIGN KEY(idTipoColision) REFERENCES tipo_de_colision(idTipoColision)
);

-- relacion formaDe, entre Siniestro y Modalidad
CREATE TABLE siniestro_forma_de_modalidad (
 idSiniestro INTEGER NOT NULL,
 idTipoModalidad INTEGER NOT NULL,
 PRIMARY KEY(idSiniestro, idTipoModalidad),
 FOREIGN KEY(idSiniestro) REFERENCES siniestro(idSiniestro),
 FOREIGN KEY(idTipoModalidad) REFERENCES modalidad(idTipoModalidad)
);

-- entidad Tipo de Pavimento
CREATE TABLE tipo_de_pavimento (
 idTipoPavimento INTEGER NOT NULL,
 descripcion VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(idTipoPavimento)
);

-- entidad Estudio
CREATE TABLE estudio (
 idEstudio INTEGER NOT NULL,
 causaProbable VARCHAR(255) DEFAULT NULL,
 estadoVia VARCHAR(255) DEFAULT NULL,
 estadoIluminacion VARCHAR(255) DEFAULT NULL,
```

```
 seguridadPeatonal BOOLEAN DEFAULT NULL,
 idTipoPavimento INTEGER NOT NULL,
 idSiniestro INTEGER NOT NULL,
 PRIMARY KEY (idEstudio, idSiniestro),
 FOREIGN KEY(idTipoPavimento) REFERENCES tipo_de_pavimento(idTipoPavimento),
 FOREIGN KEY(idSiniestro) REFERENCES siniestro(idSiniestro)
);

-- entidad Persona
CREATE TABLE persona (
 dni INTEGER NOT NULL,
 nombre VARCHAR(255) NOT NULL,
 apellido VARCHAR(255) NOT NULL,
 nacimiento DATE NOT NULL,
 PRIMARY KEY(dni)
);

-- relacion testigo, entre Siniestro y Persona
CREATE TABLE siniestro_testigo_persona (
 idSiniestro INTEGER NOT NULL,
 dni INTEGER NOT NULL,
 PRIMARY KEY(idSiniestro, dni),
 FOREIGN KEY(idSiniestro) REFERENCES siniestro(idSiniestro),
 FOREIGN KEY(dni) REFERENCES persona(dni)
);

-- relacion cinturon, entre Estudio y Persona
CREATE TABLE estudio_cinturon_persona (
 idEstudio INTEGER NOT NULL,
 dni INTEGER NOT NULL,
 si_o_no BOOLEAN NOT NULL,
 PRIMARY KEY(idEstudio, dni),
 FOREIGN KEY(idEstudio) REFERENCES estudio(idEstudio),
 FOREIGN KEY(dni) REFERENCES persona(dni)
);

-- entidad Tipo de Delito
CREATE TABLE tipo_de_delito (
 idTipoDelito INTEGER NOT NULL,
 descripcion VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(idTipoDelito)
);

-- entidad Antecedente Penal
CREATE TABLE antecedente_penal (
 idAntecedente INTEGER NOT NULL,
 fecha Date NOT NULL,
 dni INTEGER NOT NULL,
 idTipoDelito INTEGER NOT NULL,
 FOREIGN KEY(dni) REFERENCES persona(dni),
 FOREIGN KEY(idTipoDelito) REFERENCES tipo_de_delito(idTipoDelito),
```

```
 PRIMARY KEY(idAntecedente)
);

-- entidad Tipo de Infraccion
CREATE TABLE tipo_de_infraccion (
 idTipoInfraccion INTEGER NOT NULL,
 descripcion VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(idTipoInfraccion)
);

-- entidad Infraccion de Transito
CREATE TABLE infraccion_de_transito (
 idInfraccion INTEGER NOT NULL,
 idDireccion INTEGER NOT NULL,
 idTipoInfraccion INTEGER NOT NULL,
 FOREIGN KEY(idDireccion) REFERENCES direccion(idDireccion),
 FOREIGN KEY(idTipoInfraccion) REFERENCES tipo_infraccion(idTipoInfraccion),
 PRIMARY KEY(idInfraccion)
);

-- entidad Persona con Licencia
CREATE TABLE persona_con_licencia (
 dni INTEGER NOT NULL,
 FOREIGN KEY(dni) REFERENCES persona(dni),
 PRIMARY KEY(dni)
);

-- entidad Licencia
CREATE TABLE licencia (
 nroLicencia INTEGER NOT NULL,
 dni INTEGER NOT NULL,
 expedicion DATE NOT NULL,
 expiracion DATE NOT NULL,
 FOREIGN KEY(dni) REFERENCES persona_con_licencia(dni),
 PRIMARY KEY (nroLicencia, dni)
);

-- entidad Compania de Seguros
CREATE TABLE compania_de_seguro (
 cuit INTEGER NOT NULL,
 nombre VARCHAR(255) NOT NULL,
 PRIMARY KEY(cuit)
);

-- entidad Tipo de Cobertura
CREATE TABLE tipo_de_cobertura (
 idTipoCobertura INTEGER NOT NULL,
 descripcion VARCHAR(255) NOT NULL,
 PRIMARY KEY(idTipoCobertura)
);
```

```
-- entidad Tipo de Vehiculo
CREATE TABLE tipo_de_vehiculo (
 idTipoVehiculo INTEGER NOT NULL,
 descripcion VARCHAR(255) NOT NULL,
 PRIMARY KEY(idTipoVehiculo)
);

-- entidad Categoria de Coche
CREATE TABLE categoria_de_vehiculo (
 idCategoria INTEGER NOT NULL,
 descripcion VARCHAR(255) NOT NULL,
 PRIMARY KEY(idCategoria)
);

-- entidad Vehiculo
CREATE TABLE vehiculo (
 nroPatente CHARACTER(6) NOT NULL,
 fechaFabricacion DATE NOT NULL,
 idCategoria INTEGER NOT NULL,
 idTipoVehiculo INTEGER NOT NULL,
 dni INTEGER NOT NULL,
 FOREIGN KEY(idCategoria) REFERENCES categoria_de_vehiculo(idCategoria),
 FOREIGN KEY(idTipoVehiculo) REFERENCES tipo_de_vehiculo(idTipoVehiculo),
 FOREIGN KEY(dni) REFERENCES persona(dni),
 PRIMARY KEY(nroPatente)
);

-- entidad Seguro
CREATE TABLE seguro (
 idSeguro INTEGER NOT NULL,
 cuit INTEGER NOT NULL,
 idTipoCobertura INTEGER NOT NULL,
 nroPatente INTEGER NOT NULL,
 FOREIGN KEY(cuit) REFERENCES compania_de_seguro(cuit),
 FOREIGN KEY(idTipoCobertura) REFERENCES cobertura(idTipoCobertura),
 FOREIGN KEY(nroPatente) REFERENCES vehiculo(nroPatente),
 PRIMARY KEY(idSeguro)
);

-- relacion cedula, entre Vehiculo y Persona con Licencia
CREATE TABLE cedula (
 nroPatente CHARACTER(6) NOT NULL,
 dni INTEGER NOT NULL,
 FOREIGN KEY(nroPatente) REFERENCES vehiculo(nroPatente),
 FOREIGN KEY(dni) REFERENCES persona_con_licencia(dni),
 PRIMARY KEY(nroPatente, dni)
);

-- relacion protagoniza, entre Siniesto y Vehiculo
CREATE TABLE siniestro_protagoniza_vehiculo (
 nroPatente CHARACTER(6) NOT NULL,
```

```
 idSiniestro INTEGER NOT NULL,
 FOREIGN KEY(nroPatente) REFERENCES vehiculo(nroPatente),
 FOREIGN KEY(idSiniestro) REFERENCES siniestro(idSiniestro),
 PRIMARY KEY(nroPatente, idSiniestro)
);

-- relacion accidente, entre Siniestro, Vehiculo y Persona
CREATE TABLE siniestro_vehiculo_persona (
 nroPatente CHARACTER(6) NOT NULL,
 idSiniestro INTEGER NOT NULL,
 dni INTEGER NOT NULL,
 culpable BOOLEAN NOT NULL,
 FOREIGN KEY(nroPatente) REFERENCES vehiculo(nroPatente),
 FOREIGN KEY(idSiniestro) REFERENCES siniestro(idSiniestro),
 FOREIGN KEY(dni) REFERENCES persona(dni),
 PRIMARY KEY(nroPatente, idSiniestro)
);

-- relacion infraccion, entre Persona, Vehiculo e Infraccion de Transito
CREATE TABLE persona_en_vehiculo_comete_infraccion (
 nroPatente CHARACTER(6) NOT NULL,
 dni INTEGER NOT NULL,
 idInfraccion INTEGER NOT NULL,
 FOREIGN KEY(nroPatente) REFERENCES vehiculo(nroPatente),
 FOREIGN KEY(dni) REFERENCES persona(dni),
 FOREIGN KEY(idInfraccion) REFERENCES infraccion_de_transito(idInfraccion),
 PRIMARY KEY(nroPatente, dni, idInfraccion)
);
```

4. Consultas SQL

Dos eran las funcionalidades que esperaban poder realizarse. La primera de ellas pedía lo siguiente:

Obtener, con un número de licencia específico, información sobre los accidentes en los que ha participado el conductor propietario de la misma, con detalles de fecha, lugar, tipo de accidente, participación y modalidad. También se deberá indicar la cantidad de automóviles que está habilitado a conducir.

La consulta SQL correspondiente es la siguiente:

```
SELECT s.fecha, c.nombre, d.altura, lo.nombre, pr.nombre, m.tipo, tc.tipo
FROM siniestro_vehiculo_persona svp,
        persona p,
        persona_con_licencia pl,
        licencia l,
        siniestro s,
        siniestro_forma_de_modalidad f,
        modalidad m,
        direccion d,
        calle c,
        localidad lo,
        provincia pr,
        siniestro_damnifica_tipo_de_colision dam,
```

```
        tipo_de_colision tc
WHERE l.nroLicencia = 0 -- la licencia del Chano
-- join entre licencia, persona_con_licencia, persona y siniestro_vehiculo_persona
AND svp.dni = p.dni
AND p.dni = pl.dni
AND pl.dni = l.dni
-- join entre modalidad, siniestro_forma_de_modalidad, siniestro y siniestro_vehiculo_persona
AND s.idSiniestro = svp.idSiniestro
AND f.idSiniestro = s.idSiniestro
AND f.idTipoModalidad = m.idTipoModalidad
-- join entre siniestro, direccion, calle, localidad y provincia
AND s.idDireccion = d.idDireccion
AND d.idCalle = c.idCalle
AND c.idLocalidad = lo.idLocalidad
AND lo.idProvincia = pr.idProvincia
-- join entre siniestro, siniestro_damnifica_tipo_de_colision y tipo_de_colision
AND s.idSiniestro = dam.idSiniestro
AND dam.idTipoColision = tc.idTipoColision;
```

El segundo requerimiento era:

Obtener, dada una modalidad de accidente (atropello, vuelco, incendio, caída del ocupante, etc), un listado de licencias de conducir y la cantidad de veces que cada una de estas licencias incurrió en la modalidad consultada.

En esta oportunidad, decidimos abreviar la consulta utilizando el comando `JOIN`, en lugar de hacer las juntas manualmente. La consulta que resuelve el problema es:

```
SELECT m.idTipoModalidad, j.nroLicencia, j.dni, COUNT(*)
FROM (modalidad m JOIN
        siniestro_forma_de_modalidad sm JOIN
        siniestro_vehiculo_persona svp JOIN
        persona_con_licencia pl JOIN
        licencia l) j
WHERE m.idTipoModalidad = 0
GROUP BY j.nroLicencia;
```

5. Triggers

Realizamos un trigger para verificar una de las restricciones en castellano. Específicamente:

Los conductores que pertenecen a un estudio deben ser conductores involucrados en el siniestro de ese estudio.

```
CREATE TRIGGER conductores_estudiados_en_siniestro BEFORE INSERT ON estudio_cinturon_persona
BEGIN
SELECT CASE
WHEN (NOT EXISTS (SELECT *
            FROM siniestro_vehiculo_persona svp
            WHERE svp.dni = NEW.dni AND svp.idSiniestro =
                      (SELECT e.idSiniestro FROM estudio e WHERE e.idEstudio = NEW.idEstudio)))
THEN RAISE(ABORT, 'El conductor no estuvo involucrado en el siniestro del estudio.')
END;
END;
```

6. Conclusión

El ejercicio del diseño una base de datos de una escala considerable, nos dejó algunas enseñanzas. Por un lado, nos permitió adquirir mayor confianza en el trabajo con el modelado de problemas vía bases de datos relacionales, y la implementación en SQL. Pero más importante, nos permitió sentir en carne propia las dificultades del diseño, que hacen que de esto un proceso iterativo, en el cual sólo se llega a la solución final a través de refinamientos. En nuestro caso particular, la confección del DER la desarrollamos a lo largo de varias semanas, realizando sucesivas veces el antedicho refinamiento. Gracias a esto, la implementación fue fluida, sin mayores inconvenientes, más allá de los del aprendizaje del lenguaje de consultas.

7. Anexo: implementación

En esta sección indicamos en qué archivo está implementada cada una de las partes de la base de datos.

El archivo `tablas.sql` contiene las instrucciones para crear las tablas e insertar los datos. Los archivos `query1.sql` y `query2.sql` contienen las dos queries implementadas. Finalmente, el archivo `triggers.sql` contiene el trigger implementado.

Toda la implementación fue probada utilizando SQLite.