# Bases de Datos

Segundo cuatrimestre 2018

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

## Trabajo Práctico 1

Ciudad Gótica



| Integrante | LU | Correo electrónico |
|---|---|---|
| Kapobel Rodrigo | 695/12 | rok_35@live.com.ar |
| Juan Cruz Sosa | 733/12 | nirvguy@gmail.com |
| Augusto Beccar García | 267/13 | abg101@gmail.com |
| Nicolas Hernandez | 122/12 | nicoh22@hotmail.com |



**Palabras claves:** MER, MR, Postgres, Triggers.

**Reservado para la cátedra**

| Instancia | Docente | Nota |
|---|---|---|
| Primera entrega | | |
| Segunda entrega | | |

# Índice

# 1. Introducción

En este trabajo práctico se captura la realidad de la ciudad gótica definida por enunciado y se la implementa en una base de datos relacional. Para esto primero se diseñó el MER (Modelo Entidad Relación) y el MR (Modelo Relacional) y se definen las restricciones y asunciones a tener en cuenta. En base a lo mencionado se procede implementar la base de datos en Postgres SQL. Las restricciones serán implementadas mediante function triggers.

Como parte del enunciado, también, se pide responder a una serie de consultas, las cuales influyen intrinsecamente en el diseño realizado, ya que debemos poder responder a las mismas. Las consultas se listan a continuación:

- Listado de incidentes en un rango de fechas, mostrando los datos de las per- sonas y policías involucrados con el rol que jugó cada uno en el incidente
- Dada una organización delictiva, el detalle de incidentes en que participaron las personas que componen dicha organización
- La lista de todos los oficiales con sus rangos, de un departamento dado.
- El ranking de oficiales que participaron en más incidentes
- Los barrios con mayor cantidad de incidentes.
- Todos los oficiales sumariados que participaron de algún incidente.
- Las personas involucradas en incidentes ocurridos en el barrio donde viven
- Los superheroes que tienen una habilidad determinada
- Los superheroes que han participado en algún incidente.
- Listado de todos los incidentes en donde estuvieron involucrados superheroes y fueron causados por los .ªrchienemigos"de los superheroes involucrados.

# 2. Desarrollo

## 2.1. MER

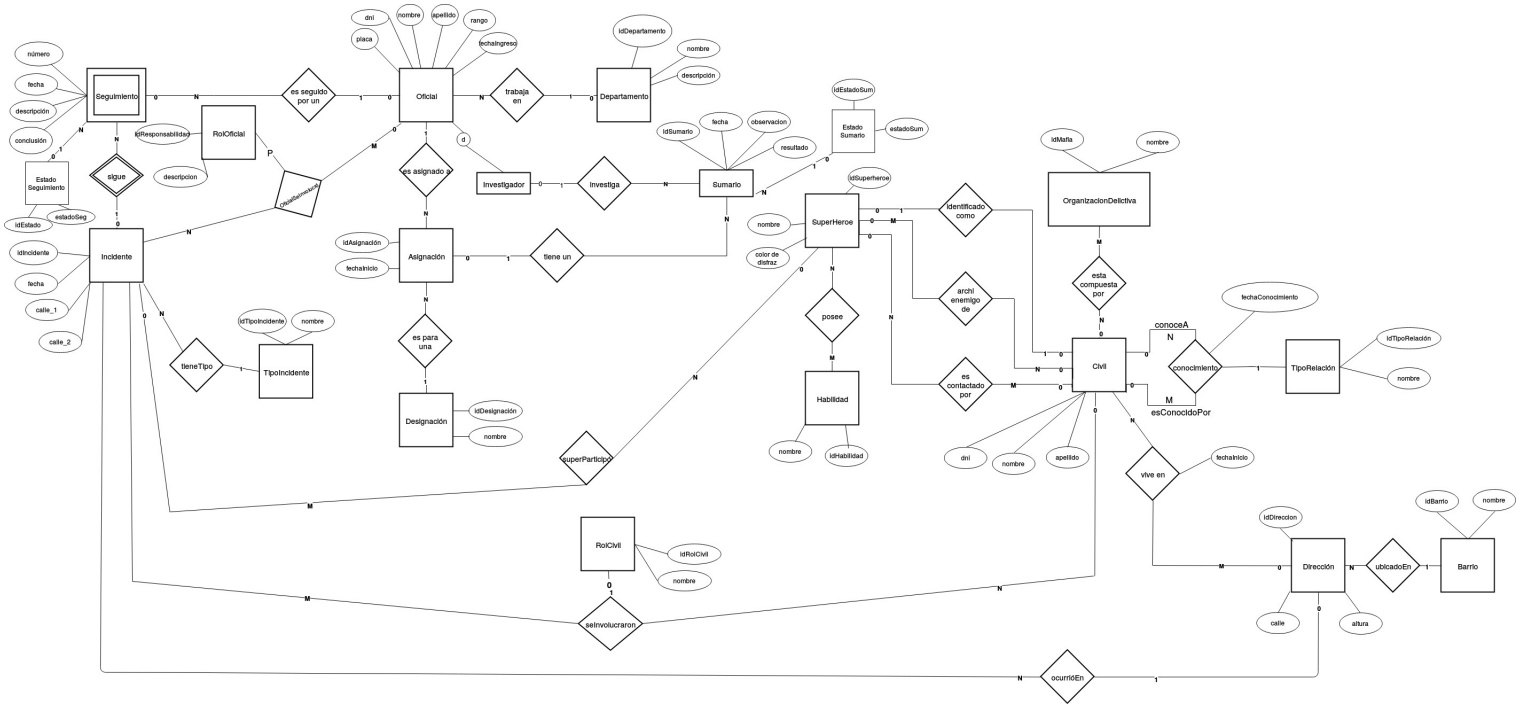


Figura 1: Ciudad Gótica

## 2.2. Modelo Relacional

### 2.2.1. Entidades

Civil(dni, nombre, apellido)

Superheroe(idSuperheroe, nombre, color de disfraz, dni)

Habilidad(idHabilidad, nombre)

OrganizacionDelictiva(idMafia, nombre)

Direccion(idDireccion, calle, altura, idBarrio)

Barrio(idBarrio, nombre)

TipoRelacion(idTipoRelacion, nombre)

Incidente(idIncidente, fecha, calle_1, calle_2, idTipoIncidente, idDireccion)

TipoIncidente(idTipoIncidente, nombre)

Seguimiento(numero, fecha, descripción, conclusión, estado,idIncidente, placa, idEstadoSeg)

EstadoSeguimiento(idEstadoSeg, estado)

Departamento(idDepartamento, nombre, descripción)

Oficial(placa,dni, nombre, apellido, rango, fechaIngreso, tipo, idDepartamento)

RolOficial(idResponsabilidad, descripción)

RolCivil(idRolCivil, nombre)

Asignación(idAsignación, fechaInicio, idDesignación, placa)

Sumario(idSumario, fecha, observación, resultado, placa, idAsignación, idEstadoSum)

EstadoSumario(idEstadoSum, estado)

Designacion(idDesignación, nombre)

#### 2.2.2. Interrelaciones

OficialSeInvolucró(placa, idIncidente, idResponsabilidad)

Posee(idSuperheroe, idHabilidad)

ArchienemigoDe(idSuperheroe,dni)

EsContactadoPor(idSuperheroe,dni)

SuperParticipó(idSuperheroe, idIncidente)

EstaCompuestaPor(idMafia, dni)

Conocimiento(conocedor,conocido, fechaConocimiento, idTipoDeRelación)

seInvolucraron(dni, idIncidente, idRolCivil)

ViveEn(dni, idDirección, fechaInicio)

#### 2.2.3. Consideraciones adicionales

Como en la relación *identificadoComo* ambas entidades participan parcialmente, decidimos poner la foreign key en Super Héroe, asumiendo que esto nos generaría menos elementos nulos en la base de datos.

#### 2.2.4. Restricciones del modelo

- **Asignación**:
  - Fecha de inicio mayor o igual a fecha de ingreso del oficial.
  - Asumimos que solo puede tener una asignación a una designación al mismo tiempo (misma fecha inicio). Cuando termina una asignación, un Oficial empieza otra asignación en otra.
- **Sumario:**
  - Fecha del sumario mayor o igual a fecha de ingreso del oficial a investigar.
  - Investigador del sumario no puede investigarse a si mismo.
  - Fecha del sumario mayor o igual a la fecha de inicio de la asignación.
  - Si el estado es concluido, entonces tiene un resultado.
- **Oficial:**
  - *dni* únicos.
  - Si se involucra en un incidente, la fecha de este último tiene que ser mayor o igual a la fecha de ingreso del oficial.
- **Seguimiento:**
  - Solo puede ser seguida si su estado es *en proceso*.
  - Si es seguida, lo es por un Oficial cuya fecha de ingreso sea menor o igual a la fecha del seguimiento.
  - Fecha del seguimiento mayor a fecha del incidente del relacionado.
  - Una vez que un proceso está en estado *cerrado* no puede cambiar de estado y deja de estar seguido por un oficial.

- La conclusion solo tiene un valor si el estado del seguimiento es cerrado.

- **Rol Oficial:**
  - Asumimos que un oficial en un incidente puede cumplir muchos roles. Por eso es $P$ su cardinalidad en la ternaria entre esta, Oficial e Incidente de la interrelacion $OficialSeInvolucro$.

- **Super héroe:**
  - No puede ser archienemigo de la misma persona de la que es *identificado como*.
  - No puede ser identificado como una persona que pertence a una organizacion delictiva.
  - No puede participar como super héroe en el incidente al mismo tiempo que su correspondiente civil (si es que se conoce su identidad) se involucra como civil.

- **Desambiguación Civil/Persona:**
  - Elejimos llamar Civil a tal entidad porque entendemos que los oficiales, son personas, pero en nuestro modelo no serán Civiles. Creemos que Civil representa el aspecto cívico de un habitante de ciudad Gótica con sus propias interrelaciones y atributos. En este modelo asumimos que no va a existir un oficial que sea civil. Por lo que estas entidades no comparten *dni*.
  - En este modelo simplificamos para que un Civil no pueda volver a vivir en una dirección.

## 2.3. Consultas

### 2.3.1. Query 1

Listado de incidentes en un rango de fechas, mostrando los datos de las personas y policías involucrados con el rol que jugó cada uno en el incidente.

```
SELECT i."idIncidente" as "idIncidente",
   i.fecha,
   i.calle_1 as calle_1,
   i.calle_2 as calle_2,
'Oficial' as tipo,
        o.placa as "placaOficial",
o.dni as dni,
o.nombre as nombre,
o.apellido as apellido,
ro.descripcion as "rol"
FROM tp1."Incidente" i
JOIN tp1."OficialSeInvolucro" osi ON i."idIncidente" = osi."idIncidente"
JOIN tp1."Oficial" o ON osi.placa = o.placa
JOIN tp1."RolOficial" ro ON osi."idResponsabilidad" = ro."idResponsabilidad"
WHERE i.fecha < '2018-10-07' and i.fecha > '2018-09-15'
UNION
SELECT i."idIncidente" AS "idIncidente",
   i.fecha,
   i.calle_1 as calle_1,
   i.calle_2 as calle_2,
'Civil' as tipo,
        NULL as "placaOficial",
c.dni as dni,
c.nombre as nombre,
c.apellido as apellido,
rc.nombre as "rol"
FROM tp1."Incidente" i
```

```
JOIN tp1."SeInvolucraron" si ON i."idIncidente" = si."idIncidente"
JOIN tp1."Civil" c ON si.dni = c.dni
JOIN tp1."RolCivil" rc ON si."idRolCivil" = rc."idRolCivil"
WHERE i.fecha < '2018-10-07' and i.fecha > '2018-09-15'
ORDER BY 1 ASC, 5 ASC;
```

#### 2.3.2. Query 2

Dada una organización delictiva, el detalle de incidentes en que participaron las personas que componen dicha organización

```
SELECT i."idIncidente",
       ti.nombre as "tipoIncidente",
       i.fecha as "fechaIncidente",
       i.calle_1,
       i.calle_2,
       d.calle as "departamentoCalle",
       d.altura as "alturaDepartamento",
       b.nombre as "nombreBarrio",
       c.dni as "dniCivil",
       c.nombre as "nombreCivil",
       c.apellido as "apellidoCivil",
       rc.nombre as "rolCivil"
FROM tp1."Incidente" i
JOIN tp1."TipoIncidente" ti ON i."idTipoInicidente" = ti."idTipoInicidente"
JOIN tp1."Direccion" d ON i."idDireccion" = d."idDireccion"
JOIN tp1."Barrio" b ON d."idBarrio" = b."idBarrio"
JOIN tp1."SeInvolucraron" si ON i."idIncidente" = si."idIncidente"
JOIN tp1."Civil" c ON si.dni = c.dni
JOIN tp1."RolCivil" rc ON si."idRolCivil" = rc."idRolCivil"
JOIN tp1."EstaCompuestaPor" ecp ON c.dni = ecp.dni
JOIN tp1."OrganizacionDelictiva" od ON od."idMafia" = ecp."idMafia"
WHERE od."idMafia" = 3;
```

#### 2.3.3. Query 3

La lista de todos los oficiales con sus rangos, de un departamento dado.

```
SELECT o.placa as "placaOficial",
       o.dni as "dniOficial",
       o.nombre as "nombreOficial",
       o.apellido as "apellidoOficial",
       o.rango as "oficialRango",
       o."fechaIngreso",
       o.tipo as "oficialTipo"
FROM tp1."Oficial" o
INNER JOIN tp1."Departamento" d ON d."idDepartamento" = o."idDepartamento"
WHERE d."idDepartamento" = 1;
```

#### 2.3.4. Query 4

El ranking de oficiales que participaron en más incidentes

```
SELECT o.placa, o.dni, o.nombre, o.apellido, COUNT(distinct i."idIncidente")
FROM tp1."Incidente" i
JOIN tp1."OficialSeInvolucro" oci ON oci."idIncidente" = i."idIncidente"
JOIN tp1."Oficial" o ON oci.placa = o.placa
GROUP BY o.placa, o.dni, o.nombre, o.apellido
HAVING COUNT(distinct i."idIncidente") >= ALL (SELECT COUNT(distinct i2."idIncidente")
   FROM tp1."Incidente" i2
   JOIN tp1."OficialSeInvolucro" oci2 ON oci2."idIncidente" = i2."idIncidente"
   JOIN tp1."Oficial" o2 ON oci2.placa = o2.placa
   GROUP BY o2.placa);
```

#### 2.3.5. Query 5

Los barrios con mayor cantidad de incidentes.

```
SELECT b."idBarrio", b.nombre, COUNT(distinct i."idIncidente")
FROM tp1."Incidente" i
JOIN tp1."Direccion" d ON d."idDireccion" = i."idDireccion"
JOIN tp1."Barrio" b ON d."idBarrio" = b."idBarrio"
GROUP BY b."idBarrio", b.nombre
HAVING COUNT(distinct i."idIncidente") >= ALL (SELECT COUNT(distinct i2."idIncidente") from
   FROM tp1."Incidente" i2
   JOIN tp1."Direccion" d2 ON d2."idDireccion" = i2."idDireccion"
   JOIN tp1."Barrio" b2 ON d2."idBarrio" = b2."idBarrio"
   GROUP BY b2."idBarrio");
```

#### 2.3.6. Query 6

Todos los oficiales sumariados que participaron de algún incidente.

```
SELECT distinct o.placa,
o.dni,
o.nombre,
o.apellido,
o.rango
FROM tp1."Incidente" i
JOIN tp1."OficialSeInvolucro" osi ON i."idIncidente" = osi."idIncidente"
JOIN tp1."Oficial" o ON  osi.placa = o.placa
JOIN tp1."Asignacion" a ON o.placa = a.placa
JOIN tp1."Sumario" s ON a."idAsignacion" = s."idAsignacion";
```

#### 2.3.7. Query 7

Las personas involucradas en incidentes ocurridos en el barrio donde viven

```
SELECT distinct civil.dni as dniCivil,
civil.nombre as nombreCivil,
civil.apellido as apellidoCivil,
binc.nombre as nombreBarrio
FROM tp1."Incidente" i
JOIN tp1."SeInvolucraron" si ON i."idIncidente" = si."idIncidente"
JOIN tp1."Direccion" dinc ON i."idDireccion" = dinc."idDireccion"
JOIN tp1."Barrio" binc ON dinc."idBarrio" = binc."idBarrio"
```

```
JOIN tp1."Civil" civil ON si.dni = civil.dni
JOIN
(SELECT  c.dni, b."idBarrio", b.nombre
FROM tp1."Civil" c, tp1."ViveEn" ve, tp1."Direccion" d, tp1."Barrio" b
WHERE c.dni = ve.dni and ve."idDireccion" = d."idDireccion" and d."idBarrio" = b."idBarrio"
and ve."fechaInicio" =
(SELECT max(ve1."fechaInicio")
FROM tp1."Civil" civ, tp1."ViveEn" ve1, tp1."Direccion" d1, tp1."Barrio" b1
WHERE civ.dni = ve1.dni and civ.dni = c.dni
    and ve1."idDireccion" = d1."idDireccion" and d1."idBarrio" = b1."idBarrio"
GROUP BY civ.dni) ) civil_barrio
ON civil.dni = civil_barrio.dni
WHERE binc."idBarrio" = civil_barrio."idBarrio";
```

#### 2.3.8. Query 8

Los superheroes que tienen una habilidad determinada

```
SELECT sh."idSuperHeroe",
   sh.nombre,
   sh.color_disfraz,
FROM tp1."Superheroe" sh
JOIN tp1."Posee" p ON sh."idSuperHeroe" = p."idSuperHeroe"
JOIN tp1."Habilidad" h ON p."idHabilidad" = h."idHabilidad"
WHERE h."idHabilidad" = 5;
```

#### 2.3.9. Query 9

Los superheroes que han participado en algún incidente.

```
SELECT distinct sh."idSuperHeroe",
sh.nombre,
sh.color_disfraz
FROM tp1."Incidente" i
JOIN tp1."SuperParticipo" sp ON i."idIncidente" = sp."idIncidente"
JOIN tp1."Superheroe" sh ON sp."idSuperHeroe" = sh."idSuperHeroe";
```

#### 2.3.10. Query 10

Listado de todos los incidentes en donde estuvieron involucrados superheroes y fueron causados por los *archienemigos* de los superheroes involucrados.

```
SELECT distinct i."idIncidente",
                ti."nombre" as "tipoIncidente",
                i.fecha as "fechaIncidente",
                d.calle   as "calleDireccion",
                d.altura as "alturaDireccion",
                i.calle_1 as "calle_1",
                i.calle_2 as "calle_2",
                b.nombre as "nombreBarrio"
FROM tp1."Incidente" i
JOIN tp1."TipoIncidente" ti ON i."idTipoInicidente" = ti."idTipoInicidente"
JOIN tp1."SuperParticipo" sp ON i."idIncidente" = sp."idIncidente"
```

```
JOIN tp1."Superheroe" sh ON sp."idSuperHeroe" = sh."idSuperHeroe"
JOIN tp1."SeInvolucraron" si ON i."idIncidente" = si."idIncidente"
JOIN tp1."Civil" c ON si.dni = c.dni
JOIN tp1."archienemigoDe" arch ON arch."idSuperHeroe" = sh."idSuperHeroe"
JOIN tp1."Direccion" d ON i."idDireccion" = d."idDireccion"
JOIN tp1."Barrio" b ON d."idBarrio" = b."idBarrio"
WHERE si."idRolCivil" = 2 and arch.dni = c.dni;
```

## 2.4. Modelo Físico

Triggers (Restricciones):

```
CREATE FUNCTION tp1.archienemigo_no_es_el_mismo() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF EXISTS (SELECT * FROM tp1."Superheroe" sh, tp1."Civil" c
    where new."idSuperHeroe" = sh."idSuperHeroe" and c.dni = new.dni
    and sh.dni is not null and sh.dni = new.dni ) THEN
      RAISE EXCEPTION 'no puede ser archienemigo de si mismo';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.asignacion_fecha_mayor_a_oficial() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF EXISTS (SELECT * FROM tp1."Oficial" o
    where new.placa = o.placa
    and new."fechaInicio" < o."fechaIngreso" ) THEN
      RAISE EXCEPTION 'fecha de asignacion menor a fecha de ingreso del oficial';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.civil_no_superparticipo() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF EXISTS (SELECT * FROM tp1."Superheroe" sh, tp1."Civil" c , tp1."SuperParticipo" sp
    where sp."idSuperHeroe" = sh."idSuperHeroe"
    and c.dni = sh.dni and c.dni = new.dni
    and new."idIncidente" = sp."idIncidente" ) THEN
      RAISE EXCEPTION 'no puede participar como superheroe y como civil al mismo tiempo';
    END IF;
    RETURN NULL;
```

```
  END;

$$;

CREATE FUNCTION tp1.dni_oficiales_civiles() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Civil" c, tp1."Oficial" o
    where new.dni = c.dni or new.dni = o.dni) THEN
      RAISE EXCEPTION 'No puede haber un oficial con mismo dni que un civil';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.oficial_se_involucro_fecha() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Oficial" o, tp1."Incidente" i
    where new.placa = o.placa and i."idIncidente" = new."idIncidente"
    and i.fecha < o."fechaIngreso" ) THEN
      RAISE EXCEPTION 'Fecha de oficial menor a fecha de incidente';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.seguimiento_al_cerrarse_no_puede_cambiar() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF EXISTS (SELECT * FROM tp1."EstadoSeguimiento" e
    where new."idEstadoSeguimiento" = e."idEstadoSeguimiento"
    and e."idEstadoSeguimiento" != 3
    and old."idEstadoSeguimiento" = 3 ) THEN
      RAISE EXCEPTION 'seguimiento al cerrare no puede cambiar de estado';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.seguimiento_conclusion() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."EstadoSeguimiento" e
    where new."idEstadoSeguimiento" = e."idEstadoSeguimiento"
    and ( (e."idEstadoSeguimiento" = 3 and new.conclusion is NULL)
    or (e."idEstadoSeguimiento" != 3 and new.conclusion is not NULL) )
    ) THEN
```

```
      RAISE EXCEPTION 'al cerrarse se tiene una conclusion';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.seguimiento_fecha_incidente() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Incidente" i
    where i."idIncidente" = new."idIncidente"
    and new.fecha < i.fecha ) THEN
      RAISE EXCEPTION 'fecha incidente menor a fecha de seguimiento';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.seguimiento_fecha_oficial() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Oficial" o
    where o.placa = new.placa and new.fecha < o."fechaIngreso" )
    THEN
      RAISE EXCEPTION 'fehcha de ingreos de oficial mayor a fecha de seguimiento';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.seguimiento_seguida_si_en_proceso() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."EstadoSeguimiento" e
    where new."idEstadoSeguimiento" = e."idEstadoSeguimiento"
    and ((e."idEstadoSeguimiento" = 2 and new.placa is NULL)
    or (e."idEstadoSeguimiento" != 2 and new.placa is not NULL) ) )
    THEN
      RAISE EXCEPTION 'Solo puede ser seguido cuando esta en proceso';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.sumario_concluyo_tiene_resultado() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."EstadoSumario" e
    where new."idEstadoSumario" = 3 and new.resultado IS NULL )
```

```
    THEN
      RAISE EXCEPTION 'Si concluyo tiene resultado';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.sumario_es_tipo_investigador() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF (SELECT o.tipo FROM tp1."Oficial" o
    where new.placa = o.placa and o.tipo != 'Investigador')
    THEN
      RAISE EXCEPTION 'El oficial que investiga debe tener tipo Investigador';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.sumario_fecha_mayor_asignacion() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Asignacion" a
    where new."idAsignacion" = a."idAsignacion" and new.fecha < a."fechaInicio" )
    THEN
      RAISE EXCEPTION 'fecha de sumario menor a la de asignacion';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.sumario_fecha_mayor_investigador() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Oficial" i
    where new.placa = i.placa and new.fecha < i."fechaIngreso" )
    THEN
      RAISE EXCEPTION 'fecha sumario menor a fecha de ingreso del investigador';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.sumario_investigador_no_investigado() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Asignacion" a
    where new."idAsignacion" = a."idAsignacion" and new.placa = a.placa )
    THEN
```

```
      RAISE EXCEPTION 'Un investigador no puede investigarse a si mismo';
    END IF;
    RETURN NULL;
  END;
$$;

CREATE FUNCTION tp1.superheroeo_no_delincuente() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
BEGIN
    IF EXISTS (SELECT * FROM tp1."Civil" c, tp1."EstaCompuestaPor" comp
    where new.dni = c.dni and comp.dni = new.dni)
    THEN
      RAISE EXCEPTION 'No puede haber un oficial con mismo dni que un civil';
    END IF;
    RETURN NULL;
  END;

$$;

CREATE FUNCTION tp1.superparticipo_no_civil() RETURNS trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (SELECT * FROM tp1."Superheroe" sh, tp1."Civil" c , tp1."SeInvolucraron" si
    where new."idSuperHeroe" = sh."idSuperHeroe" and c.dni = sh.dni
    and c.dni = si.dni and new."idIncidente" = si."idIncidente" )
    THEN
      RAISE EXCEPTION 'no puede participar como superheroe y como civil al mismo tiempo';
    END IF;
    RETURN NULL;
  END;
$$;

Tables:

CREATE TABLE tp1."Asignacion" (
    "idAsignacion" serial NOT NULL,
    "fechaInicio" date NOT NULL,
    "idDesignacion" integer NOT NULL,
    placa integer NOT NULL
);

CREATE TABLE tp1."Barrio" (
    "idBarrio" serial NOT NULL,
    nombre character varying(250) NOT NULL
);

CREATE TABLE tp1."Civil" (
    dni integer NOT NULL,
```

```
    nombre character varying(250) NOT NULL,
    apellido character varying(250) NOT NULL
);

CREATE TABLE tp1."Conocimiento" (
    conocedor integer NOT NULL,
    conocido integer NOT NULL,
    "fechaConocimiento" date NOT NULL,
    "idTipoRelacion" integer NOT NULL
);

CREATE TABLE tp1."Departamento" (
    "idDepartamento" serial NOT NULL,
    nombre character varying(250) NOT NULL,
    descripcion text DEFAULT ''::text NOT NULL
);

CREATE TABLE tp1."Designacion" (
    "idDesignacion" serial NOT NULL,
    nombre character varying(250) NOT NULL
);

CREATE TABLE tp1."Direccion" (
    "idDireccion" serial NOT NULL,
    calle character varying(250) NOT NULL,
    altura integer NOT NULL,
    "idBarrio" integer NOT NULL
);

CREATE TABLE tp1."EsContactadoPor" (
    "idSuperHeroe" integer NOT NULL,
    dni integer NOT NULL
);

CREATE TABLE tp1."EstaCompuestaPor" (
    "idMafia" integer NOT NULL,
    dni integer NOT NULL
);

CREATE TABLE tp1."EstadoSeguimiento" (
    "idEstadoSeguimiento" serial NOT NULL,
    estado character varying(250) NOT NULL
);

CREATE TABLE tp1."EstadoSumario" (
    "idEstadoSumario" serial NOT NULL,
    estado character varying(25) NOT NULL
);
```

```
CREATE TABLE tp1."Habilidad" (
    "idHabilidad" serial NOT NULL,
    nombre character varying(250) NOT NULL
);

CREATE TABLE tp1."Incidente" (
    "idIncidente" serial NOT NULL,
    fecha date NOT NULL,
    calle_1 character varying(250) NOT NULL,
    calle_2 character varying(250) NOT NULL,
    "idTipoInicidente" integer NOT NULL,
    "idDireccion" integer NOT NULL
);

CREATE TABLE tp1."Oficial" (
    placa integer NOT NULL,
    dni integer NOT NULL,
    nombre character varying(250) NOT NULL,
    apellido character varying(250) NOT NULL,
    rango character varying(250) NOT NULL,
    "fechaIngreso" date NOT NULL,
    tipo character varying(250),
    "idDepartamento" integer NOT NULL
);

CREATE TABLE tp1."OficialSeInvolucro" (
    placa integer NOT NULL,
    "idIncidente" integer NOT NULL,
    "idResponsabilidad" integer NOT NULL
);

CREATE TABLE tp1."OrganizacionDelictiva" (
    "idMafia" serial NOT NULL,
    nombre character varying(250) NOT NULL
);

CREATE TABLE tp1."Posee" (
    "idSuperHeroe" integer NOT NULL,
    "idHabilidad" integer NOT NULL
);

CREATE TABLE tp1."RolCivil" (
    "idRolCivil" serial NOT NULL,
    nombre character varying(250) NOT NULL
);

CREATE TABLE tp1."RolOficial" (
    "idResponsabilidad" serial NOT NULL,
    descripcion character varying(250) NOT NULL
```

```
);

CREATE TABLE tp1."SeInvolucraron" (
    dni integer NOT NULL,
    "idIncidente" integer NOT NULL,
    "idRolCivil" integer NOT NULL
);

CREATE TABLE tp1."Seguimiento" (
    numero integer NOT NULL,
    fecha date NOT NULL,
    descripcion text,
    conclusion text,
    "idIncidente" integer NOT NULL,
    placa integer,
    "idEstadoSeguimiento" integer NOT NULL
);

CREATE TABLE tp1."Sumario" (
    "idSumario" serial NOT NULL,
    fecha date NOT NULL,
    observacion text,
    resultado text,
    placa integer NOT NULL,
    "idAsignacion" integer NOT NULL,
    "idEstadoSumario" integer NOT NULL
);

CREATE TABLE tp1."SuperParticipo" (
    "idSuperHeroe" integer NOT NULL,
    "idIncidente" integer NOT NULL
);

CREATE TABLE tp1."Superheroe" (
    "idSuperHeroe" serial NOT NULL,
    nombre character varying(250) NOT NULL,
    color_capa character varying(250) NOT NULL,
    dni integer,
    color_disfraz character varying(250) DEFAULT ''::character varying NOT NULL
);

CREATE TABLE tp1."TipoIncidente" (
    "idTipoInicidente" serial NOT NULL,
    nombre character varying(250) NOT NULL
);

CREATE TABLE tp1."TipoRelacion" (
    "idTipoRelacion" serial NOT NULL,
    nombre character varying(250) NOT NULL
```

```
);

CREATE TABLE tp1."ViveEn" (
    dni integer NOT NULL,
    "idDireccion" integer NOT NULL,
    "fechaInicio" date NOT NULL
);

CREATE TABLE tp1."archienemigoDe" (
    "idSuperHeroe" integer NOT NULL,
    dni integer NOT NULL
);

ALTER TABLE ONLY tp1."Civil"
    ADD CONSTRAINT "Civil_pkey" PRIMARY KEY (dni);

ALTER TABLE ONLY tp1."Departamento"
    ADD CONSTRAINT "Departamento_pkey" PRIMARY KEY ("idDepartamento");

ALTER TABLE ONLY tp1."Direccion"
    ADD CONSTRAINT "Direccion_pkey" PRIMARY KEY ("idDireccion");

ALTER TABLE ONLY tp1."Incidente"
    ADD CONSTRAINT "Incidente_pkey" PRIMARY KEY ("idIncidente");

ALTER TABLE ONLY tp1."OrganizacionDelictiva"
    ADD CONSTRAINT "Organización_delictiva_pkey" PRIMARY KEY ("idMafia");

ALTER TABLE ONLY tp1."TipoRelacion"
    ADD CONSTRAINT "TipoDeRelacion_pkey" PRIMARY KEY ("idTipoRelacion");

ALTER TABLE ONLY tp1."TipoIncidente"
    ADD CONSTRAINT "TipoIncidente_pkey" PRIMARY KEY ("idTipoInicidente");

ALTER TABLE ONLY tp1."archienemigoDe"
    ADD CONSTRAINT archienemigo_de_pkey PRIMARY KEY ("idSuperHeroe", dni);

ALTER TABLE ONLY tp1."Asignacion"
    ADD CONSTRAINT asignacion_pkey PRIMARY KEY ("idAsignacion");

ALTER TABLE ONLY tp1."Conocimiento"
    ADD CONSTRAINT conocimiento_pkey PRIMARY KEY (conocedor, conocido);

ALTER TABLE ONLY tp1."Designacion"
    ADD CONSTRAINT designacion_pkey PRIMARY KEY ("idDesignacion");

ALTER TABLE ONLY tp1."EsContactadoPor"
    ADD CONSTRAINT es_contactado_por_pkey PRIMARY KEY ("idSuperHeroe", dni);
```

```
ALTER TABLE ONLY tp1."EstaCompuestaPor"
    ADD CONSTRAINT esta_compuesta_por_pkey PRIMARY KEY ("idMafia", dni);

ALTER TABLE ONLY tp1."EstadoSeguimiento"
    ADD CONSTRAINT "estadoSeguimiento_pkey" PRIMARY KEY ("idEstadoSeguimiento");

ALTER TABLE ONLY tp1."EstadoSumario"
    ADD CONSTRAINT estado_sumario_pkey PRIMARY KEY ("idEstadoSumario");

ALTER TABLE ONLY tp1."Barrio"
    ADD CONSTRAINT "idBarrio" PRIMARY KEY ("idBarrio");

ALTER TABLE ONLY tp1."Oficial"
    ADD CONSTRAINT oficial_dni_key UNIQUE (dni);

ALTER TABLE ONLY tp1."Oficial"
    ADD CONSTRAINT oficial_pkey PRIMARY KEY (placa);

ALTER TABLE ONLY tp1."OficialSeInvolucro"
    ADD CONSTRAINT oficial_se_involucro_pkey PRIMARY KEY (placa, "idIncidente", "idResponsabilidad");

ALTER TABLE ONLY tp1."Habilidad"
    ADD CONSTRAINT pk_habilidad PRIMARY KEY ("idHabilidad");

ALTER TABLE ONLY tp1."Posee"
    ADD CONSTRAINT posee_pkey PRIMARY KEY ("idSuperHeroe", "idHabilidad");

ALTER TABLE ONLY tp1."RolCivil"
    ADD CONSTRAINT rol_civil_pkey PRIMARY KEY ("idRolCivil");

ALTER TABLE ONLY tp1."RolOficial"
    ADD CONSTRAINT rol_oficial_pkey PRIMARY KEY ("idResponsabilidad");

ALTER TABLE ONLY tp1."SeInvolucraron"
    ADD CONSTRAINT se_involucraron_pkey PRIMARY KEY (dni, "idIncidente");

ALTER TABLE ONLY tp1."Seguimiento"
    ADD CONSTRAINT seguimiento_pkey PRIMARY KEY (numero, "idIncidente");

ALTER TABLE ONLY tp1."Sumario"
    ADD CONSTRAINT sumario_pkey PRIMARY KEY ("idSumario");

ALTER TABLE ONLY tp1."SuperParticipo"
    ADD CONSTRAINT super_participo_pkey PRIMARY KEY ("idSuperHeroe", "idIncidente");

ALTER TABLE ONLY tp1."Superheroe"
    ADD CONSTRAINT superheroe_pkey PRIMARY KEY ("idSuperHeroe");

ALTER TABLE ONLY tp1."ViveEn"
```

```
    ADD CONSTRAINT vive_en_pkey PRIMARY KEY (dni, "idDireccion");

CREATE INDEX "fki_idDireccion" ON tp1."Incidente" USING btree ("idDireccion");

CREATE INDEX "fki_idTipoIncidente" ON tp1."Incidente" USING btree ("idTipoInicidente");

CREATE INDEX fki_superheroe_dni ON tp1."Superheroe" USING btree (dni);

CREATE CONSTRAINT TRIGGER check_archienemigo_de_si_mismo
AFTER INSERT OR UPDATE ON tp1."archienemigoDe"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.archienemigo_no_es_el_mismo();

CREATE CONSTRAINT TRIGGER check_concluyo_tiene_resultado
AFTER INSERT OR UPDATE ON tp1."Sumario"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.sumario_concluyo_tiene_resultado();

CREATE CONSTRAINT TRIGGER check_dni_no_civiles
AFTER INSERT OR UPDATE ON tp1."Oficial"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.dni_oficiales_civiles();

CREATE CONSTRAINT TRIGGER check_es_tipo_investigador
AFTER INSERT OR UPDATE ON tp1."Sumario"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.sumario_es_tipo_investigador();

CREATE CONSTRAINT TRIGGER check_fecha_inicio_mayor_a_oficial
AFTER INSERT OR UPDATE ON tp1."Asignacion"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.asignacion_fecha_mayor_a_oficial();

CREATE CONSTRAINT TRIGGER check_fecha_mayor_asigancion
AFTER INSERT OR UPDATE ON tp1."Sumario"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.sumario_fecha_mayor_asignacion();

CREATE CONSTRAINT TRIGGER check_fecha_mayor_investigador
AFTER INSERT OR UPDATE ON tp1."Sumario"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.sumario_fecha_mayor_investigador();

CREATE CONSTRAINT TRIGGER check_fecha_oficial_involucrado
AFTER INSERT OR UPDATE ON tp1."OficialSeInvolucro"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.oficial_se_involucro_fecha();

CREATE CONSTRAINT TRIGGER check_fecha_seg_incidente
```

```
AFTER INSERT OR UPDATE ON tp1."Seguimiento"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.seguimiento_fecha_incidente();

CREATE CONSTRAINT TRIGGER check_fecha_seg_oficial
AFTER INSERT OR UPDATE ON tp1."Seguimiento"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.seguimiento_fecha_oficial();

CREATE CONSTRAINT TRIGGER check_investigador_no_se_investiga
AFTER INSERT OR UPDATE ON tp1."Sumario"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.sumario_investigador_no_investigado();

CREATE CONSTRAINT TRIGGER check_seg_conclusion
AFTER INSERT OR UPDATE ON tp1."Seguimiento"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.seguimiento_conclusion();

CREATE CONSTRAINT TRIGGER check_seguimiento_cerrado_no_cambia
AFTER UPDATE ON tp1."Seguimiento"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.seguimiento_al_cerrarse_no_puede_cambiar();

CREATE CONSTRAINT TRIGGER check_seinvolucraron_no_sh
AFTER INSERT OR UPDATE ON tp1."SeInvolucraron"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.civil_no_superparticipo();

CREATE CONSTRAINT TRIGGER check_solo_seguido_en_proceso
AFTER INSERT OR UPDATE ON tp1."Seguimiento"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.seguimiento_seguida_si_en_proceso();

CREATE CONSTRAINT TRIGGER check_superheroeo_no_delincuente
AFTER INSERT OR UPDATE ON tp1."Superheroe"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.superheroeo_no_delincuente();

CREATE CONSTRAINT TRIGGER check_superparticipo_no_civil
AFTER INSERT OR UPDATE ON tp1."SuperParticipo"
NOT DEFERRABLE INITIALLY IMMEDIATE FOR EACH ROW EXECUTE PROCEDURE
tp1.superparticipo_no_civil();

ALTER TABLE ONLY tp1."OficialSeInvolucro"
    ADD CONSTRAINT "OficialSeInvolucro_placa_fkey" FOREIGN KEY (placa)
REFERENCES tp1."Oficial"(placa) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY tp1."archienemigoDe"
```

```
    ADD CONSTRAINT archienemigo_de_dni_fkey FOREIGN KEY (dni)
REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."archienemigoDe"
    ADD CONSTRAINT archienemigo_de_id_sh_fkey FOREIGN KEY ("idSuperHeroe")
REFERENCES tp1."Superheroe"("idSuperHeroe");

ALTER TABLE ONLY tp1."Asignacion"
    ADD CONSTRAINT asignacion_id_designacion_fkey FOREIGN KEY ("idDesignacion")
REFERENCES tp1."Designacion"("idDesignacion");

ALTER TABLE ONLY tp1."Asignacion"
    ADD CONSTRAINT asignacion_placa_fkey FOREIGN KEY (placa) REFERENCES tp1."Oficial"(placa)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY tp1."Conocimiento"
    ADD CONSTRAINT conocimiento_conocedor_fkey FOREIGN KEY (conocedor)
REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."Conocimiento"
    ADD CONSTRAINT conocimiento_conocido_fkey FOREIGN KEY (conocido)
REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."Conocimiento"
    ADD CONSTRAINT conocimiento_id_tipo_de_relacion_fkey FOREIGN KEY ("idTipoRelacion")
REFERENCES tp1."TipoRelacion"("idTipoRelacion");

ALTER TABLE ONLY tp1."Direccion"
    ADD CONSTRAINT direccion_id_barrio_fkey FOREIGN KEY ("idBarrio")
REFERENCES tp1."Barrio"("idBarrio");

ALTER TABLE ONLY tp1."EsContactadoPor"
    ADD CONSTRAINT es_contactado_por_dni_fkey FOREIGN KEY (dni)
REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."EsContactadoPor"
    ADD CONSTRAINT es_contactado_por_id_sh_fkey FOREIGN KEY ("idSuperHeroe")
REFERENCES tp1."Superheroe"("idSuperHeroe");

ALTER TABLE ONLY tp1."EstaCompuestaPor"
    ADD CONSTRAINT esta_compuesta_por_dni_fkey FOREIGN KEY (dni)
REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."EstaCompuestaPor"
    ADD CONSTRAINT esta_compuesta_por_id_mafia_fkey FOREIGN KEY ("idMafia")
REFERENCES tp1."OrganizacionDelictiva"("idMafia");

ALTER TABLE ONLY tp1."Incidente"
    ADD CONSTRAINT "incidente_idDireccion_fkey" FOREIGN KEY ("idDireccion")
```

```
REFERENCES tp1."Direccion"("idDireccion");

ALTER TABLE ONLY tp1."Incidente"
    ADD CONSTRAINT "incidente_idTIpoIncidente_fkey" FOREIGN KEY ("idTipoInicidente")
REFERENCES tp1."TipoIncidente"("idTipoInicidente");

ALTER TABLE ONLY tp1."Oficial"
    ADD CONSTRAINT "oficial_idDepartamento_fkey" FOREIGN KEY ("idDepartamento")
REFERENCES tp1."Departamento"("idDepartamento");

ALTER TABLE ONLY tp1."OficialSeInvolucro"
    ADD CONSTRAINT oficial_se_involucro_id_incidente_fkey FOREIGN KEY ("idIncidente")
REFERENCES tp1."Incidente"("idIncidente");

ALTER TABLE ONLY tp1."OficialSeInvolucro"
    ADD CONSTRAINT oficial_se_involucro_id_responsabilidad_fkey FOREIGN KEY ("idResponsabilidad")
REFERENCES tp1."RolOficial"("idResponsabilidad");

ALTER TABLE ONLY tp1."Posee"
    ADD CONSTRAINT posee_id_habilidad_fkey FOREIGN KEY ("idHabilidad")
REFERENCES tp1."Habilidad"("idHabilidad");

ALTER TABLE ONLY tp1."Posee"
    ADD CONSTRAINT posee_id_sh_fkey FOREIGN KEY ("idSuperHeroe")
REFERENCES tp1."Superheroe"("idSuperHeroe");

ALTER TABLE ONLY tp1."SeInvolucraron"
    ADD CONSTRAINT se_involucraron_dni_fkey FOREIGN KEY (dni)
REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."SeInvolucraron"
    ADD CONSTRAINT se_involucraron_id_incidente_fkey FOREIGN KEY ("idIncidente")
REFERENCES tp1."Incidente"("idIncidente");

ALTER TABLE ONLY tp1."SeInvolucraron"
    ADD CONSTRAINT se_involucraron_id_rol_civil_fkey FOREIGN KEY ("idRolCivil")
REFERENCES tp1."RolCivil"("idRolCivil");

ALTER TABLE ONLY tp1."Seguimiento"
    ADD CONSTRAINT "seguimiento_idEstadoSeg_fkey" FOREIGN KEY ("idEstadoSeguimiento")
REFERENCES tp1."EstadoSeguimiento"("idEstadoSeguimiento");

ALTER TABLE ONLY tp1."Seguimiento"
    ADD CONSTRAINT "seguimiento_idIncidente_fkey" FOREIGN KEY ("idIncidente")
REFERENCES tp1."Incidente"("idIncidente");

ALTER TABLE ONLY tp1."Seguimiento"
    ADD CONSTRAINT seguimiento_placa_fkey FOREIGN KEY (placa)
REFERENCES tp1."Oficial"(placa) ON UPDATE CASCADE ON DELETE CASCADE;
```

```
ALTER TABLE ONLY tp1."Sumario"
    ADD CONSTRAINT "sumario_estado_idEEstadoSum" FOREIGN KEY ("idEstadoSumario")
REFERENCES tp1."EstadoSumario"("idEstadoSumario")
ON UPDATE RESTRICT ON DELETE RESTRICT;

ALTER TABLE ONLY tp1."Sumario"
    ADD CONSTRAINT sumario_id_asignacion_fkey FOREIGN KEY ("idAsignacion")
REFERENCES tp1."Asignacion"("idAsignacion");

ALTER TABLE ONLY tp1."Sumario"
    ADD CONSTRAINT sumario_investigador_placa_fkey FOREIGN KEY (placa)
REFERENCES tp1."Oficial"(placa) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY tp1."SuperParticipo"
    ADD CONSTRAINT super_participo_id_incidente_fkey FOREIGN KEY ("idIncidente")
REFERENCES tp1."Incidente"("idIncidente");

ALTER TABLE ONLY tp1."SuperParticipo"
    ADD CONSTRAINT super_participo_id_sh_fkey FOREIGN KEY ("idSuperHeroe")
REFERENCES tp1."Superheroe"("idSuperHeroe");

ALTER TABLE ONLY tp1."Superheroe"
    ADD CONSTRAINT superheroe_dni_fkey FOREIGN KEY (dni) REFERENCES tp1."Civil"(dni)
ON UPDATE CASCADE;

ALTER TABLE ONLY tp1."ViveEn"
    ADD CONSTRAINT vive_en_dni_fkey FOREIGN KEY (dni) REFERENCES tp1."Civil"(dni);

ALTER TABLE ONLY tp1."ViveEn"
    ADD CONSTRAINT vive_en_id_direccion_fkey FOREIGN KEY ("idDireccion")
REFERENCES tp1."Direccion"("idDireccion");
```

# 3. Conclusión

Se logró diseñar un modelo del problema que cumple con el enunciado. Este modelo facilita la respuesta de las consultas requeridas. La implementación física en un motor rdbms resultó casi en una conversion 1 a 1 con respecto al modelo relacional. Se utilizaron las funcionalidades de triggers para garantizar el cumplimiento de restricciones que surgieron del análisis del problema a resolver. Una de las tareas que facilita todo el proceso de comprensión del problema y su posterior implementación en una base de datos es claramente el diseño de un modelo entidad relación previo, ya que luego puede pasarse, mediante reglas bien establecidas, a un modelo relacional y permite establecer el conjunto de asunciones y restricciones que deben tomarse sin ambigüedades.