Modelo Relacional con Casos de Corrupción

Luis Miguel Domínguez Pérez

Índice

1. Reglas del modelo.

1.- Cada caso de corrupción tiene un nombre que lo diferencia de los demás (Gurtel, ERES, Púnica Malaya, tarjetas black. Puyol,...), una breve descripción, una estimación de los millones que se han desviado y el ámbito (Banco, Ayuntamiento, Caja, Comunidad, Estado,...), un caso puede tener varios ámbitos.

2.- Cada caso es investigado por un juez del que se conoce su nombre, dirección, fecha de nacimiento y fecha en que comenzó a ejercer como juez. Una vez concluida la investigación del caso se emite un dictamen que se registra.

3.- En cada caso hay una serie de personas implicadas, cada una de ellas con un cargo principal determinado en el momento en que se produjo el caso. De cada uno de éstos se conoce su DNI, nombre, dirección y patrimonio. La corrupción ha llegado a tal extremo que una persona puede estar implicada en varios casos.

4.- Nos va a interesar saber cuántos implicados son familia, cada implicado puede ser familia de otro o más implicados, nos interesa saber el parentesco.

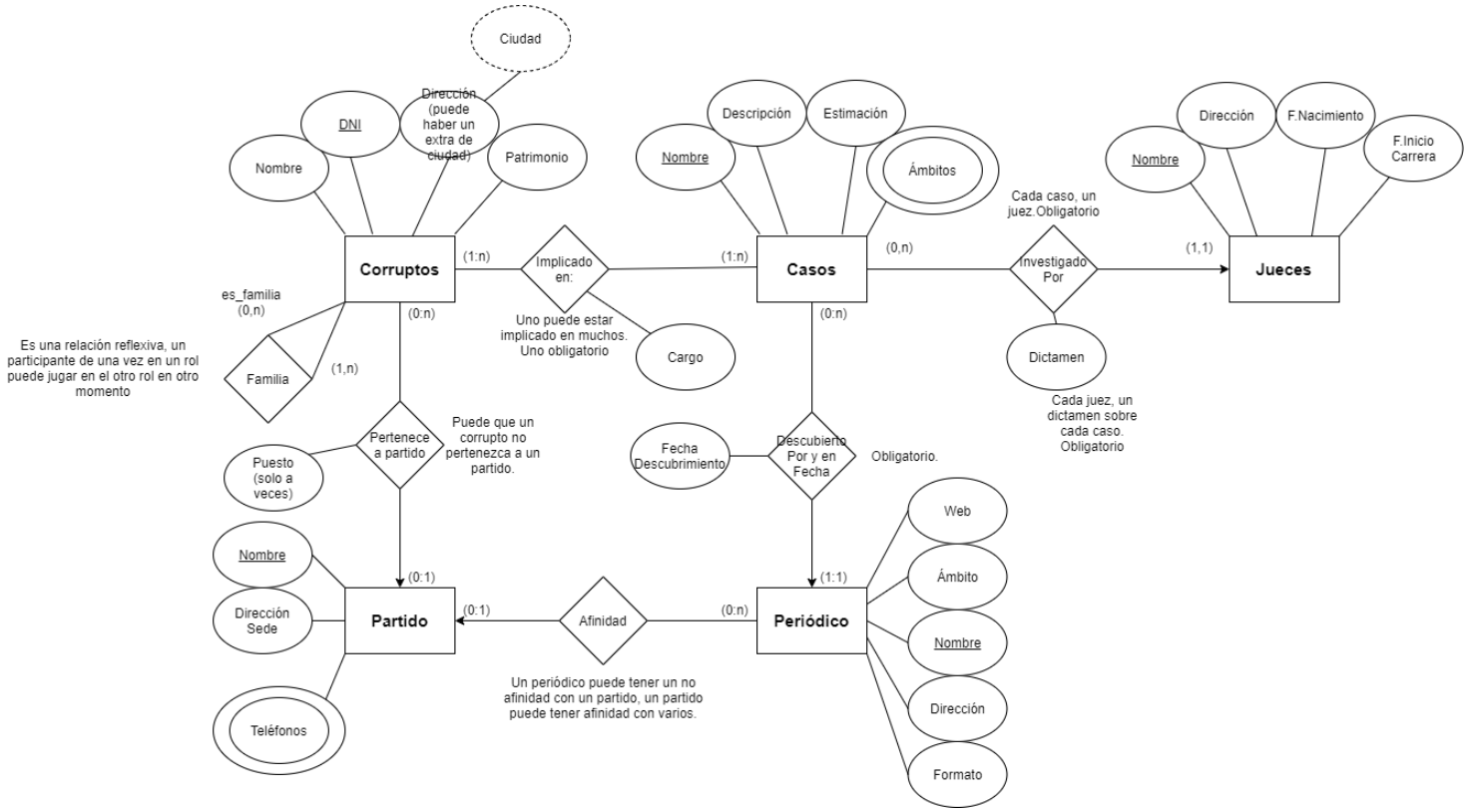
5.- Las personas pueden o no pertenecer a un partido político determinado y, a veces, desempeñan un puesto en él. De cada partido se conoce su nombre, dirección de la sede central y teléfonos.

6.- Cada caso de corrupción es descubierto por un periódico en una fecha determinada. Del periódico sabemos el nombre, la dirección, si es en papel o digital, página web si la tiene y ámbito (local, comarcal, nacional o internacional). Cada periódico puede tener o no una afinidad con un partido político. Un partido político puede tener afinidad con varios periódicos o con ninguno.

7.- Nos interesará saber en qué ciudad hay el máximo número de corruptos.

2. Modelo Entidad-Relación.

En este apartado vamos a elaborar el modelo entidad relación siguiendo las reglas establecidas por el enunciado. Mostraremos primero el esquema general del modelo y tras ello iremos comentando, una por una todas las entidades y las relaciones que median entre ellas. Mientras lo hacemos, iremos haciendo algunas suposiciones de cómo pueden quedar las tablas al pasar al modelo lógico, aunque siempre hay que estar abiertos a que puedan acabar siendo de otra manera.



Es difícil observar toda la información en este tamaño, por ello se adjunta la imagen a tamaño completo junto a los entregables del proyecto con el nombre 'Modelo_E-R'.

Comenzaremos desde arriba a la izquierda e iremos comentando cada entidad enlazándola con las otras en función de las relaciones que impliquen y sus respectivos atributos. Para identificar el tipo de dato de cada atributo utilizaremos las etiquetas (INT = entero, VARCHAR = cadena de caracteres, DATE = fecha) y se pondrá entre paréntesis junto al nombre del atributo.

- Entidad: Corruptos

Tal y como es definido en el enunciado, cada corrupto que aparezca en la BBDD debe estar identificado por un DNI (INT), que es nuestro caso juega el papel de Clave Primaria (PK) al ser siempre única. Como atributos de cada uno de los corruptos aparecerán su nombre (VARCHAR), su dirección (VARCHAR), campo del que deriva la ciudad (VARCHAR) en la que habita como atributo secundario para poder tener el cómputo de corruptos por ciudad.

- Relación: Familia (Corruptos-Corruptos)

Cada corrupto puede estar emparentado con otro por lo que debemos tener en cuenta una relación reflexiva de cada uno de los integrantes de esta entidad con cualquiera de ellos. Hay que tener en cuenta que cada uno de los que aparezcan en la entidad Corruptos puede estar emparentado con (0,n) otros integrantes. Esta relación seguramente derivará en una tabla auxiliar con dos columnas, la primera que refleje el nombre del corrupto (VARCHAR) y la segunda uno de sus familiares (VARCHAR).

- Relación: Implicado (Corruptos-Casos)

Cada corrupto puede estar implicado en n casos, por lo que sugiere una relación (1,n) por esta parte, y por la otra, cada caso debe tener al menos un Corrupto asignado aunque puede tener varios por lo que vuelve a ser (1,n). Esta relación tiene un atributo que depende de ella y es el Cargo (VARCHAR) en el caso determinado para cada implicado en el momento en que se produjo el caso. Como esta es una relación (n-n) requerirá de la creación de una tabla auxiliar.

- Relación: Pertenencia (Corruptos-Partido)

Cada corrupto puede pertenecer o no a algún partido político y sólo a uno de ellos (0-1). Sin embargo, cada partido puede tener varios integrantes entre los corruptos o no tener ninguno (0-n). De esta relación se puede desprender un atributo que es el puesto (VARCHAR) que ocupa cada uno de los corruptos en el partido al que pertenezca. Como esta relación sólo ocurre ocasionalmente y no es una relación (n-n), queda bajo nuestra elección que venga representada por una tabla. Si introducir la clave primaria y el atributo derivado de la relación de la tabla del 1 (Partido) en la otra (Corruptos) va a introducir muchos nulos sería conveniente que esta relación quede reflejada en una tabla auxiliar, como probablemente hagamos.

- Entidad: Partido

De cada partido se conoce su nombre (VARCHAR), que actuará como clave primaria. Aparte está registrada la dirección (VARCHAR) de su sede, y sus teléfonos (INT), que puede ser un atributo multivalorado. Al ser esta ultimo atributo de esta naturaleza, seguramente crearemos una tabla auxiliar para reflejarlos (en tercera forma normal no debe haber atributos multivalorados en una tabla y por ello no podemos incluirla como un atributo en la tabla de la entidad Partido).

- Entidad: Periódico

De un periódico se conoce su nombre (VARCHAR), que hace de clave primaria, su web (VARCHAR), su ámbito (VARCHAR), su dirección (VARCHAR) y su formato (VARCHAR). Todos estos atributos son univalorados y no deberían presentar un problema a la hora de que conformen una tabla.

- Relación: Afinidad (Partido-Periódico)

Cada periódico puede ser afín a un partido político como mucho y puede que a ninguno (0-1). Cada partido puede no tener ningún periódico afín o tener varios (0-n). Como es una relación (1-n) probablemente elijamos introducir la clave primaria de Partidos en la tabla de Periódicos. No hay ningún atributo que se desprenda de esta relación.

- Relación: Descubrimiento (Casos-Periódico)

Cada caso es siempre descubierto por un periódico en una fecha (1-1), pero no todos los periódicos han descubierto algún caso aunque puede haber alguno que haya descubierto varios (0-n). De esta relación se desprende un atributo, la fecha (DATE) de descubrimiento. Al ser una relación (1-n) insertaremos la clave primaria del Periódico como clave foranea (FK) en los Casos y haremos lo mismo con el atributo derivado de la relación. Esto no generará ningún campo nulo.

- **Entidad: Casos**

Cada caso viene identificado por una clave primaria que será el nombre (VARCHAR). Aparte de esto consta de una descripción (VARCHAR), una estimación de la cantidad implicada en millones (INT) y un ámbito (VARCHAR) que es un atributo multivalorado.

- **Relación: Investigado**

Cada caso es investigado por un solo juez (1-1) y cada juez ha juzgado o ningún caso o varios de los listados (0-n). Se desprende un atributo de esta relación, el dictamen (VARCHAR) que es emitido por el juez sobre el caso. Al ser una relación (1-n) y tener cada caso obligatoriamente un juez asignado y un dictamen sobre él se considerará introducir la PK de la tabla Jueces y el atributo derivado de la relación en la tabla Casos.

- **Entidad: Jueces**

Determinada por el nombre (VARCHAR) del juez que hace de clave primaria. Como información adicional encontramos su dirección (VARCHAR), su fecha de nacimiento (DATE) y su fecha de inicio de carrera (DATE).

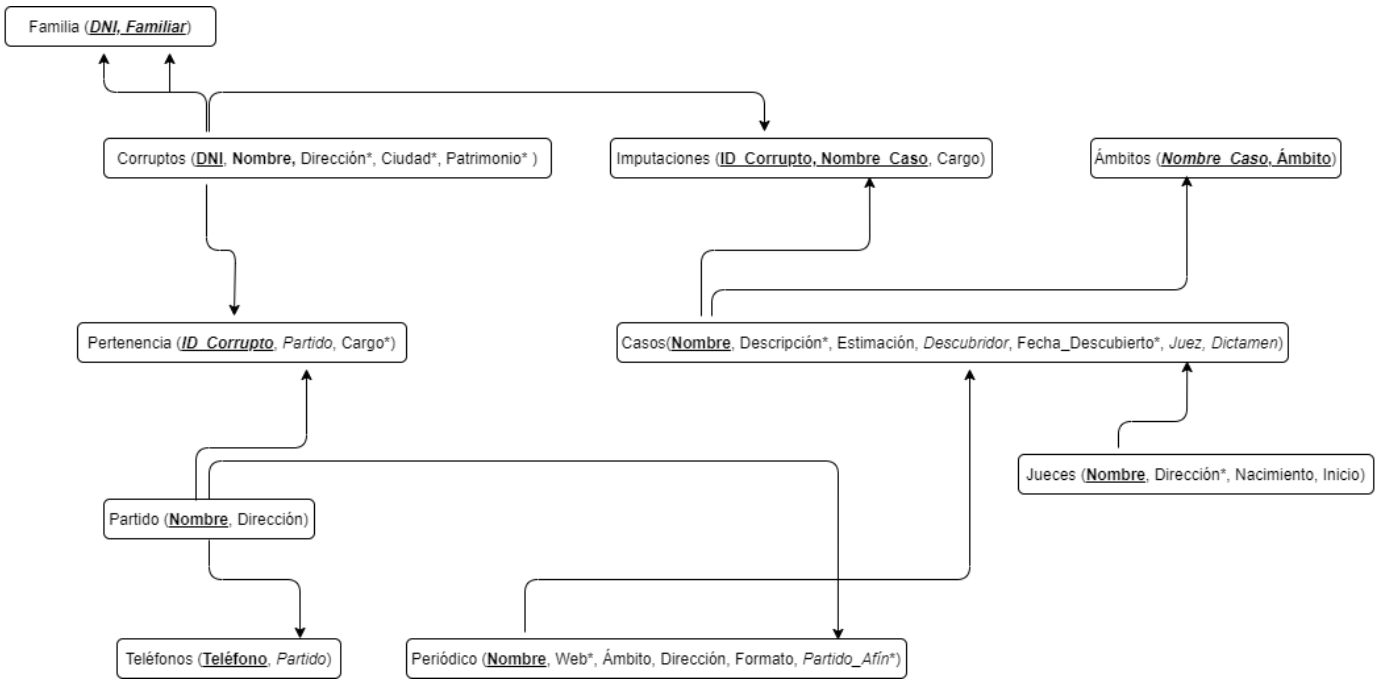
3. Modelo Relacional.

En este punto vamos a comenzar a condensar lo desarrollado en el modelo anterior construyendo tablas que cumplan la tercera forma normal. En mi caso lo más práctico me parece mostrar primero la query de creación de la tabla en MySQL a la que he llegado y tras ello desgranarla detalladamente explicando el por qué de cada decisión.

Es muy útil para este apartado y debe estar en todo momento presente el cumplimiento de la tercera forma normal a la hora de dar el modelo de las tablas del modelo relacional.

3.1. Modelo Lógico.

Analizando las dependencias de los atributos entre las tablas podemos llegar a que el modelo lógico que deberíamos seguir para crear nuestras tablas se puede esquematizar de la siguiente manera:



Donde las claves primarias están subrayadas y en negrita, las claves candidatas en negrita, los campos opcionales están indicados con '*' y las relaciones entre claves primarias y foráneas de otras tablas con flechas que salen de la primaria y apuntan a su dependiente.

3.2. Creación y detalles de las tablas.

Tabla Corruptos

```
CREATE TABLE Corruptos (
DNI INT PRIMARY KEY,
Nombre VARCHAR(30) NOT NULL,
Direccion VARCHAR(60),
Ciudad VARCHAR(30),
Patrimonio NUMERIC(10,2),
CHECK (length(DNI)=8)
);
```

Esta tabla proviene directamente de una de las entidades.

- Para comenzar con ella, he decidido nombrar el DNI como clave primaria (no puede haber valores nulos ni valores repetidos) ya que todos los corruptos deben tener uno y todos deben ser diferentes. La he tipado como un entero.

- El nombre es otra variable presente en la tabla de tipo cadena de caracteres y he impuesto la restricción de que no debe ser nulo en ningún caso.

- La dirección es una variable tipo cadena de caracteres que puede estar presente, aunque admite valores nulos. Incluirá dirección dentro de un municipio pero no el municipio como tal que será otra variable diferente para así facilitar búsquedas por municipio. Como una dirección no es única y necesita el municipio como identificador adicional ninguna de las dos tiene dependencia mutua y por lo tanto seguiría cumpliendo la tercera forma normal.

- El patrimonio del corrupto depende de él y sólo de él por lo tanto iría en esta tabla también y es de tipo Numérico. En mi caso he impuesto que tenga hasta 10 cifras enteras y dos decimales. Puede ser nula y podemos no conocerla.

- Podríamos haber pensado incluir el partido político y el cargo del imputado en esta tabla pero como puede ser una variable que introduzca muchos nulos en la tabla (no todos pertenecen a un partido y no todos de los que pertenecen a un partido tienen un cargo) prefiero elaborar una tabla auxiliar.

Tabla Familia

```
CREATE TABLE Familiares (
DNI INT,
Familiar INT,
FOREIGN KEY (DNI)
REFERENCES Corruptos(DNI)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Familiar)
REFERENCES Corruptos(DNI)
ON DELETE CASCADE
ON UPDATE CASCADE,
PRIMARY KEY(DNI,Familiar)
);
```

Esta tabla proviene de una relación reflexiva de la tabla de corruptos con ella misma.

- Tiene dos atributos principales, que provienen de los DNI de los integrantes de la tabla de corruptos, por ello son importados como claves foráneas referenciado adecuadamente su origen. Se les impone la restricción de que si los DNI de la tabla a los que hacen referencia son eliminados o modificados se eliminan o modifican los registros pertinentes de esta tabla para que así en ningún momento quede desactualizada.

- Como clave primaria se elige la combinación de los dos ya que un corrupto puede tener varias relaciones y aparecer varias veces en cualquiera de las dos columnas. Cada combinación es única aunque cada una de ellas debe aparecer dos veces en sentido especular (con los papeles cambiados en el sentido de la columna que ocupan), si A es familiar de B debe aparecer que B sea familiar de A.

Tabla Partidos

```
CREATE TABLE Partidos (
Nombre VARCHAR(60) PRIMARY KEY,
Direccion VARCHAR(60) NOT NULL UNIQUE
);
```

Esta tabla proviene directamente de una entidad.

- Cada registro queda completamente identificado por su nombre ya que no hay dos partidos con el mismo y por lo tanto constituye por sí misma una clave primaria.

- La dirección es un atributo que depende completamente de la clave primaria y por lo tanto adecuado para la tabla. Incluye la restricción de no ser nulo ya que la dirección de la sede de un partido debería ser siempre conocida y única.

Tabla Teléfonos

```
CREATE TABLE Telefonos (
Telefono INTEGER PRIMARY KEY,
Partido VARCHAR(60),
FOREIGN KEY (Partido)
REFERENCES Partidos(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Esta tabla proviene de un atributo multivaluado de la entidad Partidos, por ello la sacamos con una tabla auxiliar.

- La clave primaria va a ser el teléfono, ya que cada uno es único y el partido no puede ser ya que cada uno puede tener varios números asociados y aparecer varias veces. Por supuesto, el teléfono es de tipo entero.

- Aparace una clave foránea, el partido a cuya sede pertenece el teléfono . Se declara adecuadamente referenciando la clave primaria de la tabla adecuada y se incluyen restricciones para que las eliminaciones o modificaciones de las entradas de la tabla de partido actualicen o eliminen los registros de la tabla derivada.

Tabla Pertenencia

```
CREATE TABLE Pertenencia (
ID_Corrupto INT PRIMARY KEY,
Partido VARCHAR(60) NOT NULL,
Cargo VARCHAR(30),
FOREIGN KEY (ID_Corrupto)
REFERENCES Corruptos(DNI)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Partido)
REFERENCES Partidos(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Esta es la tabla que extraemos de la relación entre corruptos y partidos políticos. Como podía generar muchos nulos decidimos regularlo como una tabla aparte.

- La clave primaria he decidido que sea el DNI del corrupto ya que cada uno puede pertenecer a un único partido y tener sólo un cargo y por lo tanto aparecerá sólo una vez. Aparte de esto, también es FK (Foreign Key) y se agregan las restricciones necesarias para que los registros se actualicen o se eliminen ante modificaciones o eliminaciones de los registros a los que han referencia en la tabla original.

- El partido aparece como FK referenciando al nombre de los partidos de la tabla de partidos. Se agregan las restricciones necesarias para que actúe adecuadamente en las eliminaciones o modificaciones de los campos de la tabla

referenciada. Exigimos que no sea nulo ya que todos los corruptos que aparezcan en esta tabla deben pertenecer a algún partido político.

- El cargo es una variable que puede ser nula o no serla, puede haber algún político que no desempeñe ningún cargo en el partido.

Tabla Imputaciones

```
CREATE TABLE Imputaciones (
ID_Corrupto INT,
Nombre_Caso VARCHAR(30),
Cargo VARCHAR(30) NOT NULL,
FOREIGN KEY (ID_Corrupto)
REFERENCES Corruptos(DNI)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Nombre_Caso)
REFERENCES Casos(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE,
PRIMARY KEY (ID_Corrupto, Nombre_Caso),
CHECK (Cargo IN ('Imputado','Condenado'))
);
```

Esta tabla proviene de la relación entre las entidades corruptos y casos y se encarga de recoger los cargos con los que han sido imputados los corruptos en cada caso. Como cada corrupto puede haber sido imputado en varios casos y un caso puede tener muchos corruptos esta relación requiere de la creación de una tabla auxiliar.

- ID_Corrupto es una variable FK que proviene de la tabla de corruptos y referencia a la columna DNI. Se agregan las restricciones oportunas para que se actualicen o eliminen registros cuando sea necesario.

- Nombre_Caso es otra variable FK que hace referencia y se actualiza de manera acorde a la clave primaria de la tabla de Casos.

- La combinación de estas dos variables forma la clave primaria por las dependencias cruzadas que nombramos anteriormente, un corrupto puede aparecer varias veces en la tabla en diferentes casos y cada caso puede aparecer varias veces con diferentes corruptos asociados.

- El cargo es el atributo que proviene de esta relación y por el que se ha creado la tabla.

Tabla Casos

```
CREATE TABLE Casos (
Nombre VARCHAR(30) PRIMARY KEY,
Descripcion VARCHAR(200),
Estimacion INTEGER NOT NULL,
Descubridor VARCHAR(60) NOT NULL,
Fecha_Descubierto DATE,
Juez VARCHAR(30) NOT NULL,
Dictamen VARCHAR(90) NOT NULL,
FOREIGN KEY (Descubridor)
REFERENCES Periodicos(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (Juez)
REFERENCES Jueces(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Esta tabla proviene directamente de una entidad, Casos.

- Cada caso lo identificamos por un nombre, que constituye la PK de la tabla.

- Cada uno tiene una descripción que lo caracteriza.

- Para cada uno hay una estimación de los millones de euros involucrados.

- Todo caso es descubierto por algún periódico. Este atributo junto con el de la fecha de descubrimiento viene de la relación entre casos y periódicos. Como es una relación (1:n) y cada caso es el que es descubierto por un sólo periódico introducimos la clave primaria de la tabla periódicos en la tabla de casos como FK junto con el atributo asociado al descubrimiento, que en este caso es la fecha.

- Cada caso es descubierto por un solo juez. La relación entre jueces y casos es similar a la que existe entre periódicos y casos, y la tabla de casos vuelve a tener el lado 1 de la relación (1:n), por lo que inserto la clave primaria de la tabla Jueces en la de Casos como FK. El atributo asociado a la relación, el dictamen, también se inscluirá en la tabla.

- A las FK se les asignan las restricciones necesarias para la correcta actualización de las tablas al modificar la referencia principal.

Tabla Ámbitos

```
CREATE TABLE Ambitos (
Nombre_Caso VARCHAR(30),
Ambito VARCHAR(30),
FOREIGN KEY (Nombre_Caso)
REFERENCES Casos(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE,
PRIMARY KEY (Nombre_Caso, Ambito),
CHECK (Ambito IN ('Red de trafico','Malversacion','Trafico de influencia'))
);
```

Cada caso puede tener varios ámbitos, por lo que es necesario crear una tabla auxiliar para reflejarlo ya que no puede haber elementos multivaluados en una tabla de un modelo relacional.

- Cada entrada es una pareja de nombre de caso y ámbito, y entre los dos son clave primaria. Como cada uno de los casos o cada uno de los ámbitos pueden aparecer varias veces en la tabla ninguno de los dos atributos por separado podría ser PK. Referenciamos adecuadamente que el nombre de los casos es una FK e imponemos las restricciones necesarias.

- También imponemos restricciones sobre el dominio del ámbito, forzando a que esté entre 3 posibles opciones.

Tabla Periódicos

```
CREATE TABLE Periodicos (
Nombre VARCHAR(60) PRIMARY KEY,
Web VARCHAR(60),
Ambito VARCHAR(30) NOT NULL,
Direccion VARCHAR(60) NOT NULL,
Formato VARCHAR(30) NOT NULL,
Partido_Afin VARCHAR(60),
FOREIGN KEY (Partido_Afin)
REFERENCES Partidos(Nombre)
ON DELETE CASCADE
ON UPDATE CASCADE,
CHECK (Ambito IN ('Local','Comarcal','Nacional','Internacional')),
CHECK (Formato IN ('papel','digital'))
);
```

Esta tabla proviene directamente de una entidad.

- Cada periódico viene identificado por un nombre y por lo tanto consituye la PK de la tabla.

- Cada periódico tiene una dirección web asociada.

- Cada periódico cuenta con una dirección de sede, exigimos que no sea nula.

- Cada periódico pertenece a un ámbito, no puede ser nulo.

- El formato del periódico debe ser conocido.

- Cada periódico puede tener un partido afín. La relación entre partidos y periódicos es (1:n), pudiendo ser cada periódico afín a un partido o no y cada partido puede tener varios periódicos afines. En nuestro caso, vamos a asumir que esta relación no va a introducir muchos nulos en la relación y por lo tanto incluimos la clave primaria de partido como FK en esta tabla para hacer valer la relación. Imponemos las restricciones adecuadas.

Tabla Jueces

```
CREATE TABLE Jueces (
Nombre VARCHAR(30) PRIMARY KEY,
Direccion VARCHAR(60),
Nacimiento DATE NOT NULL,
Inicio DATE NOT NULL,
CHECK (Inicio > Nacimiento)
);
```

Esta tabla proviene directamente de una entidad.

- Cada juez se identifica por un nombre que constituye la PK de la tabla.

- Cada juez tiene una dirección.

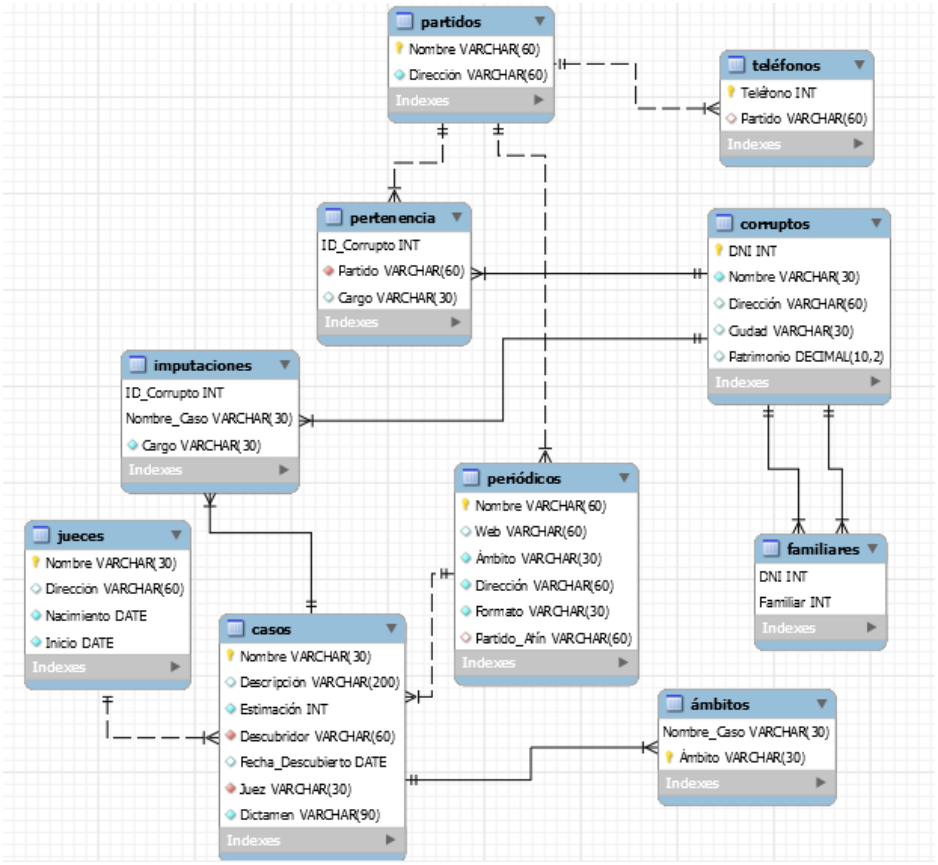
- Cada juez cuenta con una fecha de nacimiento y de inicio de carrera que no pueden ser nulas.

- Imponemos la restricción de que la fecha de inicio de la carrera de cada juez sea posterior a la de su nacimiento.

Podemos encontrar todo el script de la creación de tablas tanto por separado como en el script global del proyecto adjunto a los ficheros de la entrega o en el enlace al repositorio en Git del proyecto.

3.3. Diagrama MySQL

El siguiente diagrama de tablas ha sido creado por la feature de Reverse Engineering de MySQL una vez elaboradas las tablas:



4. Inserción de datos en tablas.

En esta sección se muestra brevemente cómo se está realizando la inserción de los datos en las tablas. Este apartado no suscita un interés especial ya que sencillamente se han introducido datos en las tablas para que las queries elaboradas posteriormente puedan mostrar un comportamiento que nos permita demostrar que funcionan como deben. La mayoría de los datos han sido insertados desde código con queries como, por ejemplo:

```
INSERT INTO Pertenencia VALUES
      (33333333,'Coalicion␣por␣la␣Honradez','Secretario␣General'),
      (11111111,'Coalicion␣por␣la␣Honradez','Portavoz'),
      (88888888,'Coalicion␣por␣la␣Honradez',NULL),
      (55555555,'Partido␣del␣Dinero','Diputado'),
      (22222222,'Partido␣del␣Dinero',NULL);
```

Otras de las tablas se han llenado de datos a través de los almacenados en un fichero externo. Este fichero debe estar en un directorio considerado seguro por MySQL, y cuya ruta podemos obtener con la query:

```
SELECT @@GLOBAL.secure_file_priv;
```

La consulta para introducir los datos desde este directorio a una tabla es:

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL␣Server␣8.0/Uploads/Jueces.txt'
INTO TABLE Jueces
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

Podemos encontrar todo el script de la inserción de datos tanto por separado como en el script global del proyecto adjunto a los ficheros de la entrega o en el enlace al repositorio en Git del proyecto.

5. Consultas.

En este apartado vamos a cubrir una por una y comentadas todas las consultas realizadas sobre la base de datos que hemos creado. El orden de complejidad va en aumento y se cubren muchas de las funcionalidades aprendidas durante el curso.

1: Podemos declarar en el 'select' las columnas que queramos proyectar en la consulta.

```
SELECT DNI,Nombre,Ciudad
FROM Corruptos;
```

2: Con la instrucción distinct filtramos sólo los elementos diferentes de un determinado campo.

```
SELECT DISTINCT Ciudad
FROM Corruptos;
```

3: Con 'where' imponemos condiciones que filtran los registros que se nos muestran. Pueden tener diferentes condiciones combinadas.

```
SELECT *
FROM Corruptos
WHERE Ciudad='Madrid' AND Patrimonio between 150000.00 AND 250000.00;
```

4: Con la instrucción count podemos contar todas las filas del dataset que en este caso cumplan una determinada condición.

```
SELECT count(*)
FROM Corruptos
WHERE Ciudad='Madrid';
```

5: Filtramos los elementos no nulos con 'IS NOT NULL' y ordenamos los resultados obtenidos según un criterio, en este caso sobre la cantidad de una columna numérica. En caso de ordenar por una columna string se seguirá un orden alfabético.

```
SELECT Nombre,Patrimonio
FROM Corruptos
WHERE Patrimonio IS NOT NULL
ORDER BY Patrimonio DESC;
```

6: Podemos utilizar agrupaciones sobre campos pero en el 'select' sólo podemos proyectar campos agrupados o el parámetro de agrupación. Se pueden seguir imponiendo filtros sobre los registros.

```
SELECT Ciudad,count(*)
FROM Corruptos
WHERE Patrimonio > 100.00
GROUP BY Ciudad;
```

7: Cuando queremos poner una condición sobre campos agregados como un 'count()' tenemos que utilizar 'HAVING' en lugar de 'WHERE' y recordar que siempre va tras la declaración de la agrupación.

```
SELECT Ciudad,count(*)
FROM Corruptos
WHERE Patrimonio > 100.00
GROUP BY Ciudad
HAVING count(*) > 1;
```

8: Podemos utilizar alias sobre las columnas generadas y combinar agrupaciones, ordenaciones y filtros como queramos

```
SELECT Ciudad, sum(Patrimonio) as Suma_patrimonio
FROM Corruptos
WHERE Ciudad IS NOT NULL
GROUP BY Ciudad
ORDER BY Suma_patrimonio;
```

9: Imponemos una condición con un 'OR'.

```
SELECT ID_Corrupto,Nombre_Caso
FROM Imputaciones
WHERE Nombre_Caso = 'Caso␣Salchichon' OR Nombre_Caso = 'Caso␣Cromos';
```

10: Podemos utilizar la siguiente sintaxis para hacer un 'JOIN' entre dos tablas.

```
SELECT *
FROM Imputaciones,Corruptos
WHERE Corruptos.DNI = Imputaciones.ID_Corrupto;
```

11: Hacemos un 'JOIN' al estilo de la última query pero poniendo alias a las tablas unidas y a una de las columnas proyectadas e imponiendo un filtro adicional sobre dos columnas llamadas con el prefijo de la tabla correspondiente para desambiguar.

```
SELECT C.Nombre,I.Nombre_Caso AS participo_en
FROM Imputaciones AS I,Corruptos AS C
WHERE C.DNI = I.ID_Corrupto  AND (I.Nombre_Caso = 'Caso␣Salchichon' OR I.
   Nombre_Caso = 'Caso␣Cromos');
```

12: 'JOIN' y proyección con alias y una condición de filtro sobre los registros. Utilizamos el 'lower' para regularizar los resultados y una expresión regular que nos permite recuperar los registros que no incluyan la palabra calle rodeada por cualquier tipo de campos (%).

```
SELECT C.Nombre,C.Direccion,I.Nombre_Caso AS participo_en
FROM Imputaciones AS I,Corruptos AS C
WHERE C.DNI = I.ID_Corrupto AND lower(C.Direccion) NOT LIKE '%calle%';
```

13: Podemos imponer condiciones sobre una subquery, en este caso, estamos filtrando los nombres de los jueces cuya fecha de inicio sea inferior a todas las fechas de descubrimiento de los casos.

```
SELECT Nombre
FROM Jueces
WHERE Inicio < all (SELECT Fecha_Descubierto FROM Casos);
```

14: Podemos hacer lo mismo pero filtrando los que cumplan la condición con al menos uno de los casos.

```
SELECT Nombre
FROM Jueces
WHERE Inicio < some (SELECT Fecha_Descubierto FROM Casos);
```

15: Creamos una vista temporal. Esto no es una tabla pero puede ser llamada en consultas sucesivas para hacer consultas desde ella. Son utilizadas si vamos a utilizar una determinada consulta como subquery en repetidas ocasiones. Estamos juntando los corruptos con los casos en los que están implicados.

```
CREATE VIEW Implicados as
                (SELECT I.Nombre_Caso as Caso, C.Nombre as Nombre
         FROM Imputaciones as I,Corruptos as C
         WHERE I.ID_Corrupto=C.DNI);
```

16: Unimos la tabla anterior con los jueces que los han procesado con la sintaxis de 'JOIN' que utilizamos anteriormente. Esta nueva tabla contiene información de los corruptos, los casos en los que han estado implicados y los jueces que los han procesado.

```
SELECT I.Nombre, C.Juez as procesado_por
FROM Implicados as I,Casos as C
WHERE I.Caso=C.Nombre;
```

17: Vamos a mostrar cómo conseguir una consulta similar de una manera más limpia y utilizando una sintaxis más clásica para los 'JOIN'. En este caso, utilizamos un 'INNER JOIN' en las dos ocasiones, por lo que no aparecerá ningún corrupto ni cas que no aparezca en alguna de cada una de las tablas unidas. En este caso, estamos proyectando cada uno de los corruptos y el juez que lo ha procesado.

```
SELECT C.Nombre, Ca.Juez as procesado_por
FROM Corruptos as C
INNER JOIN Imputaciones as I ON C.DNI=I.ID_Corrupto
INNER JOIN Casos as Ca ON I.Nombre_Caso = Ca.Nombre;
```

18: Unimos las tablas de corruptos y pertenencia para conocer qué corruptos pertenecen a qué partidos y con qué cargos. Al ser un 'INNER JOIN' no aparecerán corruptos en la tabla final que no pertenezcan a ningún partido ni ningún partido.

```
SELECT C.Nombre, P.Partido, P.Cargo
FROM Corruptos as C
INNER JOIN Pertenencia as P ON C.DNI = P.ID_Corrupto;
```

19: Similar pero con un 'LEFT OUTER JOIN'. Como la tabla 'LEFT' son los corruptos, aparecerán todos, aunque su ID no aparezca en la tabla de pertenencia y sus campos sean nulos. Estas dos consultas se han hecho para mostrar el correcto funcionamiento de los diferentes 'JOIN'.

```
SELECT C.Nombre, P.Partido, P.Cargo
FROM Corruptos as C
LEFT OUTER JOIN Pertenencia as P ON C.DNI = P.ID_Corrupto;
```

Hay periódicos que hayan descubierto casos con políticos de partidos afines implicados?

20: En mi caso voy a hacer un 'JOIN' que refleje las relaciones entre 4 tablas, corruptos, imputaciones, pertenencia y periódicos. De este 'JOIN' proyectaremos las columnas Nombre del corrupto, periódico descubridor, partido del corrupto y partido afín del periódico. Una vez hecho esto, sencillamente imponiendo un filtro entre la igualdad o desigualdad de los partidos podemos conocer lo planteado.

```
SELECT Nombre, Descubridor, Partido as Partido_del_politico, Partido_Afin as
   Partido_del_periodico
FROM (SELECT Co.Nombre, Ca.Descubridor, Pe.Partido, Per.Partido_Afin FROM Casos as
   Ca
          INNER JOIN Imputaciones as I ON Ca.Nombre=I.Nombre_Caso
          INNER JOIN Corruptos as Co ON Co.DNI=I.ID_Corrupto
          INNER JOIN Pertenencia as Pe ON Co.DNI=Pe.ID_Corrupto
          INNER JOIN Periodicos as Per ON Ca.Descubridor=Per.Nombre) as aux
WHERE Partido = Partido_Afin;
```

No hay ningún resultado. Podemos ver el resultado complementario.

```
SELECT Nombre, Descubridor, Partido as Partido_del_politico, Partido_Afin as
   Partido_del_periodico
FROM (SELECT Co.Nombre, Ca.Descubridor, Pe.Partido, Per.Partido_Afin FROM Casos as
   Ca
          INNER JOIN Imputaciones as I ON Ca.Nombre=I.Nombre_Caso
          INNER JOIN Corruptos as Co ON Co.DNI=I.ID_Corrupto
          INNER JOIN Pertenencia as Pe ON Co.DNI=Pe.ID_Corrupto
          INNER JOIN Periodicos as Per ON Ca.Descubridor=Per.Nombre) as aux
WHERE Partido != Partido_Afin;
```

21: Podríamos haber hecho esto con una vista temporal.

```
CREATE VIEW Corruptos_Periodicos as
(SELECT Co.Nombre, Ca.Descubridor, Pe.Partido, Per.Partido_Afin FROM Casos as Ca
INNER JOIN Imputaciones as I ON Ca.Nombre=I.Nombre_Caso
INNER JOIN Corruptos as Co ON Co.DNI=I.ID_Corrupto
INNER JOIN Pertenencia as Pe ON Co.DNI=Pe.ID_Corrupto
INNER JOIN Periodicos as Per ON Ca.Descubridor=Per.Nombre);
```

```
SELECT Nombre, Descubridor, Partido as Partido_del_politico, Partido_Afin as
   Partido_del_periodico
FROM Corruptos_Periodicos
WHERE Partido = Partido_Afin;
```

```
SELECT Nombre, Descubridor, Partido as Partido_del_politico, Partido_Afin as
   Partido_del_periodico
FROM Corruptos_Periodicos
WHERE Partido != Partido_Afin;
```

22: Podemos hacer un update de los registros que cumplan una determinada condición. En nuestro caso estamos multiplicando por 1000000 la estimación de millones de euros implicados (transformándolo a euros) de los casos cuyo ámbito sea 'Malversación'.

```
UPDATE Casos
INNER JOIN Ambitos ON Casos.Nombre = Ambitos.Nombre_Caso
SET Casos.Estimacion = Casos.Estimacion * 1000000
WHERE Ambitos.Ambito = 'Malversacion';
```

23: Por último, vamos a elimnar un tipo de registro de una de las tablas con una condición y comprobar que los 'CASCADE' se cumplen. Primero mostramos la tabla Casos que incluye el campo Descubridor que hace referencia al nombre del periódico que lo descubrió y es FK con una condición de eliminar en cascada en caso de que desaparezca uno de los registros del atributo al que referencia en su tabla original.

Mostramos la tabla original.

```
SELECT Nombre,Descubridor FROM Casos;
```

Eliminamos uno de los periódicos

```
DELETE FROM Periodicos
WHERE nombre = 'El␣Cotilla␣de␣Castuera';
```

Los registros referentes a ese periódico eliminado han desaparecido.

```
SELECT Nombre,Descubridor FROM Casos;
```