Base de datos 1

Alumno: Santiago Vietto

Docente: Leandro Luis Juárez

DNI: 42654882

Institución: UCC

Año: 2021

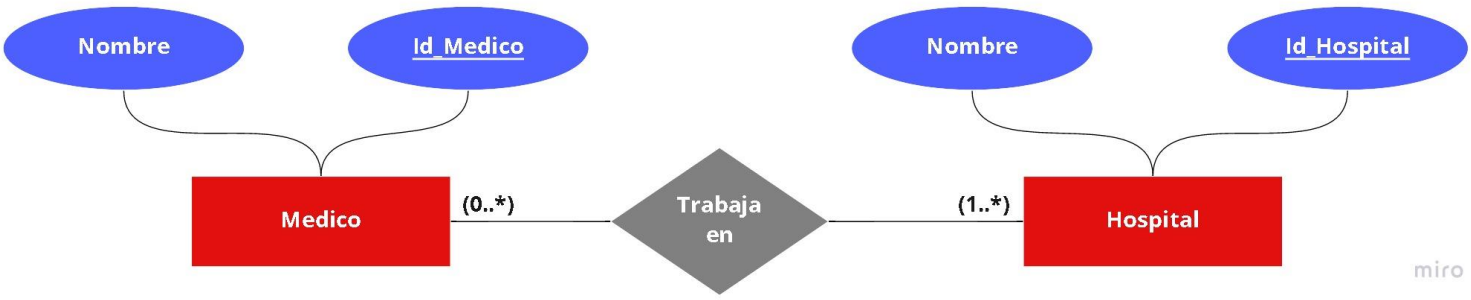
Diseño de base de datos

_ En el practico tenemos dos partes principales, una es el diseño de la base de datos, de alguna manera como va a estar estructurada o cómo se va a organizar o guardar la información dentro. Y en la segunda parte es como vamos a consultar esa información para poder utilizarla. Por ejemplo en un Loggin de una página, una cosa es como guardamos la información de los usuarios dentro de una entidad usuarios, y la otra forma es como le pedimos a la base de datos esa información para después mostrarla por ejemplo en el perfil de usuario.

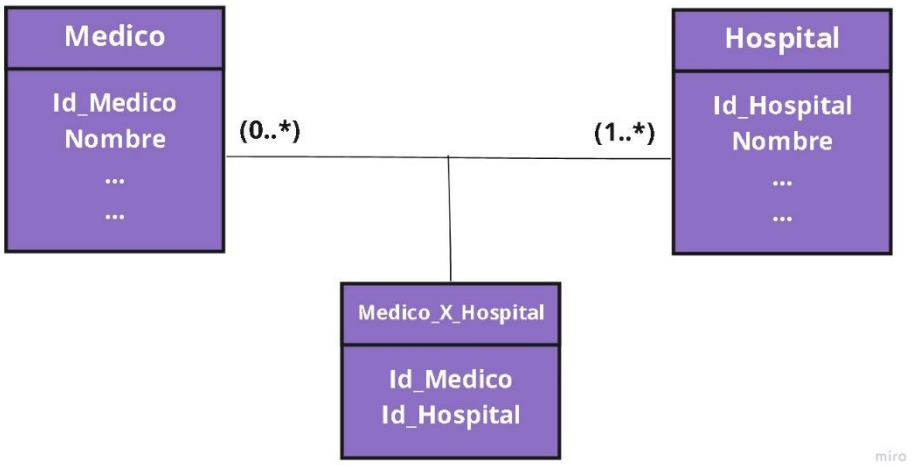
_ Es muy importante como guardamos la información, porque dependiendo la forma y el diseño que le hagamos a la base de datos, eso nos va a facilitar o complicar la forma en como extraemos la información. Al fin y al cabo la información se va a terminar guardando de una forma u otra, vamos a tener restricciones, pero el problema va a estar en el costo que vamos a tener en obtener o extraer esa información.

_ Nosotros podemos guardar la información de manera eficiente o deficiente y también podemos consultar de manera eficiente o deficiente, la diferencia está en que si guardamos la información de manera deficiente o desordenada, es muy costoso después corregir o cambiar eso una vez que el software o sistema este andando, en cuanto a tiempo de trabajo y dinero. Ahora, al momento de consultar los datos, a esas preguntas las podemos hacer de forma eficiente o deficiente, pero corregir las consultas es mucho menos costoso, por eso se hace mucho hincapié en el diseño de la base de datos para que una vez que estén listas no debamos tocarlas más.

_ Vamos a utilizar unos diagramas que se llaman diagramas entidad-relación (E-R) y unos diagramas que se llaman diagramas de clases. El diagrama E-R, que es similar al diagrama de flujo, es un diagrama de alguna manera un poco informal, en donde el diagrama formal sería el diagrama de clases. Pero por más que el diagrama E-R sea informal, nosotros vamos a estandarizar ciertas cuestiones para que podamos leer todos los diagramas. Las entidades las representamos como rectángulos donde le van a salir sus atributos en forma de elipses, que luego esto se va a representar en una tabla donde cada entidad tiene tuplas o filas que son las distintas instancias de esa entidad. Como vemos en el ejemplo tenemos una entidad medico con sus atributos, también una entidad hospital con sus atributos, y tienen una relación entre ellos que es como un rombo, en donde por lo general se le pone un verbo dentro pero que no es tan importante porque dependiendo como lo leamos va a decir una cosa u otra:



_ También tenemos la cardinalidad, que es un tema puntual, porque este tema es como la POO, algebra, análisis matemático, que una vez que se hicieron hace mucho tiempo ya no cambian más. En este caso la cardinalidad me indica como son las relaciones de ambas entidades en cuanto a cantidad, como vemos en el ejemplo tenemos 0 que significa ninguno, el 1 significa al menos uno, y el * es muchos, entonces (0..*) se lee un médico puede o no trabajar en ningún hospital o trabajar en muchos hospitales, o sea puede haber médicos que no estén asignados a ningún hospital, ahora si lo leemos del otro lado un hospital tiene al menos un médico o puede tener muchos médicos, es decir, nunca va a haber un hospital que no tenga un médico.



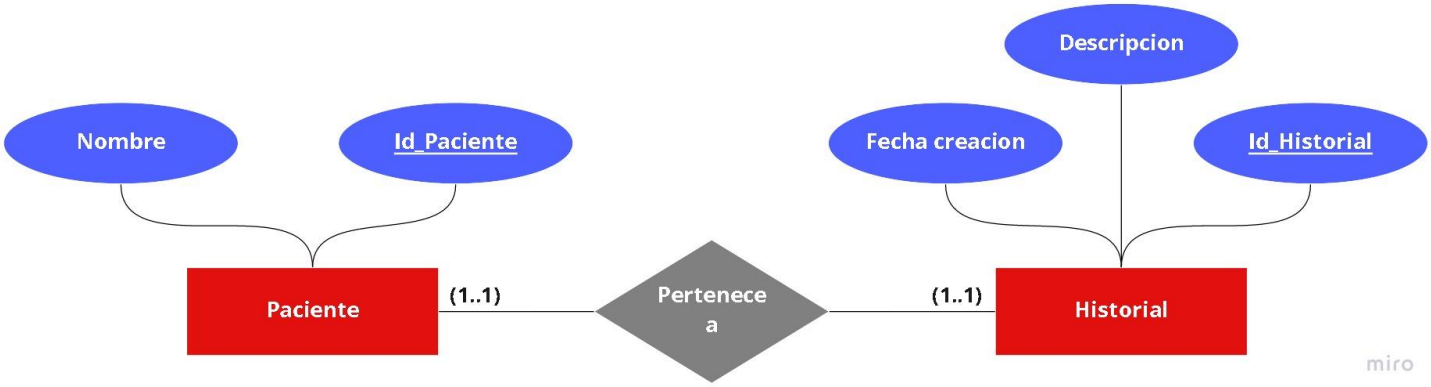
_ Además vemos que en un hospital debe haber al menos un médico o puede haber varios. Estas palabras son las que nos ayudan a definir las restricciones de cómo se relacionan las entidades del sistema.

_ El procedimiento normal con el que se realizan los diseños es, me junto con el cliente, me da un requerimiento, y con toda la información que nos da armamos un diagrama entidad-relación, y una vez que tenemos nuestro diagrama E-R armamos un diagrama de clases de base de datos, donde estos tienen muchas restricciones para que nadie las modifique. Cuando nosotros hablamos con el cliente, tenemos que saber cómo tarea nuestra, identificar qué cosas dentro de ese dominio van a ser entidades, que cosas van a ser atributos y que cosa va a ser una relación. Los elementos básicos son:

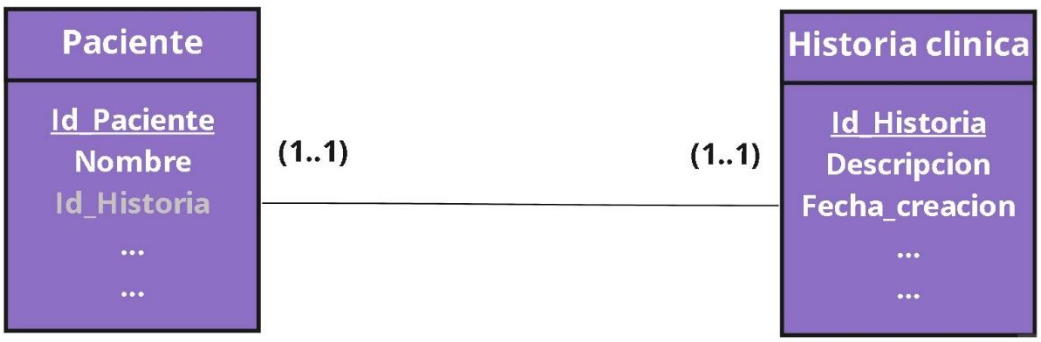
- Entidades: es cualquier elemento que nosotros necesitemos representar dentro del sistema.
- Atributos: son ciertas características o propiedades que se le atribuyen a cada entidad.
- Relaciones: es una asociación o vinculación entre varias entidades. Estas pueden tener atributos.

Tipos de relaciones

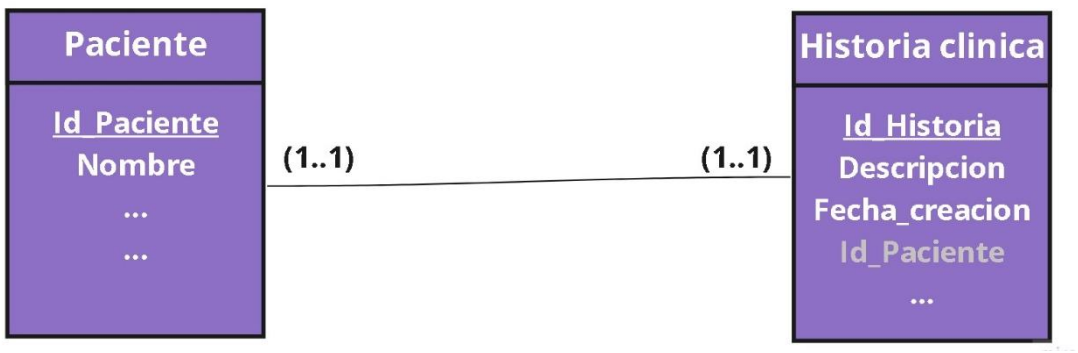
Uno a uno: esta es la más sencilla de todas, en donde nos imaginamos que para cada tupla de una tabla tiene su correlación con otra tupla de la otra tabla. La forma de diagramarlo es la siguiente, en este ejemplo tenemos una entidad paciente con sus atributos y tiene una historia clínica que es otra entidad también con sus atributos. Un paciente tiene una y solo una historia clínica, y la historia clínica le pertenece a uno y solo un paciente.



_ Como vemos la relación entre las entidades está representada por el rombo. Para traducir esta relación a la diagrama de clases. Tenemos dos opciones en el uno a uno, la opción uno es agregar a la tabla paciente, además de todos sus atributos, la clave Id_Historia, pero esto no es bueno porque si la historia clínica empieza a crecer, el paciente va a tener más información de la que debería tener:



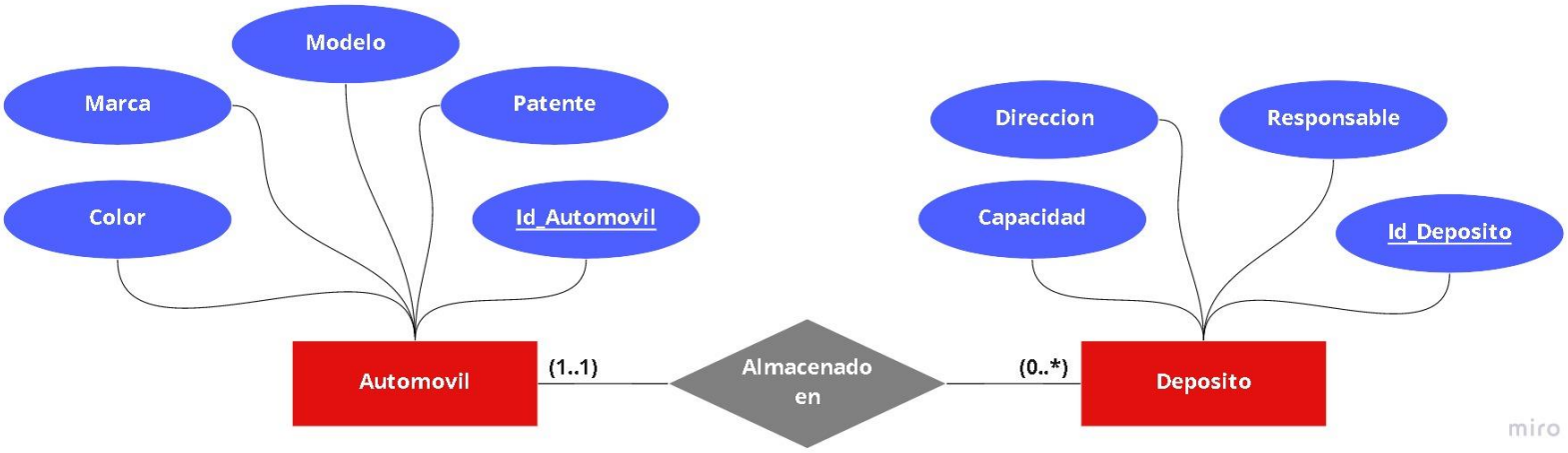
_ Ahora como son uno a uno, si al diagrama E-R lo leemos al revés, quedaría distinto por lo tanto en el diagrama de clases, a la historia clínica le agrego el Id_Paciente y con este buscamos el paciente y traemos la información que necesitemos.



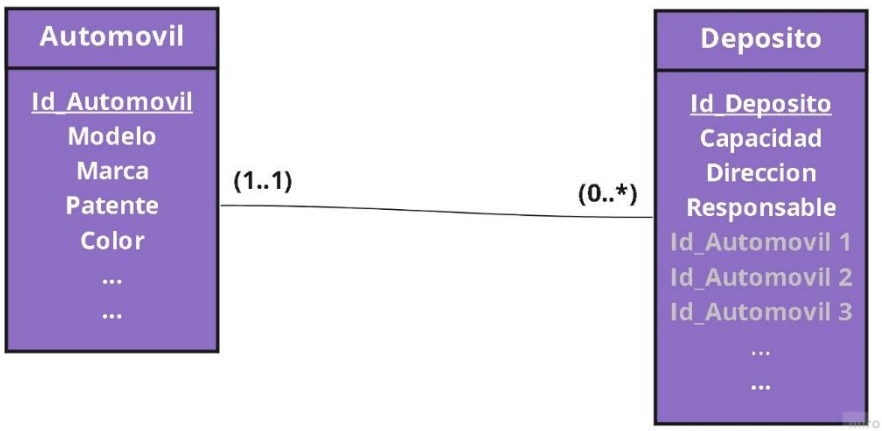
_ Es lo mismo de las dos formas, un paciente puede saber cuál es su historial médico, o una historial médico puede saber de qué paciente es.

_ En el diagrama E-R cada entidad tiene sus atributos, y después con el diagrama de clases transformamos la relación, en donde tenemos que tener en cuenta con que atributos vamos a unir a las entidades. Y como es uno a uno es indistinto que Id va a donde.

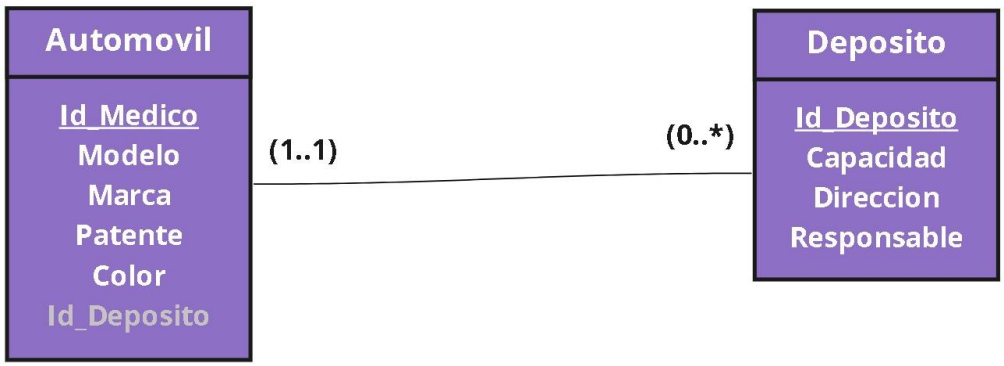
Uno a muchos: imaginamos por ejemplo que tenemos un depósito de autos, tenemos una entidad automóvil con sus atributos en donde un auto puede estar en uno y solo un depósito, ahora en el depósito que es otra entidad, puede estar vacio o puede tener muchos autos. A la hora de cargar un auto, el campo deposito si o si es obligatorio, y además debe elegir un solo lugar.



_ Ahora lo difícil de esto es llevarlo a un diagrama de clases. Una opción seria poner el Id de los autos en el depósito. Lo que pasa acá es que en ningún momento dijimos que los depósitos de autos tienen la misma cantidad de lugares, ya que pueden ser filas de 10 lugares, otro de 5, otro de 20, y lo que va a pasar es que vamos a tener un montón de lugares que existen pero que están vacíos o llenos, y asi desperdiciamos mucho espacio.

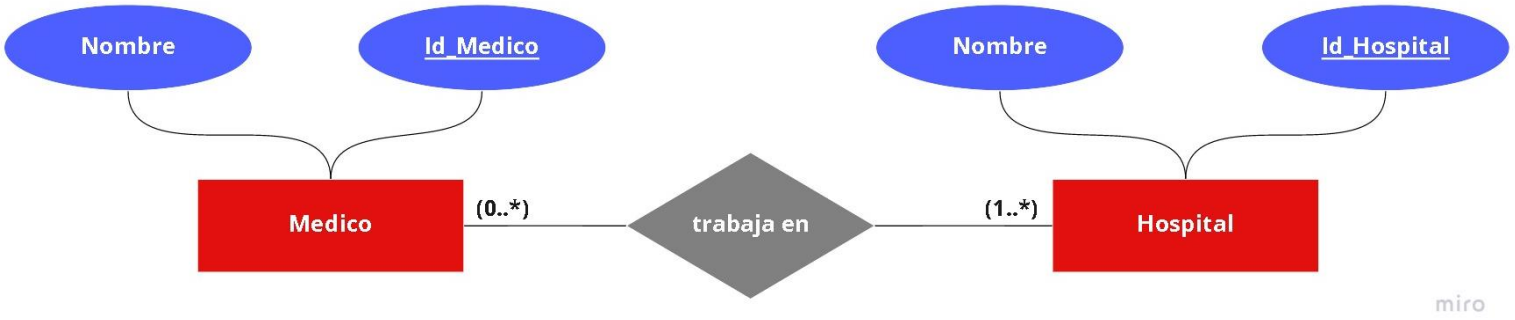


_ Ahora si lo resolvemos de la forma correcta, en la relación uno a muchos, a la entidad automóvil le asignamos un depósito, en donde cumplimos con que el auto este en un depósito, y el depósito no tiene idea que auto esta adentro porque no le importa en realidad, ya que lo que importa es que el auto sepa donde esta y no que el depósito sepa que autos tiene.



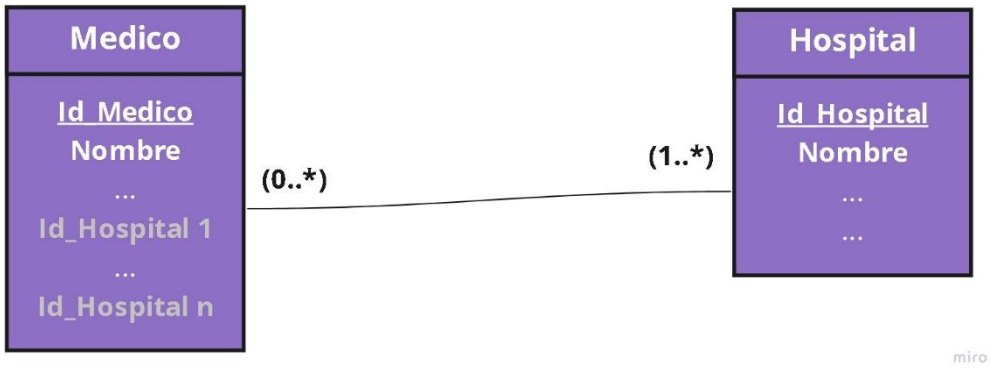
_ En resumen, cuando tenemos una relación uno a muchos, el Id que une las entidades va en la entidad que es uno a uno.

Muchos a muchos: en este ejemplo tenemos una entidad médico y una entidad hospital. Vemos que un médico puede no trabajar en ningún hospital o trabajar en muchos o en todos, o sea tenemos una tabla de médicos donde algunos están asignados a un hospital y algunos no, si en lugar de un 0 tenemos un 1 significa que en esa tabla no puede haber médicos que no tengan un hospital asignado ("los médicos tienen que estar al menos en un hospital" o "los hospitales tienen que tener al menos un médico", estos se traducen en 1, después "los médicos pueden tener un hospital asignado o no" esto es un 0, y si en ambos dicen que pueden tener a su vez muchos asignados es un muchos *). Además acá tenemos que un hospital si o si funciona con al menos un médico y puede tener muchos médicos asignados. Muchos médicos pueden estar relacionados con muchos hospitales.

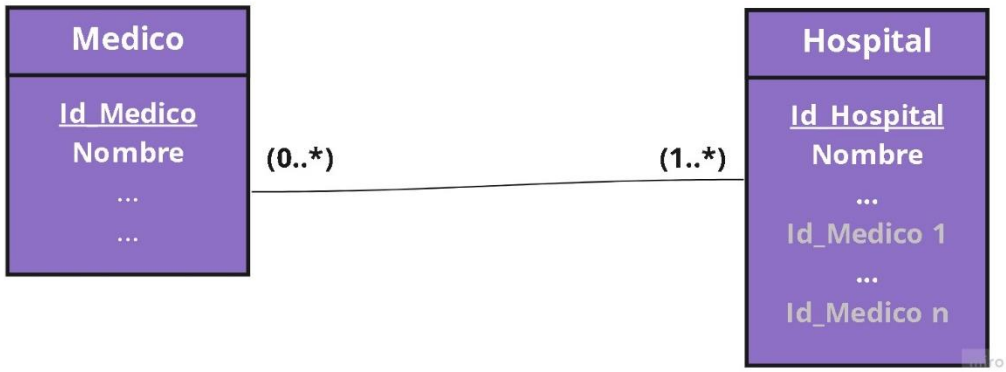


_ Ahora para hacer el diagrama de clases, determinamos donde ponemos el Id. La opción uno en este caso es poner el Id del lado del médico, pero un médico puede estar en uno, varios o todos los hospitales, entonces deberíamos poner todos los Id_Hospital donde el

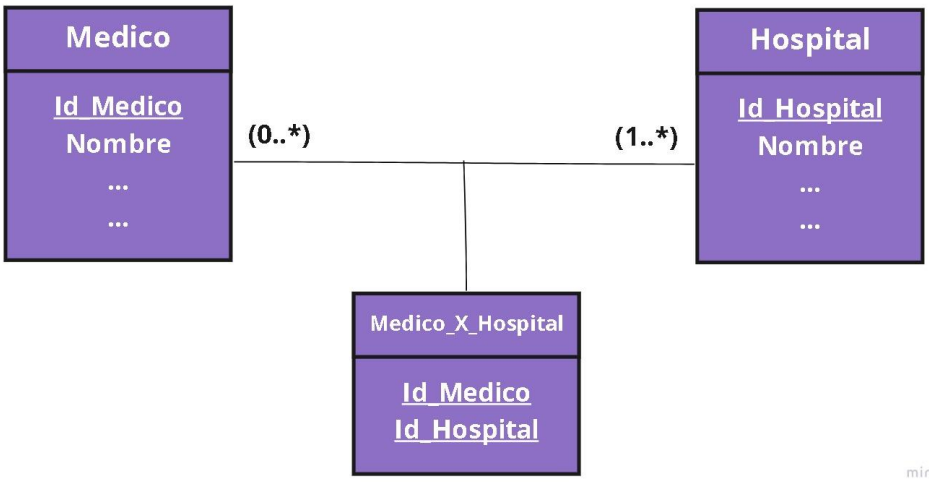
medico trabaja, en donde el medico que trabaje en un hospital va a tener lugares vacíos en la tabla, y el que trabaje en todos los va a tener llenos, hablando de tuplas.



_ La segunda opción seria al revés, colocar el Id de los médicos en la tabla hospital. Pero nos sucede lo mismo que el caso anterior, ya que no podemos llenar la tabla de Id_Medico.



_ La solución es utilizar una tabla pasarela , esta es una tabla intermedia que se crea, no es una entidad como tal, sino que es una tabla que me va a ayudar a unir a dos entidades cuando la relación es mucho a muchos. Entonces creamos una nueva tabla en donde el Id de la tabla y los datos que tiene son el Id del médico y el Id del hospital. Por ejemplo queremos buscar en que hospital trabaja el doctor Juárez, buscamos en la tabla de médicos el Id del mismo y una vez obtenido lo ponemos en la tabla pasarela y esta me trae los Id de los hospitales donde este médico trabaja, en donde con los Id de los hospitales le hacemos una consulta a la tabla hospital para que me de los nombres de los mismos. De la misma manera lo podemos hacer al revés, con buscar los médicos que trabajan en un cierto hospital.

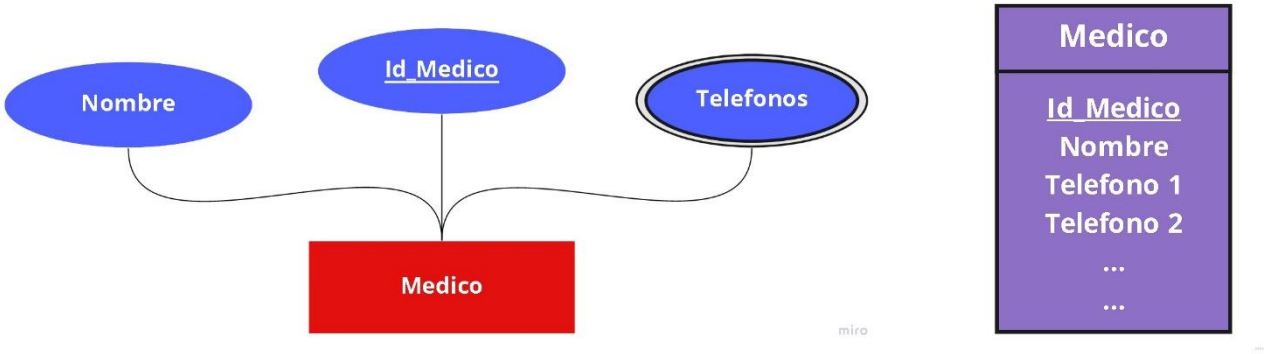


_ En las tablas pasarelas agarramos los Id de las tablas que vamos a unir. Siempre debemos subrayar los atributos que forman la clave primaria, y como vemos en este caso subrayamos ambas porque son la clave compuesta de las dos entidades que unimos, donde la combinación de ambas no se puede repetir.

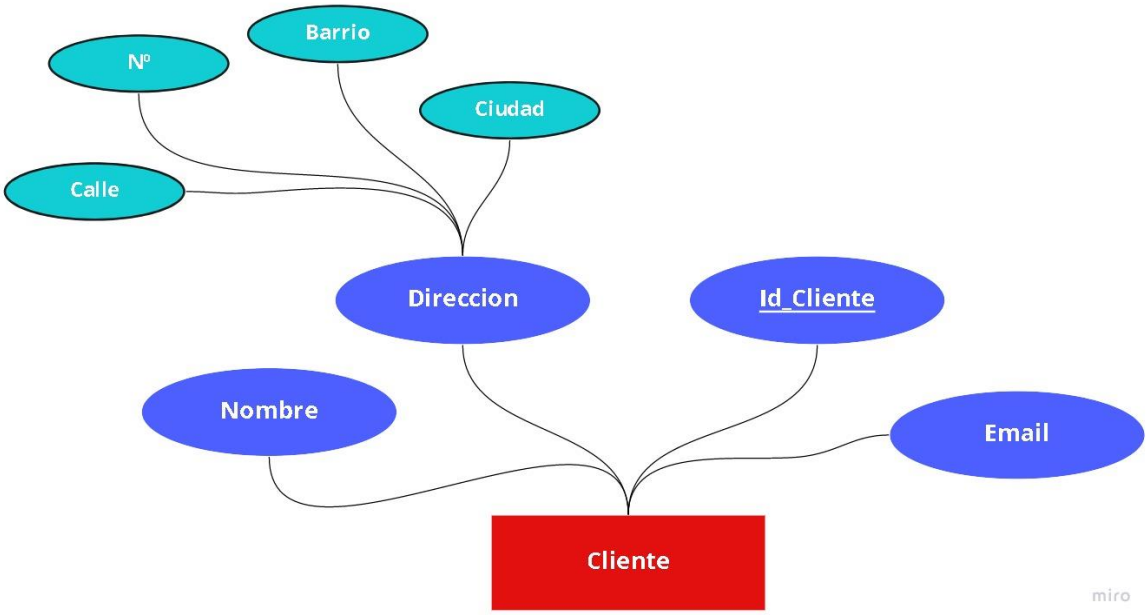
Atributos

_ Estos tipos de atributos son importantes porque hacen que la tabla o la base de datos no sea del todo performante, pero a nivel de diagrama E-R tenemos que dejar marcado porque es información que el cliente nos da.

Multivalorados: adquieren varios valores. Por ejemplo el cliente nos dice que quiere guardar tres mails o dos teléfonos por las dudas, siempre que estén en plural son multivalorados, y distinto es por ejemplo si nos dice teléfono móvil y teléfono fijo, donde nos especifica que tiene dos teléfonos distintos. Por lo general nosotros no sabemos cuánto es la cantidad de estos atributos, por el momento hacemos de cuenta que la cantidad es dos.

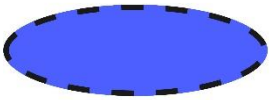


Compuestos: que están formados por elementos dentro. El atributo no es del todo atómico y se puede sub dividir en otras partes. Por ejemplo una dirección puede estar formada por la calle, numero, barrio, ciudad, etc, o también por ejemplo el nombre completo que está formado por nombre 1, nombre 2, apellido 1, apellido 2, etc. En el diagrama de clases ponemos los atributos que lo forman al atributo compuesto.



_ Asi mismo podemos combinar que un atributo sea multivalorado y a la vez compuesto, es decir, que tenga varias direcciones por lo tanto calle n, barrio n, Nº n, ciudad n. Pero cuando sucede esto vamos a asumir que es lo suficientemente importante como para que se transforme en una entidad, donde dentro de la entidad direcciones la llenamos de datos.

Derivados: son atributos donde su valor dependen de la relación con otra entidad o de otro atributo. Algunas veces queda plasmado en la base de datos y otras no porque no hace falta guardarlo porque va cambiando todo el tiempo. Estos se denotan con línea entre cortada. Un atributo es derivado entonces cuando su valor está determinado por el valor de otros atributos o por una relación. Básicamente que el dato no se carga como a mano. Estos no solo se calculan con otros atributos, sino que también se calculan con el resultado de una relación.



Ejercicio ejemplo

Sistema para control de vacunación (parte 1): (entidades, atributos y relaciones)

_ Se necesita un software para gestionar la aplicación de vacunas en la región norte del país. Para ello, es necesario diseñar la estructura de una base de datos relacional teniendo en cuenta los siguientes aspectos de dominio. El sistema debe gestionar los pacientes atendidos, los médicos asignados al plan de vacunación, las vacunas utilizadas y las aplicaciones de las mismas. Es necesario llevar un registro de los pacientes que fueron vacunados y los que serán vacunados en el futuro. De ellos se necesita saber su nombre, apellido, DNI y fecha de nacimiento. De las vacunas disponibles, se necesita saber el nombre de la misma y la fecha de vencimiento. De los médicos, se debe registrar su DNI, matricula profesional, nombre y apellido. Ahora:

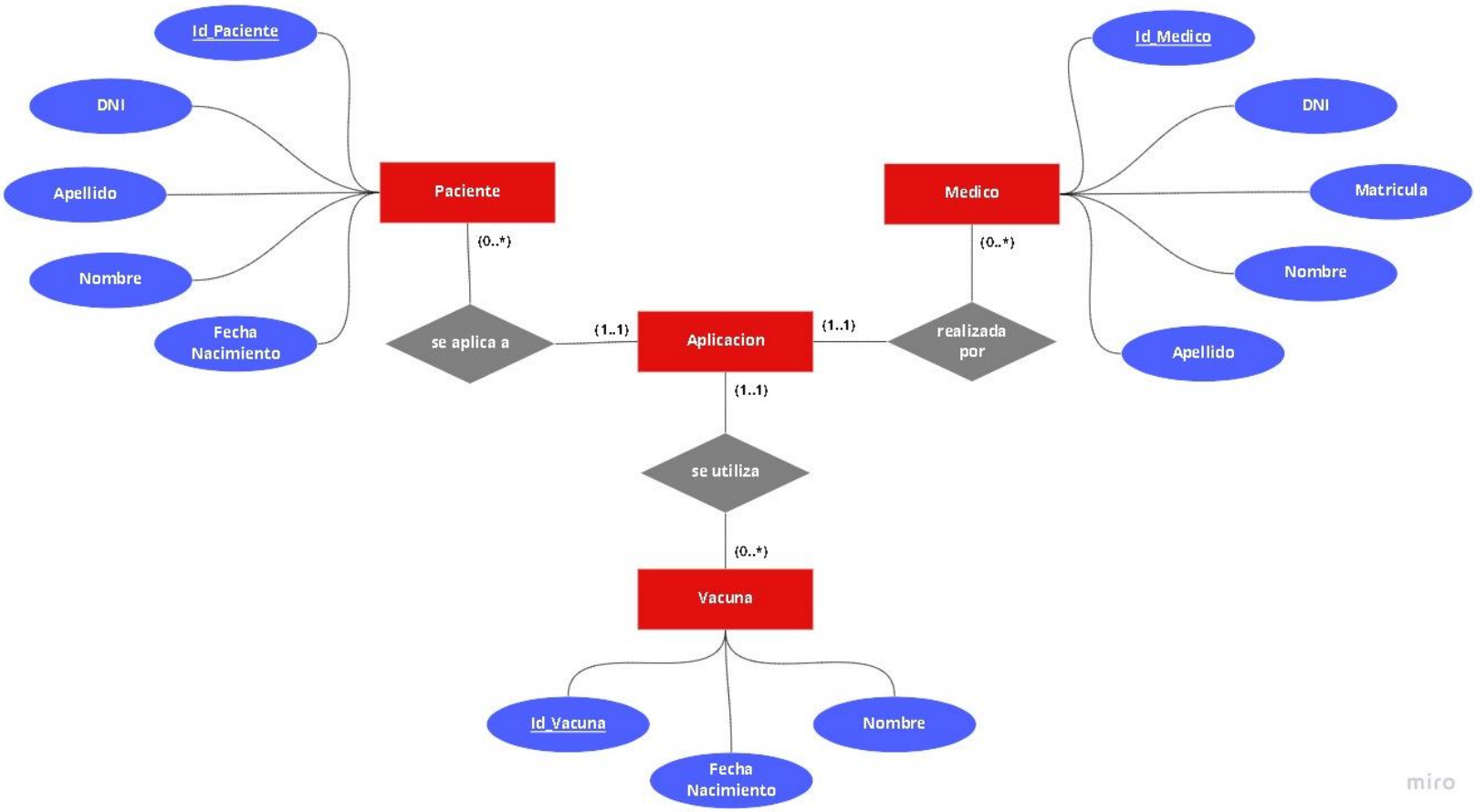
- Cada aplicación es realizada por uno y solo un médico. Un médico no puede haber colocado ninguna vacuna o varias de ellas.
- En cada aplicación se utiliza una y solo una dosis de una vacuna. Una vacuna determinada puede haberse aplicado en varias aplicaciones o aun no haber sido usada.
- Cada aplicación es realizada a una y solo un paciente. Un paciente puede haber recibido varias aplicaciones, todas las disponibles o aún ninguna.

Respuesta:

- Entidades: pacientes, médicos, vacunas y aplicación.
- Atributos:
  - Pacientes: nombre, apellido, DNI y fecha de nacimiento.
  - Vacunas: nombre y la fecha de vencimiento.
  - Médicos: DNI, matricula profesional, nombre y apellido.
- Relaciones:
  - La aplicación se le aplica a un paciente.
  - La aplicación es realizada por un médico.
  - En la aplicación se utiliza una vacuna.

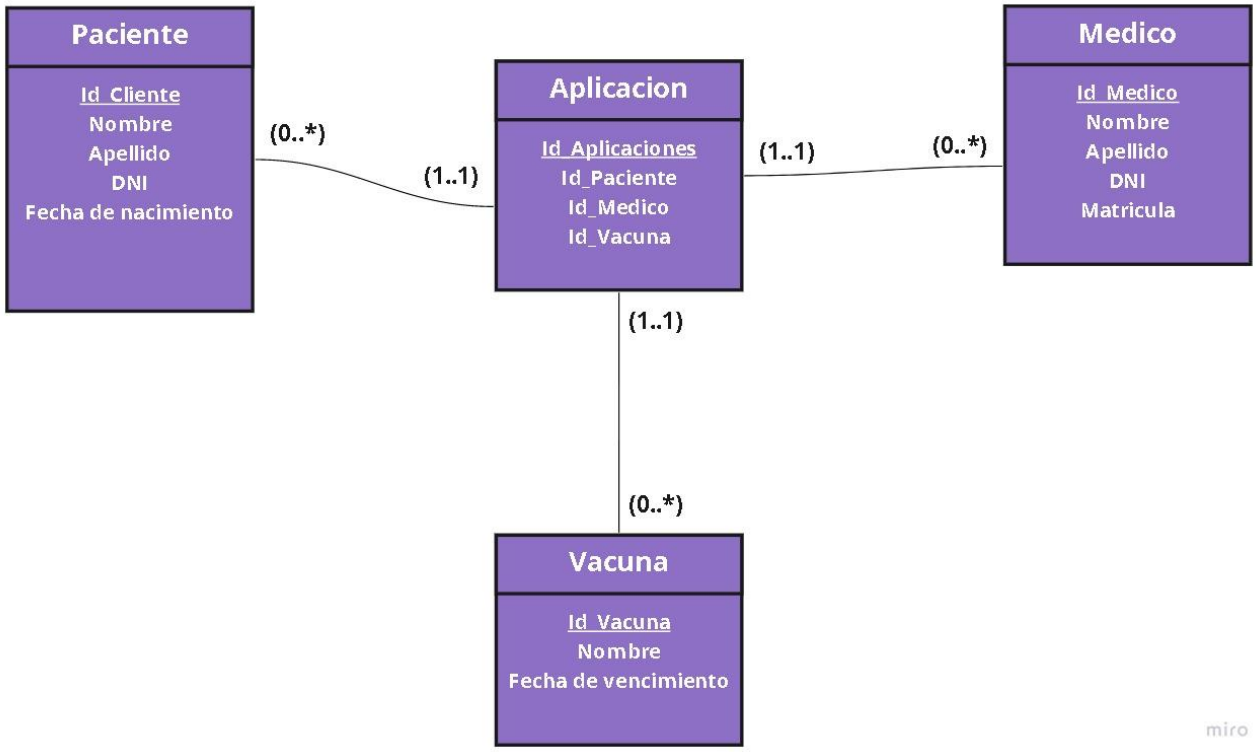
_ Hay muchos atributos que los carga una persona, por ejemplo el DNI, y a veces surge el por qué no usar el DNI como clave primaria, y lo que sucede por ahí es que al DNI se lo puede ingresar mal y nos puede generar un problema, hasta por ahí se repite el número, entonces es importante colocarles a las entidades un Id que no se repita, y que si se borra que desaparezca, para tener un control univoco sobre estas.

_ El diagrama E-R seria:



_ Para agregar, el estado de vacunación del paciente puede ser un atributo derivado, pero en este caso no es relevante agregarlo.

_ Y por último el diagrama de clases seria:



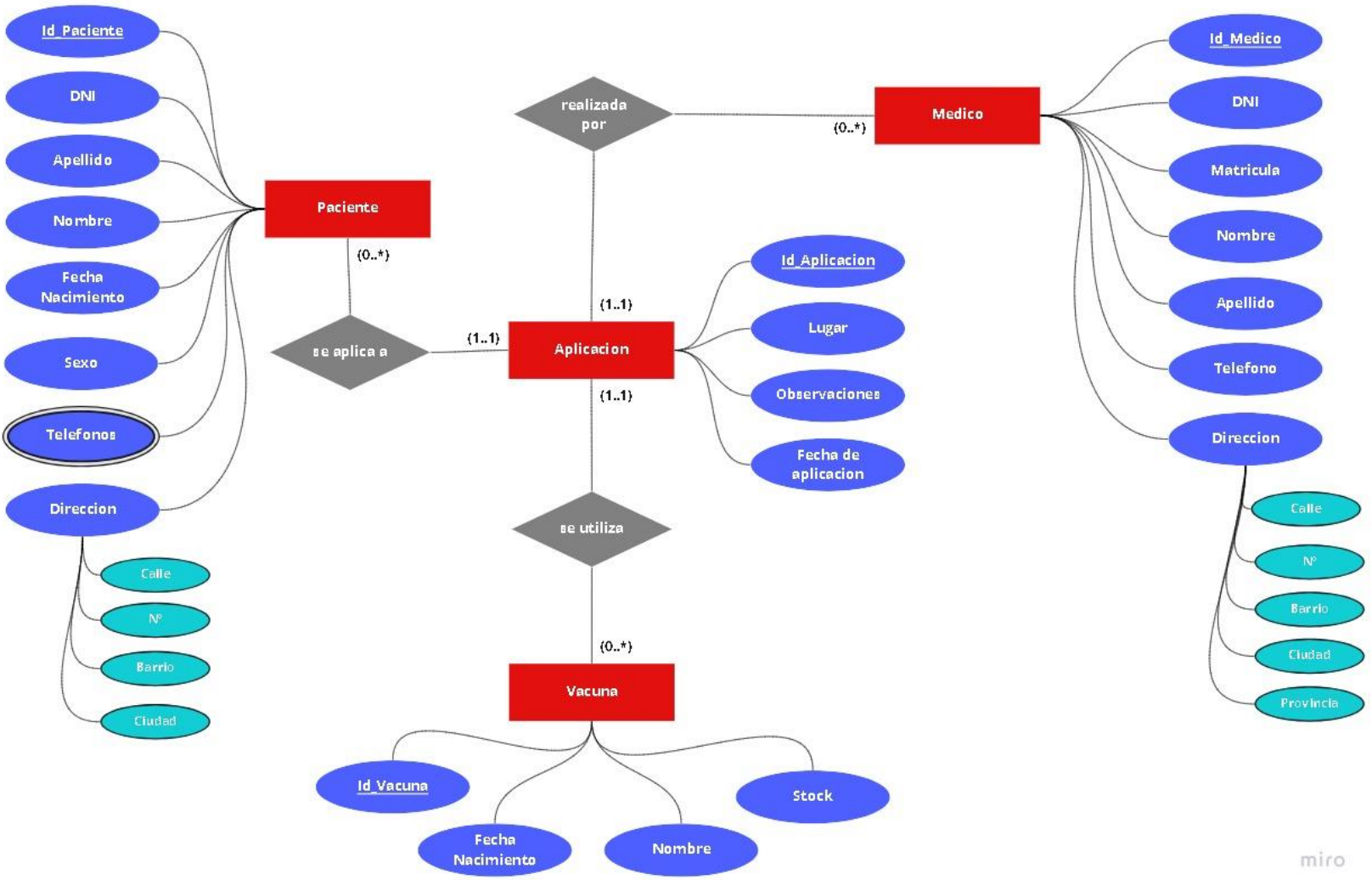
_ Ahora tenemos nuevos requerimientos, que obligan a cambiar el diseño propuesto. A medida que medida que vamos avanzando en el desarrollo de los softwares los requerimientos pueden cambiar.

Modificación de requerimientos (parte 2): el sistema debe gestionar los pacientes atendidos, los médicos asignados al plan de vacunación, las vacunas utilizadas y las aplicaciones de las mismas. Es necesario llevar un registro de los pacientes que fueron vacunados y los que serán vacunados en el futuro. Los datos necesarios de los mismos son los siguientes, nombre, apellido, DNI, fecha de nacimiento, teléfonos de contacto, dirección y sexo. De las vacunas disponibles, se necesita saber el nombre de la misma, su fecha de vencimiento, su fecha de utilización, la cantidad de dosis disponibles de cada una de ellas. De los médicos, se debe registrar su DNI, matrícula de profesional, nombre, apellido, teléfono de contacto, dirección. Es necesario registrar cada aplicación y dejar asentado, que medico la realizo, que vacuna fue utilizada, a que paciente se le realizo la aplicación, la fecha de aplicación, lugar donde se realizó la misma, observaciones y un identificador de cada una de las aplicaciones. Ahora:

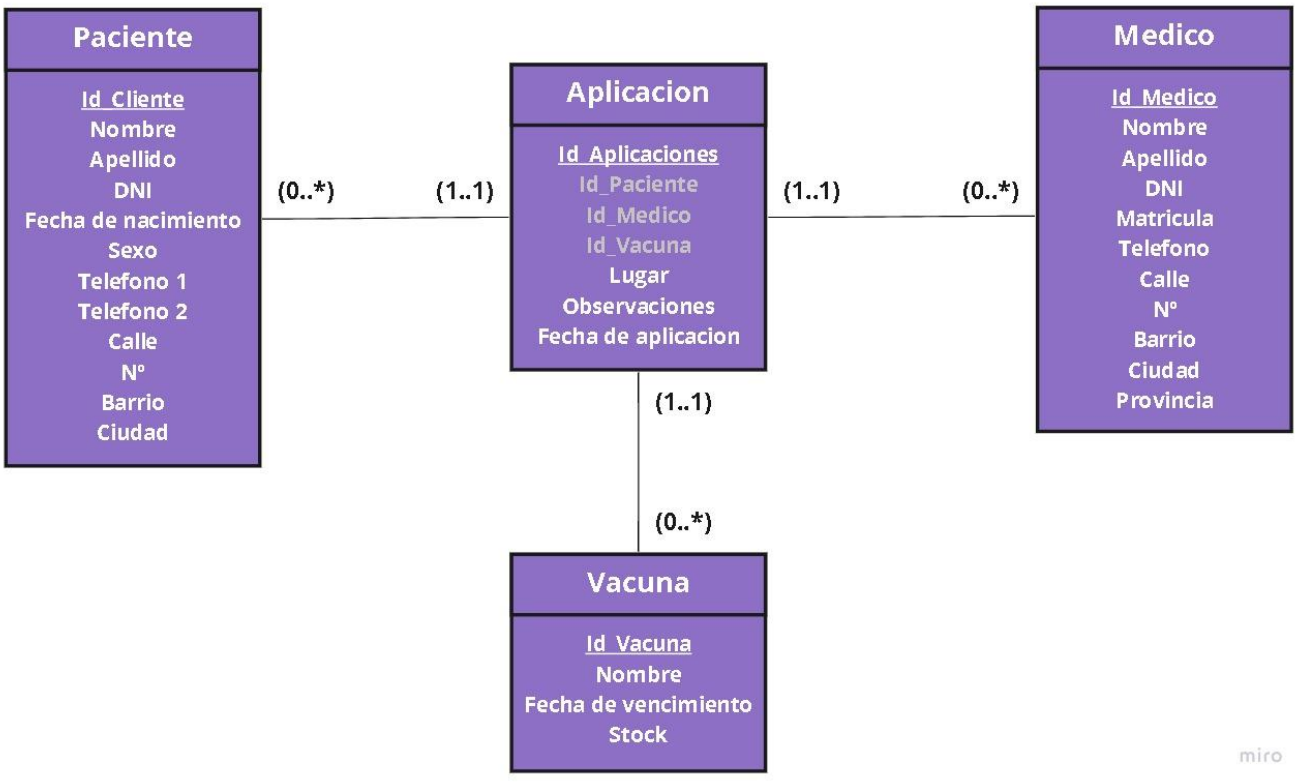
- Cada aplicación es realizada por uno y solo un médico. Un médico puede no haber colocado ninguna vacuna o varias de ellas.
- En cada aplicación se utiliza una y solo una dosis de una vacuna. Una vacuna determinada puede haberse aplicado en varias aplicaciones o aun no haber sido usada.
- Cada aplicación es realizada a uno y solo un paciente. Un paciente puede haber recibido varias aplicaciones, todas las disponibles o aún ninguna.

Asumimos que la dirección de los pacientes está formada por calle, Nº, barrio y ciudad. Asumimos que la dirección de los médicos está formada por calle, Nº, barrio, ciudad y provincia.

_ El diagrama E-R seria:



_ Y a continuación vemos el diagrama de clases, donde los rombos entre aplicación y médico y cliente se ven reflejados con los Id dentro del cuadro de aplicación:



_ Siempre es conveniente realizar el diagrama E-R antes del diagrama de clases, donde ambos tienen que tener la misma información. En este diagrama vemos que todas las relaciones son uno a muchos o muchos a uno dependiendo del punto de vista, es por eso que todos los Id de las entidades están en el que es uno a uno, que en este caso es la entidad aplicación.

_ Como regla practica debemos prestar atención cuando se mencionan los atributos por ejemplo, no es lo mismo, el usuario tiene nombre apellido, etc, que son atributos distintos, o cuando, el usuario tiene nombre completo formado por nombre apellido, etc, que sería un atributo compuesto. También por ejemplo, el usuario tiene teléfono fijo, teléfono móvil, etc, o decir que, el usuario tiene teléfonos, donde esto sería un atributo multivalorado.

_ Si el cliente nos dice que una entidad tiene como mucho una cosa, lo interpretamos como una relación 0 a 1 (0..1).

Dependencias funcionales

_ Las claves primarias son un conjunto de atributos, y son una forma univoca de identificar a las tuplas de una entidad, en donde esto al fin y al cabo es una restricción porque evita que subamos otra tupla con la misma clave primaria. Además de esa restricción, también hay otro tipo de restricciones que son restricciones para el valor que puede tomar un atributo. Es decir, las dependencias funcionales permiten expresar restricciones que identifican de manera univoca el valor de determinado atributo, además podemos decir que es una conexión entre uno o más atributos.

_ Si tenemos por ejemplo A→C significa que el valor que puede tomar C está determinado por A, o que A determina a C, o que C depende funcionalmente de A. en donde siempre que yo tenga un valor x de A, vamos a tener el mismo valor en C.

_ R es el conjunto de los atributos de la entidad, en donde si X determina a R, X es la clave primaria, y esta evita de que entren nuevos atributos. Esta no se puede repetir en las tuplas.

_ Necesitamos ver las dependencias funcionales porque son el paso siguiente a la normalización.

Normalización

_ Cuando hacíamos los diseños de base de datos, entendemos que los atributos del usuario es muy probable que vayan con la entidad usuario. Ahora vemos otra forma de ordenar los datos, en donde esto es otra forma de ordenar los datos que nos permite abstraernos del dato en sí.

_ Existen un montón de formas normales. Las formas normales son el estado al que llega un sistema de gestión de base de datos. Y dependiendo de lo que estemos desarrollando, dependiendo las reglas que pongamos para el diseño, o que consideremos que está bien o está mal, vamos a ser más o menos estrictos en las formas normales, donde las formas normales le agregan reglas para que la base de datos este correcta.

_ Nosotros vamos a considerar que una base de datos esta normalizada, si está en forma normal de Boyce-Codd (todas sus entidades). Y para que este en forma normal de Boyce-Codd antes tiene que estar en primera forma normal.

Primera forma normal (1FN)

_ Un modelo, una entidad, o tabla, está en primera forma normal si todos sus atributos son atómicos. Y con atómicos hacemos referencia a que no se puedan subdividir teniendo en cuenta el dominio, en donde por ejemplo claramente cuando hay atributos compuestos o multivalorados, hace que el modelo no esté en primera forma normal. Esto va a depender de lo que termine necesitando el cliente.

_ En la siguiente tabla/entidad vemos que no está en 1FN porque estamos forzando a la entidad a tener dos emails juntos:

ID	Nombre	Puesto	Salario	Emails
111	Juan Pérez	Jefe de Área	3000	juan@hotmail.com; juan@gmail.com;
222	José Sánchez	Administrativo	1500	jsanchez@outlook.com
333	Ana Díaz	Administrativo	1500	adiaz@gmail.com; DiazAna@live.com;

_ Por ende una forma medio incorrecta de resolver esto es duplicar los registros con valores repetidos. Donde el atributo email quedo atómico, y quedo todo en 1FN. Además utilizando esta opción, nos vemos obligado a usar como clave primaria (ID , Emails).

ID	Nombre	Puesto	Salario	Emails
111	Juan Pérez	Jefe de Área	3000	juan@hotmail.com
111	Juan Pérez	Jefe de Área	3000	juan@gmail.com
222	José Sánchez	Administrativo	1500	jsanchez@outlook.com
333	Ana Díaz	Administrativo	1500	adiaz@gmail.com
333	Ana Díaz	Administrativo	1500	DiazAna@live.com

_ En esta forma estamos arruinando la tabla primaria, el espacio en la tabla, pero lo más importante es que si necesitamos hacer un cambio en una tupla, debemos hacerlo en todas las tuplas. Por lo tanto la repetición de los datos puede generar una inconsistencia.

_ La otra forma es separar el atributo que viola 1FN en una tabla, en donde agarramos el atributo que era multivalorado, lo llevamos a una tabla nueva y con el Id del empleado, unimos los mails con ambos. Entonces por un lado en la tabla principal no tenemos más el atributo que no era atómico, y le dimos atomicidad al separarlo. Pero lo importante de la nueva entidad que creamos es saber cuál es la clave primaria de la entidad , ya que si usamos como clave primaria el Id no vamos a poder cargar nuevos datos de los que están, es por eso que usamos una clave compuesta que es el Id y el email.

ID	Nombre	Puesto	Salario
111	Juan Pérez	Jefe de Área	3000
222	José Sánchez	Administrativo	1500
333	Ana Díaz	Administrativo	1500

Id_ empleado	Email
111	juan@hotmail.com;
111	juan@gmail.com;
222	jsanchez@outlook.com;
333	adiaz@gmail.com;
333	DiazAna@live.com;

_ Entonces cuando tenemos un atributo multivalorado creamos una nueva entidad.

_ Ahora para el caso de los atributos compuestos, creamos un atributo más, es decir, los subdividimos. Por ejemplo:

Nro. Pedido	Cód. Vendedor	Cód. Producto	Nombre y apellido	Sucursal	Horario
1310001	001	PR01	**Juan, Gonzales**	S1	Turno 1
1310002	002	PR03	**Karina, Fasetta**	S2	Turno 1
1310003	003	PR03	**Raúl, Gonzales**	S1	Turno 2
1311001	001	PR01	**Juan, Gonzales**	S1	Turno 1
1311002	002	PR02	**Karina, Fasetta**	S2	Turno 1
1311002	002	PR02	**Karina, Fasetta**	S2	Turno 1
1312001	001	PR02	**Juan, Gonzales**	S1	Turno 1

_ El resultado sería:

Nro. Pedido	Fecha	Cód. Producto	Cód. Vendedor	Nombre	Apellido	Sucursal	Horario
001	10/13	PR01	001	Juan	Gonzales	S1	Turno 1
002	10/13	PR03	002	Karina	Fasetta	S2	Turno 1
003	10/13	PR03	003	Raúl	Gonzales	S1	Turno 2
001	11/13	PR01	001	Juan	Gonzales	S1	Turno 1
002	11/13	PR02	002	Karina	Fasetta	S2	Turno 1
003	11/13	PR02	002	Karina	Fasetta	S2	Turno 1
001	12/13	PR02	001	Juan	Gonzales	S1	Turno 1

_ Una vez que el modelo ya está en primera forma normal, empezamos a buscar dependencias funcionales. Podemos determinar dependencias funcionales falsas si no conocemos los datos, por ejemplo turno determina a la fecha, Nº de documento me determina código postal, apellido determina sucursal, en donde no hay relación lógica. Pero en realidad tenemos dos dependencias funcionales en el ejemplo pasado:

- Nro. pedido, Fecha → Nro. Pedido, Fecha, Cód.. producto, Cód.. Vendedor, Nombre, Apellido, Sucursal, Horario
- Cód. Vendedor → Nombre, Apellido, Sucursal, horario

_ Es importante recordar que las tuplas nuevas que aparezcan también tienen dependencias funcionales. Es de todo el modelo no de la primer tabla.

_ Ahora con las dependencias funcionales trabajamos con Boyce-Codd.

Forma normal de Boyce-Codd (FNBC)

_ Decimos que una entidad está en forma normal de Boyce-Codd si sus dependencias funcionales cumplen al menos una de las siguientes condiciones:

- α es súper clave de R (clave primaria)
- $\alpha \rightarrow \beta$ es una dependencia funcional trivial

_ Si tenemos una relación R con una dependencia funcional NO TRIVIAL $\alpha \rightarrow \beta$ y que además no sea súper clave, aplicamos estas dos reglas:

- ($\alpha \cup \beta$): creamos una nueva entidad donde alfa y beta son las dos partes de la dependencia funcional.

- (R – (β − α)): y al final a todo R le quito beta menos alfa, donde el 80% de las veces es solo beta.

_ Por ejemplo tomamos la dependencia funcional del ejemplo anterior, que no es ni clave primaria ni trivial:

Cód.. Vendedor → Nombre, Apellido, Sucursal, horario

_ Aplicamos la regla de unión:

$(\alpha \cup \beta)$

Cód. Vendedor	Nombre	Apellido	Sucursal	Horario
001	Juan	Gonzales	S1	Turno 1
002	Karina	Fasetta	S2	Turno 1
003	Raúl	Gonzales	S1	Turno 2
001	Juan	Gonzales	S1	Turno 1
002	Karina	Fasetta	S2	Turno 1
002	Karina	Fasetta	S2	Turno 1
001	Juan	Gonzales	S1	Turno 1

_ Luego a R le restamos beta menos alfa, y en este caso solo restamos beta:

$(R - (\beta - \alpha))$

Nro. Pedido	Fecha	Cód. Producto	Cód. Vendedor
001	10/13	PR01	001
002	10/13	PR03	002
003	10/13	PR03	003
001	11/13	PR01	001
002	11/13	PR02	002
003	11/13	PR02	002
001	12/13	PR02	001

_ Donde las dependencias funcionales son:

- Nro. pedido, Fecha → Nro. Pedido, Fecha, Cod. producto, Cód.. Vendedor
- Cód.. Vendedor → Nombre, Apellido, Sucursal, horario

_ Cada vez que aplicamos Boyce-Codd el R se va achicando. En la normalización no usamos los datos, sino que usamos los atributos.

Axiomas dependencias funcionales:

- Sí $\beta \subseteq \alpha$, entonces $\alpha \rightarrow \beta$ (reflexividad)
- Sí $\alpha \rightarrow \beta$, entonces $\gamma\ \alpha \rightarrow \gamma\ \beta$ (aumentatividad)
- Sí $\alpha \rightarrow \beta$, y $\beta \rightarrow \gamma$, entonces $\alpha \rightarrow \gamma$ (transitividad)

Ejemplo:

- R = (A, B, C, G, H, I) y F = { A $\rightarrow$ B A $\rightarrow$ C CG $\rightarrow$ H CG $\rightarrow$ I B $\rightarrow$ H}

_ Algunos miembros de F+:

- A $\rightarrow$ H: por transitividad de A $\rightarrow$ B and B $\rightarrow$ H.
- AG $\rightarrow$ I 4 por aumentatividad A $\rightarrow$ C con G, para lograr que AG $\rightarrow$ CG y después transitividad con CG $\rightarrow$ I.
- CG $\rightarrow$ HI 4 por aumentatividad CG $\rightarrow$ I se infiere CG $\rightarrow$ CGI, y por aumentatividad de CG $\rightarrow$ H se infiere CGI $\rightarrow$ HI, y después por transitividad.

Ejercicio ejemplo normalización

Turno_empleado	
Id_empleado	
Categoria_libro	
Nombre_libro	
Direccion_alumno (Calle, Nro, Barrio)	Compuesto
Nombre_completo_alumno (Nombre, Apellido)	Compuesto
Fecha_devolucion	
ID	
DNI_alumno	
Editorial	
Estado_prestamo	
Nombre_empleado (Nombre, Apellido)	Compuesto
Telefono_alumno	Multivalorado
Id_libro	
Fecha_inicio_prestamo	
Cantidad_ejemplares	
Autores (Autor 1, Autor 2)	Compuesto
Email_alumno	Multivalorado
Id_alumno	

_ Los atributos multivalorados desaparecen de la entidad principal y creamos dos entidades con esos atributos. Y con los atributos completos se agregan.

_ Primera forma normal:

Turno_empleado
Id_empleado
Categoria_libro
Nombre_libro
Direccion_alumno_Calle
Direccion_alumno_Nro
Direccion_alumno_Barrio
Nombre_alumno
Apellido_alumno
Fecha_devolucion
ID
DNI_alumno
Editorial
Estado_prestamo
Nombre_empleado
Apellido_empleado
Telefono_alumno
Id_libro
Fecha_inicio_prestamo
Cantidad_ejemplares
Autor 1
Autor 2
Email_alumno
Id_alumno

Id_alumno
Telefono_alumno

Id_alumno
Email_alumno

_ Las dependencias funcionales en la **entidad principal** son:

- ID→R

- Id_alumno→Nombre_alumno, Apellido_alumno, DNI_alumno, Direccion_alumno_Calle, Direccion_alumno_Nro, Direccion_alumno_Barrio.

- Id_Libro→Nombre_libro, Categoria_libro, Autor, 1, Autor 2, Editorial, Cantidad_ejemplares.

- Id_Empleado→Nombre_empleado, Apellido_empleado, turno_empleado.

_ Dependencias funcionales **entidad telefono_alumno**:

- Id_alumno, Telefono_alumno → Id_alumno, Telefono_alumno

_ Dependencias funcionales **entidad email_alumno**:

- Id_alumno, Email_alumno → Id_alumno, Email_alumno

_ Las dependencias que no cumplen con la FNBC son:

- Id_alumno→Nombre_alumno, Apellido_alumno, DNI_alumno, Direccion_alumno_Calle, Direccion_alumno_Nro, Direccion_alumno_Barrio.

- Id_Libro→Nombre_libro, Categoria_libro, Autor, 1, Autor 2, Editorial, Cantidad_ejemplares.

- Id_Empleado→Nombre_empleado, Apellido_empleado, turno_empleado.

_ Y el resto son claves primarias, en las dos entidades nuevas además son triviales porque se repiten.

_ La solución para Id_alumno seria:

(α ∪ β)

Direccion_alumno_Calle
Direccion_alumno_Nro
Direccion_alumno_Barrio
Nombre_alumno
Apellido_alumno
DNI_alumno
Id_alumno

(R – (β − α))

Turno_empleado
Id_empleado
Categoria_libro
Nombre_libro
Fecha_devolucion
ID
Editorial
Estado_prestamo
Nombre_empleado
Apellido_empleado
Id_libro
Fecha_inicio_prestamo
Cantidad_ejemplares
Autor 1
Autor 2
Id_alumno

_ La solución para Id_libro seria:

(α ∪ β)

Categoria_libro
Nombre_libro
Editorial
Id_libro
Cantidad_ejemplares
Autor 1
Autor 2

(R – (β − α))

Turno_empleado
Id_empleado
Fecha_devolucion
ID
Estado_prestamo
Nombre_empleado
Apellido_empleado
Id_libro
Fecha_inicio_prestamo
Id_alumno

_ La solución para Id_empleado seria:

(α ∪ β)

Turno_empleado
Id_empleado
Nombre_empleado
Apellido_empleado

(R – (β − α))

Id_empleado
Fecha_devolucion
ID
Estado_prestamo
Id_libro
Fecha_inicio_prestamo
Id_alumno

Resultado: el nuevo modelo seria:

Id_empleado
Fecha_devolucion
ID
Estado_prestamo
Id_libro
Fecha_inicio_prestamo
Id_alumno

- ID→R

Direccion_alumno_Calle
Direccion_alumno_Nro
Direccion_alumno_Barrio
Nombre_alumno
Apellido_alumno
DNI_alumno
Id_alumno

- Id_alumno→Nombre_alumno, Apellido_alumno, DNI_alumno, Direccion_alumno_Calle, Direccion_alumno_Nro, Direccion_alumno_Barrio

Categoria_libro
Nombre_libro
Editorial
Id_libro
Cantidad_ejemplares
Autor 1
Autor 2

- Id_Libro→Nombre_libro, Categoria_libro, Autor, 1, Autor 2, Editorial, Cantidad_ejemplares

Turno_empleado
Id_empleado
Nombre_empleado
Apellido_empleado

- Id_Empleado→Nombre_empleado, Apellido_empleado, turno_empleado

Id_alumno
Telefono_alumno

- Id_alumno, Telefono_alumno → Id_alumno, Telefono_alumno

Id_alumno
Email_alumno

- Id_alumno, Email_alumno → Id_alumno, Email_alumno