Unidad 2

Base de datos relacionales

Gestión de bases de datos

Índice

Introducción

Los primeros atisbos del diseño relacional de bases de datos se dan en los años 60 y 70 por parte del investigador de IBM Edgar Frank Codd. Este publicará un trabajo al que da el nombre de "un modelo relacional de datos para grandes bancos de datos compartidos"; donde propone una serie de instrucciones lógicas y algebraicas que nos ayudan a poder gestionar de manera eficiente la gestión e la información dentro de las bases de datos.

Este modelo tiene su principal característica en las relaciones que se establezcan entre los datos, los que nos definen la estructura lógica necesaria para entender de qué manera se encuentra almacenada la información. Además, el modelo relacional nos hace ver la base de datos como una serie de relaciones que se manipulan usando un lenguaje relacional estándar. Este lenguaje es el llamado SQL (Structured Query Language) que se basa en la realización de consultas y se encuentra implementado en los sistemas de bases de datos más importantes, los llamado SGBD (Sistemas Gestores de Bases de Datos).

Los SGBD nos ayudan a que haya una comunicación entre la base de datos, el usuario, y las aplicaciones que necesiten del uso de esta. Además, estos sistemas proporcionan distintos beneficios en cuanto a seguridad, integridad de los datos y manejo de funcionalidades se refiere.

Los objetivos que Codd escribe en su trabajo son los siguientes:

- **Independencia lógica y física de los datos.** Cada vez que haya un cambio físico de almacenamiento las aplicaciones y usuarios deben de notar total transparencia. Así mismo, los cambios en los datos no implicarán un cambio en la estructura de las aplicaciones.
- **Flexibilidad y seguridad.** Dependiendo de que usuario y con que aplicación se recupere la información almacenada, se mostrará de distintas maneras.
- **Estandarización lógica.** Organizaremos los datos en tablas que entre ellas comparten la misma filosofía de creación y estarán relacionadas entre ellas también.
- **Simplicidad.** Este modelo pretende que haya una mayor simplicidad a la hora del manejo de los datos frente a las demás opciones además de usar un lenguaje lo más natural dentro de lo que cabe.

Al finalizar esta unidad

- Ubicaremos el diseño de bases de datos relacionales en el marco del análisis de desarrollo del software.
- Nos familiarizaremos con el concepto de diccionario de datos.
- Comprenderemos la utilidad del diagrama entidad/relación como herramienta primordial en el diseño de bases de datos relacionales.
- Dominaremos los conceptos y la terminología asociados al diagrama entidad/relación.
- Describiremos apropiadamente la cardinalidad y modalidad de una relación.
- Conoceremos los elementos que definen el diagrama entidad/relación extendida.

2.1. Modelización conceptual del software

Todos los sistemas de información parten de un problema a solucionar. Para poder solucionar esta problemática es necesario que se realice un examen de las necesidades que tenemos que cubrir.

Roger S. Pressman propuso una metodología para efectuar dicho análisis. En este análisis se toma el diccionario de datos como centro de arquitectura y se centra en el contexto relacional, dividiendo la problemática en tres distintos ámbitos representados mediante herramientas gráficas:

- **Descripción de objetos de datos:** se representan los objetos de datos y sus relaciones, Se usa el diagrama de entidad/relación (DER).
- **Especificación de proceso:** identificamos todas las funciones que debe desarrollar el sistema y como se relacionan entre ellas además de la forma en la que la información va a ser transformada cuando pase por dichas funciones. Esto se expresa con el diagrama de flujo de datos (DFD).
- **Especificación de control:** va mostrando los estados en los que el sistema se va a encontrar a medida que avanzamos en el proceso, así como las transiciones que hacen que estos estados cambien basándose en el diagrama de transición de estados (DTE).

Imagen 1. Metodología de Roger S. Pressman

En el temario de base de datos y dado a que es la problemática que nosotros tenemos, nos vamos a centrar en el primero.

2.1.1. Modelización de datos

A la hora del proceso de desarrollo del sistema de información se irán modelando los datos en varias fases:

1. Creamos el modelo conceptual de datos nada más comenzar con el proceso de análisis.
2. Damos forma a este modelo con el diagrama entidad/relación, conformando un modelo lógico de datos.
3. Cuando comencemos a diseñar el *software*, iremos perfeccionando el diagrama entidad/relación, pasando a ser un modelo físico de datos.
4. De manera final, implantaremos este modelo en nuestro sistema gestor de bases de datos.

2.1.2. Diccionario de datos

El diccionario de datos o *metabase* se encarga de almacenar los datos sobre los datos, los llamado metadatos. El diccionario de datos en una herramienta de uso recurrente para un desarrollador de software a la hora de encontrar información sobre distintos elementos de una base de datos. En este diccionario se categorizan los datos de manera lógica, incluyendo descripciones, significado, estructura y consideraciones de edición, uso y seguridad de dichos datos.

```
...
Tabla: TDatos
Campos:
        cDNI                    CHAR(9)                 NOT NULL UNIQUE
        cNombre                 VARCHAR(30)             NOT NULL
        cApellidos              VARCHAR(60)             NOT NULL
        cDirección              VARCHAR(100)
        cTeléfono               CHAR(9)                 NOT NULL
        dNacimiento             DATE                    NOT NULL
Clave primaria: cDNI
Claves ajenas:
        N/A
Índices:
        iDatos_PK               cDNI ASC
        iDatos_Apellidos_Nombre cApellidos ASC + cNombre ASC
        iDatos_Nacimiento       dNacimiento DESC
_____________________________________________________________________

Tabla: TPrueba
Campos:
        cFirma                  VARCHAR(15)             NOT NULL
        cDNI                    CHAR(9)                 NOT NULL
        dFecha                  DATE                    NOT NULL
Clave primaria: cFirma ASC + cDNI ASC + dFecha DESC
Claves ajenas:
        cFirma          ————————————→   TEjemplo.cSignatura
        cDNI            ————————————→   TDatos.cNIF
Índices:
        iPrueba_PK              cFirma ASC + cDNI ASC + dFecha DESC
        iPrueba_FK_DNI          cDNI ASC
        iPrueba_FK_Firma        cFirma ASC
_____________________________________________________________________

...
```

Imagen 2. Ejemplo de un fragmento de un diccionario de datos

2.1.3. Modelo conceptual de datos (MCD)

En este modelo se representa una visión más estática y estructurada de los datos, identificando la estructura interna de datos y las relaciones que existen entre las propias entidades.

Características del MCD:

- Tiene que contener toda la información que se maneja en el sistema.
- El estado final de los datos debe de estar representado.
- Todos los cambios que se realicen en el sistema de información deben de ir reflejados en el modelo de datos, y a su vez, al revés también.

Toda esta modelización conceptual de datos se lleva a cabo en la gran mayoría de las ocasiones mediante un diagrama entidad/relación, que veremos más adelante.

2.1.3. Arquitectura ANSI/SPARC

El grupo ANSI en el año 1975 propuso la estandarización de la arquitectura o modelo X3/SPARC o ANSI-SPARC. Un estándar de diseño abstracto para los sistemas de gestión de bases de datos para dar independencia de los usuarios respecto a los datos. Aún que en la gran mayoría de base de datos utilizan esta arquitectura, nunca se convirtió en un estándar formal.

La arquitectura define tres nivel de abstracción:

- **Externo:** punto de vista de cada usuario. Consta de las distintas visiones que tienen los usuarios de la base de datos.
- **Conceptual:** punto de vista de todos los usuarios. El esquema es la visión común de la base de datos: especifica el contenido de información de la base de datos independientemente de las consideraciones de almacenamiento.
- **Interno:** punto de vista de la máquina. Es la visión que el ordenador tiene de la base de datos: especifica cómo se representan los datos, en qué orden se almacenan los registros, qué índices y punteros se han creado y qué esquema de dispersión se ha utilizado, si es el caso.

Todo sistema de gestión de bases de datos que separe los tres niveles deberá tener correspondencias entre los esquemas para transformar las peticiones de los usuarios y los resultados, de un nivel al siguiente. La mayoría de los sistemas de gestión de bases de datos no separan los tres niveles por completo. La independencia de datos hace que cada nivel de la arquitectura sea inmune a los cambios en los niveles de debajo.

La independencia de datos lógica se refiere a la inmunidad de los esquemas externos frente a los cambios en el esquema conceptual. La independencia de datos física se refiere a la inmunidad del esquema conceptual frente a los cambios en el esquema interno.

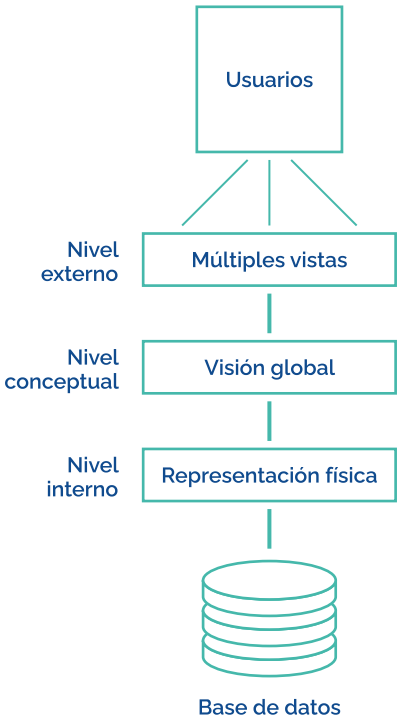


Imagen 3. Niveles de la arquitectura ANSI-SPARC

Externo

```
DCL  1 EMPP,
     2 EMP# CHAR(6),
     2 SAL FIXED BIN(31);
```

Externo

```
     01 EMPC.
        02 EMPNO PIC X(6).
        02 DEPTNO PIC X(4).
```

Conceptual

```
        EMPLEADO
            NUMERO_EMPLEADO             CHARACTER (6)
            NUMERO_DEPARTAMENTO         CHARACTER (4)
            SALARIO                     NUMERIC (5)
```

Interno

```
        EMP_ALMACENADO      BYTES=20
        PREFIJO             TYPE=BYTES(6), OFFSET=0
        EMP#                TYPE=BYTES(6), OFFSET=6, INDEX=EMPX
        DEPT#               TYPE=BYTES(4), OFFSET=12
        SUELDO              TYPE=FULLWORD, OFFSET=16
```

Imagen 4. Representación de los datos en la arquitectura ANSI/SPARC

Los niveles de datos externo y conceptual se conocen como modelos lógicos, mientras que el nivel interno se conoce como físico.

A su vez, los modelos lógicos (externo y conceptual) pueden dividirse en:

- **Conceptuales:** describen el mundo real sin tener en cuenta la tecnología: redes, sistemas operativos, etc. Ejemplos: modelo entidad-relación o UML para objetos.
- **Convencionales:** orientados a la implementación del mundo real en un SGBD. Ejemplo, jerárquico, red, relacional.

De manera concreta, las principales características de uno y otro se resumen en la siguiente tabla:

Diferencias conceptual VS convencional	
Conceptual	**Convencional**
No tiene por qué implementarse en SGBD	Están implementados, en SGBD
Es independiente del SGBD	Es dependiente del SGBD
Tiene un nivel de abstracción	Es más cercano a la máquina
Mayor capacidad semántica	Menor capacidad semántica
Enfocado al diseño de alto nivel	Enfocado a la implementación
Interfaz usuario/informático	Interfaz informático/sistema

2.2. Diseño de base de datos

Podemos definir el diseño de una base de datos como el proceso en el que se define la estructura de los datos que debe tener la base de datos de un sistema de información determinado.

El proceso de diseño de una base de datos generalmente implica varias etapas:

1. **Análisis de requisitos** para determinar el propósito y el alcance de la base de datos. Esta primera etapa se determina el problema y los requisitos que debe disponer.
2. **Diseño conceptual** para crear una representación de alto nivel de los datos y las relaciones entre los elementos de datos sin tener en cuenta el tipo de base de datos. El diagrama entidad-relación (E/R) es la mejor forma de representar el diseño conceptual con los datos obtenidos en la etapa de requisitos.
3. **Diseño lógico** para transformar el modelo conceptual en una representación más detallada utilizando un sistema de gestión de base de datos específico (relacional, orientado a objetos, documental, etc.).
4. **Diseño físico** para optimizar la base de datos en términos de rendimiento, seguridad y capacidad de almacenamiento.
5. **Implementación** y población de la base de datos con datos.
6. **Pruebas** y validación de la base de datos para garantizar que cumpla con los requisitos y funcione como se espera.
7. **Mantenimiento** y actualización de la base de datos con el tiempo para reflejar los cambios en las necesidades de datos de la organización.

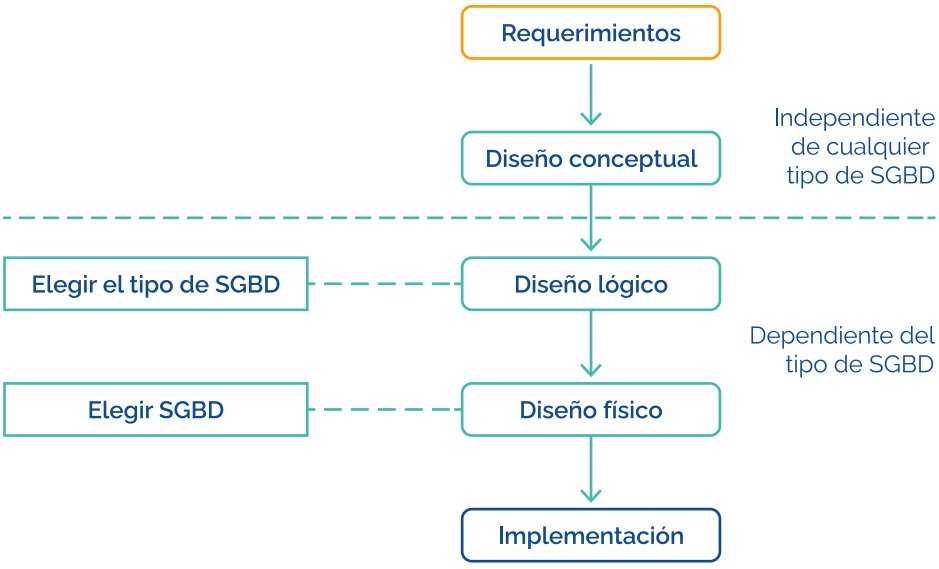


Imagen 5. Etapas del diseño de base de datos

2.3. Diagrama entidad/relación (E/R)

En la etapa de diseño conceptual se realiza la representación de alto nivel del problema general que se plantea y se analiza en la etapa de requerimientos. La mejor formar de realizar esta representación es mediante el diagrama entidad/relación.

Según Métrica-3 un diagrama E/R se define como:

> "Una técnica cuyo objetivo es la representación y definición de todos los datos que se introducen, almacenan, transforman y producen dentro de un sistema de información, sin tener en cuenta las necesidades de la tecnología existente, ni otras restricciones."

Los elementos básicos que intervienen en un diagrama E/R son las entidades, atributos, relaciones y conexiones.

Elementos básicos de un diagrama E/R	
Descripción	**Símbolo**
Entidad: se representa mediante un rectángulo.	
Atributo: se representa mediante eclipse.	
Relación: se presentan mediante un rombo.	
Conexión: se representan mediate una línea entre atributos y relaciones.	

Es necesario que se recalque la importancia de la independencia respecto al resultado e implementación que se nos dé al final. Este diagrama nos servirá para obtener la solución al problema que se plantea de manera indiferente al SGBD que se vaya a usar, con lo cual, el diagrama no tiene que reflejar ninguna característica del tipo de SGBD.

2.3.1. Entidad

Una **entidad** es un concepto abstracto que hace referencia a un objeto real o abstracto. Su existencia depende solo de sí misma y es necesario que su identificación sea clara y precisa. Se trata del elemento principal del modelo entidad/relación y se nombran con sustantivos en singular que encierra un concepto que el analista debe de identificar. Un ejemplo puede ser: "Trabajador".

Las entidades contienen elementos concretos referentes a sí mismas, las ***ocurrencias*** o ***instancias***. Por ejemplo, cada trabajador en específico es una ocurrencia del ejemplo de entidad anterior "Trabajador".

Pero dentro de cada ocurrencia, para poder distinguirlas, habrá que tener otros datos asociados (nombre, dirección, DNI, nº de teléfono, etc...). Estos datos son los que llamamos **atributos**, y tienen un valor asociado ("Pepe", "C/Frasquito N/16", "3333333354ª", "45555556", etc....).

Cualquier entidad debe de cumplir estas características:

- Todas las ocurrencias deben de tener el mismo número de atributos, sin importar que carezcan de algún valor. Más adelante veremos como trata esta situación.
- No puede haber dos ocurrencias que tengan los mismos valores para todos sus atributos, porque sería una duplicidad.
- Los atributos que sean opcionales irán con la siguiente especificación: *[0..1]*

Cuando queremos representar gráficamente una entidad, se hace del siguiente modo: se dibuja un rectángulo con el nombre la entidad en su interior (en mayúsculas siempre que sea posible) y entonces tenemos dos modos de representar los atributos de dicha entidad, mediante óvalos o mediante círculos pequeños. Si realizamos los atributos como óvalos, el nombre de cada atributo va dentro, mientras que, si son círculos, van al lado.

Otra posibilidad de representación gráfica es utilizar el mismo rectángulo de la entidad para definir los atributos, el rectángulo se dividirá en dos partes, la primera parte para indicar el nombre de la entidad y la segunda parte para indicar los atributos que contiene. Opcionalmente se podrá indicar el tipo de dato (Número, cadena, fecha, booleano) separado por : para diferenciar el nombre del campo y el tipo de dato.

Un ejemplo sería:

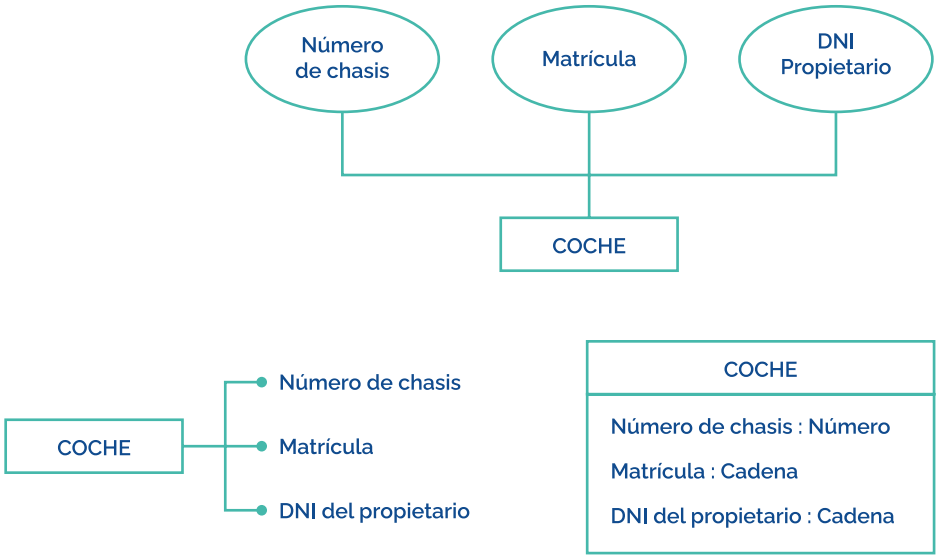


Imagen 6. Representaciones gráficas de entidades y atributos.

Podemos diferenciar dos tipos de entidades:

- **Fuerte o regular.** Es totalmente independiente de las otras entidades. Son la mayoría de las entidades.
- **Débil.** Sus ocurrencias dependen de las existencias de las ocurrencias de otra entidad distinta.

Una forma de distinguir entre un tipo de entidad fuerte y una débil es hacernos la siguiente pregunta. ¿Puede existir la entidad sin que exista previamente otra entidad? En el caso de tener que eliminar una entidad fuerte, se tendría que eliminar la entidad débil dependiente. Por ejemplo, Libro y Ejemplar. Libro es una entidad fuerte, puede existir sin tener un ejemplar, en cambio, Ejemplar es una entidad débil, si no hay un libro no puede haber un ejemplar.

Las entidades débiles deben de serlo con respecto a todas las entidades con las que se relacionan, y su representación es la siguiente:

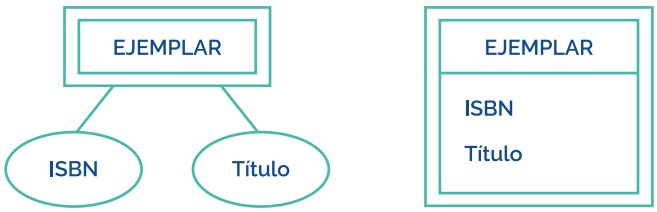


Imagen 7. Representación gráfica de una entidad débil

2.3.2. Relación

Las ocurrencias de varias entidades distintas pueden tener aspectos en común, lo que da lugar a una relación entre ellas. Las relaciones se nombran usando la tercera persona de singular de indicativo de un verbo y un ejemplo podría ser entre "camionero" y "camión" con la relación "conduce".

La representación gráfica de una relación se realiza dibujando un rombo con el nombre de la relación dentro, como el ejemplo siguiente:

Imagen 8. Representación gráfica de una relación

Dependiendo del número de entidades en las que sus concurrencias tienen alguna relación, tenemos varios tipos de relaciones:

- **Binarias.** Relacionan entre sí ocurrencias de solamente dos entidades distintas.

  Por ejemplo, podríamos hablar de un alumno que cursa alguna materia:

Imagen 9. Relación binaria

- **Ternarias.** Se relacionan las ocurrencias de tres entidades distintas entre sí.

  Por ejemplo, siguiendo con el ejemplo anterior, un alumno cursa una materia que pertenecerá a un ciclo específico:

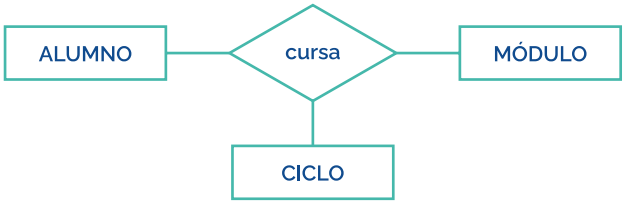


Imagen 10. Relación ternaria

- **N-arias.** Se puede dar el caso dependiendo de la complejidad de nuestra base de datos de que tengamos que establecer una relación de más de tres entidades al mismo tiempo. Podemos aplicar como ejemplo el mismo anterior, en el que alumno cursa el módulo de un ciclo en específico y en un centro en particular:

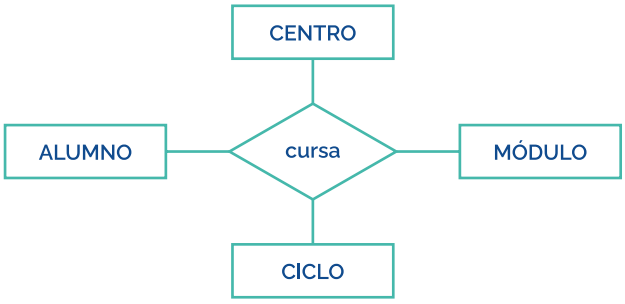


Imagen 11. Relación cuaternaria

- **Reflexivas.** Se relacionan entre sí ocurrencias que tienen cabida en la misma entidad. Esto sucede porque, aunque estén dentro de una misma entidad, la realidad es que estas ocurrencias ocupan funciones distintas. Tenemos el ejemplo en el que hay varios alumnos en una clase y uno debe ser el delegado de todos los demás. No hay necesidad de crear otra entidad porque lo único que cambia es si es delegado o no, pero todo lo demás se conserva igual y por lo tanto se establece una relación reflexiva:

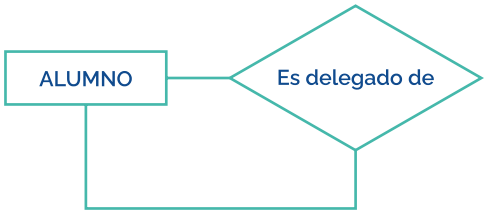


Imagen 12. Relación reflexiva

NOTA

Un pequeño consejo para el diseño de las bases de datos es que identificaremos relaciones ternarias o n-arias cuando la acción que se realice afecte de manera simultánea a todas las ocurrencias de las distintas entidades implicadas.

2.3.3. Cardinalidad y modalidad

Si lo que queremos es dotar de un cierto contenido lógico y estructurado a nuestra relación, tenemos que especificar como se relacionan las ocurrencias de las entidades en cuestión. Esto se realiza gracias a que hay una serie de ámbitos, límites y restricciones que están preestablecidos.

La **cardinalidad o tipo de correspondencia** nos va a indicar el número de máximo de datos de una ocurrencia en relación con la otra ocurrencia con la que se relacione. La cardinalidad maneja varias situaciones para poder abarcar el máximo posible:

- **Uno a mucho (1:N).** En este caso la primera ocurrencia en la relación, la del lado donde apunta el 1, se puede relacionar con muchos del extremo, mientras que la ocurrencia del lado N, solo puede relacionarse con una como máximo del otro extremo.

Vamos a imaginar que tenemos una asignatura que puede ser impartía por varios profesores, pero un profesor solo puede impartir de manera normal una asignatura, pues su representación gráfica sería la siguiente:

Imagen 13. Relación 1: N

- **Muchos a muchos (M:N).** En ambos casos, ambas ocurrencias pueden relacionarse al mismo tiempo con varias ocurrencias del extremo opuesto.

  Tenemos la casuística de que un alumno puede tener varios módulos que cursar, pero un mismo módulo puede ser cursado por varios alumnos, se representaría gráficamente del siguiente modo:

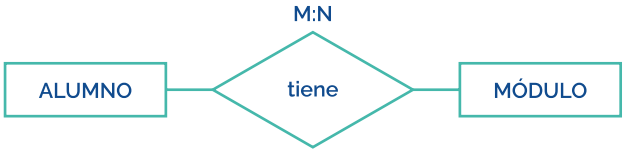


Imagen 14. Relación M:N

- **Uno a uno (1:1).** Una ocurrencia de un extremo solo se puede relacionar con otra del extremo opuesto. Lo mismo en ambas direcciones. En este caso sabemos que un director puede dirigir un solo departamento, y un departamento solo puede tener un director, como el siguiente ejemplo:

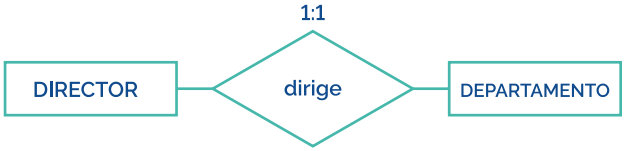


Imagen 15. Relación 1:1

La cardinalidad nos ayuda a establecer los límites superiores en una relación, pero realmente no existe una obligatoriedad de que sea justo esa la cantidad necesaria.

Para solucionar este problema tenemos la ***modalidad***. Esta característica nos va a permitir establecer dentro de la cardinalidad unos límites de la propia ocurrencia, es decir, unos máximos y unos mínimos. Esta modalidad la indicamos a ambos lados de la relación, es decir, en cada ocurrencia y el valor máximo siempre va a coincidir con el valor que se haya asignado mediante la cardinalidad de la relación. Hay varios tipos, son los siguientes:

- **Cero a uno (0,1)**. La ocurrencia de la entidad contraria se puede relacionar con ninguna ocurrencia de esta entidad y como máximo con una, nunca con varias.
- **Uno a uno (1,1)**. Sí o sí, la ocurrencia contraria en la relación debe de relacionarse con una ocurrencia de nuestra entidad. Solo con una.
- **Uno a mucho (1,N)**. La ocurrencia contraria se va a relacionar con al menos una ocurrencia de nuestra entidad y puede que con varias.
- **Cero a muchos (0,N)**. En la entidad contraria, la ocurrencia se relacionará con nuestra entidad pudiendo relacionarse con ninguna o con varias ocurrencias a la vez, es indiferente.

Un ejemplo de relación binaria 1:N donde tenemos en un lado (0,1) y en otro (0,N) correspondiente a una tienda de ordenadores:

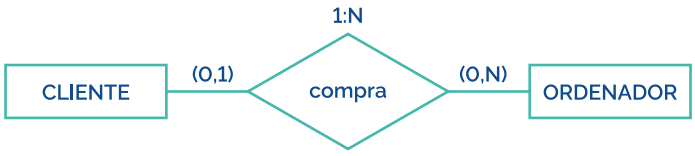


Imagen 16. Relación (0,1) a (0,N)

En este ejemplo, su modalidad se lee en el lado opuesto, es decir, se explica del siguiente modo:

- Un cliente puede comprarse ninguno o varios ordenadores, pero no tiene por qué comprar ninguno y puede que se compre más de uno.
- Un ordenador (uno en específico, no un modelo), puede ser comprado por un cliente únicamente, pero también puede ser que no sea comprado nunca, por eso lo de este tipo de modalidad.

Este otro ejemplo refleja una situación distinta:

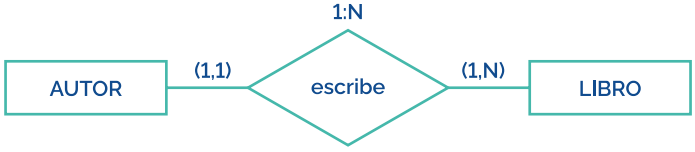


Imagen 17. Relación (1,1) a (1,N)

En el ejemplo anterior, tenemos lo siguiente:

- Un autor puede escribir uno o varios libros (mínimo uno, si no, no sería autor).
- Mientras, un libro únicamente puede ser escrito por un autor, y siempre será mínimo uno y máximo uno porque si no se escribe, no existe.

Un ejemplo donde tenemos una relación M:N y en un extremo su modalidad que es (0,N) y (1,N). El siguiente es cuando un alumno cursa un módulo. Este alumno puede cursar un módulo mínimo (si no, no sería alumno), pero también puede cursar varios al mismo tiempo.

También tenemos que saber que un módulo puede ser cursado por varios alumnos, pero también puede ser que no lo curse ninguno porque nadie se haya matriculado de ese módulo, por mucho que exista:

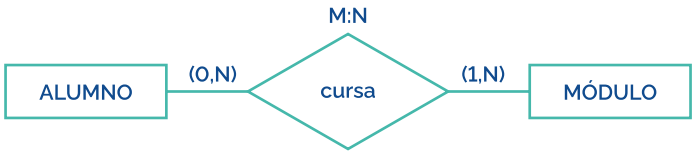


Imagen 18. Relación (0,N) a (1,N)

Un último ejemplo en una relación que acapara cuatro entidades y en las que la empezando de arriba abajo tienen la siguiente modalidad: (1,N) por que un alumno puede cursar el ciclo en uno o varios centros (varios ciclos a la vez), (1,N) como hemos dicho antes se pueden cursar uno o varios módulos, (1,N) se puede cursar más de un ciclo al mismo tiempo pero mínimo uno y (0,N) porque como antes, puede que no haya alumnos y puede que haya varios que cursen distintos módulos y ciclos.

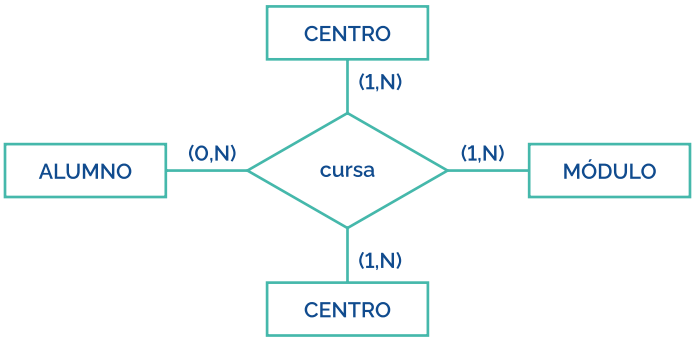


Imagen 19. Relación cuaternaria

2.3.4. Atributos de relación

Hay ocasiones en las que ciertas propiedades o alguna característica afecta al mismo tiempo a dos entidades y a su vez tiene que ver con la relación que hay formada entre dichas entidades. Es en este momento cuando se crea un atributo de relación y afecta directamente a esta, lo que hace que afecte al mismo tiempo a las dos o más entidades que toman parte de la relación. Se representa como un atributo normal pero no en cada entidad, sino en la relación. En este ejemplo tenemos que para identificar a un empleado tendremos en cuenta su código de empleado, su NIF, su nombre y su salario. En cuanto a curso, se almacena el código del curso, el nombre y el número de horas que dura.

Ahora bien, tenemos que almacenar también cuando empezó dicho curso, que es tanto cuando lo empezó el empleado y cuando comienza como tal. Es por eso por lo que este atributo se almacena en la relación y no en cada entidad, porque es común.

Gráficamente se representa igual que en una entidad, pero pertenece a la relación:

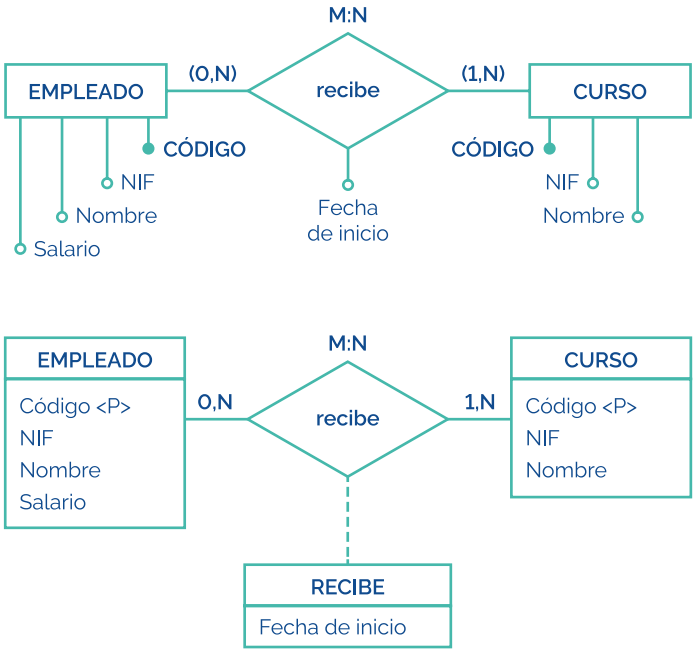


Imagen 20. Atributo de relación

2.3.5. Clave primaria y claves candidatas

Siempre que tengamos una entidad, habrá un atributo o un conjunto de atributos que con su valor nos ayudarán a identificar la entidad de todas las demás.

En cada entidad habrá un atributo que sea único para cada dato que se introduzca y que nos ayude a diferenciarlo de todos los demás, por ejemplo, el DNI de cada persona es único para ella y no puede haber dos iguales. Esto es lo que llamamos la **clave primaria**. Nos ayuda a identificar un dato de la entidad como antes hemos dicho y es única, no puede haber dos iguales.

En el siguiente ejemplo tenemos un socio de alguna empresa u organización que tiene los siguientes atributos: código de socio, nombre, tipo, domicilio, teléfono y apellido.

Bien, si lo que queremos es distinguirlo de los demás, veremos que solo hay una cosa que será única para él y no para ningún otro socio, que es el código de socio.

Si representamos mediante círculos los atributos, la clave primaria se representa como un círculo relleno, mientras que, si lo representamos mediante óvalos, esta clave es la palabra que esté subrayada. Si utilizamos un mismo rectángulo con sus atributos, diferenciaremos la clave primaria con <P> o <PK> y en negrita.

Aquí vemos una descripción gráfica:

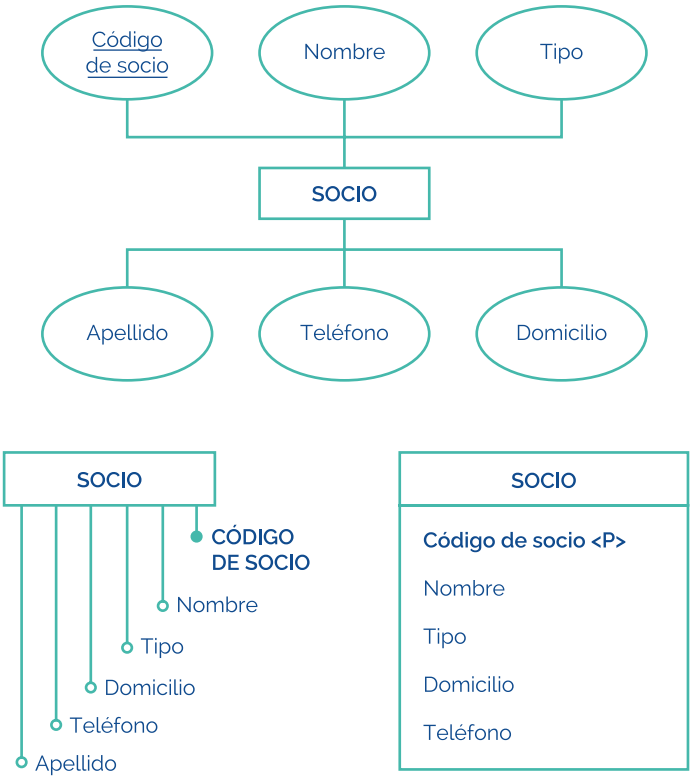


Imagen 21. Representaciones gráficas de clave primaria

No siempre es posible identificar los datos por un solo atributo porque este se puede repetir. Pero puede que no haya ninguna que excluya a las demás. En este caso de manera casi general, es una combinación de varios atributos los que hacen que haya una identificación única. Ahora tendríamos una clave primaria formada por varios atributos. El ejemplo de a continuación nos muestra la entidad 'empleado' en la que tenemos como clave primaria tanto código de usuario como código de empleado. Esto se debe a que en solitario no son únicas, pero su combinación sí, porque una va ligada a la otra.

Imagen 22. Clave primaria compuesta

A parte de la clave primaria que existe en una entidad, puede surgir un atributo de la misma entidad que cumpla condiciones similares a ser una posible clave, este tipo se denomina **clave candidata**. Por ejemplo, una entidad CLIENTE puede tener como clave primaria el atributo Código cliente, además tiene los atributos NIF, Nombre y Apellido. Si observamos el atributo NIF que no es clave primaria, si posee la características de poder identificar de forma única a un cliente, con lo cual, NIF será una clave candidata.

Para representar una clave única, se subrayará en discontinuo el nombre del atributo. Si utilizamos un mismo rectángulo con sus atributos, diferenciaremos la clave candidata con <U> y sin negrita.

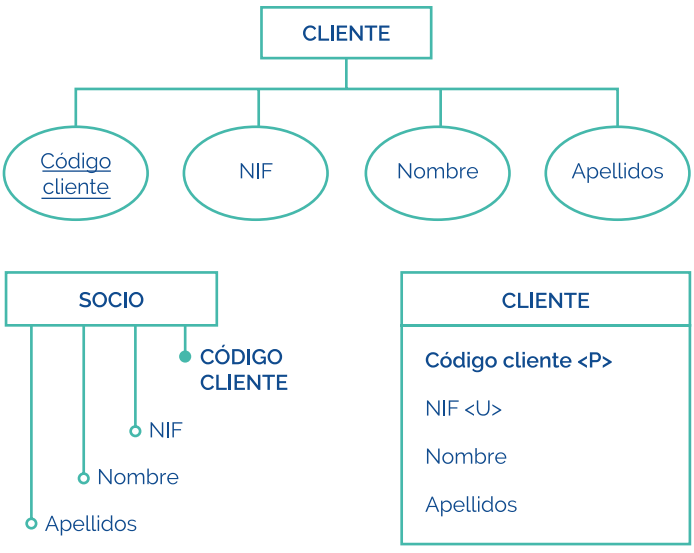


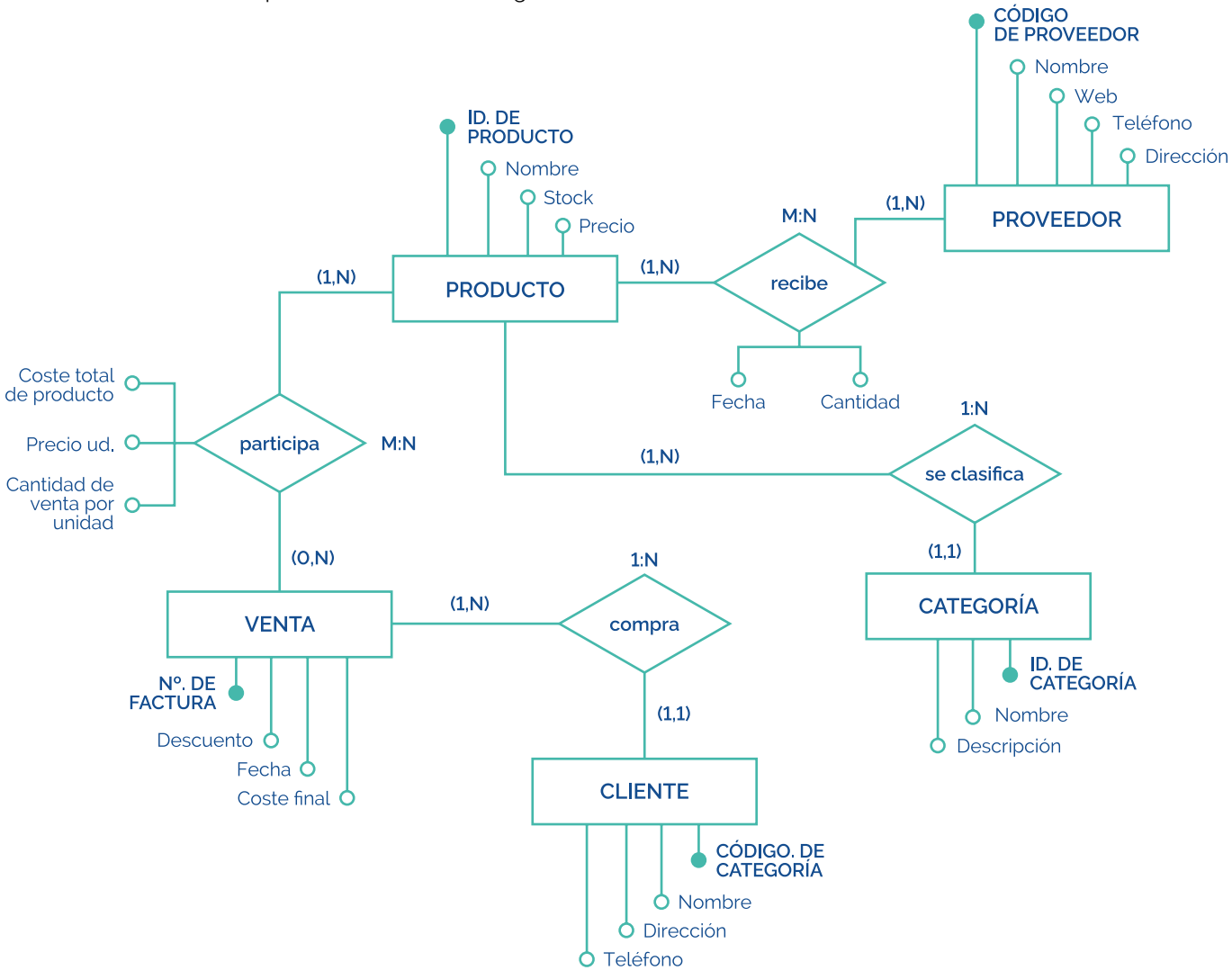
Imagen 23. Clave candidata

2.3.6. Ejemplo de diagrama entidad/relación

Tenemos un sistema de venta, donde necesitamos que se tengan en cuenta los siguientes requisitos:

- Tenemos unos proveedores que nos ofrecen distintos productos. De estos proveedores tenemos que almacenar su código de proveedor, nombre, web, teléfono y dirección.
- De cada producto almacenamos su id de producto, su nombre, el stock que hay y su precio.
- Los productos pertenecen a una cierta categoría. De la que almacenamos el id de la categoría, su nombre y una breve descripción.
- Además, los productos participan en una venta, y tenemos que almacenar coste total del producto venido, el precio por unidad y la cantidad de venta de cada producto.
- De la venta almacenaremos el número de factura, el descuento que se realice, la fecha y el coste final.
- Estas ventas son porque ha comprado un cliente, del que almacenaremos el código del cliente, el nombre, su dirección y su teléfono.

Esto quedaría finalmente del siguiente modo:



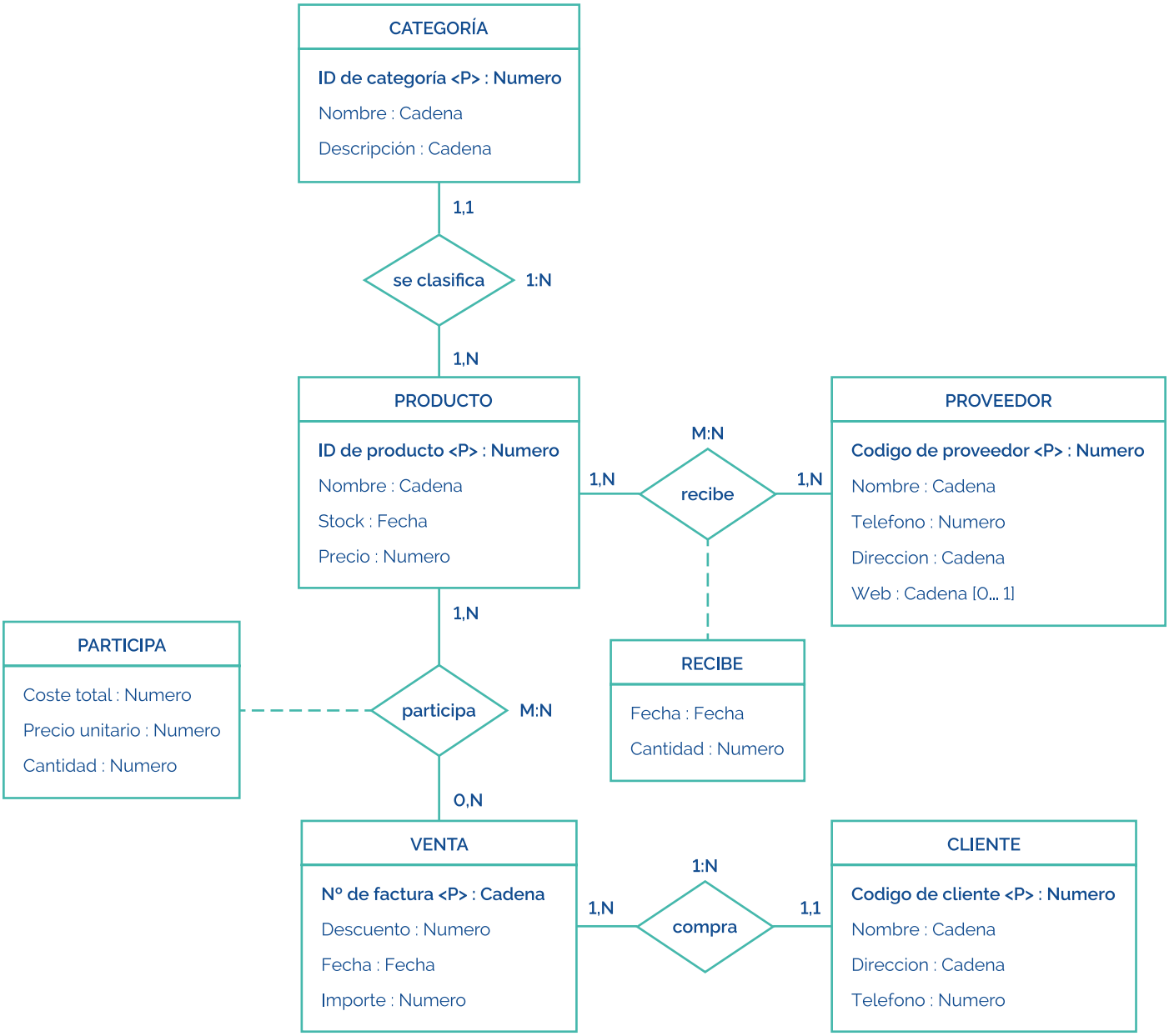


Imagen 24. Ejemplo de diagrama entidad/relación en las dos formas de representación gráfica

Vamos a explicar lo que hemos realizado:

- Se han creado las entidades 'PROVEEDOR', 'PRODUCTO', 'CATEGORIA', 'VENTA' y 'CLIENTE'.
- Se añaden las relaciones entre ellos, como podemos ver hay entre proveedor y producto, entre producto y categoría, entre producto y venta y entre venta y cliente.
- Se añaden los atributos de las entidades como hemos dicho que queremos almacenar.
- Se seleccionan las claves primarias de dichas entidades.
- Añadimos, porque en este caso los hay, los atributos de relación.
- Se determinan los tipos de datos genéricos.
- Se establece que atributos son opcionales.

Si explicamos en más profundidad, tenemos lo siguiente:

- 'PROVEEDOR' tiene como clave primaria código de proveedor, porque es lo que lo identifica.
- 'PRODUCTO' tiene como clave primaria id de producto.
- Se establece entre los dos la relación 'tiene' que es muchos a muchos. Un producto puede ser traído por uno o varios proveedores, pero siempre por uno mínimo. A la misma vez, un proveedor nos puede traer uno o más productos de igual modo.
- Ahora añadimos la entidad 'CATEGORIA' donde su clave primaria será id de categoría.
- Entre 'PRODUCTO' y 'CATEGORIA' se establece la relación 'se clasifica' que es muchos a uno. Una categoría puede englobar uno o varios productos, pero mínimo uno, de manera lógica. De modo similar, un producto solo puede pertenecer a una categoría, sí o sí a una.
- Siguiendo con las relaciones de producto, se crea la entidad 'VENTA', que tiene como clave primaria número de factura.
- Entre 'PRODUCTO' y 'VENTA' creamos la relación 'participa'. En esta relación guardaremos los atributos de relación: coste total de producto, precio unidad y cantidad de venta por unidad.
- La relación es muchos a muchos ya que un producto puede venderse en ninguna o varias ventas, y una venta mínima venderá un producto, pero puede vender varios.
- Por último, creamos la entidad 'CLIENTE' cuya clave primaria será código de cliente.
- Entre 'CLIENTE' y 'VENTA' se establecerá la relación 'compra', que es muchos a uno porque una venta será por un único cliente, pero un mismo cliente puede realizar varias ventas, mínimo una.

2.4. Diagrama entidad/relación extendido

Del modo que se planteó en un primer momento el diagrama entidad/relación había una serie de cuestiones a las que no daba solución con respecto a los problemas que el modelo de conceptual de datos plantea. Esto hace que algunos expertos añadiesen aún más restricciones para que su magnitud semántica se vea encarecida en ciertos casos. Realmente esto no es un hecho que se lleve a cabo de manera unánime en el diseño de las bases de datos, pero tenemos algunos elementos muy reconocibles del diagrama de entidad/relación extendido.

2.4.1. Restricciones sobre las relaciones

Sobre las relaciones que hay con las entidades pueden surgir diferentes casos que implica aplicar un nivel de restricción. Nos podemos encontrar con:

- **Exclusividad**. Si existe una relación, no puede existir otra. Un ejemplo sería el siguiente, en el que tenemos que un vehículo puede o consumir litros de gasoil o gastar litros de gasolina, pero nunca las dos al mismo tiempo:

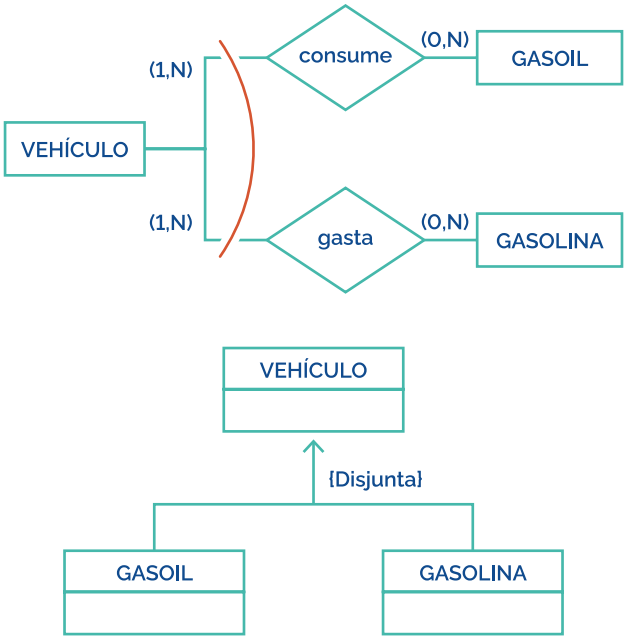


Imagen 25. Exclusividad

Las restricciones de exclusividad pueden venir dadas por dos opciones:

- **Disjunta:** una entidad solo puede pertenecer a una única relación con una entidad.
- **Solapada:** lo contrario a disjunta. Una entidad puede pertenecer a más de una relación al mismo tiempo.

- **Exclusión.** Si existe más de una relación para dos entidades, ambas ocurrencias de cada entidad solo pueden usar una única relación. Esta restricción se realiza con una línea discontinua entre ambas relaciones.

  Un ejemplo sería el siguiente, en el que tenemos que un profesor que imparte un curso o recibe un curso, pero nunca las dos al mismo tiempo:

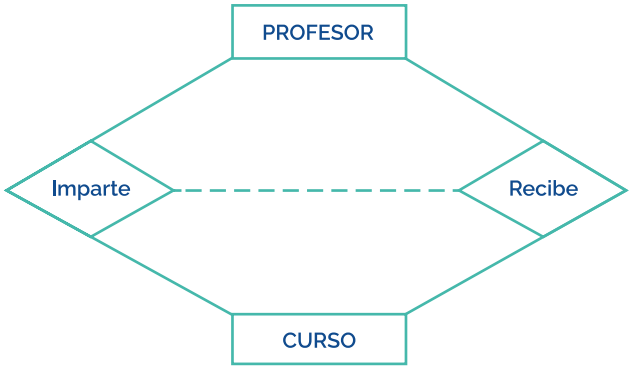


Imagen 26. Exclusión

- **Inclusión.** Se utiliza para asociar dos ocurrencias entre sí a través de una relación, que previamente puede haber estado asociado a través de otra relación distinta. Se realiza mediante una línea discontinua entre las relaciones y con acabado con una flecha hacia la relación que puede ocurrir previamente.

  Un ejemplo, que un profesor antes de impartir un curso puede haber recibido el curso.

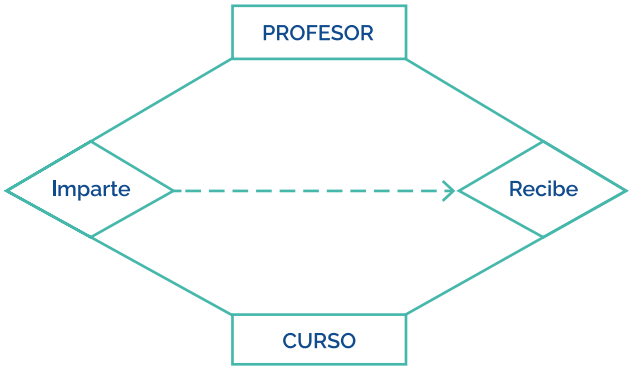


Imagen 27. Inclusión

- **Inclusividad.** Se utiliza para asociar dos ocurrencias entre sí a través de una relación, que previamente se ha asociado a través de otra relación distinta, este hecho tiene que suceder obligatoriamente. Se realiza mediante una flecha envolvente y apuntando hacia la relación previa.

  Un ejemplo, que un profesor antes de impartir un curso tiene que recibir el curso obligatoriamente.

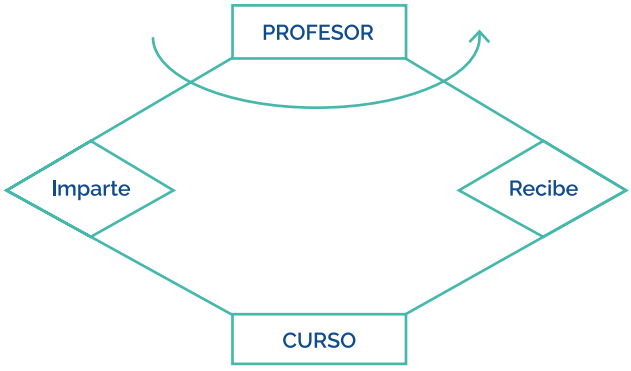


Imagen 28. Inclusividad

2.4.2. Jerarquías y generalizaciones

Existe una primera entidad con una ocurrencia y se relaciona con varias ocurrencias de distintas entidades al mismo tiempo. Esta entidad se llama supertipo y tienen atributos compartidos con las demás entidades o subtipos, mientras que las entidades de menor jerarquía cuentan con estos atributos del supertipo y con otros propios.

Por ejemplo, tenemos una empresa con empleados, los cuales pueden ser o asalariados fijos o estar contratados por horas. Estos empleados tienen algunas cosas en común, pero otras que no, y en este caso se usaría una jerarquía como la siguiente mostrada:

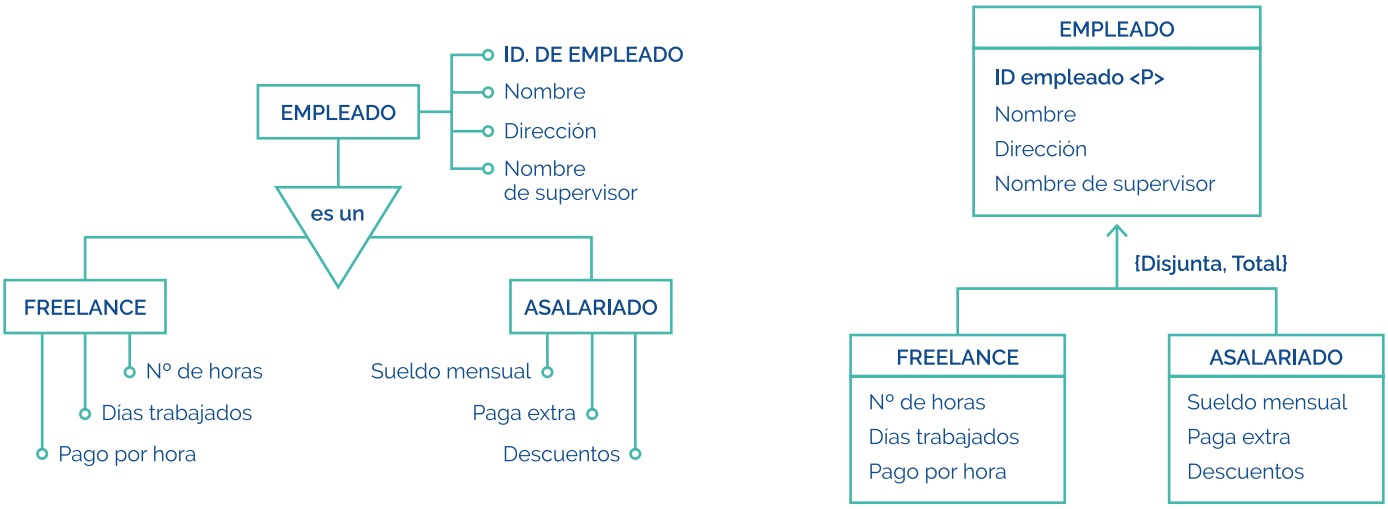


Imagen 29. Jerarquía

Las restricciones de jerarquía pueden venir dadas por dos opciones:

- **Total.** La superentidad debe pertenecer a alguna subentidad.
- **Parcial.** La superentidad puede no pertenecer a ninguna subentidad.

Casi siempre que hablamos de jerarquía partiremos de la entidad supertipo y por lo general las entidades subtipo nos ayudarán a crear una ***especialización*** que ayudan a dar un sentido más específico a la información que se agregue. Por otra parte, si el inicio de la jerarquía reside en los subtipos se hablará de una ***generalización*** que nos ayudará a crear un conjunto de entidades para su mejor entendimiento.

2.4.3. Agregación

En este caso varias ocurrencias de entidades distintas en conjunto crean una ocurrencia para otra entidad distinta. Un ejemplo sería este en el que agruparíamos las entidades empresa y solicitante en otra que se puede llamar entrevista:

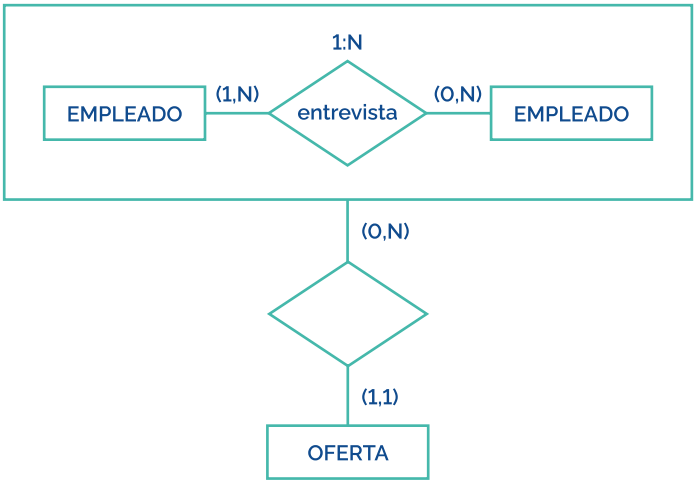


Imagen 30. Agregación

2.5. Caso práctico

Durante la etapa de análisis de requisitos se plantea el siguiente problema:

La fundación Vacunak nos ha encargado la creación de una base de datos para el registro de información sobre las vacunas que es necesario inocular a los animales de los parques naturales que gestiona.

Cada vacuna se identifica mediante un código alfanumérico, tiene un nombre único, una unidad de medida y se puede almacenar, opcionalmente, una descripción de esta. Cada vacuna requiere su aplicación en una o varias dosis. Cada una de estas dosis se identifica mediante un número que, no obstante, se puede repetir para diferentes vacunas. Además, se puede almacenar textualmente la utilidad de cada una de las dosis.

Es necesario registrar en la base de datos las dosis de vacunas que es necesario aplicar a cada especie y a qué edad es necesaria la inoculación. Cada especie se identifica mediante su nombre científico y se desea almacenar, además, su nombre popular, que es único, y el género (que se anotará textualmente) al que pertenece. Debemos almacenar en la base de datos, también, las edades a las que es necesario aplicar las dosis de vacunas. Cada edad se identifica mediante un número entero y una unidad de tiempo (día, semana, mes o año), por ejemplo, catorce meses, dos años, etc.

Pues bien, para cada especie, en una determinada edad, puede ser necesario aplicar varias dosis de vacunas y por cada una de estas dosis se almacenará un número real que indicará el tamaño de la dosis en la unidad de medida especificada para la vacuna. Puede ser necesario suministrar la misma dosis de vacuna a la misma especie en diferentes momentos de su vida. Una dosis de vacuna puede ser necesario inocularla a varias especies a la misma edad.

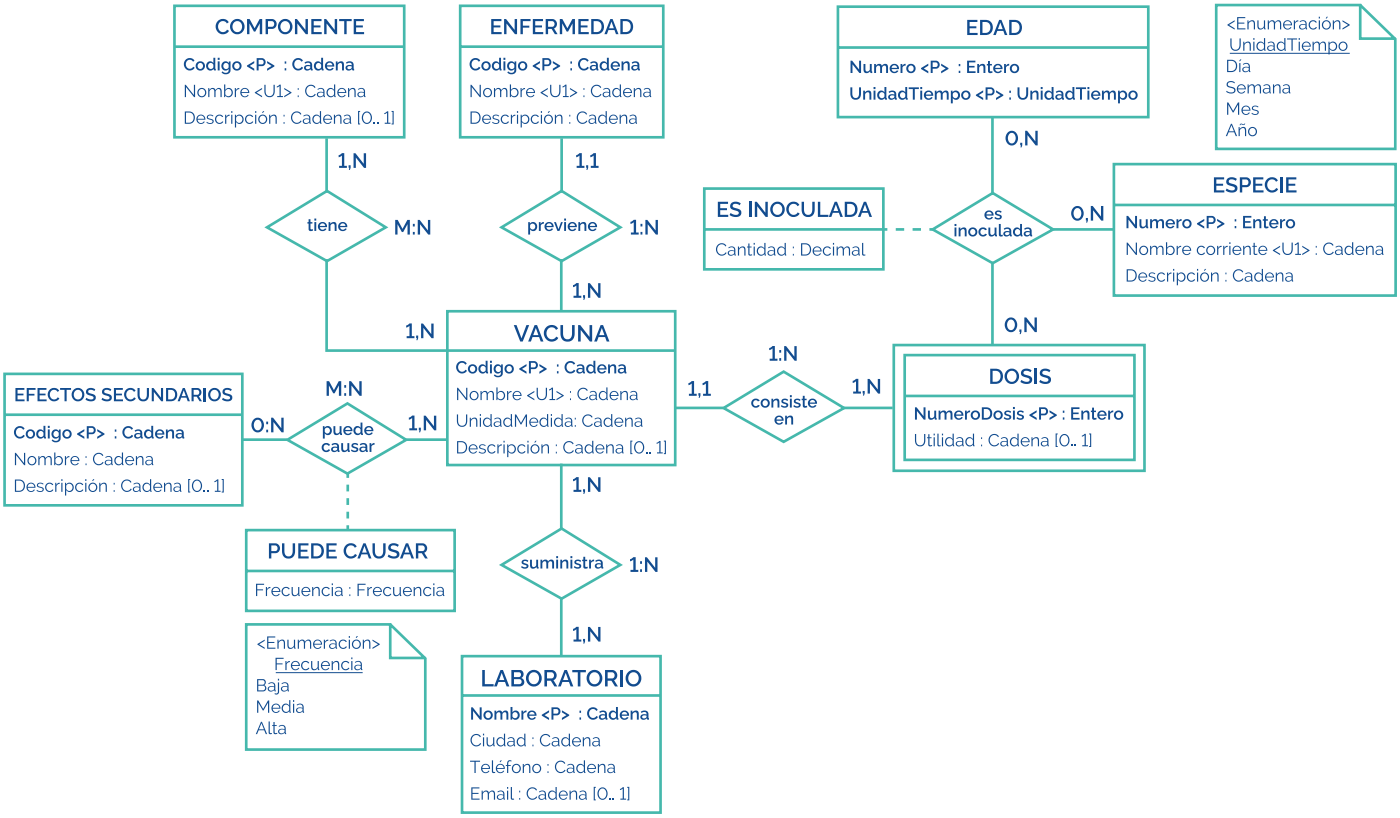
Cada vacuna se usa para prevenir una determinada enfermedad, aunque puede ocurrir que para una enfermedad sea posible emplear más de una vacuna. Solo registraremos en la base datos aquellas enfermedades que se puedan prevenir mediante la inoculación de, al menos, una vacuna. Cada enfermedad se identifica mediante un código alfanumérico y tiene asignado un nombre único y una descripción.

Algunas vacunas pueden originar efectos secundarios. Cada efecto secundario tiene asignado un código alfanumérico único, un nombre y una descripción opcional. Puede que una vacuna tenga más de un efecto secundario y un mismo efecto secundario puede ser producido por una o varias vacunas. Queremos saber la frecuencia con la que puede provocar una vacuna cada uno de sus efectos secundarios y registraremos esta frecuencia como baja, media o alta.

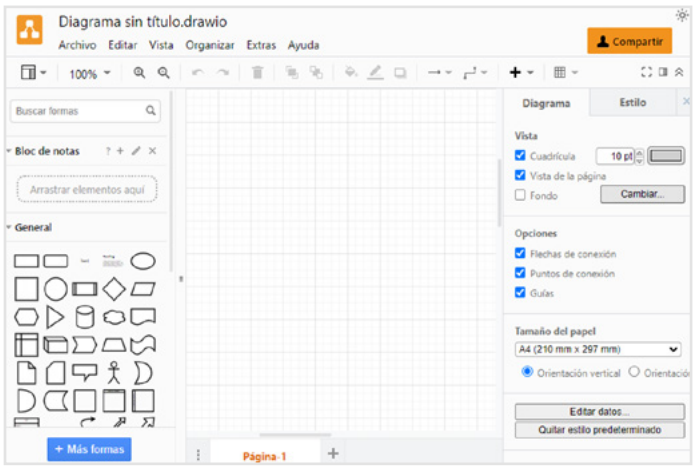
Asimismo, queremos almacenar la composición de las vacunas. Cada vacuna consta de uno o varios componentes, cada uno de los cuales se identifica mediante un código numérico y tiene un nombre único y una descripción opcional. Un componente puede aparecer en una o varias vacunas.

Cada vacuna es suministrada a Vacunak por un único laboratorio. Cada laboratorio se identifica mediante su nombre y se desea almacenar, además, la ciudad en la que está ubicado, un número de teléfono de contacto y un correo electrónico, si este es conocido. Solo almacenaremos en la base de datos laboratorios que suministran al menos una vacuna, si bien un laboratorio puede suministrar varias vacunas.

Solución caso práctico:



Para elaborar los diagramas se puede emplear el software gráfico de diagramas UML gratuito **Draw.io** https://app.diagrams.net/. Se puede utilizar tanto de forma online como offline, ya que dispone de una versión de escritorio.



www.universae.com