Departamento de Cs. e Ingeniería de la Computación

Universidad Nacional del Sur

Bases de Datos

Segundo Cuatrimestre de 2023

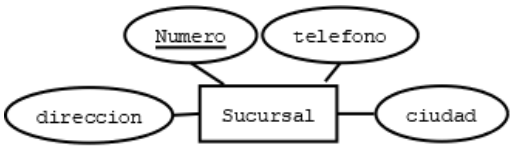
Pautas generales para el diseño del modelo Entidad-Relación

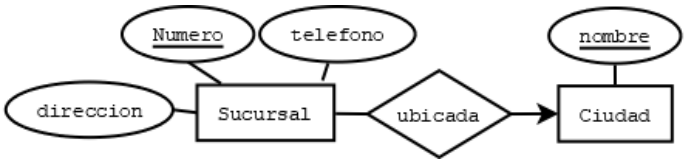
A continuación se presentan algunas pautas generales para tener en cuenta al momento de diseñar un modelo de datos utilizando el modelo Entidad-Relación. También se incluyen ejemplos para ilustrar diferentes casos y aclarar alguna dudas y confusiones que suelen ser comunes. Tenga en cuenta que al tratarse de reglas generales, puede que no resulten adecuadas para ciertos casos particulares que muchas veces dependen de las restricciones del problema que se esta modelando. En general no existe una única forma de modelar un problema, algunas pueden tener ventajas sobre otras o adecuarse mejor a las restricciones del problema. Es parte del diseño, evaluar diferentes alternativas posibles y quedarse con la que resulte mas adecuada. Como todo modelo abstracto, el modelo Entidad-Relación tiene límites en cuanto a las restricciones que se pueden representar.

¿Atributo o entidad?

Muchas veces surge la duda si representar un dato como un atributo o una entidad. Si un dato lo representamos como una entidad en lugar de un atributo (aun cuando quede una entidad con un solo atributo) tiene la ventaja de facilitar las búsquedas o consultas por ese dato, dado que cada valor estará representado una única vez en la base de datos.

Por ejemplo, supongamos que queremos representar una bases de datos de un comercio que tiene varias sucursales. La sucursales se identifican por un número, están ubicadas en una ciudad y tienen una dirección y un número de teléfono. En una ciudad puede haber varias sucursales. Consideremos dos alternativas:





En el caso de que la ciudad se represente como un atributo de la sucursal, ese dato sera almacenado como un campo de texto que contendrá el nombre de la ciudad. Con lo cual la misma ciudad podrá estar almacenada de manera diferente en diferentes sucursales. Por ejemplo la ciudad de Bahía Blanca podría estar almacenada con el texto "Bahia Blanca" en el atributo ciudad de una sucursal y en otra sucursal estar almacenado en el atributo ciudad con el texto "B. Bca.". Si se quieren hacer consultas por el atributo ciudad (por ejemplo, encontrar todas las sucursales situadas en la ciudad de Bahía blanca), al poder estar la misma ciudad almacenada de forma diferente en distintas sucursales será mas difícil responder esa consulta.

En el caso de que la ciudad se represente como una entidad con un atributo nombre como clave, cada ciudad estará almacenada una única vez, o al menos resultará mas fácil de controlarlo.

Los atributos en general deben ser valores atómicos y no deben tener una estructura que se pueda descomponer en otros valores. Una excepción a esto es el caso de las fechas y horas que pueden representarse directamente como atributos y no tiene sentido representarlos como entidades. Como veremos mas adelante, las fechas y horas son tratados de manera atómica (como un tipo de dato) por la implementación de los servidores de base de datos. Tienen una única representación, siempre se interpretan de la misma manera y podemos sacar provecho a las funciones provistas para estos tipos de datos.

¿Entidad o relacion?

¿Como determinar si un dato hay que representarlo como una entidad o como una relación? Una regla general que podemos aplicar es la siguiente: si la llave (o clave) no se puede formar con los atributos propios del dato y se necesitan atributos de otras entidades del problema esto indicaría que se trata de una relación y no de una entidad.

Por ejemplo, supongamos que queremos guardar información sobre un torneo de fútbol que se juega en un complejo deportivo con varias canchas y varios equipos. Sabemos que cada equipo tiene un nombre identificatorio y las canchas se identifican por un número. Los equipos y las canchas los podemos representar como entidades ya que tienen atributos propios para poder identificarlos (o formar una llave). Otra información a almacenar son los partidos. De estos se quiere registrar la fecha y hora en que se jugó, que equipos participaron y la cancha donde se jugó. Como el complejo tiene varias canchas, se pueden jugar varios partidos simultáneamente, por lo cual la fecha y hora no alcanza para identificar el partido (la fecha y hora no alcanzan para forma una llave). Es decir, además de la fecha y hora se necesita conocer la cancha donde se jugó, o alguno de los equipos que participaron, para poder identificar el partido. Por otra parte un partido no puede existir por si solo, si no existe una cancha donde jugarlo y dos equipos que lo disputen. Esto nos indica que lo mas adecuado es representar a un partido como una relación entre una cancha y dos equipos.

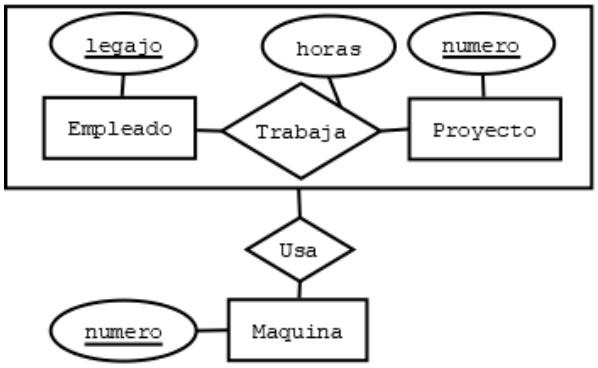
Una excepción a esta regla es el caso de la entidad débil. La diferencia es que la existencia de la entidad débil siempre esta condicionada a la existencia de una (y no varias) entidad fuerte. La llave de la entidad débil se forma con la llave de la entidad fuerte de la cual depende mas uno o más atributos (discriminador) de la entidad débil que permiten identificarla dentro de la entidad fuerte.

Por ejemplo, supongamos que queremos representar una base de datos sobre series de televisión. Cada serie tiene un nombre identificatorio que es único (no hay dos series con el mismo nombre). Cada serie puede tener varios capítulos. Cada capítulo tienen un número correlativo que establece un orden (1, 2, 3,...) y permite identificar el capítulo dentro de la serie. Con el número de capítulo sólo no podemos identificarlo, dado que el mismo número de capítulo puede aparecer en diferentes series (todas las series tienen un capítulo 1, por ejemplo). Además para que exista el capítulo tiene que existir la serie a la cual corresponde. Para representar esta situación podemos modelar el capítulo como una entidad débil que depende de la serie. Para identificar un capítulo necesitamos el nombre de la serie y el número de capítulo (discriminador) dentro de la serie.

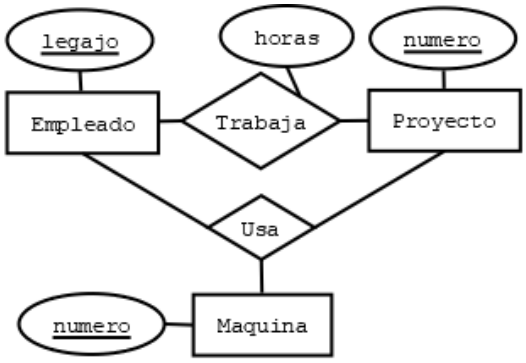
Si la serie puede tener mas de una temporada y queremos representar esta información en la base de datos, ocurre una situación similar a la que ocurre con los capítulos. Una temporada se identifica con un número correlativo dentro de una serie y el mismo número de temporada se repite en diferentes series (para identificar la temporada ademas del número necesitamos el nombre de la serie). Una temporada puede tener muchos capítulos y para identificar el capítulo necesitamos conocer el nombre de la serie y el número de la temporada.¿Como modificaría el diagrama anterior para incorporar esta información?

Agregación (agregados)

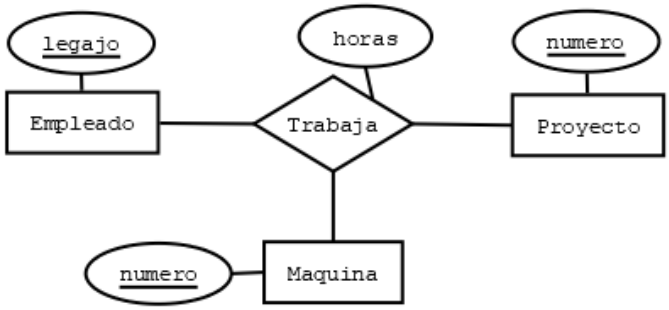
La agregación permite relacionar la relación dentro del agregado con otras entidades (o agregados) y siempre se hace sobre una sola relación (no se pude hacer un agregado de muchas relaciones). La restricción que se logra imponer a través del agregado de una relación, es la de poder relacionar otra entidad con las entidades que solo participan de la relación dentro del agregado. Por ejemplo supongamos que queremos representar información sobre trabajadores de una empresa que trabajan en varios proyectos (una determinada cantidad de horas en cada uno) y pueden utilizar diferentes máquinas cuando trabajan en un proyecto. A continuación se presentan 3 opciones para representar esta información.



(opción 1)



(opción 2)



(opción 3)

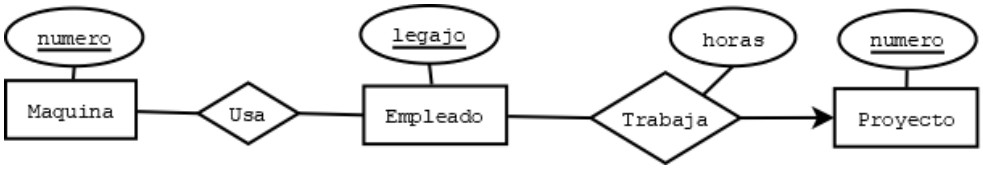
La opción 1 tiene una relación "Trabaja" entre las entidades "Empleado" y "Proyecto". La relación "Usa" vincula la entidad "Maquina" con el agregado de la relación "Trabaja". Esto permite que un empleado que trabaja en un proyecto tenga a asociado, o no, diferentes máquinas a través de la relación trabaja. Además, solo es posible vincular una máquina con un (o varios) par Empleado-Proyecto que este en la relación "Trabaja".

En la opción 2 la relación ternaria "Usa" también permite vincular, o no, diferentes máquinas a un empleado en un proyecto. Pero al ser una relación independiente de la relación "Trabaja", por medio de la relación "Usa" se pueden vincular un empleado con un proyecto en el cual el empleado no trabaja, es decir un empleado y un proyecto que no están vinculados por la relación "Trabaja".

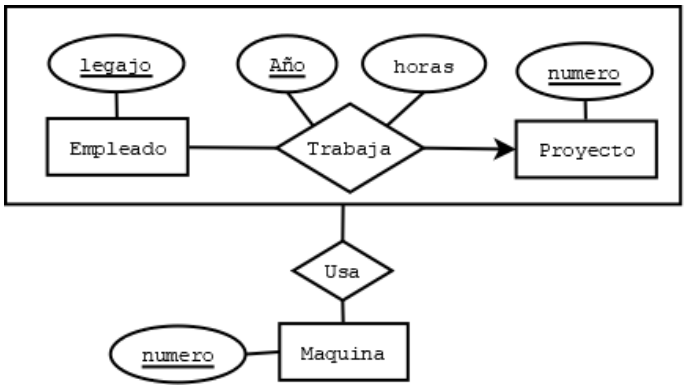
En la opción 3 se tiene una sola relación ternaria "Trabaja" que vincula empleado, proyecto y máquina. La diferencia en este caso, es que para vincular un empleado con un proyecto hay que asociarle siempre al menos una máquina (no es opcional como en las dos opciones anteriores).

Como regla general no tiene sentido hacer una agregación de una relación muchos a uno, siempre y cuando no existan atributos propios de la relación que formen parte de la llave de la relación. Por ejemplo, teniendo en cuenta el modelo anterior supongamos que cada empleado solo puede trabajar en un proyecto de la empresa. En este caso si tiene que usar una o mas máquinas podemos relacionarlas directamente con

el empleado, dado que el proyecto en el que trabaja es uno solo y podemos obtenerlo a través del empleado por medio de la relación "Trabaja". En este caso el agregado de la relación "Trabaja" no tiene sentido.



Ahora supongamos que cada empleado puede trabajar en muchos proyectos pero solo puede trabajar en un proyecto por año. En este caso si tiene sentido hacer un agregado porque, si bien la cardinalidad de relación es muchos a uno, depende del atributo "año" que forma parte de la llave de la relación. Es decir, un empleado puede participar de muchos proyectos en diferentes años y cada año que participa en un proyecto puede usar diferentes máquinas.

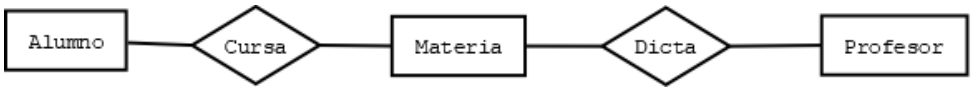


¿Una relación n-aria o varias relaciones binarias?

Una relación n-aria (n>2) se puede representar con varias relaciones binarias entre las entidades involucradas. Si se hacen varias relaciones binarias estas son independientes unas de otras, es decir algunas entidades pueden no estar relacionadas con otras porque están en relaciones binarias diferentes.

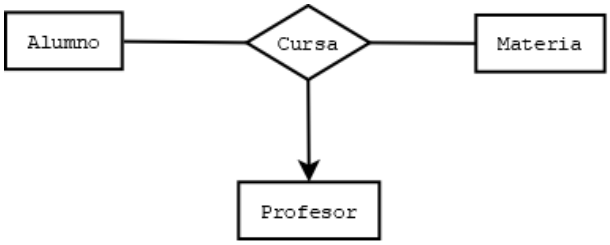
Si se utiliza una relación n-aria para que exista la relación tienen que estar todas las n entidades relacionadas y esto permite imponer ciertas restricciones de cardinalidad que no son posibles si se utilizan relaciones binarias independientes.

Por ejemplo supongamos que queremos representar información sobre alumnos, materias y profesores. Sabemos que una materia es cursada por muchos alumnos y puede ser dictada por varios profesores. A su vez un alumno puede cursar muchas materias y un profesor puede dictar muchas materias. Una forma de representar esta información sería utilizando dos relaciones binarias muchos a muchos:



En el modelo anterior no podríamos saber con que profesor cursa cada alumno porque para cada materia puede haber muchos profesores que la dicten. Es mas, uno o mas alumnos podrían cursar un materia que no tenga ningún profesor que la dicte (no este vinculada a ningún profesor por medio de la relación "Dicta") Supongamos que además queremos incorporar la restricción de que cada alumno curse cada materia con un único profesor, aunque la materia la dicten varios profesores (diferentes cursos). En el esquema anterior no es posible incorporar esta restricción, porque si modificamos la cardinalidad de relación "Dicta" para que haya un solo profesor, entonces forzamos a que cada materia sea dictada por único profesor, cuando

puede haber varios. Para modelar esta situación es necesario utilizar una relación ternaria que vincule las 3 entidades, y definir la cardinalidad para que dado un alumno y una materia haya un solo profesor.



La diferencia de este último modelo con el anterior es que para poder vincular una materia a un profesor tenemos que asociarle al menos un alumno (tienen que estar las 3 entidades para que exista la relación). En la práctica esto no resulta conveniente porque en un sistema real la asociación de la materia con el profesor se hace previamente a que los alumnos se inscriban para cursar la materia. Es decir es necesario vincular el profesor con la materia cuando no hay (o no se conoce aun) ningún alumno inscripto para cursarla. ¿Como modificaría los modelos anteriores para solucionar esta situación?