Módulo 3. MODELO RELACIONAL.

Ya hemos visto que cualquier problema que involucre manejo de datos puede ser modelado a través del modelo Entidad / Relación. Esto es así, gracias a que dicho modelo es independiente de la base de datos que se vaya a utilizar para implementar la solución.

Dentro de la gama de Sistemas Manejadores de Bases de Datos con que contamos para lograr soluciones a problemas empresariales (y de otro tipo), están los DBMS relacionales, los cuales son llamados así, debido a una serie de factores formales y matemáticos en los cuales está basada la construcción de dicho modelo.

Los DBMS relacionales implementan lo que se llama el modelo relacional.

Actualmente las bases de datos relacionales son las más extendidas en el mundo y llevan muchos años en el mercado de la tecnología. Esto es así debido a lo expresado anteriormente y que corresponde al hecho de que dicha tecnología está fundamentada en conceptos matemáticos muy profundos que la hacen una tecnología muy sólida.

Conceptos Importantes.

En la Tabla No. se muestran los términos inherentes al modelo relacional comparados con los que se utilizan en un ambiente tradicional de archivos.

Terminología Técnica del Modelo Relacional	
En el Modelo Relacional se dice…….	**.. y en Archivos Tradicionales se dice……**
Base de Datos	Conjunto de Archivos de Datos
Tabla	Archivo de Datos
Tupla	Registro
Atributo	Campo

Tabla No. Terminología Técnica del Modelo Relacional vs Sistema de Archivos

Una base de datos no es más que el conjunto de archivos en un ambiente de archivos tradicionales. Las tablas de una base de datos corresponden a cada uno de los archivos de datos en el ambiente tradicional. Cada tabla consta de una o varias tuplas, que es el mismo concepto de registros en el ambiente tradicional. Y cada tupla está constituida por un conjunto de atributos, los cuales corresponden a los campos en el ambiente anterior. A partir de este punto, seguiremos hablando con la terminología propia del modelo relacional.

Ejemplo de Modelo Relacional

A continuación se muestra un ejemplo sencillo de lo que es el modelo relacional, junto con los conceptos involucrados en él.

Tabla Departamento	
Código	**Nombre**
10	Antioquia
20	Cundinamarca
30	Valle del Cauca
40	Magdalena
50	Nariño

Tabla Ciudad				
Código	**Nombre**	**Extensión**	**Nro. Habitantes**	**Código Departamento**
100	Medellín	300	2,100,000	10
200	Ginebra	15	54,500	30
300	Villeta	35	23,600	20
400	Puerto Berrío	34	45,600	10
500	Pasto	125	560,000	50
600	Aracataca	43	25,800	40
700	Bogotá	690	7,800,000	20

Lo que se ha mostrado es una *Base de Datos* de Municipios del País. Dicha Base de Datos se compone de 2 *tablas*: Departamento y Ciudad.

La tabla Departamento tiene 5 *tuplas*, donde cada tupla se compone de 2 *atributos*: código y nombre de departamento.

La tabla Ciudad tiene 7 *tuplas*, cada tupla contiene 5 *atributos*: Código, nombre, extensión (en kms 2), número de habitantes y código de departamento.

El concepto de tupla se refiere a una serie de datos relacionados entre sí. Por ejemplo, en el ejemplo anterior la segunda tupla de la tabla Departamento se compone de 2 datos (atributos) relacionados entre sí: código 20 es el código del departamento cuyo nombre es Cundinamarca.

En términos generales, el concepto de tupla se refiere a un conjunto de atributos (A1, A2, A3, ….., An), donde cada atributo está relacionado el uno con el otro.

En el ejemplo anterior, es importante recalcar que la forma como están relacionadas las dos tablas es a través de un campo en común: código de departamento. Este campo, en este caso, es el campo conocido como clave foránea y el cual pasaré a continuación a reseñar.

Esquema Relacional

Para poder mostrar la estructura de una base de datos relacional vamos a utilizar una simbología relativamente sencilla pero demasiado clara. Esta simbología que se muestra a continuación es lo que se conoce como el esquema relacional.

El esquema de una base de datos se compone principalmente del conjunto de esquemas de las tablas que están incluidas en la base de datos. Así que debemos aprender a escribir el esquema de tablas individuales.

Para escribir el esquema de una tabla, vamos a utilizar la siguiente sintaxis:

Nombre Tabla = {Atributo 1, Atributo 2,……, Atributo n }

Por ejemplo, el esquema de la base de datos del ejemplo anterior sería el siguiente:

Departamento = { código, nombre }
Ciudad = { código, nombre, extensión, no.habitantes, código departamento }
Código departamento referencia a Departamento (código)

En el anterior esquema, además de lo enunciado en la sintaxis expuesta, existen otros conceptos que más adelante los abordaremos como son clave primaria y clave foránea.

Dominio de un Atributo

El dominio de un atributo se define como los posibles valores que puede tomar un atributo dado.

Este es un concepto que conserva su definición en el ámbito de las matemáticas.

En la siguiente tabla se muestran algunos ejemplos de atributos con sus respectivos dominios. Además se dejan en blanco algunos otros para que usted los resuelva.

Ejemplo de dominios de atributos	
Atributo	**Dominio**
Edad de empleado	Valores numéricos enteros entre 18 y 65.
Nombre de paciente	En teoría, cualquier conjunto de caracteres.
Estrato Social	1, 2, 3, 4, 5 o 6
Paciente fumador?	Si, No
Religión empleado	
Modelo Avión	
Nombre carrera a estudiar	
Género de libro	
Número celular médico	

Tabla No. Ejemplos de Dominios de Atributos.

Conocer el dominio de un atributo sirve para enriquecer la semántica del modelo, tanto Entidad / Relación como relacional. Específicamente es importante en el modelo relacional ya que en este punto del proceso del diseño de una base de datos debemos determinar de que tipo y que longitud van a tener los atributos de una tabla.

Claves en el Modelo Relacional

Así como en el modelo Entidad / Relación existen atributos claves para las entidades, es decir, atributos que identifican en forma única a cada una de las instancias de la entidad, en el modelo relacional existen también tipos de claves que cumplen funciones parecidas dentro de las tablas.

Las claves que existen en el modelo relacional son las siguientes:

- Superclave
- Clave Candidata
- Clave Primaria
- Clave Foránea

Como se verá en la explicación, los dos primeros tipos de claves son más conceptuales que prácticos, mientras que los dos últimos son los más utilizados en las bases de datos relacionales.

A continuación se mostrarán los diferentes tipos de claves que existen en el modelo relacional, ilustrados a través de un mismo ejemplo, el cual se enuncia a continuación. Vamos a trabajar los 4 tipos de claves con el siguiente esquema de tabla:

Vehículo = { placa, marca, modelo, capacidad, no.kilometros, no.motor }

Superclave

Imaginemos las tuplas que existirán en la tabla Vehículo y será posible que existan 2 o más tuplas con el mismo valor de placa? No.

Será posible que existan 2 o más tuplas con el mismo valor de número de motor? No. Al igual que la placa, el número del motor es único para cada carro.

Será posible que existan 2 o más tuplas con el mismo valor de placa y modelo (concatenados sus valores)? No.

Será posible que existan 2 o más tuplas con el mismo valor de número de motor y capacidad (concatenados sus valores)? No.

Los anteriores cuatro casos son solo algunos de los casos de superclaves que existen para la tabla Vehículo.

La definición de superclave es la misma que la de clave de una entidad.

Superclave es aquel conjunto de atributos que identifica en forma única cada una de las tuplas de una tabla.

Desde este punto de vista, será el modelo del vehículo una superclave? Lo será la placa y el número del motor (concatenados sus valores)? Será el modelo y la capacidad superclaves?

Cuantas superclaves posibles tiene la tabla Vehículo?

Clave Candidata

Dentro de la tabla Vehículo, existen 2 claves candidatas:

Placa y Número de Motor.

Estas dos claves candidatas surgen de revisar las superclaves e identificar los atributos que son necesarios en todas las superclaves. Si analizamos con detenimiento todas las superclaves, en todas están incluidos los atributos placa y/o número de motor.

Por ejemplo, si analizamos la superclave placa, modelo (concatenados sus valores) nos damos cuenta de que en esta superclave el atributo modelo sobra; teniendo solo el número de la placa nos basta para identificar en forma única cada tupla de Vehículo.

Si analizamos la superclave número motor, capacidad, marca (concatenados sus valores) nos damos cuenta de que nos basta con el número del motor para cumplir con la función de clave.

Esta es la razón de las claves candidatas enunciadas anteriormente.

Se define una clave candidata como una super clave mínima.

Clave Primaria

La tabla Vehículo tiene dos posibles claves primarias:

Placa o Número de Motor.

Una tabla dentro de una base de datos relacional solo puede tener una clave primaria.

Pero cual es, entonces, la clave primaria de esta tabla? Es la que el analista escoja como identificador de cada una de las tuplas de la tabla. Por eso es que las claves candidatas se llaman así…..porque son candidatas a convertirse en claves primarias de sus respectivas tablas. Y es el analista el jurado encargado de escoger entre las claves candidatas la clave primaria.

Y cual es el criterio para escoger la clave primaria? Simplemente se determina cual va a ser el dato por el cual se va a identificar de una manera más frecuente e intuitiva cada una de las tuplas de la tabla. En este caso, cree usted que sería muy práctico identificar cada vehículo por su número de motor, sabiendo que éste se compone de 20 caracteres o más? En cambio, el número de la placa es más fácil de manejar por parte del usuario. Por eso, en un momento dado se elige la placa como la clave primaria.

La clave primaria de una tabla es aquella clave candidata que el analista escoja para cumplir con la función de identificación única de cada una de las tuplas de la tabla.

En cuales casos sería conveniente definir el número del motor como la clave primaria de la tabla Vehículo?

Para mostrar la clave primaria de una tabla a través de su esquema relacional, simplemente subrayamos el atributo correspondiente. Para el caso de Vehículo sería así:

Vehículo = { placa, marca, modelo, capacidad, no.kilometros, no.motor }

Y para el caso de las tablas Departamento y Ciudad del ejemplo inicial del módulo sería así:

Departamento = { código, nombre }
Ciudad = { código, nombre, extensión, nohabitantes, codigo depto }

Clave Foránea

En el ejemplo expuesto, la tabla Vehículo no posee clave foránea. Más adelante veremos el ejemplo concreto.

Una clave foránea es aquel o aquellos atributos de una tabla a través de los cuales dicha tabla se relaciona con otra u otras tablas.

Pero la tabla Ciudad tiene una clave foránea que le permite relacionarse con la tabla Departamento. Y es a través de esta clave foránea que logramos saber una ciudad dada a qué departamento pertenece.

Volvamos a mirar el esquema de la base de datos de los municipios del país. Para poder definir la clave foránea en mención, se escribe de la siguiente manera:

Departamento = { código, nombre }
Ciudad = { código, nombre, extensión, nohabitantes, codigo_depto }
codigo_depto referencia a Departamento (código)

Como se puede apreciar, lo que se hace es definir la clave foránea *inmediatamente después* de escribir el esquema de la tabla dueña de la clave foránea. A través de la frase "código_depto referencia a Departamento (código)" se define que el campo código_depto es un campo que está relacionando a la tabla Ciudad con Departamento, a través de su clave primaria, es decir, tabla Departamento, campo código.

Toda clave foránea de una tabla debe referenciar (o apuntar) a la clave primaria de la tabla con la cual se está relacionando.

Lo anterior es especialmente importante ya que si existiera una clave foránea que apuntará a otro atributo no clave, habría indicios de un mal diseño de la tabla. Este aspecto se podrá entender mucho mejor en el módulo de Normalización de Datos.

Reglas de Traducción de un Modelo Entidad / Relación a un Modelo Relacional

Dentro del mundo organizacional, la rutina diaria de un departamento de sistemas (y en realidad, de cualquier departamento) exige resultados en forma frecuente y rápida. Esto ha hecho que los desarrolladores de software, a veces, obvien ciertos pasos que les exige la Ingeniería de Software con tal de obtener esos resultados esperados por los jefes inmediatos.

La anterior situación es gravísima y se refleja igual en el mundo de las bases de datos. Muchos diseñadores de bases de datos están tan enfrascados en su mundo técnico del diseño que, a veces, no se acuerdan de que existe un paso previo y es el análisis. Además, los jefes no son conscientes de que el desarrollo del análisis asegurará menos tropiezos posteriores al implementar la base de datos.

En el mundo de las bases de datos, el modelo Entidad / Relación sería la etapa del Análisis y el modelo Relacional sería el Diseño. Muchas veces los diseñadores de bases de datos empiezan su labor de diseño directamente en el motor de bases de datos, sin un previo análisis de la situación a solucionar, es decir, sin haber realizado el Modelo Entidad / Relación. Es decir, el Modelo Entidad / Relación es menospreciado

como una actividad que consume tiempo y aporta muy poco en el desarrollo de las bases de datos.

Qué equivocación tan grande! Hacer el Modelo Entidad / Relación es asegurarnos de que la solución al problema va a suplir en un 100 % las necesidades del usuario. Al fin y al cabo, el diagrama Entidad / Relación se hace con la ayuda del usuario final y con base en sus requerimientos. Y si a esto le sumamos que existen una reglas que nos permiten traducir modelos Entidad / Relación a modelos relaciones, asegurándonos un diseño eficiente de bases de datos, la importancia del Modelo Entidad / Relación es muchísimo mas trascendental.

Es decir, si nos aseguramos que realizamos muy bien el Modelo Entidad / Relación, y le aplicamos las reglas de traducción mencionadas, aseguraremos que el resultado final será un primer diseño de bases de datos que permita solucionar el problema. Y lo que es más importante, y más adelante se entenderá el por qué, es que ese diseño inicial en realidad será el definitivo porque desde el punto de vista del problema modelado en el diagrama Entidad / Relación, será el más eficiente.

En este apartado se pretende detallar las 8 reglas de traducción necesarias para traducir un modelo Entidad / Relación a un Modelo Relacional.

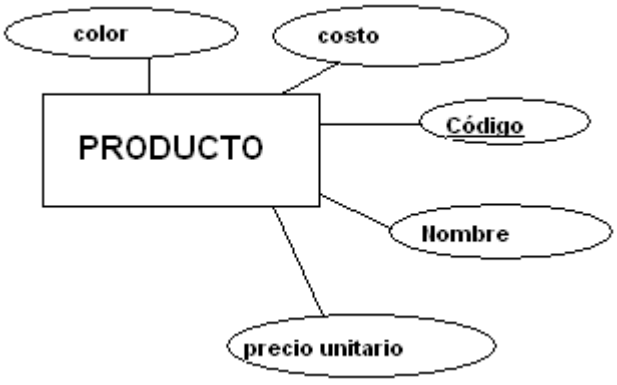
Como traducir una Entidad Normal / Fuerte?

Una entidad normal es aquella que no es ni fuerte ni débil. La mayoría de las entidades de un modelo Entidad / Relación son normales. Las entidades fuertes son aquellas que le “prestan” su clave a las entidades débiles para formar su propia clave.

La regla de traducción para ambos tipos de entidades es la misma y es la regla más directa y simple.

Una entidad normal / fuerte se traduce al modelo relacional como una tabla con el mismo nombre de la entidad y con sus mismos atributos. Además la clave primaria de la tabla es la clave de la entidad.

Ejemplo:



La entidad mostrada en el anterior ejemplo es una entidad normal y, por lo tanto, se traduce de la siguiente forma:

Producto = { código, nombre, precio_unitario, costo, color }

Como traducir una Entidad Débil?

La entidad débil siempre reflejará su dependencia con otra entidad (entidad fuerte) que le permita construir su propia clave. Como recordaremos, la entidad débil debe valerse de la clave de la entidad fuerte y, haciendo la analogía con los conceptos del modelo relacional, simplemente lo que está pasando es que esta entidad débil de alguna manera debería tener una especie de clave foránea que referencie la clave primaria de su entidad fuerte.

Una entidad débil se traduce como una tabla que tiene los mismos atributos de la entidad débil junto con el atributo clave de su entidad fuerte. La clave primaria de dicha tabla será compuesta por esta clave de la entidad fuerte y el discriminante.

Recuerde que el discriminante de una entidad débil corresponde a aquel o aquellos atributos de la entidad débil que, junto con la clave de la entidad fuerte, forman la clave de la entidad. Y recordar que este discriminante en el modelo Entidad / Relación se dibuja con un subrayado punteado.

Ejemplo:

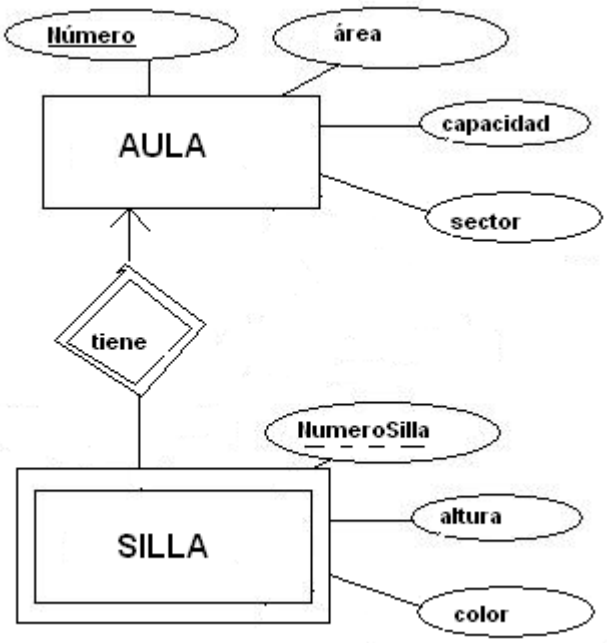
Para el desarrollo del ejemplo, mirar la gráfica de la página siguiente.

La entidad AULA, por ser una entidad fuerte, es traducida con la regla anterior:

Aula = { número, área, capacidad, sector }

Pero, en cambio, la entidad SILLA, por ser débil se traduce de la siguiente manera:

Silla = { número_aula, número_silla, altura, color }
número_aula referencia a Aula (número)



Como se puede apreciar en la traducción anterior, la clave de la tabla resultante de la traducción de una entidad débil siempre será compuesta por 2 o más atributos. Y además, la dependencia de la entidad débil con su fuerte se reflejará a través de una clave foránea que se genera en la tabla "débil" y que apunta a la clave primaria de la tabla "fuerte".

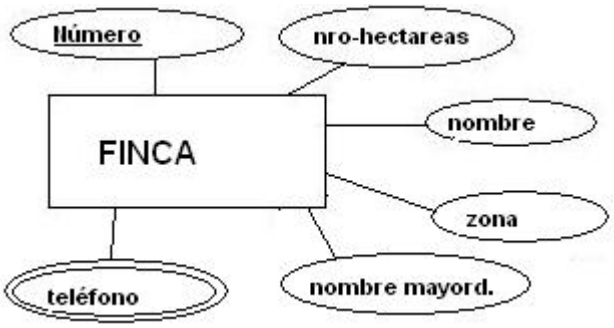
Nota: Como se puede apreciar en la gráfica del ejemplo, a la relación entre una entidad fuerte con su respectiva entidad débil se le grafica con doble rombo, a diferencia de una relación entre dos entidades normales que se grafica con simple rombo.

Como traducir un atributo multivalorado?

Para facilitar la consulta de una tabla a través de un atributo en particular, es importante que este atributo no sea multivalorado, es decir, que no posea varios valores por tupla. Esa es la razón de ser esta regla de traducción. El por qué es importante evitar atributos multivalorados se comprenderá mucho mejor en el módulo de Normalización de Datos.

Un atributo multivalorado se traduce como una nueva tabla cuyos atributos son la clave de la entidad y el atributo multivalorado. La clave primaria de esta tabla será compuesta por ambos elementos.

Ejemplo:



Como se nota, la entidad Finca es una entidad normal que se traduce con la regla correspondiente.

Finca = { número, nombre, no.hectáreas, zona, nombre-mayordomo }

Y al atributo multivalorado teléfono se le aplica su regla de traducción correspondiente.

Finca-Tel = { número-finca, teléfono }
número-finca referencia a Finca (número)

En esta nueva tabla se genera una clave foránea (número-finca) para poder relacionar la tabla Finca-Tel con su tabla padre Finca.

Si una entidad posee N atributos multivalorados, la traducción de dichos atributos dará como resultado N tablas nuevas, una por cada atributo multivalorado.

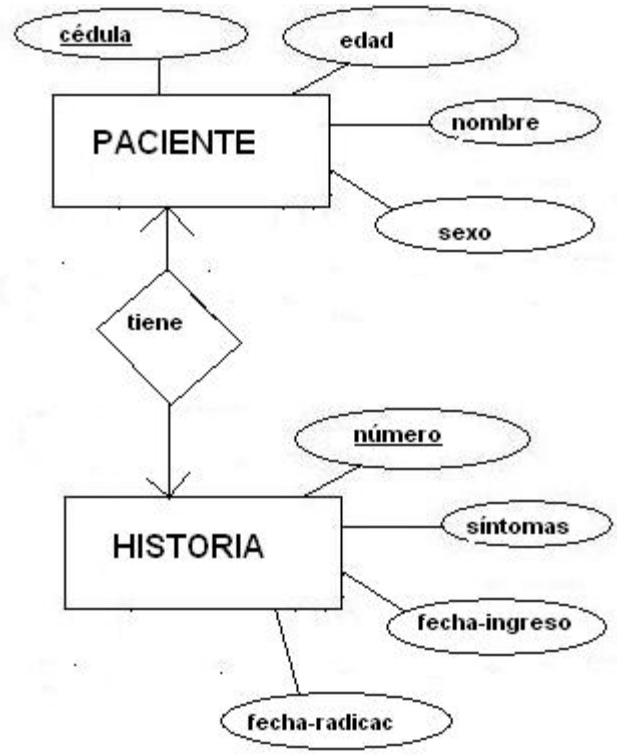
Lo resaltado anteriormente es importante ya que no podemos asegurar que, si una entidad tiene N atributos multivalorados, todos ellos tendrán la misma cantidad de valores por tupla y por lo tanto, si los metemos a todos dentro del diseño de una misma tabla, tendríamos ciertos atributos nulos para algunas tuplas de la tabla.

Como traducir una relación uno a uno?

A pesar de que la relación uno a uno se presenta esporádicamente en los modelos del mundo real, se tiene una regla de traducción para ella. Como se podrá ver, la relación uno a uno tiene tres posibilidades de traducción, dos de las cuales son válidas.

La relación uno a uno entre 2 entidades se traduce colocando la clave de cualquiera de las entidades como clave foránea de la tabla correspondiente a la otra entidad.
Si las cardinalidades son diferentes, la entidad de menor cardinalidad heredará la clave primaria de la otra entidad.

Ejemplo:



Para traducir la entidad Paciente utilizamos la regla de traducción de las entidades normales:

Paciente = { cedula, nombre, edad, sexo }

Lo mismo hacemos para traducir la entidad Historia Clínica:

Historia = { numero-historia, fecha-radicacion, fecha-ingreso, síntomas }

Hasta este punto tenemos dos tablas sin ningún tipo de relación. Son tablas aisladas. Por lo tanto nos falta reflejar la relación que existe entre ellas, es decir, reflejar que un paciente tiene una sola historia clínica y una historia clínica pertenece a un solo paciente. Como se pudo ver antes, la regla de traducción del atributo multivalorado promulga que, en lo posible, una tabla no debe poseer atributos multivalorados, a no ser que no se vayan a hacer consultas por ese atributo. Teniendo como premisa esto, debemos relacionar ambas tablas, teniendo cuidado de no generar atributos multivalorados.

Para este fin, existen tres posibilidades de relación:

- Poner la clave de la entidad Paciente como clave foránea en la tabla Historia Clínica.

  Las tablas quedan de la siguiente manera:

Paciente = { cedula, nombre, edad, sexo }
Historia = { numero-historia, fecha-radicacion, fecha-ingreso, síntomas,
Cedula-paciente }
Cedula-paciente referencia a Paciente(cédula)

- Poner la clave de la entidad Historia como clave foránea en la tabla Paciente.

  Las tablas quedan de la siguiente manera:

  Paciente = { cedula, nombre, edad, sexo, nro-historia }
  Nro-historia referencia a Historia(numero-historia)
  Historia = { numero-historia, fecha-radicacion, fecha-ingreso, síntomas }

- Las dos posibilidades anteriores conjugadas.

  Las tablas quedan de la siguiente manera:

  Paciente = { cedula, nombre, edad, sexo, nro-historia }
  Nro-historia referencia a Historia(numero-historia)

  Historia = { numero-historia, fecha-radicacion, fecha-ingreso, síntomas,
  Cedula-paciente }
  Cedula-paciente referencia a Paciente(cédula)

Existen varios aspectos para aclarar sobre las tres posibilidades anteriores:

a) Definitivamente nos damos cuenta que si existen las dos primeras posibilidades, la tercera posibilidad es redundante. Con poner una sola clave foránea en alguna de las tablas basta para relacionar ambas tablas. Por lo tanto, la tercera posibilidad se desecha.
b) Siempre que una tabla tenga claves foráneas, éstas deben ser definidas INMEDIATAMENTE DESPUES de la definición de la tabla. Dichas claves foráneas se definen tal y como se muestra en el ejemplo, es decir, enunciando el nombre de la clave foránea y diciendo a que tabla y campo hace referencia (o apunta a). Por ejemplo, cedula-paciente referencia a Paciente(cedula).
c) Analizando la primera posibilidad, nos damos cuenta que la clave foránea generada no queda siendo multivalorada ya que, por modelamiento sabemos, ***que cada historia clínica corresponde a un solo paciente; es decir, cada una de las tuplas que hay en la tabla Historia, van a contener un solo valor para el campo Cedula-paciente.*** Por lo tanto, esta posibilidad es válida.
d) Si miramos la segunda posibilidad, nos damos cuenta exactamente de lo deducido en el numeral c). Esto lo concluimos porque sabemos ***que cada paciente tiene una sola historia clínica, es decir, cada tupla de la tabla Paciente van a tener un solo valor en el campo Nro-historia.*** Por lo tanto, esta posibilidad también es válida.

Por lo tanto concluimos que cualquiera de las dos primeras posibilidades es la traducción correcta.

Como traducir una relación uno a muchos?

La relación uno a muchos es un tipo de relación frecuente en el modelamiento del mundo real. A diferencia de la relación uno a uno, esta relación tiene una sola posibilidad de traducción válida. Nunca se debe olvidar al hacer estas traducciones que no es conveniente dejar atributos multivalorados en las tablas, mucho menos si son claves foráneas.

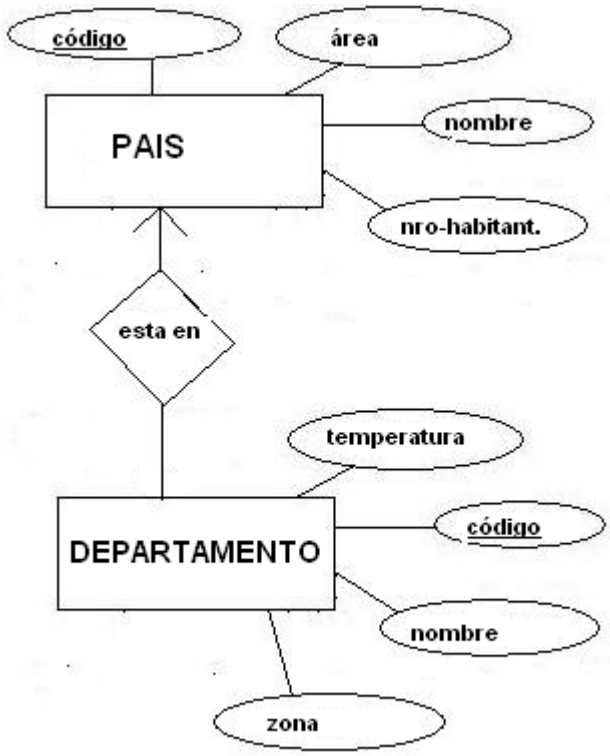
Una relación uno a muchos se traduce colocando la clave de la entidad del lado de "uno" de la relación como clave foránea en la tabla correspondiente a la entidad del lado de "muchos" de la misma.

Ejemplo:

Si analizamos la relación anterior, nos damos cuenta de que es uno a muchos porque un país tiene muchos departamentos, pero cada departamento corresponde a un solo país. Por lo tanto, la entidad del lado de "uno" de la relación es Pais, mientras que Departamento es la entidad del lado de "muchos".

Si aplicamos exactamente la regla de traducción expuesta anteriormente obtenemos el siguiente diseño de tablas:

Pais = { codigo, nombre, area, numero-habitantes }
Departamento = { codigo-dep, nombre, zona, temperatura-promedio, codpais }
Codpais referencia a Pais(codigo)



Como podemos analizar del diseño anterior, el campo codpais, el cual es la clave foránea, no es multivalorado ya que cada tupla de Departamento solo contendrá un único código de país.

Cabría la otra posibilidad que es colocar el código de departamento como clave foránea de la tabla País, pero si analizamos detenidamente esta opción nos daremos cuenta de que no es correcta. **Por qué?**

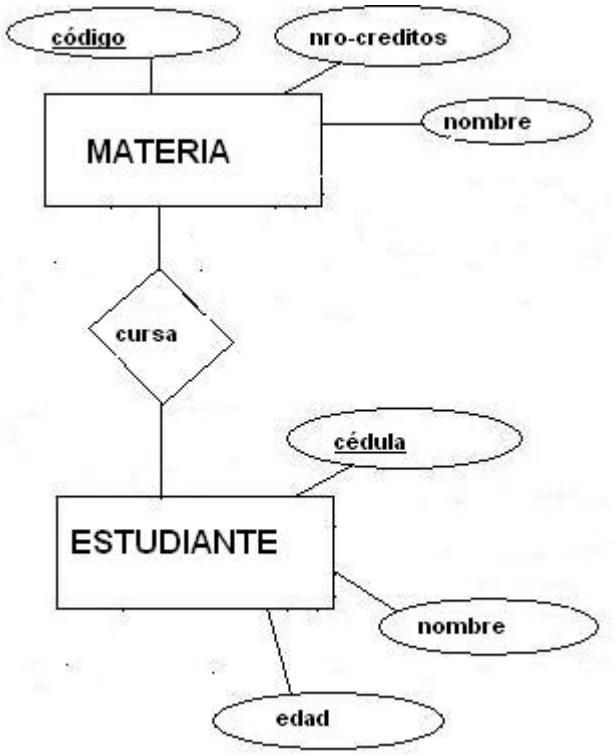
Como traducir una relación muchos a muchos?

La relación muchos a muchos es la más común dentro del modelamiento del mundo real.

La relación muchos a muchos entre dos entidades se traduce como una nueva tabla, la cual contendrá las claves primarias correspondientes a las tablas de las dos entidades. La clave primaria de esta nueva tabla estará compuesta por estos dos componentes.

Ejemplo:

Como podemos ver en el diagrama siguiente, la relación entre Materia y Estudiante es una relación Muchos a Muchos. Por lo tanto, su traducción daría como resultado una nueva tabla, aparte de las tablas correspondientes a Materia y a Estudiante.



Materia = { codigo, nombre, nro-creditos }
Estudiante = { cedula, nombre, edad }

MateEstu = { codigo-mat, cedula-est }
Codigo-mat referencia a Materia(codigo)
Cedula.est referencia a Estudiante(cedula)

Como se puede ver, se genera una nueva tabla llamada MateEstu la cual está compuesta de las claves primarias de las otras dos tablas, y cuya clave primaria es la composición de estos dos campos (por eso aparecen ambos subrayados).

Analizar por qué hay que generar una tercera tabla.

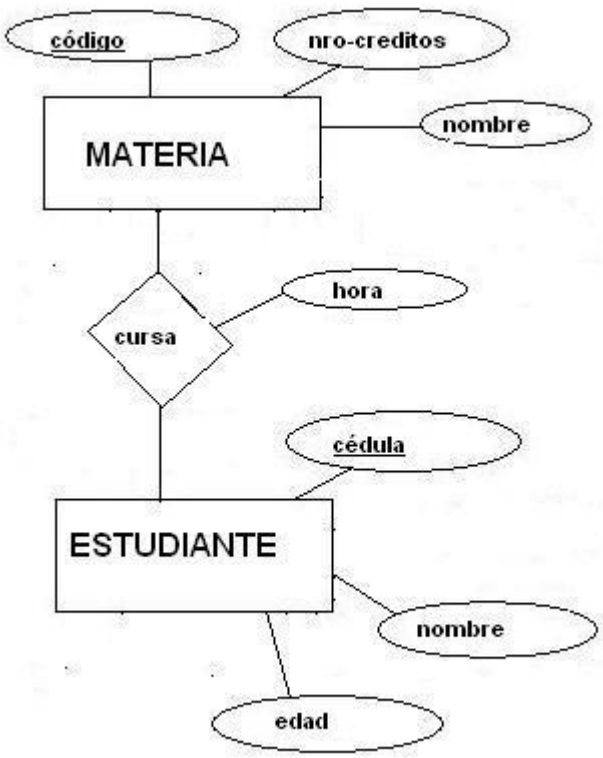
Como traducir una agregación?

Como se mencionó en el módulo 2, la agregación es una situación que solamente se da en relaciones muchos a muchos. Por lo tanto, para traducir una agregación, se utiliza la regla de traducción de la relación muchos a muchos, expuesta anteriormente.

Recordemos que una agregación es la situación en la cual una relación muchos a muchos tiene atributos propios.

Una agregación se traduce poniendo los atributos de la relación como atributos de la tercera tabla generada por la traducción de la relación muchos a muchos.

Ejemplo:



En el anterior ejemplo, vemos como el atributo Hora es una agregación de la relación Cursa. Por lo tanto, este atributo formará parte de la tabla MateEstu vista en la regla de traducción anterior. Por lo tanto, la tabla MateEstu queda de la siguiente forma:

MateEstu = { codigo-mat, cedula-est, hora }
Codigo-mat referencia a Materia(codigo)
Cedula.est referencia a Estudiante(cedula)

Vale aclarar que si una agregación tiene varios atributos, todos van en la tercera tabla generada por la relación muchos a muchos.

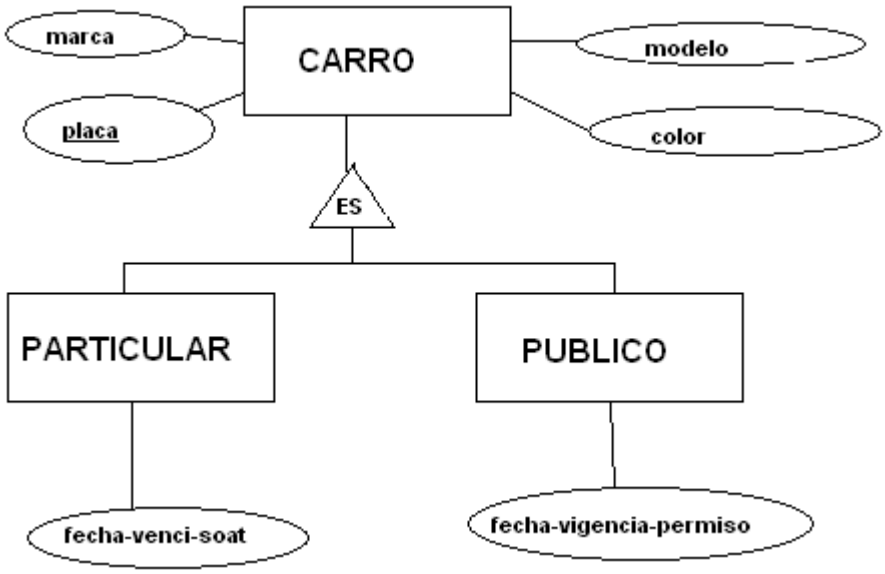
Como traducir una especialización disjunta?

Como recordaremos en temas anteriores, hablábamos de la diferencia sustancial que existe entre una especialización disjunta y una solapada. Recordemos que la disjunta es aquella especialización donde las instancias de la entidad padre se pueden ver reflejadas en una sola entidad hija, no en dos o más.

Una especialización disjunta con N entidades hijas tiene dos posibles traducciones igualmente válidas:

- Se traduce en N+1 tablas, cada tabla representando una entidad del diagrama. La entidad padre se traduce literalmente, es decir, aplicando la regla de traducción de la Entidad Normal. Cada entidad hija se traduce como una tabla cuyos atributos son la clave de la entidad padre junto con los atributos propios. La clave primaria de estas "tablas hijas" será la clave de la entidad padre.
- Se traduce en N tablas, una por cada entidad hija. Cada tabla generada se compondrá de los atributos de la entidad padre junto con los atributos propios de la entidad hija. Y la clave primaria de cada tabla es la clave de la entidad padre.

Ejemplo:



Apliquemos la primera posibilidad de traducción que existe:

Carro = { placa, marca, modelo, color }
Particular = { placa, fecha-venci-soat }
Placa referencia a Carro(placa)
Publico = { placa, fecha-vigencia-permiso }
Placa referencia a Carro(placa)

Como se puede ver en este caso, se está aplicando el concepto de herencia que hablamos en su momento, es decir, las entidades hijas heredaban los atributos de la entidad padre. Acá se manifiesta esto a través del concepto de que las "tablas hijas" tienen clave foráneas que apuntan a la "tabla padre" y de esta forma desde las hijas se captura la información del padre.

Como comentario adicional, este es el único caso en el cual la clave primaria de una tabla es a la vez clave foránea de la misma.

Aplicando la segunda posibilidad de traducción tenemos lo siguiente:

Particular = { placa, marca, modelo, color, fecha-venci-soat }
Publico = { placa , marca, modelo, color, fecha-vigencia-permiso }

Desprevenidamente podemos concluir que esta segunda traducción tiene redundancia de datos y que por lo tanto no es válida. Pero SÍ ES VALIDA Y NO TIENE REDUNDANCIA. Por qué?

La decisión de utilizar una traducción u otra dependerá del diseñador de la base de datos que tendrá criterios suficientes para determinar la mejor opción de acuerdo a las consultas que se van a hacer sobre las tablas.

Como traducir una especialización solapada?

Recordemos que una especialización solapada es aquella donde las instancias de la entidad padre pueden verse reflejadas en más de una entidad hija.

La especialización solapada, a diferencia de la disjunta, tiene una sola opción de traducción y corresponde a la primera regla de traducción enunciada en la especialización disjunta, es decir, N+1 tablas cada tabla representando una entidad del diagrama. La entidad padre se traduce literalmente, es decir, aplicando la regla de traducción de la Entidad Normal. Cada entidad hija se traduce como una tabla cuyos atributos son la clave de la entidad padre junto con los atributos propios. La clave primaria de estas "tablas hijas" será la clave de la entidad padre.

Si a este tipo de especialización le aplicamos la segunda posibilidad de traducción de la disjunta, el diseño quedaría con redundancia de datos. Por qué?
Ejemplo:

Como podemos ver, la especialización del diagrama siguiente es solapada ya que un profesor puede ser a la vez de medio tiempo e investigador.

Aplicando la regla de traducción, tenemos lo siguiente:

Profesor = { cedula, nombre, edad, profesión }
MedioTiempo = { cédula, horario-atencion }

Cédula referencia a Profesor(cédula)
Investigador = { cédula, tema-proyecto, horas-semanales }
Cédula referencia a Profesor(cédula)

Como podemos ver, la información que es común a cualquier tipo de profesor se tendrá en la tabla Profesor y lo que es específico de cada tipo se tiene en las tablas MedioTiempo e Investigador. Así podemos concluir que en este diseño no se genera redundancia de información.

Conclusión:

Aplicando estas reglas de traducción a cualquier modelo entidad relación, obtendremos como resultado un diseño de base de datos eficientemente diseñado. La razón de esta eficiencia se explicará en el siguiente módulo referente al tema de Normalización de Datos.