# 3

# EL MODELO RELACIONAL.

## Origen y ventajas del modelo relacional

#### Origen

El modelo relacional fue propuesto por el matemático inglés y doctor en informática Edgar Frank Codd en 1970 mientras trabajaba como investigador en el laboratorio de investigación de IBM en San José, California.

En aquella época, IBM y sus competidores, estaban volcados en comercializar los modelos jerárquicos y en red, cuyas estructuras de datos y almacenamiento eran complejas y difíciles de entender.

Codd y sus colaboradores desarrollan un prototipo de sistema gestor de bases de datos relacional, denominado System R, a comienzos de 1970, demostrando la practicidad del modelo relacional, de sus estructuras y operaciones, además de técnicas para control de concurrencia, optimización de consultas, gestión de transacciones, seguridad, recuperación, usuarios e interfaces de usuario.

System R es la base de DB2, el SGBD Relacional de IBM.

Siguiendo los documentos de investigación publicados por Codd y su equipo, se desarrolla INGRES, otro proyecto de investigación del modelo relacional, en la Universidad de Berkeley en California. Actualmente está disponible como POSTGRES o post-INGRES. Los orígenes de ORACLE están en 1977, atraído por las posibilidades comerciales de System R.

#### Ventajas

El modelo relacional se basa en la noción matemática de relación. Su estructura en muy simple, lo que lo hace muy sencillo de entender de forma intuitiva.

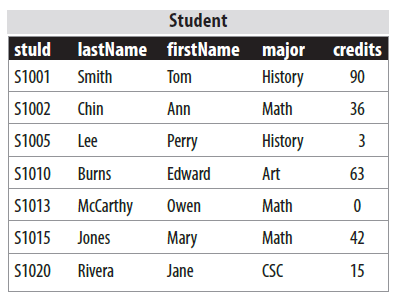
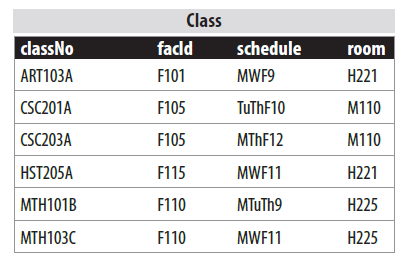
Permite separar el modelo conceptual de la base de datos del físico (el almacenamiento), lo que hace que se entienda con facilidad.

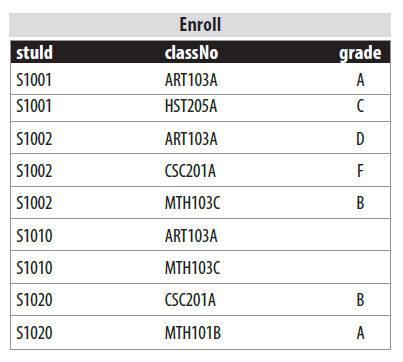
Las operaciones con datos se expresan de forma sencilla, no requieren conocer las estructuras de almacenamiento, con pocos comandos, pero poderosos, que permiten realizar operaciones muy simples o muy complejas.

Por todo esto es el modelo más popular de bases de datos.

## Estructuras del modelo relacional

Las estructuras utilizadas en el modelo relacional son las tablas y las relaciones entre ellas. Ejemplo:





## Tablas

Las tablas son unos objetos bidimensionales que se usan para contener información acerca de los objetos a almacenar en la base de datos.

Usando la terminología del modelo E-R, tanto las entidades como las relaciones se representan en tablas.

En las tablas, en las **filas** de la tabla están las **ocurrencias** de las entidades y de las relaciones y en las **columnas** los **atributos**.

Por ejemplo, la entidad estudiantes se representan en la tabla **Student**, en las columnas están los atributos de esa entidad, y en las filas cada una de las ocurrencias que se almacenan.

Las columnas contienen valores de un único atributo, siendo el dominio de ese atributo los valores que puede contener, los valores permitidos para ese atributo. Los dominios pueden ser iguales o distintos para diferentes columnas, o atributos.

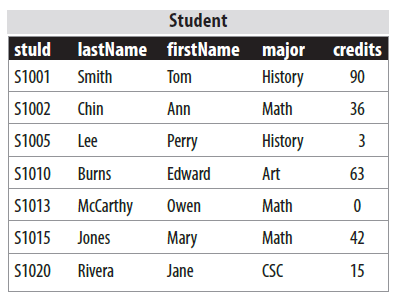
En el ejemplo de la tabla **Student**, *lastName* es una cadena de caracteres y *credits* es un número.

En el modelo relacional, a las tablas se les denomina relaciones, a las columnas de una tabla atributos y a las filas de una tabla, tuplas.

En las tablas del modelo relacional:

* Cada celda (intersección fila / columna), que es el valor del atributo de una ocurrencia, sólo tiene un valor.
* Cada columna tiene un nombre distinto, el nombre del atributo que representa.
* Todos los valores en una misma columna tienen el mismo dominio pues son valores del mismo atributo.
* Cada fila (o tupla) es distinta, no hay filas (o tuplas) duplicadas.
* El orden de las filas (o tuplas) es irrelevante.
* El orden de las columnas (atributos) es irrelevante.

Ejemplo de estas restricciones:



* En la tabla **Student**, no se pueden almacenar valores múltiples en una misma celda. No se pueden almacenar dos especialidades en la columna *major*.
* Los nombre escritos en la columnas (studId, lastName, etc.) son los atributos de la relación.
* No se permite que en el campo *credits* aparezca un nombre, porque un texto no es del domino de *credits*.
* No puede haber filas duplicadas, por ejemplo, S1001 sólo aparece una vez.
* No importa que la fila de S1001 esté antes que la de S1002.
* Si en la tabla se pone el campo *firstName* antes que *lastName*, sigue siendo la misma tabla (relación).

## Claves

En una tabla debe poderse separar del resto una de las filas.

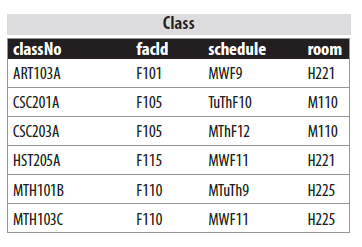
Debe de haber un atributo, o una combinación de atributos, que haga distinta a las filas (o tuplas).

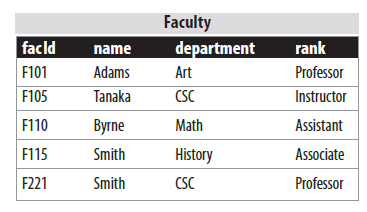
Deberá haber al menos una clave candidata y por tanto una clave primaria para cada tabla (relación).

En el peor de los casos podría ser todo el conjunto de atributos el que tuviese que examinarse para separar una de las filas, una de las tuplas.

Una clave externa, es un atributo o combinación de atributos de una relación (o tabla) que no es la clave primaria de la relación, pero que es la clave primaria, habitualmente, de otra relación.

Estas claves externas son muy importantes en el modelo relacional pues se usan para representar las conexiones entre tablas, las relaciones del modelo E-R.

Ejemplo:



En la tabla Class el atributo o columna facId es clave externa en Class. No es la clave primaria de la tabla (es classNo) pero es clave primaria de una tabla diferente, Faculty.

Tener dicho atributo en la tabla Class permite conectar una clase con el profesor que imparte la clase, es decir una relación entre las dos tablas.

Las claves externas, pueden conectar la fila con otra fila de la misma tabla. Si recordáis, en las relaciones unarias del modelo E-R, las relaciones se establecen entre la misma entidad.

Ejemplo:

Empleados

|  |  |  |  |
| --- | --- | --- | --- |
| empId | nombre | apellidos | jefeId |
| E001 | Roberto | Pereda Aguilera | E003 |
| E002 | Sergio | Viejo González | E003 |
| E003 | José Luis | Santiago Fernández |  |
|  |  |  |  |

## Restricciones de integridad

Es importante preservar la integridad, es decir la exactitud y consistencia de los datos, no permitiendo a los usuarios introducir datos que harían que esta exactitud dejase de serlo.

El modelo relacional permite definir restricciones de integridad, reglas o restricciones que aplican a los datos. Una parte del trabajo del SGBD es garantizar que se cumplan las restricciones cuando se introduzcan nuevos datos en la base de datos.

#### Restricciones de dominio

Son las que aplican a los valores permitidos para un atributo de una tabla. Los valores tienen que ser del dominio definido, y este dominio se indica cuando se crea la tabla.

Además, los SGBD relacionales nos permiten agregar restricciones con la opción CHECK (verificación) que permite indicar que el valor además debe de estar dentro de un conjunto específico de valores o cumplir una condición.

#### Restricciones de clave primaria

Establece que en una relación ninguna clave primaria puede tener un valor nulo. Una clave primaria es un identificador mínimo que se usa para recuperar una tupla o fila de entre todas las filas. Si se permitiese un valor nulo significaría que ese atributo no es en realidad necesario para identificar una fila, lo que contradice la definición de clave primaria, que debe estar formada por los atributos necesarios para identificar o distinguir tuplas (o filas).

#### Restricciones de clave externa

Conocida como integridad referencial. Indica que si en una tabla o relación, se define una clave externa, el valor almacenado debe ser coincidir con la clave primaria de alguna tupla en la relación a la que apunta o si no debe ser nulo. El modelo relacional permite especificar al crear una tabla cual será o serán claves externas.

En el ejemplo anterior jefeId no puede ser E004, pues no existe. Además, se permite que E003, que no tiene jefe o aún no está en la tabla, tenga valor nulo.

#### Restricciones generales

Se pueden expresar como restricciones de tabla, que son reglas que deciden los valores permitidos dentro de una tabla y a nivel de base de datos, las aserciones, que son condiciones que deben ser verdaderas a nivel de toda la base de datos.

Cada vez que se produce un cambio en los datos, el SGBD verifica que los cambios no violan las restricciones antes de permitirles volverse permanentes.

## Vistas

En la arquitectura en tres niveles, en el nivel superior, se define como ve el usuario la estructura de la base de datos.

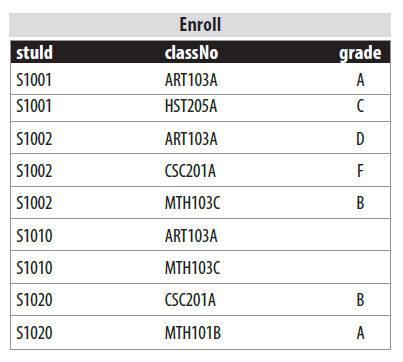
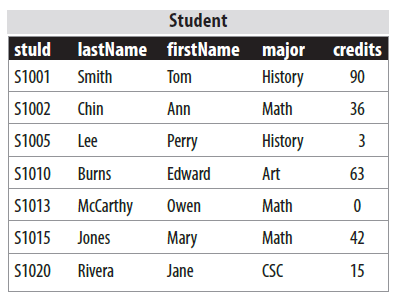
Una vista no permite eso, modificar la forma en la que un usuario va a ver la estructura de la base de datos, las tablas.

Para ello, con las vistas, se crean tablas “virtuales”. Tablas que en realidad no existen, pero que se pueden construir con las tablas existentes.

Este mecanismo es deseable pues permite ocultar a los usuarios partes de la base datos y por tanto el usuario no estará al tanto de ciertos atributos de la tabla que faltan en su vista.

También permite que los usuarios accedan a los datos de forma personalizada, de forma que los nombres de tablas y atributos les resulten familiares.

Por ejemplo, una vista podría combinar varias tablas, cambiar los nombres de los atributos y no mostrar algunos atributos (como grade, major o credits)



Matriculas

|  |  |  |  |
| --- | --- | --- | --- |
| codEstudiante | nombre | apellidos | clase |
| S1001 | Tom | Smith | ART103A |
| S1001 | Tom | Smith | HST205A |
| S1002 | Ann | Chin | ART103A |
| S1002 | Ann | Chin | CSC201A |
| S1002 | Ann | Chin | MTH103C |
| S1010 | Edward | Burn | ART103A |
| S1010 | Edward | Burn | MTH103C |
| S1020 | Jane | Rivera | CSC201A |
| S1020 | Jane | Rivera | MTH101B |

O podría ocultar información y no devolver todos los registros.

## Transformación del modelo ER al modelo relacional

En este apartado trataremos el diseño lógico de la base de datos. Partiremos del diseño conceptual expresado mediante el modelo E-R y veremos como transformarlo en estructuras del modelo relacional.

### 3.1 Transformación de entidades y relaciones del modelo ER

Las **entidades del modelo E-R** cuando se traducen al modelo relacional originan **tablas**.

Las **relaciones del modelo E-R** cuando se transforman pueden dar lugar a **claves externas** en una tabla ya existente o a una **nueva tabla**.

En el caso de las relaciones, es necesario tener en cuenta su cardinalidad (1:N, 1:1, M:N) y si son binarias o n-arias para hacer la transformación adecuada.

Las relaciones binarias 1:1 o 1:N dan lugar a claves externas.

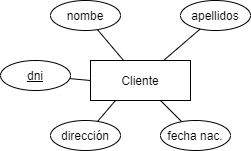
Las relaciones binarias M:N y todas las n-arias generan nuevas tablas.

#### Transformación de entidades

Todas las entidades del modelo se deben transformar en una tabla. Los atributos de la entidad serán los atributos de la tabla y la clave primaria la clave primaria de la tabla.

Ejemplo:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| nombre | apellidos | dni | dirección | fecha nac |
| Roberto | Pereda | 10870021 | C\La Lila, 121 | 16/06/1983 |
| Carlos | Álvarez | 10879665 | C\Los Fresnos,1 | 03/09/1988 |
| Rubén | Suarez | 11765133 | Avd. del Mar, 7 | 01/03/2001 |



Cliente

#### Transformación de relaciones

Una vez transformadas todas las entidades en relaciones, hay que transformar todas la relaciones en las que intervienen esas entidades.

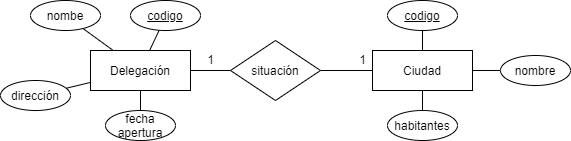
##### Binarias

Si la relación interviene en una relación del modelo E-R, con cardinalidad 1:1 o 1:N se le añaden nuevos atributos a la relación como claves externas.

##### Binarias 1:1

Sólo es necesario añadir a cualquiera de las tablas una clave externa que referencie a la otra tabla.

Ejemplo:



Las entidades crean las tablas:

DELEGACION (código, nombre, dirección, fecha-ap)

CIUDAD (código, nombre, habitantes)

Para añadir la relación hay dos opciones, añadirla a DELEGACION, con lo que la tabla quedaría

DELEGACIÓN (código, nombre, dirección, fecha-ap, código-ciudad) donde {código-ciudad} referencia CIUDAD

CIUDAD (código, nombre, habitantes)

Delegación

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| codigo | nombre | dirección | fecha-ap | codigo-ciudad |
| OF001 | Suc. Uría | Uria, 18 | 16/03/73 | OVD |
| OF002 | Suc. Coruña | Bolivia, 10 | 01/07/95 | LCG |

Ciudad

|  |  |  |
| --- | --- | --- |
| codigo | nombre | habitantes |
| OVD | Oviedo | 220k |
| LCG | La Coruña | 245K |

Dado que una delegación está situada en una sola ciudad, el atributo codigo-ciudad sólo tiene un único valor para cada valor de código de delegación.

O alternativamente

DELEGACIÓN (código, nombre, dirección, fecha-ap)

CIUDAD (código, nombre, habitantes, codigo-del) donde {codigo-del} referencia DELEGACION.

Delegación

|  |  |  |  |
| --- | --- | --- | --- |
| codigo | nombre | dirección | fecha-ap |
| OF001 | Suc. Uría | Uria, 18 | 16/03/73 |
| OF002 | Suc. Coruña | Bolivia, 10 | 01/07/95 |

Ciudad

|  |  |  |  |
| --- | --- | --- | --- |
| codigo | nombre | habitantes | codigo-del |
| OVD | Oviedo | 220k | OF001 |
| LCG | La Coruña | 245K | OF002 |

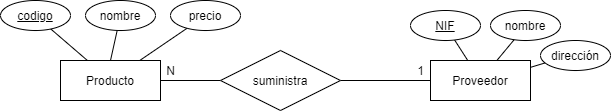
En esta transformación como ciudad sólo tiene una delegación, el atributo código-del sólo toma un valor para cada valor de código de ciudad.

En ambas transformaciones podemos saber la ciudad en la que está cada delegación y que delegación tiene cada ciudad. Es decir, se refleja el significado de la relación situación del modelo E-R.

##### Binarias 1:N

Una vez se han trasformados las entidades en tablas, solo hay que añadir a la entidad del lado N una clave externa que referencie a la otra tabla.

Ejemplo:



Producto Proveedor

|  |  |  |  |
| --- | --- | --- | --- |
| código | nombre | precio | proveedor |
| P0001 | Nike Air | 45.2 | A7998312 |
| P0002 | Puma Black | 32.9 | A7998312 |

|  |  |  |
| --- | --- | --- |
| NIF | nombre | dirección |
| A7998312 | BuyShoes | Pol. La Lila |
| B3321223 | Tradeinn | Pol. Somonte |

PRODUCTO (código, nombre, precio, proveedor) donde {proveedor] referencia PROVEEDOR

PROVEEDOR (NIF, nombre, dirección)

Esto nos permite saber para cada producto que proveedor los suministra y para cada proveedor que productos suministra.

Teniendo en cuenta que cada producto sólo tiene un único proveedor, el atributo de Producto, proveedor, tiene un valor único para cada valor de la clave primaria de Producto.

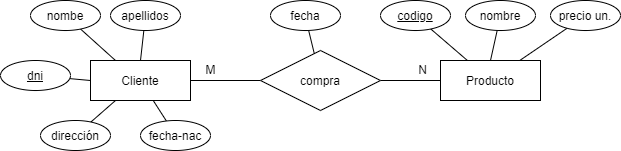
Si hubiésemos puesto la clave externa en la tabla Proveedor, la solución habría sido incorrecta porque tendría que poder tomar varios valores, todos los productos que suministra.

Si en una de las entidades es opcional en la relación (modalidad 0,1) y la transformación consiste en poner una clave externa en la tabla que corresponde a la otra entidad, entonces dicha clave externa debe poder tomar valores nulos.

##### Binarias M:N

Una relación binaria M:N se transforma en una tabla. La clave primaria está formada por las claves primarias de las entidades que relaciona. Los atributos de la relación son el resto de atributos de la tabla.

Ejemplo:



CLIENTE (dni, nombre, apellidos, dirección, fecha-nac)

PRODUCTO (código, nombre, precio\_un)

COMPRA (dni, código, fecha) donde {dni} referencia CLIENTE y {código} referencia PRODUCTO

Obsérvese que la clave de la tabla COMPRA consta tanto de la clave de CLIENTE como de la clave de PRODUCTO.

Con la solución presentada se refleja correctamente la relación compra y sus atributos. Permite saber para un cliente que productos ha comprado y en que fechas y para cada producto, que clientes los han comprado y en que fechas.

En este caso no podemos utilizar claves externas para transformar la relación porque los atributos deberían poder tomar varios valores y eso no lo permite el modelo relacional.

##### Ternarias

La transformación de las relaciones ternarias presenta similitudes importantes con la transformación de las binarias M:N.

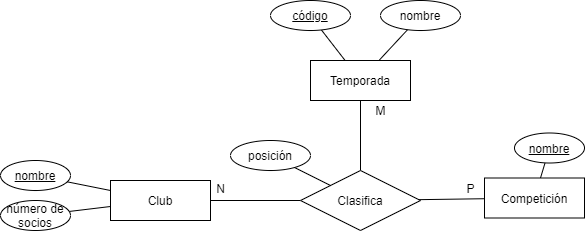
No será posible representar la relación mediante claves externas, sino que será necesario siempre crear una nueva tabla. Para que la tabla refleje toda la información necesaria para la relación, es necesario que contenga las claves de las tres entidades que relaciona y los atributos de la relación.

La clave primaria depende de la cardinalidad (también llamada conectividad) que se indique en el modelo E-R.

##### Ternarias M:N:P

La tabla que representa la relación tiene como clave primaria todos los atributos que son claves primarias de las tres entidades relacionadas.

Ejemplo:



CLUB (nombre, número\_socios)

TEMPORADA (código, nombre)

COMPETICION (nombre)

CLASIFICACION (nombre\_club, código\_temporada, nombre\_competición, posición) donde {nombre\_club} referencia CLUB, {código\_temporada} referencia TEMPORADA y {nombre\_competición} referencia COMPETICION

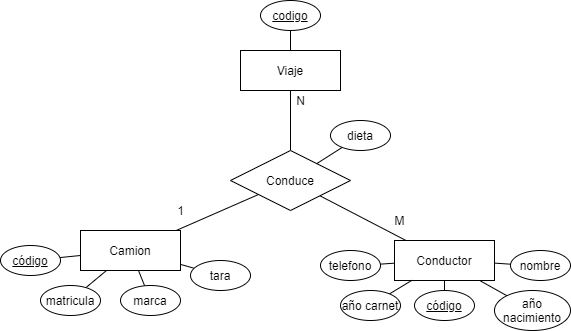
Para identificar una relación en la tabla, la clave debe de ser las de las tres entidades, porque si nos faltase una, se tendrían valores repetidos en la clave primaria y eso no está permitido en el modelo relacional.

Por ejemplo, si no incluyésemos la clave primaria de competición, no podríamos tener que Barcelona, en la temporada 2021 compitió en La Liga y además que Barcelona en la temporada 2021 compitió en La Champions pues la segunda entrada daría la clave primara Barcelona, 2021 como repetida y el modelo no lo permitiría.

##### Ternarias M:N:1

En este caso, la clave primaria de la tabla que se genera de la transformación tiene como clave primaria los atributos de las entidades de los lados etiquetados con M y N.

Ejemplo:



VIAJE (código)

CAMION (código, matricula, marca, tara)

CONDUCTOR (código, nombre, teléfono, año\_carnet, año\_nac)

CONDUCE (código\_viaje, código\_conductor, código\_camion, dieta) donde {código\_viaje} referencia VIAJE, {código\_conductor} referencia CONDUCTOR y {código\_camión} referencia CAMION

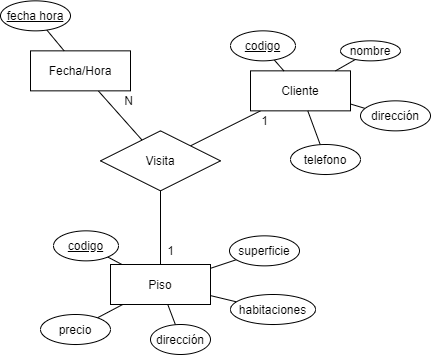
No es necesario que la clave incluya un código de camión, pues fijado un viaje y un conductor, sólo puede haber un único camión y por lo tanto no habrá claves repetidas.

##### Ternarias N:1:1

En este caso, la clave primaria de la tabla que se genera de la transformación tiene como clave primaria el atributos de la entidad del lado N y una cualquiera de las que están conectadas por 1.

Hay dos posibles claves candidatas, el diseñador debe elegir la clave primaria para la tabla.

Ejemplo:



Una opción es no incluir la clave primaria de la entidad piso:

FECHAHORA (fecha-hora)

CLIENTE (código, nombre, dirección, teléfono)

PISO (código, dirección, precio, superficie, habitaciones)

VISITA (fecha-hora, código-cliente, código-piso) donde {fecha-hora} referencia FECHAHORA, {código-cliente} referencia CLIENTE y {código-piso} referencia PISO

Aunque la clave primaria no incluya el código del piso, con una fecha y un cliente, hay un único piso que se pueda visitar en esa fecha y hora y con ese cliente.

La otra opción es no incluir el código de cliente:

FECHAHORA (fecha-hora)

CLIENTE (código, nombre, dirección, teléfono)

PISO (código, dirección, precio, superficie, habitaciones)

VISITA (fecha-hora, código-piso, código-cliente) donde {fecha-hora} referencia FECHAHORA, {código-piso} referencia PISO y {código-cliente} referencia CLIENTE

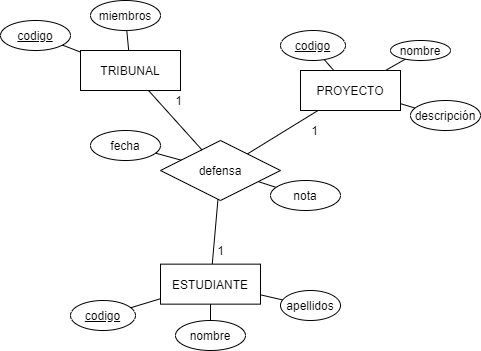
La relación se cumple pues dada una fecha y un piso, sólo un cliente puede visitarlo.

##### Ternarias 1:1:1

En este caso, la clave primaria de la tabla deberá tener como clave primaria los atributos de dos entidades cualesquiera de las tres que forman la relación.

Hay por lo tanto tres claves candidatas posibles de las que el diseñador deberá elegir una como primaria para la nueva tabla.

Ejemplo:



Hay tres alternativas para clave primaria de la relación defensa:

* Primera opción

TRIBUNAL (código, miembros)

PROYECTO (código, nombre, descripción)

ESTUDIANTE (código, nombre, apellidos)

DEFENSA (código-tribunal, código-proyecto, código-estudiante, fecha, nota) donde {código-tribunal} referencia TRIBUNAL, {código-proyecto} referencia PROYECTO y {código-estudiante} referencia ESTUDIANTE

* Segunda opción

TRIBUNAL (código, miembros)

PROYECTO (código, nombre, descripción)

ESTUDIANTE (código, nombre, apellidos)

DEFENSA (código-tribunal, código-estudiante, código-proyecto, fecha, nota) donde {código-tribunal} referencia TRIBUNAL, {código-estudiante} referencia ESTUDIANTE y {código-proyecto} referencia PROYECTO

* Tercera opción

TRIBUNAL (código, miembros)

PROYECTO (código, nombre, descripción)

ESTUDIANTE (código, nombre, apellidos)

DEFENSA (código-estudiante, código-proyecto, código-tribunal, fecha, nota) donde {código-estudiante} referencia ESTUDIANTE, {código-proyecto} referencia PROYECTO Y {código-tribunal} referencia TRIBUNAL

##### N-arias

La transformación de las relaciones n-arias consiste siempre en la creación de una nueva tabla que contiene como atributos las claves primarias de las entidades que relaciona además de los atributos de la relación.

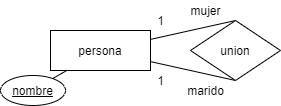
Podemos distinguir los siguientes casos:

1. Todas conectadas con muchos (M:N:P:Q por ejemplo): La clave primaria de la nueva tabla son las claves primarias de las n entidades que relaciona.
2. Si una o más entidades están conectadas con 1: La clave primaria es la de todas las entidades menos 1, siendo la que no se incluye una de las conectadas con 1.

##### Recursivas

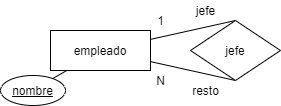
Al igual que sucede con las binarias, si el tipo de cardinalidad es 1:N o 1:1 se utiliza una clave externa, y si es N:N una nueva tabla.

Ejemplo de cardinalidad 1:1



PERSONA (nombre, nombre-cónyuge) donde {nombre-cónyuge} referencia PERSONA

Ejemplo de cardinalidad 1:N

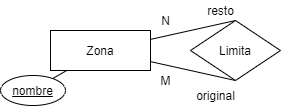


En las relaciones binarias la clave primaria de la entidad con cardinalidad 1 se incluía en la entidad con cardinalidad N como clave externa.

En este caso el atributo nuevo que se incluye es el que referencia al jefe del empleado que es que tiene cardinalidad 1.

EMPLEADO (nombre, nombre-jefe) donde {nombre-jefe} referencia EMPLEADO

Ejemplo de cardinalidad M:N



ZONA (nombre)

LIMITA (zona-original, zona-resto) donde {zona-original} referencia ZONA, y {zona-resto} referencia ZONA

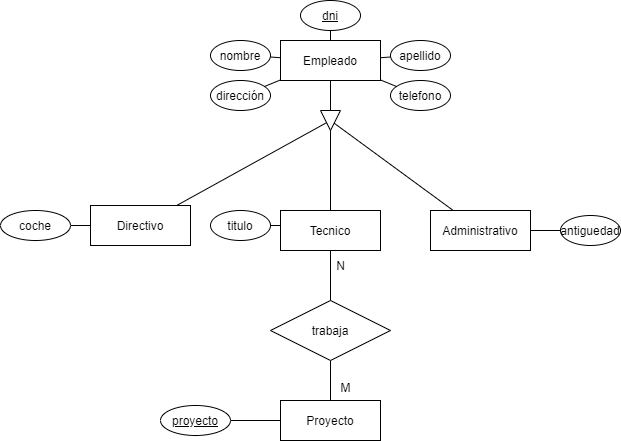
#### Especialización / Generalización

Cada una de las entidades superclase y subclase se convierten en una tabla.

La tabla para la entidad superclase tiene su clave primaria y los atributos comunes.

Las tablas de las entidades subclase tienen como clave primaria la clave de la entidad superclase y los atributos específicos de la subclase.

Ejemplo:



EMPLEADO(dni, nombre, apellido, dirección, teléfono)

DIRECTIVO(dni, coche) donde {dni} referencia EMPLEADO

ADMINISTRATIVO (dni, antigüedad) donde {dni} referencia EMPLEADO

TÉCNICO (dni, título) donde {dni} referencia EMPLEADO

PROYECTO(proyecto)

TRABAJA(dni, proyecto) donde {dni} referencia TÉCNICO y {proyecto} referencia PROYECTO