Tema 3. El modelo relacional.

1. El modelo relacional.

El objetivo principal del modelo relacional es proteger al usuario de la obligación de conocer las estructuras de datos físicas con las que se representa la información de una base de datos. Desvincular estas estructuras de datos, que son complejas por naturaleza, del diseño lógico (Modelo Relacional), permite que la base de datos se pueda implementar en cualquier gestor de bases de datos relacional (Oracle, MySQL, DB2, etc.)

Las características fundamentales del modelo relacional son:

- La relación es el elemento fundamental del modelo. Los usuarios ven la base de datos como una colección de relaciones. Estas relaciones se pueden operar mediante el Álgebra Relacional.
- El modelo relacional es independiente de la forma en que se almacenan los datos y de la forma de representarlos, por tanto, la base de datos se puede implementar en cualquier SGBD y los datos se pueden gestionar utilizando cualquier aplicación gráfica. Por ejemplo, se pueden manejar las tablas de una base de datos MySQL u Oracle con Microsoft Access.

1.1. Las relaciones en el modelo relacional.

Se define una relación como un conjunto de atributos, cada uno de los cuales pertenece a un dominio, y que posee un nombre que identifica la relación. Se representa gráficamente por una tabla con columnas (**atributos)** y filas (**tuplas**). El conjunto de tuplas de una relación representa el **cuerpo** de la relación y el conjunto de atributos y el nombre representan el **esquema**.

Definición de relación

1.2. Otros conceptos del modelo relacional

A continuación se definen los conceptos necesarios para transformar el modelo conceptual (diagrama entidad-relación) en el modelo lógico (modelo relacional).

- **Atributo**: Características que describen a una entidad o relación.
- **Dominio**: Conjunto de valores permitidos para un atributo. Por ejemplo, cadenas de caracteres, números enteros, los valores Sí o No, etc.
- **Restricciones de semántica**: Condiciones que deben cumplir los datos para su correcto almacenamiento. Hay varios tipos:

  - **Restricciones de clave**: Es el conjunto de atributos que identifican de forma única a una entidad.
  - **Restricciones de valor único** (UNIQUE): Es una restricción que impide que un atributo tenga un valor repetido. Todos los atributos clave cumplen esta restricción. No obstante es posible que algún atributo no sea clave y requiera una restricción de valor único. Por ejemplo, el número de bastidor de un vehículo no es clave (lo es la matrícula) y sin embargo, no puede haber ningún número de bastidor repetido.
  - **Restricciones de integridad referencial**: Se da cuando una tabla tiene una referencia a algún valor de otra tabla. En este caso la restricción exige que exista el valor referenciado en la otra tabla. Por ejemplo, no se puede poner una nota a un alumno que no exista.
  - **Restricciones de dominio**: Esta restricción exige que el valor que puede tomar un campo esté dentro del dominio definido. Por ejemplo, si se establece que un

campo DNI pertenece al dominio de los números de 9 dígitos 1 letra , no es posible insertar un DNI sin letra, puesto que la restricción obliga a poner al menos 1 letra.

- **Restricciones de verificación (CHECK)**: Esta restricción permite comprobar si un valor de un atributo es válido conforme a una expresión.
- **Restricción de valor NULO (NULL o NOT NULL):** Un atributo puede ser obligatorio si no admite el valor NULO o NULL, es decir, el valor falta de información o desconocimiento. Si admite como valor el valor NULL, el atributo es opcional.
- **Disparadores o triggers:** Son procedimientos que se ejecutan para hacer una tarea concreta en el momento de insertar, modificar o eliminar información de una tabla.
- **Restricciones genéricas adicionales o aserciones (ASSERT)**. Permite validar cualquiera de los atributos de una o varias tablas.

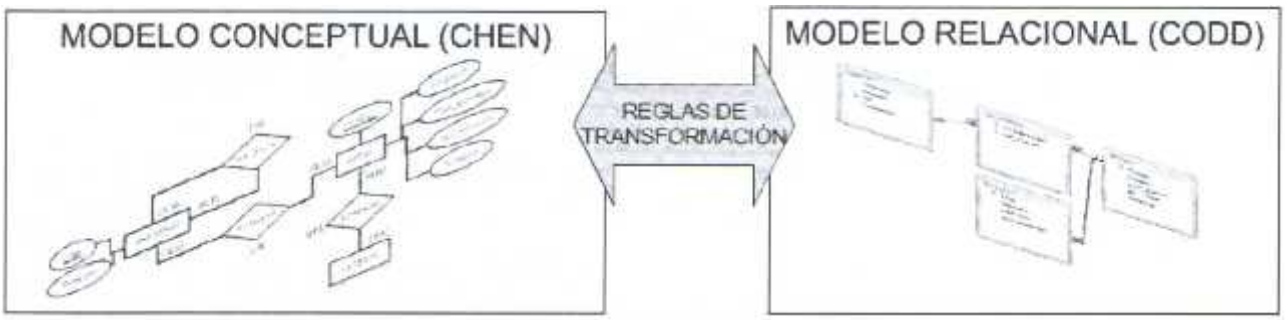
- **Clave**: Una clave es un conjunto de atributos que identifican de forma única una ocurrencia de entidad. En este caso, las claves pueden ser simples (atómicas) o compuestas. Además, hay varios tipos de clave:
  - **Superclave**: Identifican a una entidad (pueden ser no mínimas). Por ejemplo, para un empleado, las superclaves posibles son el DNI, o el DNI+Nombre o el DN+Nombre+Numero de la Seguridad Social, etc.
  - **Clave Candidata**: Es la mínima Superclave (en el caso anterior el DNI, o el Número de la seguridad social). •
  - **Clave Primaria.** Es la clave candidata elegida por el diseñador como clave definitiva (en el ejemplo anterior se elegiría el DNI por ser la más representativa para un empleado).
  - **Clave foránea**: Es un atributo de una entidad, que es clave en otra entidad. Por ejemplo, la nota en un módulo de una asignatura corresponde a un DNI, que es clave de otra entidad. En este caso el DNI es una clave foránea en la tabla notas.

1.3. modelo relacional-Operaciones

En relación con la integridad referencial (FOREING KEY), además de definir las claves ajenas, es necesario determinar las consecuencias que se producen en las operaciones de **borrado y actualización** que se realizan sobre las tuplas de una relación.

- NO ACTION (o RESTRICT): rechazar la operación de borrado o modificación.
- CASCADE: propagar la modificación (o borrado) de las tuplas de la tabla qué referencia.
- SET NULL: poner valor nulo en la clave ajena de la tabla que hace referencia.
- SET DEFAULT: poner un valor por defecto en la clave ajena de la tabla qué referencia.
- Los modos borrar y modificar son independientes, es decir, cada uno tomará una de las cuatro opciones por separado

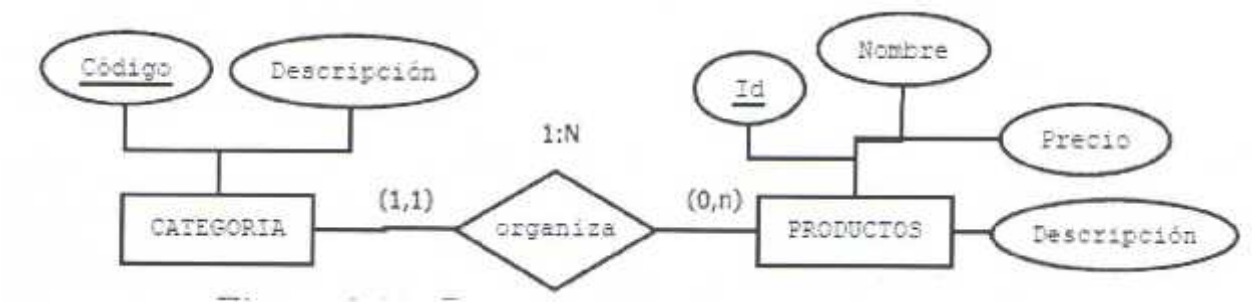
2. Transformación de un diagrama E/R al modelo relacional



Transformación del modelo E/R en relacional.

2.1. Transformación de entidades fuertes.

Para cada entidad **A, entidad fuerte,** con atributos (a_1 , a_2, , a_n) se crea una tabla A (con el nombre en plural) con **n** columnas correspondientes a los atributos de A, donde cada fila de la tabla A corresponde a una ocurrencia de la entidad A. **La clave primaria de la tabla A la forman los atributos clave de la entidad A.**



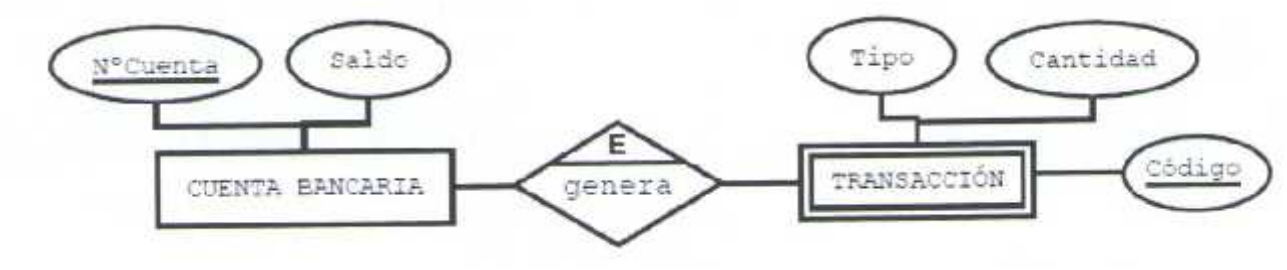
Paso a tablas de entidades fuertes.

En el diagrama E-R de la figura las tablas generadas son:

CATEGORÍAS(Código, Descripción)
PRODUCTOS (Id,Nombre,Precio,Descripción)

2.2. Transformación de entidades débiles

Para cada **entidad débil D**, con atributos $\underline{cd_1, cd_2, \ldots, cd_t}$, d_{t+i}, d_{t+2}, … d_n, donde cd_t son los atributos clave de la entidad D, y **una entidad fuerte F** de la que depende D con atributos clave ($\underline{cf_1, cf_2, \ldots cf_m}$): se crea una tabla D con m+n columnas cd_1, cd_2, , cd_n, , d_{t+1} d_{t+2},... d_n, , cf_1, cf_2 ,... , cf_n) correspondientes a los atributos de D y a los atributos clave de F. Si solo tiene dependencia de existencia, la clave primaria de la tabla D será la unión de los atributos clave de la entidad D. Si la entidad débil D, además, tiene una dependencia de identificación, la clave primaria de la tabla D será la unión de los atributos cd_1, cd_2, …., cd_t,cf_1, cf_2,...cf_m, es decir, la unión de los atributos clave de la entidad débil D y de la entidad fuerte F.



Paso a tablas de una entidad débil.

En el diagrama E-R de la figura las tablas generadas son:

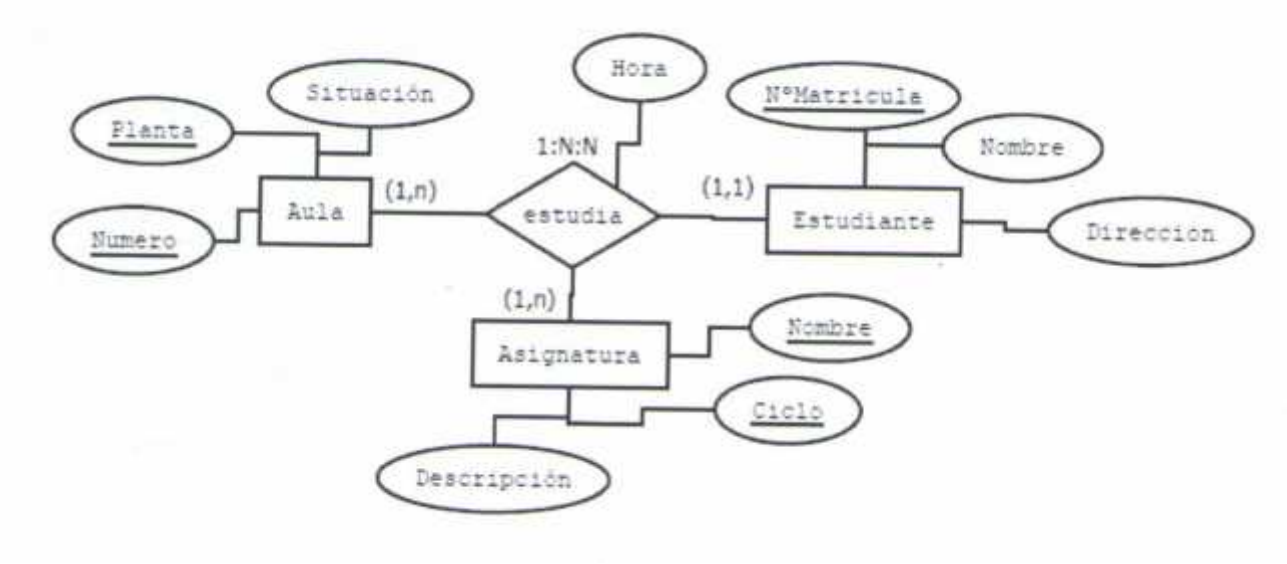
CUENTAS_BANCARIAS(N°Cuenta, saldo)
TRANSACCIONES (Código,Tipo,Cantidad,NºCuenta)

2.3 Transformación de relaciones.

Por cada relación R entre entidades E_1, E_2,... , E_N.

La clave de E_i es $C_i = a_i1,a_i2,.....a_iN$

Regla general para las relaciones: se crea una tabla con todos los campos claves de las entidades relacionadas y los atributos de la relación. La clave primaria de la tabla generada es la suma de los atributos claves de las entidades relacionadas, y cada clave incorporada a la tabla, será una clave foránea que referencia a la tabla de la que se importa.



Paso a tablas de una relación.

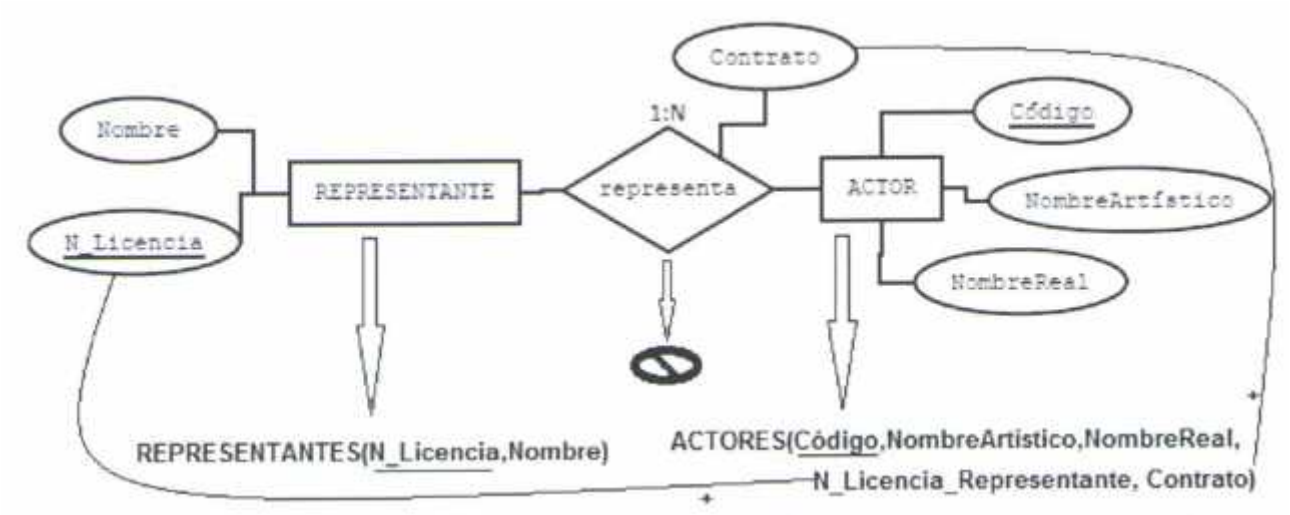
Por ejemplo, en la figura hay una relación ternaria con dos entidades con clave compuesta, **aula y asignatura**, y otra, **estudiante**, que tiene una clave simple. La transformación al modelo relacional exige la creación de una tabla para la relación. La tabla ESTUDIOS, tendrá como columnas los atributos clave del aula, los de asignatura y el atributo clave de estudiante, todos ellos formando la clave primaria y, al mismo tiempo, actuando como claves foráneas de sus respectivas tablas. Además, se incorpora el atributo de relación "hora".

AULAS(Numero, Planta, Situación)
ESTIDIANTES(Nº Matricula,Nombre,Direccion)
ASIGNATURAS(Nombre, Ciclo **,**Descripcion)
ESTUDIOS(Numero, Planta, N°Matricula, Nombre, Ciclo, Hora)

No siempre se aplica la regla general para crear una tabla por cada relación. Generalmente, se pueden encontrar las siguientes excepciones a la regla general.

2.3.1. Relaciones con cardinalidad 1:N.

En este caso, no se crea una tabla para la relación, sino que se añade a la tabla de la entidad que actúa con participación máxima N la clave de la entidad que actúa con participación máxima 1 (como clave foránea). Si además, la relación tuviera atributos se importarían también a la entidad que actúa con participación máxima N:



Excepción I: relaciones 1:N

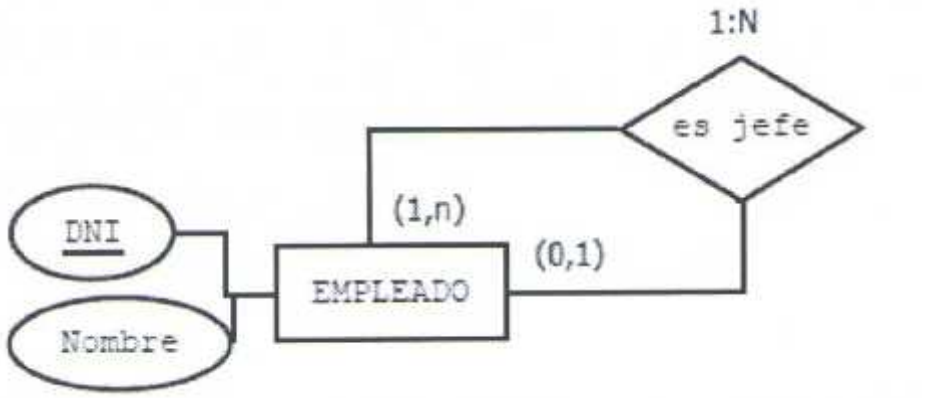
La transformación quedaría como se ilustra en la figura. Se puede observar que en este caso, no se ha creado una tabla para la relación, sino que se ha añadido a la tabla ACTORES la clave foránea **N_Licencia_Representante** que referencia al campo N_Licencia de la tabla REPRESENTANTES. También se ha añadido el campo Contrato, atributo de la relación, a la tabla ACTORES.

ACTORES(Codigo, NombreArtísitico, NombreReal, N_Licencia_Representante,Contrato)
REPRESENTANTES(N_Licencia,Nombre)

2.3.2. Relaciones con cardinalidad N:M.

2.3.3. Relaciones reflexivas con cardinalidad 1:N.

En este caso, tampoco se crea una tabla para la relación. Hay que crear una tabla con el nombre de la entidad, añadiendo otra vez la clave cambiada de nombre.



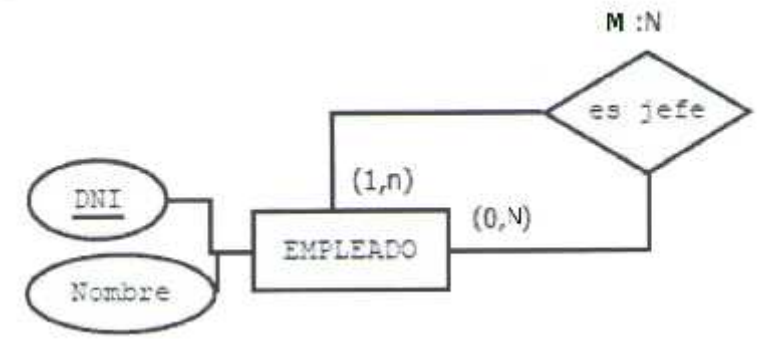
Excepción II. Relaciones reflexivas 1:N.

En el ejemplo de la figura, el empleado solo puede tener un jefe, por tanto, se incorpora el DNI del jefe del empleado (**DNISupervisor**) como clave foránea.

EMPLEADOS(DNI,Nombre,DNISupervisor)

2.3.4. Relaciones reflexivas con cardinalidad N:M

En las relaciones reflexivas con cardinalidad M:N, se aplica la regla general para la transformación de relaciones. Suponiendo un relación N:M en la relación anterior

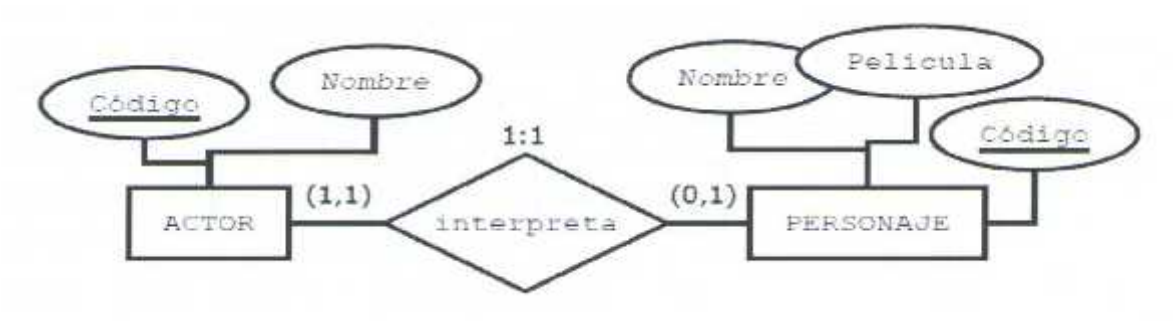


Excepción III. Relaciones reflexivas N:N.

EMPLEADOS(DNI,Nombre)
JEFES(DNISupervisor,DNIEmpleado)

2.2.5 Relaciones 1:1.

Este tipo de relaciones tampoco generan tabla. El paso a tablas se realiza de forma muy parecida a las relaciones 1-N. En este caso, tampoco se genera tabla para la relación y se tiene la libertad de poder incorporar la clave de una de las dos entidades a la otra.



Excepción III. Relaciones 1:1.

En este caso existen las siguientes opciones:

- Incorporar la clave de Personajes como clave foránea en la tabla actores:

ACTORES(Codigo, Nombre, CodigoPersonaje)
PERSONAJES(Codigo,Nombre,Película)

- Incorporar la clave de Actores como clave foránea en la tabla Personajes:

ACTORES(Codigo, Nombre)
PERSONAJES(Codigo,Nombre,Película,CodigoActor)

- Incorporar la clave de Actores como clave foránea en la tabla Personajes y la clave de Personajes a la tabla de Actores como clave foránea[1]:

ACTORES(Codigo, Nombre,CódigoPersonaje)
PERSONAJES (Codigo,Nombre,Película,CodigoActor)

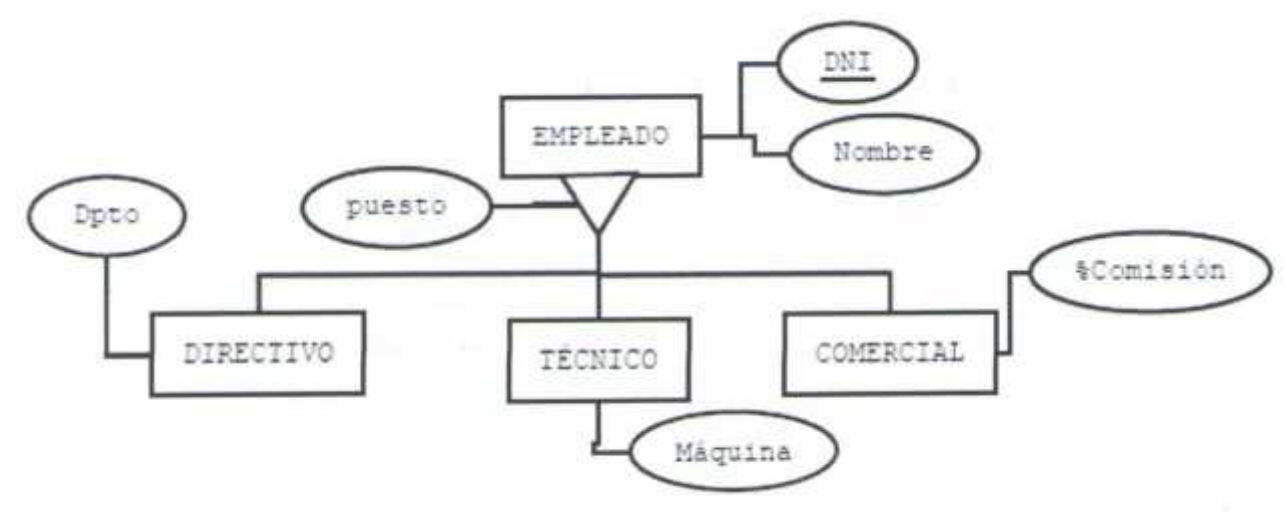
[1] *Téngase en cuenta, que en este caso se está, introduciendo una pequeña redundancia, pero que puede ser de mucha utilidad para simplificar futuras consultas.*

2.2.6. Participaciones 0,x

Normalmente las participaciones son importantes para calcular la cardinalidad de la relación, y transformar conforme a las reglas expuestas hasta ahora. Incluso en muchas ocasiones, las participaciones se omiten en los diagramas entidad-relación. No obstante, es necesario tener en cuenta cuándo la participación tiene un mínimo de 0, para adoptar un campo de una tabla como opcional NULL, u obligatorio NOT NULL.

2.2.7.Generalizaciones y especializaciones.

Para transformar las generalizaciones se puede optar por 4 opciones. Cada opción se adaptará mejor o peor a los diferentes tipos de especialización (Exclusiva, Inclusiva, Total, Parcial).



Paso a tablas de generalizaciones.

1. Se puede crear una tabla para la superclase y otras tantas para cada subclase. incorporando el campo clave de la superclase a las tablas de las subclases.

EMPLEADOS(DNI, Nombre,Puesto)
DIRECTIVOS(DNI,Dpto)
TECNICOS(DNI,Máquinas)
COMERCIALES(DNI,Comisión)

2. Se puede crear una tabla para cada subclase incorporando todos los atributos de la clase padre, y no crear una tabla para la superclase.

DIRECTIVOS(DNI,Nombre,Puesto,Dpto)
TÉCNICOS(DNI,Nombre,Puesto,Máquinas)
COMERCIALES (DNI,Nombre,Puesto,Comisión)

3. Se puede crear una sola tabla para la superclase, incorporando los atributos de todas las subclases y añadir, para distinguir el tipo de la superclase, un campo llamado "tipo", que contendrá el tipo de subclase al que representa cada tupla. Este tipo de opción se adapta muy bien a las especializaciones exclusivas.

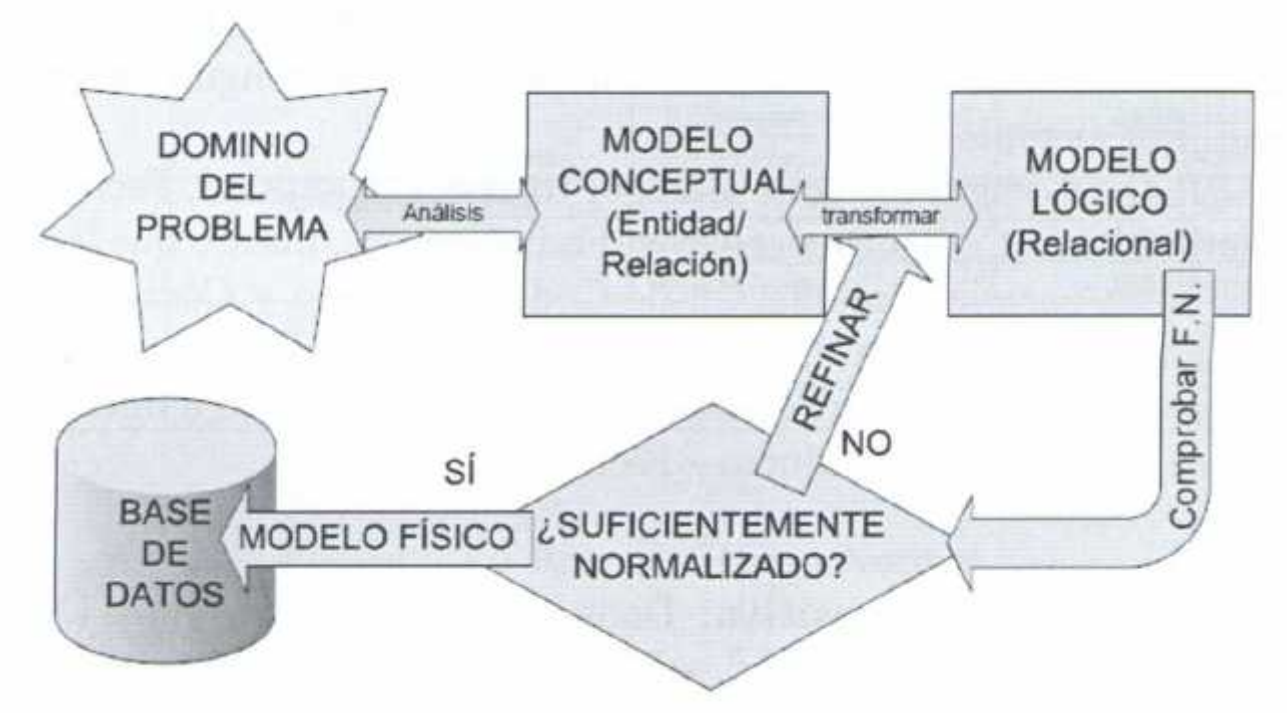
EMPLEADOS(DNI, Nombre,Puesto, Dpto, Máquinas, Comisión, Tipo)

4. Se puede crear una sola tabla para la superclase como en la opción anterior, pero en lugar de añadir un solo campo "tipo", se añaden varios campos que indiquen si cumple un perfil, de este modo se soportan las especializaciones inclusivas.

EMPLEADOS(DNI, Nombre,Puesto, Dpto, Máquinas, Comisión, EsDirectivo, EsTécnico, EsComercial)

3. Normalización

Habitualmente, el diseño de una base de datos termina en el paso del modelo entidad-relación al modelo relacional. No obstante, siempre que se diseña un sistema, no solo una base de datos, sino también cualquier tipo de solución informática, se ha de medir la calidad de la misma, y si no cumple determinados criterios de calidad, hay que realizar, de forma iterativa, sucesivos refinamientos en el diseño, para alcanzar la calidad deseada. Uno de los parámetros que mide la calidad de una base de datos es la forma normal en la que se encuentra su diseño. Esta forma normal puede alcanzarse cumpliendo ciertas restricciones que impone cada forma normal al conjunto de atributos de un diseño. El proceso de obligar a los atributos de un diseño a cumplir ciertas formas normales se llama **normalización**.



Refinamiento de un diseño de base de datos.

Las formas normales pretenden alcanzar dos objetivos:

1. Almacenar en la base de datos cada hecho solo una vez, es decir, evitar la redundancia de datos. De esta manera se reduce el espacio de almacenamiento.
2. Que los hechos distintos se almacenen en sitios distintos. Esto evita ciertas anomalías a la hora de operar con los datos.

En la medida que se alcanza una forma normal más avanzada, en mayor medida se cumplen estos objetivos. Hay definidas 6 formas normales, cada una agrupa a las anteriores, de forma que, por ejemplo, la forma normal 3 cumple la forma normal 2 y la

forma normal 1. Antes de abordar las distintas formas normales, es necesario definir los siguientes conceptos:

- **Dependencia funcional:** Se dice que un atributo Y depende funcionalmente de otro atributo X, o que X → Y, si cada valor de X tiene asociado en todo momento un único valor de Y. También se dice que X implica Y, y por tanto, que X es el implicante. Por ejemplo:

PRODUCTOS (CódigoProducto, Nombre, Precio, Descripcion)

CódigoProducto → Nombre, puesto que un código de producto solo puede tener asociado un único nombre, dicho de otro modo, a través del código de producto se localiza un único nombre.

- **Dependencia funcional completa:** Dado una combinación de atributos $X(X_1, X_{2,...})$, se dice que Y tiene dependencia funcional completa de X, o que X => Y, si depende funcionalmente de X, pero no depende de ningún subconjunto del mismo. Por ejemplo:

COMPRAS (CódigoProducto, CódigoProveedor, Cantidad, FechaCompra)

CódigoProducto,CódigoProveedor => FechaCompra, puesto que la FechaCompra es única para la combinación de CódigoProducto y CódigoProveedor (se puede hacer un pedido al día de cada producto a cada proveedor) , y sin embargo, se pueden hacer varios pedidos del mismo producto a diferentes proveedores, es decir, CódigoProducto -/->FechaCompra.

- **Dependencia funcional transitiva**: Dada la tabla T, con atributos (X,Y,Z), donde

X —› Y, Y → > Z e Y-/-> X[2], se dice que Z depende transitivamente de X. o que,

X - → Z.

Ejemplo 1:

PRODUCTOS (CódigoProducto, Nombre, Fabricante, País)

CódigoProducto → Fabricante

Fabricante → País

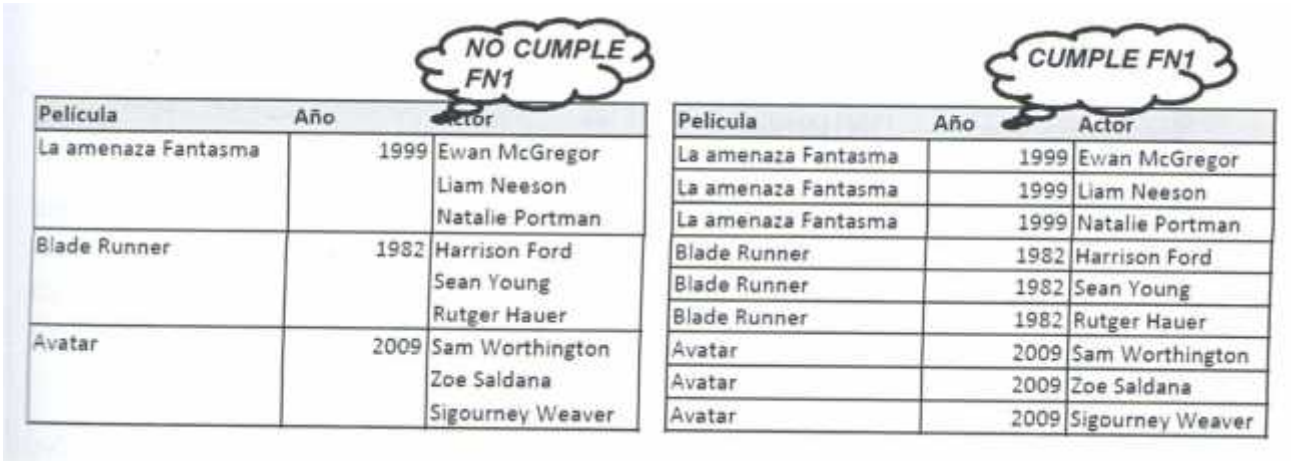
CódigoProducto - → País, es decir, País depende transitivamente de CodigoProducto.

[2]Y no depende de X

A continuación, se describe de forma intuitiva las siguientes formas normales:

3.1. Primero forma normal (FN1).

En esta forma normal **se prohíbe** que en una tabla haya atributos **que puedan tomar más un valor**. Esta forma normal es inherente al modelo relacional, puesto que las tablas gestionadas por un SGBD relacional, están construidas de esta forma



Película	Año	Actor
La amenaza Fantasma	1999	Ewan McGregor
		Liam Neeson
		Natalie Portman
Blade Runner	1982	Harrison Ford
		Sean Young
		Rutger Hauer
Avatar	2009	Sam Worthington
		Zoe Saldana
		Sigourney Weaver

Película	Año	Actor
La amenaza Fantasma	1999	Ewan McGregor
La amenaza Fantasma	1999	Liam Neeson
La amenaza Fantasma	1999	Natalie Portman
Blade Runner	1982	Harrison Ford
Blade Runner	1982	Sean Young
Blade Runner	1982	Rutger Hauer
Avatar	2009	Sam Worthington
Avatar	2009	Zoe Saldana
Avatar	2009	Sigourney Weaver

Primera forma normal.

Por ejemplo, en la figura se puede ver la diferencia entre una tabla que cumple la FN1 y otra que no, actor solo puede tener un valor para cada fila.

3.2. Segunda forma normal (FN2).

Un diseño se encuentra en FN2 si está en FN1 y además, **cada atributo que no forma parte de la clave tiene dependencia completa de la clave principal.**

Ejemplo:

COMPRAS (CódigoProducto, CódigoProveedor, NombreProducto, Cantidad, FechaCompra).

CódigoProducto —› NombreProducto, por tanto, al no ser dependencia funcional completa, no está en FN2.

El NombreProducto se puede obtener de la tabla

PRODUCTOS (CódigoProducto, Nombre, Fabricante, País) por tanto no es necesario en la tabla compras

COMPRAS (CódigoProducto, CódigoProveedor, Cantidad, FechaCompra).

3.3. Tercera forma normal(FN3).

Un diseño se encuentra en FN3 si está en FN2 y además, no hay ningún atributo no clave que depende de forma transitiva de la clave, es decir, todos los atributos que no forman parte de la clave primaria son independientes entre si (no dan información sobre otros atributos de la relación)

Ejemplo:

PRODUCTOS (CódigoProducto, Nombre, Fabricante, País).

CódigoProducto → Fabricante

Fabricante → País

CódigoProducto - -/-> País

País depende transitivamente de CódigoProducto, por tanto, no está en tercera forma normal.

Para que esten en 3FN

PRODUCTOS (CódigoProducto, Nombre, codigoFabricante).

FABRICANTES (codigoFabricante, Fabricante, pais).

Ejemplo :

EMPLEADOS(CodigoEmpleado, Nombre,Apellidos,Dirección, CodigoDeparpatmemto, NombreDepartamento)

No esta en 3FN porque

CodigoEmpleado → CodigoDepartamento

CodigoDepartameto → NombreDepartamento

CodigoDepartamento - -/-> NombreDepartamento

Para que la tabla este en 3FN creamos la tambla DEPARTAMENTOS

EMPLEADOS(CodigoEmpleado, Nombre,Apellidos,Dirección, CodigoDeparpatmento)

DEPARTAMENTOS(CodigoDeparpatmento, NombreDepartamento)

3.4. Forma Normal de Boyce-Codd (FNBC)

Esta forma normal exige que el modelo esté en FN3, y que además, todo implicante de la tabla, sea una clave candidata.

Ejemplo:

NOTAS (DNIAlumno, DNIProfesor,NombreProfesor, Nota).

DNIProfesor —> NombreProfesor

NombreProfesor → DNIProfesor

DNIProfesor,DNIAlumno —› Nota

En este caso, la tabla está en 3FN porque no hay dependencias funcionales transitivas, y sin embargo no está en FNBC porque NombreProfesor y DNIProfesor son implicantes, y no son claves candidatas. Para obtener la tabla en FNBC habría que quitar de la tabla el atributos NombreProfesor.

NOTAS (DNIAlumno,DNIProfesor Nota).

PROFESOR (DNIProfesor, nombreProfesor)

3.5.Otras formas normales.

Existen más formas normales (FN4, FN5, FNDK, FN6 cuyo alcance excede el de este libro y cuya aplicación en el mundo real es únicamente teórica). Las formas normales 4 y 5, se ocupan de las dependencias entre atributos multivaluados, la Forma Normal Dominio Clave (FNDK) trata las restricciones y los dominios de los atributos, y finalmente la FN6 trata ciertas consideraciones con las bases de datos temporales.

Bibliografía.

- LOPEZ MONTALBÁN, I. DE CASTRO VÁZQUEZ,M. OSPINO RIVAS, J. (2014). Bases de Datos. Madrid:Garceta.
- Antonio Postigo Palacios. Bases de datos .Paraninfo.