1º CFGS Desarrollo De Aplicaciones Multiplataforma

UD 3. Diseño lógico de Bases de Datos. El modelo relacional.

Módulo: Bases de Datos

Centro de Enseñanza
Gregorio Fernández

Macarena Cuenca Carbajo

1. Introducción

- **Diseño lógico**: 2ª etapa del diseño de una base de datos.
- Transformación de modelos.
- Finalmente se aplicará la teoría de normalización.
- El modelo lógico más popular es el **modelo relacional**.

2. Modelo relacional

- **Edgar Frank Codd** definió las bases del modelo relacional a finales de los 60.
- Codd pretendía que los usuarios trabajaran de forma sencilla e independiente del funcionamiento físico de la base de datos.
- Se apoya en los trabajos de los matemáticos Cantor y Childs (cuya *teoría de conjuntos* es la verdadera base del modelo relacional).
- Según Codd los datos se agrupan en **relaciones** (actualmente llamadas **tablas**).
- Oracle es una de las empresas que empieza a implementar sus teorías. Hoy en día casi todas las bases de datos siguen este modelo.

2. Modelo relacional

- Objetivos:
  - **Independencia física**. La forma de almacenar los datos, no debe influir en su manipulación lógica.
  - **Independencia lógica**. Las aplicaciones que utilizan la base de datos no deben ser modificadas porque se modifiquen elementos de la base de datos.
  - **Flexibilidad**. La base de datos debe ofrecer fácilmente distintas vistas en función de los usuarios y aplicaciones.
  - **Uniformidad**. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas).
  - **Sencillez**. Facilidad manejo.

3. Elementos Modelo Relacional: Relación

- Se representa mediante una **tabla**. Representa una entidad.
- Una relación está formada por un conjunto de atributos llamados **columnas** y un conjunto de tuplas llamadas **filas**.
- **Columnas**. Las columnas tienen un nombre y pueden guardar un conjunto de valores. Una columna se identifica por su nombre. El orden de las columnas en una tabla es irrelevante.
- **Filas**. Se corresponden con la idea clásica de registros. Su orden es irrelevante. Requisitos:
  - Cada fila se debe corresponder con un elemento del mundo real.
  - No puede haber dos filas iguales (con todos los valores iguales).

3. Elementos Modelo Relacional: Relación

- De las tablas se derivan los siguientes conceptos:
  - **Cardinalidad**. Es el número de filas de la tabla.
  - **Grado**. Es el número de columnas de la tabla.
  - **Valor**. Viene representado por la intersección entre una fila y una columna.
  - **Valor Null**. Representa la ausencia de información.

3.1. Elementos Modelo Relacional: Dominio

- Posibles valores que puede tomar un atributo.
- Dos atributos distintos pueden tener el mismo dominio.
- A los dominios se les asigna un nombre y así podemos referirnos a ese nombre en más de un atributo.
- Existen dos tipos de dominios:
  - **Generales**. Son aquellos cuyos valores están comprendidos entre un máximo y un mínimo. Un ejemplo puede ser el *Salario*, que comprende todos los números enteros positivos de cuatro cifras.
  - **Restringidos**. Son los que pertenecen a un conjunto de valores específico. Por ejemplo, *Sexo*, que puede tomar los valores H o M.

3.2. Elementos Modelo Relacional: Claves

- Toda fila de una tabla debe tener una clave que permita identificarla. La clave debe cumplir dos requisitos:
  - **Identificación univoca**. El valor de la clave ha de identificar a cada fila de forma única.
  - **No redundancia**. El valor de la clave no se puede repetir.
- En el modelo relacional existen varios modelos de claves: candidatas, primarias y ajenas.

3.2. Elementos Modelo Relacional: Claves

- **Clave candidata**. Puede existir más de un conjunto de atributos que cumpla los requisitos anteriores. Este conjunto se conoce como **claves candidatas**. Uno de esos conjuntos será elegido como clave primaria.
- **Clave primaria (primary key)**. Es la columna o conjunto de columnas que permiten identificar cada fila de la tabla. Es la clave candidata que el usuario ha elegido para identificar las filas de la tabla. No puede tener valores nulos.
- **Clave ajena o secundaria (foreign key)**. Está formada por una o más columnas de una tabla cuyos valores corresponden con los de la clave primaria de otra tabla. Las claves ajenas representan las **relaciones** entre tablas.
  - El modelo relacional especifica la **regla de integridad referencial** para las claves ajenas, que establece que los valores de la clave ajena son nulos o han de coincidir con los valores de la clave primaria de otra tabla.

3.2. Elementos Modelo Relacional: Claves

- Ejemplo:

DEPARTAMENTO		
NUMDEPT	**NOMDEPT**	**PRESUPUESTO**
D1	Marketing	1000
D2	Desarrollo	1200
D3	Investigación	5000

EMPLEADO			
NUMEMP	**APELLIDO**	**NUMDEPT**	**SALARIO**
E1	López	D1	1500
E2	Fernández	D2	1600
E3	Martínez	D3	1800
E4	Sánchez	D2	2000

3.3. Elementos Modelo Relacional: Nulos

- En los lenguajes de programación se utiliza el valor nulo para reflejar que una variable, un objeto,.. no tiene ningún contenido.
- Las bases de datos relacionales también permiten utilizar el valor nulo. Su significado no cambia: valor vacío.
- En las claves secundarias indica que la fila actual no está relacionada con ninguna.
- En otros atributos indica carencia de valor.
- Es importante indicar que el texto vacío (“”) no significa lo mismo que el nulo; como tampoco el valor cero significa nulo.
- En todas las bases relacionales se utiliza el operador llamado **es nulo** (*is null*) que devuelve verdadero si el valor con el que se compara es nulo y falso en caso contrario.

3.4. Elementos modelo E/R: Vistas

- Una **vista** es una tabla ficticia cuyas filas se obtienen a partir de una o varias tablas.
- Lo que se almacena de una vista es su definición.

4. Restricciones

- Condiciones de obligado cumplimiento por las filas de la base de datos.
- Con ellas se intenta introducir capacidad semántica al modelo, con el fin de que refleje mejor la realidad.
- Tipos:
- **Inherentes**: son definidas por el hecho de que la base de datos sea relacional.
  - No puede haber dos filas iguales.
  - El orden de las filas no es significativo.
  - El orden de las columnas no es significativo.
  - Cada columna sólo puede tomar un valor en el dominio en el que está inscrita.
- **Semánticas**: definidas por los usuarios.

4.1. Restricciones: Primary Key

- Restricción de **clave principal o primaria.**
- Marca una o varias columnas como identificadores de la tabla.
- De esta forma las filas de la tabla no podrán repetir valores en esas columnas ni tampoco dejarlos vacías.

4.2. Restricciones: Unique

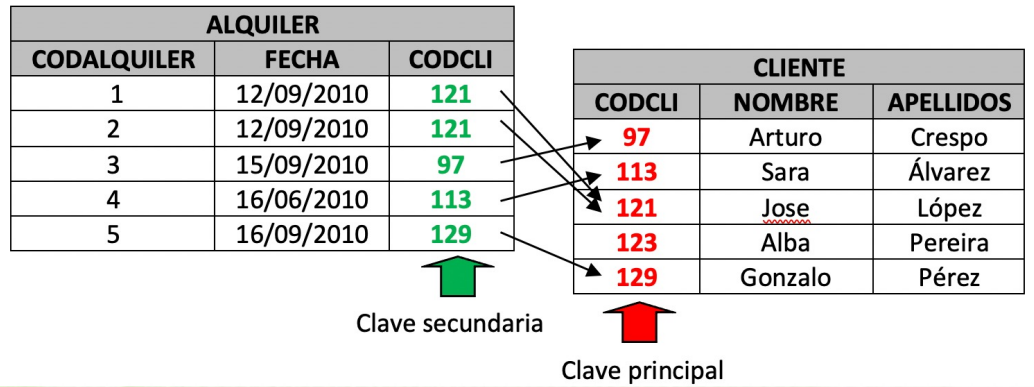
- Impide que los valores de las columnas marcadas de esta forma, puedan repetirse.
- Esta restricción debe indicarse en todas las claves alternativas.
- Al marcar una clave primaria, esta restricción se añade automáticamente sobre las columnas que forman la clave.

4.3. Restricciones: Not Null

- Prohíbe que la columna marcado de esta forma quede vacío (es decir, impide que pueda contener el valor null).
- Esta restricción se añade automáticamente sobre las columnas que forman la clave primaria.

4.4. Restricciones: Foreign Key

- Sirve para crear una **clave externa, secundario o foránea.**
- Puede estar formada por uno o más columnas.
- Las columnas marcadas de esta forma sólo podrán contener valores que estén relacionados con la clave principal de la tabla que relacionan (llamada **tabla principal**) o valores nulos.
- Ejemplo:



4.4. Restricciones: Foreign Key

- Puede haber problemas en las operaciones de borrado y modificación de registros.
- Para solventarlos se puede:
  a) Prohibir la operación (*no action*).
  b) Transmitir la operación en cascada (*cascade*). Es decir, si se modifica o borra un cliente, también se modificarán o borrarán los alquileres relacionados con él.
  c) Colocar nulos (*set null*) Las referencias al cliente en la tabla de alquileres se colocan como nulos (es decir, alquileres sin cliente).
  d) Usar el valor por defecto (*default*). Se coloca un valor por defecto en las claves externas relacionadas. Este valor se indica al crear la tabla.

4.5. Restricciones: Check y Triggers

- Con las **Check** podemos poner condiciones lógicas que debe de cumplir un dato concreto para darlo por válido. Por ejemplo, que la fecha de alquiler sea menor o igual que la fecha actual o que la edad de un cliente sea mayor de 18 años.
- Los **Triggers** son pequeños programas almacenados en la base de datos que se ejecutan automáticamente cuando se cumple una determinada condición. Sirven para realizar una serie de acciones cuando ocurre un determinado evento (cuando se añade una fila, cuando se borra un dato, cuando un usuario abre una conexión…).

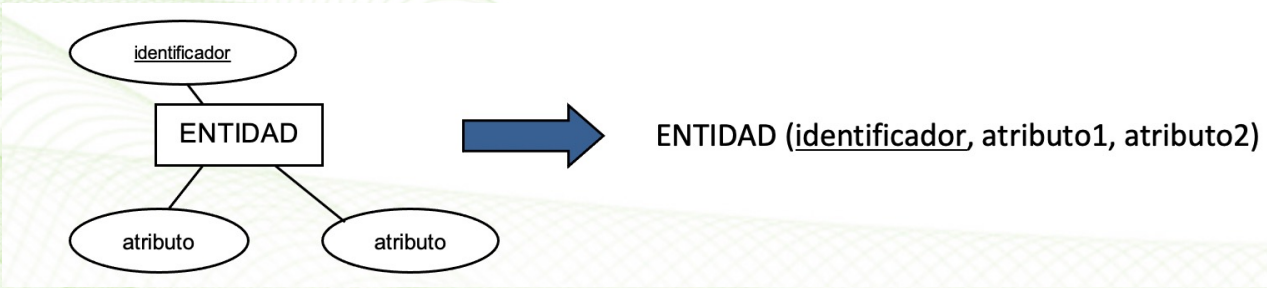
  Permiten especificar restricciones muy potentes.

5. Transformación del Modelo E/R al Modelo Relacional

- Entidades fuertes.
- Relaciones.
- Entidades débiles.
- ISA.

5.1. Transformación de entidades fuertes

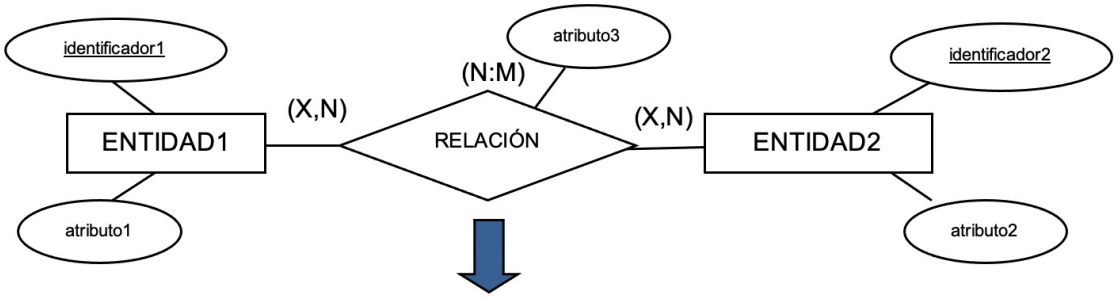
- **Entidades**. Pasan a ser tablas.
- **Identificador/es principal/es**. Pasa/n a ser clave/s primaria/s.
- **Identificadores alternativos**. Pasan a ser claves candidatas (columnas UNIQUE). Si se desea que no tomen valores nulos habrá que indicarlo.
- **Atributos**. El resto de atributos pasan a ser columnas de la tabla.



5.2. Transformación de relaciones

Relación N:M

- Se transforma en una tabla cuyos atributos son los atributos de la relación y las claves de las entidades relacionadas (que pasarán a ser claves externas). La clave de la tabla la forman todas las claves externas.

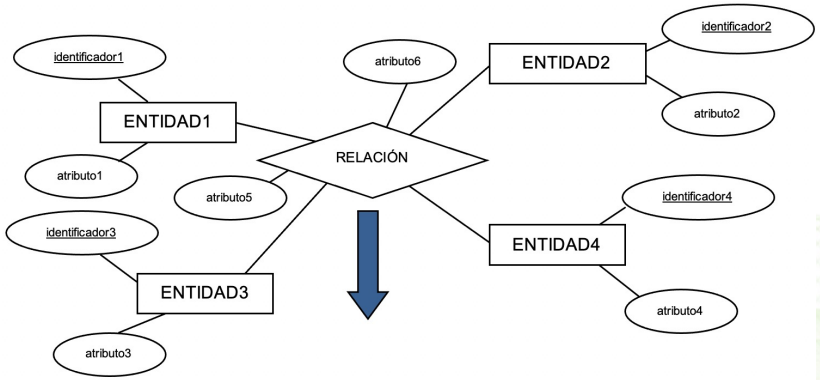


ENTIDAD1 (identificador1, atributo1)
ENTIDAD2(identificador2, atributo2)
RELACIÓN(identificador1, identificador2, atributo3)

5.2. Transformación de relaciones

Relación de orden N

- Las relaciones ternarias, cuaternarias y N-arias en general se transforman en una tabla que contiene los atributos de la relación más las claves de las entidades relacionadas. La clave la forman todas las claves externas.

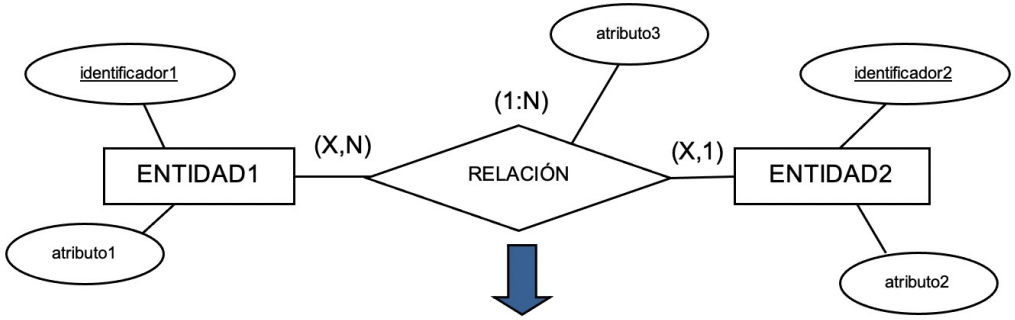


ENTIDAD1 (identificador1, atributo1)
ENTIDAD2(identificador2, atributo2)
ENTIDAD3(identificador3, atributo3)
ENTIDAD4(identificador4, atributo4)
RELACIÓN(identificador1, identificador2, identificador3, identificador4, atributo5, atributo6)

5.2. Transformación de relaciones

Relación 1:N

- Las relaciones de este tipo, no requieren ser transformadas en una tabla en el modelo relacional. En su lugar la tabla del lado **varios** (tabla relacionada) incluye como clave externa, la clave principal de la entidad del lado **uno** (tabla principal). Es lo que se conoce como **propagación de clave**.



ENTIDAD1 (identificador1, atributo1, **identificador2, atributo3**)
ENTIDAD2(identificador2, atributo2)

5.2. Transformación de relaciones

Relación 1:1

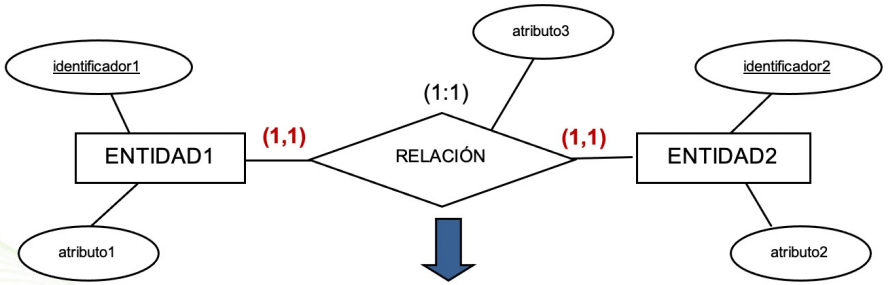
- Relaciones entre dos **entidades con todas las cardinalidades a 1.**
- Hay dos posibilidades:

  a) Propagar la clave en cualquier dirección. Sería conveniente tener en cuenta accesos más frecuentes. Hay que tener en cuenta que dicha clave será clave alternativa además de ser clave secundaria (UNIQUE, NOT NULL).

  b) Generar una única tabla con todos los atributos de ambas entidades colocando como clave principal cualquiera de las claves de las dos entidades. La otra clave será marcada como clave alternativa (UNIQUE, NOT NULL). El nombre de la tabla sería el de la entidad más importante desde el punto de vista conceptual.

5.2. Transformación de relaciones

Relación 1:1



Solución1 (ENTIDAD2 -> ENTIDAD1):
ENTIDAD1 (identificador1, atributo1, **identificador2, atributo3**)
ENTIDAD2 (identificador2, atributo2)

Solución2 (ENTIDAD2):
ENTIDAD2 (identificador2, atributo2, atributo3, identificador1, atrituto1)

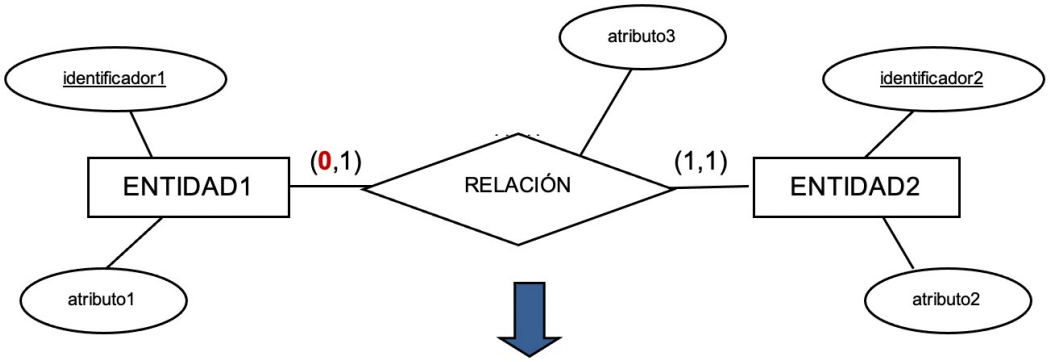
5.2. Transformación de relaciones

Relación 0:1

- Relaciobnes entre dos entidades con cardinalidad máxima de 1 en ambas direcciones, pero ***en una de ellas la cardinalidad mínima es 0***.
- En este caso la solución difiere respecto al anterior caso.
- No conviene generar una única tabla ya que habría numerosos valores nulos en la tabla (debido a que hay ejemplares que no se relacionan en las dos tablas).
- La solución sería generar dos tablas, una para cada entidad, y propagar la clave de la tabla con cardinalidad (1,1) a la tabla con cardinalidad (0,1). Dicha clave sería clave alternativa (UNIQUE, NOT NULL) de esa tabla.

5.2. Transformación de relaciones

Relación 0:1



ENTIDAD1 (identificador1, atributo1, **identificador2, atributo3**)
ENTIDAD2 (identificador2, atributo2)

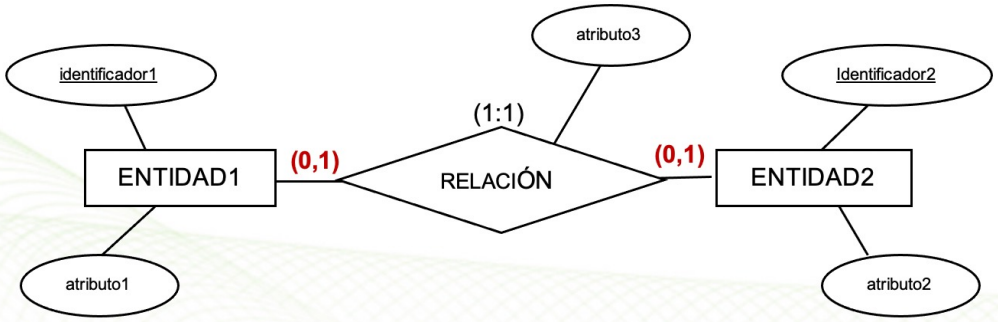
5.2. Transformación de relaciones

Relación 0:0

- En el caso de que en ambos extremos nos encontremos con relaciones (0,1), entonces la mejor solución será crear una tabla, cuyos atributos son los atributos de la relación y las claves de las entidades relacionadas (que pasarán a ser claves externas).
- La clave de la tabla será sólo una de las claves externas. La otra clave externa será clave alternativa (UNIQUE, NOT NULL).

5.2. Transformación de relaciones

Relación 0:0

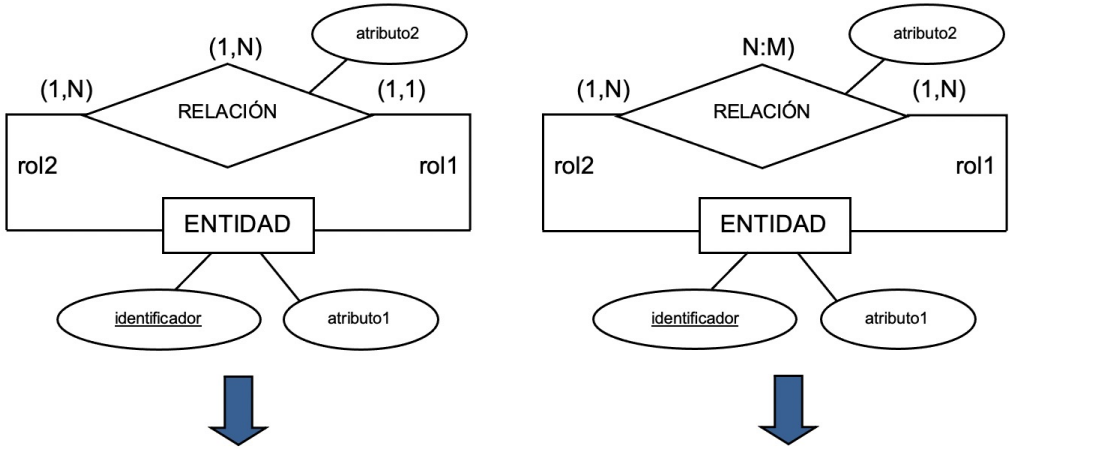


ENTIDAD1 (identificador1, atributo1)
ENTIDAD2 (identificador2, atributo2)
RELACIÓN (identificador1, identificador2, atributo3)

5.2. Transformación de relaciones

Relación reflexiva

- Se trata de la misma forma que una relación “normal”, sólo que hay que imaginar que la tabla se divide en dos, una por cada rol.

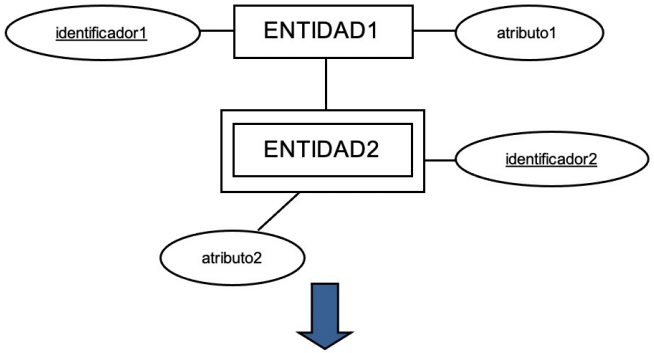


ENTIDAD (identificador, atributo1, identificador_ rol1, atributo2)

ENTIDAD (identificador, atributo1)
RELACIÓN (identificador_ rol1, identificador_ rol2, atributo2)

5.3. Transformación de entidades débiles

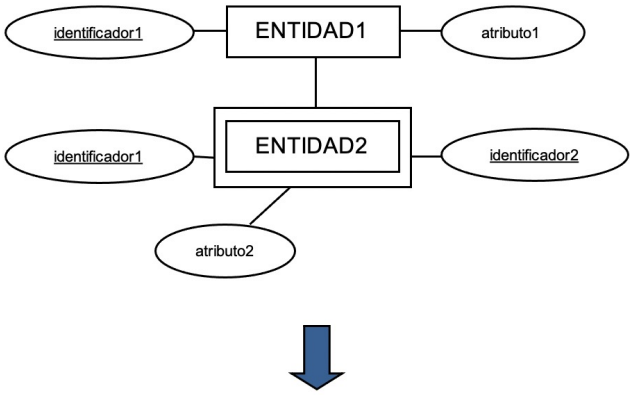
- Toda entidad débil incorpora una relación implícita con una entidad fuerte.
- Esta relación al ser (1:N) no necesita incorporarse como tabla en el modelo relacional, bastará con que se propague la clave de la entidad fuerte a la entidad débil, pasando a ser ésta clave externa en la tabla de la entidad débil.



ENTIDAD1 (identificador1, atributo1)
ENTIDAD2 (identificador2, atributo2, identificador1)

5.3. Transformación de entidades débiles

- En ocasiones, el identificador de la entidad débil tiene como parte de su identificador al identificador de la entidad fuerte (**dependencia en identificación**). En esos casos no hace falta añadir de nuevo como clave externa el identificador de la entidad fuerte.



ENTIDAD1 (identificador1, atributo1)
ENTIDAD2 (identificador1, identificador2, atributo2)

5.4. Transformación de ISA

- Normas:
  - Tanto las superentidades como las subentidades generarán tablas en el modelo relacional.
    - **NOTA**: En realidad, la superentidad generaría tabla únicamente si la ISA es parcial, ya que podría ocurrir que un ejemplar de dicha entidad no fuera de ninguno de los tipos de las subentidades. O dicho de otra forma, en el caso de que la ISA sea de tipo total, se podría incluso no hacer la superentidad y pasar todos sus atributos a las subentidades.
  - Los atributos se colocan en la tabla correspondiente.
  - Si cada entidad tiene clave propia (GENERALIZACIÓN), el identificador de la superentidad se coloca como clave foránea en cada subentidad. Además será clave alternativa.
  - En el caso de que todas las entidades tengan la misma clave (ESPECIALIZACIÓN), todas ellas tendrán la misma columna como identificador.
  - Si la ISA es exclusiva o no, no cambia el esquema relacional, pero sí habrá que tenerlo en cuenta para las restricciones futuras en el esquema interno (casi siempre se realizan mediante *triggers*), ya que en las exclusivas no puede haber repetición de la clave de la superentidad en ninguna subentidad.

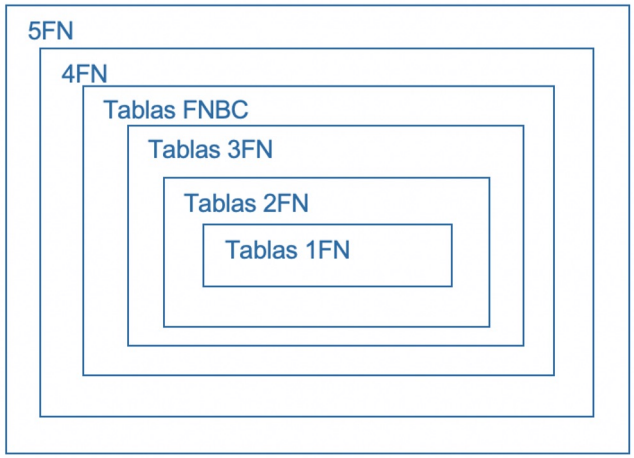
6. Normalización

- Una vez obtenido el esquema relacional, normalmente tendremos una “buena” base de datos.
- Pero otras veces, debido a fallos en el diseño o a problemas indetectables, tendremos un esquema relacional con **redundancias** y **dependencias incoherentes**.
- La **normalización** en el diseño lógico de una base de datos, implica realizar una serie de procesos formales que separan los datos en múltiples tablas relacionadas.
- El resultado de cada uno de estos procesos es conocido como **forma normal**.
- Los diseños normalizados, en principio son los “mejores” diseños. Pero puede ocurrir, si el diseño está demasiado normalizado, que lleguemos a tener un gran número de tablas, haciendo muy complejas las consultas y consumiendo bastantes recursos para su ejecución, obteniendo una respuesta lenta.

6.1. Formas normales

- En la normalización, se pretende obtener distintas **formas normales**.
- Se dice que una tabla está en una determinada forma normal si satisface una serie de restricciones.
- Las formas normales son:
  - Primera forma normal (1FN), segunda forma normal (2FN) y tercera forma normal (3FN), definidas por Codd.
  - La forma normal de Boyce-Codd (FNBC) y
  - La cuarta y quinta forma normal (4FN y 5FN), definidas por Fagin.
- La teoría de formas normales es una teoría matemática.

6.1. Formas normales



- Una forma normal incluye a las de nivel inferior, es decir, una tabla que esté en 2FN estará en 1FN, una que esté en 3FN estará en 1FN y 2FN, como se muestra en la figura.
- Muchos diseñadores opinan que es suficiente con llegar a la 3FN. Otros opinan que hay bases de datos “peores” en quinta forma normal que en tercera.

6.2. Dependencias funcionales

- Lo primero que hay que hacer es **identificar las dependencias** de cada atributo de las entidades.

Dependencia funcional

- Un conjunto de atributos (**Y**) depende funcionalmente de otro conjunto de atributos (**X**) si cada valor de **X** tiene asociado un único valor de **Y**.
  Simbólicamente se denota por **X → Y**.
- Por ejemplo: *EMPLEADO (DNI, nombre, salario, dirección)*

  Se puede asegurar que *DNI → nombre*, es decir, el nombre de un empleado depende funcionalmente del DNI, ya que para un DNI concreto sólo hay un nombre posible.

  De la misma forma, *salario ↛ nombre*, ya que para un mismo salario pueden existir varios nombres, es decir, varios empleados pueden cobrar los mismo.
- Al conjunto **X** del que depende funcionalmente el conjunto **Y** se le llama **determinante**.
  Al conjunto **Y** se le llama **implicado.**

6.2. Dependencias funcionales

Atributos equivalentes

- Dos conjuntos de atributos **X** e **Y**, son equivalentes funcionalmente, si se cumple que **X → Y** e **Y → X .** Simbólicamente se denota por **X ↔ Y**.

Dependencia funcional completa

- Caso particular de dependencia funcional.
- Un conjunto de atributos (**Y**) tiene una dependencia funcional completa sobre otro conjunto de atributos (**X**) si **Y** tiene dependencia funcional de **X** pero no depende de ningún subconjunto del mismo**.** Es decir, no se puede obtener de **X** un conjunto de atributos más pequeño que consiga una dependencia funcional de **Y.**
- Una dependencia funcional completa se denota por **X ⇒ Y.**
- Por ejemplo: *LIBRO* (*ISBN, cod_socio, fecha_préstamo*)

  ISBN, cod_socio → fecha_prestamo

  ISBN −/→ fecha_prestamo

  cod_socio −/→ fecha_prestamo

Para un libro y un socio sólo existe una fecha de préstamo, mientras que el mismo libro se puede prestar varios días y un mismo socio puede tomar prestado uno o varios libros más de un día. Por todo ello:

ISBN, cod_socio ⇒ fecha_prestamo)

6.2. Dependencias funcionales

Dependencia funcional transitiva

- Otro caso particular de dependencia funcional. Se produce cuando tenemos tres conjuntos de atributos **X, Y** y **Z** y se cumple lo siguiente:

  X → Y **Y** depende funcionalmente de **X**

  Y → Z **Z** depende funcionalmente de **Y**

  Y –/→ X **X** no depende funcionalmente de **Y**

- Entonces **Z** tiene una dependencia transitiva respecto de **X** a través de **Y**. Se denota de la forma **X – → Z**.
- Por ejemplo: *EMPLEADO (nombre, dirección, código_postal):*

  nombre → dirección

  dirección → código_postal

  dirección –/→ nombre

- Cada empleado sólo tiene una dirección y cada dirección sólo un código postal, mientras que varios empleados pueden vivir en la misma dirección.

  Por lo tanto código_postal depende transitivamente de nombre: *nombre – → código_postal.*

6.3. Primera Forma Normal (1FN)

- ***Una tabla está en primera forma normal si cada columna es atómica, es decir, no se puede descomponer, además tienen un solo valor en cada fila, y no hay grupos repetitivos, es decir, columnas con datos similares.***
- En esta etapa debemos asegurarnos de que todos nuestros campos son indivisibles y eliminar todos los datos que sean repetidos o que tengan una dependencia funcional, como por ejemplo *fecha de nacimiento* y *edad*.

6.3. Segunda Forma Normal (2FN)

- ***Una tabla está en segunda forma normal, si está en primera forma normal y además cada columna que no sea clave, depende funcionalmente de forma completa de cualquiera de las claves.***
- Es decir, toda la clave principal debe hacer dependientes al resto de atributos. Si hay atributos que depende sólo de parte de las claves, entonces todos ellos formarán otra tabla que ya si estará en 2FN.

6.3. Tercera Forma Normal (1FN)

- ***Una tabla está en tercera forma normal, cuando está en 2FN y además ningún atributo que no sea clave depende transitivamente de las claves de la tabla.***
- Es decir, una tabla no está en 3FN cuando algún atributo no clave depende funcionalmente de atributos que no son clave.

Actividades

Ejercicios Tema3