1. Introducción.

El modelo de datos relacional es posterior a los modelos jerárquicos y de red. Nació como consecuencia de los trabajos publicados en 1969-70 por E. F. **Codd**, aunque los primeros SGBD relacionales no aparecen en el mercado hasta principios de los años ochenta.

El modelo relacional se ocupa de tres aspectos de los datos: su **estructura**, su **integridad** y su **manipulación.** La estructura lógica y el modo de realizar las operaciones de E/S en el enfoque relacional son distintos con respecto a los enfoques jerárquico y de red. Algunas de las diferencias con las bases de datos jerárquicas o en red son:

> A la hora de operar con los datos, en lugar de hacerlo sobre registros, actúa sobre la relación (tabla).
> Las bases de datos relacionales no permiten ni grupos repetitivos ni apuntadores en los niveles externo y conceptual; en el nivel interno el sistema tiene libertad para usar cualquier estructura que desee.

Las bases de datos relacionales se basan en el uso de tablas (también se las llama **relaciones**). Las tablas se representan gráficamente como una estructura rectangular formada por filas y columnas.

Cada fila posee una ocurrencia o ejemplar de la instancia o relación representada por la tabla (a las filas se las llama también **tuplas o registros**).

Cada columna almacena información sobre una propiedad determinada de la tabla (se le llama también **atributo)**, nombre, dni, apellidos, edad,... Cuando no se conoce el valor de un atributo se le asigna el valor **nulo.** Los valores nulos indican contenidos de atributos que no tienen ningún valor. En claves foráneas indican que el registro actual no está relacionado con ninguno. Las bases de datos relacionales admiten utilizar ese valor en todo tipo de operaciones.

NOMBRE

atributo 1	**atributo 2**	**atributo 3**	**....**	**atributo n**	
valor 1,1	valor 1,2	valor 1,3	....	valor 1,n	← **tupla 1**
valor 2,1	valor 2,2	valor 2,3	....	valor 2,n	← **tupla 2**
.....	**.....**	.....	**....**	.....	....
valor m,1	valor m,2	valor m,3	....	valor m,n	← **tupla m**

2. Conceptos del Modelo Relacional.

Relación (Tabla). Es el elemento central del modelo relacional. Son los objetos principales sobre los que debe recogerse información y generalmente denotan personas, lugares, cosas o eventos de interés. Una relación tiene un nombre, un conjunto de atributos que representan sus propiedades y está formada por un conjunto de tuplas que incluyen los valores que cada uno de los atributos toma para cada una de las tuplas de la

relación. Una relación se representa mediante una tabla bidimensional (las columnas representan los atributos y las filas representan las tuplas o registros).

NIF	NOMBRE	LOCALIDAD
1111	Manuel	Málaga
3333	Gabriel	Granada
5555	Marco	Málaga
7777	Carlos	Cádiz

Tabla PROVEEDORES

CODIGO	NIF-PRO	CONCEPTO
11	5555	Teclado
22	7777	Impresora
33	1111	Monitor

Tabla ARTICULOS

Tupla o registro. Corresponde a una fila de la tabla. Representa cada una de las ocurrencias de la relación (*equivale a lo que conocemos como ocurrencia de un registro, en ficheros clásicos).* El número de tuplas se denomina **cardinalidad**, la cardinalidad varía con el tiempo.

Dominio. Es una colección de valores, de los cuales uno o más atributos obtienen sus valores reales. Pueden ser finitos (dias de la semana, meses del año, letras del alfabeto, etc..) o infinitos (números reales, dias del calendario – siempre que no esten limitados por el sistema operativo o el SGBD-, etc..)

Atributo. Corresponde a una columna de la tabla (*equivale a un campo de un registro)* y se definen sobre dominios. El número de atributos se llama **grado**. El grado no varía con el tiempo, si añadimos un atributo a una relación, podemos considerar que se trata de otra relación nueva.

Clave candidata es un atributo K (o conjunto de atributos) de una relación R que cumple dos propiedades:

- ***Unicidad****:* No existen dos tuplas en R con el mismo valor de K.
- ***Minimalidad:*** Si K es compuesto, no será posible eliminar ningún componente de K sin destruir la propiedad de unicidad.

Por ejemplo, el atributo compuesto (NIF,LOCALIDAD) no es una clave candidata de la relación PROVEEDORES, ya que podemos eliminar el atributo LOCALIDAD sin destruir la propiedad de unicidad, es decir, siguen sin existir dos tuplas con el mismo valor de NIF.

Clave primaria. Es posible que una relación posea más de una clave candidata, en ese caso, se escoge una de ellas como **clave primaria** y el resto se denominan **claves alternativas**. En la práctica la elección de la clave primaria suele ser sencilla. *Toda relación, sin excepción, tiene una clave primaria* y suele representarse subrayando y/o añadiendo el carácter # al atributo (o conjunto de atributos) correspondiente.

Por ejemplo: Artículos(código#, concepto)

Clave foránea, ajena o **extranjera** es un atributo (o conjunto de atributos) de una relación R1 que a la vez es clave primaria de otra relación R2. Se utiliza para referenciar a la tupla de R2 cuya clave primaria coincida con el valor de la clave foránea de R1. Ambas claves deben definirse sobre el mismo dominio.

Por ejemplo, el atributo NIF-PROV de la relación PRECIOS es clave foránea ya que se utiliza para referenciar a una tupla de PROVEEDORES mediante la clave primaria NIF.

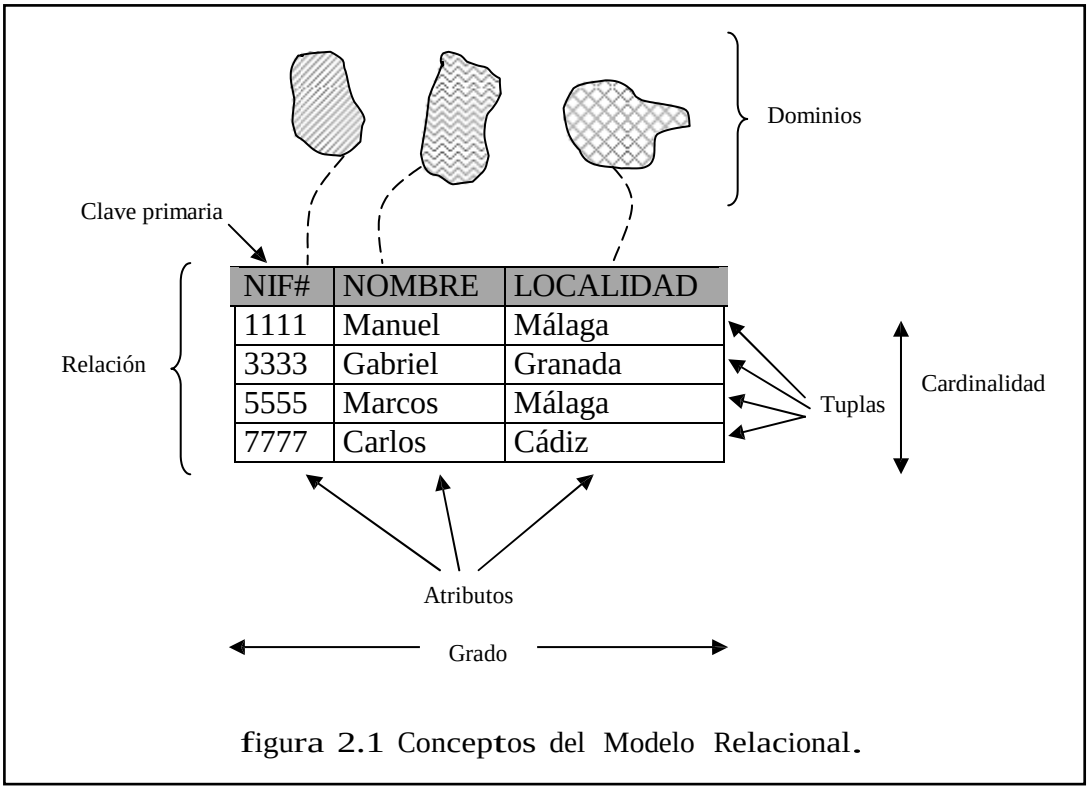


figura 2.1 Conceptos del Modelo Relacional.

3. Restricciones Inherentes del Modelo Relacional.

a) No existen tuplas repetidas (obligatoriedad de clave primaria).

La relación se ha definido como un conjunto de tuplas, y en matemáticas los conjuntos por definición no incluyen elementos repetidos. Hay que decir, sin embargo, que muchos de los SGBD relacionales sí admiten duplicidad de tuplas.

b) El orden de las tuplas y el de los atributos no es relevante.

c) Cada atributo de cada tupla solo puede tomar un único valor sobre el dominio sobre el que está definido.

d) Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo (regla de integridad de entidad)

Estas restricciones son las que marcan la diferencia entre una tabla y una relación, ya que una tabla presenta las filas y las columnas en un orden, del cual carecen las relaciones. Por otro lado, una tabla podría contener filas repetidas. De todos modos, como dijimos anteriormente, es muy común utilizar el término tabla para referirse a una relación.

Las 12 reglas de Codd

Preocupado por los productos que decían ser sistemas gestores de bases de datos relacionales (RDBMS) sin serlo, Codd publica las 12 reglas que debe cumplir todo DBMS para ser considerado relacional. Estas reglas en la práctica las cumplen pocos sistemas relaciónales. Las reglas son:

1. **Información**. Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.

2. **Acceso garantizado**. Todo dato es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato.

3. **Tratamiento sistemático de los valores nulos**. El DBMS debe permitir el tratamiento adecuado de estos valores.

4. **Catálogo en línea basado en el modelo relacional.** Los metadatos deben de ser accesibles usando un esquema relacional.

5. **Sublenguaje de datos completo**. Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación.

6. **Actualización de vistas.** El DBMS debe encargarse de que las vistas muestren la última información.

7. **Inserciones, modificaciones y eliminaciones de dato nivel.** Cualquier operación de modificación debe actuar sobre conjuntos de filas, nunca deben actuar registro a registro.

8. **Independencia física.** Los datos deben de ser accesibles desde la lógica de la base de datos aún cuando se modifique el almacenamiento.
   la forma en la que se almacenan

9. **Independencia lógica.** Los programas no deben verse afectados por cambios en las tablas.

10. **Independencia de integridad.** Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.

11. **Independencia de la distribución.** El sublenguaje de datos debe permitir que sus instrucciones funcionen igualmente en una base de datos distribuida que en una que no lo es.

12. **No subversión**. Si el DBMS posee un lenguaje que permite el recorrido registro a registro, éste no puede utilizarse para incumplir las reglas relacionales.

4. Restricciones Semánticas o de Usuario.

a. ***Restricción de Clave Primaria*** *(PRIMARY KEY)*, permite declarar un atributo o conjunto de atributos como la clave primaria de una relación.

b. ***Restricción de Unicidad*** *(UNIQUE)*, permite que una clave alternativa o secundaria pueda tomar valores únicos para las tuplas de una relación (como si de una clave primaria se tratara). Se entiende que la clave primaria siempre tiene esta restricción.

c. ***Restricción de Obligatoriedad*** *(NOT NULL)*, permite declarar si uno o varios atributos de una relación debe tomar siempre un valor.

d. ***Restricción de Integridad Referencial o de Clave Foránea*** *(FOREIGN KEY)*, se utiliza para que mediante claves foráneas podamos enlazar relaciones de una base de datos. La integridad referencial nos indica que si una relación tiene una clave foránea que referencia a otra relación, cada valor de la clave foránea o ajena tiene que ser igual a un valor de la clave principal de la relación a la que referencia, o bien, ser completamente nulo. Los atributos que son clave foránea en una relación no necesitan tener los mismos nombres que los atributos de la clave primaria con la cual ellos se corresponden.

El diseñador de la base de datos deberá poder especificar qué operaciones han de rechazarse y cuáles han de aceptarse, y en este caso, qué operaciones de compensación hay que realizar para mantener la integridad de la base de datos. Para ello el diseñador debe plantearse tres preguntas por cada clave foránea:

1. ***¿Puede aceptar nulos esa clave foránea?*** Por ejemplo, (tomando como referencia las relaciones PROVEEDORES, ARTICULOS) ¿tiene sentido la existencia de un articulo cuyo proveedor se desconoce? Evidentemente, no. En algunos casos esta respuesta podría ser distinta, por ejemplo, en una base de datos con las relaciones EMPLEADOS y DEPARTAMENTOS, podría existir un empleado no asignado de momento a un departamento.

2. ***¿Qué deberá suceder si se intenta eliminar una tupla referenciada por una clave foránea?*** Por ejemplo, si se intenta eliminar un proveedor del cual existe algún artículo. En general, para estos casos existen por lo menos tres posibilidades:

   Restringir: La operación de eliminación se restringe sólo al caso en el que no existe alguna tupla con clave foránea que la referencie, rechazándose en caso contrario. En nuestro ejemplo, un proveedor podrá ser borrado, si y sólo si, por ahora, no suministra artículos.

   Propagar en cascada: La operación de borrado se propaga en cascada eliminando también todas las tuplas cuya clave foránea la referencien. En nuestro ejemplo, se eliminaría el proveedor y todas las tuplas de artículos suministrados por él.

Anular: Se asignan nulos en la clave foránea de todas las tuplas que la referencien y se elimina la tupla referenciada. En nuestro ejemplo, no tiene mucho sentido, pero consistiría en asignar nulos al NIF-PROV de todas las tuplas de articulos pertenecientes al proveedor que queremos borrar, y posteriormente borrar al proveedor.

3. ***¿Qué deberá suceder si hay un intento de modificar la clave primaria de una tupla referenciada por una clave foránea?*** Por ejemplo, si se intenta modificar la clave de un proveedor del cual existe algún artículo. Se actua con las mismas tres posibilidades que en el caso anterios:

   Restringir.
   Propagar en cascada.
   Anular.

e. ***Restricción de Valor por Defecto*** *(DEFAULT)*, permite que cuando se inserte una tupla o registro en una tabla, para aquellos atributos para los cuales no se indique un valor exacto se les asigne un valor por defecto.

f. ***Restricción de Verificación o Chequeo*** *(CHECK);* en algunos casos puede ocurrir que sea necesario especificar una condición que deben cumplir los valores de determinados atributos de una relación de la BD, aparte de las restricciones vistas anteriormente.

g. **Aserciones** (*ASSERTION*): Esta restricción generaliza a la anterior, lo forman las aserciones en las que la condición se establece sobre elementos de distintas relaciones (por ello debe tener un nombre que la identifique).

h. ***Disparadores*** *(TRIGGERS);* a veces puede interesar espeficar una acción distinta del rechazo cuando no se cumple una determinada restricción semántica. En este caso, se recurre al uso de disparadores o triggers que nos permiten además de indicar una condición, especificar la acción que queremos que se lleve a cabo si la condición se hace verdadera. Los disparadores pueden interpretarse como reglas del tipo evento-condición-acción (ECA) que pueden interpretarse como reglas que especifican que cuando se produce un evento, si se cumple una condición, entonces se realiza una determinada acción.

5. Un ejemplo de arquitectura relacional.

La figura que viene a continuación muestra la arquitectura de un sistema relacional, bajo las recomendaciones ANSI/SPARC:

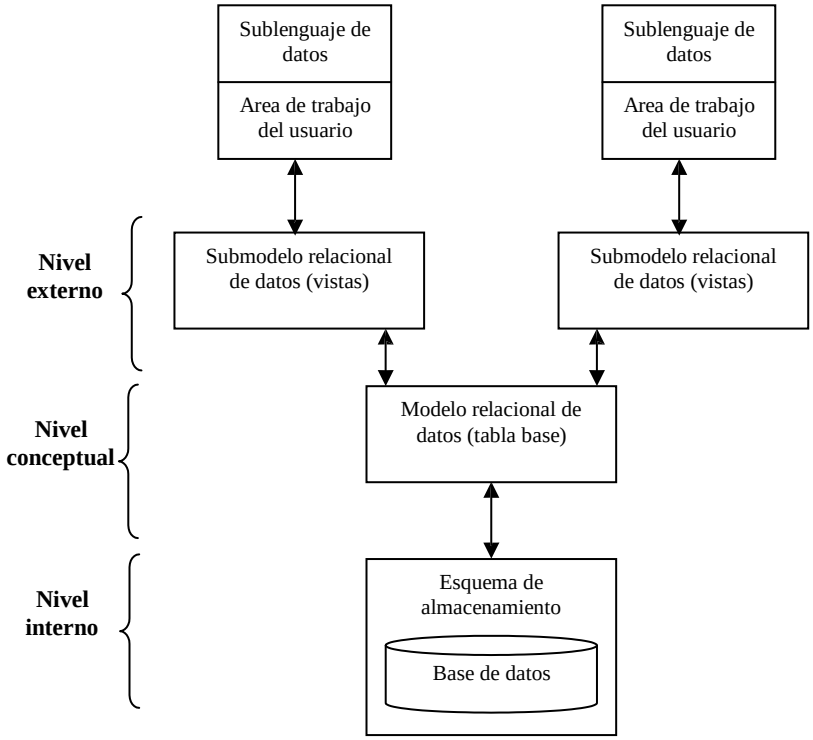
Podemos apreciar las siguientes partes:

El **nivel conceptual**, definido por el ***modelo relacional de datos***. Se trata de una colección de relaciones, también llamadas ***tablas*** con todas sus restricciones asociadas**.**

El **nivel externo**, definido por el llamado ***submodelo relacional*** de datos o ***vista***. Se puede acceder a datos de una relación o de una vista, que es una relación que no tiene existencia por sí misma y se deriva de una o más tablas. Cada programa utiliza un buffer, llamado

área de trabajo del usuario, para depositar o recuperar los datos antes de guardarlos en la base de datos o antes de ser procesados.

El **nivel interno**, donde cada tabla se representa por un archivo distinto.



Un **sublenguaje de datos** para el manejo de los datos del sistema relacional. Los sublenguajes de datos se pueden clasificar en dos grupos:

- Lenguaje **de procedimiento**. Basado en el **álgebra relacional**, permite al usuario manipular relaciones por medio de operadores algebraicos relacionales para obtener los resultados requeridos.
- Lenguaje **sin procedimiento**. Basado en el **cálculo relacional**, permite al usuario especificar exactamente qué datos desea, sin tener que detallar ni codificar la forma de obtenerlos a partir de las relaciones disponibles en la base de datos.

6. Diseño lógico.

El objetivo del diseño lógico es conseguir a partir del modelo conceptual un modelo lógico de datos, en nuestro caso formado por un conjunto de relaciones o tablas, que sea fácilmente implementable, fácil de usar, independiente de las estructuras lógicas y físicas de los datos y que limite la posibilidad de tener datos incoherentes.

Para ello, las etapas a seguir serán fundamentalmente dos:

1. Transformar los objetos conceptuales en relaciones.
2. Normalización de las relaciones obtenidas.

6.1 Transformación de los objetos conceptuales en relaciones.

Para realizar este punto se han de seguir los siguientes pasos:

Eliminación de atributos múltiples

Todos los atributos múltiples, es decir, los atributos que pueden tomar más de un valor en el dominio en el cual están definidos, se transformarán en una tabla débil por existencia, la cual mantendrá una relación uno a muchos con la tabla sobre el cual estaba definido. Esta tabla, creada por aplicación de esta regla, tendrá como atributos:

el atributo múltiple por el cual la regla se ha aplicado, y además
se deberá tener en cuenta que si el atributo de la tabla débil creada no pudiera identificar sin ambigüedad a las ocurrencias de esta, entonces se procederá de alguna de las dos formas siguientes:

- La tabla débil creada se considera que es débil por identificación con respecto a la tabla con la que mantiene relación, heredando, por tanto, sus atributos identificadores. Si el atributo múltiple estaba definido sobre una relación, heredará los identificadores de las tablas, que participaban en la relación.
- Se añadirá un nuevo atributo (externo, o no, al dominio del problema) que permita identificar sin ambigüedad a las ocurrencias de esta tabla débil.

Eliminación de atributos compuestos

Todos los atributos compuestos deben ser descompuestos en los atributos simples que formen parte o intervengan en los atributos compuestos correspondientes.

En este proceso de descomposición, se eliminará el atributo compuesto quedando los atributos simples e interviniendo de la misma forma en la tabla o relación.

Transformación de entidades.

Regla 1: Todas las entidades presentes en el esquema/modelo conceptual se transformarán en tablas en el esquema relacional manteniendo, el número y tipo de atributos, así como la característica de identificador de estos atributos.

Por ejemplo, Clientes (CodCli , NomCli, DirCli,TfnoCli)

"Como norma general, los atributos claves de las tablas se representarán subrayados, las claves alternas con doble subrayado, y las claves foráneas en negrita."

Transformación de Relaciones uno a uno.

Dependiendo de la forma en que participa cada entidad (cardinalidad) en la relación será necesario aplicar una regla diferente:

Regla 2.1: Si en una relación binaria las entidades participan con cardinalidad mínima y máxima igual a uno, entonces:

a) Sí las dos entidades tienen la misma clave :

Las dos entidades se transforman en una única tabla formada por la agregación de los atributos de las dos entidades, y
La clave de la tabla es el identificador de las entidades (es el mismo en ambas).

b) Si las entidades tienen diferente clave, cada entidad se transforma en una tabla y :

Cada tabla tendrá como clave principal el identificador de cada una de las entidades de la que se deriva.
Cada tabla tendrá como clave foránea el identificador del otro tipo de entidad con el cual está relacionado.

Regla 2. 2. Si en una relación binaria alguna de las entidades participa de forma parcial (0 ó 1), entonces, cada entidad se transforma en una tabla y la clave de la entidad que participa totalmente pasa como clave foránea de la otra tabla. En este caso esta clave foránea no podrá tomar valores nulos para diferentes tuplas de esta tabla.

Regla 2.3. Si en una relación binaria ambas entidades participan de forma parcial, entonces, cada entidad se transforma en una tabla y se procede de alguna de las dos formas siguientes:

a) Las claves de cada una de las entidades pasan a formar parte como atributos de las tablas correspondientes a la otra entidad. Estos atributos actuarán como claves foráneas en estas tablas.

b) Se crea una nueva tabla correspondiente a la relación y cuyos atributos serán las claves de las dos entidades relacionadas. La clave principal de esta nueva tabla será la clave de una de las entidades relacionadas y la clave de la otra entidad relacionada actuará como clave foránea. (En el mundo real, se opta en la mayoría de los casos, por esta segunda opción).

Ejemplo: Presupuestos – Pedidos.

Transformación de relaciones uno a muchos.

Regla 3.1: Si en una relación binaria 1:N ambas entidades participan de forma total (1 : N), entonces, cada entidad se transforma en una tabla, y la clave de la entidad que participa con cardinalidad uno pasa a formar parte de la tabla correspondiente a la entidad que participa con cardinalidad muchos. Este atributo será definido como clave foránea de esta tabla manteniendo una referencia con la tabla correspondiente a la entidad que participa con cardinalídad uno. Si la relación tuviera atributos asociados, estos atributos pasan a formar parte de la tabla correspondiente a la entidad que participa con cardinalidad muchos.

Regla 3.2: Si en una relación binaria 1:N ambas entidades participan de forma parcial (0 : 1 – 0 : N) o la entidad que participa con cardinalidad uno lo hace de forma parcial, entonces, cada entidad se transforma en una tabla y se genera una nueva tabla correspondiente a la relación. Esta tabla estará formada por las claves de las entidades que intervienen en la relación y por todos los atributos asociados a la relación. La clave principal de esta tabla será el atributo identificador correspondiente a la entidad que interviene con cardinalidad muchos, y será necesario definir como claves foráneas los atributos claves correspondientes a la entidad con cardinalidad uno. ambas claves principales son foráneas

Transformación de relaciones muchos a muchos.

Regla 4: En una relación binaria N : N cada entidad se transforma en una tabla y se genera una nueva tabla para representar a la relación. Esta tabla estará formada por las claves de las dos entidades que intervienen en la relación y por todos los atributos asociados a la relación. La clave principal de esta tabla será la agregación de los atributos claves correspondientes a las entidades que intervienen en la relación.

Además en esta nueva tabla, los atributos claves correspondientes a las entidades que intervienen en la relación, actuarán como claves foráneas a las tablas obtenidas a partir de las entidades que intervenían en la relación.

Transformación de relaciones N-arias.

Las relaciones N-arias son aquellas donde en la misma relación participan más de dos entidades. En este caso de relaciones se aplica la **Regla 4,** es decir, cada entidad se transforma en una tabla y se genera una nueva tabla para representar a la relación. Esta tabla estará formada por las claves de las entidades que intervienen en la relación y por todos los atributos asociados a la relación. La clave principal de esta tabla será la agregación de los atributos claves correspondientes a las entidades que intervienen en la relación, que a su vez también actuarán como claves foráneas.

Aunque lo que viene a continuación se deduce de las reglas anteriores, resaltaremos como tipos de relaciones específicas:

Transformación de una jerarquía.

Los tipos y los subtipos, no son objetos que se puedan representar explícitamente en el modelo relacional. Puede optarse por tres formas:

Opción a: consiste en crear una tabla para el supertipo que tenga como clave primaria el identificador principal y una tabla para cada uno de los subtipos que tenga el identificador de supertipo como clave ajena. Generalmente la clave del supertipo es también clave primaria en cada una de las tablas de los subtipos.
En esta solución las reglas de modificación y de borrado de cada subtipo con el supertipo son en cascada.
La solución es apropiada cuando los subtipos tienen muchos atributos distintos y se quieren conservar los atributos comunes en una tabla. También se deben implantar los mecanismos que aseguren que se conserva toda la semántica, por ejemplo, si la relación es exclusiva hay que verificar que sólo se inserte un subtipo.
Opción b: se crea una tabla para cada subtipo, los atributos comunes aparecen en todos los subtipos y la clave primaria para cada tabla es el identificador del supertipo. Esta opción mejora la eficiencia en los accesos a todos los atributos de un subtipo, sean los comunes al supertipo o los específicos. Este caso no es deseable en el caso de generalización solapada ya que genera redundancia de los atributos comunes que habría que controlar.
Opción c: agrupar en una tabla todos los atributos de la entidad supertipo y de los subtipos.
La clave primaria de esta tabla es el identificador de la entidad. Se añade de un atributo que indique a qué subtipo pertenece cada ocurrencia (el atributo discriminante de la jerarquía).
Esta solución puede aplicarse cuando los subtipos se diferencien en pocos atributos y las relaciones entre los subtipos y otras entidades sean las mismas. Para el caso de que la jerarquía sea total, el atributo discriminante no podrá tomar valor nulo (ya que toda ocurrencia pertenece a alguna de las entidades del subtipo)
En este caso habrá que comprobar que si el discriminante toma un valor concreto, se llenen sólo los atributos que le correspondan por subtipos, quedando los de los otros subtipos a nulo.

Transformación de relaciones de dependencia.

Se utiliza el mecanismo de propagación de clave, creando una clave ajena, con nulos no permitidos, en la relación de la entidad dependiente, con la característica de obligar a una modificación y un borrado en cascada. Además en el caso de dependencia en identificación la clave primaria de la relación en la que se ha transformado la entidad débil debe estar formada por la concatenación de las claves de las dos entidades participantes en la relación.

6.2 Normalización.

El Proceso de Normalización se encarga de seguir una serie de pasos o normas que, tras aplicarlas todas ellas, se obtienen los datos agrupados en diferentes tablas, donde cada una de ellas tiene la estructura óptima para su implementación, gestión y explotación para la futura aplicación.

Aunque un diseñador experimentado puede llegar a obtener relaciones normalizadas directamente, nosotros seguiremos un método formal.

Originalmente E.F.Codd (1970) definió tres niveles de normalización llamados primera forma normal (1FN), segunda forma normal (2FN) y tercera forma normal (3FN), existiendo una revisión de esta última propuesta por Boyce y el propio Codd que se llamó forma normal de Boyce-Codd (FNBC), posteriormente algunos autores (Fagin principalmente) definieron más niveles (4FN, 5FN...). En la práctica, normalizar hasta estos últimos niveles no suele tener sentido, ya que se generan gran cantidad de relaciones produciendo una implementación física difícil de mantener, teniéndose que realizar la operación contraria, es decir, la desnormalización.

*Una tabla se dice que está en una Forma Normal (**F.N**) cuando satisface un conjunto de restricciones impuestas por esa Norma.*

Las ventajas que se obtienen tras la normalización de datos son:

Facilidad de uso, ya que los datos están agrupados en tablas que identifican claramente un objeto y todas sus propiedades.
Independencia de Datos. Los programas no están ligados a las estructuras de datos.
Facilidad de Gestión. Los lenguajes manipulan la información de forma sencilla al estar basados en el Algebra y Calculo Relacional.
Mínima Redundancia. La información no estará duplicada innecesariamente dentro de las estructuras.
Flexibilidad. Se puede obtener cualquier información de las tablas , de relaciones entre las tablas o mediante operaciones de algebra relacional.
Seguridad. Los controles de acceso a la información son más fáciles de implementar, e incluso, pueden no depender de la aplicación.

6.2.1 Definiciones e ideas básicas.

Antes de seguir es importante definir algunos conceptos que posteriormente se utilizarán.

Dependencia funcional. Dados dos atributos (o conjuntos de atributos) A y B de una relación R, se dice que B es funcionalmente dependiente de A, si cada valor de A tiene asociado un único valor de B. En otras palabras: si en cualquier instante, conocido el valor de A podemos conocer el valor de B. Tanto A como B pueden ser conjuntos de atributos. La dependencia funcional se simboliza del siguiente modo:

no se cumple la dependencia funcional cuando el antecedente se repite para distintos consecuentes

R.A $\longrightarrow$ R.B

no se puede repetir — si se puede repetir

En las dependencias funcionales, al atributo (o conjunto de atributos) A se le denomina **Antecedente** y al al atributo (o conjunto de atributos) B se le denomina **Consecuente**.

Por ejemplo, en la relación *Pedidos : Fecha* y *CodPro* dependen funcionalmente de *NumPed*.

$$NumPed \longrightarrow (Fecha, CodPro)$$

Dependencia funcional completa *se dice que un atributo B tiene dependencia funcional completa de un grupo de atributos A (al, a2,...) de la misma relación, si B depende funcionalmente de todo A, pero no de un subconjunto de A.*

$$(R.a1, R.a2) \longrightarrow R.B$$

Por ejemplo, en la relación *Precios*, el precio de un artículo dependía funcionalmente completa del conjunto de atributos código de artículo, código del proveedor:

$$(CodPro, CodArt) \longrightarrow Precio$$

es decir, no se puede conocer el precio si no se conoce tanto el código de artículo como el código del proveedor. Si se conoce solo uno de los dos atributos

Dependencia transitiva: *sean A, B y C tres atributos o conjuntos de atributos de una relación. Si B depende funcionalmente de A y C de B, pero A no depende funcionalmente de B, se dice que C depende transitivamente de A.*

Por ejemplo: en la relación *Pedidos* los atributos nombre y dirección del proveedor dependen transitivamente de la clave NumPed, es decir, no dependen funcionalmente de ella sino del atributo código del proveedor. Por otro lado NumPed no depende de CodPro.

$$\underset{A}{NumPed} \longrightarrow \underset{B}{CodPro} \longrightarrow \underset{C}{(NomPro, DirPro)}$$

Agrupación de Dependencias Funcionales: Cuando dos D.F. tienen el mismo Antecedente, se pueden agrupar formando una única D.F.

Por ejemplo : Sea una relación R, sobre la cual se definen dos D.F.:

$$DNI \longrightarrow Nombre \text{ y } DNI \longrightarrow Dirección, Población$$

entonces se pueden agrupar en una sola D.F

$$DNI \longrightarrow Nombre, Dirección, Población$$

Con el proceso contrario se obtendrá la **"Descomposición por Yunción".**

6.2.2 Propiedades de las D.F.

Sean X, Y , W y Z conjuntos de atributos.

Reflexiva :
Si X Y, entonces Y ⟶ X
Por ejemplo : Y = {Nombre, Dirección}, X={Nombre}
Aumentativa :
Si X ⟶ Y, entonces ZX ⟶ ZY
Transitiva :
Si X ⟶ Y , Y ⟶ Z, entonces X⟶ Z
PseudoTransitiva :
Si X ⟶ Y , YZ ⟶ W, entonces ZX⟶ W
Ejemplo : X = NumMatr, Y = NomAlumno, Z = NumExamen, W = NotaExamen
Descomposición :
Si X ⟶ Y, Z Y, entonces Y ⟶ Z
Ejemplo: X = {DniAlu }Y = {NomAlu,DirAlu,PobAlu} Z = {NomAlu,PobAlu}

- ***Descomposición por Yunción.***

  Sea R una tabla formada por los conjuntos de atributos A, B y C, tales que entre ellos se establece la D.F. A ⟶B, C

  Se dice entonces que R se puede Descomponer por Yunción en dos tablas R1 y R2, en donde :
  R1(A,B) A ⟶B y R2(A,C) A ⟶ C
  ya que
  R R1 * R2

  Ejemplo : A = DniAlu, B = NomAlu y C = PobAlu

  A esto anterior se le llama **Descomposición por Yunción Natural**, ya que al hacer el Producto Cartesiano de R1 y R2 y seleccionar aquellas tuplas en las que el Antecedente de ambas tablas sea el mismo se vuelve a obtener R.

7. Formas normales.

La aplicación de una F.N. tiene como entrada una tabla y da como resultado dos o más tablas, y se ha de tener en cuenta:

La tabla objeto de la aplicación de la F.N se desestima en el nuevo esquema relacional considerado.
No se introducen nuevos atributos en el esquema relacional resultante de la normalización.
Los atributos de la tabla objeto de la aplicación de la F.N pasan a formar parte de una o más de las tablas resultantes.
En la aplicación de la norma se ha debido eliminar, al menos, una dependencia existente entre los atributos de la tabla objeto de la aplicación de la F.N.

7.1 Primera forma normal.

Una relación está en primera forma normal si el valor de cada atributo de cada fila no se puede descomponer, es decir contiene valores atómicos (ni compuestos ni múltiples).

Un atributo compuesto es aquel que se puede descomponer en atributos distintos, por ejemplo, datos personales, que se podría descomponer en nombre, dirección, etc...

Un atributo múltiple es aquel en el que su contenido guarda dos o más datos referidos a lo mismo, por ejemplo, el atributo notas, que podría guardar información de la forma: 8, 5, 3, ...

Que no haya grupos repetitivos

Inconvenientes de una tabla que no está en 1ª FN :

- Hay que definir el número de veces que el atributo se repite. Además este número es fijo y por tanto no puede incrementarse si la aplicación lo necesitase.
- Reserva de espacio innecesaria para estos atributos.

Pasos para poner una tabla en 1ª F.N :

- Buscar los atributos que forman la clave primaria de la tabla.
- Buscar los atributos repetitivos (Matrices)
- Por cada conjunto de atributos anteriores se crea una nueva tabla, con estos atributos y los que forman la clave primaria.
- A la tabla inicial se le quitan los atributos repetitivos.

Por ejemplo :

R(CodArt,DesArt,Talla1,Talla2,Talla3,Talla4,Talla5,Talla6)

Se obtiene :

R1(CodArt,DesArt)

R2(CodArt,Talla)

7.2. Segunda forma normal.

Una relación estará en 2FN si ya está en 1FN y cada uno de sus atributos depende totalmente de la clave y no de parte de ella. De una forma más rigurosa podemos decir que una relación está en 2FN si está en 1FN y cada atributo que no pertenezca a una clave candidata concreta tiene una ***dependencia funcional completa*** de dicha clave candidata.

Una tabla **NO** está en 2ª FN si se cumple que:

- Existe una D.F donde el antecedente está formado por dos o más atributos, y además,
- Existen atributos en el consecuente de la tabla que dependen de parte no completa del antecedente anterior.

Que esté en 1FN

Pasos para poner una tabla en 2ª F.N :

todos los atributos tienen que depender de forma completa del conjunto de la pk, si alguno depende solo de una parte, se saca a una relacion aparte

Truco: si la pk tene un solo atributo, ya está en 2fn, no hay k mirar nada, esto no significa que todos las 2fn tengan k tener solo 1 atributo, ya que esto se consigue si hay un grupo de pk y no hay ningun atributo k se determine de un subgrupo de este, sino se saca a otra relacion

- Partiendo de una relación en 1FN pasamos a 2FN identificando los atributos que no dependen completamente de alguna clave candidata y formando con ellos una nueva relación quitándolos de la antigua, la clave principal de la nueva relación estará formada por la parte de la antigua de la que dependen totalmente.
- Dada una tabla R cuya clave o antecedente es un agregado de la forma (R.x,R.y), y en esta tabla existe una D.F incompleta de la forma
  R.y R.z, donde R.z es un atributo que no forma parte de la clave, el proceso de descomposición se realizará de la siguiente forma :

  a. De la tabla R se elimina el atributo R.z.
  b. Se contruye una nueva tabla R1, con R.y y R.z, siendo R.y la clave primaria de R1.

Por ejemplo:
R (CodBco,CodSuc,CodCon,NCC,NomBanco,NomCliente,Saldo)
donde existe la D.F:
CodBco,CodSuc,CodCon,NCC NomBanco,NomCliente,Saldo y
como NomBanco depende solo del CodBco , se obtiene:
CodBco,CodSuc,CodCon,NCC NomCliente,Saldo
CodBco NomBanco

7.3 Tercera forma normal.

Una relación estará en 3FN si ya está en 2FN y cada uno de sus atributos depende solamente de la clave y no de otros atributos. De una forma más rigurosa podemos decir que una relación estará en 3FN si está en 2FN y cada atributo que no pertenezca a una clave candidata concreta ***no depende transitivamente*** de dicha clave candidata.
Que esté en 2FN

Partiendo de una relación en 2FN pasamos a 3FN identificando los atributos que dependen de otro atributo distinto de una clave candidata y formando con ellos otra relación, quitándolos de la antigua. La clave principal de la nueva relación será el atributo del cual dependen. Este atributo en la relación antigua pasará a ser una ***clave extranjera*** o ***ajena***.
no puede haber dependencias transitivas, todos los atributos tienen que depender unicamente del pk

Pasos para poner una tabla en 3ª F.N :

Dada una tabla R de clave R.x y dos conjuntos de atributos R.y y R.z no primos[1], y en esta tabla están presentes las siguientes D.F :
R.x R.y , R.y R.z, y por tanto existe la D.F. transitiva R.x R.z, el proceso de descomposición se realizará de la siguiente forma :

- de la tabla R se elimina R.z
- se crea una nueva tabla R1 con R.y y R.z, donde R.y será la clave primaria y donde solo existirá una D.F. completa de la forma R.y R.z
- R.y se hará clave foránea en la relación R.

Por ejemplo:

[1] Se dice que un atributo es **primo** cuando pertenece a una clave candidata.

R (NFact, CodArt, NomArt, Precio, DtoArt, Cantidad)

donde existe la D.F:

Nfact CodArt, NomArt, Precio, DtoArt, Cantidad y

sabemos que :

CodArt NomArt, DtoArt

Se obtiene:

NFact Precio, Cantidad, CodArt

CodArt NomArt, DtoArt

7.4 Forma normal Boyce/Codd.

No tiene que cumplir las formas normales anteriores, pero al ser mas restrictiva tbn lo estará.

Para definir BCNF vamos a introducir un término nuevo. Se denomina **determinante funcional** a uno o un conjunto de atributos de una tabla R del cual depende funcionalmente de forma completa cualquier otro atributo de la misma tabla.

Una tabla R está en FNBC si está en FN1 y cada determinante funcional es una clave candidata de la tabla R. Es decir, de todas las D.F. establecidas sobre R, el antecedente siempre es una clave. candidata

Por ejemplo en la relación:

LINEASPEDIDO (NumPed, Fecha, CodPro, NomPro, DirPro, CodArt, Descrip, Cantidad, Precio)

Serían determinantes los siguientes atributos:

- NumPed, ya que de él dependen funcionalmente Fecha y CodPro.
- CodPro, ya que de él dependen funcionalmente NomPro y DirPro.
- CodArt, ya que de él dependen funcionalmente Descrip y Precio.
- NumPed+CodArt, ya que de ellos depende de forma funcionalmente completa Cantidad.

Pasos para poner una tabla en 3ª F.N :

- Dada una tabla R en la cual están presentes uno o más determinantes funcionales, R.x,R.y,…,R.z, posiblemente formados por un agregado de datos de la forma: R.[] = {R.a,R.b,…,R.j}, y en esta tabla existen D.F. que no son completas de la forma R.[] R.n, donde R.n es un atributo (o conjunto de atributos) de la tabla R, el proceso de descomposición se realizará de la siguiente forma :

  a) de la tabla R se elimina el atributo R.n

  b) se construye una nueva tabla R1, con R.n como atributo no primo y R.m como atributo primo, donde R.m R.[], de tal forma, que existe una D.F de la forma : R.m R.n.

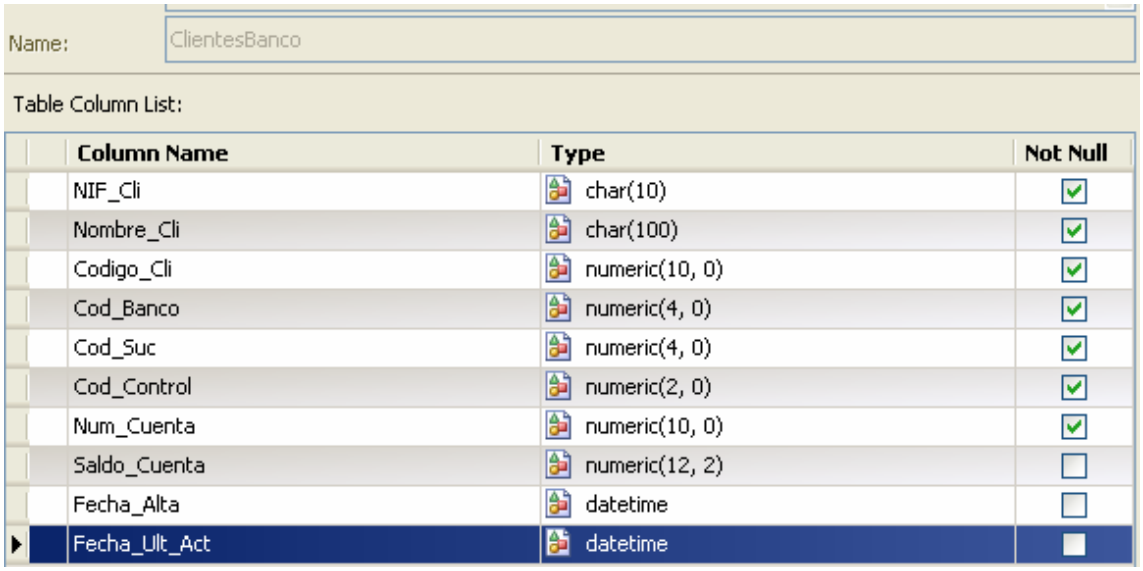
- Una relación que no está en FNBC debe descomponerse en otras dos de la siguiente forma: Sea una relación R(A,B,C,D,E,...) que no está en FNBC, siendo C un determinante, de forma que C ⟶ D, pero C no es una clave candidata. Se forman dos nuevas relaciones: R1(A,B,C,E,...) y R2(C,D). De nuevo se comprueba

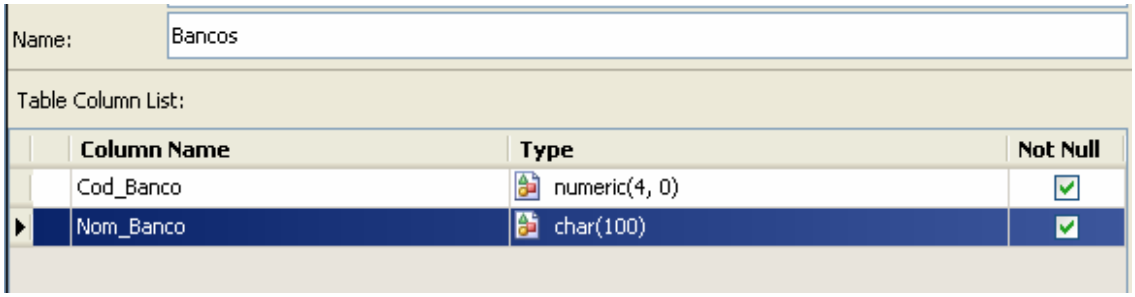
si R1 y R2 están en FNBC; en caso de no ser así se vuelve a repetir el mismo proceso.

Se sabe que una relación en FNBC está siempre en 3FN, sin embargo, una relación en 3FN, no tiene porqué estar en FNBC.

Práctica:

A partir del siguiente modelo relacional





1. Indica el nombre de las relaciones
2. Los dominios utilizados
3. Los atributos de la tabla **Bancos**
4. El grado de la tabla **ClientesBanco**
5. Enumera las posible claves candidatas de la tabla **ClientesBanco**
6. Indica la clave primaria y las alternativas o secundarias de la tabla **ClientesBanco**
7. Indica la clave foránea de la tabla **ClientesBanco** a la tabla **Bancos.**