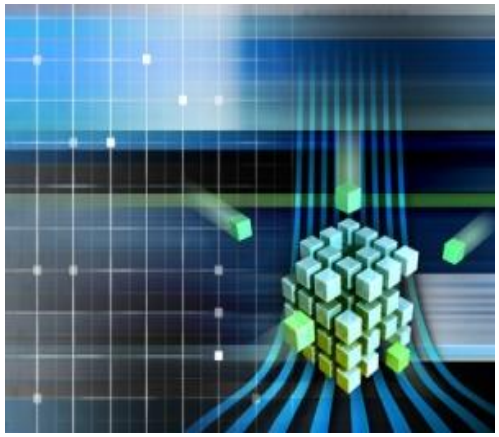
Tema 6
Diseño en el Modelo Relacional:
Teoría de la Normalización

Grado en Ingeniería Informática



Bases de Datos

2020/21

Departamento de Tecnologías de la Información
Universidad de Huelva

Diseño en el modelo relacional: Teoría de la Normalización

Objetivos

- Entender el concepto de dependencia funcional y el papel que juega dentro de un esquema relacional
- Adquirir una sólida base teórica, como es la teoría de la normalización, para el refinamiento del diseño lógico de bases de datos relacionales

Contenido

Diseño en el modelo relacional: Teoría de la Normalización

Duración

- 2 clases

Bibliografía

- Capítulo 14 [Elmasri 02]

Diseño en el modelo relacional: Teoría de la Normalización

6.1 Introducción

- En el modelo relacional, al igual que en casi todos los modelos de datos, existen distintas alternativas para almacenar la información. Por tanto, podemos obtener diferentes esquemas relacionales que serán probablemente distintos, de modo que algunos representarán la realidad mejor que otros

- Habrá que estudiar las propiedades que debe tener un esquema relacional para que represente adecuadamente la realidad y, también, los posibles problemas que podemos encontrar si no realizamos un buen diseño

- En el diseño de una base de datos, para llegar al modelo relacional, podemos seguir dos caminos:
  - Diseñando el esquema relacional a partir de la información de la que disponemos, realizando un análisis y obteniendo las tablas correspondientes y las restricciones de integridad
  - Realizando, en primer lugar, un diseño conceptual (por ejemplo, mediante el modelo E-R extendido) y transformando éste en un esquema relacional.

Diseño en el modelo relacional: Teoría de la Normalización

- Entre los problemas principales que se pueden presentar una vez obtenido el esquema relacional por cualquiera de los dos métodos anteriores están:
  - Incapacidad para almacenar ciertos hechos
  - Redundancias que pueden producir inconsistencias
  - Pérdida de información
  - Anomalías de inserción, borrado y modificación

- Como ejemplo, supongamos la tabla ESCRIBE, donde la clave primaria la forman la unión de los atributos (idLibro, autor) :

idLibro	título	editorial	Año	autor	nacionalidad
1000	Fundamentos de Bases de Datos	McGraw Hill	1994	Korth	EEUU
1000	Fundamentos de Bases de Datos	McGraw Hill	1994	Silberschatz	EEUU
1001	Sistemas de Bases de Datos	Addison-Wesley	1991	Date	EEUU
1002	La Biblia del Turbo Pascal	Anaya	1990	Duntemann	Inglaterra
1003	C Manual de Referencia	McGraw Hill	1991	Shildt	Canadá
1004	Sistemas Operativos	McGraw Hill	1993	Milenkovic	Croacia
1004	Sistemas Operativos	McGraw Hill	1993	Brezic	Croacia
1005	O-O Databases	Acm Press	1989	Kim	EEUU
1005	O-O Databases	Acm Press	1989	Lochovsky	Canadá

- En dicha tabla pretendemos almacenar información sobre autores que han escrito libros, junto con cierta información de los libros que han escrito. Veamos los problemas que se pueden producir al almacenar los datos de esta manera:
  - Existe gran cantidad de redundancia. Cuando un libro está escrito por más de un autor, se repiten el título, la editorial y el año. En cuanto a los autores, se repite la nacionalidad cada vez que almacenamos un autor
  - Se producen anomalías de modificación. Se puede cambiar el título de un libro sin modificarlo en el resto de tuplas con el mismo idLibro (inconsistencia)
  - Anomalías de inserción. Si quisiéramos introducir un autor del que no hubiera ningún libro en la base de datos, no sería posible, ya que el identificador del libro forma parte de la clave primaria y, por tanto, no puede tener valor nulo. Tampoco podríamos introducir obras anónimas. Por otra parte, la inserción de un libro escrito por dos autores implica la inserción de dos tuplas
  - Anomalías de borrado. Si damos de baja un libro escrito por un autor que sólo ha escrito dicho libro, perderíamos la información del autor. También, si borramos de nuestra base de datos un autor, desaparecería la información de todos los libros que hubiera escrito, a menos que estuviese escrito por más de un autor

- Esta tabla presenta, según se ha visto, numerosos problemas, puesto que no cumple un principio básico en todo diseño de bases de datos: HECHOS DISTINTOS SE DEBEN ALMACENAR EN OBJETOS DISTINTOS

Diseño en el modelo relacional: Teoría de la Normalización

- Ante las dudas de si un esquema es correcto o no, sería conveniente disponer de un método formal de análisis que determinara si el esquema es correcto, y que, en caso contrario, nos permitiera obtener uno equivalente que no conllevara los problemas descritos anteriormente.

  Este método formal se conoce como **Teoría de la Normalización**

- Este método trata de evitar las redundancias y las anomalías de actualización, inserción y borrado, proporcionando tablas más estructuradas.
- Por tanto, la forma correcta de estructurar la información anterior es:

LIBRO (idLibro, título, editorial, año)
CP: idLibro

AUTOR (nombre, nacionalidad)
CP: nombre

ESCRIBE (idLibro, nombre)
CP: (idLibro, nombre)
Caj: idLibro → LIBRO
Nombre → AUTOR

- Como contrapartida, las consultas resultarán más costosas, ya que la información está repartida en varias tablas

6.2 Dependencias funcionales

- Las *dependencias funcionales* (DF) son **restricciones adicionales** que permiten recoger contenido semántico inherente a los atributos de una tabla.

- Mediante las dependencias funcionales se pueden expresar determinadas relaciones existentes entre los atributos de una tabla y cuya semántica se intenta incorporar a la base de datos.
- Para el **proceso de normalización** es fundamental identificar todas las dependencias funcionales existentes entre los atributos de las tablas
  - Hay que procurar **conservar las DF a lo largo del proceso de diseño**, de modo que el esquema lógico resultante tras la normalización debe tener las mismas DF que el esquema de partida.
- Definición: Sea el esquema de relación *R* definido sobre el conjunto de atributos *A*, y sean *X* e *Y* subconjuntos de *A* llamados *descriptores*
  - Se dice que *Y* **depende funcionalmente de** *X* o que *X* **determina o implica a** *Y*, y se representa ***X*** → ***Y***, si, y sólo si, cada valor *x* de *X* tiene asociado, en todo momento, un valor *y* de *Y*
  - Es decir, para toda extensión o estado *r* de *R* y para todas las tuplas t_i, $t_j \in r$ se cumple:

$$\text{si } t_i[X] = t_j[X] \text{ entonces } t_i[Y] = t_j[Y]$$

	X	Y
i	v1	v2
	…	…
j	v1	v2

Diseño en el modelo relacional: Teoría de la Normalización

- En una DF, el descriptor que se encuentra a la izquierda del símbolo de implicación se denomina **determinante** o **implicante** y el que se encuentra a la derecha del símbolo se denomina **implicado**
  - En la DF: ***X*** → ***Y***, *X* es el *determinante* e *Y* es el *implicado*

- Por tanto, siempre que dos tuplas de *R* tengan el mismo valor en el descriptor *X*, también coincidirán en el valor del descriptor *Y*

- Las Dependencias Funcionales cumplen las siguientes propiedades:
  - Los atributos que forman el determinante y el implicado pertenecen a la misma relación
  - Son independientes del estado de una relación y, por tanto, son propiedades del esquema de una relación

- Si en un esquema de relación *R* tenemos la dependencia funcional *X* → *Y*, no podemos, en general, conocer el valor de *Y* a partir de un valor dado de *X*
  - Sólo podemos decir que si existen dos tuplas que tengan el mismo valor de *X*, entonces el valor de *Y* también será igual en ambas tuplas

Diseño en el modelo relacional: Teoría de la Normalización

- Ejemplo:

PISO (id_piso, ciudad, planta, garaje, metros, precio)

- En la relación PISO se cumple que:
  - El precio del piso depende funcionalmente de la superficie del mismo
  - El identificador del piso determina la ciudad, la planta, si tiene o no garaje, la superficie y el precio del mismo

- Se tienen por tanto las siguientes DF:

DF = { id_piso $\rightarrow$ ciudad, planta, garaje, metros, precio

metros $\rightarrow$ precio }

6.2.1 Dependencia funcional trivial

- Una dependencia funcional es **trivial** cuando el implicado es un subconjunto del implicante, no existiendo por tanto la posibilidad de que no se satisfaga tal DF
  - Para el proceso de normalización sólo nos fijaremos en las DF no triviales
- Ejemplo:

  ciudad, planta → planta

6.2.2 Dependencia funcional plena o completa

- En una DF *X* → *Y*, si el descriptor *X* está compuesto por varios atributos, se dice que *Y* tiene dependencia funcional **completa** o **plena** de *X*, si depende funcionalmente de *X* pero no depende de ningún subconjunto (propio) del mismo.
- Un ejemplo de DP plena puede ser la relación **PUBLICA** (artículo, revista, número, página), en la cual se desea mantener la semántica siguiente: el atributo página almacena la página inicial en la que comienza un artículo en una revista. Un mismo artículo puede aparecer publicado en distintas revistas y en distintos números (evidentemente, también en páginas distintas). También en la misma revista pero en números distintos. Una revista publica varios artículos en un mismo número. De ahí se deduce la siguiente DF plena:

  artículo, revista, número → página

- Un ejemplo de dependencia no plena puede ser:

idLibro, numSocio → editorial

- Esta dependencia no es plena, ya que idLibro → editorial. En este caso se dice que numSocio es un atributo *redundante*, *ajeno* o *extraño* a la dependencia

6.2.3 Dependencia funcional elemental

- Es una **dependencia funcional completa no trivial en la que el implicado es un atributo único**. Serán las dependencias que utilizaremos en el proceso de normalización
- Ejemplo:

determinante *implicado*

artículo, revista, número → página

↓

NO se puede **descomponer** en varias DF con determinante simple

- Cualquier dependencia funcional en la que el implicado sea **compuesto** puede descomponerse en las dependencias funcionales con el implicado simple que la forman. Por ejemplo:

$$\text{id\_piso} \rightarrow \text{ciudad, planta} \begin{cases} \text{id\_piso} \rightarrow \text{ciudad} \\ \text{id\_piso} \rightarrow \text{planta} \end{cases}$$

6.2.4 Dependencia funcional transitiva

- Sea la relación *R* en la que existen las siguientes dependencias funcionales entre los descriptores *X*, *Y*, *Z* :

$X \rightarrow Y$ $\quad$ $Y \not\rightarrow X$ $\quad$ $Y \rightarrow Z$

Se dice entonces que Z tiene una **dependencia transitiva** respecto de *X* a través de *Y*: $\quad$ ***X → Z***

- Ejemplo: Sea la relación **LIBRO** (idLibro, editorial, país). Suponiendo que una editorial publica en un único país se tienen las siguientes dependencias funcionales:

idLibro → editorial
editorial ↛ idLibro
editorial → país

Podemos decir que la dependencia idLibro → país es una dependencia transitiva a través de editorial.

6.3 Formas normales básicas

- El proceso de normalización puede provocar la aparición de nuevas relaciones que deben satisfacer dos condiciones fundamentales:
  - Las relaciones resultantes deben ser equivalentes a la relación original, y
  - Deben ser mejores en el sentido de haber alcanzado una forma normal superior al esquema de partida
- **Ej.:** consideremos el siguiente pedido de productos a un proveedor

Fecha: 12 de enero de 2006		**Pedido nº:** **Proveedor nº:** **Nombre del Proveedor:** **Dirección:**		002550 00115 Casi todo a 1€ Polígono Sur, s/n
Deseamos envíen:				
Nº Producto	**Descripción**	**Precio Unitario**	**Cantidad**	**Total**
969715	Taza de café	1.2 €	100	120 €
439124	Gafa de Sol	15 €	10	150 €
439126	Pañuelo de papel	0.6 €	500	300 €
			Importe Total	**570** €

- Se cumplen las siguientes **DF**:

  - númeroPedido → fecha
  - númeroPedido → numProveedor
  - númeroPedido → precioTotalPedido
  - númeroProveedor → nombreProveedor
  - númeroProveedor → direcciónProveedor
  - númeroProducto → descripProducto
  - númeroProducto → precioProducto
  - númeroPedido, númeroProducto → cantidadProducto
  - númeroPedido, númeroProducto → precioTotalProducto

- Se parte de un modelo relacional que recoge toda la semántica en una sola tabla:

PEDIDO (**númeroPedido**, fecha, numProveedor, nombreProveedor, direcciónProveedor, númeroProducto, descripProducto, precioProducto, cantidadProducto, precioTotalProducto, precioTotalPedido)
CP: númeroPedido

6.3.1 Primera forma normal (1FN)

- Una tabla está en primera forma normal si no tiene grupos repetitivos o **atributos multivalorados** ⇒ cada atributo toma **un solo valor** proveniente del dominio de dicho atributo
  - La 1FN se considera parte de la definición formal de relación
- En nuestro ejemplo existe un grupo repetitivo de productos por cada pedido. La eliminación del grupo repetitivo produce las siguientes tablas en 1FN:

PEDIDO (**númeroPedido**, fecha, numProveedor, nombreProveedor, direcciónProveedor, precioTotalPedido)

CP: númeroPedido

PRODUCTO (**númeroPedido**, **númeroProducto**, descripProducto, precioProducto, cantidadProducto, precioTotalProducto)

CP: (númeroPedido, númeroProducto)

CAj: númeroPedido → PEDIDO

- **Problemas**:
  - La información sobre un producto sólo se podrá almacenar si hay un pedido para ese producto, ya que el número de pedido forma parte de la clave principal
  - Este hecho genera anomalías en las operaciones de inserción, borrado y actualización

6.3.2 Segunda forma normal (2FN)

- Una tabla o relación está en 2FN si:
  - Está en 1FN, y
  - **Cada atributo no principal tiene dependencia funcional completa respecto de la clave primaria**. Se entiende por atributo principal o primo aquél que pertenece a la clave primaria
- En nuestro ejemplo
  - La tabla PEDIDO está en 2FN ya que su clave primaria es simple
  - La tabla PRODUCTO:
    - Los atributos *cantidadProducto* y *precioTotalProducto* dependen de ambos atributos de la clave
    - Los atributos *descripProducto* y *precioProducto* dependen ambos del atributo *númeroProducto* pero no de *númeroPedido*, por lo que no se cumple la definición de 2FN

Diseño en el modelo relacional: Teoría de la Normalización

- Para normalizar la tabla PRODUCTO, habrá que descomponerla en 2 tablas: una con los atributos que tienen dependencia completa y otra con el resto de los atributos, así, de tener:

  PRODUCTO (**númeroPedido, númeroProducto**, descripProducto, precioProducto, cantidadProducto, precioTotalProducto)

  CP: (númeroPedido, númeroProducto)

  CAj: númeroPedido → PEDIDOS

Pasamos a tener:

PRODUCTO (**númeroProducto**, descripProducto, precioProducto)

CP: númeroProducto

DETALLE (**númeroPedido**, **númeroProducto**, cantidadProducto, precioTotalProducto)

CP: (númeroPedido, númeroProducto)

CAj: númeroProducto → PRODUCTO

CAj: númeroPedido → PEDIDO

Para completar el esquema, tan sólo hay que añadir la relación PEDIDO

PEDIDO (**númeroPedido**, fecha, numProveedor, nombreProveedor, direcciónProveedor, precioTotalPedido)

CP: númeroPedido

Diseño en el modelo relacional: Teoría de la Normalización

- Los problemas planteados en la 1FN se resuelven con la 2FN, por lo que
  - La información sobre productos se mantiene en tablas separadas.
  - Por tanto, se podrán insertar, borrar y modificar productos sin necesidad de tener pedidos

- **Problemas**:
  - La tabla PEDIDO, aunque está en 2FN, plantea los mismos problemas de inserción, modificación y borrado:
    - No podemos insertar una información sobre proveedores a menos que hagan un pedido
    - Se perderá la información de un proveedor si se borra el único pedido que tiene
    - Cada tupla de pedido almacena información de los proveedores; de esta forma, si cambiase la información de un proveedor, habría que cambiarla en todas las tuplas

6.3.3 Tercera forma normal (3FN)

- Una tabla o relación está en 3FN si:
  - Está en 2FN, y
  - **Ningún atributo no primo depende transitivamente de la clave principal de la relación**.

- En nuestro ejemplo
  - En las tablas PRODUCTO y DETALLE no se produce ninguna dependencia transitiva, por tanto están en 3FN.
  - Analicemos la tabla PEDIDO:

    PEDIDO (**númeroPedido**, fecha, numProveedor, nombreProveedor, direcciónProveedor,

    precioTotalPedido)

    CP: númeroPedido
  - Algunas de las dependencias que existen son, entre otras:

    númeroPedido → numProveedor
    numProveedor ↛ númeroPedido
    numProveedor → nombreProveedor
  - Por tanto, existen dependencias transitivas, como: númeroPedido → nombreProveedor

Diseño en el modelo relacional: Teoría de la Normalización

- La normalización en 3FN será:

PEDIDO (**númeroPedido**, fecha, precioTotalPedido, numProveedor)
CP: númeroPedido
CAj: numProveedor → PROVEEDOR

PROVEEDOR (**numProveedor**, nombreProveedor, direcciónProveedor)
CP: numProveedor

Diseño en el modelo relacional: Teoría de la Normalización

- En resumen, de la tabla de partida

  PEDIDO (**númeroPedido**, fecha, numProveedor, nombreProveedor, direcciónProveedor, númeroProducto, descripProducto, precioProducto, cantidadProducto, precioTotalProducto, precioTotalPedido)
  CP: númeroPedido

- Mediante el proceso de normalización hemos obtenido las siguientes tablas:

PRODUCTO (**númeroProducto**, descripProducto, precioProducto)
CP: númeroProducto

DETALLE (**númeroPedido**, **númeroProducto**, cantidadProducto, precioTotalProducto)
CP: (númeroPedido, númeroProducto)
CAj: númeroProducto → PRODUCTO
CAj: númeroPedido → PEDIDO

PEDIDO (**númeroPedido**, fecha, precioTotalPedido, numProveedor)
CP: númeroPedido
CAj: numProveedor → PROVEEDOR

PROVEEDOR (**numProveedor**, nombreProveedor, direcciónProveedor)
CP: numProveedor

6.3.4 Forma normal de Boyce-Codd (FNBC)

- La FNBC es más estricta que la 3FN ya que:
  - Toda relación que esté en FNBC también está en 3FN
  - Sin embargo, una relación en 3FN, no está necesariamente en FNBC

- **Una tabla o relación está en FNBC si y sólo si, en las dependencias funcionales existentes, todo determinante es una clave candidata**

- En el ejemplo anterior se comprueba que todas las tablas obtenidas se encuentran en FNBC, ya que todos los determinantes son clave candidata

- En la práctica, casi todas las relaciones que están en 3FN también lo están en FNBC. Excepción:
  - Si en R existe una dependencia X → Y, donde X no es clave primaria e Y forma parte de la clave primaria ⇒ R está en 3FN pero no en FNBC

Diseño en el modelo relacional: Teoría de la Normalización

- Consideremos este ejemplo:

  - En una biblioteca se desean almacenar los artículos de investigación, junto con los datos de la revista en que aparezca. Supongamos que los títulos de los artículos no pueden repetirse en una misma revista, aunque un mismo artículo se puede publicar en distintas revistas
  - Podríamos tener la siguiente tabla:

    ARTICULO (idArtículo, título, revista, página)

  - Con las siguientes dependencias funcionales:

    idArtículo → título
    título → idArtículo
    idArtículo, revista → página
    revista, página → idArtículo

  - Puesto que idArtículo y título son equivalentes, también se dan las siguientes dependencias implícitas:

    título, revista → página
    revista, página → título

Diseño en el modelo relacional: Teoría de la Normalización

- Las claves candidatas serán:

  (idArtículo, revista)
  (título, revista)
  (revista, página)

- Se pueden producir anomalías de inserción, ya que podríamos insertar un artículo que tuviera el mismo título de uno ya existente, no cumpliéndose, por tanto, la dependencia idArtículo ↔ título. Ejemplo:

idArtículo	título	revista	página
i	t1	r1	x
…	…	…	…
j	t1	r2	y

Diseño en el modelo relacional: Teoría de la Normalización

- Como se puede ver, la tabla ARTICULO no se encuentra en FNBC ya que idArtículo y título son determinantes y, sin embargo, no son claves candidatas

- La solución consiste en dividir la tabla original **ARTICULO** (idArtículo, título, revista, página) en las siguientes dos tablas:

ARTICULO (idArtículo, título)
CP: idArtículo
Único: título

LOCALIZACION (revista, página, idArtículo)
CP: (revista, página)
Único: (idArtículo, revista)
CAj: idArtículo → ARTICULO
VNN: idArtículo