Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

NORMALIZACIÓN

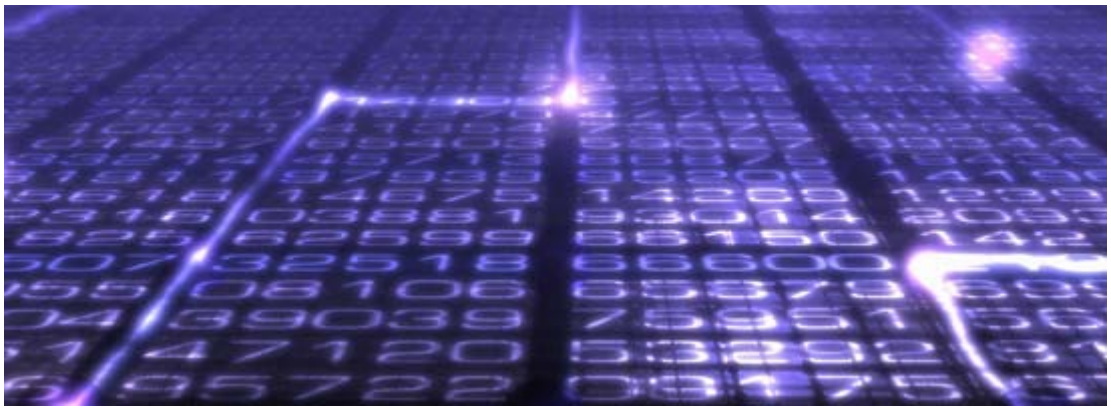
NORMALIZACIÓN BASADA EN DEPENDENCIAS FUNCIONALES

Índice

Presentación

Ya sabemos que en el proceso de normalización se representan las restricciones semánticas del mundo real mediante dependencias funcionales, multivaluadas y de reunión. El resultado es un esquema lógico de la base de datos que cumple las propiedades deseables de un buen diseño de base de datos: el **esquema relacional normalizado**.

Anteriormente vimos que para conseguirlo puede ser necesario descomponer las relaciones iniciales. Estas descomposiciones deben **conservar las dependencias** y **ser de producto sin pérdidas**. En este tema, trataremos las situaciones representadas por dependencias funcionales y en el siguiente por las dependencias multivaluadas y las dependencias de reunión.



En este tema aprenderás a:

- Identificar las situaciones en las que es necesario descomponer una relación.
- Descomponerlas para que sean sin pérdidas y, en la medida de lo posible, conserven las dependencias.

Acabaremos la unidad presentado un ejercicio resuelto que ilustre todo el proceso y que sirva de guía para la resolución de ejercicios.

Las formas normales

Las **formas normales** son condiciones que se establecen a una relación para verificar que está bien diseñada y, como consecuencia, que tiene un buen comportamiento ante las operaciones de actualización de la base de datos.

Si una relación no cumple un requisito, deberá descomponerse en otras que sí cumplan individualmente los requisitos.

1FN, 2FN Y 3FN

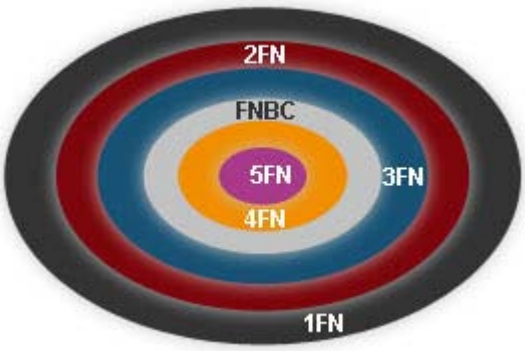
Existen diferentes formas normales. Inicialmente se propusieron tres: 1FN, 2FN y 3FN definidas por Codd en 1972. Analizan las relaciones basándose en las **claves candidatas** de éstas y las dependencias funcionales que se establecen entre sus **atributos**. Como resultado del análisis de estas formas normales obtenemos un esquema perfecto siempre que haya una sola clave candidata.

FNBC

En 1974 Boyce y Codd redefinieron la 3FN que manejaba de forma satisfactoria las relaciones en las que existía más de una clave candidata.

4FN y 5FN

La 4FN y 5FN son debidas a Fagin (1977 la primera y 1979 la segunda). Estas formas normales están basadas en el **estudio de dependencias no funcionales**: multivaluadas (DMV) en el primer caso y de reunión (DR) en el segundo.



La imagen representa las formas normales según lo estrictas que son las restricciones que definen. Si una relación está en 2FN es porque cumple las restricciones de la 2FN y de la 1FN y así sucesivamente. Esto implica que, para que una relación esté en 5FN, debe estar obligatoriamente en las 5 anteriores.

Más sobre las formas normales

En detalle

Más sobre las formas normales

La única **forma normal crítica** para el modelo relacional es la 1FN, por tanto, las restricciones que define ésta deben ser siempre seguidas. El resto de formas normales son opcionales, pero avanzar en el proceso de normalización ayuda a evitar los problemas de actualización. Es recomendable alcanzar al menos la tercera forma normal (3FN). Seguir avanzando en el proceso de normalización debe ser estudiado por el administrador de la base de datos para decidir si la descomposición obtenida mejora la situación.

1FN

La siguiente relación almacena la información de los jugadores de fútbol. La clave de esta relación es

#jugador	nombreJug	#equipo	nombreEquipo	alta	baja	presupuesto	#presidente	nombrePresi
J24	romaric	eq12	rennes	1-7-2000	1-9-2002	40	p040	gougle
J24	romaric	eq12	rennes	1-7-2000	1-9-2002	40	p040	gougle
J24	romaric	eq76	cordoba	2-9-2002	1-7-2006	30	p093	orizaola
J372	rafa lozano	eq76	cordoba	1-7-2000	1-8-2002	30	p093	orizaola
J372	rafa lozano	eq98	villarreal	1-9-2002	1-8-2004	45	p054	roig
J372	rafa lozano	eq35	real madrid	1-9-2004	1-8-2006	120	p921	perez

En este caso los atributos **#equipo**, **estadio**, **alta**, **baja**, **sueldo**, **#presidente** y **nombrePresi** tienen varios valores para cada jugador, que almacenan la información de todos aquellos equipos a los que ha pertenecido el jugador.

Una relación R está en primera forma normal (1FN) si y sólo si cada atributo de una determinada tupla tiene un único valor.

Dicho de otro modo: esto se cumple si cada celda de la relación tiene sólo un valor. Para pasar una relación a 1FN disponemos de 2 alternativas:

- **Rellenar los datos no repetitivos** (#jugador y nombreJugador) para garantizar que cada atributo tiene un único valor dentro de cada tupla o fila. Esta opción da lugar a varios datos redundantes, que pueden derivar en problemas de actualización ya vistos anteriormente. La relación sería la que se muestra a continuación y la clave **#jugador** y el **#equipo**.

Tabla resultante

- Para **evitar las redundancias** llevamos los atributos repetitivos y la clave de los no repetitivos a una nueva tabla, teniendo así ahora dos relaciones: **jugador** (#jugador, nombreJug) y **equipo** (#equipo, estadio, alta, baja, sueldo, #presidente, nombrePresi).

Problema con las dependencias parciales

Siguiendo con el 1FN, supongamos que escogemos la primera opción (**rellenar los datos no repetitivos**):

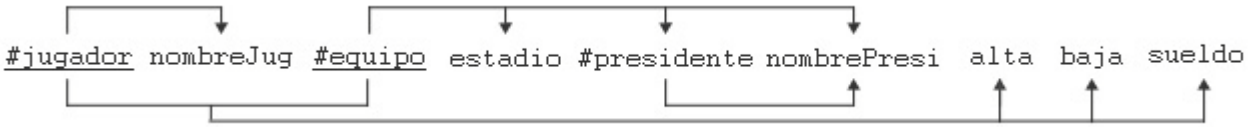
Jugador (#jugador, nombreJug, #equipo, estadio, alta, baja, sueldo, #presidente, nombrePresi)

Analicemos cuáles son las **dependencias funcionales** que se dan en esta relación.

Dependencias funcionales

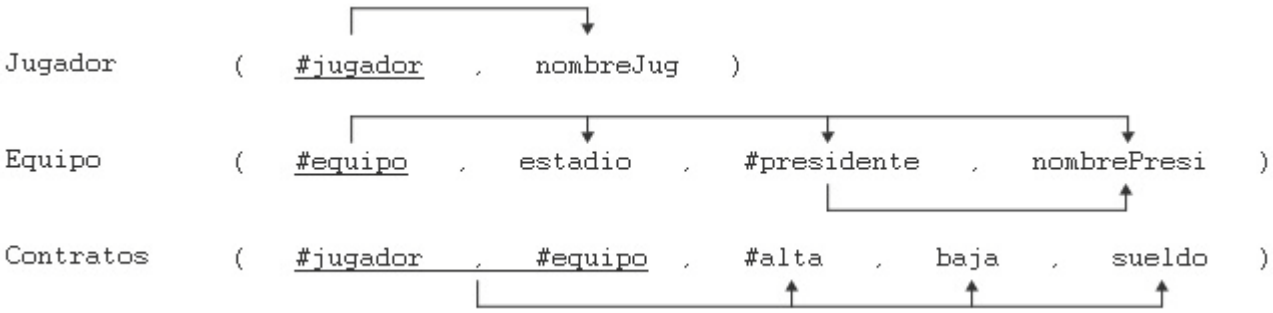
- #jugador ⟶ nombreJug
- #equipo ⟶ estadio, presupuesto, #presidente, nombrePresi
- #jugador,#equipo ⟶ alta, baja, sueldo
- #Presidente ⟶ nombrePresi

Las DF que se establecen entre sus atributos podemos verlas en el siguiente diagrama:



Fijémonos en la DF #equipo ⟶ estadio, presupuesto, #presidente, nombrePresi. Analicemos los **problemas** de actualización derivados de esta dependencia funcional: no podemos introducir en la base de datos la información de un equipo hasta que no fiche algún equipo; si un equipo solo ha fichado a un jugador y eliminamos este jugador de la base de datos, perdemos también el resto de información del equipo (estadio, presupuesto,…) y, por último, como la información del equipo aparece muchas veces, la modificación de algún dato de éste puede producir incoherencias (y un retardo considerable).

La **solución** a estos problemas es **descomponer la relación anterior** como se indica en este diagrama:



Solución

De forma intuitiva hemos agrupado la información de los **jugadores** por un lado, la de los **equipos** por otro y la de los **contratos** por otro. Nos hemos quitado los problemas de actualización anteriores y no perdemos información puesto que la tabla original se puede recomponer realizando el producto natural de estas tres tablas.

Definiciones previas

- **Atributo primo**: aquel que pertenece a la clave primaria.
- **DF completa**: dada la DF X ⟶ Y, se dice que Y depende funcionalmente de manera completa de X, si depende funcionalmente de X pero no de algún subconjunto propio de X.

Una relación R está en 2FN si está en 1FN y si sus atributos **no primos** dependen de forma completa de la clave principal.

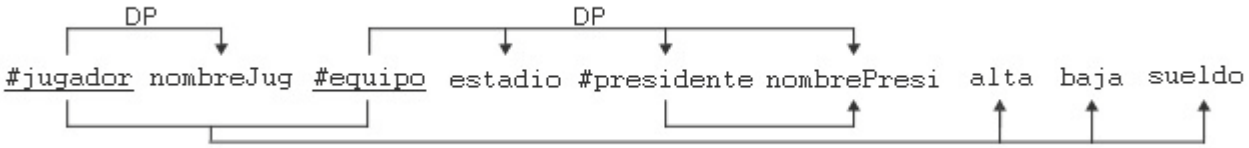
Para que una relación esté en 1FN y no esté en 2FN debe tener una clave primaria compuesta. Una relación que esté en 1FN y no esté en 2FN siempre puede reducirse a un conjunto equivalente de tablas en 2FN.

Para asegurarnos de que la descomposición a realizar sea sin pérdidas, es decir, que podemos obtener la relación original realizando el producto natural de las relaciones descompuestas, seguimos la regla:

Si existe una DF X ⟶ Y no completa (**dependencia parcial**) en R (relación inicial):

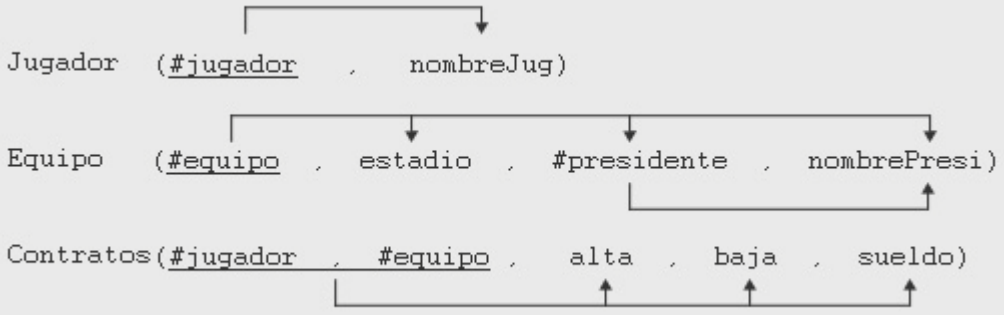
- Se crea una nueva relación resultante de eliminar de R los atributos parcialmente dependientes: $R_0 = R–Y$.
- Se crea una nueva relación R_1 con estos últimos, donde también se añade el determinante de la DF (que no abandona la relación original): $R_1 = \{Y \cup X\}$

Las DF de la base de datos de fútbol no completas se muestran en la siguiente figura:



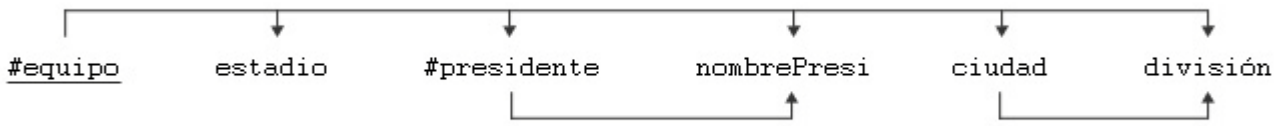
Creamos una tabla que contenga cada dependencia parcial (**DP**) y otra con las claves más el resto de atributos, de esta forma ya no existen dependencias parciales, como se ve en el siguiente **gráfico**.

Gráfico


Jugador (#jugador , nombreJug)
Equipo (#equipo , estadio , #presidente , nombrePresi)
Contratos(#jugador , #equipo , alta , baja , sueldo)

Problema con las dependencias transitivas

Vamos a representar en nuestra base de datos el hecho de que cada equipo pertenece a una ciudad y supondremos que, para que un equipo sea de primera división, la ciudad tiene que ser una capital de provincia. Hay que añadir dos atributos a la tabla de equipo: **ciudad**, **división**. Y la dependencia funcional: ciudad ⟶ división. La tabla **equipo** con sus DF es:



Esta relación contiene problemas derivados de la independencia mutua entre sus atributos no primos. Las dependencias #presidente ⟶ nombrePresi y ciudad ⟶ división son perjudiciales y se denominan **dependencias transitivas** (DT).

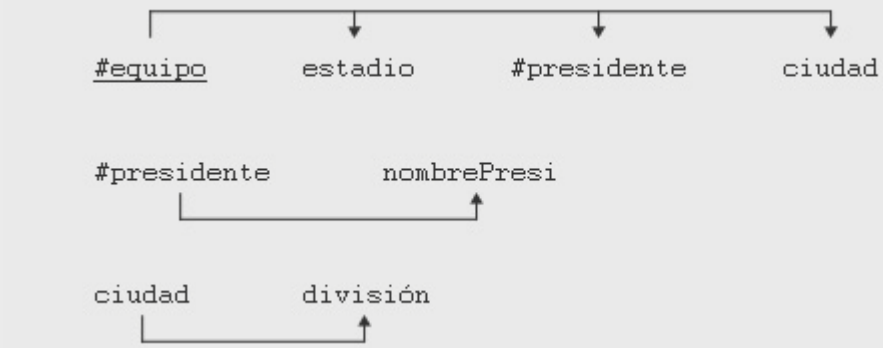
Analicemos los problemas de actualización en la relación **equipo** derivados de la DT ciudad ⟶ división:

- No podemos introducir la división de una determinada ciudad hasta que no tengamos algún equipo de esta ciudad en la liga de fútbol.
- Si eliminamos el único equipo de una ciudad, también eliminamos la información de la división a la que pertenece dicha ciudad.
- La división de una ciudad aparece repetida tantas veces como equipos tenga dicha ciudad. Para modificar la división de una ciudad habrá que localizar todas las tuplas en las que aparece, lo que provoca incrementar la posibilidad de incoherencias y añadir un retardo considerable a la operación.

La solución pasa por quitar las **DT** de la relación **equipo**. De esta forma se resuelven los problemas mencionados. Quedaría el siguiente **esquema**.

Gráfico

Solución



3FN

Una relación R está en 3FN si está en las anteriores formas normales, y todos los atributos no primos dependen de manera no transitiva de la clave primaria.

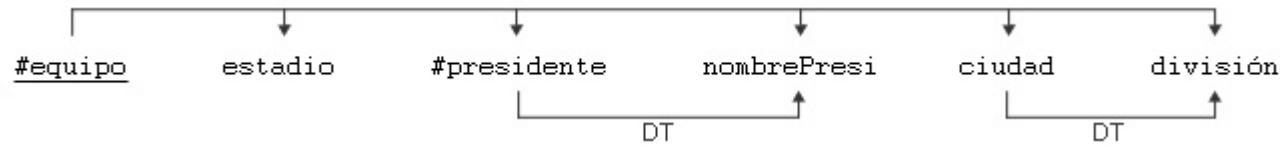
Dicho de otro modo: una relación está en 3FN **si no existen dependencias transitivas** entre sus atributos no primos. Para pasar una relación en 2FN a 3FN habrá que eliminar las dependencias transitivas. Para ello se descompone la relación R en otras dos relaciones a partir de la DF transitiva X $\longrightarrow$ Y:

- $R_0 = R - Y$
- $R_1 = X \cup Y$

Realizar la descomposición de esta forma nos permite:

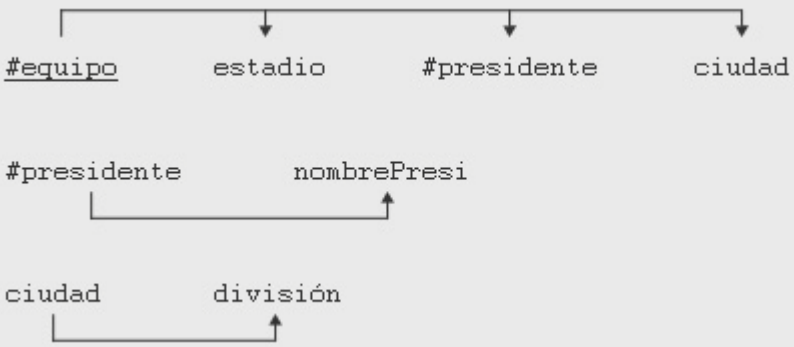
- Poder recuperar la relación original: es una **descomposición sin pérdidas**.
- Mantener las DT dentro de una relación y, por tanto, la información de la restricción semántica del mundo real que representa, lo que nos permite no perder información. Así, la descomposición **conserva las dependencias**.

Para descomponer la tabla:



tendríamos que crear una tabla para cada dependencia transitiva y otra con el resto de atributos más los determinantes de las **relaciones transitivas**.

Relaciones transitivas



De esta forma evitamos los problemas derivados de las dependencias transitivas: ahora, por ejemplo, puedo insertar la división a la que pertenece una determinada ciudad sin necesidad de que tenga un equipo de fútbol. Además, conseguimos mantener esta información en la base de datos.

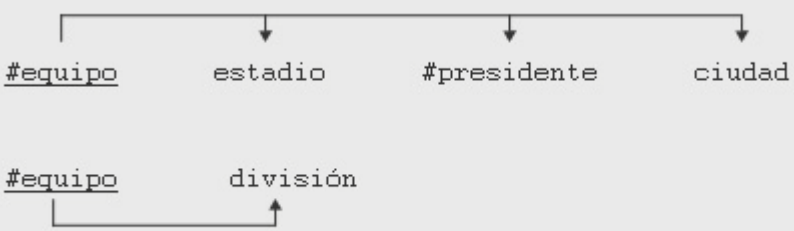
Gráfico

Descomposición en 3FN

Gráfico

Descomposición en 3FN

Si la descomposición hubiese sido la siguiente, seguiría estando en 3FN pero ahora no se representa en nuestra base de datos el hecho de que todas las capitales de provincia son las que están en primera división.



2FN y 3FN generales

La 2FN y 3FN se basan en el análisis de **dependencias parciales** o **transitivas** sobre la **clave principal** de la relación. Las definiciones de la 2ª y 3ª forma normal pueden generalizarse si se tienen en cuenta las **claves candidatas** en lugar de sólo la clave principal.

Puede generalizarse también el concepto de atributo primo, redefiniéndolo como aquel que pertenece a una clave candidata.

Así pues, podría decirse que una relación está en:

2FN si está en 1FN y si sus atributos no primos dependen de forma completa de cualquier clave candidata. 3FN si está en las anteriores formas normales, y si ningún atributo no primo depende transitivamente de cualquiera de las claves candidatas.

Usar las **definiciones generales** puede hacer más compleja la normalización, ya que hay más restricciones que analizar. Sin embargo, pueden identificarse redundancias que quedarían ocultas de usar sólo las claves primarias.

Ejercicio resuelto: planteamiento

La empresa *Fromliotte* dedicada a la consultoría informática tiene implementado el siguiente esquema relacional. Esta base de datos gestiona la asignación de tareas derivadas de los proyectos de desarrollo que tiene en marcha. El diseñador de la base de datos ha decidido introducir toda la información que tiene que almacenar en una sola tabla como esta:

ASIGNACION_TAREAS (dni_emp, nombre_emp, dirección_emp, cargo_emp, salario_emp, tarea, tipo_tarea, descripción_tarea, categoría_tarea, coste-hora, fecha-inic, fecha-fin, estado, proyecto, estimación-horas)

El significado es el siguiente: el empleado identificado por **dni**, realizará la tarea **tarea**, lo que le va a llevar una dedicación de estimación-horas **horas**. La fecha de fin indica el día en el que como máximo la tarea tiene que estar finalizada y el estado indica en qué situación se encuentra: en proceso, acabada ó cancelada.

Las restricciones semánticas que se cumplen son:

- Existen 3 tipos de tareas: planificación, diseño y desarrollo. Los jefes de proyecto solo desarrollan tareas de tipo planificación, los programadores senior realizan las tareas de tipo diseño y los junior las de tipo desarrollo.
- El coste-hora de la tarea está definido para cada categoría.

Queremos obtener un esquema equivalente al anterior pero que esté en 3FN. Para ello extraemos del enunciado del problema el conjunto de dependencias funcionales que se cumplen. Del propio significado de la relación se extraen las siguientes:

- Dni_emp $\longrightarrow$ nombre_emp, dirección_emp, cargo_emp, salario_emp.
- Tarea $\longrightarrow$ tipo_tarea, descripción_tarea, categoría_tarea, coste_hora, fecha_inic, fecha_fin, estado, proyecto
- Dni_emp, tarea $\longrightarrow$ estimación-horas

Derivadas de las **restricciones semánticas**.

Dependencias funcionales

- cargo_emp $\longrightarrow$ tipo_tarea
- categoría_tarea $\longrightarrow$ coste_hora

Ejercicio resuelto: solución 2FN

Análisis de 1FN

Al ser la clave primaria de la relación *dni_emp, tarea* se deduce que cada atributo de la relación contendrá un valor único. La relación, por tanto, se encuentra en 1FN.

Análisis de 2FN

Empezamos el análisis de las **dependencias funcionales**. Para que una relación esté en 2FN debe ocurrir que todo atributo no primo dependa de forma completa de las claves candidatas. El primer paso, por tanto, es detectar cuáles son los atributos no primos y, por tanto la existencia de claves candidatas en la relación.

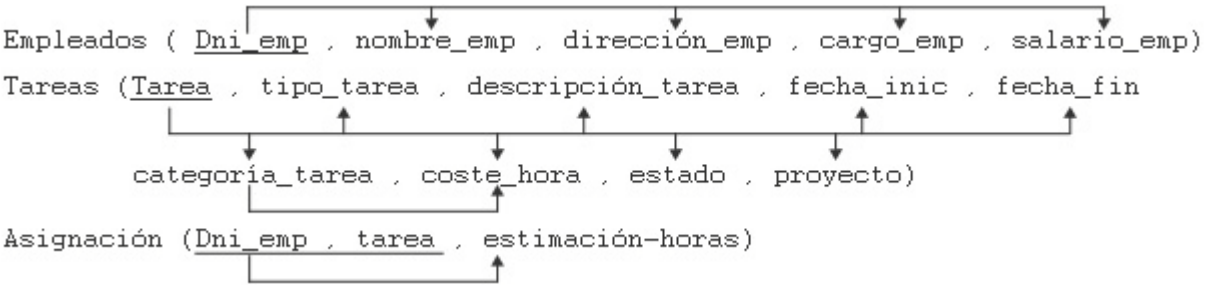
Teniendo en cuenta el conjunto de DF obtenidas en el apartado anterior, el único conjunto de atributos que determina funcionalmente al resto de la relación *asignación_ tareas* es *dni_emp, tarea*. La clave primaria de la relación es, por tanto *dni_emp, tarea* y no existen claves candidatas.

El siguiente paso es detectar la existencia de **dependencias parciales**. En nuestro caso son:

Dni_emp $\longrightarrow$ nombre_emp, dirección_emp, cargo_emp, salario_emp.

Tarea $\longrightarrow$ tipo_tarea, descripción_tarea, categoría_tarea, coste_hora, fecha_inic, fecha_fin, estado, proyecto

La relación no está, por tanto en 2FN. Para descomponerla sin pérdidas creamos una relación para cada DT y otra para el resto de los atributos más los determinantes de las DT. Como se puede comprobar en el siguiente diagrama las relaciones ya no tienen ninguna dependencia parcial por lo que este esquema se encuentra en 2FN.

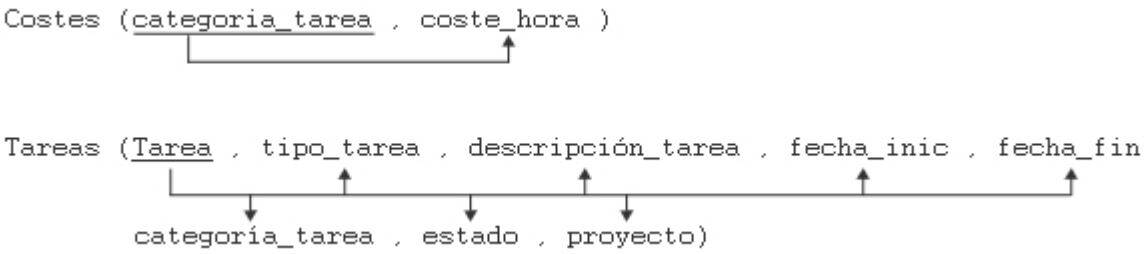


Ejercicio resuelto: solución 3FN

Análisis 3FN

Hay que comprobar que las tres relaciones se encuentran en 3FN, es decir, si todo atributo no primo depende de forma no transitiva de la clave primaria de la relación.

Del esquema de la pantalla anterior donde se representan las relaciones con las DF que se cumplen en cada una de ellas se deduce que empleado y asignación están en 3FN. Sin embargo la DF categoría_tarea ⟶ coste_hora es una dependencia transitiva que hace que tengamos que descomponer la relación **tarea** para ponerla en 3FN. Creamos una tabla para la DT y otra con el resto de los atributos más el determinante de la DT:



Obtenemos así un esquema en 3FN puesto que ahora no existen dependencias transitivas. El esquema resultado es:

- Empleados (Dni_emp, nombre_emp, dirección_emp, cargo_emp, salario_emp)
- Tareas(Tarea, tipo_tarea, descripción_tarea, categoría_tarea, fecha_inic, fecha_fin, estado, proyecto)
- Asignación (Dni_emp, tarea, estimación-horas)
- Costes (categoría_tarea, coste_hora)

Comentarios

En detalle

Comentarios

¿Conserva esta descomposición las dependencias?

La DF cargo_emp ⟶ tipo_tarea ha desaparecido del esquema por lo que esta DF no se conserva. El administrador de la base de datos debería ahora analizar si es más conveniente evitar las redundancias que se producían antes ó mantener la DF anterior.

Resumen

El proceso de **normalización** es un método sistemático para obtener un esquema de relaciones que evite los problemas de actualización debidos a las **redundancias** que se producen en las tablas. Está compuesto de varias FN: 1FN, 2FN, 3FN, FNBC, 4FN, 5FN.

Las **formas normales** (FN) son **autocontenidas**: una relación que esté en 5FN lo está en todas las anteriores. Lo contrario no es cierto.

Una relación que contiene **valores únicos** en todos sus atributos está en 1FN. Si no es el caso hay que descomponerla en dos: una que contenga los atributos del grupo repetitivo más la clave de la relación original y otra que contenga el resto de los atributos de R.

X $\longrightarrow$ Y es una **dependencia parcial** (DP) si X es un subconjunto de la clave primaria. Una relación está en 2FN si está en 1FN y no contiene dependencias parciales.

X $\longrightarrow$ Y es una **dependencia transitiva** (DT) si X e Y no son atributos primos, es decir, pertenecientes a una clave de la relación (primaria ó candidata). Una relación está en 3FN si está en 2FN y no contiene dependencias transitivas.

Cuando encontremos una **DF perjudicial** X $\longrightarrow$ Y en una relación que impida que ésta esté en una determinada FN tenemos que descomponerla en dos tablas: una que contenga los atributos X e Y, y otra que contenga el resto de los atributos de la relación original más X. De esta forma nos aseguramos de que la descomposición es sin pérdidas y la DF X $\longrightarrow$ Y se sigue conservando en el nuevo esquema.