Profesora: Olga Cuervo Miguélez

TEMA 5

DISEÑO CONCEPTUAL. NORMALIZACIÓN

Módulo: **Bases de Datos**

Ciclo: **DAM**

15. Normalización

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

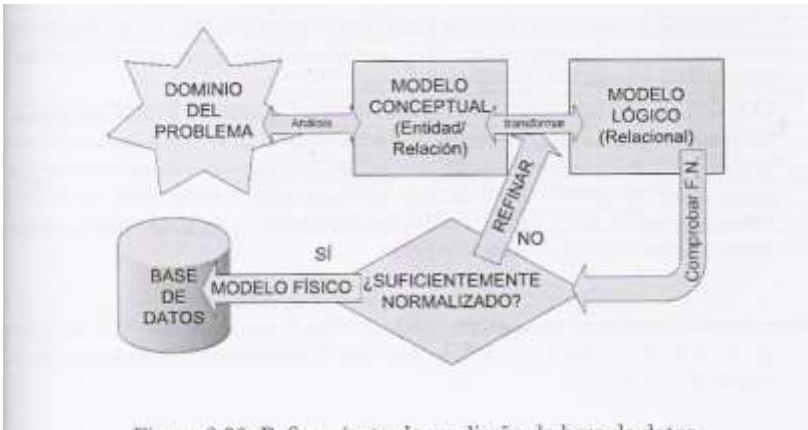


Figura 2.36: Refinamiento de un diseño de base de datos.

¿Crees que tu base de datos ya podría implementarse directamente sobre el SGBD relacional elegido?

Una vez obtenido el esquema relacional resultante del modelo entidad relación que representaba la base de datos, la respuesta sería afirmativa ya que normalmente tendremos una buena base de datos. Pero si queremos que nuestra base esté correctamente diseñada y en el futuro funcione con plena fiabilidad, es necesario que testemos el esquema lógico obtenido para comprobar que sus tablas estén normalizadas y no se van a producir inconsistencias ni anomalías al manipular los datos. Esta comprobación la realizaremos mediante el **proceso de normalización**. Debido a posibles fallos en el diseño o a problemas indetectables en esta fase del diseño, tendremos un esquema que puede producir una base de datos que incorpore estos problemas:

15.1. Problemas del Esquema Relacional

- **Redundancia**. Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos. Cuando es excesiva es evidente que el diseño hay que revisarlo, es el primer síntoma de problemas y se detecta fácilmente.
- **Ambigüedades**. Datos que no clarifican suficientemente el registro al que representan. Los datos de cada registro podrían referirse a más de un registro o incluso puede ser imposible saber a qué registro exactamente se están refiriendo. Es un problema muy grave y difícil de detectar.
- **Pérdida de restricciones de integridad.** Normalmente debido a dependencias funcionales. Más adelante se explica este problema. Se arreglan fácilmente siguiendo una serie de pasos concretos.

- **Anomalías en operaciones de modificación de datos.** El hecho de que al insertar un solo elemento suponga repetir tuplas en una tabla para variar unos pocos datos. O que eliminar un elemento suponga eliminar varias tuplas necesariamente (por ejemplo que eliminar un cliente suponga borrar seis o siete filas de la tabla de clientes, sería un error muy grave y por lo tanto un diseño terrible).

- El **principio fundamental** reside en que las tablas deben referirse a objetos o situaciones muy concretas, relacionados exactamente con elementos reconocibles por el sistema de información de forma inequívoca. Cada fila de una tabla representa inequívocamente un elemento reconocible en el sistema. Lo que ocurre es que conceptualmente es difícil agrupar esos elementos correctamente. En cualquier caso **la mayor parte de problemas se agravan si no se sigue un modelo conceptual y se decide crear directamente el esquema relacional. En ese caso, el diseño tiene una garantía casi asegurada de funcionar mal.**

- Cuando aparecen los problemas enumerados, entonces se pueden resolver usando reglas de normalización. Estas reglas suelen forzar la división de una tabla en dos o más tablas para arreglar ese problema.

La **normalización** es un proceso que consiste en asignar atributos a las entidades verificando que se cumplan ciertas reglas. Reduce las redundancias de datos y ayuda a eliminar las anomalías que se derivan de las redundancias.

La **normalización** se basa en lograr la independencia de los datos con respecto a las aplicaciones que los usan, obteniendo unas tablas con una estructura óptima y eficaz, de manera que se optimizan los procesos de inserción, modificación y borrado en la base de datos.

Veremos que el proceso de normalización se basa en el análisis de las dependencias entre atributos. Para ello se tendrá en cuenta los conceptos de: **dependencia funcional, dependencia funcional completa, dependencia funcional elemental, dependencia funcional trivial, dependencia transitiva, dependencia multivaluada** y **dependencia de unión**. Estos conceptos los veremos en el siguiente apartado.

¿Y cómo se aplica la normalización? Es un proceso que se realiza en varias etapas secuenciales. Cada etapa está asociada a una **forma normal,** que establece unos requisitos a cumplir por la tabla sobre la que se aplica. Existen varias formas normales: **Primera, Segunda, Tercera, Boyce-Codd, Cuarta, Quinta y Dominio-Clave**. Como hemos indicado, el paso de una forma normal a otra es consecutivo, si no se satisface una determinada forma normal no puede pasarse al análisis de la siguiente. Según vamos avanzando en la normalización, los requisitos a cumplir serán cada vez más restrictivos, lo que hará que nuestro esquema relacional sea cada vez más robusto.

Para referirnos a algunas de estas formas normales se suele usar la abreviatura siguiente: **1FN, 2FN, 3FN, BCFN, 4FN, 5FN**

Como norma general, para garantizar que no existan problemas en la actualización de datos, es recomendable aplicar el proceso de normalización hasta **Tercera Forma Normal** o incluso hasta **Forma Normal de Boyce-Codd**.

En algunas aplicaciones, sin embargo, llegaremos hasta la 4FN, y en aplicaciones muy especializadas de investigación estadística será necesario llegar más allá.

Este proceso se basa en la descomposición sin pérdida de las tablas que están en una forma normal inferior, obteniendo una forma normal superior. Se descompone la tabla en otras con menor cantidad de atributos, sin que haya pérdida de información.

En los siguientes apartados se describen las características y requisitos de cada una de las formas normales.

15.2. Dependencias Funcionales

Para aplicar correctamente la normalización es necesario partir del concepto de dependencia.

La dependencia es un conjunto de restricciones que se imponen a determinados atributos de las tablas.

Se denominan **dependencias** a las relaciones que existen entre los atributos en el mundo real y que son recogidas en el modelo lógico de la base de datos.

Vamos a estudiar los diferentes **tipos de dependencias**:

- Dependencia funcional.
  - Dependencias funcional completa.
  - Dependencia funcional elemental.
  - Dependencia funcional trivial.
- Dependencia transitiva.
- Dependencia multivaluada.
- Dependencia de unión.

15.3. Formas Normales

Las formas normales se corresponden a una **teoría de normalización** iniciada por el propio Codd y continuada por otros autores (entre los que destacan Boyce y Fagin). Codd definió en 1970 la primera forma normal, desde ese momento aparecieron la segunda, tercera, la Boyce-Codd, la cuarta y la quinta forma normal.

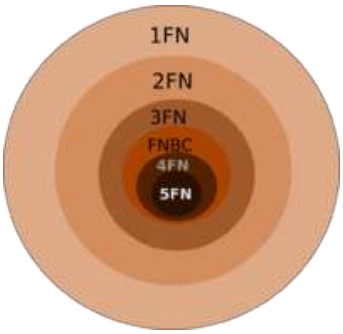
Una tabla puede encontrarse en primera forma normal y no en segunda forma normal, pero no al contrario. Es decir los números altos de formas normales son más restrictivos (la quinta forma normal cumple todas las anteriores).

La teoría de formas normales es una teoría absolutamente matemática, pero en este tema se describe de forma más intuitiva.

Hay que tener en cuenta que muchos diseñadores opinan que basta con llegar a la forma Boyce-Codd, ya que la cuarta, y sobre todo la quinta forma normal, es polémica. Hay quien opina que hay bases de datos peores en quinta forma normal que en tercera. En cualquier caso debería ser obligatorio para cualquier diseñador llegar hasta la tercera forma normal.

Las formas normales pretenden alcanzar 2 **objetivos**:

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



15.3.1. Primera forma normal (1FN)

Es una forma normal inherente al esquema relacional. Es decir toda tabla realmente relacional la cumple. Se dice que una tabla se encuentra en primera forma normal **si impide que un atributo de una tupla pueda tomar más de un valor.**

EJEMPLO

TRABAJADORES		
DNI	**Nombre**	**Departamento**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección Gestión

Visualmente es una tabla, pero no una tabla relacional (lo que en terminología de bases de datos relacionales se llama **relación**). No cumple la primera forma normal. Estaría primera forma normal si los datos fueran:

SOLUCION

TRABAJADORES		
DNI	**Nombre**	**Departamento**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección
12345345G	Andrea	Gestión

Esa tabla sí está en primera forma normal.

15.3.2. Dependencias Funcionales

Dependencia Funcional

DEFINICIÓN: Se dice que un conjunto de atributos (**Y**) depende funcionalmente de otro conjunto de atributos (**X**) si para cada valor de **X** hay un único valor posible para **Y**. Simbólicamente se denota por **X→Y**. Y se lee: **X determina a Y**.

Dicho de un modo más coloquial, las **Dependencias Funcionales (DF)** se producen cuando tenemos una tabla con una serie de atributos. Se dice que un atributo tiene dependencia funcional de otro cuando a cada valor del primero le corresponde un solo valor del segundo.

EJEMPLO

El nombre de una persona depende funcionalmente del DNI (*DNI→Nombre*), para un DNI concreto sólo hay un nombre posible. En la tabla ejemplo anterior, el departamento no tiene dependencia funcional del DNI,

(*DNI* $\not\rightarrow$ *Departamento*), ya que para un mismo DNI puede haber más de un departamento posible.

Al conjunto **X** del que depende funcionalmente el conjunto **Y** se le llama **determinante**. Al conjunto Y se le llama **implicado**.

Las DP pueden ser:

Dependencia Funcional Completa

Un conjunto de atributos (**Y**) tiene una dependencia funcional completa sobre otro conjunto de atributos (**X**) si **Y** tiene dependencia funcional de **X** y además no se puede obtener de **X** un conjunto de atributos más pequeño (subconjunto) que consiga una dependencia funcional de **Y** (es decir, no hay en **X** un determinante formado por menos atributos).

Dicho de otra forma, cuando un atributo depende de otro que es un atributo compuesto (un conjunto de atributos), se dice que la dependencia funcional es completa si el atributo dependiente no depende de ningún subconjunto del atributo compuesto.

EJEMPLO

En una tabla de clientes, el conjunto de atributos formado por el **nombre** y el **dni** producen una dependencia funcional sobre el atributo **apellidos** *(dni,nombre→apellidos)*. Pero esta DP no es plena o completa ya que el **dni** individualmente, también produce una dependencia funcional sobre **apellidos** *(dni→apellidos)*.

El **dni** sí produce una dependencia funcional completa sobre el campo **apellidos** *(dni ⇒ apellidos)*.

CLIENTES		
DNI	**Nombre**	**Apellidos**
12121212A	Andrés	Pérez
12345345G	Andrea	Domínguez

Luego podemos decir que, **DNI** produce una dependencia funcional completa sobre los campos **nombre** y **apellidos** *(dni ⇒ nombre,apellidos).*

Una dependencia funcional completa se denota como **X⇒Y**

Dependencia Funcional Elemental

Se produce cuando X e Y forman una dependencia funcional completa y además Y es un único atributo.

Dependencia Funcional Trivial

Es la dependencia que tiene un atributo con relación a otro compuesto cuando forma parte de él. Se produce cuando Y es un subconjunto de X.

autor, cod → autor

alumno, asignatura, nota → nota

libro → libro

Dependencia Funcional Transitiva

Para explicar esta dependencia tenemos que incluir los conceptos atributos principales y atributos no principales:

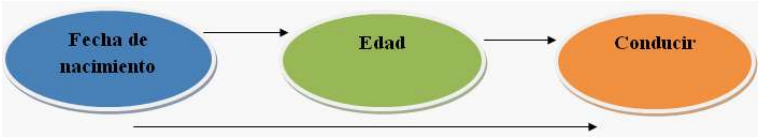
Atributos Principales y Atributos No Principales

Un **atributo** es **principal (AP)** cuando forma parte de alguna de las claves candidatas de la relación.

Un **atributo** es **no principal (ANP)** cuando no forma parte de ninguna clave candidata.

Se produce cuando tenemos tres conjuntos de atributos **X**, **Y** y **Z**. **Y** depende funcionalmente de **X** (**X→Y**), **Z** depende funcionalmente de **Y** (**Y→Z**). Además **X** no depende funcionalmente de **Y** (**Y ↛ X**). Entonces ocurre que **X** produce una dependencia funcional transitiva sobre **Z**. Esto se denota como: (**X– →Z**).

Es decir, es una dependencia de un atributo no principal de otro no principal.



EJEMPLO

Clase-Tutor-Dpto		
(X)numClase	**(Y)codTutor**	**(Z)codDepartamento**
11	Luz	Fol
21	Luz	Fol
22	Olga	Informática
23	Olga	Informática

Sí:

- ✓ *numClase→codTutor (pq cada clase sólo tiene un tutor),* es decir, **codTutor** depende funcionalmente de **numClase**.
- ✓ *codTutor→codDepartamento (pq cada tutor sólo pertenece a un departamento),* es decir, **codDepartamento** depende funcionalmente de **codTutor**.
- ✓ *codTutor* ↛ *numClase (pq un tutor es tutor de varias clases),* es decir, **numClase** NO depende funcionalmente de **codTutor**.

Entonces, **X-→Z** (el **codDepartamento** depende transitivamente de **numClase**).

15.3.3. Segunda forma normal (2FN)

Ocurre si una tabla está en **primera forma normal** y **además cada atributo que no sea clave, depende de forma funcional completa respecto de cualquiera de las claves.** Toda la clave principal debe hacer dependientes al resto de atributos, si hay atributos que depende sólo de parte de la clave, entonces esa parte de la clave y esos atributos formarán otra tabla.

EJEMPLO

ALUMNOS				
DNI	**codCurso**	**nombre**	**apellido1**	**nota**
12121219A	34	Pedro	Valiente	9
12121219A	25	Pedro	Valiente	8
34517775G	34	Ana	Fernández	6
56745378J	25	Sara	Crespo	7
56745378J	34	Sara	Crespo	6

Clave Primaria (**PK**): {DNI, codCurso}

Atributos Principales (**AP**): DNI, codCurso

Atributos No Principales (**ANP**): nombre, apellido1, nota.

Dependencias Funcionales (**DF**):

- ✓ DF1: *DNI→nombre, apellido1* (depende de forma PARCIAL de la clave).

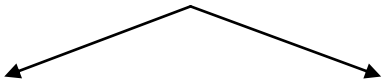
- ✓ DF2: *DNI,codCurso* → *nota* (depende de forma COMPLETA de la clave).

¿Está la tabla ALUMNOS en 2FN?

La tabla ALUMNOS está en 1FN pero existen ANP (*nombre, apellido1*) que dependen de forma PARCIAL de la PK ó existen ANP (*nombre, apellido1*) que NO dependen de forma COMPLETA de la PK.

SOLUCION

ALUMNOS (DNI, codCurso, nombre, apellido1, nota)



ALUMNOS (DNI, nombre, apellido1) **ASISTENCIA** (DNI, codCurso, nota)

ALUMNOS		
DNI	**Nombre**	**Apellido1**
12121219A	Pedro	Valiente
34517775G	Ana	Fernández
56745378J	Sara	Crespo

ASISTENCIA		
DNI	**codCurso**	Nota
12121219A	34	9
12121219A	25	8
34517775G	34	6
56745378J	25	7
56745378J	34	6

15.3.4. Tercera forma normal (3FN)

Ocurre cuando una tabla está en **2FN** y **además ningún atributo que no sea clave depende transitivamente de las claves de la tabla.** Es decir, no ocurre cuando algún atributo no principal depende funcionalmente de atributos no principales es decir, atributos que no son clave.

EJEMPLO

ALUMNOS

DNI	nombre	apellido1	codProvincia	provincia
12121349A	Salvador	Velasco	34	Palencia
12121219A	Pedro	Velasco	34	Palencia
34517775G	Ana	Fernández	47	Valladolid
56745378J	Sara	Crespo	47	Valladolid
34568583S	Sara	Serrat	08	Barcelona

Clave Primaria (**PK**): {DNI}

Atributos Principales (**AP**): DNI

Atributos No Principales (**ANP**): nombre, apellido1, codProvincia, provincia.

Dependencias Funcionales (**DF**):

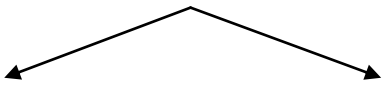
- ✓ DF1: *DNI→nombre, apellido1, codProvincia* (depende de forma COMPLETA de la clave).
- ✓ DF2: *codProvincia → provincia*
- ✓ DF3: *nombre, apellido1, codProvincia* $\not\rightarrow$ *DNI*
- ✓ DF4: *DNI - →provincia* (depende de forma TRANSITIVA de la clave).

¿Está la tabla ALUMNOS en 3FN?

La tabla ALUMNOS está en 2FN (todos los ANP dependen de forma COMPLETA de la PK pero existen ANP (provincia) que dependen de un ANP (codProvincia).

SOLUCION

ALUMNOS (DNI, nombre, apellido1, codProvincia, provincia)



ALUMNOS (DNI, nombre, apellido1,codProvincia)

PROVINCIAS (codProvincia, provincia)

ALUMNOS

DNI	Nombre	Apellido1	CodProvincia
12121349A	Salvador	Velasco	34
12121219A	Pedro	Velasco	34
34517775G	Ana	Fernández	47
56745378J	Sara	Crespo	47
34568583S	Sara	Serrat	08

PROVINCIAS	
CodProvincia	Provincia
34	Palencia
47	Valladolid
08	Barcelona

15.3.5. Forma normal de Boyce-Codd (FNBC ó BCFN)

Ocurre si una tabla está en **tercera forma normal y además todo determinante es una clave candidata** (una clave candidata tiene las mismas características que una clave primaria).

EJEMPLO

ORGANIZACIÓN		
codTrabajador	**codDepartamento**	**codResponsable**
Alex	Producción	Felipa
Arturo	Producción	Martin
Carlos	Ventas	Julio
Carlos	Producción	Felipa
Gabriela	Producción	Higinio
Luisa	Ventas	Eva
Luisa	Producción	Martin
Manuela	Ventas	Julio
Pedro	Ventas	Eva

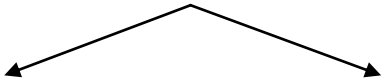
Restricciones:

- Un trabajador o trabajadora puede trabajar en varios departamentos.
- En un departamento hay varios responsables.
- Cada trabajador en un departamento sólo tiene asignado un responsable.
- *El o la responsable sólo puede ser responsable en un departamento*, detalle importante que produce una dependencia funcional ya que: ***Responsable → Departamento***

Por lo tanto hemos encontrado un determinante (*responsable*) que no es clave candidata. No está por tanto en FNBC. En este caso la redundancia ocurre por mala selección de clave. La redundancia del departamento es completamente evitable.

SOLUCION

ORGANIZACIÓN (codTrabajador, codResponsable, codDepartamento)



PERSONAL (codTrabajador, codResponsable) **RESPONSABLES** (codResponsable, codDepartamento)

PERSONAL	
codTrabajador	**codResponsable**
Alex	Felipa
Arturo	Martin
Carlos	Julio
Carlos	Felipa
Gabriela	Higinio
Luisa	Eva
Luisa	Martin
Manuela	Julio
Pedro	Eva

RESPONSABLES	
codResponsable	**codDepartamento**
Felipa	Producción
Martin	Producción
Julio	Ventas
Higinio	Producción
Eva	Ventas

En la forma normal de Boyce-Codd hay que tener cuidado al descomponer ya que se podría perder información por una mala descomposición.

Dependencia Multivaluada

Una dependencia multivaluada de una tabla con atributos X, Y, Z de X sobre Z (X →>Z) ocurre cuando los posibles valores de Y sobre cualquier par de valores X y Z (X,Z) dependen sólo del valor de X y son independientes de Z.

EJEMPLO

Cursos-Profesores-Materiales

(X)numCurso	(Y)Profesor	(Z)Material
17	Eva	1
17	Eva	2
17	Julia	1
17	Julia	2
25	Eva	1
25	Eva	2
25	Eva	3

La tabla **Cursos-Profesores-Materiales está en FNBC** ya que no hay dependencias transitivas y todos los atributos son clave sin dependencia funcional entre ellos. Sin embargo, hay redundancia. Los materiales se van a repetir para cualquier profesor dando cualquier curso, y aunque los profesores van a utilizar todos los materiales del curso (de no ser así no habría ninguna redundancia).

Los materiales del curso dependen del curso y no del profesor en una dependencia multivaluada. Para el par (numCurso, Profesor) podemos conocer los materiales pero por el curso y no por el profesor.

15.3.6. Cuarta forma normal (4FN)

Una tabla está en Cuarta Forma Normal o 4FN si está en FNBC y las únicas dependencias funcionales multivaluadas que existen son las dependencias funcionales de la clave con los atributos que no formen parte de la clave. Es decir que toda dependencia funcional multivaluada es una dependencia funcional.

Un teorema de Fagin indica cuando hay tres pares de conjuntos de atributos X, Y y Z si ocurre X->>Y|Z tienen dependencia multivaluada sobre X), entonces las tablas X,Y y X,Z reproducen sin perder información lo que poseía la tabla original. Este teorema marca la forma de dividir las tablas hacia una 4FN.

SOLUCION

Para la tabla anterior la solución serán dos tablas:

numCurso	Material
17	1
17	2
25	1
25	2
25	3

numCurso	Profesor
17	Eva
17	Julia
25	Eva

Dependencia en unión

Para comprender este tipo de dependencias es necesario explicar previamente los conceptos:

- ✓ **Proyección**: Creación de una tabla cuyos elementos forman un subconjunto de una tabla dada. Se incluyen todas las filas y algunas columnas.
- ✓ **Unión**: Formar, a partir de dos tablas, una nueva con todos los campos de una de ellas y los registros de ambas, excepto los repetidos. Ambas tablas han de tener el mismo grado y las mismas columnas.

Se dice que hay dependencia de unión o producto si una tabla tiene dependencia de unión con varias de sus proyecciones y se puede obtener la tabla por medio de la unión de dichas proyecciones.

15.3.7. Quinta forma normal (5FN)

Es la más compleja y polémica de todas. Polémica, pues no está claro en muchas ocasiones que sea una solución mejor que él no llegar a este nivel de normalización. Fue definida también por Fagin.

Es raro encontrarse este tipo de problemas cuando la normalización llega a 4FN. Se deben a restricciones muy concretas.

La quinta forma normal se refiere a dependencias que son extrañas, las **dependencias de unión**. Tiene que ver con tablas que pueden dividirse en subtablas, pero que no pueden reconstruirse.

Se emplea cuando en una misma tabla existe información redundante, con pocos atributos o cuando una tabla posee una gran cantidad de atributos y se hace por ello inmanejable.

Se dice que una tabla está en **Quinta Forma Normal o 5FN** si está en 4FN y las únicas dependencias que existen son las dependencias de unión de una tabla con sus proyecciones relacionándose entre sí mediante la clave primaria o cualquier clave candidata.

Para conseguir que una tabla que está en 4FN esté en 5FN, se divide la tabla en tantas como sea necesario. Estas tablas tendrán en común los campos que formaban la clave principal en la tabla original.

Para conseguir que una tabla que está en 4FN que tenga gran cantidad de atributos o pocos atributos y muchos registros, esté en 5FN, se divide la tabla en tantas como sea necesario. Estas tablas tendrán en común con una o más de las otras tablas la existencia de una o más claves ajenas.

EJEMPLO

Indican códigos de material suministrado por un proveedor y utilizado en un determinado proyecto.

Si ocurre una restricción especial como por ejemplo:

Cuando un proveedor nos ha suministrado alguna vez un determinado material, si ese material aparece en otro proyecto, haremos que el proveedor nos suministre también ese material para ese proyecto.

Proveedores-Materiales-Proyectos		
Proveedor	Material	Proyecto
1	1	2
1	2	1
2	1	1
1	1	1

Eso ocurre en los datos como el proveedor número 1 nos suministró el material número 1 nos suministró el material número 1 para el proyecto 2 y en el proyecto 1 utilizaremos el material 1, aparecerá la tupla proveedor 1, material 1 y proyecto 1.

La dependencia que produce esta restricción es lejana y se llama de reunión. Para esa restricción esta división en tablas sería válida:

SOLUCION

Proveedor	Material
1	1
1	2
2	1

Material	Proyecto
1	2
2	1
1	1

Esa descomposición no pierde valores en este caso, sabiendo que si el proveedor nos suministra un material podremos relacionarle con todos los proyectos que utilizan ese material.

Resumiendo, una tabla no está en quinta forma normal si hay una descomposición de esa tabla que muestre la misma información que la original.

16. Desnormalización

Cada vez que avanzamos en el nivel de normalización se necesitan más uniones entre las entidades y eso ralentiza la respuesta del sistema. Como la respuesta rápida a las demandas del usuario debe tenerse en cuenta a la hora del diseño de una base de datos, a veces es necesario **desnormaliza**r alguna parte de la misma.

Desnormalizar es transformar una base de datos en un nivel de normalización inferior. No obstante es necesario tener en cuenta que a cambio de un desempeño más rápido deberemos soportar una mayor redundancia de datos.

ÍNDICE