INDICE

9. 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**.

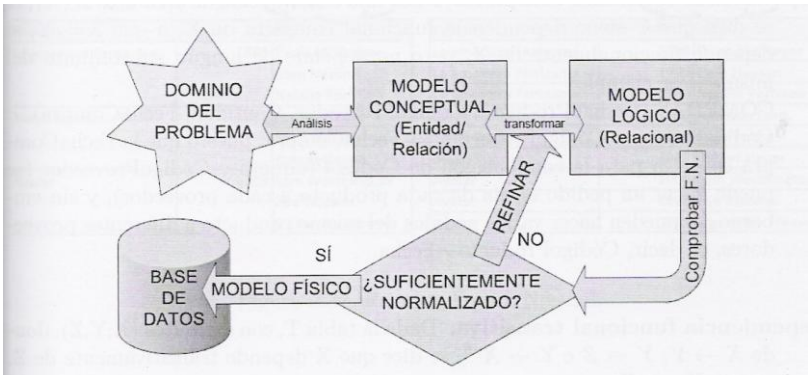


Figura 2.36: Refinamiento de un diseño de base de datos.

Una vez obtenido el esquema relacional resultante del modelo entidad relación que representaba la base de datos, normalmente tendremos una buena base de datos. Pero otras veces, debido a 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:

- **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 haya que 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.

9.1. 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).

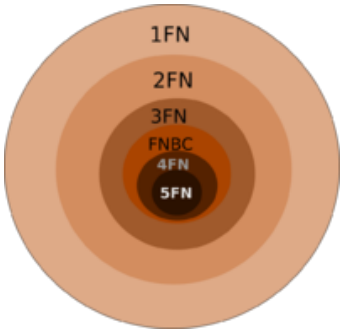
La teoría de formas normales es una teoría absolutamente matemática, pero en este tema se describe de forma más intuitiva.

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.***



9.2. 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

TRABAJADOR		
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

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

Esa tabla sí está en primera forma normal.

9.3. 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**.

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* ↛ *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**.

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 que consiga una dependencia funcional de **Y** (es decir, no hay en **X** un determinante formado por menos atributos).

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 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 Transitiva

Es más compleja de explicar, pero tiene también utilidad. 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**).

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**).

9.4. 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.

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.

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**):

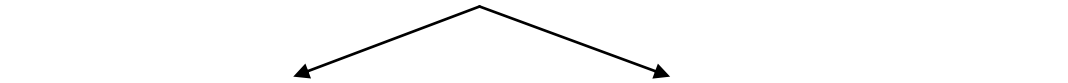
- DF1: *DNI→nombre, apellido1* (depende de forma PARCIAL de la clave).
- DF2: *DNI,codCurso → nota* (depende de forma COMPLETA de la clave).

¿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

9.5. 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, ocurre cuando algún atributo depende funcionalmente de 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**):

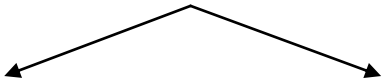
- DF1: *DNI→nombre, apellido1, codProvincia* (depende de forma COMPLETA de la clave).
- DF2: *codProvincia → provincia*
- DF3: *nombre, apellido1, codProvincia* $\not\rightarrow$ *DNI*
- DF4: *DNI - →provincia* (depende de forma TRANSITIVA de la clave).

¿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, 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

9.6. Forma normal de Boyce-Codd (FNBC)

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 uno 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 las formas de Boyce-Codd hay que tener cuidado al descomponer ya que se podría perder información por una mala descomposición.

9.7. Otras formas normales

Existen más formas normales (FN4, FN5, FNDK, FN6) cuya aplicación en el mundo real es únicamente teórica. Las formas normales 4 y 5, se ocupan de las dependencias entre atributos multivaluados, la Forma Normal Dominio Clave (FNDK) trata las restricciones y los dominios de los atributos, y finalmente la FN6 trata ciertas consideraciones con las bases de datos temporales.