

# 3. NORMALIZACIÓN EN BASES DE DATOS

---

*Carlos Augusto Meneses Escobar*

# CONTENIDO

- 1. Conceptos básicos.**
- 2. Terminología**
- 3. Formas Normales**
- 4. Reglas de Codd**
- 5. Nota Final**
- 6. Bibliografía**

# 1- CONCEPTOS

## *Que es?*

- “ Proceso en el cual se aplican reglas a las **relaciones** (*tablas*) obtenidas en el modelo Relacional para garantizar un buen diseño de bases de datos”.

# 1- CONCEPTOS

*Que se pretende al normalizar?*

- Evitar la **redundancia** de los datos.
- Evitar **inconsistencias** en la actualización de los datos.
- Conservar la **integridad referencial**.

# 1- CONCEPTOS

*En cada tabla se debe cumplir:*

- Cada columna debe tener su nombre único.
- No puede haber dos filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del mismo tipo.

# 1- CONCEPTOS

Columna

Nombre de la tabla: Trabajo

<i>Código</i>	<i>Nombre</i>	<i>Posición</i>	<i>Salario</i>
1	Edgardo Trujillo	Gerente	19000
2	Lidimarie Fonsi	Empleada	12000
3	Jean Piaget	Empleado	13500
4	Jerome Bruner	Empleado	14000

Fila

## 2- TERMINOLOGÍA

**Relación** = tabla o archivo

**Tupla** = registro, fila o renglón

**Atributo** = columna o campo

Clave = llave o código de identificación

Clave Primaria = clave principal o candidata elegida

Clave Ajena = clave externa o clave foránea

Clave Alternativa = clave secundaria

RDBMS = *Relational Data Base Manager System*.

1FN = *First Normal Form*. o 1FN (1a Forma Normal).

## 2- TERMINOLOGÍA

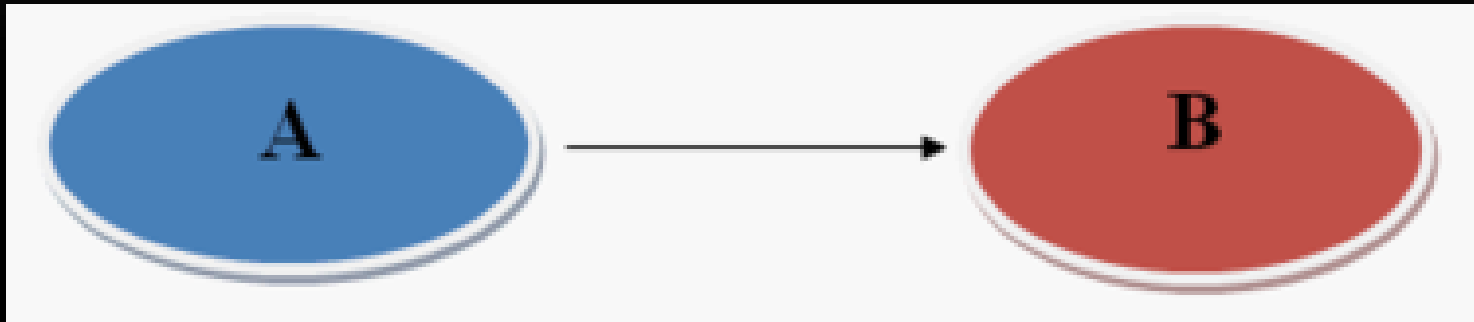
- Los términos **Relación**, **Tupla** y **Atributo** derivan del **álgebra** y **cálculo relacional**, que constituyen la fuente teórica del modelo de base de datos relacional.
- Todo atributo en una tabla tiene un dominio (que representa el conjunto de valores que el mismo).



## 2- TERMINOLOGÍA

- Una **instancia** de una tabla es un subconjunto del **producto cartesiano** entre los dominios de los atributos.
- Suele haber algunas diferencias con la analogía matemática, ya que algunos RDBMS permiten filas duplicadas.
- Una **tupla** puede razonarse matemáticamente como un elemento del producto cartesiano entre los dominios.

## 2.1. DEPENDENCIA FUNCIONAL



- *Definición:*
- Un atributo B es “**dependiente funcional**” de A, si A “**determina**” a B (Si para un valor de A existe uno y solo uno de B).

## 2.1. DEPENDENCIA FUNCIONAL

*Por ejemplo:*

Si conocemos el valor de *FechaDeNacimiento* (determinante) podemos conocer el valor de *Edad*.

*FechaDeNacimiento*  $\rightarrow$  *Edad*

De la normalización (lógica) a la implementación (física o real) puede ser sugerible tener éstas dependencias funcionales para lograr la eficiencia en las tablas.

## 2.2. PROPIEDADES DE LAS DEPENDENCIAS FUNCIONALES

### Axiomas de Armstrong:

#### 1. Dependencia funcional Reflexiva

- Si "y" está incluido en "x" entonces  $x \rightarrow y$
- Si la dirección o el nombre de una persona están incluidos en el DNI, entonces con el DNI podemos determinar la dirección o su nombre.

## 2.2 PROPIEDADES DE LAS DEPENDENCIAS FUNCIONALES

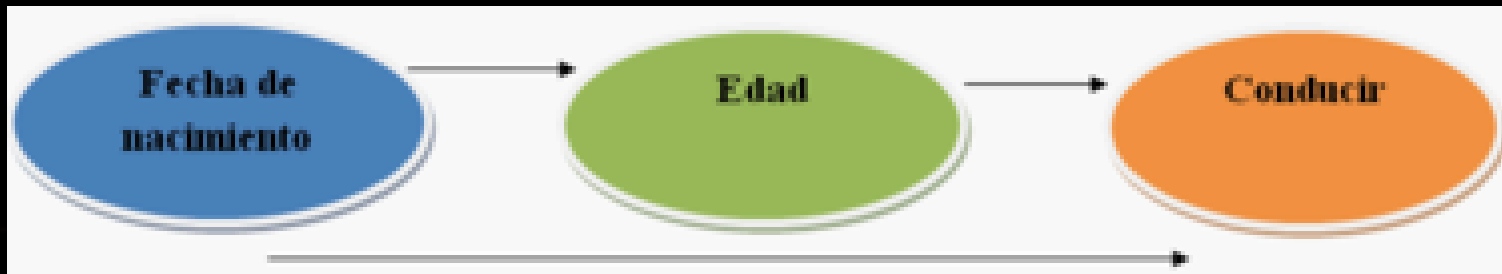
### 2. Dependencia funcional Aumentativa

- Si  $x \rightarrow y$ , entonces  $xz \rightarrow yz$
- Si  $DNI \rightarrow nombre$ . Entonces,  
 $DNI,dirección \rightarrow nombre,dirección$

## 2.2. PROPIEDADES DE LAS DEPENDENCIAS FUNCIONALES

### 3. Dependencia funcional Transitiva

- Si  $x \rightarrow y$ ,  $y \rightarrow z$ , entonces  $x \rightarrow z$
- Si  $FechaNacimiento \rightarrow edad$ . y  $Edad \rightarrow Conducir$ . Entonces,  $FechaNacimiento \rightarrow Conducir$ .



## 2.2 PROPIEDADES DE LAS DEPENDENCIAS FUNCIONALES

### Propiedades Deducidas

#### 4. UNION

- Si  $x \rightarrow y$ ,  $x \rightarrow z$ , entonces  $x \rightarrow yz$

#### 5. SEUDO - TRANSITIVA

- Si  $x \rightarrow y$ ,  $wy \rightarrow z$ , entonces  $wx \rightarrow z$

#### 6. DESCOMPOSICION

- Si  $x \rightarrow y$ ,  $z$  está incluida en  $y$ ,  
entonces  $x \rightarrow z$

### 3.1. PRIMERA FORMA NORMAL (1FN)

- El **objetivo** es eliminar los atributos multivalor y llevarlos a una forma plana (de tabla con atributos atómicos).
- Elimina los valores repetidos dentro de la base de datos.
- Todas las dependencias funcionales tienen en la parte izquierda diferente llave y hay que eliminar la **redundancia**.



## 3.1. PRIMERA FORMA NORMAL (1FN)

Una tabla está en 1FN si:

- Todos los atributos son atómicos (los elementos del dominio son indivisibles)
- La tabla contiene una clave primaria (sin atributos nulos).
- No existe variación en el número de columnas por cada fila.
- Una columna no puede tener múltiples valores. Los datos son atómicos.
- Una tabla está en 1FN cuando todos los atributos de clave están definidos y cuando todos los restantes dependen de la clave primaria

### 3.1. PRIMERA FORMA NORMAL (1FN)

Por *ejemplo*:

**Alumno** (*codigo*, *nombre*, *tel*, *semestre*)

**Materia** (*codMateria*, *nomMateria*, *creditos*)

**Cursando** (*codigo*, *codMateria*, *nombre*, *nota*)

El campo *nombre* es redundante en la tabla Cursando.

## 3.1 PRIMERA FORMA NORMAL (1FN)

La primera regla de normalización se expresa generalmente en forma de dos indicaciones separadas:

1. Todos los atributos, valores almacenados en las columnas, deben ser indivisibles.
2. No deben existir grupos de valores repetidos.

El valor de una columna debe ser una **entidad atómica**, indivisible, excluyendo así las dificultades que podría conllevar el tratamiento de un dato formado de varias partes.

Supongamos que tienes en una tabla una columna Dirección para almacenar la dirección completa, dato que se compondría del nombre de la calle, el número exterior, el número interior (puerta), el código postal, el estado y la capital.

## 3.1 PRIMERA FORMA NORMAL (1FN)

id	Nombre	Dirección	Teléfono	URL
1	Anaya	J:l: Luca	92199932	Anaya.com
2	Pericles	C/Luna # 20-28018 Tlaxcala	99299492	Pericles.com



Calle	Número	Puerta	CP	Población	Provincia
Luna	20		28018	Tlaxcala	Tlaxcala

Figura 1. Tabla con un atributo divisible en varias partes

## 3.2 SEGUNDA FORMA NORMAL (2FN)

Una tabla está en 2FN si:

- Está en 1FN.
- Los atributos no **primos** (los primos forman parte de la clave principal) dependen de forma **completa** (de la totalidad y no de un subconjunto) de la clave principal (Se conoce como **dependencia completamente funcional**).
- una tabla 1FN automáticamente está en 2FN si su clave primaria está basada solamente en un atributo simple.
- Una tabla en 2FN aún puede contener dependencias transitivas.

## 3.2 SEGUNDA FORMA NORMAL (2FN)

Por *ejemplo*:

$\{dni, idProyecto\} \rightarrow horasTrabajo$

(con el *dni* de un empleado y el *id* de un proyecto sabemos las horas de trabajo por semana)

es completamente dependiente. Mientras que,

$dni \rightarrow horasTrabajo$

$idProyecto \rightarrow horasTrabajo$

no mantienen la dependencia. Sin embargo,

$\{dni, idProyecto\} \rightarrow nombreEmpleado$

es parcialmente dependiente .

## 3.2 SEGUNDA FORMA NORMAL (2FN)

Otro *ejemplo*: Se tiene la tabla

**Biblioteca** (*codLibro, codEstudiante, titulo, autor, nombreEstudiante, fechaPrestamo*)

*codLibro*  $\rightarrow$  *titulo, autor*

*codEstudiante*  $\rightarrow$  *nombreEstudiante*

*codEstudiante, codLibro*  $\rightarrow$  *fechaPrestamo*


**Libro** (*codLibro, titulo, autor*)

**Estudiante** (*codEstudiante, nombreEstudiante*)

**Prestamo** (*codLibro, codEstudiante, fechaPrestamo*)

## 3-2 SEGUNDA FORMA NORMAL (2FN)

id	Nombre	calle	Número	Puerta	CP	Estado	Capital	Teléfono	URL
1	Anaya	Jl: Luca	15	2	28917	Tepic	Nayarit	93488345	Anaya.com
2	Pericles	Luna	20		28120	San Blas	Nayarit	88238188	Pericles.com
3	Mieres	Tajin	12	1	28120	San Blas	Nayarit	94989982	Mieres.es



Estado	CP	PK
CP	Estado	Capital
28917	Tepic	Nayarit
28120	San Blas	Nayarit
23009	Jean	Jaen



### 3.3. TERCERA FORMA NORMAL (3FN)

Una tabla está en 3FN si:

- Está en 2FN. y
- Si no existe ninguna **dependencia funcional transitiva** entre los atributos que no son clave (primos).
- “Un atributo C es dependiente funcional transitivo de A a través de B”, si “tanto B como C son dependientes funcionales de A” y C es dependiente funcional de B”.

### 3.3 TERCERA FORMA NORMAL (3FN)

Por *ejemplo*: Se tiene la tabla

**Empleado**(*codEmpleado, nombre, categoria, salario*)

*codEmpleado*  $\rightarrow$  *nombre, categoria, salario*

*categoria*  $\rightarrow$  *salario*

**Empleado** (*codEmpleado, nombre, nivel*)

**NivelSalarial** (*nivel, salario*)

### 3.3 TERCERA FORMA NORMAL (3FN)

Una tabla se encuentra en 3FN si está en 2FN y no contiene dependencias transitivas, lo cual significa que las columnas que no forman parte de la clave primaria deben depender sólo de la clave, nunca de otra columna no clave.

### 3.4 FORMA NORMAL DE BOYCE - CODD (FNBC)

Una tabla está en FNBC si:

- Si cada dependencia funcional no trivial (dependencia trivial :  $A \rightarrow A$ ) tiene una clave candidata como determinante. Es decir, los únicos determinantes son claves candidatas.

Es una versión un poco más fuerte de lo que es la 3FN.

## 3.4 FORMA NORMAL DE BOYCE - Codd (FNBC)

*Ejemplo: Hay 2 claves candidatas*

*{ID Tutor, ID Estudiante}*

*{Nro Seguro Social Tutor, ID Estudiante}*

ID Tutor	Número de seguro social del tutor	ID Estudiante
1078	088-51-0074	31850
1078	088-51-0074	37921
1293	096-77-4146	46224
1480	072-21-2223	31850

## 3.4 FORMA NORMAL DE BOYCE - CODD (FNBC)

- La tabla cumple 3FN pero no esta en FNBC
- Los 3 atributos son primarios pues pertenecen a claves candidatas
- La tabla No tiene atributos no primarios.
- La FNBC no permite ninguna dependencia funcional en la cual el conjunto determinante de atributos no sea una clave candidata (o superconjunto de eso).

### 3.4 FORMA NORMAL DE BOYCE - CODD (FNBC)

- La dependencia de *ID Tutor* en *Número de seguro social del tutor* es ese tipo de dependencia. Por consiguiente, la tabla no está en FNBC.
- Una solución sería:  
**Estudiante** (*Id Estudiante*, *Nro Seguro Social*)  
**Tutor** (*Nro Seguro Social*, *Id Tutor*)

## 3.4 FORMA NORMAL DE BOYCE - CODD (FNBC)

- Otra forma para saber si una tabla cumple la FNBC:
  - a) Si no existen claves candidatas compuestas (con varios atributos), está en FNBC.
  - b) Si existen varias claves candidatas compuestas y éstas tienen un elemento común, no está en FNBC.



## 3.5 CUARTA FORMA NORMAL (4FN)

- ❑ Se asegura de que las **dependencias multivaluadas** (donde la existencia de dos o más relaciones independientes muchos a muchos causa redundancia ) estén correcta y eficientemente representadas .
- ❑ Una tabla se encuentra en 4FN si, y sólo si:
  - Está en 3FN o FNBC, y
  - No posee dependencias multivaluadas no triviales.

## 3.5 CUARTA FORMA NORMAL (4FN)

Por *ejemplo*:

**ServicioDomicilio** (*restaurante, comida, sector*)

- ✓ “restaurantes distribuyen comidas por sectores de la ciudad.”.
- ✓ Existe una llave primaria sin atributos No primarios →
- ✓ Cumple la 3FN. Pero, hay redundancia entre restaurante y comida por diferentes sectores →

**RestauranteComida** (*restaurante, comida*)

**RestauranteSector** (*servicio, empresa*)

## 3.6 QUINTA FORMA NORMAL (5FN)

- ❑ También conocida como **Forma Normal de Proyección – Union (PJ – NF)**.
- ❑ Se busca reducir redundancia en BD que guardan hechos multi-valores aislando semánticamente relaciones múltiples relacionadas.
- ❑ Una tabla se encuentra en 5FN si, y sólo si:
  - Está en 4FN, y
  - Cada dependencia de unión (join) en ella es implicada por las llaves candidatas

## 3.6 QUINTA FORMA NORMAL (5FN)

Por *ejemplo*:

**PrestarServicio** (*medico, servicio, empresa*)

“Los medicos prestan servicios a empresas que afilian a pacientes.”. → Si fracciona la tabla en más simples se reduce la redundancia pues una empresa que contrata con medico están implícitos los servicios:

**MedicoServicio** (*medico, servicio*)

**EmpresaServicio** (*servicio, empresa*)

**MedicoEmpresa** (*medico, empresa*)

## 4. REGLAS DE CODD

12 reglas de verificación para que se considere realmente un sistema relacional.

### **Regla No. 1 - La Regla de la información (Metadatos: diccionario y catalogo)**

*Toda la información en un RDBMS está explícitamente representada de una sola manera por valores en una tabla.*

Se incluyen nombres de tablas, nombres de vistas, nombres de columnas, y los datos de las columnas deben estar almacenados en tablas de la BD

## 4. REGLAS DE CODD

### **Regla No. 2 – La regla del acceso garantizado**

*Cada ítem de datos debe ser lógicamente accesible al ejecutar una búsqueda que combine el nombre de la tabla, su clave primaria, y el nombre de la columna. Deberá encontrarse uno y solamente un valor.*

Por esta razón la definición de claves primarias para todas las tablas es obligatoria.

## 4. REGLAS DE CODD

### **Regla No. 3 – Tratamiento sistemático de los valores nulos**

*La información inaplicable o faltante puede ser representada a través de valores nulos.*

Un RDBMS (Sistema Gestor de Bases de Datos Relacionales) debe ser capaz de soportar el uso de valores nulos en el lugar de columnas cuyos valores sean desconocidos o inaplicables.

## 4. REGLAS DE CODD

### **Regla No. 4 - La regla de la descripción de la base de datos**

*La descripción de la base de datos es almacenada de la misma manera que los datos ordinarios, esto es, en tablas y columnas, y debe ser accesible a los usuarios autorizados.*

La información de tablas, vistas, permisos de acceso de usuarios autorizados, etc, se debe almacenar de la misma manera: En tablas. Estas tablas como todas deben ser accesibles a través de sentencias de SQL (o similar).



## 4. REGLAS DE CODD

### **Regla No. 5 - La regla del sub-lenguaje Integral**

*Debe haber al menos un lenguaje que sea integral para soportar la definición de datos, manipulación de datos, definición de vistas, restricciones de integridad, y control de autorizaciones y transacciones.*

Esto significa que debe haber por lo menos un lenguaje con una sintaxis bien definida que pueda ser usado para administrar completamente la base de datos.

## 4. REGLAS DE CODD

### **Regla No. 6 - La regla de la actualización de vistas**

*Todas las vistas que son teóricamente actualizables, deben ser actualizables por el sistema mismo.*

La mayoría de las RDBMS permiten actualizar vistas simples, pero deshabilitan los intentos de actualizar vistas complejas.

## 4. REGLAS DE CODD

### **Regla No. 7 - La regla de insertar y actualizar**

*La capacidad de manejar una base de datos con operandos simples aplica no sólo para la recuperación o consulta de datos, sino también para la inserción, actualización y borrado.*

Las cláusulas para leer, escribir, eliminar y agregar registros (SELECT, UPDATE, DELETE e INSERT en SQL) deben estar disponibles y operables, independientemente del tipo de relaciones y restricciones que haya entre las tablas.

## 4. REGLAS DE CODD

### **Regla No. 8 - La regla de independencia física**

*El acceso de usuarios a la BD a través de terminales o programas de aplicación, debe permanecer consistente lógicamente cuando quiera que haya cambios en los datos almacenados, o sean cambiados los métodos de acceso a los datos.*

El comportamiento de los programas y de la actividad de usuarios vía terminales debe ser predecible basado en la definición lógica de la BD, y debería permanecer inalterado, independientemente de los cambios en la definición física de ésta.

## 4. REGLAS DE CODD

### **Regla No. 9 - La regla de independencia lógica**

*Los programas de aplicación y las actividades de acceso por terminal deben permanecer lógicamente inalteradas cuando quiera que se hagan cambios (según los permisos asignados) en las tablas de la base de datos.*

La independencia lógica de los datos especifica que los programas de aplicación y las actividades de terminal deben ser independientes de la estructura lógica, por lo tanto los cambios en esta no deben alterar estos programas de aplicación.

## 4. REGLAS DE CODD

### **Regla No. 10 - La regla de la independencia de la integridad**

*Todas las restricciones de integridad deben ser definibles en los datos, y almacenables en el catalogo, no en el programa de aplicación.*

### **Las reglas para asegurar la integridad referencial:**

- Ningún componente de una clave primaria puede tener valores en blanco o nulos.
- Para cada valor de clave foránea deberá existir un valor de clave primaria concordante.

## 4. REGLAS DE CODD

### **Regla No. 11 - La regla de la distribución**

*El sistema debe poseer un lenguaje de datos que pueda soportar que la base de datos esté distribuida físicamente en distintos lugares sin que esto afecte o altere a los programas de aplicación.*

El soporte para bases de datos distribuidas significa que una colección de relaciones, BDs corriendo en distintas máquinas y con distintos sistemas operativos y conectadas por redes, pueda funcionar como si estuviera disponible en una única base de datos en una sola máquina.

## 4. REGLAS DE CODD

### **Regla No. 12 - Regla de la no-subversión**

*Si el sistema tiene lenguajes de bajo nivel, estos lenguajes de ninguna manera pueden ser usados para violar la integridad de las reglas y restricciones expresadas en un lenguaje de alto nivel (como SQL).*

Algunos productos solamente construyen una interfaz relacional para sus bases de datos No relacionales, lo que hace posible la subversión (violación) de las restricciones de integridad. Esto no debe ser permitido.



## NOTA FINAL

- En algunos casos, es necesario considerar la desnormalización para mejorar el rendimiento. La *desnormalización* es la duplicación intencionada de columnas en varias tablas, lo cual aumenta la redundancia de datos. No debe suponerse automáticamente que todas las uniones tardan demasiado tiempo.

## REFERENCIAS BIBLIOGRÁFICAS

- *A Relational Model of Data for Large Shared Data Banks* Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387 [1]
- E.F.Codd (junio de 1970). "*A Relational Model of Data for Large Shared Databanks*". Communications of the ACM.
- C.J.Date (1994). "*An Introduction to Database Systems*". Addison-Wesley.