Normalización de bases de datos

El proceso de **normalización de bases de datos** consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Disminuir problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

En el modelo relacional es frecuente llamar *tabla* a una relación, aunque para que una tabla sea considerada como una relación tiene que cumplir con algunas restricciones:

- Cada tabla 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 Terminología relacional equivalente

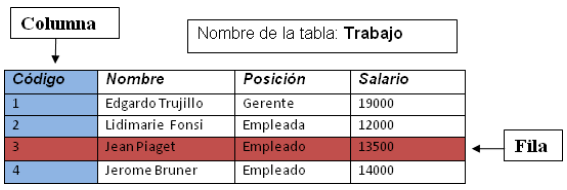

Figura 1.0: Trabajo (Código, Nombre, Posición, Salario), donde Código es la Clave Primaria.

- Relación = tabla o archivo
- Registro = registro, fila , renglón o tupla
- Atributo = columna o campo
- Clave = llave o código de identificación
- Clave Candidata = superclave mínima
- Clave Primaria = clave candidata elegida
- Clave Ajena (o foránea) = clave externa o clave foránea
- Clave Alternativa = clave secundaria
- Dependencia Multivaluada = dependencia multivalor
- RDBMS = Del inglés *Relational Data Base Manager System* que significa, *Sistema Gestor de Bases de Datos Relacionales*.
- 1FN = Significa, *Primera Forma Normal* o 1NF del inglés *First Normal Form*.

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, el cual representa el conjunto de valores que el mismo puede tomar. Una instancia de una tabla puede verse entonces como un subconjunto del producto cartesiano entre los dominios de los atributos. Sin embargo, suele haber algunas diferencias con la analogía matemática, ya que algunos RDBMS permiten filas duplicadas, entre otras cosas. Finalmente, una tupla puede razonarse matemáticamente como un elemento del producto cartesiano entre los dominios.

2 Dependencia

***B** es funcionalmente dependiente de **A**.*

2.1 Dependencia funcional

Una dependencia funcional es una conexión entre uno o más atributos. Por ejemplo si se conoce el valor de *DNI* tiene una conexión con *Apellido* o *Nombre* .

Las dependencias funcionales del sistema se escriben utilizando una flecha, de la siguiente manera:

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 la dependencia funcional

Existen tres axiomas de Armstrong:

2.2.1 Dependencia funcional reflexiva

Si "y" está incluido en "x" entonces x $\rightarrow$ y

A partir de cualquier atributo o conjunto de atributos siempre puede deducirse él mismo. 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.

2.2.2 Dependencia funcional Aumentativa

$x \rightarrow y$ entonces $xz \rightarrow yz$

DNI $\rightarrow$ *nombre*

DNI,dirección $\rightarrow$ *nombre,dirección*

Si con el DNI se determina el nombre de una persona, entonces con el DNI más la dirección también se determina el nombre y su dirección.

Dependencia funcional transitiva.

2.2.3 Dependencia funcional transitiva

Sean *X, Y, Z* tres atributos (o grupos de atributos) de la misma entidad. Si *Y* depende funcionalmente de *X* y *Z* de *Y*, pero *X* no depende funcionalmente de *Y*, se dice entonces que *Z* depende transitivamente de *X*. Simbólicamente sería:

$X \rightarrow Y \rightarrow Z$ entonces $X \rightarrow Z$

FechaDeNacimiento $\rightarrow$ *Edad*

Edad $\rightarrow$ *Conducir*

FechaDeNacimiento $\rightarrow$ *Edad* $\rightarrow$ *Conducir*

Entonces tenemos que *FechaDeNacimiento* determina a *Edad* y la *Edad* determina a *Conducir*, indirectamente podemos saber a través de *FechaDeNacimiento* a *Conducir* (En muchos países, una persona necesita ser mayor de cierta edad para poder conducir un automóvil, por eso se utiliza este ejemplo).

"C será un dato simple (dato no primario), B,será un otro dato simple (dato no primario), A, es la llave primaria (PK). Decimos que C dependera de B y B dependera funcionalmente de A."

2.3 Propiedades deducidas

2.3.1 Unión

$x \rightarrow y$ y $x \rightarrow z$ entonces $x \rightarrow yz$

2.3.2 Pseudo-transitiva

$x \rightarrow y$ y $wy \rightarrow z$ entonces $wx \rightarrow z$

2.3.3 Descomposición

$x \rightarrow y$ y z está incluido en y entonces $x \rightarrow z$

3 Claves

Una **clave primaria** es aquella columna (o conjunto de columnas) que identifica únicamente a una fila. La clave primaria es un identificador que va a ser siempre único para cada fila. Se acostumbra a poner la clave primaria como la primera columna de la tabla pero es más una conveniencia que una obligación. Muchas veces la clave primaria es numérica auto-incrementada, es decir, generada mediante una secuencia numérica incrementada automáticamente cada vez que se inserta una fila.

En una tabla puede que tengamos más de una columna que puede ser clave primaria por sí misma. En ese caso se puede escoger una para ser la clave primaria y las demás claves serán **claves candidatas**.

Una **clave ajena (foreign key o clave foránea)** es aquella columna que existiendo como dependiente en una tabla, es a su vez clave primaria en otra tabla.

Una **clave alternativa** es aquella clave candidata que no ha sido seleccionada como clave primaria, pero que también puede identificar de forma única a una fila dentro de una tabla. Ejemplo: Si en una tabla clientes definimos el número de documento (id_cliente) como clave primaria, el número de seguro social de ese cliente podría ser una clave alternativa. En este caso no se usó como clave primaria porque es posible que no se conozca ese dato en todos los clientes.

Una **clave compuesta** es una clave que está compuesta por más de una columna.

La visualización de todas las posibles **claves candidatas** en una tabla ayudan a su optimización. Por ejemplo, en una tabla PERSONA podemos identificar como claves su DNI, o el conjunto de su nombre, apellidos, fecha de nacimiento y dirección. Podemos usar cualquiera de las dos opciones o incluso todas a la vez como clave primaria, pero es mejor en la mayoría de sistemas la elección del menor número de columnas como **clave primaria**.

4 Formas normales

Las formas normales son aplicadas a las tablas de una base de datos. Decir que una base de datos está en la forma normal **N** es decir que todas sus tablas están en la forma normal **N**.

En general, las primeras tres formas normales son suficientes para cubrir las necesidades de la mayoría de las bases de datos. El creador de estas 3 primeras formas normales (o reglas) fue Edgar F. Codd.[1]

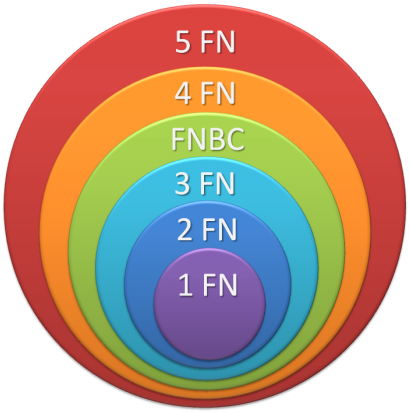

Diagrama de inclusión de todas las formas normales.

4.1 Primera Forma Normal (1FN)

Una tabla está en Primera Forma Normal si:

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos.
- La tabla contiene una clave primaria única.
- La clave primaria no contiene atributos nulos.
- No debe existir variación en el número de columnas.
- Los Campos no clave deben identificarse por la clave (Dependencia Funcional)
- Debe Existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados
- Una tabla no puede tener múltiples valores en cada columna.
- Los datos son atómicos (a cada valor de X le pertenece un valor de Y y viceversa).

Esta forma normal elimina los valores repetidos dentro de una Base de Datos.

4.2 Segunda Forma Normal (2FN)

Dependencia Funcional. Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales. (Todos los atributos que no son clave principal deben depender únicamente de la clave principal).

En otras palabras podríamos decir que la segunda forma normal está basada en el concepto de dependencia completamente funcional. Una dependencia funcional $x \to y$ es completamente funcional si al eliminar los atributos A de X significa que la dependencia no es mantenida, esto es que $A \in X, X - \{A\} \nrightarrow Y$. Una dependencia funcional $x \to y$ es una dependencia parcial si hay algunos atributos $A \in X$ que pueden ser eliminados de X y la dependencia todavía se mantiene, esto es $A \in X, X - \{A\} \to Y$.

Por ejemplo {DNI, ID_PROYECTO} $\to$ HORAS_TRABAJO (con el DNI de un empleado y el ID de un proyecto sabemos cuántas horas de trabajo por semana trabaja un empleado en dicho proyecto) es completamente funcional dado que ni DNI $\to$ HORAS_TRABAJO ni ID_PROYECTO $\to$ HORAS_TRABAJO mantienen la dependencia. Sin embargo {DNI, ID_PROYECTO} $\to$ NOMBRE_EMPLEADO es parcialmente dependiente dado que DNI $\to$ NOMBRE_EMPLEADO mantiene la dependencia.

4.3 Tercera Forma Normal (3FN)

La tabla se encuentra en 3FN si es 2FN y si no existe ninguna dependencia funcional transitiva entre los atributos que no son clave.

Un ejemplo de este concepto sería que, una dependencia funcional X->Y en un esquema de relación R es una dependencia transitiva si hay un conjunto de atributos Z que no es un subconjunto de alguna clave de R, donde se mantiene X->Z y Z->Y.

Por ejemplo, la dependencia SSN->DMGRSSN es una dependencia transitiva en EMP_DEPT de la siguiente figura. Decimos que la dependencia de DMGRSSN el atributo clave SSN es transitiva vía DNUMBER porque las dependencias SSN→DNUMBER y DNUMBER→DMGRSSN son mantenidas, y DNUMBER no es un subconjunto de la clave de EMP_DEPT. Intuitivamente, podemos ver que la dependencia de DMGRSSN sobre DNUMBER es indeseable en EMP_DEPT dado que DNUMBER no es una clave de EMP_DEPT.

Formalmente, un esquema de relación R está en 3 Forma Normal Elmasri-Navathe,[2] si para toda dependencia funcional $X \to A$, se cumple al menos una de las siguientes condiciones:

1. X es superllave o clave.
2. A es atributo primo de R ; esto es, si es miembro de alguna clave en R .

Además el esquema debe cumplir necesariamente, con las condiciones de segunda forma normal.

4.4 Forma Normal de Boyce-Codd (FNBC)

La tabla se encuentra en FNBC si cada determinante, atributo que determina completamente a otro, es clave candidata. Deberá registrarse de forma anillada ante la presencia de un intervalo seguido de una formalización perpetua, es decir las variantes creadas, en una tabla no se llegaran a mostrar, si las ya planificadas, dejan de existir.

Formalmente, un esquema de relación R está en FNBC, si y sólo si, para toda dependencia funcional $X \to A$ válida en R , se cumple que

1. X es superllave o clave.

De esta forma, todo esquema R que cumple FNBC, está además en 3FN; sin embargo, no todo esquema R que cumple con 3FN, está en FNBC.

4.5 Cuarta Forma Normal (4FN)

Una tabla se encuentra en 4FN si, y sólo si, para cada una de sus dependencias múltiples no funcionales X->->Y, siendo X una super-clave que, X es o una clave candidata o un conjunto de claves primarias.

4.6 Quinta Forma Normal (5FN)

Una tabla se encuentra en 5FN si:

- La tabla está en 4FN
- No existen relaciones de dependencias no triviales que no siguen los criterios de las claves. Una tabla que se encuentra en la 4FN se dice que está en la 5FN si, y sólo si, cada relación de dependencia se encuentra definida por claves candidatas.

5 Reglas de Codd

Codd se percató de que existían bases de datos en el mercado las cuales decían ser relacionales, pero lo único que hacían era guardar la información en las tablas, sin estar estas tablas literalmente normalizadas; entonces éste publicó 12 reglas que un verdadero sistema relacional debería tener, en la práctica algunas de ellas son difíciles de realizar. Un sistema podrá considerarse "más relacional" cuanto más siga estas reglas.

5.1 Regla No. 1 - La Regla de la información

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

Cualquier cosa que no exista en una tabla no existe del todo. Toda la información, incluyendo nombres de tablas, nombres de vistas, nombres de columnas, y los datos de las columnas deben estar almacenados en tablas dentro de las bases de datos. Las tablas que contienen tal información constituyen el Diccionario de Datos. Esto significa que todo tiene que estar almacenado en las tablas.

Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas. Por tanto los metadatos (diccionario, catálogo) se representan exactamente igual que los datos de usuario. Y puede usarse el mismo lenguaje (ej. SQL) para acceder a los datos y a los metadatos (regla 4)

5.2 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.

Esto significa que dado un nombre de tabla, dado el valor de la clave primaria, y dado el nombre de la columna requerida, deberá encontrarse uno y solamente un valor. Por esta razón la definición de claves primarias para todas las tablas es prácticamente obligatoria.

5.3 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.

5.4 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, debe ser almacenada exactamente de la misma manera: En tablas. Estas tablas deben ser accesibles igual que todas las tablas, a través de sentencias de SQL (o similar).

5.5 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.

5.6 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.

5.7 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 de datos'.

Esto significa que 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 o no.

5.8 Regla No. 8 - La regla de independencia física

El acceso de usuarios a la base de datos 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 de aplicación y de la actividad de usuarios vía terminales debería ser predecible basados en la definición lógica de la base de datos, y éste comportamiento debería permanecer inalterado, independientemente de los cambios en la definición física de ésta.

5.9 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 la estructura lógica no deben alterar o modificar estos programas de aplicación.

5.10 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.

5.10.1 Las reglas de integridad

1. Ningún componente de una clave primaria puede tener valores en blanco o nulos (ésta es la norma básica de integridad).

2. Para cada valor de clave foránea deberá existir un valor de clave primaria concordante. La combinación de estas reglas aseguran que haya integridad referencial.

5.11 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 arbitraria de relaciones, bases de datos corriendo en una mezcla de distintas máquinas y distintos sistemas operativos y que esté conectada por una variedad de redes, pueda funcionar como si estuviera disponible como en una única base de datos en una sola máquina.

5.12 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.

6 Véase también

- 1NF - 2NF - 3NF - BCNF - 4NF - 5NF - DKNF - 6NF - Denormalización
- Edgar Frank Codd
- Base de datos

7 Referencias

[1] *A Relational Model of Data for Large Shared Data Banks* Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387

[2] *Fundamentals of DATABASE SYSTEMS* Addison-Wesley;, ISBN-10: 0321122267, ISBN-13: 978-0321122261,

- 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.

8 Text and image sources, contributors, and licenses

8.1 Text

- **Normalización de bases de datos** *Fuente:* http://es.wikipedia.org/wiki/Normalización_de_bases_de_datos?oldid=77298217 *Colaboradores:* Dovidena, Angus, Rosarino, Ecelan, Dodo, Jynus, Truor, Rsg, Tostadora, Elwikipedista, Valyag, Xatufan, Almorca, FAR, Airunp, Taichi, Edtruji, Magister Mathematicae, Goofys, RobotQuistnix, Akhram, Pabloab, Yrbot, Maleiva, Mortadelo2005, Barct, Gaeddal, GermanX, Deniel77, Eskimbot, Banfield, Milestones, Er Komandante, Filipo, Mencey, Kuanto, Paintman, Jago84, Juandiegocano, BOTpolicia, CEM-bot, Laura Fiorucci, Alexav8, Pacovila, Antur, Hernan Beati, Eluseche, Tyrannosaurusreflex, PabloCastellano, Yeza, RoyFocker, Isha, Egaida, Vitorres, Gusgus, Verajm, Rayleon, Jugones55, Miguelo on the road, Mansoncc, TXiKiBoT, Millars, Humberto, Netito777, Merlyn333, Nioger, Bedwyr, Changa, Pólux, BL, Snakeyes, Technopat, C'est moi, Queninosta, Gatra, Libertad y Saber, Matdrodes, DJ Nietzsche, BlackBeast, Lucien leGrey, U.gonzalez, Fama.arciniega, Muro Bot, Edmenb, Jkarretero, SieBot, Mushii, RafaRamirez, PaintBot, Loveless, BOTarate, Manwë, Tirithel, Montehermoso-spain, Javierito92, Cumanacr, Alagoro, Leonpolanco, Pan con queso, Botito777, Walter closser, Darkicebot, VanBot, Rrupo, UA31, AVBOT, David0811, MarcoAurelio, Diegusjaimes, Arjuno3, Saloca, Andvarp, Andreasmperu, Luckas-bot, Spirit-Black-Wikipedista, Ptbotgourou, Elielsardanons, Thiamath, ArthurBot, SuperBraulio13, Acardh, Manuelt15, Xqbot, Jkbw, Rubinbot, Botarel, D'ohBot, Hprmedina, TobeBot, Diego diaz espinoza, Fgtez, CVBOT, Angelito7, Humbefa, ManuelMontiel, EmausBot, Savh, Grillitus, Jhtan, Emiduronte, Jcaraballo, ChuispastonBot, MadriCR, Felipecanol, WikitanvirBot, SHeLoo, Rezabot, TeleMania, Carliitaeliza, Elvisor, MahdiBot, Addbot, Laverde0 y Anónimos: 550

8.2 Images

- **Archivo:DependenciaFunional.png** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/2/25/DependenciaFunional.png *Licencia:* CC-BY-SA-3.0-2.5-2.0-1.0 *Colaboradores:* Trabajo propio *Artista original:* Edtruji
- **Archivo:DependenciaFunional2.png** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/3/36/DependenciaFunional2.png *Licencia:* CC-BY-SA-3.0 *Colaboradores:* Trabajo propio *Artista original:* Edtruji
- **Archivo:FormasNormalesBD.png** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/5/59/FormasNormalesBD.png *Licencia:* CC-BY-SA-3.0 *Colaboradores:* Trabajo propio *Artista original:* Syscall
- **Archivo:TablaRelacional2.png** *Fuente:* http://upload.wikimedia.org/wikipedia/commons/a/a1/TablaRelacional2.png *Licencia:* CC-BY-3.0 *Colaboradores:* Trabajo propio *Artista original:* Edgardo Trujillo

8.3 Content license

- Creative Commons Attribution-Share Alike 3.0