Fundamentos bases de datos relacionales: Conceptos básicos para su comprensión.

Julio Lopez-Nunez.
Ingeniero en Informática – Universidad Francisco de Aguirre.
Candidato a Doctor en Política Educativa – Universidad de Playa Ancha. Valparaíso-Chile.
Julio.Lopez@alumnos.upla.cl
ORCID ID: https://orcid.org/0000-0002-7920-1563

Resumen

El proceso de normalización de bases de datos, como parte fundamental de la construcción de un modelo de datos bajo el paradigma relacional, nos ayuda a prever errores de actualización vía la eliminación de redundancia e inconsistencia de los datos. Al normalizar una base de datos, la primera impresión se interpreta como el hecho de estar creando más entidades de lo necesario. Este simple considerando motiva el desarrollo de este trabajo. Centrado en la explicación del proceso de construcción de un almacén de datos, en un contexto de educación superior, la propuesta es entregar de forma sencilla los fundamentos y buenas prácticas que deben acompañar dicho proceso. Se trata de una breve revisión de la literatura científica enfocada en la enseñanza del modelamiento de datos.

Palabras claves: Modelo Entidad Relación, Normalización Bases de Datos, Formas Normales, Buenas Prácticas.

Introducción

En la actualidad podemos ver en ejecución una multiplicidad de sistemas computacionales, con diferentes propósitos, publico objetivo y plataformas de acceso. Sin embargo, una característica en común de todos ellos dice relación con el tratamiento de la información (datos) que estos generan. No por nada el concepto de Internet de las Cosas (en inglés, ***Internet of Things***, abreviado **IoT**[1]) cobra una importancia trascendental dada la magnitud de los datos que transitan por las redes que conforman la internet. Este volumen de datos, casi de manera natural, nos interpela a responder ¿Dónde y cómo almacenamos esa información? ¿Es posible crear una estructura de datos que permita realizar una búsqueda sencilla, optima y veloz dentro de este gigantesco conjunto de datos? ¿Cuál es la forma más eficiente que debe tener un almacén de datos? Aunque no lo crea, estas preguntas fueron respondidas mucho antes del nacimiento de la internet, por un científico (Edgar Frank "Ted" Codd) nacido en el año 1923 y que, en la década del 70 (hace medio siglo) diseño una serie de reglas para construir un almacén de datos capaz de resolver todas estas interrogantes. Codd utilizó la voz anomalías para señalar los problemas de actualización y comportamientos inesperados en las consultas realizadas a los datos de las tablas de una base de datos (Opell, 2010).

Partamos por el principio, un almacén de datos (Base de datos para nosotros) se trata de un conjunto de datos, con un origen en común, un contexto dado y, almacenados de forma sistemáticamente con la clara intención de ser usados posteriormente. Por ejemplo, los datos que generan los usuarios de Tinder.

Entonces, los usuarios de esta famosa aplicación de citas tienen una serie de características (atributos en nuestro contexto de bases de datos), los cuales pueden estar asociados a temáticas de identificación (Nombre, edad, seco, etc), como también a gustos o preferencias (edad y sexo de una potencial cita). Además, esta aplicación entrega la posibilidad de guardar los ***match*** [2]que todo usuario espera alcanzar. Sin duda, otro cumulo de datos que esta aplicación genera, guarda y administra, son las conversaciones que sus usuarios puedan mantener entre aquellos ***match*** que logran alcanzar.

---

[1] Concepto que se refiere a una interconexión digital de objetos cotidianos con internet. La idea que intenta representar queda bastante bien ilustrada por su nombre, cosas cotidianas del hogar (por ejemplo) que se conectan a Internet.

[2] En el contexto de Tinder, hacer un match básicamente se trata de dar un "me gusta" a un perfil de otro usuario de la aplicación, si esa otra persona también da "me gusta" al perfil del usuario señalado anteriormente, esto se transforma en un match. En sencillo, dos usuarios dan "me gusta" al perfil del otro.

Como es posible apreciar, esta aplicación bastante básica en su concepción genera una cantidad de datos que resulta difícil de poder administrar con eficiencia. Para resolver esta situación se plantea que una de las formas mas sencilla de lograr el objetivo sería vía la utilización de una base de datos relacional, la cual debe estar normalizada en tercera forma normal.

Formas Normales

Lo señalado en el párrafo anterior nos lleva de regreso a lo propuesto por Codd, con una propuesta bastante sencilla, la cual se comprenderá una vez que entendemos que una base de datos relacionales es una estructura que brinda persistencia a los datos en el modelo relacional, el cual tiene un origen en un modelo matemático, basado en teoría de conjunto, y lógica de predicados. Antes de revisar algún ejemplo es necesario señalar lo siguiente, las formas normales son cinco (5), mencionada la primera como primera forma normal con sigla 1NF, y así hasta la quinta forma normal con sigla 5NF, más una especialización de la tercera forma normal (3NF), denominada forma normal Boyce- Codd, con sigla NFBC. Sin embargo, el diseño de base de datos, tal y como se realiza en la actualidad en la industria, presta especial atención a la normalización hasta la 3NF.

Dependencia Funcional

Ahora bien, para crear una base de datos relacional, es necesario comprender el concepto de dependencia funcional. Esto nos llevara de forma expedita a la comprensión de las formas normales a las cuales referencia la propuesta de Codd (15 reglas diseñadas para la estructura de una base de datos).

De regreso a Tinder, los datos que esta aplicación administra son los que se pueden apreciar en la Tabla 1. Un antecedente importante, se debe prestar atención que cada usuario puede tener muchos ***match***. Por tanto, también es posible tener muchas conversaciones, una por cada usuario con el que realizo un ***match***.

Tabla 1.

Nombre de Usuario	Sexo	Edad	Residencia	Teléfono 1	Teléfono 2	Descripción	Preferencia	Usuarios en Match	Edad de sus Match	Conversación con Match	Fecha Ultimo Chat
Usted	M	21	Maipú	555-444	9-7465	Tranquilo	F	User A	20	XXXXXX	22-01
Sus compañeros	M	22	Cerrillos	243-578	9-6689	Skaeter	M	User B, User C	21 22	XXXXXX	17-03
Yo	F	21	Recoleta	987-456		Hogareño	M	User A	20	XXXXXX	31-01
Otros usuarios del resto del mundo	F	25	Santiago	555-985	9-4257	Punk	M	User A, User C, User E	20 22 29	XXXXXX, YYYYYY, ZZZZZZZ	04-07 12-10 05-05

Entonces, la anterior tabla nos permitirá comprender que son las dependencias funcionales. Centre su mirada en la columna "Edad de sus ***Match***", para cada usuario de Tinder esta columna podría tener cero o muchos datos, dependiendo de que tan exitosa sea su participación en Tinder. Sin embargo, el dato en cuestión, la edad de las personas con quien realizo ***match***, dependerá en estricto rigor de los usuarios con quien se realizó ese ***match***, no de usted. Entonces, la edad de sus ***match*** depende funcionalmente de las características de la persona con quien se realizo match. En síntesis, una dependencia funcional se comprende como una relación que se da entre dos atributos ***x*** e ***y***, cuando cada valor de ***x*** tiene asociado un solo valor de ***y*** (Hueso, 2014).

Clave Primaria

Un segundo concepto para desarrollar es la Clave Primaria (***Pk*** por el inglés *Primary Key*), lo que debemos encontrar es aquel atributo (columna de la Tabla 1) que contenga valores únicos, imposibles de encontrar en el resto de la tabla. Para el caso de Tinder ¿Es posible que dos personas tengan el mismo usuario?, la respuesta es NO. En la base de datos de Tinder no pueden existir nombres de Usuario repetidos. Misma situación ocurre con su numero de celular, para le caso de WhatsApp o su correo electrónico Gmail. Siempre se trata de números-nombres únicos. No ocurre lo mismo con la clave de su correo, dicho atributo si puede existir en más de una oportunidad en la base de datos de Gmail. Sin esta condición de exclusividad de un atributo, es imposible tener una base de datos normalizada capaz de ser eficiente en la gestión de los datos. Algo importante, una Clave Primaria no siempre esta compuesta por solo un atributo, algunas claves están compuestas por dos o mas atributo y, cuya combinación, es única dentro del almacén de datos.

Normalización Base de Datos

Una vez definido el concepto de clave primaria y dependencia funcional, se procederá a normalizar[3] el anterior almacén de datos (Tabla 1 en nuestro ejemplo), solo hasta la 3NF. Para esto debemos comenzar por la primera forma normal. Se dice que una tabla de datos (como es nuestro caso) se encuentra en 1NF cuando se ajusta a la consigna "Libre de grupos repetitivos". De tal forma, esto se debe entender como que las filas de nuestra tabla no se repiten. Que dentro de cada casilla existen valores únicos (atómicos) y, que los atributos son únicos. Nada de esto ocurre, salvo la primera condición. La casilla "Usuarios en Match" contiene mas de un valor en la fila (también reconocido con el termino registro) dos y cuatro. Lo mismo ocurre con "Edades de sus match" y "Conversaciones con Match". Además, las columnas (atributos) cinco y seis representan lo mismo (teléfono del usuario). Para resolver esto se deben extraer aquellas columnas que no cumplen con lo indicado y crear una nueva tabla, conservando el atributo que contiene valores únicos, es decir conservando el (los) atributo(s) denominado(s) como Clave Primaria. Nuestro ejemplo quedaría de la siguiente forma.

Tabla 1.

Nombre de Usuario	Sexo	Edad	Residencia	Descripción	Preferencia	Rango Edad
Usted	M	21	Maipú	Tranquilo	F	20-29
Sus compañeros	M	22	Cerrillos	Skaeter	M	20-29
Yo	F	21	Recoleta	Hogareño	M	20-29
Otros usuarios del resto del mundo	F	25	Santiago	Punk	M	20-29

Tabla 2.

Nombre de Usuario	Teléfono
Usted	555-444
Usted	9-7465
Sus compañeros	243-578
Sus compañeros	9-6689
Yo	987-456
Otros usuarios del resto del mundo	555-985
Otros usuarios del resto del mundo	9-4257

Tabla 3.

Nombre de Usuario	Usuarios en Match	Edad de sus Match	Conversación con Match	Fecha Ultimo Chat
Usted	User A	20	XXXXXX	22-01
Sus compañeros	User B,	21	XXXXXX	17-03
Sus compañeros	User C	22		
Yo	User A	20	XXXXXX	31-01
Otros usuarios del resto del mundo	User A,	20	XXXXXX	04-07
Otros usuarios del resto del mundo	User C,	22	YYYYYY	12-10
Otros usuarios del resto del mundo	User E	29	ZZZZZZZ	05-05

El llevar nuestra Tabla 1 a la primera forma normal (1NF) implico que pasamos de un almacén de datos, constituido por una tabla (entidad), a tener un repositorio conformado por tres entidades. El objetivo es llevar nuestro diseño de base de datos a la tercera forma normal, para concretar esto debemos pasar por la segunda forma normal (2NF). Esta forma normal señala que si y solo si, dada una clave primaria y cualquier atributo que no sea un constituyente de la clave primaria, el atributo no clave depende de toda la clave primaria en vez de solo de una parte de ella. Esta revisión la debes hacer sobre las tres entidades que ahora componen nuestro modelo de base de datos. Nuevamente debemos ubicar la clave primaria de cada entidad, en el caso de la Tabla 3 la clave primaria sería una composición de los atributos uno y dos, si usted revisa esta combinación podrá comprobar que se trata de una combinación única dentro de dicha entidad. Entonces, asumiendo que la clave primaria sería una composición de los atributos uno y dos, no todos los atributos que no forman parte de la clave (atributo tres, cuatro y cinco) dependen de **toda** la clave.

El caso del atributo tres (Edad de los *match*) depende funcionalmente solo del usuario con el cual se realizó el *match*. Entonces, al igual que el caso anterior, la forma de resolver la situación es extrayendo los atributos que no permiten cumplir con lo señalado por la 2NF. Ahora, la tercera forma normal (3NF) dice que ningún atributo no-primario (es decir que no forma parte de la clave) es dependiente transitivamente de una clave

[3] Se refiere al proceso de diseñar una base de datos, usualmente partiendo de un documento fuente. El cual se trataría de cualquier reporte donde se puedan identificar diferentes entidades relacionadas. Para unja mejor comprensión, una Entidad es cualquier cosa donde se pueda guardar información. Las entidades se localizan viendo los atributos que las describen. De esta forma, los atributos de las entidades son las características que describen a una entidad.

primaria. La misma Tabla 3, para el caso del atributo "Fecha ultimo chat", es un atributo que depende de toda la clave, pero de forma transitiva, esto se produce dado que este atributo depende funcionalmente del atributo "Conversación con match", si no hay conversación, no existe fecha de conversación. Entonces, nuestra entidad (Tabla 3) quedaría con solo dos atributos, los que conforman la clave. Y, en una nueva entidad (Tabla 4) podemos registrar la conversación y la fecha del ultimo chat.

Tabla 3.

Nombre de Usuario	Usuarios en Match
Usted	User A
Sus compañeros	User B,
Sus compañeros	User C
Yo	User A
Otros usuarios del resto del mundo	User A,
Otros usuarios del resto del mundo	User C,
Otros usuarios del resto del mundo	User E

Tabla 4

Conversación con Match	Fecha Ultimo Chat
XXXXXX	22-01
XXXXXX	17-03
XXXXXX	31-01
XXXXXX	04-07
YYYYYY	12-10
ZZZZZZZ	05-05

Modelo Entidad Relación

Ahora bien, considerando que el almacén de datos, en particular sus entidades ya se encontrarían en 3NF, se hace necesario presentar un modelo de base de datos relacional, también llamado modelo entidad relación[4], el cual se trataría de una representación lógica de lo que podemos construir dentro de un sistema (software) gestor de bases de datos (Chen, 1976). Lo ya mencionado es de suma importancia, para lograr un modelado correcto, se debe validar que el conjunto de las relaciones cumpla con las reglas que establecen las formas normales con tal de eliminar la inconsistencia y redundancia de los datos (Elmasri y Navathe, 2007; Date, 1993; Silberschatz, Korth, Sudarshan, 2006).

En la misión de construir un almacén de datos que optimice el trabajo, para Martin (2009) es significativo usar nombres que dejen ver intenciones. Por tanto, se encomienda que los nombres de los atributos de una entidad (también la propia entidad) faciliten el entendimiento del dato u objeto referenciado. Si un nombre requiere un comentario, significa que no revela su contenido (Martin, 2009). Esta regla se orienta a la comprensión del modelo propuesto, logrando esclarecer el contenido de una entidad o de sus atributos.

El uso de la notación *snake_case*, en la cual los nombres de atributos y entidades se definen con palabras que se separan por un guion bajo (_), se basa en los resultados de un estudio empírico presentado en la 18º Conferencia Internacional sobre Comprensión de Programas de la IEEE (Institute Engineers of Electrical and Electronics) del año 2010. Esta investigación mostro evidencia que permite señalar que las convenciones afectan la comprensión de código. En particular, el estudio muestra resultados significativos en favor de la comprensión de un modelo cuando es utilizada esta propuesta de nombre compuesto por más de una palabra y unidos por guion bajo (Sharif y Maletic, 2010).

Por tanto, nuestro ejemplo este compuesto por las entidades Usuario (Tabla 1), Contacto_Usuario (Tabla 2), Match_Logrados (Tabla 3) y Registro_Conversaciones (Tabla4). Para el caso de los atributos, se propone que el lector pueda asignar sus propios nombres en acuerdo a la regla ya presentada. Sin embargo, es necesario aclarar que la nomenclatura propuesta para los nombres de los atributos señala que estos deben ser sustantivos en singular. Si utilizamos plural se puede entender como que se trataría de un atributo con un valor NO Atómico (en contra de la 1NF).

Cardinalidad de las relaciones

Cardinalidad hace referencia a la cantidad de filas (también reconocidos como registros o tuplas) pertenecientes a una entidad y que estarían relacionados con las tuplas pertenecientes a otra entidad. Esto se visualiza en el contexto de una relación binaria (solo entre dos entidades). Las formas que puede tomar la cardinalidad son de tres tipos. Existen cardinalidades del tipo uno es a uno, donde solo una tupla de una entidad se relación con solo una tupla de otra entidad. Dada las entidades Esposo y Esposa, una tupla de la entidad esposo se relaciona

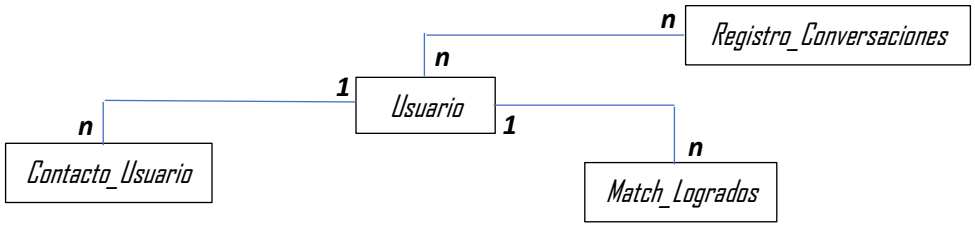
[4] Reconocido también como ERD (en inglés Entity Relationship Diagram), se trata de un diagrama que muestra cómo se deben relacionar las entidades que componen una base de datos.

con solo una tupla de la entidad Esposa (relación habitual en aquellos estados-nación donde la monogamia [5]es la norma). Un segundo tipo de cardinalidad es aquella reconocida como de uno a muchos ó muchos a uno. Este tipo de cardinalidad es la que se puede encontrar entre la relación de una Madre o Padre con su(s) Hijos (as). Finalmente, la tercera y última forma que puede tomar la cardinalidad es aquella reconocida como de muchos a muchos. Supongo el caso de la relación Tío (a) y Sobrino (a).

Retomando nuestro ejemplo de Tinder, dada las cuatro entidades resultantes del proceso de normalización, las relaciones que se pueden encontrar y, sus respectivas cardinalidades, serían las siguientes:

- Usuario (Tabla 1) se relaciona con Contacto_Usuario (Tabla 2); cuya cardinalidad es de uno a muchos (1: n).
- Usuario (Tabla 1) se relaciona con Registro_Conversaciones (Tabla4); cuya cardinalidad es de muchos a muchos (n: n).
- Usuario (Tabla 1) se relaciona con Match_Logrados (Tabla 3); cuya cardinalidad es de uno a muchos (1: n).

Una vez declara la lista de relaciones que componen nuestro modelo entidad relación, su representación grafica se detalla en la siguiente imagen.



Por último, el derrotero hace la construcción de un modelo físico de la base de datos que representa nuestro modelo entidad relación, debe considerar el hecho que una entidad, participante de en una relación, puede adquirir un papel caracterizado por su fortaleza o debilidad frente a la relación que se enfrenta. En concreto, una entidad débil es aquella que no puede existir sin participar en la relación; es decir, aquella que no puede ser unívocamente identificada solamente por sus atributos. Suponga la relación Padre/Madre e Hijo(a), esta relación está marcada por la fortaleza de la primera entidad, sin su existencia, es imposible que la segunda pueda existir. Frente a este hecho, cuando construimos el modelo físico de la base de datos, la entidad débil "hereda" la *Primary Key* de la entidad fuerte, asumiendo dicho atributo con el rotulo de *Foreing Key* (Clave Foránea). Y, su principal objetivo es el de agilizar los procesos de búsqueda dentro de la base de datos. Ahora, frente a la presencia de una relación con cardinalidad n: n, en el momento de construir el modelo físico de la base de datos, esta relación se transforma en una nueva entidad heredando las *PK* de ambas entidades que conforman dicha relación.

A modo de cierre, la presentación de este trabajo espera contribuir a la importancia que se debe asignar al proceso de normalización de las bases de datos (Giménez, 2019), cual es sustento teórico que existe tras las formas normales de Codd, la necesaria comprensión sobre el impacto que provocaría el uso indiscriminado de claves candidatas artificiales y, la difícil tarea de enseñar el modelamiento de una base de datos (Damiano *et al.*, 2019). Esto último sirve como excusa para mencionar la enorme utilidad que prestan los aplicativos que hoy existen en la red, con instrucciones claras e intuitivas para la mejor comprensión del proceso de normalización, notaria son las características de LDBN[6], software que reúne gran parte de las características que los docentes de bases de datos esperarían de una herramienta de apoyo a su gestión dentro del aula (Espinal y Puebla, 2010).

Consideraciones Finales

Una base de datos es la representación de un contexto de la vida real, esto nos lleva a considerar que cuando se crean entidades de datos y, sus respectivos atributos, solo haga uso de lo existente en el contexto estudiado.

[5] Modelo de relaciones afectivo-sexuales basado en un ideal de exclusividad sexual por un periodo de tiempo indefinido entre dos personas unidas por un vínculo sancionado por el matrimonio, por la ley o por el derecho consuetudinario.

[6] LDBN - Learn DataBase Normalization, software en plataforma que LDBN fue desarrollado por Nikolay Georgiev (2008) como parte de su tesis de maestría en la Universidad de Umeå, Suecia, bajo la supervisión de Stephen J. Hegner. Disponible en http://ldbnonline.com/Ldbn.html .

Evitar crear entidades y, particularmente crear atributos, se transforma en una distorsión de la realidad. Este hecho provocara más inconvenientes más que beneficios para la comprensión del modelo de datos propuesto.

Lo importancia de normalizar una base de datos radica en la posibilidad de evitar la redundancia e inconsistencia de los datos que su modelo debe gestionar. Pese a que el proceso de normalización implica aumentar de forma significativa la cantidad de entidades de datos del modelo propuesto originalmente, esta medida beneficia particularmente el tiempo de respuesta que si sistema mostrara durante los procesos de búsqueda de información.

Las formas normales propuestas por Codd se basan en entidades de datos con dependencias funcionales y transitivas conocidas. A lo anterior se adiciona la presencia de claves primarias candidatas que no necesariamente se componen por solo un atributo. A partir de la segunda forma normal, la propuesta de Codd se estructura en base a entidades de datos cuya clave primaria estaría compuesta por dos o más atributos. Un modelo de datos que considera la presencia de entidades con claves primarias constituidas por solo un atributo, de forma natural se encontrarían en tercera forma normal.

Referencias

Elmasri, R. y Navathe, S. (2007). *Fundamentos de Sistemas de Bases de Datos*. España: Pearson.

Espinal, Y., y Puebla, M. (2010). Sistema para la integración del proceso de normalización de bases de datos relacionales con gestores de bases de datos (sinorges). *Avances en Sistemas e Informática*. 7 (3), 17-25.

Damiano, L., Muñoz, R., Maldonado, C., Romero, S., Guevara, A., Bueno, M., Quinteros, S. y Peretti, J. (2019). Propuestas para enseñar el diseño de estructuras de datos en bases de datos relacionales. En I Simposio Argentino de Educación en Informática (SAEI 2019)-JAIIO 48 (Salta) (pp. 71-84).

Date, C. (1993). *Introducción a los Sistemas de Bases de Datos (Volumen 1)*. Estados Unidos: Addison Wesley.

Silberschatz, A., Korth, H., Sudarshan, S. (2006). *Fundamentos de Bases de Datos*. Estados Unidos: Mc Graw Hill.

Sharif, B., Maletic, J. (2010). An Eye Tracking Study on camelCase and under_score Identifier Styles. En *2010 IEEE 18th International Conference on Program Comprehension* (pp. 196-205).

Martin, R. (2009). *Código limpio*. España: Prentice Hall.

Chen, P. P. (1976). The Entity—Relationship Model—Towards a. Unified View of Data. *ACM Transactions on Database Systems*. 1 (1), 311-339.

Oppel A. (2010). *Fundamentos de base de datos*. Mexico, D.F.: Mc Graw Hill.

Giménez, J. (2019). Buenas prácticas en el diseño de bases de datos. Arandu-UTIC. *Revista Científica Internacional de la Universidad Tecnológica Intercontinental*, 6(1), 193-210.

Hueso I. (2014). *Gestión de bases de datos*. España: RA-MA Editorial.