XV JENUI. Barcelona, 8-10 de julio de 2009
ISBN: 978-84-692-2758-9
http://jenui2009.fib.upc.edu/

¿Todavía interesa normalizar bases de datos? Reflexionando sobre su enseñanza

César Domínguez Pérez[1], Arturo Jaime Elizondo[1], Tomás A. Pérez Fernández[2]
(1) Dpto. de Matemáticas y Computación de la Universidad de la Rioja. Ed. Vives, Luis de Ulloa s/n, 26004 Logroño
{cesar.dominguez, arturo.jaime}@unirioja.es
(2) Dpto. de Lenguajes y Sistemas Informáticos de la Universidad del País Vasco. Fac. Informática de San Sebastián, P. Lardizabal 1, 20018 Donostia-San Sebastián
tomas.perez@ehu.es

Resumen

El objetivo principal de este trabajo es hacer una reflexión sobre la normalización de bases de datos en la actualidad, incluyendo su utilidad práctica, el enfoque seguido en su enseñanza y su utilización en proyectos reales. Identificamos cómo muchos libros de fundamentos de bases de datos pasan por alto aspectos fundamentales en el proceso de normalización como el cálculo de las claves de una relación. Presentamos un algoritmo informal de obtención de claves basado en una representación gráfica que nos resulta útil para la docencia. Incluimos un estudio de herramientas relacionadas con la normalización y finalmente describimos casos prácticos que nos permiten reflexionar sobre cuestiones poco claras relacionadas con esta actividad.

1. Introducción

La normalización de bases de datos (BD) fue introducida por Edgar F. Codd a principios de los 70 [4] poco después de crear el modelo relacional [3]. La normalización pretendía añadir calidad a los diseños de BD que seguían el modelo relacional, evitando la aparición de datos redundantes y los problemas de actualización que conllevan. Normalizar esquemas de relación (tablas) de una BD supone, en primer lugar, comprobar una serie de normas, llamadas formas normales, y en segundo lugar, descomponer los esquemas que no las cumplen para obtener esquemas normalizados.

Hoy en día, y tras casi 40 años desde su aparición, el modelo relacional goza de gran aceptación. La mayoría de BD diseñadas actualmente son relacionales. Sin embargo, ¿podemos afirmar lo mismo de la normalización?

El interés de la normalización figura de forma explícita en las recomendaciones de la ACM sobre el currículo en informática [1]. La bibliografía especializada en BD [8,5,6] le concede siempre un espacio razonable. Además, es habitual encontrar un capítulo sobre normalización en los programas de titulaciones españolas en informática. Por todo lo anterior, da la sensación de ser un tema vigente. Lo que no está tan claro es si las técnicas de normalización se aplican con tanta profusión en el mundo profesional. Nuestra experiencia incluye los proyectos que realizan nuestros alumnos en las empresas y lo que nos trasladan algunos egresados y conocidos. Nos da la sensación de que el interés de terminar los proyectos a tiempo, el énfasis central en el diseño conceptual y quizá cierta percepción de actividad poco relevante puede estar relegando la normalización de BD al olvido. De estar en lo cierto, cabe preguntarnos si realmente merece la pena el esfuerzo de aprendizaje de estas técnicas.

Este trabajo pretende hacer una reflexión sobre la enseñanza del tema: qué podemos mejorar en su enfoque para desmitificar la actividad (como algo que parece "muy" teórico con "mucha" matemática) y al mismo tiempo facilitar su aplicación, por ejemplo, difundiendo qué herramientas son útiles tanto si el objetivo es aprender como si es automatizar la actividad.

Cuando nos preparamos este tema, hace unos años, nos llamó la atención identificar algunas cuestiones que se daban por supuestas. Una de las más llamativas es la detección de claves de una relación. Tampoco se presentan, en general, ejemplos asociados a dominios concretos, al estilo de los utilizados en capítulos sobre diseño conceptual. Si los autores no nos muestran algún ejemplo cercano a la realidad donde se ilustre su utilidad ¿conseguirán motivar al lector (futuro profesional) sobre su interés práctico?

Libro	Total nº páginas	Normalización nº páginas	%	Cuestiones/ Ejercicios	Algoritmo claves
Elmasri y Navathe 2007 [8]	988	74	7,49%	72	No
Connolly y Begg 2007 [5]	1269	56	4,41%	17	No
Date 2004 [6]	983	74	7,53%	30	No
Silberschatz y otros 2007 [22]	522	40	7,66%	30	No
Rob y Coronel 2004 [19]	838	40	4,77%	30	No
Kroenke 2003 [11]	671	30	4,47%	25	No
Piattini y otros 2006 [16]	946	184	19,45%	12	**Sí**
Rivero y otros 2005 [18]	574	101	17,60%	22	**Sí**
Promedio	**848,9**	**74,9**	**9,2%**	**29,7**	

Tabla 1. Tratamiento bibliográfico de la normalización de BD

En este trabajo, vamos a tratar de defender la vigencia y la utilidad de la normalización y el interés de dotar al enfoque de su enseñanza de una visión más cercana a la ingeniería. El artículo se estructura de la siguiente forma. En la siguiente sección presentamos un estudio del tratamiento de la normalización en varios libros sobre fundamentos de BD. En la sección 3 presentamos un algoritmo para la obtención de las claves de una relación y en la sección 4 hacemos un breve estudio de otros algoritmos de detección de claves. La sección 5 contiene un estudio comparativo de herramientas relacionadas con la normalización de BD. En la sección 6 planteamos una serie de dudas y problemas encontrados normalizando. El artículo termina con una sección de conclusiones.

2. La normalización en la bibliografía

La Tabla 1 recoge una revisión del tratamiento de la normalización en varios libros utilizados en la enseñanza universitaria de BD. Reflejamos el número total de páginas del libro y el número de páginas dedicadas a la normalización. Se contabilizan también cuántas cuestiones y ejercicios se plantean en cada obra sobre este tema y si se detalla o no algún algoritmo de obtención de claves.

El análisis de la tabla permite observar que, en general se dedica bastante espacio a la exposición del tema. Además se incluye un número razonable de cuestiones y ejercicios, artificiales en su mayoría. También podemos observar cómo la identificación de las claves es mayoritariamente ignorada. Esto nos sorprende, ya que la distinción entre los atributos que forman parte de alguna clave y los que no, es una cuestión central en el proceso de normalización. Tan sólo dos de los libros analizados proponen un algoritmo de obtención de claves. Casualmente el original de ambos está escrito en español.

Desde nuestro punto de vista hay dos tipos de lectores interesados en temas como la normalización. Por un lado estamos los lectores académicos, tanto profesores como alumnos. Nos interesa que los conceptos estén bien explicados, con ejemplos claros y un conjunto de ejercicios que nos permita profundizar en lo aprendido. Por otro lado están los profesionales, diseñadores de BD, que necesiten revisar estos conceptos. Creemos que, en general, la bibliografía podría mejorar su enfoque hacia ambos colectivos. Quizá incluyendo un ejemplo completo, donde se muestre cómo se van dando todos los pasos (incluyendo la detección de claves) y basado en un dominio (sin usar letras en lugar de nombres de atributo).

También es interesante que la bibliografía oriente sobre aquellas herramientas existentes para realizar ésta y otras actividades. Creemos que disponer de esta información sería de gran ayuda, evitando la inversión de un tiempo considerable rastreando la red sin saber a ciencia cierta si existe o no algo de verdadera utilidad.

3. Algoritmo que obtiene todas las claves de una relación

La determinación de las claves de una relación es esencial para aplicar el proceso de normalización. Por ejemplo, para identificar las DF que no están en 2FN hay que conocer qué atributos no forman parte de ninguna clave de la relación. A pesar de ello, acabamos de ver que los libros orientados a la enseñanza universitaria de BD no suelen

presentar algoritmos para obtener dichas claves. Ante esta situación, es probable que el lector entienda que tal actividad es trivial. Consideremos el caso de la Figura 1, donde se ha utilizado la misma notación que [8] para las DF. Cada línea horizontal corresponde a una DF. Por ejemplo, la línea cuyas flechas apuntan a *cubierta* e *item* representa la DF *{signatura, ejemplar}* → *{cubierta, item}*. Esta relación pertenece a una BD para una biblioteca. *Cubierta* contiene el valor "rustica" o "rígida". *Item* es un código asignado a cada volumen. *Signatura* es un código para clasificar las obras. *Ejemplar* distingue los volúmenes de una misma obra. *Estante* es un código asignado a cada anaquel. *Sección* es un código que identifica un área de la biblioteca. La identificación de *todas* las claves de esta relación no resulta evidente.

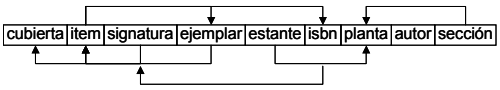

Figura 1. DF de la relación usada como ejemplo (cobertura mínima).

El algoritmo más simple consistiría en buscar claves entre todas las posibles opciones (todos los subconjuntos de atributos de la relación). Esto supondría hacer $\sum_{i=1..n} C_n^i$ (combinaciones) cálculos de cierres de subconjuntos, donde n es el número de atributos de la relación. Para el ejemplo de la Figura 1 son 510 pruebas, lo cual no resulta práctico.

Desde hace unos años utilizamos un algoritmo informal (sin demostración formal de corrección) que nos ha funcionado en todos los casos estudiados. El algoritmo se basa en una interpretación gráfica e intuitiva, útil para nuestros propósitos docentes. En este trabajo no pretendemos describir en detalle los pasos del algoritmo y su casuística. Mostraremos, sobre el caso de la Figura 1, un ejemplo de cómo se pueden identificar las claves basándonos en una representación simple de las DF. Dicha representación, denominada en [16] *matriz de implicaciones,* se muestra en la Figura 2. Partimos siempre de una *cobertura mínima* de las DF.

La matriz de implicaciones consta de una columna para cada atributo de la relación y una fila para cada DF, etiquetada con su parte izquierda. Cuando un atributo (columna) está implicado por la parte izquierda de una DF (fila) lo representamos mediante un punto. El signo $^+$ se debe a que los puntos de cada fila determinan el cierre del conjunto de atributos de su etiqueta.

En la bibliografía encontramos algoritmos para calcular la cobertura mínima y el cierre [8,16] que no vamos a revisar aquí.

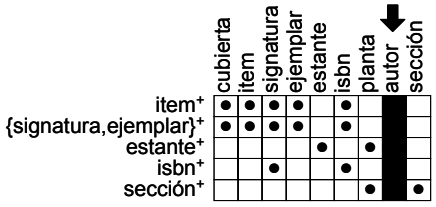

Figura 2. Matriz de implicaciones. Los puntos marcan los atributos implicados por los atributos de la etiqueta de fila. Todas las claves incluirán el atributo *autor*.

Si existe alguna columna sin puntos, *autor* en la Figura 2, el atributo de la columna formará parte de todas las claves. Eliminaremos dicha columna.

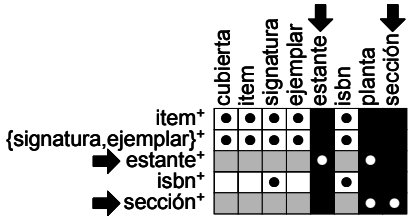

Figura 3. *Estante* y *sección* (etiquetas de filas) formarán parte de todas las claves.

La columna elegida en cada paso es una de las de *menor número de puntos*. Iremos incorporando en las claves los atributos "implicadores" (parte izquierda de una DF) de las columnas que se vayan eligiendo. Por ejemplo, en la Figura 3 la columna *sección* tiene un solo punto (implicado por una sola DF), así que se tomará su "implicador" (fila *sección*) como parte de todas las claves. Cada vez que incluyamos en la clave un "implicador", eliminaremos las columnas de los atributos "alcanzados" por éste. Por ejemplo, en la Figura 3 hemos incluido en la clave el atributo (fila) *sección* y hemos eliminado las columnas *planta* y *sección* alcanzadas por éste. A continuación elegimos la columna *estante* (un solo punto) e incluimos la fila *estante* en todas las

claves. Eliminamos las columnas alcanzadas: *estante*. De momento sabemos que toda clave incluirá los atributos *{autor, sección, estante}*.

Si la columna elegida contiene más de un punto, da pie a identificar diferentes claves. Por ejemplo, si tuviera tres puntos, escribiríamos tres tablas, en cada una habríamos seleccionado una de las DF afectadas y eliminado las columnas alcanzadas por ella. Cada tabla completará el trozo de clave que se tiene hasta el momento de una forma diferente. Este proceso se reitera seleccionando una DF cada vez e incluyendo su parte izquierda en la clave que se está construyendo.

En nuestro ejemplo las siguientes columnas con menos puntos son *cubierta*, *item* y *ejemplar* con dos puntos cada una (ver Figura 4). Además sucede que eligiendo cualquiera de las partes izquierdas de las DF afectadas (*item* o *{signatura, ejemplar}*), alcanzamos todas las columnas que quedan. Por lo tanto, hemos encontrado las claves *{autor, sección, estante, item}* y *{autor, sección, estante, signatura, ejemplar}*.

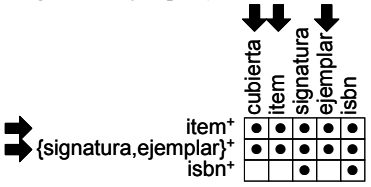

Figura 4. *Item* forma parte de una clave y {signatura, ejemplar} de otra. Cualquiera de los dos grupos completa la determinación de todas las columnas.

El proceso continúa si alguna parte izquierda introducida en una clave está determinada parcialmente por otra DF. Esto ocurre con la parte izquierda *{signatura, ejemplar}* (incluida en la segunda clave) y la DF *isbn* → *signatura*. Entonces obtendremos una nueva clave sustituyendo en la clave *{autor, sección, estante, signatura, ejemplar}* el atributo *signatura* por *isbn*. Por lo tanto las claves descubiertas son *{autor, sección, estante, item}, {autor, sección, estante, signatura, ejemplar}* y *{autor, sección, estante, isbn, ejemplar}*. Hay que tener en cuenta que en algún caso podría repetirse un mismo atributo en una clave y habría que eliminarlo.

Finalmente es necesario comprobar que ninguna de las claves obtenidas esté contenida en otra clave, ya que se trataría de una superclave y no de una clave. Esto último no ocurre en nuestro ejemplo.

4. Otros algoritmos para hallar las claves

Existen algoritmos publicados en revistas de investigación cuya corrección ha sido demostrada formalmente [9,20]. Todos ellos nos parecen inapropiados para su utilización en clase debido a su complejidad o extensión.

Hemos encontrado dos libros que incluyen propuestas informales de algoritmos de obtención de claves comparables con el que acabamos de presentar. El primer libro [16] presenta un algoritmo que simplifica el publicado en [9]. Dicho algoritmo comienza, como el nuestro, con la matriz de implicaciones, pero sólo la utiliza en el paso inicial, continuando con un proceso de cálculo de intersecciones entre los atributos de las filas no alcanzadas. Al prescindir de una representación gráfica quizá es menos intuitivo que el presentado anteriormente. También incluye un procedimiento informal basado en grafos donde se clasifican los atributos según partan o lleguen flechas. Esto sirve para descartarlos de las claves (*cubierta* y *planta* en nuestro ejemplo), o incluirlos en todas las claves (*autor, estante* y *sección*). Sin embargo no queda claro qué hacer con los atributos donde al mismo tiempo parten y llegan flechas (*item, signatura, ejemplar* e *isbn*).

El algoritmo del segundo libro [18] se basa en una representación de las DF en forma de grafo. El grafo se va coloreando a medida que se van construyendo las claves. Como los propios autores citan, la representación puede resultar difícil de usar con ejemplos complicados.

5. Herramientas para normalizar

No es una tarea sencilla encontrar una herramienta adecuada que automatice el proceso de la normalización de BD o que pueda utilizarse en el aprendizaje de esta teoría. En general, los libros sobre BD no hacen referencia a ninguna herramienta y los sistemas gestores de BD no incluyen ninguna utilidad para esta tarea.

Herramienta	Disponible Encontrada	Cierre, cobertura mínima, claves	2FN 3FN FNBC	Muestran proceso	Comentarios
RENO [16]	Sí	Sí	Sólo FNBC	No	Acompaña a libro
NORMIT [14]	Sí	Sí	Sí	Sí	Sin nuevos ejercicios
DBNormalization [21]	Sí	Sí	Sí	Sí	Guía el proceso
lbdn [10]	Sí	No cierre ni claves	Sí	Sí	Proyecto Google code
"Web-based" tool [12]	Sí	No	Hasta 3FN	No	Applet de Java
Micro [7]	No	Si	Si	-	Prototipo
NOCAT [17]	No	Sí	Sí	-	Prototipo
DBDesignTools [13]	No	Sí	Sí	-	Librería en C++
JMathNorm [23]	No	No claves	Si	No	En Mathematica

Tabla 2. Herramientas para la normalización de BD

Una excepción es el libro de Piattini y otros [16] que cita la herramienta RENO (adjuntada en la versión anterior del libro). Esta herramienta tiene una interfaz agradable que permite introducir tus propios problemas y realiza las tareas más habituales de normalización: cálculo del cierre, de la cobertura mínima, de las claves candidatas y tests de formas normales. Además, descompone relaciones en FNBC. Se echa en falta la descomposición a las formas normales intermedias. Su utilidad docente podemos compararla a una calculadora, que muestra el resultado pero no el proceso ni los pasos intermedios para conseguirlo.

La Tabla 2 incluye una selección de las herramientas que hemos encontrado. La tabla incluye si están disponibles para su uso, qué posibilidades incorporan y si muestran los pasos dados para la obtención de los resultados o se comportan al estilo de una calculadora.

Todas las herramientas accesibles, salvo una [14], calculan resultados automáticamente. Algunas, además incluyen determinadas ayudas on-line, que permiten acceder a explicaciones y ejemplos [21,14,10]. Desde nuestro punto de vista, la herramienta más adecuada para el aprendizaje de este tema sería DBNormalization tools [21]. Esta herramienta es software libre y, además, está disponible a través de la web, guía el proceso de obtención de cierres, de claves y de las diferentes formas normales (hasta FNBC). Tanto RENO [16] como lbdn [10] son buenas alternativas aunque no incluyen algunas nociones de normalización.

Utilizando las herramientas nos hemos encontrado algunos resultados inesperados de acuerdo al proceso de normalización que podemos encontrar por ejemplo en [8,5]. El estudio detallado de las diferencias entre los resultados obtenidos por unas y otras herramientas y su razonamiento excede los propósitos de este artículo. En todo caso hay que utilizarlas con cierta cautela puesto que no se nos garantiza la ausencia de errores.

6. Problemas y dudas normalizando BD

En esta sección vamos a presentar una serie de reflexiones, que no pretende ser exhaustiva, sobre el uso de la normalización. Nos apoyaremos en simplificaciones de casos que surgieron en proyectos desarrollados por alumnos, algunos de ellos en empresas, en los que hemos participado bien como tutores o como miembros del tribunal calificador.

6.1. El diseño conceptual

El diseño conceptual, que apareció históricamente después de la normalización [2], concentra quizá el mayor esfuerzo de la tarea de diseño de BD. En esta fase también se consideran las mismas anomalías que resuelve la normalización para tratar de evitarlas. El diseño lógico posterior es una pura transformación del esquema conceptual en un esquema relacional. Si conseguimos un buen diseño conceptual, el diseño lógico resultante cumplirá en la mayoría de los casos los test de normalización habituales (hasta FNBC) ¿Quiere esto decir que podemos prescindir de normalizar? Hay que tener en cuenta que en estos casos el test de las formas normales suele ser fácilmente comprobable. Por lo tanto, desde nuestro punto de vista, incluso en este caso merecerá la pena el esfuerzo de normalizar y documentar en qué forma normal quedan nuestras tablas.

6.2. Esquemas que no están en 3FN ó FNBC

No siempre el esquema relacional resultante de los diseños conceptual y lógico está en 3FN ó FNBC. En esos casos, cuando la normalización merecerá "de verdad" la pena, también pueden surgir dudas.

El caso (frecuente) de la Figura 5 se le presentó a un alumno mientras realizaba un proyecto en una empresa. La relación PERSONA de la parte superior, incluye la dirección y el código postal y, por lo tanto, existe la DF reflejada en la misma[1]. Esto supone que dicha relación no está en 3FN.

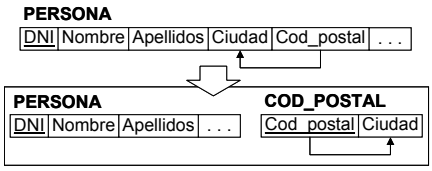

Figura 5. La DF entre código postal y parte de la dirección no está en 3FN.

Sabemos que si dejamos la dirección y el código postal en la misma tabla cabe la posibilidad de que un mismo código postal acabe figurando, por error, junto a distintas ciudades. Por ello, el alumno decidió dividir la tabla como se muestra en la parte inferior de la Figura 5.

¿Es correcto este último diseño? Si lo es, ¿por qué encontramos el código postal junto a la dirección en ejemplos propuestos en libros que tratan el diseño de BD (normalización incluida) [22,19,5]? ¿El hecho de que la ciudad asociada a cada código postal parezca mantenerse inalterable da pie a evitar esta división?

6.3. Detección de errores de diseño conceptual

Si el proceso de normalización se complica podría deberse a errores cometidos en el diseño conceptual. La determinación de las DF exige reflexionar por segunda vez sobre las propiedades inherentes al dominio en el que estamos trabajando. Así que podría surgir algún detalle que se nos hubiera pasado por alto.

Un alumno que realizó un proyecto sobre campeonatos de motociclismo se encontró con el caso de la Figura 6, que recoge parte del diseño conceptual y su transformación a tablas. Vemos que las claves ajenas IdMecánico e IdEscudería aparecen juntas en la tabla PILOTO. Al determinar las DF, el alumno observó que cada mecánico pertenece a una sola escudería. Esta DF ocasiona que la tabla no cumpla 3FN y, por lo tanto, hubo de dividirla como se muestra en la propia Figura 6.

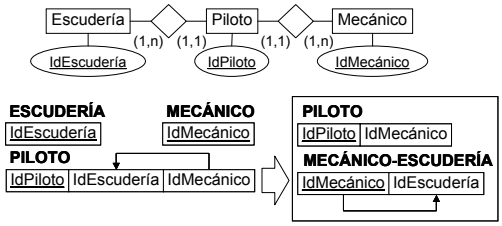

Figura 6. DF entre varias claves ajenas.

La tabla MECANICO-ESCUDERÍA contiene una relación entre mecánicos y escuderías no detectada en el diseño conceptual. Podemos corregir dicho diseño obteniendo el de la Figura 7, donde se incluye la relación entre mecánico y escudería y se prescinde de la relación entre piloto y escudería, ahora innecesaria.

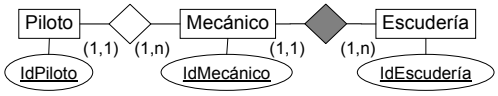

Figura 7. La solución en el esquema conceptual.

6.4. Desnormalizar o no desnormalizar

Desnormalizar consiste en renunciar a tener la BD en una forma normal más alta en beneficio de la eficiencia. La normalización exige a veces situar en diferentes tablas datos relacionados entre sí. Volverlos a juntar supone ejecutar la operación de reunión (*join*). Si se decide mantener dichos datos juntos en la misma tabla, se ahorran tales operaciones. Ello supone documentar y gestionar correctamente la redundancia de datos que ello implica. Esta actividad se sitúa habitualmente dentro del denominado diseño físico de BD.

El ejemplo de la Figura 8 surgió en un proyecto de fin de carrera donde se desarrollaba una aplicación de gestión de un almacén. La BD registraba, entre otras cosas, datos de los representantes. Como vemos en la figura la tabla incluye un comentario o descripción (por ejemplo: llamar por la tarde) para cada uno de los atributos:

[1] El conjunto de atributos subrayados de cada relación representará su clave única.

teléfono, fax, e-mail y web. De manera excepcional, varios representantes pueden compartir el mismo dispositivo y la descripción será siempre la misma para todos ellos.

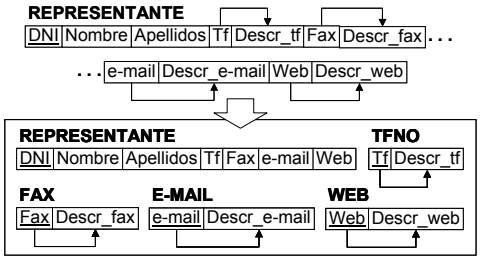

Figura 8. DF con repeticiones excepcionales de los valores de sus partes izquierdas.

Las DF señaladas en la tabla de la parte superior de la Figura 8 no están en 3FN. Por lo tanto la relación se divide conforme se muestra en la parte inferior de la misma figura, quedando la relación original totalmente fragmentada.

Al ser pocos los casos donde varios representantes comparten dispositivo, el alumno decidió quedarse con la tabla original en 2FN. Como sabemos, esto puede producir anomalías en los datos aunque el diseño se vea más compacto y quizá hasta más fácil de interpretar. ¿Estamos tomando la decisión correcta o deberíamos escuchar opiniones recientes tendentes a no desnormalizar nunca [15]? Aunque se pueden proponer mejores diseños conceptuales, por ejemplo generalizando los cuatro tipos de dispositivos en una entidad común, tales diseños no surgen aplicando normalización.

6.5. Normalizando sólo con la clave principal

Algunos autores consideran como una opción la normalización restringida a las claves principales de las relaciones [8,5,19]. Ello da lugar a ciertas confusiones. Por ejemplo, si juntamos la idea anterior con la definición de una clave principal artificial por cada relación (por ejemplo, de tipo auto-numerado), conseguimos hacer que la normalización sea una tarea trivial. En el ejemplo de la Figura 1 la tabla incumple 2FN. Sin embargo, al introducir el atributo artificial Id como clave principal, si sólo consideramos las DF asociadas a dicha clave: ¡la tabla estaría en FNBC! No pretendemos aquí valorar la utilidad de tales claves artificiales, si no que cuando se utilicen, debería seguirse teniendo en cuenta el resto de DF en el proceso de normalización.

6.6. Errores de normalización y diseño de BD

Los errores que se cometen durante la normalización, pueden afectar muy negativamente al diseño final de la BD. Como además se han seguido una serie de pasos, no del todo intuitivos, puede resultar difícil detectar el problema.

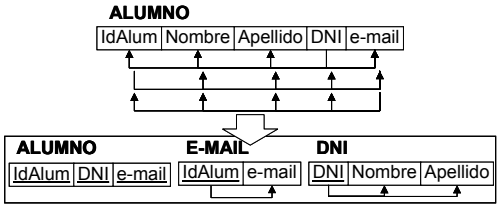

Figura 9. Error tremendo normalizando una tabla.

La Figura 9 presenta la normalización realizada por un grupo de alumnos que diseñaba una BD para un proyecto sobre una academia. Aunque la tabla inicial está en FNBC, identificaron erróneamente como única clave {IdAlum, e-mail, DNI} en lugar de tres claves tomando a cada atributo por separado. La división en tres tablas mostrada en la figura es, sin duda, de difícil interpretación. Sorprendentemente, ninguno de los tres alumnos sospechó que pudieran haber cometido un error.

7. Conclusiones

En este trabajo hemos reflexionado sobre la utilidad de la normalización de BD y el enfoque que se sigue en su docencia. A pesar de la experiencia acumulada en la enseñanza y aplicación de este tema encontramos todavía ciertos déficits en su tratamiento bibliográfico, en los algoritmos existentes y en la difusión de las herramientas disponibles. Para tratar de paliar esta situación hemos mostrado un algoritmo informal de cálculo de todas las claves basado en la tabla de implicaciones que utilizamos en nuestra práctica docente. Lo hemos comparado, desde el punto de vista práctico, con otros algoritmos existentes. También hemos realizado un estudio comparativo de las herramientas disponibles para el estudio y cálculo de formas normales. Por último, hemos mostrado algunos casos prácticos donde reflexionamos sobre cuestiones que no parecen estar claras en la bibliografía de BD.

Como trabajo futuro nos proponemos formalizar el algoritmo de cálculo de claves y compararlo en detalle con otros algoritmos existentes. También nos planteamos estudiar la usabilidad del algoritmo a través de la opinión de los alumnos. La sensación subjetiva es que los alumnos se sienten más cómodos utilizando el algoritmo y no encuentran problemas al aplicarlo.

Agradecimientos

Parcialmente subvencionado por la Universidad de La Rioja, proyecto API09/05.

Referencias

[1] ACM Computing Curricula Recommendations. http://www.acm.org/education/curricula-recommendations

[2] Chen, P. *The entity-relationship model: Toward a unified view of data*. ACM Transactions on Database Systems, 1 (1), pp. 9-36, 1976.

[3] Codd, E.F. *A relational model of data for large shared data banks*. Communications of the ACM, 13 (6), pp. 377-387. 1970.

[4] Codd, E.F. *Further normalization of the data base relational model*. Data Base Systems: Courant Computer Science Symposia Series 6, pp 33-64. Prentice Hall, 1971.

[5] Connolly, T.M., Begg, C.E. *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión*. 4ª Edición. Addison Wesley, 2005.

[6] Date, C.J. *An introduction to database systems*. 8ª Edición. Prentice Hall, 2004.

[7] Du, H., Wery L. *Micro: A normalization tool for relational database designers*. Journal of Network and Computer Applications, 22 (4) 1999, pp 215-232.

[8] Elmasri, R., Navathe, S.B. *Fundamentos de sistemas de bases de datos*. 5ª Edición. Addison Wesley, 2007.

[9] Fadous, R., Forsyth, J. *Finding candidate keys for relational data bases*. En Proc. ACM SIGMOD International Conference on Management of Data, pp. 203-210, 1975.

[10] Georgiev, N. *A web-based environment for learning normalization of relational database schemata*. 2008. http://code.google.com/p/ldbn

[11] Kroenke, D.M. *Procesamiento de bases de datos*. 8ª Edición. Prentice Hall, 2003.

[12] Kung, H.J., Tung, H.L. *A web-based tool to enhance teaching/learning database normalization*. En Proc. of the 2006 Southern Association for Information Systems Conference, pp. 251-258, 2006. http://personal.georgiasouthern.edu/~hjkung/3NF/

[13] Martín, O. *Documentación de DBDesign Tools*. 2006. http://sourceforge.net/projects/dbdesigntools/

[14] Mitrovic, A. *The effect of explaining on learning: a case study with a data normalization tutor*. En Proc. AIED2005, pp. 499-506, IOS Press, 2005. http://www.cosc.canterbury.ac.nz/tanja. mitrovic/normit.html

[15] Pascal, F. *The costly illusion: normalization, integrity and performance*. Practical Database Foundations Series, 2005.

[16] Piattini, M., Marcos, E., Calero, C., Vela, B. *Tecnología y diseño de bases de datos*. RAMA, 2006. http://www.dbdebunk.com/publications.html.

[17] Pouyioutas, P. *NOCAT—a normalisation CASE tool*. Transactions on Information and Communications Technologies, 10, pp. 277-285, 1995.

[18] Rivero, E., Martínez, L., Alonso, I. *Bases de datos relacionales: Fundamentos y diseño lógico*. Colección Ingeniería, 20. Publicaciones de la Universidad Pontificia Comillas, 2005.

[19] Rob, P., Coronel, C. *Sistemas de bases de datos. Diseño, implementación y administración*. Thomson , 2004.

[20] Saiedian, H., Spencer, T. *An efficient algorithm to compute the candidate keys of a relational database schema*. The Computer Journal, 39 (2), pp. 124-132, 1996.

[21] Selikoff, S. *The Database Normalization Tool*. 2003. http://dbtools.cs.cornell.edu/norm_index.html

[22] Silberschatz, A., Korth, H.F., Sudarshan, S. *Fundamentos de diseño de bases de datos*. 5ª Edición. Mc Graw Hill, 2007.

[23] Yazici, A., Karakaya, Z. *JMathNorm: A Database Normalization Tool Using Mathematica*. En Procc. ICCS 2007, Part II, LNCS 4488, pp. 186-193, 2007.