Incremento de Semántica en Fases Iniciales de Transformación de Modelos de Datos

Viviana Ferraggine[1], Federico Casanova[2] y Pablo Rinaldi[2]

1 INTIA, Facultad de Ciencias Exactas, U.N.C.P.B.A., Paraje Arroyo Seco
Tandil, Buenos Aires, Argentina
2 PLADEMA, C.I.C. - U.N.C.P.B.A., Paraje Arroyo Seco
Tandil, Buenos Aires, Argentina
{vferra, fcasano, prinaldi} @exa.unicen.edu.ar

Abstract. El Diagrama de Entidades y Relaciones Extendido es una pieza clave en el modelado de datos, a partir del cual mediante un proceso de transformación genera el Esquema Físico de Datos basado en un Modelo Objeto-Relacional. En el presente trabajo se identifican y se proponen soluciones a varios problemas encontrados en dicho proceso, originados por la falta de herramientas y metodologías que permitan expresar la semántica de determinadas situaciones. Varias de estas soluciones son aun objeto de especificación, en este trabajo se detallan las fases iniciales del proceso de transformación sobre las cuales ya se ha cubierto completamente dicha especificación y se describen las transformaciones necesarias en las siguientes fases.

Keywords Modelo de Entidades y Relaciones Extendido, Esquema Lógico Estandar, Esquemas Físico de Datos, SQL.

1 Introducción

Las fases de modelado conceptual, lógico y físico de datos son fundamentales en el proceso de diseño de bases de datos relacional u objeto-relacional. Para lograr un diseño completo y exacto, se necesitan metodologías que permitan representar y trasladar todos los aspectos relevantes del Universo de Discurso (UdeD) al esquema físico final [2], [3], [6], [10].

El modelado conceptual de datos permite a los diseñadores reconocer y representar los objetos del negocio y los vínculos semánticos entre ellos, y sobre esa base poder proporcionar un mapeo directo entre el mundo real percibido y su representación, sin distorsión o ambigüedad [16].

Tanto el Modelo de Entidades y Relaciones original (MER) [5], como su extensión más difundida, el Modelo de Entidades y Relaciones Extendido (MERExt), simbolizan conceptos de datos en términos de entidades y relaciones [11], [23], [24]. El minimalismo de su notación ha contribuido a su éxito, proporcionando

mecanismos de abstracción suficientes para especificar ciertas reglas del negocio y restricciones inherentes, y los lenguajes gráficos o visuales como el Diagrama de Entidades y Relaciones Extendido (*DERExt*) facilitan la discusión, comunicación y validación por parte de los expertos en el dominio [7], [8]. Dichos *DERExt* pasarán luego a formar parte de la documentación de diseño y desarrollo del proyecto, resultando en piezas claves a futuro para el entendimiento de la semántica del modelo de datos plasmado en el esquema físico.

Muchos autores han escrito acerca de las reglas de transformación del *DERExt* al Esquema Relacional (Esq R) proponiendo algoritmos con diferentes variantes [4], [9], [11], [14], [23], [24]. En particular, algunos reconocen la etapa de diseño lógico estándar, cuya definición resulta la más apropiada para la etapa de transformación del *DERExt* a un Esquema Lógico Estándar (*ELE*) inicial, basado en un Modelo Objeto-Relacional [9]. Estos algoritmos resultan mejorables ya que una variedad de conceptos plasmados en el *DERExt* se pierde al obtener el *ELE*, y por su incapacidad de permitir, de ser necesario, un proceso inverso de reconstrucción sin ambigüedades de un *DERExt* a partir de un *ELE* o de un Esquema Físico de Datos (*EFD*).

Como mencionamos, el problema surge cuando se intentan aplicar las reglas de transformación y mantener la semántica plasmada en el *DERExt* en la siguiente fase de la metodología, esto es debido a que en su gran mayoría utilizan el modelo relacional o el modelo binario para transformar el *DERExt* en un *EFD*. Las reglas que se encuentran en los textos por lo general sólo especifican transformaciones de elementos simples del *DERExt*, esto conlleva a una pérdida importante de semántica en dicho proceso, dejando en manos de quien implementa el esquema físico en la base de datos la definición de mecanismos para controlar y comprobar los aspectos que no pudieron ser representados.

Si bien se han propuesto innovaciones en la notación del *ELE* [17], basándose en reglas de construcción del *DERExt* y extendiendo el algoritmo para su conversión en un *ELE* enriquecido [18], [19], aún hay aspectos y construcciones que no han sido suficientemente explorados ni analizados.

En este trabajo se propone una forma de representar en el *ELE* todos los aspectos incluidos en el *DERExt*, enriqueciendo la notación de las reglas conocidas para la construcción del mismo, y extendiendo el algoritmo de transformación a fin de resolver las cuestiones relativas a ambigüedades y falta de ortogonalidad y cumplir plenamente con las cualidades de completitud que se pretenden de la metodología. Se tomaron como base las reglas de transformación más difundidas [4], [11], [15], [21], [22], [23], [24]. Esto permite que dicho *ELE* pueda ser traducido automáticamente a un *EFD*, utilizando los recursos del lenguaje SQL declarativo. Esta traducción se realiza pasando por una etapa intermedia denominada Esquema Lógico Estándar Tardío (ELET) que permite incorporar ya decisiones de implementación respecto del SQL a utilizar, sin perder el nivel de abstracción en la etapa final.

2 Propuesta de Mejoras al Proceso de Transformación del Modelo Conceptual de Datos a Esquema Físico de Datos

El proceso de transformación convencional desde un *DERExt* hasta un *EFD* implementado en un estándar SQL declarativo se puede dividir en 3 etapas [23]:

1. Transformación del *DERExt* al *ELE*: es el proceso de transformación de los conceptos semánticos modelados, basada en los elementos constitutivos del *DERExt*, entidades, relaciones y atributos con la información de todos sus componentes.
2. Transformación del *ELE* al Esquema Lógico Estándar Tardío (*ELET*): es el proceso de Diseño Lógico Tardío, donde se identifican, caracterizan y especifican las decisiones relacionadas con la forma de implementación de atributos multivaluados, atributos compuestos, relaciones 1:1, jerarquías y otras construcciones que dependen en gran medida de cuestiones físicas de implementación en una determinada clase de base de datos.
3. Transformación del *ELET* al *EFD*: es el proceso en el cual se toman el *ELET* y se lo transforma en SQL estándar.

A continuación, se enuncian las propuestas de mejoras sobre las diferentes transiciones en las transformaciones descriptas. Dichas propuestas se basan principalmente en adicionar en cada una de las transformaciones la traza de las decisiones de diseño que se van aplicando, exceptuando aquéllas que pueden ser deducidas.

2.2 Transformación del *DERExt* al ELE

En aras de facilitar la representación de una mayor variedad de situaciones del *DERExt*, en esta sección se propone un conjunto de reglas para una notación que refleje todos los conceptos semánticos modelados, basada en los elementos constitutivos del diagrama, sus extensiones y en las reglas de transformación incluidas en los algoritmos más difundidos [4], [11], [24].

El algoritmo propuesto combina tres aspectos clave: la formación sistemática de nombres de componentes del *DERExt* y conceptos, el orden en el que aparecen en cada esquema de componente y una especificación precisa de todos los conceptos relevantes del modelo.

Se adicionaron elementos que, aun no siendo estrictamente relacionales, permiten que el usuario experto pueda obtener un *ELE* estrictamente relacional, o bien que opte por uno post-relacional (con características de objetos) [1], [4], [13], [16]. En lo que sigue, se presentarán los algoritmos de conversión en un *ELE* basado en las transformaciones del *DERExt*.

Formalmente un *ELE* es un conjunto de componentes C donde C = {C1, C2, ..., Cm} representa a cada uno de los componentes típicos del modelo y además restricciones de unicidad, de no nulidad, dependencias funcionales, restricciones de integridad referencial, restricciones de cardinalidad y otras restricciones de integridad. Un esquema de componente se expresa $esq(C_i)$ y representa el conjunto de conceptos que definen el componente C_i [17], [18].

Se define como concepto a cada una de las partes constitutivas de *esq(C$_i$)*. Se propone conservar un orden prefijado para cada concepto resultante de la transformación; se utilizarán estratégicamente paréntesis '( )', y llaves '{ }' para definir conjuntos de conceptos del mismo tipo; punto y coma ';' como separador de conceptos diferentes, y comas simples ',' como separadores de conceptos de un conjunto o bien de las partes de un concepto. Cuando no haya existencia de un determinado concepto, su lugar deberá ser respetado colocando el ';' correspondiente. El único concepto que debe estar presente sin excepciones es el del identificador de cada componente, los restantes podrían estar ausentes. Cuando un concepto pueda admitir nulos, se indicará precedido por '¬'.

2.2.1. Reglas de Transformación para Entidades (*DERExt* – ELE)

En la aplicación de las reglas para obtener *esq(C)* siendo C un tipo de entidad, las características que deben preservarse son el nombre del tipo de entidad y su contexto: regular (R), subtipo (SB), supertipo (SP), supertipo de una jerarquía de más de un nivel (SPB) o débil (D). Los conceptos correspondientes son: lista de todos sus atributos con sus diferentes características y multiplicidades, clave primaria y claves alternativas. En el proceso estándar, los conceptos y el nombre se preservan aplicando los algoritmos convencionales de conversión, sin embargo, el contexto se infiere en función de particularidades de la clave primaria, incluyendo las dependencias en las que se la involucra, lo que genera algunas de las problemáticas ya mencionadas.

Se propone incorporar un prefijo al esquema lógico de las entidades, que se corresponda con la clasificación de contextos de las entidades ya mencionada (R, SB, SP, SPB ó D). Esto tiene el doble propósito de, por un lado, evitar las ambigüedades que surgen en los procesos de reingeniería (por ejemplo, la distinción entre entidades débiles y atributos multivaluados o relaciones unarias y jerarquías), y por otro, contribuir a la posible implementación de algoritmos automatizados de transformación de *ELE* a *EFD* sin pérdida de semántica.

Para la transformación de los diferentes tipos de entidades es necesario realizar al menos dos acciones: creación del tipo de componente entidad y transformación de su lista de conceptos. En un proceso automatizado los pasos a seguir deben guardar el siguiente orden de Transformación de Tipos de Entidades: Paso 1 Regulares (R), Paso 2 Débiles (D), Paso 3 Supertipos (SP), Paso 4 Subtipos (SB)

2.2.2. Reglas de Transformación para Relaciones (*DERExt* – ELE)

Las reglas destinadas a definir los *esq(C)* correspondientes a tipos de relaciones deben permitir representar directa o indirectamente los siguientes conceptos: nombre de relación; identificador principal e identificador o identificadores alternativos, si los hubiese; lista de los atributos propios del tipo de relación, con sus diferentes características y multiplicidades; el orden: unaria, binaria, ternaria o de orden superior; y sus correspondientes cardinalidades mínimas y máximas respecto de las entidades involucradas. Bajo el supuesto de una interpretación semántica de cardinalidades siguiendo el estilo Chen (look across) [5], [7], [8].

Para resolver el problema que implica no poder conservar todos los aspectos definitorios de ciertos tipos de relaciones, se propone obtener un *esq(C)* para cada uno de ellos, prefijándolo adecuadamente de acuerdo a la cardinalidad máxima de las

entidades que intervienen. De esta manera, para las relaciones unarias o binarias con cardinalidades N:1, N:N o 1:1, el prefijo será NN, N1, o 11 respectivamente, siguiendo en parte la idea presentada en [20].

Las relaciones ternarias enlazan 3 componentes entidad, entonces siguiendo el mismo espíritu que para las relaciones unarias o binarias, se sugiere nombrarlas por la tripla de cardinalidades: NNN, NN1, N11, 111. La cardinalidad máxima en el prefijo será siempre N, aunque en esta propuesta se admite que además de indefinida (N) pueda ser conocida y especificada mediante un entero mayor que 1, semántica que se pierde cuando se realizan las transformaciones siguiendo los algoritmos clásicos.

El *DERExt* original para las cardinalidades mínimas acepta valores mínimos 0 o 1, proporcionando una visión restringida de las diferentes situaciones. Se propone incorporar una versión extendida que permite generalizar todos los casos de valores mínimos y máximos de cardinalidades, considerando $0 \leq$ mín $\leq$ máx<, donde mín podrá tomar los valores 0, 1 ó *a* y máx podrá tomar los valores N ó *b* (donde N es el indefinido estándar, y *a* y *b* representan una multiplicidad constante respectivamente, con *b* >= 1). Esta información podría ser fácilmente extendida y completada indicando valores de intervalos para las cardinalidades mínimas y máximas (no se considera en este trabajo).

Se utiliza la notación de puntos (<Componente>.<Atributo>) para indicar el componente del cual proviene el atributo, conservando así el origen de un atributo respecto de la entidad de la cual proviene.

Al aplicar las reglas para transformar *DERExt* en un conjunto de *esq(C)* se debe establecer un orden. Se incluyen en primer lugar las claves primarias de las entidades participantes debidamente prefijadas con el nombre de la entidad u objeto del cual proceden, separadas por ‘,’. Del segundo al quinto lugar se colocan los identificadores alternativos, atributos descriptores simples, compuestos y multivaluados respectivamente. El sexto lugar se reserva para representar los pares de cardinalidades (min, max) correspondientes a cada entidad relacionada, colocando ambos límites entre ‘( )’ y asociados posicionalmente a cada clave proveniente de las entidades relacionadas, las cuales se enumeran en el séptimo y último lugar (reservado para representar las claves extranjeras). Las transformaciones de los componentes del 1 al 5, siguen las reglas que se detallarán en la siguiente sección.

2.2.3 Reglas de Transformación para Atributos (*DERExt* – ELE)

En la aplicación de las reglas destinadas a definir los *esq(C)* correspondientes a atributos tanto en entidades como en relaciones, los atributos simples se convierten en atributos simples. Los atributos compuestos se convierten en una lista encerrada por ‘( )’ de sus partes componentes, separadas por ‘,’ y precedida por el nombre del compuesto. Los atributos multivaluados se encierran entre ’{ }’. Adicionalmente, un atributo multivaluado compuesto se debe transformar considerando primeramente su composición y luego su carácter multivaluado. Un atributo compuesto con partes multivaluadas resultaría de transformar primero las partes del compuesto incluidos los multivaluados, y luego su transformación como compuesto. Cuando un atributo puede ser nulo dada la opcionalidad indicada en el diagrama, se lo deriva siguiendo las reglas anteriores, y se precede por ‘¬’.

2.3 Caso de Ejemplo

Para ejemplificar parte de lo descripto se utiliza el diagrama de la Fig. 1, que muestra la relación (0,1) : (0,N) *Pertenece* entre las entidades *EMPLEADO* y *DEPARTAMENTO* con dos atributos en la relación, que indican a partir de qué fecha inicia su actividad en el departamento, si lo tiene asignado, y una posible fecha de finalización si ha dejado de pertenecer al mismo. La semántica de ambos atributos está ligada a la cardinalidad de la relación (0,1), que como vimos se pierde con los algoritmos de transformación estándar.

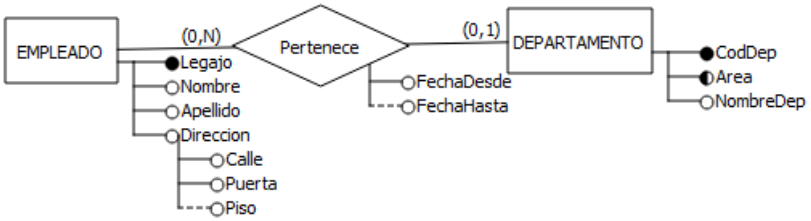

Fig. 1. Ejemplo de una situación muy simple y real, relativa a empleados y departamentos

2.3.1. Transformación *DERExt*-ELE según Proceso Estándar

Las reglas de transformación de *DERExt* a ELE, presentadas en los textos de la temática, se obtienen las relaciones donde se indican con subrayado simple las claves, subrayado doble las claves alternativas, su correspondiente Restricción de Integridad Referencial (RIR) indicada formalmente y restricciones de no nulidad (NN) (atributos que no pueden ser nulos) [4], [11], [15], [21], [22], [23], [24].

EMPLEADO (Legajo, Nombre, Apellido, Calle, Puerta, Piso, CodDep,
FechaDesde, FechaHasta)
DEPARTAMENTO(CodDep, Area, NombreDep)
RIR: EMPLEADO(CodDep) << DEPARTAMENTO(CodDep)
NN: EMPLEADO(Legajo, Nombre, Apellido, Calle, Puerta)
DEPARTAMENTO(CodDep, Area, NombreDep)

En esta transformación se perdieron dos conceptos semánticos importantes, la relación que existe entre los atributos de la relación EMPLEADO FechaDesde y FechaHasta con la cardinalidad de la relación Pertenece (0,1) y el descriptor de atributo compuesto Dirección.

2.3.2 Transformación *DERExt*-ELE segun propuesta

R_EMPLEADO[Legajo;;Nombre,Apellido, Direccion(Calle,Puerta,¬Piso);] (1)

R_DEPARTAMENTO[CodDep;AreaNombreDep;;] (2)

N1_Pertenece[EMPLEADO.Legajo;;FechaDesde,¬FechaHasta;;;(0,N),(0,1);
EMPLEADO.Legajo, DEPARTAMENTO.¬CodDep] (3)

En (1) y (2) se transformaron las entidades regulares (R_) Empleado y Departamento como dos componentes con la lista de atributos separados por ";" en el siguiente orden: 1 Concepto clave primaria (simple o compuesto). 2 Concepto clave alternativa (simple o compuesto). 3 Concepto descriptor simple. 4 Concepto descriptor compuesto. 5 Concepto descriptor multivaluado.

En (3) se transformó la relación Pertenece y los conceptos tienen el mismo orden que para las entidades (de 1 a 5). El sexto concepto, como ya se mencionó, representa las cardinalidades de las entidades involucradas, cuya ubicación se corresponde a la de las claves extranjeras de las entidades involucradas, para el ejemplo (0,N),(0,1). El séptimo y último concepto es el que corresponde a las claves extranjeras, en el ejemplo EMPLEADO.Legajo y DEPARTAMENTO.¬CodDep que es a su vez un atributo simple opcional.

Cabe destacar que en este caso, de una relación 1:N, y a diferencia del proceso estándar, se genera un componente para dicha relación.

2.3 Transformación del *ELE* al ELET

Una vez generado el *ELE* el proceso continúa con la siguiente etapa de transformaciones en las que se completan las decisiones cercanas a la implementación en SQL tanto en componentes de tipo entidad, relación o atributo.

Estas decisiones serán reflejadas en el ELET, que o bien puede continuar con una forma sintáctica de especificación similar a la del ELE, o ser implementado en un lenguaje intermedio de mayor nivel tal como XML o JSON.

Las especificaciones de esta fase de transformación se encuentran actualmente en desarrollo, por lo que a continuación se describirán los procesos y consideraciones a tener en cuenta en esta etapa.

2.3.1. Reglas de Transformación para Componentes Entidades (ELE al ELET)

En los componentes de tipo entidad existen opciones de transformación para subtipo (SB), supertipo (SP) y supertipo de una jerarquía de más de un nivel (SPB). Es posible tener en cuenta cuatro opciones de implementación distintas:

Opción A: crear una tabla para el supertipo (con clave primaria, y atributos comunes a todos los subtipos) y otra tabla para cada subtipo (con clave igual a la del supertipo, y atributos específicos del subtipo).

Opción B: crear una tabla por cada subtipo, incluyendo en cada una, los atributos del supertipo, y adicionando los atributos específicos del subtipo.

Opción C: crear una única tabla que contenga la clave primaria y atributos del supertipo, y los atributos de cada subtipo (completando con nulos, los atributos faltantes), a su vez, indicando el o los discriminante/s de subtipo al que pertenece/n el ejemplar, según corresponda.

Opción D: crear una tabla correspondiente al supertipo, y otra tabla que contenga los atributos de cada uno de los subtipos, especificando el tipo al que pertenece, según corresponda.

La elección de qué opción tomar, corre por cuenta de quien lleve adelante el diseño físico de ese modelo de datos, quien deberá considerar los pro y contras de cada opción aplicada al dominio en cuestión.

2.3.2. Reglas de Transformación para Componentes Relaciones (ELE al ELET)

Para los componentes de tipo relación es necesario adicionar las acciones referenciales y las opciones de matching para las claves extranjeras.

En el caso de las acciones referenciales permiten definir la acción a realizar ante una modificación o borrado sobre las claves en la tabla referenciada y según el SQL estándar es posible optar para cada acción por No Action, Restrict, Cascade, Set Null o Set Default. Por otro lado, se tienen los tipos de matching que se pueden definir sobre una clave extranjera (compuesta por más de un atributo) que admite nulos y que establece en qué modo se cumple la integridad referencial, las opciones posibles son Simple, Partial o Full [12].

Otro concepto que resulta importante es el de rolename. Este es el nombre que se utiliza ante la repetición de nombres de columnas a la hora de realizar la transformación. Esta repetición no es aceptada por SQL, por lo que se requiere un nombre que la reemplace, en casos como en relaciones unarias donde se incluiría dos veces la clave primaria y con entidades que poseen múltiples relaciones N:1 (recordar que la entidad de cardinalidad N tiene tantos atributos claves extranjeras como relaciones).

2.3.3. Reglas de Transformación para Componentes Atributos (ELE al ELET)

Dentro de los atributos, encontramos diferentes formas de transformarlos, según el rol que ocupan dentro del componente entidad o relación (claves principales, claves secundarias, claves extranjeras o atributos normales). Las siguientes transformaciones son válidas para cada tipo sin importar su rol, dado que el mismo es otorgado mediante sentencias especiales definidas por SQL.

Los atributos simples son agregados a la definición de la tabla a la que pertenecen, adicionando sólo el tipo de dato en el ELET. Los atributos multivaluados pueden ser transformados de acuerdo a 3 formas:

Opción A: agregar una cierta cantidad (especificada por el usuario) de atributos simples a la declaración de la tabla.

Opción B: como un arreglo, donde en la declaración simplemente se especifica como si fuese un atributo simple, pero al tipo de dato se le agrega la indicación de que se trata de un arreglo de elementos de ese tipo.

Opción C: creación de una tabla, compuesta simplemente por el atributo multivaluado (esta vez especificado como un atributo simple), y por la clave principal de la tabla que especifica la entidad que tiene ese atributo multivaluado. De esta forma, en esta nueva tabla, tanto la clave extranjera (referenciando a la clave primaria de la tabla principal) como el atributo multivaluado, forman parte de la clave primaria.

Por último, para transformación de atributos compuestos, existen dos formas diferentes de obtener la transformación a SQL. La primera forma consiste en definir cada atributo que compone al atributo compuesto como un atributo simple, en la tabla principal. No existe diferencia alguna con la derivación de un atributo simple. La segunda forma es crear un nuevo tipo de datos. Este tipo de datos va a contener todos los campos que tenga el atributo compuesto, y en la tabla principal se creará como un atributo cuyo tipo de dato es el creado previamente.

2.4 Transformación del *ELET* al EF/ SQL

Este proceso parte de la especificación del *ELET* y genera el esquema físico SQL final. Para llevar a cabo este proceso se requiere establecer un conjunto de reglas de transformación, las cuales se encuentran actualmente en definición. Para el ejemplo de la sección 2.3.1 y 2.3.2 el SQL generado se detalla a continuación.

```
CREATE TYPE Direccion_type AS(Calle varchar(40) NOT NULL, Puerta integer NOT
NULL, Piso integer);

CREATE TABLE EMPLEADO ( Legajo integer NOT NULL, Nombre varchar(40) NOT
NULL, Apellido varchar(40) NOT NULL, Direccion Direccion_type, FechaDesde date,
FechaHasta date, CodDep integer,   CONSTRAINT pk_empleado PRIMARY KEY
(Legajo),  CONSTRAINT ck_pertenece (check (CodDep IS NULL AND FechaDesde IS
NULL AND FechaHasta IS NULL) OR (CodDep IS NOT NULL AND FechaDesde IS
NOT NULL AND FechaHasta IS NOT NULL) OR (CodDep IS NOT NULL AND
FechaDesde IS NOT NULL));

CREATE TABLE DEPARTAMENTO (CodDep integer NOT NULL, Area char(5)
UNIQUE, NombreDep varchar(40) NOT NULL, CONSTRAINT pk_departamento
PRIMARY KEY (CodDep));

ALTER TABLE EMPLEADO ADD CONSTRAINT
FK_EMPLEADO_DEPARTAMENTO FOREIGN KEY (CodDep)    REFERENCES
DEPARTAMENTO (CodDep);
```

3 Conclusiones y Futuros Trabajos

En este trabajo se ha especificado un *ELE* incluyendo todos los aspectos del modelo conceptual de datos, proponiendo sólo cambios menores en la notación de las reglas conocidas para la construcción del diagrama, extendiendo el algoritmo de transformación para resolver cuestiones relativas a ambigüedades y falta de ortogonalidad.

Se han incorporado cualidades de completitud que se pretenden de la metodología, para posibilitar que el usuario experto pueda optar por una transformación tanto hacia un modelo relacional puro como a uno post-relacional. Se han descripto los lineamientos que deben seguir las reglas de especificación que establezcan la correspondencia de un *ELE* con el ELET, y de éste con el EF/SQL.

Este trabajo es un avance respecto al objetivo más global de obtener en un proceso automatizado la transformación desde un modelo conceptual hasta EF/SQL, el cual incremente la cantidad de cuestiones semánticas que contempla disminuyendo a un mínimo las decisiones de implementación que quedarán bajo responsabilidad de quien la lleve adelante en la base de datos.

Debido a esta incorporación de mayor semántica, los artefactos intermedios de las fases de este proceso constituyen también una mejora sustancial en la documentación del modelo de datos, y en la comunicación de desarrollo y post desarrollo de un proyecto de implementación.

Referencias

1. An, Y., Hu, X., & Song, I. Y.: Maintaining mappings between conceptual models and relational schemas. Journal of Database Management (JDM), 21(3), 36--68, (2010).
2. Badia, A.: Entity-Relationship modeling revisited. ACM SIGMOD Record, 33(1), 77--82, (2004).
3. Badia, A., & Lemire, D.: A call to arms: revisiting database design. ACM SIGMOD Record, 40(3), 61--69, (2011).
4. Batini, C., Ceri, S., & Navathe, S. B.: Conceptual database design: an Entity-relationship approach (Vol. 116). Redwood City, CA: Benjamin/Cummings, (1992).
5. Chen, P. P. S.: The entity-relationship model—toward a unified view of data. ACM Transactions on Database Systems (TODS), 1(1), 9--36, (1976).
6. Codd, E. F.: The relational model for database management: version 2. Addison-Wesley Longman Publishing Co., Inc., (1990).
7. Cuadra, D., Nieto, C., Martínez, P., Castro, E., & Velasco, M.: Preserving relationship cardinality constraints in relational schemata. In Database Integrity: Challenges and Solutions (pp. 66-112). IGI Global, (2002).
8. Cuadra, D., Martínez, P., Castro, E. Al-Jumaily, H.: Guidelines for representing complex cardinality constraints in binary and ternary relationships, Software & Systems Modeling Journal, Vol.12, 4, 871-889, (2013).
9. de Miguel Castaño, A., Velthuis, M. G. P., & Martínez, E. M.: Diseño de bases de datos relacionales. Editorial Alfaomega, (2000).
10. Date, C. J.: An introduction to database systems. Pearson Education India, (2006).
11. Elmasri, R., & Navathe, S.: Fundamentals of Database Systems, 6th Ed. Adison-Wesley Pub. Comp., (2010).
12. Date, C. J., & Darwen, H.: A guide to the SQL Standard: a user's guide to the standard relational language SQL. Addison-Wesley. (1989).
13. Goelman, D., & Song, I. Y.: Entity-relationship modeling re-revisited. In International Conference on Conceptual Modeling (pp. 43-54). Springer, Berlin, Heidelberg, (2004).
14. Halpin, T., & Morgan, T.: Information modeling and relational databases. Morgan Kaufmann, (2010).
15. Hansen, G. W., & Hansen, J. V.: Database management and design. Upper Saddle River, New Jersey: Prentice Hall, (1996).
16. Parent, C., Spaccapietra, S., & Zimányi, E.:. Spatio-temporal conceptual models: data structures+ space+ time. In Proceedings of the 7th ACM international symposium on Advances in geographic information systems (pp. 26-33). ACM, (1999).
17. Pieris, D.: Modifying the Entity relationship modelling notation: towards high quality relational databases from better notated ER models. Preprint arXiv:1306.5690, (2013).
18. Pieris, D.: A novel ER model to relational model transformation algorithm for semantically clear high quality database design. arXiv preprint arXiv:1306.6734, (2013).
19. Pieris, D.: Extending the ER Model to relational Model novel transformation Algorithm: transforming relationship Types among Subtypes. arXiv preprint arXiv:1307.4519, (2013).
20. Pieris, D., & Rajapakse, J.: Logical database design with ontologically clear entity relationship models. In Information and Automation for Sustainability (ICIAfS), 2012 IEEE 6th International Conference on (pp. 209-214). IEEE, (2012)
21. Ramakrishnan, R., & Gehrke, J.: Database management systems. McGraw Hill, (2000).
22. Silberschatz, A., Korth, H. F., & Sudarshan, S.: Database system concepts, 5th Ed., New York: McGraw-Hill, (2005)
23. Teorey, T.J.: Database Modeling and Design. The Entity-Relationship Approach", Morgan Kaufmann, San Mateo, (CA), (1990).
24. Teorey, T. J., Lightstone, S. S., Nadeau, T., & Jagadish, H. V.: Database modeling and design: logical design. Elsevier, (2011).