INFORMÁTICA

TRANSFORMACIÓN DE ESQUEMAS RELACIONALES ORIENTADOS A OBJETOS: UN ENFOQUE BASADO EN EL OBJETO

Resumen / *Abstract*

Las innovaciones en la industria de las bases de datos son permanentes. Entre las más recientes , se tienen las bases de datos distribuidas, el paradigma orientado a objetos (OO), y la tecnología XML*. Todas tienen un objetivo común, mejorar la calidad de los servicios. Por otro lado se observa que la inmensa mayoría de los sistemas de bases de datos (BD) actuales tienen una estructura relacional y aún siguen almacenando voluminosas informaciones de sumo valor para las organizaciones. La transformación de los esquemas de bases de datos permanecen siendo un campo de investigación de primordial importancia, ya que se observan numerosas aplicaciones sometidas hoy en día a procesos de reingeniería. Aunque la industria de software ya ofrece algunas herramientas para automatizar este proceso, aún queda mucho por hacer. Y en este contexto, la comprensión del formalismo matemático y de la orientación de los enfoques propuestos no solo podría permitir una mejor comprensión de estos, sino también apoyar a la realización de trabajos futuros en este campo. El presente artículo describe un enfoque de transformación basado en el objeto y en cómo refinarlo mediante traducción de consultas SQL en código C++.

Innovations in databases industry are in upgrading, since emerging keys technologies like distributed databases, object oriented paradigm, and the XML technology enable to perform services and products efficiency. However it looks that the today large majority of databases systems in use is relational and remains important for theirs organisations, so that companies have to face the problem of schema and data transformation when migrating theirs applications. Nowadays databases schema transformation remains an ongoing investigation field of terest due to the amount of applications which are reengineered. Although there exist some tools for process automation, understanding the mathematic formalism and related methods is important for the today and future advancements with reference to the aforementioned. It is carried out in this paper a transformation object-based method and how it can be refined by traducing SQL queries statements in C++ code.

Palabras clave / *Key words*

Bases de datos, transformaciones de esquemas, ingeniería inversa, esquema relacional, esquema orientado a objetos

Databases, schema transformation, reverse engineering, relational schema, object oriented schema

Marc Desiré Atangana, *Ingeniero Informático, Centro de Estudios de Ingeniería de Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba*
e-mail:amarc@ceis.cujae.edu.cu
amarc.de@gmail.com

Roberto Sepúlveda Lima, *Ingeniero Electricista, Doctor en Ciencias Técnicas, Profesor Titular, CEIS, Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, Cuba*
e-mail:sepul@ceis.cujae.edu.cu

Recibido:Octubre del 2006
Aprobado:Diciembre del 2006

INTRODUCCIÓN

Una vez más la industria de BD está conociendo cambios impetuosos. Tras el gran éxito de las bases de datos relacionales, son las bases de datos orientadas a objetos (BDOO) las que cobran a diario mayor importancia, dando la enorme flexibilidad que brinda este paradigma para el desarrollo de aplicaciones complejas en dominios tan variados como la representación de estructuras moleculares, el DAO**, y los sistemas de representación espacial.

** XML:eXtensible Markup Language o lenguaje de marcación extensible.*
***DAO:Diseño asistido por ordenador.*

Ya constituye una práctica desarrollar aplicaciones de BDOO. Pero la tarea no es tan fácil cuando se trata de realizar reingeniería a una aplicación ya hecha, pues surgen dificultades de traducción de datos y de esquemas de datos, donde se debe conservar la semántica de las estructuras.

Son cuantiosos los estudios de transformación de esquemas y de datos que se han logrado con el desarrollo de tecnologías de BD, siendo los más recientes y aún de actualidad los que investigan en la conversión de esquemas relacionales en diagramas OO o en formato XML. El presente trabajo apunta a mostrar cómo obtener un esquema de datos OO y refinarlo por la traducción de consultas SQL en código C++, tras una exposición de los conceptos básicos de transformación de esquemas. Se contemplan otros trabajos relacionados y el artículo concluye con algunas recomendaciones.

CONCEPTOS BÁSICOS DE TRANSFORMACIÓN DE ESQUEMAS

Numerosos procesos de ingeniería pueden ser modelados como una transformación de estructuras de datos. Más bien, la producción de un esquema puede ser considerada como una derivación de este esquema desde un esquema fuente a través de lo cual se ejecuta una cadena de operaciones elementales llamadas *transformaciones de esquemas*.[1]

Sumar un tipo de asociación, eliminar un identificador, traducir un nombre o remplazar un atributo con un tipo de entidad equivalente son, entre otros, ejemplos de operadores básicos mediante los cuales, proceden los procesos de ingeniería tales como la construcción de un esquema conceptual, la normalización de esquema, la traducción de esquema, la conversión de datos y la integración de esquemas.[2]

Jean Henrard generaliza el concepto de transformación, considerando cualquier proceso de ingeniería, ya sea una actividad de ingeniería directa o inversa, como un conjunto de transformaciones.[1]

La transformación de esquema es un concepto de gran actualidad en el ámbito del diseño de BD cuyo objetivo es la producción de un esquema de datos desde otro esquema fuente. Por tal razón, se exponen a continuación sus principios básicos.

Principios básicos

Una transformación es generalmente considerada como un operador mediante el cual una estructura de datos fuente *C* es sustituida por una estructura *C*'. Existen varios tipos de transformación de esquema:

- *La transformación de semántica incremental* que incluye técnicas a través de las cuales se insertan nuevas especificaciones en el esquema.
- *La transformación de semántica simplificativa* que incluye técnicas a través de las cuales se elimina ciertas especificaciones en el esquema.
- *La transformación de semántica conservadora* que preserva la especificación semántica de los objetos de un esquema a otro.

Siendo funciones, las transformaciones pueden estar compuestas con el fin de obtener operadores más potentes. Complejas combinaciones de transformación pueden estar construidas mediante planes de transformación que son procedimientos o *scripts* de alto nivel que describen cómo aplicar un conjunto de transformación con el fin de llevar a cabo una determinada tarea o alcanzar una meta.

Estos *scripts* están compuestos por operadores que se ejecutarán siempre y cuando ciertos predicados (sobre propiedades del esquema) estarán cumplidos.

Es importante tener en cuenta que un plan de transformación puede ser considerado como una estrategia para una transformación de alto nivel que se aplica en todo el esquema. Una de las transformaciones que suelen aplicar los diseñadores de BD es el mapeo de esquema entidad-relación (ER) al esquema relacional.

El mapeo del esquema E-R hacia el esquema relacional consiste en la transformación del MCD* que resulta del análisis, para obtener la forma lógica en la cual se almacenarán los datos (Modelo Lógico de Datos-MLD**).

En la actualidad son los mapeos del modelo relacional al orientado a objetos, del orientado a objetos al relacional, y del relacional al formato de datos XML, respectivamente, los que cobran más atención. Otro campo al parecer incipiente pero que augura un futuro prometedor, es el mapeo de esquemas orientados a objetos al formato XML.

Un formalismo matemático para la transformación de esquemas

Formalmente, una transformación $\sum$ consiste en un mapeo de elementos *T* y *t* tal que,

- *T* es un mapeo de estructuras que sustituye una construcción origen *C* en el esquema *S* por *C*'; *C*' es el objetivo de *C* a través de *T*, y se nota *C*' = *T*(*C*). Es preciso que *C* y *C*' sean clases de construcción que puedan ser definidas mediante estructuras de predicados. *T* es, por lo tanto, definido por una precondición mínima *P* que cualquiera construcción *C* deberá satisfacer para ser transformada por *T*, y una poscondición máxima *Q* que *T*(C) satisface. *T* puede ser por consiguiente descrito como un par $T = \langle P, Q \rangle$, donde *P* y *Q* son predicados de comparación de modelos que identifican los componentes y las propiedades de *C* y *T*(*C*), y más específicamente, los componentes de *C* que son preservados en *T*(*C*), los componentes de *C* que son rechazados de *T*(*C*), los componentes de *T*(*C*) que no existen en *C*. *T* define la sintaxis de transformación.
- *t* es una instancia de mapeo que expresa cómo producir la instancia *T*(*C*) correspondiente a algunas instancias de *C*. Si *c* es una instancia de *C*, entonces *c*' es la instancia correspondiente de *T*(*C*) que se expresa por *c*' = *t*(*c*). De una cierta manera, *t* es la semántica de la transformación.

La transformación $\sum$ puede, por consiguiente, ser descrita para $\langle T, t \rangle$ o $\langle P, Q, t \rangle$.

*MCD:Modelo conceptual de datos.
**MLD: Modelo lógico de datos.

Carácter reversible de la transformación de esquemas

La noción de reversibilidad es una característica importante de una transformación. Si una transformación es reversible, entonces los esquemas fuente y el objetivo poseen el mismo poder descriptivo, y describen el mismo universo del discurso, aunque con una representación o sintaxis diferente.

Uno de los primeros autores por presentar la transformación de esquema como un concepto básico en el diseño de las bases de datos fue Batini, en 1992.[2] Este define la reversibilidad en términos de equivalencia de información entre los esquemas fuente y destino. Según este autor: "Los esquemas *S*1 *y S*2 *t*ienen el mismo contenido informativo (o son equivalentes) si para cada consulta *Q* que puede ser expresada sobre *S1,* existe una consulta *Q'* que puede ser expresada sobre *S2* y dar la misma respuesta, y viceversa*".*

Es evidente comprobar que las transformaciones *SR* (esta noción es explicada más adelante) son mapeadas entre esquemas que tienen el mismo contenido informativo, respecto a la definición anterior.

Las transformaciones reversibles son también llamadas conservadoras de semánticas.

Una transformación $\sum 1 = <P1, Q1, t1> = <T1, t1>$ es reversible, si y solo si existe una transformación $\sum 2 = <P2, Q2, t2> = <T2, t2>$, tal que, para cualquiera construcción C, y cualquiera instancia *c* de *C*, se tiene: $P1(C) \Rightarrow [T2(T1(C)) = C]$ y $[t2(t1(c)) = c]$.

$\sum 2$ es el inverso de $\sum 1$, pero lo recíproco no es cierto. Por ejemplo, una instancia arbitraria *c*' de *T*(C) no satisficiera la propiedad *c*' = *t*1(*t2*(*c*')).

Si es reversible, entonces (al igual que) es *simétricamente reversible*. En este caso , y ambas transformaciones pueden ser definidas mediante la notación única . En este caso a $\sum$ se le llama transformación *SR*. Esta es la forma más deseable de transformación, y numerosas técnicas en la literatura se fundamentan de ella.

Sin embargo, en el ámbito de diseño de BD, y en particular en el nivel de implementación, se emplean a menudo transformaciones que no son completamente reversibles, debido a la dificultad de adaptar la transformación SR. Existen dos enfoques para explorar una transformación SR: la generalización y la especialización.

La generalización consiste en la selección de una construcción objetivo que describe instancias más generales que las que brinda el esquema fuente. Un caso típico es la sustitución de asociaciones 1:1 para 1: *N* común en los esquemas CODASYL* y relacionales.

La especialización consiste en seleccionar una construcción objetivo que ofrece más propiedad que las requeridas por la construcción fuente. Por ejemplo, un atributo multievaluado puede ser generalmente representado por un atributo-lista en las estructuras de datos COBOL y CODASYL.

*CODASYL: Conference on Data System Language, es un modelo de datos basado en la lógica de registro. También denominado modelo de red, que fue el primero en aparecer comercialmente, a principios de 1970.

TRANSFORMACIÓN DE ESQUEMA RELACIONAL A ESQUEMA ORIENTADO A OBJETOS

La ingeniería inversa de una base de datos relacional es el proceso que trata con las actividades de extracción de estructuras de datos correspondientes a un determinado modelo conceptual, tal como el modelo ER, el modelo ER extendido (ERE), el modelo orientado a objeto (OO) y otros.

El cómo manejar la información colectada, cuál debe ser su naturaleza, y cuánto debe ser, son uno de los artefactos ineludibles usuales a considerar al elegir un enfoque de transformación.

La información puede ser derivada automáticamente u obtenida a través de un usuario. Las dependencias de inclusión, las dependencias funcionales, y los atributos de datos son algunos de ejemplos de dicha información.

Durante mucho tiempo, las técnicas de transformación de esquemas giraron en torno a un concepto: la relación. Pero con el impulso y la imposición del paradigma objeto en la esfera de desarrollo de aplicaciones, en los últimos años, nació un nuevo enfoque de transformación que se centra en el objeto, el objeto como elemento canónico básico de modelado del universo del discurso.

En el enfoque centrado en la relación, se recupera, en primer lugar, el esquema de relación y toda la información disponible. Se examina atentamente la estructura relacional de cada relación, para identificar una estructura objeto correspondiente. Concerniente a lo referido, se considera que el grado de complejidad será mayor mientras se tenga más información deteriorada o un esquema de relación estropeado.

Por otro lado, en un enfoque centrado en el objeto, la atención se dirige directamente al objetivo, es decir, una construcción orientada a objeto que sea integralmente factible. Una vez determinada cuál es la construcción a extraer, se procede a casos de estudios.

La ventaja del enfoque centrado en el objeto es que solo la información requerida se tiene en cuenta para identificar una construcción objeto particular, es decir, la clase. El inconveniente es que puede haber una pérdida semántica de grado variable según los enlaces que puede tener la construcción con estructuras ajenas.

Al parecer, el enfoque centrado en el objeto está cobrando mucha más atención hoy en día, debido a ciertos factores, por ejemplo:

- Su flexibilidad en la formalización y la implementación,
- Una facilidad de entendimiento que justifica la propensión de su enseñanza en las instituciones de formación,
- La novedad y el gran uso del paradigma OO en la industria de la ingeniería de software.

Sin embargo, cuándo una actividad de transformación de esquema y/o de datos se enmarca en un proceso de envergadura tal como el de la ingeniería inversa, siempre es necesario escalar este proceso por una actividad previa de análisis y recubrimiento de estructuras (explicitas e implícitas) de datos.

Se expone seguidamente una técnica de transformación de esquema centrada en el objeto, basada en la propuesta de Ramanathan y Julia,[3,4] que ofrece una gran flexibilidad y es menos tediosa de procesar.

Enfoque centrado en el objeto

Ramanathan propone una técnica centrada en el objeto bastante flexible por ser implementable,[3] pero tiene el defecto de fundamentarse en ciertas suposiciones.

La primera tarea consiste en coleccionar información, es decir, el agrupamiento de toda construcción interesada en el proceso de extracción de estructuras. Estas construcciones que constituyen elementos o características del esquema relacional son:

- Un conjunto de relaciones.
- Las llaves primarias (PK) y llaves foráneas (FK).
- La nulidad de las FK.
- El esquema debe estar normalizado al menos en la tercera forma normal (3FN).

Para ilustrar las actividades realizadas en cada etapa, se considerara el siguiente esquema relacional tomado del ejemplo: [4]

Employee (SSN, Name, Age, Sex, Dept#)
- *PK: SSN FK: Dept#(Department) NN*

Department (Dept#, Name, Location)
- *PK: Dept# FK: None*

Project (Proj#, Name, LeadSSN, Budget, TargetDt)
- *PK: Proj# FK: LeadSSN(Employee) NN*

WorksOn (SSN, Proj#, StartDt)
- *PK: SSN, Proj# FK: SSN(Employee), Proj#(Project)*

HardwareProj (Proj#, Supplier#)
- *PK: Proj# FK: Proj#(Project), Supplier #(Supplier)*

SoftwareProj (Proj#, Language, LOC)
- *PK: Proj# FK: Proj#(Project)*

Supplier (Supplier#, Name, Phone, Address)
- *PK: Supplier# FK: None*

Dependent (SSN, DepName, DepAge)
- *PK: SSN, DepName FK: SSN(Employee)*

En la salida, se obtiene un esquema conceptual orientado a objeto. El algoritmo de mapeo lleva a cabo las siguientes tareas:

1. Identificar los objetos clases.

2. Identificar asociaciones. Existen tres tipos de asociaciones que pueden ser representadas en un modelo objeto: las asociaciones (simples), las generalizaciones/especializaciones, y las agregaciones. La búsqueda de cada una de esas construcciones suele ser una etapa distinta en el proceso de mapeo.

• Identificación de asociaciones. Dado que el modelo objeto permite el modelado de asociaciones en clases, basta establecer un vínculo entre dos objetos clases o identificar asociaciones que corresponden a determinadas clases de una asociación.

• Identificación de herencia. La estructura de herencia captura asociaciones de generalización/especialización entre objetos clases que ya han sido identificados.

• Identificación de agregación. Las asociaciones de agregación modelan la composición de un objeto con otros objetos. Tales construcciones complejas deben ser pertinazmente identificadas.

3. Establecer las cardinalidades.

El proceso de transformación

• Identificación de clases. Ver figura1.

Reglas:

• Cada relación, cuya llave primaria (PK) está compuesta por un solo atributo es una clase objeto.

• Cada relación, cuya PK está compuesta por más de un atributo, y por lo menos uno de estos atributos no es una llave foránea es una clase objeto.

• Cada relación, cuya PK tiene más de un atributo y todas son FK es una clase asociación.

Por ejemplo:

• Employee, Department, Project, HardwareProj, SoftwareProj, Supplier.

• Dependent.

• WorksOn.

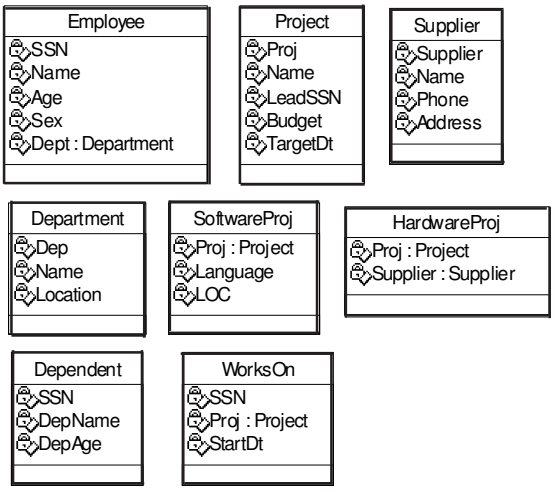

Fig. 1 *Identificación de clases.*

Identificación de asociaciones. Ver figura 2.

Reglas:

• Las llaves foráneas (FK) de una clase asociación representan el vínculo entre varias clases por las cuales a cada una de ellas corresponde una de esas llaves.

• Una clase objeto representa la asociación con las clases correspondientes a sus FK, a no ser que una de estas llaves sea parte de la PK de la clase objeto.

Por ejemplo:

• *WorksOn-Employee, WorksOn-Project.*

• *Employee-Department, Project-Employee, HardwareProj-Supplier.*

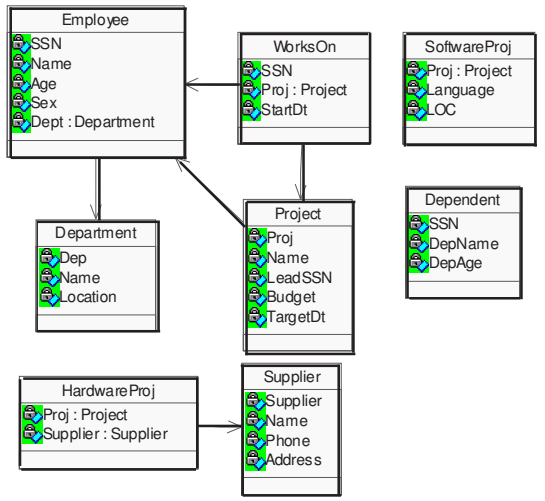

Fig. 2 *Identificación de asociaciones.*

• Identificación de la herencia. Ver figura 3.

Reglas:

La clase *C*1 hereda de la clase *C*2 si ambas tienen la misma PK y, la PK de *C*1 es llave foránea para *C*2.

Por ejemplo:

• HardwareProj-Project, SoftwareProj-Project.

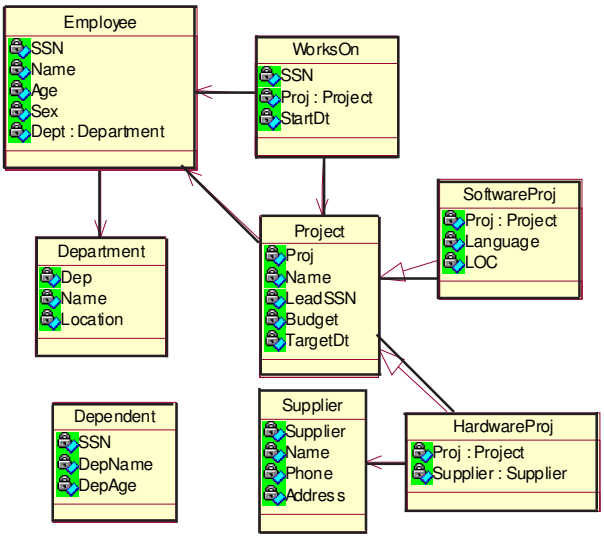

Fig.3 *Identificación de la herencia.*

• Identificación de agregación. Ver figura 4.

Reglas:

• Para una clase cuya PK posee más de un atributo, tal que al menos uno de ellos es llave foránea y, al menos uno de ellos no es llave foránea, esta clase está agregada por la clase correspondiente al más grande subconjunto de llaves foráneas, en su PK.

Por ejemplo:

• Employee-Dependent.

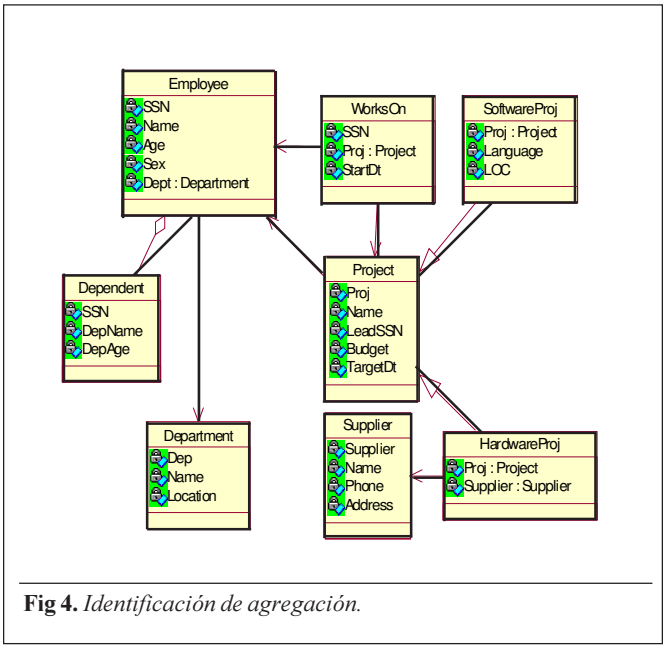

Fig 4. *Identificación de agregación.*

• Identificación de cardinalidad. Ver figura 5.

Reglas:

• Dada una asociación entre *C*1 y *C*2, si la PK de *C*1 es FK de *C*2, entonces las instancias de *C*2 están, cuanto más, asociadas con una instancia de *C*1 si la llave ajena es anulable y, con exactamente una instancia de *C*1 si la llave ajena no es anulable.

• Si por otra parte la PK de *C*2 no es FK de *C*1, entonces más de una instancia de *C*2 podría ser asociada con cada instancia de *C*1.

Por ejemplo:

Las cardinalidades aparecen en el diagrama de clase completo.

A continuación, se presenta el diagrama de clases resultante.

Contexto de aplicabilidad de la técnica de Ramanathan

La técnica centrada en el objeto cuyo proceso de transformación ha sido previamente expuesto se basa en ciertas suposiciones imprescindibles para ser llevada a cabo. Precisamente, se considera que el esquema relacional está al menos en 3FN.*

*Un esquema está en 3FN si en cada relación, no hay dependencias transitivas y cada atributo no clave es totalmente dependiente funcionalmente de la clave primaria PK de la relación.

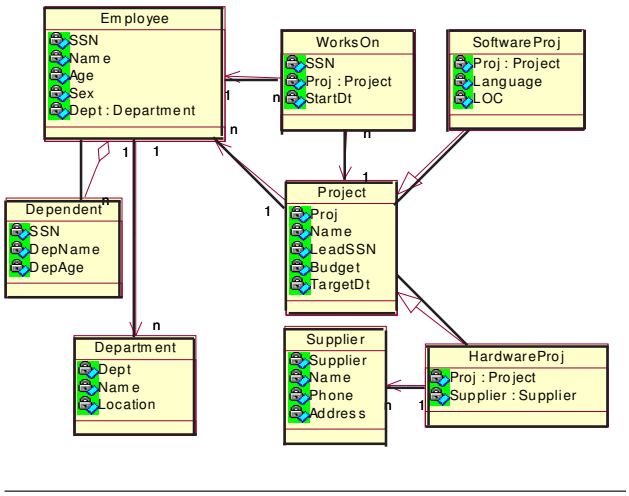

Fig. *5 Diagrama de clases resultante de la transformación.*

Aunque durante el ciclo de desarrollo de una aplicación centrada en los datos, se prevé en general diseñar una BD normalizada, es decir, que esté al menos en 3FN, ocurre a menudo factores que pueden justificar la degradación posterior de esta BD, por instancia su denormalización y optimización mediante creación de índices, dominios y procedimientos almacenados.

En definitiva, en las aplicaciones reales (no académicas) de bases de datos, ineluctablemente se producen cambios en las necesidades de negocio, temprano o tarde, que afectan de una forma u otra los datos y/o los esquemas de datos.

Las aplicaciones mencionadas son, en general, grandes contenedores de información de naturaleza variable, y son las que más se mueven para sincronizarse con la innovación tecnológica.

Por consiguiente, sería engorro, costoso y finalmente ineficiente procesar una renormalización de esas aplicaciones, antes de transformar sus datos y esquemas de datos, para finalmente integrarlos en nuevas arquitecturas.

Las soluciones actuales, para tal labor, evitan lo más posible hacer suposiciones sobre el esquema y los datos, prefiriendo plasmar sus esfuerzos en los procesos de refinamientos antes y tras la transformación.

Esto permite hacer una consideración sobre la solución de Ramanathan y Julia. Se observa, en efecto, que los trabajos se han realizados esencialmente en el aspecto estático del objeto (los miembros de datos o atributos), dejando el objeto sin revestirlo de su carácter dinámico (las funciones miembros o métodos) de suma importancia para su manipulación.

Los atributos definen la identidad del objeto, los métodos justifican su comportamiento. Esos son artefactos que hacen la fuerza y el dinamismo del paradigma OO, junto a otros tal como el polimorfismo y la herencia, por citar ejemplos.

Contribución

De lo previamente expuesto resulta que la solución de Ramanathan y Julia, de ser fácil asimilar, constituye solo la fase primaria de transformación, aun quedando por refinarse, toda dificultad radica en el cómo proceder para efectuar ese refinamiento.

Convencionalmente, la manipulación de un esquema de BD relacional se posibilita mediante la implementación de consultas SQL y de procedimientos almacenados. Es por esta razón que las funciones miembros de las clases deben ser derivadas a partir de ello.

En este artículo, se considera como lenguaje objetivo el C++. El formato básico de una consulta suele ser de la forma `SELECT...FROM...WHERE`, donde la cláusula SELECT especifica los nombres de los atributos de relaciones que entregan el resultado de la ejecución de la consulta. La cláusula FROM especifica las variables de tuplas de relación, mientras la cláusula WHERE especifica el predicado que debe ser cumplido por las variables de tuplas corrientes, para que los valores de atributos cuyo nombre ha sido referenciado se obtengan en el resultado final.

Las consultas SQL de los ejemplos siguientes son definidas sobre el esquema relacional del ejemplo de Ramanathan y Julia.

Ejemplo 1: Obtener la lista de todos los empleados.

```
SELECT Employee.Name FROM Employee
```

Traduciendo en C++, se obtiene el código de abajo, donde Employee representa un puntero sobre la clase Employee.

```
get_EmployeeName (Employee)
{
  While (! Employee.Name)
   {Writeln (Employee.Name ;)}
}
```

Ejemplo 2: Obtener los nombres de todos los empleados, cada uno con los nombres de los respectivos proyectos sobre los cuales se desempeña.

```
SELECT DISTINCT Employee.Name,
Project.Name
From Employee, Project, WorksOn
Where Employee.SSN = WorksOn.SSN
AND  Project.proj = WorksOn.proj
```

El código C++ equivalente sería:

```
get_EmpNameProjectName (Employee, Project,
WorksOn)
{
  While (! Employee.Name)
   {
     Writeln (Employee.Name);
     While ((Employee.SSN = WorksOn.SSN)
&& (Project.Proj = WorksOn.Proj))
      {
        Writeln (Project.Name);
      }
   }
}
```

TRABAJOS RELACIONADOS

Existen en la literatura varias perspectivas de mapeo de esquemas relacionales, cada una con sus características propias. Algunos autores proponen técnicas algebraicas[5] o de transformación de grafos,[6] otros proceden meramente sobre la interpretación conceptual de ambos modelos.[7-10] Pocas propuestas exploran la traducción de las consultas relacionales. En este aspecto[10] se propone un enfoque basado en tres fases: construir un grafo relacional, transformar el grafo relacional a un grafo objeto, traducir el grafo objeto en consultas OQL*. por su parte, se desarrolló un modelo algebraico para traducir las consultas SQL en consultas OQL.

Fong, por su parte, propone un algoritmo de transformación de esquemas que procede en tres etapas: traducir el esquema relacional a uno orientado a objetos; descargar las tuplas de relaciones en los archivos secuenciales; recargar esos archivos en la BDOO. Es un algoritmo centrado en el objeto muy semejante a la técnica de Ramanathan explorada en este trabajo ponencia, limitándose a procesar aspectos estáticos del objeto, y dejando inexplorado el aspecto dinámico. Ciertos proyectos de investigación han propuesto la traducción del esquema relacional a un modelo intermediario (Ej.: modelo ERE), antes de procesar la conversión hacia el modelo conceptual OO,[11,12] pero el inconveniente mayor de este enfoque está en la duplicación del esfuerzo.

Recientes trabajos han sido realizados basándose en un enfoque transformacional espacial de datos para llevar un esquema relacional a su equivalente orientado a objeto.[13]

CONCLUSIONES

Las investigaciones y los trabajos relacionados con la transformación de esquemas ya llevan más de dos décadas. Han ido evolucionando junto con la tecnología, y seguirán siendo un campo de interés, ya que no parece factible pensar en el desarrollo de un estándar pues los cambios de paradigmas, en este ámbito, son recurrentes y a veces radicales. En este artículo, se ha presentado un procedimiento de transformación de un esquema relacional para obtener un diagrama de clase estático.

De igual manera, se ha mostrado la posibilidad de que este pueda ser refinado por una traducción en código C++ de las consultas SQL en funciones miembros (métodos) a incorporar en el diagrama.

Se recomienda profundizar los estudios de transformación y refinamiento de esquema, específicamente el desarrollo de un modelo de transformación de esquemas relacionales que posee un traductor de consultas SQL en lenguajes orientados a objetos.

*OQL:Object Query Language o lenguaje de consulta a objeto.

REFERENCIAS

1. **HENRARD, JEAN:** "Program Understanding in Database Reverse Engineering", Thesis Submitted for the Degree of Doctor of Science (Computer Science Option). Institut d'Informatique de Rue Grandgagnage, 21. B-5000 Namur Belguim, August, 2003.
2. **HAINAUT, JEAN LUC:** *Schema Transformation Techniques for Database Reverse Engineering*, Institut d'Informatique de Rue Grandgagnage, 21. B-5000 Namur Belguim, July, 1993.
3. **KONTOGIANNIS, KOSTAS AND RAIHAN AL-EKRAM:** *Data Reverse Engineering: Transformation of Relational Schemas to Object-Oriented Schemas*, Course Presentation for ECE 750: Software Re-engineering, December 20, 2001.
4. **RAMANATHAN SHEKAR AND JULIA HODGES:** *Reverse Engineering Relational Schemas to Object-Oriented Schemas*, Technical Report No. MSU-960701, July 1, 1996.
5. **ANDREAS, BEHM:** *Migrating Relational Databases to Object Technology*, Department of Computer Science, University of Zurich, May 16, 2001.
6. **JAHNKE, H. JENS:** *Managing Inconsistency in Evolutionary Database Reengineering Processes*, Department of Computer Science, University of Victoria. AG Softwaretecchnik, University of Paderborn,2002.
7. **HAINAUT, J. L.:** *Database Reverse Engineering*, 2002.
8. **FONG, JOSEPH:** *Converting Relational to Object-Oriented Databases*, Computer Science Department, City University of Hong Kong, Kowloon, Hong Kong, 1996.
9. **SOON, LAY KI; IBRAHIM HAMIDAH AND ALI MAMAT:** *Constructing Object-Oriented Classes From Relations,* Faculty of Information Technology, Multimedia University, Selangor, Malaysia, 2005.
10. **STANISIC, PREDRAG:** *Database Transformation from Relational to Object-Oriented Database and Corresponding Query Translation*. Proceedings of the Workshop on Computer Science and Information Technologies CSIT'99, Faculty of Computing Mathematics and Cybernetics. Moscow State University. Moscow, Russia, 1999.
11. **TAPIA, CARLOS DE; MIGUEL DEL CORRAL Y NÉSTOR SANCHO BEJARANO:** *Transformación de esquemas relacionales a bases de datos orientadas a objetos,* Departamento de Informática y Automática, Universidad de Salamanca, Mayo, 2002.
12. **McBRIEN, P. AND A. POULOVASSILIS:** *Automatic Migration and Wrapping of Database Applications – A Schema Transformation Approach*, Department of Computer Science, Technical Report, King's College London, 1998.
13. **SIEW, T.K. AND Y.C. WANG:** *An Object-Oriented Approach for Transformation of Spatial Data from Relational Database to Object-Oriented Database*, Proceeding Of The International Conference on Asian Digital Libraries (ICADL), Kuala Lumpur, Malaysia, 2003.