Representación Semántica de un Esquema Relacional Obtenida Mediante Reingeniería de Bases de Datos

Gustavo Larriera <glarrier@ei.edu.uy>

Laboratorio de Sistemas de Información (siLAB)
Universitario Autónomo del Sur - Montevideo, Uruguay
http://www.silab.ei.edu.uy/

Octubre 28, 1998

Resumen. El proceso de reingeniería de una base de datos consiste en revertir las dos últimas fases comúnmente aplicadas en el proceso de "ingeniería hacia adelante". Específicamente, deben revertirse secuencialmente la fase lógica, donde a partir de un esquema conceptual se elabora un esquema lógico, y la fase física, donde el esquema lógico es optimizado para un DBMS en particular, generandose el esquema físico de la base de datos. Se denomina a la primera fase de reversión, fase de extracción; a la segunda fase de reversión se la denomina fase de conceptualización. En forma genérica entonces podemos ver a la reingeniería de una base de datos como un proceso que genera un modelo conceptual a partir del esquema físico. En este trabajo se describen algoritmos a ser utilizados en dichas fases de la reingeniería; tales algoritmos utilizan como entrada información de una base de conocimiento que representa la información del esquema físico. La salida final consiste en una base de datos semántica, donde se almacenan los objetos conceptualizados y los vínculos existentes entre ellos. La base de datos semántica obtenida, representa a un grafo semántico a partir del cual se puede derivar, entre otros modelos, un modelo de entidades y relacionamientos.

Palabras clave: *Database Reverse Engineering; Relational Databases; Semantic Models; Semantic Discovery; Relational Model; Conceptual Design; Logic Programming; PROLOG.*

---

1 Introducción

La Reingeniería de Bases de Datos (DBRE) consiste en un conjunto de técnicas y herramientas que permiten construír una descripción conceptual (e.g. un modelo de entidades y relacionamientos) a partir de una base de datos en producción. El uso de la DBRE permite, entre otras cosas, reconstruír y/o actualizar documentación perdida, incompleta o inexistente de bases de datos, facilitar el proceso de migración de datos y colaborar en la exploración y extracción de datos en bases poco documentadas [Lar98].

Durante el proceso de reingeniería de una base de datos -denominada *base de datos fuente*- se distinguen dos fases [HCT*93]: (i) La *fase de extracción*, durante la cual se recuperan las estructuras de datos implementadas en el esquema físico (e.g. tablas, atributos, claves primarias, claves foráneas), y (ii) La *fase de conceptualización*, durante la cual se explicitan las estructuras conceptuales que derivaron en las estructuras de datos implementadas. La fase de conceptualización produce como salida un esquema conceptual utilizando algún modelo semántico (e.g. un modelo de entidades y relacionamientos). En este artículo se detalla el proceso de conceptualización utilizado por una herramienta de DBRE propuesta en [CLR97]. Se asume que durante la fase previa de extracción se detectaron los objetos de interés, i.e. tablas, claves (primarias y foráneas) y dependencias (funcionales y de inclusión). Se asume

que tales objetos han sido previamente almacenados en una estructura de datos denominada *base de conocimiento* [Lar98]. El proceso de conceptualización utiliza a la base de conocimiento como entrada y produce como salida una representación de un esquema conceptual semántico, que se almacena en una base de datos denominada *base de datos semántica.* Las principales contribuciones de este trabajo son: (i) La descripción de la base de datos semántica; (ii) La propuesta de algoritmos para realizar la conceptualización. La base de datos semántica y los algoritmos de conceptualización se han implementado en PROLOG [CM94, DEC96]. No es el objetivo de este trabajo detallar cómo generar una representación gráfica (e.g. EER) del esquema conceptual producido.

El resto del artículo se estructura de la siguiente forma. La Sección 2 describe a las fases metodológicas que se encuentran en el proceso de reingeniería de bases de datos. En la Sección 3 se formaliza la especificación semántica a ser utilizada y se describe el diseño lógico de la base de datos semántica. En la Sección 4 se muestran los algoritmos utilizados para cargar la base de datos semántica a partir de la base de conocimiento. Finalmente, la Sección 5 presenta algunas conclusiones y trabajo futuro.

2 Fases de Reingeniería

Para comprender los procesos de reingeniería de bases de datos, resulta de interés conocer los diseños de proceso "hacia adelante" que se llevan a cabo cuando se diseña una base de datos. En forma simplificada, se puede ver al proceso de diseño como formado por dos fases que se realizan en secuencia [HCT*93]. La fase lógica utiliza como entrada a un esquema conceptual (e.g. un modelo de entidades y relacionamientos) y produce como salida un esquema lógico (e.g. un conjunto de relaciones y restricciones de integridad). La fase física acepta como entrada al esquema lógico y produce un esquema físico optimizado para un DBMS específico. Durante el proceso de reingeniería de una base de datos se distinguen a su vez dos fases, denominadas *fase de extracción* y *fase de conceptualización*, que revierten respectivamente a la fase física y a la fase lógica [HCT*93]:

- **Fase de extracción.** En la fase de extracción, los procesos acceden a la base de datos fuente para recuperar información de las estructuras de datos implementadas en el esquema físico. Los principales objetos de interés son, por ejemplo las tablas, columnas, claves primarias y claves foráneas. Cuando la base de datos fuente está implementada en un DBMS relacional, la información puede obtenerse del diccionario de datos o catálogo. Toda la información extraída se almacena en *aserciones de trabajo* de una base de conocimiento KB [Lar98]. La KB almacena al esquema lógico extraído del esquema físico de la base de datos fuente, y es utilizada como entrada para los procesos de la fase de conceptualización.

- **Fase de conceptualización.** Sobre el esquema lógico extraído en la fase de extracción se aplican diferentes procesos que permiten generar un esquema conceptual. Estos procesos ocurren en la *fase de conceptualización*, durante la cual se explicitan las estructuras conceptuales que derivaron en las estructuras de datos implementadas. Esta fase produce como salida un esquema conceptual utilizando algún modelo semántico. Este modelo semántico se almacena en una estructura a la que genéricamente se denomina *base de datos semántica.* En la base de datos semántica se almacenarán los objetos conceptuales y los vínculos existentes entre ellos. La base de datos semántica, los algoritmos de extracción y de conceptualización se describen en las siguientes secciones.

3 La Base de Datos Semántica

Cuando una base de datos es sometida a un proceso de reingeniería, la información en bruto que debe considerarse consiste en los objetos existentes y los vínculos que puedan haber entre ellos. Por ejemplo, la relación *ALUMNO(Id, Nombre, IngresoFecha)* en una base de datos de una universidad consta en principio de 4 objetos (la relación y sus tres atributos) y los vínculos entre los objetos: por ejemplo, existe una vinculación "es atributo de" entre cada atributo y la relación. En forma general, denominaremos *especificación semántica* a la colección de información de los objetos y los vínculos entre ellos. La especificación semántica se almacena en una *base de datos semántica (SDB)*, a los efectos de ser procesada por los algoritmos de conceptualización encargados de generar un modelo conceptual.

3.1 Objetos y Links

La información representada en la especificación semántica consiste en *objetos* y *links*. Los objetos, denotados como conjunto O, pertenecen a una de las siguientes clases: (i) Objetos atómicos o *átomos*, son aquellos objetos simples que no incluyen a otros objetos (e.g. una columna de una tabla); (ii) Objetos compuestos o *moléculares*, son aquellos objetos que están formados por otros objetos. Notaremos a las clases de átomos y moléculas, respectivamente como A y M. Se cumple que $A \cup M \equiv O$ y $A \cap M = \varnothing$.

Los *links* representan asociaciones definidas entre los objetos. Los links entre objetos pueden representar conceptualmente asociaciones diferenciadas según sean átomos o moléculas los objetos vinculados. En forma más general, podemos definir al conjunto de links L y luego a las clases de links, según cuáles sean los tipos de objetos vinculados. Por ejemplo, en el EER son moléculas los entity types y los relationship types; los links son los role-links y los attribute-links, que respectivamente vinculan a un entity tipe con un relationship type, y un atributo con un entity type (o con un relationship type).

Def. [Links] Se define al *conjunto de links* como un conjunto de pares ordenados $L = \{ (o_1, o_2) \mid o_1, o_2 \in O \}$. Se dice que $(o_1, o_2) \in L$ es un *autolink* cuando $o_1 = o_2$. ♦

Def. [Clases de links] Sea $(o_1, o_2) \in L$. Decimos que (o_1, o_2) es un *link intermolecular* cuando $o_1, o_2 \in M$. Es un *link interatómico* cuando $o_1, o_2 \in A$. Es un *link mixto* cuando o bien $o_1 \in A$ y $o_2 \in M$, o bien $o_1 \in M$ y $o_2 \in A$. Notaremos L_{IM}, L_{IA} y L_{MX} respectivamente a los conjuntos de links intermolecualres, interatómicos y mixtos. L_{IM}, L_{IA} y L_{MX} forman una partición en L. ♦

Como instancias de L_{IM}, tenemos a la *generalización* y a la *agregación*, con la semántica habitual que se les da en los modelos de datos [TL82]. Sin embargo, consideramos a la generalización en un sentido más amplio ya que la entendemos como un link entre dos moléculas y no hacemos distinción entre moléculas entity type y relationship type, como ocurre en el modelo clásico de entidades y relacionamientos.

Como instancias de L_{IA}, tenemos a la relación *subparte*, que permite especificar una jerarquía

de atributos compuestos. El link *dimensión* establece una jerarquía entre atributos que puede utilizarse para representar aspectos multidimensionales [GMR98]. Por ejemplo, podemos considerar una dimensión *"ubicación*" formada por *(país, ciudad, código postal)* para los alumnos de la universidad. La representación de links interatómicos no será tratada en el presente artículo.

Entre los links de L_{MX} vamos a considerar al que usualmente se denomina *attribute link*, que vincula a una molécula con cada atributo de ella.

3.2 Base de Datos Semántica

A los efectos de representar la SDB vamos a definir un grafo, cuyos vértices son los objetos (átomos y moléculas) y los diversos links aparecen como aristas con dirección.

Def. [Base de datos semántica, SDB] Se define la base de datos semántica SDB = (O, L) como un grafo dirigido donde los vértices son v_1, ..., v_n, $v_i \in$ O y las aristas son $(o_i, o_j) \in$ L. ♦

Podemos implementar la SDB utilizando PROLOG de la siguiente manera: (i) Las moléculas y átomos se expresan mediante los hechos `objmolec/2` y `objatom/2`; (ii) Las tres clases de links se expresan respectivamente mediante los hechos `lim/4`, `lia/4` y `lmx/4`; (iii) La colección completa de objetos se deriva mediante la regla `obj/1`; (iv) La colección completa de links se deriva mediante la regla `lnk/1`.

```
objmolec(OId, ObjLabel).
objatom(OId, ObjLabel).
obj(X) :- objmolec(X, _).
obj(X) :- objatom(X, _).
lim(LId, OFrom, OTo,Label).
lia(LId, OFrom, OTo,Label).
lmx(LId, OFrom, OTo,Label).
lnk(X) :- lim(X, _, _, _).
lnk(X) :- lia(X, _, _, _).
lnk(X) :- lmx(X, _, _, _).
```

4 Carga de la Base de Datos Semántica

El proceso de carga de los elementos de la SDB consiste en insertar objetos y links. Algunos elementos se pueden obtener durante la fase de extracción, directamente de la KB, como ser los links que asocian a una molécula con sus atributos. Sin embargo, otros links deben deducirse mediante la aplicación de algoritmos específicos. Por ejemplo, un link "is_a" puede detectarse en plena fase de conceptualización mediante la aplicación de algoritmos basados en las dependencias de inclusión [PTB*96]. Los algoritmos necesarios para la fase de extracción generan al esquema lógico de la base fuente (e.g. modelo relacional).

4.1 Carga de Objetos y Links Mixtos

La inserción de moléculas y átomos en la SDB es sencilla: cada tabla en el esquema físico se considera como objeto molecular y cada atributo, como objeto atómico. Para ello se debe

acceder a la información del esquema físico almacenada en un subconjunto de las aserciones de trabajo de la KB [Lar98]. Tales aserciones, usadas como entrada son:

```
kbtable(TblName). /* identificadores de tablas del esquema físico */
kbattribute(TblName, AttName). /* atributos de las tablas */
```

Adicionalmente, se generan los links mixtos que representan la asociación entre cada molécula y sus átomos.

Algoritmo 1. [Inserción de moléculas, átomos y links mixtos en la SDB]
begin
do $\forall\ t_i(a_{i1}, ..., a_{iN}) \in$ kbtable$\otimes$ kbattribute
$M := M \cup \{t_i\}$;
$A := A \cup \{a_{ij}\}, \forall\ j \in \{1,...,N\}$;
$L_{MX} := L_{MX} \cup \{(t_i, a_{ij})\}, \forall\ j \in \{1,...,N\}$;
enddo;
end;

4.2 Carga de links intermoleculares

Este proceso es más sofisticado, pues la detección de un link intermolecular implica analizar restricciones de integridad existentes en el esquema o en las aplicaciones. Fundamentalmente, se estudian las dependencias de inclusión que pueden detectarse a partir de equijoins en las aplicaciones [And94] [PTB*96]. La inserción de links intermoleculares se basa en la información de dependencias de inclusión disponibles. Cada dependencia de inclusión establece un link entre las moléculas a las que corresponden los atributos involucrados en la dependencia. Asumiremos que se han detectado las dependencias de inclusión y que se han almacenado en la KB, en una aserción de trabajo `kbincldep/3`.

Algoritmo 2. [Inserción de links intermoleculares en la SDB]
begin
do $\forall\ idep_i(a_{1i}, a_{2i}) \in$ kbincldep, $\exists\ m_1, m_2 \in M$: $(m_1, a_{1i}) \in L_{MX}$, $(m_2, a_{2i}) \in L_{MX}$
$L_{IM} := L_{IM} \cup \{(m_1, m_2)\}$;
enddo;
end;

4.3 Carga de Links Interatómicos

Los links entre átomos pueden representar diferentes conceptualizaciones, por ejemplo jerarquías dimensionales o estructuración de atributos. Durante la fase de extracción se requieren algoritmos específicos para detectar este tipo de conceptualizaciones. En [PKB*94] se sugiere el análisis de expresiones GROUP BY de SQL para determinar agrupamientos o subcategorías de atributos con posibilidad de descubrir nuevas moléculas de interés; mientras que en [CLR97] proponemos determinar jeraquías dimensionales a partir de GROUP BY. No

se abordará el tema en el presente trabajo.

4.4 Conceptualización de Links

Mediante un análisis de la SDB se puede descubrir semántica adicional en los links. Específicamente, vamos a determinar cuando un link entre moléculas representa una jerarquía de generalización, o cuando representa un relacionamiento entre moléculas. En [RH97] se propone la identificación de links que representan herencia cuando se conocen las declaraciones de claves foráneas en el esquema, a los efectos de generar un modelo conceptual orientado a objetos. En [And94] se utiliza información sobre claves e identificadores de objetos para determinar de qué tipo son los objetos y los links entre ellos, mediante un conjunto de reglas que generan ERC+. En [PTB*96] se esboza un algoritmo para generar un MER simple.

Someramente, el procedimiento consiste en analizar los links intermoleculares salientes de una molécula, surgidos a partir de dependencias de inclusión. Para cada dependencia de inclusión Mi.A< < Mj.B se revisa si el conjunto de atributos A es una clave, está incluído en una clave, o no pertenece a clave alguna. Cuando A es clave de M_i tenemos que el link representa una jerarquía de generalización: M_i is-a M_j. Cuando A forma parte de la clave de M_i tenemos un *property link* [And94]. Concretamente, M_i representa un relationship-type si otros subconjuntos de atributos en su clave son a su vez parte izquierda de otras dependencias de inclusión. Si A no participa en la clave, entonces solamente podemos deducir un property link, es decir un vínculo binario entre las moléculas involucradas.

Algoritmo 3. [Conceptualizar links]
begin
do $\forall$ $idep_i(a_{1i}, a_{2i}) \in$ kbincldep, $\exists$ $m_1, m_2 \in$ M: $(m_1, a_{1i}) \in L_{MX}$, $(m_2, a_{2i}) \in L_{MX}$
if $a_{1i} = pkey(m_1) \wedge a_{2i} = pkey(m_2)$ **then**
(m_1, m_2) *es isa-link*;
else
(m_1, m_2) *es property-link*;
endif;
enddo;
end;

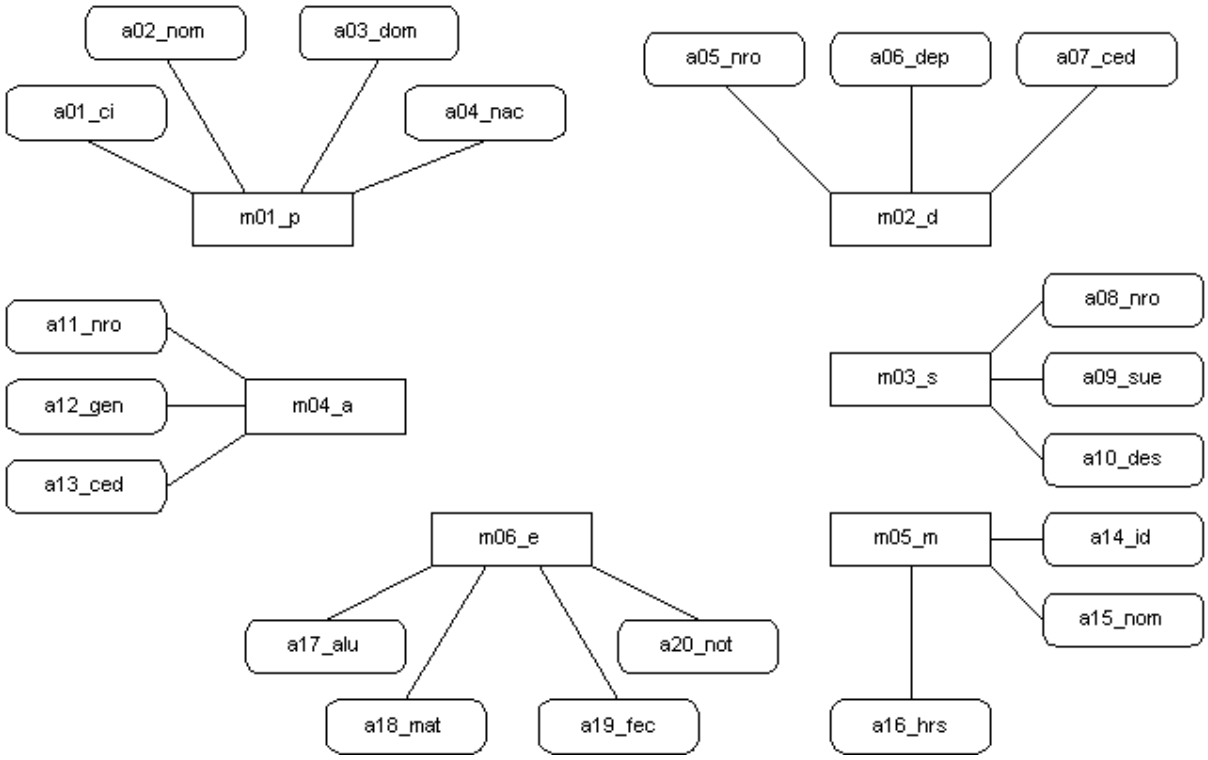
El Algoritmo 3 puede extenderse si se dispone de las *claves candidatas*, además de las claves primarias. Dicha variante se basa en [And94] donde se estudia el concepto de *OId (object id)* como identificador de instancias de objetos. En dicha propuesta es posible determinar si una molécula representa a un entity type, a un relationship type o a un dependent type (atributo multivaluado o entidad débil). Esta variante no se considerará, debido a que no se pretende en la presente propuesta generar directamente un modelo conceptual EER o similar.

4.5 Un Ejemplo

A modo de ejemplo, considérese que se quiere modelar los datos de la gente de una universidad (alumnos y docentes) y los exámenes que los alumnos rinden de las distintas materias. La KB contiene:

```
/* Knowledge Base */
kbtable(persona).
kbtable(docente).
kbtable(sueldo).
kbtable(alumno).
kbtable(materia).
kbtable(examen).
kbattribute(persona, ci).
kbattribute(persona, nom).
kbattribute(persona, dom).
kbattribute(persona, nacim).
kbattribute(docente, nro).
kbattribute(docente, depto).
kbattribute(docente, ced).
kbattribute(sueldo, nrodoc).
kbattribute(sueldo, sueldo).
kbattribute(sueldo, desde).
kbattribute(alumno, nro).
kbattribute(alumno, gen).
kbattribute(alumno, cedula).
kbattribute(materia, id).
kbattribute(materia, nom).
kbattribute(materia, hrs).
kbattribute(examen, alum).
kbattribute(examen, mat).
kbattribute(examen, fecha).
kbattribute(examen, nota).
/* claves primarias */
kbpkey(pk01, persona, ci, 1).
kbpkey(pk02, docente, nro, 1).
kbpkey(pk03, sueldo, nrodoc, 1).
kbpkey(pk04, alumno, nro, 1).
kbpkey(pk05, materia, id, 1).
kbpkey(pk06, examen, alum, 1).
kbpkey(pk06, examen, mat, 2).
kbpkey(pk06, examen, fecha, 3).
```

La ejecución del Algoritmo 1 produce la siguiente SDB representada gráficamente: Se han insertado las moléculas, los átomos y los links mixtos que vinculan a las moléculas con los átomos, representado las moléculas mediante rectángulos, los átomos usando óvalos y los links mixtos mediante líneas continuas.



A continuación se lista la extensión de la SDB correspondiente al SDG de la fig. 1:

```
/* objetos moleculares */
objmolec(m01_p, persona).
objmolec(m02_d, docente).
objmolec(m03_s, sueldo).
objmolec(m04_a, alumno).
objmolec(m05_m, materia).
objmolec(m06_e, examen).
/* objetos atomicos */
objatom(a01_ci, cedula).
objatom(a02_nom, nombre).
objatom(a03_dom, domicilio).
objatom(a04_nac, nacimiento).
objatom(a05_nro, numero).
objatom(a06_dep, depto).
objatom(a07_ced, cedula).
objatom(a08_nro, numero).
objatom(a09_sue, sueldo).
objatom(a10_des, desde).
objatom(a11_nro, numero).
objatom(a12_gen, generacion).
objatom(a13_ced, cedula).
objatom(a14_id, idmateria).
objatom(a15_nom, nombre).
objatom(a16_hrs, horas).
objatom(a17_alu, alumno).
objatom(a18_mat, materia).
objatom(a19_fec, fecha).
objatom(a20_not, nota).
/* links mixtos */
lmx(l01_pci, m01_p, a01_ci, p_ced).
```

```
lmx(l02_pnom, m01_p, a02_nom, p_nom).
lmx(l03_pdom, m01_p, a03_dom, p_dom).
lmx(l04_pnac, m01_p, a04_nac, p_nac).
lmx(l05_dnro, m02_d, a05_nro, d_nro).
lmx(l06_ddep, m02_d, a06_dep, d_dep).
lmx(l07_dced, m02_d, a07_ced, d_ced).
lmx(l08_snro, m03_s, a08_nro, s_nro).
lmx(l09_ssue, m03_s, a09_sue, s_sue).
lmx(l10_sdes, m03_s, a10_des, s_des).
lmx(l11_anro, m04_a, a11_nro, a_nro).
lmx(l12_agen, m04_a, a12_gen, a_gen).
lmx(l13_aced, m04_a, a13_ced, a_ced).
lmx(l14_mid, m05_m, a14_id, m_id).
lmx(l15_mnom, m05_m, a15_nom, m_nom).
lmx(l16_mhrs, m05_m, a16_hrs, m_hrs).
lmx(l17_ealu, m06_e, a17_alu, e_alu).
lmx(l18_emat, m06_e, a18_mat, e_mat).
lmx(l19_efec, m06_e, a19_fec, e_fec).
lmx(l20_enot, m06_e, a20_not, e_not).
```

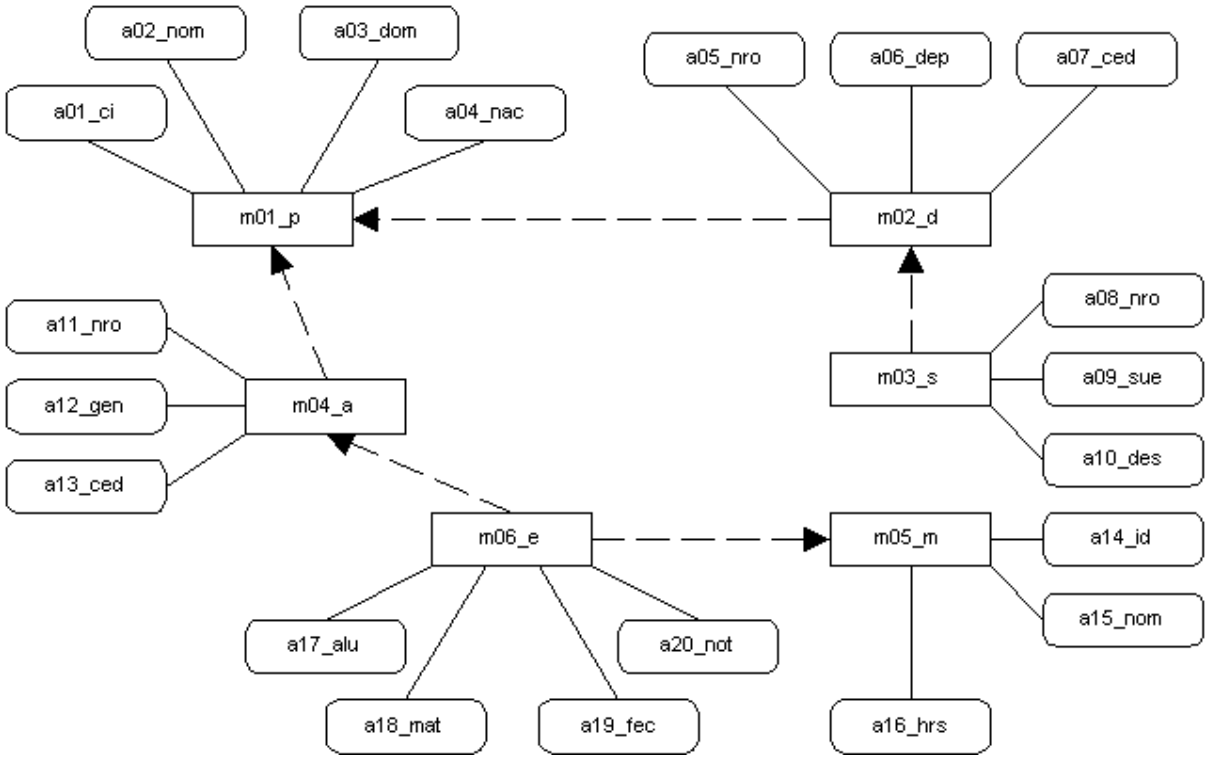
A partir de las dependencias de inclusión -que se detectan directamente de los equijoins [PTB*96]- se establecen vínculos entre las moléculas. En nuestro ejemplo podemos considerar las siguientes:

```
/* dependencias de inclusión */
kbincldep(idep01, a07_ced, a01_ci ).
kbincldep(idep02, a13_ced, a01_ci ).
kbincldep(idep03, a08_nro, a05_nro).
kbincldep(idep04, a17_alu, a11_nro).
kbincldep(idep05, a18_mat, a14_id ).
```

La ejecución del Algoritmo 2 con los datos del ejemplo y los hechos `kbincldep/3`, insertan en la SDB los links intermoleculares:

```
/* links intermoleculares */
lim(l21_dp, m02_d, m01_p, dp).
lim(l22_ap, m04_a, m01_p, ap).
lim(l23_sd, m03_s, m02_d, sd).
lim(l24_ea, m06_e, m04_a, ea).
lim(l25_em, m06_e, m05_m, em).
```

Respectivamente representan a los links entre Docente-Persona, Alumno-Persona, Sueldo-Docente, Examen-Alumno y Examen-Materia. En la fig. 2 se han agregado los links intermoleculares, representados por lineas quebradas.



La ejecución del Algoritmo 3 conceptualiza a todos los links intermoleculares como property-links. A pesar de que intuitivamente podemos pensar que se cumplen las generalizaciones *Docente is-a Persona* y *Alumno is-a Persona*, obsérvese que las dependencias de inclusión que generan a los links intermoleculares `l21_dp` (entre Docente y Persona) y `l22_ap` (entre Alumno y Persona) no involucran a las claves primarias respectivamente de Docente y de Alumno. Por tanto el Algoritmo 3 solamente puede deducir que `l21_dp` y `l22_ap` son property-links.

5 Implementación (Esbozo)

En esta sección se esbozan algunos aspectos de la implementación de las estructuras y algoritmos prototipadas en PROLOG. El Algoritmo 1 puede implementarse mediante las dos reglas siguientes `alg1a/0` y `alg1b/0`.

```
alg1a :-
   kbtable(TblName),
   genIdMolec(IdMolec),
   assert(sdbobjmolec(IdMolec, TblName, TblName)),
   fail.

alg1b :-
   kbattribute(TblName, AttrName),
   genIdAtom(IdAtom),
   genIdLinkMX(IdLink),
   sdbobjmolec(IdMolec, _, TblName),
   assert(sdbobjatom(IdAtom,"", TblName, AttrName)),
   assert(sdblmx(IdLink, IdMolec, IdAtom, "")),
   fail.
```

Se asumen disponibles predicados que generan identificadores en el sistema para las moléculas, átomos y links: `genIdMolec/1`, `genIdAtom/1` y `genIdLinkMX/1`. No se detallan tales predicados.

La siguiente es la implementación del Algoritmo 2, donde `genIdLinkIM/1` es análogo a los ya mencionados:

```
alg2 :-
   kbincldep(_, TblNameFrom, _, TblNameTo, _),
   sdbobjmolec(OIdFrom, _, TblNameFrom),
   sdbobjmolec(OIdTo, _, TblNameTo),
   genIdLinkIM(IdLnk),
   assert(sdblim(IdLnk, OIdFrom, OIdTo, LblLnk)),
   fail.
```

6 Conclusiones

Se ha presentado una estructura denominada *base de datos semántica (SDB)* que se utiliza durante las fases de reingeniería de una base de datos relacional. Durante la fase de conceptualización, a partir del esquema físico (tablas, columnas y claves primarias) y de dependencias de inclusión detectadas -almacenados en *una base de conocimientos (KB)*- se detectaron los objetos y vínculos entre ellos.

Los objetos de interés pueden ser objetos simples denominados *átomos*, u objetos compuestos denominados *moléculas*. Los vínculos o links entre los objetos pueden clasificarse en *links intermoleculares*, *links interatómicos* y *links mixtos*, sgún asocien respectivamente a dos moléculas, dos átomos o una molécula con un átomo. Todos los objetos y links se insertan en la SDB utilizando dos algoritmos para tal fin. Los links luego son analizados a los efectos de determinar jerarquías de generalización en el modelo semántico. De esta forma, el modelo conceptual resultante es un conjunto de objetos vinculados por generalización/especialización o por simples asociaciones binarias.

En el presente trabajo se han prototipado la SDB y los algoritmos de carga utilizando al lenguaje de programación PROLOG. La KB con información del esquema físico se ha cargado manualmente en hechos PROLOG.

Los algoritmos presentados no discriminan a las clases de objetos como asociadas a un entity type o a un relationship type, tal como se conocen en el modelo clásico de entidades y relacionamientos. Algunas propuestas mencionadas en las referencias permiten tal cosa cuando se trabaja con la noción de object identifier (OID) a partir de la detección de claves candidatas en el esquema físico.

Como posibles extensiones a este trabajo se pueden abordar las siguientes:

1. A nivel general, se pretende integrar al conjunto de algoritmos propuestos, al conjunto de algoritmos usados en la fase de extracción. Tales algoritmos se están desarrollando en forma paralela.

2. A nivel de implementación, disponer de una interfaz con la base de datos fuente. Esto permitiría, entre otras cosas, obtener directamente del catálogo o diccionario de datos, información sobre el esquema físico, información de claves foráneas y equijoins disponibles en la definicón de vistas. Adicionalmente, el acceso a la extensión de la

base de datos permitiría realizar conteos de datos que refinarian al modelo obtenido, por ejemplo con mapeos. Tal es el enfoque en [PTB*96].

3. A nivel de algoritmos, el uso de la noción de OID permitiría determinar si las moléculas representan a un entity o a un relationship set. La propuesta de [And94] se enfoca en esa idea.

4. A nivel de salidas, también resulta de interés estudiar qué modelos conceptuales pueden derivarse de la SDB, como ser MER, EER, ERC+, etc.

5. A nivel de aplicaciones, parece interesante considerar herramientas de navegación en la SDB, visualizadores gráficos, aplicaciones de browsing que permitan consultar los datos de la extensión de la base de datos fuente, a partir de los objetos de la SDB.

6. Finalmente, en la presente propuesta no se han tratado los links interatómicos que podrían representar atributos estructurados y jerarquías dimensionales.

---oOo---

Referencias

[Amb87] T. Amble. *Logic Programming and Knowledge Engineering.* Addison-Wesley, 1987.

[And94] M. Andersson. Extracting an Entity Relationship Schema from a Relational Database Through Reverse Engineering, *Proc. 13th Int. Conf. on ER Approach,* Manchester UK. Dec. 1994.

[CM94] W. Clocksin, C. Melish. *Programming in Prolog,* 4th Edition. Springer, 1994.

[Cod70] E. Codd. A Relational Model of Data for Large Shared Data Banks. *Comm. ACM 13, 6.* Jun. 1970.

[Cod79] E. Codd. Extending the Database Relational Model to Capture More Meaning. *ACM 0362-5915.* 1979.

[BCN92] C. Batini, S. Ceri, S. Navathe. *Conceptual Database Design: An Entity-Relationship Approach.* Addison-Wesley. 1992.

[CA97] I. Comyn-Wattiau, J. Akoka. Reverse Engineering of Relational Database Physical Schemas. 1997.

[Che76] P. Chen. The Entity Relationship Model - Toward a Unified View of Data. *ACM TODS, 1(1),* 1976.

[CLR97] M. Colman, G. Larriera, R. Ruggia. Database Reverse Engineering: Proposal of an Open Tool Based on a Semantic Model, *1er. Congreso Uruguayo de Informática,* 1997.

[DEC96] P. Deransart, A. Ed-Dbali, L. Cervoni. *Prolog: The Standard.* Springer, 1996.

[GMR98] M. Golfarelli, D. Maio, S. Rizzi. Conceptual Design of Data Warehouses from E/R Schemes. *Proc. of the Hawaii Int. Conf. On System Sciences.* Jan. 1998, Kona, Hawaii.

[HCT*93] J. Hainaut, M. Chandelon, C. Tonneau, M. Joris. Contribution to a Theory of Database Reverse Engineering, *Working Conferenc. on Reverse Engineering.* Baltimore, May. 1993.

[HK87] R. Hull, R. King. Semantic Database Modeling: Survey, Applications, and Research Issues. *ACM Computing Surveys, Vol. 19, No. 3,* Sep. 1987.

[Joh94] P. Johannesson. A Method for Transforming Relational Schemas into Conceptual Schemas. *Proc. of the 10th Int. Conf. on Data Engineering,* págs. 190-201, Houston, Texas, Feb. 1994. IEEE Computer Society.

[Lar98] G. Larriera. Descripción de una base de Conocimiento para una Herramienta de Reingeniería de Bases de Datos. 1998.

[PKB*94] J. Petit, J. Kouloumdjian, J. Boulicaut, F. Toumani. Using Queries to Improve Database Reverse Engineering. *Proc. of the 13th Int. Conf. on ER Approach*. Manchester, UK. Dec. 1994.

[PM88] J. Peckham, F. Maryanski. Semantic Data Models. *ACM Computing Surveys, Vol. 20, No. 3,* Sep. 1988.

[PTB*96] J. Petit, F. Toumani, J. Boulicaut, J. Kouloumdjian. Towards the Reverse Engineering of Denormalized Relational Databases, *Proc. of 12th Int. Conf.*

on Data Engineering, 1996.

[RH97] S. Ramanathan, J. Hodges. Extraction of Object-oriented Structures from Existing Relational Databases, *ACM SIGMOD Record, Vol. 26 No. 1,* 1997.

[TL82] D. Tsichritzis, F. Lochovsky. *Data Models.* Prentice-Hall, 1982.