# Descripción de una Base de Conocimiento para una Herramienta de Reingeniería de Bases de Datos

**Gustavo Larriera**
siLAB Laboratorio de Sistemas de Información
Universitario Autónomo del Sur - Facultad de Informática
Montevideo, Uruguay
glarrier@ei.edu.uy - http://www.ei.edu.uy/silab/

***Resumen****. Las herramientas de reingeniería de bases de datos permiten construir una especificación conceptual de una base de datos relacional a partir de su estructura, código de operaciones SQL y propiedades de sus datos. Dentro de la arquitectura de tales herramientas se dispone de familias de algoritmos que realizan las tareas de reingeniería. Los resultados obtenidos por tales algoritmos se deben almacenar en una base de conocimientos a los efectos de poder extraer diversos modelos conceptuales.*

*En el presente trabajo se describe el diseño de una base de conocimientos a ser utilizada por una herramienta de reingeniería de base de datos. Esta base de conocimientos permite derivar un modelo conceptual/semántico de la base de datos reingenierizada, con más poder expresivo que el Modelo de Entidades y Relacionamientos, usado clásicamente en los procesos de ingeniería reversa.*

***Palabras clave:*** *Database Reverse Engineering, Databases, Reverse Engineering, 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.

En este artículo se describe la Base de Conocimiento (KB) utilizada por una herramienta de DBRE que está siendo desarrollada. Dicha herramienta construye un esquema conceptual/semántico a partir de un esquema de una base de datos relacional operativa sometida a estudio. Dentro de la arquitectura de la herramienta, la KB es la estructura responsable de generar diferentes representaciones conceptuales de la base en estudio. Como caso particular, pero sin limitarse a él, la KB genera un Modelo de Entidades y Relacionamientos (MER) [Che76].

Las principales contribuciones de este trabajo son: (i) La descripción de dicha Base de Conocimiento; (ii) la representación de la misma utilizando lógica de predicados. La implementación de la KB se ha realizado utilizando el lenguaje PROLOG [CM94, DEC96].

El trabajo presentado ha sido desarrollado en el contexto del proyecto "Técnicas y herramientas para Ingeniería Reversa de Bases de Datos" del Laboratorio de Sistemas de Información, Facultad de Informática del Universitario Autónomo del Sur.

El resto del artículo se estructura de la siguiente forma. La Sección 2 introduce las componentes arquitectónicas de la herramienta de DBRE encargadas de construír la especificación semántica. La Sección 3 describe a la base de conocimiento. La Sección 4 presenta un caso de estudio. Finalmente, la Sección 5 presenta conclusiones y trabajo futuro.

# 2 Componentes de la herramienta DBRE

La herramienta DBRE [CLR97] es un software abierto para reingeniería de bases de datos. La herramienta se considera "abierta" en el sentido de que está diseñada para soportar diferentes familias de algoritmos de reingeniería, en forma intercambiable. Esto permite sustituír cualquier algoritmo por otro, en tanto se respete la interfaz del mismo hacia el resto de los módulos. Dentro de la arquitectura de la herramienta DBRE se dispone de un módulo encargado de la construcción de la especificación semántica de la base en estudio, denominado *módulo constructor de la especificación semántica (MCES)*. Este módulo nutre de información a la Base de Conocimiento (KB). Dentro del MCES están los *algoritmos de reingeniería* y el *algoritmo de derivación de la especificación semántica.* Este último toma la salida de los algoritmos de reingeniería y carga a la KB (ver Fig. 1).

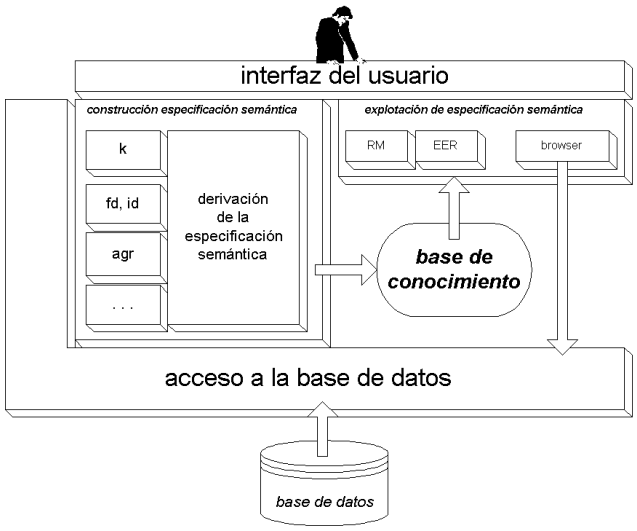


**Fig. 1 - Arquitectura de la herramienta DBRE.**

Los algoritmos de reingeniería se agrupan en *familias* según el tipo de objetos que detectan: *detectores de claves* (algoritmos-K) [PTB*96], *detectores de dependencias* funcionales y de inclusión (algoritmos-FD y algoritmos-ID) [PTB*96] [Chi95] y *detectores de agrupamientos de atributos* (algoritmos-AGR). Los algoritmos de reingeniería obtienen información de diversos orígenes: el esquema de la base en estudio, la extensión de la base y las operaciones SQL existentes en las aplicaciones. La información obtenida, asi también como el grado de confiabilidad de la misma -dependiente del origen de donde se obuvo dicha información- se almacena como resultado intermedio, en las denominadas *aserciones de trabajo*. Estas aserciones de trabajo son utilizadas por el algoritmo de derivación semántica, para construir la especificación semántica, que se almacenará en la KB.

# 3 La Base de Conocimiento

La Base de Conocimiento (KB) almacena toda la información generada durante el proceso global de reingeniería, realizado por los módulos que construyen la especificación semántica de la base de datos que está siendo reingenierizada. La información almacenada en la KB se clasifica en dos tipos: (i) *Aserciones del esquema*, que almacenan datos primarios del esquema y restricciones de integridad de la base de datos reingenierizada, y (ii) la *especificación semántica*, que contiene toda la información de los objetos conceptuales de la base (e.g. entidades, atributos, etc.) y las asociaciones entre ellos (e.g. relacionamientos, tipos de mapeos, etc.).

## *3.1 Las aserciones del esquema*

Las *aserciones del esquema* contienen la información del esquema físico de la base de datos, dependencias funcionales, dependencias de inclusión, condiciones de join, y atributos de agrupamiento. Podemos distinguir dos grupos de aserciones del esquema:

(i) Las *aserciones del esquema básicas*, obtenidas directamente del esquema físico y de las expresiones SQL. En esta categoría tenemos las aserciones que representan información estructural de la base de datos reingenierizada (información sobre tablas y sus atributos), aserciones que representan vínculos entre las tablas (expresiones de join SQL) y aserciones que representan agrupamientos de atributos (expresiones GROUP BY de SQL).

(ii) Las *aserciones del esquema derivadas*, obtenidas mediante aplicación de algoritmos. En esta categoría tenemos a las aserciones construidas por los algoritmos detectores de las diversas familias.

Por ejemplo, las siguientes son las aserciones necesarias cuando se utilizan los algoritmos de [PTB*96]:

- ***table(TblName):*** Tablas de la base de datos. Se pueden obtener directamente del esquema físico.
- ***attribute(TblName, AttName, Unique?, NotNull?):*** Atributos de las tablas y sus propiedades intensionales. Se pueden obtener directamente del esquema físico.
- ***join(TblName1, AttLst1, TblName2, AttLst2):*** Condiciones de join entre dos tablas. Se pueden obtener de las cláusulas WHERE de operaciones SQL SELECT.
- ***incldep(TblName1, AttLst1, TblName2, AttLst2):*** Dependencias de inclusión entre dos tablas. Se pueden obtener mediante algoritmos-ID.
- ***funcdep(TblName, AttLst1, AttLst2):*** Dependencias funcionales. Se pueden obtener mediante algoritmos-FD.
- ***groupby(TblName, AttLst):*** Agrupamientos de atributos de una tabla. Se pueden obtener mediante algoritmos-AGR.
- ***pkey(TblName, AttLst):*** Claves primaria de tabla. Se pueden obtener directamente del esquema físico y mediante algoritmos-K.
- ***dtable(TblName):*** Nuevas tabla derivadas. Algunos algoritmos permiten derivar nuevas tablas a partir de las existentes.

Adicionalmente, algunos algoritmos propuestos para la detección de dependencias de inclusión, se basan en el análisis de los datos existentes en la extensión de la base. Por ejemplo, el algoritmo denominado *IND-Discovery* [PTB*96] realiza conteos sobre proyecciones de tablas y sobre resultados de equijoins. El uso de tales algoritmos requiere entonces la representación en las aserciones de trabajo, de los siguientes predicados:

- ***n(TblName, AttLst, Cnt):*** Conteos de tuplas según proyección de atributos. Se pueden obtener ejecutando SELECT COUNT sobre la extensión de la tabla.
- ***njoin(TblName1, AttLst1, TblName2, AttLst2, Cnt):*** Conteos de tuplas resultantes de join. Se pueden obtener ejecutando SELECT COUNT sore el resultado de join de tablas.

Es importante tener en cuenta que la información obtenida surge de la aplicación de diversos algoritmos pertenecientes a diferentes familias. Mediante esa forma de trabajo, algunos algoritmos obtienen

información en forma más confiable que otros. Por lo tanto, resulta de importancia disponer de alguna manera de medir y comparar la calidad de los resultados obtenidos. Para ello se introduce un valor de medida al que se denomina *grado de confiabilidad,* que puede representarse como un valor porcentual. Por ejemplo, los resultados basados en información estructural de la base son más confiables que los resultados obtenidos mediante el análisis de operaciones de consulta. Por simplicidad, no se ha puesto el grado de confiabilidad en cada una de las aserciones de trabajo mostradas.

## *3.2 Especificación semántica*

La especificación semántica captura la información conceptual de la base de datos fuente, mediante un modelo semántico, que representa un amplio espectro de *objetos* de la base y *vínculos* (links) entre ellos. Los objetos pueden ser *átomos* (representaciones de objetos simples, como ser atributos de entidades) o pueden ser *moléculas* (representaciones de objetos compuestos, como ser entidades). Los vínculos representan *asociaciones* definidas entre los objetos, ya sean átomos o moléculas.

Disponemos de dos *categorías de vínculos,* a saber: (i) Vínculos entre moléculas y (ii) vínculos entre moléculas y átomos. Para cada vínculo se considera a su vez, un *vínculo inverso*. Esquemáticamente:

| | M | A |
|---|---|---|
| M | generalization_of / specialization_of<br>component_of / composed_of<br>association | has_attribute |
| A | attribute_of | is_part_of / comprises |

Cada vínculo entre objetos, tiene asociado un *origen*. El origen expresa la fuente de la cual se obtuvo la información. Tenemos en cuenta que la carga de los datos de la KB es realizada por diferentes módulos de construcción, pertenecientes a diferentes familias de algoritmos de reingeniería: detectores de claves, detectores de dependencias y detectores de agrupamientos de atributos [CLR97]. Almacenar la información de origen resulta de interés para clasificar la calidad y fiabilidad de la información obtenida, mediante la asignación del grado de confiabilidad.

A los efectos de describir a los vínculos consideremos a los siguientes conjuntos: M es el conjunto de todas las moléculas, A es el conjunto de todos los átomos, O es el conjunto de todos los orígenes, C = { one, many } (conjunto de cardinalidades), U = { unique, distinct } (conjunto de unicidades), V = { one, many } (conjuno de valoraciones). Los siguientes son los predicados que representan a la especificación semántica:

- **generalization_of(m1, m2, o)** expresa que la molécula $m1 \in M$ es una generalización de la molécula $m2 \in M$. El vínculo inverso es **specialization_of.**

- **component_of(m1, m2, o)** expresa que la molécula $m1 \in M$ es una componente de la molécula $m2 \in M$. El vínculo inverso es **composed_of**.

- **association(m1, m2, c, o)** expresa que la molécula $m1 \in M$ está asociada a la molécula $m2 \in M$ con determinado mapeo $c \in CxC$.

- **attribute_of(a, m, u, o)** expresa que $a \in A$ es atributo de la molécula $m \in M$, con cierta unicidad $u \in U$. El vínculo inverso es **has_attribute(m, a, v, u, o).**

- **is_part_of(a1, a2, v, o)** expresa que el atributo $a1 \in A$ es parte del atributo $a2 \in A$, con cierta valoración $v \in V$. El vínculo inverso es **comprises_of.**

### *3.3 Derivación de la especificación semántica*

La especificación semántica se deriva de las aserciones del esquema. Los átomos y moléculas pueden derivarse de la siguiente manera:

```
objatom(AttributeName, 'TYPE_ATTRIBUTE') :- attribute(_, AttributeName, _, _).
objmolec(TableName, 'TYPE_TABLE') :- table(TableName).
```

La derivación de los distintos tipos de vínculos, a los efectos de construir un esquema conceptual, puede realizarse mediante algoritmos variados, a saber: reestructuración del esquema desnormalizado en un esquema en 3NF apto para ser representado mediante un MER extendido (EER) [PTB*96], reglas de refinamiento aplicadas sobre un diagrama de conexión entre los objetos de la base [And94], o procesos que buscan detectar optimizaciones físicas realizadas en el esquema a los efectos de aplicarlas en reversa para lograr el modelo conceptual [HTJ*94].

# 4 Un ejemplo

Considérese el siguiente ejemplo. Sean las siguientes tablas de una base de datos de personal asignado a proyectos, a ser reingenierizada (observar que el esquema está desnormalizado) y un conjunto de operaciones SQL utilizadas para extraer información:

| **P /* Personas */** | | | | |
|---|---|---|---|---|
| **id** | **nom** | **dom** | **zip** | **ciudad** |
| e1 | Juan | Cuareim 1522 | 11100 | Montevideo |
| e2 | Maria | Salto 450 | 11000 | Montevideo |
| e3 | Pedro | Obligado 876 | 11300 | Montevideo |
| e4 | Isabel | Ellauri 450 | 11300 | Montevideo |
| e5 | Luis | Yi 4850 | 11100 | Montevideo |

| **D /* Departamentos */** | | | |
|---|---|---|---|
| **dep** | **jefe** | **nomdep** | **proy** |
| d1 | e1 | Bdatos | p1 |
| d2 | e2 | Pgmcion | p2 |
| d3 | e3 | Soperativ | p3 |

| **A /* Asignaciones de empleados a proyectos */** | | | | |
|---|---|---|---|---|
| **emp** | **dep** | **proy** | **fecha** | **nomproy** |
| e1 | d1 | p1 | 1/1/97 | Geminis |
| e2 | d2 | p2 | 6/6/97 | Aries |
| e4 | d1 | p1 | 1/1/97 | Geminis |
| e5 | d2 | p2 | 8/8/97 | Aries |

| **H /* Historia salarial de empleados */** | | |
|---|---|---|
| **nro** | **fecha** | **sueldo** |
| e1 | 1/1/97 | $2000 |
| e2 | 8/8/97 | $1500 |
| e3 | 1/1/97 | $1800 |
| e4 | 6/6/97 | $1900 |
| e2 | 1/1/98 | $2000 |
| e1 | 1/1/98 | $2500 |

Considérese el siguiente conjunto de sentencias SQL utilizadas para recuperar información de la base de datos, del cual se extraen las expresiones Q de equijoin y G de agrupamientos:

```
Q1: select * from H, P where H.nro = P.id ;
Q2: select * from D, H where D.jefe = H.nro ;
Q3: select * from A, H where A.emp = H.nro ;
Q4: select * from A, D where A.dep = D.dep ;
Q5: select * from D, A where D.proy = A.proy ;
Q6: select * from P group by ciudad, zip ;
Q7: select * from H group by nro, fecha ;

Q = { H.nro ⊗ P.id , D.jefe ⊗ H.nro , A.emp ⊗ H.nro , A.dep ⊗ D.dep , D.proy ⊗
A.proy }

G = { {P.ciudad, P.zip} , {H.nro, H.fecha} }
```

Usando algoritmos-ID, algoritmos-FD y algoritmos-K, se pueden detectar respectivamente los conjuntos de claves primarias, dependencias de inclusión y dependencias funcionales (no derivadas directamente a partir de claves primarias):

```
K = { P(id) , D(dep) , A(emp,dep,proy) , H(nro,fecha) }

I = { H[nro] ⊆ P[id] , D[jefe] ⊆ H[nro] , A[emp] ⊆ H[nro] , D[proy] ⊆ A[proy] ,
S[dep] ⊆ A[dep] , S[dep] ⊆ D[dep] }

F = { D:dep → nomdep , A:proy → nomproy , A: emp → proy}
```

Algunos algoritmos permiten detectar nuevas tablas. Por ejemplo, el algoritmo IND-Discovery [PTB*96] perteneciente a la familia de algoritmos-ID, genera una nueva tabla S cuando, al revisar la extensión de las tablas, aparece una intersección no vacía en las proyecciones según expresiones de equijoin. Por ejemplo, usando el equijoin *A[dep] Ä D[dep]*, se calcula la cantidad de tuplas en la intersección no vacía: *count(A[dep] Ç D[dep])* y se compara el conteo contra el conteo de proyecciones *count(A[dep])* y *count(D[dep])*.La nueva tabla derivada representa a los departamentos que tienen empleados asignados a sus proyectos.

| **S /* A[dep] Ç D[dep] */** |
|---|
| **dep** |
| d1 |
| d2 |

La información detectada se almacena en las aserciones del esquema, de la siguiente manera[1]:

```
% Tablas del esquema y nuevas tablas derivadas
table(p).
table(d).
table(a).
table(h).
dtable(s).

% Atributos de las tablas y sus propiedades
attribute(p, id, true, true).
attribute(p, nom, true, true).
attribute(p, dom, false, false).
attribute(p, zip, false, false).
attribute(p, ciudad, false, flase).
attribute(d, dep, true, true).
```

[1] No se incluyen en el ejemplo las aserciones intermedias de trabajo *n* y *njoin* utilizadas por el algoritmo IND-Discovery de [PTB*96] para detectar dependencias de inclusión.

```
attribute(d, jefe, false, true).
attribute(d, nomdep, true, true).
attribute(d, proy, false, false).
attribute(a, emp, false, true).
attribute(a, dep, false, true).
attribute(a, proy, false, true).
attribute(a, fecha, false, true).
attribute(a, nomproy, false, true).
attribute(h, nro, false, true).
attribute(h, fecha, false, true).
attribute(h, sueldo, false, true).
attribute(s, dep, true, true).

% Claves primarias
pkey(p, [id]).
pkey(d, [dep]).
pkey(a, [emp, dep, proy]).
pkey(h, [nro, fecha]).
pkey(s, [dep]).

% Dependencias funcionales derivadas
funcdep(p, [zip], [ciudad]).
funcdep(a, [proy], [nomproy]).
funcdep(a, [emp], [proy]).

% Dependencias de inclusion
incldep(h, [nro], p, [id]).
incldep(d, [jefe], h, [nro]).
incldep(a, [emp], h, [nro]).
incldep(d, [proy], a, [proy]).
incldep(s, [dep], a, [dep]).
incldep(s, [dep], d, [dep]).

% Agrupamientos de atributos
groupby(e, [ciudad, zip]).
groupby(h, [nro, fecha]).

% Expresiones de equijoins
join(h, [nro], p, [id]).
join(d, [jefe], h, [nro]).
join(a, [emp], h, [nro]).
join(a, [dep], d, [dep]).
join(d, [proy], a, [proy]).
```

Para construir el esquema conceptual a partir de la información anterior, se pueden utilizar algoritmos que traduzcan al esquema relacional en estructuras conceptuales, habitualmente en 3NF [Joh94, MM90]. La aplicación del algoritmo *Restruct* de [PTB*96] sobre la información almacenada en las aserciones anteriormente mostradas, produce como salida un esquema relacional normalizado a 3NF que puede traducirse en estructuras de EER (Extended Entity-Relationship). Como ejemplo, se muestra una parte de la especificación semántica[2] derivada con dicho algoritmo:

```
generalization_of(p, e). /* Empleado is-a Persona */
generalization_of(e, j). /* Jefe is-a Empleado */
association(j, d, (many, one)). /* Relationship jefe-de-departamento */
association(j, p, (many, one)). /* Relationship jefe-de-proyecto */
association(e, p, (many, many)). /* Relationship empleado-en-proyecto */
...
```

Los atributos de entidades y de relacionamientos se derivan con la regla:

[2] No se han incluído en los predicados los orígenes de donde se obtuvo la información.

```
attribute_of(A, M, U) :-  objatom(A, 'TYPE_ATTRIBUTE'),
                          objmolec(M, 'TYPE_TABLE'),
                          attribute(M, A, U, _).
```

Generar un modelo de entidades y relacionamientos a partir de la especificación semántica almacenada en la KB no presenta dificultades mayores, ya que la especificación semántica modeliza todos los conceptos presentes en el Modelo Entidad-Relación Extendido. A modo de ejemplo, se muestra un fragmento del algoritmo de generación, bajo la forma de reglas PROLOG que implementan el esquema Entidad-Relación en función de los predicados que representan la especificación semántica:

```
entity_type(O) :-   objmolec(O, 'TYPE_TABLE'),
                    not association(O,_,_).

relationship_type(O) :-   objmolec(O, 'TYPE_TABLE'),
                          association(O,_,_).

link_Ent_Rel(R,E,Card) :- objmolec(R, 'TYPE_TABLE'),
                          objmolec(E, 'TYPE_TABLE'),
                          association(R,E,Card).
...
```

# 5 Conclusión

En este artículo se ha realizado la descripción de la base de conocimiento (KB) utilizada por una herramienta abierta de ingeniería reversa de base de datos. La KB es la estructura que almacena toda la información generada a partir de algoritmos de reingeniería, pertenecientes a diversas familias: detectores de claves, detectores de dependencias (funcionales y de inclusión) y detectores de agrupamientos de atributos. La información obtenida de una base de datos sometida al proceso de reingeniería se puede dividir en *infomación del esquema físico* y en *información semántico conceptual* derivada a partir del esquema físico, de las operaciones SQL de recuperación de datos y del conocimiento experto del usuario.

La implementación de la KB se ha realizado mediante un conjunto de predicados denominado *aserciones del esquema*, que almacenan la información del esquema físico de la base de datos, y otro conjunto de predicados denominado *especificación semántica*, que almacenan la información referente a los objetos conceptuales y los vínculos entre ellos. Esta especificación semántica se deriva de las aserciones de esquema mediante algoritmos propuestos por diversos autores.

Como trabajo futuro a corto plazo, se pretende desarrollar los algoritmos necesarios para manejar información de agrupamiento de atributos, presentes en cláusulas GROUP BY de SQL. Los agrupamientos de atributos permitirían detectar *jerarquías dimensionales* en el modelo conceptual, no existentes en los modelos clásicos (MER, EER) utilizados en otras propuestas. La detección de estas jerarquías dimensionales serán de aplicación inmediata en la construcción de datawarehouses.

---oOo---

# Referencias


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

[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.

[Chi95] R. Chiang. A Knowledge-based System for Performing Reverse Engineering of Relational Databases, 1995.

[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.

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

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

[HTJ*94] J. Hainaut, C. Tonneau, M. Joris, M. Chandelon. Transformation-based Database Reverse Engineering, *Proc. 12th Int. Conf. on ER Approach,* 1994.

[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.

[MM90] V.M. Markowitz, J.A. Makowsky. Identifying Extended Entity-Relationship Object Structures in Relational Schemas. *IEEE Transactions on Software Engineering*, 16(8):777-790, Aug. 1990.

[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.