

# Universal Data Model

## Proyecto de Grado

Facultad de Ingeniería. INCO  
Universidad de la República Oriental del Uruguay

### Tutores

Sandro Moscatelli  
Nicolás Jodal

---

Pablo Alza	3.104.068-7
Guzmán Montaña	2.599.150-5
Alexander Wolff	3.914.665-3

---

## RESUMEN

En 1995, David C. Hay publica su primer libro "Data model patterns: conventions of thought" [1], en el cual presenta de forma muy breve un modelo de datos relacional con un conjunto fijo de tablas al que denomina Universal Data Model (UDM). El autor proclama que el modelo permite representar cualquier realidad. Esta atractiva característica del modelo UDM sumado a la falta de estudios sobre el mismo, motivan este proyecto.

A lo largo del proyecto se investigan ciertos aspectos relevantes del modelo como ser la capacidad de UDM (potencia) para representar las realidades que permite representar un MER y la eficiencia de las operaciones de acceso a datos. En este estudio se presentan algunas modificaciones del modelo que fueron realizadas con el objetivo de mejorar la potencia del mismo.

Por último se integra UDM en un escenario de uso donde se pueden apreciar sus virtudes. En este escenario UDM participa como motor de almacenamiento de una herramienta del tipo "Model by example"

**Palabras clave:** Modelo de datos, Modelado de realidades, Técnicas de desarrollo.

---

# Indice

## 1. Introducción

1.1 Motivación

1.2  
Objetivos

## 2. Estado del Arte

2.1 Universal Data model

2.2 Model by Example

## 3. Estudio del Modelo

3.1 Evolución en la Investigación del Modelo.

3.1.1 Etapa 1

3.1.2 Etapa 4

3.1.3 Etapa 5

3.2 Representación de un Esquema Entidad –  
Relación en UDM v5.1

3.2.1 Algoritmo para el mapeo de un esquema Entidad  
Relacional al modelo UDM

3.2.2 Ejemplo del uso del algoritmo

## 4. Estudio de Performance del Modelo

## 5. Aplicación “Model by Example”

5.1 Análisis de versión de UDM a usar

5.2 Análisis del editor a usar

## 6. Conclusiones

6.1 Conclusiones Generales

6.2 Conclusiones Particulares

6.3 Trabajo Futuro

## 7. Bibliografía

---

## 1 Introducción

### 1.1 Motivación

El modelo relacional [2] se basa en el hecho de almacenar proposiciones de la realidad en tablas y en donde se tiene una (o varias) tabla por cada conjunto homogéneo de estas proposiciones (una tabla para Cliente, una o mas tablas para Factura, etc.). Cualquier cambio de la realidad puede ocasionar una reestructuración de las tablas relacionales existentes.

En contraposición al modelo relacional, algunos investigadores [1] han sugerido que es posible tener un modelo de datos denominado Universal Data Model (UDM) con un conjunto fijo de tablas relacionales a través de las cuales se puede representar cualquier proposición de la realidad. Esta característica de UDM, lo hace muy atractivo para ser usado en ciertos escenarios con alta frecuencia de cambios sobre estructuras de datos. Dado que cualquier cambio en la realidad representada no implica una reestructura de tablas relacionales del modelo.

La principal motivación de este proyecto es investigar ciertos aspectos relevantes de UDM y usar el modelo en un escenario donde se aprecien sus virtudes.

### 1.2 Objetivos

En primer lugar se quiere investigar que tan poderoso resulta el modelo, en el sentido que se quiere determinar si permite representar las realidades que son representables a través de un MER.

En particular es de interés investigar si el modelo permite modelar:

- Atributos derivados
- Atributos Multivaluados
- Relaciones entre mas de 2 objetos

Otro aspecto a investigar sobre el modelo es la eficiencia (performance) de las operaciones de acceso a datos sobre el mismo. En particular es de interés estudiar el tiempo de respuesta de la operación equivalente en UDM a la operación "Join" sobre el modelo relacional.

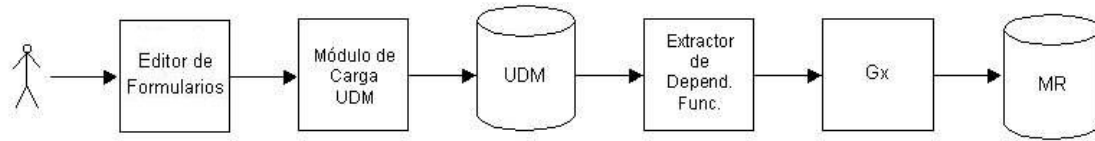
El último objetivo del proyecto es utilizar el modelo UDM en el siguiente escenario:

Como motor de almacenamiento para una herramienta del tipo "Model by example" en la cual un usuario final expresa en forma electrónica el conocimiento de una realidad como por ejemplo un editor de formularios en donde el usuario dibuja formularios y el editor produce la salida en algún formato conocido. Luego la herramienta crea la base de datos que se corresponde con la realidad.

---

La arquitectura y operativa de la herramienta es la siguiente:

**Figura**  
**R.1.**  
**Arquitectur**  
**a y**  
**operativa**



El proceso es el siguiente: El usuario dibuja formularios (forms) e ingresa datos de ejemplo. El Módulo de Carga UDM carga la información de los forms en el modelo UDM. El Módulo Extractor de Dependencias Funcionales, infiere las mismas a partir de la información almacenada en el modelo UDM y luego se crea un archivo XML [6] que sirve de interfase con GeneXus (GX) [5].

Luego GeneXus normaliza las estructuras de datos y crea el esquema relacional. Además genera los programas para realizar altas, bajas, modificaciones y consultas sobre la base relacional creada.

De esto surge un objetivo secundario que es la elección de un editor de formularios.

## 2 Estado del arte

### 2.1 Universal Data Model (UDM)

En 1995, David publica su primer libro "Data model patterns: conventions of thought"[1]. El libro presenta la postura de que las estructuras de datos de muchos negocios y entidades gubernamentales son similares. En un afán por obtener modelos de datos genéricos para cada negocio, surge la idea de diseñar un modelo que permita representar cualquier realidad. El producto de esta idea es el modelo UDM (Universal Data Model), un meta-modelo que se basa de un conjunto fijo de 7 tablas relacionales.

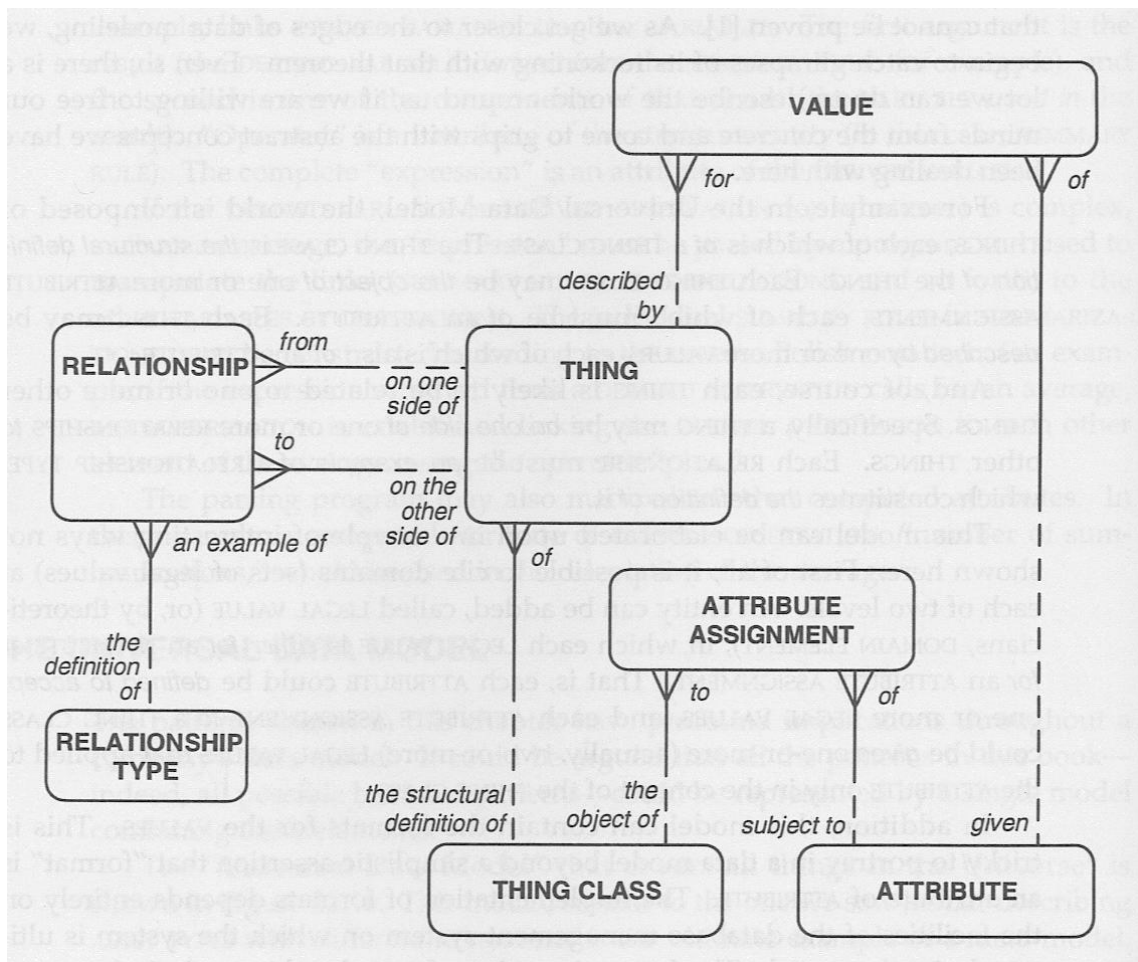
Cabe destacar que si bien el autor proclama que UDM permite representar cualquier realidad, hasta el momento de este proyecto de grado no existe análisis sobre el modelo que prueben que esto se cumple. Mas en general aún, resulta que la información sobre UDM que se puede encontrar en búsquedas de Internet es muy escasa. A comienzos del proyecto se consultó directamente a David C. Hay sobre el estado actual de UDM y el mismo confirmó que no existen estudios del modelo.

A continuación se presentan los conceptos de UDM a través de los cuales se representa una realidad:

En UDM la realidad está compuesta de OBJETOS (THING) y cada uno de estos pertenece a una CLASE DE OBJETO (THING CLASS). La CLASE DE OBJETO define la estructura del OBJETO. Cada CLASE DE OBJETO tiene asociado un conjunto de ATRIBUTOS. Cada OBJETO se describe a través de los valores que toman sus ATRIBUTOS. A la vez, se dan RELACIONES entre OBJETOS donde cada RELACION pertenece a un TIPO DE RELACION.

La siguiente figura que se tomó de [1], representa las entidades del modelo UDM y sus relaciones:

---



## 2.2 Model by example

Durante el proceso de desarrollo de un producto de software orientado a bases de datos, uno de las etapas del proceso es la obtención del modelo de datos lógico (por ejemplo modelo relacional). Típicamente se haría un diseño conceptual de la realidad (por ejemplo usando MER [2]) y luego se expresaría el mismo a través del correspondiente esquema relacional. Esta técnica de obtención del modelo de datos es bien conocida y usada pero no la única.

Una alternativa es la técnica "Model by example" que permite obtener el modelo de datos a través de un enfoque diferente. En lugar de ser un analista de sistemas quien obtiene el modelo de datos a partir de un diseño, el mismo se obtiene de la siguiente forma: primero un usuario final describe en forma electrónica el conocimiento necesario para obtener el modelo de datos. En un segundo paso una herramienta desarrollada para operar bajo esta técnica obtiene el modelo de datos a partir del conocimiento electrónico expresado por el usuario.

A modo de ejemplo se presenta un posible escenario de uso de la técnica "Model by example", en el cual se tiene como objetivo deducir el modelo de

**Figura R.3. Diseño de tres formularios** datos de un sistema de facturación a partir de un conjunto de formularios electrónicos, que el usuario diseña y rellena con datos.

El proceso en este caso sería: el usuario diseña formularios electrónicos para el almacenamiento de clientes, productos y facturas e ingresa datos en los formularios.

A continuación se muestra el diseño de estos formularios.

### Cientes

Documento	Nombre	Direccion	Telefono
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>


Tabla de repetición

### Productos

Codigo	Descripcion	Precio
<input type="text"/>	<input type="text"/>	<input type="text"/>

Tabla de repetición

### Factura

Numero:  Fecha:  

**Ciente**

Documento:  Nombre:

Direccion:

Sección

**Detalle**

NroL	Producto	Precio	Cantidad	Sub Total
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Tabla de repetición

**Importe Total:**

Lo que se esperaría en este caso de una herramienta desarrollada para operar según la técnica de "Model by example" es que deduzca el modelo de datos a partir del diseño de los formularios y de los datos de prueba ingresados en los mismos y lo exprese a través de un modelo de datos en concreto.



### 3 Estudio del modelo

#### 3.1 Evolución en la Investigación del Modelo

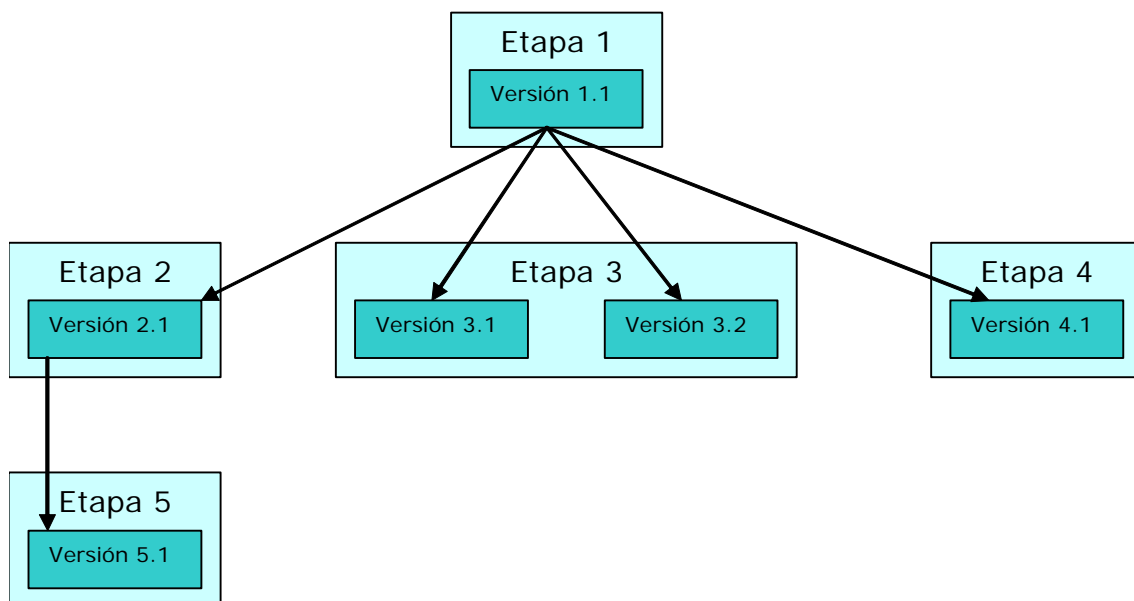
Figura R.4. Esta sección se refiere al entendimiento e investigación de UDM en cuanto a su potencia, donde el término potencia en este contexto se define como la capacidad de poder representar cualquier realidad.

Arbol de evolución de El proceso de investigación se desarrolla en 5 etapas. Se enumeran las etapas según el orden cronológico en el cual fueron sucediendo. La primera etapa tiene como objetivo estudiar el modelo original de David Hay presentado en [1] y determinar sus posibles limitaciones o no en el aspecto de si permite representar cualquier realidad.

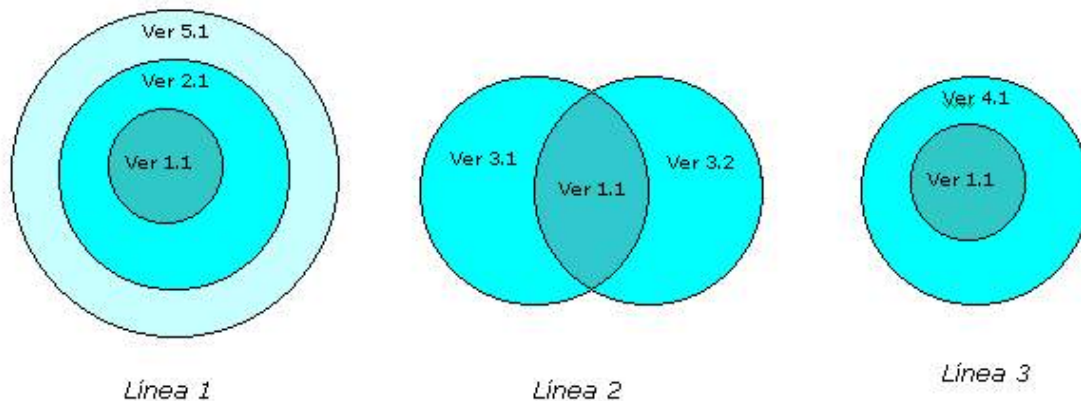
Como se detalla mas adelante en esta sección, el modelo original no tiene la potencia esperada. Esto motiva un estudio de evolución del mismo que da lugar a las etapas restantes en las cuales se agregan o se quitan elementos que permiten abarcar o modelar nuevos aspectos de la realidad. En cada una de estas etapas se genera en carácter de prototipo una o mas versiones del modelo donde se toman o no conceptos incluidos en la etapa anterior.

La nomenclatura utilizada para la numeración de las versiones de UDM toma la forma Etapa.Numero, por ejemplo la versión 3.2 correspondería al segundo modelo prototipo de la etapa 3.

A continuación se muestra un diagrama de etapas de investigación y versiones del modelo. Las flechas que van de una versión v1 a una versión v2 indican que v2 es una evolución de v1.



**Figura R.5.** De la figura anterior se aprecia que el conocimiento no es Organización incremental en la evolución de las etapas. La organización incremental de las versiones sería entonces la siguiente: de las



De la figura anterior se desprende que hubo 3 líneas de evolución del modelo: *Línea 1*, *Línea 2* y *Línea 3*.

En cada Línea de Evolución se encuentran versiones de UDM que comparten una idea. Las versiones de UDM que se encuentran en la *Línea 1* tiene como característica sobresaliente, el hecho de que las relaciones entre Clases se modelan de una forma similar al modelo relacional.

Las versiones de UDM que se encuentran en la *Línea 2*, se destacan por representar las relaciones entre clases y las claves como objetos pertenecientes al propio UDM.

Las versiones de UDM que se encuentran en la *Línea 3*, se caracterizan por representar de forma natural los diferentes constructores del modelo Entidad Relación.

A continuación se describe en forma breve las diferentes etapas de investigación.

En la etapa 1 se hace una interpretación en un esquema relacional del diagrama conceptual de la *figura R.2* de la página 5 y se determinan sus limitaciones.

En la etapa 2 se genera la versión 2.1 de UDM, esta versión levanta las limitaciones de la versión original de no poder determinar las clases que participan de cierta relación así como también la incapacidad de representar el concepto de clave.

En la etapa 3 se generan las versiones 3.1 y 3.2 de UDM que se caracterizan por representar las relaciones entre clases y las claves como objetos pertenecientes al propio UDM.

En la etapa 4 se genera la versión 4.1 de UDM que se caracteriza por representar de forma natural los diferentes constructores de un modelo conceptual. Para lograr esto se decide modificar y extender lo que sea necesario en la estructura del modelo de la versión 1.1 para modelar cualquier realidad.

En la etapa 5 se genera la versión 5.1 de UDM que es una versión mejorada de la versión 2.1. El enfoque seguido en esta etapa es de

Figura R.6. representar lo mejor posible la información almacenada en formularios ya que será usado como motor de almacenamiento en la aplicación y mantener la representividad como modelo conceptual.

A los efectos de mostrar los resultados mas relevantes, se presentan las etapas 1, 4 y 5.

### 3.1.1 Etapa 1

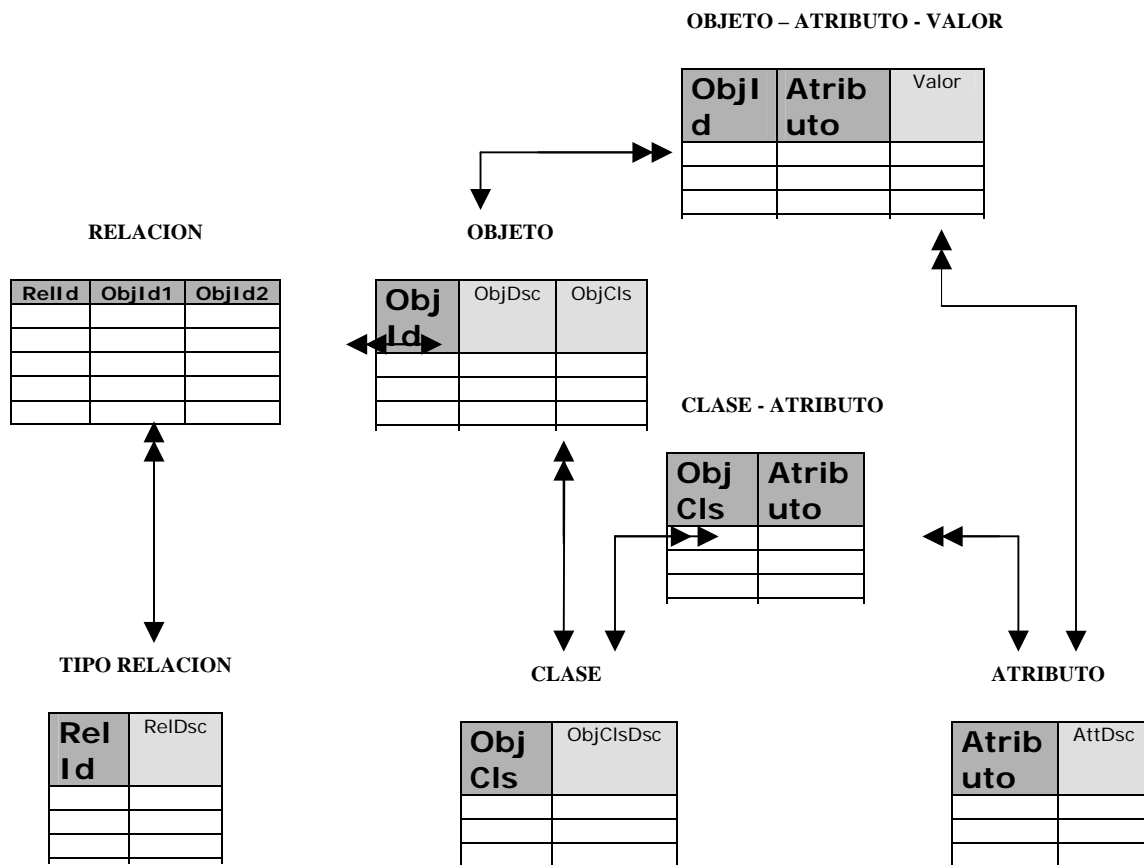
A partir de [1], se hace una interpretación del diagrama conceptual de la figura R.2 (pág 5) obteniendo el correspondiente esquema relacional.

Cabe destacar que la figura R.2 al igual que el resto de los diagramas que aparecen en el libro [1] usan la notación CASE\*Method [13] para representar modelos conceptuales. A los efectos de poder realizar la traducción al esquema relacional se explican brevemente los símbolos de la notación CASE\*Method que aparecen en el diagrama conceptual.

En la figura se aprecian los siguientes símbolos:

- Recuadros con bordes redondeados: estos son equivalentes a las entidades del MER, el texto que aparece dentro del recuadro se traduce en el nombre de la entidad en el MER.
- Líneas en parte punteadas que terminan en 3 puntas y conectan los recuadros: estas son equivalentes a relaciones "1-N" del MER donde el recuadro que queda del lado de las 3 puntas es el que participa "N veces" de la relación.

El resultado es el siguiente esquema relacional:



La notación usada para identificar las columnas de la clave primaria para cada tabla son en formato negrita y en fondo gris oscuro.

Estudiando el esquema, se observa que tiene las siguientes limitaciones:

- Si bien este esquema permite almacenar relaciones entre Objetos y a partir de estas relaciones se podría deducir como se relacionan las Clases de Objetos, la idea es poder modelar que Clases de Objetos intervienen en una relación sin pasar por un proceso de deducción a partir de los Objetos relacionados. Esto último implicaría tener de antemano la información ingresada sobre que Objetos se relacionan con que Objetos.
- No se sabe cuales son los atributos de las Clases que participan en una relación.
- No representa relaciones entre 3 o mas entidades.
- No representa el concepto de clave de una entidad.

Estas limitaciones son temas de estudio en las siguientes etapas de investigación.

### **3.1.2 Etapa 4**

En esta etapa de investigación del modelo se genera la versión 4.1 del modelo que satisface el requerimiento inicial propuesto: Lograr un modelo UDM equivalente al modelo conceptual MER[2]. (Esta equivalencia queda demostrada en *Representación de un esquema ER en la versión 4.1 de UDM* del informe principal)

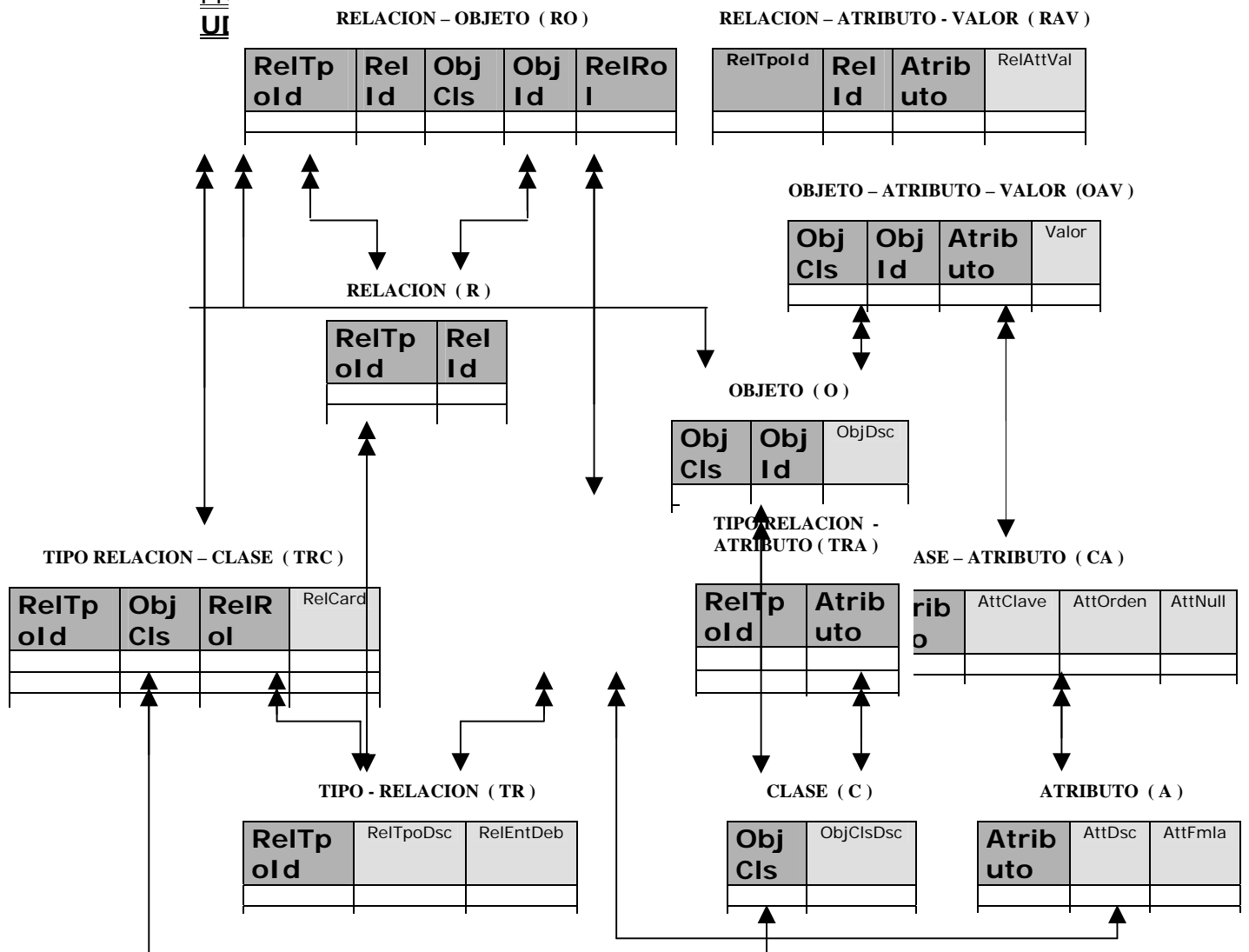
Esta versión de UDM se caracteriza por representar de forma natural los diferentes constructores del modelo conceptual MER[2]. Para lograr esto se decide modificar y extender lo que sea necesario en la estructura del modelo de la versión 1.1. El diagrama es el siguiente:

---

**Figura R.7.**

Prototipo

**UI**



Esta versión del modelo introduce conceptos propios de almacenamiento en lo referente a representación de relaciones entre entidades. Para cada Tipo de Relación en la tabla TIPO RELACION – CLASE (TRC) se especifica las dos o mas Clases que participan en esa relación. Cada una de estas se representa con un registro en esta tabla, habiendo tantos registros para el tipo de relación, como Clases de Objetos en la relación a representar. En la tabla RELACION (R) se almacenan las instancias de relación.

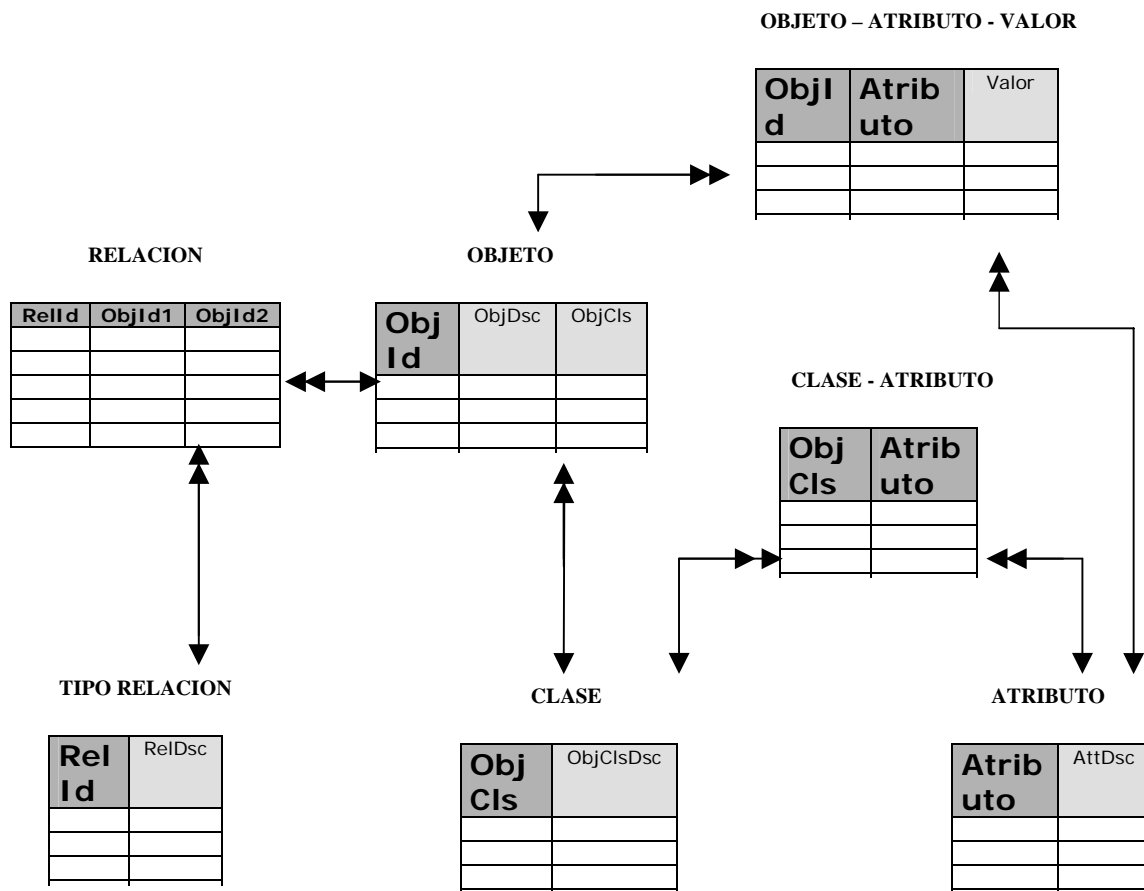
Para cada una de estas existen tantos Objetos participantes (especificado en RELACION – OBJETO (RO)) como Clases en el Tipo de Relación.

### 3.1.3 Etapa 5

En la sección 5.1 *Análisis de versión de UDM a usar* (pág. 24) en la aplicación, se concluye que la Línea 1 de evolución es la que mas se acerca a los requerimientos de la aplicación.

Con el objetivo de terminar la línea 1 de evolución del modelo, surge la etapa 5 de investigación en la cual se genera la versión 5.1 que es una evolución de la versión 2.1.

El diagrama de la versión 2.1 es el siguiente:



A continuación se detallan los cambios que incorpora la versión 5.1:

Se agrega una restricción de integridad referencial que evita que se le asigne un Valor a un Atributo de una Clase que no pertenece a la misma. Por ejemplo el modelo versión 2.1 permitía que se le asigne el Valor 10 al Atributo Importe de la Clase cliente, donde claramente el Atributo importe no pertenece a la Clase Cliente.

La versión actual, especifica la cardinalidad para una relación en la columna RelTpoCard de la tabla TIPO-RELACION.

En este modelo no se especifica cuales atributos son los que participan en una relación (como sí se hacía en la versión 2.1 anterior en la *Línea 1 de*

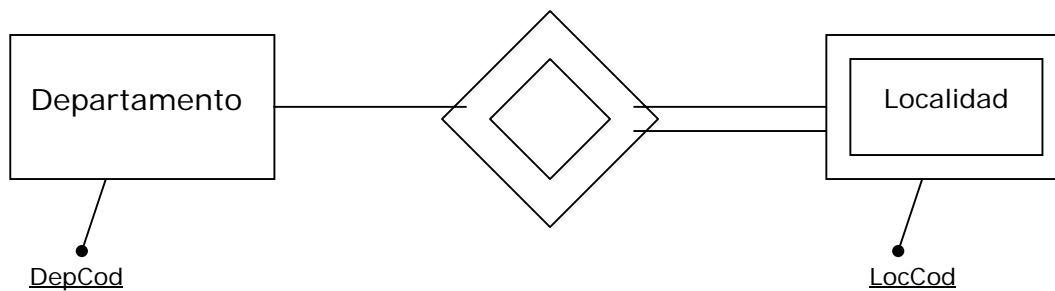
*Evolución*). Debido a que esto no es necesario ya que alcanza con especificar las relaciones entre Clases de objetos.

La explicación es la siguiente: Al saber que un objeto está relacionado con otro objeto, siempre será por su clave primaria y no es necesario especificarlo en la Clase origen de la relación. Esto lleva a que tampoco se incluyan valores para estos Atributos en la tabla OBJETO – ATRIBUTO – VALOR para la Clase origen, que si se incluyera, sería redundante debido a que esto mismo ya estaría reflejado en la tabla RELACION.

Otra mejora que tiene la versión 5.1 respecto a la versión 2.1 es la capacidad para poder representar Entidades Débiles.

Para esto se agrega la columna *RelTpold* en la tabla CLASE – ATRIBUTO. Para representar una entidad débil se suma la clave de la Clase padre a la Clase que representa la entidad débil y se marca en la columna *RelTpold* de la tabla CLASE-ATRIBUTO, con la identificación de la relación que lo relaciona con la Clase padre.

Para explicar esto se muestra el siguiente ejemplo de Departamento - Localidad:

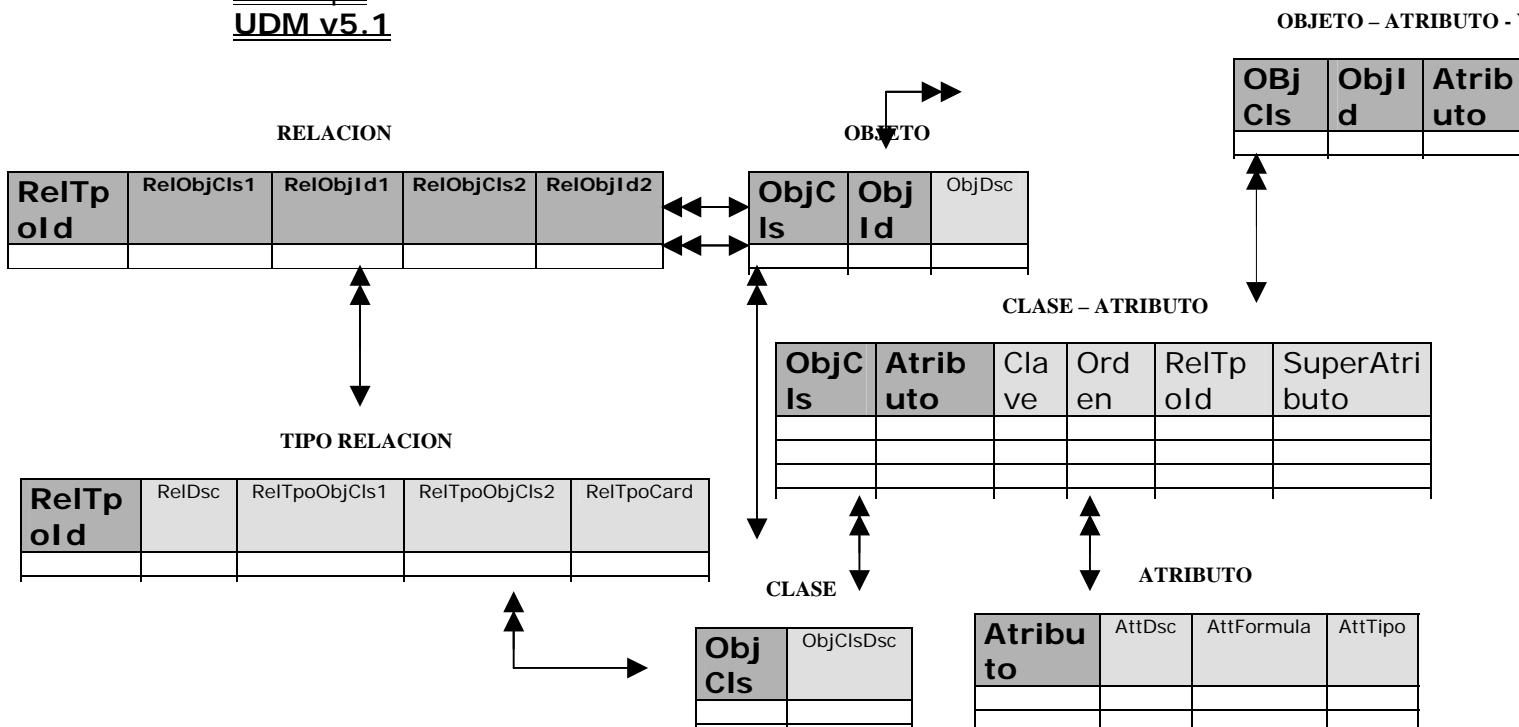


La tabla CLASE – ATRIBUTO para representar esta realidad es la siguiente:

ObjCls	Atributo	Clave	Orden	AtrNull	RelTpold
Departamento	DepCod	P	1	N	
Localidad	DepCod	P	1	N	DepartLocal
Localidad	LocCod	P	2	N	

El diagrama de la versión 5.1 es el siguiente:

**Figura R.8.**  
**Prototipo**  
**UDM v5.1**



En esta versión solamente se puede modelar directamente relaciones con cardinalidad 1 a 1 y relaciones con cardinalidad 1 a N entre clases. Para modelar relaciones M a N deben crearse clases ficticias. Las relaciones entre clases son almacenadas en la tabla TIPO RELACION, en donde se especifica cuales son las clases que participan en la relación en forma ordenada.

La columna *Clave* de la tabla CLASE – ATRIBUTO puede estar vacío o contener una “P”. En este último caso se está especificando que el Atributo pertenece a la clave primaria de la Clase. Cuando un Atributo pertenece a la clave de una Clase, se especifica en la columna *Orden*, el orden del Atributo en la composición de la clave.

Hasta ahora se han presentado las mejoras que introduce la versión 5.1 con respecto a su predecesora la versión 2.1 en lo referente a características de modelado. A continuación se presenta una extensión del modelo que le permite representar “Atributos Subtipos”.

Un “Atributo Subtipo” es un atributo que semánticamente es igual a otro atributo (“Atributo Supertipo”) a pesar de poder llamarse diferente.

El valor que toma el Atributo Subtipo es el valor que toma el correspondiente “Atributo Supertipo”. Esto lleva a la necesidad de una columna mas en la tabla CLASE-ATRIBUTO: la columna SuperAtributo.

Un Atributo es Subtipo cuando está cargada la columna *SuperAtributo* de la tabla CLASE – ATRIBUTO, en donde se especifica cual es el Atributo Supertipo en la Clase destino.

Vale remarcar que esta extensión no tiene como objetivo cubrir una carencia en un aspecto de potencia del modelo. Por el contrario, es una



facilidad que brinda el modelo para representar de forma cómoda una situación que se da frecuentemente en el diseño de formularios.

A continuación se presenta un ejemplo clásico Factura-Cliente para la versión 5.1. Aquí se muestran los conceptos vertidos en esta versión junto con el concepto "Atributo Subtipo".

La tabla CLASE-ATRIBUTO para un ejemplo Factura-Cliente es la siguiente

CLASE- ATRIBUTO						
ObjCls	Atributo	Clave	Orden	AtrNull	RelTpo Id	SuperAtributo
Cliente	ClienteCodigo	P	1	N		
Cliente	ClienteNombre			S		
Cliente	ClienteDireccion			S		
Vendedor	VendedorCodigo	P	1	N		
Vendedor	VendedorNombre			S		
Factura	FacturaNumero	P	1	N		
Factura	FacturaFecha			S		
Factura	FacturaCliCod			N	FacturaCliente1	ClienteCodigo
Factura	FacturaCliNom			S	FacturaCliente1	ClienteNombre
Factura	FacturaVendCod			N	FacturaVendedor1	VendedorCodigo
Factura	FacturaVendNom			S	FacturaVendedor1	VendedorNombre

En el ejemplo se tienen tres Clases: Cliente, Vendedor y Factura con sus respectivos Atributos claves especificados con una "P". Luego para la Clase Factura se dice que los "Atributos Subtipos" FacturaCliCod y FacturaCliNom están participando en la relación FacturaCliente1 especificada en la columna *RelTpoId*. Para cada uno de los "Atributos Subtipos" se especifica en la columna *SuperAtributo* cual es el Atributo de la Clase destino que lo relaciona. En este caso la Clase destino sería "Cliente", información que se extrae de la tabla TIPO – RELACION. Además la Clase Factura participa en otra relación determinada por FacturaVendedor1 con un comportamiento similar. Algunos de los registros de la tabla OBJETO – ATRIBUTO – VALOR y de la tabla RELACION para el ejemplo serían los siguientes:

OBJETO – ATRIBUTO - VALOR			
ObjCls	ObjId	Atributo	Valor
Cliente	Cliente1	ClienteCodigo	CLI01
Cliente	Cliente1	ClienteNombre	Juan
Cliente	Cliente1	ClienteDireccion	Minas 999
Vendedor	Vendedor1	VendedorCodigo	VEN01
Vendedor	Vendedor1	VendedorNombre	Pedro
Factura	Factura1	FacturaNumero	FAC01
Factura	Factura1	FacturaFecha	01/01/04

RELACION				
RelTpoId	RelObjCls1	RelObjId1	RelObjCls2	RelObjId2
FacturaCliente	Factura	Factura1	Cliente	Cliente1
FacturaVendedor	Factura	Factura1	Vendedor	Vendedor1

TIPO - RELACION

RelTpoId	RelDsc	RelTpoObjCls1	RelTpoObjCls2	RelTpoCard
FacturaCliente	FacturaCli	Factura	Cliente	N1
FacturaVendedor	FacturaVend	Factura	Vendedor	N1

Se observa que no hay registros en la tabla OBJETO – ATRIBUTO – VALOR para los Atributos FacturaCliCod, FacturaCliNom, FacturaVendCod, FacturaVendNom de la Clase Factura pues son subtipos. Estos harían referencia al Cliente y al Vendedor que estaría relacionado con el objeto Factura1 y esto se representa en la tabla RELACION. Esto evita la redundancia mencionada anteriormente.

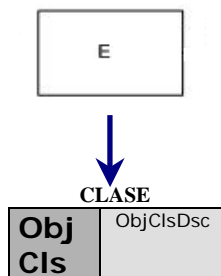
### 3.2 Representación de un esquema Entidad Relación en UDM versión 5.1

#### Introducción

En esta sección se muestra un mapeo entre un esquema ER y el Universal Data Model (UDM) versión 5.1 resultado de la etapa 5 de la evolución del modelo. Para ello se toma cada constructor de diagrama Entidad Relación y se muestra como se mapea en UDM. Por último se presenta un algoritmo, que toma un MER como entrada, y se define los pasos a seguir para hacer el mapeo completo del esquema ER a UDM.

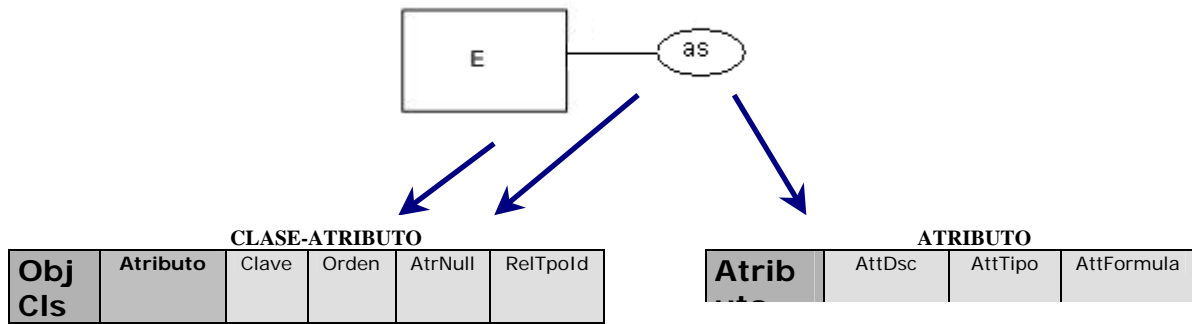
#### ENTIDAD

Cada entidad  $E$  se representa como una clase de objeto  $c$ , es decir se inserta una tupla en la tabla CLASE DEL OBJETO.



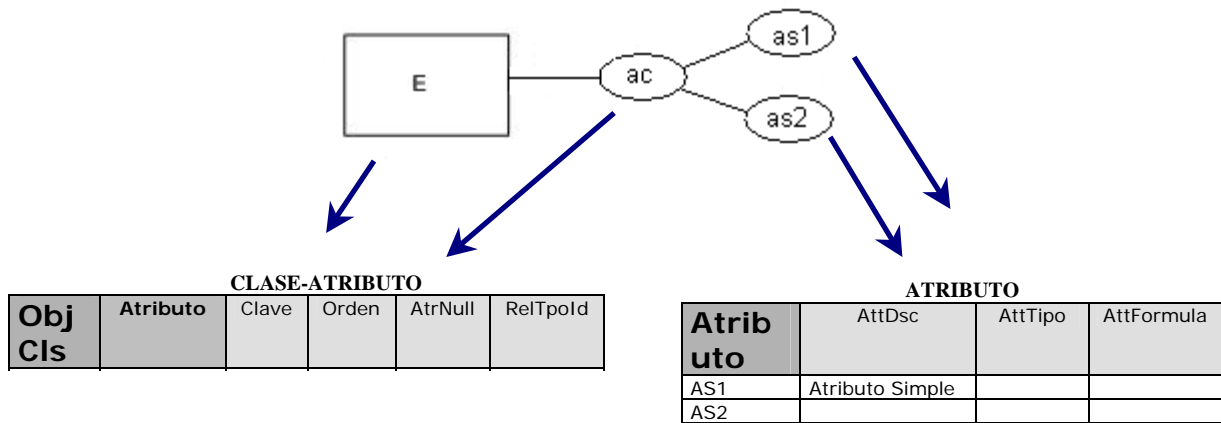
#### ATRIBUTO Simple

Cada atributo simple  $AS$  de una entidad  $E$  se representa como un atributo  $a$  (se inserta una tupla en la tabla ATRIBUTO cargando  $a$  en la columna *Atributo*, si ya existe no se hace nada). Para relacionar el atributo  $a$  con la clase de objeto  $c$  correspondiente con la entidad  $E$  se inserta una tupla en la tabla CLASE-ATRIBUTO con los valores  $c$  en la columna *ObjCls* y el valor  $a$  en la columna *Atributo*.



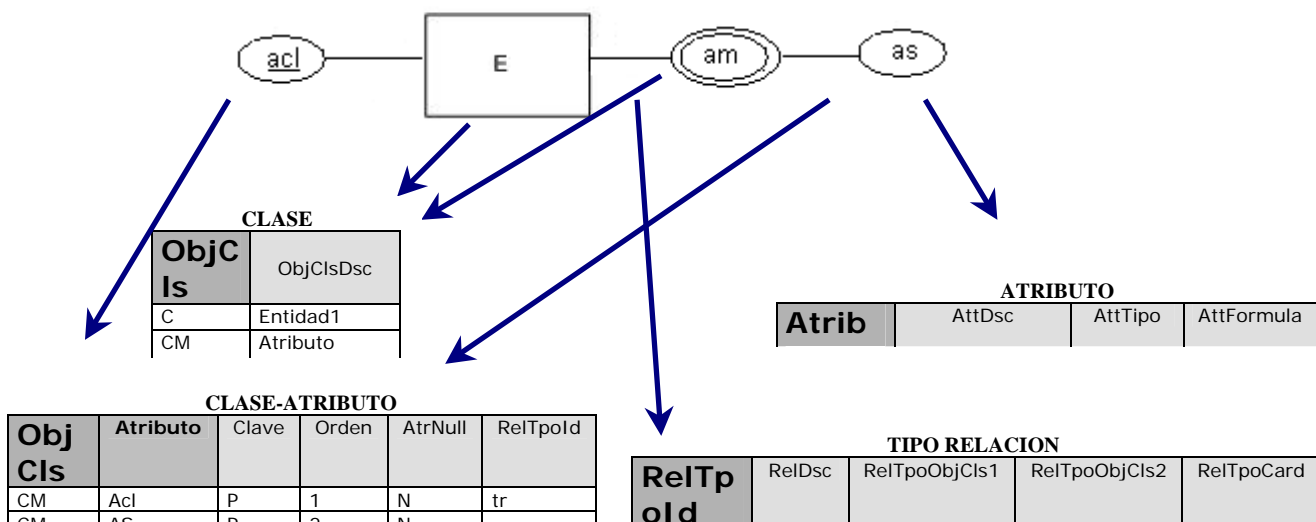
### ATRIBUTO COMPUESTO

Se representa cada uno de los atributos simples *AS* que componen el atributo compuesto *AC* de la entidad *E* como atributos simples *a*. Se inserta en la tabla *ATRIBUTO* con el valor *as* en la columna *Atributo* para cada atributo simple *AS*. Se inserta en la tabla *CLASE-ATRIBUTO* cargando *c* en la columna *ObjCls* y el valor de cada *as* en la columna *Atributo*.



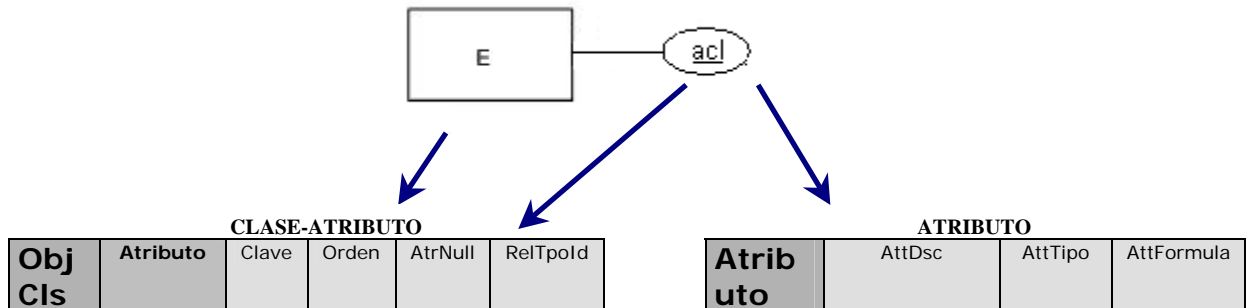
### ATRIBUTO MULTIVALUADO

Cada atributo multivaluado *AM* de una entidad *E* se representa como una clase de objeto *cm*. Todos los atributos simples de *AM* son los atributos de *cm*. La clave de la clase de objeto *c* correspondiente al mapeo de la entidad *E* también forma parte de la clase *cm*. Todos los atributos de *cm* son claves. Se tiene una relación *tr* entre la clase *cm* y la clase *c* correspondiente al mapeo de la entidad *E*.



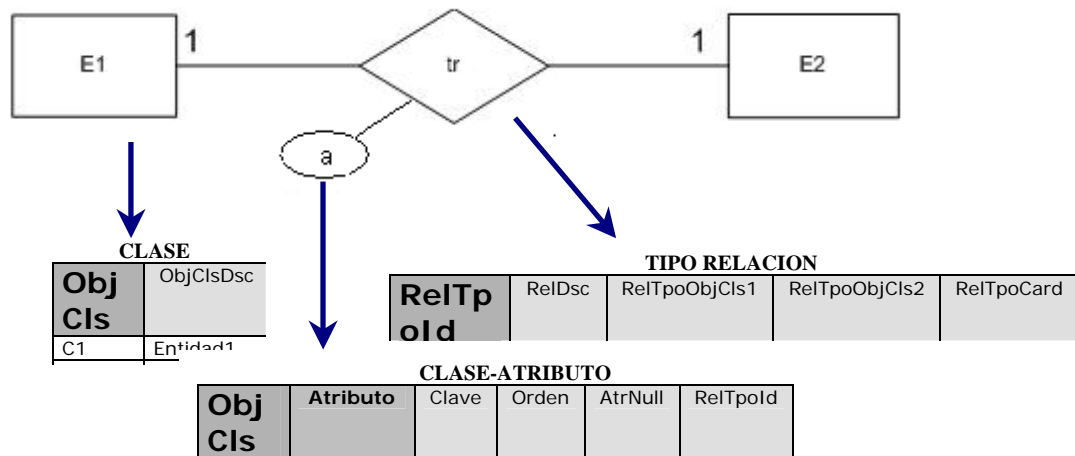
### ATRIBUTO CLAVE

Un atributo clave *ACL* se representa igual que un atributo *acl* ya sea simple o compuesto. Para decir que forma parte de la clave, se carga en la columna *Clave* de la tabla ATRIBUTO con el valor "P". También se carga la columna *Orden* para indicar el orden que ocupa *acl* en la clave.



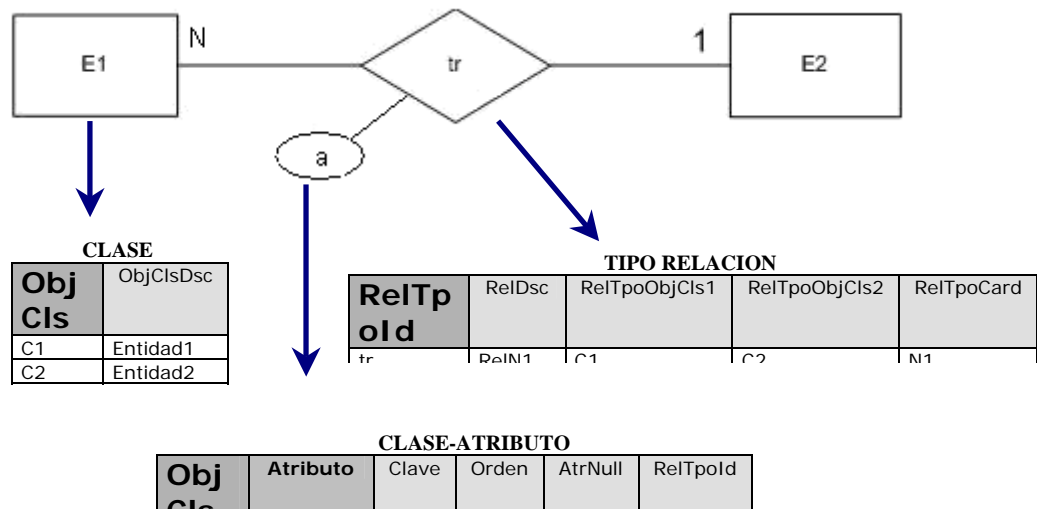
### RELACION 1:1

Una relación *R* de tipo 1:1 entre las entidades *E1* y *E2* se representa como un TIPO DE RELACION *tr*, esto es se inserta una tupla en TIPO-RELACION con *R*. En las columnas *RelTpoObjCls1* y *RelTpoObjCls2* se cargan los valores *c1* y *c2* que se corresponden a las clases, mapeo de las entidades *E1* y *E2* respectivamente. En la columna *RelTpoCard* se almacena el valor '11'. Los atributos *a* de *R* se almacenan como parte de la clase *c1*.



### RELACION 1:N

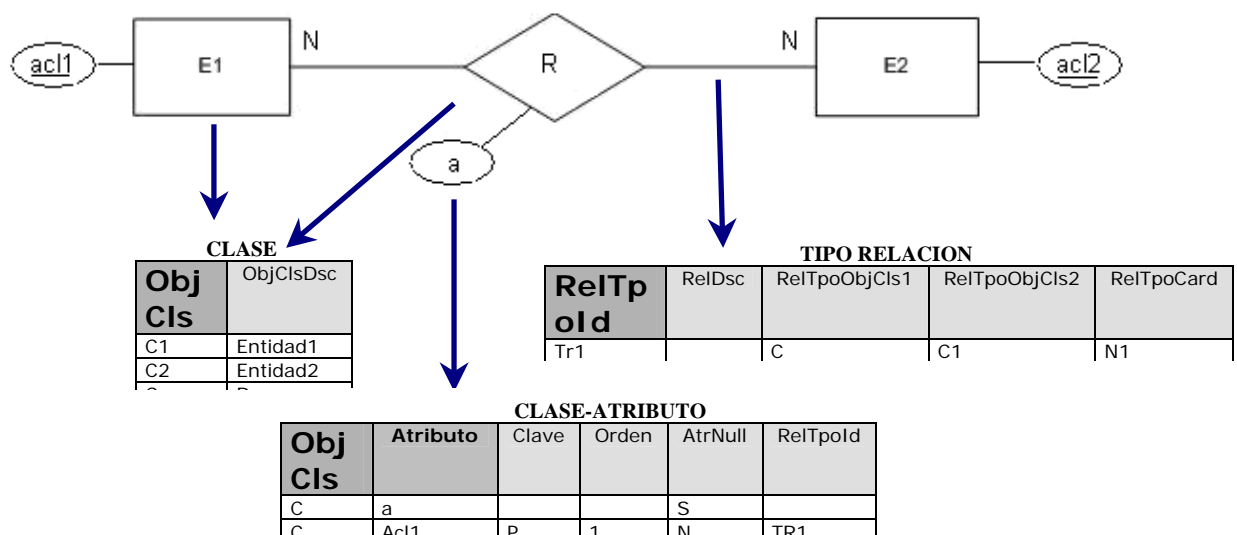
Una relación *R* de tipo 1:N entre las entidades *E1* y *E2* se representa como un TIPO DE RELACION *tr*, esto es se inserta una tupla en TIPO-RELACION con *R*. En las columnas *RelTpoObjCls1* y *RelTpoObjCls2* se cargan los valores *c1* y *c2* que se corresponden a las clases, mapeo de las entidades *E1* y *E2* respectivamente. En la columna *RelTpoCard* se almacena el valor 'N1'. Los atributos *a* de *R* se almacenan como parte de la clase *c1*.



### RELACION M:N

Una relación  $R$  de tipo M:N entre las entidades  $E1$  y  $E2$  se representa como una CLASE DE OBJETO  $c$ . Se generan relaciones N:1 entre  $c$  y las clases de objeto  $c_1$  y  $c_2$  correspondiente al mapeo de las entidades  $E1$  y  $E2$  respectivamente. Esto es se inserta una tupla en TIPO-RELACION con el valor  $tr1$  en la columna  $RelTpold$ , el valor  $c$  en la columna  $RelTpoObjCls1$ , el valor  $c_1$  en la columna  $RelTpoObjcls2$  y el valor N1 en la columna  $RelTpoCard$ .

Si la relación  $R$  tiene atributos  $a$ , se insertan en ATRIBUTO y en CLASE ATRIBUTO para la clase  $c$ . La clave de  $c$  esta formada por la clave de  $c_1$  mas la clave de  $c_2$ .

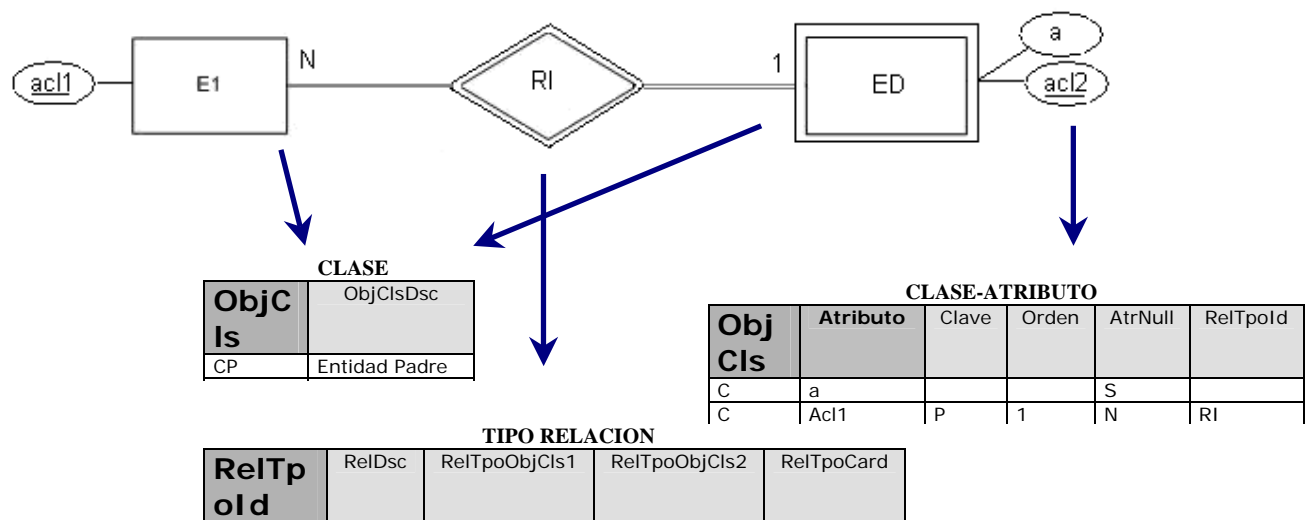


### ENTIDAD DEBIL

Cada entidad débil  $ED$  no puede ser representada por si misma sino que precisa una relación identificador  $RI$  con su dueña para identificarse. Se

trata la entidad débil con su relación identificadora juntas. La entidad débil se representa como una entidad común. Si se tiene una entidad débil *EB* se genera como una clase de objeto *c*. Para representar la relación identificador *RI* se genera como cualquier otra relación 1:N. Se genera una tupla en TIPO RELACION con el valor *ri* en la columna *RelTpold*, el valor *c* en la columna *RelTpoObjCls1* y el valor *cp* en la columna *RelTpoObjCls2*, la clase de objeto correspondiente a la clase padre.

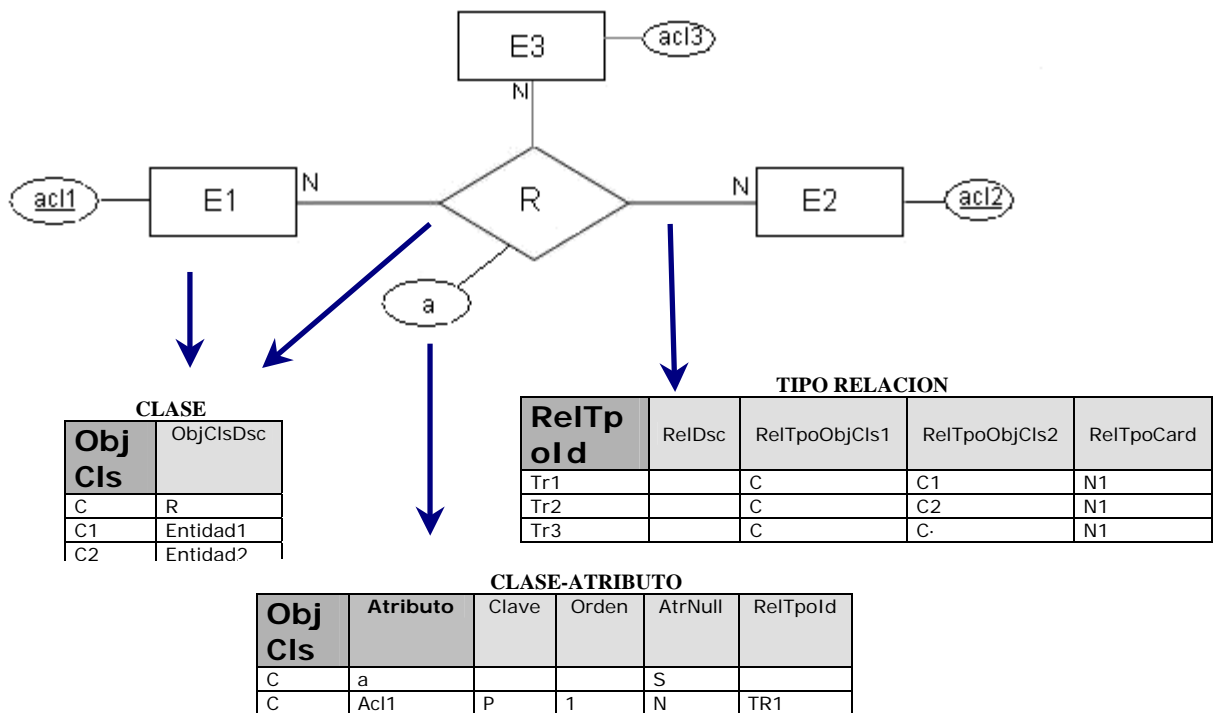
Para representar que la entidad es débil se suma la clave de la clase *cp* a la clase *c* en la tabla CLASE-ATRIBUTO. Se inserta una tupla en CLASE-ATRIBUTO con el valor *c* en la columna *ObjCls*, el valor *acl1* en la columna *atributo*, el valor 'P' en la columna *clave* y el valor *ri* en la columna *RelTpold*.



### RELACION N-aria

Una relación *R* de tipo N-aria entre *n* entidades *E1..En* se representa como una clase de objeto *c*, esto es se almacena una tupla en CLASE DE OBJETO con *c*. Para cada entidad *Ei* participante de la relación *R* se carga una tupla en TIPO RELACION con el valor *tri* en la columna *RelTpold*, con el valor *c* en la columna *RelTpoObjCls1* y en la columna *RelTpoObjcls2* se carga el valor *ci*, la clase correspondiente a *Ei*.

Si la relación *R* tiene atributos *a*, se insertan en la tabla ATRIBUTO y en la tabla CLASE ATRIBUTO para la clase *c*. Los atributos claves de las clases *ci* se agregan como atributos claves de la clase *c*.



### 3.2.1 Algoritmo para el mapeo de un esquema ER a UDM

A continuación se presenta un algoritmo que muestra los pasos a seguir para hacer el mapeo completo de un esquema Entidad Relación al modelo UDM. El orden es incremental en complejidad, procesando primero las entidades y las relaciones; y luego los constructores más complejos como entidad débil, atributos multivaluados y relaciones N-arias.

**Paso 1:** para cada entidad  $E$  del esquema se crea una clase de objeto  $c$ . Se toman todos los atributos simples y compuestos de  $E$  y se crean los atributos en el UDM. Se relacionan los atributos creados con  $c$ , insertando los atributos en la tabla CLASE-ATRIBUTO. Los atributos clave de  $E$  se cargan con el valor "P" en la columna *Clave* en la tabla CLASE-ATRIBUTO.

**Paso 2:** Para cada relación  $R$  de tipo 1:1 del esquema, se identifican las clases de objeto  $c$  y  $d$  que corresponden a las entidades  $S$  y  $T$  participantes de la relación  $R$ . Se almacena la relación  $R$  en TIPO RELACION. Se almacenan  $c$  y  $d$  como participante de la relación. Si la relación tiene atributos  $a$  se almacenan en ATRIBUTO y se insertan en CLASE ATRIBUTO como parte de la clase  $c$ .

**Paso 3:** Para cada relación  $R$  de tipo 1:N del esquema, se identifican las clases de objeto  $c$  y  $d$  que corresponden a las entidades  $s$  y  $t$  participantes de la relación. Se almacena la relación  $R$  en TIPO RELACION. Se almacenan  $c$  y  $d$  como participante de la relación, se almacena el valor N1 en *RelTpoCard*. En *ReltpoObjcls1* se carga la clase que participa del lado N. Si la relación tiene atributos, se insertan en ATRIBUTO y en CLASE ATRIBUTO como parte de la clase de objeto que participa del lado N.

**Paso 4:** Para cada relación  $R$  N:M del esquema se crea una CLASE de OBJETO  $cr$ . Se identifican las clases de objeto  $c$  y  $d$  que se corresponden a las entidades  $S$  y  $T$  participantes de la relación  $R$ . Se crea la relación  $tr1$  en TIPO RELACION entre  $cr$  y  $c$  con la columna *RelTpoCard* cargado con el valor N1. Se crea la relación  $tr2$  en TIPO RELACION entre  $cr$  y  $d$  con la columna *RelTpoCard* cargado con el valor N1. Si la relación tiene atributos, se almacenan en ATRIBUTO y se insertan en CLASE ATRIBUTO como parte de la clase  $cr$ .

**Paso 5:** Para cada entidad débil  $EB$  del esquema con entidad dueña  $D$  se crea una clase de objeto  $c$ . Se toman todos los atributos simples y compuestos y se crean los atributos  $a$  en el UDM. Se relacionan estos atributos  $a$  con la clase  $c$ . Se crea la relación identificador  $r$  en TIPO RELACION. En *ReltpoObjcls1* se almacena la clase  $c$  y en *ReltpoObjcls2* se almacena  $d$  correspondiente a  $D$  como participantes de la relación. La clave de  $d$  es cargada como parte de  $c$  y forma parte de la clave de la clase  $c$ .

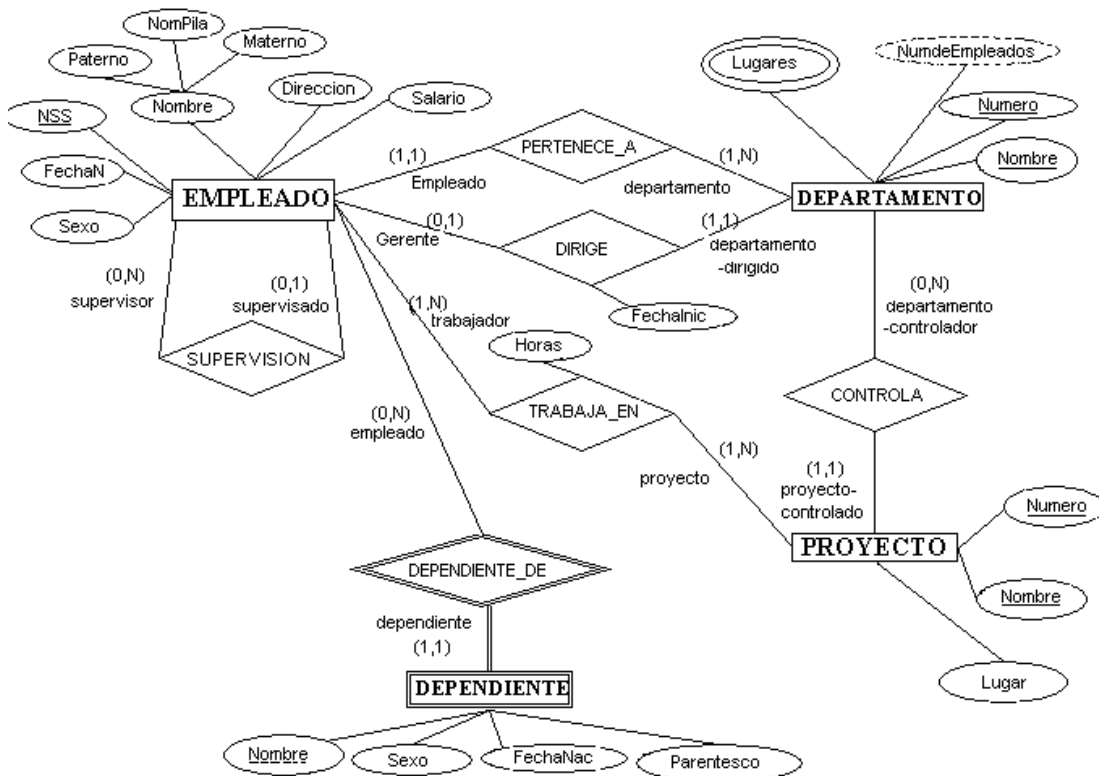
**Paso 6:** para cada atributo multivaluado  $AM$  de la entidad  $E$  se crea una nueva clase de objeto  $c$ . Se insertan los atributos simples y compuestos de  $AM$  en ATRIBUTO. Se relacionan los atributos con  $c$  insertando en CLASE ATRIBUTO. Se crea una relación entre  $c$  y  $ce$ , la clase de objeto que se mapea con  $E$ . Se suman los atributos clave de  $ce$  como parte de  $c$ . Se marca todos los atributos de  $c$  como claves.



**Figura R.9. Paso 7:** Para cada relación n-aria  $R$ , se crea una nueva Diagrama clase de objeto  $c$ . Para cada clase participante  $c_i$  de la Entidad relación  $R$  se crea una nueva tipo de relación  $tr_i$ , Relación de cargando la columna  $ReltpoObjcls1$  con el valor  $c$ ,  $ReltpoObjcls2$  con  $c_i$  y con la columna  $ReltpoCard$  cargado con el valor N1. Si  $R$  tiene atributos, se almacenan en ATRIBUTO y se cargan en CLASE ATRIBUTO como parte de  $c$ . Las claves de las entidades participantes se cargan como parte de  $c$  y forman la clave de  $c$ .

### 3.2.2 Ejemplo de uso del algoritmo

Para mostrar el uso del algoritmo se presenta como ejemplo el usado en [2] El ejemplo muestra la realidad de una compañía donde diagrama Entidad Relación es el siguiente:



Con la aplicación del algoritmo se crean las Clases de Objeto empleado, departamento y proyecto como mapeo de las entidades correspondientes. Se crean los atributos Nombre, NSS, dirección, NumeroD, etc. Se asocian estos atributos a las Clases que pertenecen, por ejemplo Numero es un atributo de Departamento, dirección es un atributo de Empleado. Los atributos Numero de Departamento, NSS de Empleado y Numero del Proyecto se marcan como clave cargando "P" en la columna Clave de la tabla CLASE-ATRIBUTO. Para mapear la relación Dirige se inserta en la tabla TIPO-RELACION una tupla con Dirige como identificador de la relación y las clases participantes Empleado y Departamento. El atributo FechaNac de la relación se da de alta en la tabla ATRIBUTO y como parte de Empleado en la tabla CLASE-ATRIBUTO.

Las tablas se ven de la siguiente forma después de aplicado el algoritmo

**CLASE**

<b>ObjCls</b>	<b>ObjClsDsc</b>
DEPARTAMENTO	DEPARTAMENTO
EMPLEADO	EMPLEADO
PROYECTO	PROYECTO
DEPENDIENTE	DEPENDIENTE
...	

**ATRIBUTO**

<b>Atributo</b>	<b>AttDsc</b>	<b>AttTipo</b>	<b>AttFormula</b>
NSS	NSS		
Dirección	Dirección		
FechaN	FechaN		
NomPila	NomPila		
Salario	Salario		
Sexo	Sexo		
Nombre	Nombre		
Numero	Numero		
Lugar	Lugar		
FechaInic	FechaInic		
Parentesco	Parentesco		

**CLASE- ATRIBUTO**

<b>ObjCls</b>	<b>Atributo</b>	<b>Clave</b>	<b>Orden</b>	<b>AtrNull</b>	<b>RelTpoId</b>
DEPARTAMENTO	Numero	P	1	N	
DEPARTAMENTO	Nombre			S	
EMPLEADO	NSS	P	1	N	
EMPLEADO	Direccion			S	
EMPLEADO	FechaN			S	
EMPLEADO	NomPila			S	
EMPLEADO	Salario			S	
PROYECTO	Numero	P	1	N	
PROYECTO	Lugar			S	
EMPLEADO	FechaInic			S	
DEPENDIENTE	NSS	P	1	N	DEPENDIENTE_DE
DEPENDIENTE	Nombre	P	2	N	
DEPENDIENTE	Parentesco			S	

**TIPO - RELACION**

<b>RelTpoId</b>	<b>RelDsc</b>	<b>RelTpoObjCls1</b>	<b>RelTpoObjCls2</b>	<b>RelTpoCard</b>
DIRIGE	DIRIGE	EMPLEADO	DEPARTAMENTO	11
CONTROLA	CONTROLA	PROYECTO	DEPARTAMENTO	N1
DEPENDIENTE_DE		DEPENDIENTE	EMPLEADO	N1

## 4 Estudio de performance del modelo UDM

El hecho de que UDM se base en un conjunto fijo de tablas relacionales para representar cualquier realidad abre una gran interrogante en cuanto a la eficiencia (performance) de las operaciones de acceso a datos sobre el modelo. Esta inquietud motiva un estudio de performance que tiene como objetivo central comparar sobre UDM y el modelo relacional, los tiempos de respuesta de algunas operaciones de acceso a datos que son de interés. Un objetivo secundario de este estudio es medir el espacio en disco que ocupan los datos en uno y otro modelo.

Para el estudio de performance se toma como metodología representar una misma realidad en el modelo UDM y en el modelo relacional. La realidad elegida es la de un sistema de facturación en el cual intervienen las clásicas entidades: factura, cliente y producto. Después se realiza una carga masiva de datos sobre ambos modelos para estar en condiciones de comparar tiempos de respuesta de ciertas operaciones en uno y otro modelo. El volumen de datos que se carga es: 1.000 clientes, 1.000 productos, 200.000 facturas y 800.000 líneas de factura.

En una primera medición de performance varias operaciones resultaron ineficientes debido a falta de índices. Para solucionar este problema se crearon los índices faltantes y se midieron los tiempos de las operaciones nuevamente.

El DBMS (DataBase Managment System) usado para las pruebas fue SQL Server 2000, ejecutando sobre un equipo Intel con 1GB de RAM y 2 procesadores.

A continuación se listan las operaciones y su tiempo de respuesta en segundos en uno y otro modelo:

<b>Operación</b>	<b>Relacional</b>	<b>UDM</b>
Realizar una carga masiva de datos	6900	26640
Seleccionar todas las tuplas de una tabla	16	76
Búsqueda por clave	1	1
Búsqueda por un atributo indexado	3	1
Búsqueda por un atributo no indexado	4	1
Búsqueda por condición AND	1	1
Búsqueda por condición OR	8	9
Join	1	1
Eliminar todas las tuplas de una tabla	19	560

En otro orden, la medición de espacio usado en disco se basa en medir el espacio ocupado en Megabytes por el conjunto de datos cargado por la operación de carga masiva de datos, en uno y otro modelo. El resultado de la medición arroja que la base relacional necesita 155 MB de espacio en disco para almacenar los datos cargados mientras que la base UDM necesita 1909 MB.

A modo de conclusión de este estudio resulta que la mayoría de las operaciones de acceso a datos sobre UDM se comportan de forma similar en el modelo relacional. En particular la operación "Join" que era de sumo interés resulta ser muy eficiente.

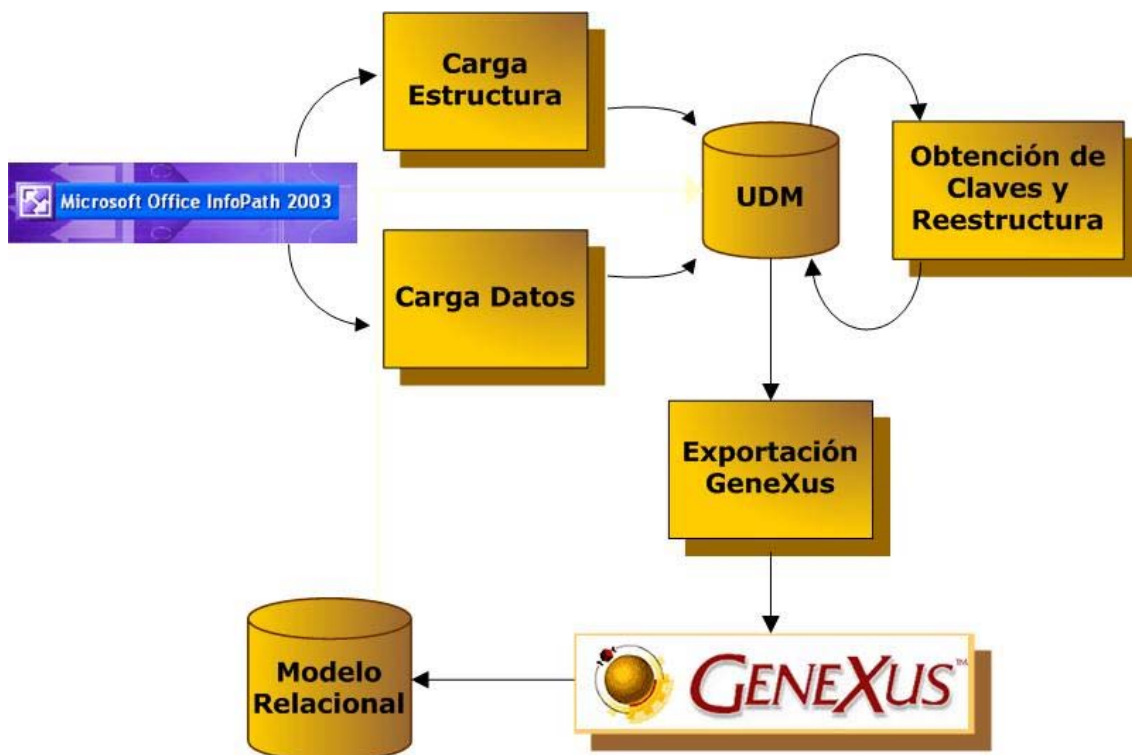
**Figura R.10.**  
Arquitectura

Por otro lado se tiene que el modelo UDM necesita mucho mas espacio en disco que el modelo relacional para almacenar un conjunto de datos semánticamente equivalente. De cualquier forma esta no es una característica altamente preocupante debido a que el precio del almacenamiento magnético es cada día mas bajo.

### 5 Aplicación "Model by Example"

La aplicación desarrollada en este proyecto aplica la técnica "Model by Example", esto es a partir de formularios diseñados y cargados por el usuario se crea una base de datos y se generan programas que permiten realizar altas, bajas, modificaciones y consultas sobre dicha base de datos.

La aplicación consiste en cargar la información de formularios producidos desde Microsoft InfoPath SP1 al modelo de datos UDM, trabajar sobre la información almacenada buscando claves, y luego exportarlo a GeneXus para que este se encargue de la creación de la base de datos y los programas. La aplicación tiene la siguiente arquitectura:



La forma de trabajar es: El usuario dibuja formularios en Microsoft Office InfoPath 2003 e ingresa datos de ejemplo en los mismos. El Módulo de *Carga Estructura* y *Carga Datos* carga la información de los formularios en el modelo UDM. El Módulo *Obtención de Claves y Reestructura*, infiere las

claves a partir de la información almacenada en el modelo UDM y luego se crea un archivo XML que sirve de interfase con GeneXus. GeneXus normaliza las estructuras de datos y crea el esquema relacional. Además genera los programas para realizar altas, bajas, modificaciones y consultas sobre la base relacional creada.

La aplicación consiste en los siguientes módulos:

### ***Carga de la Estructura de Datos del Formulario***

Este módulo tiene como objetivo el de cargar la información relativa a la estructura de los formularios infiriendo aspectos de metadata desde los formularios. Esta información es almacenada en las tablas del modelo UDM que cargan aspectos de metadata.

### ***Carga de los Datos Ingresados en el Formulario***

Este módulo carga los datos ingresados por el usuario en los formularios. Los formularios rellenos corresponden a archivos XML. La información es almacenada en las tablas de datos del modelo UDM. Es importante que el usuario cargue datos para que el siguiente módulo pueda trabajar correctamente.

### ***Obtención de Clave Primaria y Reestructura***

Este módulo se encarga de inferir claves a partir de los datos ingresados por el usuario. También reorganiza la estructura de los formularios para que se parezca a las transacciones de GeneXus.

### ***Exportación a GeneXus***

El módulo de exportación lee la información almacenada en el modelo UDM y crea un archivo XML para la consolidación en la herramienta GeneXus.

## ***5.1 Análisis de versión de UDM a usar***

Esta sección se refiere a la elección de la versión de Universal Data Model mas adecuada para usar en la aplicación "Model by Example". Esta elección se basa fundamentalmente en la facilidad de cada versión de UDM para representar las estructuras de los formularios. Por tanto el primer paso en este camino consiste en hacer un análisis de las estructuras que se presentan en los formularios.

El análisis de formularios estándar arroja que existe en general una o varias secciones (o áreas) dentro de un formulario, donde sección es el espacio definido en el formulario en donde se agrupa un conjunto de campos. Puede suceder también que exista una construcción anidada de secciones.

---

Tomando como raíz el propio formulario, dentro de este puede haber campos y/ o secciones. A su vez, dentro de cada sección puede haber otras secciones y/o campos. Se aprecia que las relaciones que existen entre las secciones pueden llegar a ser de tipo 1:1 o de tipo 1:N. Esto es debido a que una sección puede contener muchas instancias de una sección hija, pero la sección hija pertenece a una única sección padre.

Entonces la estructura de un formulario queda definida por las secciones, los campos y las relaciones que se dan entre secciones. La forma en que se representaría la estructura de un formulario en UDM sería: representar las secciones como Clases, los campos como Atributos y las relaciones entre secciones como Relaciones entre Clases.

Las secciones y los campos se representarían de forma similar en cualquiera de las versiones de UDM con lo cual la representación de estos elementos no es un factor de peso a los efectos de inclinarse por determinada versión de UDM. Si es importante para la elección, la representación de las relaciones entre secciones. Una característica que tienen las relaciones entre secciones del formulario, es que las mismas nunca tienen cardinalidad N:M. Tampoco se presentan relaciones n-arias entre secciones.

De lo anterior se deduce que la versión de UDM que modela de forma mas natural las estructuras de los formularios sería una versión de UDM que siga los lineamientos de la *Línea 1* de evolución. La restante posibilidad sería elegir una versión de UDM de la *Línea 3* de evolución. Esto último no se prefiere debido a que un modelo de la *Línea 3* de evolución podría representar directamente todo tipo de relaciones entre entidades (relaciones M:N, relaciones n-arias, etc) y esto no es necesario ya que redundaría en un modelo UDM subutilizado por la aplicación. Por el contrario un modelo de la *Línea 1* de evolución solo tendría la posibilidad de representar directamente relaciones 1:N y 1:1 que son ni mas ni menos el tipo de relaciones que se pueden dar entre secciones de formularios.

Esto origina la etapa 5 de investigación en donde se hacen algunos ajustes al último prototipo (versión 2.1) de la *Línea 1* de evolución.

## **5.2 Análisis del editor a usar**

La forma en que el usuario expresa el conocimiento de una realidad en la herramienta "Model by example" que se quiere desarrollar es a través de un editor que cumpla una serie de requerimientos que se listan a continuación.

El "Requerimiento 1" es la capacidad de representar estructuras de datos de la realidad. Esto es que el editor debe proveer una forma de expresar conceptos que se puedan traducir en Clases, Atributos y Relaciones entre Clases de UDM.

El "Requerimiento 2" es la capacidad de ingreso de datos de la realidad representada. Esto es que el editor debe proveer una forma de ingresar datos que se puedan traducir en Objeto, Valores de Atributos y Relaciones entre Objetos de UDM.

El "Requerimiento 3" es que la salida del editor sea legible e interpretable de forma programática. Esto tiene que ver con la posibilidad de acceder en

---

forma programática a la información referente a estructuras de datos de la realidad representada así como los datos de ejemplo ingresados por el usuario.

En base a estos requerimientos se estimó que habían 2 tipos de editores que podían satisfacerlos: Los editores de planillas electrónicas y los editores de formularios electrónicos. A continuación se presenta un análisis de estos 2 tipos de editores en cuanto al cumplimiento de los requerimientos establecidos.

Editores de planillas electrónicas: la planilla electrónica tiene un gran punto a favor en el sentido de que es una herramienta ampliamente difundida pero tiene algunos problemas para cumplir con el Requerimiento 1:

- No existe una forma clara de representar clases subordinadas (por ejemplo el caso "Factura" y "Línea de Factura")
- No existe una forma efectiva y sencilla de representar relaciones entre Clases

Editores de formularios electrónicos: si bien no es una herramienta tan difundida como la planilla electrónica, los formularios electrónicos satisfacen ampliamente los requerimientos 1 y 2. En cuanto al cumplimiento del requerimiento 3, no se puede establecer en forma general que todos los editores lo contemplen debido a que es una característica que depende de la implementación de cada editor de formularios. A los efectos de establecer si se cumple este requerimiento serán evaluados 3 editores de formularios: "Infopath" de la empresa Microsoft[4], "FormDocs"[11] de la empresa FormDocs LLC y "SpectraForms" [12] de la empresa Starpoint Software.

FormDocs: Es posible acceder a los datos del formulario a través de XML o también a través de las librerías "FormDocs Developer Kit". No es posible acceder a la estructura del Formulario.

SpectraForms: Es posible acceder a los datos del formulario a través de XML o también a través de las librerías haciendo uso del producto "SpectraForms ActiveX Control". No es posible acceder a la estructura del Formulario.

Infopath: Permite acceder a la estructura del formulario y a los datos a partir de archivos de la familia XML.

Del este análisis se deduce que Infopath es el editor elegido debido a que cumple los requerimientos planteados.

---

## 6 Conclusiones

### 6.1 Conclusiones Generales

A grandes rasgos el proyecto tuvo dos líneas de trabajo. La primera, puramente de investigación, en donde se hizo un estudio profundo del modelo original y de las distintas versiones en cuanto a su entendimiento, composición, potencialidad y performance. La segunda, de implementación, en donde se puso a prueba el modelo dentro de un escenario: "Model by Example".

En la fase de investigación teórica del modelo se obtienen las versiones 4.1 y 5.1 de UDM que tienen la potencia del MER. Se entiende que cualquiera de estas 2 versiones tiene la madurez teórica necesaria para considerar su uso. La versión 4.1 se caracteriza por la capacidad de representar realidades expresadas en un MER ya que la traducción de conceptos de un MER a UDM 4.1 es relativamente sencilla. En contraposición, la versión 5.1 se caracteriza por su naturalidad para representar esquemas relacionales y estructuras de formularios. Por esto último, la versión 5.1 fue el modelo utilizado para la aplicación "Model by Example" desarrollada.

Además de la evaluación de potencia, se realizó sobre la versión 5.1 una evaluación sobre su uso práctico y una evaluación de los tiempos de respuesta (performance) de las operaciones de acceso a datos sobre el modelo.

La evaluación práctica se basó en poner a prueba el modelo como modelo de datos dentro del diseño de la aplicación. Su utilización práctica en el escenario usado fue satisfactoria ya que respondió correctamente ante el alto dinamismo en cuanto a reorganización de estructuras de datos que representan distintas realidades. La evaluación de performance se basó en comparar tiempos de respuesta de ciertas operaciones de acceso a datos en el modelo relacional y en el modelo UDM. El estudio arrojó que la mayoría de las operaciones se comportan de forma similar en uno y otro modelo.

El modelo 5.1 permite además almacenar aspectos de prototipado de la aplicación.

Culmina el proyecto habiendo alcanzado los objetivos planteados, dentro de los cuales el principal consistió en el estudio y el alcance de un nuevo modelo de datos hasta ahora no incursionado. Esto último, fue debido a características consideradas como adversas de antemano (por ejemplo la performance) o la no existencia de escenarios para su aplicación. Se llega a la conclusión de que estas características prejuzgadas como adversas no lo son tanto y aún considerándolas como tales, existen muchos escenarios donde el modelo puede llegar a ser explotado. Se cree que, el Universal Data Model puede a ser utilizado en ambientes donde se procure la innovación ante todo, teniendo buenos resultados y mejores aún en el futuro, a la luz de las nuevas tecnologías que reducen aún mas la brecha de las condiciones adversas.

---



## 6.2 Conclusiones Particulares

En la propuesta del proyecto se definieron una serie de interrogantes a investigar en el proyecto. Estas tienen que ver con aspectos particulares de modelado y específicamente de evaluación del Universal Data Model frente a características de otros modelos de datos. Se citan literalmente las interrogantes que también pueden ser consultadas en la sección Objetivos del Capítulo de Introducción del presente informe.

1 - *Determinar si UDM permite representar las realidades que son representables a través de un MER.*

Se llegó a representar en UDM, todas las realidades que son representables a través de un MER. Queda demostrado en la sección "Representación de un esquema ER en UDM" la equivalencia del Universal Data Model con el Modelo Entidad Relación (MER), en donde se muestra cada constructor del MER y su mapeo a UDM. También se presenta un algoritmo que toma un MER como entrada, y se definen los pasos a seguir para hacer el mapeo completo del esquema ER a UDM.

*En particular es de interés investigar si el modelo UDM permite modelar:*

- *Atributos derivados*

En el UDM se incluye una columna en la tabla ATRIBUTO el cual especifica la fórmula para calcular el atributo derivado. En caso de estar vacío esta columna se trata de un atributo no derivado. Esta columna es "AttFmla" para la versión 4.1 de UDM y "AttFormula" para la versión 5.1. La inclusión de los valores que podría tomar esta columna para el Atributo de la Clase, depende de la implementación y estaría almacenado en la tabla OBJETO – ATRIBUTO – VALOR. En el caso de la aplicación desarrollada se considera la inclusión de fórmulas planas del tipo "Cantidad \* Precio" y no se almacenan estos datos en la tabla OBJETO – ATRIBUTO – VALOR por ser considerada redundante.

- *Atributos Multivaluados*

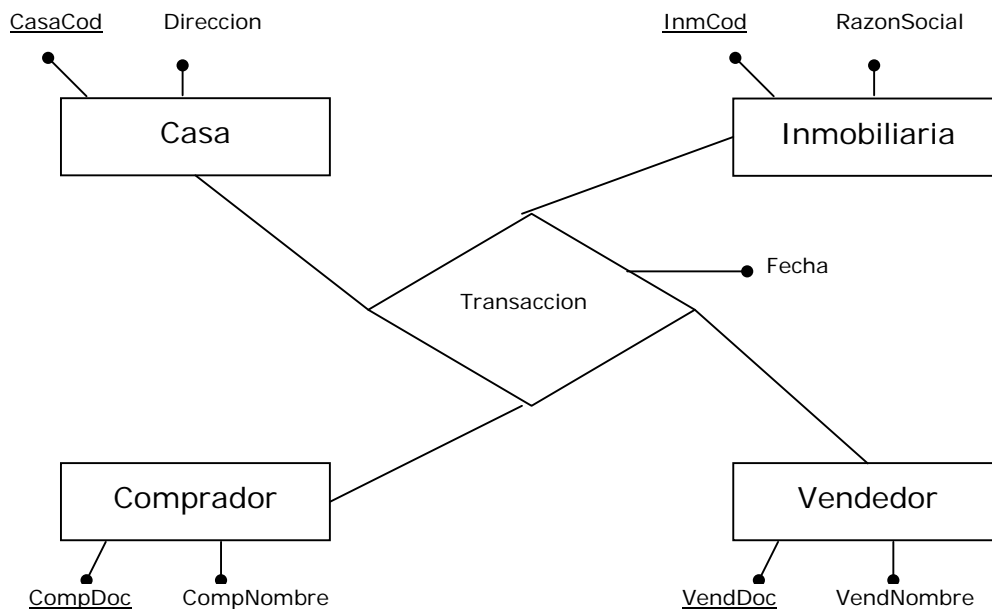
Cada atributo multivaluado *AM* de una entidad *E* de un modelo Entidad Relación se representa en UDM como una clase de objeto **cm**. Todos los atributos simples de *AM* son los atributos de **cm**. La clave de la clase de objeto **c** correspondiente al mapeo de la entidad *E* también forma parte de **cm**. Todos los atributos de **cm** son claves. Se tiene una relación entre **cm** y **c**. Por lo que se puede mapear los atributos multivaluados al modelo UDM.

---

- *Relaciones entre mas de 2 objetos(un ejemplo de este caso sería representar una transacción inmobiliaria que involucra una casa, una inmobiliaria, un comprador y un vendedor)*

En el Modelo Entidad Relación, una relación  $R$  n-aria entre  $n$  entidades  $E_1..E_n$  se representa en UDM como una clase de objeto  $c$ . Esto es, se inserta una tupla en la tabla CLASE con la nueva Clase  $c$ . Para cada entidad  $E_i$  participante de la relación  $R$  se inserta una tupla en la tabla TIPO RELACION con  $tr_i$  como identificador de la relación y  $c$  y  $c_i$  como las Clases de objeto participantes en la relación, donde  $c_i$  es la clase correspondiente a  $E_i$ . Si la relación  $R$  tiene atributos  $a$ , se insertan en ATRIBUTO y en CLASE ATRIBUTO para la clase  $c$ . Los atributos claves de las clases  $c_i$  se agregan como atributos claves de  $c$ . En la columna RelTpoId se almacena la identificación de relación correspondiente  $tr_i$ .

El MER del ejemplo propuesto sería:



Las tablas UDM tendrían la siguiente información:

### Tabla CLASE

ObjCls	ObjClsDsc
Casa	Casa
Comprador	Comprador
Inmobiliaria	Inmobiliaria
Vendedor	Vendedor
Transaccion	Transaccion

### Tabla TIPO RELACION

RelTpoId	RelTpoObjCls1	RelTpoObjCls2	RelTpoCard
TranCasa	Transaccion	Casa	N1
TranComprador	Transaccion	Comprador	N1
TranInmobiliaria	Transaccion	Inmobiliaria	N1
TranVendedor	Transaccion	Vendedor	N1

### Tabla CLASE-ATRIBUTO

ObjCls	Atributo	Clave	Orden	AttNull	RelTpold
Casa	CasaCod	P	1	N	
Casa	Direccion			S	
Comprador	CompDoc	P	1	N	
Comprador	CompNombre			S	
Inmobiliaria	InmCod	P	1	N	
Inmobiliaria	RazonSocial			S	
Vendedor	VendDoc	P	1	N	
Vendedor	VendNombre			S	
Transaccion	CasaCod	P	1	N	TranCasa
Transaccion	CompDoc	P	2	N	TranComprador
Transaccion	InmCod	P	3	N	TranInmobiliaria
Transaccion	VendDoc	P	4	N	TranVendedor
Transaccion	Fecha				

2 - Investigar la eficiencia (performance) de las operaciones de acceso a datos sobre UDM.

Dentro del conjunto de las operaciones de acceso a datos que se decidió estudiar su performance, resultó que la mayoría de las mismas se comportaron de forma similar al modelo relacional. Las operaciones de "Carga masiva de datos", "Eliminar todas las tuplas de una tabla" y "Seleccionar todas las tuplas de una tabla" fueron la excepción a este comportamiento. La operación "Join" que era de particular interés conocer su performance resultó en tiempos de respuesta muy buenos.

3 - Utilizar el modelo UDM en un escenario donde se puedan aprovechar sus puntos fuertes.

Específicamente se desea evaluar el modelo UDM Como motor de almacenamiento para una herramienta del tipo "Model by example" en la cual un usuario final expresa en forma electrónica el conocimiento de una realidad y la herramienta crea la base de datos que se corresponde con la realidad y programas para realizar altas, bajas, modificaciones y consultas sobre la base de datos creada.

Se desarrolló la aplicación para ser usada con la versión 5.1 de la etapa 5 de investigación. Los módulos de "Carga de la Estructura de Datos del Formulario" y "Obtención de Claves Primaria y Reestructura" crean y reorganizan las estructuras de datos en forma dinámica sobre el modelo UDM. Se tiene entonces una aplicación que puede almacenar distintas realidades en un modelo de datos de 7 tablas relacionales: Universal Data

Model. Es decir, cualquiera sea la realidad que se tenga esto no implica una reestructura de tablas relacionales en donde se almacena la información.

### **6.3 Trabajo Futuro**

A partir de la investigación y desarrollo realizado surgen distintos lineamientos en cuanto al trabajo futuro a realizar. Estos pueden orientarse en cuanto al estudio de posibilidades de uso del UDM en distintos escenarios o a aspectos específicos de implementación en el escenario usado, no llevados a cabo en su totalidad por la aplicación desarrollada.

#### **6.3.1 Aspectos de Implementación en el escenario usado**

Aquí surgen una serie de características no implementadas pero si analizadas y que sería deseable que la aplicación "Model by Example" desarrollada cumpliera.

- **Cargar en la base de datos relacional generada por GeneXus, los datos ingresados por el usuario en los formularios.**

La aplicación almacena en el modelo UDM información relativa a aspectos de metadata de los formularios y también los datos cargados en los formularios. Sería de utilidad que una vez generada la base de datos relacional por GeneXus, la información almacenada en el UDM de datos se cargara automáticamente en las respectivas tablas del modelo relacional obtenido.

- **Deducción de Relaciones entre Clases a partir de otro criterio**

La aplicación desarrollada puede distinguir claramente las relaciones entre secciones dentro de un formulario. Esto tiene un fundamento teórico atrás obtenido a partir de un análisis efectuado para los formularios como estructura de datos. Así pues, se determina claramente la existencia de relaciones entre la sección principal del formulario (o formulario en sí) y las secciones subordinadas al primero. No sucede lo mismo para determinar las relaciones existentes entre distintos formularios. Se debe tomar algún criterio para poder determinarlas. En la aplicación desarrollada, el criterio tomado fue basado principalmente en las conexiones del formulario principal con los formularios secundarios mediante Cuadros de Lista de Selección o a través de Reglas.

Otro criterio que puede adoptarse, podría estar basado en el nombrado de campos. En la aplicación desarrollada existe la funcionalidad de renombrado de campos (Nombre de Sección + Nombre de Campo) de modo que el usuario pueda hacer uso del mismo nombre para distintos campos en distintos formularios. La aplicación podría utilizar esto para deducir las relaciones entre secciones de distintos formularios. Por ejemplo la sección

---

“Cliente” de un formulario “Factura” sería semánticamente el mismo que la sección “Cliente” del formulario de “Clientes” y así determinar que la “Factura” está relacionada con los “Clientes”.

Esto abre las puertas a un nuevo estudio para la toma de criterio para la deducción de relaciones en herramientas “Model by Example”.

- **Deducción de Claves Primarias para Clases tomando criterios estadísticos y/o operando con el usuario.**

La aplicación deduce las claves primarias de las Clases en el UDM a partir de los datos almacenados en el mismo. Esta deducción no admite ningún margen de error que pueda haber sucedido debido a por ejemplo, un error de “typeo”. Sería necesario que la aplicación maneje algún criterio estadístico para tolerar este tipo de errores.

También puede manejarse como idea, que el usuario decida a través de una interacción con paneles de advertencia, que hacer en estos casos y que Claves optar.

- **Que InfoPath cargue directamente en el UDM, tanto la estructura como los datos ingresados en el formulario.**

El usuario interactúa con la aplicación a través de un wizard en donde procesa los archivos generados por MicroSoft InfoPath para la carga de la información.

Sería de gran utilidad que este procedimiento se hiciera directamente desde InfoPath.

- **Controles o propiedades de Infopath que no se extrajo información**

Existe una serie de controles que ofrece Microsoft InfoPath al momento de diseñar un formulario, los cuales no se absorben por la aplicación. Algunos de estos son: Botones de Opción, Casilla de Verificación, Orígenes de Datos distintos a XML (base de datos), Filtros, Formulas verticales (Suma, Contar), Maestro Detalle 11, Maestro Detalle N1.

El hecho de obtener por parte de la aplicación, información especificada en estos controles y sus propiedades, amplia poderosamente la capacidad de prototipación y reflejo de lo que el usuario está expresando a través de los formularios. Esto sería una característica muy interesante a tener en cuenta en el desarrollo de herramientas del tipo “Model by Example”

### **6.3.2 Estudio de posibilidades de uso del UDM en distintos escenarios**

En el proyecto se hizo uso del UDM y su evaluación como motor de almacenamiento para una herramienta del tipo “Model by example”. Existe

---

una amplia gama de escenarios liberado a la creatividad, que pueden surgir para explotar las ventajas que ofrece el modelo UDM. Algunas de estas pueden ser:

- **Integrar el modelo dentro de un modelo relacional tradicional a los efectos de almacenar datos no previstos en design-time.**

Por ejemplo en una aplicación del tipo CRM poder almacenar datos sobre Personas que puedan ser definidos en run-time por los implementadores o usuario del sistema.

Este punto fue especificado como objetivo alternativo en la propuesta de proyecto.

- **Desarrollo de un Manejador de Base de datos que use UDM como modelo de datos.**

Sería interesante contar con un DBMS que use UDM como modelo de datos ya que en un futuro los tiempos de procesamiento de la información podrían ser menores y este punto puede dejar de ser una limitación.

Aquí se podrían definir las estructuras necesarias para la optimización de consultas y acceso a datos. También podría soportar algún lenguaje de consulta para el mantenimiento y acceso de la información.

- **UDM como modelo de datos en sistemas inteligentes**

Una de las principales ventajas que ofrece el Universal Data Model es la capacidad de poder representar cualquier realidad basándose en un esquema o metadato fijo. Esto puede llegar a ser útil en este tipo de sistemas en donde la realidad a aprender puede ser impredecible y universal. El modelo Universal Data Model debería ser analizado para evaluar su uso en este tipo de sistemas.

---

## 7 Bibliografía

- [1] Data Model Patterns, conventions of thought. David C. Hay. Primera edición. Dorset House Publishing Company. 1995.
- [2] R. Elmasri, S.B.Navathe Sistemas de Base de Datos, Conceptos Fundamentales .Segunda edición. Addison Wesley. 1997.
- [3] Ayuda de Microsoft Office InfoPath 2003 Service Pack 1
- [4] Ayuda de Microsoft Office InfoPath 2003 OnLine.  
<http://office.microsoft.com/es-mx/FX010857923082.aspx>
- [5] sitio técnico de Genexus  
<http://www.gxtechnical.com/>  
<http://www.gxtechnical.com/gxdl>  
<http://www.gxtechnical.com/wiki>
- [6] XML  
<http://www.w3.org/XML/>
- [7] W3 Schools. XML tutorial  
<http://www.w3schools.com/xml/default.asp>
- [8] W3 Schools. XSLT tutorial  
<http://www.w3schools.com/xslt/default.asp>
- [9] W3 Schools. Xpath tutorial  
<http://www.w3schools.com/xpath/default.asp>
- [10] Len Silverston. The Data Model Resource Book, Vol1: A Library of Universal Data Models for all Enterprises. Edicion revisada. Wiley. 2001
- [11] FormDocs  
<http://www.formdocs.com/index.htm>
- [12] SpectraForms  
<http://www.compliancenaavigator.com/SpectraForms>
- [13] Richard Barker. Case\*Method: Entity Relationship Modelling .Primera edición. Addison Wesley. 1989.
- [14] Oracle Designer  
<http://www.oracle.com/technology/products/designer/index.html>
-