Idioms en E-R

Juan Marcelo Flores Soliz,
marcelo@memi.umss.edu.bo
Programa MEMI, Facultad de Ciencias y Tecnología, Universidad Mayor de San Simón
Fax: 591-4-4250383, teléfono: 591-4-4252439
Cochabamba-Bolivia

Resumen
Es bastante común en el diseñador, aquel que construye modelos E-R, recurrir a patrones de solución o modelos preconcebidos para dar solución a algún problema planteado de forma específica. Este trabajo presenta algunas estructuras de modelaje para soluciones bastante comunes concebidas de una forma sencilla y pequeña en extensión y complejidad, a llamarse ***Idioms****, a veces triviales durante el desarrollo de modelos E-R. Estos idioms pueden ser usados como plantillas que se pueden acomodar para ayudar en la solución de infinidad de problemas específicos sin mucho esfuerzo.*

Palabras clave:
Modelos ER, Diagramas ER, Patrones, Estructuras sintácticas, Idiom, Reuso

1. Introducción

Una de las mayores riquezas que ha aportado el paradigma Orientado Objetos (OO), es la de ver al mundo como una colección de estructuras (por no decir Objetos). Las estructuras del paradigma orientado a objetos, son piezas de construcción de estructuras mayores, al igual que las estructuras que construyen al mundo real. Las estructuras observadas se constituyen en piezas fundamentales para la construcción de modelos que representarán el mundo real en un mundo cerrado y difícil llamado *software*.

El objetivo de este trabajo es demostrar que algunas técnicas en el desarrollo de Bases de Datos relacionales incrementan su productividad notablemente, cuando se insertan conceptos como los que el Paradigma orientado a objetos sostiene. Por ejemplo: la visión modular del mundo, la visión estructural, como conjunto de piezas estructurales más que como observación individual de entidades y sus relaciones. De esta manera la visión de aquellos que manejan el concepto E-R se verá reforzada sin perder coherencia, además de no ver invadidos sus conceptos propios y sencillos, que en definitiva son los que le mantienen el poder y preferencia.

En la sección *2.* se consideran algunas motivaciones prácticas para el trabajo presentado, luego en *3.*, *4.* y *5.* se analiza el concepto de patrón en el paradigma OO y lo que podría ser en E-R, en 6. se introduce con el concepto lingüístico de idiom, en *7.* este concepto es apropiado para nombrar a los fenómenos centrales de este trabajo, en *8.* se presenta la historia de una práctica real y actual y al fin en *9.* se presenta el catálogo de las estructuras a las que se refiere de forma central este trabajo.

2. Motivaciones

¿Qué diseñador no se ha preguntado, si el problema que está analizando no ha sido ya resuelto por otro diseñador? ¿No es acaso una curiosidad frecuente saber cómo ha resuelto el problema el diseñador de la empresa competidora?
Estas preguntas por lo general no tienen mucha trascendencia, pues muchas veces casi nunca tienen respuesta. Sin embargo, el trabajo a continuación intenta mostrar que la mayoría de las veces, las soluciones encontradas por distintos diseñadores son exactamente las mismas.

La idea de reflejar el comportamiento de las soluciones relacionadas con la estructura de objetos en un paradigma Orientado a Objetos es bastante conocida, llamado además patrones de diseño de objetos [3]. Estos patrones se comportan como soluciones genéricas o como modelos de soluciones genéricas en los cuales el diseñador debe aplicar las particularidades de su observación del problema que quiere resolver. Sin embargo las motivaciones para la búsqueda de patrones en el diseño OO también existen con casi igual razón en el diseño E-R, y aún más, puesto que la mayoría de los diseñadores en el mundo aún modelan diagramas E-R con la finalidad de usar DBMS relacionales. Esta característica seguirá siendo popular por bastante tiempo aún, puesto que la tecnología OO en los DBMS aún es bastante cara y además por que realmente no hay justificaciones razonables para que las empresas que confían en sus sistemas de información con tecnología de DBMS relacional se cambien a una tecnología OO[1].
La ventaja en la mayor abstracción lograda con los patrones radica en que los elementos de construcción de los modelos ya no son tan sueltos y débiles, sino estructuras más sólidas y estables,

[1] Es posible suponer un mejor rendimiento en la tecnología OO, sin embargo, ¿Por qué hacer inversiones grandes cuando el beneficio en cuanto a rendimiento y confiabilidad es dudosamente ventajoso?

pero es seguro que otra gran ventaja radica en que al manejar estructuras de ese tipo, es decir de construcciones sólidas pero basadas en el lenguaje original E-R, da la posibilidad de NO perder los factores de implementación del modelo que E-R provee.

3. Patrones de diseño

Existe actualmente un concepto ampliamente difundido y usado a la vez en toda la comunidad de desarrolladores de software orientado a objetos, es el concepto de patrón.
El concepto de patrones se refuerza con la concepción de un mundo formado por objetos, que a su vez conforman estructuras superiores al simple objeto observado. Si es posible observar esas estructuras o esa colección de objetos, seguramente observaremos que se repiten, es entonces cuando se dice que se ha observado un patrón, una estructura recurrente que se puede reutilizar para observaciones similares[3].

La idea de observar el mundo a través de patrones es aplicada no solamente cuando se observa el universo de discurso[2] [4], sino su gran utilidad radica en la posibilidad de aplicarlo en el modelaje del software bajo desarrollo[3] [4].
Está claro que la utilización de patrones de por sí implica una búsqueda en el ahorro de esfuerzo en el diseño de los modelos. Implica además de hecho una reutilización de los logros obtenidos con esfuerzos anteriores.

Una gran ventaja de la utilización de patrones es que las publicaciones sobre los trabajos de diseñadores en todo el mundo, está ayudando a estandarizar la visión de los patrones, mejorar su estructura con respecto a experiencias diversas o mejores, es así que se puede hablar de la existencia de una biblioteca de patrones que buscan soluciones a problemas diversos o que simplemente modelan o abstraen el universo de discurso o el software bajo desarrollo de una forma predecible según el caso estudiado[3], además de crear un lenguaje apropiado de mucho más alto nivel.

4. Análisis con el enfoque Relacional

Es interesante notar que los conceptos básicos y más simples de objeto, del paradigma orientado a objetos, se confunden con los conceptos más ortodoxos de entidad. Por lo tanto, no es extraño, que algunas técnicas usadas para reconocer entidades sean también usadas para reconocer objetos, incluyendo a las técnicas usadas para reconocer relaciones E-R, usadas también para reconocer relaciones entre objetos [4].

Entonces, si una entidad E-R en su concepción más simple y un objeto también en su concepción básica pueden ser confundidos, es posible también que las soluciones planteadas para los mismos problemas tengan un parecido estructural, sin embargo existe un problema serio: el lenguaje.
El lenguaje en el cual se escribe los modelos de objetos es diferente al lenguaje en el cual se escriben los modelos E-R, de hecho el nivel de abstracción de los lenguajes que modelan objetos es mayor que el nivel de abstracción que modela el mundo a través de E-R [2]. Veamos una justificación para ello.
El mundo visto a través de modelos orientado a objetos usando UML, tiene básicamente un conjunto finito y pequeño de posibles relaciones, a nombrar algunas: asociación, agregación, composición, generalización (o especialización), abstracción, realización, implementación [1][5]. Estas relaciones son estructuras sintácticas y semánticas completas, que abstraen de forma completa las observaciones en el universo de discurso (por lo menos esa es la esperanza).

A su vez los modelos E-R a través de la sintaxis de los diagramas E-R, no tienen los elementos sintácticos para representar tales relaciones, todo el modelo E-R debe ser construido con los simples elementos que posee, la relación entre entidades E-R, un elemento sintáctico y semántico muy simple y además único y muy guiado al desarrollo de software. Una construcción de gran abstracción en E-R es la relación de tipo IS A [4], sin embargo esta relación puede construirse con los elementos simples y nativos de los diagramas E-R tornándose esta construcción en una formación equivalente a una sin esa abstracción.

5. Patrones en E-R

Una conclusión evidente de los párrafos anteriores es que es posible construir estructuras complejas en E-R que representan algún tipo de equivalencias con estructuras OO [2].
Las estructuras E-R sin embargo serán mucho más complicadas que las OO puesto que su construcción demandará más piezas de construcción básicas (entidades tipo y relaciones) que las necesitadas por los patrones OO.

Esta característica hace muy difícil el uso de patrones con el nivel de abstracción que tiene OO. Además que tratar de "traducir" los patrones OO a sintaxis E-R sería un error puesto que los patrones

[2] UoD, Universe of Discourse [4]
[3] SuD, Software under Development [4]

OO usan de forma indistinta los objetos persistentes y aquellos que no requieren persistencia, por ejemplo: los objetos de la interfaz de usuario. El problema radica en que un modelo E-R sólo representa por excelencia aquellas entidades que requieren persistencia.

En resumen, es posible pensar en patrones para los modelos E-R, bajo las siguientes consideraciones:

- Los "patrones" E-R serían construcciones difíciles y complicadas
- Los "patrones" E-R no deberían tratar de traducir los patrones OO
- Los "patrones" E-R reflejarían solamente las estructuras de datos persistentes.

Una vez que los "patrones" E-R no tienen la misma concepción que los patrones OO, es conveniente dejarlos de llamar de esa forma. Veamos que podríamos llamarlos y que características concretas podrían tener.

Es claro que no deberían ser construcciones complicadas, pues su complejidad dificultaría su aplicación. ¿A quien le gustaría usar estructuras elaboradas poco comprensibles para solucionar sus ya complicados problemas muy poco comprensibles? Una posible solución a este problema es pensar en estructuras que no tengan mucha complejidad como los patrones y además que intenten mostrar soluciones completas a problemas concretos y completos. Estas estructuras podrían ser simplemente estructuras que posibilitan una mayor abstracción en el desarrollo de modelos E-R sin intentar constituirse en estructuras de solución de observaciones concretas.
A estas estructuras las llamaremos Idioms y las veremos en detalle en la siguiente sección.

6. IDIOMS

Idioms, no es una linda palabra, pero es muy significativa. En términos lingüísticos, *Idiom* es referido a una construcción sintáctica, es decir es una frase u oración cuyo significado no está en la interpretación de las palabras exactas que la componen, sino en la idea que representan, que puede ser un hecho histórico conocido por un grupo de personas.

La definición semántica de *idioms* dice en realidad que es posible construir estructuras sintácticas con semántica diferente. Veamos un ejemplo:

`[a piece of cake]`

Esta frase u oración tiene un significado coherente por si sola, sin embargo, desde la perspectiva de *Idiom* su significado es totalmente distinto al que plantea inicialmente palabra por palabra. En la perspectiva de *Idiom* esta oración es entendible como: ¡fácil!, o con un *idiom* equivalente en español: ¡pan comido!

7. Idioms en E-R

De forma similar que todos los lenguajes, el enfoque E-R basa sus conceptos en estructuras y reglas sintácticas muy precisas, a decir verdad las estructuras y reglas sintácticas muy formales en las cuales se basa el enfoque E-R no permiten interpretaciones ambiguas como suceden en los lenguajes humano-sociales.

Sin embargo, al igual que en los lenguajes humano-sociales, es posible pensar en estructuras o formaciones sintácticas con el lenguaje E-R. Estas estructuras sintácticas formarían estructuras semánticas mucho más complejas de las que como componentes individuales podrían lograr. Además, si a estas estructuras le añadimos algunas propiedades deseables, entonces, tenemos en los *idioms* un recurso de gran poder semántico y de mayor poder de abstracción que los símbolos y reglas de los cuales se componen.

La mayoría de los desarrolladores de sistemas de información, como parte del sistema a construir, elaboran los modelos de bases de datos o aspectos del sistema que necesitan persistencia en el tiempo y durante esa construcción, más precisamente en las primeras etapas, es decir, en las etapas de modelaje conceptual o análisis, los desarrolladores experimentados ya no consideren aspectos de normalización de forma explícita, pues la experiencia de desarrollo ha otorgado a los desarrolladores facultades de concebir entidades-tipo y relaciones que ya están normalizadas o que cumplen las formas normales recomendadas.
Este aspecto es importante resaltar una vez que se podría decir que en términos de sintaxis y semántica, los desarrolladores han concebido un medio experto para la escritura correcta de los modelos, sin analizar detenidamente las características de los símbolos que la componen (las representaciones de las entidades, sus tipos y sus relaciones). Este aspecto además, puede ser un indicio de que los desarrolladores de bases de datos con E-R han encontrado una forma superior del lenguaje que no sólo es la formación del modelo sino de su correctitud sin pasar por esa tediosa etapa de verificación de las formas normales.

Más aún otro hecho: los desarrolladores, en su mayoría, ha encontrado algunas estructuras preconcebidas en el lenguaje E-R, tal que recurre a esas estructuras cada vez que las necesita con la suposición de su correctitud. Analicemos este

hecho que parece ser más sorprendente que el primero: el uso de estructuras o formaciones sintácticas con semántica correcta, indica claramente el uso provechoso de estructuras preconcebidas o más bien el reuso de estructuras halladas con esfuerzos anteriores.
Estos hechos refuerzan la idea de encontrar y formalizar la idea de contar con estructuras sintácticas y semánticas superiores a las que el simple lenguaje pueda ofrecer como combinación correcta de símbolos.

En este estudio, veremos un tipo de estructuras bastante simples que las llamaremos *Idioms*. Las llamaremos así por las siguientes razones:

- Son referidas a formaciones o estructuras sintácticas que no deben ser interpretadas símbolo por símbolo[4], sino que deben ser interpretadas bajo otros conceptos de mayor abstracción.
- No se deben confundir con patrones, puesto que no pretenden formar una idea completa de solución ante una observación o problema planteado, de hecho los *Idioms* pueden ser usados para formar estructuras aún más complejas como los patrones referidos.

Es deseable que los *Idioms* tengan las propiedades siguientes:

Existe un conjunto finito de Idioms
Esta propiedad es referida a que el conjunto de *Idioms* es enumerable. Más aún, para que el conjunto de *Idioms* sea atractivo, debe ser pequeño.

Los conjuntos de Idioms y los aspectos del software bajo desarrollo tienen una relación completa muchos a muchos
Esta propiedad es referida a que toda observación vista con una mirada E-R básico, también puede ser vista con una mirada desde la perspectiva de *Idioms*.

La semántica de un Idiom no altera la semántica original de sus componentes.
Esta propiedad es referida al hecho de que una construcción sintáctica del *Idiom* preserva su correctitud (semántica) aún si es interpretada bajo un desconocimiento de la semántica del *Idiom*.

El uso de Idioms, proporciona una estructura de navegación correcta y completa.
Un *Idiom* no sólo es una estructura sintáctica de los modelos de gran abstracción (conceptual), sino que el *Idiom* penetra en los modelos más concretos de diseño y construcción de software. Es decir, un modelo construido en bases a *Idioms* contiene cualidades deseables de navegabilidad tal que, ¡minimiza costosos JOINS a tiempo de usar SQL!

[4] Aunque por la correctitud de los elementos con los que fueron escritos, bajo la interpretación de símbolo por símbolo, preservarían su correctitud y semántica.

8. Idioms, la experiencia

Es importante la productividad lograda con la aplicación de los *Idioms*. La experiencia descrita a continuación es referida al desarrollo de un proyecto de software para la Universidad Estatal Boliviana, más precisamente; un sistema de información académico y administrativo de carácter genérico y flexible tal que pueda adaptarse a un conjunto de Instituciones Universitarias, a la vez de cumplir con todas sus expectativas particulares.
El proyecto fué denominado Proyecto de actualizaciones tecnológicas al Sistema de información universitario. Este proyecto tiene la misión de cubrir las necesidades de administración académica institucional y cubre los siguientes aspectos:

- Administración de postulaciones, registrando su información personal y académico obtenido en el ciclo secundario, además de registro de información referida a eventos de postulación a alguna carrera universitaria.
- Administración estudiantil, referido a la información estudiantil una vez que el postulante es aceptado e incorporado a la vida universitaria.
- Administración de infraestructura universitaria, referido a registro de información sobre infraestructura inmobiliaria y equipamiento, que oferta la universidad para las actividades académicas.
- Administración de matriculación, referido al registro de información generada por eventos como el cobro de matriculas y cargos económicos al estudiante.
- Administración de estructura académica, referido a la información que constituye la oferta académica de la institución, además de información referente al desarrollo curricular académico.
- Administración de inscripciones, referido a la información sobre eventos de registro académico estudiantil.
- Administración de personal académico, referido a la información sobre docentes y profesionales académicos.
- Administración de horarios, referido a la administración de infraestructura con respecto a la demanda generada por el registro académico.

- Gestión de seguridad, referida a información necesaria con fines de seguridad de la información almacenada.

Es fácil darse cuenta que este sistema de información no es trivial. Una gran desventaja adicional en términos de desarrollo de software es el dinamismo innato del tipo de instituciones para la cual se desarrolla el software, su trasformación institucional es contínua y sus demandas de información y procesos no son muy estáticos en el tiempo, su carácter público, abierto y de co-gobierno (docente-estudiantil) otorga a esta universidad facultades innatas para un contínuo desarrollo político, institucional y administrativo muy poco atractivos para los desarrolladores de software.

El proyecto, en sus inicios ha elaborado un subconjunto de los subsistemas (aspectos mencionados líneas arriba), demandando alrededor de un mes en el desarrollo, en la parte conceptual, de cada subsistema: diagramas de procesos y diagramas E-R, siguiendo el proceso recomendado por CDM[5] propio de Oracle.
Los modelos obtenidos fueron con desconocimiento de los *Idioms* y aunque los desarrolladores poseían gran experiencia, tanto en diseño como en programación, los modelos tuvieron que ser revisados muchas veces hasta lograr una estabilidad entre lo conceptual y lo técnicamente posible (o tecnológicamente posible).
Una de las trabas grandes en el proceso de desarrollo es sin lugar a dudas las ataduras que fueron creadas tanto en los conceptos como en los procesos correspondientes a fines de los 80s e inicios de los 90s, e incluso anteriores.
La observación de estos procesos y la observación de las estructuras logradas han permitido pensar en los *Idioms,* aspecto central de este trabajo.
Para el desarrollo de los restantes subsistemas, parte del mismo equipo de desarrolladores ha logrado producir los modelos y software en un tiempo hasta cuatro veces inferior, es decir un modelo que demanda un mes de desarrollo, es posible realizarlo en una semana con la idea de *Idioms*, además de que los modelos ER obtenidos tienen cualidades adicionales como correctitud en la navegabilidad y normalización.

Algunos conceptos debieron ser rotos para este tipo de desarrollo: por ejemplo la famosa regla "7±3" [8] que indica que un diagrama debe contener a lo sumo 7±3 elementos, puesto que si violamos esta regla se corre un riesgo alto de ilegibilidad del modelo, o peor, de no haber realizado una correcta división o modularidad del sistema.
Con la idea de *Idioms*, los modelos que se construyen pueden tener hasta el triple de elementos sin perder legibilidad, puesto que el desarrollador ya no ve elementos sueltos del diagrama, sino modela y observa estructuras sintácticas con una semántica diferente y completa.
En otras palabras, se incrementa la productividad con añadido de capacidades para afrontar desarrollo de software de cada vez mayor envergadura (típico en nuestros días en contraste a aquellos cuando fue enunciada la regla 7±3), además de otorgar al desarrollador capacidades de visión de la arquitectura del sistema, los *Idioms* también proporcionan de forma implícita capacidad de implementar las famosas reglas de cohesión y acoplamiento.[8]
Sin embargo una de las más importantes reglas pasadas a su procesamiento en segundo plano fué la de "buscar normalización como parte esencial de correctitud y eficiencia del modelo"[2], Los *Idioms* proveen de forma implícita las formas normales recomendadas para un buen diseño. Por ejemplo: la aplicación del *Idiom clasificación* (ver más adelante) ayuda a reconocer los atributos multivaluados y lograr la forma normal recomendada para esta observación.
Sin embargo es bueno reflexionar sobre algunas formas normales, por ejemplo: sobre la de no permitir el almacenamiento de valores de atributos que pueden calcularse. Recordemos que esta forma normal persigue el objetivo de hacer eficiente el almacenamiento confiando en la capacidad de cálculo. Esta forma normal atenta en alguna medida los requerimientos actuales de velocidad de recuperación y comunicación en nuestros días de Internet.
En definitiva la aplicación de los *Idioms* "estirando" algunas reglas y paradigmas establecidos ha permitido lograr un gran rendimiento en cuanto a tiempo de desarrollo y su calidad.
A continuación viene un cuadro (cuadro Nº 1)que muestra en datos numéricos el producto de desarrollo realizado sin *Idioms*, y otro con *Idioms*, ejemplificando el tiempo como medida de esfuerzo. Este cuadro no toma en cuenta el proceso de revisión sufrido por los modelos realizados sin *Iidioms*, que en su mayoría fueron por factores de navegabilidad del modelo y aquellos por minimizar los costosos *joins*.
El proyecto ha sido denominado ISKAY a partir de su segunda etapa, coincidiendo con la aplicación de los *Idioms* y en la actualidad está desarrollándose en sus etapas más avanzadas,

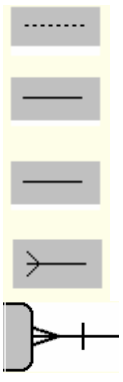
[5] Custom Development Method, por Oracle Corp.

Aspectos modelados en ERD Incluye número de Tipos/entidades	Tiempo de desarrollo Sin IDIOMS	Tiempo de desarrollo Con IDIOMS
calendario académico 4 controles 3 convalidación 6 reglas registro notas 3 datos biográficos estudiante 25 datos biográficos empleado 4 datos generales 48 distribución grupos 3 escala notas 6 estructura académica 31 estudiante excluido 6 horario 15 infraestructura 8 mantenimiento registro académico 3 parámetros de oferta asignatura 4 peso parámetro notas 6 registro certificación 6 registro titulación 13 reglas inscripción 10 seguimiento académico docente 6 seguimiento académico estudiante 3 titulación 15	8 meses, para un TOTAL de 228 Tipos-entidades	
websiss 9 postulación 12 matriculación 19 inscripciones 9 admisión 12 caja 23 ciclo de vida 6 seguridad 43 Auditoría 16 Auditoría Seguridad 9		1 mes, para un TOTAL de 159 Tipos-entidades

Cuadro N° 1

como la de realizar interfaces de usuario e interfaces para recuperación y minería de datos. En la página siguiente (Figura Nº 1) se muestra un ejemplo, un fragmento de los diagramas E-R, en el cual se describe el modelo como un conjunto de *Idioms* relacionados entre sí.

Para la observación del ejemplo y la descripción misma de los *Idioms* se describe la sintaxis ER particular (o dialecto ER) elegida para la representación de los diagramas, ésta es la propuesta por Oracle, y básicamente es como sigue:



Obligatoriedad

Opcional

Requerido

Cardinalidad

Uno

Muchos

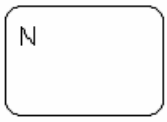
Dependencia

La barra en la línea de relación define un tipo-entidad débil (la del lado de la barra)

Características de atributos

#	Llave primaria
*	Requerido
º	Opcional

Tipo-Entidad



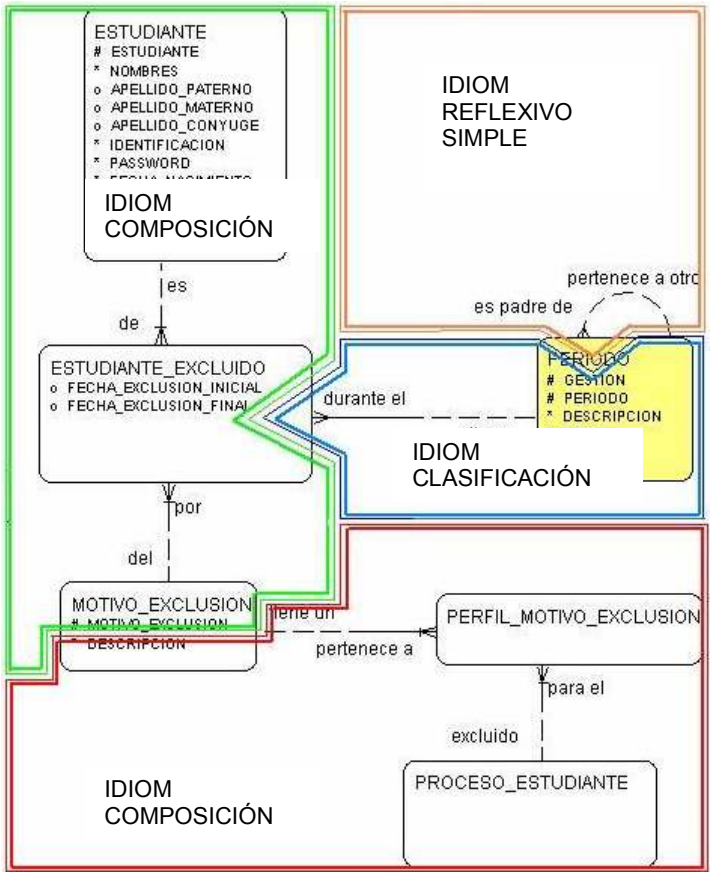


Figura N° 1

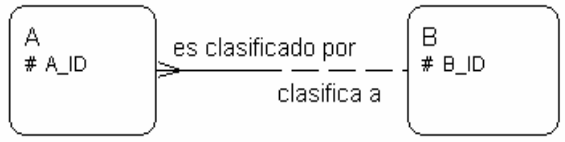
En este fragmento, se puede ver el uso de los *idioms* de una forma superior a la escritura normal ER; es decir, los *Idioms* constituyen piezas de mayor abstracción y representatividad en el Universo de Discurso. Por tanto, la construcción de modelos ER no se realiza con los conceptos ER primitivos sino con *Idioms*, aunque en la escritura final se recurre irremediablemente a ER.

9. *Finalmente: Los Idioms*

Nombre del Idiom: *Clasificación o Catálogo*

Uso: Cuando se tiene una entidad-tipo, con un atributo que puede tener múltiples valores que además son predecibles aunque no necesariamente en un número estable.

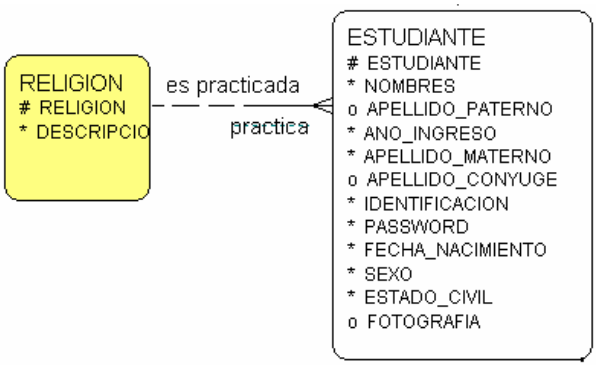
En lenguaje ERD



Semántica: La entidad-tipo B clasifica los posibles valores que puede tener la entidad-tipo A en los valores de su atributo B. En otras palabras, la entidad-tipo B se constituye en un catálogo de recursos de entidades que tipifica.

Características y ventajas de este Idiom:

- Permite el crecimiento del conjunto de entidades clasificadoras.
- Otorga una cerradura al conjunto de entidades clasificadoras.
- El clasificador o catálogo puede ser reusado múltiples veces.

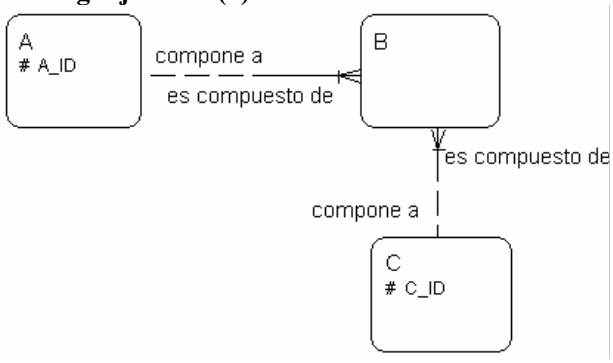


Ejemplo: La entidad-tipo RELIGION, clasifica los múltiples valores que puede tomar esa observación en la entidad-tipo ESTUDIANTE.

Nombre del Idiom: *Composición*

Uso: Cuando se tiene una entidad cuya existencia es dependiente de otras entidades diferentes. En definitiva, cuando se quiere la representación de entidades compuestas (una relación que en E-R no existe explícitamente), pero que es fácilmente observable. (La aridad de la composición puede ser diversa)

En lenguaje ERD (*)

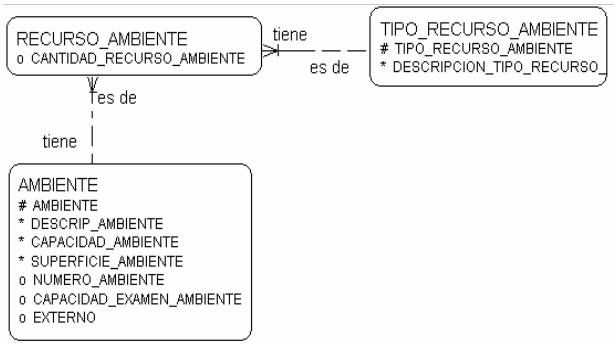


(*) Es posible añadir Entidades-tipo a la composición aumentando la aridad de la composición.

Semántica: Una entidad B, es el resultado de la composición de dos entidades; C y A, cuyo dinamismo es el que permite la creación de entidades B. En otras palabras se usa este idiom, cuando se quiere registrar la historia de las acciones de C y A.

Características y ventajas de este Idiom:

- Permite el registro de la historia de las relaciones entre dos (o más) entidades
- De forma particular, cada nueva entidad compuesta refleja implícitamente el dinamismo de las entidades de cuales se compone.

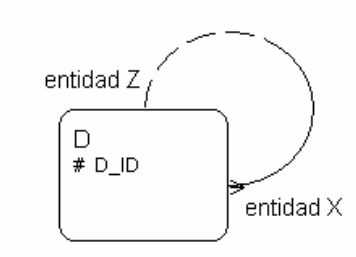


Ejemplo: Entidades del tipo AMBIENTE, se componen con entidades de tipo TIPO_RECURSO_AMBIENTE, con la finalidad de componer una nueva entidad tipificada como RECURSO_AMBIENTE. Notar la dependencia de existencia de entidades del último tipo con respecto a las otras

Nombre del Idiom: *Reflexión simple*

Uso: Cuando se tiene la observación de relaciones entre entidades del mismo tipo. Un caso muy típico es cuando se desea representar relaciones de orden en entidades similares.
Otro uso muy frecuente es cuando se quiere hacer uso de recursividad, es decir de la exploración de un camino trazado por elementos similares (las entidades del mismo tipo)

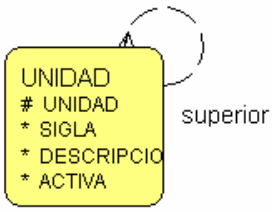
En lenguaje ERD



Semántica: Entidades de tipo A, se relaciones con sus semejantes también de tipo A.

Características y ventajas de este Idiom:

- Permite aclarar el hecho de que no existen relaciones unarias en esencia.
- Permite Modelar relaciones recursivas: es decir definir un medio de recursión (la relación) y un punto de parada de recursión (la opcionalidad de la relación). Nótese que si este idiom se representa sin la posibilidad de relaciones con una entidad NULL (sin la opcionalidad), debe crearse una entidad ficticia que sirva de entidad de parada en la recursión.

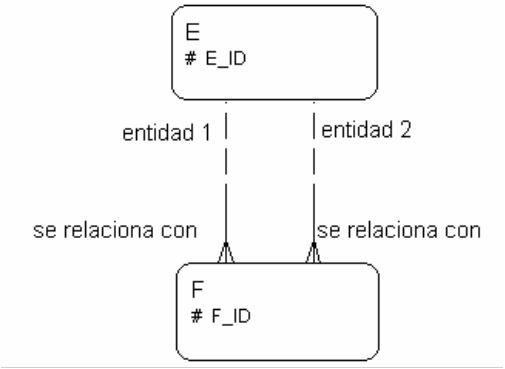


Ejemplo: Entidades del tipo UNIDAD, pueden tener una relación de orden llamada *superior*, en la que una de ellas cumple el rol de superior con respecto a la(s) otra(s).

Nombre del Idiom: *Reflexión compuesta*

Uso: Cuando se tiene la observación de relaciones entre entidades del mismo tipo, que además se desea saber la historia de esas relaciones o calificar esas relaciones con algunos atributos propios. Un caso típico de uso es; cuando se desea representar relaciones de orden calificadas.

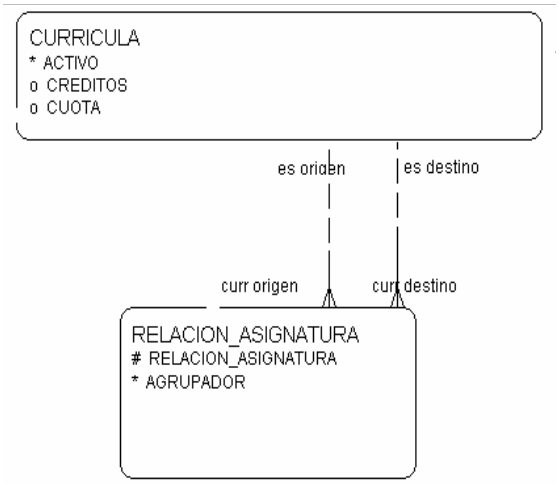
En lenguaje ERD



Semántica: Entidades de tipo E, se relacionan con entidades de tipo E con relaciones de la forma F

Características y ventajas de este Idiom:

- Permite registrar historias de relaciones reflexivas además de calificarlas
- También permite Modelar relaciones recursivas (similares al idiom reflexivo simple).

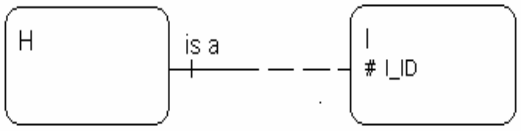


Ejemplo: Entidades del tipo CURRICULA (asignaturas en esencia) tienen relaciones de orden que además están calificadas cuando se las ven en RELACION_ASIGNATURA

Nombre del Idiom: *is a*

Uso: El **is a** clásico, cuando se observa una especialización de una entidad con respecto a otra o mirándolo de otra perspectiva, cuando se observa una generalización de una entidad con respecto a otra

En lenguaje ERD (*)(**)

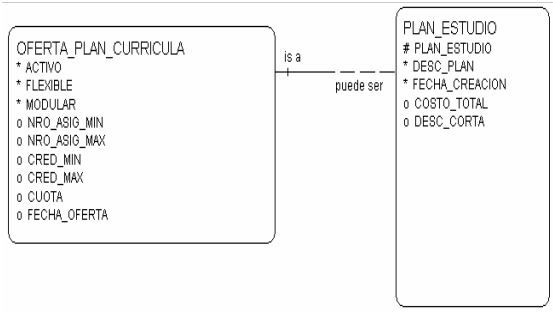


(*) Es bueno notar que esta es sólo una posible implementación del concepto IS A, siendo posible otras implementaciones diferentes.
(**) Es posible añadir entidades-tipo (de lado contrario a *is a*) aumentando la aridad de la generalización.

Semántica: Una entidad de tipo H es una (is a) entidad de tipo I

Características y ventajas de este Idiom:

- Permite extender las características de una entidad en particular sin obligar a las otras similares.
- Permite definir una relación de especialización entre entidades. (Natural en el paradigma OO).
- Es probable confundir este concepto asociándolo muy estrechamente al mecanismo de Herencia que utilizan los lenguajes OO para reflejar este tipo de relación, sin embargo este concepto les da problemas a la hora de considerar generalización múltiple (muy natural en el mundo real). Lo cual no sucede con E-R y/o SQL que podrían no usar este concepto para implementar IS A (generalización)
- Es posible tener múltiples relaciones IS A para una entidad especializada (H) definiendo de esta manera una generalización múltiple.

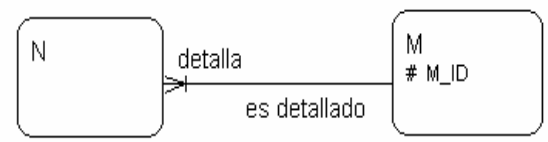


Ejemplo: Cada Entidad OFERTA_PLAN_CURRICULA extiende características de entidades de PLAN_ESTUDIO

Nombre del Idiom: Maestro - detalle

Uso: Cuando se quiere registrar entidades diferentes que en su conjunto detallan a una sola. Es decir un conjunto de entidades subordinadas a detallar a su maestro.

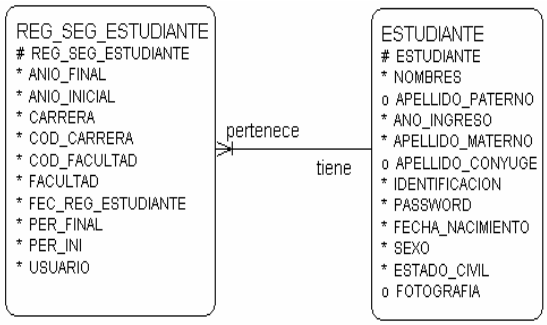
En lenguaje ERD



Semántica: Una entidad M es detallada por N entidades

Características y ventajas de este Idiom:

- Permite la representación de entidades compuestas por varias entidades del mismo tipo.
- Muy similar al concepto maestro-detalle en el manejo de interfaz de usuario. (en la representación de bloques de datos subordinados a otros).

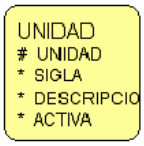


Ejemplo: Cada entidad ESTUDIANTE es detallada con su(s) correspondientes REGISTROS DE SEGUIMIENTO ESTUDIANTE, por cada carrera que estudia.

Nombre del Idiom: Básico

Uso: Idiom Básico o Unidad de construcción. Representa el hecho de identificar claramente entidades de un tipo sobre el cual girará las observaciones de los Idioms restantes. Es el punto de partida de cada fragmento de un diagrama ER.

Ejemplo:



10. Algunas conclusiones

¡Los *Idioms* en el diseño E-R funcionan! son mejores cuando son vistos como piezas pequeñas de construcción de grandes modelos.

Todos los *Idioms* presentados en realidad corresponden a familias de *idioms* (jugando con la rigidez u opcionalidad de las relaciones), sin embargo los *Idioms* que presentan relaciones rígidas no son muy recomendados, puesto que a la hora del diseño habrá que solucionarlos enfrentándose a las restricciones tecnológicas.

Los *Idioms* de alguna manera ayudan a enriquecer el modelo E-R con sus construcciones sintácticas, sin embargo, es seguro que la cualidad más importante de los *Idioms*, radica en su poder semántico, guiando la observación del UoD de una forma menos traumática y más realista desde el punto de vista del modelaje conceptual, plasmando sus beneficios en los modelos siguientes de menor abstracción. Es así que la capacidad de modelaje se ve incrementada con un poder semántico razonablemente cercano al poder deseado en modelos de datos semánticos[6] [10], quizás por la similitud de las abstracciones deseadas planteadas en [10] con los *Idioms*.

Es posible que existan otros *Idioms*, sin embargo, los presentados en este trabajo cubren ya una gama grande de posibilidades de diseño en al área de sistemas de información reactivos [11]. Por otro lado el aspecto atrayente de la concepción de los I*dioms* es su sencillez y su poco número, así un diseñador puede tenerlos presentes sin mucho esfuerzo concentrándose en los aspectos del problema más que del *Idiom*.

[6] El análisis y contrastación de modelos de base de datos semánticos con los Idioms, fué realizada despues de la experiencia que permitió obtener los Idioms (nota del autor).

Sin embargo las estructuras presentadas de los *Idioms* no son estructuras completas capaces de resolver problemas completos, simplemente son estructuras simples que ayudan a dividir y observar el mundo de forma mas abstracta, es decir sin pensar en dirección de las relaciones (la navegabilidad), la multiplicidad exacta (en términos del conteo de entidades que participan en las relaciones) y sin siquiera pensar en los roles, puesto que el *Idiom* los provee. Sin embargo al concebir el mundo en base a estos *Idioms* que de por si llevan consideraciones de implementación implícitos, el resultado es que se usan herramientas de mayor abstracción sin perder el hilo de la implementación, mas bien teniéndola siempre presente, con posibilidades grandes de éxito en su codificación.

11. Referencias bibliográficas

[1] The Unified Modeling Languaje, Object Management Group, www.omg.org
[2] Diseño de Bases de Datos con UML, Paul Dorsey-Joseph R. Hudicka, Oracle Press, 1999
[3] Design Patterns, E. Gamma, R. Helm, R. Johnson and J. Vlissides. Addison-Wesley, 1995.
[4] Requirements Engineering: Frameworks for Understanting, R.J. Wieiringa, Wiley, 1996.
[5] UML y Patrones, Craig Larman, Prentice-Hall, 2nd edition 2002.
[6] Oracle Designer-database Generator, OraclePress, 2001
[7] Fundamentos de Sistemas de Bases de Datos, Ramez A. Elmasri-Shamkant B. Navathe, Addison Wesley, 3ra. edición, 2002
[8] Análisis Estructurado Moderno, E. Yourdon, Yourdon Press, 1992.
[9] The Entity-Relationship Model-Toward a Unified View of Data, Peter Pin-Shan Chen, ACM Transactions on DataBase Systems, Vol 1, Nº1.
[10] Evolution of Data Modeling for Databases, Shamkant B. Navathe, Communications of the ACM, Vol.35,Nº 9, 1992.
[11] Design Methods for Reactive Systems: Yourdon, Statemate and the UML, R.J. Wieringa, Morgan Kaufmann, 2003
[12] Documentación técnica de: ISKAY proyecto de desarrollo de software, Programa MEMI, UMSS, 2005 (todos los ejemplos son reales, obtenidos de los modelos de software desarrollados para ISKAY)

Agradecimientos especiales a:
Carlo Calderón y Gustavo Jiménez, por sus aportes e implementaciones reales de los Idioms.
Vladimir Costas, por su aporte en la definición conceptual de la relación is a.
Pablo Azero, por su ayuda en la definición del concepto de Idiom.