

Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

Práctica Laboral



Tema: Diseño de un metamodelo para el modelo
Entidad-Relación Extendido

Autores:

Aicenis M. Castro Oves-García

Hayder L. Endo Pérez

Tercero de Ciencia de la Computación

Curso 2018-2019

Resumen

Teniendo en cuenta que el modelo Entidad-Relación (ER) y sus extensiones (EER) es uno de los modelos de datos de alto nivel más populares en la actualidad para el modelado de esquemas conceptuales para aplicaciones de Base de Datos, se hace útil contar con un metamodelo que lo describa y sea independiente de su definición gráfica, siendo esta perspectiva el objetivo principal del presente trabajo. El metamodelo presentado está descrito en un modelo Ecore del marco de trabajo para el modelado de Eclipse (EMF), posibilitando así su vínculo con GMF para la generación de editores gráficos a partir de la definición propuesta.

Abstract

Taking into account that the Entity-Relationship (ER) model and its extensions (EER) is one of the high-level data model currently most popular for the modeling of conceptual schemes for Database applications, it is useful to have a metamodel that describes it and that can be independent of its graphic definition, that's why this perspective becomes the main objective of the present work. The presented metamodel is described in an Ecore model of the Eclipse Modeling Framework (EMF), thus enabling its link with GMF in order to generate graphic editors based on the proposed definition.

Índice general

| | |
|---|-----------|
| Introducción | 1 |
| 1. Fundamentación Teórica | 3 |
| 1.1. Creación de un esquema conceptual | 3 |
| 1.2. Modelo de datos Entidad-Relación | 4 |
| 1.2.1. Entidades | 4 |
| 1.2.2. Atributos | 5 |
| 1.2.3. Relaciones | 6 |
| 1.3. El modelo Entidad-Relación extendido (EER) | 9 |
| 1.3.1. Conceptos incorporados al modelo ER | 9 |
| 1.4. Arquitectura dirigida por modelos | 12 |
| 1.5. Eclipse | 12 |
| 1.6. Eclipse Modeling Framework (EMF) | 13 |
| 1.7. Graphical Modeling Framework (GMF) | 14 |
| 2. Implementación de un metamodelo para el modelo ER y EER | 18 |
| 2.1. Otros metamodelos | 18 |
| 2.2. Modelado del esquema conceptual | 22 |
| 2.3. Implementación del metamodelo en EMF | 24 |
| Conclusiones | 28 |
| Recomendaciones | 29 |
| Bibliografía | 30 |

Introducción

Uno de los modelos de datos de alto nivel más completo y popular es el modelo Entidad-Relación, al posibilitar un modelado conceptual con gran expresividad, sencillez y brindar una representación diagramática con facilidad de interpretación. El uso de herramientas CASE (Computer Aided Software Engineering) para el modelado de esquemas conceptuales usando los conceptos de modelado del modelo ER tienen relevante importancia para el desarrollo de aplicaciones de base de datos porque permiten la descripción de los requisitos del dominio por parte de los usuarios independientemente del procesamiento que los transforma u otras consideraciones de eficiencia. Además, con una herramienta CASE se agiliza el proceso de desarrollo de software a partir de los modelos construidos. En orden de poder contar con herramientas CASE que permitan la representación de esquemas conceptuales mediante el modelo EER y que tengan en cuenta todas las restricciones y relaciones que este define para el modelado, se torna fundamental tener una definición de sus elementos y características.

El proceso de metamodelado brinda la definición de lenguajes de modelado mediante un metamodelo, que no es más que el modelo que describe un modelo de datos de alto nivel. La descripción de un lenguaje de modelado, permite contar con una definición de sus elementos, relaciones y restricciones de forma resumida y sencilla, facilitando su comprensión y aplicación en la representación de esquemas conceptuales para una aplicación dada. Por ello, cobra notable importancia la descripción del modelo Entidad Relación Extendido, dado que es un modelo de datos de alto nivel que expresa con bastante precisión y claridad el esquema conceptual.

Debido a que existen diferentes notaciones diagramáticas alternativas para la representación de un esquema conceptual, es de gran utilidad el diseño de un metamodelo que describa los conceptos del modelo ER y su extensión (EER) y facilite la generación de un editor gráfico para el modelado independiente de una notación específica.

Actualmente, el reuso de diferentes elementos de software es una alternativa que busca potenciarse, debido al reto de desarrollar productos de calidad en limitados períodos de tiempo que implican una alta productividad. El modelado es una estrategia clave en este ámbito, pues mediante su transformación en las diferentes etapas del desarrollo de software permite de forma progresiva la obtención del producto final, por lo que potenciar el modelado se convierte en el propósito principal de la Arquitectura Dirigida por Modelos (del inglés Model Driven Architecture) propuesta por la OMG (Object Management Group). La MDA brinda las guías para el desarrollo de aplicaciones a partir de modelos, de esta forma va quedando en el pasado la percepción de los modelos únicamente como documentación.

Además, la interoperabilidad entre las diversas plataformas y tecnologías existentes, junto a la rápida evolución de las tecnologías que limita en muchos casos el reuso de software, son problemas a los que se enfrenta actualmente la industria de software. En este orden, MDA busca enfatizar en el dominio de los problemas y no en las tecnologías que brinden una solución, separando los aspectos que lo describen de la

tecnología, así como desarrollar la estandarización para facilitar la interoperabilidad entre las diferentes herramientas.

Las líneas de trabajo propuestas por MDA son prometedoras para el proceso de desarrollo de software, y revitalizan el rol de las herramientas CASE en este ámbito, haciendo necesaria la inclusión de nuevas funcionalidades que permitan una automatización e intercambio durante el proceso de desarrollo.

Objetivo general:

Confeccionar un metamodelo que describa el modelo ER y elementos del modelo EER independiente de la definición gráfica.

Objetivos específicos:

1. Determinar los aspectos teóricos necesarios para el diseño del metamodelo deseado.
2. Diseñar un esquema conceptual que describa el modelo ER y sus extensiones.
3. Implementar un metamodelo mediante EMF que represente los elementos descritos en el esquema conceptual.

El informe está dividido en dos capítulos, el capítulo 1 resume una fundamentación teórica necesaria para el desarrollo del objetivo planteado, en él se incluyen los elementos que integran el modelo ER y EER, así como una breve introducción a las tecnologías de modelado a usar para la implementación del metamodelo. El capítulo 2 incluye el esquema conceptual a realizar y el metamodelo que lo describe.

Capítulo 1

Fundamentación Teórica

En este capítulo se abordan los principales aspectos teóricos necesarios para el desarrollo de un metamodelo que describa con un alto índice de fidelidad los conceptos característicos del Modelo Entidad-Relación (ER) y el Modelo Entidad-Relación Extendido (del inglés Enhance Entity Relationship (EER)). Inicialmente se explica detalladamente los elementos del modelo ER y EER que serán modelados. Se explica el uso de modelos para el desarrollo de herramientas, siendo este el concepto fundamental de la Arquitectura Dirigida por Modelos (del inglés *Model Driven Architecture* (MDA)) de la que forma parte el proyecto de Eclipse para el desarrollo de las tecnologías basadas en modelos. Otras secciones dan una breve introducción al proyecto de modelado del software Eclipse (*The Modeling Project*) que brinda la posibilidad de representar el modelo de dominio de la aplicación a través del *Eclipse Modeling Framework* (EMF) a partir del cual se puede generar un editor gráfico con el marco de trabajo para el modelado gráfico del proyecto de modelado de Eclipse (GMF).

1.1. Creación de un esquema conceptual

Entre los primeros pasos del proceso de diseño de una base de datos están la recopilación de requerimientos en cuanto a datos de los usuarios, así como los requisitos funcionales, que no son más que las transacciones y operaciones que se realizarán sobre la base de datos. La creación de un esquema conceptual entra una vez guardados todos los requisitos necesarios. El esquema conceptual se desarrolla mediante los conceptos fundamentales del modelo de datos de alto nivel que lo modela, es decir que describe la representación de los datos de forma abstracta. Algunas de las ventajas del modelado de un esquema conceptual son la no inclusión de detalles de implementación lo que facilita la comunicación con usuarios sin conocimientos técnicos. Además, una vez realizado el modelo se puede chequear si se satisfacen todos los requisitos recopilados inicialmente y evitar conflictos entre ellos.

1.2. Modelo de datos Entidad-Relación

Uno de los modelos de datos de alto nivel que desarrolla el esquema conceptual de una base de datos es el modelo Entidad-Relación (ER), el cual se caracteriza por describir los datos como entidades, atributos y relaciones, según (Elmasri and Navathe, 2011). Este modelado facilita la comprensión de los datos y sus funciones, además mejora el mantenimiento y el control de posibles errores.

1.2.1. Entidades

Una entidad es un objeto del mundo real, su existencia no tiene dependencia alguna y puede ser física o conceptual. Los atributos describen la entidad y su valor es almacenado en la Base de Datos, según (Elmasri and Navathe, 2011).

Cuando se tienen varias entidades con los mismos atributos, pero diferentes valores se agrupan en un tipo de entidad, que tiene un nombre que identifica el conjunto de entidades y los atributos que las definen. Al tener un conjunto de entidades con diferentes valores para sus atributos interviene la necesidad de diferenciarlas entre sí, esto se implementa mediante una restricción de unicidad para los atributos, marcando un atributo como clave, es decir, cada entidad del conjunto tendrá el atributo definido como clave con valor diferente para cada entidad, de esta forma se garantiza que cada entidad sea única en su conjunto y además pueda identificarse.

Un tipo de entidad puede tener más de un atributo clave, este caso se da cuando un atributo solo no puede garantizar que cada entidad sea única, o sea, cuando no puede identificarla.

Cuando un tipo de entidad no tiene atributo clave es denominado tipo de entidad débil.

Por tanto, podemos ver la existencia de dos tipos de entidades, los tipos de entidades fuertes, cuando existe uno o varios atributos identificadores para el tipo de entidad, y los tipos de entidades débiles.

Los tipos de entidades débiles identifican sus entidades a través de la relación con otro tipo de entidad denominado propietario o identificado y la combinación de uno o varios atributos denominados clave parcial en el tipo de entidad débil y el/los atributos clave de la entidad propietaria. Un tipo de entidad débil no puede identificarse sin una entidad propietaria, por tanto, tiene una dependencia de existencia respecto a la relación con la entidad propietaria, relación que se denomina identificadora y será explicada en la subsección 1.2.3 en página 6 referente a las relaciones y sus tipos. Tener en cuenta que un tipo de entidad propietaria puede ser a su vez un tipo de entidad débil y puede tener más de un tipo de entidad propietaria.

La notación para representar entidades en los diagramas de ER es un rectángulo con el nombre asignado a la entidad dentro y en el caso de las entidades débiles, se rodea el rectángulo con líneas dobles.

1.2.2. Atributos

Los atributos son las propiedades que describen una entidad o un tipo de entidad. Existen varios tipos de atributos que podrían agruparse en tres categorías que se solapan, es decir, que caracterizan al atributo a la vez, la composición del atributo, dígame simple o compuesto, el valor, ya sea monovalor o multivalor, y el almacenamiento que podría ser almacenado o derivado. Además, existe el tipo de atributo complejo, que no es más que un atributo compuesto por atributos de tipo multivalor y compuestos arbitrariamente, según (Elmasri and Navathe, 2011).

Atributos simples

Cuando nos referimos a un atributo como una unidad que no se divide en partes que tendrían un valor diferente cada una tenemos un atributo simple.

Atributos compuestos

Si deseamos referirnos las partes en las que se pueda dividir un atributo, de valores diferentes, estamos frente a un atributo compuesto, es decir, un atributo de tipo compuesto esta integrado por varios atributos simples. Incluso, podríamos tener una jerarquía de atributos simples que componen al compuesto.

Atributos monovalor

Tenemos atributos monovalor cuando el atributo referido tiene y puede tomar solo un valor.

Atributos multivalor

Un atributo es de tipo multivalor cuando puede tener varios valores para la misma entidad. Los atributos de este tipo pueden restringir el número de valores para una entidad específica si se les añade límites superior e inferior.

Atributos almacenados

Los atributos son de tipo almacenados cuando su valor es dado y guardado en la base de datos y su determinación no depende de otros atributos ni de entidades relacionadas, es decir, que su valor no se va a derivar de otros valores dados, ya sea a partir de atributos o entidades que se relacionen con la entidad que contiene el atributo en sí.

Atributos derivados

Opuesto a los atributos almacenados, están los atributos derivados, denominados así a causa de que su valor se puede determinar a partir de su relación con un atributo almacenado o con una entidad relacionada con la entidad a la cual caracteriza. Viso de otra forma, el valor de estos atributos puede derivarse de otros valores ya almacenados.

La notación para modelar los atributos es un óvalo con el nombre del atributo dentro y una línea recta que lo enlaza con el tipo de entidad que describe. Un atributo compuesto se une a los atributos que lo componen mediante líneas rectas. Los atributos multivalor encierran el nombre del atributo en dos óvalos. Los atributos clave se modelan de igual forma, pero con el nombre subrayado y los atributos de clave parcial subrayados con una línea discontinua.

1.2.3. Relaciones

Un tipo de relación es el conjunto de varias relaciones, donde cada relación asocia o relaciona 2 o más entidades de diferentes tipos de entidades, y a su vez un tipo de relación relaciona n tipos de entidades. Es decir, se tiene un tipo de relación R , que agrupa r_n relaciones, donde cada relación r_i puede relacionar n entidades e_i , con $n > 2$ y cada e_i pertenece a diferentes tipos de entidades E_n . Los conceptos de relación y tipo de relación quedan reflejados en la figura.

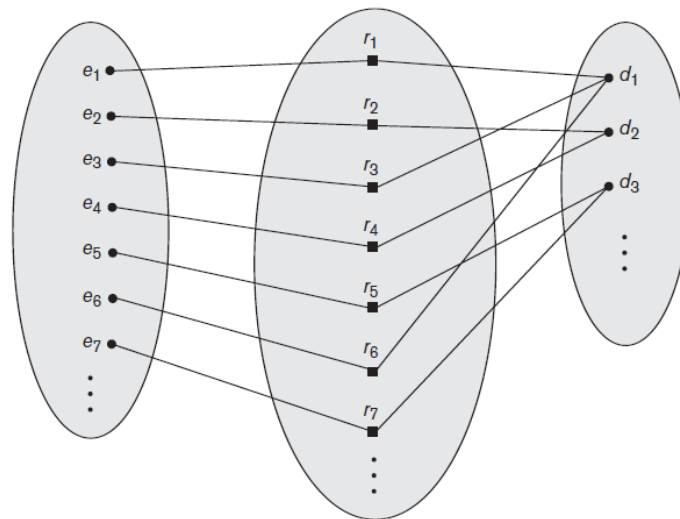


Figura 1.1: Tipo de relación y relaciones, fuente: (Elmasri and Navathe, 2011)

Para referirse a las entidades asociadas por una relación se usa el termino participa, o sea, son las entidades que participan en la relación, igualmente los tipos de entidades que relaciona un tipo de relación son los tipos de entidades participantes en el tipo de relación.

La cantidad de tipos de entidades que participan en un tipo de relación es a su vez el grado del tipo de relación. Por ejemplo, cuando un tipo de relación asocia dos tipos de entidades es de grado dos y se denomina binario, igualmente, si es de grado tres, el tipo de relación es ternario. De forma análoga para referirse a la relación entre las instancias de dos o tres tipos de entidades, usamos los términos relación binaria y ternaria respectivamente.

La notación para modelar los tipos de relación es un rombo con el nombre del tipo de relación dentro y líneas que lo conectan con los tipos de entidades participantes,

según (Elmasri and Navathe, 2011).

Relaciones recursivas

Se da una relación recursiva cuando el mismo tipo de entidad participa dos o más veces en un tipo de relación con diferentes papeles a desempeñar, es decir, cuando una relación asocia dos o más entidades del mismo tipo de relación.

Roles

En la asociación que establece un tipo de relación, cada tipo de entidad que participa tiene un rol o papel a desempeñar, esto se representa mediante el nombre de rol, el cual puede ser omitido cuando los nombres de los distintos tipos de entidad que intervienen en el tipo de relación reflejan por sí solos un rol. El nombre de rol cobra particular importancia ante las relaciones recursivas, caso donde se relacionan entidades pertenecientes al mismo tipo de entidad por tanto se hace necesario especificar qué rol tiene cada entidad en la relación, según (Elmasri and Navathe, 2011).

Razón de cardinalidad

En una relación binaria, debe especificarse el número máximo de instancias de relación en las que pueden participar las entidades relacionadas, esto se modela mediante la razón de cardinalidad. Para una relación r_i que asocia entidades entre dos tipos de entidades E_1 y E_2 existen cuatro variantes de razón de cardinalidad.

Según (Elmasri and Navathe, 2011):

- 1:1 Caso donde solo una entidad e_i de E_1 puede desempeñar su rol en la relación con una única entidad e_j de E_2 , y solo la entidad e_j de E_2 puede estar relacionada con e_i de E_1 y ejercer su rol en la relación únicamente sobre ella.
- 1:N Como máximo una única entidad e_i de E_1 puede estar relacionada con varias entidades e_1, e_2, \dots, e_k de E_2 , y las entidades e_1, e_2, \dots, e_k de E_2 estarán relacionadas solo con la entidad e_i de E_1 .
- N:1 Varias entidades e_1, e_2, \dots, e_k de E_1 podrán relacionarse con solo una entidad e_j de E_2 . Por la otra parte de la relación una única entidad e_j de E_2 se relacionará con muchas entidades de E_1 .
- M:N Las entidades e_1, e_2, \dots, e_k de E_1 están relacionadas con e_1, e_2, \dots, e_k de E_2 , y en sentido opuesto de igual forma.

Restricciones de participación

Otra característica de las relaciones son las restricciones de participación, así denominadas debido a que modelan la dependencia de existencia de una entidad respecto a su relación con otra, es decir, especifica si para que exista una entidad, tiene que participar en al menos una instancia de un tipo de relación, de ser así la relación tiene

participación total, porque se está aplicando una restricción donde todas las entidades del tipo de entidad E_1 tienen que estar relacionadas con al menos una entidad del tipo de entidad E_2 . La participación es parcial cuando solo parte del conjunto de entidades de E_1 participa en la relación.

La participación total se modela conectando el tipo de relación con el tipo de entidad participante mediante líneas dobles y las participaciones parciales son representadas con una conexión con línea sencilla, según (Elmasri and Navathe, 2011).

Atributos de los tipos de relación

Los tipos de relación pueden tener atributos para guardar datos que no se pueden determinar desde un tipo de entidad, es decir, cuando un valor deseado no se puede obtener teniéndolo como atributo de cualquiera de las entidades que intervienen en la relación pero sí al combinarlas o a partir de la relación entre ellas, es conveniente agregar el atributo al tipo de relación, según (Elmasri and Navathe, 2011).

Relación identificativa

Una relación identificativa se da para un tipo de entidad débil, siendo la que lo relaciona con su propietario. En este caso la restricción de participación del tipo de entidad débil es total, pues su existencia depende de su relación identificativa con el tipo de entidad propietario. Debido a que un tipo de entidad débil puede tener más de un tipo de entidad propietario, existen las relaciones identificativas de grado superior a dos.

La relación identificativa se modela mediante un rombo similar al resto de los tipos de relaciones pero rodeado con líneas dobles, según (Elmasri and Navathe, 2011).









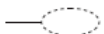
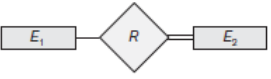

| Symbol | Meaning |
|--|---|
|  | Entity |
|  | Weak Entity |
|  | Relationship |
|  | Identifying Relationship |
|  | Attribute |
|  | Key Attribute |
|  | Multivalued Attribute |
|  | Composite Attribute |
|  | Derived Attribute |
|  | Total Participation of E_2 in R |
|  | Cardinality Ratio 1 : N for E_1, E_2 in R |

Figura 1.2: Notación, fuente: (Elmasri and Navathe, 2011)

1.3. El modelo Entidad-Relación extendido (EER)

El modelo Entidad-Relación extendido surge por la necesidad de incorporar nuevos conceptos a la semántica de modelado de datos capaces de modelar requerimientos más complejos de las aplicaciones. Por tanto, el modelo EER no es más que el modelo ER con nuevos conceptos incluidos.

1.3.1. Conceptos incorporados al modelo ER

Subclase y Superclase

Cuando en el conjunto de entidades de un tipo de entidad se tiene un grupo del cual se necesita tener una representación individual en la aplicación de base de datos tenemos una subclase. Cada una de las entidades incluidas en la subclase conservan las mismas características definidas en su tipo de clase, dígame atributos y los tipos de relación en los que participa, el cual pasa a ser la superclase de la subclase determinada, es decir, cada una de las entidades de la subclase, representan las mismas entidades en el tipo de clase definido como superclase. Por tanto, existe una relación entre una subclase y su superclase denominada relación superclase/subclase o clase/subclase, según (Elmasri and Navathe, 2011).

La existencia de una entidad miembro de una subclase implica que pertenece a una superclase. Agregar que cada entidad de una superclase no tiene que pertenecer

a alguna subclase.

Cuando tenemos una relación superclase/subclase tenemos presente una herencia, porque cada entidad miembro de una subclase hereda los atributos y tipos de relaciones en los que participa su superclase al ser miembro de esta también.

Las subclases también pueden ser consideradas un tipo de entidad, dado que además de las características heredadas de la superclase, puede tener atributos y relaciones propias. Los atributos que son propios de una subclase son conocidos como atributos específicos o locales, y los tipos de relaciones en los que participa se llaman tipos de relación específicos

Especialización

Cuando tenemos un conjunto de subclases de un tipo de entidad estamos frente a una especialización, pues las entidades que las componen tienen características que las distinguen dentro de la superclase y desempeñan un papel concreto como subclase, además pueden tener características adicionales.

El hecho de que las entidades especializadas tengan atributos propios posibilita que las subclases compartan muchos de sus atributos con las entidades de la superclase y a su vez pueda tener atributos que no se pueden aplicar a la superclase.

Los tipos de relaciones específicas de las subclases facilitan el modelado de relaciones que sólo pueden existir entre miembros de la subclase y no entre todos los de la superclase.

Otros aspectos de la especialización, comunes con la generalización posteriormente explicada, son los casos donde tenemos una única subclase, aquí se manifiesta una deferencia en la notación, pues representamos este vínculo solamente uniendo la superclase con la subclase mediante una línea recta y el símbolo de subconjunto sobre ella. También existen las subclases de predicado definido, donde se determinan las entidades que formarán parte de las subclases mediante una condición sobre algunos atributos de la superclase, es decir, las entidades de la superclase que tengan el atributo por el cual se filtra en la condición con el valor especificado en esta, pasan a ser de una subclase determinada y las que no, se quedan en la superclase sin pertenecer a otra subclase a menos que haya una condición que las seleccione. En la figura 1.2 se muestra como modelar estas subclases de condición definida.

Para modelar la especialización se usa una línea recta saliente de la superclase hacia un círculo y otras líneas rectas que parten del círculo a las clases especializadas con el símbolo de subconjunto encima, en sentido de la superclase.

Dentro del círculo va una letra, puede ser *d* u *o*, la *d* representa una restricción de disyunción, que implica que una entidad de la superclase puede formar parte de solo una de las subclases, un ejemplo donde se manifiesta esta restricción es en las subclases de predicado definido donde la condición obliga a que las entidades miembros de una subclase, no pertenezcan a otra. Por otra parte, la *o*, indica que puede existir solapamiento entre las entidades de las subclases, o sea, una misma entidad podría ser miembro de varias subclases al mismo tiempo.

Otra restricción expresada en el modelado es la restricción de integridad, la cual indica la cantidad de entidades de la superclase que serán miembros de las subclases.

Es total cuando cada miembro de la superclase tiene que formar parte de al menos una subclase, esto se modela conectando la superclase con el círculo mediante una línea doble. Cuando podemos tener entidades que no pertenezcan a ninguna subclase, tenemos una especialización parcial y se representa conectando la superclase y el círculo con una línea simple, según (Elmasri and Navathe, 2011).

Generalización

La generalización no es más que tomar las características comunes entre entidades, agrupar esas similitudes en una superclase y que estas entidades hereden de la superclase que generalizó sus características. En los casos donde especializamos subclases a partir de una superclase también tenemos generalización, porque podemos ver la superclase como la entidad o tipo de entidad que generalizó las características de las subclases especializadas.

Las restricciones antes explicadas para la especialización también se aplican a la generalización, pues ambos procesos están implícitos cuando se ejecuta cualquiera. En el caso de la restricción de integridad, debemos tener en cuenta que cuando se crea una superclase a partir de la generalización, el cubrimiento es total, porque la superclase se derivó de las subclases, por tanto las entidades que la integran también están en las subclases, según (Elmasri and Navathe, 2011).

Herencia simple

Podemos tener una especialización donde cualquiera de las subclases o todas tengan subclases, esto implica que esa subclase tiene otras clases especializadas, por tanto, ella pasaría a ser clase generalizadora para sus subclases y al mismo tiempo sigue siendo una subclase respecto a su superclase. En casos así, estamos en presencia de una especialización jerárquica donde la única restricción es que cada subclase tiene solo una clase padre, dicho de otra forma, las subclases van a heredar de una única superclase inmediata. Esta jerarquía de especializaciones es lo que se conoce como herencia simple.

Unión

Cuando necesitamos que una subclase este formada por un subconjunto de entidades pertenecientes a distintos tipos de entidades, es decir, un subconjunto de la UNION de los diferentes tipos de entidades, esa subclase es denominada de tipo unión o categoría. Las entidades de una categoría tienen al menos dos superclases que representan diferentes tipos de entidades, y en dependencia de la superclase a la que pertenezca una entidad determinada de la categoría, heredará los atributos y relaciones correspondientes.

La diferencia de esta relación de UNION respecto a una herencia múltiple está en que en la herencia múltiple las entidades de la subclase compartida heredan los atributos y relaciones de todas las superclases con las que esta enlazada, a diferencia de la UNION donde cada una de las entidades de la subclase pertenecen a la superclase

1, o a la superclase 2, o a la superclase n, y heredan las características de la superclase correspondiente, no las de todas porque representan objetos diferentes.

La restricción de integridad también es aplicada a las categorías, es total cuando la categoría contiene todas las entidades de sus superclases y parcial si solo toma un subconjunto de ellas, según (Elmasri and Navathe, 2011).

1.4. Arquitectura dirigida por modelos

La Arquitectura Dirigida por Modelos (*Model Driven Architecture*(MDA)) fue definida por la OMG (*Object Management Group*) como resultado de la necesidad de integración en todo el ciclo de vida de un sistema para poder transitar desde el modelado de negocios hasta el diseño del sistema, la construcción de componentes, el ensamblado o la implementación. MDA separa los requerimientos funcionales del sistema de los de la implementación de esas funcionalidades en una plataforma específica, para esto, MDA define una arquitectura para modelos que provee guías para estructurar las especificaciones de la aplicación en forma de modelo, según (OMG, 2001).

Por tanto, permite que la semántica esencial de la aplicación a desarrollar tenga un modelo independiente de la plataforma, es decir, formaliza las especificaciones estructurales y funcionales del sistema abstrayéndose de los detalles de implementación, de esta se facilita la migración a otro software intermedio o a nuevas versiones de este mejorando así la portabilidad de las aplicaciones en general.

Brinda gran flexibilidad a los desarrolladores, una arquitectura consistente, cierto grado de generación automática y facilidades para el mantenimiento. Además, las declaraciones formales de la semántica de los sistemas lograda a través del modelado garantizan mayor calidad y durabilidad de los diseños. Estos modelos pueden ser contruidos, vistos y manipulados vía UML y transmitidos a través de XML y almacenados en repositorios *Meta Object Facility* (MOF).

MDA gira entorno a estándares de OMG como MOF, UML, *System Modeling Language* (SysML), *Object Constraint Language* (OCL) y *XML Metadata Interchange* (XMI). Además, clasifica los metamodelos en tres categorías:

1. *Computation Independent Model* (CIM)
2. *Platform Independent Model* (PIM)
3. *Platform Specific Model* (PSM)

Entre los proyectos más activos dentro del MDA están *Eclipse Modeling Framework* (EMF) y *Graphical Modeling Framework* (GMF), que son descritos posteriormente. También existen otros como M2M, M2T, según (Soley, 2000).

1.5. Eclipse

Eclipse es un proyecto de software de código abierto (*open source*) con el propósito general de proveer una plataforma de herramienta altamente integrada. El desarrollo

del trabajo en Eclipse está dividido en varios proyectos de alto nivel como el *Eclipse Project*, *the Modeling Project*, *the Tools Project*, and *the Technology Project*. El *Eclipse Project* contiene el núcleo de componentes necesarios para desarrollar usando Eclipse. Sus componentes son agrupados en una unidad conocida como *Eclipse Software Development Kit* (SDK). El resto de los proyectos de Eclipse son usados para fines específicos. El proyecto usado para desarrollar las tecnologías basadas en modelos es *The Modeling Project*, su núcleo es *Eclipse Modeling Framework* (EMF) que soporta el concepto principal de MDA que usa modelos para el desarrollo y la integración de herramientas. En EMF un modelo es usado para conducir la generación de código y la serialización para el intercambio de datos, según (Steinberg, 2009). Una descripción más detallada sobre EMF se da en la siguiente sección.

1.6. Eclipse Modeling Framework (EMF)

EMF representa el vínculo entre Java, XML y UML, pues permite a partir de cualquiera de estas tres formas definir un modelo a partir del cual se podrán generar las otras, por ejemplo el código en Java para un modelo determinado, según (Steinberg, 2009). El modelo usado para representar modelos en EMF es conocido como Ecore, el cual es en sí un modelo de EMF, por tanto es su propio metamodelo. Aclarar que el Ecore tiene términos similares a UML porque es un subconjunto simplificado y más pequeño del UML.

Un modelo Ecore es construido a partir de cualquiera de las formas de entrada antes mencionadas, dígame interfaces en Java, un esquema en XML o un diagrama en UML, en este último caso puede ser usando un editor gráfico Ecore que provee el *Ecore Tool project*, importándolo desde UML, pues EMF tiene soporte para el Rational Rose, o exportándolo desde UML, en este caso es invocado desde la herramienta UML. Cuando se hace referencia a exportar, se exporta a Ecore XMI. Entonces, tenemos otra forma, que sería la representación primaria o estándar para construir el modelo Ecore, o también representación canónica, el XML *Metadata Interchange* (XMI), que es un estándar para serializar metadatos de forma concisa a partir de XML, según (Steinberg, 2009). Con esta representación a diferencia de la dada por Java, XML o UML, no se guarda información adicional a la recogida en el modelo Ecore, por tanto constituye una serialización directa de Ecore.

Clases del Ecore:

1. EClass. Representa una clase del modelo. Las clases son identificadas por su nombre pueden tener varios atributos y referencias. Como soporte para la herencia una clase puede referirse a otras clases como sus supertipos. Representa un concepto del dominio.
2. EAttribute. Representa un atributo del modelo. Tiene un nombre y tipo.
3. EReference. Representa uno de los extremos de una asociación entre clases. Tiene un nombre y un tipo, en este caso el tipo es de tipo EClass.

4. EDataType. Usado para representar el tipo de los atributos, puede ser un tipo de dato primitivo o de tipo objeto definido en Java.

Con el modelo generado con EMF queda representado el modelo de dominio de la aplicación.

1.7. Graphical Modeling Framework (GMF)

GMF es un marco de trabajo para la creación de editores gráficos para el modelado. Se divide en dos componentes principales the tooling y el entorno de ejecución. “*The tooling*” consiste en editores para crear o editar modelos describiendo la notación, la semántica y aspectos del editor gráfico, además de un generador para la implementación de los editores gráficos. Los plug-ins generados dependen del entorno de ejecución de GMF, según(Eclipsepedia, 2006).

Con el uso de GMF pueden crearse herramientas gráficas para el modelado a partir de la creación de un modelo de dominio que define la información no gráfica que manejará el editor, luego requiere crear un modelo de definición que define los elementos gráficos que serán mostrados, además, hay que crear un modelo de mapeo, que define como se mapearán los elementos del modelo de dominio y los elementos gráficos. Con estos recursos creados queda generar el editor gráfico y GMF provee un modelo generador que permite la definición de algunos detalles de implementación para la fase de generación.

GMF permite que la definición gráfica pueda ser reutilizada para varios dominios y esto se logra usando un modelo de mapeo separado para enlazar las definiciones gráficas con el conjunto de herramientas gráficas definido para el dominio seleccionado.

El modelo de domino necesario para generar el editor gráfico mediante GMF es modelado con EMF, por tanto, el vínculo entre GMF y EMF se torna fundamental para el desarrollo de proyectos con GMF.

Por tanto, la generación del editor gráfico, además de tener dependencia con EMF, se vincula con el GMF *Runtime*, GEF y la plataforma de Eclipse. GMF provee un puente entre EMF y GEF. .

Algunas características de los editores generados con GMF sobre su entorno de ejecución según (Frederic Plante, IBM, 2006)son:

- *Collapsed and Expanded Compartment*. Las figuras compuestas o con compartimientos(*compartment*) pueden ser expandidas o resumidas (*collapsed*).

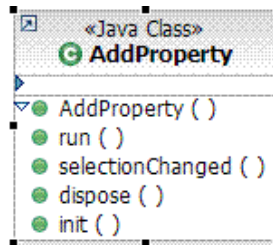


Figura 1.3: Attribute Compartment Collapsed.

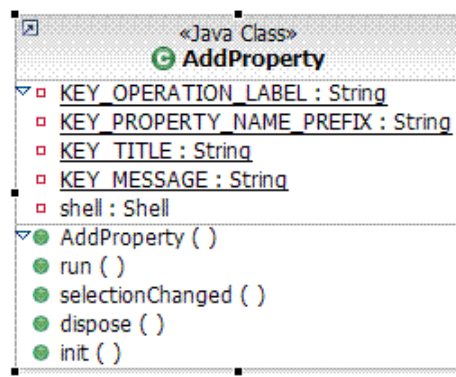


Figura 1.4: Attribute Compartment Expanded.

- *Direct Editing.* El texto en las etiquetas puede ser editado directamente o interpretado para la ejecución de comandos, por ejemplo se puede añadir una operación como se muestra en la figura.

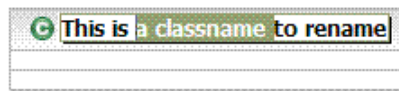


Figura 1.5: Edición del nombre de una clase.

- *Diagram Assistants.* Los diagramas asistentes son objetos gráficos que se muestran sobre un diagrama dada la ubicación del mouse en ese lugar sin actividad un tiempo determinado. Sus capacidades y comportamiento depende del objeto bajo el mouse. Hay disponibles dos tipos de diagramas asistentes: *Pop-up and Connection Handles*.



Figura 1.6: Pop-up.

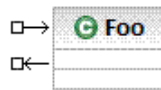


Figura 1.7: Connection Handles.

- *Common Tools.* EL editor gráfico tendrá su propia paleta donde algunas herramientas estarán disponibles, como la herramienta de notas (*Note tool*), que permite la creación de cajas de texto en el diagrama para notas, la herramienta de notas adjuntas (*Note Attachment tool*) que permite conectar las notas entre sí y con elementos del diagramas, así como modificar su forma, entre otras como la herramienta de zoom.

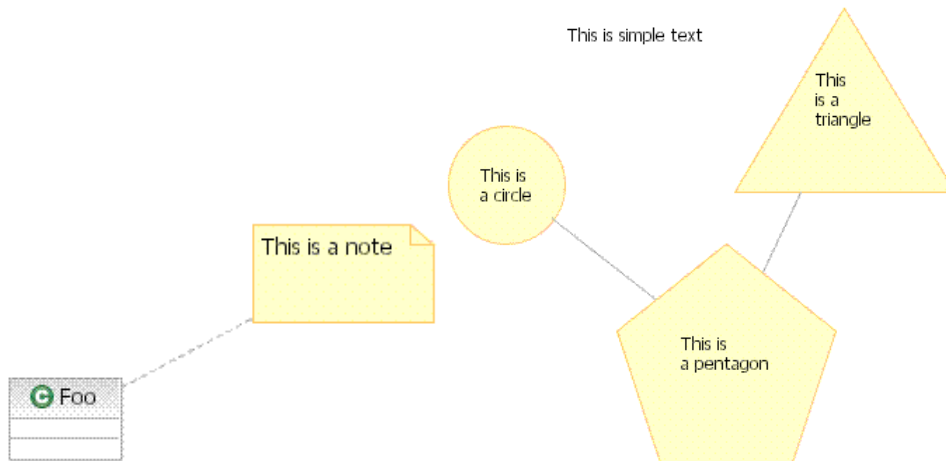


Figura 1.8: Note tool and Note Attachment tool

- *Common Menus Commands.* El GMF Runtime habilita varios digramas estándares relacionados con los comandos del menú como un *Font menu*, el *Fill color* y el *Line Color*, el *Line Style*, un menú Select para seleccionar elementos del diagrama, un *Auto Size*, *View* entre otros.

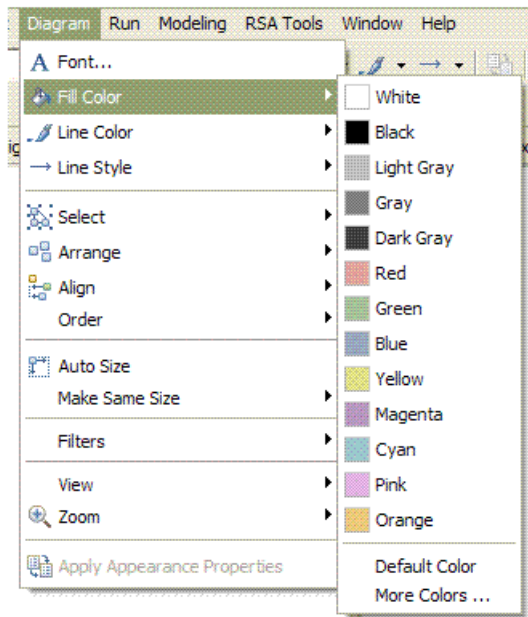


Figura 1.9: Common Menus Commands

- *Common Toolbars*. Habilita controles en una barra de herramientas como soporte a los comandos accesibles desde la barra de menú.



Figura 1.10: Common Toolbars

El entorno de ejecución de GMF aporta la posibilidad de reutilizar componentes estandarizados y de gran utilidad para los editores gráficos de modelado. Además, permite la definición de un metamodelo con total separación entre la semántica y la notación.

En este capítulo se abordaron los principales conceptos que caracterizan el modelo de datos de alto nivel EER, dado que se pretende construir un metamodelo que lo describa como lenguaje de modelado. Se presentaron las tecnologías a emplear para la implementación de dicho metamodelo, las cuales forman parte de las herramientas que brinda la MDA. El marco de trabajo para el desarrollo de aplicaciones mediante modelos que brinda Eclipse es una herramienta prometedora para la obtención de un metamodelo que describa un lenguaje de modelado determinado, pues a partir de esta representación en un modelo Ecore, puede obtenerse el código en Java correspondiente o un fichero XMI para el intercambio de los datos e incluso, con el uso de GMF que forma parte del proyecto de modelado de Eclipse, se puede obtener un editor gráfico donde pueda emplearse el lenguaje representado en el Ecore para el diseño de esquemas conceptuales.

Capítulo 2

Implementación de un metamodelo para el modelo ER y EER

En este capítulo inicialmente se analizan algunos metamodelos para el modelo EER que se han realizado, con el fin de identificar algunas de sus fortalezas y limitantes. Además, se describe el esquema conceptual propuesto que representa los elementos del modelo EER y su implementación usando EMF siguiendo un esquema nodo-relación para su futura descripción usando metáforas gráficas.

2.1. Otros metamodelos

Uno de los problemas de integración que enfrenta el almacenamiento de datos es la gerencia e integración de metadatos, que implica el uso de varias herramientas y productos, cada uno con su propia definición y formato para estos, por lo que su creación, el intercambio y gerencia requiere de tiempo y están propensos a errores. Debido a que cada herramienta de gestión y análisis de datos requiere diferentes metadatos y metamodelos para resolver los problemas de almacenamiento, no es posible tener solo un repositorio de metadatos que implemente solo un metamodelo para todos los metadatos de una organización. Ante esta problemática la OMG propuso iniciativas para estandarizar la representación de metadatos en el entorno del almacenamiento de datos (*data warehouse*) según (Chang, 2000). El *Common Warehouse Metamodel* (CWM) está compuesto por un conjunto de sub-metamodelos que representan el almacenamiento común de metadatos en las áreas de mayor relevancia del almacenamiento de datos (*data warehousing*) y la inteligencia para negocios (*business intelligence*) dígase según (OMG, 2003):

- Recursos de datos (*data resources*), que incluye metamodelos para representar recursos de datos orientados a objetos, relacionales, multidimensionales y XML.
- Análisis de datos (*data analysis*), incluyendo metamodelos que representen transformaciones de datos, OLAP (*On-line Analytical Processing*), nomenclatura de negocios, minería de datos y otros.

- Gestión de almacenamiento (*Warehouse Management*) que abarca metamodelos para procesar el almacenamiento (*warehouse process*) y resultados de las operaciones de almacenamiento.

Los elementos del metamodelo CWM se encuentran agrupados en paquetes, el paquete llamado *Object Model* está basado en el metamodelo de UML excluyendo aquellos aspectos que no son relevantes en el dominio del almacenamiento de datos. Por tanto, CWM hereda de UML, incluso, la notación de UML es usada para la representación diagramática de CWM y las restricciones adicionales que incluye CWM están representadas en *Object Constraint Language* (OCL) como define la especificación UML. Esto es posible debido a que UML define un mecanismo de extensión que permite la especificación de lenguajes especializados basados en UML.

Muchos de los usos del CWM van más allá de la creación y gestión de los almacenes de datos, por lo que el *Information Management Metamodel* (IMM) es otro estándar de la OMG que incluye lo alcanzado por el CWM y se centra en facilitar el modelado de información a partir de la estandarización de los conceptos y notaciones tradicionales de modelado, así como permitir la trazabilidad de los términos comerciales a los modelos orientados a objetos, modelos de datos, esquemas XML, Ontologías y otros. Además encaja con otros estándares relacionados con la gestión de la información como *Ontology Definition Metamodel* (ODM) según (Rivett, 2011).

CWM e IMM proponen metamodelos para el modelo EER, el metamodelo de CWM contiene clases para representar los conceptos de Entidad, Atributo y Relación, tiene como limitante la no inclusión del modelado de clases o atributos para entidades y relaciones débiles, ni atributos compuestos o unión. La representación de otras características del modelo EER como Entidades Asociativas, Atributos multivalor y derivados, y las relaciones con atributos es posible a través de la notación de UML, por ejemplo, un atributo multivalor puede representarse usando corchetes. Además, elementos como la razón de cardinalidad de las relaciones, la restricción de participación, los roles, herencia simple, relaciones n-arias y restricción de disyunción no están implícitos en el metamodelo CWM, pero están definidos en el metamodelo de UML. Otra limitante de CWM es que no define ninguna restricción OCL en su metamodelo y al heredar de UML que requiere estas restricciones para el diseño de base de datos, también debe contar con algunas en orden de evitar la introducción de elementos del UML innecesarios en ese dominio que podrían causar errores. Además, tener en cuenta que CWM define una clase llave foránea (*ForeignKey*) la cual no forma parte de los elementos del modelo EER. Ver figura 2.1.

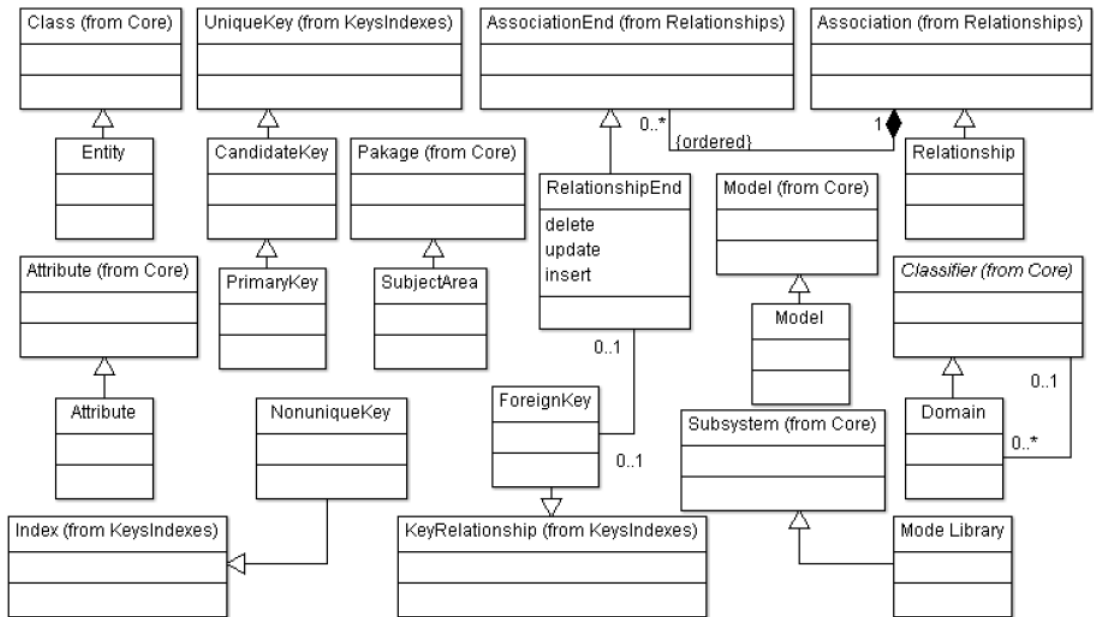


Figura 2.1: Metamodelo de CWM para el modelo EER, fuente: (Fidalgo et al., 2013)

En el caso del metamodelo de IMM para el modelo EER, no hereda de UML como el metamodelo CWM ni incluye elementos que no forman parte del modelo EER y como limitante, no brinda una representación para elementos como las Entidades Asociativas, Unión, Atributos Llave, Compuestos, Multivalor, relaciones con atributos, especialización. Ver figura 2.2.

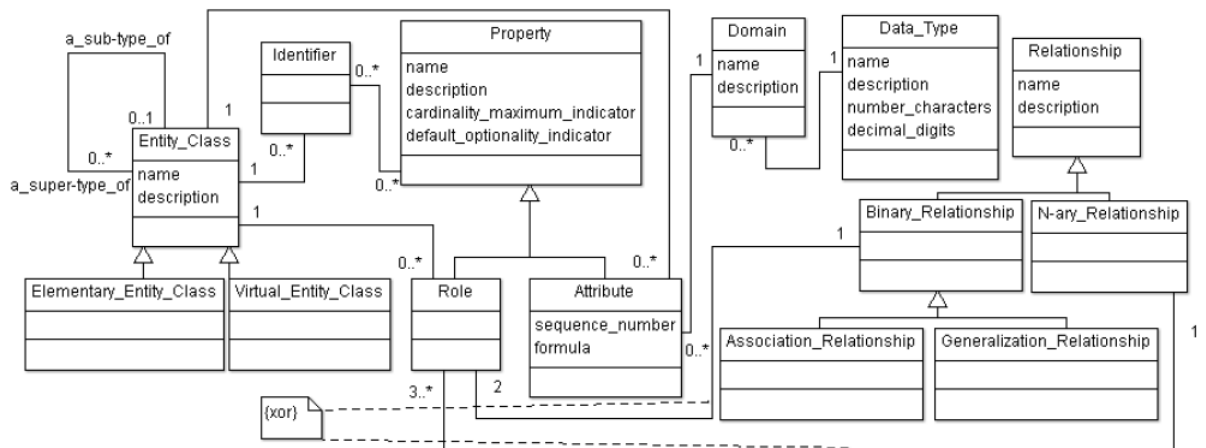


Figura 2.2: Metamodelo de IMM para el modelo EER, fuente: (Fidalgo et al., 2013)

Otro metamodelo que describe el modelo EER es el propuesto en (Fidalgo et al., 2013). Este metamodelo incluye los conceptos fundamentales del modelo EER y es modelado siguiendo un esquema nodo-relación para permitir su implementación usan-

do EMF y a partir del modelo Ecore obtenido generar el código correspondiente para un editor gráfico mediante el uso de GMF. Puede representar entidades, débiles o no, entidades asociativas, atributos y sus diferentes clasificaciones, relaciones, herencia simple, unión, es decir, están implícitos los principales conceptos que describen el modelo EER, sin embargo debido a su forma de generalizarlos pierde algunas restricciones del modelo, por ejemplo las entidades débiles se pueden ver representadas en el modelo Ecore (ver figura 2.3) como un atributo booleano en la Eclass que representa las Entidades y según (Elmasri and Navathe, 2011) una entidad débil es una entidad que depende de una entidad fuerte mediante una relación identificativa. Además al omitir la representación de las entidades débiles como entidades, queda excluido el modelado de sus relaciones identificativas y de el atributo llave parcial necesario para identificarlas en conjunto con el atributo llave de su clase propietaria (ver subseccion 1.2.1 en página 4), así como las restricciones referentes a este tipo de entidades.

Este metamodelo posibilita representar atributos en las relaciones, necesidad que surge a causa de la imposibilidad de determinar el valor de ese atributo en cualquiera de las entidades que intervenga en la relación pero sí al relacionarlas, según (Elmasri and Navathe, 2011). Por tanto, no cumple objetivo alguno que un atributo de una relación sea de tipo llave pues los tipos de relaciones no necesitan atributos que los identifiquen y el metamodelo propuesto por (Fidalgo et al., 2013) no restringe este hecho.

El metamodelo de (Fidalgo et al., 2013) generaliza de una forma muy abarcadora los conceptos del modelo EER pero al hacerlo así, excluye algunas restricciones que implica, pero hasta el momento estos conceptos no han sido estandarizados por tanto su metamodelo es un avance en este dominio pues describe muchos elementos faltantes en los metamodelos antes propuestos, el CWM y IMM de OMG.

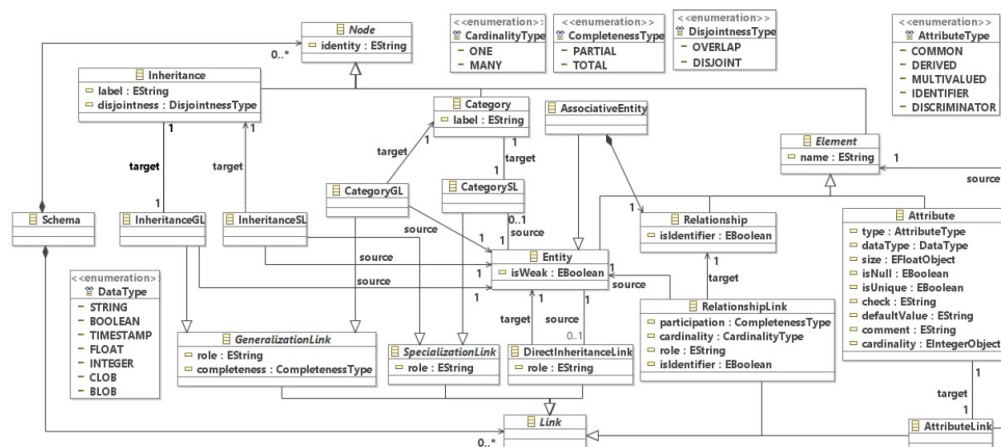


Figura 2.3: Metamodelo en Ecore de Robson Fidalgo y otros, fuente: (Fidalgo et al., 2013)

2.2. Modelado del esquema conceptual

Los elementos fundamentales del modelo EER expresados en el modelo del dominio son:

1. Entidades
2. Atributos
3. Relaciones

Los tipos de entidades representados son:

1. Entidades regulares
2. Entidades débiles
3. Entidades asociativas

Los tipos de relaciones modeladas son:

1. Identificación
2. Unión
3. Herencia
4. Relaciones Binarias
5. Relaciones n-arias

Una entidad tiene un atributo nombre, de ella heredan los tipos de entidades débiles, fuertes y las entidades asociativas y tiene relaciones de asociación con los diferentes tipos de relación modelados. Además, se relaciona mediante una agregación con la entidad Atributo, dado que una entidad puede estar compuesta por uno o varios atributos.

En la descripción de los atributos se diferencia entre los atributos que pueden ser clave, clave parcial o ninguna de estas categorías. Los atributos tienen un tipo, nombre y su valor, para representar sus diferentes clasificaciones, se utiliza una categorización, además, así se garantiza que los atributos de las relaciones no sean de tipo llave o tipo llave parcial.

Las relaciones son modeladas mediante la jerarquía representada por una entidad llamada Relación siendo el nombre de la relación su atributo, de la cual heredan los diferentes tipos de relaciones que son a su vez entidades. En el caso de la relación de herencia, se modela la restricción de disyunción y de participación mediante dos atributos pertenecientes a la entidad representativa de la herencia. La posibilidad de tener solo una clase padre y una o varias subclases que hereden sus características es el concepto que define la herencia simple y es modelado en el esquema conceptual mediante dos relaciones entre la entidad “Herencia” y “Entidad”, una indica la existencia de una única entidad padre y otra los posibles hijos.

De forma análoga se modela la relación de identificación y unión, en cada caso se tienen dos relaciones entre la entidad representante del tipo de relación a modelar y el tipo entidad. Para la relación de unión se representa mediante la razón de cardinalidad la existencia de más de una superclase y una subclase que agrupa las diferentes entidades de los diferentes tipos de entidades que la especializan.

En el modelado de la relación de identificación, a diferencia de los tipos de relaciones antes explicados, se tiene una relación entre la entidad representante de la relación de identificación y un tipo de entidad débil, de esta forma se garantiza la presencia de una entidad débil en una relación de identificación. La segunda relación es entre la entidad “Identificación” y un tipo de entidad general, representando la presencia de una o más entidades propietarias en la relación de identificación.

Las relaciones binarias también tienen una entidad en el esquema conceptual que las describe, igualmente, con dos relaciones que parten de ella a un tipo de entidad general, estas relaciones representan la existencia de dos tipos de entidades relacionados. El rol, la cardinalidad mínima y máxima de cada tipo de entidad participante en la relación se representan como atributos de las relaciones que las conectan con la entidad representativa de la relación binaria en el esquema.

La entidad que describe los tipos de relaciones n-arios hereda de la entidad “Relación Binaria”, de esta forma se tiene una relación n-aria a partir de una binaria, es decir, cuando intervienen en la relación más de dos entidades. De forma similar a los tipos de relaciones binarias, los n-arios modelan el rol y las cardinalidades mínimas y máximas como atributos de la relación que los representa.

La notación diagramática usada para representar el modelo del dominio para el caso del modelo EER es la definida por Elmasri y Navathe para el mismo.

Respecto a los metamodelos anteriores, el actual supera las limitantes de CWM al no heredar del UML o representar elementos que no formen parte del modelo EER, además describe la mayoría de los elementos del modelo EER, incluidos los que se omiten en los metamodelos de CWM e IMM, excepto la herencia múltiple que no se representó debido a que no todos los lenguajes de programación la implementan. Además, respecto al metamodelo propuesto por (Fidalgo et al., 2013), sí modela el tipo de entidad débil como una entidad, así como las relaciones identificativas y las restricciones respecto a su llave parcial (*ver subsección 1.2.1 en página 4*).

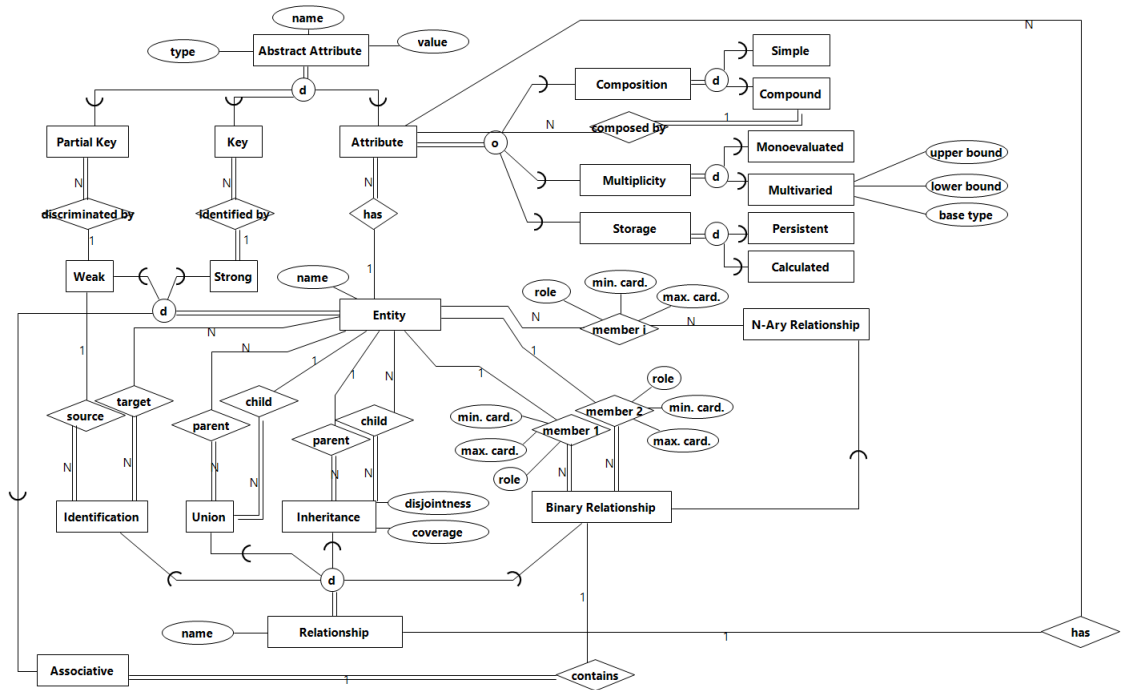


Figura 2.4: Esquema Conceptual

2.3. Implementación del metamodelo en EMF

En orden de poder utilizar el marco de trabajo de Eclipse para la construcción de una herramienta de modelado a partir de un metamodelo definido con EMF se debe seguir un esquema nodo-relación, por lo que la representación de los elementos del esquema conceptual deben seguir una jerarquía de esta forma, según (Steinberg, 2009).

Respecto al esquema conceptual propuesto en la subsección anterior, se hicieron cambios en la representación del metamodelo implementada con EMF, cambios que mantienen la semántica del modelo EER a describir, pero se adecúan a los requerimientos de diseño del modelo Ecore donde se implementó.

Se conforma una jerarquía donde los elementos fundamentales del modelo EER, dígame entidades, atributos y relaciones heredan de una clase *Node*, y el vínculo entre estos elementos se modela mediante clases de tipo *Link*. La clase *AttributeLink*, que representa la relación entre un atributo y la entidad a la que está asociado o su vínculo con una relación si pertenece a ella. La clase *AttributeCompositionLink*, representa los vínculos que puede tener un atributo compuesto, es decir, un atributo compuesto es un atributo que puede tener uno o varios atributos, por tanto, esta clase *Link* restringe este hecho. Las clases *IdentifiedByLink* y *DiscriminatedByLink* modelan la restricción de que las entidades débiles tienen una llave parcial y las entidades regulares(fuertes) tienen una llave que las identifican respectivamente, según Elmasri and Navathe (2011).

RelationshipLink describe la relación entre una entidad y una relación, es decir,

cuando se quiere modelar una relación binaria o n-aria entre entidades, cada tipo de entidad que interviene tiene un vínculo con el tipo de relación y en ese vínculo se guarda la información referente a la razón de cardinalidad, rol y a la restricción de disyunción y de participación por ese extremo de la relación.

La clase *DirectInheritanceLink* permite modelar una relación de herencia directa entre dos entidades, o sea, representa el enlace entre una entidad que hereda de otra y en él se guarda el rol de la relación.

Las clases *GeneralizationLink* y *SpecializationLink* modelan los vínculos entre las entidades en relaciones de herencia simple y Unión, en el caso de la herencia simple se tiene una clase generalizadora que especializa a varias subclases, opuesta a la unión donde una entidad representa la UNIÓN de varios tipos de entidades, *ver sección 1.3.1 en página 11*.

IdentifiedSideLink representa el enlace, en las relaciones de identificación, entre una entidad débil por un extremo y la entidad propietaria por otro, consultar restricciones sobre las relaciones de identificación en página 8.

Para modelar el solapamiento entre las diferentes categorías de atributos, dígame simple o compuesto, monoevaluado o multivalor y persistente o derivado, de forma tal que no se pierda la semántica del modelo EER, se separaron estas clasificaciones en clases y en clases de tipo enumeración como se muestra en la *Figura 2.5*.

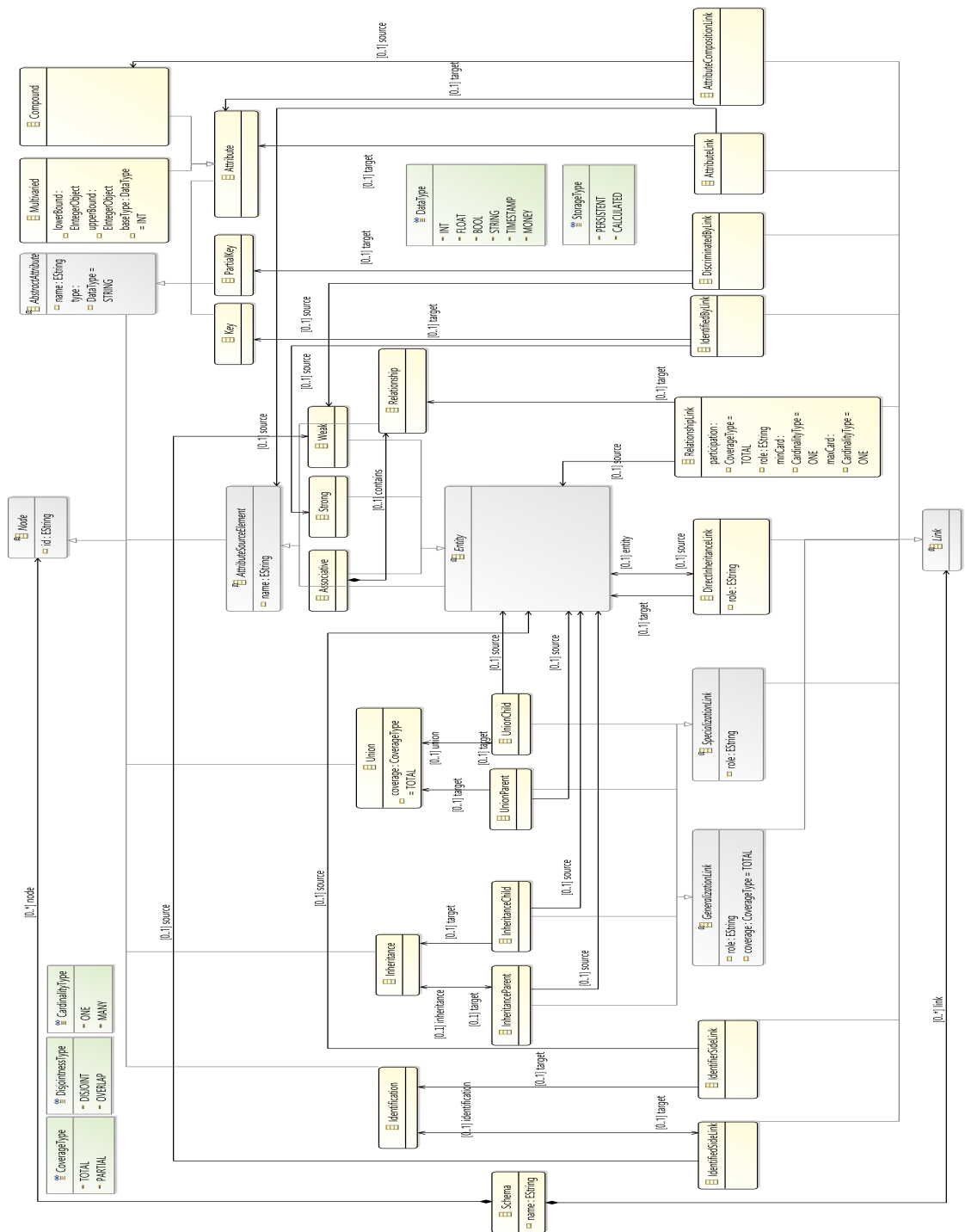


Figura 2.5: Metamodelo

En este capítulo se analizó el trabajo existente de otros autores referente al meta-modelado del modelo EER para analizar algunas de las limitantes y ventajas presentadas hasta el momento. Se propuso un esquema conceptual que describe el modelo EER donde quedaron reflejados sus principales conceptos. Además, se explicó la forma de implementación del metamodelo deseado, donde se mantuvo la semántica del modelo EER representándolo mediante una jerarquía nodo-relación requerida para el modelado mediante EMF. El metamodelo obtenido supera algunas de las limitantes presentes en los metamodelos de CWM e IMM y respecto al propuesto por (Fidalgo et al., 2013), cubre algunas restricciones que son generalizadas en este último. El modelo Ecore obtenido brinda una definición del lenguaje de modelado EER que cumple con muchas de sus restricciones más importantes. Además, puede obtenerse el código en Java correspondiente y facilita el intercambio al poder generar una versión del proyecto en XMI facilidad que brinda EMF en cuanto a interoperabilidad.

Conclusiones

1. La síntesis de los conceptos del modelo EER, así como el análisis de las tecnologías que implementan la arquitectura dirigida por modelos, constituyeron la base del actual estudio y posibilitaron el diseño del metamodelo deseado. Es notable la importancia que ha cobrado el desarrollo de proyectos de software basados en una arquitectura dirigida por modelos, de esta forma se obtienen las especificaciones de una aplicación en forma de modelo dejando los requerimientos funcionales independientes de la implementación que se les pueda dar en una plataforma específica. El marco de trabajo para el modelado que brinda Eclipse aporta gran interoperabilidad al proyecto emprendido, debido a que a partir de un modelo Ecore, puede obtenerse el código en Java correspondiente y un fichero XMI para el intercambio. Por tal motivo, se seleccionó para la implementación del metamodelo que describe el modelo EER.
2. La obtención de un esquema conceptual que describe el modelo EER es el paso inicial para la implementación del metamodelo usando el marco de trabajo que brinda Eclipse para el modelado. El esquema conceptual propuesto abarca los principales conceptos del modelo EER y sus restricciones. Respecto a trabajos previos que lo han generalizado mediante el metamodelado, el esquema conceptual propuesto supera algunas de las limitantes presentadas referentes a elementos y restricciones del modelo EER que no quedaban recogidas de forma implícita en el metamodelo. Los elementos que modela el esquema conceptual propuesto son solamente los que caracterizan el modelo EER y no se incluyen elementos ajenos para facilitar el modelado.
3. La implementación propuesta del metamodelo está representada a través del modelo Ecore que brinda EMF. El metamodelo sigue una jerarquía nodo-relación requerida por la herramienta pero mantiene la semántica del modelo EER. Siendo actualmente la interoperabilidad un factor importante en las herramientas para el desarrollo de software, a partir del Ecore resultante, se puede obtener una versión del proyecto en XMI para el intercambio, así como el código en Java correspondiente. Además, se puede integrar el uso de GMF para la obtención de editores gráficos de modelado de esquemas conceptuales mediante el lenguaje de modelado definido en el Ecore. La obtención de un metamodelo que describa el modelo EER constituye un paso hacia la estandarización. Este modelo cuenta con varias notaciones diagramáticas y al ser unos de los lenguajes de modelado de alto nivel más populares por su simplicidad y precisión, se torna necesario contar con una definición que lo describa independientemente de la forma en que se modele con él.

Recomendaciones

Se recomienda la generación de una definición gráfica para el metamodelo construido, permitiendo la elección de diferentes representaciones diagramáticas.

Bibliografía

- Chang, D. D. T. (2000). Common Warehouse Metamodel (CWM), UML and XML. page 56. 2.1
- Eclipsepedia (2006). Graphical Modeling Framework/Tutorial/Part 1 - Eclipsepedia. 1.7
- Elmasri, R. and Navathe, S. (2011). *Fundamentals of database systems*. Addison-Wesley, Boston, 6th ed edition. OCLC: ocn586123196. 1.2, 1.2.1, 1.2.2, 1.1, 1.2.3, 1.2.3, 1.2.3, 1.2.3, 1.2.3, 1.2, 1.3.1, 1.3.1, 1.3.1, 1.3.1, 2.1, 2.3
- Fidalgo, R. N., Alves, E., España, S., Castro, J., and Pastor, O. (2013). Metamodeling the Enhanced Entity-Relationship Model. 4(3):15. 2.1, 2.2, 2.1, 2.3, 2.2, 2.3
- Frederic Plante, IBM (2006). Introducing the GMF Runtime. 1.7
- OMG (2001). OMG Document – ormsc/01-07-01 (Model Driven Architecture (MDA)). 1.4
- OMG (2003). cwm_1_book.book - <https://www.omg.org/spec/CWM/1.1/PDF/>. 2.1
- Rivett, P. (2011). Information Management Metamodel. page 22. 2.1
- Soley, R. (2000). Model Driven Architecture. *Model Driven Architecture*, page 12. 1.4
- Steinberg, D., editor (2009). *EMF: Eclipse Modeling Framework*. The eclipse series. Addison-Wesley, Upper Saddle River, NJ, 2nd ed., rev. and updated edition. 1.5, 1.6, 2.3