COLECCIÓN

SISTEMAS DE INFORMACIÓN

1

UMET
UNIVERSIDAD METROPOLITANA

MODELACIÓN DE DATOS

UN ENFOQUE SISTÉMICO

CARLOS ERNESTO GARCÍA GONZÁLEZ
LUISA MANUELA GONZÁLEZ GONZÁLEZ
ABEL RODRÍGUEZ MORFFI
BEATRIZ EUGENIA LÓPEZ PORRERO

MODELACIÓN DE DATOS

UN ENFOQUE SISTÉMICO

CARLOS ERNESTO GARCÍA GONZÁLEZ
LUISA MANUELA GONZÁLEZ GONZÁLEZ
ABEL RODRÍGUEZ MORFFI
BEATRIZ EUGENIA LÓPEZ PORRERO

SISTEMAS DE INFORMACIÓN

Con el auspicio de la Fundación Metropolitana

Proyecto Observatorio Metropolitano de Inteligencia Competitiva, Ciencia, Tecnología, Innovación y Saberes. Red UMET - UCf- UCLV

MODELACIÓN DE DATOS

UN ENFOQUE SISTÉMICO

CARLOS ERNESTO GARCÍA GONZÁLEZ
LUISA MANUELA GONZÁLEZ GONZÁLEZ
ABEL RODRÍGUEZ MORFFI
BEATRIZ EUGENIA LÓPEZ PORRERO

Diseño de carátula: D. I. Yunisley Bruno Díaz

Composición de textos: D. I. Yunisley Bruno Díaz

Corrección: MSc. Alicia Martínez León

Dirección editorial: Dr. C. Jorge Luis León González

ISBN: 978-959-257-499-1

Con el auspicio de la Fundación Metropolitana

Editorial: “Universo Sur”.

Universidad de Cienfuegos. Carretera a Rodas, Km 3 ½.

Cuatro Caminos. Cienfuegos. Cuba.

CP: 59430

E-mail: eus@ucf.edu.cu

Introducción

La primera aproximación para describir la estructura de los datos para ser manipulada por un Sistema de Gestión de Bases de Datos (SGBD) se conduce con la ayuda de algún modelo conceptual, que deben ser naturales o al menos comprensibles para los usuarios del universo de discurso, término usado con mucha frecuencia para delimitar un área del mundo real que es objeto de modelación.

Estos modelos deben ofrecer una variedad de conceptos de modelación con énfasis en las interrelaciones lógicas entre los datos. El objetivo de tal modelación es encontrar los conceptos esenciales presentes en la fase de análisis de requerimientos. Esta tarea no es trivial pues a veces los usuarios describen un mismo concepto con términos diferentes (sinónimos) o usan un mismo término para conceptos diferentes (homónimos) (García & Montes de Oca, 2015). El resultado del diseño conceptual es el *esquema conceptual* de la base de datos, el cual es una descripción de alto nivel de la estructura de la base de datos, que es un conjunto estático de representaciones lingüísticas y gráficas, invariables en el tiempo, que describen la estructura de los datos (Batini, Ceri & Navathe, 2011).

Relación (ER, del inglés Entity Relationship)[1] (Chen, 1976). Los conceptos que le sirven de base a este modelo son los de entidad e interrelación. La decisión de si un concepto del universo de discurso debe ser una entidad, una interrelación, o simplemente una propiedad de cualquiera de ellos, no siempre es obvia y en efecto, puede existir más de una descripción conceptual que refleje los requerimientos de los usuarios.

A partir del modelo ER original presentado por Chen (1976) muchos investigadores han propuesto numerosas extensiones con el objetivo de aumentar su poder de expresión (Teorey, Yang & Fry, 1986; Chen, 2006; Murthy, Delcambre & Maier, 2006; Combi, Degan & Jensen, 2008; Elmasri, Weeldrey & Hevner, 2012). Estas

1 Se asume una ligera imprecisión en su denominación en español (Entidad-Interrelación) para conservar la sigla ER tan frecuentemente citada.

extensiones se han ido incorporando a un supuesto modelo ER abstracto, que no siempre es el modelo original de Chen; no es una tarea sencilla unificar todas estas extensiones en un solo modelo, genéricamente denominado modelo Entidad Relación Extendido (EER, acrónimo del inglés Extended Entity Relationship).

Varias de las extensiones del modelo ER original persiguen mayor riqueza semántica en la descripción del universo de discurso. Detrás de las diferencias sintácticas de estas extensiones está el enriquecimiento semántico de las construcciones. Muchos de los cambios sintácticos propuestos están relacionados con la abstracción de generalización y en menor grado, con la inclusión de nuevas construcciones. También existen diferencias en cuanto a las notaciones empleadas por diferentes autores en la representación gráfica utilizada en los diagramas (Song, Evan, & Park, 1995; Hay, 1999), para lo cual la notación de Elmasri & Navathe (2012) se considera un referente adecuado.

Un concepto tratado con poca formalidad, con un alcance no bien definido, y con enfoques acotados por cada autor, es el de dependencia de existencia. Este concepto se ha asociado intuitivamente a un comportamiento, se ha considerado que existe dependencia de existencia entre dos entidades cuando la eliminación de una entidad dominante conlleva a la eliminación de la entidad subordinada. Este comportamiento es inherente a otras construcciones; algunas de ellas tratadas de manera sutil o informal en la literatura, pero en otros casos, como es su presencia en algunos tipos de interrelaciones de asociación, no ha sido abordado en las extensiones al modelo ER, y sin embargo, su consideración ayuda a modelar muchos hechos asociados con su inmutabilidad en el tiempo.

Es bien conocido que el modelo ER al ser un modelo semántico tiene por objetivo la descripción de un universo de discurso mediante la distinción de hechos cada vez más singulares. Ello se logra con sutilezas en construcciones que describan la mayor diversidad posible de tipos de hechos, que si bien desde el punto de vista formal pueden tener comportamientos similares,

desde el punto de vista de la descripción del universo de discurso puedan expresar dicha singularidad semántica.

En este libro se reconoce la necesidad de una sistematización de construcciones, notaciones, conceptos y del comportamiento embebido en los mismos, con atención especial a las diferentes variantes de dependencia de existencia, todo lo cual ha motivado su inclusión en una herramienta de ayuda al diseño de bases de datos.

Por otra parte, se ha desarrollado un número creciente de herramientas CASE (acrónimo del inglés Computer Aided Software Engineering) con el objetivo de apoyar a los diseñadores en el proceso de diseño de base de datos. Estas herramientas se caracterizan por una interfaz de usuario que da soporte al modelo conceptual y por un diccionario de datos donde se almacenan los esquemas resultantes. Entre sus beneficios están automatizar el proceso de obtención de un esquema de base de datos; minimizar los costos y el tiempo de implementación; utilizar metodologías y modelos de datos consolidados; contribuir a eliminar errores que el diseñador pueda cometer en las distintas fases del diseño, entre otros aspectos.

Una brecha identificada está relacionada con las capacidades de validación de esquemas de estas herramientas, que en opinión de Piattini & Díaz (2012a), es la más poderosa ayuda que las herramientas CASE deben proveer, ya que tienen un impacto directo en la calidad de los esquemas obtenidos durante el proceso de diseño.

Idealmente, el proceso de creación de un esquema ER mediante una herramienta CASE, requiere de un soporte mínimo en términos de la validación de los diferentes esquemas, que comprende la validación de la sintaxis, la estructura y la semántica del esquema conceptual, así como la verificación de posibles inconsistencias en el esquema lógico.

La validación estructural debe comprobar que el conjunto de restricciones de cardinalidad asociado a un esquema conceptual

pueda ser satisfecho de manera total, en cuyo caso el esquema se considera consistente. La validación semántica determina si los tipos de entidades y tipos de interrelaciones del esquema conceptual representan exactamente los conceptos del dominio que se esté modelando (Kolapalli & Dullea, 2012; Dullea, Song & Lamprou, 2003).

Con relación a la detección de inconsistencias en esquemas lógicos como parte integrada al proceso de modelación conceptual, no se encontraron referencias en la literatura consultada. Tradicionalmente la detección de anomalías en esquemas lógicos ha sido abordada con un enfoque bottom-up, que comprueba las propiedades de cada esquema por separado, conocido como Teoría de la Normalización, y que fue tratado incluso, previo al surgimiento del modelo ER (Codd, 1970; Ullman, 1988; Maier, 2011).

En este trabajo se valoró un grupo amplio de herramientas CASE comerciales para comprobar sus capacidades de validación. Como resultado del estudio se pudo constatar que todas las herramientas realizan de una manera u otra la validación sintáctica del esquema ER; sin embargo, un número reducido herramientas ofrece de forma parcial capacidades de validación estructural; todas carecen de soporte de validación semántica y lógica.

El libro ha sido estructurado de la siguiente manera:

En el primer capítulo se realiza una evaluación del marco teórico relacionado con la modelación conceptual de bases de datos y se analiza cada una de las construcciones del modelo ER y sus extensiones, haciendo notar cómo se manifiesta la dependencia de existencia en estas construcciones. También se efectúa un análisis crítico de los métodos de validación de esquemas conceptuales, así como de las herramientas de ayuda al diseño de bases de datos.

En el segundo capítulo se hace una sistematización del concepto de dependencia de existencia, se establece el comportamiento correspondiente de las entidades con dependencia de

existencia en cada una de las construcciones del modelo EER. Como resultado del análisis de la semántica de las interrelaciones de asociación con dependencia de existencia se propone una nueva construcción.

En el tercer capítulo se propone un algoritmo para detectar y corregir un tipo de inconsistencia que puede manifestarse en el esquema lógico de una base de datos, lo cual contribuye a eliminar errores en la fase de implementación. En el cuarto, se elabora una descripción de la arquitectura de la herramienta ERECASE, se destacan las principales funcionalidades que la distinguen del resto de las herramientas CASE evaluadas.

Capítulo I. Modelación de datos con Entidad Relación

1.1 Modelación de datos

Paralelo al avance de la tecnología de bases de datos relacionales se han desarrollado metodologías y técnicas para su diseño. Todas ellas tienen en común la descomposición del proceso en fases, con objetivos bien definidos y técnicas establecidas para conseguir estos objetivos.

La modelación conceptual es la etapa en que se crea un esquema conceptual que describe de manera concisa, los requerimientos de información desde el punto de vista de los usuarios (Bienemann, Schewe & Thalheim, 2013). Los datos se expresan mediante conceptos de un modelo de alto nivel que son independientes de un SGBD.

Un modelo de datos es una colección de conceptos y notaciones para describir datos, sus interrelaciones, su semántica, restricciones de integridad y operaciones sobre los mismos (Brodie, Mylopoulos & Schmidt, 2012; Ullman & Widom, 2013; Silberschatz, Korth, & Sudarshan, 2014). Un modelo de datos ofrece una manera de describir una base de datos para un determinado nivel de abstracción.

La representación de los datos a través de conceptos que no incluyen detalles de su almacenamiento o implementación suelen estar cercanos al lenguaje natural, y son fáciles de entender, de modo que pueden utilizarse para la comunicación con usuarios no especialistas. De esa manera, el esquema se utiliza como referencia para garantizar que se satisfagan los requerimientos de los usuarios y que no existan contradicciones entre dichos requerimientos. Utilizando este enfoque, los diseñadores de bases de datos solo deben concentrarse en especificar las propiedades de los datos, sin tener en cuenta detalles de implementación y almacenamiento.

1.2 El modelo Entidad Relación

El modelo ER es de datos semántico que se ha convertido en uno de los más populares desde su presentación por Chen (1976). Es extensamente usado durante el análisis de requisitos para la modelación conceptual de la base de datos, parte de su popularidad se debe a la claridad y simplicidad de su notación gráfica en forma de diagramas, lo que unido a una semántica bien definida, lo convierte en una excelente herramienta de comunicación entre analistas de sistemas, diseñadores y usuarios finales (Batini, Barone, Garasi & Viscusi, 2012).

Los modelos más comunes para bases de datos tienen un formato de representación para entidades, sus atributos y para interrelaciones entre entidades. La comunidad de especialistas en base de datos reconoce el concepto de entidad como el más importante en la modelación, denominado indistintamente entidad u objeto. El modelo para una entidad es construido a partir de un conjunto relativamente pequeño de primitivas de modelación.

Aunque la modelación orientada a objetos se ha hecho popular, en la actualidad el modelo ER no ha perdido su vigencia. Algunos estudios han demostrado (Davies, Green, Rosemann, & Gallo, 2004; Davies, Green, Rosemann, Indulska, & Gallo, 2006) que el modelo ER sigue siendo el modelo conceptual de datos más utilizado; la continuidad de una conferencia internacional sobre el modelo ER (International Conference on Conceptual Modeling), la edición de libros y artículos sobre la temática en años recientes, es un testimonio de la actualidad de este modelo que justifica su utilización.

1.2.1. El modelo ER original

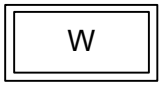
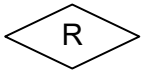
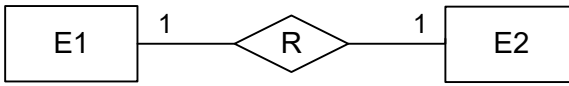
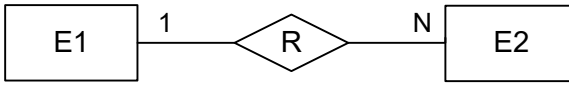
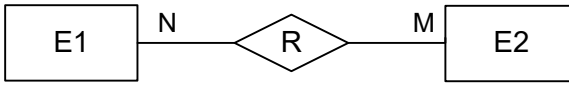
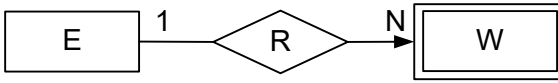
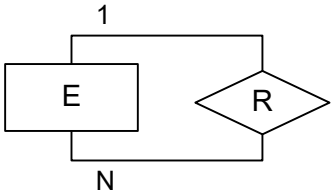
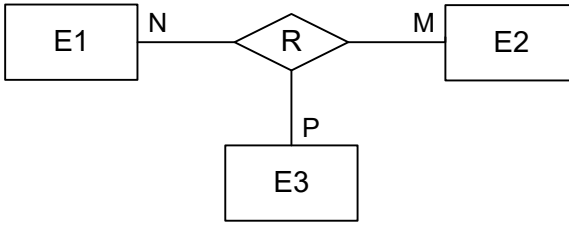
Las construcciones del modelo ER que se presentan Chen (1976), son las entidades, interrelaciones y atributos. Una entidad es algo que puede ser perfectamente identificado y es descrita por sus atributos. Las entidades que poseen los mismos atributos se clasifican en conjuntos de entidad o tipos de entidad.

Una interrelación, en el modelo original es una asociación entre entidades, lo que indica que solo existían interrelaciones de asociación. Según su grado estas podían ser recursivas, binarias o ternarias. Al conjunto de interrelaciones existentes entre dos o más conjuntos de entidades se le denomina conjunto de interrelación o tipo de interrelación. En el modelo original solo era posible establecer restricciones de cardinalidad máxima en las interrelaciones de asociación, lo que limitaba su capacidad de expresividad.

Las entidades y las interrelaciones se describen a través de conjuntos de pares (atributo, valor) que representan sus propiedades o características relevantes. Se reconoce la necesidad del empleo de atributos para identificar de manera única a cada entidad de un conjunto de entidad. En determinadas ocasiones una entidad puede depender de entidades de otros conjuntos de entidades para su existencia, debido a que no posee suficientes atributos propios para identificarse; Chen (1976), denominó entidades débiles a las entidades con estas características. De acuerdo con lo anterior, si la interrelación se utiliza para identificar un conjunto de entidad débil, entonces se denomina interrelación de entidad débil, en caso contrario, interrelación de entidad regular o fuerte. Del mismo modo, propone dos formas de clasificar las entidades en regulares y débiles. No se hace una diferenciación entre atributos simples o compuestos y monovaluados o multivaluados.

Para expresar un esquema ER en forma de un diagrama se utiliza un rectángulo y un rombo para representar a los conjuntos de entidad y conjuntos de interrelación, respectivamente. Los conjuntos de entidad interrelacionados están conectados al rombo por líneas rectas. Cada línea es marcada con un “1”, “N” o “M” para indicar interrelaciones del tipo 1:1, 1:N o M:N. Un conjunto de entidad débil es encerrado dentro de un rectángulo de doble línea y es identificado por otro conjunto de entidad por una flecha que parte del rombo hacia el conjunto de entidad dependiente. En el resto del informe se utilizarán los términos tipo de entidad por conjunto de entidad y tipo de interrelación por conjunto de interrelación. En la tabla 1.1 se muestra la notación original del modelo ER.

Tabla 1.1. Notación original propuesta por Chen para los diagramas ER.

Símbolo	Significado
	Tipo de entidad E.
	Tipo de entidad débil W.
	Tipo de interrelación R.
	Cardinalidad 1:1 para E1:E2 en R.
	Cardinalidad 1:N para E1:E2 en R.
	Cardinalidad N:M para E1:E2 en R.
	Tipo de entidad W identificada por un tipo de entidad regular E a través del tipo de interrelación débil R.
	Tipo de interrelación recursiva R.
	Tipo de interrelación ternaria R.

En el modelo básico no existían reglas para transformar el esquema conceptual en esquemas lógicos, sino que el diagrama ER era traducido hacia un diagrama de estructura de datos, que era una representación a nivel de registros. Desde la década de 1980 se ha producido un rápido incremento en el desarrollo de sistemas de bases de datos que motivó a muchos investigadores (Teorey, et al., 1986; Chen, 2006; Hartmann & Link, 2007) a introducir conceptos adicionales que ayuden a captar cada vez más la semántica de un universo del discurso, este esfuerzo para extender el modelo básico se conoce de manera simbólica como modelo Entidad Relación Extendido.

1.2.2. Extensiones al modelo Entidad Relación

Según Ullman & Widom (2013), el modelo ER es muy simple y sus conceptos básicos son entidades e interrelaciones. Las entidades ya fueron abordas en el modelo original y por tanto, las extensiones se refieren fundamentalmente a las interrelaciones y a las reglas de transformación hacia el modelo lógico. Aunque el modelo básico solo hace referencia a estos dos conceptos, también incluye otros como el de dependencia de existencia, para entidades débiles.

El concepto de *cardinalidad* del modelo ER original fue extendido con la introducción de las restricciones de cardinalidad mínima[2], por Webre (1981). Estas restricciones expresan si las entidades tienen una participación opcional u obligatoria en una interrelación de asociación. De esta manera, la cardinalidad de un tipo de entidad es el número mínimo y máximo de entidades del tipo de interrelación en las que puede intervenir una de sus entidades. Típicamente la cota inferior es 0 o 1 y la superior 1 o N (muchos); esta forma de indicar la cardinalidad se denota con el par (mín, máx). La participación obligatoria de una entidad en una interrelación de asociación expresa una dependencia de existencia, que hasta el momento solo se había reconocido en las entidades débiles.

2 También conocidas como restricciones de participación o de membresía.

En las diversas notaciones del modelo EER, los valores de las restricciones de cardinalidad son mostrados en los diagramas ER sobre las líneas que relacionan a cada uno de los tipos de entidades con el tipo de interrelación de asociación. La ubicación de la cardinalidad en el diagrama ER es conocido en la literatura como convenio look across y look here [3], términos utilizados por primera vez en Ferg (1991). La comprensión del tipo de convenio utilizado para expresar las restricciones de cardinalidad posibilita una interpretación correcta de la semántica de las interrelaciones. Sin embargo, en la literatura se observa que estos términos no se utilizan de manera uniforme, lo que puede generar confusiones a lectores y a usuarios de herramientas CASE al interpretar las restricciones de cardinalidad de una notación determinada del modelo EER. En la tabla 1.2 se puede observar cómo quedan representadas, para algunas notaciones del modelo ER, las restricciones de cardinalidad que modelan los siguientes hechos: en un departamento trabajan de uno a varios empleados y un empleado puede trabajar para un solo departamento. De acuerdo con esta especificación de requisitos, y utilizando las definiciones de cardinalidad mínima (min-card) y cardinalidad máxima (max-card) enunciadas por Batini, et al. (2011), las restricciones de cardinalidad de los tipos de entidad EMPLEADO y DEPARTAMENTO en el tipo de interrelación TRABAJA_EN son:

min-card (EMPLEADO, TRABAJA_EN) = 0

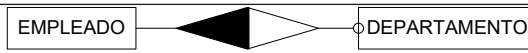
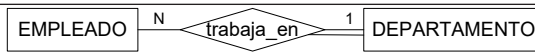
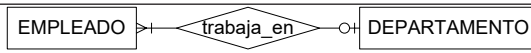
max-card (EMPLEADO, TRABAJA_EN) = 1

min-card (DEPARTAMENTO, TRABAJA_EN) = 1

max-card (DEPARTAMENTO, TRABAJA_EN) = N

3 Se prefiere utilizar intencionalmente los términos en inglés ya que su traducción al español no es muy conocida.

Tabla 1.2. Convenio para representar las restricciones de cardinalidad en algunas notaciones del modelo ER.

Diagrama ER	Notación	Convenio
	Chen (notación actual)	min-card: look here max-card: look across
	Teorey	min-card: look across max-card: look across
	Batini, Navathe y Ceri	min-card: look here max-card: look here
	Elmasri y Navathe	min-card: look here max-card: look across
	McFadden y Hoffer	min-card: look across max-card: look across

Nótese en la tabla anterior la variedad de formas de representar las cardinalidades descritas en Teorey, et al., 1986; Batini, et al., 2011; Chen, 2011; Elmasri & Navathe, 2012; McFadden & Hoffer, 2011. Una explicación más detallada sobre diversas notaciones del modelo ER y sus correspondientes representaciones de los convenios look across y look here puede encontrarse en Hay (1999); y Song, et al., (1995).

Para las notaciones que permiten modelar el *grado* de las interrelaciones de asociación, se observa que en especial para las interrelaciones ternarias o de mayor grado, su semántica queda mejor expresada si las restricciones de participación utilizan el convenio look here (Nieto, Martínez, Cuadra & de Miguel, 2012).

Dentro de las *construcciones*, la abstracción de generalización fue una de las más importantes contribuciones del modelo EER y en menor medida, otras como la agregación y la categorización, significaron aportes a la sintaxis y semántica del modelo ER.

Generalización/especialización: la abstracción de generalización fue abordada por Smith & Smith en (1977), al proporcionar una base formal para los subtipos dentro de un modelo de datos. Posteriormente, otros autores (Elmasri, et al., 2012; Ling, 1985; Navathe & Cheng, 1983; Sakai, 1983) introdujeron esta construcción en diversas extensiones al modelo ER.

La principal propiedad de una jerarquía de generalización/especialización es la herencia, lo que implica que todas las propiedades del tipo de entidad genérico son heredadas por los tipos de entidades especializados. La herencia constituye un aporte substancial a la modelación de la realidad, pues permite definir subconjuntos de tipos de entidades que tienen atributos e interrelaciones propias, a la vez que conservan todos los atributos e interrelaciones del tipo de entidad más general (García & Montes de Oca, 2015).

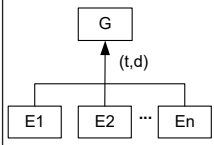
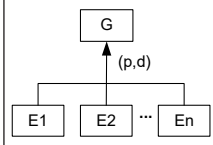
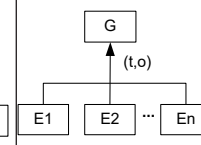
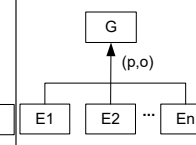
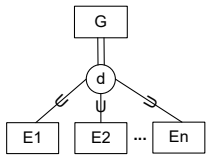
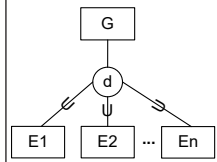
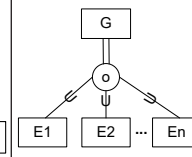
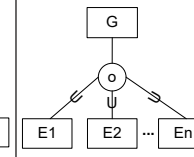
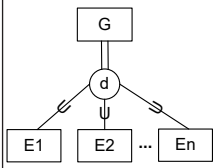
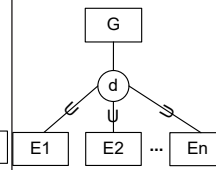
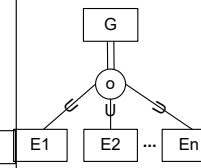
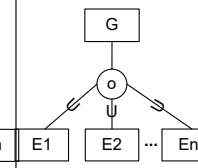
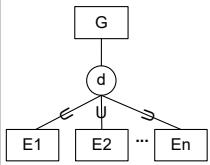
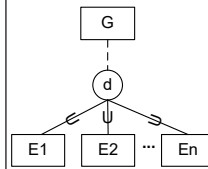
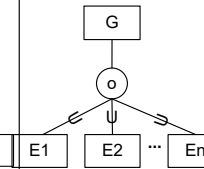
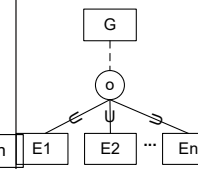
Las interrelaciones ISA (acrónimo del inglés ISA cuyo significado es "es un") son un tipo especial de interrelaciones semánticas que se utilizan para modelar subconjuntos y cuya propiedad más significativa es la herencia. Para Manila & Räihä (2012), las interrelaciones ISA se modelan como entidades débiles lo que reduce la oportunidad de modelar hechos de diferente naturaleza. La identificación de un tipo de entidad débil en una interrelación ISA se hace a través de la llave primaria del tipo de entidad genérico, por lo que las entidades de los tipos de entidades genérico y especializado tendrán la misma llave primaria.

Hay dos restricciones que pueden aplicarse a una jerarquía de generalización/especialización: la restricción de participación y la restricción de disyunción (Parsons & Li, 2007). La imposición de estas restricciones determinará el significado que cada especialización tiene en el mundo real y la aplicación de las reglas correspondientes que regirán el comportamiento de las entidades que forman parte de este tipo de interrelación (Artale, Calvanese, Kontchakov, Ryzhikov & Zakharyaschev, 2007).

Diversas notaciones del modelo EER (Finkelstein, 1998; Barker, 2011; Batini, et al., 2011; McFadden & Hoffer, 2011; Elmasri & Navathe, 2012; Bruce, 2013; Korth, Silberschatz, & Sudarshan,

2014) permiten modelar jerarquías de generalización/especialización (véase tabla 1.3).

Tabla 1.3. Notaciones para la construcción de Generalización/Especialización.

Notación	Restricciones de participación y de disyunción			
	Total y Disjunta	Parcial y Disjunta	Total y Solapada	Parcial y Solapada
Batini				
Teorey (notación actual)				
El-masri y Navathe				
Wand y Weber				

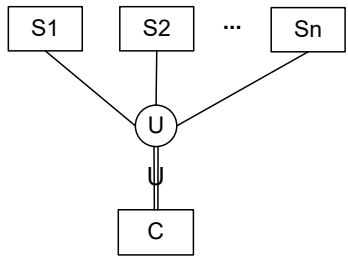
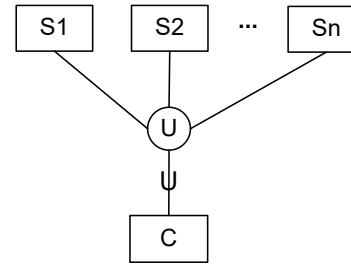
Donde G es el tipo de entidad más general y E_1, E_2,…, En son los tipos de entidades especializados.

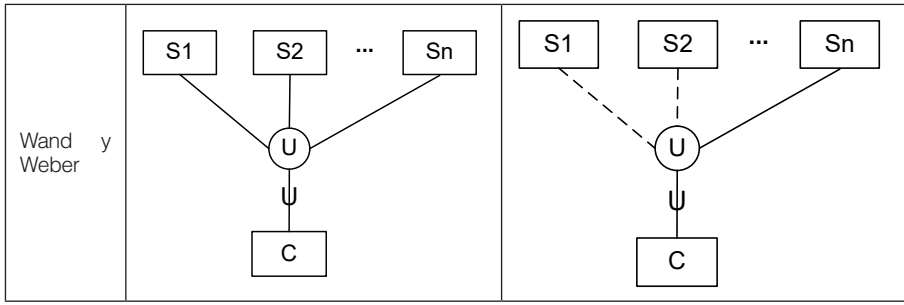
Categorización: el concepto de categorización tiene su base en

el modelo Entidad Categoría Relación (ECR) descrito en Elmasri, et al. (1985), y se define a través de tipos de entidades con diferentes niveles jerárquicos. Una categoría es una subclase de la unión de dos o más superclases que pueden tener diferentes atributos llaves porque pueden ser de diferentes tipos de entidades (Elmasri & Navathe, 2012). Entre sus características se encuentran que la cardinalidad máxima que se establece entre los tipos de entidades de orden superior y el tipo de entidad categorizado es 1:1; la participación del tipo de entidad categorizado siempre es total, mientras que cada tipo de entidad de orden superior puede tener una participación total o parcial.

La categorización ha sido poco tratada en la literatura, razón por la cual numerosas notaciones del modelo EER no la soportan. Notaciones como las de Wand & Weber (2011) y la de Elmasri & Navathe (2012), proporcionan una representación gráfica para este tipo de construcción (véase tabla 1.4).

Tabla 1. 4. Notaciones para la construcción de categorización.

Notación	Restricción de participación de la categoría	
	Total	Parcial
Elmasri y Navathe		



Donde *C* es el tipo de entidad que representa a la categoría y S_1, S_2,…, Sn son los tipos de entidades de nivel superior.

Según la notación de Elmasri & Navathe (2012), los tipos de entidades de orden superior se conectan a un círculo con el símbolo U, que representa la operación unión de conjuntos. La línea que conecta el tipo de entidad categorizado al círculo tiene el símbolo de subconjunto indicando la dirección de la categorización. Una línea simple indica participación parcial de un tipo de entidad de orden superior, mientras que una línea doble indica participación total. De ser necesario un predicado, este se muestra junto a la línea que conecta el círculo con el tipo de entidad de orden superior al que es necesario aplicarle el predicado.

A modo de comparación, se relacionan algunas características de las construcciones de generalización/especialización y categorización:

- Una categoría tiene dos o más tipos de entidades de orden superior, mientras que la generalización/especialización solo tiene una.
- En la categorización la "herencia" de atributos funciona de manera más selectiva que en la generalización/especialización y es denominada "herencia selectiva" (Umanath & Scamell, 2011).

Agregación: una agregación es una abstracción en la cual una interrelación entre objetos es considerada como un objeto de mayor nivel (Smith & Smith, 1977). Existen diversos casos en

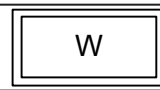
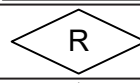
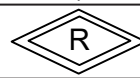
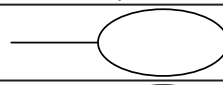
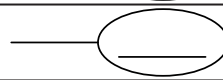
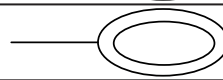
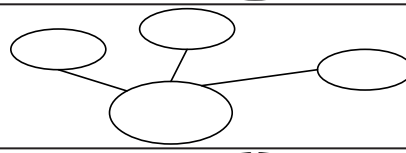
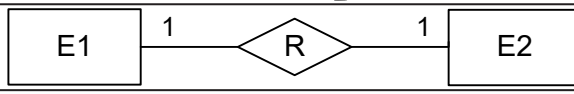
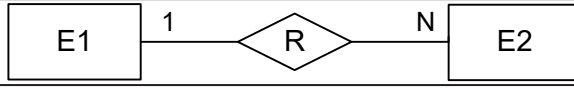
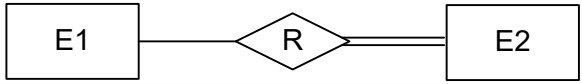
los que este concepto se expresa en el modelo ER. En su forma más simple, es un conjunto al que se desea asociar varias propiedades, por ejemplo un empleado puede ser un agregado de los componentes número de identidad, nombre y dirección. En su forma más compleja, una interrelación entre entidades se considera como una entidad agregada de mayor nivel (Shanks, Tansley, Nurelini, Toblin, & Weber, 2008; Keet & Artale, 2012; Teorey, Buxton & Simsion, 2013), la cual define un nuevo tipo de entidad, denominado tipo de entidad agregada, a partir de otros tipos de entidades que representan sus partes componentes. La agregación establece una interrelación de tipo "Is-a-part-of" (acrónimo del inglés Es una parte de) entre un "todo" y las "partes" que lo componen.

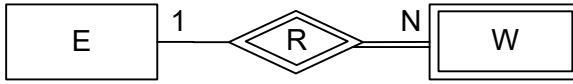
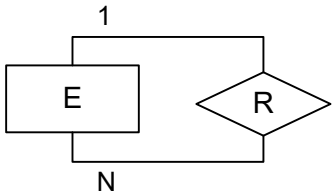
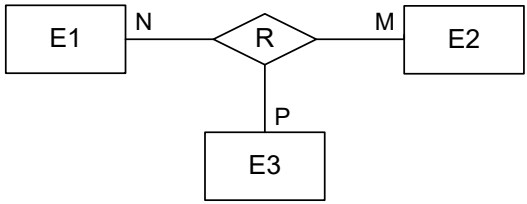
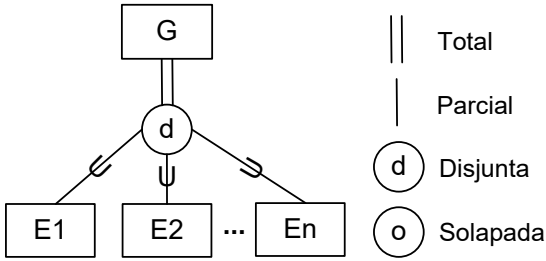
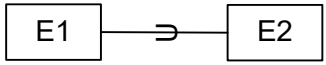
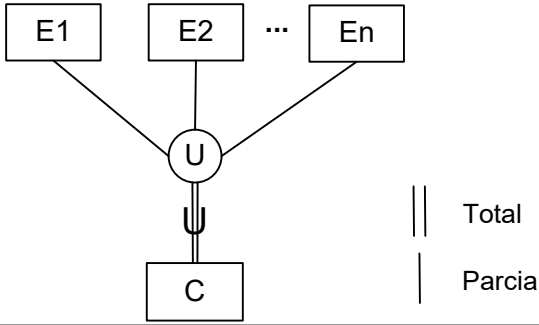
La *composición* es una forma específica de agregación que representa una asociación entre entidades donde hay una pertenencia fuerte y una existencia coincidente entre el "todo" y las "partes" (Connolly & Begg, 2014b). En una composición, el "todo" es responsable de la creación y destrucción de sus "partes". Este concepto aparece con las aplicaciones ingenieriles al posibilitar el ensamble de entidades compuestas (Girju, Badulesco & Moldovan, 2006).

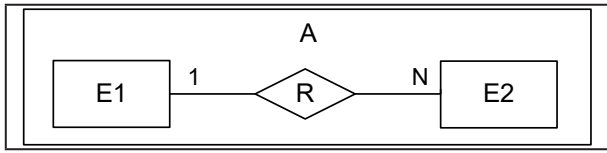
En las diversas notaciones del modelo EER se observa que un número reducido de ellas permiten soportar los conceptos de agregación y composición. Por ejemplo, en Elmasri & Navathe (2012); y Silberschatz, et al. (2014), el tipo de entidad agregada se representa como un rectángulo que contiene al tipo de interrelación de asociación y a los tipos de entidades que la conforman. En Markowitz & Shoshani (1992), se permite expresar la agregación mediante la asociación entre tipos de interrelaciones. En Umanath & Scamell (2011), la composición se representa mediante un círculo con la letra 'A' en su interior. El tipo de entidad que representa al "todo" y los tipos de entidades que representan las "partes" se unen al círculo mediante una línea.

En la tabla 1.5 se muestran las notaciones más utilizadas en el modelo entidad relación extendido (EER).

Tabla 1.5. Notación de Elmasri y Navathe para diagramas EER.

Símbolo	Significado
	*Tipo de entidad regular E.*
	*Tipo de entidad débil W.*
	*Tipo de interrelación regular R.*
	*Tipo de interrelación débil R.*
	*Atributo.*
	*Atributo identificador.*
	*Atributo multivaluado.*
	*Atributo compuesto.*
	*Atributo derivado.*
	*Cardinalidad 1:1 para E1:E2 en R.*
	*Cardinalidad 1:N para E1:E2 en R.*
	*Cardinalidad N:M para E1:E2 en R.*
	*Participación total de E2 en R.*

	*Tipo de entidad W identificada por un tipo de entidaa través del tipo de interrelación débil R.*
	*Tipo de interrelación recursiva R.*
	*Tipo de interrelación ternaria R.*
	*Jerarquía de generalización/ especialización.*
	*Interrelación de subconjunto (ISA).*
	*Interrelación de categorización.*



Tipo de entidad agregada A.

Hasta aquí se ha hecho un resumen de las extensiones inherentes a la fase de modelación conceptual que más han trascendido.

Otra extensión de relevancia para la modelación de datos, por su ayuda al diseño de bases de datos relacionales, es el de las *reglas de transformación* de un esquema ER a esquemas lógicos. En ese sentido, diversos autores (Teorey, et al., 1986; Elmasri & Navathe, 2012; Ponniah, 2013; Ullman & Widom, 2013; Teorey, Lightstone & Nadeau, 2014; Silberschatz, et al., 2014), han propuesto un conjunto de reglas para transformar esquemas ER al modelo relacional. Estas reglas siguen uno de dos enfoques en lo relativo a la transformación de las interrelaciones de asociación: 1) las que propagan la llave extranjera hacia alguno de los esquemas de relación; 2) las que evitan el uso de llaves extranjeras. Cada enfoque tiene sus ventajas y desventajas.

El enfoque basado en la propagación de la llave extranjera tiene como ventaja que reduce la cantidad de esquemas relacionales que se generan en el esquema lógico. Como desventaja, se destaca en Wilmot (1984), que la inclusión de las llaves extranjeras "contamina" los esquemas con atributos que no son propios, trae consigo inconvenientes en la independencia de los datos. Un problema que puede generarse en la utilización de este enfoque es el relacionado con la referencia cíclica entre llaves extranjeras de esquemas relacionales que no admiten el valor nulo. Esta inconsistencia en el esquema lógico trae como consecuencia problemas en el momento de poblar con datos los esquemas resultantes (tablas). Este tipo de inconsistencia no ha sido abordado en la literatura, y las herramientas CASE actuales no realizan ningún tipo de validación en este sentido.

Una de las ventajas del enfoque basado en la no propagación de las llaves extranjeras es que no se generan inconsistencias

de referencia cíclica entre esquemas. Como desventaja se encuentra que al transformar todas las interrelaciones en un nuevo esquema de relación se incrementan los costos de almacenamiento y de programación (Wilmot, 1984).

Nótese que entre las reglas de transformación a esquemas lógicos, el enfoque de la propagación de la llave extranjera es el más abordado en la literatura y el más utilizado por las herramientas CASE de diseño de bases de datos, razón por la cual es tomado como referente en esta investigación.

La generación de esquemas relacionales a partir de un esquema conceptual ER involucra también el establecimiento de diversos tipos de restricciones de integridad que rigen el comportamiento de las entidades. Entre las restricciones de integridad, la *dependencia de existencia* no ha sido tratada con suficiente sistematicidad en las construcciones del modelo EER. Ha sido común considerar que existe dependencia de existencia entre dos entidades, cuando la eliminación de una entidad (entidad dominante) conlleva a la eliminación de otra entidad (entidad subordinada) (Silberschatz, et al., 2014).

En diversas construcciones del modelo EER la dependencia de existencia se manifiesta de manera explícita o implícita. En una jerarquía de generalización/especialización la eliminación de una entidad del tipo de entidad más general implica que se elimine todas las entidades de los tipos de entidades al cual pertenece. En una categorización, toda entidad de la categoría depende de la existencia de una entidad que pertenece a algunos tipos de entidades de orden superior. En este caso el comportamiento de las entidades dependientes no puede especificarse a través de acciones referenciales sobre las llaves extranjeras, sino mediante restricciones procedimentales como pueden ser con disparadores. En la agregación, una entidad agregada depende de la existencia de cada una de las entidades que la componen. En la composición las entidades que pertenecen a los tipos de entidades de las “partes” están interrelacionadas con una y solo una entidad del tipo de entidad agregado, lo que

pone de manifiesto una dependencia de existencia de las "partes" con respecto al "todo".

Como se puede apreciar, las extensiones al modelo ER original ayudan a describir un universo de discurso con mayor expresividad que las ofrecidas en el modelo original, la sistematización de estas extensiones con un enfoque y una notación coherente, sienta las bases para el desarrollo de herramientas CASE. Un concepto en el que se aprecia la necesidad de dicha sistematización es el de dependencia de existencia, abordado en ocasiones de manera genérica y sus implicaciones en las interrelaciones de asociación no han recibido la atención que requiere.

La sistematización a que se hace referencia en el estudio del modelo, no abarca solamente a las construcciones y su comportamiento de forma aislada, sino también a las regularidades que se deben observar cuando se usan para modelar una realidad concreta y que conducen a la necesidad de validaciones con enfoques integradores.

1.3 Validación de esquemas conceptuales

El proceso de diseño de una base de datos comienza con la ingeniería de requisitos. Una actividad clave durante este proceso es la creación de un esquema conceptual, el cual describe las propiedades que el sistema de información debe tener y sirve como base para las siguientes fases del diseño. Aunque la modelación conceptual representa solo una pequeña parte del desarrollo de un sistema, esta tiene un impacto importante en el resultado final; insuficiencias en el diseño conceptual constituyen un factor de peso en el fracaso de un sistema de información (Graeme Shanks, Tansley & Weber, 2003).

1.3.2. Validación estructural de esquemas conceptuales

En la modelación conceptual, las restricciones de integridad son utilizadas para especificar la manera en que los elementos de una base de datos van a estar asociados entre sí, y por tanto especifican cuáles instancias de la base de datos son válidas (Olivé, 2012b).

Las restricciones de cardinalidad están entre los tipos de restricciones de integridad más ampliamente utilizadas en la modelación conceptual (Calì, 2007). Su función es restringir el número de interrelaciones en que una entidad participa en un tipo de interrelación.

Existen varias maneras de definir la cardinalidad de una interrelación de asociación. Se adopta la definición de cardinalidad para interrelaciones de asociación de grado *n* formulada en (Olivé, 2012a). Por ejemplo, para interrelaciones binarias:

- *Card*(E_1;E_2;R)=(*min, max*) indica el número mínimo y máximo de entidades del tipo entidad E_2 que pueden estar interrelacionadas en *R*, con alguna entidad del tipo entidad E_1.
- *Cmin*(p_1:E_1,p_2:E_2:R) y Cmax(p_1:E_1,p_2:E_2:R) denotan el valor de la cardinalidad mínima y máxima del tipo de entidad E_1 a través del rol p_1.

El uso de 0 y 1 para la cardinalidad mínima y de 1 y N para la cardinalidad máxima es el enfoque más ampliamente utilizado en (Chen, 1976; Markowitz & Shoshani, 1992; Elmasri & Navathe, 2012; Garcia-Molina, Ullman, & Widom, 2012; Date, 2013; Ullman & Widom, 2013; Ponniah, 2013; Silberschatz, et al., 2014; Teorey, et al., 2014). Otros enfoques generalizan las restricciones de cardinalidad de manera que permiten valores mayores que 1 para la cardinalidad mínima y máxima (Lenzerini & Santucci, 1983; Lenzerini & Nobili, 1990; Calvanese & Lenzerini, 1994; Thalheim, 1992, 1999, 2013; Mira Balaban, Maraee & Sturm, 2007). Algunos autores generalizan el concepto de cardinalidad posibilitando que esta puede ser expresada como un conjunto de números {min_1,…, max_1,…,min_n,…,max_n} (Hartmann, 1998, 2000, 2001a, 2001b).

El proceso de adquisición de las restricciones de cardinalidad está lejos de ser una tarea trivial (Hartmann, Link & Trinh, 2009), la semántica se recopila a partir de varias fuentes que pudieran generar conflictos. La práctica demuestra que la integración de la información captada puede conducir a obtener restricciones

de cardinalidad inconsistentes, por lo que el problema radica en: 1) ¿cómo identificar las inconsistencias?, 2) ¿cómo reportarlas al diseñador?, 3) ¿cómo corregirlas? (Hartmann, 2001a).

Sea *S* un esquema ER con un conjunto *C* de restricciones de cardinalidad asociadas. Se dice que un esquema *S* es satisfecho desde el punto de vista estructural si admite al menos una instancia válida de la base de datos. Sin embargo, puede suceder que para algunas restricciones de cardinalidad el conjunto de instancias válidas sea vacío o infinito. Se dice entonces que un esquema *S* es totalmente satisfecho si admite un conjunto no vacío y finito de instancias válidas. Esquemas que no sean totalmente satisfechos se consideran incorrectos (Olivé, 2012a).

Formalmente, un esquema ER *S* con un conjunto *C* de restricciones de cardinalidad es consistente o totalmente satisfecho si existe al menos una base de datos $DB = (r_1,\ldots,r_k)$ acorde a *(S,C)* en la cual todas las instancias ri sean no vacías (Thalheim, 1992).

En la literatura se reportan varios trabajos que abordan el problema de determinar la consistencia de un conjunto de restricciones de un esquema conceptual, utilizando dos enfoques en su solución: el de programación lineal y el basado en la teoría de grafos. El primero reduce el problema a encontrar una solución a un sistema de inecuaciones lineales. El segundo, permite la detección de restricciones de cardinalidad inconsistentes mediante la búsqueda de ciclos que cumplan con determinadas condiciones.

1.3.2.1. Detección de inconsistencias utilizando programación lineal

Uno de los primeros trabajos relacionados con la satisfacción de las restricciones de cardinalidad en un esquema ER aparece en Lenzerini & Nobili (1990). El método consiste en transformar las restricciones de cardinalidad en un sistema de inecuaciones lineales, cuyas variables representan los valores de las cardinalidades de los tipos de entidad y tipos de interrelación en una

posible instancia. De acuerdo con elmétodo, las restricciones de cardinalidad de un diagrama ER pueden ser satisfechas si y solo si el sistema de inecuaciones tiene una solución.

Como limitaciones se encuentran que solo se puede aplicar en esquemas ER que contengan interrelaciones de asociaciones recursivas y binarias y que no determina cuáles son las restricciones de cardinalidad que hacen que el esquema conceptual sea inconsistente. En Calvanese & Lenzerini (1994), se extiende el sistema de inecuaciones basado en el método anterior para aplicarlo a esquemas ER que incluyan interrelaciones de subconjunto.

La extensión se basa en el supuesto de que las entidades de los tipos de entidades especializados pueden solaparse. Se proporciona un algoritmo con dos fases, en el cual el problema de la satisfacción de las restricciones de cardinalidad de un diagrama ER, que incluye interrelaciones de subconjunto, se reduce al problema de satisfacción de las restricciones de cardinalidad de un diagrama ER que no contenga interrelaciones de subconjunto. Entonces de manera similar al método anterior se prueba que el sistema de inecuaciones de este nuevo diagrama tenga una solución. Debido a que el sistema de inecuaciones es más complejo que el propuesto en Lenzerini & Nobili (1990), su complejidad temporal en el peor caso es exponencial. Este método fue simplificado por Cadoli, Calvanese, De Giacomo & Mancini (2004), restringiendo solamente el solapamiento a las jerarquías de subconjunto, lo cual reduce el número de tipos de entidades y de asociaciones entre ellas, pero la complejidad temporal para el peor caso sigue siendo exponencial.

Debido a la similitud entre las restricciones de cardinalidad en el modelo ER y la multiplicidad en el modelo UML (acrónimo del inglés Unified Modeling Language), los resultados obtenidos en el análisis de las restricciones de cardinalidad han sido utilizados en UML (Balaban & Maraee, 2006; Mira Balaban, et al., 2007) y se extiende al sistema de inecuaciones analizado en Lenzerini & Nobili (1990), para aplicarlo a diagramas UML que pueden in-

cluir: asociaciones binarias, jerarquías de clases, jerarquías de generalización, asociaciones de orden *n* y clases de asociación. La extensión se basa en un preprocesamiento que reduce el problema de la satisfacción de las restricciones de integridad del diagrama UML al problema de satisfacción de las restricciones de integridad analizado al comienzo de este epígrafe. La ventaja de este método en comparación con el método de Calvanese & Lenzerini (1994), radica en su simplicidad y eficiencia, lo que lo hace adecuado para su inclusión en una herramienta de diseño basado en UML. Como limitación está que el método no es aplicable cuando existen ciclos en una jerarquía de clases.

1.3.2.2. Detección de inconsistencias utilizando la teoría de grafos

Lenzerini & Nobili (1990), también proponen un método para identificar restricciones de cardinalidad inconsistentes, basado en la representación del esquema ER mediante un multígrafo dirigido y en la búsqueda de ciclos que cumplan con una determinada condición. En el multígrafo dirigido *G = (V, A)* asociado al esquema conceptual *S*, el conjunto de vértices *V* está formado por los tipos de entidades y los tipos de interrelaciones del esquema conceptual. El conjunto de arcos *A* está determinado por las siguientes reglas: por cada conexión en S entre un tipo entidad E_1 y un tipo de interrelación R a través del rol p_1, se establecen dos arcos e_1 y $e_2 \in A$; e_1 va del nodo E_1 al nodo *R* y tiene como peso $Cmax(p_1:E_1,p_2:E_2;R)$; e_2 va del nodo R al nodo E_1 y tiene como peso $\frac{1}{Cmin(p_1:E_1,p_2:E_2;R)}$ si $Cmin(p_1:E_1,p_2:E_2;R) \neq 0$, ó ∞ si $Cmin(p_1:E_1,p_2:E_2;R) = 0$. De manera similar se establecen los arcos entre E_2 y *R* a través del rol p_2. En este trabajo los autores han considerado que $Cmin(p_1:E_1,p_2:E_2;R)$ y $Cmax(p_1:E_1,p_2:E_2;R)$ pueden tomar cualquier valor positivo siempre que se cumpla que $Cmin(p_1:E_1,p_2:E_2;R) \leq Cmax(p_1:E_1,p_2:E_2;R)$.

La determinación de restricciones de cardinalidad inconsistentes se realiza buscando ciclos críticos. Un ciclo crítico es una

secuencia de arcos (v_0,v_1) (v_1,v_2) … $(v_k\text{-}1,v_k)$ que debe cumplir con las siguientes condiciones: $v_k = v_0$; $v_1, \ldots, v_k$ son distintos; y el producto del peso de los arcos (v_0,v_1) (v_1,v_2) … $(v_k\text{-}1,v_k)$ es menor que 1.

Cada ciclo crítico encontrado en el grafo se corresponde con un conjunto de restricciones de cardinalidad que no pueden ser satisfechas en el esquema conceptual. El método solo puede ser aplicado a esquemas que contengan interrelaciones de asociaciones recursivas y binarias. Una extensión a esquemas que incluyan interrelaciones de orden *n* fue desarrollada por Thalheim (1992), donde se formaliza una notación de las restricciones de cardinalidad para las interrelaciones de asociación de orden *n*, y se generaliza el concepto de restricciones de cardinalidad, de manera que las restricciones de cardinalidad pueden ser especificadas como un conjunto de números en lugar de un intervalo de valores.

La representación del esquema conceptual mediante un grafo es similar a lo analizado por Lenzerini & Nobili (1990), aunque con una ligera variación para soportar las restricciones de cardinalidad de las interrelaciones de orden n. Una vez construido el grafo la detección de las restricciones de cardinalidad inconsistentes sigue el mismo procedimiento de Lenzerini & Nobili (1990). Para la corrección de esquemas inconsistentes se propone un algoritmo, el cual toma como entrada un esquema ER formado por los tipos entidades y tipos de interrelaciones, un conjunto de restricciones de cardinalidad asociados al esquema ER y un conjunto formado por todos los ciclos críticos detectados en el grafo.

El algoritmo elimina del esquema ER los tipos de entidades y tipos de interrelaciones que forman parte de un ciclo crítico, así como todos aquellos tipos de interrelaciones cuyos componentes forman parte de algún ciclo crítico, se obtiene como resultado un esquema consistente. Nótese que en este método no se

hace una corrección de las restricciones de cardinalidad inconsistentes sino que se eliminan del esquema ER los tipos de entidades y tipos de interrelaciones que inciden en la inconsistencia del esquema conceptual.

Otros métodos que abordan el problema de la satisfacción de las restricciones, incluyen además de la detección de inconsistencias, la identificación de las causas y un plan para la corrección del esquema ER. En Hartmann (2000, 2001a), se proponen métodos basados en los ciclos críticos para identificar las causas de las inconsistencias y se sugieren cuatro estrategias heurísticas para la corrección de los esquemas conceptuales inconsistentes. Como ventaja se señala la de proporcionar al diseñador un plan de correcciones a aplicar sobre el esquema conceptual. Estos resultados fueron extendidos por Hartmann (2003), de manera que las restricciones de cardinalidad son tratadas como restricciones suaves al basarse en un promedio de satisfacción del conjunto de restricciones de cardinalidad, o en un por ciento esperado de satisfacción del conjunto de restricciones de cardinalidad. Ambos enfoques proponen procedimientos para la corrección de esquemas, similares a los descritos por Hartmann (2001a).

Otro método que realiza la detección e identificación de las causas de las inconsistencias en un esquema conceptual ER es el propuesto por Dullea & Song (1998a, 1998b, 1999); y Dullea, et al. (2003), el cual se basa en el análisis de las restricciones de cardinalidad y su influencia en las interrelaciones de asociación de grado 1, 2 o 3 cuando estas forman parte de algún ciclo en el diagrama ER (véase tabla 1.6). Los valores utilizados para expresar las restricciones de cardinalidad son: 0 y 1 para la cardinalidad mínima; 1 y N para la cardinalidad máxima.

Tabla 1.6. Reglas para la validación estructural de esquemas conceptuales.

Reglas para la validación estructural de interrelaciones recursivas
Regla 1: Solo las interrelaciones recursivas 1:1 con restricciones de cardinalidad mínima “obligatoria-obligatoria” u “opcional-opcional” son estructuralmente válidas. **Regla 2**: Las interrelaciones recursivas 1: N o N:1 con restricciones de cardinalidad mínima “opcional-opcional” son estructuralmente válidas. **Regla 3**: Las interrelaciones recursivas 1: N de tipo jerárquica-circular con restricciones de cardinalidad mínima “opcional-obligatoria” son estructuralmente válidas. R**egla 4**: Todas las interrelaciones recursivas con cardinalidad máxima “muchos a muchos” son estructuralmente válidas independiente de las restricciones de cardinalidad mínima. **Regla 5**: Todas las interrelaciones recursivas con cardinalidad mínima “opcional-opcional” son estructuralmente válidas. **Corolario 1**: Todas las interrelaciones recursivas 1:1 con restricciones de cardinalidad mínima “obligatoria-opcional” u “opcional-obligatoria” son estructuralmente inválidas. **Corolario 2**: Todas las interrelaciones recursivas 1: N o N:1 con restricciones de cardinalidad mínima “obligatoria-obligatoria” son estructuralmente inválidas. **Corolario 3**: Todas las interrelaciones recursivas 1: N o N:1 con restricción de participación obligatoria en el lado “uno” y restricción de participación opcional en el lado “muchos” son estructuralmente inválidas.

Reglas para la validación estructural de interrelaciones binarias

Regla 6: Un camino sin ciclos que contenga interrelaciones binarias es siempre estructuralmente válido.

Regla 7: Un camino cíclico que contenga interrelaciones binarias y una o más interrelaciones "opcional-opcional" es siempre estructuralmente válido.

Regla 8: Un camino cíclico que contenga interrelaciones binarias y una o más interrelaciones mucho a uno con participación opcional en el lado "uno" es siempre estructuralmente válido.

Regla 9: Un camino cíclico que contenga interrelaciones binarias y una o más interrelaciones "muchos a muchos" es siempre estructuralmente válido.

Regla 10: Los caminos cíclicos que contengan al menos un conjunto de interrelaciones opuestas son siempre válidos.

Regla 11: Un camino cíclico que contenga interrelaciones binarias 1:1 "obligatoria-obligatoria" es siempre estructuralmente válido.

Corolario 4: Un camino cíclico que no contengan interrelaciones opuestas e interrelaciones de auto-ajuste son estructuralmente inválidos y es llamado Interrelación Circular.

Corolario 5: La presencia de una interrelación 1:1 "obligatoria-obligatoria" no tiene efecto en la validez (o invalidez) de un camino cíclico que contenga otros tipos de interrelaciones.

Reglas para la validación estructural de interrelaciones ternarias

Regla 6 (generalización): Si un camino contiene interrelaciones binarias y ternarias y no tiene ciclos entonces siempre es estructuralmente válido.

Regla 12: Un camino cíclico que contenga interrelaciones ternarias donde no existan restricciones binarias explicitas sobre las entidades de la interrelación ternaria es siempre estructuralmente válido independiente de las cardinalidades máxima y mínima de la interrelación ternaria.

Regla 13: Si la cardinalidad máxima de una interrelación binaria que es impuesta sobre una interrelación ternaria es mayor o igual que la cardinalidad máxima de la interrelación ternaria entre las dos entidades entonces la interrelación ternaria es válida.

Regla 14: Si es necesario definir una segunda interrelación binaria sobre una interrelación ternaria, entonces solo puede ser impuesta entre dos entidades donde la cardinalidad máxima es "muchos a muchos" y el efecto de la segunda interrelación no puede redefinir cualquier interrelación explícita previa o relajar cualquier interrelación binaria derivada.

Regla 15: La cardinalidad mínima de una interrelación que actúa como restricción debe seguir la cardinalidad mínima de la interrelación ternaria que está restringiendo para ser estructuralmente válida.

Corolario 6: Si la cardinalidad máxima de la interrelación binaria impuesta sobre una interrelación ternaria es menor que la cardinalidad máxima de la interrelación ternaria entre las dos entidades involucradas entonces la interrelación ternaria es inválida.

Corolario 7: Si la cardinalidad mínima de una interrelación binaria que actúa como restricción no sigue la cardinalidad mínima de la interrelación ternaria, entonces la interrelación ternaria es inválida.

Para las interrelaciones recursivas se presenta una clasificación de las mismas basada en los roles que una entidad puede desempeñar en la interrelación, lo cual posibilita formular reglas que verifican la consistencia de las restricciones de cardinalidad impuestas en interrelaciones recursivas.

Para el estudio de las interrelaciones binarias se utiliza el concepto de camino (Dullea & Song, 1997), que es la conectividad que se establece entre los tipos de entidades y los tipos de interrelaciones en un esquema ER, y muestran que en un camino cíclico cada conjunto de restricciones de cardinalidad debe ser consistente con el resto de las restricciones de cardinalidad y en ese sentido, establecen las reglas para determinar la validez estructural de las interrelaciones binarias.

Con respecto a las interrelaciones ternarias, en Jones & Song (1996, 2000, 2012); y Song & Jones (1993), se hace un amplio estudio de la coexistencia de interrelaciones ternarias y binarias en caminos cíclicos, se identifican tres tipos de coexistencias de interrelaciones ternarias y binarias: 1) interrelaciones ternarias no restringidas con interrelaciones binarias implícitas entre los tipos de entidades de la interrelación ternaria; 2) interrelaciones ternarias restringidas por interrelaciones binarias; 3) interrelaciones ternarias con interrelaciones binarias que no restringen la interrelación. Del estudio se derivan las reglas que determinan la validez estructural de las interrelaciones ternarias. Otros trabajos, como los que se presentan en McAllister (1998); Santos, Martínez & Cuadra (2007); Thalheim (1992), solo analizan las interrelaciones ternarias de manera independiente, sin tener en cuenta la interacción con otras interrelaciones.

Al hacer una valoración de los métodos descritos, se puede notar que los basados en la programación lineal solo determinan si el conjunto de restricciones estructurales impuestas en el esquema conceptual puedan ser cumplidas totalmente, o sea, que no existan instancias vacías en la base de datos. Los métodos que utilizan la teoría de grafos son7 más precisos que los anteriores, ya que permiten identificar las restricciones de cardinalidad que hacen que el esquema conceptual sea inconsistente.

Entre estos métodos se observan aquellos que utilizan restricciones de cardinalidad con valores mayores que 1 para expresar la cardinalidad mínima y máxima, lo que permite enriquecer la semántica de los esquemas ER; sin embargo este tipo de restricciones de cardinalidad es un subconjunto de las restricciones de cardinalidad menos restrictivas, que son aquellas que utilizan los valores 0 y 1 para la cardinalidad mínima y de 1 y N para la cardinalidad máxima. Al respecto Hartmann, et al. (2009), reconoce que *"un conjunto de restricciones de cardinalidad restrictivas es consistente siempre y cuando sea también consistente el conjunto de restricciones de cardinalidad menos restrictivas"*. En este sentido, el método propuesto en Dullea, et al. (2003), es más general ya que permite detectar los conjuntos de restricciones de cardinalidad inconsistentes que serían detectados con los métodos anteriormente descritos. Por esta razón, se recomienda este método para su implementación en una herramienta CASE.

1.3.3. Validación semántica de esquemas conceptuales

Para el diseño de bases de datos se dispone de numerosas herramientas que no contienen información acerca del mundo real y la forma que el mismo opera y delegan en los usuarios para que estos proporcionen dicha información. Un próximo paso para lograr herramientas inteligentes sería incluir conocimiento del mundo real, organizado por dominios de aplicación, y un método para lograrlo es utilizar una ontología (Gruber, 1995). Una ontología define los términos básicos y las relaciones comprendidas en el vocabulario de un área, así como reglas para combinar términos y relaciones entre términos. Una ontología puede tener términos de muy alto nivel o específicos a un dominio. En el área del diseño de bases de datos se ha reconocido la utilidad de las ontologías de dominio (Sugumaran & Storey, 2006) para clasificar entidades e interrelaciones y contribuir a su automatización (Siau, 2004).

En la modelación conceptual de bases de datos, una interrelación de asociación modela una relación entre dos o más en-

tidades. Por lo general, los diseñadores y herramientas CASE utilizan interrelaciones binarias, en las cuales una interrelación R puede ser expresada de la forma A <frase verbal> B, donde A y B son entidades y el significado de esta frase depende del dominio de aplicación para el cual se esté modelando la base de datos (Franconi, 2011; Yair Wand, Woo & Wand, 2008).

Varios autores reconocen la importancia que tiene el significado de la frase verbal de una interrelación de asociación para el análisis semántico de un esquema conceptual ER. En Storey, 2001, 2005; y Storey & Purao (2004), se propone una ontología para la clasificación de frases verbales de las interrelaciones, con el objetivo de lograr la integración de varias bases de datos. Como limitación se observa que el análisis del significado de las frases verbales solo se circunscribe a las interrelaciones de asociación binarias.

Uno de los trabajos más notables relacionados con la clasificación de frases verbales puede encontrarse en Purao & Storey (2005), que presenta una ontología basada en el dominio de la aplicación para clasificar la semántica de las interrelaciones. La ontología está compuesta de tres capas: el núcleo, el contexto interno y el contexto externo. La capa del núcleo captura los tipos fundamentales de interrelaciones entre entidades. La capa intermedia proporciona el contexto interno, obtenido a partir de las entidades involucradas en la interrelación. La capa externa permite una interpretación más precisa a partir del contexto del dominio donde se utiliza la interrelación. Esta ontología fue desarrollada en un prototipo que realiza las clasificaciones de las frases verbales de forma interactiva. Como restricciones se puede señalar que solo acepta interrelaciones binarias y no hace uso de las restricciones de cardinalidad de las interrelaciones. Su aplicación está orientada a la clasificación de frases verbales con el objetivo de integrar varias bases de datos a partir de fuentes heterogéneas.

La validación semántica de esquemas conceptuales con la ayuda de ontologías de dominio específico es un tema actualmente en desarrollo, en el que se pueden observar resultados discretos en lo relacionado con el desarrollo de herramientas CASE que incorporen este tipo de facilidad. El reto fundamental que impone el diseño conceptual de bases de datos asistido con ontologías, es la representación del conocimiento acerca de un dominio de aplicación determinado para su posterior uso durante la fase de modelado conceptual, lo cual constituye un tema de investigación a desarrollar, que por su magnitud no es abordado en este trabajo.

1.4. Herramientas CASE para el diseño de bases de datos

Una de las primeras etapas del ciclo de desarrollo de un sistema de base de datos, la del diseño conceptual de la base de datos, puede también implicar la selección de herramientas CASE adecuadas. Los diseñadores necesitan estas herramientas para que las actividades de desarrollo de la base de datos se lleven a cabo de la forma más eficiente y efectiva posible. En esta dirección se han propuesto varias clasificaciones de herramientas CASE basadas, entre otros aspectos, en el ciclo de vida del proyecto (análisis, diseño, implementación, prueba, mantenimiento); en el nivel de abstracción (CASE de alto nivel, CASE de bajo nivel y CASE integrado); en el grado de automatización (semiautomatizadas, interactivas o completamente automatizadas) (Connolly & Begg, 2014a; Piattini & Díaz, 2012b).

Las herramientas CASE de alto nivel soportan las etapas iniciales del ciclo de desarrollo de los sistemas de base de datos, desde la planificación hasta su diseño. Las de bajo nivel soportan las últimas etapas del ciclo de vida, desde la implementación a las pruebas y el mantenimiento operativo. Las herramientas CASE integradas soportan todas las etapas del ciclo de desarrollo y proporcionan, por tanto, la funcionalidad de las herramientas CASE de alto y bajo nivel en una única herramienta.

1.4.2. Características de herramientas CASE de ayuda al diseño de bases de datos

Entre las herramientas CASE para el diseño de bases de datos se identifican tres tipos ateniendo a la manera en que se construye el esquema conceptual: aquellas que ayudan a un diseño creativo por parte del usuario; las que permiten obtener un diseño conceptual a partir de ficheros o bases de datos existentes y aquellas que obtienen el esquema conceptual a partir de sentencias del lenguaje natural (Piattini & Díaz, 2012b).

Esta investigación se centra en el primer tipo de herramientas, en las que la actividad de modelación generalmente comienza desde cero, a partir del proceso de abstracción que el diseñador hace del universo del discurso. Estas herramientas se caracterizan por una interfaz de usuario que da soporte al modelo conceptual y por un diccionario de datos donde se almacenarán los esquemas resultantes. Para que la interfaz de usuario tenga éxito debe cumplir que sea amigable, de fácil uso y que permita expresividad semántica. Esta depende del modelo conceptual utilizado. Un modelo rico en semántica reduce la brecha entre una percepción y su representación formal. Un modelo conceptual pobre desde el punto de vista semántico requiere de más habilidades en el diseño para expresar hechos del universo del discurso en el modelo conceptual.

Aunque el proceso de diseño está basado en el uso de una interfaz, que permite construir un diagrama, se requiere de un soporte mínimo en términos de verificación de la sintaxis, la estructura y la semántica del diagrama. La verificación sintáctica comprende el chequeo de reglas inherentes al modelo ER. Un esquema conceptual es sintácticamente correcto cuando los conceptos se definen con propiedad en el esquema (Batini, Navathe, Ceri, Martín García & Romero Ibancos, 2014).

La verificación estructural o validación estructural comprueba si la consideración conjunta de todas las restricciones estructurales (restricciones de cardinalidad) del esquema conceptual no implica ninguna inconsistencia lógica en cualquiera de los posibles estados de la base de datos (Dullea & Song, 1998a, 1998b, 1999; Thalheim, 1992; Dullea, et al., 2003). Además de la verificación sintáctica y estructural las herramientas deben proporcionar algún chequeo semántico. La validación semántica determina si los tipos de entidades y tipos de interrelaciones del esquema conceptual representan exactamente los conceptos del dominio que se esté modelando (Kolapalli & Dullea, 2012). Como esta validación depende del dominio de la aplicación resulta difícil definir un criterio de validez generalizado. En la actualidad la validación semántica como ayuda al diseño de bases de datos no está disponible en la mayoría de las herramientas CASE comerciales y se muestran algunos resultados en prototipos. Con estas funcionalidades no es solo un editor gráfico sino un sistema inteligente que permite realizar diferentes procesos de validación. Estas facilidades contribuyen a mejorar la calidad de los esquemas generados.

1.4.3. Estudio comparativo de herramientas CASE de ayuda al diseño de bases de datos

Las herramientas fueron seleccionadas atendiendo a:

- Que soporten las fases iniciales del diseño de una base de datos (conceptual, lógico y físico), utilizando alguna notación del modelo ER para la elaboración del esquema conceptual.
- En el caso de herramientas integradas, por ejemplo Visual Paradigm for UML, se analizaron las funcionalidades orientadas al diseño de la base de datos.

La tabla 1.7 muestra una lista de herramientas CASE utilizadas para realizar el estudio comparativo.

Tabla 1.7. Relación de herramientas CASE evaluadas.

Herramienta CASE	Versión	Página web	Compañía
Case Studio 2	2.18	http://www.casestudio.com	CharonWare
ConceptDraw	8.0.2	http://www.conceptdraw.com/en/products/cd5/applications_uml.php	CS Odessa ConceptDraw
Data Architect	3.0.0	http://www.thekompany.com	TheKompany
Database Architect	1.8.0	http://www.gurudevelopers.com	Guru Developers
Database Design Studio	2.21.3	http://www.dds-pro.com/index.html	Chilli Source
Database Design Tool	1.5	http://janssens.dhs.org/software/	Jo Janssens
DBDesigner	4.0.5.6	http://fabforce.net	Fabulous Force
DeZign	4	http://www.datanamic.com/dezign/index.html	Datanamic
Dia	0.91	http://www.gnome.org/projects/dia/	Gnome Office
EasyCase Professional	4.21.016	http://www.esti.com	Evergreen Software Tools
ER Creator	3.2	http://www.modelcreator.com	Model Creator
ER Diagrammer	8.1.7.6	http://www.keeptool.com	KeepTool
ER/Studio	7.1	http://www.embarcadero.com	Embarcadero
ERWin	7.3.0.1	http://www.ca.com	Computer Associates
Oracle Designer	10e	http://otn.oracle.com/products/designer/index.html	Oracle Corporation
PGDesigner	2	http://www.hardgeus.com/projects/pgdesigner/	John R. Mcwley III

SmartDraw	7.01	http://www.smartdraw.com	SmartDraw
SQLDesigner	2.3.2	http://code.google.com/p/wwwsqldesigner/	Ondrej Zara
Toad Data Modeler	3.3.8.11	http://quest.com	Quest Software
Visio	2003	http://www.microsoft.com/office/visio/default.asp	Microsoft Corporation
Visual Paradigm for UML	6.0	http://www.visual-paradigm.com/product/vpuml/	Visual Paradigm International
XCase	8.1	http://www.xcase.com	Resolution Software
XTG Data Modeller	2.3.4	http://www.xtgsystems.com	XTG Systems

Los aspectos a tener en cuenta para realizar el estudio comparativo fueron los siguientes:

1. Notación y construcciones del modelo ER.
2. Especificación de restricciones para cada uno de tipos de interrelaciones soportadas.
3. Validación de esquemas.

La notación y construcciones del modelo ER utilizadas por una herramienta permiten determinar el grado de expresividad de los esquemas conceptuales que se generan. Los modelos conceptuales difieren en la elección y número de las distintas estructuras de modelado que ofrecen. En este sentido, Batini, et al. (2014), señalan que *"en general, la disponibilidad de una amplia gama de conceptos hace posible una representación más extensa de la realidad; por este motivo, los modelos más ricos en conceptos son también más expresivos".* El grado de expresividad de los esquemas conceptuales elaborados por las herramientas CASE, será evaluado a partir de la variedad de construcciones del modelo ER utilizadas por las mismas.

La manera en que se establecen las restricciones para los distintos tipos de interrelaciones (asociación, jerarquías de generalización, débiles) permite evaluar en qué grado la herramienta expresa la semántica del universo del discurso en el esquema conceptual. Las facilidades de validación de esquemas de una herramienta CASE traen como consecuencia una mejora en la calidad de los esquemas generados.

1.4.3.1. Notación y construcciones

Del estudio anterior se pudo constatar que la mayoría de las herramientas comienzan el proceso de diseño con la elaboración de un esquema conceptual utilizando alguna notación del modelo ER; otras usan los términos de tabla y relación para realizar el diseño físico de la base de datos y el resto solo dibujan el diagrama ER y no permiten generar el esquema físico de la base de datos. En la tabla 1.8 se muestra una agrupación en dependencia de las fases de diseño soportadas por las herramientas evaluadas.

Tabla 1.8. Fases de diseño soportadas por las herramientas.

Criterio	Herramientas
Comienza la fase de diseño a partir del esquema conceptual.	Case Studio 2, Data Architect, Database Design Studio, DBDesigner, DeZign, EasyCase, ER Creator, ERStudio, ERwin, Oracle Designer, Toad Data Modeler, Visual Paradigm for UML, XCase, XTG Data Modeller.
Comienza la fase de diseño a partir del esquema físico.	ER Diagrammer, Database Architect, Database Design Tool, PGDesigner, SQL Designer.
Permiten dibujar el diagrama ER.	ConceptDraw, Dia, SmartDraw, Visio.

Es de interés enfocar el análisis de aquellas herramientas que inician el proceso de diseño a partir de un esquema conceptual utilizando alguna notación del modelo ER. En este sentido las herramientas objeto de estudio son las que aparecen en la tabla 1.9.

Tabla 1.9. Notaciones del modelo ER utilizadas por las herramientas.

Herramienta	Notación del modelo ER
Case Studio 2	Information Engineering
Data Architect	Information Engineering
Database Design Studio	Chen
DBDesigner	Teorey
Dezign	Information Engineering
Oracle Designer	CASE*Method
EasyCase	Information Engineering, IDEF1X, Chen, Bachman
ER Creator	Information Engineering
ER/Studio	Information Engineering, IDEF1X
ERwin	Information Engineering, IDEF1X
Toad Data Modeler	Information Engineering, IDEF1X
Visual Paradigm for UML	Information Engineering
XCase	Information Engineering
XTG Data Modeller	Information Engineering

Según Chen (1983), y Song, et al. (1995), las notaciones del modelo ER se pueden clasificar en modelos binarios o en modelos n-arios. En los modelos binarios no se pueden expresar interrelaciones de asociación de grado mayor que dos, mientras que en los modelos n-arios sí. La tabla 1.10 muestra las notaciones utilizadas por las herramientas evaluadas según esta clasificación.

Tabla 1.10. Notaciones de las herramientas evaluadas que permiten interrelaciones binarias y n-arias.

Criterio	Notación del modelo ER
Solo permiten interrelaciones binarias.	Information Engineering, IDEF1X, Bachman, CASE*Method.
Permiten interrelaciones de grado mayor que dos.	Chen, Teorey.

En el estudio se pudo constatar que la mayoría de las herramientas evaluadas utilizan modelos binarios. Estas notaciones se caracterizan por:

- Soportar las siguientes construcciones: entidad regular, entidad débil, interrelaciones de asociación recursivas y binarias, interrelaciones débiles con dependencia de identificación y jerarquías de generalización/especialización.
- No permitir atributos en las interrelaciones. Cualquier objeto que tenga al menos un atributo es considerado una entidad, de esta manera, las interrelaciones "muchos a muchos" que tengan al menos un atributo deben ser modeladas como un nuevo tipo de entidad.
- No se permiten interrelaciones ternarias. Para modelar este tipo de interrelación el diseñador puede expresarla con interrelaciones binarias siempre que no exista pérdida de la semántica (Dey, Storey, & Barron, 1999; Jones & Song, 2000, 2012), o utilizar una clase de asociación conectada por interrelaciones binarias a cada uno de los tres tipos de entidades (Dey, et al., 1999; Hitchman, 2004), o utilizar un tipo de entidad débil conectada por medio de interrelaciones débiles a cada uno de los tres tipos de entidades (García-Molina, et al., 2012). Nótese que el uso de estas alternativas depende en gran medida de la experiencia del diseñador.

Por lo general, los modelos n-arios son más expresivos que los modelos binarios y permiten la modelación directa de interrelaciones "muchos a muchos" con atributos y de interrelaciones ternarias. En la tabla 1.11 se muestran las construcciones soportadas por las herramientas evaluadas.

Tabla 1.11. Notaciones y construcciones utilizadas por las herramientas evaluadas.

Herramienta	Notación del modelo ER	Tipos de entidades	Tipos de interrelaciones
Case Studio 2	Information Engineering	• fuerte	• interrelación de asociación binaria 1:1, 1:N, N:M. • interrelación débil con dependencia de identificación.

Data Architect	Information Engineering	• fuerte	• interrelación de asociación recursiva y binaria. • interrelación débil con dependencia de identificación.
Database Design Studio	Chen	• fuerte • débil	• Interrelación de asociación recursiva y binaria. • interrelación débil. • interrelación de subconjunto. • jerarquía de generalización / especialización.
DBDesigner	Teorey	• fuerte • débil	• interrelación de asociación recursiva 1:1, 1:N, N:M • interrelación de asociación binaria 1:1, 1:N, N:M • interrelación débil con dependencia de identificación. • interrelación de subconjunto.
DeZign	Information Engineering	• fuerte	• interrelación de asociación recursiva y binaria. • interrelación débil con dependencia de identificación.

EasyCase	Information Engineering	• fuerte	• interrelación de asociación recursiva y binaria. • Interrelación de subconjunto.
	IDEF1X	• fuerte • débil	• interrelación de asociación recursiva y binaria. • interrelación débil con dependencia de identificación. • jerarquía de generalización / especialización.
	Chen	• fuerte • débil	• Interrelación de asociación recursiva y binaria. • interrelación débil. • interrelación de subconjunto.
	Bachman	• fuerte	• Interrelación de asociación recursiva y binaria, • interrelación de subconjunto.
ER Creator	Information Engineering	• fuerte	• interrelación de asociación recursiva y binaria. • Interrelación débil con depedencia de indentificación.

ER/Studio	Information Engineering	• fuerte • débil	• interrelación de asociación recursiva 1:1, 1:N. • interrelación de asociación binaria 1:1, 1:N, N:M. • interrelación débil con dependencia de identificación. • interrelación de subconjunto. • jerarquía de generalización/especialización.
	IDEF1X	• fuerte • débil	• interrelación de asociación recursiva 1:1, 1:N. • interrelación de asociación binaria 1:1, 1:N, N:M. • interrelación débil con dependencia de identificación. • interrelación de subconjunto. • jerarquía de generalización/especialización.

ERwin	Information Engineering	• fuerte • débil	• interrelación de asociación recursiva 1:1, 1:N, N;M. • interrelación de asociación binaria 1:1, 1:N, N:M. • interrelación débil con dependencia de identificación. • interrelación de subconjunto. • jerarquía de generalización/ especialización.
	IDEF1X	• fuerte • débil	• interrelación de asociación recursiva 1:1, 1:N. N:M. • interrelación de asociación binaria 1:1, 1:N, N:M. • interrelación débil con dependencia de identificación. • interrelación de subconjunto. • jerarquía de generalización/ especialización.

Oracle Designer	CASE*Method	• fuerte	• interrelaciones de asociación, recursivas y binarias. • Interrelación de subconjunto. • Interrelación de generalización/especialización.
Toad Data Modeler	Information Engineering	• fuerte	• interrelación de asociación recursivas y binarias 1:1, 1;N, N:M. • interrelación débil con dependencia de identificación.
	IDEF1X	• fuerte	• interrelación de asociación recursivas y binarias 1:1, 1;N, N:M, • interrelación débil con dependencia de identificación.
Visual Paradigm for UML	Information Engineering	• fuerte	• interrelación de asociación recursivas y binarias 1:1, 1;N, N:M. • interrelación débil con dependencia de identificación.
XCase	Information Engineering	• fuerte	• interrelación de asociación recursivas y binarias 1:1, 1;N. • interrelación débil con dependencia de identificación. • Interrelación de subconjunto.

XTG Data Modeller	Information Engineering	• fuerte	• interrelación de asociación recursivas y binarias 1:1, 1;N. • interrelación débil con dependencia de identificación.

Para modelar interrelaciones de asociación de hasta grado dos, la mayoría de las herramientas soportan la interrelación de asociación binaria con razón de cardinalidad 1:1, 1:N y N:M. Durante el proceso de diseño, el orden en que se establece la interrelación entre los dos tipos de entidades (para las interrelaciones de asociación con cardinalidad máxima 1:1 y 1:N), determina el sentido de la migración del atributo identificador, lo cual hace que el proceso de transformación del esquema conceptual al esquema relacional sea más sencillo. Sin embargo, esto trae como consecuencia que en la modelación conceptual estén presentes términos que corresponden al modelo lógico y físico de la base de datos, como son: llave primaria, llave extranjera, índices, entre otros. Un tratamiento diferente se observa en la herramienta Database Design Studio donde el orden en que se establece la interrelación de asociación entre los tipos de entidades no es significativo por lo que el proceso de obtención del esquema lógico se basa en reglas de transformación como las que se formulan en Elmasri & Navathe (2012); Date (2013); Ponniah (2013); Teorey, et al., (2014).

Para establecer las restricciones de cardinalidad algunas notaciones permiten especificar de manera independiente los valores para la cardinalidad mínima y máxima, en otras las restricciones de cardinalidad se expresan como combinaciones de la cardinalidad mínima y máxima, lo cual oscurece su interpretación, tal es el caso de la notación IDEF1X. Aunque la notación Information Engineering permite la especificación independiente de los valores de la cardinalidad mínima y máxima, en las herramientas ER/Studio y ERwin el tratamiento es similar al de la notación IDEF1X (véase la tabla 1.12).

Tabla 1.12. Formas de especificar las restricciones de cardinalidad.

Criterio	Notación del modelo ER	Herramienta
Especificación independiente de los valores de la cardinalidad mínima y máxima.	Information Engineering, Chen, Teorey, Bachman, CASE*Method.	Case Studio 2, Data Architect, Database Design Studio, DBDesigner, DeZign, EasyCase, Oracle Designer, Toad Data Modeler.
Combinación de los valores de la cardinalidad mínima y máxima.	IDEF1X, Information Engineering.	EasyCase, ERCreator, ERStudio, ERwin, Visual Paradigm for UML, XCase, XTG Data Modeller.

Con respecto a las jerarquías de generalización/especialización se observa que alrededor del 50% de las herramientas evaluadas no ofrecen este tipo de construcción, lo que limita la capacidad del diseñador para modelar de forma natural hechos utilizando la abstracción de generalización.

De las herramientas que soportan jerarquías de generalización/especialización se analizó cómo estas especifican las restricciones de participación y de disyunción. Se pudo observar que algunas herramientas permitían establecer las restricciones de participación y de disyunción, mientras que otras, solo una de ellas. Por ejemplo, en la herramienta ERStudio para las notaciones IDEF1X e Information Engineering se pueden especificar los dos tipos de restricciones, mientras que en la herramienta ERwin en la notación IDEF1X solo se puede establecer la restricción de participación y en la notación Information Engineering solo se permite establecer la restricción de disyunción. En el tabla 1.13, aparece para cada herramienta, cómo se especifican las restricciones en las interrelaciones de asociación y en las jerarquías de generalización/especialización.

Tabla 1.13. Especificación de restricciones para las interrelaciones de asociación y las jerarquías de generalización/especialización.

Herramienta	Notación del modelo ER	Restricciones de cardinalidad en interrelaciones de asociación	Restricciones de participación y disyunción en jerarquías de generalización
Case Studio 2	Information Engineering	Especificación independiente de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.
Data Architect	Information Engineering	Especificación independiente de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.
Database Design Studio	Chen	Especificación independiente de los valores de la cardinalidad mínima y máxima.	Permite establecer restricciones de participación y de disyunción.
DBDesigner	Teorey	Especificación independiente de los valores de la cardinalidad mínima y máxima.	Solo soporta la interrelación de subconjunto. Permite establecer la restricción de participación.
DeZign	Information Engineering	Especificación independiente de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.

EasyCase	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	Solo soporta la interrelación de subconjunto. No se permite establecer restricciones.
	IDEF1X	Combinación de los valores de la cardinalidad mínima y máxima.	No se permite establecer restricciones.
	Chen	Especificación independiente de los valores de la cardinalidad mínima y máxima.	Solo soporta la interrelación de subconjunto. No se permite establecer restricciones.
	Bachman	Especificación independiente de los valores de la cardinalidad mínima y máxima.	Solo soporta la interrelación de subconjunto. No se permite establecer restricciones.
ER Creator	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.
ER/Studio	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	Permite establecer restricciones de participación y de disyunción.
	IDEF1X	Combinación de los valores de la cardinalidad mínima y máxima.	Permite establecer restricciones de participación y de disyunción.

ERwin	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	Solo permite establecer restricciones de disyunción.
	IDEF1X	Combinación de los valores de la cardinalidad mínima y máxima.	Solo permite establecer restricciones de participación.
Oracle Designer	CASE*Method	Especificación independiente de los valores de la cardinalidad mínima y máxima.	No se permite especificar restricciones. Por defecto la interrelación tiene restricción de participación total y la restricción de disyunción es disjunta.
Toad Data Modeler	Information Engineering	Especificación independiente de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.
	IDEF1X	Especificación independiente de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.
Visual Paradigm for UML	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.
XCase	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	Solo soporta la interrelación de subconjunto. Permite establecer la restricción de participación.

XTG Data Modeller	Information Engineering	Combinación de los valores de la cardinalidad mínima y máxima.	No soporta este tipo de interrelación.

1.4.3.2. Validación de esquemas

Las capacidades de validación de los diferentes esquemas de bases de datos fueron verificadas en cada una de las herramientas objeto de estudio. Como resultado se pudo comprobar que solo un número reducido de ellas llevan a cabo algún tipo de validación parcial del esquema conceptual. Un resumen de este estudio se muestra en la tabla 1.14.

Tabla 1.14. Soporte para la validación estructural y semántica proporcionada en cada herramienta.

Herramienta	¿Realiza validación del esquema conceptual?	
	Estructural	Semántica
Case Studio 2	No	No
Data Architect	No	No
Database Design Studio	Detecta ciclos en el diagrama ER.	No
DBDesigner	Detecta ciclos en el diagrama ER.	No
DeZign	No	No
Oracle Designer	No	No
EasyCase	No	No
ER Creator	No	No
ER/Studio	No	No
ERwin	No	No
Toad Data Modeler	No	No
Visual Paradigm for UML	No	No
XCase	No	No
XTG Data Modeller	No	No

Como puede observarse, las herramientas DBDesigner y Database Design Studio realizan un proceso de validación estructural que consiste en la detección de ciclos en el diagrama ER. El criterio que se sigue para decidir la validez de un ciclo no tiene en cuenta las restricciones de participación de las interrelaciones y siempre asume que las llaves extranjeras no admiten el valor nulo cuando se obtienen los esquemas relacionales.

En cuanto al soporte de validación semántica, esta facilidad no está disponible en ninguna de las herramientas evaluadas. Esta facilidad tampoco está disponible para los esquemas lógicos que se obtienen en el proceso de transformación.

Del estudio realizado de las herramientas de ayuda al diseño de bases de datos se puede apreciar:

- Algunas se ocupan de aspectos que no corresponden a la fase de modelación conceptual de la base de datos.
- Disponen de un conjunto reducido de construcciones del modelo ER.
- Por lo general, se nota la ausencia de procesos de validación de los esquemas de bases de datos.

Capítulo II. Sistematización del concepto de dependencia de existencia

2.1. La dependencia de existencia en construcciones del modelo ER

Como resultado del estudio realizado del modelo ER se ha podido constatar que el concepto de dependencia de existencia ha sido tratado de manera informal e incompleta en las construcciones del mismo; específicamente en las interrelaciones de asociación no ha sido abordado en sus extensiones, sin embargo su consideración ayuda a modelar muchos hechos interesantes asociados con su inmutabilidad en el tiempo. En este capítulo se hace una sistematización del comportamiento de las entidades con dependencia de existencia para una amplia variedad de construcciones del modelo ER, el cual queda expresado en forma de sentencias en el lenguaje SQL3 (American National Standards Institute, 1999), de manera que estas puedan ser implementadas de forma automática en una herramienta CASE para diseñadores de bases de datos. Como resultado de la extensión de la semántica de las interrelaciones de asociación con dependencia de existencia se propone una notación con su correspondiente regla de transformación a esquemas relacionales.

El concepto de dependencia de existencia se ha tratado de forma intuitiva e incompleta en varias construcciones del modelo ER que tienen su última expresión en las llaves extranjeras en el nivel lógico. La dependencia de existencia puede tener diferentes comportamientos que se han intentado describir con varias definiciones (Guizzardi, 2005; Snoeck & Dedene, 1998, 2001; Snoeck, Michiels, & Dedene, 2003; Guizzardi & Wagner, 2008; McBrien, 2010; Olivé, 2012a; Silberschatz, et al., 2014; Simsion & Witt, 2015), algunas de las cuales se analizan a continuación.

Definición 1. Sea ϵ el predicado que denota existencia. Se tiene que una instancia x es dependiente de existencia de otra instancia y (denotada como ed(x,y)), siempre que se cumpla que y

debe existir para que x exista, o formalmente: ed(x,y) =def (ϵ(x) → ϵ(y)) [4] (Guizzardi, 2005; Guizzardi & Wagner, 2008).

En los enfoques asociados a la dependencia de existencia en el contexto del modelo ER, se ha supuesto de una manera u otra esta definición, que se manifiesta en diversas construcciones como son: entidades débiles, jerarquías de generalización/especialización y agregaciones. Esta definición sugiere de manera implícita una relación entre llaves para expresar la dependencia.

Definición 2. Sean E_1 y E_2 tipos de entidades. Se dice que E1 tiene dependencia de existencia de *E2* (se denota $E_1 \leftarrow E_2$), si y solo si, una entidad e_1 de E_1 está asociada con solo una y siempre la misma entidad e_2 de E_2. Se llamará a e1 entidad dependiente o subordinada y a e_2 entidad dominante, donde E_1 es el tipo de entidad dependiente y E_2 es el tipo de entidad dominante (Snoeck & Dedene, 1998, 2001; Snoeck, et al., 2003).

En esta definición, la dependencia de existencia exhibe un comportamiento extendido con respecto a la definición 1, debido a que se establece con una y siempre la misma entidad dominante durante todo el tiempo de vida de la entidad dependiente, lo que se expresa en la inmutabilidad del vínculo entre estas instancias. Esta inmutabilidad se materializa en hacer que en el atributo correspondiente a la parte "uno" de la interrelación no sea actualizable. Esta simple observación permite ampliar las implicaciones que también se derivan de la definición 1, relativa a la inmutabilidad de las llaves extranjeras que se han generado como parte de las construcciones que reflejan dependencia de existencia.

Aunque la definición anterior se refiere a interrelaciones de asociación con cardinalidades 1:1 y N:1, la idea que sugiere sobre la no actualización del atributo correspondiente a la parte "uno", es generalizable a una interrelación de asociación N:M, haciendo notar que en este caso hay que indicar cuál de los lados de la interrelación no es actualizable, o lo que es lo mismo, que tiene dependencia de existencia. Si bien esta observación puede resultar natural, las interrelaciones de asociación N:M no han

4 El operador → significa implicación.

sido objeto de estudio en este trabajo. De acuerdo con las definiciones anteriores, se formaliza el comportamiento asociado a la dependencia de existencia para diversas construcciones del modelo ER.

2.1.1. Entidades débiles

Sea *E* un tipo de entidad regular o fuerte, *W* un tipo de entidad débil y *R* un tipo de interrelación débil como se muestra en la figura 2.1.

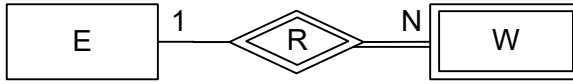


Figura 2.1. Diagrama ER de una interrelación débil.

Fuente: Elmasri & Navathe (2012).

La regla para transformar la interrelación débil *R* a esquemas relacionales es la siguiente: para el tipo de entidad débil W se crea un esquema relacional *D* donde se incluyen todos los atributos descriptores del tipo de entidad *W* como atributos de *D*. Además, se incluye como llave extranjera de *D* la llave primaria del esquema relacional correspondiente al tipo de entidad fuerte *E*. La llave primaria del esquema relacional *D* queda constituida por la combinación de la llave primaria del tipo de entidad *E* y la llave parcial del tipo de entidad débil *W*. Para reflejar la dependencia de existencia se debe especificar como acciones referenciales sobre la llave extranjera del esquema relacional *D* la eliminación y actualización en cascada de las entidades del tipo de entidad débil *W* de las entidades del tipo de entidad fuerte *E*.

De modo general, los esquemas relacionales y las sentencias SQL para transformar la interrelación débil de la figura 2.1 quedan definidos como:

E (atributos_identificadores_fuertes, otros_atributos)

D (atributos_identificadores_fuertes, atributos_identificadores_parciales, otros_atributos)

```
CREATE TABLE E (
        atributos_identificadores_fuertes,
        otros_atributos,
        PRIMARY KEY (atributos_identificadores));
CREATE TABLE D (
        atributos_identificadores_fuertes,
        atributos_identificadores_parciales,
        otros_atributos,
        FOREIGN KEY (atributos_identificadores_fuertes)
                REFERENCES E (atributos_identificadores)
                ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY KEY (atributos_identificadores_fuertes,
                atributos_identificadores_parciales));
```

Donde, atributos_identificadores es una colección de definiciones que especifica el nombre de la columna y el dominio para los respectivos atributos que constituyen la llave del tipo entidad *E* y otros_atributos es la declaración del resto de los atributos del tipo de entidad que no constituyen la llave primaria. De manera similar para el esquema *D*, teniendo en cuenta que atributos_identificadores_parciales es la definición de los atributos que constituyen la llave parcial del tipo de entidad débil W los cuales pueden estar presente o no.

Si la cardinalidad máxima del tipo de entidad *E* es "uno" entonces es necesario agregar la restricción UNIQUE siguiente a la definición de la tabla *D*:

```
UNIQUE (atributos_identificadores_fuertes)
```

2.1.2. Generalización/especialización

Diversos autores (Elmasri & Navathe, 2012; Ponniah, 2013; Batini, et al., 2014; Connolly & Begg, 2014b; Silberschatz, et al., 2014; Teorey, et al., 2014; Simsion & Witt, 2015) han definido varias opciones para transformar una jerarquía de generalización/especialización en esquemas relacionales. La selección de la opción más apropiada depende, entre otros factores, de las restricciones de participación y disyunción, del número de entidades que participan en la interrelación y de si los tipos de entidades especializados están implicados en otras interrelaciones. El número de esquemas relacionales que se obtienen al transformar este tipo de interrelación depende de la opción seleccionada, que puede comprender desde la generación de un esquema relacional por cada tipo de entidad hasta la obtención de un solo esquema.

A pesar de la diversidad de opciones definidas por cada autor, existe consenso en que la opción de generar un esquema de relación por cada tipo de entidad que participa en la interrelación es la más general, pues permite representar todas las combinaciones de restricciones de participación y de disyunción, razón por la cual será tomada como referente.

Sea G el tipo de entidad más general y $E_1, E_2, \ldots, E_n$ los tipos de entidades especializados que participan en una jerarquía de generalización/especialización (véase la figura 2.2) en la que se especifican las restricciones de participación y disyunción según la notación de Elmasri & Navathe (2012).

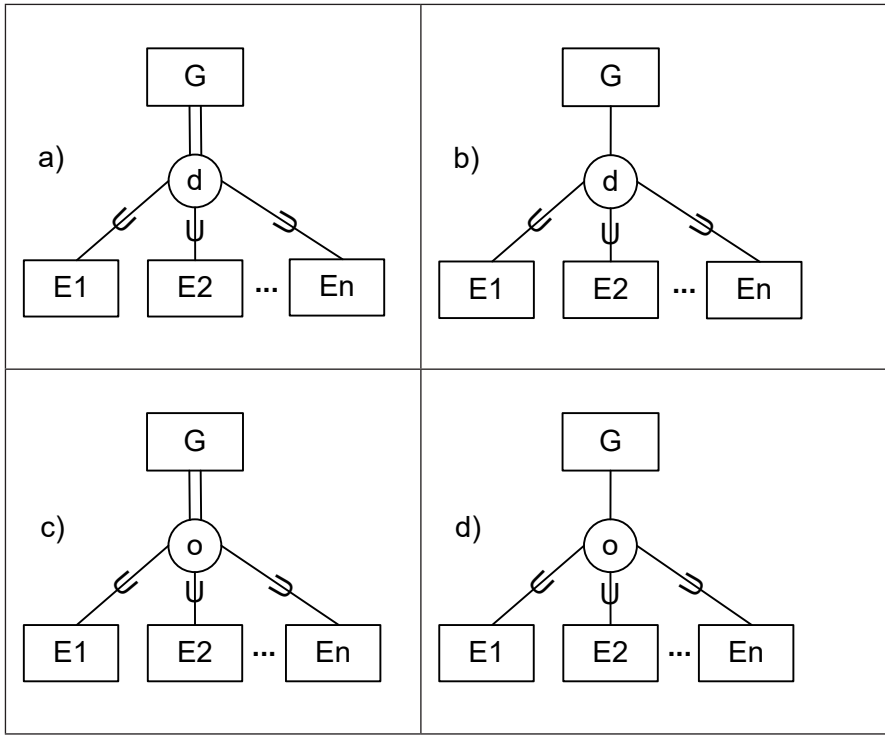


Figura 2.2. Representación de una jerarquía de generalización/especialización donde se especifican las restricciones de participación y de disyunción: a) total y disjunta, b) parcial y disjunta, c) total y solapada, d) parcial y solapada.

Fuente: Elmasri & Navathe (2012).

Los esquemas relacionales y las sentencias SQL para transformar la jerarquía de generalización/especialización de la Figura 2.2 quedan definidos como:

G (atributos_identificadores, otros_atributos_generales)

E1 (atributos_identificadores, otros_atributos_específicos)

. . .

En (atributos_identificadores, otros_atributos_específicos)

CREATE TABLE G (

atributos_identificadores,

otros_atributos_generales,

PRIMARY KEY (atributos_identificadores));

Se debe crear una tabla Ei ($i = \overline{1,n}$) para cada uno de los esquemas relacionales obtenidos a partir de los tipos de entidades especializados.

CREATE TABLE Ei (

atributos_identificadores,

otros_atributos_específicos,

FOREIGN KEY (atributos_identificadores)

REFERENCES G (atributos_identificadores)

ON DELETE CASCADE ON UPDATE CASCADE,

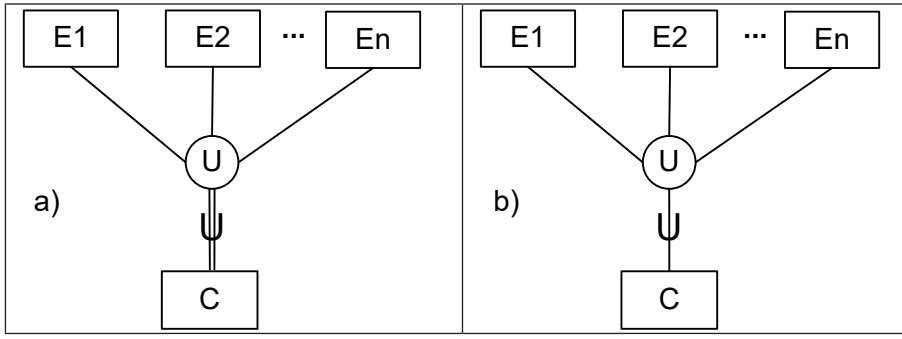
PRIMARY KEY (atributos_identificadores));

Obsérvese que se definen tantas tablas Ei como especializaciones se requieran. Nótese las acciones referenciales de eliminación y actualización en cascada en cada una de las tablas de las especializaciones que garantizan la dependencia de existencia entre las entidades de los tipos de entidad especializados con las correspondientes entidades del tipo de entidad genérico.

2.1.3. Categorización

Para la transformación de la categorización se crea un esquema relacional por cada tipo de entidad de orden superior y uno por el tipo de entidad que representa a la categoría. Se debe crear una llave subrogada[5] en el esquema relacional correspondiente a la categoría, ya que por lo general los dominios de valores de los atributos identificadores de los tipos de entidades de orden superior son diferentes. Esta llave subrogada se adiciona como llave extranjera a cada esquema relacional correspondiente a los tipos de entidades de orden superior, la cual debe admitir el valor nulo para indicar que una entidad específica no pertenece a la categoría. Como las entidades del tipo de entidad que representa la categoría tienen dependencia de existencia de las entidades que categoriza, las acciones referenciales deben ser controladas a través de la definición de disparadores asociados a las tablas que representan a los tipos de entidades de nivel superior, ya que no es posible hacerlo mediante la cláusula FOREIGN KEY debido a que se hace referencia a más de una tabla.

Sea *C* el tipo de entidad categorizado y $E_1, E_2, \ldots, E_n$ los tipos de entidades de orden superior que participan en una interrelación de categorización (véase la figura 2.3), en la que se especifican las restricciones de participación, según la notación de Elmasri & Navathe (2012).



5 El concepto de subrogado fue utilizado en el modelo RM/T de Codd (Codd, 1990) y representa un identificador que distinguirá a cada entidad al margen de sus atributos durante toda su vida.

Figura 2.3. Representación de la interrelación de categorización donde se especifican las restricciones de participación: a) total, b) parcial.

Fuente: Elmasri & Navathe (2012).

Para cada una de las opciones descritas se presentan los esquemas relacionales y sentencias SQL generales que realizan la transformación de una interrelación de categorización, especificando además, el comportamiento de la dependencia de existencia de las entidades del tipo de entidad categorizado.

C (llave_subrogada, otros_atributos)
E1 (atributos_identificadores_E1, otros_atributos, llave_subrogada)
E2 (atributos_identificadores_E2, otros_atributos, llave_subrogada)
. . .
En (atributos_identificadores_En, otros_atributos, llave_subrogada)

CREATE TABLE C(

llave_subrogada dominio,

otros_atributos,

PRIMARY KEY (llave_subrogada));

Se debe crear una tabla Ei ($i = 1, n$) para cada uno de los esquemas relacionales obtenidos a partir de los tipos de entidades de orden superior.

CREATE TABLE Ei(

atributos_identificadores,

otros_atributos,

llave_subrogada dominio,

FOREIGN KEY (llave_subrogada)

REFERENCES C (llave_subrogada)

ON DELETE RESTRICT ON UPDATE RESTRICT,

PRIMARY KEY (atributos_identificadores));

Si bien la transformación de esta construcción aparece tratada en la literatura, no se completa el comportamiento implícito asociado a la dependencia de existencia de las entidades del tipo de entidad C (categoría), por lo que se propone expresarlo a través de disparadores que se definen en cada una de las tablas que se corresponden con los esquemas de los tipos de entidades de nivel superior de esta construcción.

Para cada tabla Ei ($i = \overline{1,n}$) crear el siguiente disparador:

CREATE TRIGGER Eliminar_Entidad_En_Ei

AFTER DELETE ON Ei

DELETE C

WHERE llave_subrogada = Old.llave_subrogada;

2.1.4. Agregación

En esta sección se presentan las reglas para transformar esquemas relacionales de dos tipos de agregaciones presentes en el modelo EER: la agregación como tipo de entidad agregada de mayor nivel y la agregación que modela una interrelación entre las “partes” y un “todo”.

La transformación de un tipo de entidad agregada a esquemas relacionales se realiza siguiendo un conjunto de reglas, como por ejemplo, las definidas en (Elmasri & Navathe, 2012; Ponniah, 2013; Toby Teorey, et al., 2014), se obtiene un nuevo esquema, como consecuencia de externalizar la interrelación de asociación contenida en el tipo de entidad agregada[6]. Si la interrelación de asociación y el tipo de entidad agregada tienen atributos propios, entonces estos se adicionan al nuevo esque-

6 La interrelación de asociación puede ser de cualquier grado.

ma, además se añaden las acciones referenciales que permiten establecer el comportamiento de la dependencia de existencia de las entidades del tipo de entidad agregada.

Como una entidad agregada puede contener una interrelación de asociación de cualquier grado, se tomará como caso de estudio una interrelación de asociación binaria. De manera similar se pueden definir las transformaciones para interrelaciones de asociaciones recursivas, ternarias o de grado mayor.

Sea *A* un tipo de entidad agregada formada por la interrelación de asociación binaria $R(E_1, E_2)$ representada en la Figura 2.4.

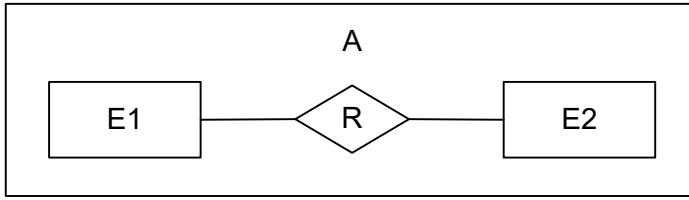


Figura 2.4. Diagrama ER que muestra una agregación.

Fuente: Elmasri & Navathe (2012).

Al aplicar las reglas de transformación descritas al diagrama ER de la figura 2.4 se obtienen los siguientes esquemas relacionales y sentencias SQL:

E1 (atributos_identificadores_E1, otros_atributos)
E2 (atributos_identificadores_E2, otros_atributos)
A (atributos_identificadores_E1, atributos_identificadores_E2, atributos_R, atributos_A)

CREATE TABLE E1(

atributos_identificadores_E1,

otros_atributos,

```
        PRIMARY KEY (atributos_identificadores_E1));
CREATE TABLE E2(
        atributos_identificadores_E2,
        otros_atributos,
        PRIMARY KEY (atributos_identificadores_E2));
CREATE TABLE A(
        atributos_identificadores_E1,
       atributos_identificadores_E2,
        atributos_R,
        atributos_A,
        PRIMARY KEY (llave_elegida_según_reglas)
        FOREIGN KEY (atributos_identificadores_E1)
              REFERENCES E1 (atributos_identificadores_E1)
              ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (atributos_identificadores_E2)
              REFERENCES E2 (atributos_identificadores_E2)
              ON DELETE CASCADE ON UPDATE CASCADE);
```

Nótese que para la elección de la llave primaria de la tabla *A* se aplican las mismas reglas para transformar una interrelación de asociación binaria como las definidas en (Elmasri & Navathe, 2012; Teorey, et al., 2014). De acuerdo con lo anterior, la cláusula PRIMARY KEY definirá como llave primaria al atributo identificador del tipo de entidad E_1 o al atributo identificador de E_2 si la razón de cardinalidad de *R* es 1:1 o 1:N. En otro caso la llave primaria de la tabla A estará formada por los atributos identificadores de los tipos de entidades E_1 *y* E_2.

Obsérvese que la existencia de una entidad de A depende de una y siempre la misma entidad de E_1 y de E_2, así como la eliminación de una entidad de E_1 o de una entidad de E_2 conllevará a eliminación de las tuplas en que estaban involucradas. Este comportamiento se garantiza a través de las acciones referenciales de eliminación y actualización en cascada impuestas a cada una de las llaves extranjeras en la relación *A*.

La transformación de una composición que modela una interrelación entre un "todo" y sus "partes" según Umanath & Scamell (2011), es la siguiente: se crea un esquema relacional para el tipo de entidad que representa el agregado; para cada tipo de entidad que representa a las partes se crea un esquema relacional, donde se incluye un atributo como llave extranjera que hace referencia a la llave primaria del esquema de relación correspondiente al agregado. Se definen las acciones referenciales asociadas a la dependencia de existencia entre las "partes" y el "todo".

Sea *T* el tipo de entidad que representa al agregado (todo) y P_1, P_2,...,P_n los tipos de entidades que representan las partes (véase la Figura 2.5).

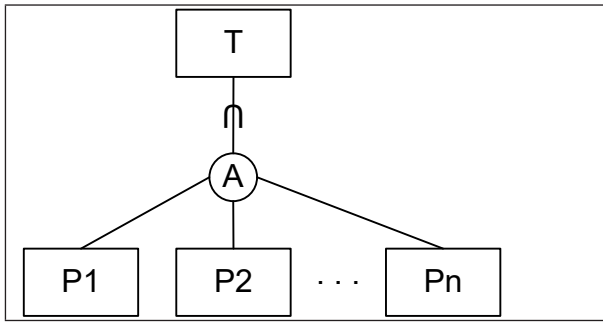


Figura 2.5. Representación de una interrelación todo/partes.

Fuente: Elmasri & Navathe (2012).

Al aplicar las reglas de transformación descritas al diagrama ER

de la Figura 2.5 se obtienen los siguientes esquemas relacionales y sentencias SQL:

T(atributos_identificadores_T, otros_atributos)

P1(atributos_identificadores_P1, otros_atributos, *atributos_identificadores*_T)

P2(atributos_identificadores_P2, otros_atributos, *atributos_identificadores*_T)

. . .

Pn(atributos_identificadores_Pn, otros_atributos, *atributos_identificadores*_T)

CREATE TABLE T(

atributos_identificadores_T,

otros_atributos,

PRIMARY KEY (atributos_identificadores_T));

Se debe crear una tabla Pi ($i = \overline{1,n}$) para cada uno de los esquemas relacionales obtenidos a partir de los tipos de entidades que representan a las "partes".

CREATE TABLE Pi(

atributos_identificadores_Pi,

otros_atributos,

atributos_identificadores_T,

PRIMARY KEY (atributos_identificadores_Pi)

FOREIGN KEY (atributos_identificadores_T)

REFERENCES T (atributos_identificadores_T)

ON DELETE CASCADE ON UPDATE CASCADE);

Como se había señalado anteriormente, en este tipo de construcción las "partes" tienen una fuerte dependencia de existencia con respecto al "todo". Este comportamiento se expresa a través de las acciones referenciales de eliminación y actualización en cascada especificadas en cada una de las tablas que corresponden a las "partes".

2.1.5. Asociaciones

La formalización del concepto enunciado en la definición 2 ha dado lugar al análisis de la dependencia de existencia en interrelaciones de asociación en el modelo ER.

Varios autores están de acuerdo en que la restricción de participación total en las asociaciones implica un cierto tipo de dependencia de existencia (Elmasri & Navathe, 2012; Silberschatz, et al., 2014); no obstante, la particularidad de la dependencia de existencia en asociaciones no se ha estudiado lo suficiente en el modelo ER y se expresa con una extensión del concepto, acorde a la definición 2 y conduce a la imposición de restricciones de actualización a la restricción de participación total que exprese dependencia de existencia.

Para ilustrarlo, considérese el diagrama EER de la figura 2.6, donde se representa una empresa con clientes a los que se les brinda distintos servicios, entre ellos la realización de proyectos de software, en que cada proyecto es solicitado exactamente por un cliente y solo uno. Nótese que esto es un hecho muy sensible, incluso por la existencia de un posible contrato económico. Dicha solicitud es registrada hasta que se le asigne un equipo de trabajo. Supóngase que los empleados que se dedican a la realización de estos proyectos en un momento determinado solo deben estar trabajando en uno a la vez.

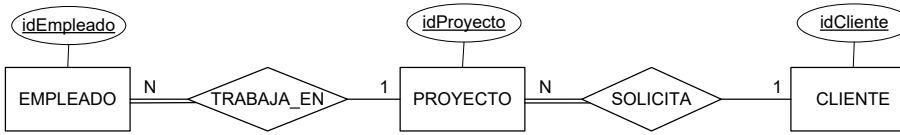


Figura 2.6. Control de proyectos de una empresa.

Fuente: Elmasri & Navathe (2012).

La representación de ambas asociaciones es idéntica en cuanto a estructura y aparentemente tienen el mismo comportamiento, sin embargo existe una diferencia sustancial en la semántica de las mismas. De acuerdo con las restricciones del problema ya expuestas, cada empleado, en un momento dado, debe estar trabajando en solo un proyecto, pero en el tiempo puede trabajar consecutivamente en varios proyectos, lo cual trae como consecuencia que no tengan una dependencia de existencia del proyecto. En el diagrama se refleja que la asociación TRABAJA_EN entre EMPLEADO y PROYECTO es obligatoria para EMPLEADO ya que un empleado siempre debe estar trabajando en un proyecto; pero su participación en un proyecto dado puede ser cancelada y asignarle otro proyecto a dicho empleado, incluso volver a trabajar en un proyecto asignado anteriormente, sin que se violen las restricciones del problema. Aquí se observa que la asociación TRABAJA_EN es modificable. Por otra parte, en la asociación SOLICITA, un proyecto que es pedido por un cliente seguirá perteneciendo al mismo cliente durante todo el tiempo de vida, lo que pone de manifiesto la inmutabilidad del vínculo entre un proyecto y un cliente. Por tanto, se puede concluir que la asociación SOLICITA no es modificable y expresa dependencia de existencia, según la extensión del concepto expresada en definición 2 usada como referencia.

Consecuentemente, el diagrama de la figura 2.6 puede considerarse semánticamente incompleto, ya que hay un comportamiento de los datos que no queda reflejado desde el punto de vista conceptual. Al transformar la asociación SOLICITA a esquemas relacionales utilizando reglas de transformación, como las propuestas en (Teorey, et al., 1986; Elmasri & Navathe, 2012), se obtienen los siguientes esquemas de relaciones:

Cliente (idCliente, otros_atributos_cliente)

Proyecto (idProyecto, otros_atributos_proyecto, *idCliente*)

Donde idCliente en el esquema Proyecto debería ser un atributo no actualizable, puesto que PROYECTO tiene una asociación con dependencia de existencia de CLIENTE. En la implementación, el hacer que un atributo sea no actualizable no resulta sencillo sin que sea muy costoso, como sería por ejemplo con la imposición de alguna restricción.

Cuando se está en presencia de asociaciones con dependencia de existencia, se debe tener en cuenta que existe un comportamiento en el universo del discurso que implica una modelación que va más allá de la simple consideración de una llave extranjera que no admite el valor nulo. Para que esta situación quede reflejada correctamente, en esta investigación se propone modelarlo como un hecho independiente. En lugar de generar los esquemas relacionales anteriores, se propone crear un nuevo esquema al cual se le agrega una llave artificial o subrogado para sutilmente manifestar, mediante la semántica de la misma, que modificaciones en los valores de los atributos que se refieren a CLIENTE o PROYECTO originan un nuevo hecho, y su semántica debe ser expresada con el nombre que se le asigne a dicho esquema.

Cliente (idCliente, otros_atributos_cliente)

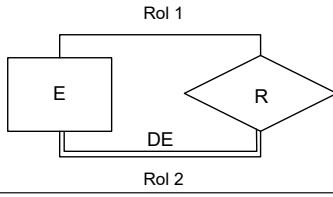
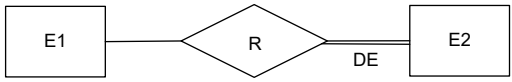
Proyecto (idProyecto, otros_atributos_proyecto)

Solicita (llave_subrogada, *idProyecto, idCliente*)

Nótese que la llave subrogada se puede editar para mostrar su semántica, por lo que se pudiera utilizar nroContrato para nombrar este atributo. Al introducir en el esquema SOLICITA la llave subrogada nroContrato se está mostrando una solución desde el punto de vista práctico para insinuar que la asociación entre CLIENTE y PROYECTO no sea modificable y de esta manera cada nuevo valor de la llave subrogada representa otro hecho, en este caso un nuevo contrato entre un cliente y un proyecto.

Para modelar las interrelaciones de asociación con dependencia de existencia se propone una notación para los diagramas EER con su correspondiente regla de transformación hacia el modelo relacional (García, Rodríguez, Cabrera, & González, 2008a). En la notación propuesta, el tipo de entidad dependiente se identifica con las siglas "DE" sobre la línea que conecta al rectángulo correspondiente al tipo de entidad con el rombo relativo a la asociación *R* (véase la tabla 2.1).

Tabla 2.1. Notación para modelar asociaciones recursivas y binarias con dependencia de existencia.

Símbolo	Significado
	Interrelación de asociación recursiva con dependencia de existencia.
	Interrelación de asociación binaria con dependencia de existencia.

En relación con la notación mostrada en la tabla 2.1, resulta oportuno señalar las siguientes características de la interrelación de asociación con dependencia de existencia abordadas en el presente trabajo:

- El grado de la asociación es 1 o 2 con razón de cardinalidad 1:1 o 1:N.
- El tipo de entidad E2 es el tipo de entidad dependiente, de acuerdo a la definición 2 siempre tiene participación obligatoria. En el caso de una interrelación de asociación recursiva, la dependencia de existencia se expresa a través de uno de sus roles.
- La asociación puede tener atributos propios, donde los atributos temporales desempeñan un papel importante, pues

permiten reflejar la duración de la dependencia, surgieren como parte de la interrelación la presencia de atributos para el inicio y fin de la relación; el primero no actualizable y el segundo inicialmente nulo y actualizable.

Se proponen como reglas de transformación para asociaciones con dependencia de existencia (RTDE) las siguientes:

- Para *asociaciones recursivas*: sea *R* una asociación recursiva con dependencia de existencia entre las entidades del tipo de entidad *E*. Para transformar la asociación *R* se crea un nuevo esquema relacional *S*. Se incluye como llave primaria en *S* un atributo subrogado, y como llaves extranjeras se incluyen las llaves primarias por cada uno de los roles de la asociación. También se incluyen los atributos de la asociación *R*. Para cada una de las llaves extranjeras en *S* se definen las acciones referenciales de eliminación y actualización en cascada.

- Para *asociaciones binarias*: sea *R* una asociación binaria con dependencia de existencia entre los tipos de entidades E_1 y E_2. Para transformar la asociación *R* se crea un nuevo esquema relacional *S*. Se incluye como llave primaria en S un atributo subrogado, y como llaves extranjeras se incluyen las llaves primarias de los tipos de entidades E_1 y E_2. También se incluyen los atributos de la asociación *R*. Para cada una de las llaves extranjeras en S se definen las acciones referenciales de eliminación y actualización en cascada. El comportamiento relativo a la dependencia de existencia de E_2 de E_1 se materializa mediante disparadores asociados al esquema *S*.

Se utiliza la notación de la tabla 2.1, el diagrama de la figura 2.7 se expresa con más exactitud la semántica de la interrelación SOLICITA.

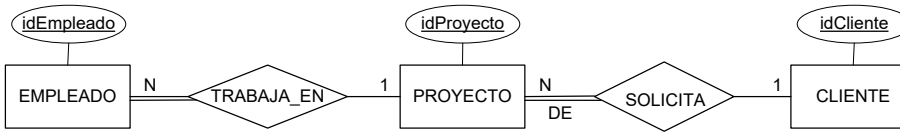


Figura 2.7. Mejora de la semántica del diagrama ER con el empleo de una asociación con dependencia de existencia.

Al transformar hacia los esquemas relacionales el diagrama EER de la figura 2.7 aplicando la regla RTDE para la asociación SOLICITA, se obtienen las siguientes sentencias SQL:

```
CREATE TABLE Cliente(
        idCliente dominio,
          otros_atributos_cliente,
          PRIMARY KEY (idCliente));
CREATE TABLE Proyecto(
          idProyecto dominio,
          otros_atributos_proyecto,
          PRIMARY KEY (idProyecto));
CREATE TABLE Solicita(
          nroContrato INTEGER,
          idCliente dominio,
          idProyecto dominio,
          PRIMARY KEY (nroContrato),
          FOREIGN KEY (idCliente)
                REFERENCES Cliente(idCliente)
                ON DELETE CASCADE ON UPDATE RESTRICT,
         FOREIGN KEY (idProyecto)
                REFERENCES Proyecto(idProyecto)
         ON DELETE CASCADE ON UPDATE RESTRICT));
```

El comportamiento de las entidades del tipo de entidad PRO-

YECTO que dependen en existencia de las entidades del tipo de entidad CLIENTE se garantiza con la definición del siguiente disparador en la tabla SOLICITA.

```
CREATE TRIGGER Propaga_Eliminación_DE
        AFTER DELETE ON Solicita
        FOR EACH ROW
         DELETE FROM Proyecto
                WHERE idProyecto = OLD.idCliente);
```

Como se ha podido apreciar, la modelación de atributos no modificables ha sido resuelta mediante el empleo de llaves subrogadas, ya que se ha podido constatar que estas permiten sugerir hechos con una existencia propia inmutable. Al tener una existencia propia se pueden identificar y los recursos que utiliza la tecnología de bases de datos para identificar tuplas son precisamente las llaves. En otras palabras, la solución que se ofrece es semántica, al sugerir que esa llave representa esa combinación, entonces la inmutabilidad del atributo que establece la dependencia de existencia se traspasa a la llave.

Las diferentes manifestaciones del concepto de dependencia de existencia del modelo ER se adhieren a la definición 1, que indica un comportamiento en cascada. En el caso de la categorización, aunque la construcción y su transformación a esquemas relacionales aparece tratado en la literatura, no se completa el comportamiento implícito de la dependencia de existencia de la categoría.

La dependencia de existencia en asociaciones, no ha sido abordada en extensiones al modelo ER y el comportamiento asociado para esta construcción conduce a la existencia de atributos no modificables; comportamiento no considerado en la definición 1 y que se propone como una extensión al concepto de dependencia de existencia mediante la definición 2, descrita en la metodología MERODE en un contexto para el desarrollo de

aplicaciones basado en modelos de dominio.

Se propuso una nueva construcción para modelar interrelaciones de asociación con dependencia de existencia y su correspondiente regla de transformación a esquemas relacionales.

Se sistematiza el comportamiento de la dependencia de existencia en las siguientes construcciones: interrelaciones débiles, jerarquías de generalización, categorización y agregación, proporcionándose las sentencias SQL correspondientes con el objetivo de incluirlas en una herramienta de ayuda al diseño de bases de datos.

Capítulo III. Validación de esquemas lógicos

3.1. Inconsistencia de referencias cíclicas

Para introducir el problema de este tipo de inconsistencia se seleccionaron ejemplos de esquemas conceptuales como casos de estudio. A cada esquema se le hicieron validaciones estructurales. Posterior a este proceso y aplicando un conjunto de reglas de transformación, se obtuvo el esquema lógico correspondiente, que fue utilizado en la caracterización del tipo de inconsistencia objeto de estudio en este acápite.

Los diagramas ER que se muestran en este capítulo utilizan la notación de Elmasri & Navathe (2012). De igual manera, son de los autores mencionados, las reglas de transformación que se aplican para obtener los esquemas relacionales. Para cada esquema conceptual se muestra el esquema lógico correspondiente compuesto por los esquemas de relaciones, en los cuales se adopta como convenio que las llaves primarias estarán subrayadas y las llaves extranjeras en cursiva.

Para apoyar el análisis se hace una representación gráfica del esquema relacional utilizando un grafo relacional (De Miguel & Piattini, 1993; Elmasri & Navathe, 2012) compuesto de un conjunto de nodos multiparticionados, donde cada nodo representa un esquema relacional. Para cada esquema de relación se muestra su nombre y sus atributos, se indica su llave primaria (se subrayan los atributos que la componen con trazo continuo). Se dibuja, además, un arco dirigido que conecta los atributos que constituyen la llave extranjera en un esquema de relación con los atributos que forman la llave primaria en el esquema relacional referenciado. En caso que la llave extranjera no admita valores nulos el arco estará etiquetado con la frase “No nulo”.

3.1.1. Caso de estudio 1

El diagrama ER de la figura 3.1 modela el hecho de que un empleado tiene que ser dirigido por otro empleado y que un empleado puede dirigir a ninguno, uno o varios empleados.

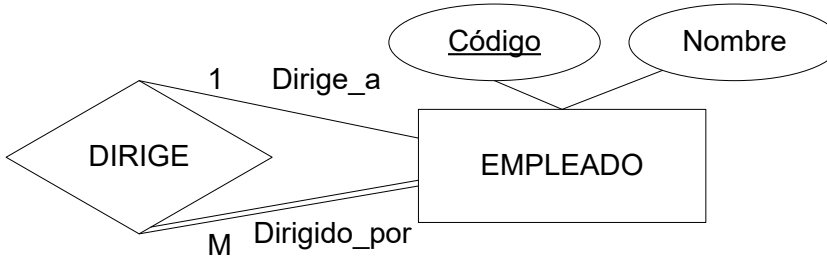


Figura 3.1. Diagrama ER del caso de estudio 1.

Al transformar el esquema conceptual se obtiene el siguiente esquema relacional:

Empleado(Código, Nombre, Director)

Como puede observarse en el grafo relacional de la figura 3.2, la llave extranjera *Director* no admite el valor nulo debido a la participación obligatoria del tipo entidad EMPLEADO en la interrelación DIRIGE a través del rol *Dirigido_por*. esto trae como consecuencia que no es posible insertar una tupla correspondiente a un empleado sin antes haber insertado el empleado que lo dirige, así se está en presencia de un ciclo, por lo que el esquema de relación es inconsistente.

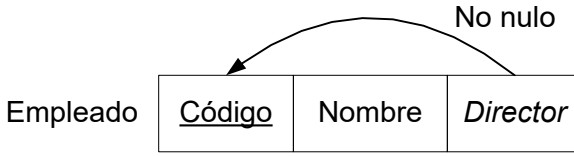


Figura 3.2. Grafo relacional del caso de estudio 1.

3.1.2. Caso de estudio 2

En el diagrama de la figura 3.3 se modelan los siguientes hechos: los empleados tienen que pertenecer a un departamento y un departamento tiene que ser dirigido por un empleado.

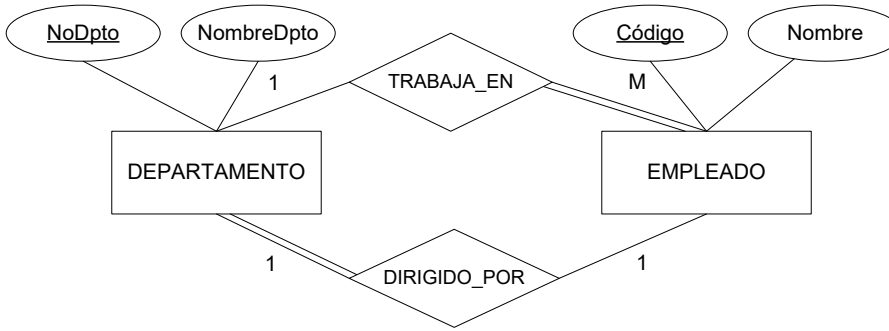


Figura 3.3. Diagrama ER del caso de estudio 2.

Al transformar el esquema conceptual se obtienen los siguientes esquemas relacionales:

Empleado(Código, Nombre, NoDpto)
Departamento(NoDpto, NombreDpto, *JefeDpto*)

En la figura 3.4 se muestra el grafo relacional que se corresponde con los esquemas de relaciones anteriores.

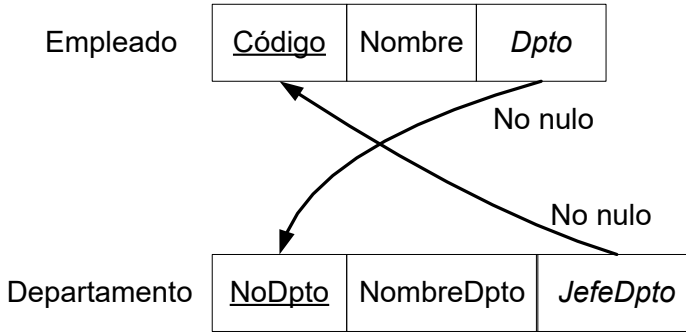


Figura 3.4. Grafo relacional del caso de estudio 2.

En este caso, debido a la participación obligatoria de los tipos de entidad EMPLEADO y DEPARTAMENTO en interrelaciones TRABAJA_EN y DIRIGIDO_POR respectivamente, las llaves extranjeras correspondientes no permiten valores nulos. Por esta razón no es posible insertar un departamento a menos que se

haya insertado el empleado jefe en el esquema relacional Empleado, al mismo tiempo no es posible insertar un empleado a menos que se haya insertado el departamento al que pertenece el mismo en el esquema relacional Departamento. Como puede observarse, hay una dependencia mutua que genera un conflicto que no permite la inserción de tuplas en ninguno de los dos esquemas. Este tipo de inconsistencia en los esquemas relacionales genera un problema de implementación dado por la presencia de llaves extranjeras con referencias mutuas y que no admiten valores nulos, el cual puede ser ilustrado mediante el siguiente código en SQL.

```
CREATE TABLE Empleado(
        Código      INTEGER NOT NULL,
        Nombre      CHARACTER(32) ,
        Dpto        INTEGER NOT NULL,
        PRIMARY KEY ( Código ));
CREATE TABLE Departamento(
        NoDpto      INTEGER NOT NULL,
        NombreDpto CHARACTER(32) ,
        JefeDpto    INTEGER NOT NULL,
        PRIMARY KEY ( NoDpto ));
ALTER TABLE Empleado ADD CONSTRAINT FK_Empleado_
Departamento
        FOREIGN KEY (Dpto)
                REFERENCES Departamento (NoDpto );
ALTER TABLE Departamento ADD CONSTRAINT FK_Depar-
```

tamento_Empleado

FOREIGN KEY (JefeDpto)

REFERENCES Empleado (Código)

3.1.3. Caso de estudio 3

En el diagrama de la figura 3.5 se modelan los siguientes hechos: los empleados tienen que pertenecer a un departamento; un departamento tiene que financiar un proyecto y un proyecto tiene que ser apoyado por un empleado.

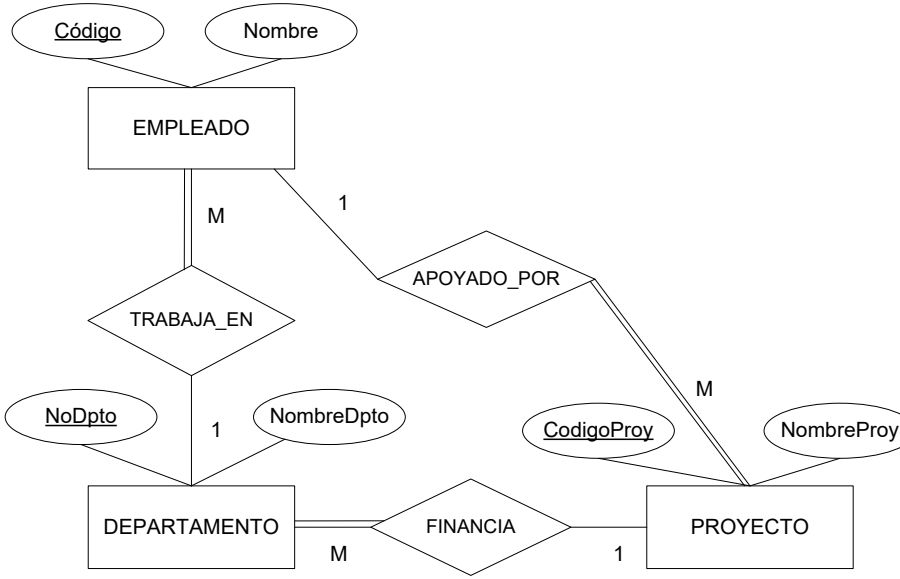


Figura 3.5. Diagrama ER del caso de estudio 3.

Transformando el esquema conceptual se obtienen los siguientes esquemas relacionales:

Empleado(Código, Nombre, Dpto)
Departamento(NoDpto, NombreDpto, Proy)
Proyecto(CódigoProy, NombreProy, Emp)

Nótese en la figura 3.6 que las llaves extranjeras de los esquemas obtenidos no admiten valores nulos debido a la participación obligatoria de los conjuntos de entidades del lado "mucho" de las interrelaciones. Como puede observarse existe una referencia cíclica entre los esquemas, y no es posible insertar una tupla en ninguno de esquemas, por lo que también se presenta un problema de implementación debido a esquemas relacionales inconsistentes.

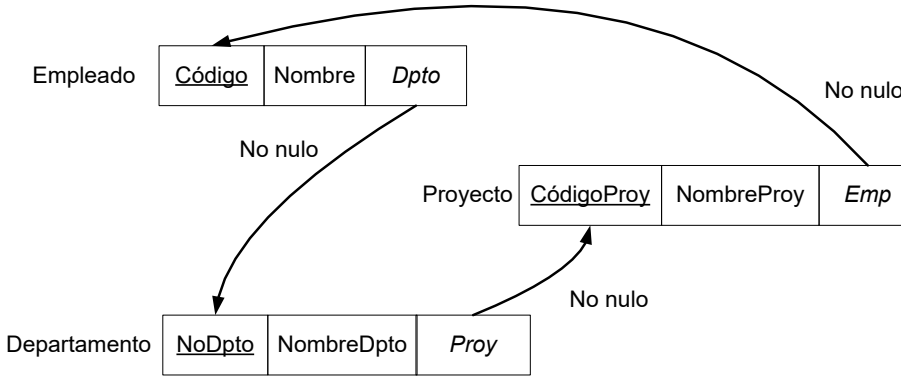


Figura 3.6. Grafo relacional del caso de estudio 3.

Los ejemplos mostrados anteriormente permiten concluir que a pesar de que un esquema conceptual ER sea estructuralmente válido, es posible obtener esquemas relacionales que presenten inconsistencias de referencias cíclicas.

3.2. Caracterización de la inconsistencia de referencias cíclicas

Como resultado del análisis de los casos de estudio anteriores, la inconsistencia de referencias cíclicas se define de la siguiente

manera:

Definición: Sean R_1, R_2,…, R_n esquemas relacionales de un esquema lógico *L* de una base de datos, *L* manifiesta una inconsistencia de referencias cíclicas si existe una llave extranjera en R_i, que no admite el valor nulo y que hace referencia a la llave primaria de R_{i+1}, para i=$\overline{1..n}$; y si también existe una llave extranjera en R_n, que no admite el valor nulo, y que hace referencia a la llave primaria en R_1, donde R_n y R_1 no son necesariamente diferentes.

Nótese en la definición que la inconsistencia puede manifestarse dentro de un mismo esquema, o entre dos o más esquemas relacionales. Por lo tanto, la solución que se propone en este trabajo consiste en encontrar todas las secuencias de esquemas relacionales que cumplan con las condiciones expresadas en la definición anterior y a continuación realizar un proceso de corrección de las inconsistencias de referencias cíclicas, para obtener como resultado un esquema lógico consistente.

Para ilustrar el proceso de corrección de inconsistencias, considérese de nuevo el esquema lógico obtenido a partir del diagrama ER de la Figura 3.3.

Empleado(Código, Nombre, *NoDpto*)
Departamento(NoDpto, NombreDpto, *JefeDpto*)

La inconsistencia de este esquema se puede comprobar a través del análisis de los estados S_1 *y* S_2, compuestos por tuplas de los esquemas relacionales Empleado y Departamento.

S_1 = { r $_{Empleado}$ = [10, “nombre1”, 101], r $_{Departamento}$ = [101, “dpto1”, 10] };

S_2 = { r $_{Empleado}$ = [11, “nombre2”, null], r $_{Departamento}$ = [200, “dpto2”, 11] }.

El estado S_1 no es posible satisfacerlo debido a las referencias mutuas de las llaves extranjeras de los esquemas Empleado y Departamento. El estado S_2 tampoco puede ser satisfecho debido a que la llave extranjera del esquema Empleado no admi-

te valores nulos. Para eliminar esta inconsistencia se propone crear un nuevo esquema relacional en lugar de una de las restricciones de integridad referencial.

Por ejemplo, se puede seleccionar la llave extranjera que representa a la interrelación TRABAJA_EN y en su lugar se creará un nuevo esquema, de manera que el esquema lógico resultante es:

Empleado(Código, Nombre)
Departamento(NoDpto, NombreDpto, *JefeDpto*)
Trabaja_en(Código, *NoDpto*)

Considere ahora el siguiente estado:

$S_1 = \{ r_{Empleado} = [10, \text{“nombre1”}], r_{Departamento} = [101, \text{“dpto1”}, 10], r_{Trabaja\_en} = [10, 101] \}$

Nótese que el estado S_1 es consistente y se logra insertando primero una tupla en el esquema Empleado, a continuación se inserta una tupla en el esquema Departamento y por último se inserta una tupla en el esquema Trabaja_en.

3.3. Detección y corrección de inconsistencias de referencias cíclicas

En esta sección se describe el algoritmo de detección y corrección de inconsistencias de referencias cíclicas en esquemas lógicos, el cual utiliza un seudógrafo dirigido para representar los esquemas relacionales generados a partir del esquema conceptual. El algoritmo obtiene una solución del esquema lógico sin inconsistencias de referencias cíclicas bajo el criterio de generar la menor cantidad de nuevos esquemas relacionales.

3.3.1. Representación del esquema lógico

La representación de un esquema lógico de una base de datos mediante un seudógrafo dirigido es directa si se asume que los vértices son los esquemas relacionales y cada arista dirigida indica la interrelación que existe entre la llave extranjera de un esquema relacional con su correspondiente llave primaria en otro esquema relacional. Tomando como referencia la definición de seudógrafo dada en (Grassmann & Tremblay, 2011), el esquema

lógico se representa como:

Sea $G_{EL} = (V, A, f, p_1, p_2)$ el seudógrafo dirigido y ponderado que representa el esquema lógico de una base de datos, donde:

- $V \neq \varnothing$ es el conjunto finito de vértices que representan los esquemas relacionales generados a partir del esquema conceptual. Cada vértice contiene la información que permite describir los atributos del esquema relacional.
- A es el conjunto finito de aristas.
- Una función $f : A \rightarrow VxV \cup \{(u,v); u,v \in V\}$, que asigna a cada arista un par ordenado de vértices (no necesariamente diferentes), donde "aÎA si f(a) = (u,v) se dice que la arista a representa la relación existente entre una llave extranjera del esquema relacional u con la llave primaria del esquema relacional v.
- $p_1 : A \rightarrow$ { "NoNulo", "Nulo"} es una función donde $\forall a \in A$ si $p_1(a)$="NoNulo" significa que la llave extranjera no admite el valor nulo y $\forall a \in A$ si $p_1(a)$="Nulo" significa que la llave extranjera admite el valor nulo.
- $p_2 : A \rightarrow \mathbb{Z}^*_+$ es una función que $\forall a \in A$ indica la cantidad de veces que la arista a forma parte de un ciclo elemental.

3.3.2. DETCOI: Algoritmo de detección y corrección de inconsistencias de referencias cíclicas

A partir de la caracterización de la inconsistencia de referencias cíclicas se propone un algoritmo (García, Rodríguez, Cabrera & González, 2008b, 2008c, 2010) que detecta y corrige este tipo de inconsistencia en el esquema lógico de una base de datos. La entrada de este algoritmo es el esquema conceptual ER y la salida es un seudógrafo dirigido que representa el esquema lógico el cual no contiene inconsistencias de referencias cíclicas.

Algoritmo DETCOI: Detección y Corrección de Inconsistencias de Referencias Cíclicas.

1. **Crear el seudógrafo que representa el esquema lógico de la base de datos.**
2. **Detectar inconsistencias de referencias cíclicas en el seudógrafo.**
3. **Corregir inconsistencias de referencias cíclicas en el seudógrafo.**

A continuación se describe cada paso del algoritmo propuesto.

PASO 1. Crear el seudógrafo G_{EL} que representa el esquema lógico de la base de datos.

Cada vértice vi $\in$ V es creado a partir de aplicar un conjunto de reglas de transformación al esquema conceptual y contendrá como información básica los atributos del esquema relacional. Los atributos se identifican por su tipo: descriptivo, llave primaria y/o llave extranjera. Las aristas del seudógrafo se crean de la siguiente manera: "a Î A, $p_2(a) = 0$ y $p_1(a) =$ {''Nulo''} o $p_1(a) =$ {''NoNulo''} indicando si la llave extranjera admite o no el valor nulo.

PASO 2. Detectar inconsistencias de referencias cíclicas en el seudógrafo G_{EL}.

2.1. Buscar en el seudógrafo G_{EL} todos los ciclos elementales utilizando una variante iterativa del algoritmo DFS (Depth First Search).

2.2. Para cada ciclo elemental Cj = (a_{j_1}, a_{j_2}, … , a_{j_r}), $\forall j \in \overline{1,k}$ (k es la cantidad de ciclos elementales detectados en el Paso 2.1), a_{j_i} Î A, $\forall i \in \overline{1,r}$ (r ≤ m, m es la cantidad de aristas) hacer:

2.2.1. T = $\varnothing$. (T es el conjunto de ciclos inconsistentes).

2.2.2. Si $\forall i \in \overline{1,r}$, p1($a_{j_i}$) = "NoNulo" entonces marcar el ciclo elemental Cj como inconsistente, de forma tal que T = T U { Cj }.

PASO 3. Corregir inconsistencias de referencias cíclicas en el seudógrafo G_{EL}.

3.1. Para cada arista a∈A, tal que f(a) = (u,u) hacer:

3.1.1. Eliminar del vértice u los atributos que forman la llave extranjera y que hacen referencia al propio vértice a través de la arista a.

3.1.2. Crear el vértice w e insertarlo en el seudógrafo G_{EL}, de forma tal que V = V U {w}.

3.1.3. Crear dos nuevas aristas a' y a'', de forma tal que A = A U {a', a''}, f(a') = (w,u) y f(a'') = (w,u).

3.1.4. Actualizar la información del vértice w, que consiste en insertar un atributo llave extranjera por cada arista a' y a'', que hace referencia a la llave primaria del vértice u. Definir una de las dos llaves extranjeras como llave primaria del vértice w.

3.1.5. Hacer p1(a') = p1(a'') = "NoNulo" y p2(a') = p2(a'') = 0.

3.1.6. Eliminar la arista a del seudógrafo G_{EL}, de forma tal que A = A \ {a}.

3.2. Para cada ciclo elemental Cj = ($a_{j_1}, a_{j_2}, \ldots, a_{j_r}$) en T, $\forall j \in \overline{1,k}$, para r ≥ 2 hacer:

3.2.1. Buscar a_{j_q} tal que $p_2(a_{j_q}) = \max_{i=1..r}\{p_2(a_{j_i})\}$ (en lo adelante la arista a_{j_q} se denotará simplemente como arista a).

3.2.2. Eliminar del vértice u el atributo llave extranjera que hace referencia al vértice v a través de la arista a.

3.2.3. Crear un vértice w e insertarlo en el seudógrafo G_{EL} de forma tal que V = V U {w}.

3.2.4. Crear dos nuevas aristas a' y a'', de forma tal que A = A U {a',a''}, f(a') = (w,u) y f(a'') = (w,v).

3.2.5. Actualizar la información del vértice w insertando dos atributos llaves extranjeras; una llave extranjera hará referencia a través de la arista a' a la llave primaria del vértice u, la otra llave extranjera hará referencia a través de la arista a'' a la llave primaria del vértice v; definir como atributo llave primaria del vértice w a la llave extranjera que hace referencia al vértice u.

3.2.6. Hacer $p_1(a') = p_1(a'') =$ ''NoNulo'' y $p_2(a') = p_2(a'') = 0$.

3.2.7. Para todas las aristas $a_{j_i} \neq a$ con $i = \overline{1,r}$ hacer

$p_2(a_{j_i}) = p_2(a_{j_i}) - 1$.

3.2.8. $T = T \setminus \{C_j\}$.

3.2.9. $M = \varnothing$. Para cada ciclo elemental $C_j = (a_{j_1}, a_{j_2}, \ldots, a_{j_r})$ en T, $\forall j \in \overline{1,k}$, si la arista a está presente en el ciclo C_j entonces $M = M \cup \{C_j\}$.

3.2.10. Eliminar la arista a del seudógrafo G_{EL} de forma tal que $A = A \setminus \{a\}$.

3.2.11. $T = T \setminus M$. Terminar si $T = \varnothing$, de lo contrario ir al Paso 3.2.1.

En el paso 3.2 se realiza el proceso de corrección de ciclos elementales de longitud mayor o igual que dos. Se justifica a continuación que este proceso no genera nuevos ciclos.

Sea una secuencia ordenada de vértices distintos $v_1, v_2,\ldots, v_k$, para $1 \leq k \leq n$, donde n es la cantidad de vértices del seudógrafo dirigido. Sea el ciclo elemental formado por las aristas ($a_1, a_2,\ldots, a_k$) tal que:

$f(a_1) = (v_1,v_2)$, $f(a_2) = (v_2,v_3)$,…, $f(a_{k-1}) = (v_{k-1},v_k)$, $f(a_k) = (v_k,v_{k+1})$, donde $v_1 = v_{k+1}$

En un ciclo elemental sin lazos se cumple que el semigrado inicial $d+(v_i) > 0$ y que el semigrado final $d-(v_i) > 0$, para $i = \overline{1,k}$ (Agnarsson & Greenlaw, 2014).

Supóngase que en el paso 3.2.1 se selecciona la arista a_i tal que $f(a_i) = (v_i,v_{i+1})$, donde $1 \leq i \leq k$.

En esencia, el proceso de corrección desde el paso 3.2.3 hasta 3.2.11 inserta un nuevo vértice w entre los vértices v_i y v_{i+1}, con aristas a' y a'' tal que f(a') = (w, v_i) y f(a'') = (w, v_{i+1}); se actualizan los pesos p_1 y p_2 de las aristas a' y a''; por último se elimina la arista a_i, por lo que el camino resultante queda formado por las aristas:

($a_1, a_2,\ldots, a_{i-1}, w, a_{i+1},\ldots, a_k$) tal que:

$f(a_1) = (v_1,v_2),\ldots, f(a_{i-1}) = (v_{i-1},v_i)$, $f(a') = (w,v_i)$, $f(a'') = (w,v_{i+1})$, $f(a_{i+1}) = (v_{i+1},v_{i+2}),\ldots, f(a_k) = (v_k,v_{k+1})$

Al calcular el semigrado inicial y final de los vértices v_i y w se obtiene que:

$d+(v_i) = 0$, $d-(v_i) = 2$

$d+(w) = 2$, $d-(w) = 0$

Nótese que $d+(v_i) = 0$ y que $d-(w) = 0$, por lo que estos vértices no pueden formar parte de un ciclo elemental. Obsérvese además que el vértice w es un vértice no alcanzable debido a que $d-(w) = 0$. Esta condición establece dos caminos elementales:

$(a_1, a_2, \ldots, a_{i-1})$ y $(a'', a_{i+1}, \ldots, a_k)$

En consecuencia el ciclo elemental original ha sido transformado en dos caminos elementales, con lo queda justificado que el proceso de corrección no genera nuevos ciclos en el seudógrafo. Como resultado de la aplicación del algoritmo descrito se obtiene un nuevo esquema lógico que no presenta inconsistencias de referencias cíclicas y que puede ser traducido al esquema físico de la base de datos, sin que este último presente los problemas de implementación mencionados en el epígrafe 3.1.

En el cálculo de la complejidad temporal del algoritmo DETCOI se considera: *n* cantidad de vértices, m cantidad de aristas, y *k* cantidad de ciclos inconsistentes detectados.

La creación del seudógrafo se realiza en el paso 1. Tomando en consideración que para la representación del seudógrafo se utilizan listas de adyacencias, la complejidad de este paso es O(n+m). En el paso 2.1 la complejidad del algoritmo de detección de ciclos en un seudógrafo dirigido es $O(n^3m^2)$. En el 2.2, la creación de la lista de ciclos inconsistentes tiene un complejidad de O(km). En el paso 3.1 se examinan todas las aristas que sean lazos o bucles en el seudógrafo; si se considera el peor caso en que todas la aristas son lazos o bucles, entonces la complejidad es O(m). La corrección de los ciclos inconsistentes de longitud mayor o igual que dos que se realiza en el paso 3.2 tiene una complejidad de $O(k^2m)$. Luego la complejidad del paso 3 es $O(k^2m)$. Teniendo en cuenta el análisis de cada uno de los pasos anteriores, la complejidad temporal del algoritmo DETCOI queda expresada como:

$$O(O(n+m) + O(max(n^3m^2, km)) + O(k^2m)) = \quad (1)$$

$$O(O(n+m) + O(max(n^3m^2, km, k^2m)) = \quad (2)$$

$$O(O(n+m) + O(max(n^3m^2, k^2m)) = \quad (3)$$

$$O(max(n+m, n^3m^2, k^2m)) \quad (4)$$

Al analizar la expresión (4) se puede afirmar que para un seu-

dógrafo conexo se cumple que $n^3m^2 > n+m$, $\forall n,m \in \mathbb{Z}_+$, $n > 1$ y $m > 0$. Nótese que la desigualdad no se cumple solamente en el caso de que el seudógrafo esté formado por un vértice y a lo sumo por una arista (bucle), lo cual significa que el esquema lógico está formado por un esquema relacional, y en ese caso la complejidad es constante. Por tanto, la complejidad computacional del algoritmo DETCOI es:

$$O(max(n^3m^2, k^2m)) \qquad (5)$$

3.3.3. Un ejemplo de aplicación del algoritmo

A partir del diagrama ER que aparece en la figura 3.7, se muestra un esquema lógico con inconsistencias de referencias cíclicas y a continuación el esquema lógico que se obtiene cuando se aplica el algoritmo descrito anteriormente.

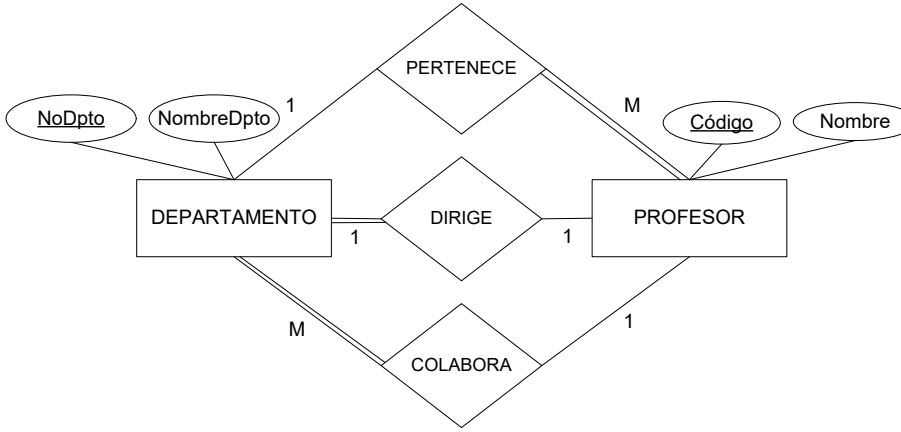


Figura 3.7. Diagrama ER del ejemplo.

Al transformar el diagrama ER de la figura 3.7 se obtienen los siguientes esquemas relacionales:

Profesor(Código, Nombre, *Dpto*)
Departamento(NoDpto, NombreDpto, *JefeDpto, Colaborador*)

En la figura 3.8 se muestra la representación de este esquema en el seudógrafo G_{EL}.

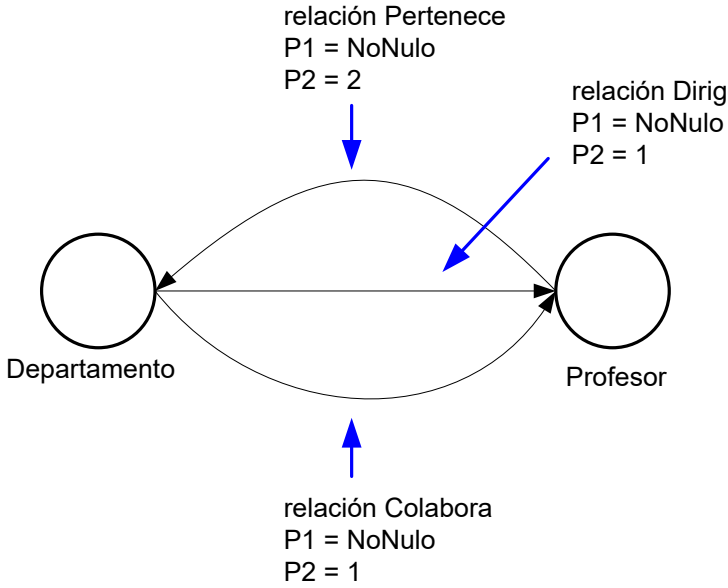


Figura 3.8. Esquema lógico con inconsistencias de referencias cíclicas.

El seudógrafo contiene dos ciclos elementales de longitud dos entre los vértices Departamento y Profesor que son inconsistentes: el ciclo Profesor → Pertenece → Departamento → Dirige → Profesor y el ciclo Profesor → Pertenece → Departamento → Colabora → Profesor. Nótese que la arista Pertenece participa en dos ciclos por lo que el valor del peso p_2 es 2, y como es el valor mayor en comparación con las otras aristas del grafo, entonces según el algoritmo DETCOI esta arista se elimina y en su lugar se crea un nuevo vértice llamado por ejemplo Pertenece, donde una arista se dirigirá hacia el vértice Profesor y la otra arista hacia el vértice Departamento. La figura 3.9 muestra una solución luego de la aplicación del algoritmo DETCOI.

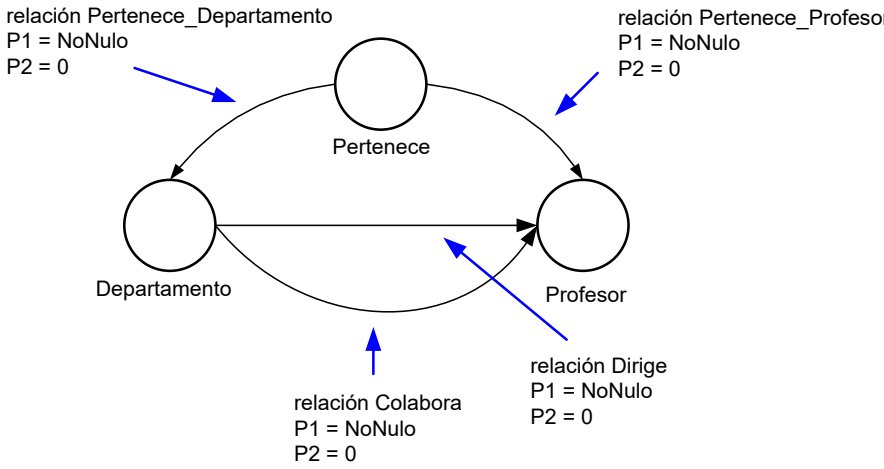


Figura 3.9. Seudógrafo G_{EL} resultante de la aplicación del algoritmo DETCOI.

Del seudógrafo de la Figura 3.9 se obtienen los siguientes esquemas relacionales consistentes:

Profesor(Código, Nombre, *Dpto*)
Departamento(NoDpto, Nombre*Dpto*, *JefeDpto*, *Colaborador*)
Pertenece(Código, *NoDpto*)

Aunque los métodos actuales de validación estructural de esquemas conceptuales permiten detectar restricciones de cardinalidad inconsistentes, se demuestra que es posible la presencia de inconsistencias a nivel del esquema lógico obtenido a partir de esquemas estructuralmente válidos. Se realizó la caracterización del tipo de inconsistencia de referencias cíclicas que puede presentarse en el esquema lógico de una base de datos.

Se diseñó un algoritmo para detectar y corregir en el esquema lógico inconsistencias de referencias cíclicas, el cual fue implementado en una herramienta CASE, nombrada ERECASE (Registro CENDA 2965-2008), que ha sido desarrollada en Grupo de Bases de Datos del Centro de Estudio de Informática perteneciente a la Facultad de Matemática-Física-Computación de la Universidad Central de Las Villas, Cuba.

Capítulo IV. ERECASE, una herramienta de ayuda a la modelación de bases de datos relacionales

4.1. Presentando la herramienta ERECASE

En este capítulo se presenta una herramienta de ayuda a la modelación de bases de datos relacionales, la cual ha sido el resultado de un proyecto de investigación que dirigió el autor de este libro y que tuvo como colofón la defensa de su proyecto doctoral (García, 2010).

La herramienta ERECASE integra los conocimientos abordados en los capítulos anteriores, en un esfuerzo por proporcionar a los diseñadores una ayuda en el proceso de diseño de diseño de bases de datos relacionales, aportando métodos de validación de los esquemas conceptual y lógico, así como nuevas construcciones al modelo Entidad Relación Extendido.

Es una herramienta para el diseño de bases de datos que utiliza como modelo conceptual el modelo Entidad Relación. Como característica novedosa permite la validación estructural del diagrama ER basándose en las cardinalidades máximas y mínimas de las interrelaciones de asociación.

Esta nueva herramienta CASE se crea para el diseño de bases de datos con el objetivo que permita la validación estructural de los diagramas Entidad Relación Extendido (ERE). Para cumplir con este objetivo se propusieron los siguientes objetivos específicos:

- Crear un espacio de trabajo para la edición del diagrama ERE.
- Implementar reglas para la validación estructural de construcciones del modelo ERE.
- Implementar las transformaciones del modelo ERE al modelo relacional.

- Implementar la traducción del modelo relacional a un modelo físico de bases de datos.

La herramienta se caracterizar por:

- La inclusión de un conjunto de construcciones de este modelo para lograr una mejor expresividad en el diagrama.
- La realización de validaciones estructurales en un esquema en la etapa de diseño para evitar inconsistencias.
- La transformación de un esquema al modelo relacional.
- La generación de código SQL para la creación de la base de datos física en un Sistema Gestor de Bases de Datos (SGBD) determinado.

4.2. Funcionalidades de ERECASE

Los resultados de índole teórico obtenidos en esta investigación fueron incorporados a la herramienta ERECASE mencionada anteriormente, de manera que la versión actual de misma exhibe las siguientes funcionalidades:

1. Se soporta el uso de la agregación en los diagramas EER.
2. Se incluye una nueva construcción: la interrelación de asociación con dependencia de existencia.
3. Se implementa un algoritmo que realiza la validación estructural del esquema conceptual EER.
4. Se implementa un nuevo algoritmo que realiza la detección y corrección de inconsistencias de referencias cíclicas.

La figura 4.1 muestra la arquitectura de ERECASE (García, Rodríguez, Cabrera & González, 2009), formada por los siguientes módulos: Editor del diagrama EER, Validación Estructural, Generación del Esquema Lógico, Detección y Corrección de Inconsistencias y Generación del Esquema Físico.

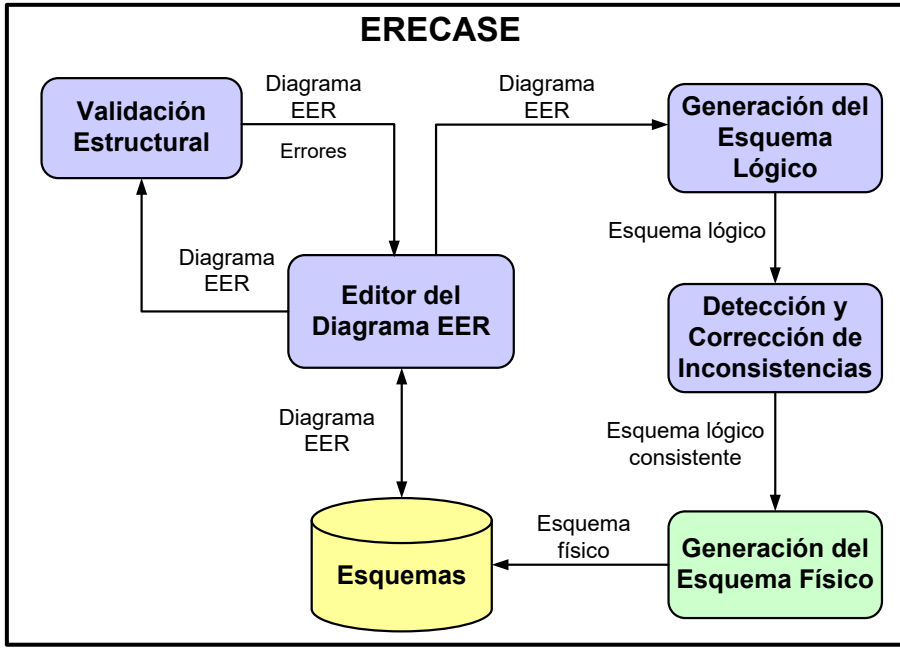


Figura 4.1. Vista arquitectónica de ERECASE con sus funcionalidades.

El Editor del Diagrama EER tiene como función la manipulación de los diversos objetos gráficos en la ventana de edición, los que se corresponden con cada una de las construcciones disponibles: entidades fuertes y débiles; interrelaciones de asociación recursivas, binarias y ternarias; jerarquías de generalización/especialización; interrelaciones ISA e interrelaciones débiles, agregación y la interrelación de asociación con dependencia de existencia; actualización del explorador de objetos y la notificación de los errores sintácticos y estructurales en la ventana de errores de la aplicación. Este módulo incorpora un chequeo sintáctico del diagrama EER que comprende:

- La exclusividad de nombres: los nombres de las construcciones deben ser únicos.
- El chequeo de identificación: todos los tipos de entidades, que no sean débiles, deben tener al menos un atributo identificador.

- El chequeo de identificación de tipos de entidades débiles: todos los tipos de entidades débiles deben estar identificados mediante al menos un tipo de interrelación débil.
- No permitir ciclos entre tipos de entidades débiles que participan en interrelaciones débiles.
- No permitir ciclos entre tipos de entidades que participan en interrelaciones de subconjunto.

La Validación Estructural detecta ciclos formados por los tipos de entidades e interrelaciones cuyas restricciones de cardinalidad son inconsistentes según el criterio definido en Dullea, et al. (2003). Esta funcionalidad es novedosa y distingue a ERECASE del resto de las herramientas evaluadas en el epígrafe 1.4.

La Generación del Esquema Lógico permite además la transformación de la construcción de agregación y también se añade la regla RTDE (Regla de Transformación para la Dependencia de Existencia) presentada como resultado en el capítulo 2, y que se aplica a interrelaciones de asociaciones recursivas o binarias con dependencia de existencia.

La Detección y Corrección de Inconsistencias, detecta y corrige de manera automática el tipo de inconsistencia de referencias cíclicas, que puede manifestarse en el esquema lógico. Este módulo implementa el algoritmo obtenido como resultado en el capítulo 3. Esta facilidad incorporada a la herramienta ERECASE no está presente en ninguna de las herramientas evaluadas en el epígrafe 1.4.

La Generación del Esquema Físico comprende además la generación de las sentencias SQL para la agregación y la interrelación de asociación con dependencia de existencia.

El proceso de diseño de una base de datos que utilice la interfaz de usuario de la herramienta ERECASE comienza con la creación de un nuevo diagrama o con la lectura de uno ya existente. Durante el proceso de edición se construye el diagrama EER, al que se le aplica un método de validación estructural que detecta restricciones de cardinalidad inconsistentes, las cuales

son notificadas al diseñador para su corrección manual. Una vez que el esquema es estructuralmente válido, entonces es posible obtener el esquema lógico, que es sometido al proceso de validación lógica, consistente en detectar inconsistencias de referencias cíclicas. Si se detecta esta inconsistencia, el diseñador decide si se corrige o no. Si se elige no realizar ninguna corrección, entonces se obtiene un esquema físico con problemas de implementación y su chequeo es responsabilidad de los programadores de aplicación; en caso contrario se efectúa el proceso de corrección de las inconsistencias, que posibilita obtener el esquema físico. Los procesos descritos se muestran en el diagrama de actividades de la figura 4.2.

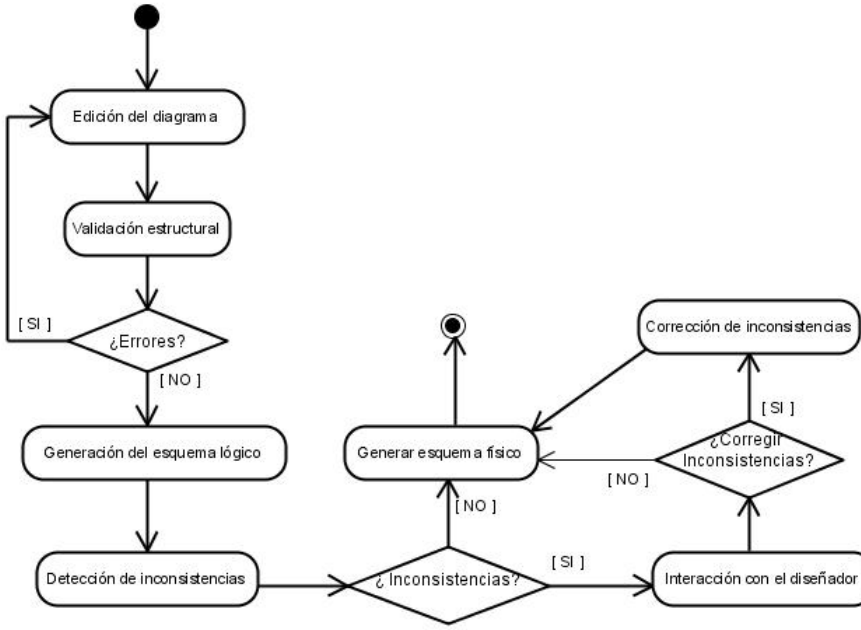


Figura 4.2. Diagrama de actividades que describe el proceso de diseño de una base de datos en la herramienta ERECASE.

Como puede observarse, las facilidades desarrolladas en la herramienta están centradas en el uso de construcciones del modelo ER y en la validación de esquemas.

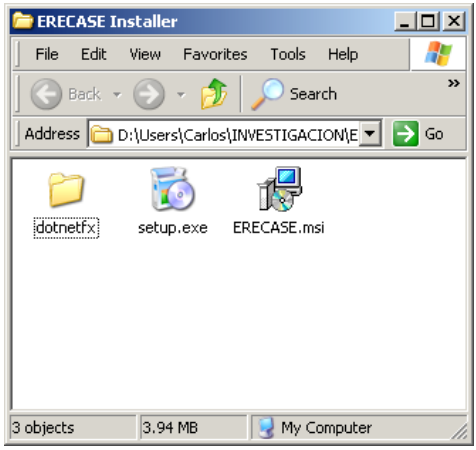
Una vez que se han expuesto los aspectos relacionados con las funcionalidades que tiene la herramienta ERECASE se describe el proceso de instalación y uso de dicha herramienta.

4.3. Instalación de ERECASE

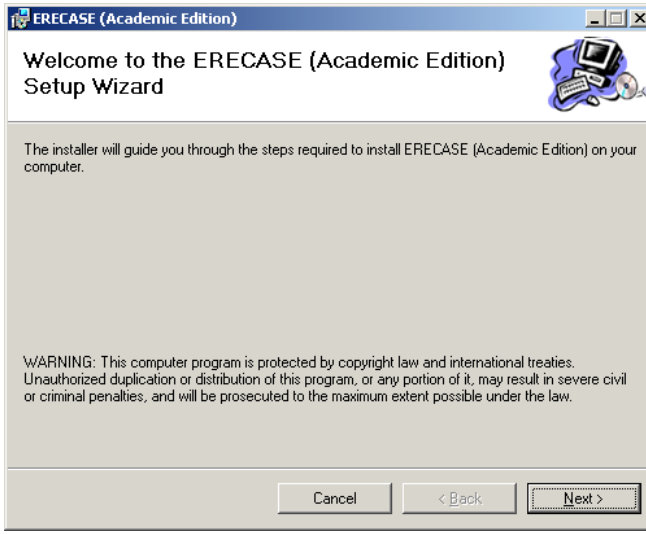
Los requisitos de hardware y de software para la instalación de ERECASE se muestran en la siguiente tabla.

Recurso	Mínimo	Recomendado
CPU	Pentium III 500	Pentium 4 1600
Memoria	256 MB	512 MB
Espacio en disco	30 MB	30 MB o más
Sistema operativo	Microsoft Windows XP, 2003, Vista, 2008, 7	

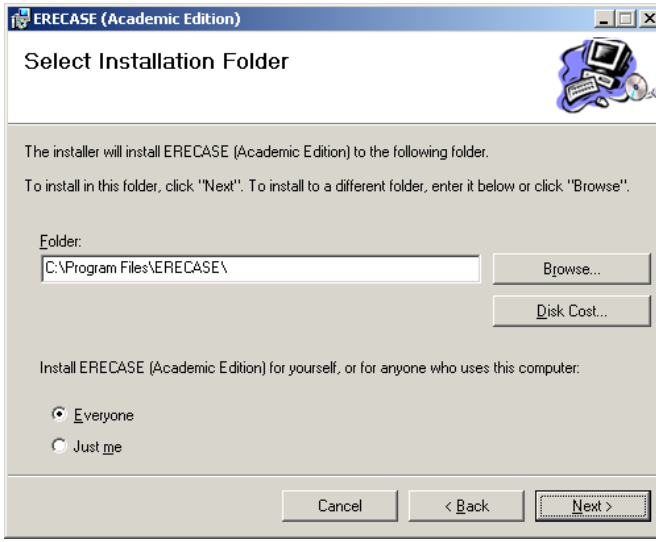
Para la instalación de la herramienta ERECASE es necesario primeramente tener el software, luego ejecutar el SETUP.EXE para comenzar, como se muestra en la siguiente figura.



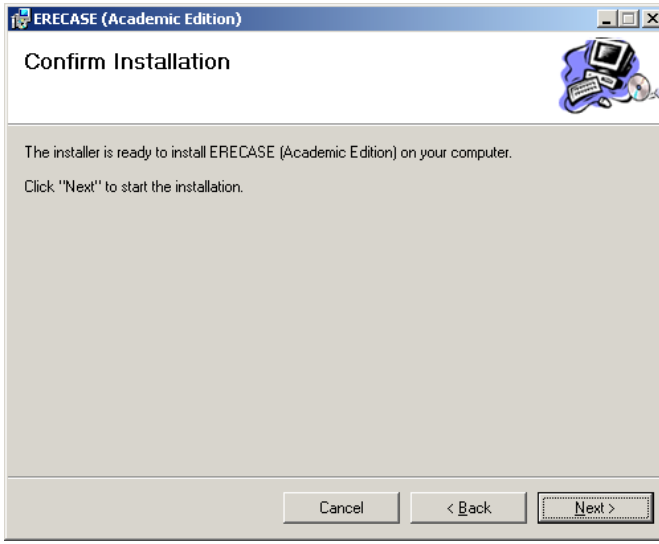
Se muestra la siguiente ventana de presentación, hacer clic en Next.



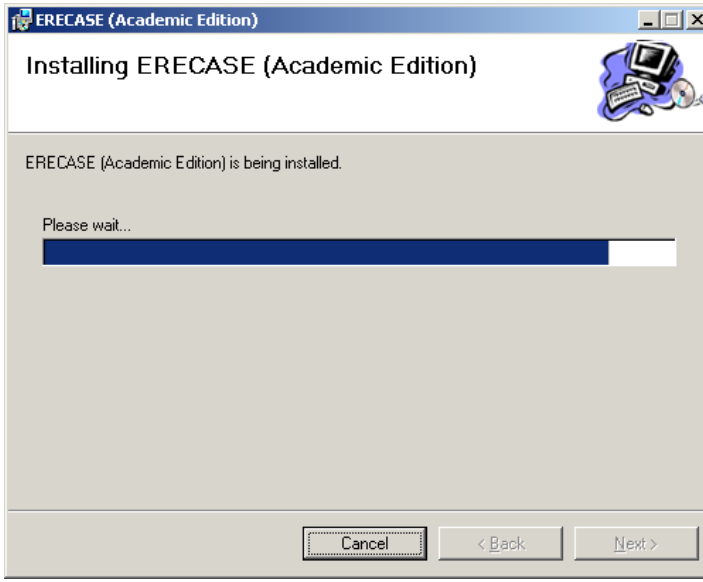
En este diálogo se selecciona la carpeta donde se va a realizar la instalación, por ejemplo C:\Program Files\ ERECASE como se muestra en la siguiente figura.



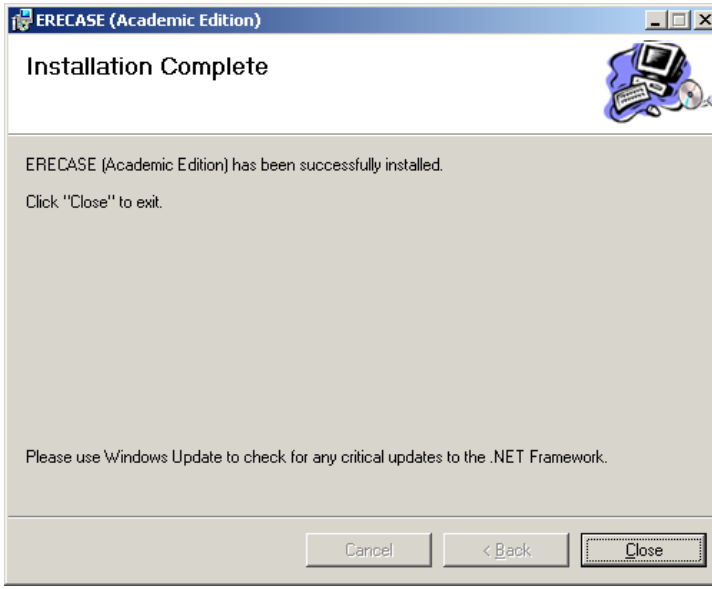
A continuación hacer clic en Next.



Para comenzar la instalación hacer clic en Next.



Al terminar la instalación se presenta la siguiente ventana:

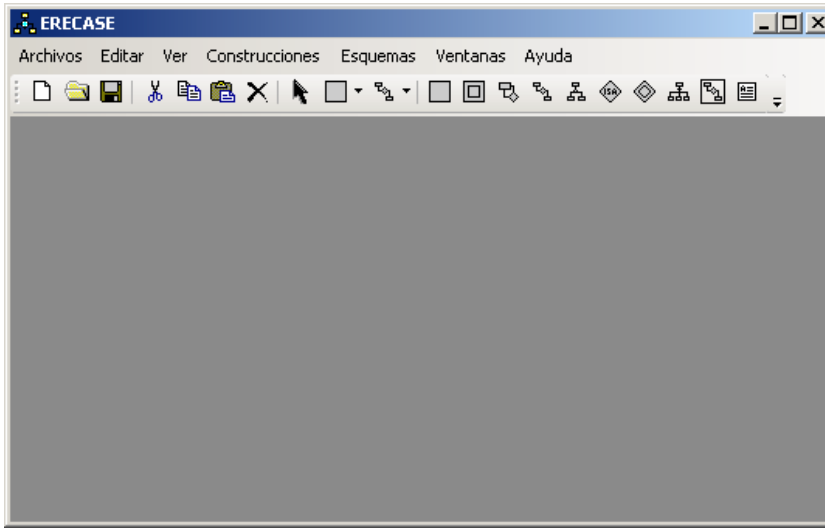


Hacer clic en Close para terminar el programa de instalación.

Ya después de instalado, al ejecutarse el software, este brinda una opción para seleccionar el idioma con el cual desea trabajar; es decir, el idioma en el que aparecerán cada una de las palabras de la herramienta.

Por último se muestra la ventana inicial de la herramienta ERE-

CASE, la que será descrita en el próximo epígrafe.



4.4. La interfaz de usuario de ERECASE

ERECASE está concebida como una aplicación MDI (del inglés Multi Document Interface) que le permite al diseñador trabajar con varios diagramas simultáneamente. Las partes fundamentales de la interfaz son (véase figuras 4.3 y 4.4):

- *Contenedor del diagrama*: constituye el espacio para la construcción del diagrama ERE en general. El contenedor puede ser ampliado según las necesidades del diseño usando las barras de desplazamiento.
- *Explorador del diagrama*: permite acceder de forma rápida a las propiedades de una construcción determinada; si las propiedades de la construcción son editables.
- *Menú y barra de herramientas*: agrupa las acciones y opciones necesarias para el trabajo con la herramienta y la personalización.

- *Esquemas y generación de script SQL:* muestra la estructura de los esquemas a partir de un esquema ERE y la opción de generar el script asociado al diagrama planteado.
- *Editores de Propiedades:* son claves en el modelado de datos, pues son los que permiten manipular todas las características de las construcciones del diagrama ERE, como los atributos, el nombre de las entidades, la descripción, etc.

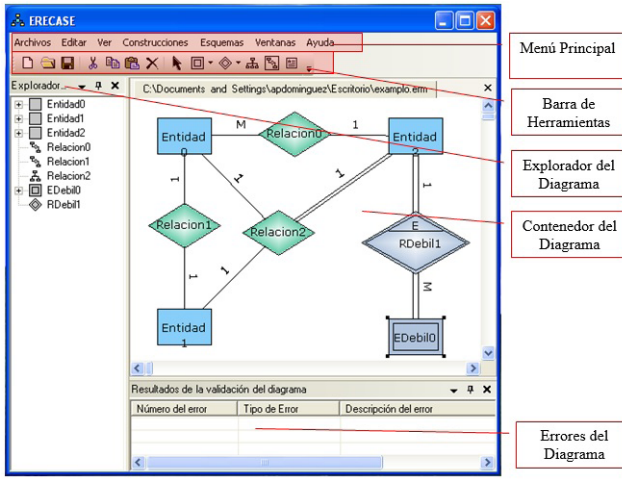


Figura 4.3. Interfaz gráfica de ERECASE.

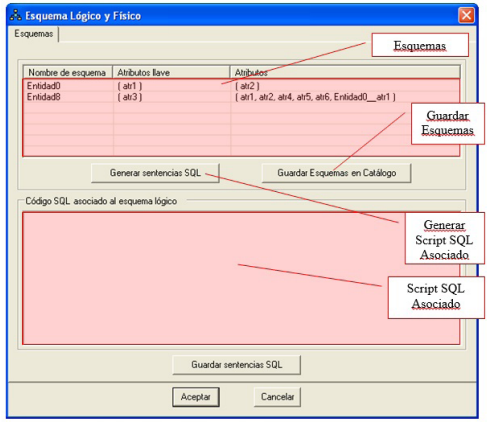


Figura 4.4. Esquemas y Generación de Script SQL en ERECASE.

4.5. Comenzando a trabajar con ERECASE

Al ejecutarse el software, este brinda una opción para seleccionar el idioma con el cual desea trabajar; es decir, el idioma en el que aparecerá cada una de las palabras de la herramienta. Para comenzar a trabajar con el ERECASE se debe crear un nuevo diagrama ER, para esto debe ir a la opción **Archivos** del menú y dentro de esta a **Nuevo Diagrama (Ctrl+N)** o hacer clic en el botón correspondiente en la barra de herramientas.

Seguidamente aparecerá un cuadro de diálogo (Figura 4.5) donde se debe especificar el nombre del nuevo diagrama y el directorio donde se guardará toda la información referente a este.

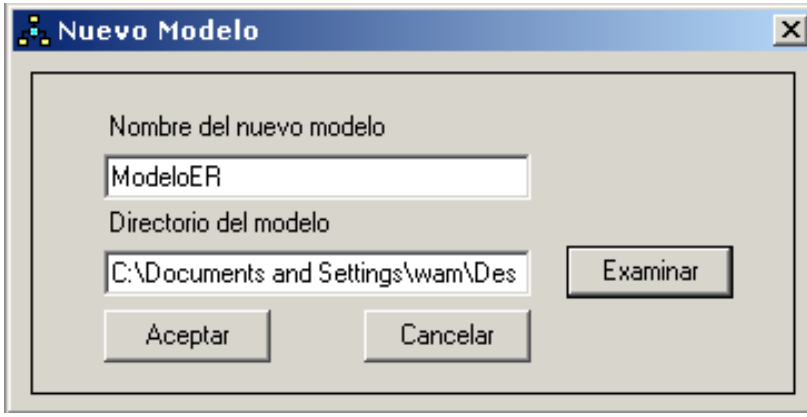


Figura 4.5. Creando un nuevo diagrama.

Los cambios realizados al modelo serán guardados en un archivo con el nombre especificado anteriormente y con extensión **.erm.**

Para abrir un diagrama ER ya creado y continuar trabajando, se debe seleccionar la opción Abrir Diagrama (Ctrl+A) en el menú Archivos o hacer clic en el botón correspondiente en la barra de herramientas.

Luego aparecerá un diálogo clásico para abrir ficheros en el cual se debe seleccionar el diagrama sobre el cual desea trabajar. Automáticamente se crea un catálogo del sistema SIADBDD,

donde se puede almacenar el diagrama ERE correspondiente a la base de datos, así como los esquemas generados a partir del diagrama.

Si se abre un diagrama ya existente o se encuentra trabajando en uno nuevo, se tiene la opción Guardar Diagrama en el menú Archivos que permite ir guardando cada uno de los cambios que vayan haciendo, también existe un botón correspondiente en la barra de herramientas.

En el menú Archivos existe un submenú Cerrar Diagrama que cierra el diagrama que se tenga seleccionado, antes de cerrarlo aparece un diálogo el cual confirma si se desean guardar los cambios, al seleccionar Si se guarda automáticamente en la localización donde se encuentra el diagrama; si se seleccionar No, no se guardan los cambios, en ambos casos se cierra el diagrama.

El submenú Terminar que puede encontrar similarmente en el menú Archivos, si tiene algún diagrama abierto aparece un diálogo similar al que aparece cuando se da Cerrar Diagrama, si no tiene ningún diagrama cierra la herramienta, este submenú cumple la misma función que el botón Cerrar que aparece en la esquina superior derecha de la herramienta representado por una cruz.

El menú Ver posibilita mostrar los paneles *Explorador del diagrama y Resultados de la validación del diagrama* lo cual se logra dando clic izquierdo en los submenús Explorador del diagrama y Resultados de la validación del diagrama respectivamente.

4.6. El editor gráfico

El editor gráfico es el componente principal de diseño de ERECASE, es donde el diseñador crea y edita las construcciones del diagrama ERE, comenzando desde cero o usando partes de otros esquemas ya creados.

¿Cómo trabajar con el editor gráfico de ERECASE?

La característica principal del editor radica en seleccionar objetos (construcciones ERE) y hacer clic en la ventana de edición para insertarlos, estos objetos se pueden redimensionar y la edición de sus propiedades se logra a través de cuadros de diálogos o exploradores.

Los objetos que se pueden adicionar al esquema en diseño se encuentran en la barra de herramientas de la aplicación (véase figura 4.3). Las mismas construcciones se pueden acceder usando el menú Construcciones en el menú principal.

ERECASE contiene un explorador del diagrama que permite visualizar las construcciones insertadas en el esquema y constituye una vía de acceso rápido a las propiedades de estas. A continuación se explican los aspectos y la metodología para la creación de las construcciones en un diagrama ERE utilizando ERECASE.

4.6.1. Insertando entidades

Para insertar una entidad, se debe elegir el tipo de entidad a insertar (fuerte o débil) en el menú Construcciones > Entidades > Entidad Fuerte o Entidad Débil o en el botón correspondiente en la barra de herramientas (véase figura 4.6) se debe hacer clic en el tipo de entidad correspondiente y dar clic en el editor para que aparezca la entidad, si se da clic momentáneamente aparece la entidad con un tamaño estándar, si se quiere especificar un tamaño determinado arrastrar el ratón con el clic primario presionado en el espacio en blanco y luego soltarlo.

Figura 4.6. Selección de Entidades.

Al liberar el botón del ratón aparecerá la entidad en la zona especificada, la cual se puede mover, redimensionar o invocar el editor de las propiedades correspondiente (véase figura 4.7 y figura 4.8), haciendo doble clic sobre la entidad.

La figura 4.7 muestra la ventana de diálogo de propiedades para asignar un nombre apropiado a la entidad, además de una sección que permite agregar un comentario a dicha entidad si se desea.

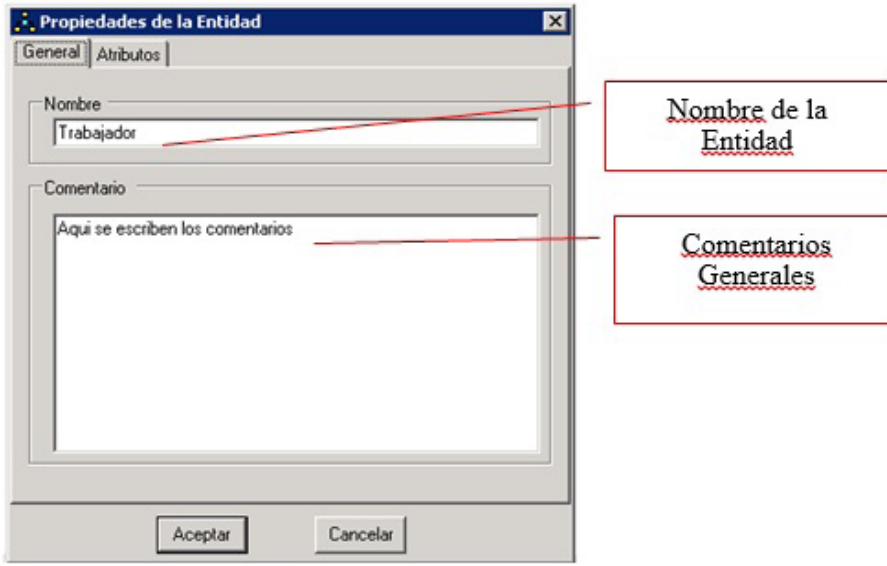


Figura 4.7. Editor de propiedades de una Entidad.

La figura 4.8 muestra cómo mediante el editor de propiedades de las entidades se pueden especificar sus atributos, además señala cuál/cuáles de ellos constituirán identificadores, lo cual se logra marcando en las cajas de chequeo correspondientes a los atributos que serán identificadores en la columna Identificador.

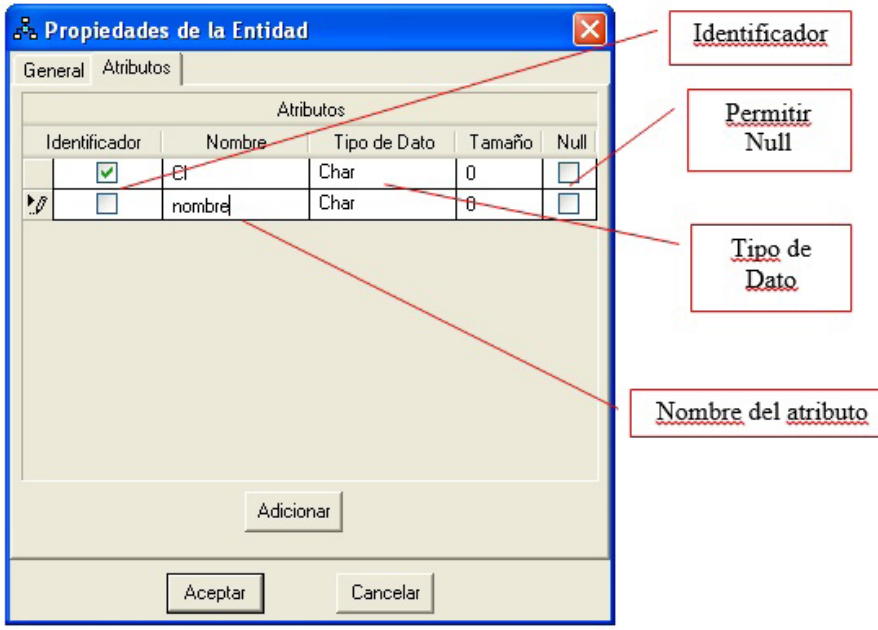


Figura 4.8. Editor de propiedades de una Entidad.

Los botones que presenta este editor son los clásicos Aceptar, Cancelar y Adicionar, este último es para cuando se están definiendo los atributos, se insertan uno a uno.

4.6.2. Insertando interrelaciones

Para insertar una interrelación se debe seleccionar el tipo que se desee en el menú Construcciones > Interrelaciones o en el botón correspondiente en la barra de herramientas (véase la figura 4.9) y luego se debe hacer clic sobre el espacio en blanco y aparecerá el rombo que representa la interrelación seleccionada, automáticamente aparecerá una línea discontinua que parte del rombo y sigue al cursor del ratón, esperando que se le indique las entidades que participarán en dicha relación. Si se selecciona una interrelación no deseada se puede presionar la tecla **Esc** y se elimina la inserción de la interrelación.

Figura 4.9 Selección de interrelaciones.

Para conectar una entidad a una relación se debe hacer clic sobre la entidad y aparecerá una línea de conexión entre ambas. El programa determina automáticamente el número de conexiones que debe tener una interrelación, según cuál sea esta.

Las interrelaciones jerárquicas o de dependencia necesitan especificar una entidad padre o genérica, tienen una particularidad en el momento de conectarlas. Esta particularidad consiste en que la primera entidad seleccionada constituye la entidad hija de la interrelación y la segunda la entidad padre. Por ejemplo: si se inserta una interrelación ISA, que representa herencia, según la semántica de A ISA B, entonces se selecciona primeramente la entidad A y luego se selecciona la entidad B, esto quiere decir que A es un subconjunto de B.

Las interrelaciones al igual que las entidades se pueden mover, redimensionar y editar sus propiedades dependiendo del tipo. A las relaciones binarias se le pueden editar sus propiedades, para ello se utiliza un diálogo (véase las figuras 4.10, 4.11 y 4.12). En la figura 4.10 se muestra cómo se especifica el nombre adecuado a la interrelación binaria.

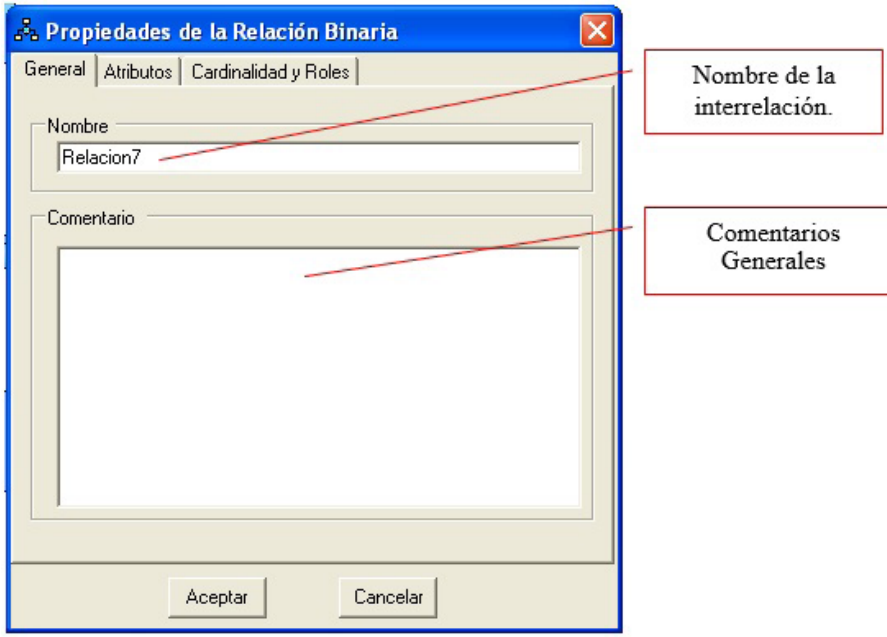


Figura 4.10. Editor de interrelaciones binarias.

En la figura 4.11 se muestra el mismo editor de las interrelaciones binarias, pero muestra cómo se especifican los atributos, a diferencia de las entidades a las interrelaciones no hay que especificarles atributos llaves.

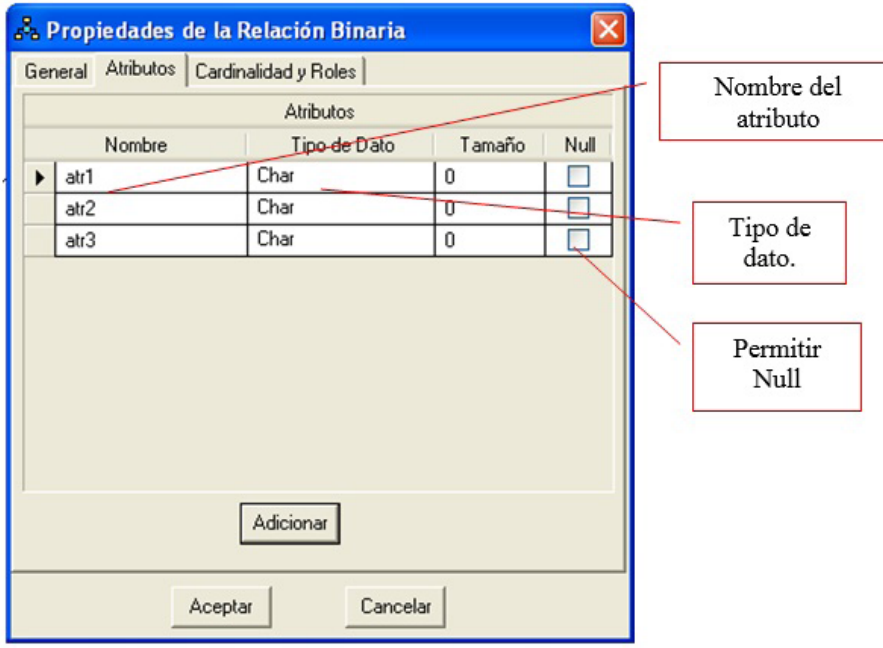


Figura 4.11. Editor de las relaciones binarias.

La figura 4.12 muestra cómo se le especifican las cardinalidades y los roles a las relaciones binarias. Las cardinalidades pueden ser mínima y máxima, la mínima varía de 0 a 1 y las máxima de 1 a m (muchos). Los roles hacen aún más descriptiva la problemática que se desea representar.

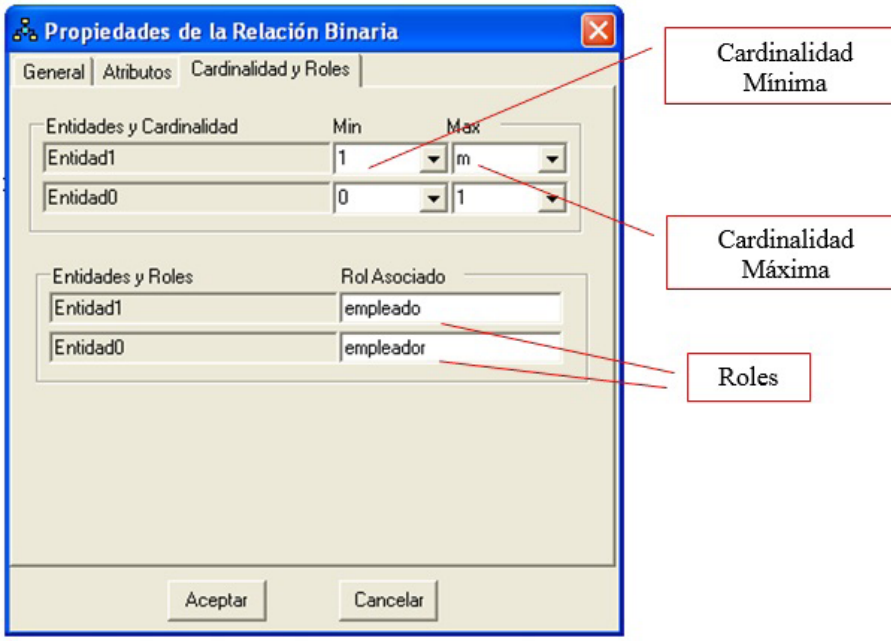


Figura 4.12. Editor de las relaciones binarias.

Las interrelaciones recursivas involucran una sola entidad, estas relaciones también cuentan con un editor de propiedades que es similar al de las relaciones binarias, este editor solo se diferencia en cuanto a las especificaciones de los roles y las cardinalidades (véase figura 4.13). En esta interrelación es imprescindible la especificación de los roles, pues como interviene una única entidad puede causar confusión al interrelacionarse un elemento de ella con otro elemento de la misma entidad.

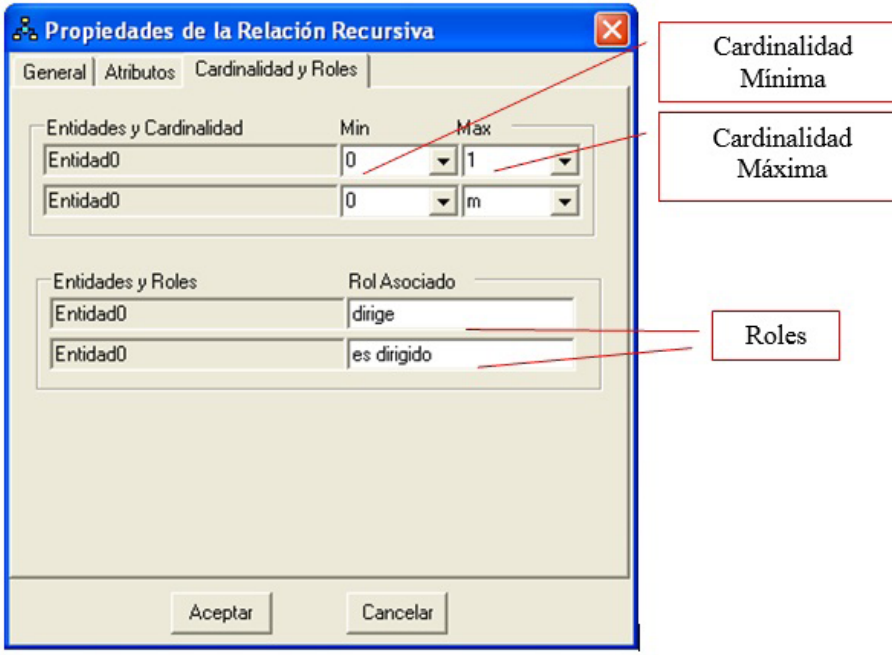


Figura 4.13. Editor de interrelaciones recursivas.

Las interrelaciones ternarias también cuentan con un editor de propiedades, el cual es similar al editor de las interrelaciones binarias y recursivas, de igual forma se diferencian al especificar las cardinalidades y los roles, puesto que las ternarias involucran tres entidades. La figura 4.14 muestra el editor de las interrelaciones ternarias, específicamente donde se especifican las cardinalidades y los roles.

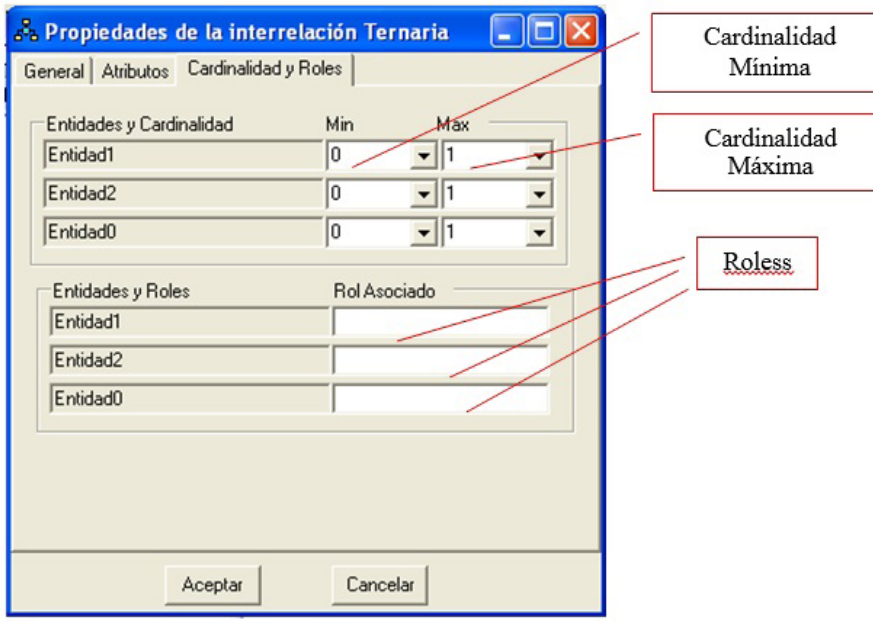


Figura 4.14. Editor de interrelaciones ternarias.

4.6.3. Insertando generalizaciones

Para insertar una generalización primero se debe seleccionar del menú principal la opción Construcciones > Generalización o en el botón correspondiente en la barra de herramientas (véase figura 4.15), después de seleccionarla con solo dar clic sobre el espacio en blanco aparece un círculo y desde este sale una línea discontinua que sigue el cursor de ratón para la selección de las entidades participantes.

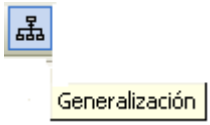


Figura 4.15. Botón Generalización de la barra de herramientas.

Como ocurre con las relaciones la generalización creada se debe conectar con las entidades correspondientes. Para las generalizaciones, la primera entidad seleccionada constituye

la entidad padre o más general, el resto de las entidades son entidades hijas o específicas. Se le debe indicar que se ha concluido de conectar la generalización haciendo doble clic sobre el espacio en blanco. Las generalizaciones también tienen propiedades editables (véase figura 4.16).

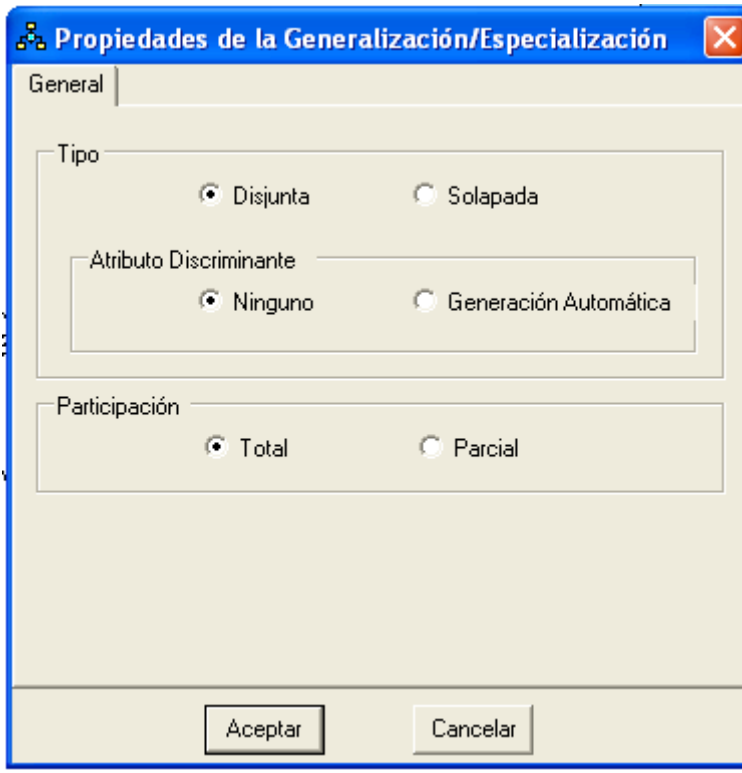


Figura 4.16. Editor de propiedades de la generalización.

4.6.4. Insertando agregaciones

Para crear una agregación, luego de haber seleccionado la opción correspondiente en el menú principal (Construcciones > Agregación) o en el botón correspondiente en la barra de herramientas (véase figura 4.17), se debe especificar un rectángulo que contenga una sola interrelación de asociación, así como las entidades que participen en dicha interrelación. Al liberar el bo-

tón del ratón se muestra la asociación dentro de un rectángulo y a partir de ese momento la interrelación y las entidades pasan a formar parte de la agregación, sin poder participar en otra agregación que fuese insertada posteriormente. La agregación creada toma un nombre automáticamente. Cuando se mueve una agregación todas las construcciones que la componen se mueven junto con esta.

Figura 4.17. Botón agregación de la barra de herramientas.

La agregación tiene un editor en el cual se le especifica el nombre (véase figura 4.18) y sus atributos (véase figura 4.19).

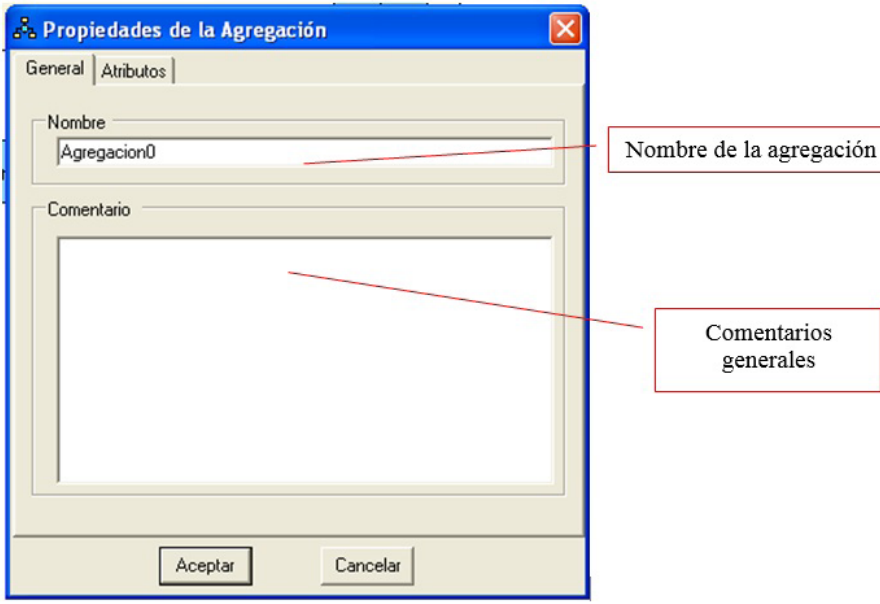


Figura 4.18. Editor de las propiedades de la agregación.

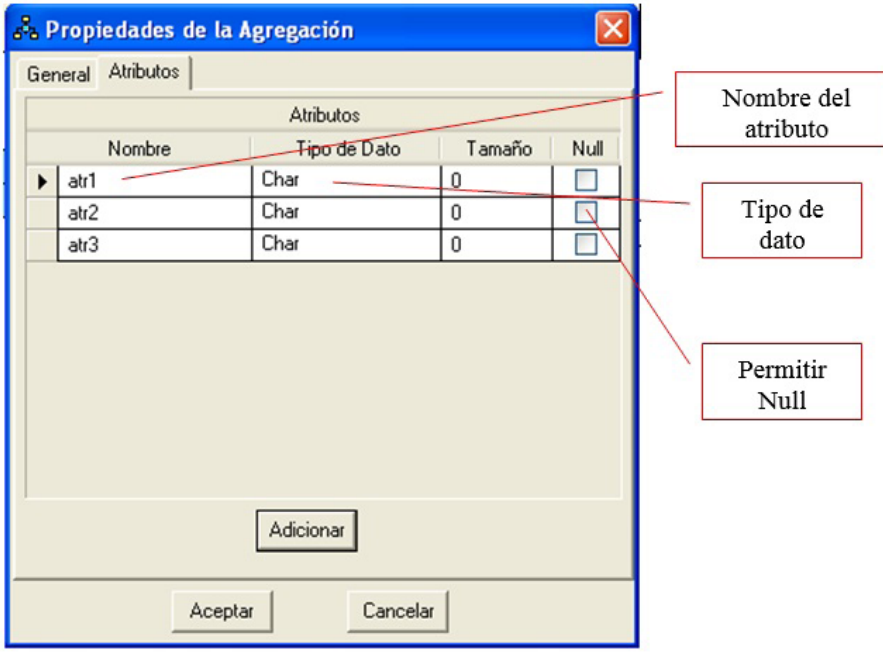


Figura 4.19. Editor de las propiedades de la agregación.

4.6.5. Copiando y pegando

El editor de ERECASE posee un menú Editar el cual contiene cuatro submenús clásicos Cortar, Copiar, Pegar y Borrar, todos tienen un botón correspondiente en la barra de herramientas.

La herramienta permite el copiado y pegado de objetos pero con ciertas particularidades, esto se debe a la complejidad que puede tener un esquema determinado.

La herramienta solo permite copiar, cortar y pegar entidades ya que como las relaciones siempre tienden a especificar un nuevo concepto, no soporta que se copien relaciones, esto es para asegurar que no se repitan conceptos en un mismo diagrama con diferentes nombres.

4.6.6. Redimensionado de las construcciones

Todas las construcciones en ERECASE tienen la posibilidad de redimensionarse a gusto del diseñador; excepto las agregaciones. Para esto primeramente se debe seleccionar la construcción que se quiere redimensionar, al hacerlo la construcción aparecerá marcada en sus extremos por un conjunto de puntos de selección, se debe mantener presionado clic sobre uno de ellos y arrastrar el ratón para modificar el tamaño de las relaciones, para el caso de las entidades, cuando se está sobre un borde aparece una doble flecha, ya sea horizontal o vertical, que indica que manteniendo presionado el clic primario se puede redimensionar la entidad en la dirección especificada según el borde que se esté seleccionando.

4.6.7. Selección múltiple de objetos

ERECASE permite la selección de varios objetos simultáneamente, esto es de gran ayuda a la hora de modificar la posición de un conjunto de objetos sin perder la posición relativa entre ellos. Para seleccionar varias construcciones a la vez, se debe hacer clic sobre el espacio en blanco y arrastrar el ratón hasta formar un rectángulo que contenga todos los objetos que se desea seleccionar.

Para cancelar una selección múltiple simplemente se hace clic sobre el espacio en blanco o se selecciona una construcción que no pertenezca a la selección múltiple.

4.7. Validación del diagrama conceptual

ERECASE realiza distintos tipos de validaciones al diagrama conceptual en construcción en tiempo de diseño. Tales errores se muestran en el explorador Resultados de la validación del diagrama (véase figura 4.20), se muestra un código de error, el tipo de error, la causa y/o la construcción o construcciones que incurren en tal error.

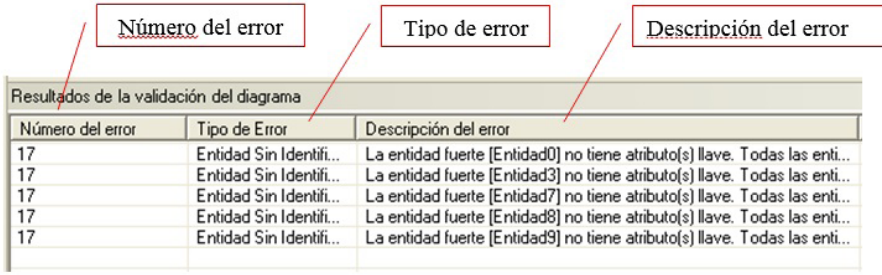


Figura 4.20. Verificación de Errores.

4.8. Transformación del esquema conceptual al esquema lógico

Para comenzar el proceso de transformación se debe seleccionar Esquemas > Generar Esquemas (F5) en el menú de la herramienta. Antes de transformar al modelo lógico, previamente se muestran los esquemas que se generaron, si el diseñador está de acuerdo con los esquemas que se generaron puede transformar al modelo lógico, de lo contrario puede cambiar los nombres de los atributos no identificadores de un esquema. Para esto debe hacer doble clic sobre un esquema y a continuación se muestra un diálogo con un árbol donde se visualizan los atributos editables, a continuación se selecciona el atributo al cual se le va a cambiar el nombre.

La generación del modelo lógico es semiautomática, esto es que, para completar el proceso, en algunas ocasiones es necesaria la intervención del diseñador para tomar decisiones que el módulo de generación de esquemas no puede ejecutar por sí solo. La intervención del diseñador se realiza a través de cuadros de diálogos que son mostrados durante la generación de los esquemas.

La forma de transformar las de interrelaciones de asociación depende de su cardinalidad mínima y máxima. Existen varios casos bien determinados que siempre tienen una misma manera de transformar, por ejemplo las asociaciones muchos-muchos, pero en algunos otros casos la transformación depende de la

semántica del modelo. Por esto se le permite al diseñador decidir qué forma usar para transformar las relaciones que no tienen un método de transformación determinado, ya sea migrando las llaves de una entidad a otra o generando un esquema aparte para la asociación.

El cuadro de diálogo que se muestra en la figura 4.21 es para que el diseñador determine el tipo de transformación de la relación especificada. Como se puede observar en la figura, el diseñador debe elegir una de las opciones de transformación brindadas por la herramienta para continuar con el proceso de generación de los esquemas lógicos. Para tal selección el diseñador tiene que dominar el mini mundo que está representando ya que si hace migrar llaves a entidades con participación opcional en la relación, puede generar muchos valores nulos dentro del esquema.

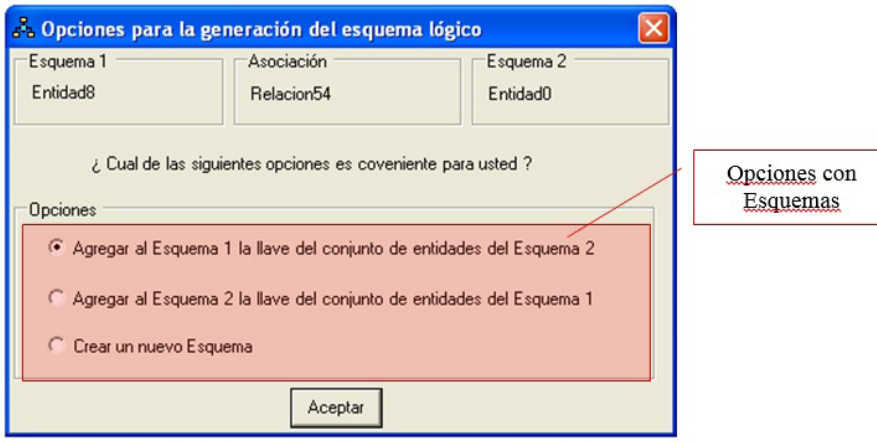


Figura 4.19. Diálogo para la determinación de esquemas.

4.9. Generación de las sentencias SQL

Para generar las sentencias de creación de las tablas y las relaciones entre ellas a partir de los esquemas obtenidos, en el diálogo de los esquemas (véase figura 4.2) se debe hacer clic en el botón Generar sentencias SQL. Automáticamente se genera

el código SQL asociado a los esquemas y se muestra en la zona de Código SQL asociado al esquema lógico. El código generado por ERECASE se puede guardar en formato de texto para su posterior utilización en el gestor de bases de datos *MS SQL Server*, no sin resaltar que las sentencias SQL utilizadas (CREATE TABLE y ALTER TABLE) son estándar para *SQL99, Oracle 9i, DB2, UDB 8.1*.

Para guardar los esquemas relacionales ERECASE tiene en la ventana de la figura 4.2 el botón Guardar Esquemas en Catálogo.

4.10. Ayuda de la herramienta

La herramienta tiene un menú Ayuda, dentro de este contiene un submenú Ayuda F1 que permite al diseñador documentarse acerca de las características conceptuales del modelo ERE como tal, además presenta propiedades de las diferentes construcciones. Dentro del menú Ayuda también se encuentra un submenú Acerca de ERECASE que trata sobre el propio trabajo con la herramienta.

Referencias bibliográficas

Agnarsson, G., & Greenlaw, R. (2014). *Graph Theory: Modeling, Applications, and Algorithms* (2nd. ed.): New Jersey: Prentice Hall.

American National Standards Institute. (1999). *ISO/IEC 9075:1999, Database Language SQL*: Washington D.C: ANSI/ISO/IEC International Standard.

Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., & Zakharyaschev, M. (2007). *Reasoning over Extended ER Models.* Paper presented at the ER 2007.

Balaban, M., & Maraee, A. (2006). *Consistency of UML Class Diagrams with Hierarchy Constraints.* Paper presented at the NGITS 2006.

Balaban, M., Maraee, A., & Sturm, A. (2007). Reasoning with UML Class Diagrams: Relevance, Problems, and Solutions – a Survey. Recuperado de http://www.cs.bgu.ac.il/~mira/CDReasoning-07.pdf

Barker, R. (2011). *CASE Method: Entity Relationship Modeling* (2nd. ed.). New York: Addison-Wesley Publishing Company.

Batini, C., Barone, D., Garasi, M. F., & Viscusi, G. (2012). *Design and Use of ER Repositories: Methodologies and Experiences in eGovernment Initiatives*. Paper presented at the ER 2012.

Batini, C., Ceri, S., & Navathe, S. B. (2011). *Conceptual Database Design: An Entity-Relationship Approach* (3rd. ed.). Redwood City: Benjamin/Cummings.

Batini, C., Navathe, S. B., Ceri, S., Martín García, A. V., & Romero Ibancos, D. (2014). *Diseño Conceptual de Bases de Datos: Un enfoque de entidades-interrelaciones* (2nd. ed.). Buenos Aires: Addison-Wesley/Diaz de Santos.

Bienemann, A., Schewe, K.-D., & Thalheim, B. (2013). *Towards a Theory of Genericity Based on Government and Binding*. Paper presented at the ER 2013. Tucson.

Brodie, M., Mylopoulos, J., & Schmidt, J. (2012). *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. (3rd. ed.). New York: Springer-Verlag.

Bruce, T. A. (2013). *Designing Quality Database with IDEF1X Information Models* (3rd. ed.). New York: Dorset House Publishing Company.

Cadoli, M., Calvanese, D., De Giacomo, G., & Mancini, T. (2004). *Finite satisfiability of UML class diagrams by constraint programming.* Paper presented at the International Workshop on Description Logics (DL'2004).

Calì, A. (2007, November 5-9, 2007). *Querying Incomplete Data with Logic Programs: ER Strikes Back.* Paper presented at the ER 2008, Auckland, New Zealand.

Calvanese, D., & Lenzerini, M. (1994). *On the Interaction Between ISA and Cardinality Constraints*. Paper presented at the Tenth International Conference on Data Engineering, Houston.

Chen, P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1), 9-36.

Chen, P. (1983). *A Preliminary Framework for Entity-Relationship Model.* Paper presented at the Information Modeling and Analysis.

Chen, P. (2011). *The ER Designer: Reference Manual* (4th. ed.). Baltimore: Chen & Associates.

Chen, P. P. (2006). *Suggested Research Directions for a New Frontier – Active Conceptual Modeling*. Paper presented at the ER 2006.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM,* 13(6), 377-387.

Combi, C., Degani, S., & Jensen, C. S. (2008). *Capturing Temporal Constraints in Temporal ER Models.* Paper presented at the ER 2008.

Connolly, T. M., & Begg, C. E. (2014a). Herramientas CASE *Sistemas de Bases de Datos. Un enfoque práctico para diseño, implementación y gestión.* Reading: Pearson Addison Wesley.

Connolly, T. M., & Begg, C. E. (2014b). *Sistemas de Bases de Datos. Un enfoque práctico para diseño, implementación y gestión.* Reading: Pearson Addison Wesley.

Date, C. J. (2013). *An Introduction to Database Systems* (9th ed.). Boston: Addison-Wesley.

Davies, I., Green, P., Rosemann, M., & Gallo, S. (2004). *Conceptual Modelling – What and Why in Current Practice.* Paper presented at the ER 2004.

Davies, I., Green, P., Rosemann, M., Indulska, M., & Gallo, S. (2006). How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering, 58*, 358-380. Recuperado de https://dl.acm.org/citation.cfm?id=1162464

De Miguel, A., & Piattini, M. (1993). *Concepción y diseño de bases de datos: Del Modelo E/R al modelo relacional.* Madrid: RA-MA.

Dey, D., Storey, V. C., & Barron, T. M. (1999). Improving Database Design through the Analysis of Relationships. *ACM Transactions on Database Systems, 24*(4), 453-483. Recuperado de https://pdfs.semanticscholar.org/1048/c8dc57e1ed379987d7c733fe1bd66ab02a3e.pdf

Dullea, J., & Song, I.-Y. (1997). *An Analysis of Cardinality Constraints in Redundant Relationships*. Paper presented at the 6th InternationalConference on Information and Knowledge Management (CIKM '97). Las Vegas.

Dullea, J., & Song, I.-Y. (1998a). *An Analysis of Structural Validity of Ternary Relationships in Entity Relationship Modeling*. Paper presented at the 7th International Conference on Information and Knowledge Management (CIKM '98). Washington, D.C.

Dullea, J., & Song, I.-Y. (1998b). *An Analysis of the Structural Validity of Unary and Binary Relationships in Entity Relationship Modeling*. Paper presented at the 4th International Conference on Computer Science and Informatics. Research Triangle Park.

Dullea, J., & Song, I.-Y. (1999). *A Taxonomy of Recursive Relationships and Their Structural Validity in ER Modeling*. Paper presented at the ER 1999. Paris.

Dullea, J., Song, I.-Y., & Lamprou, I. (2003). An Analysis of Structural Validity in Entity-Relationship Modeling. *Data & Knowledge Engineering, 47*, 167-205. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.4593&rep=rep1&type=pdf

Elmasri, R., & Navathe, S. B. (2012). *Fundamentals of Database Systems* (7th ed.). Boston: Addison-Wesley.

Elmasri, R., Weeldreyer, J., & Hevner, A. (2012). The Category Concept: An Extension to the Entity-Relationship Model. *Data & Knowledge Engineering,* 1(4), 75-116.

Ferg, S. (1991). *Cardinality Concepts in Entity-Relationship Modeling*. Paper presented at the ER 1991. San Mateo.

Finkelstein, C. (1998). Information Engineering Methodology *Handbook on Architectures of Information Systems*. Berlin: Springer-Verlag.

Franconi, E. (2011, November 5-9, 2007). *Conceptual Schemas and Ontologies for Database Access: Myths and Challenges*. Paper presented at the ER 2011. Auckland.

García, C. (2010). *Enfoque sistémico a la modelación de datos, base para el desarrollo de una herramienta CASE*. (Tesis de Doctorado). Santa Clara: Universidad Central de Las Villas.

García, C., Rodríguez, A., Cabrera, N., & González, L. (2008a). Caracterización de la dependencia de existencia en interrelaciones binarias. *Revista de Ciencias Matemáticas, 25*, 3-12. Recuperado de http://www.sinewton.org/numeros/numeros/85/Volumen_85.pdf

García, C., Rodríguez, A., Cabrera, N., & González, L. (2008b). *DETCOI: un algoritmo para la detección y corrección de inconsistencias en esquemas lógicos*. Paper presented at the XIV Congreso Latino Ibero Americano de Investigación de Operaciones, CLAIO 2008. Cartagenas de Indias.

García, C., Rodríguez, A., Cabrera, N., & González, L. (2008c). *Validación de esquemas lógicos de base de datos.* Paper presented at the V Conferencia Internacional FIE'2008 Santiago de Cuba: Universidad de Oriente.

García, C., Rodríguez, A., Cabrera, N., & González, L. (2009). *ERECASE: una herramienta de ayuda al diseño de bases de datos relacionales*. Paper presented at the I Evento Internacional la Matemática, la Física y la Computación en el siglo XXI Holguín: Instituto Superior Pedagógico.

García, C., Rodríguez, A., Cabrera, N., & González, L. (2010). Detección y corrección de inconsistencias en esquemas lógicos de bases de datos. *Revista Facultad de Ingeniería, Universidad de Antioquia,* (55), 163-171. Recuperado de http://www.scielo.org.co/pdf/rfiua/n55/n55a17.pdf

García, L., & Montes de Oca, M. (2015). Sistemas de Bases de Datos: *Modelación y Diseño* (2da. ed.). La Habana: Eitorial Félix Varela.

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2012). *Database Systems. The Complete Book* (3rd. ed.) New Jersey: Prentice Hall.

Girju, R., Badulesco, A., & Moldovan, D. (2006). Automatic Discovery of Part-Whole Relations. *Computational Linguistics, 32*(1), 83-135. Recuperado de https://dl.acm.org/citation.cfm?id=1168116

Grassmann, W. K., & Tremblay, J.-P. (2011). *Matemática Discreta y Lógica. Una perspectiva desde la Ciencia de la Computación* (3rd. ed.). New Jersey: Prentice Hall.

Gruber, T. R. (1995). Towards Principles for the Design of Ontologies used for Knowledge Sharing. *International Journal of Human-Computer Studies, 43*(5-6), 7-928. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.5775&rep=rep1&type=pdf

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models.* Thesis. Enschede: University of Twente.

Guizzardi, G., & Wagner, G. (2008). *What's in a Relationship: An Ontological Analysis*. Paper presented at the ER 2008.

Hartmann, S. (1998). *On the Consistency of Int-cardinality Constraints*. Paper presented at the ER 1998, Singapore.

Hartmann, S. (2000). *On Interactions of Cardinality Constraints, Keys and Functional Dependencies*. Paper presented at the FoIKS 2000. Burg.

Hartmann, S. (2001a). *Coping with Inconsistent Constraint Specifications.* Paper presented at the ER 2001. Yokohama.

Hartmann, S. (2001b). On the implication problem for cardinality constraints and functional dependencies. *Annals of Mathematics and Artificial Intelligence,* 33(2-4), 253-307. Recuperado de https://dl.acm.org/citation.cfm?id=590466

Hartmann, S. (2003). *Soft Constraints and Heuristic Constraint Correction in Entity-Relationship Modelling*. Paper presented at the International workshop on semantics in databases. Dagstuhl Castle.

Hartmann, S., & Link, S. (2007). *Collection Type Constructors in Entity-Relationship Modeling*. Paper presented at the ER 2007.

Hartmann, S., Link, S., & Trinh, T. (2009). Constraint acquisition for Entity-Relationship models. *Data & Knowledge Engineering, 68*(10), 1128-1155. Recuperado de https://dl.acm.org/citation.cfm?id=1598343

Hay, D. C. (1999). A Comparison of Data Modeling Techniques. Recuperado de http://www.essentialstrategies.com/publications/modeling/compare.htm

Hitchman, S. (2004). Entity Class Classification and Entity Relationships Types. *Journal of Conceptual Modeling,* (31), 16-32. Recuperado de http://web.cs.ucdavis.edu/~green/courses/ecs165a-w11/2-er.pdf

Jones, T. H., & Song, I.-Y. (1996). Analysis of Binary/Ternary Cardinality Combinations in Entity-Relationship Modeling. *Data & Knowledge Engineering, 19*(1), 39-64. Recuperado de www.sciencedirect.com/science/article/pii/0169023X9500036R/pdf?md5=cd17516ea371fe97c916005f3e8cb460&pid=1-s2.0-0169023X9500036R-main.pdf

Jones, T. H., & Song, I.-Y. (2000). Binary Equivalents of Ternary Relationships in Entity-Relationship Modeling: a Logical Decomposition Approach. *Journal of Database Management, 11*(2), 12-19. Recuperado de https://dl.acm.org/citation.cfm?id=344142

Jones, T. H., & Song, I.-Y. (2012). Ternary Relationships: Semantic Requirements and Logically Correct Alternatives *Advanced topics in database research*. Hershey: IGI Publishing.

Keet, C. M., & Artale, A. (2012). Representing and reasoning over a taxonomy of part–whole relations. *Applied Ontology, 3*, 91-110. Recuperado de https://dl.acm.org/citation.cfm?id=1412418

Kolapalli, P., & Dullea, J. (2012). *Semantic Validity and Design Issues of a Data Model*. Paper presented at the International Conference on Information & Knowledge Engineering, IKE 2012. Las Vegas.

Korth, H. F., Silberschatz, A., & Sudarshan, S. (2014). *Database System Concepts* (6th ed.). New York: McGraw-Hill Science/Engineering/Math.

Lenzerini, M., & Nobili, P. (1990). On the satisfiability of dependency constraints in entity–relationship schemata. *Information Systems, 15*(4), 453-461. Recuperado de https://sapienza.pure.elsevier.com/en/publications/on-the-satisfiability-of-dependency-constraints-in-entity-relatio

Lenzerini, M., & Santucci, G. (1983). *Cardinality constraints in the entity-relationship model*. Paper presented at the ER 1983.

Ling, T. (1985). *A normal form for entity-relationship diagrams*. Paper presented at the ER 1985, Chicago.

Maier, D. (2011). *The Theory of Relational Databases* (2nd. ed.). Rockville: Computer Science Press.

Manila, H., & Räihä, K.-J. (2012). *The Design of Relational Databases* (2nd. Ed). Reading: Addison-Wesley Publishing Company Inc.

Markowitz, V., & Shoshani, A. (1992). Representing extended entity-relationship structures in relational databases: A modular approach. *ACM Transactions On Database Systems, 17*(3), 423-464. Recuperado de http://dl.acm.org/citation.cfm?id=132273

McAllister, A. (1998). Complete rules for n-ary relationship cardinality constraints. *Data & Knowledge Engineering, 27*(3), 255-288. Recuperado de http://dl.acm.org/citation.cfm?id=302097

McBrien, P. (2010). *Temporal Constraints in Non-temporal Data Modelling Languages*. Paper presented at the ER 2010.

McFadden, F., & Hoffer, J. (2011). *Modern Database Management* (5th ed.): San Francisco: Benjamin/Cummings Publishing.

Murthy, S., Delcambre, L., & Maier, D. (2006, November 6-9, 2006). *Explicitly Representing Superimposed Information in a Conceptual Model.* Paper presented at the ER 2008. Tucson.

Navathe, S., & Cheng, A. (1983). *A Methodology for Database Schema Mapping from Extended Entity Relationship Models into the Hierarchical Model.* Paper presented at the ER 1983.

Nieto, C., Martínez, P., Cuadra, D., & de Miguel, A. (2012). Profundizando en la semántica de las cardinalidades en el modelo E/R extendido. *JISBD,* 53-54.

Olivé, A. (2012a). Cardinalitiy Constraints *Conceptual Modeling of Information Systems*. Berlin: Springer.

Olivé, A. (2012b). Integrity Constraints *Conceptual Modeling of Information Systems*. Berlin: Springer.

Parsons, J., & Li, X. (2007). *An Ontological Metamodel of Classifiers and Its Application to Conceptual Modelling and Database Design.* Paper presented at the ER 2007.

Piattini, M. G., & Díaz, O. (2012a). *Advanced Database Technology and Design* (3rd. ed.). Norwood: Artech House Inc.

Piattini, M. G., & Díaz, O. (2012b). CASE Tools: Computer Support for Conceptual Modeling. In M. G. Piattini & O. Díaz (Eds.), *Advanced Database Technology and Design*. Norwood: Artech House Inc.

Ponniah, P. (2013). *Database Design and Development* (2nd. ed.). New Jersey: Wiley-Interscience.

Purao, S., & Storey, V. C. (2005). A multi-layered ontology for comparing relationship semantics in conceptuals models of databases. *Applied Ontology*(1), 117-139. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.8700&rep=rep1&type=pdf

Sakai, H. (1983). *Entity-relationship approach to logical database design.* Paper presented at the Entity-Relationship Approach to Software Engineering.

Santos, I.-J., Martínez, P., & Cuadra, D. (2007). *On the semi-automatic validation and decomposition of ternary relationships with optional elements.* Paper presented at the International Conference on Enterprise Information Systems. Funchal.

Shanks, G., Tansley, E., & Weber, R. (2003). Using Ontology to Validate Conceptual Models. *Communications of the ACM, 46*(10), 85-89. Recuperado de ftp://ftp.gunadarma.ac.id/pub/books/Communication-ACM/Oktober-2003/p85-shanks.pdf

Shanks, G., Tansley, E., Nurelini, J., Toblin, D., & Weber, R. (2008). Representing part-whole relationships in conceptual modeling: An empirical evaluation. *MIS Quarterly, 32*(3), 553-573.

Siau, K. (2004). Relationship Construct in Modeling Information Systems: Identifying Relationship Based on Relation Element Theory. *Journal of Database Management, 15*(3). Recuperado de http://vldbarc.org/dblp/db/journals/jdm/jdm15.html

Silberschatz, A., Korth, H., & Sudarshan, S. (2014). *Sistemas de Bases de Datos* (5ta ed.). Reading: Pearson Addison Wesley.

Simsion, G., & Witt, G. C. (2015). *Data Modeling Essentials* (4th. ed.). Massachusetts: Morgan Kaufmann Publishers.

Smith, J., & Smith, D. (1977). Database abstractions: Aggregation and generalization. *ACM Transactions On Database Systems, 2*(2), 105-133. Recuperado de https://dl.acm.org/citation.cfm?id=359620

Snoeck, M., & Dedene, G. (1998). Existence Dependency: The key to semantic integrity between structural and behavioural aspects of object types. *IEEE Transactions on Software Engineering, 24*(4), 1-30. Recuperado de https://www.researchgate.net/publication/220069517_Existence_Dependency_The_Key_to_Semantic_Integrity_Between_Structural_and_Behavioral_Aspects_of_Object_Types

Snoeck, M., & Dedene, G. (2001). Core Modelling Concepts to define Aggregation. *L'Objet, 7*(1), 281-306.

Snoeck, M., Michiels, C., & Dedene, G. (2003). *Consistency by construction: the case of MERODE*. Paper presented at the ER 2003 Workshops.

Song, I.-Y., & Jones, T. H. (1993). *An analysis of binary relationships within ternary relationships*. Paper presented at the ER 1993, Arlington.

Song, I.-Y., Evans, M., & Park, E. K. (1995). A comparative analysis of entity-relationship diagrams. *Journal of Computer and Software Engennering,* 3(4), 427-459. Recuperado de http://www.ischool.drexel.edu/faculty/song/publications/p_jcse-erd.pdf

Storey, V. C. (2001). *Understanding and Representing Relationship Semantics in Database Design*. Paper presented at the NLDB 2000.

Storey, V. C. (2005). Comparing Relationships in Conceptual Modeling: Mapping to Semantic Classifications. *IEEE Transactions on Knowledge and Data Engineering,* 17(11), 1478-1489. Recuperado de ftp://ftp.gunadarma.ac.id/research/IEEE/Knowledge_Data_Engineering/Nov_05/k1478.pdf

Storey, V. C., & Purao, S. (2004). *Understanding Relationships: Classifying Verb Phrase Semantics.* Paper presented at the ER 2004.

Sugumaran, V., & Storey, V. C. (2006). The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment. *ACM Transactions on Database Systems, 31*(3), 1064-1094. Recuperado de https://pdfs.semanticscholar.org/f702/d7b609b318185e7422032a-0d31e13c9d2ecf.pdf

Teorey, T. J., Buxton, S., & Simsion, G. (2013). *Database Design. Know It All* (2nd. ed.). Massachusetts: Morgan Kaufmann Publishers.

Teorey, T. J., Lightstone, S. S., & Nadeau, T. (2014). *Database Modeling and Design: Logical Design* (4th ed.). Massachusetts: Morgan Kaufmann Publishers.

Teorey, T., Yang, D., & Fry, J. (1986). A logical design methodology for relational databases using the extended E-R model. *ACM Computing Surveys*, 18(2), 197-222. Recuperado de https://dl.acm.org/citation.cfm?id=7475

Thalheim, B. (1992). *Fundamentals of cardinality constraints*. Paper presented at the ER 1992, Karlsruhe.

Thalheim, B. (1999). *The Strength of ER Modeling.* Paper presented at the Current Issues in Conceptual Modelling.

Thalheim, B. (2013). *Integrity Constraints Entity-Relationship Modeling. Foundations of Database Technology* (2nd. Ed). Berlin: Springer-Verlag.

Ullman, J. D. (1988). *Principles of Database and Knowledge-Base Systems* (Vol. 1). Rockville: Computer Science Press.

Ullman, J. D., & Widom, J. (2013). *First Course in Database Systems* (4th. ed.). New Jersey: Prentice Hall.

Umanath, N. S., & Scamell, R. W. (2011). *Data Modeling and Database Design* (2nd. ed.).Boston: Thompson Course Technology.

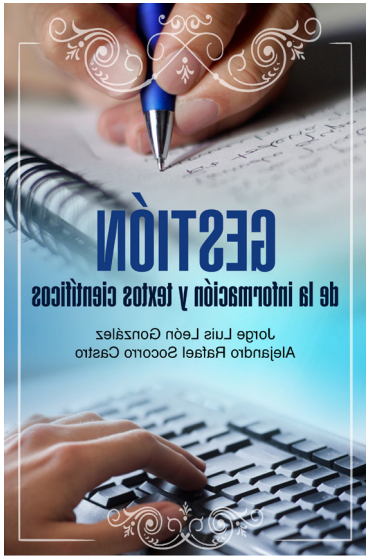
Wand, Y., & Weber, R. (2011). Research Comentary: Information Systems and Conceptual Modeling - A Research Agenda. *Information Systems Research, 19*(2), 363-376. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.407.9324&rep=rep1&type=pdf

Wand, Y., Woo, C., & Wand, O. (2008). *Role and Request Based Conceptual Modeling – A Methodology and a CASE Tool.* Paper presented at the ER 2008.

Webre, N. (1981). *An Extended E-R model and its use on a defense project.* Paper presented at the ER 1981, Washington, D.C.

Wilmot, R. B. (1984). Foreign Keys Decrease Adaptability of Database Designs. *Communications of the ACM, 27*(12), 1237-1243.

OTROS TÍTULOS DE NUESTRA EDITORIAL



Gestión de la información y textos científicos

Jorge Luis León González, Alejandro Rafael Socorro Castro

ISBN: 978-959-257-498-4

Análisis exploratorio de datos con SPSS

Raúl López Fernández (Compilador)

ISBN: 978-959-257-493-9