Universidad Central “Marta Abreu” de Las Villas
Facultad de Matemática – Física – Computación
Departamento de Ciencia de la Computación

Acerca de la dependencia de existencia en la modelación de datos

Tesis presentada en opción al grado científico de
Máster en Ciencia de la Computación

Autor: Norma Elisa Cabrera González

Tutores: Dra. Luisa M. González González
Msc. Carlos E. García González

Santa Clara
2008

Resumen

En la infraestructura de todo sistema de información, la exactitud semántica de los datos es vital para garantizar la calidad de las aplicaciones que los utilizan. La especificación de cuándo y cuáles datos son semánticamente correctos, constituye una de las tareas más importantes en la modelación de datos, donde la lógica del negocio y los requisitos que definen los usuarios son traducidas en restricciones de integridad semánticas. Los modelos de datos proporcionan diferentes medios para representar las restricciones de integridad, en la modelación conceptual la dependencia de existencia es un recurso útil para asegurar la integridad semántica entre las propiedades estáticas y dinámicas de los datos.

En general, las restricciones de integridad son captadas a nivel conceptual y deben ser transformadas adecuadamente para su soporte en el nivel lógico, particularmente la dependencia de existencia queda modelada en los esquemas relacionales a través de las llaves foráneas y su comportamiento. A pesar de la utilidad que puede tener este estudio no solo para la modelación de datos en ambientes centralizados, sino incluso en ambientes distribuidos, en la literatura se describe de forma intuitiva y no está debidamente abordada. En este trabajo se hace un análisis de las distintas construcciones que reflejan de alguna manera dependencia de existencia y se modela su comportamiento sobre la base de sus particularidades. Además, se analiza la aplicación de la dependencia de existencia en asociaciones para la fragmentación horizontal derivada en el marco de bases de datos distribuidas.

Summary

In the infrastructure of any information system, the semantic accuracy of the information is vital to guarantee the quality of the applications that use them. The specification of when and which information is semantically correct, constitute one of the most important tasks in the modeling of information, where the logic of the business and the requisites that the users define are translated in semantic restrictions of integrity. The information models provide different means to represent the integrity restrictions, in the conceptual modeling the existence dependency is a useful resource to assure the semantic integrity between the static and dynamic properties of the information.

In general, the integrity restrictions are received at conceptual level and they must be transformed appropriately for his support in the logical level, particularly the existence dependency is modeled in the schemes related through the foreign keys and his behavior. In spite of the utility that can have this study for the modeling of information in centralized ambiences, but even in distributed ambiences, in the literature it is described of intuitive form and is not tackled properly. In this work is made an analysis of the different constructions that reflect somehow existence dependency and his behavior models itself on the base of his peculiarities. Also, there is analyzed the application of the existence dependency in affiliations for the horizontal fragmentation derived in the frame of distributed databases.

Tabla de contenidos

Introducción

La modelación de la estructura de la información en una aplicación, involucra la identificación de los datos relevantes del universo de discurso, las interrelaciones entre ellos y las restricciones que deben cumplirse. Las propiedades estáticas y dinámicas de los datos se definen a partir de las especificaciones dadas por los usuarios y se expresan a través de modelos de datos. Los modelos de datos se clasifican dependiendo de los tipos de conceptos ofrecidos; los modelos conceptuales o de alto nivel, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios perciben los datos; mientras que los modelos físicos o de bajo nivel, proporcionan conceptos que describen los detalles de cómo se almacenan. Estos últimos están dirigidos al personal informático, no a los usuarios finales. Entre ambos extremos se encuentran los modelos lógicos, cuyos conceptos pueden ser comprendidos por los usuarios aunque no están demasiado alejados de la forma en que los datos se organizan físicamente. Los modelos lógicos ocultan algunos detalles de almacenamiento, pero pueden ser implementados directamente.

La descripción de los datos de un universo de discurso mediante un modelo se le denomina esquema conceptual, lógico o físico dependiendo del nivel de abstracción que corresponda. Como resultado de la modelación conceptual de los datos se obtiene el esquema conceptual que no es modificado a menudo; pequeñas transformaciones pueden traer como consecuencia grandes cambios en las etapas siguientes. Los esquemas lógicos están subordinados al modelo de datos que soporta el Sistema de Gestión de Bases de Datos (SGBD) seleccionado y no contiene detalles de implementación, esta tarea queda definida en el esquema físico, donde se especifican las estructuras de almacenamiento y la organización de los archivos.

En contraste con los esquemas, los datos cambian con mucha frecuencia, dado que se insertan, actualizan y eliminan constantemente; en un momento dado los datos almacenados determinan el estado de la bases de datos. La diferencia entre esquema y estado es importante, cuando se define el diseño de una nueva base de datos, solo se le especifica al SGBD el esquema; en ese momento la base de datos se encuentra en el “estado vacío” sin datos. Cuando se almacenan por primera vez pasa al “estado inicial” y de ahí en lo adelante, siempre que se realice una

operación se tendrá un nuevo estado. El SGBD se encarga, en cierta medida, de garantizar que todos los estados de la base de datos sean válidos, satisfaciendo la estructura y las restricciones especificadas en el esquema.

En la infraestructura de todo sistema de información, la exactitud semántica de los datos es vital para garantizar la calidad de las aplicaciones que los utilizan. La exactitud semántica de los datos, básicamente se refiere al grado en que éstos reflejan correctamente la información del universo de discurso. La especificación de cuándo y cuáles datos son semánticamente correctos, constituye una de las tareas más importantes en el proceso de modelación y diseño de la base de datos (Maier, 1983, Biskup, 1995). En este proceso, la lógica del negocio y los requisitos que definen los usuarios son traducidas en restricciones de integridad semánticas o simplemente restricciones de integridad.

Una restricción de integridad es la representación de una propiedad de los datos en las aplicaciones e imponen exigencias que deben ser cumplidas para garantizar la consistencia de los datos, los cuales, no sólo deben ser almacenados y recuperados sino que los sistemas deben garantizar su integridad semántica. La importancia de especificar y definir las restricciones de integridad fue reconocida desde los años 1970 por varios autores (Hammer and McLeod, 1975, Stonebraker, 1975, Eswaran and Chamberlin, 1975, Brodie, 1978, Codd, 1979), a partir de entonces han sido estudiadas desde diferentes puntos de vistas por (Grefen, 1992, Desloch, 1993, Gertz, 1996, Türker, 1999) y al tema se le han dedicado capítulos en libros de texto de base de datos (Ullman and Widom, 1999, Date, 2000, Elmasri and Navathe, 2003). Otros libros de texto como (Maier, 1983, Abiteboul et al., 1995) se concentran principalmente en los fundamentos teóricos de las restricciones de integridad. Los modelos de datos proporcionan diferentes medios para representar las restricciones e incluso, éstas pueden ser inherentes al modelo (Brodie et al., 1984).

El modelo de datos semántico más utilizado es el Entidad Relación Extendido (ERE), el cual tiene sus bases en el modelo Entidad Relación (ER) propuesto en (Chen, 1976) y es el resultado de muchas investigaciones, que han incorporado durante el devenir de los años nuevos conceptos, con el objetivo de enriquecer la representación de los esquemas

conceptuales. Los conceptos básicos de este modelo son las entidades y las interrelaciones entre entidades; a partir de este núcleo básico se han desarrollado numerosas extensiones, no siempre tratadas de manera consistente en la literatura sobre el tema, como es el concepto de dependencia de existencia, usado frecuentemente de manera informal. Es muy frecuente encontrar que en un universo de discurso, la presencia de ciertas entidades dependa de la existencia de otras en un esquema, lo que se puede comprender intuitivamente pero sin embargo este hecho y sus diferentes manifestaciones tienen implicaciones en los posibles estados válidos de una BD. El concepto dependencia de existencia se asocia intuitivamente a un comportamiento; se considera básicamente que existe dependencia de existencia entre dos entidades, cuando la eliminación de una, nombrada entidad *dominante*, conlleva también la eliminación de la otra, entidad *subordinada*. Explícitamente conceptos como entidades débiles, interrelaciones de identificación y jerarquías (De Miguel and Piattini, 1993, Ferg, 1985, Smith and Smith, 1977) expresan dependencia de existencia entre entidades, en otros conceptos aparece implícitamente y el comportamiento a que se hace referencia no siempre es el mencionado y hay que profundizar en cada construcción o concepto.

El principio de clasificación a través del concepto de composición (Teorey et al., 1986, Ketabchi et al., 1988) es natural e intuitivo, la utilización de la composición en la modelación de datos a menudo reduce la complejidad del esquema conceptual. A pesar de esto el principal problema radica en que su semántica está insuficientemente definida y sujeta a diferentes interpretaciones. La dependencia de existencia captura algunos de los intereses semánticos que usualmente están asociados a los conceptos de agregación (Hammer and McLeod, 1978, Brodie, 1981) y composición, en contraste con tales conceptos la semántica de la dependencia de existencia es precisa y clara.

Aparentemente la interrelación de asociaciones no es un concepto que refleja de forma evidente dependencia de existencia, sin embargo sus restricciones estructurales, dadas por las propiedades de cardinalidad y las restricciones de participación, conocidas también como cardinalidades máxima y mínima, son de interés en este sentido. Las restricciones de participación especifican si la existencia de una entidad depende de que esté relacionada con otra. Muchos autores están de acuerdo en que la restricción de participación total en las

asociaciones implica un cierto tipo de dependencia de existencia (Elmasri and Navathe, 2003). En ocasiones a simple vista no es tan fácil determinar cuando una asociación entre entidades expresa dependencia de existencia y descuidarlo puede acarrear inconsistencia semántica. Por estas razones se considera que el concepto de dependencia de existencia es un recurso útil para asegurar la integridad semántica entre las propiedades estáticas y dinámicas de los datos en la modelación conceptual.

Como puede observarse, la dependencia de existencia en restricciones que de una manera u otra está presente en varios conceptos del modelo ERE y sin embargo se describe de forma intuitiva y no siempre se le presta la importancia que la misma tiene como restricción de integridad que en definitiva representa. En general, las restricciones de integridad son captadas a nivel conceptual y deben ser transformadas adecuadamente para su soporte en el nivel lógico, particularmente la dependencia de existencia queda modelada en un esquema relacional a través de las llaves foráneas y su comportamiento. A pesar de la utilidad que puede tener este estudio no solo para la modelación de datos en ambientes centralizados, sino incluso e ambientes distribuidos, en la literatura no está debidamente abordado.

Por todo lo anterior se formula el siguiente **problema de investigación**: el concepto de dependencia de existencia es tratado de forma incompleta y no uniforme tanto en libros de textos como en artículos sobre modelación conceptual. El **objetivo general** de esta investigación es hacer un análisis de las distintas construcciones que reflejan de alguna manera dependencia de existencia y modelar su comportamiento sobre la base de sus particularidades. Para lograr este objetivo se plantean los siguientes **objetivos específicos** de la investigación:

1. Hacer un estudio de las construcciones de varios de los enfoques más populares del modelo ERE a partir del modelo original de Chen y en especial destacar aquellas construcciones que reflejan dependencia de existencia acorde a cada enfoque.
2. Proponer scripts que modelen el comportamiento de cada tipo de dependencia de existencia y facilitar de este modo su soporte por una herramienta CASE.
3. Analizar la aplicabilidad de la dependencia de existencia en asociaciones para la fragmentación horizontal derivada en el marco de bases de datos distribuidas.

Como resultado de la revisión bibliográfica y de la factibilidad del trabajo en función de los objetivos, se define la siguiente **hipótesis de investigación**: La dependencia de existencia permitirá captar restricciones semánticas del universo de discurso en la modelación conceptual de los datos.

Las **tareas de investigación trazadas** son:

1. Analizar a profundidad el modelo ER y su evolución, especialmente las construcciones que expresan dependencia de existencia.
2. Modelar el comportamiento de los diferentes tipos de dependencia de existencia presentes en el modelo ERE, a través del lenguaje estándar SQL.
3. Proponer un método para la obtención de fragmentos horizontales derivados teniendo en cuenta la presencia de dependencia de existencia.

El **valor científico** del presente trabajo consiste en:

1. Sistematizar el concepto de dependencia de existencia como restricción de integridad y reflejar su comportamiento a través de los recursos que brinda SQL3.
2. Añadir un método para la fragmentación horizontal derivada teniendo en cuenta la dependencia de existencia como criterio adicional cuando haya que decidir entre varias posibles fragmentaciones en que estén involucradas llaves extranjeras.

El **valor práctico** consiste en la obtención de un conjunto de script de ayuda a los desarrolladores de herramientas CASE para la transformación del modelo ERE.

El **valor metodológico** está dado por la descripción de diferentes extensiones al modelo ER, con el propósito de esclarecer la presencia de dependencia de existencia en las mismas.

La tesis está estructurada en tres capítulos:

En el Capítulo 1 se introduce la problemática de la modelación conceptual, como vía para captar los requerimientos de los usuarios. Se analiza el modelo conceptual ER de Chen como base para esclarecer las extensiones al mismo, centrando la atención en aquellas construcciones que expresan dependencia de existencia, por ser un recurso que refleja el contenido semántico de los datos. Además se comparan diferentes notaciones gráficas del modelo. Finalmente se muestra una formalización del concepto de dependencia de existencia presente en asociaciones, así como su representación en diagramas.

En el Capítulo 2 se presentan algunos de los recursos que brinda el lenguaje estándar SQL3 para dar soporte a la representación de las restricciones de integridad, dado que posteriormente la dependencia de existencia será modelada a través de estas facilidades.

En el Capítulo 3 se analiza el comportamiento de los datos modelados en construcciones con dependencia de existencia, al ser transformados a un esquema relacional. Se ofrecen scripts en el lenguaje estándar SQL3 de manera que puedan ser generados de forma automática en herramientas CASE para diseñadores de bases de datos. Además se analiza una aplicación de la dependencia existencia en la fragmentación horizontal derivada.

CAPÍTULO 1. PRESENCIA DE DEPENDENCIA DE EXISTENCIA EN CONSTRUCCIONES DEL MODELO ENTIDAD RELACIÓN EXTENDIDO

Con este capítulo se pretende introducir la problemática de la modelación conceptual, como vía para captar los requerimientos de los usuarios y el contenido semántico de los datos, siendo la dependencia de existencia un recurso valioso para este fin. En el epígrafe 1.2 se abordan los conceptos del modelo ER de Chen y las notaciones originales, sobre el cual se han generado muchas extensiones dando como resultado el modelo ERE, al punto de que no siempre queda claro cuáles construcciones son extensiones y cuáles no. Por lo anterior, en la sección 1.3 presenta algunas de las extensiones que se han incorporado al modelo básico y se hace un análisis de las construcciones que de alguna manera expresan dependencia de existencia. Debido a que no puede lograrse un buen entendimiento de la semántica involucrada en un modelo sin un cuidadoso estudio de la representación sintáctica en sus diagramas, el epígrafe 1.4 esta centrado en las notaciones gráficas de los diagramas ER. Con el epígrafe 1.5 y 1.6 se formaliza el concepto de dependencia de existencia en las asociaciones, así como su representación en diagramas.

1.1 Introducción a la modelación conceptual

Paralelo al avance de la tecnología de bases de datos, se han desarrollado metodologías y técnicas para su diseño. Todas ellas tienen en común por ejemplo, la descomposición del proceso en fases, con objetivos bien definidos y técnicas establecidas para conseguir estos objetivos. A pesar de ello, algunos desarrolladores insisten en pasar por alto este proceso. Batini comenta al respecto:

> *Desafortunadamente, las metodologías de diseño de bases de datos no son muy populares; la mayoría de las organizaciones y de los diseñadores individuales confían muy poco en las metodologías para llevar a cabo el diseño y esto se considera, con*

> *frecuencia, una de las principales causas de fracaso en el desarrollo de los sistemas de información. Debido a la falta de enfoques estructurados para el diseño de bases de datos, a menudo se subestima el tiempo o los recursos necesarios para un proyecto de bases de datos, las bases de datos son inadecuadas o ineficientes en relación a las demandas de la aplicación, la documentación es limitada y el mantenimiento es difícil.* (Batini et al., 1994)

Muchos de los problemas son consecuencia de falta de claridad en el diseño de la base de datos, que no permite entender con exactitud la naturaleza de los datos a un nivel conceptual y abstracto. Con frecuencia los datos se describen desde el inicio del proyecto en términos de las estructuras finales de almacenamiento, no se valora que el comprender las propiedades estructurales y semánticas de los datos es independiente de los detalles de la implementación.

La modelación conceptual, es la etapa en que se crea un esquema conceptual que describe de manera concisa, los requerimientos de información de los usuarios. Los datos se expresan mediante los conceptos de un modelo de alto nivel, independientemente del SGBD donde finalmente se almacenará la información.

Un modelo de datos según (Brodie et al., 1984) es una colección de conceptos bien definidos matemáticamente, que ayudan a considerar y expresar las propiedades estáticas y dinámicas de una aplicación. Aunque la mayoría de los modelos de datos han evolucionado intuitivamente y no han sido formalmente definidos, se asume generalmente que de una aplicación deben caracterizar:

- propiedades estáticas tales como objetos, sus propiedades (llamadas con frecuencia atributos) e interrelaciones entre objetos (consideradas en ocasiones como propiedades especiales de los objetos).
- propiedades dinámicas, tales como operaciones sobre los objetos, propiedades de las operaciones e interrelaciones entre las mismas.
- reglas de integridad sobre los objetos que reflejen un estado válido de la base de datos.

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 como el concepto más importante en la modelación, el de entidad, denominado indistintamente entidad u objeto (Teorey and Fry, 1980). El modelo para una entidad es construido a partir de un conjunto relativamente pequeño de primitivas de modelación.

La representación de los datos a través de conceptos que no incluyen detalles del 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 satisfacen los requerimientos de los usuarios y no hay contradicciones entre dichos requerimientos. Utilizando este enfoque los diseñadores de bases de datos sólo deben concentrarse en especificar las propiedades de los datos, sin tener en cuenta detalles de implementación y almacenamiento. Construyendo los esquemas conceptuales, los diseñadores descubren el significado o semántica de los datos, por tal razón estos modelos también son llamados modelos de datos semánticos.

Para (Elmasri and Navathe, 1997) es importante que el modelo de datos con el que se describa un esquema conceptual tenga las siguientes características:

- Expresividad: El modelo de datos debe ser lo bastante expresivo para distinguir los diferentes tipos de datos, interrelaciones y restricciones.
- Sencillez: El modelo debe ser lo bastante simple para que la generalidad de los usuarios no especialistas comprendan y usen sus conceptos.
- Minimalidad: El modelo debe tener un número pequeño de conceptos básicos cuyo significado sea distinto y no se solapen.
- Representación gráfica: El modelo debe contar con una representación gráfica para mostrar un diagrama del esquema conceptual que sea fácil de interpretar.

- Formalidad: Un esquema conceptual expresado en el modelo de datos debe representar una especificación formal de los datos sin ambigüedad. Por tanto, los conceptos del modelo deben definirse con exactitud y sin que haya posibilidad de confusión.

En la práctica pueden aparecer conflictos entre tales requerimientos, particularmente entre el primero de ellos con el resto. El modelo de datos semántico más popular es el ER, según describe el autor, fue concebido inicialmente unificando las facilidades del modelo de red, el relacional y el de conjunto de entidades. Consta el modelo ER de un conjunto de primitivas básicas, a través de las cuales se pueden modelar la mayoría de las características de las bases de datos. A través de los años, varios investigadores han incorporado nuevos conceptos con el objetivo de enriquecer la representación de los esquemas conceptuales. En el siguiente epígrafe se presenta el modelo ER, tal como fue definido por su autor en (Chen, 1976) y luego se abordan algunas extensiones al modelo.

1.2 Modelo Entidad Relación de Chen

El modelo básico ER propuesto por Chen, está basado en las teorías de relaciones y conjuntos. Como objetivo inicial se planteó, unificar diferentes vistas de datos en un universo de discurso, a la vez que proporcionar mayor información semántica que los modelos precedentes; sin perder el alto grado de independencia de datos alcanzado por el modelo relacional. El universo de discurso se modela como un conjunto de entidades e interrelaciones entre dichas entidades. La filosofía subyacente es que las propiedades o atributos son solamente simples hechos acerca de una entidad, mientras que una interrelación puede modelar la construcción de hechos más complejas a partir de otras entidades. En los siguientes subepígrafes se presentan los conceptos de este modelo, así como las notaciones gráficas para su representación mediante diagramas.

1.2.1 Entidades y conjunto de entidades

Las entidades son ¨ algo ¨ relevante del universo de discurso que puedan ser unívocamente identificadas y sobre las cuáles se colecciona información. Una entidad puede existir

físicamente como cierta persona, casa, automóvil o simplemente ser definida conceptualmente, como un departamento, compañía o curso académico específico.

Las entidades son agrupadas en diferentes conjuntos de entidades, como EMPLEADO, PROJECTO o DEPARTAMENTO. Una entidad pertenece a un conjunto de entidades si satisface el predicado que caracteriza a ese conjunto. Matemáticamente, se define: $E = \{e: p(e)\}$, siendo e una entidad del conjunto de entidades E y p el predicado asociado a E. Por ejemplo, si sabemos que una entidad pertenece al conjunto EMPLEADO, entonces sabemos tiene las propiedades comunes a las otras entidades de dicho conjunto. Los conjuntos de entidades pueden no ser mutuamente excluyentes. Para ilustrarlo: una entidad que pertenece al conjunto MUJER también pertenece a PERSONA. En este caso, MUJER es un subconjunto de PERSONA. En (Chen, 1976) aún cuando se reconoce tal situación, no se define una construcción para representarla; en extensiones al modelo queda reflejada a través de jerarquías, la cual se aborda en el subepígrafe 1.3.2.

1.2.2 Interrelación, rol y conjunto de interrelaciones

Las interrelaciones son conexiones que existen entre dos o más entidades. Las entidades relacionadas pueden pertenecer al mismo o a distintos conjuntos de entidades. Una interrelación puede ser vista como una relación matemática entre n entidades, cada una tomada de un conjunto de entidades $\{E_1, E_2,\ldots, E_n\}$ no necesariamente diferentes, donde cada entidad e_i es una instancia del conjunto de entidades E_i. La tupla de la forma $[e_1, e_2,\ldots, e_n]$ es una interrelación y una colección de ellas es denominada conjunto de interrelaciones. Por ejemplo, MATRIMONIO es un conjunto de interrelación entre entidades del conjunto PERSONA. Para simplificar la descripción del modelo en los esquemas y teniendo en cuenta que a nivel conceptual se trabaja con abstracciones generales, en lo adelante se nombran a los conjuntos de entidades como entidad y a los conjuntos de interrelaciones como interrelación, a menos que sea imprescindible hacer la distinción para evitar ambigüedades.

El rol de una entidad en una interrelación es el papel o función que desempeña en ella. La definición de roles permite atribuirle a las entidades su semántica, aportándole mayor

expresividad al esquema y permitiendo disminuir ambigüedades. Esto cobra mayor importancia en aquellas interrelaciones que involucran a una misma entidad más de una vez. Por ejemplo, los roles ¨ esposo ¨ y ¨ esposa ¨ en la interrelación MATRIMONIO que vincula entidades PERSONA. Si se definen los roles de las entidades, se declaran explícitamente de la siguiente manera: [$r_1/e1$, r_2/e_2, ..., r_n/e_n], donde r_i es el rol de la entidad e_i en la interrelación.

1.2.3 Atributo, valor y dominio de valores

Las entidades y las interrelaciones se describen a través de conjuntos de pares (atributo, valor) que representan sus propiedades o características relevantes. Los valores son clasificados en conjuntos de valores a los cuales se asocia un predicado que determina si un valor pertenece o no al conjunto.

Un atributo se define formalmente como una función que le hace corresponder a una entidad o interrelación, un dominio de valores o el producto cartesiano de conjuntos de valores: $f: E_i \text{ o } R_i \rightarrow V_i \text{ o } V_{i1} \text{ x... x } V_n$. Como puede observarse los atributos pueden tener disímiles características, aunque (Chen, 1976) no formaliza una clasificación para los diferentes tipos de atributos, quedan comprendidos en la definición los atributos simples o compuestos, atómicos o multivaluados.

El valor de un atributo puede ser utilizado para diferenciar las entidades dentro de un conjunto de entidades del mismo tipo, como No_Empleado para el conjunto de entidades EMPLEADO si cada empleado tiene asignado un número diferente. En ocasiones se necesita un conjunto de atributos para identificar las entidades. Básicamente un atributo clave o llave es un grupo de atributos que permiten identificar de manera única a cada entidad del conjunto de entidades. Si este conjunto de atributos no se encuentra en el universo de discurso, entonces puede definirse un atributo artificial asociado a un conjunto de valores que posibilite la distinción entre las entidades. En ocasiones se encuentran más de un grupo de atributos que permiten identificar las entidades, todos ellos pueden ser llaves alternativas. Usualmente se selecciona como la llave primaria aquella que semánticamente tenga un significado más natural o resulte cómoda a los usuarios.

1.2.4 Entidades débiles

En ciertos casos las entidades pueden no ser identificadas por los valores de sus propios atributos. Por la semántica del problema, debe usarse una interrelación para identificarlas. Por ejemplo, considere en una empresa donde los empleados en adiestramiento no son plantilla de la empresa y dependen de los empleados a quienes están subordinados (figura 1.1). Las entidades de ADIESTRADO son identificadas por sus nombres y por la llave primaria de los empleados que los apoyan, es decir, por su asociación con las entidades de EMPLEADO. En casos como este, el atributo No_Empleado de EMPLEADO junto al atributo Nombre_Adiestrado permite identificarlos.

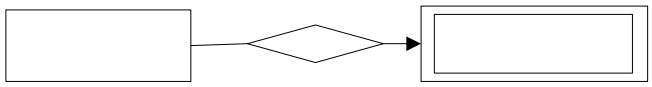


Figura 1.1 *Ejemplo de identificación de entidades a través de asociaciones con otras entidades (Chen, 76)*

Teóricamente, cualquier interrelación puede ser usada para identificar entidades. Por simplicidad, (Chen, 1976) restringe el uso sólo a las interrelaciones binarias 1:N, donde las entidades del lado N de la interrelación dependen de la existencia de una entidad al otro lado. Por ejemplo, un empleado puede tener varios dependientes y la existencia de los dependientes depende de la existencia del empleado correspondiente. Este método de la identificación de entidades por medio de interrelaciones con otras entidades puede ser aplicado recurrentemente hasta que las entidades que puedan ser identificadas por sus propios valores de atributo sean alcanzadas. Por ejemplo, la llave primaria de un departamento en una compañía puede consistir en el número de departamento y la llave primaria de la división, que por su parte consiste en el número de división y el nombre de la compañía.

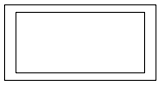
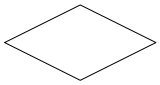
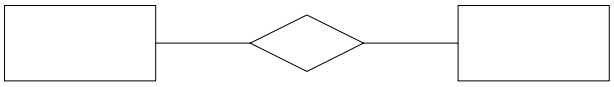
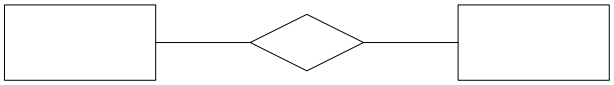
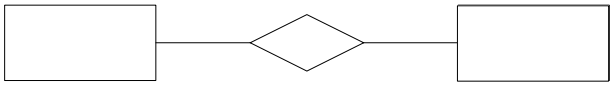
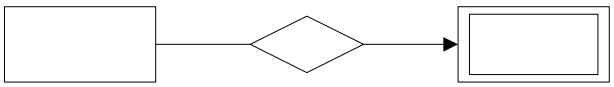
Por lo tanto, (Chen, 1976) plantea dos formas de interrelaciones entre entidad. Si la interrelación es usada para identificar un conjunto de entidades, la llama interrelación de entidad débil. En caso contrario la llama interrelación de entidad regular o fuerte. Del mismo

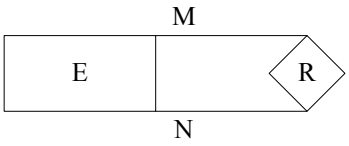
EMPLEADO E

modo, propone dos formas de clasificar las entidades. Si todas las entidades en la interrelación son identificadas por sus propios valores de atributo, las llama entidad regular. Por otro lado, si algunas entidades son identificadas por otras entidades, las llama entidad débil.

1.2.5 Notaciones gráficas para diagramas ER

En el modelo ER original, las entidades son representadas por un rectángulo con el nombre de la entidad y las interrelaciones por un rombo con el nombre de la interrelación. Las entidades interrelacionadas están conectadas al rombo a través de líneas rectas, cada línea es etiquetada con un "1", "N" o "M" para indicar interrelaciones del tipo *1:1*, *1:N* o *M:N*. La notación gráfica para un conjunto de entidades débiles definida en (Chen, 76) es un rectángulo de doble línea, conectado a través de una flecha que apunta hacia la entidad débil. Sin embargo, en extensiones al modelo lo común es encontrarlo conectado a través de una flecha que parte desde la entidad débil.

Símbolos	**Significado**
	Conjunto de entidades *E*
	Conjunto de entidades débiles *Ed*
	Conjunto de interrelaciones *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*
	*Ed* (entidad débil), identificada por *R*(interrelación de entidad débil) y *E* entidad regular



R interrelación unaria

Figura 1.2 *Resumen de la notación de diagramas ER propuesta por Chen*

1.3 Modelo Entidad Relación Extendido

Con los conceptos básicos del modelo ER se pueden modelar la mayoría de las características de las bases de datos, sin embargo no abarcan completamente los requerimientos de la modelación, principalmente para aplicaciones complejas. Al mismo le faltan subestructuras para entidades e interrelaciones, lo cual es aclamado por investigadores que desarrollan extensiones del modelo. En los siguientes subepígrafes se presenta algunas de las definiciones sobre los conceptos básicos del modelo que no fueron definidas en (Chen, 1976), así como nuevas extensiones que se han incorporado.

1.3.1 Extensiones a las construcciones básico del modelo de Chen

Las entidades débiles, según las define Chen, son aquellas que no poseen atributos suficientes para ser unívocamente identificadas o las que carecen de sentido su identificación independiente, en el universo de discurso, sin la asociación con otras entidades. Aunque Chen no hace diferencia; para (De Miguel and Piattini, 1993) resulta interesante distinguir dentro de las asociaciones débiles con dependencia de existencia y las asociaciones débiles con dependencia de identificación. Para (Ferg, 1985), un conjunto de entidades débiles tiene dependencia de identificación cuando las entidades no pueden ser identificadas por los valores de sus propios atributos y tienen que serlo por la asociación con otras entidades. Obviamente una dependencia de identificación es siempre una dependencia en existencia, no ocurre así lo contrario.

Para (Elmasri and Navathe, 2003) sólo considera las asociaciones débiles cuando poseen dependencia de identificación y nombra a la entidad dominante propietario identificador. Estos conjuntos de entidades débiles siempre tienen una restricción de participación total respecto

a la asociación de identificación, dado que una entidad débil no puede ser identificada sin su correspondiente entidad propietario. Representativo de esta situación es el ejemplo de un almacén (figura 1.3) donde se guarda la información de los pedidos realizados al mismo. A cada pedido se le asocia las líneas de pedido. En este ejemplo si se pregunta por el producto que se solicita en la línea 20, hay ambigüedad a menos que se pregunte por la línea 20 del pedido 3045.

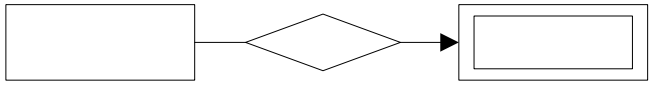


Figura 1.3 *Diagrama para dependencia de identificación, según notación (De Miguel and Piattini, 1993)*

Una entidad débil queda definida siempre a través de una interrelación que implica que su existencia depende de otra entidad de orden superior (que puede ser una entidad fuerte o débil). Por tanto, toda entidad débil tiene una dependencia de existencia de la entidad de orden superior, definiéndose entre ellas una jerarquía de dos niveles. Para existir la entidad débil, debe existir previamente la entidad de orden superior y si desaparece esta última, entonces deben desaparecer todas las entidades débiles que están asociadas a la misma. En caso de que la existencia sea sólo de existencia y no de identificación, por ejemplo en una empresa donde los datos sobre los familiares de los empleados sólo tendrán sentido mientras estos trabajen en la empresa, ver figura 1.4:

PEDIDO

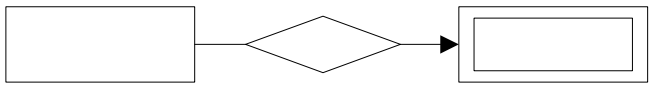


Figura 1.4 *Diagrama para dependencia de existencia, según notación (De Miguel and Piattini, 1993)*

Las interrelaciones, en el sentido ya descrito, también son nombradas asociaciones. Este concepto es introducido por (Peckham and Maryanski, 1988) y (Blaha et al., 1988) como una forma de abstracción en que una interrelación entre entidades con igual nivel de abstracción es considerada como un conjunto de mayor nivel. Los detalles de las entidades interrelacionadas se suprimen y se enfatizan los del conjunto obtenido. El grado de una asociación está dado por el número de entidades asociados en la misma. Las interrelaciones unarias, binarias y ternarias tienes grado 1, 2 y 3 respectivamente.

Las asociaciones son distinguidas por el número de posibles relaciones que una entidad determinada puede tener sobre esta. Este rango es llamado cardinalidad, Chen sólo utilizaba las restricciones de cardinalidad máxima, la cual se define como el máximo número de instancias de una entidad que pueden participar en una interrelación. Para una interrelación binaria hay cuatro opciones: "uno-a-uno", "uno-a-muchos", "muchos-a-uno", o "muchos-a-muchos". (Scheuermann and Scheffner, 1980) introduce al modelo el concepto de restricciones de participación y define la cardinalidad mínima como el número mínimo de instancias de una entidad que pueden participar en la interrelación. Los valores más usados para la cardinalidad mínima son 0 y 1, con los correspondientes términos "participación opcional" y "participación obligatoria", aunque en ocasiones pueden ser usados valores mayores. Las restricciones de participación obligatoria o total especifican si la existencia de una entidad depende de que esté relacionada con otra entidad, para (Elmasri and Navathe, 2003) expresa cierto grado de dependencia de existencia, en el capítulo tres se retoma esta idea.

1.3.2 Especialización y generalización

Una debilidad particular del modelo ER original, propuesto por Chen fue la ausencia del concepto de subtipo y generalizaciones de tales subtipos. Smith en (Smith and Smith, 1977) introducen el término generalización en su modelo semántico jerárquico y de este modo provee una base formal para los subtipos dentro de un modelo de datos abstracto. Esas ideas han sido extendidas en (Scheuermann and Scheffner, 1980), (Navathe and Cheng, 1983) y otros.

Un conjunto de entidades puede incluir subconjuntos de entidades que se diferencian de alguna forma de las otras entidades del conjunto. Por ejemplo, un subconjunto de entidades puede tener atributos que no son compartidos por todas las restantes entidades del conjunto. El proceso de designación de subgrupos dentro de un conjunto de entidades es la especialización. Un conjunto de entidades se puede especializar mediante más de una característica distintiva.

Muchas veces las especializaciones son representadas a través de interrelaciones *ISA* (Brachman, 1983). Las interrelaciones *ISA* son un tipo especial de interrelaciones semánticas, donde si ***A*** *ISA* ***B***, entonces ***B*** es un concepto más general y ***A*** es el concepto específico. La propiedad más significativa de una interrelación *ISA* es la herencia. Todo lo que sea válido para el concepto genérico es válido para el concepto específico, esto es, que todos los atributos, sus valores y restricciones son heredados desde el nivel más genérico hasta el nivel específico. Implícitamente esta construcción modela dependencia de existencia del concepto específico hacia el general.

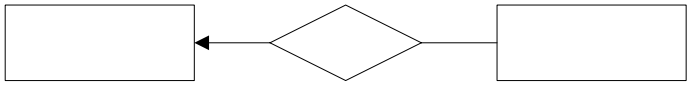


Figura 1.5 *Ejemplo de una especialización*

El concepto de generalización, también es tratado en (Peckham and Maryanski, 1988), (Blaha et al., 1988, Rumbaugh, 1987) como una abstracción en que entidades similares son relacionados con un conjunto genérico de mayor nivel. Las entidades constituyentes son consideradas especializaciones del conjunto genérico. Al nivel de conjunto genérico las diferencias de las especializaciones se suprimen, mientras que se enfatizan las similaridades. Toda instancia de un conjunto genérico puede considerarse como una instancia de uno o más entidades especializadas. Esto establece una interrelación *ISA* entre entidades y permite minimizar almacenamiento al permitir que propiedades asociadas con conjuntos de entidades genéricos sean heredadas por las especializadas. El mecanismo de generalización puede ser base para construir modelos conceptuales de forma que se modele primero en términos de clases y después en términos de subclases. La jerarquía de generalización ayuda al diseñador a organizar el proceso de acumulación de detalles y su integración en un sistema informativo coherente. Matemáticamente se define como: una entidad *E* es una generalización de un grupo de entidades $E_1, E_2, \ldots, E_n$ si cada elemento de las entidades $E_1, E_2, \ldots, E_n$ es también un elemento de la entidad *E*.

Propiedad de cubrimiento de una generalización

Cubrimiento Total o Parcial. El cubrimiento de una generalización es total (t) si cada elemento de la entidad genérica es transformada en al menos un elemento de las entidades de nivel más bajo o específicos; es parcial (p) si existe algún elemento de la entidad genérica que no es transformado en algún elemento de las entidades específicas.

Cubrimiento Exclusivo o Solapado. El cubrimiento de una generalización es exclusivo (e) si cada elemento de la entidad genérica es transformado a lo sumo en un elemento de las entidades específicas; es solapado (s) si existe algún elemento de la entidad genérica que es transformado en elementos de dos o más entidades específicas diferentes.

Los siguientes ejemplos ilustran las formas en que son combinados los distintos cubrimientos de la generalización.

- El cubrimiento de la generalización Persona de las entidades Masculino y Femenino es total y exclusivo (t, e).
- El cubrimiento de la generalización Persona de las entidades Masculino y Empleado es parcial y solapado (p, s).
- El cubrimiento de la generalización Vehículo de las entidades Bicicleta y Auto es parcial y exclusivo (p, e).
- El cubrimiento de la generalización Deportista de las entidades Futbolista y Tenista en una escuela que requiere que cada alumno participe al menos en uno de estos deportes es total y solapada (t, s).

Generalización *versus* Especialización

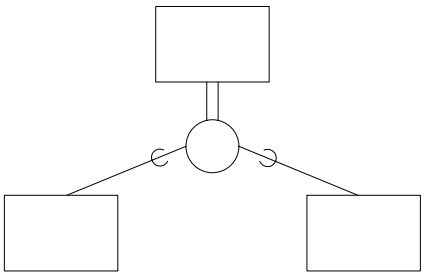
Para todos los propósitos prácticos, la generalización es una inversión simple de la especialización. En términos de diagrama ER no se distingue entre especialización y generalización.

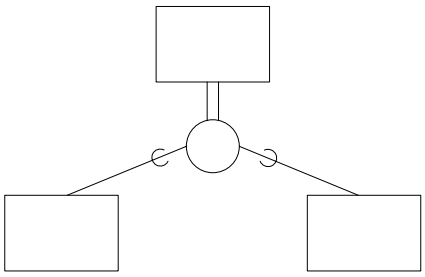
La especialización parte de un conjunto de entidades simples, enfatiza la diferencia entre las entidades dentro del conjunto mediante la creación de distintos conjuntos de entidades de nivel más bajo. Estos conjuntos de entidades de nivel más bajo pueden tener atributos, o pueden

participar en interrelaciones, que no se aplican a todas las entidades del conjunto de nivel más alto. De hecho, la razón de que el diseñador aplique la especialización es representar tales características.

La generalización procede del reconocimiento de un número de conjuntos de entidades que compartan algunas características comunes (se describen mediante el mismo conjunto de atributos y participan en las mismas interrelaciones). Basada en sus similitudes, la generalización sintetiza estos conjuntos de entidades en uno solo, el conjunto de entidades de nivel más alto. La generalización se usa para resaltar las similitudes entre los conjuntos de entidades de nivel más bajo y para ocultar las diferencias, también permite economizar la representación para que los atributos compartidos no estén repetidos.

La siguiente figura muestra cómo representa (Elmasri and Navathe, 2003) la especialización/generalización en los diagramas ER. Las entidades que definen una especialización se conectan a través de líneas a un círculo, que a su vez se conecta a la entidad generalizada con una línea simple o doble si el cubrimiento es parcial o total respectivamente. El símbolo de subconjunto sobre la línea que conecta una especialización al círculo indica la dirección de la generalización/especialización. Una d (disjoint) dentro del círculo significa cubrimiento exclusivo y una o (overlapping) cubrimiento solapado. Cuando la especialización consiste en solo una entidad no se usa en la notación el círculo, sino como se muestra en la figura 1.7.

a) 

b) 

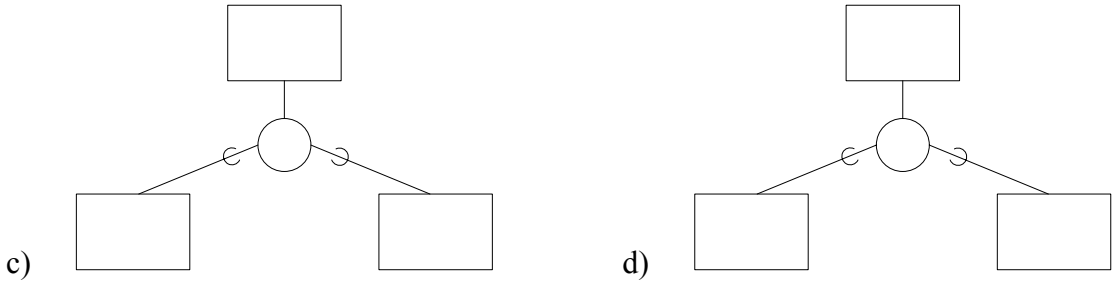


Figura 1.6 *Notación gráfica para la generalización-especialización según (Elmasri, 2003)*
a)cubrimiento total y exclusivo b)cubrimiento total y solapado c)cubrimiento parcial y exclusivo b)cubrimiento parcial y solapado

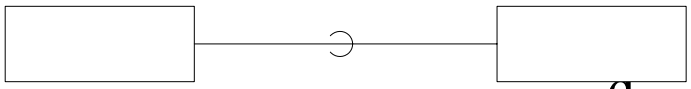


Figura 1.7 *Notación gráfica para especialización en solo una entidad según (Elmasri, 2003)*

1.3.3 Agregaciones

Esta noción de modelación fue introducida en (Hammer and McLeod, 1978) y es tratada en (Codd, 1979, Brodie, 1981, Mylopoulos et al., 1980). Una agregación en su forma más simple, es un conjunto al que se desea asociar varias propiedades, como por ejemplo un empleado puede ser un agregado de los componentes número de identidad, nombre y dirección. En su forma más compleja, según (Peckham and Maryanski, 1988), (Blaha et al., 1988) es una abstracción en que una interrelación entre entidades se considera como una entidad agregada de mayor nivel. Cuando consideramos una entidad agregada se suprimen detalles específicos de las entidades constituyentes. Toda entidad agregada puede ser descompuesta en entidades componentes, lo que establece una interrelación *member-of* entre entidades (Brodie, 1981).

Una limitación del modelo ER básico es que no facilita expresar interrelaciones entre interrelaciones. Una interrelación y las entidades que relaciona, pueden ser manejados como una entidad en un nivel de abstracción mayor, lo que posibilita que se pueda asociar con otras entidades. Este mecanismo es conocido como estructura de agregación o agregación de entidades y permite representar la interrelación *member-of*. La agregación se representa en los diagramas ER como un rectángulo englobando a la interrelación que la conforma.

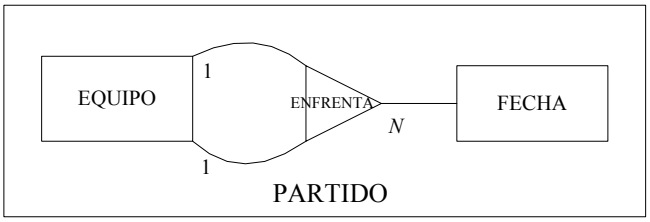


Figura 1.8 *Ejemplo de una agregación*

Al definir la entidad agregada PARTIDO se facilita la modelación de otros hechos como por ejemplo: los jugadores que participaron en un partido dado, cuál es el equipo ganador de un partido, entre otros. Si se intenta modelar la problemática de la figura 1.8 sin el uso de la agregación, habría que utilizar una interrelación de grado mayor con un oscurecimiento en su semántica. En el ejemplo anterior se considera oportuno modelar FECHA como conjunto de entidades con propósitos didácticos, aunque no es imprescindible dado que fecha puede ser un atributo de la interrelación.

1.3.4 Composición

La composición es un caso particular de la agregación, según (Ketabchi et al., 1988) es una abstracción en que una entidad es interrelacionada con sus partes componentes, estableciendo una interrelación *part-of* entre entidades (Teorey et al., 1986). Esta forma de abstracción aparece con las aplicaciones ingenieriles al posibilitar el ensamble de entidades compuestas como parte de la base de datos. La composición de entidades puede ser vista en dos niveles de abstracción: un nivel de caja negra donde los detalles del ensamble se suprimen y un nivel de caja transparente donde se exponen estos detalles. Según (Rumbaugh et al., 1991) la existencia de las entidades componentes pueden depender, en ciertos casos, de la existencia de la entidad agregada de la cual son parte. Por tanto la composición, en ocasiones modela dependencia de existencia de las partes respecto al todo.

1.4 Notaciones gráficas para los modelos básicos

La mayoría de los autores analizados en este trabajo usan en sus diagramas ER un rectángulo para representar las entidades y un rombo para las asociaciones. Algunos, pero no todos,

incluyen atributos en el diagrama, y los que lo hacen usan un óvalo con el nombre (los identificadores subrayados), un círculo pequeño con el nombre al lado (los identificadores tienen un círculo ennegrecido) o una línea con el nombre al lado (el identificador no se indica). Para la cardinalidad se usa la notación de 1:1, 1:N, M:N de Chen o la notación (min-card, max-card) discutida en (Batini et al., 1992). (Elmasri and Navathe, 2003) para mostrar una participación obligatoria usan una línea doble de conexión entre una entidad y una asociación. Otros autores ennegrecen la mitad del rombo de la asociación adyacente a la entidad cuya conectividad es "muchos" (si es "uno" no se ennegrece), y ponen un círculo pequeño en la línea cerca del rombo si la interrelación es opcional (si es obligatoria se usa una línea sin círculo).

Wertz en (Wertz, 1993) discute otros estilos de notaciones gráficas. Uno de estos, atribuido a Clive Finkelstein, no usa el rombo y pone un símbolo de pata de gallo en la línea de conexión en el extremo de "muchos" y también usa una barra vertical (a través de la línea de conexión) para indicar una participación obligatoria y un círculo abierto para una participación opcional. Otro enfoque, atribuido a Charles Bachman, tampoco usa el rombo y sí una flecha en el extremo "muchos" de la asociación, para indicar una participación opcional usa un pequeño círculo abierto en la unión del rectángulo de la entidad y la línea de conexión; y un círculo negro para una participación obligatoria. En cuanto el modelo de Chen, (Wertz, 1993) también dice, "variantes menores incluyen el uso de una flecha para indicar el lado "uno" de la interrelación y una flecha doble para indicar el lado "muchos"; un punto para indicar el lado "muchos" de una interrelación y una caja doble para indicar una entidad débil".

Las extensiones al modelo ER han sido numerosas y abordan aspectos muy concretos, razón por la cual se hace difícil una valoración con el propósito de mostrar la sistematización que se persigue en este trabajo. Situación similar presenta la nomenclatura de las construcciones del modelo. En el desarrollo de autores clásicos de libros de textos en el área de las bases de datos, implícitamente se aprecia la tendencia a lograr una sistematización del modelo. Representativos de esta tendencia son los modelos Entidad Relación Extendido (ERE) expuesto por C. J. Date en (Date, 2000), Jeffrey D. Ullman en (Ullman and Widom, 1999) o Teorey en (Teorey, 1999).

Notaciones de C. J. Date

Notaciones de Jeffrey D. Ullman

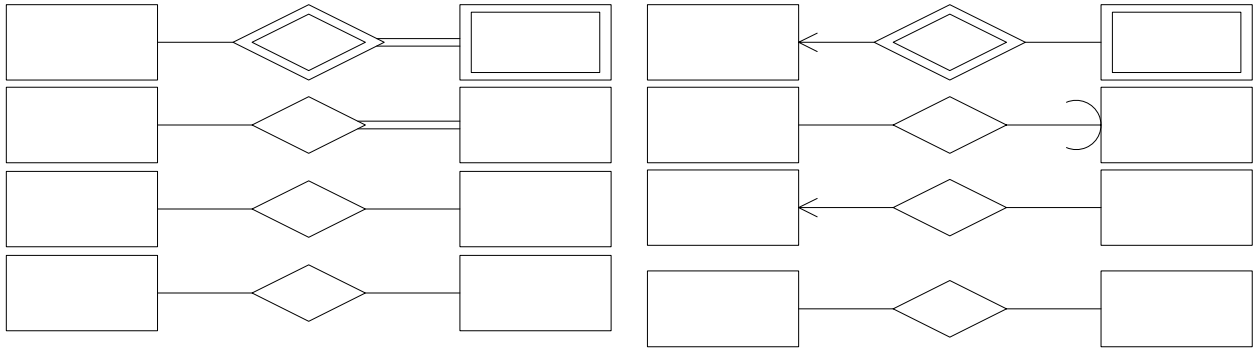
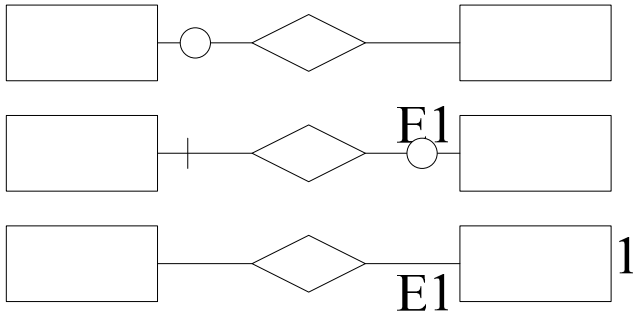


Figura 1.9 *Comparación entre notaciones gráficas de (Date, 2000) y (Ullman and Widom, 1999)*



Participación opcional de *E1* en *R*

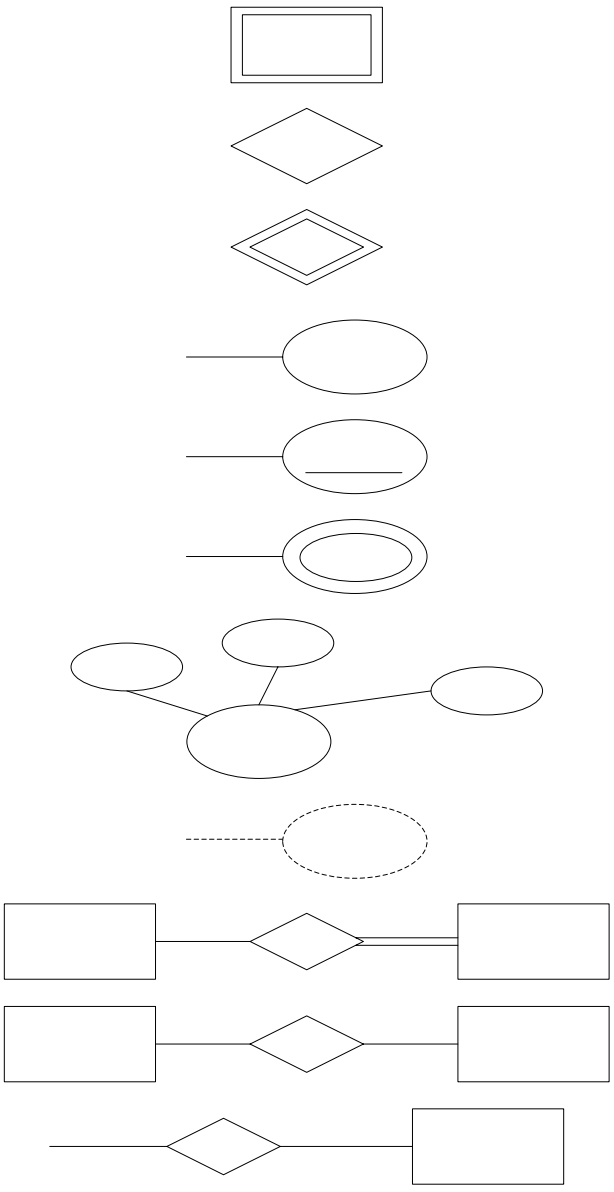
Participación total de *E1* en *R*

Desconocida

Figura 1.10 *Notación gráfica para asociaciones de(Teorey, 1999)*

Como se ha podido apreciar, durante el devenir de los años los investigadores han ido desarrollando el modelo ER, enriqueciéndolo con nuevos conceptos y el resultado está en que actualmente se dispone de una amplia variedad de construcciones. Aun cuando se utilizan las mismas estructuras, no todos los autores las nombran de igual forma, como tampoco existe una manera única de representar los diagramas. En (Teorey, 1999) se puede encontrar una muestra de la diversidad de notaciones para el modelo ERE, así como comparaciones entre ellas. En lo siguiente, este trabajo adopta la nomenclatura y notación de (Elmasri and Navathe, 2003).

Símbolo	**Significado**
	Conjunto de entidades *E*



Conjunto de entidades débiles *Ed*

Conjunto de interrelaciones *R*

Conjunto de interrelaciones de identificación

Atributo

Atributo identificador (llave)

Atributo multivaluado

Atributo compuesto

Atributo derivado

Participación total de E_2 en *R*

Razón de cardinalidad 1:*N* para E_1: E_2 en *R*

Restricción de cardinalidad (mín, máx) de la participación de *E* en *R*

Figura 1.11 *Resumen de la notación gráfica para diagramas ER según (Elmasri, 2003)*

Debe notarse que la representación de las interrelaciones con cardinalidad (min, max) es una alternativa que fusiona las representaciones anteriores.

1.5 Características estructurales de la dependencia de existencia en asociaciones

Las restricciones de participación en las asociaciones implican un cierto tipo de dependencia de existencia, pero a simple vista no es tan fácil determinar cuando una asociación entre entidades expresa dependencia de existencia y descuidarlo puede acarrear inconsistencia semántica. Tomando como base lo planteado en (Snoeck and Dedene, 1998), formalmente puede ser definida en el modelo ERE como:

Dados E_1 y E_2 conjuntos de entidades. Decimos que E_1 tiene dependencia de existencia de E_2, 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 e_1 entidad dependiente o subordinada (E_1 conjunto de entidades dependientes) y e_2 entidad dominante (E_2 conjunto de entidades dominantes).

Otra manera, algo informal, de definir dependencia de existencia (Snoeck and Dedene, 1998) puede ser: Si cada entidad del conjunto de entidades E_1 siempre está asociada como mínimo con una, a lo sumo con una y siempre la misma entidad del conjunto de entidades E_2, entonces E_1 tiene dependencia de existencia de E_2.

Como consecuencia, si E_1 tiene dependencia de existencia de E_2 ($E_1 \leftarrow E_2$), no puede insertarse una entidad e_1 en el conjunto de entidades E_1 a menos que se asocie con una entidad e_2 del conjunto de entidades E_2. Análogamente al eliminar una entidad dominante de E_2, deben eliminarse todas las entidades de E_1 asociadas a ella.

La dependencia de existencia es similar al concepto de dependencia de asociación descrita en (Kilov and Ross, 1994). La principal diferencia es cuando una entidad e_1 depende de la existencia de dos entidades e_2 y e_3, estas entidades no tienen que pertenecer necesariamente al mismo conjunto. Por ejemplo, el control de los préstamos en la biblioteca (figura 1.12), donde la existencia de una entidad *préstamo*, donde se controla la información de las planillas de préstamos en la biblioteca, depende de la existencia de un *ejemplar* disponible y el *miembro* que lo solicita.

Figura 1.12 *Diagrama para el control de los préstamos en una biblioteca*

1.6 Representación gráfica para la dependencia de existencia en asociaciones

La representación de la dependencia de existencia a través de un grafo es sintácticamente correcta. Un Grafo de Dependencia de Existencia (GDE), se define siguiendo la idea de (Snoeck and Dedene, 1998) como:

Sea *G(V, A)* un gafo dirigido donde *V* es el conjunto de los vértices que representan los conjuntos de entidades en el esquema conceptual, *A* las asociaciones que definen dependencia de existencia y *G* debe satisfacer las siguientes restricciones:

1. No hay presencia de bucles, lo que significa que una entidad no tiene dependencia de existencia de sí misma
2. No hay presencia de ciclos

La primera de las restricciones se justifica por el hecho de: si tenemos un conjunto de entidades E_1 que tiene dependencia de existencia de sí mismo, implicaría que una entidad e_1 del conjunto E_1 depende de la existencia de otra entidad e_2 del mismo conjunto; en otras palabras no puede insertarse o crearse e_1 si no se asocia o vincula a e_2, quien a su vez depende de la existencia de otra entidad del mismo conjunto. Obviamente ante tal situación E_1 será un conjunto de entidades vacío. La segunda restricción, responde a un hecho similar, pero en este caso cuando hay más de un conjunto de entidad involucrado.

Retomando el ejemplo anterior, para los conjuntos de entidades que guardan la información sobre los *miembros* y *préstamos* de la biblioteca, la representación del grafo de dependencia de existencia sería: *G* ({MIEMBRO, PRESTAMO}, {(MIEMBRO, PRESTAMO)}).

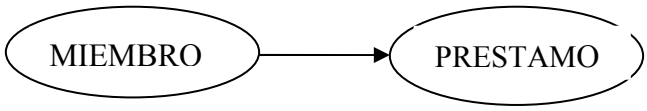


Figura 1.13 *Grafo de dependencia de existencia entre los conjuntos de entidades MIEMBRO y PRESTAMO*

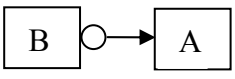
En el grafo de dependencia de existencia también se puede expresar la cardinalidad de la asociación, lo cual es defino en (Snoeck and Dedene, 1998) como la cantidad de entidades subordinadas asociadas a una entidad dominante en un momento dado.

Notación para la cardinalidad de la asociación con dependencia de existencia:

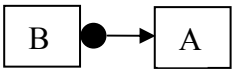
- E_1 (1) ← E_2, si E_1 ← E_2 y una entidad e_2 de E_2 tiene asociado a ella a lo sumo una entidad e_1 de E_1 en un momento dado
- E_1 (N) ← E_2, si E_1 ← E_2 y una entidad e_2 de E_2 puede tener asociado a ella varias entidades e_1 de E_1 en un momento dado

Debe notarse que las asociaciones con dependencia de existencia son siempre mandatarias para la entidad subordinada, lo cual implica que para ella la cardinalidad siempre es como mínimo uno y a lo sumo uno (1,1). Por otro lado, la participación de la entidad dominante puede ser opcional o estar involucrada con más de una entidad subordinada.

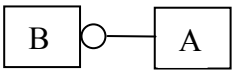
En (Snoeck and Dedene, 2000) se propone una representación gráfica para las asociaciones que expresan dependencia de existencia con el objetivo de evitar confusiones entre estas y la representación clásica de las asociaciones (figura 2.4). El círculo blanco indica la participación opcional en la asociación, una entidad dominante puede o no tener entidades subordinadas asociadas a ella. Mientras que el círculo negro nos indica que la participación es mandataria, cada entidad dominante esta asociada como mínimo con una entidad subordinada en un momento determinado. La flecha indica cardinalidad máxima mucho, o sea que una entidad dominante tiene asociada a ella en un momento dado varias subordinadas y la línea por su lado denota cardinalidad máxima uno.



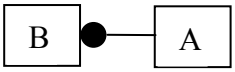
Las entidades dominantes del conjunto de entidades B se asocian con cero, una o varias entidades subordinadas del conjunto de entidades A en un momento dado



Las entidades dominantes del conjunto de entidades B se asocian con una o varias entidades subordinadas del conjunto de entidades A en un momento dado



Las entidades dominantes del conjunto de entidades B se asocian con cero o una entidad subordinada del conjunto de entidades A en un momento dado



Las entidades dominantes del conjunto de entidades B se asocian con una entidad subordinada del conjunto de entidades A en un momento dado

En cada caso las entidades de A están asociadas solamente con una y siempre la misma entidad dominante de B

Figura 1.14 *Notación gráfica para la dependencia de existencia*

Para el ejemplo de la biblioteca, la representación según esta notación, quedaría como se muestra en la figura 1.15. En un momento dado, cada *miembro* tiene cero, uno o varios *préstamos* y a su vez cada *préstamo* se asocia con solo uno y siempre el mismo *miembro*.

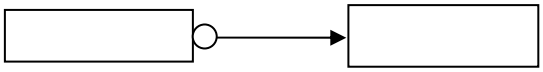


Figura 1.15 *Asociación con dependencia de existencia entre los conjuntos de entidades MIEMBRO y PRESTAMO*

1.7 Conclusiones parciales

El modelo conceptual ER consta de un conjunto de primitivas básicas, a través de las cuales se pueden modelar la mayoría de las características de las bases de datos. A lo largo de los años varios investigadores han incorporado nuevos conceptos, con el objetivo de enriquecer la representación de los esquemas conceptuales, dando como resultado el modelo ERE. Existe una amplia variedad de representaciones gráficas para las construcciones del modelo y por ello este trabajo adopta la notación de (Elmasri and Navathe, 2003), salvo para la representación de las entidades débiles, debido a la importancia que le concede (De Miguel and Piattini, 1993) a

la diferenciación entre entidades débiles con dependencia de existencia e identificación .Varias de las construcciones de dichos modelos garantizan integridad estructural a través de la dependencia de existencia, sin embargo es un concepto que frecuentemente aparece en la literatura tratado de forma incompleta y por ello se busca una sistematización del concepto en asociaciones siguiendo el enfoque de (Snoeck and Dedene, 1998), así como su representación en los diagramas conceptuales.

CAPÍTULO 2. RESTRICCIONES DE INTEGRIDAD EN SQL3

En cualquier aplicación de negocio, investigación o para toma de decisiones, un aspecto importante es la exactitud de los datos que manipula. Con el objetivo de evitar datos erróneos que no reflejen información verdadera del universo de discurso, las restricciones de integridad semánticas deben ser especificadas desde la modelación conceptual de los datos y mantenidas durante todas las etapas siguientes del diseño. Los modelos lógicos soportados por los SGBD comerciales proveen mecanismos para hacer cumplir tales restricciones de integridad. En el epígrafe 2.1 se introducen los conceptos del modelo relacional, las nociones básicas y clasificación de las restricciones de integridad. En las secciones siguientes se ofrece una descripción de los recursos que brinda el lenguaje estándar SQL: 1999 o también conocido como SQL3, para representar las restricciones de integridad.

2.1 Restricciones de integridad en el modelo relacional

El modelo de datos relacional (Codd, 1970) se basa en el concepto matemático de relación. Una relación es un conjunto de n-tuplas, donde una tupla puede representar entidades de un mismo conjunto así como también interrelaciones entre entidades de diferentes conjuntos. En este último caso deben observarse ciertos criterios sobre la representación de la interrelación en uno de los conjuntos o su representación totalmente independiente. Los criterios fundamentales que se manejan son, representar explícitamente la interrelación mediante las llaves de los conjuntos que están siendo interrelacionados y cualquier otro atributo que requiera la interrelación, o representarla implícitamente en uno de los conjuntos (esto es posible cuando la cardinalidad de la interrelación contiene al menos un uno y su grado es uno o dos). Para esta representación implícita, el conjunto que corresponde a la parte muchos o cualquiera de los involucrados en una interrelación uno-uno (el que sea mas natural), toma la llave del otro; conocidas éstas como llaves extranjeras (Wilmot, 1984). El uso o no de llaves foráneas es muy discutido y hay criterios prácticos que en ocasiones las justifican (Teorey et al., 1986).

Las relaciones pueden ser vistas naturalmente como tablas, en las cuales cada tupla es una fila, y todas las filas de una tabla tienen el mismo número de columnas, donde los valores de una columna son tomados del mismo dominio. Los dominios o tipos de datos básicos en SQL son INTEGER, SMALLINT, DECIMAL, VARCHAR o DATE, entre otras. Cada dominio incluye el valor especial NULL, significando que el valor no es conocido o no aplicable. En lo adelante, se usan los términos tabla/ fila/ columna en lugar de las nociones matemáticas relación/ tupla/ atributo a menos que sea necesario, debido a que son los términos usados en SQL. Un detalle importante es que a diferencia de las relaciones, las tablas son colecciones de filas, permitiendo filas duplicadas a menos que los duplicados sean explícitamente excluidos por una restricción de unicidad.

SQL:1999 extiende SQL-92, o también conocido como SQL2 (Melton and Simon, 1995) hacia un modelo objeto-relacional que soporta constructores de tipos como ROW, REF y ARRAY, estructuras definidas por el usuario como tipos y subtipos de datos, así como tablas y subtablas. Desafortunadamente los constructores de tipos de datos no son completamente ortogonales, por ejemplo las listas son solamente unidimensionales. Otros tipos de colecciones de datos como SET, MULTISET y LIST han sido pospuestas para siguientes versiones de estándares de SQL.

Las restricciones de integridad semánticas especifican qué estados de la base de datos y de las transiciones o secuencias de las mismas son admisibles y reflejan correctamente la verdadera semántica del universo de discurso. Tales restricciones pueden formularse como predicados asociados a un esquema de base de datos o pueden ser especificados como componentes procedurales en la definición del esquema al implementarse para un SGBD particular.

El estado de una base de datos varía cuando se insertan, actualizan o eliminan filas en las tablas. Tales operaciones ocurren dentro de transacciones. Una transacción básicamente, es una secuencia de operaciones que realizan una función lógica en una aplicación de base de datos y se adhiere a los principios de atomicidad, consistencia, aislamiento y durabilidad (Gray, 1978). Las transacciones inducen una transición de un estado de la base de datos a otro. En SGBD SQL, una transacción comienza con la primera operación declarada después de que

finalice la transacción anterior y es terminada por COMMIT, terminación con éxito o ROLLBACK que genera una interrupción. Una interrupción provoca que los cambios realizados por la transacción sean deshechos.

Es parte de la responsabilidad del esquema de recuperación del SGBD el manejar correctamente las interrupciones de las transacciones, debe notarse que al menos hay dos tipos de interrupciones. El primero se refiere a las interrupciones técnicas que son iniciadas por el SGBD, por ejemplo, solucionar una situación de bloqueo, deadlock, abortando una o varias transacciones estancadas. El segundo, que resulta ser el de interés para este estudio, es el tipo de interrupciones que están relacionadas con las restricciones de integridad, abortando aquellas transacciones que producen un estado de la base de datos semánticamente incorrecto.

Un estado de la base de datos es consistente si satisface todas las restricciones de integridad definidas. En el contexto de transacciones y operaciones de interrupción es importante distinguir dos nociones de consistencia: una a nivel de transacción y otra a nivel de definición. La consistencia a nivel de transacción requiere que una transacción conduzca a la base de datos de un estado consistente a otro también consistente, no necesariamente diferente, aunque posiblemente con estados intermedios inconsistentes. La consistencia a nivel de definición requiere que ninguna declaración genere un estado inconsistente, esta última es soportada por SQL: 1999. Una transacción donde cada declaración satisface las restricciones definidas también satisfacen las restricciones a nivel de transacción. Según los niveles de restricciones que soporta un SGBD puede brindar la posibilidad de un rollback de toda la transacción o solamente de la declaración que viola cierta restricción. Esto último permite más flexibilidad ya que las violaciones de restricciones causadas por una declaración pueden ser compensadas por declaraciones de reparación apropiadas que están incluidas en la propia transacción en ejecución.

Las restricciones de integridad se garantizan finalmente a través de declaraciones sobre las propiedades de las columnas, filas o tablas de una base de datos. Asumiendo que un esquema de base de datos es un conjunto $E = \{T_1, T_2, ..., T_n\}$ de tablas. Las restricciones de integridad

pueden estar asociadas con una o varias tablas del esquema *E* y pueden referirse a una o varias filas de cada una de dichas tablas.

- Restricciones de filas, están asociadas a exactamente una tabla T_i y pueden ser evaluadas independientemente para cada fila $f \in T_i$. Además pueden referirse a una o varias columnas de una fila. El ejemplo típico constituye una restricción de dominio.
- Restricciones de tablas, su evaluación involucra al menos dos filas de la misma tabla T_i. El ejemplo más simple es la restricción de unicidad de la llave primaria. Los ejemplos más complejos incluyen funciones agregadas donde la evaluación se efectúa sobre todas o un subconjunto de las filas de T_i.
- Restricciones entre tablas, se refieren a filas que al menos pertenecen a dos tablas T_i y T_j con $i \neq j$. En general la mayoría de las restricciones pertenecen a esta clasificación. Por ejemplo las restricciones de llaves foráneas.

2.2 Soporte que ofrece SQL: 1999 para las restricciones de integridad

El estándar SQL: 1999 (Eisenberg and Melton, 1999) soporta definiciones para las restricciones de integridad tanto al nivel declarativo como procedural, estas últimas en términos de triggers. En los epígrafes siguientes, se ofrece una descripción de ambos métodos.

2.2.1 Definición declarativa de restricciones de integridad

Una restricción de integridad puede ser especificada dentro de una declaración que crea o modifica una tabla. Cada restricción de integridad está asociada con un descriptor que consiste en el nombre de la restricción, el modo de comprobación inicial y una bandera que indica si la comprobación de la restricción puede ser aplazada hasta el final de una transacción:

1. Dentro de un esquema de base de datos, una restricción de integridad es identificada unívocamente por su nombre. Si no es especificado explícitamente, entonces el sistema proporcionará implícitamente un nombre, dependiendo de la implementación.
2. El modo de comprobación determina el momento en que la restricción de integridad tiene que ser comprobada dentro de una transacción. Se distinguen dos modos: inmediato y aplazado. Si el modo es inmediato, entonces la restricción se comprueba

con eficacia al final de cada declaración que podría violarla. De lo contrario, la comprobación es aplazada hasta el final de la transacción.

3. El modo de comprobación inicial define el modo por defecto en que es válido el inicio de cada transacción. Debe notarse que sólo las restricciones aplazables pueden pertenecer al modo aplazado, de lo contrario el modo de comprobación es inmediato. De no ser especificado, implícitamente el modo de chequeo es inmediato. En principio todas las restricciones de integridad pueden ser especificadas como aplazables.

En SQL: 1999 se definen las siguientes construcciones, que pueden ser usadas para declarar algunas restricciones específicas. Tales construcciones también las provee SQL: 1999, además de proporcionar algunas extensiones que se mostrarán más adelante:

NOT NULL impide a una columna tomar el valor nulo.

DEFAULT establece el valor por defecto de una columna.

UNIQUE define que una columna o conjunto de columnas deben tener valores únicos dentro de una tabla.

PRIMARY KEY determina la llave primaria de una tabla.

FOREIGN KEY (REFERENCES) especifica una llave foránea cuyos valores deben coincidir con cierta llave primaria especificada.

CHECK define una restricción de integridad basada en una condición.

DOMAIN crea o restringe el dominio de una columna.

ASSERTION define una restricción de integridad para una tabla independiente basada en una condición.

Semántica de las restricciones de control CHECK

En términos del lenguaje SQL, una restricción de control **CHECK** (condición) definida en una tabla T se satisface si y sólo si **NOT EXISTS**(**SELECT * FROM** *T* **WHERE NOT**(condición)) es verdadero o desconocido. Recordando que en SQL existen tres valores lógicos, verdadero, desconocido y falso.

Semántica de las restricciones de unicidad y llave primaria

Una restricción de unicidad **UNIQUE**($u_1, u_2, ..., u_n$) asociada a una tabla T se satisface si y solo si no existen dos filas f_1, f_2 en T cuyos valores en las respectivas columnas u_i sean idénticos o al menos uno sea nulo. Una definición formal, siendo $\perp$ el valor nulo:

$$\forall f_1, f_2 \in T : \left(\bigwedge_{i=1}^{n} f_1.u_i \neq \perp \wedge f_2.u_i \neq \perp \right) \Rightarrow \left(\bigvee_{i=1}^{n} f_1.u_i \neq f_2.u_i \right)$$

Las restricciones de llave primaria **PRIMARY KEY**(u_1, u_2, ..., u_n) debe satisfacer la restricción de unicidad y además no permite valores nulos para ninguna de las columnas involucradas, formalmente:

$$\forall f \in T : \left(\bigwedge_{i=1}^{n} f.u_i \neq \perp \right) \wedge \forall f_1, f_2 \in T : \left(\bigvee_{i=1}^{n} f_1.u_i \neq f_2.u_i \right)$$

Semántica de las restricciones de llave foránea

Las restricciones referenciales son importantes para expresar dependencia entre las filas de las mismas o diferentes tablas. Donde los valores de un subconjunto de columnas $c_1,...,c_n$ de la tabla que referencia o miembro, pertenecen a las columnas $u_1,...,u_n$ que constituyen la llave primaria de la tabla referenciada o propietaria. La correspondencia entre los valores de referencia puede ser clasificada en simple, parcial o total.

La referencia es simple cuando para cada fila f_1 de la tabla miembro M, al menos uno de los valores de las columnas de llaves foránea c_i es nulo o es igual al valor de la correspondiente columna referenciada u_i para alguna fila f_2 de la tabla propietaria P. Formalmente:

$$\forall f_1 \in M : \left(\bigwedge_{i=1}^{n} f_1.c_i \neq \perp \right) \Rightarrow \exists f_2 \in P : \left(\bigwedge_{i=1}^{n} f_1.c_i = f_2.u_i \right)$$

La referencia es parcial cuando para cada fila f_1 de la tabla miembro M, debe haber alguna fila f_2 en la tabla propietaria P, tal que el valor de cada columna de la llave foránea c_i es nulo o igual al valor de la correspondiente columna referenciada u_i en f_2. Formalmente:

$$\forall f_1 \in M : \left(\bigvee_{i=1}^{n} f_1.c_i \neq \perp \right) \Rightarrow \exists f_2 \in P : \left(\bigwedge_{i=1}^{n} f_1.c_i = \perp \vee f_1.c_i = f_2.u_i \right)$$

La referencia es total cuando para cada fila f_1 de la tabla miembro M, el valor de cada columna de llave foránea c_i debe ser nulo o debe haber alguna fila f_2 en la tabla propietaria P tal que el valor de cada columna c_i es igual al valor de la correspondiente columna referenciada u_i en la fila f_2. Formalmente:

$$\forall f_1 \in M : \left(\bigvee_{i=1}^{n} f_1.c_i \neq \perp \right) \Rightarrow \exists f_2 \in P : \left(\bigwedge_{i=1}^{n} f_1.c_i \neq \perp \wedge f_1.c_i = f_2.u_i \right)$$

Para ilustrar el uso de las construcciones anteriores, suponga una aplicación que maneja información sobre empleados. Los empleados son unívocamente identificados por el número del carné de identidad. De cada uno se almacena el nombre y apellidos, se asume que las combinaciones de los mismos tienen que ser únicas. Cada empleado tiene un supervisor, excepto el gerente superior que se supervisa a si mismo. Además, en la compañía se distinguen tres tipos diferentes de plazas: vendedor, gerente y oficinista; por defecto un empleado es vendedor. El sueldo debe ser mayor que $300 y se establece un proceso de estimulación según el cumplimiento del mes inferior a los $200. Finalmente, el sueldo básico más el estímulo de cada empleado debe ser menor que $1000.

```
CREATE DOMAIN Plaza CHAR (7)
DEFAULT ´vendedor´
CHECK (VALUE IN (´vendedor´, ´gerente´, ´oficinista´));

CREATE TABLE Empleado (
        ci SMALLINT PRIMARY KEY,
        nombre VARCHAR (20) NOT NULL,
        primer_apellido VARCHAR (20) NOT NULL,
        segundo_apellido VARCHAR (20) NOT NULL,
        ocupacion Plaza,
        supervisor SMALLINT REFERENCES Empleado (ci),
        salario DECIMAL(7,2) NOT NULL CHECK(salario > 300),
        estimulo DECIMAL(7,2) NOT NULL CHECK(estimulo < 200),
        UNIQUE(nombre, primer_apellido, segundo_apellido),
        CHECK(salario + estimulo < 1000));
```

Para garantizar que en el año la suma de los salarios totales, incluidos los estímulos, de todos los empleados de la empresa tiene que ser inferior a los $500 000, se puede definir la siguiente restricción utilizando una ASSERTION:

```
CREATE ASSERTION salario_maximo_anual
        CHECK ((SELECT 12*(SUM (salario + estimulo)) FROM Empleado) < 500000);
```

Las restricciones de integridad pueden ser nombradas, esto es un medio importante para eliminar fallos de la aplicación así como para administrarlas. Por ejemplo, para eliminar dinámicamente, deshabilitar o cambiar el modo de comprobación de una restricción específica. Otro modo de ver el ejemplo anterior puede ser:

```
CREATE TABLE Empleado(
        ...
        CONSTRAINT chequeo_unicidad_nombre
                UNIQUE (nombre, primer_apellido, segundo_apellido),
        CONSTRAINT chequeo_salario CHECK(salario + estimulo < 1000));
```

2.2.2 Sintaxis para la declaración de restricciones

A las restricciones declaradas debe asociársele el modo en que serán chequeadas, la especificación del mismo tiene que obedecer la siguiente gramática:

```
<modo> ::= INITIALLY {IMMEDIATE | DEFERRED} [[NOT] DEFERRABLE]
        | [[NOT] DEFERRABLE] INITIALLY {IMMEDIATE | DEFERRED}
```

SQL clasifica las declaraciones de restricciones en cuatro categorías: restricciones de columnas, restricciones de tablas, restricciones de dominio y predicados. Las restricciones de columnas son especificadas en su definición, inmediatamente después de especificar el tipo de datos que se almacenarán en las mismas. Existen cuatro tipos de restricciones: no nulos, comprobación, unicidad / llave primaria y restricciones referenciales. Tales restricciones son especificadas utilizando las siguientes sintaxis:

```
< definicion_restriccion_columna > ::= [CONSTRAINT < nombre >]
        < restriccion_columna >
        [< modo >]

< restriccion_columna > ::= NOT NULL
        | CHECK ( < condicion_busqueda > )
        | UNIQUE
        | PRIMARY KEY
        | < referencia >
```

< referencia > ::= **REFERENCES** < nombre_tabla >
(< lista_columnas >)
[**MATCH** {**SIMPLE | PARTIAL | FULL**}]
[< def_accion >]

< def_accion > ::= **ON UPDATE** < accion_ejecutar >
[**ON DELETE** < accion_ejecutar >]
| **ON DELETE** < accion_ejecutar >
[**ON UPDATE** < accion_ejecutar >]

< accion_ejecutar > ::= **NO ACTION**
| **RESTRICT**
| **CASCADE**
| **SET NULL**
| **SET DEFAULT**

Con respecto a las restricciones de columnas, SQL: 1999 amplía el anterior estándar SQL-92 de dos modos. Primero, introduce la opción restrict para restricciones de referencia y segundo, soporta una cláusula scope para definir el alcance de las referencias de columnas, que puede ser definida usando la sintaxis:

< scope > ::= **SCOPE** < tipo_nombre_tabla >
[**REFERENCES ARE** [**NOT**] **CHECKED**
[**ON DELETE** <accion_ejecutar>]]

Por defecto la referencia de una llave foránea no es comprobada. En este caso pueden ocurrir referencias a datos que aún no existen en la base de datos. A pesar de que REFERENCES ARE NOT CHECKED está predeterminado, esto puede ser explícitamente especificado. Sin embargo si la referencia debe ser comprobada debe especificarse a través de la cláusula REFERENCES ARE CHECKED. Por otro lado si no se especifica acción de referencia, implícitamente está definido ON DELETE NO ACTION.

Las restricciones a las tablas son especificadas independientemente de la definición de columnas. Se distinguen tres tipos: comprobación, unicidad / llave primaria y restricciones referenciales. La sintaxis es similar a las restricciones de columna:

< definicion_restriccion_tabla > ::= [**CONSTRAINT** < nombre >]
< restriccion_tabla >
[< modo >]

< restriccion_tabla > ::= **CHECK** (< condicion_busqueda >)
| **UNIQUE** ({< lista_columna > | **VALUE**})
| **PRIMARY KEY** (< lista_columna >)
| **FOREIGN KEY** (< lista_columna >)
< referencia >

Es importante notar que SQL: 1999 usa las restricciones de tabla para referirse a restricciones de integridad que son especificadas dentro de la definición de una tabla, pero fuera de la definición de una columna. Así, una restricción de tabla en SQL: 1999 puede ser de una fila, tabla o entre tablas según la clasificación ofrecida en el epígrafe 2.2.

Las restricciones de dominio son especificadas fuera de la definición de tablas, de acuerdo con la sintaxis:

< definicion_dominio > ::= **CREATE DOMAIN** < nombre >
[**DEFAULT** < valor_por_defecto >]
[< restricciones_dominio >]

< restricciones_dominio > ::= [**CONSTRAINT** < nombre >]
CHECK (< condicion_busqueda >)
[< modo >]

Análogamente los predicados son especificados independientemente de la definición de tablas o columnas, obedeciendo la siguiente gramática:

< definicion_predicado > ::= **CREATE ASSERTION** < nombre >
CHECK (< condicion_busqueda >)
[< modo >]

La violación de una restricción de referencia puede ocurrir ante en los siguientes casos:

1. Una operación de inserción o actualización en una tabla que referencia
2. Una operación de eliminación o actualización en una tabla referenciada

Dependiendo del universo de discurso, en ocasiones es más apropiado para mantener la integridad de referencia realizar operaciones que reviertan la inconsistencia o la compensen, en lugar de rechazar la operación que violaría la integridad de referencia. Tales operaciones de compensación son conocidas como acciones de referencia. El SQL: 1999 permite que el programador o diseñador defina acciones de referencia diferentes ante las operaciones de actualización y eliminación que afectan a las tablas que tienen referencia a otra(s) tabla(s):

CASCADE: la actualización o eliminación de una fila referenciada conduce a una actualización o eliminación respectivamente de todas las filas que referencian a dicha fila.

SET NULL: cada columna c_i de llave foránea para cada fila que hace referencia a la fila actualizada o eliminada toma el valor nulo. Esta regla no debe ser especificada si alguna columna de llave foránea no admite valores nulos.

SET DEFAULT: la actualización o eliminación de una fila referenciada conduce a que las llaves foráneas de todas las filas que la referencia, tomen sus respectivos valores por defecto.

RESTRICT: esta regla rechaza la actualización o eliminación de la fila referenciada.

NO ACTION: es implícita si ninguna regla es especificada y similar a las anteriores excepto unas diferencias sutiles, que son descritas a continuación.

Como se ha mencionado antes, SQL: 1999 requiere que el chequeo de restricciones ocurra con eficacia al final cada declaración que implique algún cambio en la base de datos. La diferencia principal entre RESTRICT y NO ACTION radica en el momento de comprobación. Las restricciones de referencia en la que se ha especificado la regla RESTRICT son comprobadas antes de todas las restricciones de referencia de otros tipos. Según (Horowitz, 1992), esta regla puede hacer que una declaración de cambio de datos se obvia y se retorna al punto anterior prematuramente ya que las acciones de referencia como CASCADE o SET NULL pueden reparar violaciones de restricción intermedias. Las restricciones de referencia con NO ACTION, por otra parte son comprobadas al final, después de que todas las otras restricciones de referencia han sido tratadas. Así, no se tendrá en cuanta una violación de restricción intermedia a menos que esto mantenga una violación hasta el final. Por lo tanto, en ocasiones la regla NO ACTION es llamada una restricción suave mientras que RESTRICT se clasifica como una restricción fuerte.

2.2.3 Definición procedural de restricciones de integridad a través de triggers

El SQL: 1999 también proporciona el concepto de triggers o procedimiento que es automáticamente invocado por el SGBD en respuesta a algún evento en la base de datos. Los

triggers pueden ser vistos como reglas de evento - condición - acción que permite a los usuarios poner en práctica la lógica de la aplicación dentro del SGBD. También pueden ser usados para supervisar modificaciones de la base de datos, propagar automáticamente modificaciones de los datos, crear alarmas o hacer cumplir restricciones de integridad. Este estudio sólo se concentra en el último rasgo. Un trigger en SQL: 1999 tiene los componentes siguientes:

Un nombre único para identificarlo dentro de la base de datos

Un evento provocador como la inserción, eliminación o actualización en una tabla

Un tiempo de activación que puede ser antes o después de ejecutar el evento provocador

Una cláusula que especifica si debe tenerse en cuenta para cada fila o declaración

Una condición, tan compleja como se quiera, que puede incluir cualquier condición válida SQL

Una acción provocada que puede ser cualquier secuencia válida de declaraciones procedurales de SQL

Finalmente, se define el timestamp como el momento en que es creado

Un trigger es implícitamente activado siempre que ocurra el evento especificado. Así, la base de datos puede reaccionar ante cambios provocados por las aplicaciones o los usuarios. Para un mismo evento pueden definirse varios triggers, en este caso el último que fue creado, o sea el de timestamp más alto, es el primero en ser ejecutado. Debe notarse que a diferencias de las reglas generales de evento - condición - acción (Buchmann, 1993, Paton and Díaz, 1999) la ejecución de la condición y la acción de los triggers en SQL: 1999 no pueden ser ni aplazada ni separada. En SQL:1999 los triggers se definen según la gramática:

< definicion_trigger > ::= **CREATE TRIGGER** < nombre >
{**BEFORE | AFTER**}
{**INSERT | DELETE | UPDATE** [**OF** < lista_columnas >]} **ON** < tabla >
[**REFERENCING** < antigua_nueva_lista >]
[**FOR EACH** {**ROW | STATEMENT**}]
[**WHEN** (< condicion_seleccion >)]
< triggered-SQL-declaracion >

< antigua_nueva > ::= {**OLD | NEW**} [**ROW**] [**AS**] < nombre >

|{**OLD | NEW**} **TABLE** [**AS**] < nombre >

< triggered-SQL-declaracion > ::= < SQL-procedimiento-declaracion >
| **BEGIN ATOMIC**
< SQL-procedimiento-declaracion-lista >
END

Según el evento de provocación especificado, un trigger es nombrado de inserción, actualización o eliminación. La condición y acción pueden ser verificadas y ejecutadas, respectivamente para cada fila afectada por el evento que lo provoca o solo una vez por evento. Si ninguna fila es afectada, entonces el trigger no es evaluado. Un trigger en el que se especifique FOR EACH ROW es nombrado trigger a nivel de fila o de lo contrario es nombrado trigger a nivel de declaración si se especifica FOR EACH STATEMENT, por defecto este es el modo definido.

Los tiempos de activación antes y después especifican cuando el trigger debería ser ejecutado, es decir antes de que el evento que lo provoca sea realizado o después del mismo. Si la acción debe ejecutarse antes, no puede contener ninguna declaración que cambie el estado de la base de datos, inserción, actualización o eliminación, como tampoco puede invocar alguna rutina que pueda modificar el estado de los datos. Sin embargo, puede contener un conjunto de declaraciones que cambian el valor de las nuevas variables que se refieren a filas que han sido modificadas por el evento.

Para ejemplificar el uso de los triggers, supongamos que la empresa del epígrafe anterior, esta organizada por departamentos, Cada departamento tiene un identificador único, un nombre único y es localizado en una de las ciudades ´Santa Clara´, ´Habana´ o ´Santiago de Cuba´. Además pueden tener un gerente. La información se almacena en la tabla Departamento y las restricciones de integridad son garantizadas a través de los triggers.

CREATE TABLE Departamento (
identificador **SMALLINT PRIMARY KEY**,
nombre **VARCHAR** (20) **NOT NULL UNIQUE**,
gerente **SMALLINT REFERENCES** Empleado (ci),
localizacion **Char**(9) **DEFAULT** ´Santa Clara´,
CHECK (localizacion **IN** (´Santa Clara´, ´Habana´, ´Santiago de Cuba´)));

```
CREATE TRIGGER al_eliminar_establecer_nulo
        AFTER DELETE ON Empresa
        REFERENCING OLD AS antes
        FOR EACH ROW
        BEGIN ATOMIC
                UPDATE Departamento
                SET gerente = NULL
                WHERE gerente = antes.ci
        END;

CREATE TRIGGER al_actualizar_cascada
        AFTER UPDATE OF ci ON Empleado
        REFERENCING OLD AS antes NEW AS despues
        FOR EACH ROW
        WHEN (EXISTS (SELECT * FROM Departamento WHERE gerente = antes.ci))
        BEGIN ATOMIC
                UPDATE Departamento
                SET gerente = despues.ci
                WHERE gerente = antes.ci
        END;
```

2.3 Conclusiones parciales

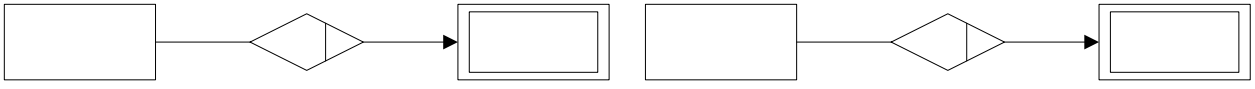
SQL3 proporciona diferentes recursos para el diseño e implementación de restricciones de integridad, tanto al nivel declarativo como procedural. A nivel declarativo, permite definir valores por defecto, impedir valores nulos, exigir unicidad en los valores de un conjunto de columnas, especificar llaves primarias y foráneas, así como acciones referenciales y restricciones de control para columnas, filas y tablas. A nivel procedural, mediante los triggers permiten poner en práctica la lógica de la aplicación dentro del SGBD. Tales recursos deben ser valorados para reflejar las restricciones de integridad derivadas del esquema conceptual en los esquemas lógicos.

CAPÍTULO 3. MODELACIÓN DE LA DEPENDENCIA DE EXISTENCIA EN ESQUEMAS RELACIONALES

En el Capítulo 1 se introduce la presencia de dependencia de existencia en algunas construcciones del modelo ERE, en este capítulo se analiza el comportamiento derivado de tales construcciones al ser transformados a un esquema relacional, con el objetivo de ofrecer scripts en el lenguaje estándar SQL3 de manera que puedan ser implementada la generación automática de la transformación, en una herramienta CASE para diseñadores de bases de datos. Además se analiza en el epígrafe 3.6 la aplicación de la dependencia existencia en la fragmentación horizontal derivada, como otro ejemplo en que este concepto ha resultado de utilidad.

3.1 Entidades débiles

Durante la transformación de un esquema conceptual a un esquema relacional de la base de datos, las entidades débiles con o sin dependencia de identificación, presentan un comportamiento similar. Para cada entidad *Débil* en el esquema ERE (figura 3.1) teniendo a *Fuerte* como entidad propietaria, se crea un esquema de relación *Débil_R* donde se incluyen todos los atributos simples (o componentes simples de los atributos compuestos) de *Débil* como atributos de *Débil_R.* Además, se incluyen las llaves primarias del esquema de relación *Fuerte_R* correspondiente a la entidad propietaria, como llaves foráneas en *Débil_R*. La llave de *Débil_R* queda constituida por la combinación de las llaves primarias de la propietaria y su llave parcial, si existe. Los atributos que constituyen la llave parcial de *Débil_R*, para las entidades débiles sólo con dependencia de existencia, tienen que tener valores únicos a diferencia de las entidades con dependencia de identificación.



a) b)

Figura 3.1 *Notación según (De Miguel and Piattini, 1993) para entidades débiles,*

a)con dependencia de existencia, b)con dependencia de identificación.

De modo general los esquemas de relación y los scripts para transformar ambas construcciones quedan definido como:

Fuerte_R(atributos_identificadores_fuerte, otros_atributos)
Débil_R(atributos_identificadores_fuerte, atributos_identificadores_parciales, otros_atributos)

```
CREATE TABLE Fuerte_T (
        Lista_atributos_identificadores,
        Lista_otros_atributos,
        PRIMARY  KEY (Lista_atributos_identificadores));

CREATE TABLE Débil_T (
        Lista_atributos_identificadores_fuertes,
        Lista_atributos_identificadores_parciales,
        Lista_otros_atributos,
        UNIQUE (Lista_atributos_identificadores_parciales),
        FOREIGN KEY (Lista_atributos_identificadores_fuertes)
                REFERENCES Fuerte_T (Lista_atributos_identificadores)
                ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY  KEY (Lista_atributos_identificadores_fuertes,
                                Lista_Atributos_identificadores_parciales));
```

Lista_atributos_identificadores es una colección de definiciones donde se especifica el nombre de la columna y el dominio para los respectivos atributos que constituyen la llave de la entidad *Fuerte* y Lista_otros_atributos es la declaración del resto de los atributos de la entidad, que no constituyen la llave primaria, de manera similar para la tabla Débil_T, teniendo en cuenta que atributos_identificadores_parciales es la definición de los atributos que constituyen la llave parcial de la entidad débil y pueden aparecer o no. El dominio puede ser uno de los tipos de datos permitidos en el estándar SQL3: INTEGER, SMALLINT, DECIMAL, VARCHAR, DATE, colecciones o los definidos por los usuarios. Para las entidades débiles sin dependencia de identificación en la generación del script SQL, la diferencia con el tratamiento a una entidad débil radica en que la cláusula **UNIQUE** (Lista_atributos_identificadores_parciales) no aparece.

3.2 Especialización y generalización

Los procesos de especialización y generalización pueden considerarse cada uno el inverso funcional del otro. La especialización, por ejemplo, nos permite definir a partir de un conjunto de entidades, otros conjuntos a los que se le asocian atributos adicionales específicos y con los que se pueden establecer asociaciones entre ellos y/o con otros. Por otro lado la generalización, es el proceso inverso de abstracción en el que se suprimen las diferencias entre varios conjuntos de entidades, se identifican los rasgos comunes y se generalizan formando un conjunto de entidades genérico. Esta construcción explícitamente modela dependencia de existencia, siendo E una generalización de un grupo de conjuntos de entidades $E_1, E_2, \ldots, E_n$, donde cada entidad de los conjuntos de entidades $E_1, E_2, \ldots, E_n$ es también un elemento del conjunto de entidades E, por tanto $E_1 \leftarrow E, E_2 \leftarrow E, \ldots, E_n \leftarrow E$.

Hay varias opciones para transformar tales conceptos del modelo ERE a un esquema lógico y depende de las características de los datos. Tomando el modelo relacional para construir el esquema lógico, en lo siguiente se usa Atrs(R) para denotar los atributos de la relación R y Clp(R) para denotar la llave primaria. La idea consiste en convertir cada una de las n especializaciones $\{E_1, E_2, \ldots, E_n\}$ y la generalización E donde sus atributos son $\{k, a_1, a_2, \ldots, a_n\}$ y k es la llave primaria, como esquemas de relación según una de las cuatro variantes (Elmasri and Navathe, 2003):

Opción 1: Crear un esquema de relación R para E con atributos Atrs(R) = $\{k, a_1, a_2, \ldots, a_n\}$ y Clp(R) = k. Crear una relación R_i para cada E_i, $1 \leq i \leq n$, con los atributos Atrs(R_i) = $\{k\} \cup \{$atributos de $E_i\}$ y Clp(R_i) = k.

Opción 2: Crear un esquema de relación R_i para cada E_i, $1 \leq i \leq n$, con los atributos Atrs(R_i) = $\{$atributos de $R_i\} \cup \{k, a_1, a_2, \ldots, a_n\}$ y Clp(R_i) = k.

Opción 3: Crear un solo esquema de relación R con atributos Atrs(R) = $\{k, a_1, a_2, \ldots, a_n\} \cup \{$atributos de $E_1\} \cup \ldots \cup \{$atributos de $E_n\} \cup \{t\}$ y Clp(R) = k. Esta opción es para un cubrimiento exclusivo donde los conjuntos de entidades especializados son disjuntos, y t es un

atributo que indica el conjunto al que pertenece cada tupla, si la hay. Esta alternativa tiene el potencial de crear un gran número de valores nulos.

Opción 4: Crear un solo esquema de relación R con atributos Atrs(R) = $\{k, a_1, a_2, ..., a_n\} \cup \{$atributos de $E_1\} \cup \ldots \cup \{$atributos de $E_n\} \cup \{t_1, t_2, ..., t_n\}$ y Clp(R) = k. Esta opción se usa cuando el cubrimiento es solapado y cada atributo t_i, $1 \leq i \leq n$, es un atributo booleano que indica si una tupla pertenece o no a una especialización E_i.

La Opción 1 puede ser utilizada para cualesquiera restricciones sobre la especialización: total o parcial, exclusivo o solapado. La Opción 2 sólo da buenos resultados con un cubrimiento total y exclusivo. Si el cubrimiento es parcial una entidad que no pertenece a ninguno de los conjuntos especializados se perderá y si es solapado una entidad que pertenezca a más de un E_i tendrá almacenados redundantemente en más de un R_i los atributos heredados de E. Con esta variante ninguna relación contiene todas las entidades del conjunto generalizado, por tanto siempre que busquemos una entidad arbitraria de E, deberemos buscar en todas las n relaciones R_i. Las Opciones 3 y 4 crean una sola relación para representar el conjunto de entidades generalizadas y todas sus especializaciones. Una entidad que no pertenezca a alguno de los subconjuntos tendrá valores nulos para los atributos específicos que no pertenecen a la generalización. Por tal razón, no se recomiendan estas opciones si se han definido muchos atributos específicos para los conjuntos de entidades especializados. Sin embargo, si hay pocos atributos específicos, estas transformaciones son preferibles ante las opciones uno y dos porque hace innecesario operaciones de unión y por tanto pueden implementar una operación más eficiente (Elmasri and Navathe, 2003).

La definición del esquema relacional y el script depende de la opción seleccionada y el cubrimiento de la generalización por parte de las especializaciones. Para este estudio se selecciona la Opción 1, ya que permite representar todos los cubrimientos.

Generalización_R (Lista_atributos_identificadores, Lista_otros_atributos_generales)
Especialización_1_R (Lista_atributos_identificadores, Lista_otros_atributos_específicos)
Especialización_N_R (Lista_atributos_identificadores, Lista_otros_atributos_específicos)

```
CREATE TABLE Generalización_T (
        Lista_atributos_identificadores,
        Lista_otros_atributos_generales,
        PRIMARY  KEY (Lista_atributos_identificadores));

CREATE TABLE Especialización_i_T (
        Lista_atributos_identificadores,
        Lista_otros_atributos_específicos,
        FOREIGN KEY (Lista_atributos_identificadores)
                REFERENCES Generalización (Lista_atributos_identificadores)
                ON DELETE CASCADE ON UPDATE CASCADE,
        PRIMARY  KEY (Lista_atributos_identificadores));
```

Se definen tantas tablas *Especialización_i_T* como especializaciones se requieran. Cuando el cubrimiento es exclusivo se generan un disparador para cada tabla *Especialización_i_T,* con el objetivo de garantizar que al insertar una fila en alguna de las tablas *Especialización_i_T* no exista en otra. Por simplicidad se llama LlaveForanea a la lista de los atributos identificadores de la generalización que además son los atributos identificadores de cada especialización.

```
CREATE TRIGGER Especialización_Exclusiva
        BEFORE INSERT ON Especialización_1_T
        REFERENCING NEW AS esp
                WHEN (NOT EXISTS (SELECT *
                                FROM Especialización_2_T
                                WHERE Especialización_2_T. LlaveForanea =
                                esp.LlaveForanea)

                                UNION (SELECT *
                                FROM Especialización_N_T
                                WHERE OtraEspecialización1. LlaveForanea =
                                espEliminada.LlaveForanea));
```

3.3 Agregaciones

La agregación permite combinar entidades que se relacionan mediante una asociación específica para formar una entidad agregada de más alto nivel. En ocasiones, es útil cuando la propia entidad agregada se debe relacionar con otras entidades y se utiliza como recurso para poder establecer asociaciones entre una interrelación y otras construcciones. Una entidad

agregada depende de cada una de las entidades que la componen, por tanto la agregación representada a través de interrelaciones *member-of* expresa dependencia de existencia.

En el ejemplo de la figura 1.7 se modela el enfrentamiento entre equipos deportivos como una agregación PARTIDO, donde un *equipo_1* se enfrenta a un *equipo_2* en una *fecha* determinada. La existencia de un *partido* específico depende de los *equipos* participantes y la *fecha* en que se enfrentaron. Además un *partido* se refiere siempre a los mismos *equipos* y *fecha*, variaciones en estas últimas implica un enfrentamiento distinto, así como la eliminación de ellos conlleva a eliminar todos los *partidos* en que estaban involucrados.

La agregación siempre genera un esquema relacional, pues externaliza la asociación englobada. Estas asociaciones son transformadas siguiendo las reglas convencionales de (Teorey et al., 1986), se crea un nuevo esquema con los identificadores de las entidades asociadas. La generación del esquema relacional y del script SQL es muy simple e intuitiva donde las acciones referenciales que se definen es la eliminación y actualización en cascada – ON DELETE CASCADE ON UPDATE CASCADE.

3.4 Composición

En la composición las asociaciones *part-of* tienen implícito algunos aspectos de dependencia de existencia, sin embargo no son conceptos equivalentes; en ocasiones algunas partes tienen dependencia de existencia y otras no. Si las partes no dependen de la existencia del todo, el tratamiento de los datos puede verse afectado con una eliminación en cascada. Por ejemplo, las *ruedas* son parte de un *vehículo*, el desensamblaje de un *vehículo* implica la eliminación de una entidad del conjunto VEHÍCULO, pero no conlleva a eliminar las *ruedas* u otras entidades de las partes constituyentes. La semántica de la dependencia de existencia es similar a la de componentes dependientes y aun cuando exista tal dependencia nada impide que la asociación del todo con las partes componentes no pueda ser modificada en el tiempo. En algunos casos, la dependencia implica una eliminación en cascada de componentes cuando la entidad compuesta es suprimida, en otros, la eliminación de una entidad compuesta no es válida mientras haya entidades componentes que su existencia dependa de ella.

El uso de la dependencia de existencia es una alternativa para la composición, si en la asociación *part-of* las partes son dependientes, simplemente puede ser sustituida por la dependencia de existencia y en este caso el comportamiento es idéntico. Cuando las partes no tengan dependencia de existencia, puede ser modelada como un conjunto de entidades, donde representa una especie de contrato entre la parte y el todo mientras exista la asociación entre ellas.

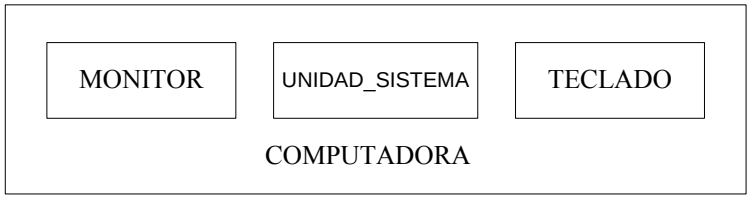


Figura 3.2 *Diagrama que representa la composición de una computadora*

En el diagrama de la figura 3.2 se modelan los datos de una computadora, constituida por un monitor, una unidad de procesamiento y un teclado. Después de la transformación que propone (Snoeck and Dedene, 1998), la figura 3.3 refleja a través de asociaciones con dependencia de existencia la dinámica de los datos, ya que cada componente puede estar en uso a lo sumo por una computadora en cierto momento del tiempo y una computadora está constituida por un monitor, un teclado y una unidad de procesamiento, por tanto la reutilización de componentes es posible.

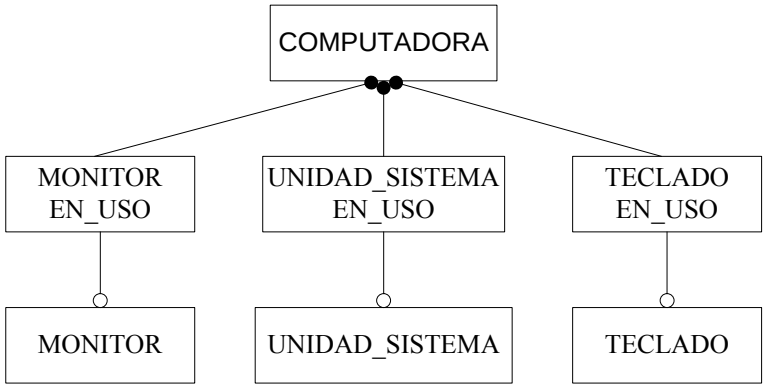


Figura 3.3 *Ejemplo de composición cuando las partes no poseen dependencia de existencia*

Con este ejemplo se puede notar que la dependencia de existencia es una alternativa para evitar las confusiones que en ocasiones aparecen entre los conceptos de agregación y composición en la modelación conceptual. Comparando la asociación *part-of* con la dependencia de existencia, esta última tiene como ventajas que su semántica es simple y fácil de formalizar, el comportamiento es rápidamente comprensible y además captura la semántica y aspectos dinámicos de los datos. Aun cuando la modelación conceptual es favorecida con esta alternativa pues evita confusiones la transformación de la composición a esquemas relacionales difiere de la agregación. Para cada entidad componente se genera un esquema relacional, similar a la agregación y un nuevo esquema relacional para las asociaciones que asocian a la entidad compuesta con otras construcciones del modelo pero la acción referencial que se define es RESTRICT, o sea, no puede eliminarse una entidad componente mientras exista una entidad compuesta que la utiliza. Finalmente en el caso de que las entidades componentes tengan dependencia de existencia se genera un trigger para garantizar la eliminación de las partes componentes cuando la entidad compuesta de la cual dependen es eliminada, dada su simplicidad no se muestra un ejemplo. Los esquemas relacionales y los scripts SQL para la composición cuando las partes no presentan dependencia de existencia pueden ser vistos a través de este ejemplo:

Monitor_R (idMonitor, modelo, fabricante, *otros_atributos*)
Teclado_R (idTeclado, fabricante, *otros_atributos*)
Unidad_Sistema_R (idUnidadSist, procesador, *otros_atributos*)
Computadora_R (idMonitor, idTeclado, idUnidadSist, *otros_atributos*)

```
CREATE TABLE Computadora _T (
        idMonitor       dominio,
        idTeclado       dominio,
        idUnidadSist    dominio,
        Lista_otros_atributos,
        FOREIGN KEY (idMonitor) REFERENCES Monitor_T (idMonitor)
                ON DELETE RESTRICT ON UPDATE CASCADE,
        FOREIGN KEY (idTeclado) REFERENCES Teclado_T (idTeclado)
                ON DELETE RESTRICT ON UPDATE CASCADE,
        FOREIGN KEY (idUnidadSist) REFERENCES Unidad_Sistema _T (idUnidadSist)
                ON DELETE RESTRICT ON UPDATE CASCADE,
        PRIMARY  KEY (idMonitor, idTeclado, idUnidadSist));
```

3.5 Asociaciones

La forma de transformar las asociaciones depende del grado de las mismas. La transformación de las asociaciones binarias, está influenciada por la cardinalidad mínima de las entidades asociadas (Teorey et al., 1986).

- Asociaciones *1:1*: En el caso de que las dos entidades tengan participación obligatoria (cardinalidad mínima 1) la llave de una de ellas pasa a formar parte del esquema que genera la otra, se recomienda que se incluya en el esquema que sea más natural. En el caso de que sólo una entidad tenga participación opcional entonces el esquema correspondiente al conjunto de entidades con participación opcional contiene la llave del conjunto de entidades con participación obligatoria. Por otro lado, en el caso que ambas entidades tengan participación opcional, es posible que cualquiera de los esquemas asuma la llave del otro teniendo en cuenta cual genera menor cantidad de valores nulos (ya se sabe que por ser opcional, no se requiere un valor en toda tupla y necesariamente origina valores nulos); si son significativos los valores nulos que se originan en cualquier dirección se justifica la creación de un esquema independiente con las llaves primarias de ambas entidades.
- Asociaciones *1:N*: Sea *R* una asociación entre las entidades E_1 y E_2 con cardinalidad uno y muchos respectivamente, la asociación *R* se transformará usualmente incluyendo la llave primaria de1 lado uno en el esquema correspondiente a E_2, lado muchos, como un atributo o atributos simples. Cuando la participación en el lado uno sea opcional, el atributo que se incluye como llave foránea en el esquema del lado muchos puede tener valores nulos, en el caso de que los valores nulos sean muchos y no sean factibles, la mejor solución es formar un tercer esquema correspondiente a la interrelación, el cual estará formado por las llaves primarias de los conjuntos de entidades relacionados, como llaves del esquema.
- Asociaciones *N:M* o interrelaciones *1:N* y *1:1* con atributos: Para estos tipos de interrelaciones la solución no depende de la cardinalidad mínima de la interrelación, en estos caso se crea un nuevo esquema tomando como llave primaria una combinación de los atributos que constituyen las llaves primarias de los conjuntos de entidades asociados por la interrelación e incluye, además, sus atributos descriptivos.

Las asociaciones unarias son transformadas de la misma manera que las asociaciones *1:1*, *1:N* ó *N:M*, dependiendo de su cardinalidad máxima. Las asociaciones *n*–arias de grado mayor que dos, por su parte, siguen las mismas reglas que las asociaciones binarias *N:M*, se crea un nuevo esquema que contiene todos los identificadores de las *n* entidades relacionadas, y atributos adicionales en el caso que existan. La llave primaria del nuevo esquema es siempre compuesta para reflejar de esta forma que para modelar un hecho de la realidad se necesita el concurso de al menos *n – 1* entidades, esto es, para formar la llave de una interrelación *n*–aria es obligatoria la inclusión de la llave primaria de las entidades con cardinalidad muchos, en el caso de que no existan o sean menos de *n – 1*, la llave de la interrelación se construye combinado las llaves primarias de los conjuntos de entidades con cardinalidad uno hasta completar *n – 1* entidades en la llave de la asociación.

3.5.1 Asociaciones con dependencia de existencia

Muchos autores están de acuerdo en que la restricción de participación total en las asociaciones implica una cierto tipo de dependencia de existencia (Elmasri and Navathe, 2003), sin embargo para (Put, 1988, Dogac et al., 1990, Snoeck and Dedene, 2000) una diferencia importante entre la restricción de participación total y dependencia de existencia radica en las reglas de actualización. Por ejemplo en una empresa (figura 3.4), si la política establecida es que todo empleado debe pertenecer a un departamento, cada entidad de EMPLEADO sólo participa en una asociación PERTENECE_A, donde la participación de EMPLEADO en PERTENECE_A es total. Por otro lado, en la empresa no todos los empleados dirigen un departamento, así que la participación de EMPLEADO en DIRIGE es parcial.

Figura 3.4 *Diagrama para controlar los empleados y departamentos en una empresa*

En el ejemplo anterior, según la definición, la asociación entre EMPLEADO y DEPARTAMENTO no expresa dependencia de existencia. En un *departamento* pueden existir varios *empleados* y cada *empleado* está asignado a lo sumo a un *departamento*. Sin embargo, la existencia de un *empleado* no depende de un *departamento* y a su vez la existencia de un *departamento* no depende de los *empleados*. En el modelo, sencillamente se refleja que el vínculo entre EMPLEADO y DEPARTAMENTO es mandatario para EMPLEADO, ya que cada *empleado* siempre deberá estar vinculado a un *departamento*. Por otro lado, un *empleado* puede ser asignado a otro *departamento*, incluso regresar a un *departamento* al que perteneció en otro momento; en este caso debido a la unicidad de los identificadores, podría perderse la información inicial o sencillamente no ser claro en la semántica del problema. Generalmente se utiliza otro atributo, como la fecha, para modelar este comportamiento. Este requerimiento es importante en los sistemas de negocios, donde "qué ha sucedido" es tan importante como "qué está sucediendo".

Considerando que la empresa tiene clientes a los que le brinda distintos servicios, entre ellos la realización de proyectos de software (figura 3.5), donde cada proyecto es solicitado exactamente por un cliente, cuya solicitud es almacenada hasta que se le asigne un equipo de trabajo. Los empleados que se dedican a la realización de estos proyectos en un momento determinado solo pueden estar trabajando en uno.

Figura 3.5 *Diagrama que modela los datos sobre los clientes, sus proyectos y empleados que trabajan en ellos.*

La representación de ambas asociaciones es idéntica, aparentemente tiene el mismo comportamiento, sin embargo existe una sustancial diferencia en la semántica de las mismas. Cada *empleado*, en un momento dado puede estar trabajando en solo un *proyecto*, pero en el tiempo puede tener otras asignaciones, en otras palabras, el vínculo TRABAJA_EN es modificable, mientras que la asociación SOLICITA no lo es. Los *proyectos* son pedidos por los *clientes* y pertenecen a ellos todo el tiempo. Por tanto el diagrama de la figura 3.5 puede

considerarse semánticamente incompleto, ya que hay un comportamiento de los datos que no queda reflejado.

Cuando se está en presencia de asociaciones con dependencia de existencia, se puede considerar que hay un comportamiento en el universo de discurso que deriva una modelación más precisa que solo la consideración de una llave foránea. Para que esta situación quede reflejada se propone modelarlo como un hecho independiente. En lugar de generar los esquemas:

Cliente_R (idCliente, otros_atributos_cliente)
Proyecto_R (idProyecto, otros_atributos_proyecto, idCliente)

Donde *idCliente* en el esquema Proyecto_R es un atributo no actualizable, debido a que PROYECTO tiene una fuerte asociación con dependencia de existencia de CLIENTE. Se sugiere agregar una llave artificial para sutilmente reflejar mediante la semántica de la misma que cambios en los atributos que se refieren a CLIENTE o PROYECTO originan un nuevo hecho. Los esquemas relacionales pueden ser renombrados de manera más natural.

Cliente_R (idCliente, otros_atributos_cliente)
Proyecto_R (idProyecto, otros_atributos_proyecto)
Contrato_R (idContrato, idProyecto, idCliente)

La generación del escript es intuitiva teniendo en cuenta que si la asociación expresa dependencia de existencia las acciones referenciales que se definen es la eliminación y actualización en cascada –ON DELETE CASCADE ON UPDATE CASCADE.

3.5.2 Otras consideraciones sobre dependencia de existencia en asociaciones

En el esquema conceptual de una base de datos se describen aspectos estáticos de los datos pero consideraciones semánticas o aspectos dinámicos, como puede ser el comportamiento en el tiempo pueden escapar en la modelación. Sobre la base de lo propuesto en (Snoeck and Dedene, 1998), todas las entidades deben vincularse a través de asociaciones que expresen dependencia de existencia. A simple vista no parece ser obvia la asociación de todas las entidades con dependencia de existencia, sin embargo esto siempre será posible.

En un esquema conceptual la asociación entre dos conjuntos de entidades expresa dependencia de existencia o no, figura 3.5. La asociación "SOLICITA" expresa dependencia de existencia, cada *proyecto* sólo puede existir dentro del contexto de un *cliente* y se refiere exactamente a uno y siempre el mismo todo el tiempo. Un *cliente* puede existir independientemente de si ha solicitado algún *proyecto* o no y puede tener varios *proyectos* en curso. Mientras que la asociación "TRABAJA_EN" no expresa dependencia de existencia. Un *empleado* puede existir fuera del contexto de un *proyecto* y un *proyecto* puede existir fuera del contexto de un *empleado*.

Cuando una asociación no expresa dependencia de existencia, puede ser transformada en un conjunto de entidades que tiene dependencia de existencia de los conjuntos de entidades que participan en la asociación (figura 3.6). En este ejemplo de la empresa, la asociación "TRABAJA_EN" con cardinalidad *1:N*, es convertida en un conjunto de entidades ASIGNACION, que es dependiente de existencia de PROYECTO y EMPLEADO.

Figura 3.6 *Diagrama que modela la asignación de los empleados a los proyectos*

De esta manera se modela lo que puede suceder durante el período de tiempo en que un *proyecto* y un *empleado* están relacionados el uno con el otro. Si en un proyecto pueden estar trabajando cero o varios empleados, entonces cada proyecto tiene cero a muchas asignaciones. Si cada empleado es asignado a exactamente un proyecto a la vez, entonces tiene exactamente una asignación por vez.

El mismo razonamiento se aplica a asociaciones *1:1* y *N:M*. Por ejemplo, en una red de computadoras, las estaciones cliente pueden estar conectadas a diferentes servidores, que a su vez brindan servicios a varios clientes (figura 3.7), la conexión puede ser modelada como una entidad independiente.

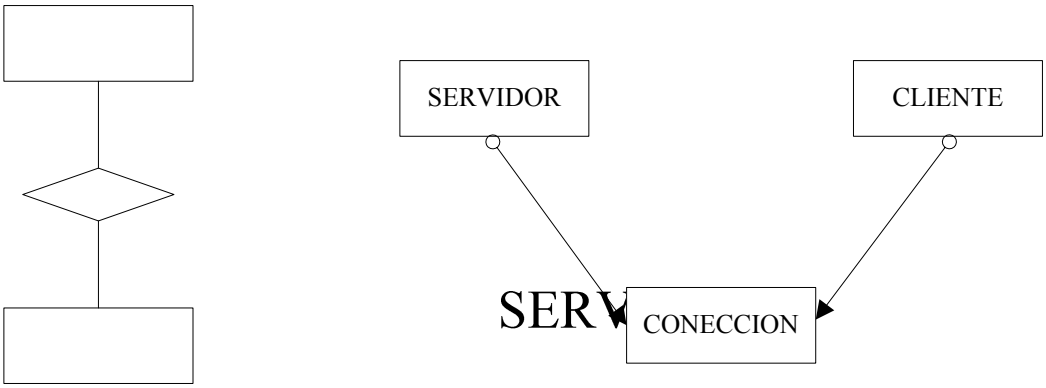


Figura 3.7 *Conversión de una asociación muchos-muchos en un conjunto de entidades con dependencia de existencia*

Las asociaciones unarias, por su parte nunca expresan dependencia de existencia, porque la dependencia de una entidad no puede depender de si misma, ni de una entidad del mismo conjunto. De no ser así se crearía una cadena infinita de dependencias sin una entidad dominante. Por tanto, una asociación unaria siempre debe transformarse en un conjunto de entidades. Por ejemplo, en la figura 3.8 se modela el matrimonio como una asociación unaria entre entidades de tipo PERSONA:

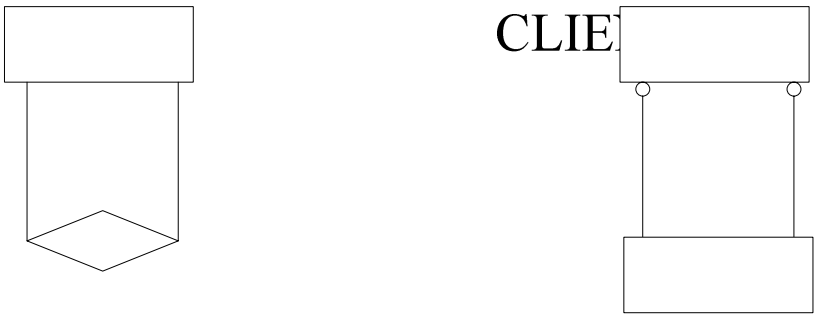


Figura 3.8 *Transformación de una asociación unaria*

3.6 Aplicación en la fragmentación horizontal derivada

En el contexto de bases de datos distribuidas, la fragmentación horizontal derivada se aplica a esquemas relaciones miembros a través de llaves foráneas que hacen referencia a esquemas que han sido fragmentados horizontalmente. Dado un enlace *Llf* donde propietaria(*Llf*) = *P* y miembro(*Llf*) = *M*, la fragmentación horizontal derivada de *M* se define como: $M_i = M \ltimes P_i$,

($1 \leq i \leq w$) donde w es el máximo número de fragmentos que pueden definirse en M y $P_i = \sigma_{Fi}$ (P) es la fragmentación horizontal primaria de P.

Es común que en una tabla M exista más de un llave foránea, por lo que hay más de una posible fragmentación horizontal derivada de M. La decisión de seleccionar una relación candidata a fragmentación horizontal derivada se basa en dos criterios (Özsu and Patrick, 1999):

1. La fragmentación usada en más aplicaciones
2. La fragmentación con mejores características de acople

El primer criterio es más simple si se considera la frecuencia con la que las aplicaciones acceden a los datos. De ser posible, debe facilitar el acceso a los usuarios potenciales de tal forma que el impacto total en el desempeño del sistema sea mínimo. Para explicar el segundo criterio es necesario considerar el efecto de este tipo de fragmentación y es que el acople entre las relaciones propietarias y miembros para responder las consultas, es asistido primero ejecutándolo en pequeñas relaciones (ejemplo fragmentos) y en segundo lugar, desarrollando enlaces potenciales de forma distribuida.

El primer criterio es muy claro debido a que al ser los fragmentos de una relación más pequeños que la relación en sí, se agiliza el acople entre cualesquiera de los fragmentos. El segundo criterio es más importante y es el fundamento de las bases de datos distribuidas debido a que además de ejecutar solicitudes en diferentes sitios, es posible ejecutar una solicitud en paralelo lo que trae como consecuencia que se minimice el tiempo de respuesta y se mejore el desempeño del sistema.

Con el objetivo de favorecer las características de acople, el algoritmo para la fragmentación derivada de un esquema relacional, teniendo en cuenta la presencia de dependencia de existencia puede definirse como:

```
Algoritmo Fragmentacion_Horizontal_Derivada
        Si es aplicable CriterioFHD1 Entonces Aplicar CFHD1
        En otro caso Aplicar CriterioFHD2
```

Fin Algoritmo Fragmentacion_Horizontal_Derivada

Los criterios en cuestión son:
CriterioFHD1: Si el esquema a fragmentar fue obtenido como resultado de la transformación realizada por ERECASE a una asociación con dependencia de existencia mediante la regla de transformación RTDE vista anteriormente, se efectúa la FHD a partir del esquema de mayor cardinalidad que tenga realizada una FHP.

CriterioFHD2: Usar el criterio de "la fragmentación usada en más aplicaciones" referido en el capítulo 1 en la sección relativa a Fragmentación Horizontal Derivada. Su implementación fue muy directa, considerando la frecuencia con la cual las aplicaciones acceden a los datos y que es capturada por un asistente para caracterizar aplicaciones.

3.7 Conclusiones parciales

La dependencia de existencia queda modelada en los esquemas relacionales a través de las llaves foráneas, por tanto se analiza el comportamiento derivado de las construcciones en que aparece al ser transformadas a un esquema relacional. Se ofrecen scripts en SQL3 de para que puedan ser implementada la generación automática de la transformación, en una herramienta CASE. Además se analiza la aplicación de la dependencia existencia en la fragmentación horizontal derivada.

Conclusiones

Como resultado de este trabajo:

1. Se caracterizan los conceptos básicos definidos por Chen en su artículo original del modelo ER para facilitar la identificación de las sucesivas extensiones a los mismos.
2. Se describen un conjunto de construcciones que representan extensiones al modelo básico y que reflejan dependencia de existencia, en particular se formaliza esta propiedad en asociaciones.
3. Se propone un conjunto scripts que modelan el comportamiento de cada tipo de dependencia de existencia y de esta forma se facilita su soporte por herramienta CASE.
4. Se muestra la aplicabilidad de la dependencia de existencia en asociaciones para la fragmentación horizontal derivada en el marco de bases de datos distribuidas.

Referencias Bibliográficas

ABITEBOUL, S., HULL, R. & VIANU, V. (1995) *Foundations of Databases*, Addison-Wesley.

BATINI, C., CERI, S. & NAVATHE, S. B. (1992) *Conceptual Database Design: An Entity-Relationship Approach.,* Redwood City, CA., Benjamin/Cummings.

BATINI, C., CERI, S. & NAVATHE, S. B. (1994) *Diseño conceptual de bases de datos: un enfoque de entidades-interrelaciones*, Díaz de Santos.

BISKUP, J. (1995) Achievements of Relational Database Schema Design Theory Revisited. *Springer-Verlag, Berlin***,** 29-54.

BLAHA, M. R., PREMERLANI, W. J. & RUMBAUGH, J. E. (1988) Relational database design using an object-oriented methodology. *Communications of the ACM archive New York, NY, USA,* 31**,** 414 - 427.

BRACHMAN, R. J. (1983) What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *Computer*.

BRODIE, M. L. (1978) Specification and Verification of Data Base Semantic Integrity. *Technical Report CSRG-91*. University of Toronto.

BRODIE, M. L. (1981) Association: A Database Abstraction. *Entity-Relationship Conf.*

BRODIE, M. L., MYLOPOULOS, J. & SCHMIDT, J. W. (1984) *Perspectives from Artificial Intelligence, Databases, and Programming Languages*, New York: Springen-Verlag.

BUCHMANN, A. P. (1993) Active Object Systems. *NATO ASI OODBS.*

CODD, E. F. (1970) A Relational Model of Data for Large Shared Data Banks. *Commun. ACM,* 13**,** 377-387.

CODD, E. F. (1979) Extending the Data Base Relational Model to Capture More Meaning. *ACM Trans. on Database Systems,* 4**,** 397-434.

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

DATE, C. J. (2000) *An Introduction to Database Systems,* Boston, MA, Addison-Wesley.

DE MIGUEL, A. & PIATTINI, M. (1993) *Concepción y diseño de bases de datos: Del Modelo E/R al modelo relacional,* Madrid, RAMA.

DESLOCH, S. (1993) Semantic Integrity in Advanced Database Management Systems. *Fachbereich Informatik.* Universit¨at Kaiserslautern.

DOGAC, A., OZKARAHAN, E. & CHEN, P. (1990) An integrity system for a relational database architecture. *Eight International Conference on Entity-Relationship Approach.* Toronto, Canada, North-Holland.

EISENBERG, A. & MELTON, J. (1999) SQL: 1999, formerly known as SQL3. *ACM. SIGMOD Rec. ,* 28**,** 131--138.

ELMASRI, R. & NAVATHE, S. B. (1997) *Fundamentals of Database Systems, third edition*, Addison-Wesley

ELMASRI, R. & NAVATHE, S. B. (2003) *Fundamentals of Database Systems, fourth edition*, Addison-Wesley.

ESWARAN, K. P. & CHAMBERLIN, D. D. (1975) Functional SpecificaSpecifications of a Subsystem for Data Base Integrity. IN D.S., K. (Ed.) *1st Int. Conf. on Very Large Data Bases.* Framinghamingham, Mass., USA.

FERG, S. (1985) Modelling the Time Dimension in an Entity-Relationship Diagram. IN CHEN, P. (Ed.) *Fourth International Conference on Entity-Relationship Approach.* Chicago, Illinois, USA, IEEE Computer Society and North-Holland.

GERTZ, M. (1996) *iagnosis and Repair of Constraint Violations in Database Systems*, Infix Verlag, St. Augustin, Germany.

GRAY, J. (1978) Notes on Data Base Operating Systems. *Advanced Course: Operating Systems.*

GREFEN, P. W. P. J. (1992) Integrity Control in Parallel Database systems. The Netherlands, University of Twente.

HAMMER, M. & MCLEOD, D. (1975) Semantic Integrity in a Relational Data Base System. *Proceedings of the International Conference on Very Large Data Bases, September 22-24, 1975, Framingham, Massachusetts, USA*. ACM.

HAMMER, M. & MCLEOD, D. (1978) *The Semantic Data Model: A Modelling Mechanism for Data Base Applications*, ACM.

HOROWITZ, B. M. (1992) A Run-Time Execution Model for Referential Integrity Maintenance. *ICDE.*

KETABCHI, M. A., BERZINS, V. & MARCH, S. T. (1988) An object-oriented semantic data model for CAD applications. *Inf. Sci.,* 46**,** 109-139.

KILOV, H. & ROSS, J. (1994) *Information Modeling: An Object Oriented Approach*, Cliffs N.J.: Prentice Hall.

MAIER, D. (1983) *The Theory of Relational Databases*, Computer Science Press.

MELTON, J. & SIMON, A. R. (1995) *Understanding the New SQL: A Complete Guide*, Morgan Kaufmann.

MYLOPOULOS, J., BERNSTEIN, P. & WONG, H. (1980) A Languaje Facility for Designig Database-Intensive Applications. *TODS,* 5:2.

NAVATHE, S. & CHENG, A. (1983) "A Methodology for Database Schema Mapping from Extended Entity Relationship Models into the Hierarchical Model," The Entity-Relationship Approach to Software Engineering.

ÖZSU, M. T. & PATRICK, V. (1999) *Principles of Distributed Database Systems, Second Edition*, Prentice-Hall.

PATON, N. W. & DÍAZ, O. (1999) Introduction. IN SPRINGER, N. Y. (Ed.) *Active Rules in Database Systems.*

PECKHAM, J. & MARYANSKI, F. J. (1988) Semantic Data Models. *ACM Comput. Surv.,* 20**,** 153-189.

PUT, F. (1988) Introducing dynamic and temporal aspects in a conceptual (database) schema. *doctoral dissertation.* Faculteit der Economische en Toegepaste Economische Wetenschappen, K.U.Leuven.

RUMBAUGH, J. E. (1987) *Relations as Semantic Constructs in an Object-Oriented Language*.

RUMBAUGH, J. E., BLAHA, M. R., PREMERLANI, W. J., EDDY, F. & LORENSEN, W. (1991) *Objecte Oriented Modeling and Design*, Prentice Hall Int´l.

SCHEUERMANN, P. & SCHEFFNER, G. (1980) "Abstraction Capabilities and Invariant Properties Modelling within the Entity-Relationship Approach", Entity-Relationship Aproach to Systems Analysis and Design. 121-140.

SMITH, J. & SMITH, D. (1977) Database abstractions: Aggregation and generalization. *ACM Transactions On Database Systems,* 2**,** 105-133.

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.

SNOECK, M. & DEDENE, G. (2000) Core Modelling Concepts to define Agregation. *L'Objet,* 7**,** 281-306.

STONEBRAKER, M. (1975) Implementation of Integrity Constraints and Views by Query Modification. IN KING, W. F. (Ed.) *SIGMOD International Conference on Management of Data.* San Jose, California, ACM.

TEOREY, T. (1999) *Database Modeling & Design*, 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**,** 197-222.

TEOREY, T. J. & FRY, J. P. (1980) The Logical Record Access Approach to Database Design. *ACM Computing Surveys (CSUR) archive New York, NY, USA,* 12**,** 179 - 211.

TÜRKER, C. (1999) *Semantic Integrity Constraints in Federated Database Schemata*, Infix Verlag, St. Augustin, Germany.

ULLMAN, J. D. & WIDOM, J. (1999) *Introducción a los sistemas de Bases de Datos,* Mexico, Prentice Hall Hispanoamericana.

WERTZ, C. (1993) *Relational Database Design*, CRC Press, Greenwich, CT.

WILMOT, R. B. (1984) Foreign Keys Decrease Adaptability of Database Designs. *Commun. ACM,* 27**,** 1237-1243.