UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
Departamento de Informática

TESIS DOCTORAL

APROXIMACIÓN FORMAL A LAS RESTRICCIONES DE CARDINALIDAD EN UN MARCO METODOLÓGICO DE DESARROLLO DE BASES DE DATOS RELACIONALES

Autor: Dolores Cuadra Fernández
Director: Paloma Martínez Fernández

Leganés, 2003

Índice

INTRODUCCIÓN

Con esta sección se pretende dar una visión preliminar del trabajo de tesis doctoral. Presentaremos el área de investigación donde se ha desarrollado dicho trabajo y describiremos los problemas que hemos encontrado y su importancia, con el fin de proponer una solución válida que los resuelva.

1.1. A MODO DE INTRODUCCIÓN

El área de investigación más general donde se engloba esta tesis doctoral se denomina *ingeniería de datos* (data engineering), área que no tiene una delimitación clara con la Ingeniería del software ya que están íntimamente relacionadas, pero que se diferencian en el ámbito donde trabajan y en su trayectoria histórica. El punto de partida de la ingeniería del software es la realización de una tarea en un ordenador. En la ingeniería de datos se comienza, la mayor parte de las veces, por tareas que existen en cualquiera de estos dos sistemas, basados en documentos o en formularios informatizados y el ingeniero debe buscar la mejor solución para poder utilizar estos datos de manera eficiente y eficaz. Por el contrario, los componentes principales que maneja un ingeniero del software son los aplicaciones, que se componen de datos y algoritmos y en general, el paso de los datos es transitorio (memoria principal). Sin embargo, el ingeniero de datos ve las aplicaciones como un nivel que se construye por encima de los datos, los cuales tienen una propiedad permanente (almacenamiento secundario). Por ello, la diferencia más significativa entre ambas disciplinas es que el ingeniero de software diseña el sistema y el ingeniero de datos construye la base sobre la cual se construye dicho sistema.

Las similitudes entre ambas disciplinas son mayores que las diferencias anteriormente presentadas. Ambas se ocupan de fomentar los principios y los pasos necesario para realizar un adecuado desarrollo de un sistema, que debe encajar perfectamente con sus requisitos y del cual se puede demostrar su correcto funcionamiento. Ambas se ocupan de extraer la semántica imprescindible de una situación del mundo real y mantenerla en la aplicación. La principal diferencia es que en la ingeniería de software la semántica de los datos se codifica, en general, dentro del código de la aplicación y en la ingeniería de datos lo que se pretende es embeberla dentro de los metadatos.

Los principios y los pasos ha seguir, con sus procedimientos y técnicas, para el buen desarrollo de sistemas, en ambas disciplinas, se rigen por lo que se denominan *metodologías*. Las cuales deberán cubrir todo el ciclo de vida del sistema facilitando la actuación sobre el mismo.

Nuestro trabajo se enmarca dentro de la ingeniería de datos y más concretamente en el desarrollo de Bases de Datos (BD). La realización de un buen análisis, diseño e implementación de una BD deberá estar apoyada en una metodología de desarrollo; por tanto, dedicamos la sección 2.4. a la exposición de las metodologías más significativas, en cuál de ellas nos basamos y porqué.

Para la creación de una metodología se deben tener en cuenta los siguientes elementos, que aunque pueden actuar de forma independiente al tener entidad por sí mismos, su utilización a través de una metodología obtendrá su rendimiento máximo.

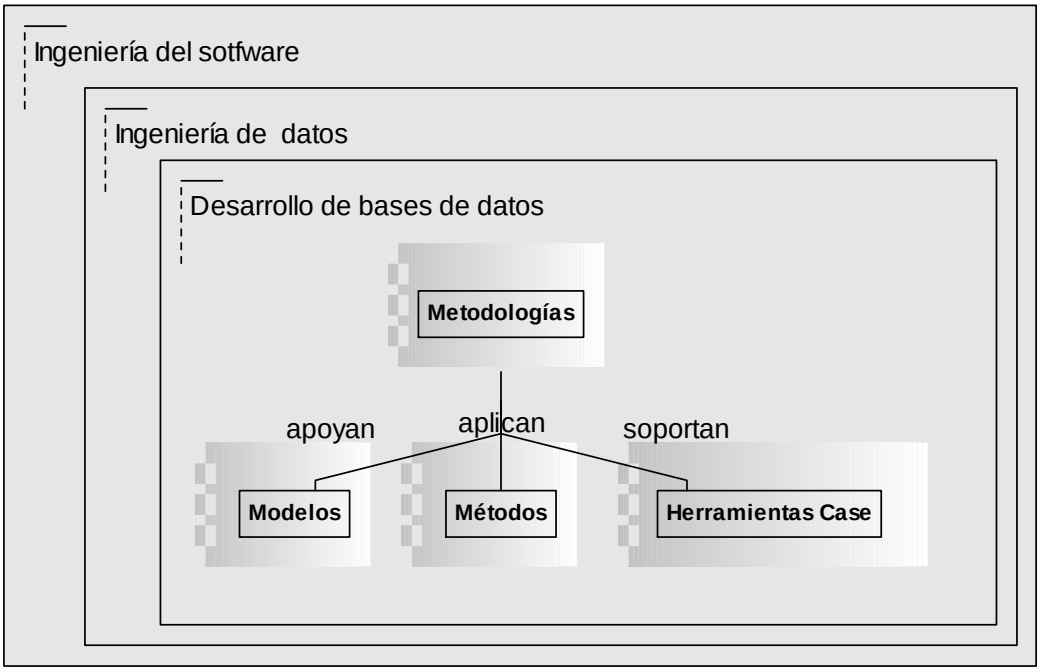


Figura 1.1: Elementos que construyen una metodología

Según la figura 1.1, las distintas fases de las que se compone una metodología de desarrollo de bases de datos se apoyan en uno o varios *modelos de datos* para su descripción. Los modelos se servirán de lenguajes para realizar dicha descripción. *Los métodos* son técnicas que vendrán a indicarnos cómo debe realizarse una determinada actividad dentro de una metodología. Y por último, una metodología deberá estar soportada por una *herramienta automatizada* para mejorar su productividad.

Para comprender, por tanto, la importancia del desarrollo de bases de datos, área en la que se encuentra este trabajo de tesis doctoral, tendremos, primeramente, que analizar la evolución de los modelos de datos.

Los sistemas de información (SI) hacen posible que una organización disponga de los datos necesarios para llevar a cabo una mejor gestión. Toda esta gestión ha ido evolucionando con la tecnología, que ha proporcionado, en primer lugar, soporte hardware, dotando de más memoria secundaria para poder almacenar grandes cantidades de datos y más velocidad de proceso para poder realizar operaciones de recuperación y de actualización de información en un tiempo prudencial. En segundo lugar, soporte software, evolucionando desde los sistemas de gestión de ficheros hasta los actuales Sistemas Gestores de Bases de Datos (SGBD) con arquitecturas diversas, centralizadas o distribuidas, y con tratamiento de distintos tipos de información, BD multimedia, BD orientadas a objetos, BD documentales, etc. Por supuesto, no podemos olvidar que la evolución de las aplicaciones en Internet también han repercutido en la aparición de BD semiestructuradas con la consiguiente implicación en el diseño de las mismas (Feyer & Thalheim, (1999); Gómez, Cachero & Pastor, (2000); Lee et. al. (1999)). Por lo que las bases de datos forman parte de un gran entramado de información (SI) que describe una organización.

La evolución de los SGBD comenzó en los años setenta; anteriormente a esta década existían dos grandes grupos de SGBD comerciales que cubrían todo el

mercado, los que daban soporte a los modelos jerárquicos y los modelos en red. Esta evolución, o mejor dicho revolución en los SGBD y por ende en los SI, surgió en primer lugar, con la aparición del modelo relacional (Codd, 1970) que aportaba, frente a los modelos existentes, una clara distinción entre la representación lógica de los datos y su representación física, y especificó un lenguaje no procedimental para realizar consultas y actualizaciones de los datos, SQL (structured query language). Estos tres primeros modelos los hemos denominado de primera generación y se presentarán en la sección 2.1.

Por otro lado, la separación entre la representación lógica y física de los datos apoyada en la arquitectura ANSI/SPARC y el modelo relacional, llevó a un gran número de investigadores a centrarse en los denominados *modelos de datos semánticos* (Kerschberg et. al., 1976; Tsichritzis & Lochovsky, 1982). Modelos que hemos denominado de segunda generación y cuyas características se presentan en la sección 2.2. La primera publicación surge en 1974 (Abrial, 1974) y a lo largo de esta década todos los esfuerzos fueron encaminados al desarrollo de mecanismos potentes para representar aspectos estructurales sobre los datos. En la actualidad, la atención se dirige hacia la incorporación de aspectos de comportamiento de los datos en los formalismos de estos modelos (por ejemplo, modelos orientados a objetos).

La aparición de los modelos semánticos ha repercutido en distintas disciplinas, por ejemplo, en Inteligencia Artificial con modelos de representación del conocimiento, entre los que se encuentran las redes semánticas y los marcos, o en la Ingeniería del Software, para dar soporte a una de las primeras fases del ciclo de vida software denominada especificación de requisitos. En el Diseño de Bases de Datos lo que se propone es que los modelos semánticos nos sirvan para recoger las especificaciones de la BD, contribuyendo tanto al diseño como a la evolución de esquemas. Por eso se incorporan a las metodologías de diseño para realizar la integración y formalización de la fase de análisis de requisitos.

La aparición de las BD orientadas a objetos a mediados de los 80 no fue una evolución de los sistemas relacionales sino una alternativa a estos, ya que no existía una correspondencia entre los modelo relacionales y los modelos semánticos, que al no estar, estos últimos, implementados en un SGBD debían ser transformados a este modelo. También surgieron por la imposibilidad de representar un número de parcelas del mundo real que requerían el almacenamiento de datos y para los que el modelo relacional no era apropiado. Los ejemplos típicos dados en la literatura son los sistemas de información geográfica (GIS), diseño asistido por ordenador (CAD) y construcción asistida por ordenador (CAM). Este tipo de aplicaciones necesitan tipos de datos mas ricos que los proporcionados por el modelo relacional y los sistemas orientados a objetos heredaban la posibilidad de crear tipos de datos de los lenguajes de programación en objetos. Esta gran revolución en los SGBD, con la creación de un estándar en 1993 (ODMG-93), fue frenada por la demanda del mercado, la cual sigue desarrollando sus sistemas en SGBD relaciones porque argumentan que el desarrollo es menos costoso.

La evolución natural nos ha llevado al campo objeto-relacional cubriendo el agujero de los tipos complejos de datos de los que los sistemas relacionales adolecían. Estos sistemas extienden las características relacionales en: soportar extensiones de tipos base, objetos complejos, herencia y reglas de producción en el contexto SQL. Todavía estas características no están totalmente integradas en los sistemas relacionales, pero el SQL 3 podría dar cobertura total a estas características.

Llegados a este punto podemos especificar que esta tesis doctoral está enmarcada dentro de una metodología de desarrollo de bases de datos objeto-relacional que recoge lo mejor de la vertiente relacional y orientada a objeto.

En el siguiente apartado expondremos los fases principales de las que consta una metodología y las dificultades encontradas en la aplicación de la misma.

1.2. DESCRIPCIÓN DEL PROBLEMA

Las fases características de una metodología de desarrollo de bases de datos solo se diferencian de otras disciplinas en las herramientas, es decir, los modelos empleados para desarrollar cada una de estas fases. Estos modelos son denominados *modelos datos* y a través del lenguaje que nos proporcionen describiremos la base de datos. A continuación pasamos a describir cada una de las fases más comunes de una metodología de desarrollo de bases de datos:

- ✓ Análisis de requisitos. Se recogen las especificaciones que describen el problema o parcela del mundo real que se desea representar y que denominaremos Universo del Discurso (UD). Esta primera fase utiliza el lenguaje natural para recoger la especificación de requisitos y puede refinarse a través de sucesivas entrevistas con los expertos del dominio. Su principal objetivo es acotar o limitar el problema.

- ✓ Diseño conceptual. Se describe el UD a través de un modelo de datos conceptual. Estos modelos poseen la característica de tener un alto nivel de abstracción y que nos servirán para validar la modelización del UD con los expertos del dominio. Los modelos conceptuales deben ser intuitivos y ofrecer distintos mecanismos para recoger toda la semántica del problema.

- ✓ Diseño lógico. En esta fase de la metodología se transforma el esquema conceptual obtenido en la fase anterior a un esquema lógico, el cual se compone de estructuras mas cercanas a la implementación. En la fase anterior no hemos especificado ningún modelo conceptual; como veremos en el capítulo 2 de este documento estudiaremos los más relevantes agrupados por familias. Sin embargo, en esta fase el modelo por excelencia es el objeto-relacional al ser uno de los modelos de datos más extendidos e implementados por los SGBD comerciales.

- ✓ Diseño físico. Se adapta el esquema objeto-relacional a las características propias del sistema gestor de bases de datos (SGBD) donde se implemente la base de datos (BD). En esta fase no existe ningún modelo que nos sirva para describir el esquema físico de la BD, al no estar estandarizado, por lo que depende del conocimiento que el diseñador tenga del SGBD.

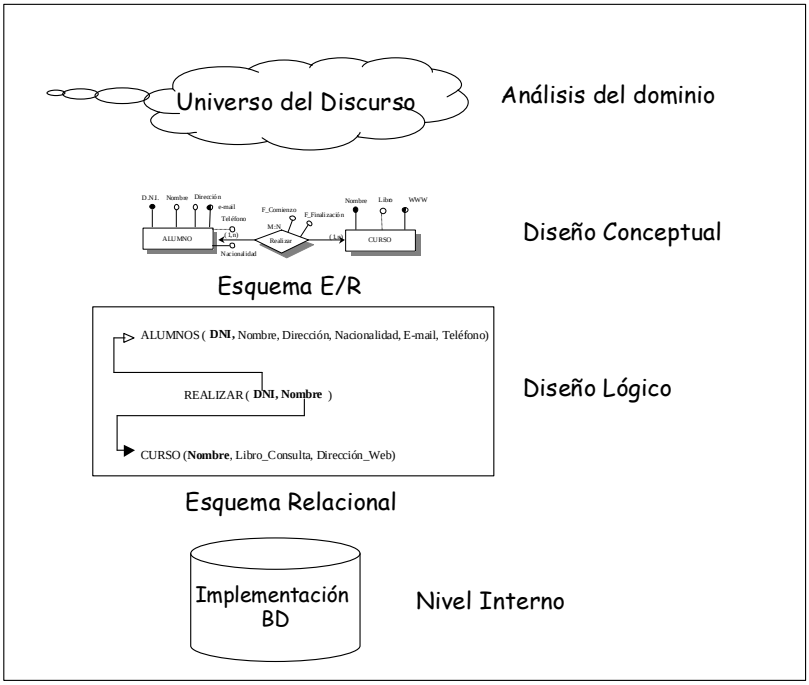


Figura 1.2: Fases comunes para las metodologías de desarrollo de bases de datos.

En general, todas las metodologías siguen la siguiente secuencia para el desarrollo de bases de datos: en primer lugar la fase de análisis de requisitos es donde se realizaran sucesivos refinamientos con los expertos del dominio hasta llegar a una descripción del problema completa. A través de la segunda fase tendremos representado el UD de forma, generalmente gráfica, al ser este el lenguaje que nos proporcionan la mayoría de los modelos de datos conceptuales. Para llegar a la fase de diseño lógico se suelen aplicar un conjunto de reglas establecidas pero no completas, como veremos en la sección 2.2, donde se transforman esquemas

conceptuales a lógicos. En este punto es importante destacar que existen algunas metodologías que las fases anteriores no las tienen en cuenta y comienzan el diseño en esta fase con la consiguiente pérdida de información ya que como veremos en la sección 2.1, los modelos lógicos son menos intuitivos y más rígidos para recoger semántica. Por último, el SGBD elegido para llevar a cabo la implementación vendrá determinado por el modelo lógico, relacional, jerárquico o Codasyl, y se adaptará a las características del producto.

Los problemas que abordaremos en esta tesis doctoral están relacionados con la fase de diseño conceptual y lógica, al entender que son las más problemáticas a la hora de elaborar un buen diseño de la base de datos y más concretamente en:

1. Ambigüedad en el uso de modelos conceptuales, al no existir una semántica bien definida para algunos de los elementos que lo componen (sección 2.3).

2. Conservación de la semántica de un esquema conceptual en su transformación a un esquema lógico (sección 2.4).

En relación con el primer problema, en la fase de diseño conceptual detectamos tal proliferación de modelos que han provocado que existan, para algunos de sus elementos, ambigüedad en su uso y se echa en falta un estándar que normalice o estabilice esta situación, proporcionado definiciones formales de los elementos de los que consta un modelo conceptual.

En relación con el segundo problema, la mayoría de las investigaciones se han encaminado a enriquecer los modelos conceptuales para recoger toda la semántica asociada a un determinado UD y de esta forma describir perfectamente el problema, pero existen pocas soluciones a la conservación de esta semántica en la transformación a un modelo lógico. Los modelos lógicos al ser más cercanos a la implementación tienen una estructura más rígida y no recogen toda la información que posee un esquema conceptual. Esto provoca que las perdidas de semántica

sean recogidas a través de otros mecanismos para que el diseño de la base de datos sea consistente con las especificaciones del UD.

También cabe destacar que estos problemas se ven aumentados cuando se utiliza una herramientas CASE que soporte la metodología que se haya elegido para el diseño de la base de datos. Esto es debido a que no cubren todas las fases de la metodología, no utilizan todos los constructores del modelo de datos e incluso mezcla constructores conceptuales y lógicos. En la sección 2.5.2 detalleremos las características de las herramientas CASE, estudiaremos algunas de las propuestas presentadas dentro de este marco para el diseño de bases de datos, los problemas encontrados y presentaremos el Proyecto PANDORA.

1.3. IMPORTANCIA DEL PROBLEMA

Una vez expuestos los dos problemas en los que se encuadra este trabajo de tesis doctoral nos planteamos la siguiente pregunta; ¿cómo repercuten en el desarrollo de una BD?

En primer lugar, sí la utilización de un determinado modelo conceptual es complicada por no tener claros sus constructores o las herramientas de las que dispones para modelar, esto hará que el esquema conceptual no esté bien diseñado y por lo tanto, el resultado final no será el deseado. Además, si dichos constructores o herramientas no se definen formalmente no se podrán realizar comparaciones entre esquemas, es decir, no se obtendrá una definición de equivalencia entre esquemas. Sin esta equivalencia la reutilización no es posible. Por supuesto el *mapping* o correspondencia entre modelos conceptuales, tampoco.

En segundo lugar, si suponiendo que el esquema conceptual se elabora de forma correcta, la transformación al modelo relacional propuesta en la mayoría de los textos que abordan en problema del diseño de BD, Teorey (1999), suele ser muy

básica tratándose sólo algunos de sus constructores y, consecuentemente, la pérdida de semántica es considerable. Esto repercute de manera negativa en:

- ✓ Desprestigio de la fase de modelado conceptual ya que aunque se realice un buen diseño solo servirá para poder validarlo con los expertos del dominio.

- ✓ Al no poder llevar todas las características del UD hasta la fase de implementación de la BD, se tendrán que buscar mecanismos, controlados fuera de la base de datos, para mantener la consistencia con las especificaciones iniciales y provocará que la semántica no recogida se contemple en las distintas aplicaciones que accedan a la BD. Determinada información, por lo tanto, estará duplicada lo que influirá en el rendimiento y sobre todo en el mantenimiento y adaptabilidad de nuestra BD.

1.4. APROXIMACIÓN A LA SOLUCIÓN

Una vez expuestas las lagunas detectadas en la fase de modelado conceptual así como en su transformación a un modelo lógico, los objetivos que perseguimos en la elaboración del trabajo de tesis doctoral *Restricciones de cardinalidad en Bases de Datos* están enmarcados dentro de una metodología de Desarrollo de Bases de Datos que tendrá soporte en una herramienta CASE denominada PANDORA[1], mejorando dicha metodología y proponiendo distintas heurísticas y procedimientos (denominados *métodos* en la figura 1.1) para su implementación en PANDORA CASE, cubriendo de esta forma las carencias observadas en la mayoría de las herramientas CASE (Castro et al., 2002). Los objetivos principales de este proyecto son:

[1] Plataforma Case para el desarrollo de Bases de Datos y su aprendizaje vía Internet. Proyecto CICYT (TIC99-0215).

- Obtener una herramienta CASE que cubra las todas las fases del ciclo de vida del desarrollo de BD (análisis, diseño e implementación). No hay que olvidar que la fase de análisis de requisitos apenas es considerada en las herramientas CASE actuales.
- Que estas fases se lleven a cabo utilizando un conductor metodológico que guíe a los usuarios en el desarrollo de BD.
- Incluir un componente didáctico que facilite el aprendizaje del diseño de BD y que además pueda utilizarse vía Web.

Los aspectos de la metodología que queremos mejorar vienen motivados por el estudio de los modelos conceptuales y la problemática detectada en la definición del constructor que representa la asociación de sus elementos, denominado *interrelación* o asociación, y concretamente de una de las restricciones que se pueden definir sobre este; la *restricción de cardinalidad.*

Los objetivos que proponemos en el trabajo de tesis doctoral son:

- ✓ En la fase de modelado conceptual: expondremos las distintas definiciones de interrelación junto con las de restricción de cardinalidad, para llevar a cabo una definición formal de ambos conceptos (sección 5.1, 5.2 y 5.3).

- ✓ En la fase de transformación de un esquema conceptual a un esquema lógico, proponemos la conservación de la semántica asociada a las interrelaciones y a sus restricciones de cardinalidad, a través de la utilización de los siguientes mecanismos: disparadores y reglas de verificación, todos ellos desarrollados en el estándar SQL99 (Melton & Simon, (2002)) y que analizaremos en la sección 5.4.

Como aportación al proyecto PANDORA, y más concretamente, dentro de las fases de diseño conceptual y lógico:

- ✓ Desarrollo de heurísticas y procedimientos para el buen desarrollo de un diseño conceptual, basados en los resultados obtenidos por el análisis de los datos de un experimento que hemos llevado a cabo para diseñadores poco expertos, alumnos de 3 ITIG y 4 II, acerca de la detección de interrelaciones en el modelo EER, del cúal hablaremos en la sección 3.2. En Martínez et al.(2000) se desarrolla una parte de lo que será un asistente de PANDORA para interrelaciones binarias.

- ✓ Aplicación de los algoritmos de conservación de semántica en la transformación de interrelaciones en el modelo ER al relacional (sección 5.4).

- ✓ Implementación en SGBD Oracle 9i de dichos algoritmos aplicando las características de dicho gestor, realizando un modelo de ejecución que nos asegure la terminación y no confluencia de los disparadores (Al-Jumaily, 2002) (sección 6.2).

1.5. VALIDEZ DE LA SOLUCIÓN

La validación es la acción y efecto de validar según la DRAE. Validar una propuesta supone darle firmeza, fuerza, seguridad o subsistencia. En la persecución de estos objetivos presentaremos para cada parte de nuestra propuesta su consecuente validación que a continuación enumeramos.

Dentro del modelado conceptual, al proponer una definición de interrelación a partir de la cual definimos las restricciones de cardinalidad asociadas se demostrará de forma teórica que este conjunto de restricciones es suficiente para recoger todas las limitaciones que podemos imponer a un tipo de interrelación, a partir de unas condiciones previas. También se presentan los resultados obtenidos a partir de la experimentación realizada a grupo de expertos con respecto a la utilización de la notación propuesta, asociada a las restricciones de cardinalidad.

Centrándonos en la transformación de los elementos propuestos a un modelo lógico sin pérdida de semántica. La validación también se presenta a través de los resultados obtenidos en la medición de distintos parámetros que afectan al rendimiento de una base de datos objeto-relacional ORACLE 9.i cuando se implementan los procedimientos necesarios para controlar la semántica dentro del SGBD.

Todos estos resultados se presentarán en la sección 6 así como la descripción de cada uno de los experimentos desarrollados.

ANTECEDENTES

Se presentará, en un primer lugar, las características más destacadas de los modelos de datos y su evolución. Daremos una visión general de los modelos de datos convencionales o lógicos que los hemos denominado de primera generación. La segunda generación de modelos serán los modelos de datos semánticos conceptuales, clasificados en las familias más representativos, de los que se estudiará sus constructores y las diferencias existentes entre ellos. Siguiendo la metodología de desarrollo de bases de datos, analizaremos las distintas reglas que se pueden aplicar en la transformación de un esquema conceptual a un esquema lógico. Y por último, presentaremos las herramientas desarrolladas para automatizar las fases de diseño e implementación de una base de datos.

2. MODELOS DE DATOS

La gestión de los datos siempre se ha considerado una importante tarea dentro de una organización, en un principio, se organizaban a fin de atender las necesidades de cada proceso, posteriormente, a un conjunto de procesos. De aquí, que surgieran los sistemas de gestión de ficheros cuya característica principal es la adaptación de los datos a los procesos. Las bases de datos son mucho más ambiciosas, buscan una interpretación de la realidad con el objetivo de representar toda la semántica del mundo real. Por lo tanto, la gestión de los datos es mas compleja y se lleva a cabo a través de los modelos de datos.

Los modelos de datos surgen para dar soporte a la interpretación de un determinado problema o parcela del mundo real. Podemos definir como modelo de datos a un conjunto de conceptos que permiten construir una representación del mundo real. El resultado de su aplicación se denomina *esquema*. Es importante diferenciar el concepto de modelo y esquema porque a veces se utilizan de manera indistinta, cuando el modelo es el instrumento y el esquema es el resultado de aplicar este instrumento. También debemos matizar la diferencia entre mundo real y universo del discurso (UD), ya que este último es la interpretación que realiza el diseñador del mundo real. Por lo que el primer paso para construir una base de datos es definir el UD fijando los objetivos sobre el mundo real que se pretende analizar.

Según De Miguel & Piattini (1993), “un modelo de datos es un conjunto de conceptos, reglas y convenciones que permiten describir los datos del UD”. Las ventajas de la utilización de un modelo de datos son las siguientes:

- ✓ A partir de los modelos de datos se definen los lenguajes de datos. Por ejemplo, el lenguaje SQL se construye añadiendo al modelo relacional una sintaxis.
- ✓ Permiten realizar una formalización de las estructuras permitidas y las restricciones con el fin de representar los datos de un sistema de información.

- ✓ Son una de las herramientas básicas para el desarrollo de una metodología de diseño de bases de datos y facilitan la evaluación del impacto en los cambios de un sistema de información.

Todo modelo de datos (figura 2.1) se caracteriza por una parte estática, estructura de los datos con sus restricciones, y una parte dinámica, que se compone de un conjunto de operadores que se definen sobre las estructuras de los datos. Los lenguajes de definición de datos se corresponden con la parte estática de un modelo mas una sintaxis y los lenguajes de manipulación con la parte dinámica más sintaxis.

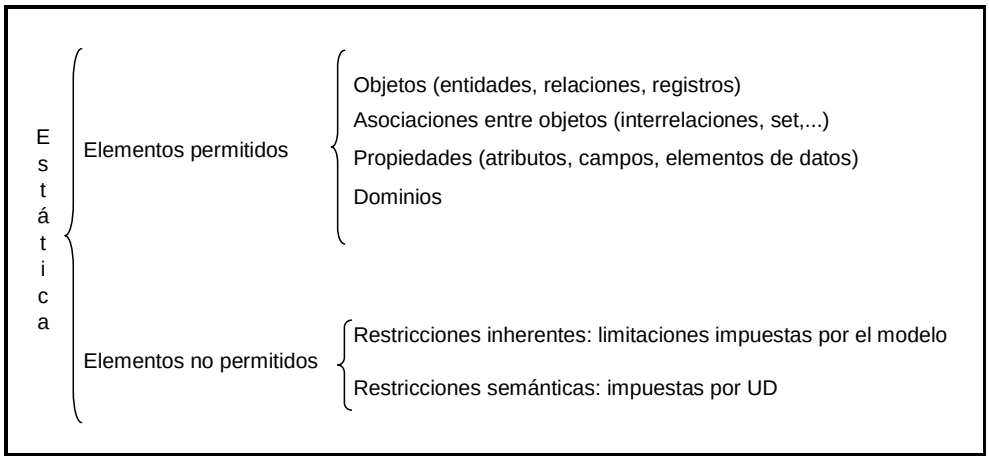


Figura 2.1: Componentes de la parte estática de un modelo de datos

En las siguientes secciones se realizará una exposición de los modelos de datos que mas han transcendido. En la sección 2.1. presentaremos los modelos que hemos denominado *de primera generación*, unidos por la cercanía de sus lenguajes, tanto de definición como de manipulación, a la implementación; creando esquemas muy cercanos a la máquina. Fueron los primeros modelos de datos y caracterizaron los principales SGBD.

Los que hemos denominado *de segunda generación*, sección 2.2., se caracteriza por ser modelos más abstractos para añadir más significado. Hemos decido presentar estos modelos a través de diferentes familias siendo el criterio de esta clasificación el modelo original que han extendido.

2.1. Primera generación: Modelos lógicos

Los modelos lógicos también son conocidos como modelos convencionales. Los tres modelos más importantes y que más relevancia han tenido en es desarrollo de bases de datos son: jerárquico, CODASYL y relacional. Los dos primeros son casos especiales dentro de lo que se denominan modelos en red. Estos modelos y el relacional fueron desarrollados aproximadamente al mismo tiempo pero los SGBD's que implementaban el modelo en red aparecieron antes que los relacionales, lo que sirvió, posteriormente, para que algunas de las características de estos sistemas influyeran en las investigaciones y desarrollos de los productos relacionales comerciales.

La propiedad mas destacable de los modelos de primera generación se encuentra en que suelen estar implementados en SGBD comerciales, pero poseen poca capacidad semántica y son dependientes del SGBD.

Vamos a describir cada uno de estos modelos según el orden de aparición y veamos las aportaciones mas importantes de cada uno de ellos.

2.1.1. Características del modelo jerárquico

El primer problema con el que se enfrentó la tecnología de las bases de datos fue poder representar las relaciones entre datos que anteriormente con la tarjetas perforadas y las cintas magnéticas no se podían reflejar. Con la aparición de los discos magnéticos que proporcionaban acceso aleatorio a los datos, se pudieron crear punteros embebidos en registros, y a través de estos relacionar los datos. Nos hemos remontado a los años 60 donde IBM creo IMS (MacGee, 1977), el primer gran modelo fuera de las consideraciones físicas de almacenamiento y uno de más destacados modelos jerárquicos implementados.

El modelo jerárquico se caracteriza por estructurarse como un árbol cuyos nodos son entidades u objetos que se enlazan a través de arcos, representando las interrelaciones o asociaciones entre los nodos.

El modelo jerárquico se compone principalmente de:

- ✓ Un conjunto de registros denominados *nodos* del grafo.
- ✓ Un conjunto de interrelaciones o asociaciones no nominadas que asocian dos nodos y que se denominan *arcos* del grafo.

Las restricciones inherentes vienen impuestas por la propia estructura de árbol; un nodo padre puede tener un número ilimitado de hijos, sin embargo un hijo solo puede tener un padre. Al ser las asociaciones no nominadas un registro solo puede relacionarse con otro registro distinto. Y no se pueden representar asociaciones *que relacionen una ocurrencia del nodo padre con cero, una o varias ocurrencias del hijo.* Además de estas restricciones impuestas por el modelo, debemos subrayar que no se pueden definir restricciones de usuario.

Esta rigidez repercute en que para representar ciertas estructuras que se dan habitualmente en el mundo real, por ejemplo, las asociaciones denominadas comúnmente *muchos a muchos* (un padre con varios hijos, un hijo con varios padres), se simulen a través de la introducción de redundancias lógicas. De esta forma el modelo se adapta a situaciones del mundo real que no responden a una jerarquía a cambio de la pérdida de eficiencia. Las desventajas son claras, la poca flexibilidad que ofrece, la redundancia lógica que provoca esta rigidez y la dependencia entre el nivel lógico y físico en la realización de operaciones de selección y actualización, donde debe controlarse perfectamente cómo se encuentran enlazados los registros.

La mayor ventaja de estos sistemas es que si el UD a representar sigue una estructura jerárquica este modelo se ajusta perfectamente y además es muy eficiente, en especial el modelo IMS. Por eso ha sido y es utilizado en la aplicaciones de gestión bancaria.

2.1.2. Características del modelo CODASYL

El modelo CODASYL intenta mejorar la rigidez del modelo jerárquico permitiendo en primer lugar que un nodo pueda tener varios padres, ampliando el espectro de aplicaciones del mundo real que pueden ser representadas con este modelo.

Aunque realmente por lo que destaca es por la creación de lenguajes para definir y manipular los datos. En el año 1973 publicó su primer informe, donde introdujo un importante numero de innovaciones en los SGBD. Este esfuerzo del comité CODASYL fue motivado por preservar la supremacía del lenguaje de programación COBOL y por ello reunió un grupo de trabajo de bases de datos (DBTG). La principal innovación de este informe surgió de la creación de tres lenguajes para la definición de los datos a diferentes niveles. El lenguaje de definición (DDL) de esquemas, que permite diseñar la base de datos a través de un lenguaje muy similar al de definición de datos de COBOL. El lenguaje de definición de subesquemas, con el que se describen subconjuntos de la BD global. Ambos lenguajes contemplan la posibilidad de establecer restricciones en las asociaciones entre nodos, por ejemplo, indicar al insertar un registro en la BD si este debe estar relacionado con un registro de otro tipo o si al borrar un registro qué ocurre con el registro que se relaciona con él. Y por último, el lenguaje de manipulación de datos, cuya ventaja radica en la independencia de la conexión mediante punteros de los registros por los que se desea navegar. Este lenguaje en comparación con el jerárquico es mucho mas abstracto aunque sigue siendo un lenguaje navegacional donde se necesita primero localizar la ocurrencia o registro que se quiere actualizar y luego realizar la operación sobre la misma. Este tipo de lenguajes van muy unidos a la estructura especifica de cada esquema y por tanto son poco adecuados para los usuarios finales.

El comité CODASYL fue el primero en definir lenguajes para los tres niveles de abstracción propuestos por ANSI/SPARC, subcomité creado por ISO/IOC para el estudio de los aspectos más relevantes de los SGBD que podrían ser susceptibles de ser estandarizados. Aunque la estandarización nunca se llevó a cabo, si se propuso la arquitectura a tres niveles para los sistemas gestores con el objetivo de conseguir mas independencia en los datos. Antes de la disolución de este comité en 1983 se publicó el último informe incluyendo lenguajes para cada uno de estos tres niveles.

Podríamos describir el modelo CODASYL como una serie de registros, *records*, que pueden ser relacionados a través de los conjuntos (*set*). Un registro está compuesto de items o agregados de datos (vectores o grupos repetitivos). Los conjuntos son

colecciones nominadas de dos o más registros que establece una vinculación entre ellos. El *set* crea asociaciones jerárquicas a dos niveles, en las cuales el nodo raíz se denomina propietario (*owner*) y los descendientes, miembros (*member*).

Las restricciones inherentes imponen la estructura del modelo que debe ser jerárquica a dos niveles por lo que el nivel propietario solo puede existir un tipo de registro. No permiten que un mismo tipo de registro sea propietario y miembro en un mismo conjunto y que una misma ocurrencia de miembro pertenezca en un mismo conjunto a más de un propietario.

Las restricciones de usuario o semánticas son reflejadas a través de las definiciones estructurales de los registros y de los conjuntos.

En comparación con el modelo jerárquico, este modelo permite definir distintos conjuntos entre los mismos miembros ya que las asociaciones son nominadas, pueden existir distintos propietarios para un mismo miembro, aunque en conjuntos diferentes, registros aislados, es decir, registros que no sean propietarios ni miembros de un conjunto y registros opcionales que son aquellos cuya ocurrencias pueden no estar relacionadas con el propietario de un conjunto.

Estas propiedades flexibilizan el modelo pero realmente lo que le caracteriza es la potencia de sus lenguajes, adaptándose a la arquitectura a tres niveles que le aporta independencia físico/lógica a los datos.

2.1.3. Características del modelo relacional

El modelo relacional tardó en ser implementado porque en un principio no existía soporte hardware que le hiciera competitivo frente a los modelos jerárquicos o en red, mucho más fáciles de implementar y más eficientes al adaptarse a la tecnología existente.

Cuando Codd en 1970 (Codd, 1970) presentó el modelo relacional resultó revolucionario por su dimensión formal, basada en la teoría de conjuntos y por la

constante referencia a la importancia de conseguir la independencia físico - lógica de los datos propuesta en la arquitectura a tres niveles de ANSI/SPARC.

El elemento fundamental de este modelo es la *relación* la cual se puede representar en forma de tabla y en ella podemos distinguir, las columnas, denominadas *atributos*, que representan las propiedades de la relación y que se caracterizan por un nombre. Las filas de las relación se llaman *tuplas* y son las ocurrencias o instancias de la misma.

Otro elemento importante son los dominios, de los cuales los atributos toman valor. Formalmente, podemos definir un dominio como un conjunto de valores homogéneos y atómicos caracterizados por un nombre. Los dominios pueden definirse por intensión o por extensión, en el primero de ellos se debe especificar un tipo de datos abstracto similar a los definidos en los lenguajes de programación, para el segundo se tiene que especificar el conjunto de valores que puede tomar o lo que se denomina en los lenguajes de programación tipos enumerados de datos.

Un atributo A es el papel que tiene un determinado dominio D en una relación; se dice que D es el dominio de A y se denota por dom(A).

Un esquema de relación R, denotado por $R(A_1{:}D_1, A_2{:}D_2,\ldots,A_n{:}D_n)$ se compone de un nombre, R, y un conjunto de atributos definidos sobre un dominio. Los nombres de los atributos que pertenecen a R son únicos sin embargo los dominios no necesariamente. Es decir, puede haber distintos atributos definidos sobre el mismo dominio.

Una ocurrencia de relación o también denominada *extensión de relación*, denotada por r(R) es un conjunto de tuplas $\{t_1,..t_m\}$ donde cada una de ellas se compone de n pares atributo valor $\{(A_i{:}V_{ij})\}$; donde V_{ij} es el valor j del dominio D_i asignado al atributo A_i. El número de tuplas pertenecientes a una relación se denomina *cardinalidad* y el número de atributos *grado*.

También podemos definir la extensión de una relación R como una relación matemática de grado n tal que:

$$r(R) \subseteq dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$$

Las características de una relación que la distinguen de una tabla y de un fichero son las siguientes:

- ✓ No puede haber filas repetidas, es decir, las tuplas deben ser distintas.
- ✓ El orden de las tuplas es irrelevante.
- ✓ La relación es plana por lo que en el cruce de una fila con una columna solo puede haber un valor.

Además de estas restricciones inherentes al modelo existe la de *integridad de entidad*, la cual establece que ningún atributo que forme parte de la clave primaria puede tomar un valor nulo. Por la propia definición de relación siempre existe al menos una clave primaria pero debemos considerar aquel conjunto de atributos que identifiquen de forma unívoca y mínima las tuplas pertenecientes a la relación. Por tanto, pueden existir distintos candidatos a clave primaria y será decisión del diseñador elegir una u otra.

La primera restricción de usuario será la elección de la clave primaria y las claves alternativas de cada relación ya que lo que se considera inherente al modelo relacional es la obligatoriedad de la clave primaria.

La segunda restricción de usuario es la de *integridad referencial* que expresa si una relación R_2 (relación que referencia) tiene un descriptor que es clave candidata de R_1 (relación referenciada), todo valor de dicho descriptor debe coincidir con algún valor de la clave candidata a la que hace referencia o ser nulo. R_1 y R_2 no son necesariamente distintas y el descriptor se denomina *clave ajena.*

La restricción de clave ajena es una restricción de comportamiento y describe asociaciones entre las relaciones para reflejar la semántica del mundo real. Además de la definición de clave ajena también se debe especificar las consecuencias que pueden tener las operaciones de borrado y modificación en una clave candidata sobre los valores de la clave ajena que la referencian. Las opciones a las que se puede optar son las siguientes: operación restringida, en cascada, con puesta a nulos, con puesta a valor

por defecto o definida por el usuario. Toda clave ajena tiene por defecto la operación de borrado y modificación restringida.

La importancia del modelo relacional no solo destaca por su definición formal apoyada en la teoría de conjuntos sino que además, a partir de este modelo se construyeron distintos lenguajes que al final desembocaron en el SQL (Structure Query Language). Lenguaje que fue norma ISO en 1992, aunque cada SGBD comercial le imprime ciertas variantes.

A diferencia de los modelos anteriormente descritos, el lenguaje SQL posee la característica de ser un lenguaje de especificación, es decir, se dice qué hacer pero no cómo hacerlo, de esto se encarga el SGBD. Además, este lenguaje sirve para especificar tanto la parte estática del modelo relacional como la parte dinámica. Y puede ser autocontenido o definirse dentro de cualquier lenguaje de programación, siendo un lenguaje huésped. Por último, se debe destacar que permite definir los elementos relacionales en los tres niveles propuestos por ANSI/SPARC, caracterizando a los SGBD que hasta ese momento solo eran de tipo jerárquico.

Este modelo en comparación con los dos modelos anteriores propone otra filosofía de ver los objetos y las relaciones entre ellos. La asociaciones que se permiten son de todo tipo, no tiene porque ser jerárquicas ni sobre distintos elementos. El único tipo de asociación que no se representa de forma natural en la de *muchos a muchos*, lo que se tendrá que simular añadiendo redundancias lógicas.

El modelo presentado por Codd en 1970 fue ampliándose con más restricciones de usuario (Codd, 1979) enriqueciendo la semántica del mismo. Además se construyeron metodologías apoyadas en la teoría de la normalización para garantizar un buen diseño sin redundancias ni pérdidas de información. En la sección 2.4. se explicarán las metodologías y los métodos aplicados para el diseño de bases de datos relacionales.

2.2. Segunda generación: modelos semánticos

Los SGBD comerciales utilizados en la década de los 50 fueron los modelos jerárquicos y en red hasta que en los 70 apareció el modelo relacional. Esto supuso una revolución en las bases de datos ya que este modelo separaba la representación lógica de los datos de su representación física. Esto permitía un potente desarrollo, la utilización de un lenguaje de definición y manipulación no procedimental y una gran variedad de resultados teóricos para realizar un buen diseño de bases de datos.

La historia acerca de las investigaciones desarrolladas entorno a los modelos semánticos es algo distinta ya que fueron primeramente introducidos como herramientas de diseño de esquemas: un esquema en primer lugar se diseñaba en un modelo semántico de alto nivel y posteriormente era trasladado a un modelo tradicional (jerárquico, en red o relacional) para su implementación. El énfasis inicial de los modelos semánticos fue representar con precisión las típicas aplicaciones de bases de datos mediante la relación existente entre los datos. En consecuencia, se trata de modelos más complejos y que dan una visión de los datos mucho mas estructurada. Estos modelos han ido evolucionando y ya no solo se conciben como herramientas de diseño sino que se consideran importantes para gestión de base de datos, que facilitan el desarrollo de interfaces de usuario, que mejoran la reutilización y que se adaptan mejor a los cambios del sistema de bases de datos.

La primera publicación acerca de modelos semánticos aparece en 1974 (Abriel, 1974), durante la subsecuente década se especifican importantes modelos y una larga lista de publicaciones acerca de los esfuerzos relativos a esta investigación. Los resultados se centraban en el desarrollo de mecanismos potentes para representar aspectos estructurales sobre los datos de negocio. En la actualidad, la atención se dirige hacia la incorporación de aspectos de comportamiento de los datos en los formalismos de estos modelos, para conseguir recoger toda la información necesaria del Universo del Discurso (UD) a representar, a través de los elementos del modelo.

Existe una analogía entre lo que hay detrás de los modelos semánticos y los lenguajes de programación de alto nivel. Estos últimos fueron desarrollados para tratar de proporcionar más riqueza y mayor nivel de abstracción. Similarmente, los modelos de datos semánticos (Chen, (1976), Smith and Smith, (1977), Hammer and McLeod,(1981)) tratan de proporcionar más potencia de abstracción en la especificación de esquemas de bases de datos para que luego sean soportados por un SGBD relacional, jerárquico o CODASYL.

Los modelos lógicos o convencionales manejan estructuras de datos que se acercan bastante a la representación física de los datos en un ordenador, son vistas de registros con punteros a los valores de cada campo y por eso a menudo se les conoce como basados en registros. Por el contrario, los modelos semánticos se presentan como una herramienta para el diseñador que le servirá para representar los datos de un determinado UD.

La utilización de modelos semánticos en la fase conceptual de una metodología de desarrollo de BD produce las siguientes ventajas:

- ✓ Incrementa la separación de los componentes lógicos y físicos de los datos.
- ✓ Con los modelos orientados a registro existen constructores semánticamente solapados en el sentido que diferentes tipos de asociaciones deben ser representados utilizando el mismo constructor. Por lo que uno de los primeros objetivos de los modelos semánticos sería proporcionar una familia coherente de constructores para la representación de estructuras que en los modelos de primera generación solo podrían representarse a través de restricciones o con la introducción de redundancias lógicos que simularan dicha situación.
- ✓ Mecanismos de abstracción. El nivel de abstracción proporcionado por estos modelos provoca que aumente el nivel de detalle en el cual se puede ver una porción del esquema. Otra dimensión de abstracción se plantea en términos de modularidad ya que los esquemas semánticos facilitan el aislamiento de información y por lo tanto, su reutilización en otros esquemas.

Lo que se propone es que los modelos semánticos nos sirvan para recoger las especificaciones de la base de datos contribuyendo tanto al diseño como a la evolución de esquemas. Por eso se incorporan a las metodologías de diseño proporcionando más facilidad para realizar la integración y formalización de la fase de análisis de requisitos.

En general, existen dos aproximaciones en los modelos de datos semánticos: una de ellas muestra a través del mismo constructor las relaciones entre objetos y sus propiedades. La otra vertiente promueve la utilización de un constructor específico para diferenciar la representación de las interrelaciones y de los atributos. Estas dos vertientes desembocan en dos filosofías que caracterizan los modelos semánticos. Los modelos que se denominan FDM (modelo de datos funcional) y que utilizaremos si deseamos emplear un lenguaje funcional. En la otra vertiente se encuentran la familia de modelos que provienen del modelo ER (Chen, 1976) y la familia de modelos de datos orientados a objetos.

A lo largo de este capítulo describiremos los modelos semánticos más representativos en cada una de estas dos vertientes (figura 2.3). La familia Object-Role, donde el modelo más destacado es ORM (Halpin 2001), que pertenece a la parte de modelos de datos donde las relaciones entre objetos y sus propiedades se representan a través del mismo constructor; y por la parte de los modelos que proponen un constructor para representar relaciones y atributos, presentaremos dos familias. La primera es un conjunto de modelos que son propuestas de extensiones al modelo ER de P. Chen, (1979) y la segunda, que contiene las características principales de los modelos de datos orientados a objetos, donde su máximo exponente es el lenguaje UML (OMG, 2000), estándar en el diseño orientado a objetos.

Nos gustaría hacer mención de una de las diferencias observadas al realizar el estudio de las distintas familias. Tanto en la familia Object-Role (OR) como en la de Orientación a Objetos (OO) las distintas propuestas han ido evolucionando a un único modelo, en el primer caso a ORM, y en el segundo a un lenguaje, UML. A diferencia de la familia ER que tiene diversas extensiones, pero no desembocan en un modelo único que unifique las mismas.

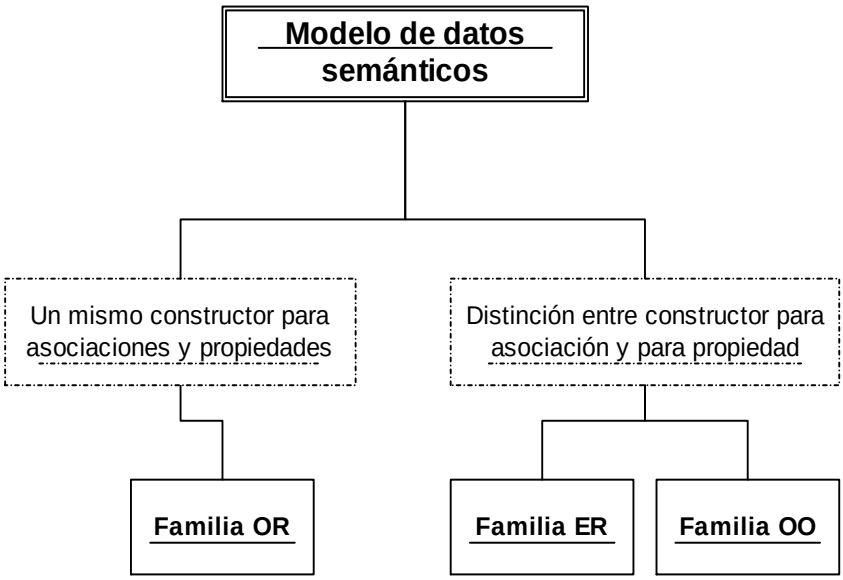


Figura 2.3: Clasificación de modelos semánticas

2.2.1. Familia Entidad/Interrelación

El modelo entidad-interrelación (ER) propuesto por Peter Chen en 1976 (Chen, 1976) fue uno de los primeros modelos de datos semánticos y la base para un gran conjunto de modelos que le han extendido. Esta proliferación de modelos viene dada por la sencillez del modelo ER, su facilidad de aprendizaje y de uso y su cercanía al mundo real, facilitando la comunicación con los expertos del dominio para validar la solución.

Según la clasificación anterior, una de sus características es la utilización del constructor *interrelación* para representan las relaciones entre objetos.

Antes de comentar las principales extensiones que se han desarrollado, analizaremos la propuesta que realizó Chen en 1976. En este primer artículo de Chen y por ende del modelo ER, destaca cuatro niveles en los que podemos ver los datos. El primero, y más abstracto, representa las entidades e interrelaciones que existen en el mundo real. En los tres restantes se estudia la organización de la información, su estructura y su representación a nivel físico. Con esta especialización en la definición de los datos se puede establecer las semejanzas con otros modelos y la equivalencia entre ellos, figura 2.4.

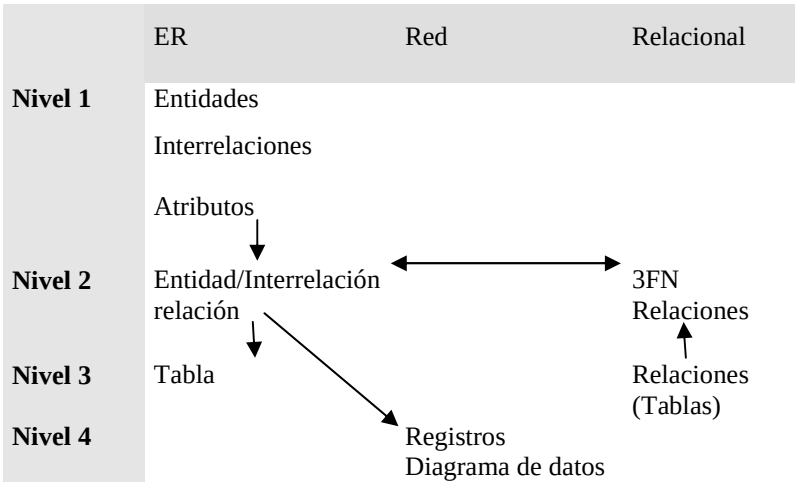


Figura 2.4: Distintos niveles de definición de datos y su representación en cada modelo de datos.

La inclusión de estos niveles se traduce al utilización de distintas fases para desarrollar una bases de datos. Hasta ese momento los modelos de datos convencionales comenzaban el desarrollo a un nivel cercano a la implementación aunque ya aseguraban la independencia físico /lógica. Al añadir un primer nivel, más abstracto, enuncia los elementos de los que se compone y el lenguaje para definirlos pero también cómo pasar de un nivel a otro. Lo que podemos traducir en que ya en este primer artículo esboza una metodología de desarrollo de bases de datos, tanto para bases de datos relacionales que posteriormente amplían Teorey, Yang & Fry (1986), tanto como para las bases de datos en red. A continuación pasaremos a esbozar las características de cada uno de estos tres niveles.

Nivel 1: Elementos del modelo ER

En el primer nivel, los elementos que se consideran son las entidades e interrelaciones. Las *entidades* son "objetos" que pueden ser claramente identificables. Una persona, compañía o evento puede ser un ejemplo de entidad. Se distingue entre *entidad* y *tipo de entidad*, donde toda entidad pertenece a un tipo de entidad. Por lo que las entidades de un esquema están clasificadas en diferentes tipos de entidades. Además, una entidad puede pertenecer a varios tipos de entidad. Por ejemplo, la entidad perteneciente al tipo

de entidad EMPLEADO puede pertenecer también al tipo de entidad PERSONA, por lo que EMPLEADO es un subconjunto de PERSONA.

Un tipo de *interrelación* es una asociación entre tipos de entidades. Un tipo de interrelación, R_i, se compone de un conjunto de tuplas denominadas interrelaciones y que se forman por una relación matemática entre n entidades, cada una de ellas tomadas de un tipo de entidades, $\{[e_1, e_2,\ldots., e_n] \mid e_1 \in E_1 \ldots e_n \in E_n\}$ donde los E_i no son necesariamente distintos.

El role de una entidad en una interrelación es la función que esta desarrolla en la interrelación. Por ejemplo, *matrimonio* es una interrelación entre dos entidades del tipo de entidad PERSONA, cada una de ellas participa con un role distinto, *marido* y *mujer*.

Si incluimos el role en la definición del tipo de interrelación no sería necesario el orden dentro de la interrelación y por tanto, una tupla sería $(r_1/e_1, r_2/e_2, \ldots, r_n/e_n)$ donde r_i es el role de e_i en la interrelación.

Los atributos pertenecen a un tipo de entidad o interrelación y representan la información que se quiere incorporar de las mismas. Esta información es expresada mediante pares atributo-valor. Y los valores pueden agruparse en conjuntos de valores como por ejemplo COLORES, NOMBRES, etc. Un atributo puede definirse formalmente como una función que hace corresponder a un conjunto de entidades o interrelaciones un conjunto de valores o el producto cartesiano de un conjunto de valores:

$$F: E_i \text{ o } R_i \rightarrow V_i \text{ o } V_1 x \ldots x V_n$$

Se destaca la diferencia entre atributos y conjunto de valores, que en la mayoría de los modelos no se considera y la importancia de los atributos en las interrelaciones ya que determinan las dependencias funcionales entre los datos.

El lenguaje de representación propuesto por Chen para la vista de los datos a Nivel 1, es decir, a nivel más abstracto, se denomina diagrama Entidad/Interrelación en la que se dibujarán todos los elementos del modelo expuestos anteriormente. Los tipos de entidad

se representarán a través de un rectángulo y los tipos de interrelación por un rombo del cual salen líneas que conectan los tipos de entidades que participan en dicha interrelación.

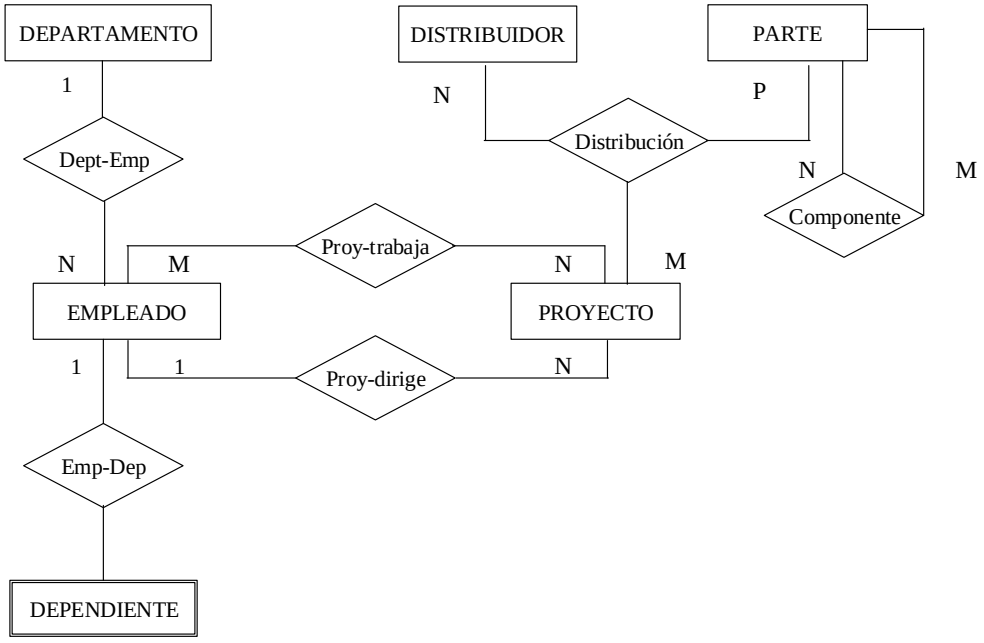


Figura 2.5: Ejemplo del diagrama ER, Chen (1974)

Los elementos que podemos representar, cómo representarlos y la semántica asociada a los mismos se explican a través del ejemplo original de Chen (figura 2.5).

1. Un tipo de interrelación puede asociar más de dos tipos de entidad. Por ejemplo, SUMINISTRADOR- PROYECTO- PIEZA.

2. Un tipo de interrelación puede estar definido sobre un solo tipo de entidad. Por ejemplo el tipo de interrelación COMPONENTE definido sobre el tipo de entidad PART.

3. Pueden existir diferentes tipos de interrelación definidos sobre el mismo conjunto de tipos de entidad. Por ejemplo, los tipos de interrelación PROYECTO-TRABAJA y PROYECTO- DIRIGE definidas sobre los tipos de entidad PROYECTO y EMPLEADO.

4. En la figura 2.5 se distinguen tres tipos de correspondencia o “mappings”: 1:n, m:n y 1:1. El tipo de interrelación DEPARTAMENTO-EMPLEADO cuya correspondencia es 1:n refleja que un departamento puede tener n (n = 0,1,2…) empleados y cada empleado trabaja para un solo departamento. El tipo de interrelación PROYECTO-TRABAJA es una correspondencia n:m que indica, cada proyecto puede tener cero, uno o más empleados asignados y cada empleado puede ser asignado a cero, uno o más proyectos. También es posible representar correspondencias 1:1 como en el tipo de interrelación Matrimonio. Por lo que define el tipo de correspondencia como el número de entidades en cada tipo de entidad que es permitido en un tipo de interrelación. Además indica que esta propiedad es característica del modelo ya que no es considerada ni en el modelo relacional ni en el modelo en red.

5. El diagrama expresa la dependencia en existencia de un tipo de entidad con respecto a otra. Por ejemplo, la flecha en el tipo de interrelación EMPLEADO-DEPENDER indica que la existencia de una entidad en el tipo de entidad DEPENDER depende de que se relacione con una entidad del tipo de entidad EMPLEADO. Es decir, si un empleado abandona la compañía, sus dependientes no serán de interés para la misma.

Lo que propone en este primer nivel es realizar la identificación de los tipos de entidad e interrelación y recoger la semántica de estas últimas a través sus propiedades: roles, grado (número de tipos de entidad que participan en el tipo interrelación) y multiplicidad. Posteriormente Chen publica una serie de artículos (Chen, 1977; Chen 1983) donde presenta una serie de heurísticas para que la identificación de los tipos de entidad e interrelación a través de sentencias en inglés. De forma resumida, indica, que en general, un nombre puede ser un tipo de entidad y un verbo, un tipo de interrelación.

Nivel 2: Definición de entidad/interrelación relación

En el segundo nivel se propone la estructura de la información recogida en el nivel anterior. Para las entidades siempre ha de existir un atributo o grupo de atributos que

identifique cada entidad. Básicamente, define como *clave de entidad* a un grupo de atributos si la correspondencia entre el conjunto de entidades y su grupo de conjunto de valores es uno a uno, o biyectiva. Si no existe este grupo de atributos o por motivos de simplificación de clave, permite crear un atributo artificial que realice la función de clave de entidad. También puede ocurrir que existan varios grupos de atributos que cumplan con este tipo de correspondencia biyectiva por lo que existen varias *claves de entidad.* En este caso se debe elegir una de ellas y para distinguirla de las demás se la denomina *clave primaria de la entidad.*

En este segundo nivel se hace corresponder a un tipo de entidad con una *entidad relación*. Debemos destacar que una entidad relación es una representación en forma de tabla de un tipo de entidad y que no se trata de un relación del modelo relacional ya que contiene mas semántica. Esto se traduce en que un tipo de entidad se convierte en una tabla donde cada fila, denominada *tupla*, está identificada por la clave primaria.

Una *interrelación relación* se identifica por los tipos de entidad que participan en ella, su clave primaria, por tanto, será representada por las claves primarias de las entidades que asocia y al igual que en las entidades estas pueden representarse mediante tablas cuyas filas se denominan *tuplas*.

En algunos casos, las entidades de un tipo de entidad no pueden identificarse unívocamente por los valores de sus propios atributos. Si utilizamos una interrelación o interrelaciones para identificarla mediante otras entidades que participan en esta interrelación esta entidad se denomina *entidad relación débil*, en caso contrario, tratamos con *entidades relación regulares*. Aunque teóricamente cualquier tipo de interrelación podría servir para identificar entidades, el modelo las restringe a interrelaciones binarias con correspondencia 1:n, en donde la existencia de n entidades de uno de los tipos de entidad participantes en la interrelación, depende de una entidad del otro tipo de entidad. Toda interrelación relación compuesta por entidades relación regulares se denominan *interrelación relación regular.*

Esta distinción entre entidad/interrelación regular y entidad/interrelación débil podrá ser utilizada para mantener la integridad de la base de datos a la hora de realizar cualquier operación de modificación.

Realmente en este segundo nivel se definen algunas de las restricciones del modelo ER. Como restricción inherente la obligatoriedad de definir una clave primaria de entidad, al igual que en el modelo relacional. El conjunto de claves de entidad, la clasificación de entidades o interrelaciones relación regulares o débiles así como la definición de clave primaria para una interrelación relación, forman parte de las restricciones semánticas del modelo.

Nivel 3: Definición de atributos y valores

La etapa 3 se correspondería con la definición a través de un lenguaje de todos los conjuntos de valores y atributos. Un ejemplo de la declaración de estos atributos y sus valores se muestran a continuación:

DECLARE	VALUES-SET	REPRESENTATION	ALLOWABLES
	N-Empleado	Integer (4)	(0,2000)
	Nombre	Char (8)	Todos
	Apellidos	Char(30)	Todos
	Edad	Integer (3)	(0,100)
	N-Proyecto	Integer (3)	(1,500)

Tabla 2.1: Definición de conjuntos de valores

Chen en su artículo no especifica el lenguaje a utilizar para la declaración de atributos y valores. Aunque como se puede apreciar en el ejemplo podría ser cualquier lenguaje de programación.

Nivel 4: Definición de las tablas

Para la etapa 4 se definirían las entidades y las interrelaciones en modo relación.

DECLARE REGULAR ENTITY RELATION EMPLEADO

ATRIBUTTE / VALUE-SET

N-Empleado / N-Empleado

Nombre Completo / (Nombre, Apellidos)

Nombre alternativo / (Nombre, Apellidos)

Edad / Edad

PRIMARY KEY

N-Empleado

DECLARE REGULAR ENTITY RELATION PROYECTO

ATRIBUTTE / VALUES

N-Proyecto / N-Proyecto

PRIMARY KEY

N-Proyecto

DECLARE REGULAR RELATIOSHIP RELATION PROYECTO-TRABAJA

ROLE / ENTITY-RELATION.PK / MAX-NO-OF-ENTITIES

TRABAJA / EMPLEADO.PK / m

PROYECTO / PROYECTO.PK / n

ATRIBUTTE / VALUES

PORCENTAGE-TIEMPO / PORCENTAGE

Tabla 2.2: Definición de tipos de entidad e interrelación

Además de esta definición de tipos de entidad e interrelación, Chen índica una serie de restricciones para mantener la integridad de los datos aunque resalta que incluso el modelo en sí mismo representa muchas de estas restricciones.

Los tres principales tipos de restricción sobre valores son especialmente tratados en este primer artículo de Chen y los describimos a continuación.

- ✓ Valores permitidos para un conjunto de valores. Como por ejemplo, los presentados en la tabla 2.1.

- ✓ Valores permitidos para un cierto atributo. Por ejemplo para indicar que la edad de un empleado se corresponde al intervalo entre los 18 y los 65 años. Es decir, propiedades características de un determinado atributo para un tipo de entidad.

  $EDAD(e) \in (18,65)$ donde $e \in$ EMPLEADO

- ✓ Valores existentes en la base de datos, los cuales se dividen en dos: representa la relación existente entre conjuntos de valores definidos y restricciones entre valores particulares. Un ejemplo de la primera restricción podría indicar que la nombres de las mujeres son un subconjunto de los nombres de personas. Y para la segunda podríamos indicar que los impuestos siempre son menores que el salario de un determinado empleado.

  $\{NOMBRE\ (e) \mid e \in MUJER\text{-}PERSONA\} \subseteq \{NOMBRE\ (e) \mid e \in PERSONA\}$

  $IMPUESTO\ (e) < SALARIO\ (e)$

Dinámica del modelo y reglas de consistencia

Otra consideración importante es el mantenimiento de la consistencia de la BD cuando se realiza una operación de actualización. Chen por tanto, además de definir la parte estática del modelo ER define la dinámica de este modelo. No específica la sintaxis, pero realiza una distinción entre recuperación y actualización y también del nivel en el que se quiera realizar dichas operaciones.

Para la operación de recuperación, en el nivel 1 se utiliza un lenguaje muy parecido al lenguaje natural, sin embargo, en el caso de que estemos en el segundo nivel considera la posibilidad de utilizar SEQUEL (Chamberlin & Raymond, 1974).

Para la inserción, modificación y borrado presenta las reglas que se deben mantener para asegurar la consistencia de la BD tanto para el primer nivel 1 como para el segundo (tablas 2.3,2.4,2.5).

Nivel 1	Nivel 2
Operación Insertar una entidad en un tipo de entidad	Operación Crear una tupla entidad con una PK Check Si la PK existe o si es admisible
Operación Insertar una interrelación en un tipo de interrelación Check Si las entidades existen	Operación Crear una tupla interrelación con PK de las entidades Check Si las PK’s existen
Operaciones Insertar propiedades en una entidad o interrelación Check Si los valores son admisibles	Operación Inserción de valores en una tupla de entidad o interrelació Check Si los valores son admisibles

Tabla 2.3: Reglas para el mantenimiento de consistencia en la operación de inserción.

Nivel 1	Nivel 2
Operación Cambiar el valor del atributo de una entidad	Operación Modificar un valor Check

	Si el atributo no es PK no hay Check Si el atributo es parte de la PK cambiar todas las interrelaciones en donde aparezca este valor. Cambiar los PK's de las entidades que utilizan este valor como parte de su PK
Operación Cambiar el valor del atributo de una interrelación	Operación Modificar el valor

Tabla 2.4: Reglas para el mantenimiento de consistencia en la operación de modificación.

Nivel 1	Nivel 2
Operación Borrado de una entidad Consecuencias Borrado de las entidades que dependan de esta Borrado de las interrelaciones en las que participe Borrado de todas sus propiedades	Operación Borrado de una tupla de entidad Consecuencia (aplicada recurrentemente) Borrar toda tupla de entidad que dependa de esta entidad Borrar las tuplas de interrelación asociada con esta entidad.
Operación Borrado de una interrelación Consecuencias Borrado de todas sus propiedades	Operación Borrar la tupla de interrelación

Tabla 2.5: Reglas para el mantenimiento de consistencia en la operación de borrado.

Las reglas presentadas solo reflejan la semántica asociada a las operaciones de manipulación. Con esto añade restricciones dinámicas al modelo, es decir, indica cómo pasar de un estado a otro de la base de datos sin dejar inconsistentes los datos.

La parte dinámica del modelo ER no llegó a cuajar ya que no se ha implementado en un SGBD. Por lo que el modelo en la actualidad se utiliza en la fase conceptual dentro de muchas de las metodologías de BD y sirve para especificar los requisitos del UD. A continuación estudiaremos las extensiones que se han realizado.

Extensiones al modelo Entidad/Interrelación

Las extensiones del modelo ER vienen motivadas por la evolución de los modelos de datos semánticos, un resumen de las principales características de estos modelos aparece

en Hull & King (1987). En esta sección comentaremos las más importantes pero en la sección 2.3.3 discutiremos acerca de diferentes aproximaciones dadas para la restricción de cardinalidad en todas las familias y la ambigüedad encontradas, elemento sobre el que se basa la propuesta de esta tesis.

Profundizando en el concepto de entidad

El modelo ECR (entity-category-relationship) fue introducido por Elmasri, Weeldreyer y Hevner (Elmasri, Weeldreyer & Hevner, 1985) para extender el modelo ER. En este trabajo presentaremos esta extensión basándonos en (Elmasri & Navathe, 2000), su última edición, donde podemos destacar para el tipo de entidad, la distinción que realiza entre una entidad con existencia física (persona, empleado, etc..) y existencia abstracta (negocio, puesto de trabajo, etc..) para diferenciar y aclarar el tipo de información que podemos guardar dentro de un tipo de entidad. A las entidades pertenecientes a un tipo de entidad las denomina *ocurrencias* aunque también pueden ser denominadas *instancias,*Teorey (1990).

A pesar de que el concepto de entidad es ampliamente utilizado y aceptado, no existe un acuerdo para realizar una definición común, así por ejemplo, Thalheim (2000), recoge doce diferentes acepciones. Sin embargo, aunque los expertos no sean capaces de acordar una única definición, lo que se pretende definir es coincidente en todos los casos, por lo que no se plantean grandes dudas acerca de lo que los diseñadores entienden habitualmente por entidad y su utilización como elemento de diseño no supone grandes inconvenientes. Tan solo subrayaremos de acuerdo con Thalheim (2000), que una entidad[2] es una abstracción de representación a efectos de modelado.

La propuesta de Thalheim (2000) es mucho mas ambiciosa que otras extensiones, por eso la nombramos especialmente, al proponer una base matemática para construir esquemas ER semánticamente bien formados. Con ello muestra métodos para realizar integración de vistas, generación de consultas, equivalencia de esquemas y transformación de esquemas. Además el modelo HERM (Higher-order entity-

[2] En este caso al hablar de entidad, Thalheim se refiere a tipo de entidad.

relationship) basa su representación en el modelo ER pero incluye comportamiento a los constructores; por lo que Thalheim define el marco de su modelo dentro de la familia ER y de la familia OO. A continuación presentamos la definición de tipo de entidad presentada en HERM.

Definición tipo de entidad :

Un tipo de entidad E es un par E = (attr(E), $\{id_j (E) \mid 1 \leq j \leq m \}$) donde attr(E) es un conjunto de atributos y $\{id_j (E) \mid 1 \leq j \leq m \}$ es el conjunto de m claves que posee E.

Para aclarar esta definición presentamos el siguiente ejemplo:

> Una persona se puede caracterizar por el siguiente conjunto de atributos y se identifica por NSS:
>
> Persona = ({SSN, Nombre (Nombre, Apellidos), Dirección (CP, Ciudad, Calle (Nombre, Número)), Género},{SSN})

Todas las extensiones del modelo ER coinciden en la representación del tipo de entidad según la propuso Chen en su primer artículo, al igual que para la representación de tipos de entidad débil. Que recordamos es a través de un rectángulo y un doble rectángulo respectivamente.

Tipo de atributos

El atributo simple siempre se define sobre un dominio tal y como Chen propuso en su primer artículo y las extensiones coinciden en este punto. Además de este tipo de atributo, en todas las extensiones se incluyen los siguientes tipos de atributos:

Simples o compuestos. Los atributos compuestos también se denominan agregaciones cartesianas.

> Ejemplo en HERM: Agregación cartesiana, nombre de una persona está compuesto por nombre y apellidos: Nombre (Nombre, Apellidos)
>
> Representación:

Figura 2.6: Representación de atributos compuesto, Elmasri & Navathe (2000) y Thalheim (2000), respectivamente.

Monovaluados o multivaluados. Esta clasificación se realiza por el valor o valores que toma un atributo para una determinada entidad o interrelación. Los multivaluados se denominan también agregaciones de conjunto.

Ejemplo en HERM: Teléfonos de una persona: Teléfonos {Teléfono}

Representación:

Figura 2.7: Representación de atributos multivaluados, Elmasri & Navathe (2000) y Thalheim (2000), respectivamente.

Obligatorios u opcionales. Estos atributos representan la obligatoriedad de poseer un valor o no.

Ejemplo en HERM: el nombre de una persona está compuesto por nombre y apellidos, y por apodo en el caso de que exista: Nombre (Nombre, Apellidos, [Apodo])

Representación:

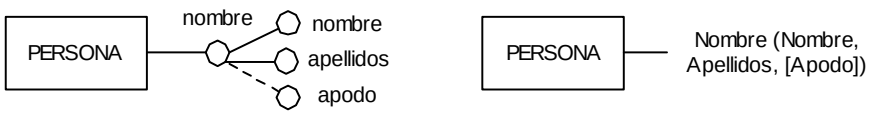


Figura 2.8: Representación de atributos opcionalez, De Miguel, Piattini & Marcos (1999) y Thalheim (2000), respectivamente.

Almacenados o derivados. Los componentes derivados son uno de los elementos mas importantes en los modelos semánticos ya que permiten la abstracción de datos y el encapsulamiento. Constan siempre de dos elementos: una especificación estructural para situar la información derivada dentro del esquema y un mecanismo para indicar como se debe rellenar dicha estructura (lo que se denomina regla de derivación). En resumen este tipo de componentes nos permite introducir información calculada dentro de un esquema. En la mayoría de los modelos semánticos, y en este caso ER los datos derivados que se contemplan son: subtipos derivados (que veremos mas adelante al definir las jerarquías) y atributos derivados, que es el caso que nos ocupa. En cualquier modelo semántico se hará necesario disponer de un lenguaje para especificar reglas de derivación.

No existe restricción a la hora de combinar cada uno de estos tipos de atributos a no ser que tratemos con la clave primaria o identificador principal de un tipo de entidad. En este caso el atributo o conjunto de atributos que pertenecen a dicha clave deben ser obligatorios y monovaluados.

Profundizando en el concepto de interrelación

Un tipo de interrelación representa la relación entre elementos. Esta definición ofrece distintas interpretaciones; por ejemplo, existen diferencias dependiendo de si las interrelaciones puedan asociarse entre sí, como en HERM (Thalheim, 2000) denominadas interrelaciones de grado superior, o si representan asociaciones entre entidades como en ECR, Elmasri & Navathe (2000) o si se consideran una agrupación de tipos de entidades e interrelaciones en clusters como en Teorey (1999). Además de las distintas definiciones existe una dualidad en el concepto de *asociación* ya que este puede, dependiendo del punto de vista adoptado, considerarse bien como una interrelación (si se subrayan los aspectos de asociación) bien como una entidad (si se subrayan los aspectos de representación). Así por ejemplo, un matrimonio puede verse como una interrelación (asociación de personas) o como una entidad (representación de un concepto social y legal). Esta dualidad en las interrelaciones suele ser fuente de confusión en el diseño.

En MacAllister (1998), por ejemplo, define un *tipo de interrelación* como la necesidad de almacenar las asociaciones entre instancias de entidades. Cada tipo de interrelación tiene dos o más roles los cuales especifican el enlace entre la entidad y la interrelación. Por lo que una instancia de una interrelación es una tupla (e_1,e_2,.....e_n) donde n representa el número de roles y cada e_i es una instancia clave para la entidad participante en el i-ésimo role (ith). Una instancia clave de una entidad es el conjunto de valores que identifica cada instancia de un tipo de entidad.

Elmasri & Navathe (2000) a los tipos de interrelaciones las denomina *tipo de vínculo*, y los define como, un tipo de vínculo R entre n tipos de entidades E_1...E_n representa un conjunto de asociaciones entre entidades de estos tipos. De forma matemática, R es un conjunto de instancias de vínculo r_i, donde cada r_i asocia n entidades individuales (e_1, ..., e_n) y cada e_j de r_i es miembro del tipo de entidad E_j, $1 \leq j \leq n$. En definitiva, un tipo de vínculo es un subconjunto del producto cartesiano E_1 x.....x E_n. Definición muy similar a la anterior pero no tiene en cuenta el role con el que participa cada tipo de entidad en la interrelación. Esto repercute en la imposibilidad de distinguir la participación de una entidad dos o más veces en un tipo de interrelación.

Según Thalheim (2000) existen dos tipos de interrelaciones, la interrelación de primer orden, que tiene la forma R = ({E_1,.....,E_n}, {B_1,...B_k}), donde R es el nombre de la interrelación, {E_1,.....,E_n} es una secuencia de tipos de entidades o clusters de las mismas y {B_1,...B_k}es un conjunto de atributos. Una instancia de R para un momento t dado, E_1^t,.....,E_n^t se denomina r y es un elemento del producto cartesiano E_1^t x....x E_n^t x dom(B_1) x…x dom(B_k). El otro tipo de interrelación que define es la de orden superior. Un tipo de interrelación es de orden (i+1) si tiene la forma R = ({E_1,.....,E_n}, {R_1,.....,R_m}, {B_1,...B_k}), donde {R_1,.....,R_m} es una secuencia de tipos de interrelaciones o clusters de orden no mayor que i. Una interrelación r es definida como un elemento del producto cartesiano E_1^t x....x E_n^t x R_1^t x....x R_m^t x dom(B_1) x…x dom(B_k).

Las definiciones presentadas son equivalentes ya que todas coinciden en que una interrelación se compone de un subconjunto del producto cartesiano de las entidades participantes en dicha interrelación.

Un enfoque distinto se plantea en Frederiks, Hofstede & Lippe (1997) cuya definición es tratada mediante la teoría matemática de la categoría. Para definir el tipo interrelación hace referencia a que en los modelos extensiones del ER, los tipos de interrelación fueron formalizados desde la perspectiva de subconjuntos del producto cartesiano, *aproximación orientada a la tupla.* Las instancias o población asociada a una interrelación sería de la forma: Pop(R)={$\langle a_1,b_1\rangle$, $\langle a_2,b_1\rangle$} (figura 2.9).

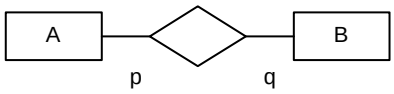


Figura 2.9: R tipo de interrelación donde A participa con el role p y B con el role q

La desventaja de esta aproximación es que la representación de las instancias es demasiado especifica. Las instancias de un tipo interrelación R podrían además ser consideradas elementos de Pop(A)xPop(B) como de Pop(B)xPop(A). Un producto cartesiano impone un orden en las distintas componentes de una interrelación. Consecuentemente, los operadores algebraicos no poseen las propiedades tales como la conmutatividad y la asociatividad. Esta observación lleva a considerar la *aproximación orientada a la correspondencia o mapping,* donde las instancias de un tipo de interrelación son tratadas como funciones de roles a valores. En esta aproximación el anterior ejemplo de instancias podría ser representado como:

Pop(R) = {{p$\rightarrow a_1$, q$\rightarrow b_1$},{ p$\rightarrow a_2$, q$\rightarrow b_1$}}

En esta aproximación el orden no es relevante y además las distintas partes de la relación permanecen totalmente distinguibles. Pero se podrían considerar otras categorías para definir las instancias de un tipo de interrelación.

Considerarlas como instancias de la categoría **FinSet** (conjunto finito).

p = {$f_1 \rightarrow a_1$, $f_2 \rightarrow a_2$}

q = {$f_1 \rightarrow b_1$, $f_2 \rightarrow b_1$}

Si se considera como instancias de **PartSet** permite que ciertas componentes de la interrelación sean indefinidas (permite valores nulos).

$p = \{f_2 \rightarrow a_2\}$

$q = \{f_1 \rightarrow b_1, f_2 \rightarrow b_1\}$

Y si la instancia pertenece a la categoría **Rel** puede ser relacionada con uno o mas objetos en una de sus componentes.

$p = \{f_2 \rightarrow a_1, f_2 \rightarrow a_2\}$

$q = \{f_1 \rightarrow b_1, f_2 \rightarrow b_1, f_2 \rightarrow b_2\}$

Las desventajas que vemos en esta aproximación es que estamos construyendo un meta modelo que unifica todos los modelos de datos conceptuales, esto nos ayudará a pasar de un modelo a otro, a comparar esquemas y realizar esquemas equivalentes. Por lo que se enfoca más a la automatización algo interno que no puede manejar un diseñador. Nosotros estamos en otro plano no tan abstracto donde la representación de los objetos y la semántica asociada es muy importante para que el diseñador pueda manejar estos objetos con total soltura y sencillez. No podemos darle una definición de interrelación tan compleja, ya que es un constructor que se maneja habitualmente y que para ello necesita tener claro.

El concepto de interrelación, por tanto, es bastante confuso. Tanto en su definición como en su interpretación. En el apartado 3 veremos las dificultades que encuentran los usuarios no expertos para manejar este concepto a través de distintos experimentos. Y en el apartado 4 fijaremos el concepto de interrelación que manejaremos en la presentación de la propuesta de este trabajo de tesis doctoral.

Propiedades asociadas al concepto de interrelación

Las propiedades que caracterizan un tipo de interrelación son: el grado, la correspondencia y la cardinalidad o multiplicidad.

El grado nos indica el número de participantes asociados en un tipo de interrelación. Esta propiedad es muy importante porque define una clasificación dentro de los tipos de interrelación, las interrelaciones binarias (IB) y las interrelaciones n-arias o de orden superior. En la mayoría de los textos referente a la familia ER se indica que las interrelaciones que mas aparecen en el mundo real son las IB y por ello son las mas importantes e incluso han caracterizado un tipo de modelos denominados binarios. Actualmente todos los modelos han considerado la importancia de poseer interrelaciones de cualquier grado al comprobar que las interrelaciones n-arias recogen mucha semántica imposible de recoger con solo IB. De estas propiedades hablaremos posteriormente en la sección 2.3.2 donde presentaremos la discusión entre las distintas propiedades de las interrelaciones desde el punto de vista de la tres familias presentadas.

Jerarquías

El modelo ER será extendido incluyendo los conceptos de relación *superclase/subclase, herencia* de tipo, y los procesos de especialización y generalización. Estos procesos de abstracción se incorporan al modelo ER provenientes de los modelos de datos semánticos para incorporar más riqueza y ampliar la semántica. La notación aquí presentada es utilizada tanto por ECR (Elmasri & Navathe) como en Teorey (1990).

Una subclase de un tipo de entidad sirve para agrupar de manera adicional un conjunto de entidades. Por ejemplo, las entidades miembros del tipo de entidad EMPLEADO pueden agruparse en RECURSOS-HUMANOS, INFORMATICOS y GESTIÓN (figura 2.10). El conjunto de entidades de cada una de estas agrupaciones (subclases) es un subconjunto de las entidades que pertenecen al tipo EMPLEADO (superclase).

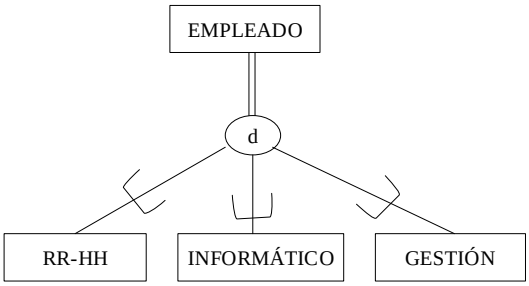


Figura 2.10: Ejemplo de especialización/generalización

La relación entre superclase y subclase debe cumplir que todo miembro de una subclase debe aparecer como miembro de la superclase y por ello hereda todos los atributos que posee el tipo de entidad superclase.

El proceso de definición de subclases se denomina especialización y el inverso a este proceso es la generalización. La necesidad de utilizar este tipo de relaciones viene motivada por dos razones, la primera de ellas surge de la necesidad de establecer atributos propios a un determinado subconjunto de la entidad y la segunda, para poder definir interrelaciones en las que solo participan los miembros de una determinada subclase.

Las restricciones que se definen sobre la especialización/generalización para enriquecer el modelo son las siguientes:

Subclases definidas por un predicado, en las cuales se determina a través de una condición la pertenencia de una entidad como miembro de una subclase. En el ejemplo, figura 2.11, podríamos incluir un atributo Tipo_Trabajo para indicar que depende del valor del mismo, una entidad pertenecerá a una subclase o a otra. Este atributo se denomina *atributo de definición o discriminante*.

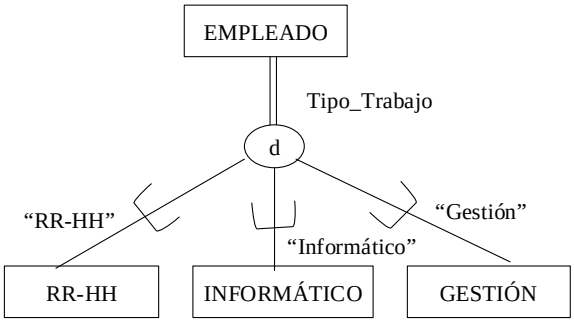


Figura 2.11: Subclases definidas por predicado

Cuando no tenemos una condición que determine la pertenencia a una clase, la subclase es definida por el usuario.

Las restricciones más importantes definidas en la especialización son: restricción de disyunción y de completitud. La primera de ellas indica que si las subclases de una

especialización son disjuntas (se representa con una “d” dentro del círculo y con una “o” en caso contrario) se debe cumplir que una entidad es miembro de cómo mucho una subclase. En la segunda, si las subclases son totales (se representa con una línea doble que une el círculo con la superclase y con una línea simple en caso contrario) se cumple que toda entidad de la superclase pertenece al menos a una subclase. Estas dos restricciones son independientes por lo que podemos tener cuatro tipos de especialización: disjunta y total, disjunta y parcial, solapada y total, y por último, solapada y parcial. En las figuras hemos representado la notación utilizada por Elmasri & Navathe (2000) en el modelo ECR. La definición de un tipo de especialización lleva consigo unas reglas de inserción y eliminación consecuencia de las restricciones impuestas.

Cuando una subclase posee subclases sobre ella forma jerarquías o retículas de especialización. Una jerarquía indica que una subclase solo puede formar parte de una relación superclase/subclase. Es decir, la especialización tiene estructura de árbol. Por el contrario, una retícula permite que una subclase pueda ser subclase en mas de una relación. La subclase que posee más de una superclase se denomina *subclase compartida* y da lugar a la herencia múltiple (hereda de varias atributos de varias especializaciones). Lo que te marca el modelo es que una subclase compartida puede heredar el mismo atributo por relaciones de especialización distintas y que por tanto no se duplicará este atributo, es decir, solo se heredará una sola vez.

Para el proceso de generalización también se puede aplicar conceptos similares de jerarquía y retícula, pero las restricciones no tienen sentido ya que una generalización siempre será total.

En contraposición a esta propuesta para representar este concepto de abstracción se encuentra HERM (Thalheim, (2000)), el cual aboga por no tener un nuevo constructor para contemplar este concepto al considerar una jerarquía como un tipo especial de interrelación.

Otra de las extensiones del modelo ER muy importantes es la relación de agregación o categoría. Una agregación representa una colección de entidades (superclases) que se

agrupan en una subclase (Elmasri & Navathe, 2000). Una categoría puede ser total o parcial, dependiendo de si la unión de las entidades pertenecientes a las superclases conforman las entidades que pertenecen a la subclase o solo son un subconjunto. En este caso una entidad de la subclase heredará los atributos de la superclase a la que pertenezca.

2.2.2. Familia de Orientación a Objetos (OO)

A diferencia de los apartados anteriores en los que presentábamos modelos de datos, explicando sus elementos, el lenguaje que utilizan para expresarlos y el proceso de modelado que proponen, el lenguaje de modelado unificado (UML) es una notación, principalmente gráfica, independiente del proceso y que permite visualizar, especificar, construir y documentar los elementos de un sistema.

La motivación que nos lleva a incluir este lenguaje dentro del estado del arte es porque UML se ha convertido en el sucesor de lo que hasta ahora eran métodos de análisis y diseño orientado a objetos. De esta forma cubrimos todas las vertientes que se han establecido para el desarrollo de las bases de datos.

Parte de las premisas en el desarrollo de UML fue que se deseaba separar los modelos de una metodología dada, por lo que una de las principales características de este lenguaje es la independencia de las metodologías. UML puede ser utilizado en distintas metodologías como Fusion, Objectory, Rational Unified Process, OMT, ECM, Catalysys, etc. La independencia antes mencionada permite que las organizaciones adapten el uso de UML a la metodología que consideren más apropiada. UML es un lenguaje para hacer modelos.

Breve historia

Los lenguajes orientados a objetos aparecieron entre la mitad de los setenta y finales de los ochenta cuando los metodologistas, enfrentados a los nuevos lenguajes de programación orientados a objetos y a unas aplicaciones cada vez más complejas, empezaron a experimentar con enfoques alternativos al análisis y al diseño. El número

de métodos orientado a objetos se incrementó durante el periodo comprendido entre 1989 y 1994. Muchos usuarios tenían problemas para encontrar un lenguaje que se adaptara a sus necesidades, provocando así la guerra de los métodos. Como consecuencia de esto comenzaron a aparecer nuevas generaciones de métodos, entre los que destacan: el método Booch, el método OOSE (Object-Oriented Software Engineering, Ingeniería del Software Orientada a Objetos) de Jacobson y el método OMT (Object Modeling Technique, Técnica de modelado de Objetos) de Rumbaugh.

El esfuerzo de UML comenzó en octubre de 1994, cuando Rumbaugh se unió a Booch en Rational. El objetivo inicial de su proyecto fue la unificación de los métodos de Booch y OMT (Object Modeling Technique). El borrador de la versión 0.8 del Método Unificado se publicó en octubre de 1995. Por esta época, Jacobson se unió a Rational y el alcance del proyecto UML se amplió para incorporar OOSE (Object Oriented Software Engineering). Los esfuerzos se plasmaron en los documentos de la versión 0.9 en junio de 1996.

Las organizaciones que contribuyeron a la definición de 1.0 de UML fueron Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments y Unisys. Esta colaboración produjo UML 1.0, un lenguaje de modelado bien definido, expresivo, potente y aplicable a un amplio espectro de dominios de problemas. UML 1.0 se ofreció para su estandarización al Object Management Group (OMG) en enero de 1997. Entre enero y julio de 1997, el grupo inicial de colaboradores se amplió para incluir prácticamente a todas las demás organizaciones que habían enviado alguna propuesta o habían contribuido en alguna medida en las respuestas iniciales al OMG, incluyendo a Andersen Colsunting, Ericsson, Object Time Limited, Platinum Technology, Ptech, Reich Tecnologies, Softeam, Sterlin Software y Taskon. Se formó un grupo de trabajo para la semántica, liderado por Cris Kobryn de MCI Systenhouse y coordinado por Ed Eykholt de Rational, para formalizar la especificación de UML y para integrar UML con otros esfuerzos de estandarización. Una versión revisada de UML (la versión 1.1) se ofreció al OMG para su estandarización en julio de 1997. En septiembre de 1997, esta versión fue aceptada por la OMG Análisis and Design Task Force (ADTK) y el OMG

Architecture Board y se sometió al voto de todos los miembros del OMG. UML 1.1 fue adoptada como estándar por el OMG el 14 de noviembre de 1997.

La RTF publicó una revisión editorial, UML 1.2, en junio de 1998. En otoño de 1998, la RTF publicó UML 1.3, que es la versión que se describe aquí.

Componentes de UML

El lenguaje unificado tiene la especificación de todos los elementos necesarios para cubrir el ciclo de vida de desarrollo software. Este ciclo es mucho más amplio que el consideramos en esta tesis por lo que intentaremos ceñirnos a la especificación de aquellos componentes que atañen al desarrollo de bases de datos. Esto no quita para que nombremos a todos ellos y expliquemos de forma breve en que consiste.

La notación de UML incluye un enorme número de símbolos utilizados para la construcción de diferentes diagramas que cubren las distintas perspectivas de una aplicación. UML propone nueve diagramas para cubrir el ciclo de vida de una aplicación:

- ✓ Casos de uso: diagrama de casos de uso.
- ✓ Estructura estática: diagrama de clases y diagrama de objetos.
- ✓ Conducta: diagrama de estados y actividades.
- ✓ Interacción: diagrama de secuencia y diagrama de colaboración.
- ✓ Implementación: diagrama de componentes y diagrama de despliegue.

Alguno de estos diagramas son utilizados para el diseño de código de programas orientados a objetos, por ejemplo, el diagrama de colaboración, otros para el análisis de requisitos, como por ejemplo, diagramas de actividad y diagramas de caso de uso. Para el análisis conceptual se utiliza el diagrama de clases pero la mayoría de los expertos en el desarrollo de bases de datos coinciden que sobre todo es más eficiente en el diseño lógico, al poder definir si un atributo es público, privado o protegido; que operciones

están encapsuladas; o si un tipo de asociación solo puede recorrerse en una dirección. Con ello podemos recoger tanto datos como conducta mucho más rico que si utilizamos un modelo relacional. El problema encontrado es que la mayoría de los SGBD's son relacionales pero no dudan en afirmar que en un futuro esto cambiará y UML tendrá un papel importante en el desarrollo de bases de datos.

Por tanto, como nuestro trabajo está enfocado al modelado de datos, y en esta sección tratamos el modelado conceptual, solo consideraremos la estructura estática de UML para el análisis de datos, donde el diagrama de clases se utilizará para el modelado de datos y el diagrama de objetos para discutir la población de los datos (ocurrencias o instancias).

Diagrama de clases

Una clase es el equivalente a un tipo de entidad en la familia de modelos ER. Por lo que se considera la unidad básica que encapsula toda la información de un objeto (un objeto es lo que consideramos una instancia de un tipo de entidad) , es decir, es una definición común para un conjunto de objetos que comparten atributos y operaciones. Una clase se representa como un rectángulo que posee tres divisiones, en donde la parte superior contiene el nombre de la clase, la parte intermedia contiene los atributos que caracterizan a la clase, y la parte inferior contiene los métodos u operaciones. Esta última sección no será representada en la fase conceptual. Su representación se muestra en la figura 2.12.

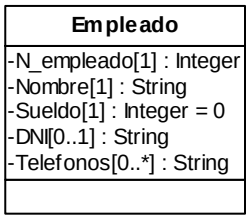


Figura 2.12: Representación de la clase Empleado

Es importante aclarar que en UML no es obligatorio elegir una identificador principal para cada clase, pero para mantener una uniformidad se suele indicar con una {P} el

atributo o atributos que forman parte del identificador principal y con {Un} aquellos que son identificadores alternativos. Aunque esta notación no estándar. También podemos observar en la figura 2.12 los distintos tipos de atributos que podemos definir, que contempla todos los estudiados en sección 2.1.

Las interrelaciones son denominadas asociaciones. También poseen las propiedades de grado, correspondencia y multiplicidad (cardinalidad) que posteriormente veremos en la sección 2.2.4. Aquí destacaremos otras restricciones que pueden definirse para las asociaciones que no suelen aparecer en la familia ER, aclarando que el diagrama de clases es n-ario y utiliza la misma representación que la de la familia ER, a través de un rombo.

Podemos definir restricciones entre asociaciones definidas sobre el mismo conjunto de clases. Una de las más importantes es la de subconjunto mostrada en la figura 2.13 a), la cual expresa que las instancias de la asociación *Dirige* se incluyen en las instancias de *Trabaja*.

Otra restricción entre asociaciones es la de exclusión, figura 2.13 b), recogiendo que una instancia de factura se relaciona con un cliente particular o con una empresa pero no con ambas.

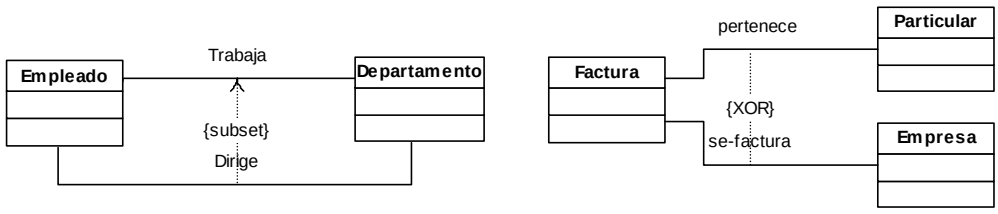


Figura 2.13: a). Representación de subconjuntos en UML. b) Representación de conjuntos disjuntos en UML.

También podemos indicar en una asociación en términos de agregación para indicar la relación todo/parte. Por ejemplo, la figura 2.14, se muestra los dos tipos de agregación más importantes que permite UML, la agregación simple y compuesta o fuerte.

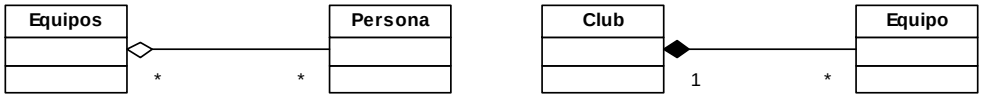


Figura 2.14: Representación de agregación simple y fuerte en UML.

La agregación simple es menos restrictiva al considerar que las *partes* no forman completamente el *todo*.

UML también considera las jerarquías (figura 2.15 a), con la propiedades estudiadas en el ER en las que se indica la totalidad/parcialidad o la exclusividad.

Figura 2.15: a) Representación de la jerarquía en UML. b) Relación entre asociaciones.

También UML permite relaciones entre asociaciones como se muestra en la figura 2.15 b), la cual representa que una matrícula se formaliza siempre y cuando un estudiante esté asociado a un curso.

2.2.3. Familia Objeto-rol

El modelo ORM (object-role modeling) es un método para el modelado y consulta de sistemas de información a nivel conceptual cuya principal característica, que además le diferencia de otras técnicas de modelado vistas anteriormente, es que no emplea de forma explícita los atributos.

Como sabemos los SGBD están implementados para modelos lógicos como el relacional, jerárquico u objetos- relacional por lo que ORM incluye procedimientos para realizar el mapping entre el nivel conceptual y lógico, igual que Teorey, Yang & Fry (1986) para la metodología de bases de datos relacionales. Por lo que una parte importante de las técnicas presentadas por Halpin (2000) se corresponde a presentar una serie de pasos para llevar a cabo el modelado en ORM y presentar la transformación a modelos lógicos. La otra parte importante en la que se basa este modelo es en especificar a través de sentencias en lenguaje natural los requisitos del UD a modelar. Las sentencias construidas son frases sencillas que evitan ambigüedades para poder identificar de forma clara cuales son la relaciones elementales entre objetos. Este método es conocido como NIAM (Natural Language Information Analysis Method) (Nijssen, 1977).

Es importante destacar que este modelo también ha sufrido distintas extensiones pero que todas ellas se han desarrollado dentro del marco objeto-role. La versión ORM que presentaremos a continuación utiliza interrelaciones sin limitaciones en su grado por lo trataremos con un modelo n-ario aunque existe una versión binaria denominada BRM (Shoval & Shreiber, 1993).

El método de modelado comprende un lenguaje (formal y gráfico) y un procedimiento que servirá para guiar la construcción de esquemas. Este proceso a menudo se denomina proceso de modelado. A continuación explicaremos la notación empleada por Halpin (2001).

Elementos

Los elementos principales son el tipo de entidad y el predicado. A continuación pasamos a dar su definición, comparándola con las otras dos familias, y cual es su representación gráfica. Además presentaremos las aportaciones que realiza el modelo para incluir nuevas restricciones añadiendo semántica a los esquemas conceptuales.

El tipo de entidad se representa con una elipse que contiene el nombre del objeto y debe tener al menos un esquema de referencia (*reference schema*) que indica cómo cada

instancia de un tipo de entidad es identificada. Por tanto, el modelo ORM impone que todo tipo de entidad tenga una referencia primaria de esquema pero permite que existan tipos de entidades independientes, es decir, que no participen en ninguna interrelación. Podemos afirmar que el tipo de entidad aquí definido es semejante a los vistos para las anteriores familias. La diferencia más importante es que los atributos se representarán por medio de asociaciones entre tipos de entidades. Por lo que en este modelo el principal constructor es el predicado.

Un predicado de n-roles se representa a través de n cajas con los nombres de cada uno de los roles. ORM da mucha importancia a esta propiedad ya que a través de ella se realiza una verbalización del esquema en lenguaje natural. Por tanto, los predicados son definidos como un tipo de interrelación o una asociación. Tienen también la propiedad del grado, la correspondencia que la denomina *restricciones de unicidad interna* y restricciones de unicidad externa, la restricción de participación y la de frecuencia que estudiaremos en la sección de comparación y discusión de familias. Pero además de estas restricciones, se pueden definir las siguientes restricciones.

Las restricciones de subconjunto muy parecidas a la definida en para el diagrama de clases de UML pero con una sutileza, el subconjunto puede ser definido sobre uno o varios roles del predicado.

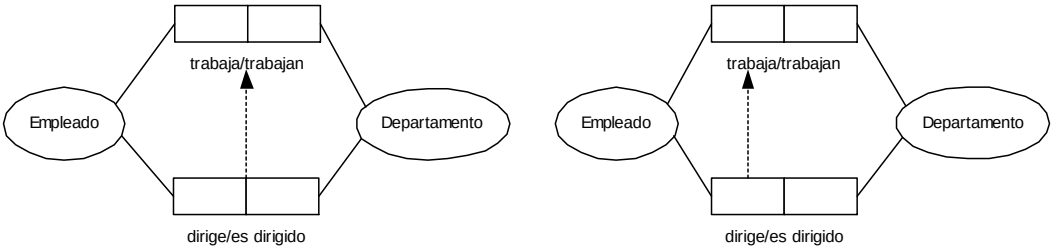


Figura 2.16: a) Subconjuntos de predicados. b) Subconjuntos de roles.

En la figura 2.16 a) se refleja que si un empleado *e* dirige un departamento *d* entonces el empleado *e* trabaja en el departamento *d*. Sin embargo, si la restricción solo afecta a u

role, figura 2.16 b), esta restricción solo recoge que si un empleado dirige un departamento, trabaja en un departamento. Esta distinción UML no la recoge.

También podemos definir la restricción de igualdad entre instancias o población de dos predicados o por las componentes pertenecientes a una parte de los roles de cada uno de los predicados. Esta restricción se representaría con una doble flecha. Y para finalizar con las operaciones de conjuntos, podríamos definir la exclusión entre poblaciones pertenecientes a la totalidad de roles de los predicados o a una parte.

Otro tipo de restricción es la OR-exclusiva que tiene el mismo significado que en UML y que se representa como aparece en la figura 2.17 a).

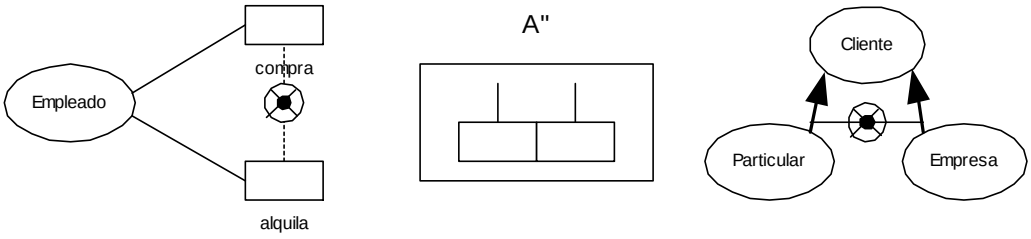


Figura 2.17: a) Representación de roles exclusivos. b) Relación entre predicados. c) Representación de jerarquías.

También son consideradas las jerarquías, con las mismas propiedades de totalidad/parcialidad y disjunción/solapamiento que aparecen en las familias ER y OO (figura 2.17 c). Y los tipos de entidad anidados que se corresponden con la objetivación de un predicado o como tratamos en UML relacionar dos predicados (figura 2.17 b).

2.3. Restricciones de integridad

Hemos realizado un apartado para este tipo de restricciones por considerar que es una de las partes mas importantes dentro de un modelo independientemente de la familia de la que provenga. Si que es cierto que cada modelo de datos ofrece mas o menos restricciones de integridad y que esto le imprimirá una mayor potencia. Con las restricciones de integridad se modela aquellos estados de la base de datos que son permitidos.

Las restricciones de integridad se dividen en estáticas, siempre deben cumplirse y dinámicas, hablan sobre aquellos estados válidos por los que pueden pasar los datos después de realizar cualquier operación de actualización sobre los mismos.

2.3.1. Primera generación: Dependencias funcionales

Para realizar un buen diseño de bases de datos es de vital importancia el modelado de la semántica del UD. La idea central para ello es la noción de restricción de integridad. La definición y el cumplimiento de la las mismas ayuda a garantizar que la BD refleje correctamente las principales características del dominio. Las dependencias son una de las mas fundamentales restricciones introducidas por Codd en 1970 y posteriormente extendido por otros modelos entre los que destacan Thalheim (1992) y Weddell (1992). Este mecanismo sirvió para extender el modelo relacional proporcionándole más contenido semántico. Se introduce en esta sección como una de las restricciones más importantes impuestas en la primera generación de modelos de datos y posteriormente veremos cómo influye en las metodologías aportando la teoría de la normalización como un método que asegura un diseño de bases de datos sin redundancias. Los resultados obtenidos acerca posteriormente se han utilizado para demostrar la correctitud de las restricciones de cardinalidad a las que están íntimamente ligadas.

Veremos la definición de las distintas dependencias que se introducen para realizar el proceso de normalización que estudiaremos en la sección 2.4.2.

Dependencia funcional

Y depende funcionalmente de X, X→ Y, donde X e Y son subconjuntos de R especifica que para dos tuplas cualesquiera t_1 y t_2 de r si $t_1[X] = t_2[X]$ se cumple que $t_1[Y] = t_2[Y]$.

Se denomina determinante o implicante al descriptor que se encuentra en la parte izquierda del símbolo de implicación, e implicado al descriptor de la parte derecha.

Dependencia funcional completa

Si X está compuesto por X_1 y X_2, se dice que Y tiene una dependencia plena o completa de X, si depende funcionalmente de X pero no depende de ningún subconjunto del mismo:

$X \rightarrow Y$ y $X_1 \not\rightarrow Y$, $X_2 \not\rightarrow Y$

Y se representa $X \Rightarrow Y$

Dependencia funcional trivial

Una dependencia funcional $X \rightarrow Y$ se dice que es trivial si Y es un subconjunto de X ($Y \subseteq X$).

Dependencia funcional elemental

Una dependencia funcional $X \rightarrow Y$ es elemental si Y es un atributo único no incluido en X, y no existe X' incluido en X tal que $X' \rightarrow Y$.

Descriptores equivalentes

Se dice que dos descriptores X e Y son equivalentes, si se cumple $X \rightarrow Y$ e $Y \rightarrow X$. Lo que se representa por $X \leftrightarrow Y$.

Dependencia funcional transitiva

Sea el esquema de relación R(X, Y, Z) en la que existen las siguientes dependencias funcionales: $X \rightarrow Y$, $Y \rightarrow Z$ e $Y \not\rightarrow X$. Se dice que Z tiene una dependencia transitiva respecto a X a través de Y. Y se representa X -- > Z.

Si además $Z \rightarrow Y$ se dice que la dependencia transitiva es estricta.

Debemos tener en cuenta que estas restricciones son semánticas y por lo tanto las tiene que determinar el diseñador previo conocimiento del dominio.

Si partimos de un conjunto F de dependencias funcionales, definimos F^+ como el conjunto de dependencias inferido de F y se denomina cierre de F. Las reglas de inferencia son seis y se presentan a continuación:

1. Regla reflexiva. Si $Y \subseteq X$ entonces $X \rightarrow Y$
2. Regla de aumento. $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
3. Regla transitiva. $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
4. Regla de descomposición. $\{X \rightarrow YZ\} \models X \rightarrow Y$
5. Regla de unión. $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
6. Regla pseudotransitiva. $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

Armstrong (1974) demostró que las tres primeras reglas son correctas y completas. La correctitud implica que todas las dependencias funcionales que podamos inferir de F a través de estas primeras reglas se cumplirán para cualquier estado de la base de datos. Por completo se entiende que con las tres reglas es suficiente para inferir el conjunto de todas las dependencias.

Una clase de dependencias funcionales especiales ya que recogen las claves candidatas que puede contener una relación son las que a continuación presentamos y que en la sección 2.4.1 volveremos a retomar para realizar el proceso de normalización.

Superclave y clave candidata

Dado un esquema de relación R (A, DF), donde A es el conjunto de atributos definido para R y DF el conjunto de dependencias funcionales existentes en R. Se denomina superclave SK de la relación R a un subconjunto no vacío de A, tal que $SK \rightarrow A$ es una consecuencia lógica de DF.

Por tanto, $SK \neq \varnothing$ y $SK \rightarrow A \in DF^+$

Decimos que K es una clave candidata de R, si, además de ser superclave de R, no existe ningún subconjunto estricto K' de K tal que K' implique también a A.

Las dependencias que a continuación presentamos tienen como característica que dependen del contexto, es decir, de en ellas influyen el resto de atributos de una relación.

Dependencias multivaludas

Sea X e Y dos descriptores o atributos de una relación. X multidetermina a Y si para cada valor X existe un conjunto bien definido de valores posibles en Y, con independencia del resto de atributos de la relación (Fagin, 1977). Se denota como X→→Y.

Estas dependencias son una generalización de las dependencias funcionales, al ser estas últimas un caso de multideterminación de un solo valor. Por tanto, también existen una reglas de inferencia para estas dependencias en las cuales se incluyen los axiomas de Armstrong y del cual se deducen todas las dependencias de una relación.

Dependencias jerárquicas

Sean X,Y y Z descriptores. X multidetermina jerárquicamente a Y y Z si la proyección de los atributos de X multidetermina a Y y X multidetermina a Z. Se denota como X→ Y| Z.

Dependencias en combinación

Una relación R tiene dependencias en combinación respecto de sus proyecciones (R1, R2,…, Rn) si R= R1*R2*..Rn y se denota por DJ*(R1, R2,.., Rn)

Estas dos últimas dependencias veremos que son la consecuencia para que una relación se encuentre en 4 y 5FN, repectivamente.

Para terminar con la definición de dependencias que recogen restricciones del UD para el modelo relacional definiremos otra dependencia importante, que aunque no influye en

la teoría del diseño de bases de datos relacional, si será muy utilizada porque generaliza el concepto de integridad referencial y expresa en el modelo relacional conceptos semánticos como la generalización que sin ella no se recogerían.

Dependencias de inclusión

Decimos que un descriptor X del esquema de relación R depende en inclusión del descriptor Y del esquema de relación S, y lo denotamos por $R[X] \subseteq S[Y]$. Si para cualquier extensión r de R y s de S se cumple:

$\Pi_X(R) \subseteq \Pi_Y(S)$ siendo X e Y descriptores compatibles, definidos sobre los mismos dominios, y no necesariamente distintos.

2.3.2. Segunda generación: restricciones de cardinalidad

¿Por qué la mención especial de este constructor?

En primer lugar no hemos querido meter la definición dada a esta restricción en cada una de las familias presentada anteriormente porque no las caracteriza. Es decir, existen modelos que adoptan la misma interpretación perteneciendo a familias distintas. Por lo que presentaremos cada una de estas interpretaciones indicando qué modelo o modelos la acogen y de que familia son.

En segundo lugar, este apartado nos introducirá de una forma directa en la propuesta realizada en este trabajo de tesis doctoral.

Se podría dar una definición genérica de esta restricción.

Definición genérica: una *restricción de cardinalidad (RC)* limita la combinación de instancias de los elementos que participan en una interrelación (familia ER), asociación (familia OO) o predicado (familia OR). Podemos distinguir RC mínimas que imponen un límite inferior a esta combinación y las RC máximas que establecen un límite superior. Las RC máximas coinciden con la definición de correspondencia dada por Chen en su primer artículo (Chen, 1976).

Analizaremos con mas detalle la restricción de cardinalidad desde dos perspectivas, la perspectiva de la representación y la de la semántica. La proliferación en la representación hace que se produzca confusión en la utilización de esta restricción. Un ejemplo claro de esta confusión e incluso de un mal uso se produce cuando el diseñador elige para realizar la fase conceptual un modelo n-ario que al no ser tan restrictivo recoge mas información del UD, y posteriormente lo traslada a una herramienta CASE. En general se encuentra con varios problemas, el primero, que la herramienta CASE no soporta el modelo de datos con el que previamente se diseño, y en segundo lugar, y más importante que se desconoce como hacer la equivalencia entre ambos modelos porque la representación es distinta y porque no se tiene muy claro la semántica que recogen.

La primera diferencia palpable que encontramos en el estudio de esta restricción es la de su representación. En los modelos denominados binarios, aquellos que solo permiten la representación de asociaciones entre dos elementos, la restricción de cardinalidad es restringida a un conjunto de valores. Estos valores son *cero o uno* cuando hablamos RC mínima y *uno o N* (como valor ilimitado que indica más de uno) para la RC máxima. Con la acotación de valores consiguen que su representación sea sencilla, y por lo tanto fácilmente implementable, por eso la mayoría de los modelo binarios están soportados por una herramientas CASE. Por ejemplo, CASE*Method (Barker, 1990), IDEF1X (Bruce, 1992), IE (Martin, 1990), Hansen & Hansen (1995) o BRM (Shoval & Shreiber, 1993).

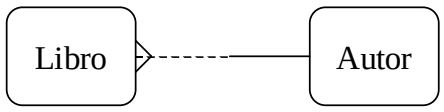


Figura 2.18: Representación de la RC, Barker (1990)

La interpretación de las RC máximas de la figura 2.18, Barker (1990), muestran que un *autor* escribe varios *libros*, y los representa con la bifurcación de la línea que une Autor con Libro, y que un *libro* puede ser escrito por un *autor*. Las RC mínimas se representan con una línea punteada para indicar el cero u opcionalidad situada en la

entidad que participa con esta cardinalidad. En nuestro caso Libro es opcional lo que significa que un *libro* puede ser anónimo. La línea continua pegada a Autor se corresponde con la cardinalidad mínima de uno, lo que se interpreta como que un *autor* al menos ha escrito un *libro*. Esta notación es muy utilizada y tiene la característica de que la localización de las RC máxima y mínima para un tipo de entidad son opuestas. La máxima se representa en el lado contrario de la entidad y la mínima junto a la entidad. Para la notación IE o también conocida por *crow's foot* se utiliza una bifurcación de línea al final de esta para indicar la RC máxima de *varios* de la entidad que se encuentra en el lado opuesto, el trazo vertical "|" siempre al final de la línea expresa un *uno* y para la RC mínima de cero un círculo y el trazo vertical para indicar un *uno*.

Algunos autores clasifican las restricciones de cardinalidad según su localización. Si la RC se coloca al lado del tipo de entidad a la que se corresponde, se denomina representación *look here,* si se coloca en el lado opuesto, se denomina *look across*. Algunos modelos utilizan la misma representación para las cardinalidades máximas y mínimas y otros, tal cual hemos podido ver en la notación utilizada por Barker (1990) combinan ambas representaciones. Como veremos mas adelante estos dos conceptos también son utilizados para distinguir entre dos formas de hallar las restricciones de cardinalidad.

En lo modelos n-arios la representación de las RC es también muy prolífica, por ejemplo, Elmasri & Navathe (2002) y Teorey (1999) representan solamente las RC máximas y utiliza la misma notación presentada por Chen (1976). La diferencia entre ellos es el lugar donde colocan la etiqueta de la restricción, el primero de ellos la emplaza al lado del tipo de entidad a la que pertenece la restricción y el segundo al lado contrario. Pero en general, la mayoría de los modelos consideran las RC mínima y máxima como un intervalo de números enteros (i, j) donde $i<j$, $i \geq 0$, $j> 0$ o incluso un conjunto de valores Hartmann (1998).

Las distintas representaciones provocan que estas restricciones sean difíciles de manejar ya que generan ambigüedad en el diseño. Pero esto no generaría un problema si la semántica de la restricción de cardinalidad estuviera bien definida.

Estas restricciones complementan a entidades y asociaciones al matizar la semántica que en éstas se recoge. Al restringir la manera en que los ejemplares de entidades e interrelaciones pueden combinarse se limitan los posibles estados de la BD. La restricción de cardinalidad es una de las más importantes restricciones que pueden establecerse en un esquema conceptual. Su función es limitar el número de ejemplares de los tipos de entidad que pueden resultar asociados por la interrelación (i.e. participar en ejemplares del tipo de interrelación). Sin embargo, la definición de esta restricción admite diversas variantes.

A continuación mostraremos una clasificación de las interpretaciones encontradas a lo largo del estudio de diversos modelos de datos conceptuales, distinguiendo en las interrelaciones, que a efectos prácticos resulta importante, entre una interrelación binaria (IB), cuando la interrelación asocie solamente dos entidades, mientras que se hablará de una interrelación de orden superior, cuando la interrelación asocie más de dos entidades.

Primera interpretación para interrelaciones binarias: *Look Here*

La restricción de cardinalidad *look here*, RC(LH), especifica el número mínimo y máximo de instancias de un tipo de entidad que pueden participar en las instancias de una interrelación. Si la RC(LH) mínima es 0 se dice que la participación del tipo de entidad dentro del tipo de interrelación es opcional, es decir, no toda instancia de la entidad interviene en la interrelación. Por el contrario un 1 o cualquier valor i>1, obliga a que cada instancia del tipo de entidad intervenga en 1 o i instancias de la interrelación. La RC(LH) máxima pondrá el límite superior a esta participación. En general, esta interpretación de la cardinalidad es denominada aproximación de MERISE por otros autores y además es colocada con una etiqueta al lado de la entidad de la se está obteniendo la restricción.

Elmasri & Navathe (2002) en su modelo ECR utiliza la RC(LH) mínima para mostrar la participación total o parcial de las instancias de un tipo de entidad dentro de un vínculo binario y es representada con una línea doble que une la entidad con la interrelación para indicar la totalidad en la participación o una línea simple para representar la parcialidad (figura 2.19). El modelo ORM (Halpin, 2001) denomina a la RC(LH) mínima *mandatory role* y tiene el mismo significado que el anteriormente expuesto para el modelo ECR pero su representación se realiza a través de un círculo relleno en al entidad que posee el role obligatorio (figura 2.19).

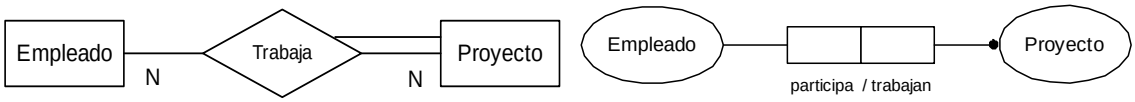


Figura 2.*19: RC(LH) mínima con distinta representación y equivalente semántica.*

En el modelo HERM, Thalheim (2000), la definición de esta restricción se enuncia formalmente:

Sea R = (R_1,…, R_k, atr(R)) donde R_i es una entidad o un tipo de interrelación y para cada i, $1 \leq i \leq k$, la restricción de cardinalidad comp (R, R_i) = (m, n) que especifica que, en cada uno de los estados de la BD, una instancia *e* de R_i^C aparece R^C al menos m y como mucho n veces, es decir,

Comp (R, R_i) = (m,n) si y solo si $m \leq |\{r \in R^C \setminus r(R_i) = e\}| \leq n$, $\forall t, \forall e \in R_i^C$

Donde |M| denota la cardinalidad de M y $r(R_i)$ es la proyección de r a R_i.

En resumen podemos indicar que esta restricción se calcula mirando las instancias de un tipo de entidad y contando las veces que aparece en la interrelación en la que participa.

Segunda interpretación para interrelaciones binarias: *Look Across*

La restricción que denominamos *look across* se fija en las instancias pertenecientes a la interrelación binaria, fija una de sus componentes o role y cuenta el número mínimo y máximo de veces que se combina con elementos distintos del otro role. Esta restricción

cuenta las combinaciones de un tipo de interrelación a través de las instancias de la misma por lo que la RC(LA) mínima siempre será de al menos 1.

Halpin (2001) refleja con la *restricción de unicidad interna* si las instancias pertenecientes a un role son únicas para un predicado. Esta restricción es equivalente a la RC(LA) máxima aun que en ORM está limitada a los valores *uno* o *varios (N)*. Por ejemplo, la figura 2.20 recoge que un empleado participa de forma única con el role *trabaja*. Un predicado puede tener varias restricciones de unicidad y al menos una se ellas debe ser declarada como primaria (figura 2.20).

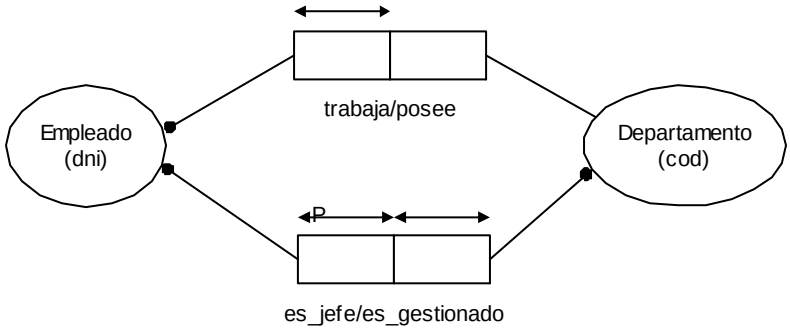


Figura 2.20: Restricción look across en el modelo ORM (Halpin, 2002)

Podemos, en este punto, pensar que aparentemente esta restricción no tiene ninguna diferencia frente a *look here*, pero existe una sutil diferencia. *Look here* toma las instancias pertenecientes a una entidad y cuenta las veces que aparecen en la interrelación en la que participa, por eso si la RC(LH) mínima es cero no se impone ninguna restricción. *Look across* toma las instancias de una interrelación y se fija en una de sus componentes, y cuenta el número de veces que aparece cada uno de los valores. Por tanto, RC(LA) mínima es al menos uno, lo que implica que este caso no proporciona ninguna restricción.

Interrelaciones de grado superior o n-arias con n>2

Con la división realizada para la exposición de las distintas aproximaciones a la restricción de cardinalidad se pretende mostrar la dificultad añadida al tratar con

interrelaciones de grado superior. El problema añadido es que como sabemos la cardinalidad impone restricción en la combinación de las entidades que participan en una asociación. Tratándose de una interrelación ternaria como la que presentamos en la figura 2.21 con las instancias que pertenecerían a la misma, esta combinación se podría interpretar de varias formas: ¿participación de un tipo de entidad, por ejemplo Calificación, en la interrelación o la participación de la componente correspondiente a calificación en la interrelación?. La restricción de cardinalidad en uno y otro caso sería, (0,n) o (1,n) para el conjunto de instancias presentado en la figura 2.21.

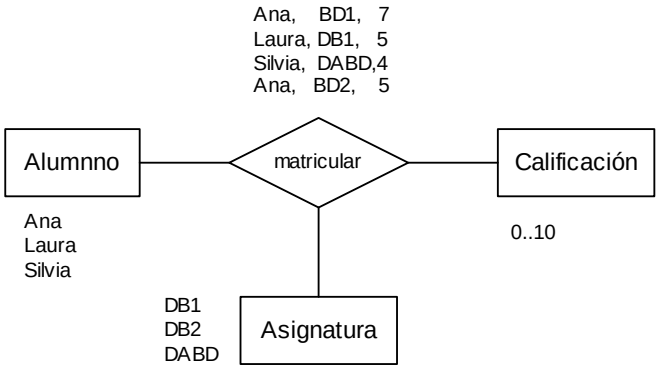


Figura 2.21: Ejemplo de interrelación ternaria y las instancias asociadas.

Pero además, si observamos las instancias de la interrelación *matrícula* comprobamos que la combinación entre una instancia de alumno y una instancia de asignatura es única. Esta restricción llevada al modelo relacional sería una dependencia funcional que aportaría una restricción estructural en el esquema. Sería importante, por tanto, recoger la restricción entre combinaciones de instancias de cada dos entidades que participen en la interrelación. En nuestro caso, tendríamos que comprobar tres restricciones; la combinación entre asignatura y alumno, alumno y calificación y, por último, entre asignatura y calificación. Pero existe otro problema, esta combinación es una combinación potencial de instancias que "podrían" pertenecer a la interrelación o una combinación que sabemos que pertenece a la interrelación. Esta discusión esta expuesta en Génova, Llorens & Martínez (2001) y Cuadra et al (2002).

Podemos distinguir dos vertientes para hallar las restricciones de cardinalidad, la aproximación de Merise, que procede la metodología propuesta por Tardieu et al (1983), que calcula la cardinalidad a través de cada tipo de entidad o componente de la interrelación. Y la aproximación de Chen que la realiza a través de la combinación. Por lo tanto podríamos distinguir cuatro interpretaciones distintas para las restricciones de cardinalidad en interrelaciones de grado superior: *look here* y *look across* con ambas aproximaciones.

Primera interpretación para interrelaciones de grado superior: Look here

Aplicando la aproximación de Merise a esta interpretación para una interrelación de cualquier grado se obtendría la participación de cada tipo de entidad.

La aproximación de Chen para un tipo de interrelación de grado n, cogería un vector de n-1 componentes, resultante del producto cartesiano de las instancias de las n-1 tipos de entidad, y contaría con cuantas instancias se relacionaría del tipo de entidad que resta.

Para el ejemplo presentado en la figura 2.21 estas cardinalidades se aplicarían de esta forma.

RC(LHm(Calificación)) = restricción de cardinalidad con la interpretación *look here* y la aproximación de Merise = número mínimo y máximo de veces que cada instancia de la entidad Calificación se encuentra en la interrelación *matricular*. En nuestro ejemplo, la cardinalidad mínima sería de 0, ya que no toda calificación aparece en la interrelación y la máxima sería *n*.

RC(LHc (Calificación)) = restricción de cardinalidad con la interpretación *look here* y la aproximación de Chen = fijar una instancia de alumno y otra de asignatura y contar el número mínimo y máximo de veces que esa combinación se relaciona con calificación. Así con cada combinación del producto cartesiano de instancias de alumno y asignatura. En el ejemplo, la cardinalidad mínima es de 0, ya que no toda combinación de instancias entre alumno y asignatura participan en matricula, y 1 ya que un alumno obtendrá una única calificación por asignatura.

En esta interpretación la restricción más fuerte es la impuesta por la cardinalidad mínima 1, ya que implicar la obligatoriedad de la participación de cada instancia o combinación de instancias en la interrelación.

Los modelos que adoptan la aproximación de LHm (*look here* Merise) son el modelo ER propuesto en la metodología de Merise (Tardieu et al., 1983). En el modelo ORM (Halpin, 2001) la restricción de role obligatorio (*mandatory role*) es la cardinalidad mínima de LHm. En ECR (Elmasri & Navathe, 2000) aunque aboga por la utilización de la máxima LHc comenta que para completar las restricciones estructurales de una interrelación de grado superior también debería incluirse la RC(LHm) denominadas en este modelo restricciones de participación.

En Ferg (1991) realiza una extensión en la representación para incluir tanto la aproximación de Chen como la de Merise con la interpretación de *look here* en los modelos de datos conceptuales sobre los lleva el estudio, que son IE (Martin, 1990), Merise (Tardieu, Rochfeld & Coletti, 1983) y ER (Chen, 1976). De esta forma recogen la restricción de participación de un tipo de entidad como la participación de n-1 tipos de entidades. También incluye la restricción de cardinalidad generalizada para poder recoger restricciones entre cualquier subconjunto formado por las componentes de una interrelación. Por supuesto incluye la conclusión que en binarias esta discusión carece de sentido y que además ningún modelos de datos conceptual recoge todas las restricciones de cardinalidad. Otra idea importante de esta investigación es la clasificación que realiza de las restricciones según incidan en la estructura de la base de datos o en las instancias de la interrelación.

En McAllister (1998) también engloba la definición *look here* para ambas aproximaciones pero introduce una representación tabular que permite recoger las cardinalidades mínimas y máximas de cualquier combinación de entidades con cualquier otra combinación de entidades que participen en una interrelación I. Para el ejemplo de la figura 2.21 tendríamos las siguientes cardinalidades según McAllister (tabla 2.12):

	Calificación (C)	Alumno (A)	Asignatura (As)	CA	CAs	AAs
C		Binarias impl.	Binarias impl.			LHm
A	Binarias impl.		Binarias impl.		LHm	
As	Binarias impl.	Binarias impl.		LHm		
CA			LHc			
CAs		LHc				
AAs	LHc					

Tabla 2.12: Restricciones de cardinalidad impuestas por McAllister (1998)

Por lo que para un tipo de interrelación de grado tres se tendrían las definiciones dadas anteriormente junto con las restricciones binarias implícitas dentro de la interrelación ternaria en un único formalismo, sin embargo, el número de cardinalidades a definir es muy alto (por ej. para tres entidades habría que definir 12 pares, para cuatro entidades 50 y para cinco 180), por lo que se proporcionan reglas que permiten en parte aliviar esta labor. Estas reglas están basadas en los axiomas de Armstrong (Armstrong, 1974) y a través de su aplicación pueden validarse las restricciones de cardinalidad. Por su parte, Thalheim (2000) analiza diferentes variantes de restricción de cardinalidad y propone la general *cardinality constraint* como una forma general de restricción de cardinalidad que subsume como casos particulares a todas las variantes expuestas hasta ahora, sin embargo, apenas se comentan las implicaciones de cada una de ellas en el proceso de diseño.

Segunda interpretación para interrelaciones n-arias: Look Across

Esta aproximación aplicada a las interrelaciones de grado superior refleja el número de veces que aparece un determinado valor en un role o componente de un tipo de interrelación adoptando la aproximación de Merise.

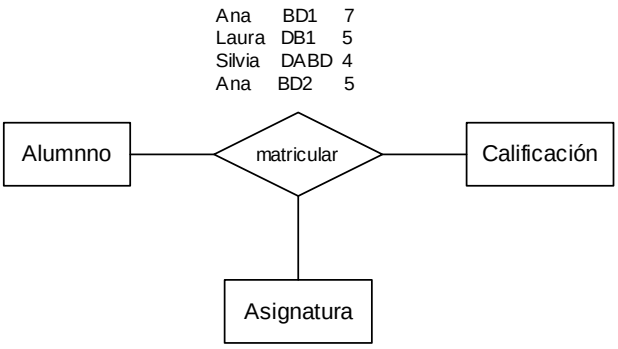


Figura 2.22: Cardinalidades look across en una interrelación n-aria

En el ejemplo de la figura 2.22 se observa que hemos quitado las instancias pertenecientes a los tipos de entidad ya que al utilizar esta aproximación solo nos debemos de fijar en las instancias de las que consta el tipo de interrelación. Con la aproximación de Merise, tomaríamos, por ejemplo, la componente o role correspondiente a la entidad *Calificación* y contaríamos el número mínimo y máximo que aparece cada valor. En este caso el número mínimo sería 1 y el máximo n al no conocer ninguna restricción impuesta por UD, RC(LAm(Alumno)) = (1,2). Esta aproximación no recoge semántica diferente a la LHm cuando tratamos con la cardinalidad máxima, para la mínima, la mayoría de los modelos prefieren utilizar la interpretación anterior porque con ella obtienen la información sobre si la participación de instancias es obligatoria u opcional para cada tipo de entidad que participa en la interrelación.

Para la aproximación de Chen con esta interpretación debemos coger una combinación de n-1 instancias, que en este caso deben estar relacionadas, es decir deben participar en la interrelación, y contar el número mínimo y máximo de veces que aparecen relacionadas con valores de la componente o role correspondiente a la entidad que resta. Esta restricción de cardinalidad se diferencia de la anterior en que la combinación de instancias deben participar en la interrelación, en Genova, Llorens & Martinez (2001) esta interpretación la denomina *actual tuple*.

Aplicando esta restricción al ejemplo de la figura 2.22 para la entidad o más bien para el role con el que actúa la entidad Calificación, tomaríamos la combinación *Ana*, *BD1* y

contaríamos cuántas veces aparece en matricular, en este caso, RC(Lac(Calificación)) = (1,1).

Dentro de esta aproximación existe un matiz muy importante que incluso puede condicionar la definición de interrelación. ¿Qué significado tiene la cardinalidad mínima de cero?. Conceptualmente la interpretación que se le daría a esta restricción se correspondería con la existencia de combinaciones de n-1 instancias dentro de la interrelación que se asocian con un valor desconocido o no aplicable.

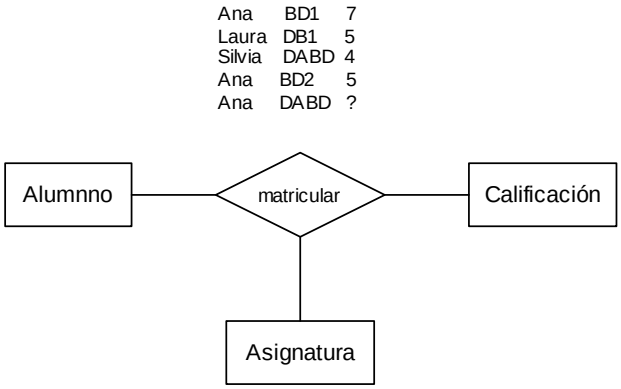


Figura 2.23: Instancias de una interrelación con valores nulos

En el ejemplo de la figura 2.23 la RC(LAc(Calificación)) = (0,n), semánticamente tiene sentido ya que la restricción mínima de cero se correspondería con aquellos alumnos que están matriculados en una asignatura pero todavía no han sido calificados. Estructuralmente estaría justificado si se permitiera definir un tipo de interrelación como un conjunto de n componentes cuyos valores podrían ser nulos (Cuadra et al, 2002).

Existen distintas aproximaciones que recogen restricciones dentro de una interrelación de grado superior que no se pueden reflejar con las cardinalidades estudiadas, por ejemplo restricciones solo entre un subgrupo de componentes de la interrelación. En Jones & Song (2000) incluyen interrelaciones binarias implícitas o explícitas para completar la semántica de la interrelaciones ternarias, el problema es que no considera las dos aproximaciones. En McAllister (1998) sin embargo recoge ambas

aproximaciones pero solo toma la interpretación de *look here* todas las restricciones asociadas a cualquier combinación de roles o componentes de cardinalidad a través de una tabla, para cualquier grado de la interrelación.

2.4. Aplicando una metodología en el desarrollo de bases de datos

Como en cualquier metodología de desarrollo podemos elegir principalmente entre un diseño descendente o ascendente o una combinación de ambos denominado mixto. El diseño descendente para el desarrollo de bases de datos relacionales parte de un conjunto de atributos individuales y considera las relaciones existentes entre ellos. El diseño ascendente comienza con agrupaciones de atributos en relaciones que se han obtenido en el diseño conceptual por la aplicación de las reglas de transformación.

Por eso esta sección la hemos dividido en dos partes muy importantes en el desarrollo de bases de datos. La primera de ellas considera que el diseño de bases de datos comienza a partir de la especificación de un esquema relacional del UD que deseamos modelar. Para realizar un buen diseño relacional se aplica la teoría de la normalización que asegura que no existan redundancias en el esquema. En la segunda aplicamos el diseño ascendente y a partir de un buen esquema conceptual aplicamos reglas de transformación para pasar a un esquema relacional. Estas reglas serán estudiadas y discutidas en esta sección.

Con la aparición de los modelos conceptuales a mediados de los 70 la teoría de la normalización ya no es el principal paso en el diseño lógico. Trabajando primeramente con un modelo de datos conceptual se producen directamente esquemas en tercera formal normal. Por lo que desde este punto del diseño, la normalización es solo utilizada como instrumento para borrar ciertas anomalías cuando transformamos un esquema conceptual en uno lógico.

2.4.1. Metodología a partir del diseño relacional

Es importante destacar que antes de que aparecieran los modelos conceptuales o más bien antes de que se instauraran como un paso previo al diseño relacional, la teoría de la normalización resultó el núcleo principal del diseño de bases de datos relacionales. La teoría de la normalización surgió de la necesidad de poseer un método formal para evitar redundancias, pérdidas de información y estados inconsistentes de la base de datos. Para conseguir estas condiciones se apoya en las formas normales que no son mas que restricciones que deben de cumplir un conjunto específico de atributos y restricciones que se apoyan en las dependencias funcionales.

Caben destacar dos escuelas a la hora de aplicar la teoría de la normalización: la que propugna los métodos llamados de análisis o descomposición, y la que aboga por el procedimiento de síntesis.

El método de descomposición fue propuesto por Codd (1970), el cual va descomponiendo una relación hasta tercera formal normal, posteriormente Rissanen (1973) desarrolla formalmente el método y Fagin (1977) y Zaniolo (1981) lo generalizan incluyendo dependencias multivaluadas y de combinación. La característica principal de este método es que se parte de todos los atributos del UD y las dependencias existentes entre ellos y se van descomponiendo para que se vayan cumpliendo las formas normales.

En 1976, Bernstein propone una alternativa denominada método de síntesis que recorre el camino inverso al del método anterior. Es decir, obtiene relaciones a partir de un conjunto de atributos y dependencias funcionales. Ambos métodos tienen la misma finalidad pero con el algoritmo de Bernstein solo nos asegura hasta la tercera forma normal (3 FN).

El proceso de normalización sigue los pasos siguientes:

1. Cálculo de las dependencias funcionales, multivaluadas, jerárquicas y en combinación.

2. Cálculo del recubrimiento minimal.

3. Cálculo de las claves candidatas distinguiendo entre atributos principales y no principales.

4. Cálculo de la forma normal en la que se encuentra la relación.

5. Aplicar los métodos de análisis o de síntesis.

A continuación pasamos a explicar cada una de los pasos teniendo en cuenta el primero de ellos fue visto en la sección 2.3.1 y que solo requiere la observación del UD para extraer las dependencias que en él se hallan.

Cálculo Recubrimiento minimal

M es un recubrimiento minimal si

- ✓ Todas sus dependencias son elementales.
- ✓ No existen atributos extraños. Un atributo A perteneciente a X es extraño en la dependencia X→Y si la dependencia (X-A) → Y se deduce del resto de dependencias de la relación mediante la aplicación de los axiomas de Armstrong.
- ✓ No existen dependencias redundantes. Una dependencia *dp* es redundante si sus implicados se deducen a partir del resto de las dependencias de la relación.

Cálculo claves candidatas

El cierre transitivo de X se define como el conjunto de atributos determinados por X aplicando los axiomas de Armstrong y se denota como X^+. Se denomina clave candidata de una relación a una superclave tal que ningún subconjunto de la misma determina a todo los atributos de la relación.

Formas normales

Existen seis niveles de normalización de una relación y de forma simplificada pasaremos a definirlas.

La primera forma normal (1 FN) es una restricción inherente del modelo relacional en la cual no se permite que un atributo para una misma tupla tome distintos valores. También podríamos decir que el cruce de una fila y una columna de una relación debe contener un único valor.

La 2FN de una relación debe cumplir que esta relación se encuentre en 1FN y que cada atributo no principal (no forma parte de una clave candidata) tiene dependencia funcional completa con respecto de una clave.

Una relación de encuentra 3FN si se encuentra en 2FN y no existe ningún atributo que dependa transitivamente de alguna clave de R.

La forma normal de Boyce-Codd (FNBC) se cumple si y solo si todo determinante es clave candidata.

Se dice que una relación se encuentra en 4FN si y solo si las únicas dependencias multivaluadas no triviales son aquellas en que la clave multidetermina un atributo.

Y por último, una relación se encuentra en 5FN si está en 4FN y toda dependencia de combinación está implicada por una clave candidata.

Una vez que hemos definido las formas normales lo único que nos queda es realizar el proceso de descomposición o de síntesis para obtener un esquema relacional normalizado.

El proceso de descomposición termina cuando las relaciones se encuentran en la forma normal objetivo o cuando la continuación en la descomposición supone pérdidas no deseadas de dependencias.

Los pasos a seguir son los siguientes:

1. Hallar el recubrimiento minimal.
2. Hallar las claves.
3. Identificar la FN en la que se encuentra la relación.
4. Agrupar las dependencias funcionales que tengan el mismo implicante o equivalente y obtener proyecciones independientes con estos grupos de dependencias.
5. Tratar las dependencias multivaluadas (descomponiéndolas o eliminándolas).
6. Tratar las dependencias en combinación. Si existe alguna que no esté implicada por la clave, habrá que descomponerla.

Para el proceso de síntesis los dos primeros pasos coinciden con el de descomposición pero no tiene en cuenta las dependencias multivaludas ni las de combinación por lo que los pasos a seguir son:

1. Hallar el recubrimiento minimal.
2. Agrupar las dependencias funcionales en particiones que tienen el mismo implicante.
3. Crear un esquema de relación para cada partición, que tenga como atributos todos los que participen en las dependencias y como grupo de dependencias las del grupo.
4. Si existen atributos que no son implicantes ni implicados en el cierre, formar un esquema de relación con éstos sin dependencias o alternativamente crear un esquema con la clave de la relación y sin dependencias.

Cabe destacar que este proceso solo asegura hasta la FNBC como ya habíamos comentado anteriormente. También mencionar que aunque no es objeto de nuestro estudio existen diversos algoritmos para automatizar todo el proceso de normalización aunque siempre debe haber intervención del diseñador para indicar la semántica del UD a modelar.

Lo que más nos interesa de la teoría de la normalización en este apartado es como método que se aplica dentro de la metodología de desarrollo de bases de datos relacionales sin pasar por la fase de modelado conceptual de la cual hablaremos a continuación.

2.4.2. Metodología a partir del diseño conceptual

Esta metodología se caracteriza por tener una fase conceptual como una de los primeros pasos a realizar para desarrollar una base de datos. Una de las propuestas más importantes apareció en 1986 y fue realizada por Teorey, Yang y Fry. En ella argumentaban el beneficio que se obtiene al utilizar el modelo ER como herramienta para el análisis de requisitos. La teoría de la normalización es demasiado compleja cuando la base de datos es muy grande, si se desarrolla un esquema conceptual previo a través del modelo ER se reduce el número de objetos y sus relaciones por lo que la fase de análisis se simplifica. Que el número de elementos para manejar sea menor viene provocado por el nivel de abstracción que proporcionan los modelos de datos conceptuales. Por tanto, el desarrollo de una base de datos con una primera fase conceptual hace que la resolución del problema sea abarcable, el esquema obtenido servirá además, como herramienta de validación con los expertos del dominio y usuarios finales y se simplifica la normalización.

A continuación presentamos los pasos básicos de toda metodología de bases de datos:

Paso 1: Especificación de requisitos. Este paso se realizará siempre con un modelo de datos que hemos denominado de segunda generación y que se corresponden con los modelos de datos conceptuales. Será el diseñador el que elija el modelo de datos a utilizar. Como ya estudiamos en la sección 2.2 existe una gran parte de constructores que son comunes a todos los modelos conceptuales, por lo que la elección de uno u otro dependerá de la complejidad para manejarlos y la amplitud que muestren para reflejar ciertas restricciones del UD. No existen unos criterios ni unas medidas para saber cual es el modelo más adecuado según las características del dominio. También en esta fase

además de construir el esquema conceptual de base de datos global pueden definirse las vistas que se necesitarán cada grupo de usuarios.

Paso 2: Transformación de un esquema conceptual a un esquema relacional. Este será el paso donde más nos detendremos en esta sección ya que forma parte de la propuesta de este trabajo de tesis doctoral. En la siguiente sección explicaremos en que consiste esta transformación y las distintas propuestas realizadas. Lo que podemos adelantar es que al contrario del paso anterior en el que el modelo de datos no se especifica en este paso el modelo está definido; el modelo relacional. ¿Porqué? Pues por que actualmente es el mas extendido. Además, se han incorporado mecanismos muy potentes en este modelo para recoger más semántica, como por ejemplo los disparadores y la creación de objetos, y sobre todo que este modelo encontrará soporte muy eficiente en los SGBD objeto – relacionales que existen actualmente en el mercado.

Paso 3: Normalización de las relaciones. Algunas de las dependencias funcionales las derivaremos de las restricciones impuestas en el modelo de datos conceptual. Pero aquellas dependencias que no podamos recoger en el esquema conceptual serán añadidas en este paso. El esquema relacional obtenido en esta fase no contendrá redundancias y preservará ña integridad de los datos.

Paso 4: Implementación. En esta fase se realizará el paso de del esquema relacional obtenido en la fase anterior a un esquema relacional específico para un SGBD. Por lo que este esquema puede ser modificado e incluso refinado para obtener mayor eficacia.

En general, el primer paso se denomina diseño conceptual, los dos siguientes pasos se corresponden con el diseño lógico y el último con el diseño físico, muy especifico del SGBD donde se implemente la base de datos.

En Kolp & Zimányi (2000) propone un método de diseño añadiendo la forma normal de inclusión basada en Goh (1992) que consta de las siguientes pasos:

- ✓ Correspondencia entre un esquema ER y esquema relacional según se expone en Elmasri & Navathe (2000), Goh (1992), Golshani (1992), Markovitz & Shoshani

(1992), Teorey (1990), Ullman (1982). Mostrando la dependencia de datos a través de las restricciones impuestas en el esquema ER.

- ✓ Normalización de las relaciones y generación de claves. Cada relación es descompuesta en relaciones en 3FN y al menos una clave debe ser encontrada aplicando el algoritmo de Bernstein (1976).

- ✓ Normalización de la base de datos: atributos superfluos y relaciones son eliminados.

Pero todavía se va mucho mas allá en la ampliación de la metodología inicial. Como hemos visto con el estudio de los modelos de segunda generación, la extensión de estos modelos no solo se encuentra en la inclusión de nuevos constructores y restricciones para enriquecer la semántica de los mismos, sino que inciden que es muy importante recoger el comportamiento de los distintos objetos y los cambios de estados que son válidos. Por tanto, no solo tendremos restricciones estáticas sino también dinámicas y además no podemos olvidar que una base de datos no es un sistema independiente sino que está rodeado de un entorno.

Surgen metodologías como la propuesta por Yassen & Thalheim (1989), Thalheim (1989), Thalheim (1991), donde se desarrolla la metodología *diseño-por-unidades* muy similar al diseño modular de ingeniería del software, orientada a objetos, que se centra tanto en la representación de los objetos como en la de procesos. Otra metodología orientada a objeto muy extendida en bases de datos con unas características semejantes es la propuesta por Rumbaugh et al (1991), OMT para el modelado y diseño orientado a objetos.

Otra propuesta orientada a objetos es la expuesta por UML mucho más revolucionaria. Al ser UML sólo un lenguaje formará parte un método de desarrollo de software, en nuestro caso de desarrollo de bases de datos. Es independiente de la metodología, aunque para utilizarlo óptimamente la metodología debe cumplir una serie de requisitos. Esta metodología debe utilizar procesos dirigidos por los casos de uso, ser centrada en

la arquitectura[3], iterativa e incremental. Un proceso es iterativo cuando este se pasa por las mismas fases varias veces, con un resultado cada vez más definido, con lo se depuran los errores y se minimizan los riesgos. El proceso incremental lo que realiza es sucesivas iteraciones, de cada una de las cuales resulta un producto cada vez más completo (proporciona más funcionalidades) y siempre, antes de pasar a la siguiente fase el producto se valida. El método, por lo tanto, distribuye los riesgos, a lo largo del proyecto, permitiendo aprender y actuar a tiempo.

UML es independiente de metodologías, por lo que puede ser usada (y lo es) en distintas metodologías como Fusion, Objectory, Rational Unified Process, OMT, ECM, Catalysys. La independencia antes mencionada permite que las organizaciones adapten el uso de UML a la metodología que consideren más apropiada. UML es un lenguaje para hacer modelos.

Nuestro trabajo de tesis doctoral no se centra en el estudio de una adecuada metodología por eso solo hemos expuesto algunas de las propuestas mas utilizadas siendo conscientes de que las metodologías con mas futuro son las orientadas a objetos. Al centramos en la representación de los datos, no creemos de importancia para nuestro trabajo la parte de procesos aunque será de vital importancia para completar el desarrollo de la base de datos. Por eso en la siguiente sección presentaremos los métodos propuestos para pasar de la fase conceptual a la fase de modelado relacional dentro de las metodologías de desarrollo de bases de datos relacionales, donde se incluyen distintas reglas de transformación que facilitan ese paso.

2.4.3. Métodos para el desarrollo de bases de datos relacionales

Cada una de las fases que componen la metodología en la que nos apoyamos para la presentación de nuestro trabajo de tesis doctoral consta de una serie de métodos para desarrollarla y para realizar el paso de una fase a la siguiente. Es común que cada fase

2 Centrado en la arquitectura: el diseño tiene como objetivo crear una estructura estable en el tiempo que puede modificarse para requisitos cambiantes. La estabilidad se consigue dividiendo la arquitectura en elementos.

se desarrolle de una forma iterativa e incremental y que no se pase a la siguiente fase hasta que no se realice una validación del producto desarrollado en esa fase.

Métodos de transformación general entre un esquema conceptual y un esquema relacional

Fase de modelado conceptual

Los objetivos principales en esta fase son delimitar el dominio especificando el universo del discurso (UD), describir la información a través de objetos y sus asociaciones.

Para conseguir los objetivos propuestos se propone utilizar las expresiones en lenguaje natural para delimitar el dominio y utilizar un modelo de datos conceptual para desarrollar un esquema que se ajuste a la especificación de requisitos.

En Chen (1976) se proponen una serie de métodos para detectar los objetos y sus relaciones en una especificación realizada en lenguaje natural. Un estudio mas exhaustivo es presentado por Batra & Zanakis (1994) donde se realiza una ampliación del método presentando un conjunto de reglas y heurísticas para desarrollar un esquema conceptual. En Halpin (2001) utilizan expresiones acotadas del lenguaje natural para traducirlas directamente a predicados de su esquema conceptual.

Aunque existen métodos para realizar un esquema conceptual siempre requieren de la experiencia del diseñador, de su creatividad y de su capacidad para resolver problemas.

Fase de modelado lógico

En este fase se transforma el esquema conceptual de la fase anterior, validado y refinado, a un esquema relacional aplicando un conjunto de reglas. Las reglas básicas, presentadas por Teorey, Yang & Fry (1986) se resumen a continuación:

1. Relación que contiene la misma información que el tipo de entidad que transforma. Esta transformación se da cuando la entidad participa en interrelaciones binarias N:M, 1:N si es la entidad que actúa con la correspondencia 1 en la interrelación o

1:1. Si la entidad participa en interrelaciones reflexivas N:M, en interrelaciones n-arias o si es supertipo o subtipo de una jerarquía también estará dentro de este caso.

2. Relación que contiene, además de la información del tipo de entidad que transforma, la clave ajena (*FK*) de otra entidad denominada padre dentro de la interrelación que las asocia. Esta transformación se aplica cuando la entidad participa en interrelaciones binarias 1:N con correspondencia N (entidad hijo).

3. Relación que proviene de la transformación de un tipo de interrelación, la cual contiene siempre como tantas claves ajenas como entidades asocie.

Como podemos observar con la aplicación de estas reglas el esquema relacional obtenido es muy pobre ya que solo tiene en cuenta los tipos de entidad y la correspondencia de los tipos de interrelación para optar por una transformación u otra. En los textos más relevantes dentro del diseño de bases de datos (Batini, Ceri & Navathe, 1992; Elmasri & Navathe, 1994; Korth & Silverschatz, 1991; Kroenke, 1992; Mannila & Räihä,1992) aparece una ampliación de estas reglas teniendo en cuenta las propiedades individuales de cada modelo de datos conceptual, por ejemplo; Elmasri & Navathe (2000) para la familia de modelos ER, Halpin (2001) para la familia de modelos objeto-rol, para la familia UML

Comentar que la pérdida de semántica es en general de casi todos los constructores, empezando por que tanto entidades como interrelaciones se transforman en un mismo constructor, la relación, y se pierde la separación entre los objetos de los cuales se quiere almacenar y las relaciones existentes entre ellos. Así como la pérdida evidente que incluye las abstracciones de los modelos de datos semánticos, como la generalización, la agregación, propiedades de conjuntos como la inclusión, la igualdad, etc…, y deberá ser el diseñador el que decida si toda esta semántica se añade a los procesos que acceden a la base de datos o se intentan almacenar con la base de datos. Lo que es indiscutible es que los requisitos que describen el UD a través del esquema conceptual deben conservase en las sucesivas fases de la metodología ya que si no fuera así, la implementación de la BD sería un subdominio del dominio a modelar. No es objeto de nuestro estudio proponer los mecanismos para la conservación de semántica

de todos los elementos proporcionados por los modelos conceptuales a través de las sucesivas fase de una metodología. El centro de la investigación son las restricciones de cardinalidad que como vimos en la sección 2.3.2 es uno de los elementos que más semántica recoge.

A continuación presentaremos la problemática encontrada en la transformación de la restricción de cardinalidad, primeramente porque en general no se tienen en cuenta las cardinalidades mínimas y en segundo lugar por la ausencia de reglas de transformación para interrelaciones de grado superior a 2 que distingan las distintas aproximaciones e interpretaciones presentadas en la sección 2.3.2.1.

Distintas aproximaciones en la transformación de restricciones de cardinalidad

En la propuesta de Teorey, Yang & Fry (1989) explica la transformación de interrelaciones dividiéndola en: interrelaciones binarias, reflexivas y n-arias de grado superior (n>2).

Las reglas de transformación para las interrelaciones binarias y reflexivas toman en cuenta el tipo de correspondencia y la participación de las entidades asociadas a través de la interrelación. Recalcamos que esta última restricción solo es recogida en determinados casos. Observemos la figura 2.24, la transformación de una interrelación binaria, la entidad que tiene asociada la correspondencia 1 se convierte en una relación con todas las propiedades asociadas a la entidad más una clave ajena de la otra entidad con la que se relaciona. Esta clave ajena permitirá valores nulos o no dependiendo de la participación opcional u obligatoria de la entidad. Pero ¿se contempla en el esquema la restricción de cardinalidad que índica *todo departamento debe tener asignado al menos un empleado*?. Esta restricción no es recogida en el esquema relacional, el modelo no proporciona ningún mecanismo para obligar a que todo empleado que se encuentre en la BD esté asignado a un departamento. Tampoco se contemplan las restricciones de cardinalidad mínima cuando transformamos interrelaciones con las características de la regla 3 (figura 2.25).

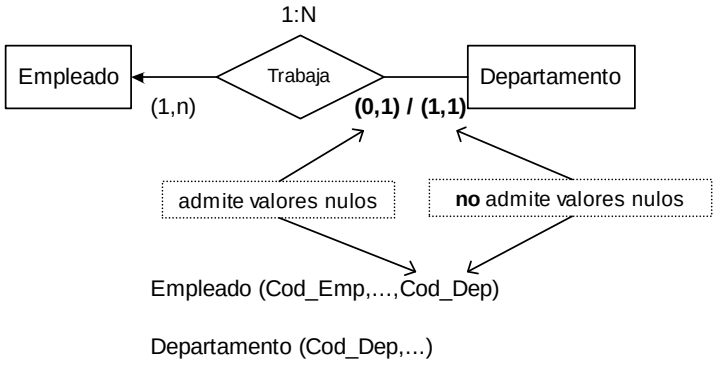


Figura 2.24: Transformación con propagación de clave de una interrelación binaria.

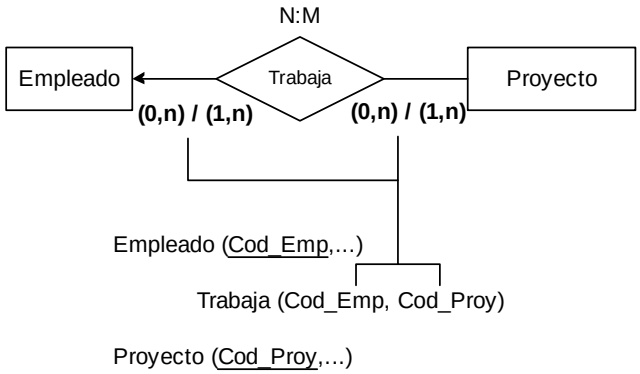
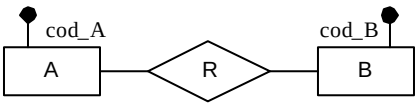


Figura 2.25: Transformación de interrelación binaria N:M

En Fahrner & Vossen (1995) se presenta un estudio de la transformación de interrelaciones binarias teniendo en cuenta la restricción de cardinalidad máxima y mínima, lo que en esta investigación se denomina transformación mediante la opción 1, Teorey lo describe con la regla 3, la opción 2 es explicada en la regla 2 e incluye una nueva transformación que se denomina de unión donde a través de una relación se recoge información tanto de las entidades como de la interrelación que las une (figura 2.26).



Opción 1

A (cod_A,...)
B (cod_B,...)
R (cod_A, cod_B,..)

Opción 2

A (cod_A,...,cod_B)
B (cod_B,...)

Opción 3

R (cod_A, cod_B,...)

Figura 2.26: Distintas opciones para transformar interrelaciones binarias.

En la tabla 2.13 presentamos un resumen de las distintas posibilidades para transformar interrelaciones binarias, poniendo siempre en primer lugar aquella que se recomienda y las restricciones que actualmente se añaden para preservar la semántica.

Cardinalidad asociada A y B	Opción	Restricción
(0/1,n) (0/1,n)	1	CC en R: (cod_A, cod_B) FK_A en R: cod_A FK_B en R: cod_B
(0,1) (0/1,n)	1	PK en R: cod_A NO en R: cod_B FK_A en R: cod_A FK_B en R: cod_B
	2	FK_B en A: cod_B O en A: cod_B
(1,1) (0/1,n)	2	FK_B en A: cod_B NO en A: cod_B
	1	PK en R: cod_A NO en R: cod_B FK_A en R: cod_A FK_B en R: cod_B
(0,1) (0,1)	1	CC en R: cod_A o cod_B NO en R: cod_A, cod_B FK_A en R: cod_A FK_B en R: cod_B
	2	FK_B en A: cod_B O en A: cod_B o FK_A en B: cod_A O en B: cod_A

Cardinalidad asociada A y B	Opción	Restricción
(0,1) (1,1)	2	FK_A en B: cod_A NO en B: cod_A
	1	CC en R: cod_A, cod_B NO en R: cod_A, cod_B FK_A en R: cod_A FK_B en R: cod_B
(1,1) (1,1)	2	FK_B en A: cod_B NO en A: cod_B **o** FK_A en B: cod_A NO en A: cod_B
	1	CC en R: cod_A, cod_B NO en R: cod_A, cod_B FK_A en R: cod_A FK_B en R: cod_B
	3	PK en R: cod_A NO en R: cod_B

CC : clave candidata
PK: clave primaria
FK_A: clave ajena que referencia a la relación A
O: opcional
NO: no opcional

2. 13: Transformación de interrelaciones binarias

Las restricciones de usuario del modelo relacional que se contemplan en la tabla son las aplicadas por la mayoría de los textos en Diseño de base de datos. Y como puede observarse la cardinalidad mínima 1 solo se contempla en algunos casos. En esta aproximación se engloba todas las posibles transformaciones que se pueden aplicar a las interrelaciones binarias dependiendo del tipo de correspondencia. Existen

interrelaciones a las que se le pueden aplicar distintas opciones de transformación y en la tabla aparecen ordenadas por la que es aplicada comúnmente, que en general se rige por evitar la presencia de valores nulos.

En la transformación de interrelaciones n-arias o de grado superior a 2 tenemos varios problemas añadidos. Por supuesto, se sigue arrastrando la pérdida de semántica asociada a un tipo determinado de restricción de cardinalidad, al transformar todo tipo de interrelación de grado superior como se indica en opción 1, figura 2.27, independientemente de los valores mínimos indicados por la restricción de cardinalidad. Sino que además, la mayoría de los libros de texto y artículos describen el proceso de transformación de una manera genérica y asumen una de las dos interpretaciones estudiadas en la sección 2.3.2, o la de Merise o la de Chen. Pero como se demuestra en McAllister (1998), Génova, Llorens & Martínez (2001), Cuadra, Castro & Martínez (2003), es importante expresar todas las cardinalidades posibles. Por lo que la transformación no ha sido nunca totalmente analizada. Estos problemas serán detallados a continuación.

Han sido varios los autores que han tratado de reducir la complejidad del problema de transformación al modelo relacional de interrelaciones n-arias, buscando su solución en el nivel conceptual, propugnando que toda interrelación n-aria se transforme directamente en el modelo conceptual en varias interrelaciones binarias establecidas con un tipo de entidad de enlace intermedia (Ullman & Widom, 1997; Dey, Storey & Barron, 1999). En sí misma, esta solución puede verse como caso particular de la regla 3. Sin embargo, realizar esa transformación en el nivel conceptual resulta prematuro y conlleva el riesgo de perdida de semántica, Silberschatz, Korth & Sudarshan (2001). Por otra parte, aquellos autores, Elmasri & Navathe (1994), Hansen & Hansen (1995), que tratan de manera diferenciada la interrelación n-aria en el modelo conceptual de lo que es su transformación al modelo relacional, habitualmente proponen como única solución. Esta solución es la propuesta por Teorey, Yang & Fry (1987) y que se muestra en la figura 2.26, aun cuando reconocen sin formalizarlo y centrándose solo en interrelaciones ternarias, que sería interesante tener en cuenta las interrelaciones binarias implícitas entre las entidades participantes.

Figura 2.27: Transformación general de interrelaciones n-arias

Si consideramos la transformación según las distintas interpretaciones debemos destacar que el trabajo de Teorey, Yang & Fry (1986) transforma el tipo de correspondencia que define Chen en su artículo Chen (1976); por lo que se correspondería con la $RC_{max}(LAc) = RC_{max}(LHc)$.

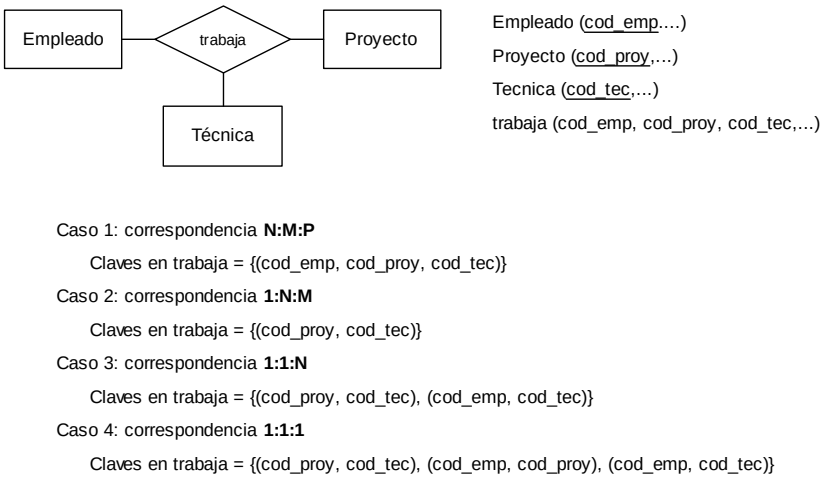


Figura 2.28: Transformación de interrelaciones ternarias dependiendo de su tipo de correspondencia.

Como se muestra en la figura 2.28 el tipo de correspondencia según la interpretación de Chen afecta al conjunto de claves candidatas de la relación, proveniente de la interrelación n-aria, en el ejemplo, de la interrelación ternaria. El diseñador será el que deberá elegir cual de ellas es clave primaria de la relación (*Primary Key*) y cuales son claves alternativas (*Unique*). Esto se traduce en un conjunto de dependencias funcionales en el modelo relacional y que es importante referenciar para realizar el proceso de normalización posterior.

En la aproximación de Merise (Tardieu, Rochfeld & Coletti, 1983) la transformación es muy parecida a la presentada en la figura 2.28, contempla el tipo de correspondencia al igual que en el caso anterior pero varía el conjunto de claves de la R por la diferencia en el significado de ambos tipos de restricción, figura 2.29.

Caso 1: correspondencia **N:M:P**

Claves en trabaja = {(cod_emp, cod_proy, cod_tec)}

Caso 2: correspondencia **1:N:M**

Claves en trabaja = {(cod_emp)}

Caso 3: correspondencia **1:1:N**

Claves en trabaja = {(cod_emp), (cod_proy)}

Caso 4: correspondencia **1:1:1**

Claves en trabaja = {(cod_emp), (cod_proy), (cod_tec)}

Figura 2.29: Conjuntos de claves asociados a la transformación del ejemplo con la aproximación de Merise.

Una de las propuestas mas interesantes que aúna ambas aproximaciones, Merise y Chen, es la presentada por Camps (2002). En su artículo presenta la transformación de interrelaciones ternarias tratando el tipo de correspondencia de ambas aproximaciones con la interpretación *look here* y trasladándola a un conjunto de dependencias funcionales. Aplicando los axiomas de Armstrong y teniendo en cuenta las restricciones asociadas a las interrelaciones binarias implícitas denominadas por Rumbaugh, Jacobson & Booch (1999) *semantically constraint binary o* SCB, construye un conjunto de patrones, donde se incluyen los estudios realizados por McAllister (1998), Thalheim (2000) y Jones-Song (2000). Para mostrar un resumen de este artículo hemos utilizado el esquema ER y su transformación al relacional siguiente, figura 2.30:

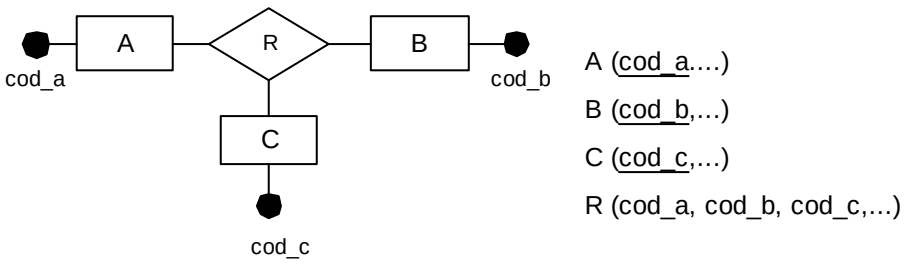


Figura 2.30: Caso general de transformación de una interrelación ternaria.

En la tabla 2.14 presentamos las restricciones de cardinalidad máxima derivadas de las dependencias funcionales que provienen de la aproximación de Chen, junto con la transformación que conserva estas dependencias y el número de patrón correspondiente en el artículo de Camps (2002).

	Chen	Merise	SCB	Camps	DF	Transformación
Caso 0	**N:M:P**	N:M:P	N:M	# 0	Ninguna	Caso general
			N:M			PK en R: (cod_a, cod_b, cod_c)
			N:M			
Caso 1	**1:N:M**	N:M:P	N:M	# 3	cod_b, cod_c --> cod_a	Caso general
			N:M			PK en R: (cod_b, cod_c)
			N:M			
Caso 2	**1:1:N**	N:M:P	N:M	# 2	cod_b, cod_c --> cod_a	Caso general
			N:M		cod_a, cod_c --> cod_b	CC en R: {(cod_b, cod_c), (cod_a, cod_c)}
			N:M			
Caso 3	**1:1:1**	N:M:P	N:M	# 1	cod_b, cod_c --> cod_a	Caso general
			N:M		cod_a, cod_c --> cod_b	CC en R: {(cod_b, cod_c), (cod_a, cod_c), (cod_a,cod_b)}
			N:M		cod_a, cod_b --> cod_c	

Tabla 2.14: Transformación cuando se fija la cardinalidad de Chen

Como se muestra en la tabla 2.14 las restricciones de cardinalidad máxima con la aproximación de Chen no influyen en el resto de las cardinalidades, es decir, con las dependencias que define no se pueden inferir más dependencias. Sin embargo, con la aproximación de Merise (tabla 2.15), se infieren tanto dependencias que influyen en la aproximación de Chen como en las SCB.

	Chen	Merise	SCB	Camps	DF	Transformación
Caso 4	N:1:1	**1:N:M**	N:1	# 6	cod_a --> cod_b, cod_c	Caso general
			N:M			PK en R: cod_a
			1:N			
Caso 5	1:1:1	**1:1:N**	1:1	# 5	cod_a --> cod_b, cod_c	Caso general
			1:N		cod_b --> cod_a, cod_c	CC en R: {(cod_a), (cod_b)}
			1:N			
Caso 6	1:1:1	**1:1:1**	1:1	# 4	cod_a --> cod_b, cod_c	Caso general
			1:1		cod_b --> cod_a, cod_c	CC en R: {(cod_a), (cod_b), (cod_c)}
			1:1		cod_c --> cod_a, cod_b	

Tabla 2.15: Transformación cuando se fija la cardinalidad de Merise

Las dependencias funcionales que define la aproximación de Merise son del tipo A→ (B,C) esto es equivalente a A→ B y A→C, de aquí se deducen las cardinalidades de las

SBC. Las cardinalidades de la aproximación de Chen vienen dadas por la aplicación de la regla de aumento de Armstrong que genera dos dependencias más: AC→B y AB→C.

Por tanto, la restricción de cardinalidad de Merise influye en las restricciones de cardinalidad de Chen y en las SCB. Por último, a través de la tabla 2.16, sepamos las restricciones asociadas a las SCB, que también repercuten en el resto de restricciones.

	Chen	Merise	SCB	Camps	DF	Transformación
Caso 7	N:1:M	N:M:P	**N:1**	# 7	cod_a --> cod_b	Transformación de la interrelación ternaria en dos relaciones
			N:M			
			N:M			
Caso 8	N:1:1	1:N:M	**N:1**	# 9	cod_a --> cod_b	
			1:N		cod_b --> cod_c	
			1:N		cod_a --> cod_c	
Caso 9	1:1:N	N:M:1	**1:1**	# 10	cod_a <-->cod_b	
			N:1		cod_c --> cod_b	
			N:1		cod_c --> cod_a	
Caso 10	1:1:N	N:M:P	**1:1**	# 8	cod_a <-->cod_b	
			N:M			
			N:M			
Caso 11	N:1:1	1:N:M	**N:1**	# 11	cod_a --> cod_b	
			N:1		cod_c --> cod_b	
			N:M			
Caso 12	**1**:1:1	1:N:M	**N:1**	# 13	cod_a --> cod_b	
			1:N		cod_b, cod_c --> cod_a	
			1:N		cod_a --> cod_c	
Caso 13	**1**:1:N	N:M:P	**N:1**	# 12	cod_a --> cod_b	
			1:N		cod_b, cod_c --> cod_a	
			1:N			

Tabla 2.16: Transformación cuando ciertas resticciones de cardinalidad en SCB.

Cada uno de los casos debería ser explicado con independencia ya que aunque la interrelación ternaria se transforma en dos relaciones binarias, según las dependencias funcionales y para la conservación de las mismas, se deberán añadir mecanismos que aseguren la consistencia de la base de datos. En Camps (2002) utiliza las aserciones como mecanismos relacionales para asegurar el cumplimiento de las DF. Cabe destacar que tampoco existe unanimidad en la transformación, para McAllister (1998) del caso 4 al 13 son interrelaciones ternarias que pueden ser descompuestas en binarias, sin embargo recientemente, Jones & Song (2000) cuestionan que los casos del 10 al 13 sean totalmente descomponibles. En resumen, podemos decir que esta aproximación es una de las mas completas pero sigue sin tratar todas las interpretaciones en la restricción de

cardinalidad (*look across*) y no le da demasiada importancia a la cardinalidad mínima, solo se centra en los tipos de correspondencia.

Basándose en las propuestas de Lazarevic & Misic (1991) y Bouzeghoub & Metais (1991) donde se realiza una extensión del modelo ER con restricciones de integridad referencial a través de acciones asociadas a las operaciones de actualización, en la primera de ellas, y la especificación de diversas restricciones de integridad y reglas de comportamiento en la segunda, Balaban & Shoval (2002), realiza una extensión del modelo ER introduciendo métodos para asegurar la consistencia de las restricciones de cardinalidad cuando se realizan operaciones de actualización. En esta aproximación se distingue entre las restricciones asociadas a interrelaciones binarias, donde solo se construyen métodos para la RC(LHm) y restricciones para interrelaciones de grado superior, donde se consideran las RC(LAc). Justifica su decisión en la amplia variedad de modelos que las utilizan aunque no descartan en un futuro ampliar la propuesta a cualquier tipo de restricción.

2.5. Herramientas Case

El acrónimo CASE, *engineering software assistant computer*, implica dos aspectos: la ingeniería del software y la asistencia de un ordenador para realizar distintas funciones dentro de la ingeniería del software. La ingeniería del software requiere actividades de análisis, diseño, implementación, y mantenimiento de sistemas de información, a las cuales podemos añadir las tareas complementarias de verificación, evaluación y de todas las decisiones que han sido tomadas y productos generados durante el ciclo de vida del proyecto. La asistencia del ordenador conforma todos los posibles apoyos que un ordenador puede aportar para facilitar el manejo del proyecto y la documentación, para controlar la complejidad de un diseño, y razonar sobre las especificaciones y modelos.

En esta sección hablaremos solo de herramientas Case para el desarrollo de bases de datos ya que es el marco principal de esta tesis doctoral. Se introducen dentro de este

marco para soportar, facilitar y mecanizar las fases de las que se compone una metodología de desarrollo de bases de datos (Bouzeghoub, Kedad & Métais, 2000).

2.5.1. Herramientas Case para el desarrollo de bases de datos

Como hemos venido comprobando durante la elaboración de este trabajo, una de las dificultades encontradas es que durante la fase de diseño conceptual el proceso de abstracción con el que debe trabajar el diseñador requiere de su creatividad, de su experiencia y de su inteligencia. Aunque los modelos conceptuales intentan ser sencillos, la elaboración de un esquema conceptual no es una tarea fácil y que requiere de varios pasos para asegurar la validez del esquema. Estos pasos son la composición, la extensión y la descomposición.

La mayoría de las herramientas CASE, dentro del modelado conceptual, contempla la fase de composición proporcionando para ello un interfaz gráfico que se apoya en un modelo de datos. Las deficiencias encontradas durante la fase de composición de un esquema conceptual es que siempre se presentan versiones reducidas de los modelos de datos, es decir, no recogen todos los elementos que componen el modelo de datos que dicen implementar y aún más importante, incluyen elementos que pertenecen a la fase lógica, como por ejemplo claves ajenas y sus opciones de borrado y modificación que claramente son elementos relacionales. Para la fase de extensión del esquema son bastantes recomendables estas herramientas ya que pueden modificar esquemas de forma muy sencilla pero seguimos encontrando limitaciones en el modelo de datos empleados ya que en su mayoría modelos binarios disminuyendo las capacidades de representación. Para la fase de descomposición de un esquema debemos proporcionar una serie de reglas para validar el esquema obtenido, tanto sintáctica como semánticamente, y un dialogo efectivo con el usuario.

Llegados a este punto podemos establecer una primera diferenciación entre las herramientas comerciales (Erwin, EasyCase, Excelerator, Oracle Designer,...) y los prototipos de investigación, aun cuando algunos de estos haya alcanzado rango de herramienta comercial.

Entre las herramientas comerciales, si bien es cierto que algunas de ellas adoptan enfoques más profesionales, también es cierto que muchas no pasan de ser simples utilidades para dibujar, disponiendo a lo sumo de una BD para el almacenamiento no redundante de los objetos que se van definiendo. Muchas de ellas no disponen siquiera de un soporte metodológico o no son lo suficientemente estrictas en su aplicación, con lo cual el diseñador no encuentra el camino correcto para realizar su tarea.

Por contra, los prototipos de investigación han sufrido los problemas derivados de una elevada especialización, su enfoque se ha centrado en determinadas problemáticas, dejando de lado el resto del proceso. En (Ram, 1994) se establece un marco para la comparación de herramientas de diseño de BD y se revisan, comparan y catalogan una gran variedad de ellas. Uno de los aspectos claves en tal comparación se refiere a los medios de representación empleados. Mientras que un gran número de herramientas dispone de interfaces gráficas, pocas son las que inciden en suministrar un diálogo con el usuario usando los medios que este emplea para su expresión, i.e., descripciones textuales en lenguaje natural. Véanse SECSI (Bouzeghoub & Gardarin, 1984), OICSI (Rolland et al, 1992), NLDA (Columbetti et al, 1985) y ANNAPURNA (Eick, 1984) como trabajos pioneros y NL-OOPS (Mich, 1996) y COLOR-X (Burg, 1996 y Burg, 1997) como alguno de los trabajos más recientes. Una de las herramientas expertas más prometedoras es View Creation System (Storey & Goldstein, 1993). En todos estas investigaciones se detectan cinco deficiencias: no existe garantía de normalización en la transformación de esquemas conceptuales a esquemas relacionales, no existen mecanismos que prevengan las asociaciones redundantes, las herramientas se decantan por los modelos binarios y podrían violar la cuarta formal normal, no se basan en los fallos detectados por los diseñadores poco expertos y no existe una validación empírica de las mismas.

Tanto para las herramientas comerciales como para los prototipos de investigación, las críticas realizadas sobre el conjunto de las herramientas CASE en el sentido de que no se adecuan al propósito con que fueron diseñadas y por tanto no pueden emplearse para la construcción de sistemas de información integrados pueden concentrarse en los siguientes puntos:

- ✓ No abarcan el ciclo de diseño completo.
- ✓ No emplean técnicas avanzadas de representación.
- ✓ No exhiben comportamiento inteligente.
- ✓ No son metodológicamente estrictas.
- ✓ No contemplan diversas notaciones y metodologías, carecen de adaptabilidad.
- ✓ No favorecen la reutilización.

Un estudio mas detallado acerca de las herramientas CASE comerciales aparece en Castro et al (2002). A continuación presentaremos el proyecto CICYT desarrollado en el grupo de trabajo LABDA del departamento de Informática de la Universidad Carlos III, que intenta paliar algunas de las deficiencias encontradas en el estudio de las herramientas CASE, añadiendo tanto el conocimiento adquirido por la docencia impartida, como el recogido por un experimento realizado a los alumnos a los que imparte las asignaturas de Diseño de Bases de Datos.

2.5.2.Pandora Case

El objetivo del proyecto PANDORA es desarrollar un entorno CASE para desarrollo de BD en todas las fases (modelado conceptual, diseño lógico e implementación) y para aprendizaje de los conceptos necesarios para realizar esta labor. La experiencia recogida en docencia de BD así como en la tecnología de BD, ha permitido detectar cuáles son los principales problemas a los que se enfrentan tanto los alumnos como los profesionales en el desarrollo de BD. Cómo se ha mostrado en la sección anterior de esta memoria, los entornos CASE existentes en la actualidad carecen de algunas capacidades (conductor metodológico, cobertura del ciclo de vida de una BD, etc.) que es necesario contemplar en este tipo de tecnología. Por ello, los objetivos concretos de PANDORA son:

- ✓ Obtener una herramienta CASE que cubra las todas las fases del ciclo de vida del desarrollo de BD (análisis, diseño e implementación). No hay que olvidar que la fase de análisis de requisitos apenas es considerada en las herramientas CASE actuales.

- ✓ Que estas fases se lleven a cabo utilizando un conductor metodológico que guíe a los usuarios en el desarrollo de BD.

- ✓ Incluir un componente didáctico que facilite el aprendizaje del diseño de BD y que además pueda utilizarse vía Web.

A continuación se describe la arquitectura del entorno PANDORA así como su funcionalidad.

El objetivo principal del proyecto PANDORA es realizar un entorno CASE con comportamiento inteligente que proporcione cobertura a las distintas fases del ciclo de vida del desarrollo de Bases de Datos relacionales y que incluya además un componente didáctico para usuarios noveles. La figura 2.31 muestra la arquitectura de PANDORA, que consta de un conjunto de módulos que pueden ser utilizados de forma independiente para llevar a cabo distintas actividades del desarrollo de una BD (también se muestran los Paquetes de Trabajo en los que se analizarán, diseñaran e implementarán cada uno de ellos).

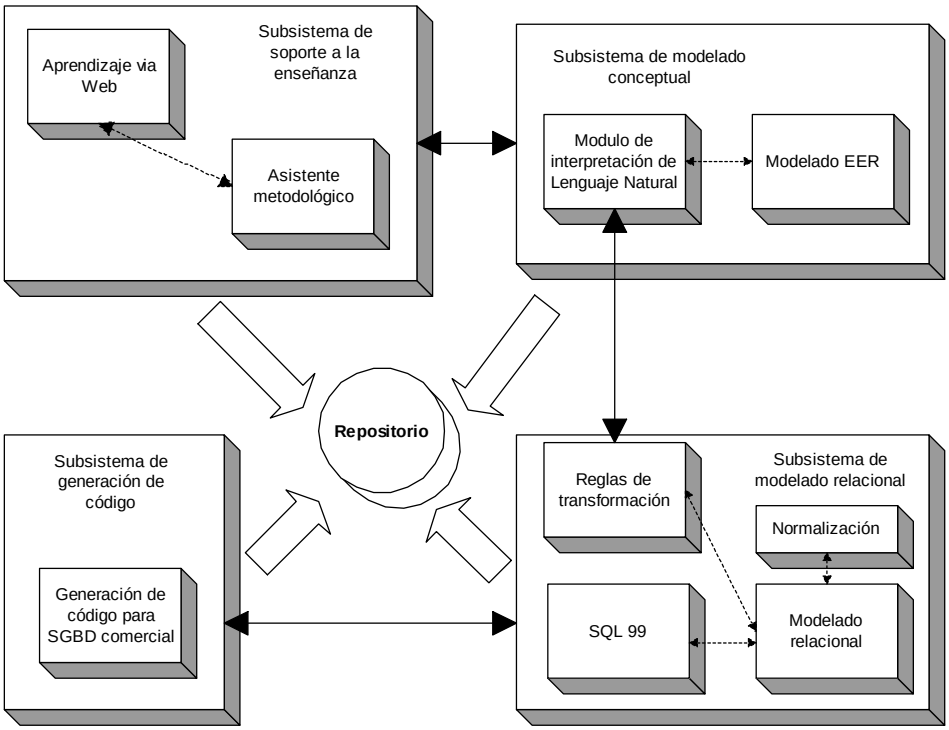


Figura 2.31: Arquitectura PANDORA

El núcleo principal de la plataforma CASE PANDORA es el repositorio (metabase), en donde se almacenan los recursos como los esquemas EER, esquemas relacionales, scripts SQL, disparadores, etc…El diseño del repositorio se ha dividido en dos metamodelos, uno para guardar los esquemas EER y otro para los esquemas relacionales, con el objetivo de distinguir las dos fases de la metodología, la conceptual y la lógica.

El subsistema de modelado conceptual está compuesto de dos módulos: el de modelado conceptual y el de análisis de lenguaje natural. El primero de ellos permite dibujar y verificar los esquemas conceptuales que el usuario a introducido a través de un interfaz gráfico. El modulo de análisis de lenguaje natural proporciona una propuesta de esquema EER a partir de las especificaciones introducidas en un texto (Martínez & Serrano, 2000). También incluye procesos de dialogo con el usuario para realizar la validación semántica del esquema conceptual, dando una especial tratamiento a la validación de las restricciones de cardinalidad, para el cual se propone la combinación de sintaxis (categorías gramáticas, colaboración entre palabras, etc…), semántica

(significado de las palabras, frases y sentencias) además de la lógica de primer orden para extraer la cardinalidad y validar esta con el usuario.

El subsistema de diseño incluye cuatro módulos: transformación de esquemas EER a esquemas relacionales, modelado relacional, algoritmos de normalización y generación de código SQL 3 (Melton & Simon, 2002). La principal contribución de este módulo es la inclusión de las restricciones necesarias (check, assertions, triggers, etc…) para la conservación de la semántica en la transformación de esquemas EER a esquemas relacionales. También incorpora un módulo de transformación experta en la cual el usuario puede ir supervisando paso a paso e interviniendo en la transformación.

En el subsistema de generación de código realiza la transformación de un esquema relacional al código en SQL específico del SGBD en el que se desee implementar la BD.

Y finalmente, el subsistema de soporte a la enseñanza proporciona una unificación para el entorno CASE en dos perspectivas. La primera, el módulo del tutor inteligente actúa como un asistente metodológico ayudando al diseñador en todas las fases del proceso de desarrollo de una base de datos. La segunda, el tutor inteligente incorpora estrategias de enseñanza y entrenamiento para el aprendizaje de los conceptos que envuelve el diseño de bases de datos a través de la Web.

PROPUESTA

Este capítulo consta de tres secciones principales. La primera describe la motivación que nos ha impulsado a realizar este trabajo de tesis doctoral. Motivación que proviene no solo de los antecedentes sino también de la experiencia docente, corroborada por experiencias empíricas, llevada a cabo por el grupo de trabajo de Bases de Datos Avanzadas. La segunda presenta el planteamiento del problema y por último, la solución que se ha propuesto.

3. HIPÓTESIS Y MOTIVACIÓN DEL TRABAJO

En este apartado vamos a explicar las razones que nos han llevado a realizar este trabajo de tesis doctoral apoyándonos en las propuestas presentadas anteriormente pero también en el diseño y análisis de un experimento llevado a cabo dentro de nuestro grupo de trabajo LABDA.

En resumen podemos indicar que dentro de una metodología de desarrollo de bases de datos, la fase de diseño conceptual es una de las mas relevantes y con más peso especifico. Se tiene evidencia empírica que indica que los errores, mal entendidos o inconsistencias, en las primeras fases de desarrollo conllevan un alto precio para realizar una corrección.

Hemos visto que los modelos utilizados, por tanto, en la primera fase de una metodología tienen la características de ser sencillos, simples y fáciles de aprender y manejar, con un lenguaje, generalmente, gráfico para poder validar el esquema con los expertos del dominio. Pero se han observado en distintos experimentos que existen distintas dificultades, que analizaremos en este capítulo, para la realización de la abstracción que exigen estos modelos sobre todo en aquellos diseñadores poco expertos.

A continuación pasaremos a estudiar las variables analizadas por distintos investigadores, para luego contar el experimento llevado a cabo por nuestro grupo de investigación.

3.1. Resultados empíricos acerca del modelado conceptual

En Batra & David (1992), se realiza un análisis de las diferencias existentes entre los diseñadores con distinta experiencia en el modelado. Un diseñador experto se

caracteriza por la descomposición de toda la descripción del problema en partes significativas, organizando toda la información concerniente a cada parte y traduciéndola a estructuras apropiadas. Cuando los expertos realizan el modelado, utilizan conocimientos adquiridos en su trayectoria sobre el dominio en cuestión para llegar a entender la aplicación, formulando preguntas en el caso de dudas acerca del problema. Además, tienen mayor riqueza de vocabulario con lo que son más capaces de entender aspectos específicos del problema y transformarlos en estructuras de la base de datos. Los diseñadores sin experiencia tienden a tener más errores en sus soluciones debido a su incapacidad para integrar varias partes de la descripción del problema en las estructuras adecuadas de la base de datos.

Goldstein & Storey (1990) afirman que educadores y profesores creen que los alumnos de asignaturas de bases de datos tienen gran dificultad en entender los concepto de este modelo y como consecuencia, les resulta complicado aplicarlos al mundo real motivo por el cual crean un herramienta de ayuda al diseño, View Creation, a partir de la cual analizan los fallos mas comunes en cuanto a la identificación de entidades, atributos e interrelaciones clasificados de la siguiente forma:

1. Utilización de un constructor en lugar de otro, sobre todo confunden los atributos de la interrelaciones.

2. Inclusión de elementos no necesarios. Interrelaciones binarias redundantes.

3. Omisión de elementos. Se evita la utilización de interrelaciones de grado superior.

Según estos autores, las razones por las que se cometen este tipo de errores son la incomprensión del modelo Entidad/Interrelación, la falta de conocimiento del UD y los lapsos mentales o falta de concentración.

En Batra & Antony (1994) extienden las causas por las que usuarios inexpertos comenten errores en la aplicación del modelo ER. Una de las causas es la incorrecta aplicación de heurísticas para detectar los elementos del modelo dentro de una especificación de requisitos. Según los autores, la aplicación de estas heurísticas son poco útiles, porque pueden llegar a producir errores sistemáticos. La segunda de las causas es la complejidad de la aplicación. En este punto, se coincide con el anterior experimento y se indica investigar en qué forma el conocimiento del dominio influye decisivamente en el correcto modelado de la base de datos. Lo más destacado de este experimento es que detecta que una de las tareas más difíciles para los individuos poco expertos es, dado un conjunto de entidades, identificar las interrelaciones existentes entre estas. Esto es debido a que para elegir bien las interrelaciones de las posibles entre un conjunto de entidades, se tienen que seguir unos criterios: que la semántica del dominio no se pierda, que estas interrelaciones no sean redundantes y que las interrelaciones siempre sean del menor grado posible, atendiendo a los otros dos criterios. En este experimento solo se toman en cuenta las interrelaciones con su grado y correspondencia, sin considerara las cardinalidades. Concretamente estos errores se clasifican en:

1. Grado mayor como menor. Cuando una interrelación debe de ser de mayor grado que como está modelada. La consecuencia de este error es que se introducen restricciones incorrectas.

2. Grado menor como mayor. Cuando una interrelación debe de ser de menor grado que como está modelada. Como consecuencia de este error, se produce redundancia y pérdida de semántica.

3. Correspondencia errónea. Se produce cuando en número de instancias de una entidad que se relacionan en una interrelación con otras entidades no es el correcto. Esto puede producir pérdida o exceso de restricciones.

Dentro de la clasificación de los errores, los que aparecen con mayor frecuencia son los errores debidos al grado de la interrelación mas que los errores por correspondencia incorrecta. Se tiende a modelar más interrelaciones de grado mayor como de grado menor que viceversa, interrelaciones binarias por ternarias. Y una buena proporción de los errores son causados por no considerar diferentes soluciones del problema una vez que se ha partido de una inicial.

La conclusión que sacan estos autores son que deberían existir herramientas para que los individuos puedan darse cuenta de los errores cometidos, y que muestren posibles situaciones iniciales del problema, es decir, un mecanismo de marcha atrás que compruebe si el punto de partida es correcto o los posibles caminos a seguir. Esta herramienta también debería comprobar los errores sintácticos por ejemplo la falta de conectividad entre algunas entidades. En definitiva, concluyen que es necesario la utilización de herramientas expertas para que esos errores se minimicen.

Mencionamos el experimento de Siau, Wand & Benbasat (1995) por que es uno de los primeros que analiza la restricción de cardinalidad. El experimento consiste en varios diagramas E/R en los que el individuo tiene que decidir si la cardinalidad de la interrelación es obligatoria u opcional. Los diagramas se dividen en dos tipos: diagramas con dominio familiar, que representan ejemplos del mundo real; y diagramas con dominio no familiar, en los que se muestran conceptos específicos desconocidos por los individuos de la muestra. Los resultados obtenidos muestran un uso predominante de la cardinalidad opcional frente a la obligatoria, incluso cuando el dominio es poco familiar y por supuesto la influencia del conocimiento del dominio en el modelado.

3.2. Experimentación para detectar errores en el tratamiento de interrelaciones ternarias

El estudio de los experimentos realizados para la detección de errores dentro del modelado conceptual se han basado en: una muestra de diseñadores inexpertos y estudian las causas y los tipos de errores en la representación de los principales elementos que constituyen el modelo Entidad/Interrelación. Aquellos experimentos que enfocan su estudio en las interrelaciones no consideran las cardinalidades sino el tipo de correspondencia y además la mayoría considera exclusivamente las interrelaciones binarias.

Por lo que los objetivos que se persiguen con este experimento son los siguientes:

- ✓ Si es costosa la detección de interrelaciones ternarias a partir de un texto dado.
- ✓ Si la interpretación de las cardinalidades asociadas a una interrelación ternaria es correcta.
- ✓ Y cuales son las causas por las que se desconoce la cardinalidad.

Para los cuales se ha diseñado el experimento con las siguientes características:

Unidad experimental: modelo E/R

Sujetos: el experimento se aplicará a alumnos de Ingeniería Técnica de Gestión que ya han cursado la asignatura de Diseño de Bases de Datos y alumnos de Ingeniería Informática que además de cursar la asignatura de Diseño donde se imparten los conocimientos del modelo E/R, la han aprobado.

Variable respuesta o variable dependiente: El resultado de un experimento siempre viene referido a una variable respuesta. Esta variable contará aquellos alumnos que detectan las interrelaciones ternarias, de entre estos los que reflejan

las cardinalidades asociadas y si no es así, cuáles son las causas. Lo errores se han tabulado como se muestra en la tabla 3.1.

Tipo de error	Peso
Ternaria completa	6
Ternaria con alguna máxima mal	5
Ternaria con alguna mínima mal	4
Ternaria con todas mal	3
Ternaria con cardinalidades desconocidas No tiene claro el concepto	2
Ternaria con cardinalidades desconocidas Otras causas	1

Las causas por las que se desconoce las cardinalidades se dividen en: 1) No se especifica claramente en el texto. 2)Desconoce el dominio y por eso le resulta difícil identificar las cardinalidades. 3) Tiene poco claro el concepto de cardinalidad. 4) Otras (indique cual).

Test	Dominio	Descripción	Tipo
1	Familiar	Discoteca	Ternaria sin atributo dentro de la misma frase.
2	No Familiar	Gestión de documentos legales	Ternaria sin atributo dentro de la misma frase
3	No familiar	Organización de Conferencia Internacional	No tiene ternarias
4	No Familiar	Revisión de publicaciones	Ternaria sin atributo en distintas frases
5	Familiar	Compañía telefónica	Ternaria con atributo
6	No familiar	Oficina de abogados	Ternaria con atributos
7	Familiar	Compañía de ferrocarril	No tiene ternarias
8	Familiar	Departamento medioambiental	Ternaria sin atributos en distintas frases

Parámetros: estos son características que permanecen invariables a lo largo de la ejecución del experimento y por lo tanto particularizan los resultados. El

cuestionario que se presenta a los alumnos tiene 8 textos para modelar con el E/R, estos 8 textos se muestran en el mismo orden en cada uno de los cuestionarios y poseen de forma individual distintas características.

Factores o variables independientes: características que se desean estudiar y que afectan a las variables respuestas o dependientes. En nuestro caso tenemos distintos factores: *dominio, presentación en el texto y ternaria con atributos.* Alternativas o niveles: posibles valores de los factores a estudiar. En este experimento tenemos para el dominio, familiar y no familiar, para la representación en el texto, en la misma frase o distinta, y por último para el atributo en ternarias, con o sin él.

Los resultados mas importantes obtenidos son presentados a continuación.

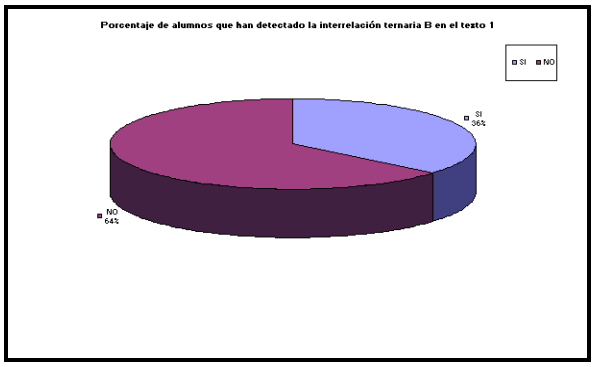


Gráfico3.1: Alumnos que detectan la IT en el texto 1

En el gráfico 3.1 vamos el porcentaje de alumnos que han detectado la interralición ternaria en el primer texto donde se supone que el dominio es familiar y la ternaria aparece dentro de la misma frase. Lo más significativo es que todos los alumnos que detectan la interrelación representa bien las cardinalidades.

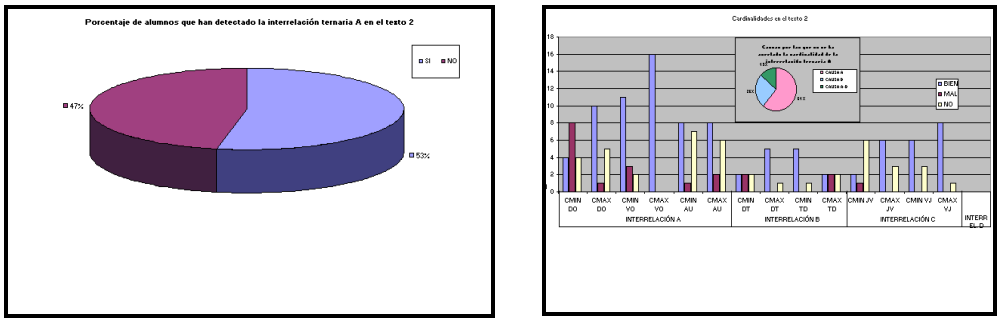


Gráfico 3.2: Alumnos que detectan las IT y porcentaje de aciertos en las cardinalidades

En el gráfico 3.2 se puede observar cuales han sido los resultados para el segundo texto. Aunque el texto se clasifica de dominio no familiar el porcentaje de aciertos es bastante alto por la colocación de las especificaciones dentro del texto.

Los resultados del texto tres no se presentan porque no son significativos para este estudio, y que en él no aparecen interrelaciones ternarias.

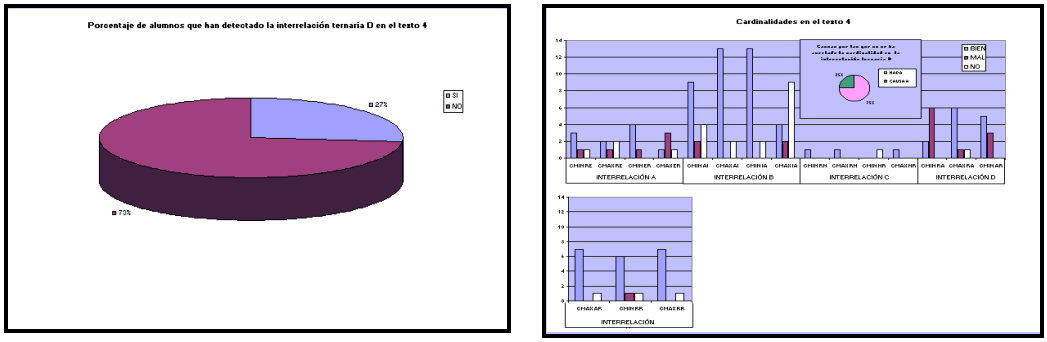


Gráfico3.4: Alumnos que detectan las IT y porcentaje de aciertos en las cardinalidades

El texto 4 es de dominio no familiar y esto se detecta en los resultados presentados en el gráfico 3.4.

4. PLANTEAMIENTO DEL PROBLEMA

El desarrollo de una base de datos viene dado por la percepción de un *sistema.* Un sistema no es absoluto, al ser una concepción en la mente de un observador, por lo que sus propiedades son subjetivas y asociadas al dominio del sistema al percibir este como un todo. De manera informal podemos definir un sistema como un conjunto de objetos relacionados entre sí. Los sistemas no tienen existencia propia sino que es necesario un observador que tenga propósito de concebir una parte del mundo real un sistema.

La necesidad de crear una base de datos viene dada por la representación formal de un sistema. En la figura 4.1 vemos los elementos que forman parte del proceso de creación de un sistema.

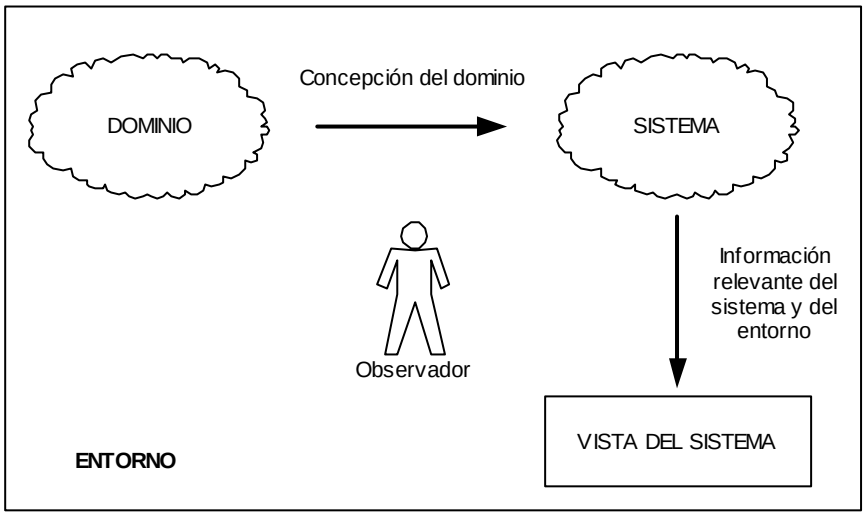


Figura 4.1: Elementos que componen la creación de un sistema

El elemento principal para la creación de un sistema es el *observador*, el cual concibe el *dominio* o área del mundo real como un sistema. Este sistema posee elementos relacionados y al menos una propiedad sistémica en relación con su entorno. La *vista del sistema* contiene información relevante acerca del sistema, de los dominios tanto del sistema como del entorno y las relaciones existentes entre ellos, necesaria para explicar el sistema.

Para realizar el proceso de concepción de un sistema se emplean los modelos. Un *modelo* es un formalismo sobre cómo describir una porción del mundo real, que solo expresa una parte de lo que puede decirse sobre su dominio y que tiene cierta utilidad. En la sección 2.1 y 2.2 presentamos los modelos de datos más utilizados en el desarrollo de una base de datos. Por tanto, podemos concluir que para formalizar un sistema es necesario el empleo de un modelo de datos.

¿Cómo realizamos esta formalización? ¿Por donde empezamos? Estas preguntas tienen su contestación a través del uso de métodos. Un *método* es una forma sistemática de trabajar para obtener el resultado deseado. Un ejemplo de resultado podría ser la implementación de un producto, la especificación de los requisitos de un sistema, o dentro del área de bases de datos, su implementación, su análisis o su diseño conceptual. En la mayoría de los casos los métodos nos proporcionan heurísticas para solucionar un problema. Una heurística es un consejo para solucionar un problema pero no garantiza llegar al resultado deseado.

Un método está compuesto de *técnicas* que irán cubriendo metas parciales hasta obtener el resultado final. Las técnicas son recetas para llevar a cabo tareas específicas que describen con detalle una forma de trabajar.

Una metodología es el estudio de los métodos. Una metodología de desarrollo de bases de datos es el estudio de métodos utilizados par obtener una base de datos.

Este es nuestro punto de partida, para poder definir nuestro sistema de bases de datos deberemos realizar una formalización a través de un modelo y para llevar a cabo este proceso nos serviremos de una metodología.

La terminología definida hasta el momento nos servirá para realizar un resumen de lo que hemos visto a lo largo de la presentación de los antecedentes y para plantear los problemas encontrados, que serán abordados en la propuesta de esta tesis doctoral.

Según la metodología de bases de datos, un sistema se describe a distintos niveles. El primer nivel es el mas abstracto y se denomina nivel conceptual. El resultado que debemos obtener es un esquema conceptual que describa nuestro sistema. Para ello utilizaremos un modelo conceptual y en general, no existe ningún método formal para construir el esquema conceptual.

¿Cuáles son las deficiencias encontradas en este nivel?

1. Derivadas de los modelos de datos conceptuales.

En un primer lugar una gran variedad de modelos ¿cuál de ellos elegimos?.

Los modelos denominados n-arios son mucho mas expresivos que los modelos binarios pero en general no tienen soporte en una herramienta CASE. ¿Adaptamos el esquema n-ario a uno binario o comenzamos directamente a modelar con un modelo binario?

En general, los elementos de los que se componen la inmensa mayoría de los modelos conceptuales son muy parecidos pero cada modelo le da un matiz o una interpretación con lo que se puede producir una gran ambigüedad. Esta ambigüedad se introduce tanto en la definición del elemento como en su representación gráfica. En este punto haremos una aclaración, el lenguaje utilizado por cada modelo de datos para definir los elementos de los que se compone, denominado ontologías, depende de cada modelo. Por ejemplo, UML es un lenguaje de modelado gráfico y por lo tanto la definición, es decir, el meta-elemento como el elemento o la representación de un elemento se realiza a través de una misma notación gráfica.

En general, los modelos de datos conceptuales utilizan lenguajes con formato gráfico lo que permite, sencillez a la hora de comunicarse con los expertos del dominio, realizar comprobaciones de consistencia y completitud y detectar errores léxicos.

¿Cuáles son los elementos más ambiguos? Como hemos visto en los distintos experimentos presentados en el apartado 3, el constructor que presenta más dificultad es la interrelación. Dentro de estas dificultades se encuentran, el problema de su detección dentro de un conjunto de especificaciones o requisitos, la determinación de sus propiedades: grado, conectividad o correspondencia y cardinalidad. Y por último, la validación de todas las interrelaciones dentro del esquema. La dificultad en el manejo de la interrelación aumenta según aumentamos el grado. Las interrelaciones n-arias además tienen el problema añadido de que la definición de cardinalidad no es única y cada una de las aproximaciones presentadas en la sección 2.3.2 recoge una semántica distinta. Por lo que la utilización de una u otra aproximación hace que la restricción de cardinalidad no sea completa.

2. Derivadas de la falta de método.

La construcción de un esquema conceptual se elabora a través de la experiencia y la audacia del diseñador, siempre apoyada en una serie de heurísticas. Los diseñadores poco expertos acusan la dificultad en la aplicación del proceso de abstracción, por lo que la aplicación de un modelo conceptual para la especificación de un sistema les resulta una tarea compleja.

Además esta tarea resulta mas compleja porque es difícil evaluar la calidad de un esquema conceptual, la detección de los errores y su validación.

El siguiente nivel de la metodología de desarrollo de bases de datos es el nivel lógico. El modelo de datos utilizado en este nivel es en general el modelo relacional, a diferencia de la proliferación de modelos existentes en el nivel anterior.

¿Cuáles son las dificultadas encontradas en este nivel?

En este caso no vienen derivadas del modelo ya que el modelo relacional está bien definido. La descripción de este modelo es a través de un lenguaje formal bien estructurado como es el utilizado por la teoría matemática de conjuntos. El problema encontrado es como pasar del esquema conceptual a un esquema relacional sin pérdida de semántica.

Esta transformación es guiada por una serie de reglas que se engloban dentro del método de transformación que nos da como resultado un esquema relacional en tercera formal normal.

El método de transformación es sencillo pero poco exhaustivo. Por ejemplo, para transformar una interrelación solo nos debemos de fijar en su correspondencia perdiendo mucha de la información recogida por la restricción de cardinalidad. Al igual que en el modelo conceptual esta perdida se hace más notable en las interrelaciones n-arias donde la mayoría de las técnicas presentadas solo dan una solución para sus transformación independiente de las restricciones de cardinalidad. Este problema proviene de que al no tener una definición clara de cardinalidad para este tipo de asociaciones no se puede presentar una técnica que contemple toda su semántica.

Después de la exposición de los problemas encontrados en los dos niveles presentados dentro de la metodología de desarrollo de bases de datos relacionales que dan un tratamiento global a la base de datos vamos a determinar las pautas que seguiremos dentro de la presentación de nuestra propuesta.

- ✓ Definición de las restricciones de cardinalidad para una interrelación n-aria apoyada en una definición formal de interrelación, presentada en la sección 5.1.

- ✓ Estudio de la validación y completitud de las restricciones de cardinalidad.

- ✓ Técnicas de transformación de las restricciones de cardinalidad definidas, sin pérdida de semántica, a un esquema relacional.

5. SOLUCIÓN PROPUESTA

La presentación de la propuesta, según hemos visto en el planteamiento del problema se dirige a las dos primeras fases, diseño conceptual y diseño lógico, de una metodología de bases de datos. Estos dos fases no son independientes, por tanto los resultados obtenidos en la primera fase repercutirán en la segunda.

En el diseño conceptual proponemos la utilización de un conjunto de restricciones de cardinalidad para que la semántica de las interrelaciones, sobre todo de grado superior, quede reflejada completamente. Para facilitar la compresión de estas restricciones, difíciles de entender y aplicar, presentamos sus definiciones a partir de la definición formal de interrelación y un conjunto de reglas para su validación.

En el diseño relacional se realizará un estudio exhaustivo de la repercusión de estas cardinalidades en las estructuras lógicas provenientes de la transformación de esquemas conceptuales en lógicos. Este estudio se basa sobre todo en las interrelaciones binarias y ternarias.

A continuación pasamos a exponer la propuesta de este trabajo de tesis doctoral.

5.1. Definición de interrelación

En esta sección presentaremos la terminología que utilizaremos para nuestra propuesta. Daremos una definición formal del concepto de interrelación, dando por sentado que cuando hablamos de entidad o interrelación nos referimos al concepto en si mismo, es decir, al tipo de entidad o de interrelación. Los elementos pertenecientes a una entidad o interrelación los denominaremos instancias o ejemplares de la misma.

Definición de entidad:

Sea $E = (A_1,\ldots,A_n, \{id_j(E) / 1 \leq j \leq n\})$ una entidad E formada por los atributos $A_1,\ldots,A_n$ cada uno de ellos definidos sobre un dominio donde el conjunto $\{id_j(E) / 1 \leq j \leq n\}$ contiene los identificadores o claves candidatas de la entidad E cuya propiedad es que caracterizan de forma unívoca cada una de las instancias de E.

Dentro de este conjunto podemos distinguir el identificador principal (IP) que se caracteriza de los demás en que es obligatorio. Es decir, una entidad debe tener uno y solo un IP. Debemos recordar que una de las restricciones inherentes a la mayoría de los modelos conceptuales es que toda entidad debe tener al menos un identificador o clave.

Definimos E^t como el conjunto de instancias de E. Un elemento e^t de E^t es un vector de s elementos donde la componente i-ésima $e_i \in dom(A_i)$ y tal que se cumple que para cualquier otra instancia e'^t y para cualquier identificador $id_j(E)$:

$$\Pi_{idj(E)}(e^t) \neq \Pi_{idj(E)}(e'^t) \text{ donde } \Pi_A(E) \text{ es la proyección de E sobre A.}$$

Definimos el conjunto de instancias clave de una entidad E como $\#E = \Pi_{IP}(E^t)$.

Definición de interrelación de grado n

Sea $R = (r_1E_1,\ldots, r_nE_j, A_{1,\ldots,} A_s)$ una interrelación de grado n con s atributos, donde cada r_i es el rol con el que participa la k-ésima entidad E_k en R.

Por tanto, el grado viene dado por el número de roles que participan en una interrelación y no por el número de entidades. En la figura 5.2(a) se muestra la representación gráfica de una interrelación de grado n. En el caso presentado por la figura, el grado coincide con el número de entidades, al no participar ninguna de ellas en la interrelación con varios roles.

Definimos R^t como el conjunto de instancias de R. Un elemento r^t de este conjunto es un vector de n componentes, donde cada componente representa un role y contiene una instancia clave de la entidad que participa con ese role. Por lo que, el conjunto de instancias R^t de una interrelación R es un subconjunto del producto cartesiano de las instancias claves de las entidades que participan en R y de los dominios de los atributos que pertenecen a R.

$$R^t \subseteq r_1E_1 \text{ x}\ldots\text{x } r_n E_j \text{ x dom}(A_1)\text{x}\ldots\text{x dom}(A_s)$$

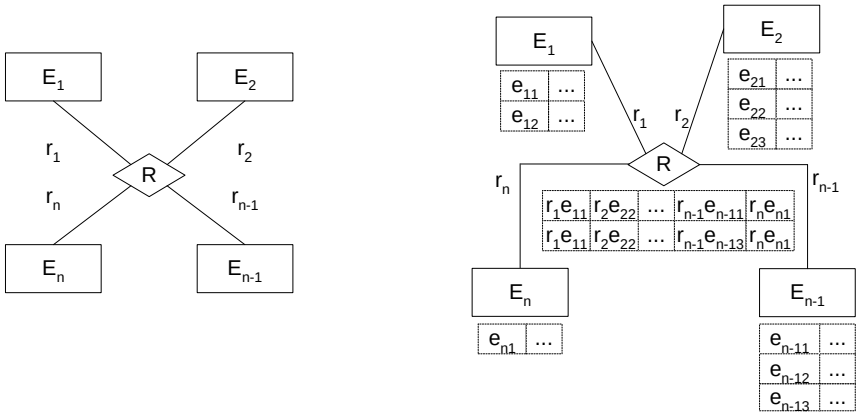


Figura 5.2: a) Representación de una interrelación de grado n sin atributos. b) Ejemplo de instancias que participan en R.

En la figura 5.2(b) se presenta un ejemplo de dos instancias pertenecientes a R. Al ir acompañada cada instancia clave con el role podemos distinguir cuando una misma entidad participa más de una vez dentro de una interrelación y además las componentes del vector no tienen necesariamente que estar ordenadas.

Antes de pasar a definir las restricciones de cardinalidad asociadas a una interrelación debemos aclarar que a partir de ahora cuando una entidad participe solo una vez dentro de una interrelación simplificaremos la notación y prescindiremos de su role. Con esta simplificación lo que buscamos es que la terminología sea lo mas clara posible pero con ello no queremos quitarle importancia a este concepto. No solo distingue la participación de cada entidad

dentro de una interrelación sino que sirve para documentar un esquema conceptual en lenguaje natural de forma automática e incluso el proceso contrario, a través de una especificación en lenguaje natural si se detectan los roles se pueden construir las interrelaciones asociadas y validarlas con el usuario.

5.2. Restricciones de cardinalidad

La primera definición de restricción de cardinalidad que enunciaremos se basa en la observación de la participación de las instancias de una entidad dentro de una interrelación. Esta participación será obligatoria, cuando todas las instancias de una entidad se asocian dentro de una interrelación, u opcional, cuando existen instancias de la entidad que no están asociadas a través de la interrelación.

Cardinalidad de participación de una entidad: Definimos la restricción de cardinalidad de participación de una entidad $C(r_iE_j, R) = 0$ o 1 como la participación opcional u obligatoria, respectivamente, de las instancias clave de E_j con el role r_i en la interrelación $R = (r_1E_1,\ldots, r_nE_j)$, interrelación de grado n de grado n.

$C(r_iE_j, R) = 1$ con $1 \leq j \leq i \leq n$ si y solo si $\forall\ a \in \#E_j$, $\exists\ b \in R^t$ tal que $b_i = a$ donde b_i denota la componente i-ésima de b, vector de n componentes. En caso contrario la participación sería opcional.

La representación de esta restricción es a través de un círculo negro para reflejar la obligatoriedad y un círculo en blanco para la opcionalidad en la participación de una entidad dentro de una interrelación. En la figura 5.3 mostramos un ejemplo de una interrelación binaria entre *Escritor* y *Libro* donde se refleja que toda escritor participa en la interrelación *escribe* al contrario que libro, el cual puede tener instancias que no se asocien con ningún escritor.

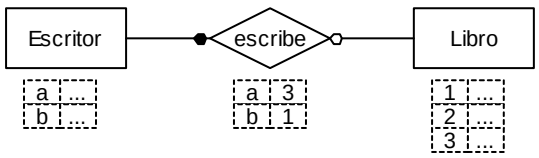


Figura 5.3: Representación de la restricción de participación de una entidad

Esta restricción impone un comportamiento o conducta dentro de nuestra base de datos. Una operación de actualización sobre Libro se realizará sin tener en cuenta la relación que le asocia con Escritor, sin embargo, una inserción, por ejemplo, en Escritor va a desencadenar una inserción en la interrelación *escribe*. Destacamos que la cardinalidad de participación de una entidad recoge semántica estática y dinámica o de comportamiento. Y que esta restricción no es de tipo estructural sino poblacional.

Cardinalidad de Merise para una interrelación: Definimos la restricción de cardinalidad de Merise para un determinado role r_i en $R = (r_1E_1,\ldots, r_nE_j)$ como $CMerise(r_iE_j,R) = (n,m)$ donde $n, m \in N$ y $1 \leq n \leq m$ y refleja que dada una instancia clave *a* de r_iE_j de R^t, aparece en R^t como mínimo n veces y como máximo m.

$CMerise(r_iE_j,R) = (n,m)$ si y solo si $n \leq |\{b \in R^t / b_i = a\}| \leq m \quad \forall a \in r_iE_j$.

Donde |M| es el número de elementos de un conjunto, donde b_i denota la componente i-ésima de b, vector de n componentes y r_iE_j la componente i-ésima de R^t.

La representación de la cardinalidad de Merise se refleja con una etiqueta en el extremo de la línea que une la entidad con el rombo de la interrelación (figura 5.4), en el lado del rombo.

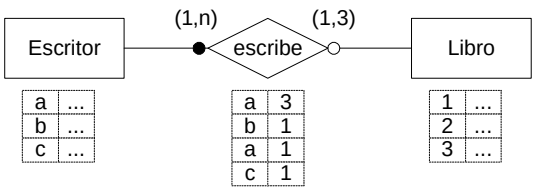


Figura 5.4: Representación de la restricción de cardinalidad de Merise

En el ejemplo presentado en la figura 5.4 se refleja que un libro, del que se conoce sus autores, está escrito por 1 y como mucho por tres escritores. La semántica que recogemos con estas dos cardinalidades tiene un matiz que la distingue de las restricciones de cardinalidad estudiadas anteriormente. Supongamos que queremos reflejar el siguiente supuesto, que los empleados que participan en proyectos, dentro de una empresa, lo hacen en al menos dos. Según las definiciones estudiadas en la sección 2.3.2 con la aproximación *look here* o *look across* representaríamos esta restricción como se muestra en la figura 5.5.

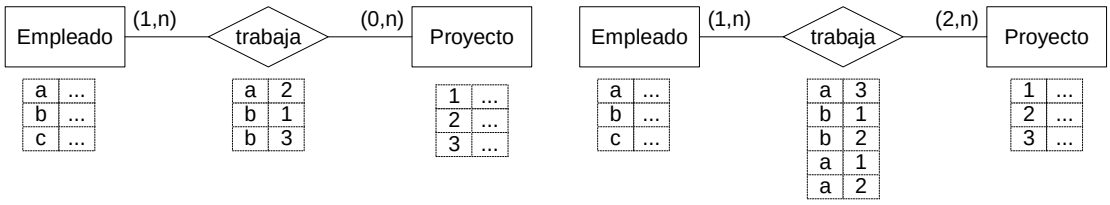


Figura 5.5: Restricciones cardinalidad (a) Con la aproximación look here (b) con la aproximación look across.

En la primera solución, figura 5.5(a) solo podemos expresar la opcionalidad de participación de Empleado. Por lo que puede ocurrir que existan instancias en *trabaja*, el empleado *a* solo se relaciona con un proyecto, que no cumplan la regla establecida en la empresa. La segunda solución, figura 5.5(b), impone que todas las instancias de empleado se asocien con dos proyectos como mínimo. Por lo que la ventaja de nuestra propuesta recae en que se precisa el número de veces que aparece una instancia dentro de una interrelación y por otro lado si toda instancia

debe participar en la interrelación. La solución a la especificación planteada se muestra en la figura 5.6.

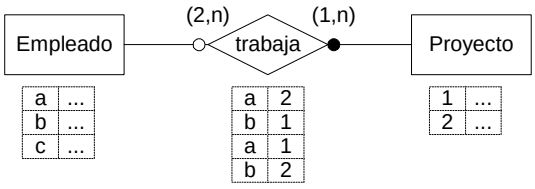


Figura 5.6: Representación de las restricciones de cardinalidad propuestas.

La restricción de cardinalidad de Merise junto con la cardinalidad de entidad (figura 5.6) reflejan que no todo empleado trabaja en los proyectos de una empresa pero aquellos que lo hacen están involucrados en al menos dos. La regla de negocio o especificación queda plenamente representada con la aplicación de ambas restricciones.

Como veremos posteriormente en la transformación de esquemas, la cardinalidad máxima de Merise nos impondrá restricciones estructurales y la mínima limitará las combinaciones en la población de la interrelación.

La última restricción de cardinalidad que definimos en nuestra propuesta es una ampliación de la presentada por Chen (1976) pero con un nuevo enfoque para evitar los problemas de trabajar con información desconocida o incompleta (Cuadra et al, 2000).

Cardinalidad de Chen para una interrelación: Definimos la restricción de cardinalidad de Chen para un determinado role r_i en $R = (r_1E_1,\ldots, r_nE_j)$ como $CChen(r_iE_j,R) = (n,m)$ donde $n, m \in N$ y $1 \leq n \leq m$ y refleja que dada una combinación de instancias clave $a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n$ de R^t, esta aparece en R^t, n veces como mínimo y m máximo.

$CChen(r_iE_j,R) = (n,m)$ si y solo si $n \leq |\{b \in R^t / (b_1,\ldots, b_{i-1}, b_{i+1},\ldots,b_n) = (a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n)\}| \leq m \quad \forall\, a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n \in r_1E_1,\ldots,r_{i-1}E_s\ldots, r_{i+1}E_t, \ldots r_nE_j$

Donde $|M|$ es el número de elementos de un conjunto, donde cada b_i denota la componente i-ésima de b, vector de n componentes y r_iE_j la componente i-ésima de R^t.

La representación de la cardinalidad de Chen se refleja con una etiqueta en el extremo de la línea que une la entidad con el rombo de la interrelación (figura 5.7), en el lado de la entidad.

Veamos a través de un ejemplo la diferencia que existe entre la restricción de Merise y de Chen en interrelaciones binarias. Supongamos que tenemos la siguiente especificación: un empleado puede estar asignado a un único departamento y cuando se crea un departamento tendrá al menos dos empleados.

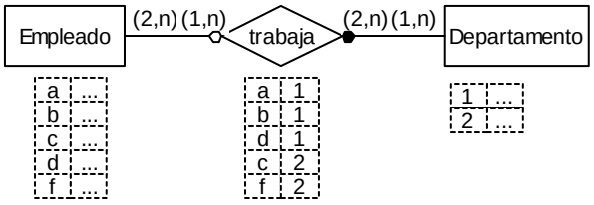


Figura 5.7: Cardinalidad de entidad, de Merise y de Chen.

Como puede observase en la figura 5.7, la restricción de Chen y Merise se diferencian en la representación pero expresan la misma semántica por lo que podemos enunciar el siguiente resultado.

Teorema 1: Si R = (rA, sB) es una interrelación binaria donde A, B son entidades no necesariamente distintas se cumple que:

$$CMerise\ (rA, R) = CChen\ (sB, R) \text{ y } CMerise\ (sB, R) = CChen\ (rA, R)$$

-Demostración-

Supongamos que CMerise (rA, R) = n y CChen (sB, R) = m donde n≠m.

Que CMerise (rA, R) = n significa que toda instancia de rA aparece n veces.

Que CChen (sB, R) = m implica que si fijamos una instancia de rA esta aparece relacionada con una instancia de sB m veces.

Por tanto n = m. En donde n puede ser la cardinalidad mínima o máxima.

Este resultado nos muestra que la distinción entre estos dos tipos de restricción en interrelaciones binarias no es necesaria. Sin embargo, en interrelaciones de grado superior las cosas cambian. Como justificamos en el apartado 2.3.2.1 es necesario representar las cardinalidades a través de la aproximación de Merise y de Chen, ya que aportan al esquema semántica distinta.

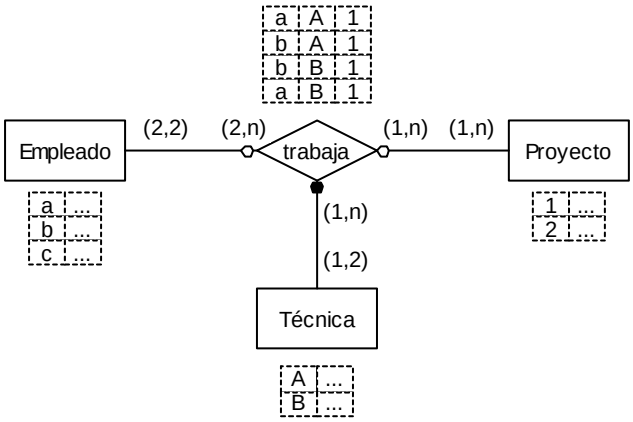


Figura 5.8: Representación de todas la restricciones de cardinalidad

En la figura 5.8 se representan todas las restricciones de cardinalidad propuestas. La semántica que recogemos es, a través de la cardinalidad de entidad, la opcionalidad u obligatoriedad de la participación de las instancias de cada una de las entidades que participa en la interrelación *trabaja*. Y con la conjunción de las cardinalidades de Merise y de Chen no se permitiría que en un proyecto utilicen la misma técnica mas de dos empleados, ni que un empleado que trabaje en un

proyecto emplee más de dos técnicas en el mismo, y un empleado aparecerá en la interrelación trabaja mas de una vez.

Al igual que la cardinalidad de Merise, la cardinalidad máxima de Chen nos impondrá restricciones estructurales y la mínima de combinación o de población dentro de la interrelación.

En la siguiente sección estudiaremos detalladamente qué cardinalidades son consistentes y si con ellas la semántica de una interrelación de grado superior está totalmente definida. Antes de este estudio queremos clasificar y comparar estas restricciones con las restricciones de cardinalidad propuestas por otros investigadores y/o adoptadas por los modelos conceptuales. Recordemos que en la sección 2.3.2 presentábamos las siguientes aproximaciones, dividiéndolas en el tratamiento de interrelaciones binarias y de grado superior.

Recordamos que cuando tratamos con interrelaciones binarias existen dos posibilidades: la interpretación a través de la entidad, que hemos denominado *look here*, o la interpretación a través de la interrelación, *look across.*

En nuestra propuesta combinamos ambas interpretaciones al considerar que la cardinalidad mínima que cada una representa recoge especificaciones distintas y esta diferencia es necesario recogerla. En la primera interpretación con una cardinalidad mínima de cero, se considera que la participación de una entidad es opcional dentro de la interrelación, pero si una instancia de entidad participa en la interrelación no cuenta el número mínimo de veces que lo hace. Por lo que en nuestra propuesta tomamos ambas cardinalidades mínimas, la cardinalidad de entidad y la cardinalidad mínima de Merise.

Esta doble semántica, proporcionada por mirar a través de la entidad o de la interrelación, recogen situaciones distintas, lo que hace que con ellas podamos expresar de una forma más *fina* determinadas características del UD para poder plasmarlas dentro de un modelo conceptual.

Cuando tratamos de interrelaciones de grado superior además de estas interpretaciones tenemos dos aproximaciones la de Merise y de Chen que de forma resumida lo que hacen es hallar las restricciones de cardinalidad de un role o entidad con respecto a los restantes roles o entidades que participan en la interrelación, o viceversa. Por lo tanto, tenemos 4 posibilidades y cada una de ella recoge información distinta. Como nos pasaba con las interrelaciones binarias existen diferencias entre interpretaciones cuando tratamos cardinalidades mínimas. Pero además, dependiendo de la aproximación elegida también influye en las cardinalidades máximas (tabla 5.1).

Como vemos en la tabla 5.1 la cardinalidad máxima cuando toma el valor de uno, recoge, para Merise y para Chen, dos tipos de dependencias muy importantes porque imponen restricciones estructurales. Con opcionalidad indicamos que la participación de instancias de E_i no es obligatoria en la interrelación sin embargo la opcionalidad (n-1) indicamos que cualquier combinación de instancias de entidades menos la de E_i pueden o no participar en la interrelación. Con la i-obligatoriedad representamos que las instancias de E_i participan en la interrelación i-veces y con la i-obligatoriedad (n-1) las que participan i-veces son la combinación de instancias del resto de entidades. Y por último, el concepto de i-participación indica cuántas veces aparece una instancia clave asociada a E_i y con la i-participación (n-1) cuantas veces lo hacen la combinación de instancias claves del resto de entidades.

Cardinalidad E_i		Merise		Chen	
		LH	**LA**	**LH**	**LA**
Mínima	0	opcionalidad	NO	opcionalidad (n-1)	NO
	i > 0	i-obligatoriedad	i-participación	i-obligatoriedad (n-1)	i-participación (n-1)
Máxima	1	E_i--> E_1,..Ei_{-1},E_{i+1},.., E_n		E_1,..Ei_{-1},E_{i+1},.., E_n -->E_i	
	i > 1	i-obligatoriedad o participación		i-obligatoriedad o participación (n-1)	

Tabla 5.1: Resumen de todas las restricciones de cardinalidad estudiadas

En resumen, el concepto de obligatoriedad a diferencia de la participación es que la primera de ellas toma instancias de la entidad o combinación de instancias potenciales sin embargo la segunda se asegura de que estas instancias estén en la interrelación.

¿Nuestra propuesta que restricciones adopta? La respuesta está en elegir las restricciones que más información y más completa recojan.

La restricción de cardinalidad se corresponde con la mínima de la restricción de cardinalidad con la interpretación de *look here* y la aproximación de Merise, con los valores 0 o 1 para indicar la opcionalidad o la obligatoriedad.

La restricción de Merise definida en nuestra propuesta recoge una interpretación *look across* y una aproximación de Merise, y la restricción de Chen también responde a la misma interpretación pero con la aproximación de Chen.

5.2.1. Estudio de las interrelaciones ternarias

Según la propuesta presentada tenemos que para una interrelación n-aria debemos expresar al menos tres tipos de restricción de cardinalidad. Aunque hemos definido formalmente cada una de estas cardinalidades basándonos en una definición formal de interrelación solo hemos logrado un paso: que la ambigüedad en la definición de interrelación y sus propiedades se disipen.

Pero si recordamos los problemas que encontrados en el manejo de interrelaciones de grado superior sigue siendo difícil de aplicar por los diseñadores poco expertos y un problema para interpretarlo en estructuras relacionales. Este problema es debido a la dificultad de identificar interrelaciones de grado superior legítimas en situaciones prácticas y la carencia de entendimiento del constructor dentro de la

teoría básica sobre la que se apoya el modelo relacional. Por supuesto, todavía no hemos visto como incide estas nociones en el estudio y soporte para implementar una interrelación n-aria en el modelo relacional. Enfocando también esta carencia a las herramientas CASE.

En esta sección nos centraremos en el estudio de las interrelaciones ternarias, estudiando la consistencia de la restricciones de cardinalidad definidas en nuestra propuesta y su validación, dando un método para el cálculo de las cardinalidades aplicando los resultados obtenidos por McAllister (1998), Dullea &Song (1998), Jones & Song (2000) y Camps (2002).

Antes de estudiar las restricciones de cardinalidad válidas debemos definir que se entiende por un restricción de cardinalidad.

Definición general de validación estructural: Un esquema conceptual es estructuralmente válido si se consideran de forma simultanea todas las restricciones estructurales impuestas en el modelo y no implican una inconsistencia lógica para cualquier estado posible.

Definición general de validación semántica: Un esquema conceptual es semánticamente válido cuando toda y cada una de las interrelaciones representan exactamente el concepto a modelar del problema del dominio.

En la validación semántica, el diagrama, compuesto por entidades y las restricciones asociadas, debe comunicar exactamente el concepto deseado. En la validación estructural existen dos factores que imponen restricciones y que son los que se deben estudiar: las restricciones de cardinalidad máxima y mínima.

En esta sección nos limitaremos a estudiar la validación estructural de las restricciones de cardinalidad asociadas a una interrelación ternaria ya que la validación semántica debe llevarse a cabo a través de entrevistas con los expertos del dominio que lo validen. Esta validación es mucho más compleja que la

validación estructural al no poder formalizarla y depender de un dialogo con el usuario experto del dominio.

¿Cuándo un conjunto de restricciones (RC) de cardinalidad asociadas a una interrelación son estructuralmente válidas? La respuesta es sencilla, si encontramos al menos una instancia que cumpla las restricciones impuestas. Por tanto, si existe alguna restricción de cardinalidad que no se cumple, el RC no será válido estructuralmente.

Definición de restricción de cardinalidad no válida: Sea $R = (r_1E_1,..,r_nE_j)$ una interrelación de grado n.

1. La restricción de cardinalidad de entidad $C(E_i, R) = 1$ es no válida si y solo si $\exists\ a \in \#E_i$ tal que $\forall\ b \in R^t\ b_i \neq a$.

2. La restricción de cardinalidad de Merise, $CMerise\ (r_iE_i, R) = (n,m)$ es no válida si y solo si:

   (a) $n>1$ y $\exists\ a \in r_iE_i$ tal que $|\{b \in R^t / b_i = a\}| < n$ ó

   (b) $m \in N$ y $\exists\ a \in r_iE_i$ tal que $|\{b \in R^t / b_i = a\}| > m$

3. La restricción de cardinalidad de Chen, $CChen\ (r_iE_i, R) = (n,m)$ es no válida si y solo si:

   (a) $n>1$ y $\exists\ a \in a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n \in r_1E_1,\ldots,r_{i-1}E_s\ldots, r_{i+1}E_t, \ldots r_nE_j$ tal que $|\{b \in R^t / (b_1,\ldots, b_{i-1}, b_{i+1},\ldots,b_n) = (a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n)\}| < n$ ó

   (b) $m \in N$ y $\exists\ a \in \exists\ a \in a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n \in r_1E_1,\ldots,r_{i-1}E_s\ldots, r_{i+1}E_t, \ldots r_nE_j$ tal que $|\{b \in R^t / (b_1,\ldots, b_{i-1}, b_{i+1},\ldots,b_n) = (a_1,\ldots, a_{i-1}, a_{i+1},\ldots,a_n)\}| > m$

Reglas de validación para interrelaciones ternarias

En primer lugar debemos aclarar que este estudio se basa en la presentación de las restricciones de cardinalidad que puedan existir entre cualquier subconjunto de componentes de una interrelación de grado n. Esto tiene un gran problema de complejidad ya que el número de restricciones que podemos definir dentro de una interrelación de grado n es de $3^n - 2^{n+1}+1$ (McAllister, 1998). Esto implica que si tenemos una interrelación de grado 3 existen 12 restricciones que podemos definir, si la interrelación es de grado 4, el número es 50 y si fuera de grado 5, serían 180. Esto es inmanejable para un diseñador aunque si la sería si se automatizara a través de una herramienta CASE.

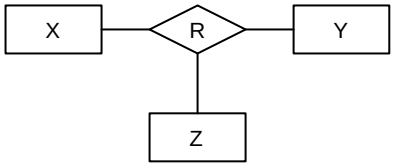
Pensando en la facilidad para el diseñador en nuestra propuesta nos centramos únicamente en las restricciones de cardinalidad más utilizadas por lo modelos conceptuales, pero sabemos que la información no es totalmente completa.

Las reglas presentadas a continuación están basada en McAllister (1998) y se derivan de los axiomas de Armstrong. Veremos la aplicación de estas reglas en las interrelaciones de grado tres, por ser de las interrelaciones de grado superior, la que más aparece en los esquemas conceptuales y por la sencillez para su exposición.

También es importante la definición de estas reglas no solo para ayudar al diseñador a validar las restricciones asociadas a una interrelación sino que también nos pueden ayudar a realizar mapping entre restricciones y por lo tanto incrementar las posibilidades de reutilización.

Sea R = (X,Y,Z) una interrelación ternaria, tal como define Jones & Song (2000) existen también restricciones binarias entre las entidades participantes en una interrelación binaria que las denomina S*emantically Constraining Binary* (SCB) y que deberemos tener en cuenta para completar la semántica de una interrelación

binaria. Para denotar tanto las restricciones de la interrelación ternaria estudiadas como a las binarias asociadas utilizaremos la siguiente notación:



Denotaremos a Cmin(X,Y) y Cmax(X,Y) cardinalidad mínima y máxima, respectivamente, de SCB entre las componentes X, Y de R, Cmin(XY, Z) es la mínima de CChen (Z, R), Cmax(XY, Z) es la máxima. De la misma forma podemos definir Cmin(X, YZ) y Cmax(X, YZ) como la cardinalidad mínima y máxima de CMerise (X, R).

Esta cambio de notación se justifica para simplificación de las reglas y para mostrar la semejanza con los axiomas se Armstrong.

Regla 1: C (X, Y) ≥ C (XZ, Y)

-Demostración-

Lo demostraremos para la mínima pero sin pérdida de generalización

Supongamos que n = Cmin (X, Y) < Cmin (XZ, Y) = m.

Que Cmin (XZ, Y) = m significa que para cada xz (combinación de instancias clave de X y Z) aparece en R^t al menos m veces.

Esto implica que las combinaciones xy al menos son m. !!! con m>n.

Regla 2: C (X, YZ) ≥ C (X,Y)

-Demostración-

Supongamos n = Cmin (X, YZ) < Cmin (X,Y) = m

Que Cmin (X, Y) = m implica que la combinación de un x en R^t aparece con m y's distintos.

Por lo tanto, al menos aparecerá este x en R^t m veces, esto hace que Cmin (X, YZ) > m !

Regla 3: Cmin (X, YZ) ≥ Cmin (X,Y) x Cmin (XY, Z)

-Demostración-

Supongamos Cmin (X, YZ) < Cmin (X,Y) x Cmin (XY, Z)

Esto implica que Cmin (X,Y)> 0 y Cmin (XY, Z)>0.

Cmin(X,Y) = n1 implica que en R^t aparecen una misma instancia clave x con *n1* y's distintos, es decir, es válida.

Asumimos que estas mismas combinaciones xy aparece en R^t exactamente *n2*. Por tanto, el resultado del numero de instancias en R^t será n1xn2.

Si decrementamos este valor no será válida Cmin(X,YZ)

Regla 4: Cmax (X, Y) ≥ Cmin (X,Y)

-Demostración trivial-

Regla 5: Cmin (X, Y) x Cmax (Y,Z) ≥ Cmin (X, YZ)

-Demostración-

Supongamos Cmin (X, YZ) > Cmin (X,Y) x Cmax (Y, Z)

Cmin(X,Y) = n1 implica que en R^t aparecen una misma instancia clave x con *n1* y's distintos, es decir, es válida.

Asumimos que de cada instancia xy anterior, construimos tantas xyz como Cmax (Y,Z). Por lo que el número de instancias en R^t es Cmin (X,Y) x Cmax (Y, Z).

Tal y como hemos supuesto Cmin (X,Y) x Cmax (Y, Z) < Cmin (X, YZ), esto significa que si Cmin(X,Y) es válido, Cmin (X, YZ) no puede ser válido !!!!

Si decrementamos este valor no será válida Cmin(X,YZ)

Regla 6: Cmin (X,Y) x Cmax (X, Z) ≥ Cmin (X, YZ)

Regla 7: Cmax (X, YZ) ≥ Cmax (XY, Z) + Cmin (X, Y) –1

Según la definición de nuestras restricciones debemos añadir las siguientes reglas:

Regla 8: Si Cmin(XY,Z) > 1 entonces Cmin (X,Z) > 1 y Cmin (Y,Z) >1

5.3. Manejando información desconocida

Las restricciones que hemos considerado en nuestra propuesta son las que más contenido semántico aportan de una interrelación. Pero todavía no hemos entrado a manejar información inaplicable o desconocida dentro de una interrelación. Supongamos que tenemos las siguientes especificaciones, los empleados de la empresa que trabajan en proyectos pueden hacerlo usando una técnica o no. Pero todo empleado que participe en un proyecto con una técnica, lo hará solo con una. Y en un proyecto la misma técnica será utilizada por dos empleados.

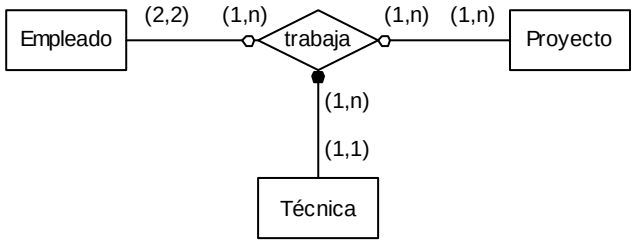


Figura 5.9: Representación de las cardinalidades de acuerdo a las especificaciones

¿Alguna de las cardinalidades representadas refleja que pueden existir empleados que trabajen en proyectos sin utilizar una técnica? Podríamos pensar que si en vez de poner un 1 en la mínima de CChen (Técnica) lo sustituimos por un 0 esto recogería la restricción pedida. Pero esto implicaría que según la figura 5.10, en la interrelación aparecerían instancias con alguna de sus componentes vacías o nulas.

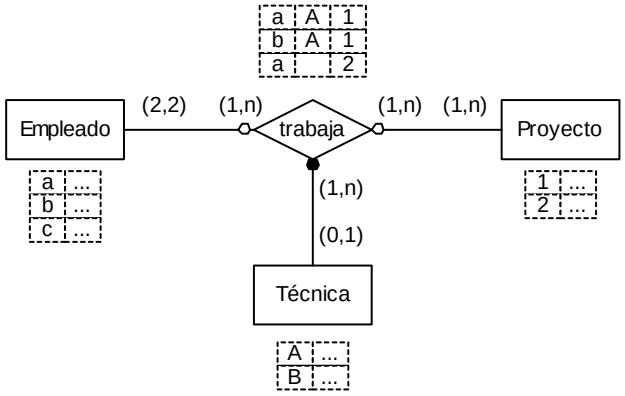


Figura 5.10: Representación de información inaplicable en trabaja

Según la definición dada y adoptada en nuestra propuesta esta representación no es posible al entrar en conflicto con la definición de interrelación. Para completar, por tanto, la definición de interrelación sin tener que manejar información desconocida introducimos el concepto de interrelación complementaria.

Definición de interrelación complementaria: Sea $R = (r_1E_1,\ldots, r_nE_j)$ una interrelación de grado n. Definimos una interrelación complementaria de R como R^C_a, donde a indica los roles sobre los que participa esta interrelación y debe cumplir:

1. a < n, es decir, el número de roles sobre las que se aplica ha de ser menor que el grado de la interrelación.

2. C(aE, R) = 0, la restricción de cardinalidad para cada una de las entidades que participan con uno de estos roles en R ha de ser opcional.

3. $R_a^{Ct} \not\subset R^t$, es decir, las instancias pertenecientes a la interrelación complementaria no pueden ser un subconjunto de la interrelación a la que complementa.

Su representación es igual que la de una interrelación pero las líneas que la unen a las entidades que participan en ella son punteadas en vez de continuas. En la figura 5.11 presentamos cómo sería la representación del ejemplo anterior introduciendo esta nueva interrelación. Como se puede observar el empleado *a* participa en el proyecto *2* pero no utiliza ninguna técnica y por lo tanto aparece como una instancia de la interrelación complementaria a trabaja. Esta instancia no puede aparecer como elemento de trabaja por que llegaríamos a una inconsistencia semántica. Si el empleado *a* trabaja en el proyecto *2* con una técnica aparecerá como elemento de *trabaja* sino utiliza ninguna técnica entonces pertenecerá a $trabaja^{C}_{EP}$.

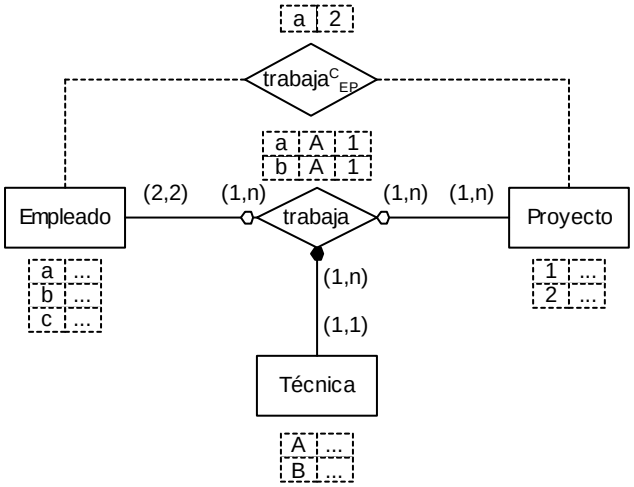


Figura 5.11: Representación de una interrelación complementaria

5.4. Aplicando una metodología

Una vez definidas la restricciones de cardinalidad que tendremos en cuenta al representar una interrelación de grado n debemos pasar al siguiente nivel de la metodología de desarrollo de bases de datos.

Este nivel se corresponde con la fase de modelado lógico que abordaremos con el modelo relacional. En esta fase se parte de un esquema conceptual válido estructuralmente como semánticamente. Lo que significa que en el esquema conceptual queda totalmente plasmada el UD.

Como hemos analizado en la sección 2.4.3 la transformación de esquemas conceptuales a esquemas relacionales se rige por una serie de reglas muy elementales que en la mayoría de los casos no reflejan toda la semántica recogida en el esquema conceptual.

No discutiremos de nuevo cuales son las pérdidas concretas de semántica ya que fueron estudiadas anteriormente, en esta sección nos centraremos en la ampliación de las reglas de transformación incluyendo las restricciones de cardinalidad definidas en la sección anterior.

La sección la dividimos en dos partes, al igual que en la presentación de las restricciones de cardinalidad, ya que existen diferencias significativas al transformar las interrelaciones binarias y las de grado superior.

5.4.1. Transformación de interrelaciones binarias

Las restricciones asociadas a las interrelaciones binarias son: la cardinalidad asociada a la entidad y la cardinalidad asociada a la interrelación con la

aproximación de Merise o de Chen ya que como demostramos anteriormente son iguales.

La semántica recogida por la cardinalidad asociada a la entidad determina la opcionalidad u obligatoriedad de la participación de las instancias de una entidad en una interrelación. Sin embargo la cardinalidad de Merise cuenta, por cada componente de la interrelación, cuantas veces aparece. Veamos como afectan en la transformación de una interrelación binaria.

Las reglas de transformación presentan tres opciones para convertir una interrelación binaria (IB) en una estructura relacional (figura 5.12):

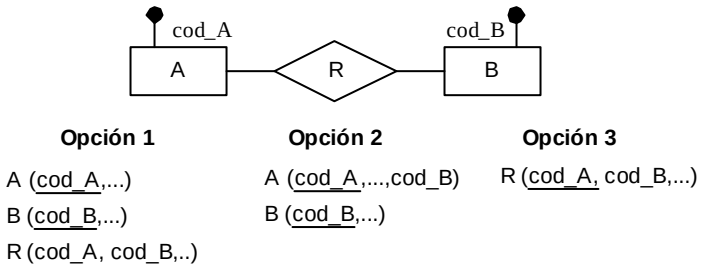


Figura 5.12: Distintas opciones para la transformación de IB

En general, la primera opción es utilizada cuando el tipo de correspondencia es N:M, la segunda cuando la correspondencia es 1:N y la tercera, que es la menos utilizada, en algunos casos de correspondencia 1:1. Como podemos observar estos tres tipos de transformación solo contemplan la correspondencia de la IB, a continuación presentaremos las restricciones que deberemos añadir a cada una de las opciones para completar la semántica de las restricciones.

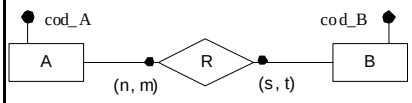
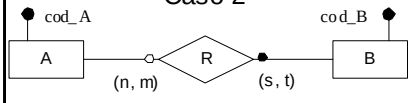
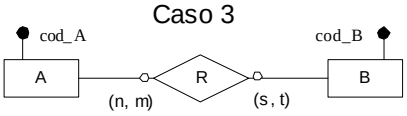
N:M	Opción	Restricción
Caso 1 	1	CC en R: (cod_A, cod_B) FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_A_R Obligatoriedad_B_R Participación_rA_R Participación_rB_R
Caso 2 	1	CC en R: (cod_A, cod_B) FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_B_R Participación_rA_R Participación_rB_R
Caso 3 	1	CC en R: (cod_A, cod_B) FK_A en R: cod_A FK_B en R: cod_B Participación_rA_R Participación_rB_R

Tabla 5.2: Transformación de IB con correspondencia N:M

Según indicamos en la sección 2.4.3.2, denotamos CC al conjunto de claves candidatas asociadas a una relación, con PK_R a la clave primaria de la relación R, con FK_R como la clave ajena de una relación que referencia a R y añadimos dos mecanismos de control, el de Obligatoriedad y el de Participación, que son los que forman parte de nuestra propuesta. Veamos en que consisten estos mecanismos.

El control de la obligatoriedad se implementará a través de una aserción.

```
CREATE ASSERTION Obligatoriedad_E_R
CHECK (NOT EXITS (SELECT cod_E FROM E WHERE
        NOT IN(SELECT cod_E FROM R)));
```

Donde E es la relación que proviene de una entidad y R es la relación donde cualquier tupla de E debe aparecer al menos una vez. El lenguaje utilizado para este mecanismo de control es el SQL3 y el elemento que hemos utilizado funciona

de la siguiente manera; cada operación de actualización que se realice sobre las tablas E o R comprobará si la condición impuesta por el Check se no cumple abortará la operación. De esta forma controlamos las actualizaciones en las tablas asegurándonos la validación de la restricción de cardinalidad C(E, R) =1.

El control de la participación de un role se basará en contar en la relación que proviene de la interrelación si aparecen el número mínimo y máximo de instancias que indica la restricción de cardinalidad CMerise (rE, R). En este control distinguiremos que la restricción de cardinalidad máxima sea distinta o no de N, valor que no impone ninguna prohibición en R.

Si CMerise (rE1, R) = (i,j) añadimos a la relación R, proveniente de la interrelación cuya correspondencia es N:M la siguiente sentencia de verificación, dependiendo:

- ✓ Si (1<i<j) y (j≠N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_R
CHECK ((i ≤ SELECT count(cod_e1) FROM R GROUP BY
cod_e2) AND (j ≥ SELECT count(cod_e1) FROM R GROUP
BY cod_e2))
```

- ✓ Si (1<i<j) y (j=N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_R
CHECK ((i ≤ SELECT count(cod_e1) FROM R GROUP BY
cod_e2))
```

- ✓ Si (1=i<j) y (j≠N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_R
CHECK ((j ≥ SELECT count(cod_e1) FROM R GROUP BY
cod_e2))
```

- ✓ Si (1≠i =j) y (j≠N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_R
CHECK ((i = SELECT count(cod_e1) FROM R GROUP BY
cod_e2))
```

Con estas sentencias de verificación añadidas a R se evitará la violación de la restricción CMerise (rE1, R) para cualquier operación de actualización de la base de datos.

Esta transformación en comparación con la propuesta por Fahrner & Vossen (1995), una de las mas completas estudiadas en la sección 2.4.3.2, que consistía en que toda transformación N:M podía descomponerse de una forma única independientemente de las cardinalidades, se ha divido en tres casos generales donde las restricciones de cardinalidad juegan un papel importante dentro de esta transformación.

El estudio de la transformación de la IB cuando la correspondencia es 1:N se presenta en la tabla 5.3, en ella se analizan las distintas posibilidades según las restricciones de cardinalidad asociadas. En comparación con Fahrner & Vossen (1995), en la que se presentaban exclusivamente dos casos al considerar irrelevantes las cardinalidades mínimas asociadas a la entidad que participa con la máxima de N, en la tabla 5.3 vemos que se han extendido a dos casos más.

En esta transformación podemos utilizar las opciones una y dos presentadas en la figura 5.12. Observamos en la tabla 5.3 que el orden que se ha seguido a la hora de priorizar opciones es la de poner primeramente aquella que evite los valores nulos en las relaciones. Dependiendo del caso y del SGBD donde se implemente la BD, el diseñador deberá tomar una decisión dependiendo de si el número de tuplas que poseen nulos no perjudican en la consistencia de la BD.

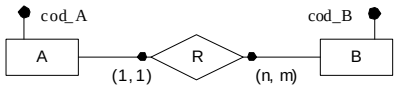
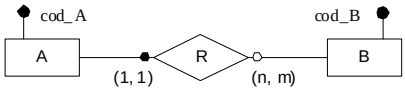
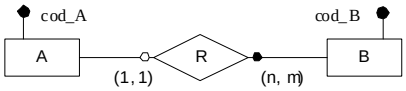
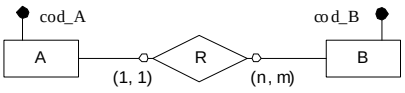
1:N	Opción	Restricción
Caso 4 	2	FK_B en A: cod_B NO en A: cod_B Obligatoriedad_B_A Participación_rB_A
	1	PK en R: cod_A NO en R: cod_B FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_A_R Obligatoriedad_B_R Participación_rB_R
Caso 5 	2	FK_B en A: cod_B NO en A: cod_B Participación_rB_A
	1	PK en R: cod_A NO en R: cod_B FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_A_R Participación_rB_R
Caso 6 	1	PK en R: cod_A NO en R: cod_B FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_B_R Participación_rB_R
	2	FK_B en A: cod_B O en A: cod_B Obligatoriedad_B_A Participación_rB_A
Caso 7 	1	PK en R: cod_A NO en R: cod_B FK_A en R: cod_A FK_B en R: cod_B Participación_rB_R
	2	FK_B en A: cod_B O en A: cod_B Participación_rB_A

Tabla 5.3: Transformación de IB cuya correspondencia es 1:N

En el caso de la transformación de IB con correspondencia 1:1, tabla 5.4, los casos siguen siendo los mismos que los estudiados por Fahrner & Vossen (1995) pero se le añaden mecanismos de control para asegurar el cumplimiento de las

restricciones de cardinalidad. Aunque no aparece como una opción debemos mencionar que para el caso 8 existe otra posibilidad propuesta por De Miguel & Piattini (1993) que consiste en propagar la clave de A a B y de B a A, no permitiendo valores nulos para las claves ajenas.

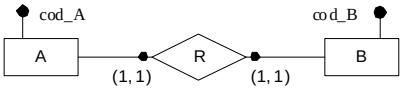
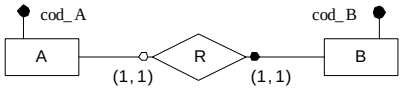
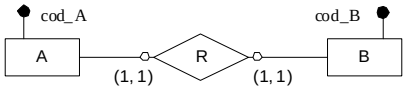
1:1	Opción	Restricción
Caso 8  	2	FK_B en A: cod_B NO en A: cod_B Obligatoriedad_B_A **o** FK_A en B: cod_A NO en A: cod_B Obligatoriedad_A_B
	1	CC en R: cod_A, cod_B NO en R: cod_A, cod_B FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_A_R Obligatoriedad_B_R
	3	PK en R: Cod_A NO en R: cod_B
Caso 9  	2	FK_A en B: cod_A NO en B: cod_A
	1	CC en R: cod_A, cod_B NO en R: cod_A, cod_B FK_A en R: cod_A FK_B en R: cod_B Obligatoriedad_B_R
Caso 10  	1	CC en R: cod_A o cod_B NO en R: cod_A, cod_B FK_A en R: cod_A FK_B en R: cod_B
	2	FK_B en A: cod_B O en A: cod_B o FK_A en B: cod_A O en B: cod_A

Tabla 5.4: Transformación de IB con correspondencia 1:1

Al igual que en los demás casos en la propuesta presentamos las distintas posibilidades de transformación, según las características del UD, las operaciones

que se realicen en la BD, el volumen de datos que se vayan a manejar y si se prima la eficacia en vez de la eficacia o viceversa, el diseñador deberá elegir aquella opción que mas ventajosa le sea.

5.4.2. Transformación de interrelaciones ternarias

La transformación de interrelaciones de grado superior, concretamente, las IT se transforman en general, y la mayoría de las propuestas así lo indican Elmasri & Navathe (1994), Hansen & Hansen (1995), Fahrner & Vossen (1995), Ramakrishnan, R. (1997), etc, en una relación compuesta por los identificadores principales de las entidades que asocian junto con los atributos que posea la interrelación. En nuestro estudio no tomaremos en cuenta estos atributos pero si queremos dejar constancia que la igual que en la transformación de IB, la clave primaria de la relación que proviene de la interrelación puede verse modificada por la inclusión de alguno de estos atributos. En la figura 5.13 presentamos la única opción para realizar la transformación de una IT a partir de la cual añadiremos las restricciones necesarias para forzar el cumplimiento de las cardinalidades asociadas a ella.

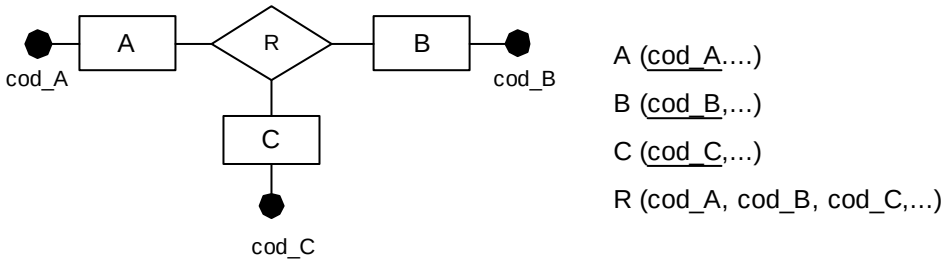


Figura 5.13: Transformación de un IT

Para la transformación de interrelaciones ternarias debemos añadir otro procedimiento de control de cardinalidad correspondiente CChen que en las IB no aparece porque es igual a CMerise.

Si CChen (rE1, R) = (i,j), hallamos la cardinalidad fijando combinaciones de instancias claves de rE2 y rE3, es decir con respecto al resto de los roles que participan en R. Por lo que denotamos *rr* a las componentes pertenecientes al resto de roles y añadimos a la relación R (figura 5.13), proveniente de la interrelación ternaria, la siguiente sentencia de verificación, dependiendo:

1. Si (1<i<j) y (j≠N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_(n-
1)_R CHECK ((i ≤ SELECT count(cod_e1) FROM R GROUP
BY rr) AND (j ≥ SELECT count(cod_e1) FROM R GROUP
BY rr))
```

2. Si (1<i<j) y (j=N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_(n-
1)_R CHECK ((i ≤ SELECT count(cod_e1) FROM R GROUP
BY rr))
```

3. Si (1=i<j) y (j≠N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_(n-
1)_R CHECK ((j ≥ SELECT count(cod_e1) FROM R GROUP
BY rr))
```

4. Si (1≠i =j) y (j≠N)

```
ALTER TABLE R ADD CONSTRAINT Participación_rE1_(n-
1)_R CHECK ((i = SELECT count(cod_e1) FROM R GROUP
BY rr))
```

A continuación estudiaremos las distintas posibilidades que podemos encontrarnos dentro de una interrelación ternaria, distinguiendo cada uno de ellos porque el valor máximo de alguna de las cardinalidades sea uno, ya que este valor

influye en la estructura de la transformación. Utilizaremos la misma notación empleada para las IB, aunque no utilizaremos los atributos opcionales ya que en R estos serán obligatorios y añadiremos una nueva restricción, *UN* que indica la unicidad de un o más atributos dentro de una interrelación. En la exposición de cada uno de los casos solo presentaremos aquellas restricciones de cardinalidad válidas. La cardinalidad de entidad al no influir en el resto de cardinalidades puede tomar cualquier valor, independientemente de las restricciones de Merise y de Chen por lo tanto en todos los casos se añade el procedimiento de control *Obligatoriedad_X_R* que se añadirá a la transformación general si existe alguna entidad posee la participación obligatoria.

El caso 0 es el más general, tanto las cardinalidades de Merise como de Chen son N:M:P, esto significa que todas las máximas son mayores que uno (tabla 5.5).

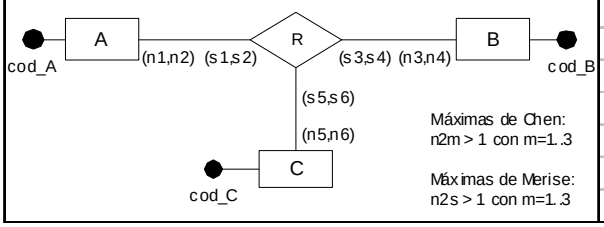
Caso	Chen	Merise	Transformación general + Restricciones
0	N:M:P	N:M:P	PK en R: cod_A, cod_B, cod_C Obligatoriedad_X_R Participación_rX_R Participación_rX_(n-1)_R
			

Tabla 5.5: Transformación de IT para el caso 0

Se añadirán tantas aserciones para la controlar la obligatoriedad como entidades obligatorias halla, lo mismo nos ocurrirá con los procedimientos Participación_rX_R y Participación_rX_(n-1)_R, introducidos de forma genérica ya que dependerán del valor que tomen las cardinalidades. Un caso especial sería, si todas las entidades participan de forma opcional en R y todas las cardinalidades son (1,n) no haría falta añadir ningún tipo de mecanismo de control solamente la clave primaria a la relación R. Este es el caso más sencillo y menos restrictivo.

El caso 1 contiene las siguientes correspondencias N:1:1 y 1:N:M, ya que al imponer en Merise una cardinalidad máxima de uno esta influye en las restricciones de cardinalidad de Chen, forzando a que tomen el valor de uno (tabla 5.6). Esto viene provocado por la equivalencia entre cardinalidad máxima uno y la dependencia funcional que provoca.

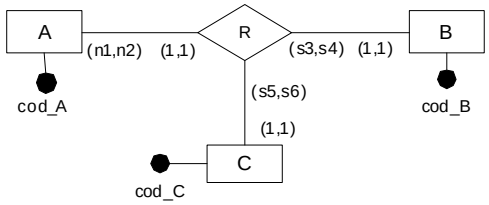
Caso	Chen	Merise	Transformación general + Restricciones
1	N:1:1	1:M:P	PK en R: cod_A UN en R: (cod_A, cod_B), (cod_A, cod_C) Obligatoriedad_X_R Participación_rB_R Participación_rC_R Participación_rA_(n-1)_R
			

Tabla 5.6: Transformación de IT para el caso 1.

En el caso 2, la imposición de dos de las cardinalidades máximas de Merise a 1, provocan que las cardinalidades máximas de Chen sean todas 1 (tabla 5.7).

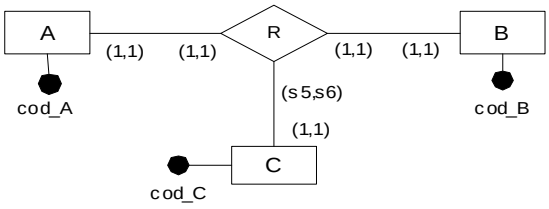
Caso	Chen	Merise	Transformación general + Restricciones
2	1:1:1	1:1:P	CC en R: cod_A, cod_B UN en R: (cod_A, cod_B), (cod_A, cod_C) (cod_B, cod_C) Obligatoriedad_X_R Participación_rC_R
			

Tabla 5.7: Transformación de IT para el caso 2.

Un caso particular del caso dos sería cuando todas las restricciones de cardinalidad máxima son 1. Las claves candidatas serían todos y cada uno de los IP de las entidades A,B y C, y se tendrían que cumplir la unicidad para cada uno de los pares formados por estos IP (tabla 5.7).

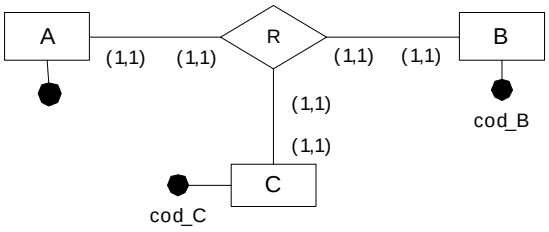
Caso	Chen	Merise	Transformación general + Restricciones
3	1:1:1	1:1:1	CC en R: cod_A, cod_B, cod_C UN en R: (cod_A, cod_B), (cod_A, cod_C) (cod_B, cod_C) Obligatoriedad_X_R
			

Tabla 5.8: Transformación de IT para el caso 3.

Los posibles casos de correspondencia que pueden darse coexistiendo las cardinalidades de Merise y Chen se presentan en las tablas 5.9, 5.10 y 5.11.

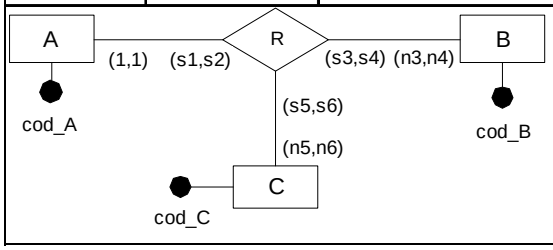
Caso	Chen	Merise	Transformación general + Restricciones
4	1:M:P	N:M:P	PK en R: cod_B, cod_C Obligatoriedad_X_R Participación_rX_R Participación_rB_(n-1)_R Participación_rC_(n-1)_R
			

Tabla 5.9: Transformación de IT para el caso 4

La tabla 5.10 presenta como varía la cardinalidad de Chen según vamos cambiando la cardinalidad máxima de Merise a cada uno de los roles que participan en la interrelación R.

.

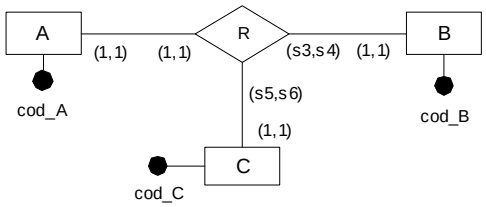
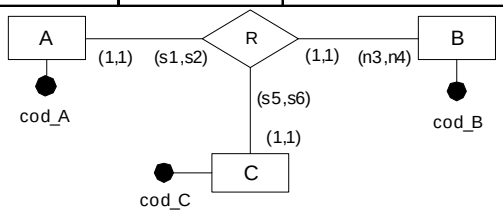
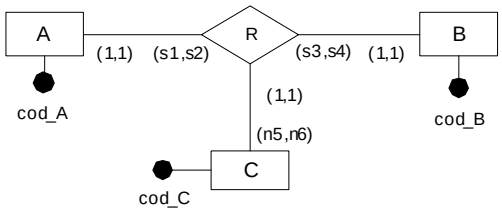
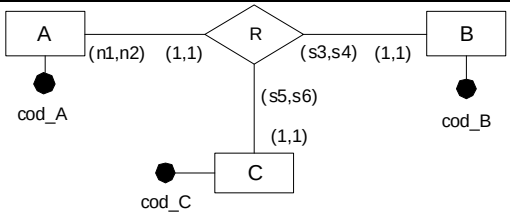
Caso	Chen	Merise	Transformación general + Restricciones
5	1:1:1	1:M:P	PK en R: cod_A UN en R: (cod_A, cod_B), (cod_A, cod_C) (cod_B, cod_C) Obligatoriedad_X_R Participación_rB_R Participación_rC_R
			
6	1:N:1	N:1:P	PK en R: cod_B UN en R: (cod_A, cod_B), (cod_B, cod_C) Obligatoriedad_X_R Participación_rA_R Participación_rC_R Participación_rB_(n-1)_R
			
7	1:1:N	N:M:1	PK en R: cod_C UN en R: (cod_A, cod_C), (cod_B, cod_C) Obligatoriedad_X_R Participación_rA_R Participación_rB_R Participación_rC_(n-1)_R
			
8	N:1:1	1:M:P	PK en R: cod_A UN en R: (cod_A, cod_B), (cod_A, cod_C) Obligatoriedad_X_R Participación_rB_R Participación_rC_R Participación_rA_(n-1)_R
			

Tabla 5.10: Transformación IT para los casos del 5 al 8

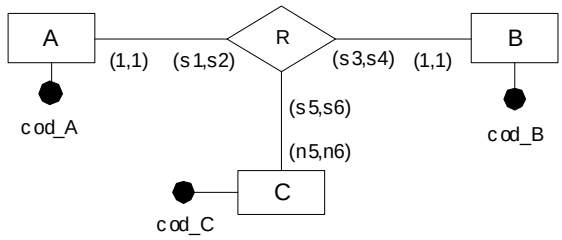
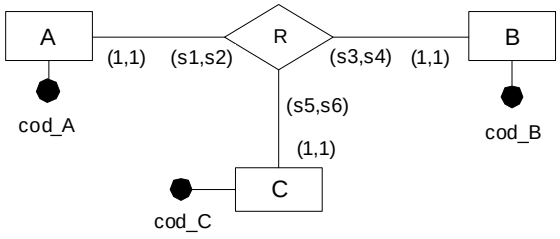
Caso	Chen	Merise	Transformación general + Restricciones
9	1:1:P	N:M:P	CC en R: (cod_B, cod_C), (cod_A, cod_C) Obligatoriedad_X_R Participación_rX_R Participación_rC_(n-1)_R
			
10	1:1:1	N:M:P	CC en R: (cod_B, cod_C), (cod_A, cod_C) (cod_A, cod_B) Obligatoriedad_X_R Participación_rX_R
			

Tabla5.11: Transformación IT para los casos 9 y 10

VALIDACIÓN

En este capítulo presentaremos la realización de dos experimentos para comprobar de forma empírica cuales son las repercusiones de nuestra contribución en el nivel conceptual como en el lógico dentro de la metodología de bases de datos. Para el nivel conceptual estudiaremos las ventajas ofrecidas por nuestra propuesta frente a otras aproximaciones. Para el nivel lógico, analizaremos el rendimiento de una base de datos real a la cual se le añaden los procedimientos de control necesarios para reflejar la semántica de las restricciones de cardinalidad propuestas.

6. EXPERIMENTACIÓN.

6.1. Experimento 1: Comparación entre la propuesta y otras aproximaciones, en la semántica que refleja las restricciones de cardinalidad

Este experimento se plantea para diseñadores expertos de bases de datos ya que los experimentos realizados nos indican que uno de los problemas más importantes dentro de los diseñadores inexpertos es la de recoger la semántica del UD a través del constructor interrelación y sus propiedades. Dentro de esto último, lo que mas le cuesta es detectar el grado de la interrelación y las restricciones de cardinalidad para interrelaciones de grado superior. Por lo tanto, para sesgar este tipo de errores tomamos una muestra de diseñadores expertos, con ellos podremos analizar si detectan las ventajas ofrecidas en nuestra propuesta para las restricciones de cardinalidad en interrelaciones ternarias frente a las dos aproximaciones dentro de los modelos de conceptuales. Las dos aproximaciones que utilizaremos en el experimento son las más conocidas y empleadas; la aproximación de Merise y la de Chen.

Antes de presentar el experimento a los seis diseñadores expertos, que participan en el mismo, les recordamos las tres aproximaciones con las que tendrán que resolver el formulario que les daremos posteriormente, a través de una clase recordatorio de treinta minutos. Les advertimos que durante la sesión recordatorio podrían preguntar cualquier duda pero que durante el desarrollo del experimento se enfrentarían solos al problema. No se formularon ninguna pregunta, con lo que en un principio entendieron las distintas aproximaciones.

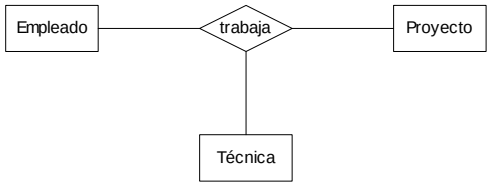
Después de la sesión recordatorio se les presentó el siguiente formulario:

Especificaciones

1. Dentro de la empresa los empleados que participan en proyectos lo hacen utilizando al menos dos técnicas.
2. Existen empleados que trabajan en proyectos sin utilizar una técnica.
3. Por política de empresa no se permite que los empleados trabajen en mas de tres proyectos.
4. La empresa solo puede abarcar 30 proyectos como máximo.
5. Para que no se solapen tareas dentro de un proyecto, una técnica dentro de un proyecto solo la puede desarrollar un empleado.
6. Toda técnica se relaciona con un empleado que la utiliza en un determinado proyecto.
7. Un empleado puede utilizar la misma técnica en diferentes proyectos.

A. Marque con una cruz aquellas especificaciones que se contemplen con la aproximación de Chen y represéntalas en el diagrama ER.

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☐ 6. ☐ 7. ☐

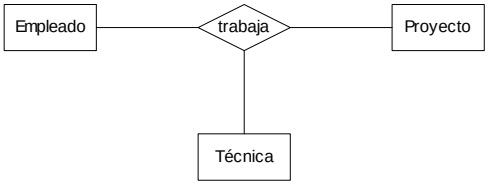


Indique el nivel de confianza:

Ninguno☐ Poco☐ Medio☐ Alto☐ Total☐

B. Marque con una cruz aquellas especificaciones que se contemplen con la aproximación de Merise y represéntalas en el diagrama ER.

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☐ 6. ☐ 7. ☐

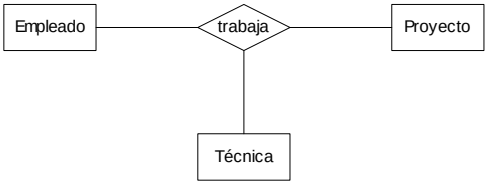


Indique el nivel de confianza:

Ninguno☐ Poco☐ Medio☐ Alto☐ Total☐

C. Marque con una cruz aquellas especificaciones que se contemplen con la aproximación de la propuesta y represéntalas en el diagrama ER.

1. ☐ 2. ☐ 3. ☐ 4. ☐ 5. ☐ 6. ☐ 7. ☐



Indique el nivel de confianza:

Ninguno☐ Poco☐ Medio☐ Alto☐ Total☐

Figura 6.1: Formulario presentado para la ejecución del experimento.

En la figura 6.1 podemos observar que el formulario consta de dos secciones principales. En la primera de ellas se muestra el conjunto de especificaciones a tener en cuenta en nuestro UD. Estas especificaciones se encuentran numeradas

para poder tabular aquellas que son detectadas y representadas por los participantes en el experimento. La segunda sección consta de tres apartados, uno para cada una de las aproximaciones a comparar, compuestas de tres preguntas: una para saber las especificaciones que han detectado dentro de cada aproximación, el nivel de confianza con que lo han hecho y la representación de estas especificaciones en el diagrama Entidad / Interrelación.

Resultados obtenidos en el primer experimento

A continuación presentaremos los resultados mas relevantes obtenidos en el análisis del experimento.

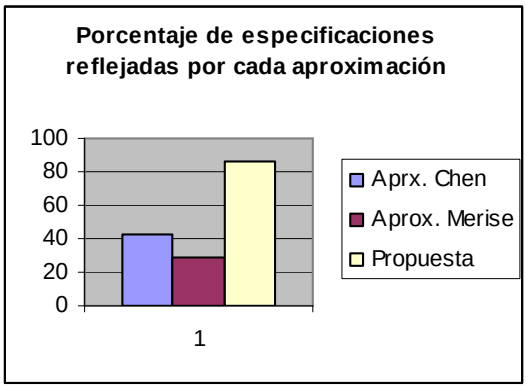


Gráfico 6.1: Número de especificaciones por aproximación

En el gráfico 6.1 podemos observar el porcentaje del número de especificaciones reflejadas por cada diseñador según la aproximación utilizada. No existe ninguna duda que la propuesta presentada en este trabajo es la que más especificaciones representa, por lo que es la más completa a la hora de reflejar la semántica del UD.

En el gráfico 6.2 mostramos el nivel de confianza que han mostrado cada uno de los diseñadores a la hora de contestar el formulario por cada aproximación. La

mayor confianza la ponen en la aproximación de Chen al ser la más utilizada por los mismos, lo que les ofrece mayor confianza; lo se podría ya esperar.

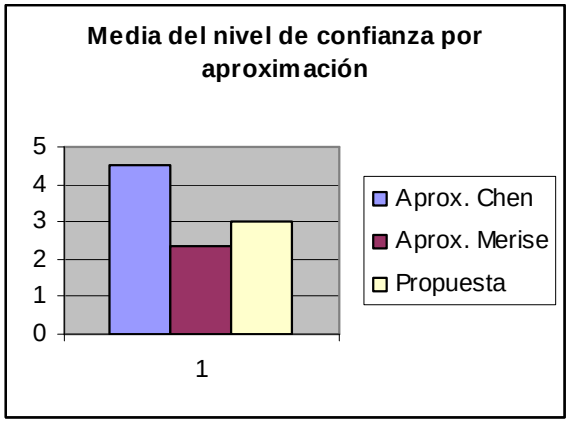


Gráfico 6.2: Nivel de confianza expuesto por cada aproximación

En el gráfico 6.3 comprobamos que los diseñadores al tener más confianza en la aproximación de Chen cometen menos errores en el diseño. El resultado obtenido por la propuesta está muy cerca de la aproximación de Chen por lo que demuestra que los diseñadores la han entendido y además saben reflejarla en el esquema.

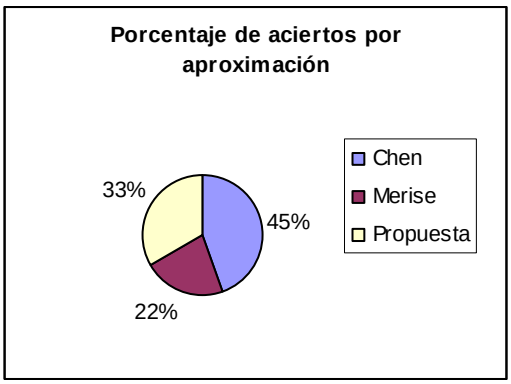


Gráfico 6.3: Aciertos en las especificaciones por aproximación

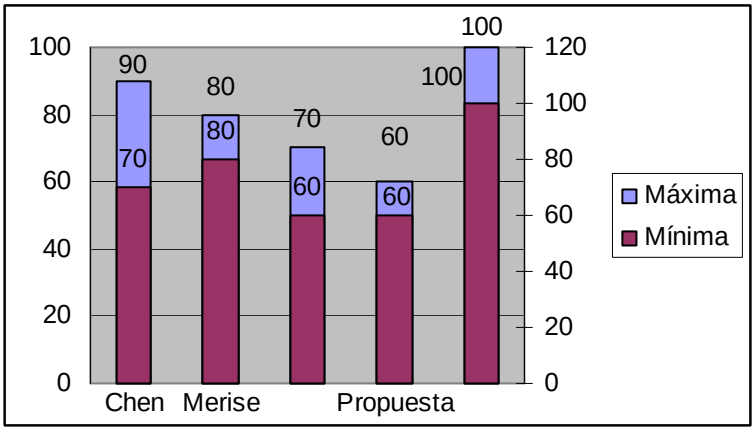


Gráfico 6.4: Porcentaje de aciertos según cardinalidad y aproximación

Las dos primera barras dibujadas en el gráfico 6.4 representan la aproximación de Chen y de Merise, las tres últimas son las consideradas en nuestra propuesta donde como se puede apreciar la que obtiene mejores resultados es la restricción de cardinalidad de entidad.

Las conclusiones a las que hemos llegado después de realizado el análisis es bastante positivo porque aunque los errores en el diseño no los hemos corregido completamente si que se distingue las diferentes aproximaciones y la semántica que refleja cada una de ellas. Por lo que aclaremos el concepto de cardinalidad lo que influye en ayudar a realizar un buen mapping entre modelos conceptuales y saber qué información se pierde al utilizar una aproximación u otra.

6.2. Experimento 2: Rendimiento de BD con la implementación de mecanismos de control de restricciones de cardinalidad

La validación de nuestro prototipo se funda en la aplicación directa de las técnicas desarrolladas en una base de datos real. El principal objetivo es verificar la validez

de las ideas presentadas. Para esto, va a ser necesario que se implemente el prototipo en un sistema determinado como ORACLE 9i, esta validación consistirá básicamente en probar, con distintos objetos de la base de datos y distintas operaciones, cómo afecta la inclusión de reglas ECA en el funcionamiento de una base de datos real. Los aspectos más importantes que tendremos en cuenta para comprobar si nuestra propuesta es válida, serán la implementación de nuestro prototipo sobre dicho sistema gestor ORACLE 9i y la comprobación del rendimiento de una base de datos real. Por eso, en este apartado vamos a explicar con profundidad el problema de rendimiento de los sistemas activos. Como cualquier otro tipo de sistema de software, los sistemas activos tienen que proporcionar sus funcionalidades de una manera eficaz. Por eso, muchos estudios se han realizado sobre la evaluación del rendimiento de estos sistemas analizando las ganancias y las pérdidas comparándolos con sistemas pasivos. De estos estudios se ha demostrado que el uso de los mecanismos proporcionados por los SGBDA resulta arma de dos filos. Es decir, que la utilización de estos mecanismos puede resultar intrincada porque el sistema de las reglas activas es un mecanismo de propósito general que introduce una sobrecarga en el SGBD, por eso, al comparar el rendimiento de la misma aplicación implementada con ó sin disparadores, la versión que usa estos disparadores es más lenta. También por las complejidades y los problemas asociados con la ejecución de los mismos, como se ha adelantado al principio de esta tesis, muchos diseñadores de las bases de datos rechazan el uso de los mecanismos activos para las grandes aplicaciones (especialmente cuando se puede sustituir aplicaciones tradicionales a estos mecanismos) y se recomienda a veces que los disparadores no deban ser intensivamente usados. Sin embargo, existen dos razones demuestran que la utilización de las reglas activas puede mejorar el rendimiento de las aplicaciones. La primera razón viene por la centralización de semántica de la aplicación en estas reglas. Debido a la centralización, se puede aplicar buenas y mejores técnicas para la optimización, y se puede evitar la redundancia de la verificación y los cambios en el entorno que se realizan fácilmente. La segunda razón, viene por considerar

que las reglas activas son instrumentos de afinación (*Tuning Instrument*) eficaz para hacer la aplicación más rápidamente. Un ejemplo de esto, es la creación de las vistas materializadas, supongamos que tenemos dos relaciones **`ORDER(ordernum, itemnum, qty, vendor)`**, y **`ITEM(itemnum, price)`**, y necesitamos frecuentemente consulta sobre la cantidad total de dinero en el orden de cada vendedor. Esta consulta puede ser muy costosa utilizando la transacción tradicional (*polling transaction*). Otra alternativa muy eficaz y rápida es crear una relación **`TOTAL_VENDOR (vendor, amount)`** donde la cantidad del dinero en el orden del vendedor se inserta cada vez que se realiza una modificación en la relación **`ORDER`** a través de utilizar las reglas activas.

Por eso, la conclusión final es que existe una necesidad grande para las bases de datos activos eficaces y existe específicamente la necesidad para llevar a cabo la implementación de las aplicaciones activas. El rendimiento de estas bases de datos juega un papel crucial y debe tenerse presente cuando los sistemas se diseñan y se llevan a cabo. Debe probarse el rendimiento de los prototipos de la base de datos activos desarrollados con respecto de una referencia (*Benchmark*) que nos permite medir el rendimiento de los SGBDA. Una referencia se podría usar para comparar el rendimiento de las distintas técnicas utilizadas en los sistemas activos como por ejemplo las distintas técnicas para detectar los eventos compuestos, las distintas arquitecturas de los sistemas activos, las tareas realizadas por el sistema activo con las mismas realizadas por el sistema tradicional, y cuánto ganaría si se escoge un DBMS activo en lugar de un DBMS pasivo.

Actualmente por las limitaciones del lenguaje de los disparadores y las limitaciones en el modelo de ejecución, no existe una referencia (*Benchmark*) de los sistemas activos relacionales. Pero sí existen algunas referencias de los sistemas orientados al objeto, como por ejemplo, BEAST y ACT-1 etc. En general, en estas referencias se centran en aspectos como la detección de eventos,

la gestión de reglas, y el modelo de ejecución, y estos aspectos no son comparables con los sistemas relacionales.

El factor principal para evaluar el rendimiento es el tiempo de la respuesta (Elapsed Time) de ejecución de un conjunto de consultas ejecutadas concurrentemente en el entorno del multiusuario (*Multiprogramming Level*), es decir, tener un número (N) de transacciones que se ejecutan al mismo tiempo sobre la base de datos. También, el diseño del sistema de área global (SGA) que tiene un efecto significativo en el rendimiento de los sistemas de bases de datos y que facilita tomar decisiones de los reemplazamientos inteligentes para aumentar la frecuencia de encontrar las sentencias pedidas de esta área. Además de esto, el rendimiento y la eficacia de los disparadores en el entorno de las bases de datos relacionales pueden ser experimentados a través de comparar el rendimiento de estos disparadores y los procedimientos almacenados. Por eso, en este trabajo hemos utilizado la misma aproximación y los factores anteriores para medir las diferencias entre el rendimiento de los disparadores y los procedimientos almacenados para evaluar la validez de nuestro prototipo de disparadores. Hemos desarrollado dos propuestas para realizar la misma transición en el estado de la base de datos de $\mathbf{S_i}$ a $\mathbf{S_j}$, una de estas propuestas utiliza los disparadores y la otra utiliza los procedimientos almacenados. Las dos propuestas son similares en la estructura y el funcionamiento, es decir, utilizan las mismas sentencias en las dos propuestas como por ejemplo, **`Cursor`**, **`If Then`**, etc., para asegurar que la diferencia en el rendimiento entre las dos propuestas viene por la naturaleza de la ejecución de cada propuesta. Además de esto, y para controlar el tiempo de la respuesta hemos utilizado transacciones que no transmiten resultados al usuario. Hemos aplicado la propuesta de disparadores dentro de una base de disparadores que contiene 600 disparadores activos para comprobar si el número de estos afectan el rendimiento de los mismos, y como se muestra a continuación:

Por la semejanza entre los disparadores del estándar SQL3 que hemos utilizado en nuestra propuesta y los disparadores de ORACLE, cada disparador conceptual se transfiere a un disparador de la plataforma ORACLE. En este apartado vamos a presentar un ejemplo de la transformación de los disparadores conceptuales a los de ORACLE, como se presenta a continuación:

(a) Por la ausencia de las opciones de las prioridades en los disparadores de ORACLE, necesitamos utilizar las variables globales como semáforos para controlar la ejecución de los disparadores evitando la no-terminación y el conflicto entre ellos, y para que los procedimientos almacenados sean invocados correctamente. Por esto, se usa una estructura de datos auxiliares (**`GlobalData`**) para definir los parámetros globales que vamos a utilizar para desactivar los disparadores sustituyendo las opciones de las prioridades. Las definiciones de las variables globales que se usan en dicho paquete son las siguientes:

- Se define como variable numérica global (**`Rowcnt`**) que actúa como contador de las tuplas modificadas, el proceso que se usa para contar el número de las tuplas modificadas que siempre se realizan a través de los disparadores de tipo (**`BEFORE/ROW`**), mientras que en los disparadores (**`AFTER/STATEMENT`**) se utiliza esta variable para construir los ciclos necesarios para llamar a las variables guardadas.

- Los parámetros booleanos globales (**`gbp`**) actúan como semáforos para controlar la ejecución de los disparadores en un conjunto de ellos para evitar la no-terminación y el conflicto. En el ejemplo que vamos a presentar, la mejor manera para definir estos semáforos, es especificar un semáforo para cada relación en el esquema relacional, esta manera reduce el número de las variables que se usan para cada disparador, como por ejemplo (**`gbp_Usuario, gbp_desfruta, gbp_perfil ,... etc.`**).

(b) Se transforma la opción **`IS_IT_ENABLE()`** que se usa como control para ejecutar la acción de cada disparador a una condición, se evalúa al principio de la ejecución. Como se muestra en cada disparador del ejemplo. Además, para cada tipo de entidad en el modelo relacional asociado con otra entidad, tiene cardinalidad mínima uno, se crean dos tablas de variables (*Array*), una para guardar las claves primarias o ajenas de las tuplas modificadas como por ejemplo (**`var_perfil`**) y la otra siempre queda vacía (**`emp_perfil`**) para reiniciar la primera tabla siempre que se termine un proceso de disparadores.

(c) Se transforma el disparador conceptual de tipo (**B/S**) de **USUARIO** al disparador de ORACLE (**`Del_T1_Usuario`**) donde se sustituye la opción (**`DISABLE(T3);`**) del disparador conceptual para desactivar el semáforo (**`gbp_disfruta:='Off';`**), y evitar la ejecución de los disparadores de la relación **DISFRUTA**. Aunque no existe disparador conceptual de tipo (**A/S**) de la entidad **USUARIO**, se necesita utilizar uno de este tipo (**`Del_T2_Usuario`**) en ORACLE para que active dicho semáforo desactivado en el anterior.

(d) Se transforma el disparador de tipo (**B/S**) de **DISFRUTA** al disparador de ORACLE (**`Del_T1_disfruta`**) que se usa para guardar las claves ajenas de las ocurrencias borradas, y se transforma el disparador de tipo (**A/S**) de la relación **DISFRUTA** al disparador de ORACLE (**`Del_T2_disfruta`**), donde se utiliza la función (**`Check_Usuario`**) para comprobar la semántica. En todos los disparadores que hemos utilizado, se comprueba la semántica de la tabla modificada al final de la ejecución, si la modificación no viola la semántica de los datos, la ejecución de estos se termina correctamente, o en el caso de que exista violación de la semántica, la ejecución se deshace, es decir, se genera un error de aplicación acompañada de un mensaje **AVISO** al usuario indicándole que su modificación viola la semántica y por tanto se anula la modificación, y el estado inicial de la base de datos se recupera. En estos dos

casos (aceptación y anulación) hay que reiniciar los valores de las variables globales utilizadas en estos disparadores, por ejemplo (**`var_perfil:=emp_perfil`**).

En la base de datos de usuario único, este usuario puede modificar los datos sin preocuparse de otros que actualizan a la vez los mismos datos. Sin embargo, en las bases de datos multiusuario, las sentencias dentro de las transacciones simultáneas múltiples pueden actualizar la misma base de datos. Por eso, en este apartado, vamos a experimentar las dos propuestas en el entorno multiusuario, es decir, tener un número (N) de la misma propuesta que se ejecutan concurrentemente sobre la base de datos. Este experimento pasa por la notación del tiempo de la respuesta (*Elapsed time*) que es el tiempo necesario para el proceso de una sentencia SQL, este proceso consiste en tres pasos que son; *Parse, Execute, y Fetch*. El primer paso es analizar la sentencia SQL o un boqueo de sentencias SQL al plan de ejecución. Este paso incluye las comprobaciones de la autorización de seguridad, la existencia de tablas, columnas, y otros objetos de referencias. En el segundo paso se ejecuta actualmente la sentencia SQL que actualizan los datos (Insert, Delete, Update), mientras en la sentencia SQL de recuperación de datos (Select) este paso identifica las tuplas seleccionadas. En el tercer paso se recuperan las tuplas seleccionadas por una consulta. Este paso se realiza sólo para la sentencia (Select).

Para determinar el tiempo de la respuesta de un experimento determinado, los archivos del rendimiento se analizan de la manera siguiente: Se considera que **`MPL`** es el nivel de la multiprogramación del experimento y **`N`** es el número de las iteraciones para ejecutar cada programa concurrentemente. Se considera que $\mathbf{T_{ij}}$ representa el tiempo de la respuesta requerida desde la instancia del arranque hasta acabar la ejecución. Un tiempo medio de la respuesta de cada nivel de programa $\mathbf{MPL_i}$ será calculado dividiendo la suma de los tiempos de dicho nivel por número de iteración.

$$\text{Elapsed Time } (MPL_i) = \left(\sum_{J=1}^{N} T_{ij}\right) / N$$

Para obtener una referencia (*Benchmark*) de nuestro experimento, hemos ejecutado una transacción (**MPL=1, N=5**) de cada propuesta para comprobar la semántica de cardinalidades al borrar tuplas de una tabla de la base de datos experimental, esta tabla contiene 10000 tuplas y el resultado ha demostrado que la diferencia entre el rendimiento de las dos propuestas varía entre 0%-2% a favor de los procedimientos almacenados, es decir, que no existe una diferencia significativa entre las dos propuestas al ejecutarlas en el nivel (**MPL=1**). Como se puede ver en la gráfico 6.2, la línea continua muestra el rendimiento de los disparadores, mientras la línea discontinua muestra el rendimiento de los procedimientos almacenados.

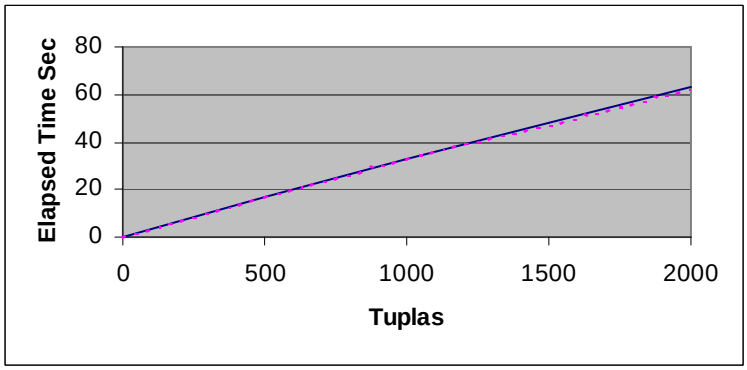


Figura 6.2: Tiempo empleado en la ejecución de disparadores y procedimeintos almacenados

Hemos utilizado las dos herramientas básicas **SQL TRACE** y **TKPROF** de ORACLE para diagnosticar el rendimiento de nuestro experimento. Estas dos herramientas le permiten al usuario evaluar la eficacia de la ejecución de las

sentencias SQL. La **SQL TRACE** proporciona información sobre el rendimiento de cada sentencia SQL individual, como *Parse*, *Execute*, *Fetch counts*, *CPU time*, *Elapsed times*, etc. Al ejecutar el programa **TKPROF** se convierte el archivo obtenido de **SQL TRACE** en un archivo legible que tiene estadísticas sobre dichos recursos utilizados para la ejecución de cada sentencia SQL.

Propuesta de Procedimientos

Call	Count	CPU	Elapsed	Disk	Query	Current	Rows
Parse	18	0.01	0.01	0	0	0	0
Execute	2017	3.14	3.65	0	86	1117	2005
Fetch	4050	31.80	31.88	0	82057	12006	4050
Total	6085	34.95	35.54	0	82143	13123	6055

Propuesta de Disparadores

Call	Count	CPU	Elapsed	Disk	Query	Current	Rows
Parse	21	0.00	0.00	0	0	0	0
Execu	3019	4.52	4.97	0	64	1040	2006

te							
Fetch	**4053**	**32.13**	**32.21**	**0**	**78068**	**12006**	**4051**
-----	**-----**	**------**	**------**	**----**	**------**	**------**	**----**
Total	**7093**	**36.65**	**37,18**	**0**	**78101**	**14153**	**6057**

Tabla 6.1. Estadísticas totales del rendimiento.

Se debe señalar que estas estadísticas proporcionan un resumen de la ejecución de dos tipos de sentencias. Las primeras, son sentencias ejecutadas por los usuarios. Las segundas, son sentencias adicionales (Recursivos SQL) que se usan algunas veces para ejecutar sentencias emitidas por un usuario, como por ejemplo, si se inserta una tupla en una tabla que no tiene bastante espacio para sostener esa tupla, entonces ORACLE hace una llamada recursiva para asignar el espacio dinámicamente. También se generan las llamadas recursivas cuando la información del diccionario de datos no está disponible en el caché y debe recuperarse del disco. También se puede detectar en estas estadísticas que el máximo tiempo empleado en la ejecución de cada propuesta, es el tiempo de recuperación (*fetch*) de las tuplas seleccionadas, es decir, el tiempo empleado por las consultas ejecutadas por la transacción.

Otro experimento que hemos hecho utilizando múltiples programas concurrentemente ejecutados (**MPL=2,3,4,....,**) y (**N=5**), cada uno de estos programas realiza la misma actualización sobre nuestra base de datos experimental. En el caso de disparadores, cada programa activa dos de ellos para hacer la actualización de nuestro experimento, es decir, que el número de los disparadores activados en una instancia determinada es (**MPL*2**). Mientras en el caso de los procedimientos, cada programa llama a un solo procedimiento para hacer la misma actualización, es decir, que el número de los procedimientos llamados en una instancia determinada es (**MPL**). Por esta razón, cada vez que se

multiplica el uso de los disparadores el rendimiento de estos resulta afectado negativamente comparando con el rendimiento de los procedimientos, debido al tiempo gastado en llamar estos disparadores múltiples y encargar y tratarlos en SGA. Un fragmento de nuestro experimento consiste en ejecutar la misma transacción de las dos propuestas para borrar 1000 tuplas y comprobar la semántica de cardinalidades de una tabla que contiene 10000 tuplas en el entorno multiusuario y el resultado se muestra en la gráfico 6.5, donde la línea continua muestra el rendimiento de los disparadores, mientras la línea discontinua muestra el rendimiento de los procedimientos almacenados.

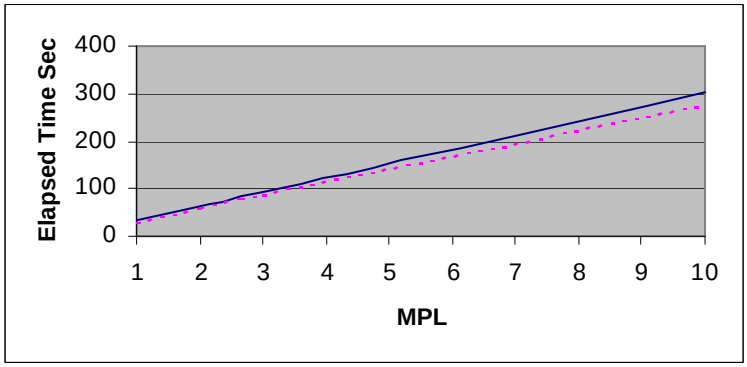


Gráfico 6.5: El rendimiento de las dos propuestas en el entorno multiusuario.

Los procedimientos almacenados y los disparadores una vez cargados en la SGA (System Global Area) permanecen allí hasta que se decida retirarlos de la memoria (*Paged-out*). La SGA es un grupo de estructuras de memoria compartida que almacena datos e información que controla cada instancia de ORACLE y esta memoria será cargada cuando se arranca la instancia ORACLE y será liberada cuando se cierre la instancia. Todos los usuarios conectados a una misma instancia pueden usar la información contenida en la SGA de la instancia. Una de las partes importantes de la SGA es el área caché SQL, el contenido de esta área se divide lógicamente entre dos partes activas e inactivas (*hot and cold*). Las

partes activas siempre estarán en la memoria porque están siendo llamadas por los programas, pero algunas de las partes inactivas pueden retirarse de la memoria, y resulta una pérdida del rendimiento al devolverlos dentro de la misma otra vez. Cuando un usuario solicita un acceso a la base de datos mediante una sentencia SQL el plan de ejecución de dicha sentencia se almacena en un área SQL. Dicha área puede ser compartida por otro usuario, que solicita acceso mediante la misma sentencia. Es decir, cuando se ejecuta un código de aplicación, ORACLE intenta primero reutilizar el código existente en el área caché que se ha ejecutado previamente y puede compartirse. Si ORACLE es incapaz de usar el código existente, entonces una nueva versión ejecutable del código de la aplicación debe construirse en el área caché. Cuando una aplicación llama a una sentencia SQL analizada (*parse*), si la representación de esta sentencia no existe en el área caché, ORACLE analiza y guarda la forma de la misma. Esta operación es conocida como análisis duro o un fallo (*miss*) en el área caché. Para reducir este fallo se puede asegurar que todas las sentencias SQL compartidas están en área caché, de esta manera no se tiene que volver a realizar la búsqueda de la consulta.

El rendimiento del área caché de ORACLE se investiga a través de la vista **`V$LIBRARYCACHE`** que contiene información de la actividad del área y para examinar estas actividades de cada *namespace,* se usa la consulta siguiente:

```
SELECT namespace
, pins
, pinhits
, reloads
FROM V$LIBRARYCACHE;
```

En el campo de *pins* se presenta el número de veces que el sistema emite una petición para coger los objetos del área caché para acceder a ellos. En *pinhits* se presenta el número de veces que el objeto está fijado y accedido en la memoria. En *reloads* se presenta el número de veces que el sistema reinicia y recoge un objeto que ha sido retirado de la memoria.

De la tabla 6.2 se muestra las actividades registradas del área caché al ejecutar las dos propuestas para realizar la misma actualización. El porcentaje *Hit Ratio* indica cuántas veces los usuarios piden realmente y pueden usar o ejecutar los objetos que están fijado en el área caché. Este porcentaje debe ser muy alto (95% o más) para cualquier *namespaces* que está experimentado. Cuando el porcentaje es bajo esto significa que existen cantidades grandes de SQL que no son compartidos. Para calcular el porcentaje del *Hit Ratio* se usa la fórmula siguiente:

Library Cache Hit Ratio = sum(pinhits) / sum(pins).

Examinando los datos en la tabla 6.2, se puede ver que el porcentaje del *hitratio* de la propuesta de procedimientos (**0,996**) significa que solo existe (**0,004**) de la ejecución como resultado del reanálisis, mientras que el porcentaje del *hitratio* de la propuesta de disparadores (**0,999**) significa que solo existe (**0,001**) de la ejecución como resultado del reanálisis. Esto significa no existe diferencia significativa en el tratamiento de la memoria en las dos propuestas, pero si existe diferencia a favor de utilizar los disparadores y como hemos adelantado en este tesis porque estos proporcionan más ventajas en términos de la flexibilidad y el control de la consistencia de la base de datos.

Propuesta de Procedimientos

NAMESPACE	PINS	PINHITS	RELOADS
SQL AREA	6270	6246	18
TABLE/PROCEDURE	137	134	0

BODY	**0**	**0**	**0**
TRIGGER	**0**	**0**	**0**

Propuesta de Disparadores

NAMESPACE	**PINS**	**PINHITS**	**RELOADS**
SQL AREA	**13263**	**13248**	**15**
TABLE/PROCEDURE	**23118**	**23113**	**3**
BODY	**3**	**3**	**0**
TRIGGER	**12**	**12**	**0**

Tabla 6.2. Estadísticas del área caché.

CONCLUSIONES Y LÍNEAS FUTURAS

El capítulo de conclusiones y líneas futuras pretende exponer si se han alcanzado los objetivos previstos al comienzo del trabajo doctoral y presentar qué caminos se pueden tomar a partir de este trabajo.

7.1. Conclusiones

Las conclusiones de este trabajo de tesis doctoral se derivan de la consecución de los objetivos planteados al inicio de este trabajo. Dentro de una metodología de desarrollo de bases de datos nuestros objetivos se enfocaban dentro sus dos primeras fases.

En la fase de diseño conceptual se han estudiado distintos modelos agrupados por familias y en ellos se ha detectado una amplia variedad de propuestas para reflejar la restricción de cardinalidad. Esto unido a que la mayoría de las herramientas CASE implementan una porción de estos modelos y no de forma concreta, repercute de forma negativa en su entendimiento, creando ambigüedad y confusión en la aplicación de estos modelos a un determinado UD.

El constructor que más dificultad entraña es la interrelación, concretamente una de sus propiedades más importantes para imponer restricciones que reflejan el UD, la cardinalidad. Hemos estudiado las distintas aproximaciones dadas por cada familia de modelos y basándonos en ellos hemos realizado la propuesta de definición, primeramente de interrelación, que nos ha servido de base para definir las restricciones de cardinalidad de entidad, la de interrelación de Chen y la de interrelación de Merise. Estos tres tipos de restricción contemplan las limitaciones más importantes dentro de las interrelaciones binarias y ternarias, es decir, reflejan la semántica “útil” necesaria para contemplar las reglas de negocio de nuestro sistema o UD.

Otra ambigüedad encontrada dentro de las interrelaciones es el tratamiento de información inaplicable o desconocida, este problema solo aparece en las interrelaciones de grado superior, que también infunden confusión a la hora de su representación. Para evitar el manejo de esta información, creamos un nuevo

constructor, la interrelación complementaria, que adjunta la información necesaria a una interrelación sin tener que violar su definición.

En la fase de modelado lógico, el modelo por excelencia es el relacional, por lo que la mayoría de las aproximaciones que exponen los modelos conceptuales presentan reglas de transformación para pasar de un esquema conceptual a un esquema relacional. Comprobamos que la mayoría de las reglas de transformación no toman en cuenta valor de las restricciones de cardinalidad mínima y máxima, perdiendo la semántica del UD, tomando solo una porción del mismo. Para evitar la pérdida de semántica presentamos mecanismos de control para cada una de las restricciones de cardinalidad propuestas.

Para validar la propuesta presentada se han realizado dos experimentaciones, cubriendo las dos partes en las que se divide la propuesta de este trabajo de tesis doctoral.

Los resultados empíricos obtenidos para el primer experimento, donde se compara nuestra propuesta con las dos aproximaciones mas utilizadas para representar las restricciones de cardinalidad, se resumen en: la detección de las cardinalidades a través de especificaciones en lenguaje natural es una tarea compleja que mejora cuando la definición de estas restricciones se clarifica y se clasifica. Por lo que nuestra ayuda a los diseñadores expertos a diferenciar, dentro de un conjunto de especificaciones, si detectan la cardinalidad de qué tipo se trata. Con lo que se potencia que puedan utilizar distintos modelos de datos conceptuales porque tienen claro como se realiza el mapping entre restricciones de cardinalidad.

Los resultados obtenidos en el segundo experimento miden como afecta al rendimiento de una base de datos real, la implementación de mecanismos de control para el cumplimiento de las restricciones de cardinalidad propuestas. Estos mecanismos de control han sido creados en el SGBD Oracle 9i empleando para ello, disparadores, como primera técnica, y procedimientos almacenados, como

segunda, programados en el lenguaje procedimental PL/SQL. Tanto los disparadores como los procedimientos se han implementado siguiendo una política pesimista en la que siempre que se realiza una operación sobre la base de datos primeramente se comprueba si viola las restricciones de cardinalidad. Si es así, se aborta la transacción que ha generado esta violación y se informa al usuario. Los resultados obtenidos muestran que los tiempos de ejecución no varían al utilizar un mecanismo u otro de control, es decir, la implementación a través de disparadores o de procedimientos almacenados no varía. El único problema encontrado es que si la cardinalidad es muy restrictiva, al utilizar una política pesimista no se pueden realizar muchas de operaciones ya que la mayoría son abortados por no cumplir las restricciones impuestas.

7.2. Difusión de resultados

La evolución de este trabajo de tesis doctoral se ve respaldado por un conjunto de publicaciones que nos han ayudado a mejor la propuesta y a ampliarla.

La primera aproximación, Cuadra et al (1999), aporta una solución para la pérdida de semántica en interrelaciones binarias en su transformación al modelo relacional. La solución propuesta se basaba en la aplicación de reglas ECA y solo consideraba la aproximación de Chen.

7.3. Líneas Futuras

Las líneas futuras más inmediatas vienen motivadas por la necesidad de incorporar la propuesta realizada en este trabajo de tesis doctoral dentro de una herramienta CASE, con ello completaríamos el ciclo para un buen desarrollo de una base de datos.

La herramienta CASE PANDORA será el prototipo donde se llevarán a cabo estas propuestas, incorporando dentro de cada una de las fases del diseño a las que da soporte:

- ✓ En la fase de diseño conceptual:
  - ☑ Representación de las restricciones de cardinalidad de entidad, de interrelación de Merise y de Chen.
  - ☑ Representación de la interrelación complementaria para el manejo de información desconocida o inaplicable.
  - ☑ Validación sintáctica y semántica de las restricciones de cardinalidad. La validación sintáctica se realizará a través de la aplicación reglas basadas en los axiomas de Armstrong y en las presentadas por McAllister (1998). La validación semántica se llevará a cabo a través de un dialogo con el diseñador.
- ✓ En la fase de diseño lógico:
  - ☑ Realización de una transformación de esquemas parametrizada por el diseñador. Para aquellas construcciones que pueden tener distintas transformaciones se dará la opción e incluso aconsejar al diseñador, cual de ellas es la recomendable.

Además de la incorporación de la propuesta a una herramienta CASE también se debería ampliar la experimentación en la implementación de los mecanismos de control en SGBD. Primero, utilizando políticas optimistas que comprueben la semántica cuando el responsable de la base de datos así lo requiera, dando más flexibilidad a las operaciones que se llevan a cabo dentro de la BD. Y en segundo lugar, incorporando estos mecanismos de control en otros SGBD's aislando el problema de las características del cada SGBD para

poder comprobar que la incorporación de semántica en las BD no ralentiza su operabilidad.

REFERENCIAS

Abrial, J.R., (1974). *Data semantics.* North-Holland, Amsterdam, pp. 1-59, 1974.

Al-Jumaily, H., Cuadra, D. & Martínez, P. (2002). T*he Execution Model for transforming the associate semantics to the Restriction of Cardinality in DBMS Relational*.Proceedings 4th International Conference on Entrepise Information Systems, Ciudad Real, Spain, 3-6 April, 2002.

Armstrong, W.W., (1974). Dependency structures of database relationship. Inf. Process. 74, pp: 580-583, 1974.

Balaban, M. & Shoval, P. (2002). MEER- An EER model enhanced with structure methods. Information Systems 27, pp 245-275, 2002.

Barker, R.(1990). Case*Method: Entity Relationship Modelling. Addison Wesley, 1990.

Batini, C., Ceri, S. & Navathe, S. B. (1992). Conceptual Database Design: An Entity-Relationship Approach. Benjamin/Cummings, CA.

Batra, D. & Zanakis , H. (1994). A Conceptual Database Design Approach Based on Rules and Heuristics. European Journal of Information Systems, Vol. 3, Nº 3, 228-239.

Batra, D. & Antony, S. R. (1994). *Novice Errors in Conceptual Database Design*. European Journal of Information Systems, Vol. 3, Nº 1, pp 57-69, 1994.

Boman et al. (1997). *Conceptual Modelling.* Prentice Hall Series in Computer Science, 1997.

Bouzeghoub, M. & Gardarin, G. (1984). The Design of an Expert System for Database Design. *New Applications of Databases* , Gardarin, G. and Gelenbe, E., De. Academic Press, London, 1984.

Bouzeghoub, M., Kedad, Z. & Métais E (2000). *CASE Tools: Computer Support for Conceptual Modelling*. In Advanced Database Technology and Design, Díaz and Piattini (Eds.), Artech House, 2000.

Bouzeghoud, E. & Metais, E. (1991 a). Semantic modeling and object modeling: two complementary paradigms. Entity-Relationship Approach (ER'91) pp. 325-348, North-Holland, Amsterdam, 1991.

Bouzeghoud, E. & Metais, E. (1991 b). Semantic modeling of object oriented database. Very Large Data Bases (VLDB'91), pp. 3-14, September 1991.

Bruce, T.A., (1992). Designing Quality Databases with IDEF1X Information Models. Dorset House, New York, 1992.

Buneman, P. et. al. (1991). Using power domains to generalize relational databases. Theoretical Computer Science 91, 23-55.

Burg, J.F.M (1997). Linguistic Instruments in Requirements Engineering. Tesis Doctoral, IOS Press, 1997.

Burg, J.F.M. & Van de Riet, R.P (1996). Analysing Informal Requirements Specifications: A First Step towards Conceptual Modeling. Applications of Natural Language to Information Systems. Van de Riet, R.P., Burg, J.F.M., Van der Vos, A.J. (Eds.). IOS Press, 1996.

Camps, R. (2002). From ternary relationship to relational tables: a case against common beliefs. ACM SIGMOD Record, Volume 31, Issue 2, pp. 46-49, (June 2002).

Castro et al. (2002). Integrating Intelligent Methodological Tutoring assistance in a CASE platform: the PANDORA experience. Informing Science + IT Education Conference, Cork, Ireland, June 19-21,2002.

Castro, E., Cuadra, D & Martínez. P. (2003). An empirical perspective of using ternary relationships in database conceptual modeling. INFORMATICS IN EDUCATION (in preparation) © Institute of Mathematics and Informatics ISSN 1648-5831.

Ceri, S. & Fraternalli, P. (1997). Designing database applications with objects and rules : the IDEA Methodology. Addison-Wesley.

Chamberlin, D.D. & Raymond, F.B. SEQUEL: A structure English query language. Proc. ACM-SIGMOD, Workshop, Ann Arbor, Michigan, Mayo, 1974.

Chen, P. P. (1976). The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems. 1, 1, 9-36.

Chen, P.P. (1977). *The Entity-Relationship Model-- A basis for the Enterprise View of Data*
Proc. of National Computer Conference, AFIPS Press, pp. 77-84, 1977.

Chen, P.P. (1983). English Sentence Structure and Entity-Relationship Diagram *Information Sciences*, Vol. 1, No. 1, pp. 127-149, Elsevier, May 1983.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. CACM 13 (6), June 1970.

Codd, E. F. (1979). Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems, 4 (4), 397-434, December 1979.

Columbetti, M et al., (1985). NLDA: A Natural Language Reasoning System for the Analysis of Data Base Requirements. *Methodology and Tools for Database Design*, in Ceri, S. (Eds.), North-Holland Publishing Company, 1985, pp. 163-180.

Cuadra et al. (1999). Control de Restricciones de Cardinalidad en una Metodología de Desarrollo de Bases de Datos Relacionales. Novática, Nº 140.

Cuadra, D., Nieto, C., Martínez, P.,Castro, E., Velasco, M.(2002). *Preserving Relationship Cardinality Constraints in Relational Schemata.* Database Integrity: Challenges and Solutions. Idea Group Publishing Ed, 2002.

Cuadra, D., Martínez, P.,Castro, E.(2003). *Dealing with Relationship Cardinality Constraints in Relational Database Design.* Effective Databases for Text & Document Management. Idea Group Publishing.

Date, C. J. (1986). An Introduction to Database Systems. Addison-Wesley, Reading, Mass.

Date, C. J. (1990). An Introduction to Database Systems. 5th ed. Addison-Wesley, Reading, Mass.

Date, C. J. (1995). An Introduction to Database Systems. 6th ed. Addison-Wesley, Reading, Mass.

De Miguel, A. & Piattini. M. (1993). Fundamentos y modelos de Bases de Datos, Ed. Rama, España, 1993.

De Miguel, A., Piattini, M. & Marcos, E, (1999). *Diseño de Bases de Datos Relacionales*. Rama Ed., 1999.

Dey, D., Storey, V.C. & Barron T.M. (1999). *Improving Database Design through the Analysis of Relationships.* ACM Transaction on Database Systems, Vol. 24, No.4, pp 453-486.

Dullea, J., & Song, IL-Y, (1998). *An Analysis of the Structural Validity of Ternary Relationships in Entity-Relationship Modelling*. Proceedings of the 7th International Conference on Information and Knowledge Management, CIKM'98, pp 331-339, November, 1998.

Eick, C (1984). From Natural Language Requirements to Good Database Definitions: A Database Design Methodology. *Proceedings of the International Conference on Data Engineering*, Los Angeles, April 1984, pp. 324-331.

Elmasri, R. & Navathe, S. (1994). *Fundamentals of Database Systems*. 2nd ed. Benjamin-Cummings, 1994.

Elmasri, R. & Navathe, S.B. (2000). Fundamentals of Database Systems. Third Edition. Addison-Wesley.

Elmasri, R., Weeldreyer, J. & Hevner, A. (1985). The category concept: an extension to the entity-relationship model. Data & Knowledge Engineering 1 (1): 75-166, June 1985.

Fagin, R. (1977). Multivalued Dependencies and a New Normal Form for Relational Databases. ACM TODS, 2, 3, 1977.

Fahrner, C. & Vossen, G. (1995). A survey of database design transformations based on the Entity-Relationship model. Data & Knowledge Engineering 15, 213-250, 1995.

Ferg, S., (1991). *Cardinality Concepts in Entity Relationship*. 10 th, International Conference on the Entity Relationship Approach, 1991.

Feyer, T. & Thalheim, B. (1999). *E/R based scenario modeling for rapid prototyping of web information services.* ER'99 Workshop on the World Wide Web and Conceptual Modeling, vol. 10, November 1999, pp. 353-263.

Frederiks, P.J.M., Hofstede, A.H.M. & Lippe, E. (1997). A unifying framework for conceptual data modelling concepts. Information and Software Technology, 39, pp. 15-25, 1997.

Génova, G., Llorens, J. & Martínez, P. (2001). *Semantics of the Minimum Multiplicity in Ternary Associations in UML*. UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools. Editado por M. Gogolla, C. Kobryn, Springer Verlag, LNCS 2185, pp. 329-342, 2001

Gómez, J., Cachero, C. & Pastor, D. (2000). *Extending a conceptual modeling approach to web application design.* CAISE-2000, North-Holland, Amsterdam, pp. 79-93, 2000.

Gorla, N., (2001). *An Object-Oriented Database Design for improved performance*. Data & Knowledge Engineering, 37, pp117-138, 2001.

Halpin, T.A. (2001). Information Modeling and Relational Databases : from Conceptual Analysis to Logical Design. San Francisco. Morgan Kaufman Ed., 2001.

Hansen, G. & Hansen, J. (1995). Database Management and Design. Prentice-Hall , 1995.

Hartmann, S. (1998). On the Consistency of Int-cardinality Constraints. Conceptual Modeling ER'98, 17th International Conference on Conceptual Modeling , Singapure. Ed. Springer, 1507, pp. 150-163, 1998.

Hartmann, S. (b) (2001). On implication problem for cardinality constraints and functional dependencies. Annals of Mathematics and Artificial Intelligence, 33, pp. 253-307, 2001.

Hartmann, S.(a) (2001). Decomposing relationship types by pivoting and schema equivalence. Data & Knowledge Engineering, 39, pp. 75-79, 2001.

Hartmann, T. et. al. (1994). Revised Version of the Conceptual Modelling and Design Language TROLL.Proceedings ISCORE Workshop, Amsterdam, 89-103.

Hull, R. & King, R. (1987). Semantic Database Modelling: Survey, Application, and Research Issues. ACM Computing Surveys 19 (3), 201-260, 1987.

Jones, T. H. & Song, IL-Y, (2000). Binary Equivalents of Ternary Relationships in Entity-Relationship Modeling: a Logical Decomposition Approach. Journal of Database Managment, April-June 2000, pp 12-19, 2000.

Kerschberg, L., Klung, A., & Tsichritzis, D., (1976). *A taxonomy of Data Models*. In Systems for Large Data Bases. North-Holland, Amsterdam, pp. 43-64, 1976.

Kolp, M. & Zimányi, E. (2000). Enhanced ER to relational mapping and interrelational normalization. Information and Software Technology 42, pp:1057-1073.

Korth, H.F. & Silberschatz, A. (1991). Database Systems Concepts. Ed. McGrawhill, New York, 1991.

Kroenke, D.M. (1992). Database Processing – Fundamentals, Design, Implementation. Ed. Macmillan, New York, 1992.

Lazarevic, B. & Misic, V. (1991). Extending the entity-relationship model to capture dynamic behaviour. European Journal Information Systems 1 (2) pp. 95-106.

Lee, S.Y. et al. (1999). *Designing good semi-structured database*. ER'99 Entity Relationship Approach, North-Holland, Amsterdam, 1999, pp.131-145.

Mannila, M. & Räihä, K. (1992). The design of Relational Databases. Ed. Addison – Wesley, 1992.

Martin, J., (1990). Information Engineering, Book II. Planning and Analysis. Prentice Hall, 1990, Ch.9.

Martínez, P. et al. (1999). *Profundizando en la semántica de las cardinalidades en el Modelo E/R Extendido.* IV Jornadas de Ingeniería del Software y Bases de Datos. Cacéres (España).

Martínez, P., et al. (2000). *Data Conceptual Modelling through Natural Language: Identification and Validation of Relationship Cardinalities.* Proceedings of the 2000 Information Resources Management Association International Conference. Idea Group Publishing, 2000.

McAllister, A. (1998). Complete Rules for n-ary Relationship Cardinality Constraints. Data & Knowledge Engineering 27, 255-288.

McGee, W. C. (1977): The Information Management System IMS/VS Part I: General Structure and Operation. IBM Systems Journal 16(2): 84-95 (1977).

Melton, J. & Simon, A.R (2002). *Basic SQL: 1999 : understanding relational language components.* Morgan Kaufmann, San Francisco.

Mich,L., (1996). NL-OOPS: from Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA. *Natural Language Engineering*, 2 (2), 1996.

Nijssen, G. M. & Halpin, T. A. (1989). Conceptual Schema and Relational Database Design – A Fact Oriented Approach. Prentice-Hall, New York.

Nijssen, G.M., 1977. Current issues in conceptual schema concepts. Proc. IFIP Working Conference on Modelling in Data Base Management Systems, ed. G.M. Nijssen, Nice, France, Holland Publishing, pp. 31-66.

ODMG-93, 1993. The Object Database Standard: ODMG-93. Morgan Kaufmann, Los Altos, CA, 1993.

OMG (2000). *Unified Modelling Language Specification*, Version 1-3. Object Management Group. ACM Computing Surveys 31(1): 63-103.

Peckham, J. & Maryanski, F. (1988). Semantic Data Models. ACM Computing Surveys 20 (3): 153-189, 1988.

Ram, S., (1994). Automated Tools for Database Design: State of the Art. CMI Working Paper, Department of MIS, University of Arizona, 1994.

Ramakrishnan, R. (1997). Database Management Systems. MacGraw-Hill International Editions, 1997.

Rolland, C. & Proix, C. (1992). A Natural Language Approach for Requirements Engineering. *Proceedings CAISE'92*, Manchester, May. 1992, pp. 257-277.

Rumbaugh, J., Blaha, M. & Premerlani, W. J. (1991). *Object Oriented Modelling and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Rumbaugh, J., Jacobson, I. & Booch, G. (1999). The Unified Modeling Language Reference Manual, ed. Addison Wesley, 1999.

Shoval, P. & Shreiber, N. (1993). Database reverse engineering: from the relational to the binary relational model. Data and Knowledge Engineering, vol.10, pp:293-315.

Storey, V.C. & Goldstein, R.C. (1993). Knowledge-based approaches to database design. MIS Quarterly, 17, 1, pp: 25-46.

Tardieu, H., Rochfeld, A. & Coletti, R. (1983). *La Méthode MERISE. Tome 1: Principles et Outils.* Les Editions d'Organisation, Paris.

Teorey, T. J., Yang & D. Fry, J. P. (1986). A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. ACM Computing Survey. Vol 18. No. 2. June 1986.

Teorey, T.J. (1999). *Database Modeling and Design: The Entity-Relationship Approach*. 3rd Edition Morgan Kaufmann, San Mateo, 1999.

Thalheim, B. (1989). The higher - order entity-relationship model and $(DB)^2$. MFDBS 1989, pp:382-397.

Thalheim, B. (1991). Concepts of the database design. Trends in database management systems. Ed. G. Vossen, K.U. Witt, Oldenbourg, Munchen, German, pp:1-48, 1991.

Thalheim, B. (2000). *Entity-relationship modeling : foundations of database technology*. Ed: Springer , 2000.

Thalheim,B. (1992). Foundations of entity-relationship modeling. Annals Mathematics Intelligent 6, pp: 197-256, 1992.

Tsichritzis, D. C. & Lochovsky, F. H., (1982). *Data Models*. Prentice-Hall Ed., Englewood Cliffs, N.J.

Ullman, J. D. & Widom, J. (1997). *A First Course In Database Systems*. Prentice-Hall International Ed., 1997.

Weddell, G.E., (1992). Reasoning about functional dependencies generalized for semantic data models. ACM Transactions Database Systems 17, pp: 32-64, 1992.