UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: CasER 2.0. Herramienta para el diseño de Bases de Datos.
Autores: Durán, Alejandra; Rius, María Florencia
Director: Thomas, Pablo
Codirector: Bertone, Rodolfo
Carrera: Licenciatura en Sistemas

Resumen

El diseño y construcción de un Modelo/Esquema de Datos es una actividad compleja cuya habilidad se adquiere con experiencia y práctica. Dicho diseño se compone de tres etapas claramente definidas: Diseño Conceptual, Diseño Lógico y Diseño Físico.
En la asignatura Introducción a las Bases de Datos de la Facultad de Informática, el diseño de Bases de Datos es un tema central que los alumnos deben aprender. Hasta el año 2008, la ejercitación práctica de este tema era realizada con papel y lápiz, con las limitaciones propias de dichos elementos. A raíz de esto, se generó una herramienta de software ad-hoc denominada CasER, con la cual se puede realizar la primera fase del diseño de Base de Datos, en forma asistida y semiautomática. Por este motivo, existe la necesidad de ampliar la funcionalidad existente en CasER, incorporando las fases de Diseño Lógico y Diseño Físico.
Esta Tesina propone la generación de la versión 2.0 de CasER, en la cual se incorpora toda la funcionalidad necesaria para cumplir las tres fases de Diseño de Bases de Datos.
Finalmente, a partir de una especificación de requerimientos, con CasER 2.0 será posible diseñar una Base de Datos en forma asistida. De esta forma podrá simplificar y mejorar notoriamente las tareas que realizan los alumnos.

Palabras Claves

Modelado Conceptual, Modelado Lógico, Modelado Físico, Modelo de Entidades y Relaciones. Diseño Conceptual. Herramienta Educacional.

Trabajos Realizados

Se han estudiado las tres fases del diseño de bases de datos. Se i desarrolló la herramienta CasER 2.0 con toda la funcionalidad necesaria para cumplir las tres fases de Diseño de Bases de Datos.

Conclusiones

En un entorno de modelado de datos, abstraer conceptos de la realidad, representarlos en un modelo gráfico, y luego obtener las tablas que representarán una base de datos, es un desafío.
La versión 2.0 de la herramienta CasER permite completar el proceso de diseño en forma asistida, por ende el diseño conceptual, el diseño lógico y el diseño físico se realiza sin utilizar papel y lápiz y en modo semiautomático, convirtiéndose en un valioso asistente en el proceso de enseñanza – aprendizaje.

Trabajos Futuros

Se pretende utilizar la herramienta para todo el proceso de diseño de BD, y obtener retroalimentación de los alumnos respecto a la usabilidad y las mejoras posibles en ese sentido.
Se prevé además la incorporación de dominios desde el modelo conceptual, lo que permitirá obtener modelos físicos con tipos de datos incluidos.
Finalmente también se pretende disponer de la generación automática de los scripts de creación de la Base de Datos en lenguaje SQL.

Fecha de la presentación: Mayo 2012

AGRADECIMIENTOS

Agradezco a mi familia y amigos por el apoyo incondicional, la comprensión

y el ánimo que me dieron a lo largo de toda la carrera.

Alejandra

Agradezco a mis padres, por ayudarme a crecer, a plantearme objetivos y concretarlos y por estar cerca aunque lejos de casa.

A mis hermanas y a mi novio, por apoyarme siempre y ser mi compañía en toda esta etapa.

A mis compañeros de la facultad, en especial a Ale que estuvo desde los primeros días a mi lado, por hacer de esta etapa la mas linda e inolvidable.

Florencia

Agradecemos a Sole y Ariel, por recibirnos en sus casas y ayudarnos a comprender su trabajo para poder continuarlo.

A Pablo y Rodolfo por apoyarnos, por la predisposición, por ser nuestra principal guía, ayuda y enseñanza durante el todo desarrollo de este este trabajo final.

Alejandra y Florencia

ÍNDICE GENERAL

TABLA DE FIGURAS

CAPITULO 1

1.1 Motivación

El diseño y construcción de un Modelo/Esquema de Datos es una actividad compleja cuya habilidad se adquiere con experiencia y práctica. Dicho diseño se compone de tres etapas claramente definidas: Diseño Conceptual, Diseño Lógico y Diseño Físico.

En la asignatura Introducción a las Bases de Datos de la Facultad de Informática, el diseño de Bases de Datos es un tema central que los alumnos deben aprender. Hasta el año 2008, la ejercitación práctica de este tema era realizada con papel y lápiz, con las limitaciones propias de dichos elementos. A raíz de esto, se generó una herramienta de software ad-hoc denominada CasER, con la cual se puede realizar la primera fase del diseño de Base de Datos, en forma asistida y semiautomática.

Desde el año 2009 CasER versión 1.0 [17] es utilizada exitosamente por la asignatura antes mencionada; no obstante, con la limitación de suplir sólo la primera fase de Diseño de Bases de Datos. Por este motivo, existe la necesidad de ampliar la funcionalidad existente en CasER, incorporando las fases de Diseño Lógico y Diseño Físico.

Esta Tesina propone la generación de la versión 2.0 de CasER, en la cual se incorpora toda la funcionalidad necesaria para cumplir las tres fases de Diseño de Bases de Datos.

Finalmente, a partir de una especificación de requerimientos, con CasER 2.0 será posible diseñar una Base de Datos en forma asistida. De esta forma podrá simplificar y mejorar notoriamente las tareas que realizan los alumnos.

1.2 Modelos De Datos

Los modelos se utilizan en todo tipo de ciencias. Su finalidad es simbolizar una parte del mundo real de forma que sea manipulable más fácilmente. Representan un esquema mental (conceptual) en el que se intenta reproducir las características de una realidad específica. En el caso de los modelos de datos, lo que intentan reproducir es el esquema de soporte de información que se desea almacenar en una Base de Datos (BD).

El diseño de una BD es un proceso complejo que abarca decisiones a distintos niveles. La complejidad de un problema se aborda mejor si se descompone dicho problema en subproblemas y se resuelve cada uno de éstos independientemente, utilizándose métodos y técnicas específicas. Una alternativa de diseño de BD consiste en descomponer esta actividad en tres etapas: diseño conceptual, diseño lógico y diseño físico [1].

La figura 1 representa el proceso de diseño de una BD. Los dos modelos fundamentales de datos son el conceptual y el lógico. Ambos son conceptuales en el sentido de que convierten parámetros del mundo real en abstracciones que permiten entender los datos sin tener en cuenta la representación física de los mismos.

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción de datos, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Los modelos de datos son el instrumento principal para ofrecer dicha abstracción. Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, los datos, las relaciones entre ellos y las restricciones que se deben cumplir.

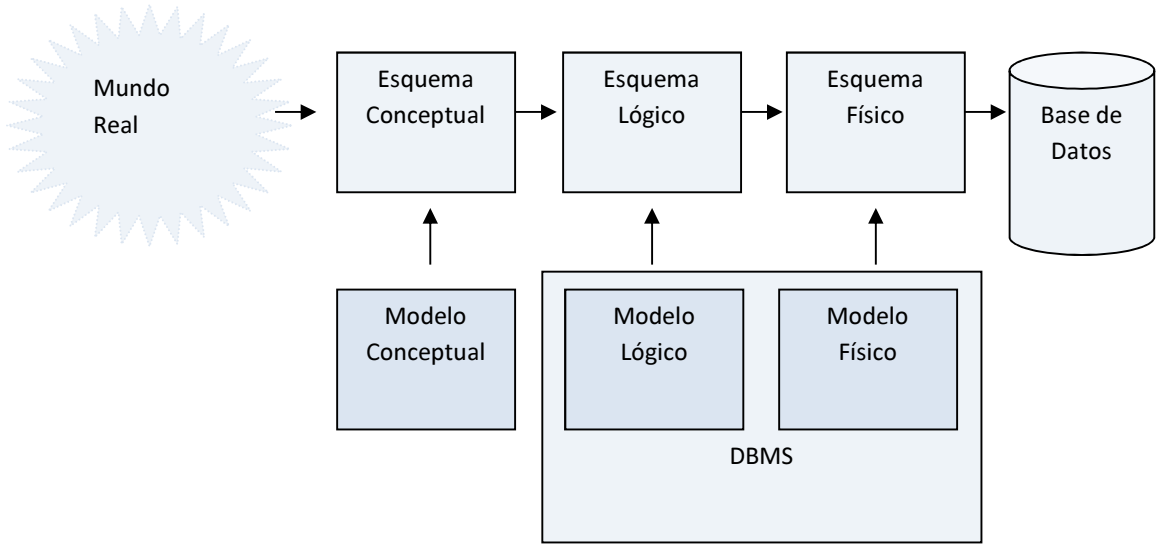

Figura 1. Proceso de Diseño de una BD.

Los modelos de datos se pueden clasificar de acuerdo a los tipos de conceptos que ofrecen para describir la estructura de la base de datos.

Por lo tanto, un Modelo de Datos es una colección de herramientas conceptuales para describir datos, sus relaciones, semántica y restricciones de consistencia. Existen tres niveles de modelado: Conceptual (Modelo Entidad-Relación), Lógico (Modelo Relacional) y Físico (Implementación en el DBMS).

Los modelos de datos de alto nivel, o modelos conceptuales, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos.

Los modelos conceptuales se usan para simbolizar la realidad a un alto nivel de abstracción. Mediante estos modelos se puede construir una descripción de la realidad fácil y simple de entender. El objetivo es la representación gráfica de los datos del mundo real mediante reglas y convenciones, que finalmente se desea almacenar en una base de datos.

Los modelos conceptuales utilizan entidades, atributos y relaciones. Una entidad representa un objeto o concepto del mundo real, por ejemplo, un empleado de la empresa o una oficina. Un atributo representa alguna propiedad de interés de una entidad, por ejemplo, el nombre o el salario del empleado. Una relación describe una interacción entre dos o más entidades, por ejemplo, la relación de trabajo entre un empleado y su oficina.

Los modelos de datos lógicos, tienen conceptos que pueden ser entendidos por los usuarios finales, aunque no están demasiado alejados de la forma en que los datos se organizan físicamente. Los modelos lógicos ocultan algunos detalles sobre cómo se almacenan los datos, pero pueden implementarse de manera directa en una computadora.

Finalmente, los modelos de datos de bajo nivel, o modelos físicos, proporcionan conceptos que describen los detalles sobre cómo se almacenan los datos en una computadora. Los conceptos de los modelos físicos están dirigidos al personal informático, no a los usuarios finales. Los modelos físicos describen: el formato de los datos, los tipos de datos y los métodos de acceso utilizados.

Los Gestores de Bases de Datos (DBMS su acrónimo en inglés) soportan la gestión de modelos lógicos, siendo los tradicionales: Relacional, en Red y Jerárquico. Estos modelos representan los datos valiéndose de estructuras de registros, por lo que también se denominan modelos orientados a registros.

Una nueva familia de modelos lógicos son los modelos orientados a objetos, que por su semántica y forma de expresar, se pueden asimilar de algún modo con los modelos conceptuales.

1.3 Fases del diseño de Bases de Datos

El diseño de Base de Batos abarca el diseño conceptual, diseño lógico y diseño físico, tal como está representado en la figura 2.

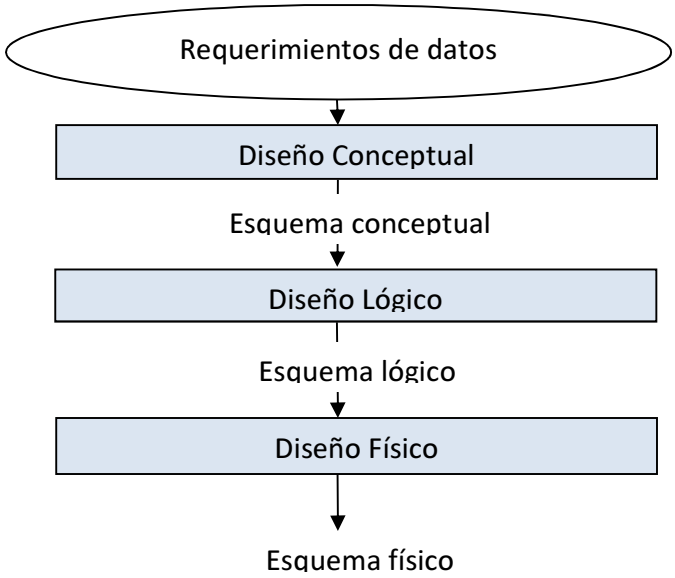

Figura 2. Fases de Diseño de una BD.

El diseño conceptual comienza con el análisis de la especificación de los requerimientos de usuario. Su objetivo consiste en describir el contenido de la información de la base de datos más que las estructuras de almacenamiento. Se determinan las entidades, atributos, posteriormente se interrelacionan las entidades, obteniéndose una representación gráfica que constituye el denominado modelo Entidad Relación (ER).

Por razones prácticas y a partir de la discusión generada en la Cátedra de Introducción a las Bases de Datos, se hace referencia a modelo o esquema (conceptual, lógico y físico), indistintamente.

A través de un modelo conceptual se representa la realidad de un problema específico en un dominio dado, por lo que debe poseer las siguientes cualidades o propiedades básicas:

- Expresividad: capturar y representar de la mejor forma posible la semántica de los datos del problema a resolver.
- Simplicidad: debe ser fácil de entender.
- Minimalidad: cada concepto tiene un significado distinto respecto del resto y no puede expresarse mediante otros conceptos.
- Formalidad: todos los conceptos deben tener una interpretación única, precisa y bien definida.

En la etapa de diseño conceptual la participación activa y comprometida del usuario sobre las decisiones de diseño tiene impacto positivo, mejora la calidad del esquema conceptual, aumenta la probabilidad de que el proyecto obtenga el resultado esperado, y reduce los costos de desarrollo.

La herramienta desarrollada en este trabajo provee la generación asistida de los esquemas conceptual, lógico y físico.

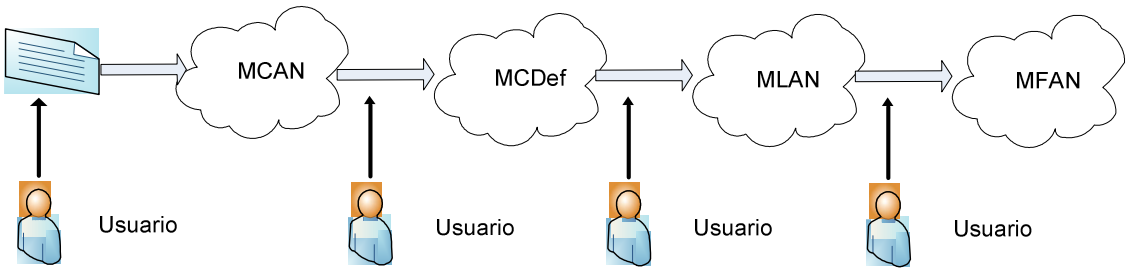

Figura 3. Generación de modelos de datos asistida.

La figura 3 resume el comportamiento propuesto para CasER 2.0. El comienzo de la actividad de diseño se inicia en la especificación general de un problema, se trabaja sobre dicho documento y se genera un Modelo Conceptual de Alto Nivel (MCAN). Luego, a partir de un proceso de

depuración y refinado a cargo del usuario, se obtiene el Modelo Conceptual Definitivo (MCDef). A partir de este modelo definitivo se realiza el proceso de pasaje a Modelo Lógico de Alto nivel (MLAN). Finalmente la última etapa del modelado consiste en realizar el pasaje a Modelo Físico de Alto nivel (MFAN).

CAPITULO 2

2.1 Importancia del Diseño Lógico.

El diseño E-R lógico tiene el propósito de convertir el esquema conceptual en un esquema que pueda ser interpretado por un DBMS. Un modelo lógico representa un esquema equivalente al conceptual pero más eficiente desde el punto de vista operativo para el encargado del diseño de la BD [2].

Al comenzar con el modelado lógico se necesita definir qué tipo de DBMS se utilizará luego para su implantación física. La cadena de pasos de conversión está relacionada con el tipo de DBMS (relacional, orientado a objetos, jerárquico y de red) a utilizar.

En este caso se toma como ejemplo un DBMS relacional, dado que actualmente el modelo relacional es más utilizado por los Sistemas de Información generados para usuarios finales.

2.2 Características del Diseño Lógico

Para realizar el diseño lógico de un modelo de datos se consideran:

- **El esquema conceptual:** resultado de la etapa anterior. El esquema conceptual representa una solución de alto nivel de abstracción respecto del problema original.
- **La descripción del modelo lógico a obtener**: se definen las reglas que se aplicarán en el proceso de conversión. Estas reglas están ligadas al tipo de DBMS que se eligió.
- **Los criterios esperados de rendimiento de la BD**: durante la fase de diseño conceptual, se consideraron los requerimientos del usuario. Pero hay otro tipo de necesidades que no se pueden definir sobre el modelo conceptual. Estas tienen que ver con requerimientos, en general, no funcionales del problema, como por ejemplo la performance de la BD. Así, una regla puede dar alternativas de solución y el analista deberá optar por aquella que permita alcanzar los niveles de rendimiento definidos para el problema.
- **Información de carga de la BD**: Cuando se genera el esquema lógico, el analista deberá observar cada entidad e interrelación definida, y ver la probable evolución de la información contenida en esas estructuras. De esta forma, la decisión final sobre el

esquema de una relación o entidad dependerá del número probable de elementos que la compondrán, con la finalidad de mantener la performance bajo control.

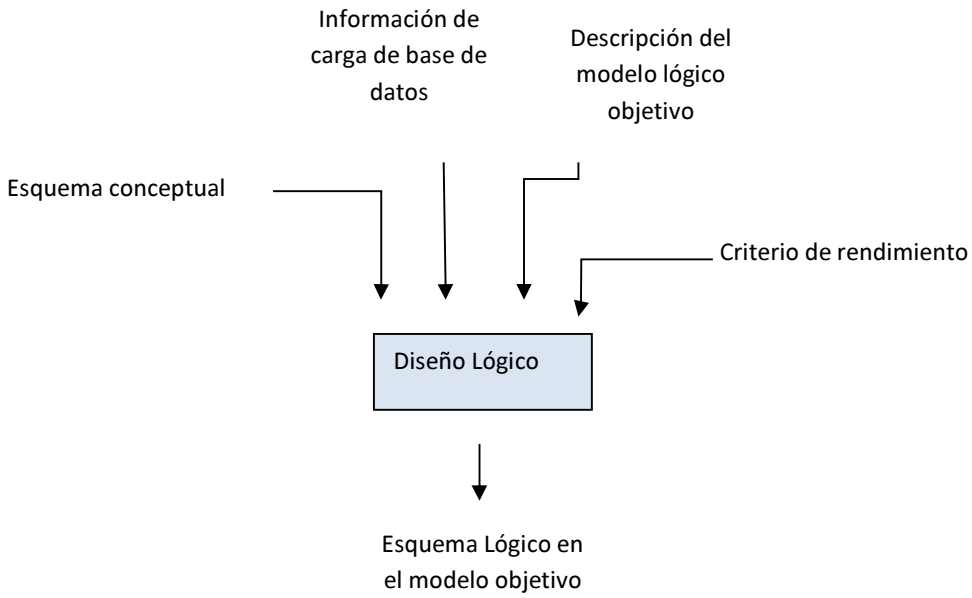

Figura 4. Entradas y Salidas en el proceso de diseño lógico.

La Figura 4 ejemplifica las entradas descriptas del modelo lógico.

Se debe tener en cuenta que en la resolución de un modelo de datos para un problema concreto, el proceso de conversión hacia el esquema lógico, se llevará a cabo una vez afianzados los requerimientos del problema original; es decir, una vez que la especificación de requerimientos esté consensuada con el cliente. Esto puede significar, según el caso, un período de tiempo considerable [1].

2.3 Decisiones sobre el diseño lógico.

Las decisiones sobre el diseño lógico están relacionadas, básicamente, a cuestiones generales de rendimiento y a un conjunto de reglas que actúan sobre características del esquema conceptual que no se presentan en los DBMS relacionales. Por ejemplo, el concepto de herencia no existe en el modelo relacional; esto implica que las jerarquías deberán ser resueltas para adaptarlas a este

contexto. Además, el modelo relacional carece de un dominio que permita definir varios atributos, es decir, no es posible representar atributos compuestos [2].

2.3.1 Decisiones sobre datos derivados

El valor para este tipo de atributo se puede derivar de los valores de uno o varios atributos que no necesariamente deben pertenecer a la misma entidad o relación. Por ejemplo, en la figura 5, el conjunto de entidades cliente que tiene un atributo préstamos representa cuántos préstamos tiene un cliente en el banco. Ese atributo se puede derivar al contar el número de entidades préstamo asociadas con ese cliente. Un atributo es derivado si contiene información que puede obtenerse de otra forma desde el modelo. Por lo general la edad de una persona es un atributo derivado. En este caso con el valor del atributo fecha-de-nacimiento se calcula la edad cuando es necesario.

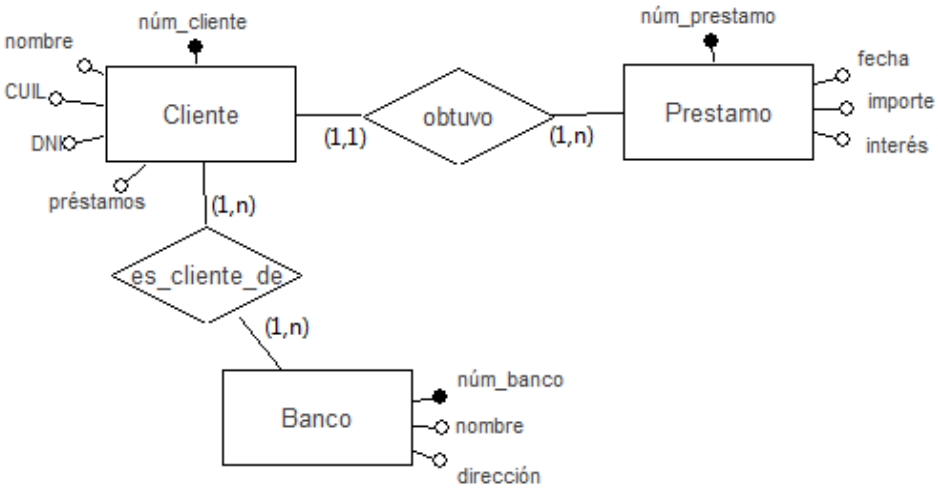

Figura 5. Atributo Derivado

La ventaja que presenta un atributo derivado es permitir obtener determinada información de manera directa, sin necesidad de realizar cálculo alguno. La figura 5 incluye el atributo derivado préstamos de la entidad Clientes. Si esa información es muy solicitada, estará disponible sin la necesidad de contabilizar cuántas veces aparece un cliente en la relación “obtuvo”.

La desventaja de un atributo derivado radica en la necesidad de ser recalculado cada vez que la información que contiene se modifica.

Es decisión del analista mantener o no un atributo derivado, de acuerdo a las ventajas / desventajas a obtener para un problema en particular.

2.3.2 Decisiones sobre ciclos de relaciones.

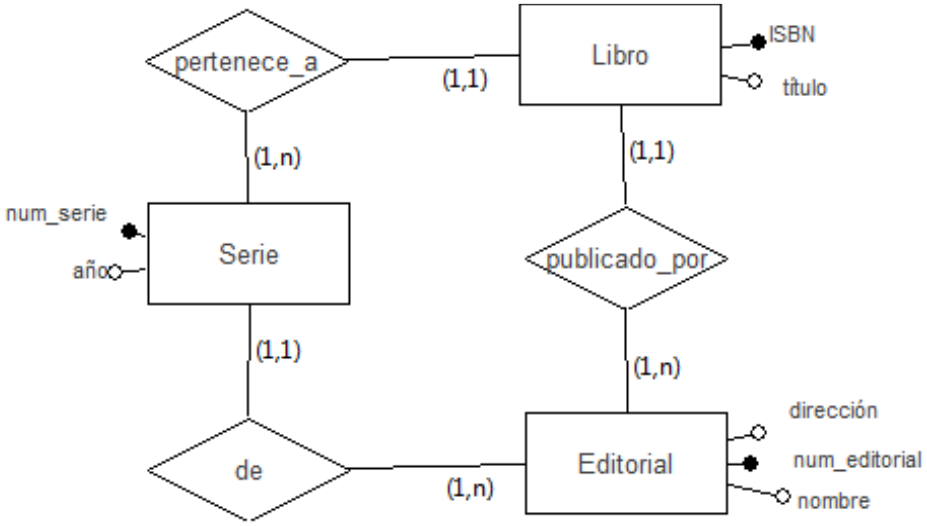

Figura 6. Ciclos de Relaciones. Redundancia.

Es necesario reconocer las relaciones de un esquema que generan repetición innecesaria de información. La figura 6 presenta un ciclo generado entre las entidades Libro, Editorial y Serie. Se repite información dado que la Editorial de la Serie, se puede obtener a través de la entidad Libro. Por lo tanto la relación "de" es redundante. Si se elimina, el modelo quedaría mínimo. Un esquema es mínimo cuando cada concepto se representa una sola vez en el modelo. Como desventaja se utilizará más tiempo de procesamiento en caso que se necesite obtener todas las Series de una Editorial, dado que se deberá obtener los Libros de cada una de las Series para obtener la información de la Editorial.

La decisión es otra vez del analista quien tiene que optar por tener el modelo mínimo, o en caso contrario, que posteriormente el modelo implique menos tiempo de procesamiento en las consultas.

2.3.3 Decisiones sobre atributos polivalentes.

Un atributo se denomina polivalente, cuando puede tomar varios valores diferentes. Por ejemplo los atributos teléfonos o títulos de una entidad Persona son algunos de ellos. En esos casos una persona podría tener varios teléfonos o varios títulos, sin un límite a priori [2].

Los DBMS relacionales no permiten que un atributo contenga valores múltiples determinados dinámicamente. Esto lleva a dos situaciones extremas, para tener suficiente espacio para almacenar los teléfonos, por ejemplo, se puede establecer que el atributo podrá contener hasta 10 valores diferentes. En general, una persona puede tener dos o tres teléfonos; con el consiguiente desperdicio de espacio. Pero peor aún, puede ocurrir que una persona tenga 11 teléfonos y no sea posible registrar a uno de ellos. Ambas situaciones son anómalas, y tienen que ver con definir una estructura estática para almacenar información que, a priori, no se puede determinar la cantidad. En este caso la solución debe implementarse con otro criterio. El criterio establecido es denominado primera forma normal.

Un modelo está en primera forma normal (1FN) si y solo si todos los atributos de entidades o relaciones son atributos simples, es decir, cada atributo puede tener a lo sumo un valor.

Entonces se puede decir que el modelo estará en 1FN sino tiene ningún atributo polivalente.

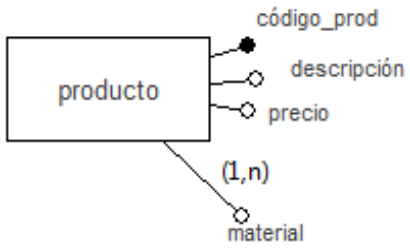

Figura 7. Atributo Polivalente

Si se observa la figura 7, la entidad Producto tiene definido el atributo material como polivalente obligatorio. Para cumplir con la 1FN, la solución en este caso consiste en quitar el atributo polivalente de la entidad Producto, generar una nueva entidad denominada Material, y establecer la relación “producto_material” entre producto y material. La relación “producto_material” se define como una relación muchos a muchos, e indica que un producto puede tener varios

materiales y que un material puede corresponder a varios productos, tal como se presenta en la Figura 8.

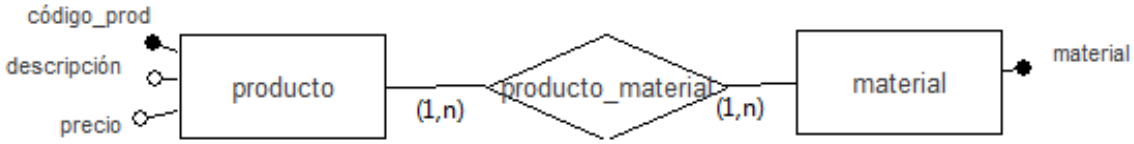

Figura 8. Eliminación de atributo polivalente

El modelo resultante se encuentra en primera forma normal. Sin embargo, no es la única solución posible.

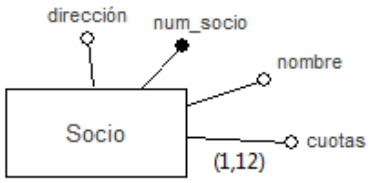

Figura 9. Atributo polivalente de 12 valores posibles

La entidad Socio tiene un atributo denominado *cuotas*, y dicho atributo tiene 12 valores posibles, uno para cada mes del año, por lo que es un atributo polivalente. Nuevamente la solución puede ser generar una nueva entidad y establecer una relación con la entidad existente. Pero a diferencia del caso anterior, en este caso la cantidad exacta de valores posible es conocida. El diseñador de la BD podría optar por definir el atributo con una estructura de 12 elementos sin generar una nueva entidad.

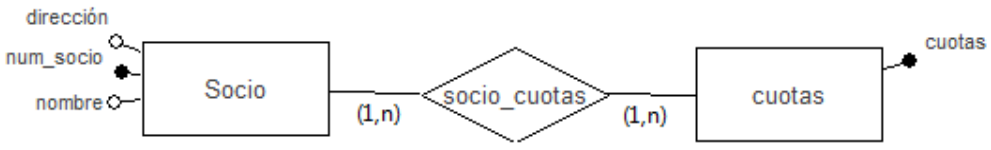

Figura 10. Eliminación de atributo polivalente

La situación de la figura 10 es válida, y se debe tener en cuenta que la forma de resolución planteada por 1FN es mucho más general y efectiva, y en ningún caso presentará inconvenientes.

2.3.4 Decisiones sobre atributos compuestos.

Un atributo compuesto está constituido por varios atributos simples. Los DBMS no soportan esto.

Durante el diseño conceptual los atributos compuestos permiten evitar tomar decisiones rápidamente con poca información sobre el dominio del problema. Por ejemplo, la dirección de una persona se puede definir como la conjunción de calle, número, piso y departamento.

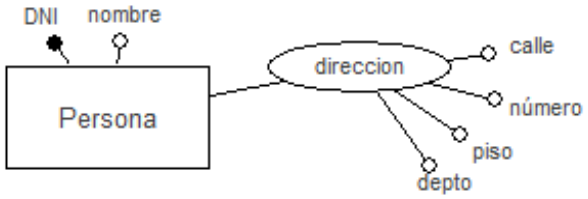

Figura 11. Atributo Compuesto

Posteriormente, cuando se pasa a diseño lógico es necesario decidir cómo convertir un atributo compuesto. La figura 11 presenta el atributo compuesto dirección de la entidad Persona. Existen tres posibilidades para convertir este atributo compuesto en el modelo lógico. La elección dependerá del analista de la BD, teniendo en cuenta criterios de rendimiento y utilización de la BD.

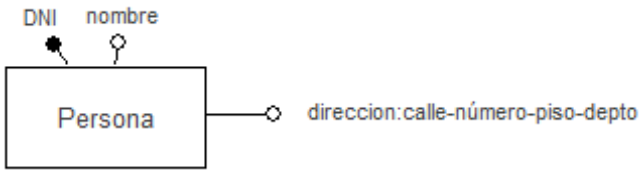

Figura 12. Eliminación de Atributo Compuesto: Concatenación

La primera alternativa consiste en generar un único atributo que resulte de la concatenación de todos los atributos simples que contiene el atributo compuesto. Como se observa en la figura 12, el atributo dirección, se define con una cadena de caracteres, donde el usuario debería ingresar todos los datos de un domicilio: calle + número + piso + departamento. Esta solución es simple y sencilla de implantar, pero se debe considerar que al unir todos los atributos simples que forman el compuesto, se pierde la identidad de cada atributo simple. Luego, establecer cuantas personas viven en un piso determinado no se puede resolver en forma directa ni fácilmente.

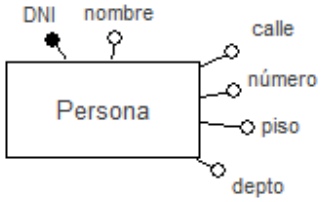

Figura 13. Eliminación de Atributo Compuesto: Atributos simples

La segunda solución, se presenta en la Figura 13. Consiste en definir todos los atributos simples sin un atributo compuesto que los contenga. En este caso, se quita el atributo dirección y en la entidad Persona se definen los atributos simples calle, número, piso y departamento. La cantidad de atributos aumenta pero esta solución permite al usuario definir cada uno de los datos en forma independiente. Generalmente esta solución es la más utilizada.

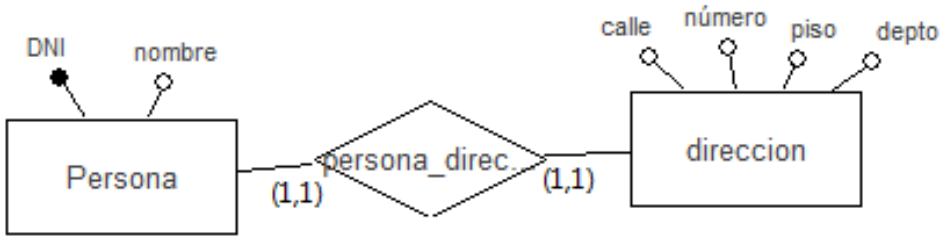

Figura 14. Eliminación de Atributo Compuesto: Genera la nueva entidad Dirección

La tercera opción consiste en generar una nueva entidad, la que representa el atributo compuesto, conformada por cada uno de los atributos simples que contiene. Esta nueva entidad debe estar

relacionada con la entidad a la cual pertenecía el atributo compuesto. Como se puede observar en la Figura 14 esta es la solución más radical, se capta mejor la esencia del atributo compuesto, pero es la opción más compleja.

En general la segunda alternativa es la más utilizada, pues se adapta mejor a la mayoría de las situaciones de la vida real.

2.3.5 Decisiones sobre jerarquías.

El concepto de herencia no está soportado por el modelo relacional (MR), por lo tanto, las jerarquías no pueden permanecer en el esquema lógico. L as decisiones respecto de las jerarquías constituyen el punto más importante al convertir un modelo conceptual en lógico.

Dentro del esquema lógico la información de la jerarquía tiene que estar representada. Por este motivo es necesario encontrar un mecanismo que las incorpore, capte el dominio de conocimiento que representan, bajo un esquema que no administre herencia.

En el pasaje de modelo conceptual a modelo lógico, las decisiones respecto del tratamiento de las jerarquías representan uno de los ítems más importantes. Se definen tres opciones para tratar de eliminar una jerarquía del esquema conceptual [2]. Estas son:

1) Eliminar a las especializaciones (subentidades o entidades hijas) y dejar solo la generalización (entidad padre), la cual incorpora todos los atributos de sus hijos. Cada uno de estos atributos deberá ser opcional (no obligatorio).

2) Eliminar la entidad generalización (padre) y dejar solo las especializaciones. Con esta solución los atributos del padre deberán incluirse en cada uno de los hijos.

3) Dejar todas las entidades de la jerarquía, convirtiéndola en relaciones uno a uno entre el padre y cada uno de los hijos. Esta solución permite que las entidades que conforman la jerarquía, mantengan sus atributos originales. Se genera la relación explícita ES_UN entre padre e hijos.

La primera y la tercera solución son siempre aplicables.

La segunda solución no se puede aplicar en todas los casos, depende de la cobertura de la jerarquía. Solo es viable si la cobertura es total y exclusiva.

Una cobertura parcial, significa que hay componentes contenidos en la entidad generalización que no están cubiertos por las especializaciones. Esto significa que, si se quita la entidad padre, dichos componentes no tendrán representación en el modelo. Esta conversión generaría un modelo lógico que no es equivalente al modelo conceptual, ya que se perdería información. Por lo tanto, la segunda alternativa no es aplicable en el caso de tratarse de cobertura parcial.

Lo mismo ocurre en el caso de la cobertura superpuesta, cuando existe algún elemento de la clase genérica que corresponde a elementos de dos o más clases subconjunto diferente; la segunda solución, nuevamente, no es práctica. Algunos componentes del padre se repiten en varios hijos, esto significa que se deberá repetir información en las subentidades generadas. Esta repetición innecesaria de información puede generar problemas.

Los subconjuntos son jerarquías especiales, con cobertura parcial exclusiva. En este caso, tanto la primera como la tercera alternativa de solución son aplicables, quedando descartada la segunda alternativa. Un ejemplo de subconjuntos se presenta en la Figura 15.

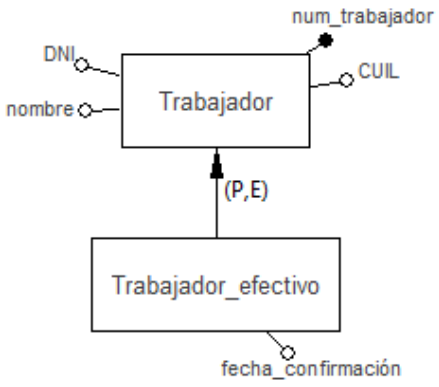

Figura 15. Ejemplo subconjunto.

Ejemplos de eliminación de jerarquías:

Se modelan los Socios del club. Los socios pueden ser de vips o vitalicios. Los socios vip tienen beneficios extras a los socios comunes, pero pagan una cuota vip algo más cara que la general. Y los socios vitalicios del club, son aquellos que tienen 20 años de asociados, ellos tienen registro libre en el club, y un invitado anual, el cual puede usar las instalaciones del club.

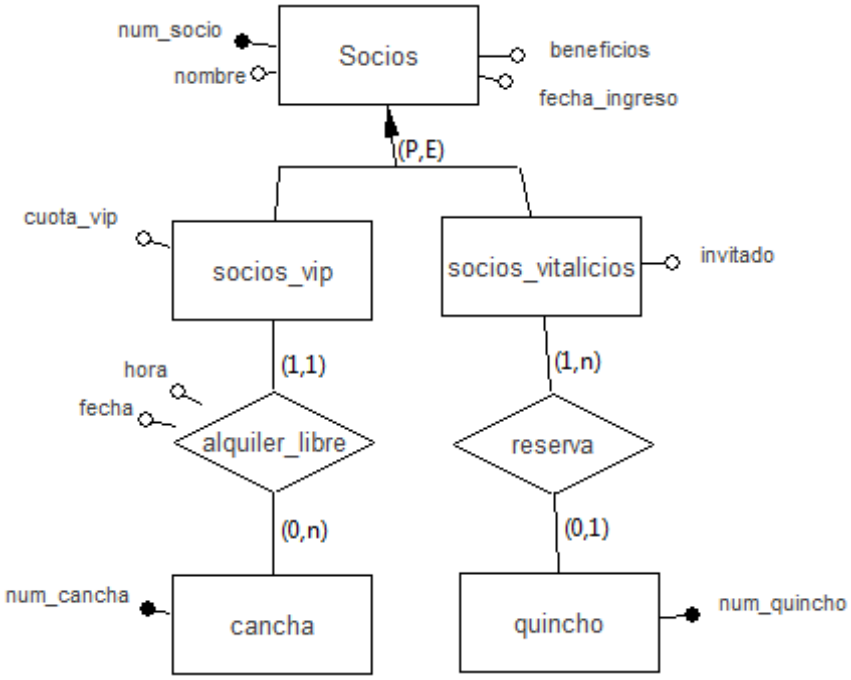

Figura 16. Jerarquía Socios del club

Al aplicar la primera alternativa, se quitan las entidades hijas de la jerarquía y se dejan solamente la entidad padre. En la figura 17 se puede observar que el modelo queda más reducido.

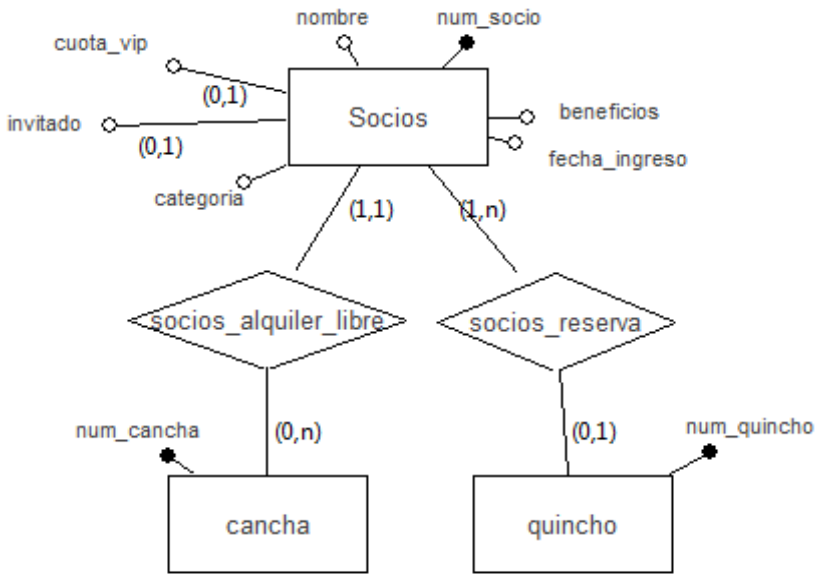

Figura 17. Eliminación de Jerarquías: Se quitan las entidades hijas.

Sobre la entidad padre quedan los atributos que estaban contenidos en las entidades hijas; estos atributos son ahora opcionales. La cardinalidad de Socios en la relación alquiler_libre se convierte en opcional en lugar de obligatoria, dado que un socio vitalicio no tiene alquiler libre.

La segunda alternativa de solución no es viable. Si se quita la entidad socios, aquellos que son socios clásicos no tienen cabida ni como socios vip ni vitalicio.

Por último, la figura 18 presenta la tercera alternativa de solución. Aquí la jerarquía se transforma en una relación explícita Es_Un entre la entidad padre y cada una de las entidades hijas. Es la solución que mejor capta los conceptos del esquema conceptual, pero también es la más extensa.

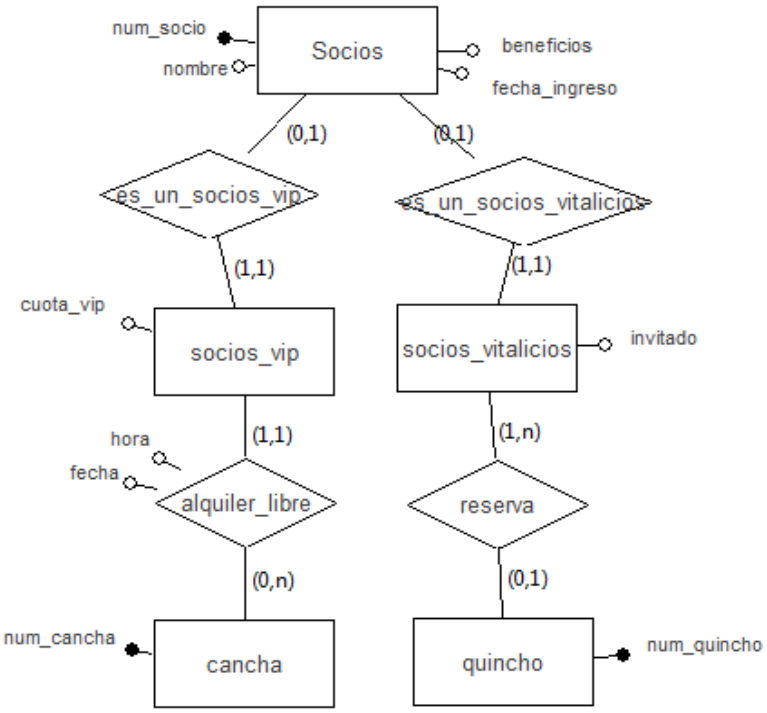

Figura 18. Eliminación de Jerarquías: Se crean relaciones explícitas Es_Un entre padre e hijas

En el caso del club, se tiene como ejemplo la jerarquía de personal del club, las categorías de personal son: deportistas, asistentes, profesores, directores.

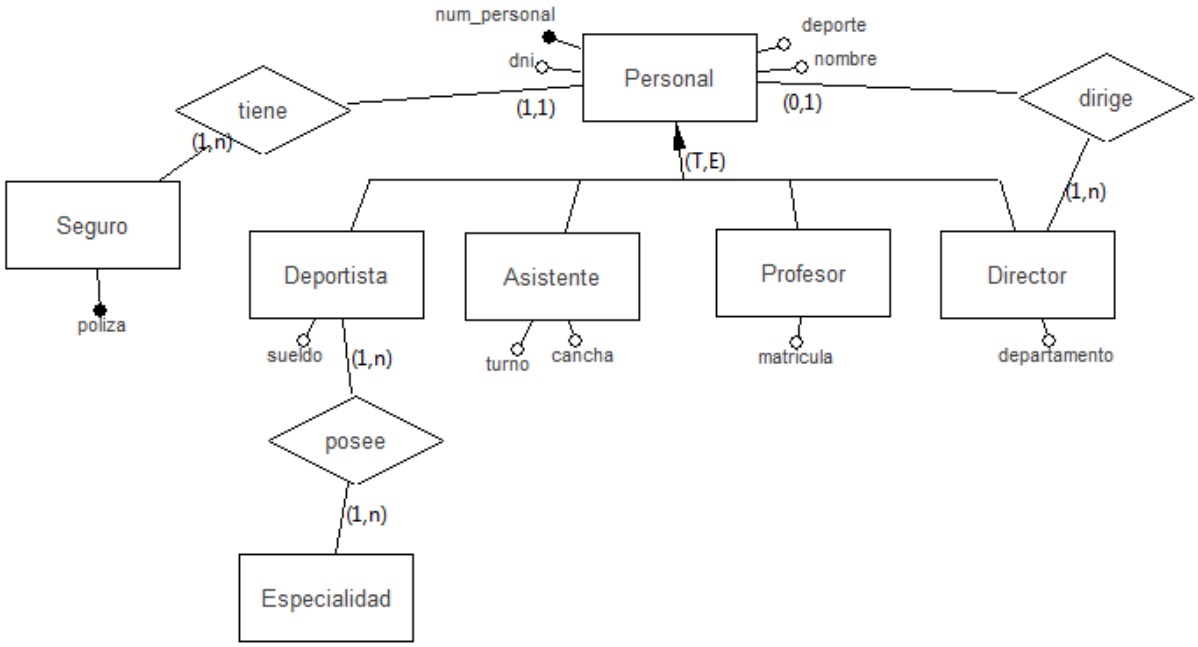

Figura 19. Jerarquía personal del club

Si se aplica la primera opción, las especializaciones se eliminan del modelo y solo se deja la entidad padre (Personal). El modelo se reduce, pero la lectura se hace más complicada, porque aparecen varios atributos no obligatorios, un atributo de categoría para identificar a que especialización pertenece y algunas de las cardinalidades se ven afectadas, como por ejemplo personal_posee (entre personal y especialidad). El resultado es el que muestra la figura 20.

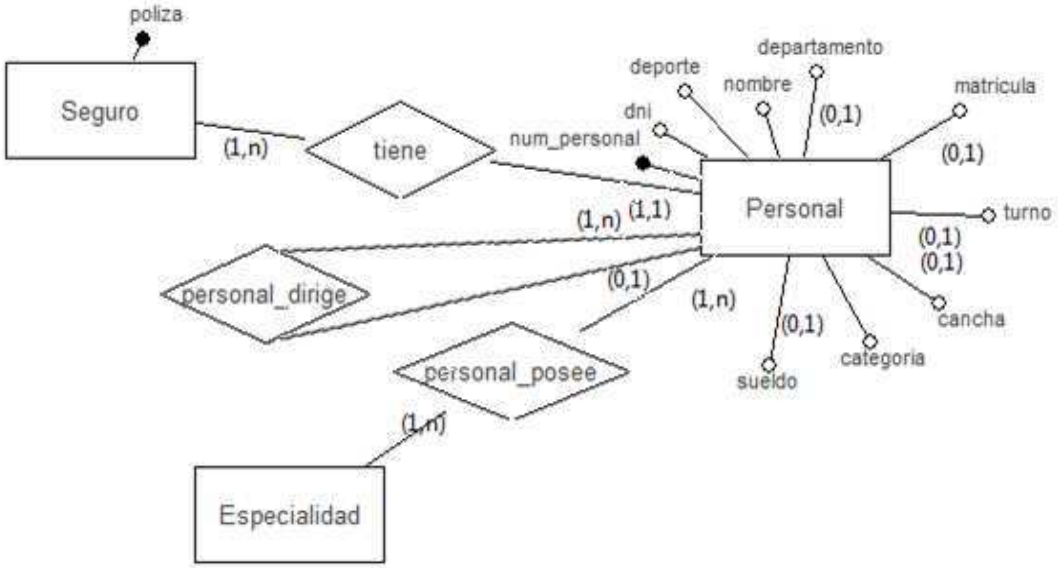

Figura 20. Jerarquía Personal del club: Se quitan las entidades hijas.

Como se trata de una cobertura total, es posible aplicar la segunda solución, con esta opción se quita la generalización (entidad Personal) y se dejan solamente las entidades hijas. Nuevamente el modelo resultante tiene mayor complejidad. La relación que había entre Personal y Seguros se convierte, ahora, en cuatro relaciones. También la relación entre Personal y Director se multiplica por cuatro, ahora con Deportistas, Asistentes, Profesores y con los Directores. La figura 21 muestra el resultado final de la segunda solución.

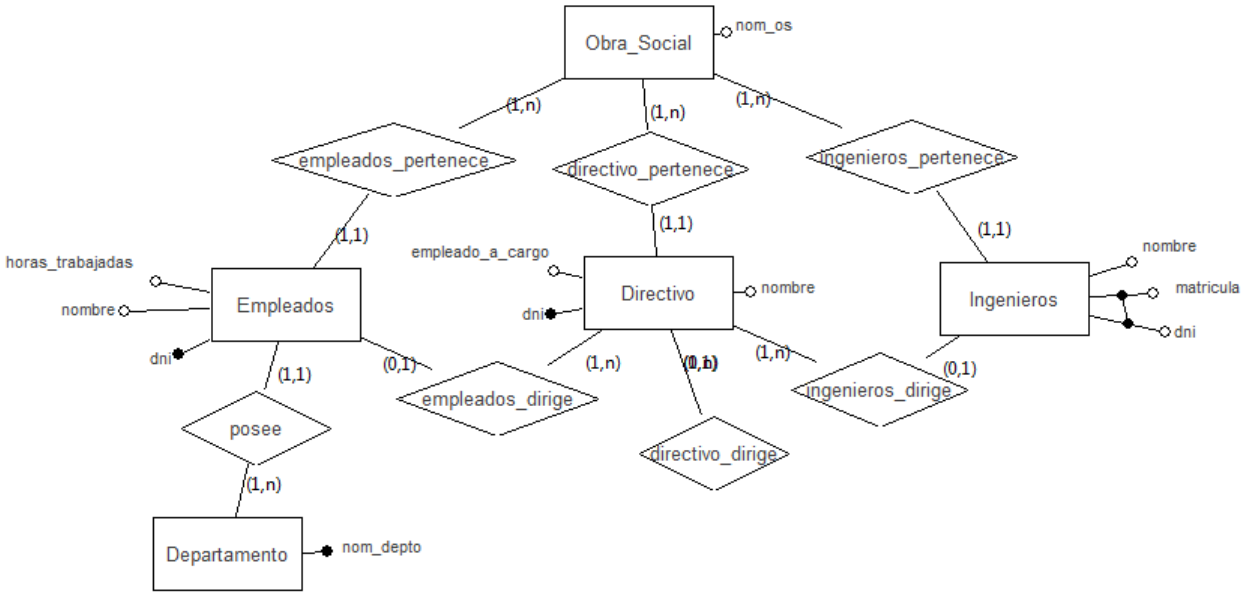

Figura 21. Jerarquía personal del club: Se quita la entidad padre

Finalmente, es la tercera solución la que capta mejor la naturaleza del problema. El número de entidades y relaciones finales no se ve afectado de sobremanera, y la información planteada en el modelo conceptual es referenciada en el modelo lógico de una forma clara y concisa. La figura 22 presenta este caso.

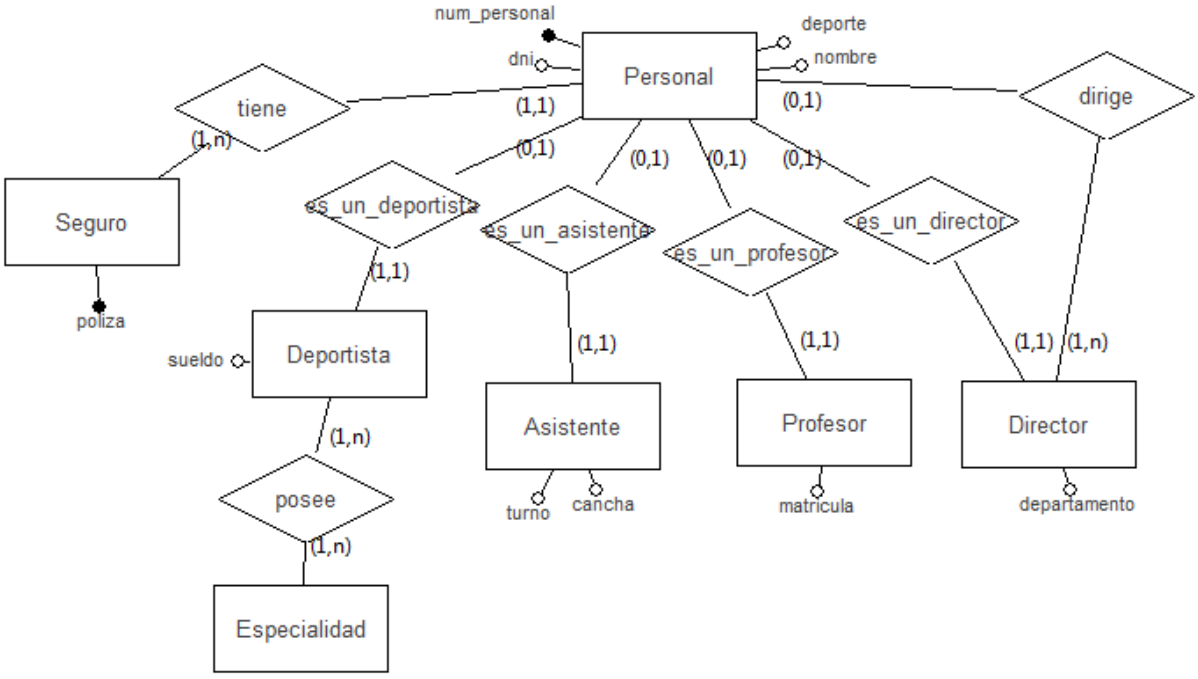

Figura 22. Jerarquía Personal del club: Se crean relaciones explícitas Es_Un entre padre e hijas

Entonces, si la cobertura de la jerarquía que se va a eliminar no es parcial las tres opciones son válidas. En el caso de coberturas parciales no es posible eliminar la entidad padre del problema. La responsabilidad de la elección de la opción para eliminar de la jerarquía es siempre responsabilidad del diseñador de la BD.

Con los ejemplos se ve que la tercera alternativa de solución es la que capta mejor el concepto de la herencia y entonces la más atractiva para aplicar. El problema podría llegar en el futuro, con

baja performance en la utilización de la BBDD, porque esta solución, es la que genera en el esquema mayor número de entidades y relaciones.

CAPITULO 3

3.1 Diseño Físico.

A continuación se describen mecanismos para generar los elementos que componen al modelo relacional (MR) de datos, a partir de las entidades, relaciones, atributos e identificadores. Luego el administrador de la BBDD deberá seleccionar los dominios de acuerdo al DBMS seleccionado.

El modelo relacional fue propuesto en 1970 por Codd, y la popularidad de este modelo creció desde ese momento. El modelo relacional de datos es un modelo simple, potente y formal para representar la realidad.

3.2 Conceptos Básicos del Modelo Relacional.

El modelo relacional MR representa una BD como una colección de tablas, las cuales están conformadas por registros. Cada una de estas tablas constituye el elemento básico del modelo, se denominan relaciones y están integradas por filas (horizontales) y columnas (verticales). Cada columna representa un atributo del registro, y cada fila representa un registro del archivo y se denomina tupla.

Un esquema de base de datos relacional es una colección de definiciones de relaciones. El esquema de cada relación es una agregación de atributos; el conjunto de todos los valores que puede adoptar un atributo en particular se denomina dominio de ese atributo. Como un dominio restringe los valores del atributo, puede considerarse como una restricción. Matemáticamente, otorgar un dominio a un atributo significa que todos los valores de ese atributo deben ser elementos del conjunto especificado. Ejemplos de tipos de dominios son: enteros, cadenas de texto, fecha, entre otros.

El grado de una relación es el número de columnas; la cardinalidad de una relación es el número de tuplas.

En el resto del trabajo se denominan tabla a las relaciones obtenidas en el modelo relacional.

3.3 Eliminación de identificadores externos.

Según [2] el primer paso en la conversión del esquema lógico hacia el esquema físico es la eliminación de los identificadores externos. Para lograr esto, se deberá incorporar dentro de la entidad que contenga identificadores externos, aquellos atributos que permitan la definición del identificador de forma interna a la entidad.

En el ejemplo del club deportivo en la figura 23 se presenta un caso donde la entidad Asistente tiene un identificador externo (num_asistente + num_deporte). Cada Asistente se diferencia del resto a partir de su num_asistente más el Deporte donde trabaja; esto significa que un Asistente puede tener el mismo num_asistente en diferentes Deportes.

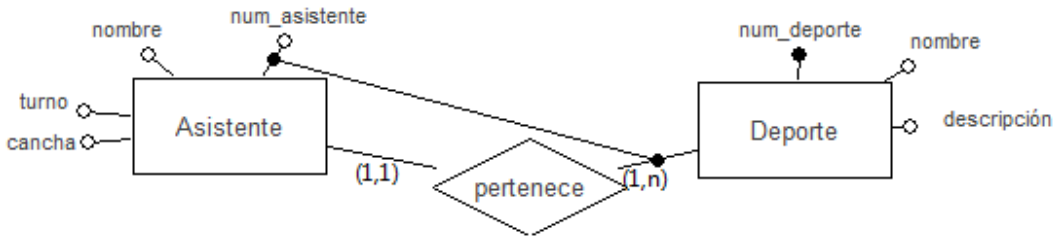

Figura 23. Identificador Externo

Por lo tanto, para realizar la conversión es necesario tomar el identificador de Deporte e incorporarlo como atributo de la entidad Asistente. Esta acción se puede ver en la figura 24.

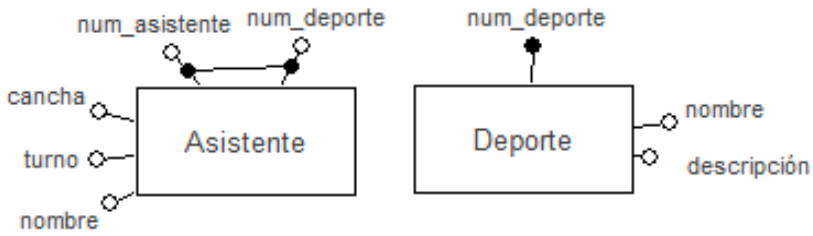

Figura 24. Eliminación de Identificador Externo

Num_deporte es el identificador de la entidad Deporte, que posteriormente podrá ser definido como clave primaria.

3.4 Selección de claves: Primaria, candidata y secundaria.

Uno de los pasos para construir el modelo físico es decidir cuáles identificadores se convierten en clave primaria (CP) o clave candidata (CC), esta es una tarea del diseñador de la BD. Estas claves que son seleccionadas en el proceso del diseño físico serán utilizadas por los usuarios para distinguir una entidad del conjunto.

A partir de los identificadores registrados en las entidades de un modelo relacional se determina el criterio de definición de la clave primaria durante proceso de generación de un modelo físico.

En el caso de una entidad que solo tiene un identificador definido, este es tomado como clave primaria de la tabla. Si la entidad tuviese definido más de un identificador, la selección de la Clave Primaria (CP) se debe realizar con el siguiente criterio:

- Entre un identificador simple y uno compuesto, se debería tomar el identificador simple, dado que es más simple su implantación física sobre un DMBS.
- Si los dos identificadores son simples, se debe optar por aquel de menor tamaño físico. Cuando los dos identificadores son compuestos se debería optar por aquel que tenga menor tamaño en bytes, nuevamente por razones operativas del DBMS.

Estos serian los criterios más adecuados para seleccionar CP, el resto de los identificadores se definen como CC.

Una tabla o archivo tiene asociado solamente una CP, en tanto que puede tener asociado varias CC y/o claves secundarias (CS). A través de la CP es posible el acceso a una única fila o tupla de la tabla o archivo de datos. Es necesario tener precauciones en la elección de la CP, la cual será utilizada para construir el índice primario. Una CP que puede ser utilizada por el usuario directamente, puede sufrir borrados o modificaciones. Esto último impactará en los índices primarios y/o secundarios y puede generar cambios que influirán negativamente en la performance final de la BD. Algunos autores consideran conveniente definir la CP a partir de un nuevo atributo que se incorpora en el modelo físico, con un dominio autoincremental.

Los DBMS tratan una CP autoincremental de una tabla en forma exclusiva y el usuario solo puede consultar el valor de la CP, no pudiéndola crear, modificar o borrar.

Esta elección, combinada con el concepto de integridad referencial que se presenta al final del capítulo, genera una CP que actúa de la forma más eficiente posible, mejorando la performance final de la BD.

3.4.1 El concepto de superclave

Un conjunto de uno o más atributos que permiten identificar de forma única una entidad de un conjunto de entidades es denominado superclave. Visto de esta manera sería igual a una CP o una CC, pero una superclave puede contener más atributos. Dado el ejemplo donde se tiene la siguiente tabla del personal del Club deportivo:

Personal = (num_personal, DNI, nombre, fecha_nacimiento, deporte)

DNI y num_personal son superclaves, porque permiten identificar unívocamente a una persona del conjunto. También pueden ser superclaves el DNI junto al nombre, o num_personal más la fecha_nacimiento.

Si un atributo es superclave, entonces también lo es cualquier superconjunto que incluya a dicho atributo. Una CP o CC es una superclave que no admite a un subconjunto de ella como superclave [2].

3.5 Conversión de entidades.

El proceso de conversión para obtener el esquema físico de una BD comienza con el análisis de las entidades definidas en el modelo lógico. En general, cada una de las entidades definidas se convierte en una tabla del modelo.

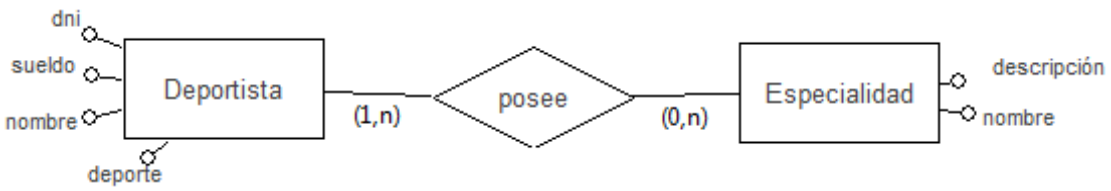

Figura 25. Deportista y Especialidad

Considerando el esquema de la figura 25, la conversión genera dos tablas:

Deportista = (nombre, DNI, sueldo, deporte)

Especialidad = (nombre, descripción)

Si se consideran las definiciones de CP, la conversión genera las siguientes tablas, cada uno con su propio identificador único:

Deportista = (id_deportista, nombre, DNI, sueldo deporte)

Especialidad = (id_especialidad, nombre, descripción)

Donde los atributos id_deportista e id_especialidad, se agregan en este momento y representan para cada uno dominios AutoIncrementales, definiendo la CP de las tablas Deportista y Especialidad respectivamente. Para indicar que son CP en el esquema se muestran subrayados. En este caso también se puede definir una CC, por ejemplo el DNI de la tabla Deportista, la representación debe hacerse efectiva sobre un DBMS; en la conversión al modelo físico las CC no son indicadas.

Existen casos en el que en el proceso de conversión la entidad no se convertirá en tabla. Es el ejemplo de la figura 26 en donde la cardinalidad es (1,1) a (1,1). La descripción indica que el Club deportivo define una comisión directiva.

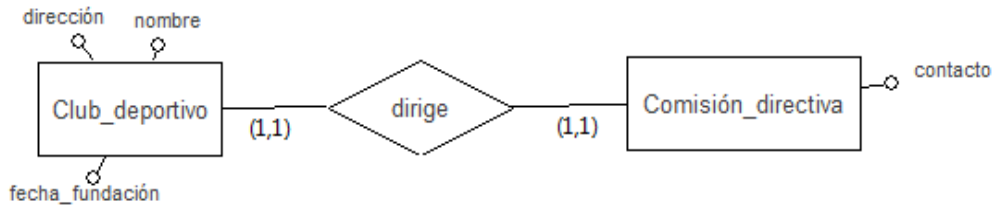

Figura 26. Club y Comisión

Si se continúa con el criterio de conversión anterior de que toda entidad se convierte en tabla, deberían generarse dos tablas, una para Clubes y otra para Comisiones. Salvo que en este caso la existencia del club determina que exista la comisión y también de forma inversa.

La cardinalidad definida entre ambas entidades es uno a uno con participación obligatoria. Por este motivo, la solución más eficiente en el proceso de conversión al esquema físico, consiste en generar una única tabla:

Comision_directiva (id_comision, contacto, nombre_club, dirección_club, fecha_fundacion_club)

O una alternativa sería:

Club_deportivo (id_club, nombre, dirección, fecha_fundación, contacto_comision)

Esta solución es la indicada cuando existe una relación uno a uno con participación obligatoria entre dos entidades.

3.6 Conversión de relaciones.

En el siguiente paso se convierten las relaciones en el proceso de pasaje al modelo físico. Existen tres casos relacionados con las cardinalidades de las relaciones del esquema:

a) N a N

La resolución en este caso es independiente de la cardinalidad mínima definida en la relación, que puede ser: obligatoria u opcional, en cualquier caso la solución es la misma.

En el ejemplo del Club deportivo de la figura 25, las entidades: Deportista y Especialidad se convierten en tablas. Para indicar que un Deportista posee una Especialidad, o que una Especialidad es poseída por un Deportista, se debe definir una nueva tabla para la relación posee.

La relación muchos a muchos se convierte en tabla, conformada con los atributos que definen la CP de cada una de las entidades que relaciona. En este caso, ambos atributos generan la CP de la nueva tabla.

Posee (id_deportista, id_especialidad).

Además podría contener su propio identificador autoincremental, un atributo simple como CP de tabla.

Posee (id_posee, id_deportista, id_especialidad)

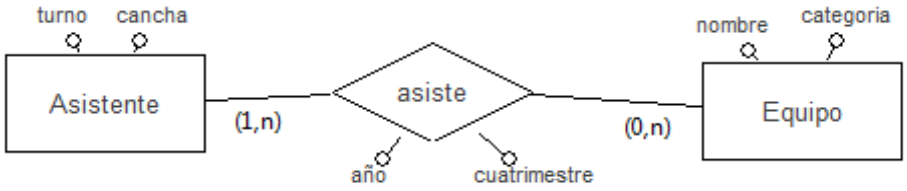

Figura 27. Ejemplo Muchos a muchos Asistente/asiste/Equipo

La figura 27 presenta otro modelo donde se define una relación muchos a muchos, pero en este caso con un conjunto de atributos en la relación a asiste. Entonces se podrían obtener como una solución posible:

Asistente = (id_asistente, turno, cancha)

Equipo = (id_equipo, nombre, categoría)

Asiste = (id_asistente, id_equipo, año, cuatrimestre)

o bien la tabla Asiste podría tener la estructura:

Asiste = (id_asiste, id_asistente, id_equipo, año, cuatrimestre)

Respecto de la solución de la relación, Asiste es una tabla que contiene las CP de Asistente y Equipo en conjunto con los atributos que estaban definidos en la relación. Se puede notar además, que la CP de Asiste está integrada por los atributos id_asistente, id_equipo y año, la asistencia de un mismo Asistente para un equipo puede repetirse en dos años diferentes y, por este motivo, no son suficientes para definir la CP.

La cardinalidad mínima del ejemplo Asistente/asiste/Equipo tiene participación parcial para equipo, sin embargo esta situación no afecta a la resolución del problema.

b) 1 a N

En este caso hay dos alternativas posibles, puede ocurrir que la relación se transforme o no en una tabla. La decisión deberá ser tomada en función de la cardinalidad mínima definida y a las decisiones de diseño del administrador.

Relación 1 a N con participación total

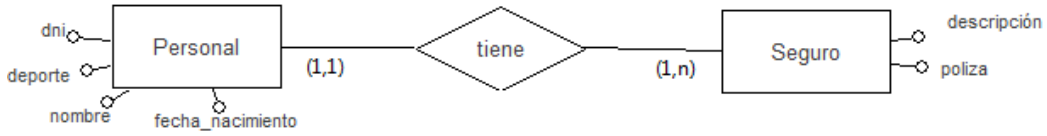

Figura 28. Ejemplo participación total. Personal/tiene/Seguro

La figura 28 presenta la entidad Personal y la entidad Seguro. Entre ambas está definida la relación tiene, con cardinalidad uno a muchos y participación total de ambos lados. Esto significa que cada empleado tiene siempre un Seguro y que cada Seguro tiene, al menos, un empleado que lo elige. La regla de conversión para entidades produce:

Personal = (id_personal, DNI, nombre, fecha_nacimiento, deporte)

Seguro = (id_seguro, descripción, poliza)

Cada seguro puede tener múltiples empleados que lo eligen, sin embargo, cada empleado selecciona solo un seguro. Entonces se puede incorporar al seguro como un atributo más de Personal, y se establece de este modo el vínculo sin necesidad de generar otra tabla:

Personal = (id_personal, DNI, nombre, fecha_nacimiento, deporte, id_seguro)

El atributo id_seguro, CP en la tabla Seguro, es una clave foránea CF, en Personal. Se denomina clave foránea a un atributo o grupo de atributos de una tabla (en este caso Personal) referida a un atributo o grupo de atributos que en otra tabla (para el ejemplo es la tabla Seguros) son CP. Sobre estos atributos se establece un concepto que se denomina integridad referencial, el cual garantiza que una entidad siempre se relaciona con otras entidades válidas, es decir, que existen en la base

de datos. Esto implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias o datos perdidos, y sin relaciones mal resueltas.

Por ejemplo: en el futuro cuando se cree un nuevo registro de la tabla Personal, la integridad referencial exige que el atributo id_seguro coincida con el id_seguro de algún registro de la tabla Seguro. En caso contrario, no se permite la operación.

Una clave foránea es además clave secundaria en la tabla donde aparece. En la Tabla Personal, puede existir varios empleados que tengan el mismo seguro definido; por lo tanto el atributo id_seguro puede repetirse para varias tuplas, conformando, por ende, una clave secundaria en Personal.

Relación 1 a N con participación parcial del lado de muchos

El siguiente caso a analizar se presenta en la figura 29. Aquí aparecen las entidades Deportista y Club deportivo, y entre ambas la relación comenzo_en. La cardinalidad definida establece que un Deportista tiene definido siempre un club donde comenzó, pero puede ocurrir que entre la lista de clubes haya alguno en el que no haya comenzado nadie, y otra en la que hayan comenzado muchos deportistas. La cardinalidad definida es uno a muchos, pero con participación parcial u opcional del lado de muchos.

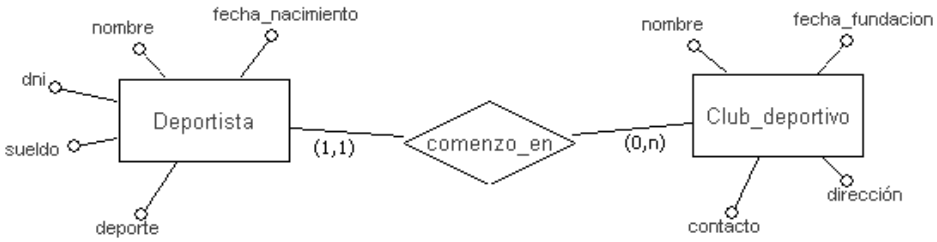

Figura 29. Participación parcial del lado de muchos. Deportista/comenzó en/Club

En la conversión al modelo físico las tablas resultantes serían:

Deportista = (id_deportista, dni, nombre, fecha_nacimiento, sueldo, deporte, id_club)

Club_deportivo = (id_club, nombre, fecha_fundación, dirección, contacto)

El atributo id_club en la tabla Deportista actúa, como clave foránea. Como todos los deportistas que comenzaron en un Club deportivo, la definición de este atributo no presenta ningún inconveniente. En tanto que si en un club no ha comenzado ningún deportista, su CP no figurará en ninguna tupla de la tabla Deportista. Si la conversión se habría realizado al poner como CF el id_deportista en la tabla Club_deportivo, como está definido que existe participación parcial del lado de muchos, es decir, al menos existe un club deportivo en la que no ha comenzado ningún deportista; el atributo id_deportista para la tupla de es Club deportivo es nulo. No es conveniente incluir la clave foránea en la tabla que posee participación parcial, dado que existirán tuplas con valores nulos para ese atributo.

Relación 1 a N con participación parcial del lado de 1

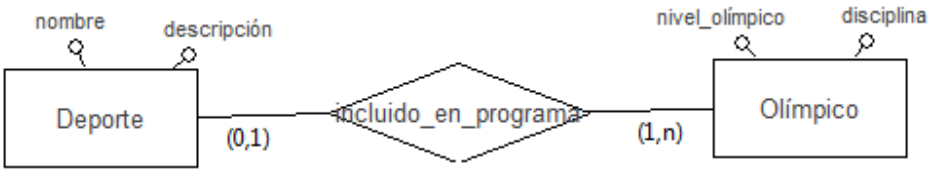

Figura 30. Participación parcial del lado de uno. Deporte/incluido en programa/Olímpico

En este caso la participación es total del lado de N, pero parcial del lado de 1. La figura 30 presenta un ejemplo de esta situación. Cada deporte, es o no es incluido en el programa de los juegos olímpicos, por lo tanto para el modelo de datos es importante reflejar esta situación.. Entonces, la cardinalidad del lado de uno es parcial u opcional.

La resolución del ejemplo es la siguiente:

Deporte = (id_deporte, nombre, descripción, id_olímpico)

Olímpico = (id_olímpico, nivel, disciplina)

En este caso, la solución planteada define el atributo id_olímpico dentro de la tabla Deporte, pero este atributo debe permitir valores nulos, para aquellas deportes que no sean incluidos en el programa de los juegos olímpicos. Igualmente el atributo id_olímpico constituye una clave foránea.

Como no se recomienda el uso de atributos nulos, se debe plantear una alternativa de solución que consiste en mantener las tablas Deporte y Olímpico, pero sin agregar el atributo id_olímpico a la tabla Deporte. Así se debe definir una nueva tabla que contendrá la CP de Deporte y Olímpico, tal como se resuelve la cardinalidad muchos a muchos. La solución sería:

Deporte = (id_deporte, nombre, descripción,)

Olímpico = (id_olímpico, nivel, disciplina)

Incluido_en_programa = (id_deporte, id_olímpico)

Un registro o tupla en la tabla Incluido_en_programa indica que un deporte es olímpico, y se evita de este modo atributos nulos.

La conclusión que se puede emitir en este punto es que la primera solución analizada genera una tabla menos, la información queda resumida y por lo tanto operar con ella es más sencillo. No obstante, se administran atributos con posibles valores nulos, y la situación es no deseada. Por lo tanto se opta por la segunda alternativa.

Relación con cobertura parcial de ambos lados

Las opciones para este caso son las mismas que para el ejemplo anterior, sucede cuando la cardinalidad es parcial de ambos lados. Hay dos soluciones factibles, aquella que genera atributos con valores nulos (no recomendable), y la variante que lo evita.

Relación uno a uno

En el caso de cardinalidad uno a uno con participación total de ambos lados, se debe generar una sola tabla que contenga los atributos de ambas entidades. Pero existen otros dos casos posibles, con participación parcial de un lado o de ambos.

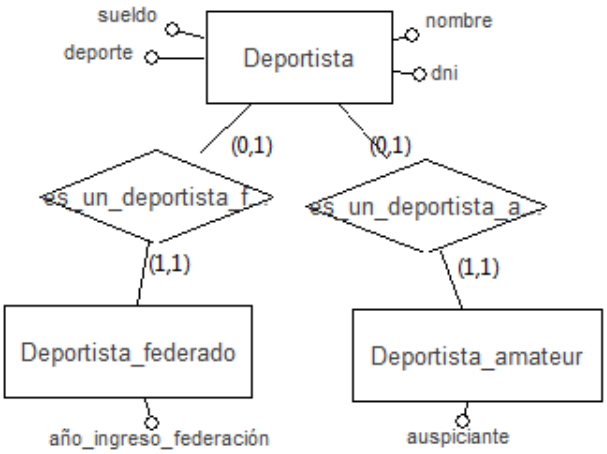

Figura 31. Cardinalidad uno a uno con participación parcial de un solo lado.

La figura 31 ejemplifica la participación parcial de un lado. En ella están definidas las entidades Deportista, Deportista federado y Deportista amateur. Sea federado o amateur los dos son un deportista, y cada deportista puede ser de uno de estos dos casos. La solución para convertir al esquema físico comienza con la definición de la tabla Deportista:

Deportista (id_deportista, nombre, dni, sueldo, deporte)

Luego se definen tres tablas, pero con una particularidad, el atributo id_deportista, CP de la tabla Deportista, también será la CP de cada una de las tres tablas definidas:

Deportista_federado (id_deportista, año_ingreso_federación)

Deportista_amateur (id_deportista, auspiciante)

Tanto un deportista federado o un amateur tiene las características de Deportista del club.

El caso de cardinalidad parcial de ambos lados no es una situación muy común. Sin embargo la solución aconsejable en este caso es generar una tabla por cada entidad y otra tabla que involucre la relación. Esta última tabla deberá incluir las CP de las entidades que relaciona.

Dos casos particulares

Estos son los casos de las relaciones recursivas y las relaciones n-arias, particularmente las ternarias.

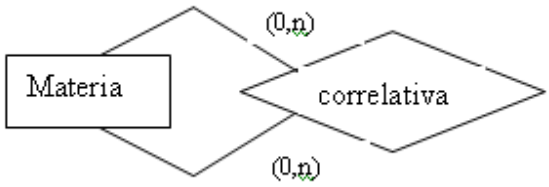

Figura 32.Relación recursiva

En la figura 32 se analiza el caso de la entidad Materias que presenta una relación recursiva que define las correlativas. Suponer que se genera la tabla MATERIAS, con los siguientes atributos:

MATERIAS = (idmateria, nombre, año, duración)

La cardinalidad de la relación Correlativas indica que una materia puede tener varias correlativas o ser correlativa de varias, genera un caso de muchos a muchos. Entonces la solución será generar una nueva tabla:

CORRELATIVAS = (idmateria_original, idmateria_correlativa)

Se debe notar que como la relación es recursiva, al armar la tabla CORRELATIVAS se toma la única CP disponible y se la replica.

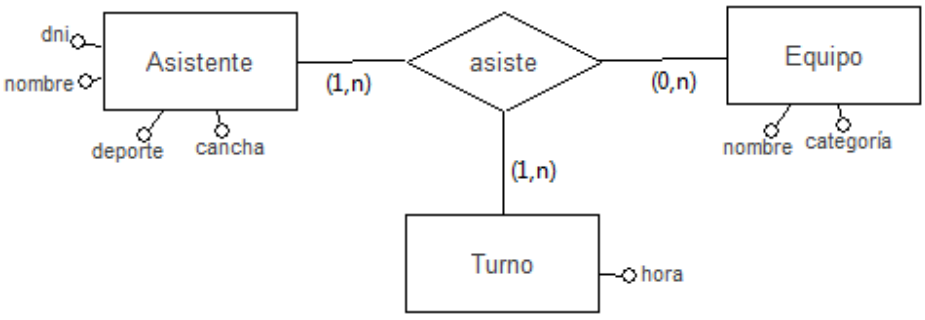

Figura 33. Relación ternaria.

La figura 33 presenta una relación ternaria asiste, entre Asistente, Equipo y Turno. Como la relación es muchos a muchos, la conversión se realiza del siguiente modo:

Asistente = (id_asistente, cancha)

Equipo = (id_equipo, nombre, categoría)

Turno = (id_turno, hora)

Asiste = (id_asistente, id_equipo, id_turno)

La solución depende, nuevamente, de la cardinalidad definida en la relación.

3.7 Integridad Referencial

La integridad referencial es una propiedad deseable de las BD relacionales. Esta propiedad asegura que un valor que aparece para un atributo en una tabla, aparezca además en otra tabla para el mismo atributo.

La integridad referencial plantea restricciones entre tablas, y sirve para mantener la consistencia entre las tuplas de dichas tablas.

Suponer que se generan las tablas:

FACTURAS = (idfactura, fecha, monto, idcliente)

CLIENTES = (idcliente, nombre, dirección)

El atributo idcliente en la tabla FACTURAS es una clave foránea. Esta clave foránea permite establecer integridad referencial entre la tabla FACTURAS y la tabla CLIENTES. Se debe notar que el atributo idcliente es CP de la tabla CLIENTES.

Para que exista integridad referencial entre dos tablas, necesariamente debe existir un atributo común. En general, el atributo común es clave foránea en una tabla y en la otra tabla es CP, aunque pueden presentarse excepciones a esta regla.

No es condición necesaria que entre dos tablas que tengan un atributo común esté definida integridad referencial (IR). Puede ocurrir que el diseñador de la BD no considere oportuno la

definición de la IR entre dos tablas del modelo. No obstante, en general, es deseable definir la IR, a fin de permitir que el DMBS controle determinadas situaciones.

En el ejemplo anterior, se debe definir la IR entre las tablas FACTURAS y CLIENTES. La IR puede plantearse para dos casos posibles: intento de borrado o intento de modificación.

Cada DBMS tiene escenarios de definición de IR diferentes, pero en general cuando se la define se puede optar por estas situaciones:

Restringir la operación: es decir, si se intenta borrar o modificar una tupla que tiene integridad referencial con otra, la operación se restringe, y no se puede llevar a cabo. En el ejemplo, si se intenta borrar una tupla de la tabla CLIENTES que tiene ligada al menos una tupla en la tabla FACTURAS, la operación no es permitida. Para poder borrar un cliente, éste no debe poseer facturas, o las facturas deben ser borradas previamente. Se debería proceder de la misma forma si se intentara modificar la CP sobre la tabla CLIENTES.

En el caso de realizar la operación "en cascada" si se intenta borrar o modificar una tupla sobre la tabla donde está definida la CP de la IR, la operación se realiza en cadena sobre todas las tuplas de la tabla que tiene definida la clave foránea (CF). Para el ejemplo anterior, eliminar un cliente significaría eliminar en la misma operación todas las facturas de dicho cliente. En caso de modificar la CP de un cliente, se realizaría la modificación de las CF en las tuplas de la tabla FACTURAS.

En el caso de establecer la CF en nulo: si se borra o modifica el valor del atributo que es CP, sobre la CF se establece valor nulo. Esta opción no es muy utilizada y no está presente en todos los DBMS, porque genera, por ejemplo, inconsistencia.

También existe la opción de no hacer nada. En este caso se le indica al DBMS que no es necesario controlar la IR. Esta opción es equivalente a no definir restricciones de Integridad Referencial.

CAPITULO 4

4.1 CasER - Versión 2.0 - Una Herramienta de Diseño de Base de Datos

Hasta ahora se ha presentado el marco teórico con los conceptos que se tomaron como base para desarrollar las funcionalidades de CasER 2.0.

La herramienta CasER versión 1.0 [17] se utiliza para modelado conceptual de bases de datos. Ahora, bajo la Versión 2.0 la herramienta extendió sus funcionalidades para modelado lógico y físico de alto nivel.

Su característica principal es la de asistir en la creación de un esquema conceptual de alto nivel, y asistir en la generación del esquema lógico y físico. De este modo se abarca todo el proceso de diseño de una BD

El modelado parte de la especificación detallada de un problema (especificación de requerimientos sobre la información a gestionar) a ser resuelto por un Sistema de Software, se asiste en la creación de un Modelo Conceptual de Alto Nivel (MCAN) y, luego de sucesivos refinamientos por parte del usuario, se obtiene un Modelo (o esquema) Conceptual Definitivo (MCDef). El paso siguiente consiste en derivar el modelo lógico (MLAN) donde la herramienta aplica todas las reglas de conversión de modelo Conceptual a modelo Lógico resumido en capítulos anteriores. Finalmente, se obtiene al esquema físico de alto nivel (MFAN).

La finalidad de CasER 2.0 consiste en asistir y agilizar el proceso de modelado de datos de alto nivel. Básicamente, la construcción de los modelos se facilita notablemente con la utilización de la herramienta. Se presentan opciones para la toma de decisiones de diseño en la resolución de esquemas lógicos.

La herramienta CasER facilita la resolución del problema permitiendo la rápida generación de gráficos de entidades, relaciones, atributos, identificadores, etc.

En sus orígenes CasER [17] [18] era reducido a una parte del modelado, por ende era necesario continuar con los esquemas lógicos y físicos de forma manual, es decir construirlos del modo tradicional utilizando papel y lápiz.

Con la herramienta versión 2.0 es posible crear esquemas más legibles en todo el proceso de diseño, con las pautas de dibujo de entidades, relaciones, atributos, identificadores, cardinalidades, entre otros, de acuerdo a los conceptos de [1] y [2].

Existen diferentes herramientas que permiten crear modelos de datos, entre otras, ERWIN [14], POWERDESIGNER [15], Workbench [16]. Sin embargo, ninguna de ellas permite la creación de un modelo conceptual, lógico y físico con los conceptos referidos en [1] y [2].

En CasER 1.0, al existir la posibilidad de relacionar los elementos seleccionados de la especificación con los elementos que componen el diagrama o esquema, se logra alinear la especificación del problema con el modelo conceptual de datos que lo soporta. Con la versión 2.0, esta característica se ve enriquecida con la posibilidad de poder continuar desde el modelo conceptual con la creación de esquemas lógicos y físicos con solo tomar algunas decisiones, característica ausente en los productos comerciales actuales. Esto convierte a CasER en una herramienta muy útil para el proceso de enseñanza y aprendizaje del modelado de bases de datos.

Por lo tanto CasER 2.0 ofrece la posibilidad de crear modelos conceptuales, lógicos y físicos, donde se reflejan conceptos teóricos dados en [1] [2] [4] [5] [6] [7] y [10].

4.2 Presentación de CasER 2.0.

En la figura 34 se señalan las partes que componen las áreas con las que el usuario puede interactuar. El área A corresponde a la especificación de requerimientos del problema. El área B contiene el árbol que representa la jerarquía de los objetos marcados en A y graficados en área C. El área C corresponde al diagrama de modelo conceptual. Finalmente el área D incluye el diagrama del modelo lógico, y el área E presenta el diagrama de tablas del modelo físico.

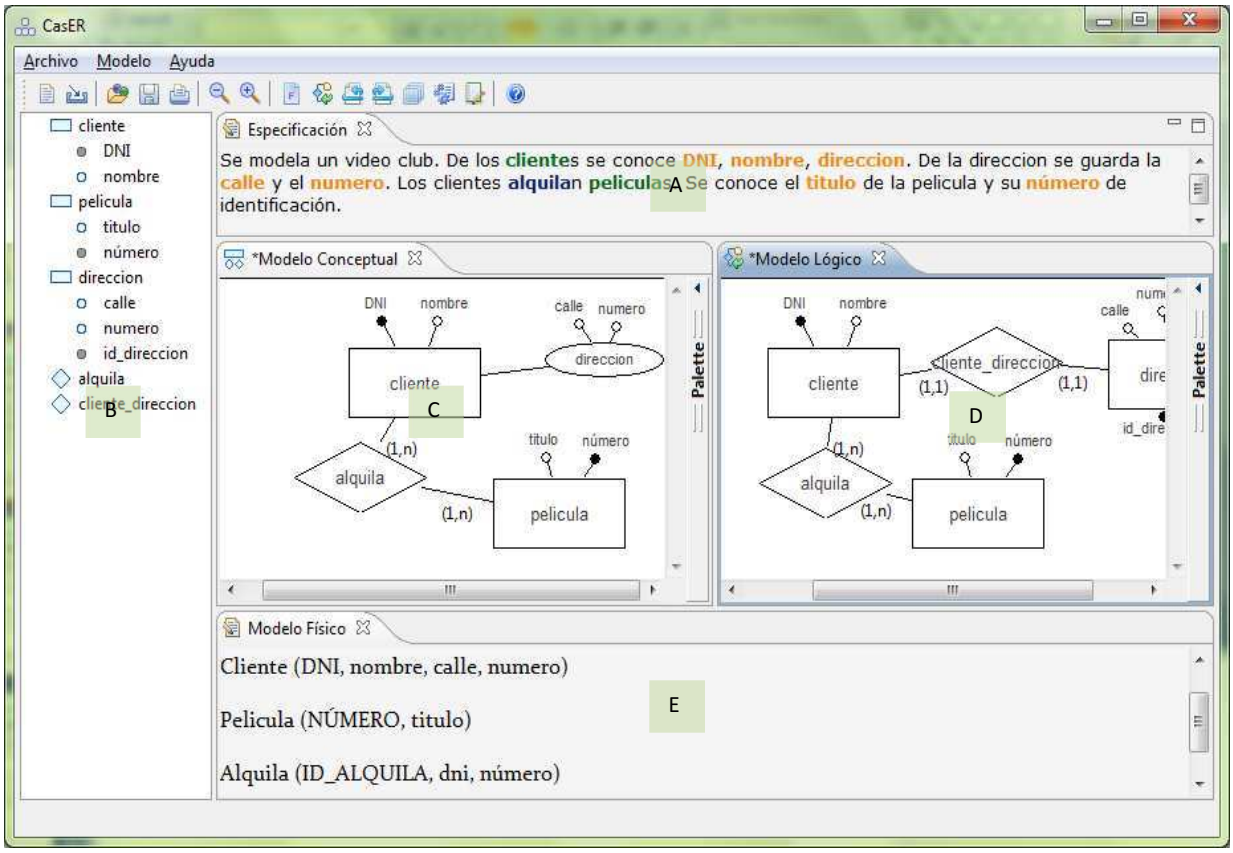

Figura 34. CasER. Pantalla inicio.

El área A corresponde a un editor de texto simple que contiene la especificación de requerimientos del problema a resolver. La funcionalidad esencial de este editor radica en la posibilidad de realizar marcas específicas sobre distintas palabras o frases que tengan impacto directo sobre los datos del problema. De esta manera, las marcas identifican dentro del texto potenciales entidades, relaciones y atributos, las cuales constituyen los tres elementos básicos del modelado conceptual.

Es posible importar un archivo de texto con la especificación de requerimientos, o generarla directamente. Sobre ella se realizan marcas y se observan los conceptos resaltados con determinados colores, según se represente una entidad, una relación o un atributo. Caser admite editar el texto en caso de que no esté completa la especificación de requerimientos, aún después de que dicho el texto posea marcas en alguna palabra o frase.

Paralelamente a la selección y marca de palabras o frases, se genera automáticamente y en forma incremental el modelo conceptual preliminar. Los elementos obtenidos se reflejan en el área B como representación gráfica del modelo, y en el área C como una jerarquía de objetos. Todos los objetos que se agregan desde la especificación, se mantienen sincronizados con los de las restantes áreas.

En el área D se ven reflejados los objetos que quedaran luego de aplicar las reglas de transformación de modelo conceptual al modelo lógico y, por último, en el área E se presentan las tablas del modelo físico resultado de aplicar las reglas de transformación al modelo lógico.

Figura 35. CasER. Barra de herramientas.

La figura 35 muestra la barra de herramientas de CasER 2.0 que permitirá realizar el proceso completo de modelado de datos. La barra de herramientas permite deshacer y rehacer acciones en la conversión de modelos conceptuales a lógicos. También permite visualizar el progreso de la conversión y la secuencia de actividades realizadas. Desde la misma barra se accede el manual de usuario.

En la siguiente sección, previo a realizar una descripción en detalle de cada parte de la herramienta, se detallan los estados por los que atraviesa un proyecto o documento de CasER y la relación entre cada parte de la herramienta según dichos estados. En lo que sigue se utilizarán los términos “editor de texto” y “especificación” indistintamente para referenciar al área A de la figura 34, y “modelo”, “diagrama” o “editor gráfico” en el caso del área C y D.

4.3 Estados de un documento

Un documento que contiene un modelo se puede encontrar en cuatro estados. Los estados posibles son “abierto o no finalizado”, “finalizado o finalizado conceptual”, “finalizado lógico” y “finalizado físico”. Un documento que se encuentra en estado abierto puede pasar al estado finalizado, mientras el cambio inverso de estado no es posible. Lo mismo ocurre para el paso de “finalizado conceptual” a “finalizado lógico” y luego, del estado “finalizado lógico” a “finalizado físico”.

El estado inicial de un documento es abierto o no finalizado, lo que significa que la interacción entre el usuario y la herramienta estará reducida en marcar elementos en la especificación para agregarlos al modelo conceptual. En el gráfico resultante es posible ordenar los objetos a medida que son agregados y, además, ver o editar algunas de sus propiedades, o eliminarlos. Si algún elemento es eliminado, la palabra asociada en la especificación se desmarca en forma automática.

Mientras el estado del documento sea no finalizado, el usuario sólo tendrá la opción de agregar elementos al modelo conceptual a partir de marcas en la especificación. Esta restricción aporta robustez y hace más consistente a la herramienta, evitándose redundancia. Otra opción es modificar la especificación agregando lo que se crea necesario y/o conveniente, y posteriormente realizar las marcas de palabras y/o frases de modo de que se generen automáticamente los gráficos respectivos.

Existe una restricción de relación unidireccional en la creación de los objetos (desde la especificación hacia el diagrama que representa al modelo).

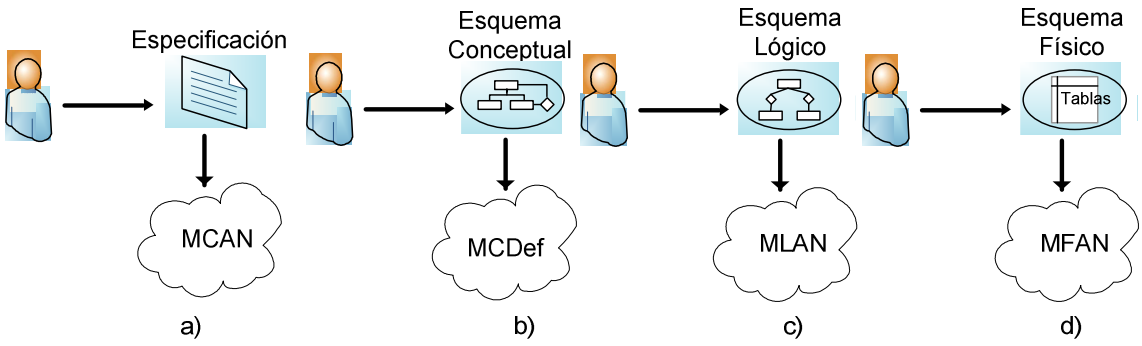

Figura 36. Relación entre el usuario y la herramienta.

Como se puede apreciar, en la figura 36 a) el trabajo del usuario se concentra principalmente en la especificación hasta obtener el Modelo Conceptual de Alto nivel (MCAN). Esto es consecuencia de lo expuesto anteriormente, dado que el documento aún no se encuentra finalizado.

4.4 Documento finalizado

El segundo estado posible para un documento es finalizado. En este estado se finalizó la relación existente entre la especificación y el gráfico o el árbol; sin embargo se mantiene la sincronización entre el árbol y el gráfico que representa al modelo.

La especificación a partir de esta etapa no se puede editar ni marcar. Sólo mantiene las marcas que fueron realizadas en el estado anterior del documento (no finalizado).

Nuevos elementos pueden crearse desde la paleta ubicada en el diagrama, pero estos no se verán reflejados en la especificación debido a que no están asociados a ella. En esta instancia, el usuario trabaja directamente con el gráfico para lograr el Modelo Conceptual Definitivo (MCDef) (Figura 36 b).

4.5 Documento de Modelo Lógico

El tercer estado en el que se encuentra un documento es finalizado lógico. Recién cuando un modelo se encuentra en el estado MCDef es posible comenzar el pasaje a modelo lógico, por lo que el botón de pasaje a este modelo se encontrará habilitado en ese momento.

Los tres pasos de conversión de modelo conceptual a lógico son:

1) Eliminación de atributos compuestos (en este paso es posible tomar una opción de tres posibles).
2) Eliminación de atributos polivalentes. Como se creará una nueva entidad por cada atributo polivalente eliminado, en este punto es necesario que el usuario ingrese la cardinalidad mínima.
3) Eliminación de jerarquías, en este caso existen dos o tres posibilidades y el usuario puede optar por una de ellas. El marco teórico fue definido en el capítulo anterior.

Luego de que se ejecuten todas las reglas de pasaje/conversión respectivas, el diagrama pasa al estado Modelo Lógico de Alto Nivel (MLAN) (Figura 36 c). Para mantener alineado los esquemas en este estado, no se puede retroceder y editar el diagrama conceptual, ni el diagrama lógico.

Es importante destacar que al comenzar el pasaje/conversión, hasta completarlo y obtener un documento finalizado lógico, es decir MLAN , es posible deshacer el trabajo realizado y optar por

otras opciones de solución. Esto permite que el pasaje/conversión sea flexible, iterativo, permita correcciones, y que se mejore de este modo el aprendizaje, sin perder robustez.

4.6 Documento de Modelo Físico (MFAN)

A partir de un documento en el estado MLAN es posible pasarlo al cuarto estado y obtener el modelo físico (MFAN). Se habilita el botón respectivo y el primer paso posible elimina los identificadores externos, en caso de que el modelo lógico los tenga.

Luego de la eliminación de los identificadores externos, comienza la generación de tablas. El usuario tiene la posibilidad de elegir entre dos alternativas para realizar la conversión:

a- El usuario participa en la conversión de cada tabla, seleccionando el pasaje a dos o tres tablas según corresponda.
b- El usuario no participa en la conversión y se generan las tablas automáticamente.

Finalmente el diagrama queda en como Modelo Físico de Alto Nivel (MFAN), tal como se aprecia en la figura 36 d).

4.7 Tipos de archivos

La herramienta permite abrir, editar y guardar los tipos de archivos: CSR y CDF, y así lo grafica la figura 37. Estos archivos se corresponden con el estado del documento. Al guardar un documento no finalizado se almacenará en el destino elegido como un archivo con extensión CSR, mientras que si fue finalizado, esta operación se hará en un archivo CDF.

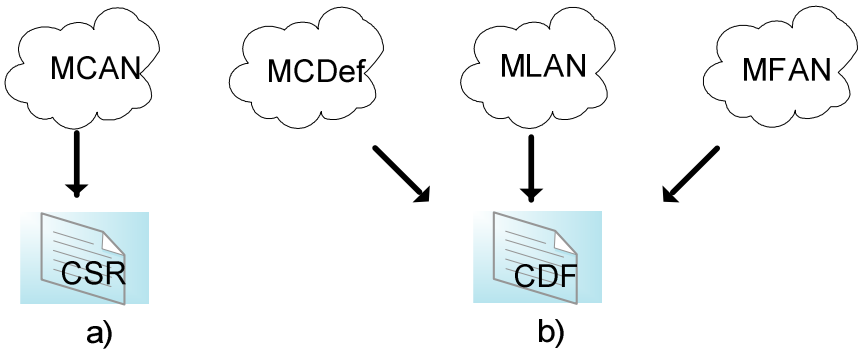

Figura 37. Extensión de archivo en relación al estado del documento.

La herramienta condiciona al usuario para guardar el documento antes de cambiar su estado por finalizado. Si éste no fue guardado como CSR, la herramienta indicará realizar dicha actividad. En consecuencia todo documento tendrá una versión no finalizada, es decir un MCAN, de modo de poder volver a realizar cambios en la especificación. Esto asegura poder obtener nuevamente el documento definitivo MCdef, un documento con extensión .cdf y continuar el proceso de modelado de datos.

Los documentos finalizados (CDF) pueden abrirse y editarse las veces que sea necesario. A pesar de denominarse modelos conceptuales definitivos, luego de ser guardados en disco, pueden volver a abrirse para editarlos y convertirlos hasta llegar al modelo lógico y físico. Esta forma de trabajo mantiene total relación con el concepto de iteración anteriormente citado, donde se realizan sucesivas modificaciones al modelo.

4.8 Comportamiento de Modelos Lógico y Físico (MLAN y MFAN)

A continuación, se presentan las restricciones que impone la herramienta y cómo son tratados los elementos que forman parte de un modelo conceptual, lógico y físico.

Las entidades, relaciones y atributos pueden ser agregados, modificados y eliminados mediante la herramienta. Estas son las operaciones fundamentales que se pueden efectuar sobre estos elementos cuando un archivo se encuentra en el estado MCAN o MCDef.

Al agregar elementos desde la especificación, se toma como nombre del elemento la palabra seleccionada. Dicho nombre se puede modificar en el momento de la creación del diagrama conceptual o posteriormente.Las entidades tienen un nombre y los datos necesarios para determinar si se trata de una especificación de otra entidad genérica (en este caso, se especifica de cual depende), o si se trata de una entidad padre (se especifica necesariamente el tipo de cobertura de la jerarquía que forma). Es posible administrar los atributos e identificadores de las entidades. Las propiedades de una entidad que establece el usuario son especificadas a través de una ventana como se presenta enla figura 38.

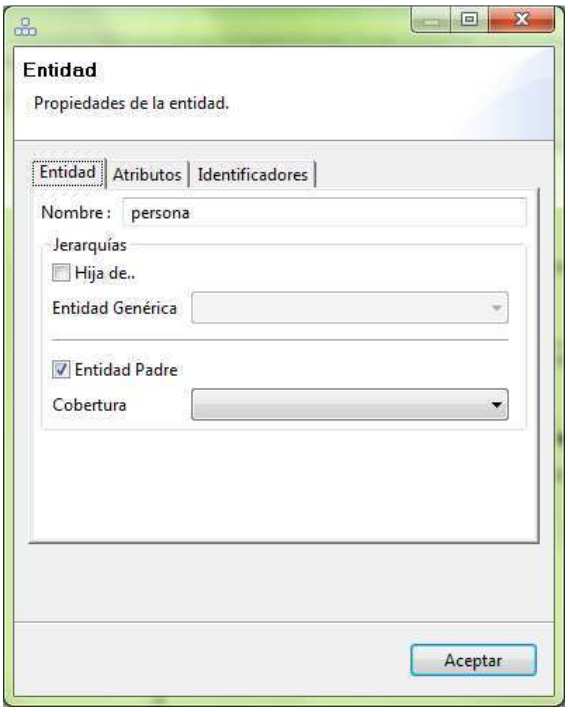

Figura 38. Propiedades de una Entidad.

Las propiedades que se establecen para las relaciones (Figura 39) son: nombre y las entidades relacionadas. Por cada una de las entidades elegidas se especifica la cardinalidad con respecto a la relación.

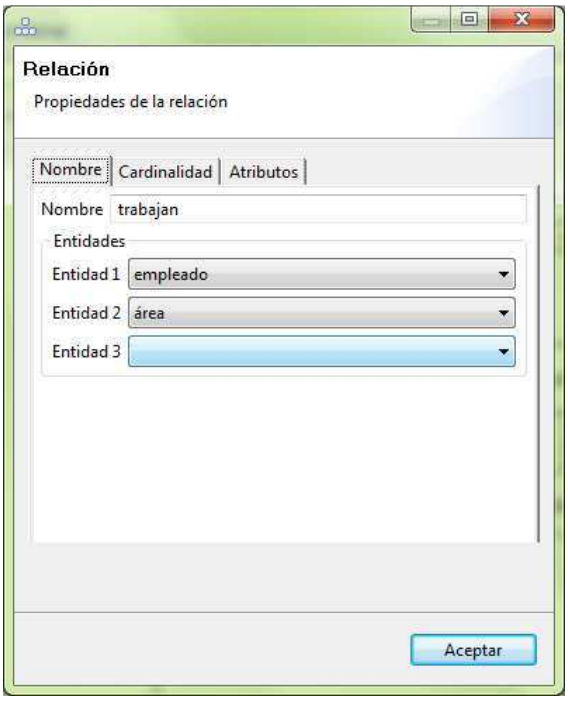

Figura 39. Propiedades de una Relación.

CasER restringe la representación de relaciones a binarias o ternarias.

En cuanto a los atributos, la herramienta brinda la posibilidad de definir si son simples o compuestos como se muestra en la figura 40. Además, se debe especificar si son monovalentes o polivalentes junto con las cardinalidades mínima y máxima respectivamente.

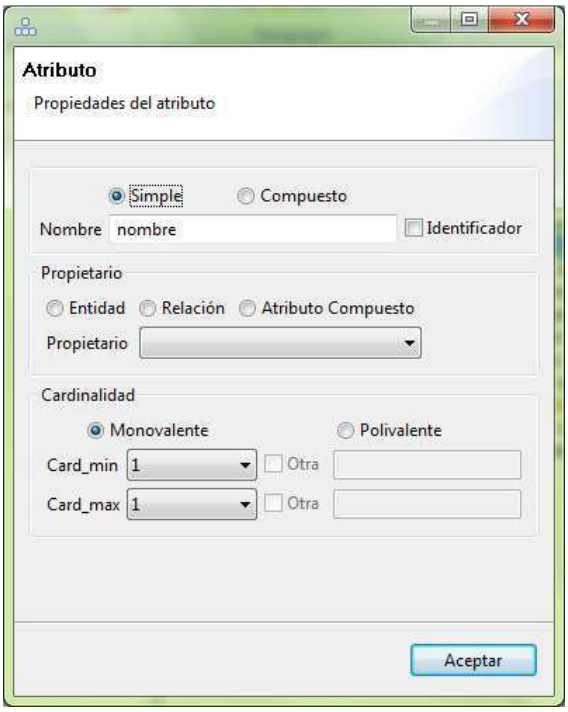

Figura 40. Propiedades de un Atributo.

CasER permite identificadores simples o compuestos, así como, internos, externos, o mixtos. Como se observa en la figura 41 a través de las propiedades de la entidad, es posible definir identificadores simples o compuestos.

Para que sea posible distinguir los identificadores, es restricción obligatoria, tener un nombre único en la entidad que identifican. Los identificadores simples toman por defecto el nombre del atributo o entidad que lo forma.

También se revisa que la elección de atributos o entidades externas que formarán parte de un identificador cumplan las condiciones necesarias para tal fin. En el caso de elegir atributos, deben ser monovalentes obligatorios. Las relaciones involucradas en identificadores externos deben ser obligatorias, con una cardinalidad máxima fija en 1.

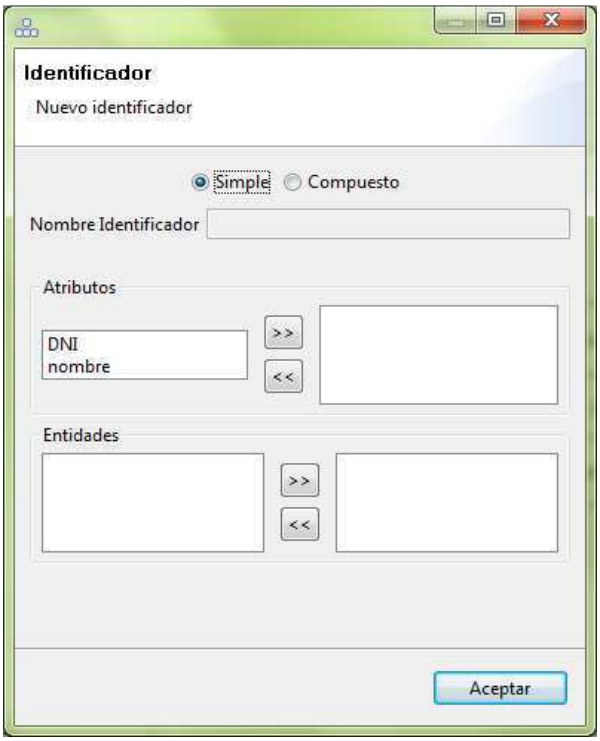

Figura 41. Propiedades de un Identificador.

4.9 Componentes de la Herramienta.

Se ha definido cuáles son los posibles estados de un documento y cómo se representan y agregan las propiedades a los elementos u objetos de un modelo (entidades, relaciones y atributos). En esta sección se describen las partes principales de la herramienta.

El árbol de objetos que presenta la herramienta (oresentado en la figura 42) permite ver cada uno de los objetos del modelo conceptual. Esto posibilita una mejor visualización y un acceso práctico y funcional para el usuario, quien puede realizar operaciones de cambio de nombres, determinar si una entidad es subconjunto de otra, especificar cardinalidades en atributos y relaciones, o directamente eliminar algún objeto si se ingresa a las propiedades de cada objeto. Cualquier cambio realizado tiene impacto en el esquema conceptual.

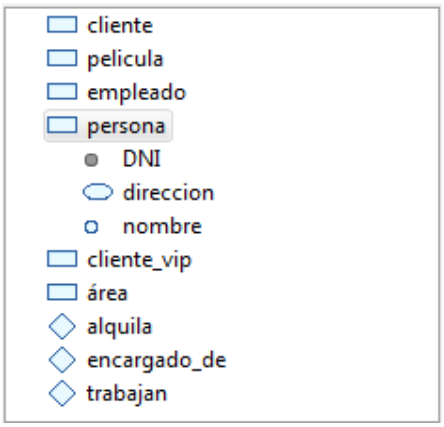

Figura 42. Árbol de objetos.

Una especificación con marcas ya realizadas contiene elementos vinculados con el diagrama de alto nivel y, por lo tanto, con el árbol. Las marcas realizadas son de distinto color; esto depende del elemento al que estén asociadas. Las entidades se marcan de color verde, las relaciones en naranja y los atributos en azul. Un ejemplo de las marcas se presenta en la figura 43.

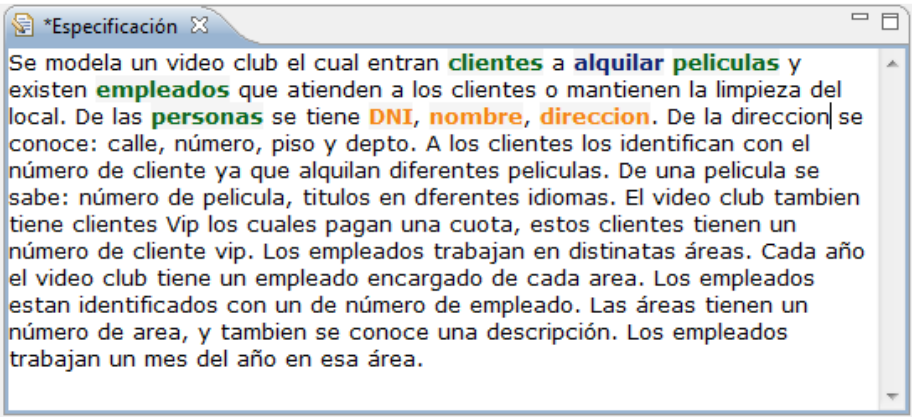

Figura 43. Especificación. Marcado de objetos.

El tercer componente de la herramienta es el editor gráfico del esquema conceptual. En él se muestran aquellos objetos que fueron marcados en la especificación.

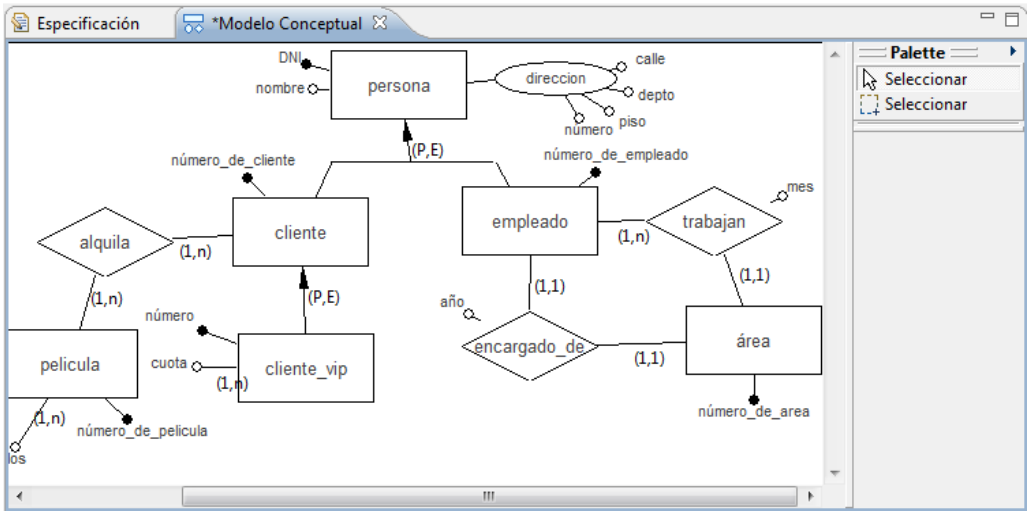

Figura 44.Esquema Conceptual en edición.

Mientras el documento no esté finalizado, se puede utilizar esta parte de la herramienta para reacomodar los objetos o para ver las propiedades de estos objetos. Es útil también como referencia para comprender el modelo que se genera.

Como se presenta en la figura 44 el “esquema en edición”, mediante la paleta de componentes ubicada a la derecha, se accede a los objetos que se desean agregar luego de finalizado el modelo.

Los botones de la barra de herramientas no están siempre disponibles para su uso. Solo un diagrama de modelo Conceptual que es definitivo (MCDef) puede ser convertido a Modelo Lógico (MFAN), representado en la figura 45, y solo un modelo Lógico puede ser transformado a modelo Físico (MFAN). Los botones se habilitan y deshabilitan en función de estas restricciones.

Otro componente importante es el editor del esquema lógico. Al comenzar el pasaje de modelo conceptual a lógico, y antes de aplicar todas las reglas, el modelo lógico es idéntico gráficamente al modelo conceptual.

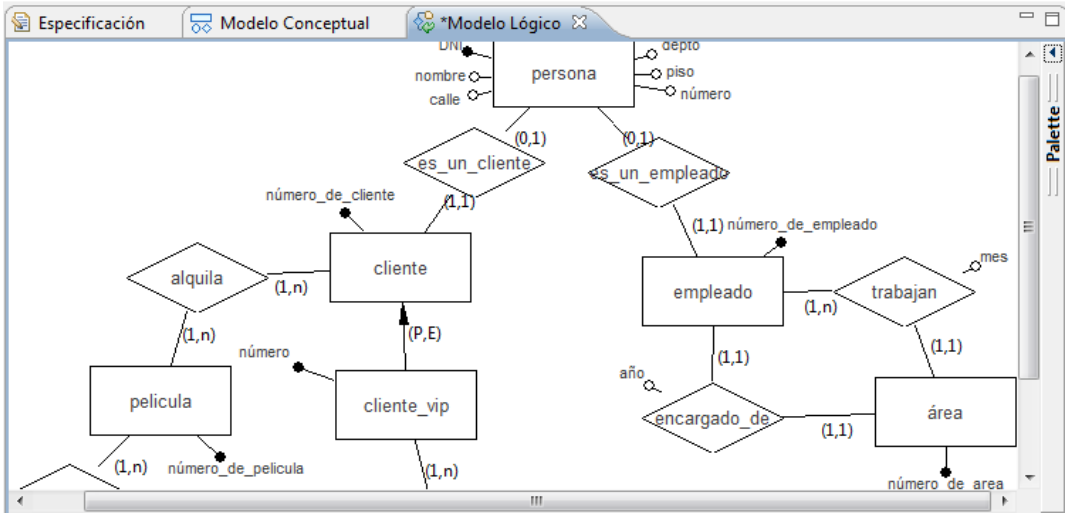

Figura 45. Documento finalizado lógico.

El esquema lógico se modifica dinámicamente a medida que se aplican cada una de las reglas de conversión de modelo conceptual al lógico. Se eliminan los atributos compuestos, los atributos polivalentes y las jerarquías. El usuario podrá decidir que método de eliminación se utiliza y también puede rever sus decisiones. El documento que se obtiene es finalizado lógico.

El quinto y último componente es el esquema físico. Las tablas del esquema se generan a partir el modelo lógico. Como primer paso se eliminan los identificadores externos en caso de que existan.

El usuario tiene la opción de realizar el pasaje físico en un único paso, o paso a paso, lo que significa que puede decidir cómo será la conversión de cada tabla en los casos en que la cardinalidad sea uno a uno en alguno de los lados de la relación.

La figura 46 muestra el modelo físico. El esquema refleja las tablas de dicho modelo. El nombre de las tablas comienza con mayúscula, las claves primarias se indican en mayúsculas y los demás campos en minúsculas.

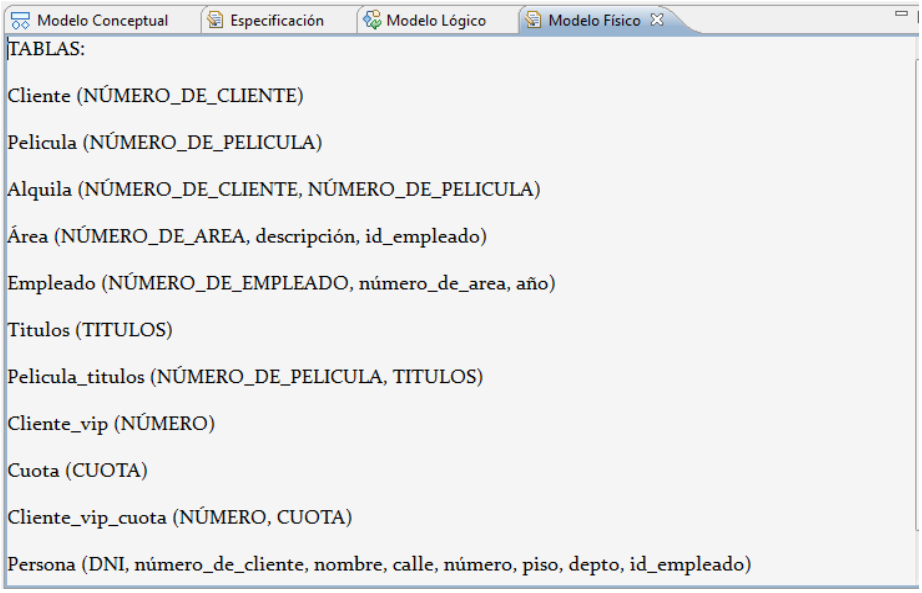

Figura 46. Documento finalizado físico.

4.10 Flujo de un documento en CasER

Para completar la descripción de la herramienta, en la figura 47 se presenta el flujo o ciclo de vida de un documento, con los distintas situaciones previamente enunciadas

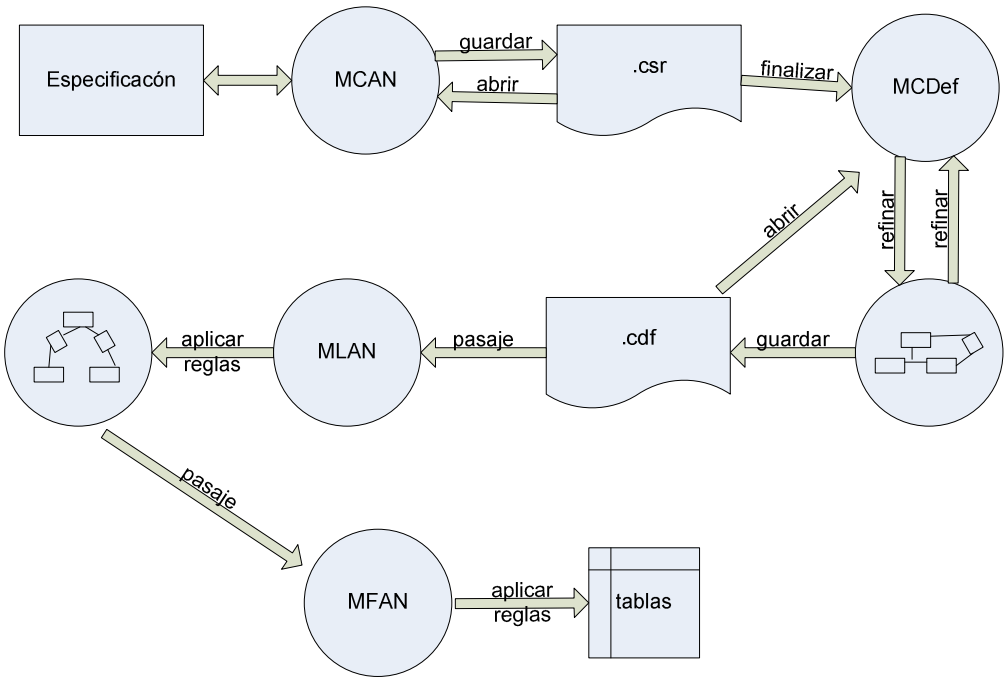

Figura 47. Flujo de un documento en CasER

Los pasos a seguir con CASER se pueden resumir del siguiente modo:

1. Crear o importar una especificación de un problema y de ser necesario editarla.
2. Marcar las potenciales entidades, relaciones y atributos expresados en la especificación.
3. Refinar el modelo generado, ya que no necesariamente todos los conceptos del problema están expresados en la especificación. Esta tarea puede realizarse en el editor gráfico, en la representación jerarquía de objetos (árbol), o en ambas.
4. Finalizar el modelo conceptual.
5. Comenzar el pasaje de modelo conceptual a lógico. Ejecutar cada paso de conversión y obtener el documento finalizado lógico.
6. Comenzar el pasaje de modelo lógico a físico. Ejecutar la eliminación de identificadores externos (si existieran) y la conversión de cada elemento básico del modelo lógico. Obtener el documento finalizado físico.

4.11 Caso práctico

El caso práctico que se presenta es un ejercicio correspondiente a los trabajos prácticos del año 2011 de la asignatura Introducción a las Bases de Datos de la Facultad de Informática . El enunciado es el siguiente

> Una distribuidora mayorista de CD y DVD informatiza su sistema de ventas. Dicho sistema debe contemplar:
>
> - Información de los clientes: DNI, apellido, nombre, fecha de nacimiento, CUIL, estado de cuenta, crédito máximo, nombre de la entidad que representa (si es que posee) e información de contacto.
> - Información de proveedores (nombre del proveedor, dirección detallada, teléfono/s, ciudad, estado de cuenta y crédito máximo).
> - Información sobre los productos (marca, descripción, tipo de producto (CD / DVD), stock actual, stock máximo, stock mínimo, precio de compra, precio de venta).
> - Información de las ventas, de cada venta se registra: número de venta, fecha de la venta, productos que incluye dicha venta, cliente al cual se le realizó la venta y monto total de la misma.

- Información de las compras, de cada compra se registra: número de compra, fecha de la compra, productos que incluye, proveedor al cual se le realizó y monto total de la misma.
  Aclaraciones:
- Tipo de producto puede ser entidad o un atributo. Explicar ventajas y desventajas de ambas decisiones.
- El monto total en Ventas/Compras no es un atributo derivado del precio de los productos que incluye.

4.11.1 Generación del Modelo Conceptual inicial.

El primer paso consiste en importar la especificación desde el menú o desde la barra de botones. Luego, a través del menú contextual, se marcan los elementos de la especificación que representan entidades, relaciones y atributos (Figura 48).

De esta manera se agregan al diagrama gráfico todos los elementos. Cuando se marcaron todas las palabras se obtiene el MCAN (Figura 49).

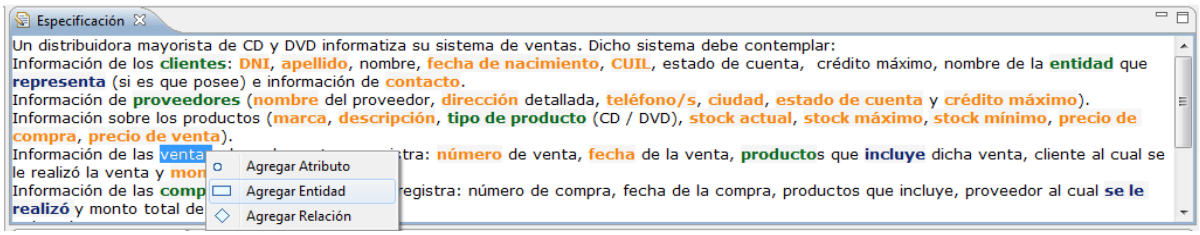

Figura 48. Figura Especificación y marcado de elementos.

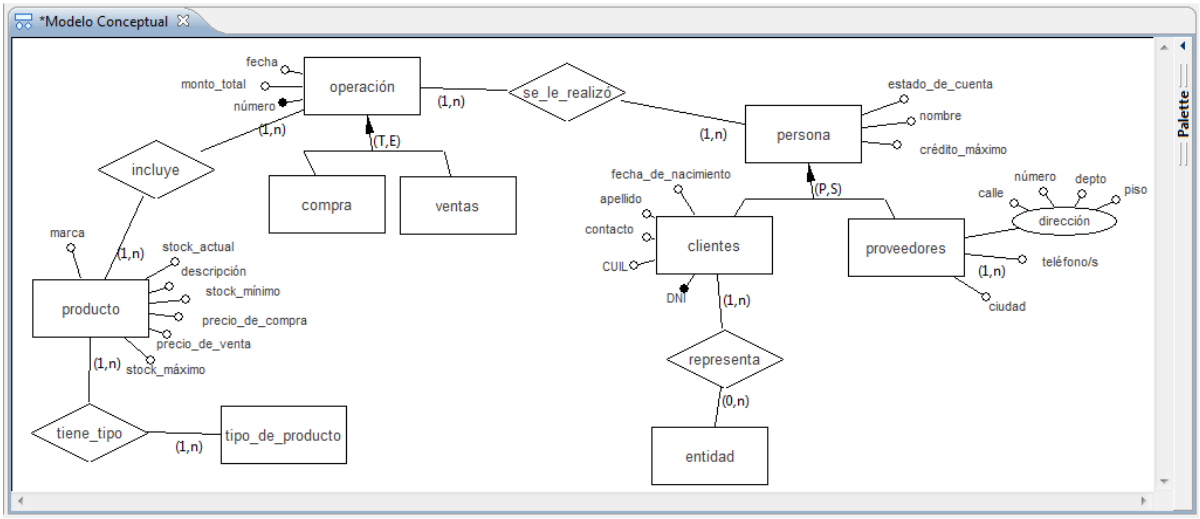

Figura 49. Modelo Conceptual.

4.11.2 Finalización del Modelo Conceptual

Como se visualiza en la Figura 49, luego de que se trabajó en la especificación de requerimientos y todos los elementos correspondientes al modelo fueron seleccionados y graficados, se puede finalizar el documento (Figura 50). Cuando esto último ocurre, se guarda la versión del MCAN que almacena la relación entre los elementos de la especificación y los del diagrama.

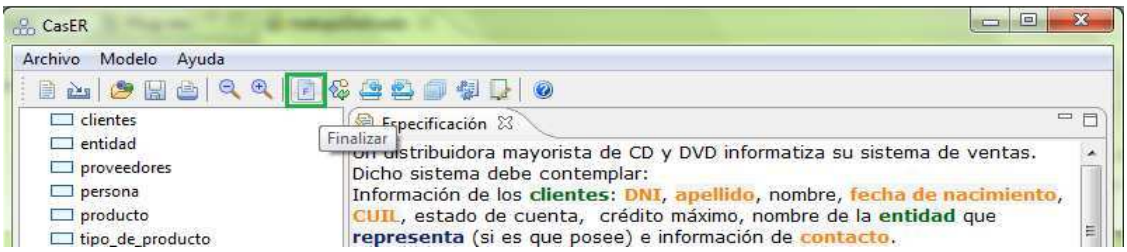

Figura 50. Finalización el modelo.

El diagrama se almacena en un archivo pero esta vez con extensión CSD. Posteriormente es posible recuperar dicho archivo para continuar agregando elementos desde la paleta de botones que se encuentra situada junto al diagrama. También es posible editar las propiedades de los objetos ya agregados.

4.11.3 Conversión del Modelo Conceptual en Lógico

Para comenzar con el pasaje a Modelo lógico, antes se debe finalizar el documento.

- Paso 1: Eliminación de Atributos Compuestos. El usuario decide el método de eliminación que se usará. En este ejemplo:

Figura 51. Eliminación de atributos compuestos. Opciones.

Para el atributo compuesto Dirección de la entidad Proveedores luego de seleccionar la Opción A, se genera un único atributo que surge de la concatenación de todos los atributos simples que contenía el anterior atributo compuesto. El resultado se presenta en la figura 52.

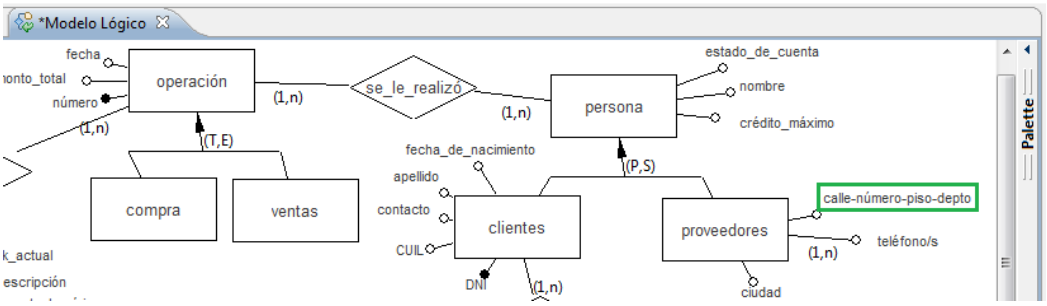

Figura 52. Eliminación de atributos compuestos. Opción A.

En la figura 53 se presenta el resultado de haber seleccionado la opción B, con la cual se definen los atributos simples sin un atributo compuesto que los resuma.

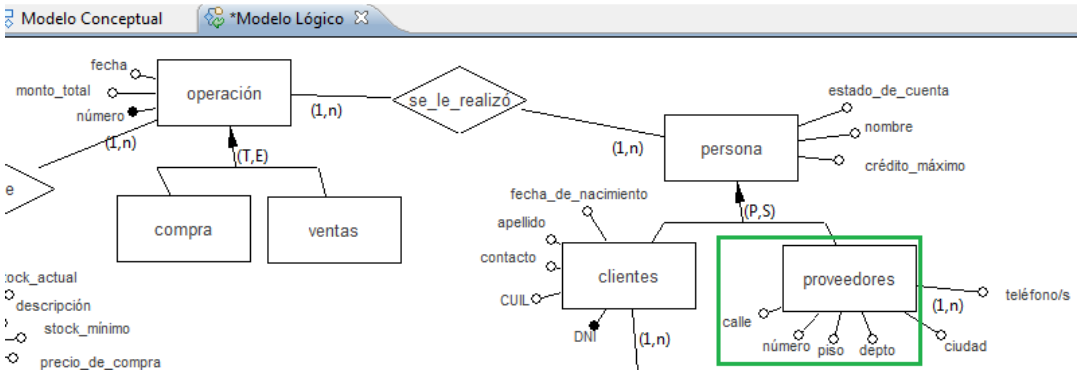

Figura 53. Eliminación de atributos compuestos. Opción B.

Finalmente la figura 54 presenta el resultado de seleccionar la opción C.

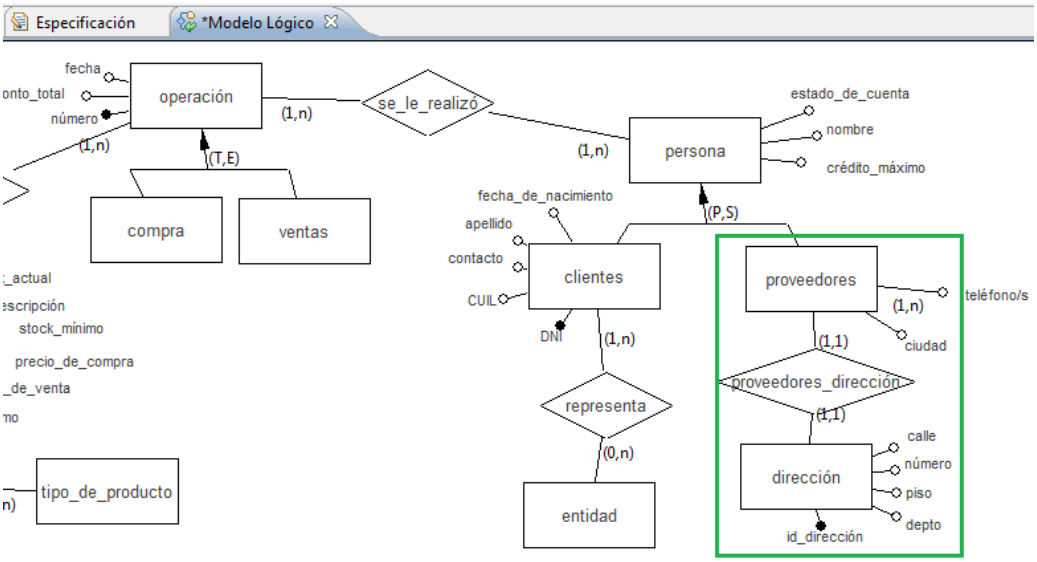

Figura 54. Eliminación de atributos compuestos. Opción C.

Se genera una nueva entidad dirección que representa el atributo compuesto. La nueva entidad contendrá todos los atributos simples que pertenecían al atributo compuesto eliminado.

Luego de dar comienzo con la conversión de modelo conceptual en modelo lógico, es posible ver el estado de progreso de dicha conversión, tal como se presenta en la figura 55.

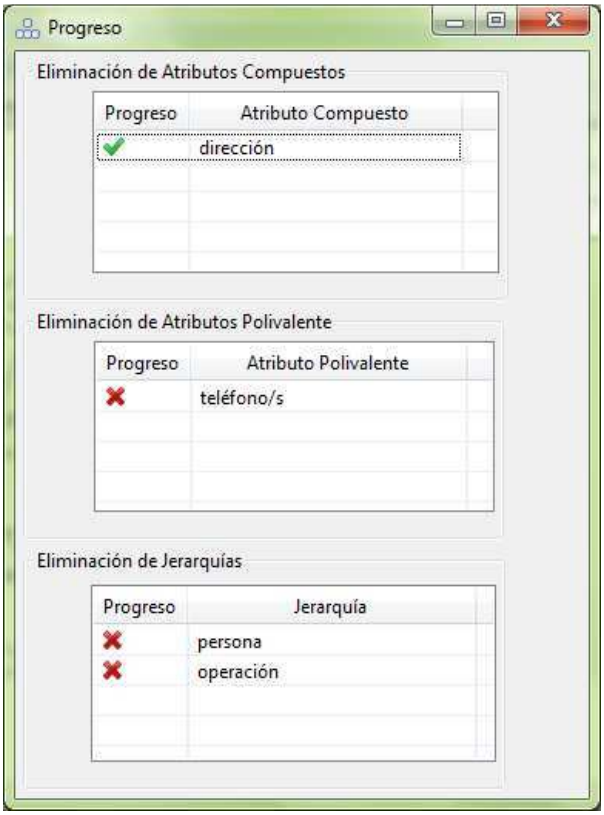

Figura 55. Ventana progreso.

Finalizada la eliminación de atributos compuestos, se comienza el paso siguiente (tal como lo indica la figura 56) que consiste en la eliminación de los atributos polivalentes.

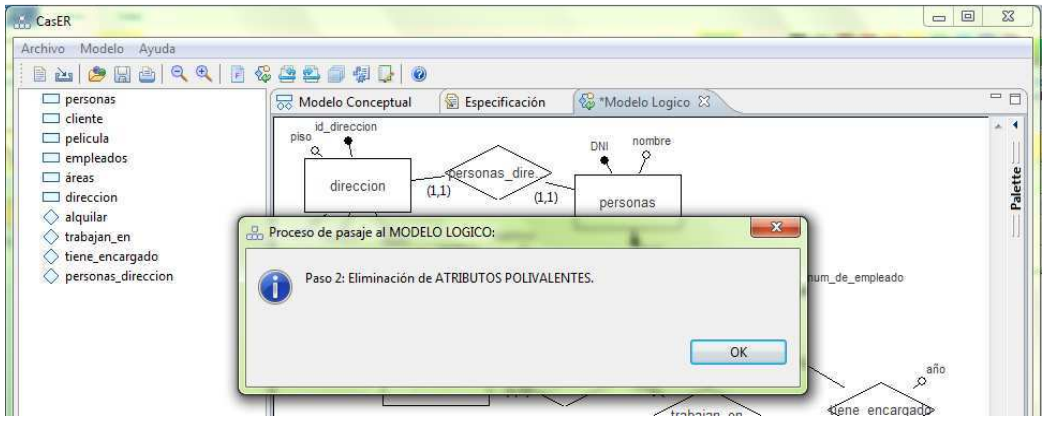

Figura 56. Paso 2: Eliminación de Atributos Polivalentes.

Un mensaje indica cual atributo polivalente se debe eliminar.

Para eliminar un atributo polivalente es necesario ingresar las cardinalidad mínima (figura 57) con la que se va a generar la nueva relación y entidad. Por defecto la cardinalidad máxima siempre será n.

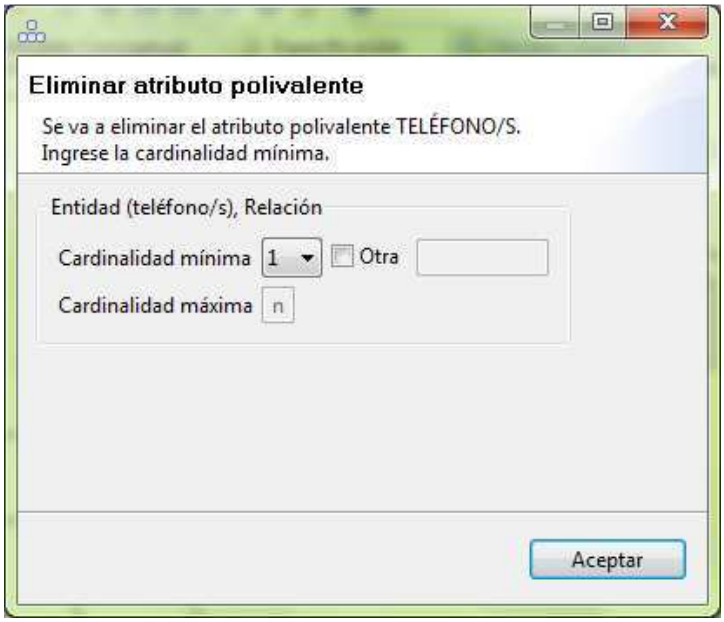

Figura 57. Ingreso de cardinalidad mínima.

Cuando no existan más atributos polivalentes para eliminar, se indica con un mensaje. Finalmente se elimina el atributo polivalente como podemos ver en la figura 58.

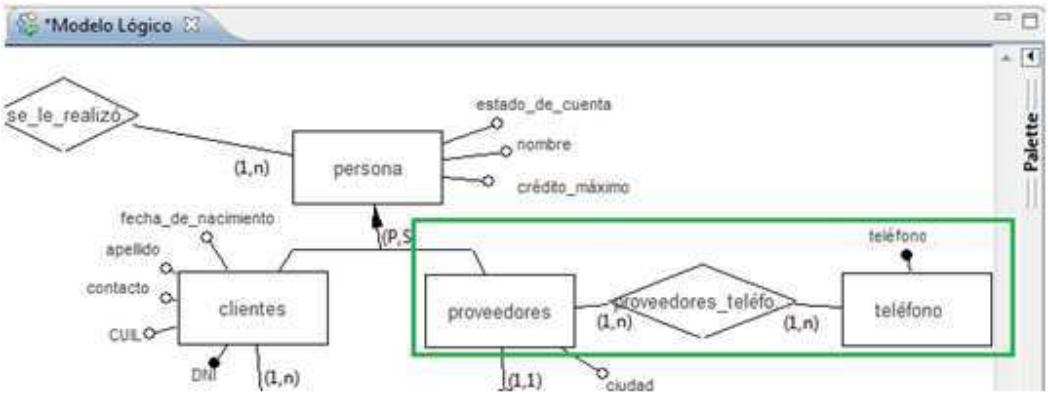

Figura 58. Eliminación de atributo polivalente.

El siguiente paso es la eliminación de Jerarquías (figura 59). De acuerdo a la cobertura, las opciones disponibles serán dos o tres.

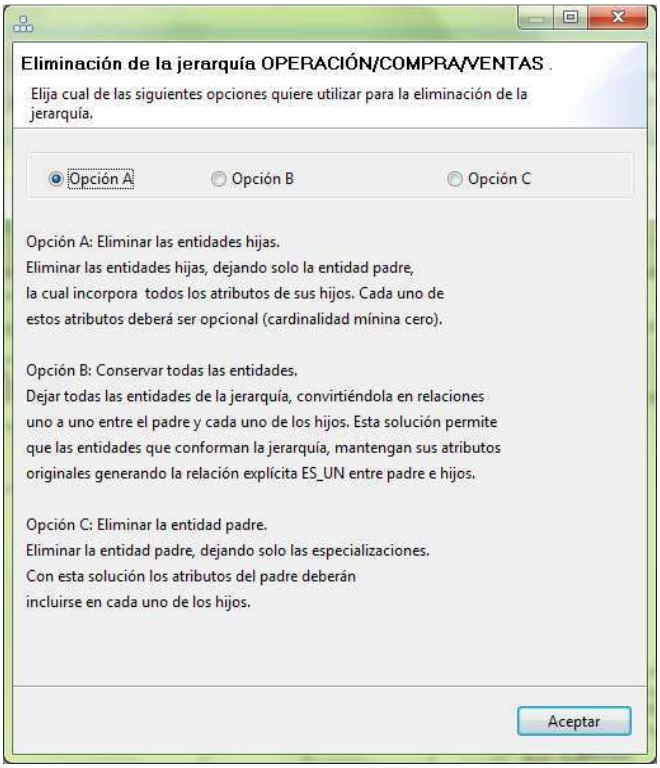

Figura 59. Opciones de Eliminación de jerarquías.

En el caso presentado se decidió conservar todas las entidades, opción A para el caso de la jerarquía Operación/Compra/Venta, y la opción B para la jerarquía Persona/Clientes/Proveedores (figura 60).

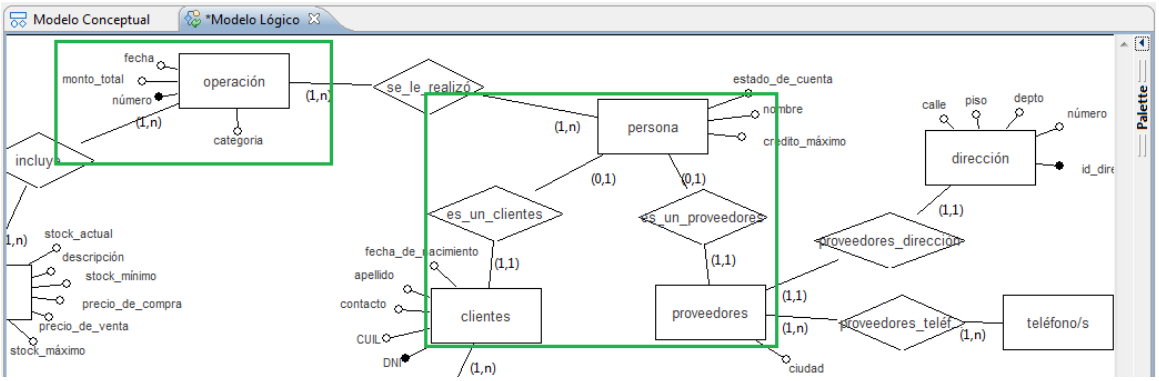

Figura 60. Eliminación de jerarquías. Opción B.

Cuando se eliminan las entidades hijas o subentidades se agrega el atributo categoría a la entidad padre. Este nuevo atributo identificará la subentidades, en este caso Venta o Compra.

Cuando no existen más jerarquías a eliminar, se finaliza el pasaje a modelo lógico para obtener el MLAN. Como se visualiza en la figura 61.

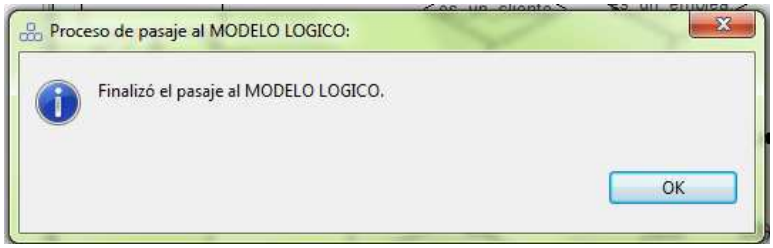

Figura 61. Finalización de pasaje a modelo lógico.

4.11.4 Generación del Modelo Físico

Al comenzar con el pasaje a Modelo Físico, se aplican las reglas presentadas anteriormente en este trabajo. En principio, si el esquema lógico contiene identificadores externos, serán eliminados.

Como se visualiza en la figura 62, el primer paso es la elección de conversión al modelo físico en un único paso generando las tablas automáticamente, o si será un pasaje paso a paso con la intervención del usuario.

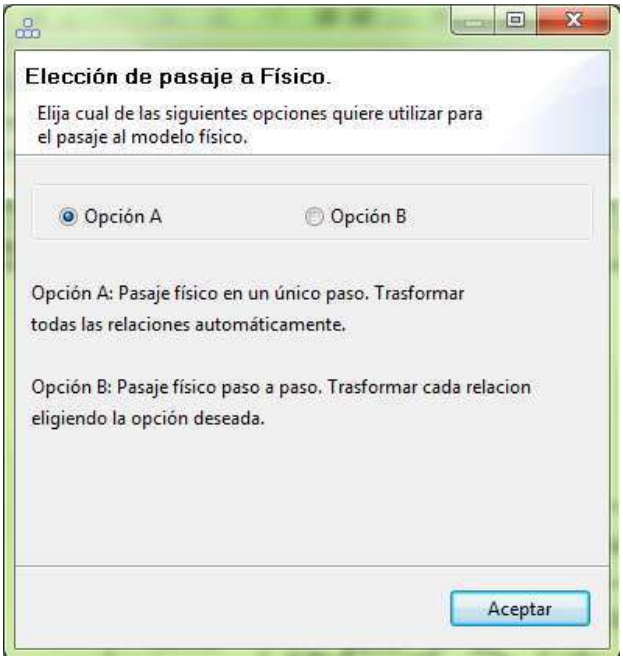

Figura 62. Elección de pasaje a modelo físico (primer paso de conversión).

Si en el primer paso la opción seleccionada es la A entonces directamente se obtiene el modelo físico de tablas de alto nivel (figura 64). En otro caso, si la opción es la B, como se puede ver en la figura 63, se presentan las opciones de conversión para cada una de las relaciones del modelo lógico, en las que el usuario puede optar entre pasar a dos o tres tablas.

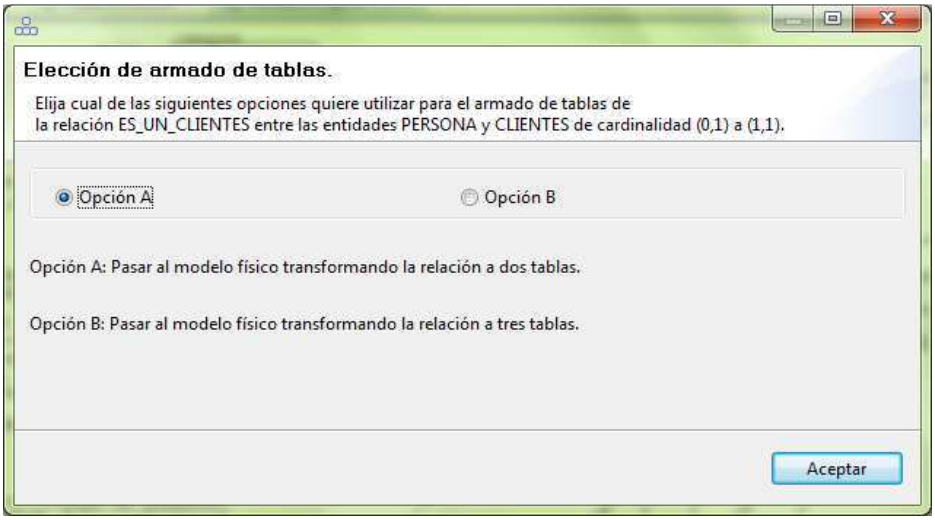

Figura 63. Elección de pasaje a modelo físico. Opción B (segundo paso de conversión)

El resultado final del proceso de diseño lo constituye el modelo físico el cual se presenta en la figura 64.

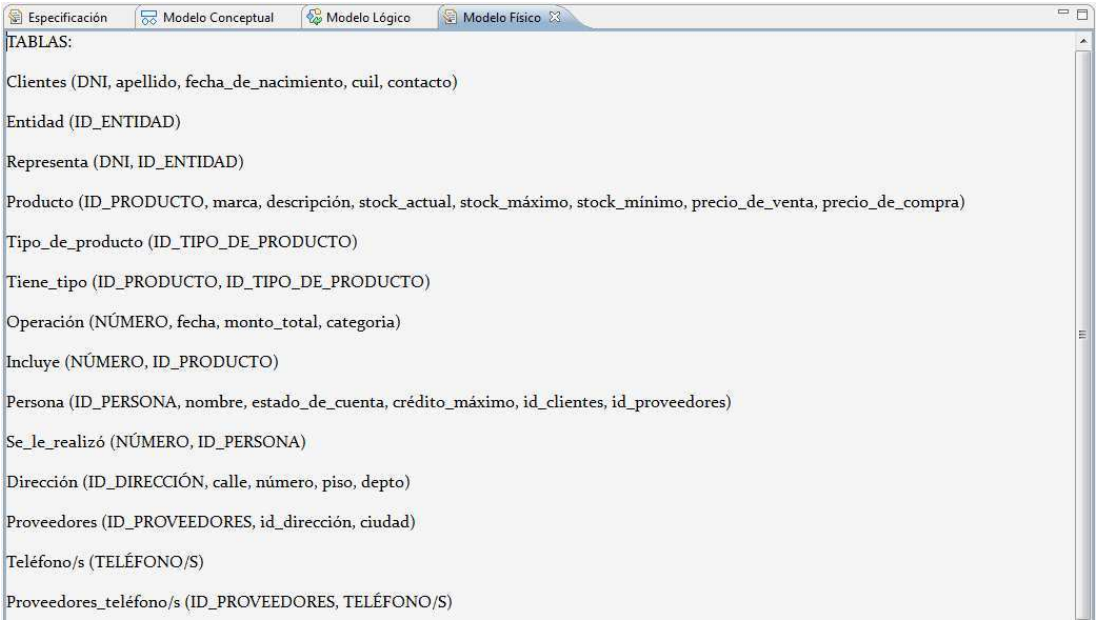

Figura 64. Modelo Físico.

Durante el proceso de modelado se genera un log de las operaciones realizadas. En el log se registran todos los pasos de conversión a modelo lógico y modelo físico. Cada deshacer y rehacer actualiza el log. Cada decisión tomada se asienta y se explica en detalle el porqué, a modo de información para una mejor comprensión de las medidas tomadas. Finalmente en el pasaje a tablas se registran cada una de las tablas del modelo físico.

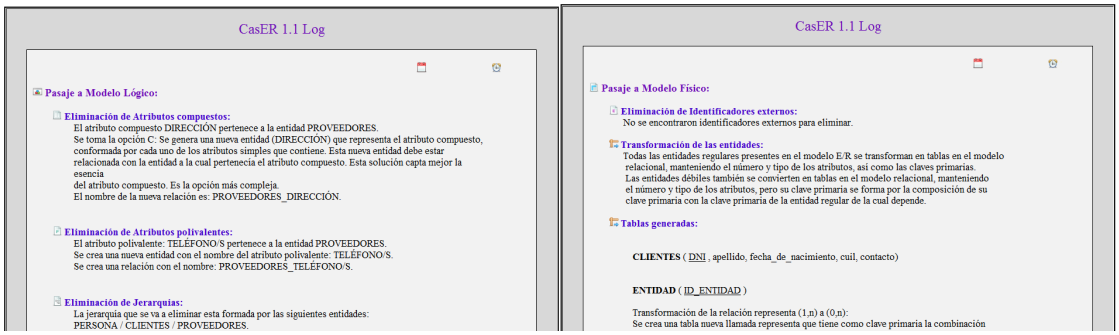

Figura 65. Log pasaje a Modelo lógico y a Modelo físico

El log (figura 65) se podrá leer a partir del comienzo del diseño lógico. Se abrirá un navegador con una bitácora o registro de todas las decisiones tomadas en el diseño lógico y en el diseño físico, con una breve descripción técnica de cada paso del pasaje. Se generar un documento HTML con el nombre del archivo abierto en la herramienta.

CAPITULO 5

5.1 Descripción de implementación.

La finalidad de éste capítulo, es presentar como fue implementada la herramienta y explicar conceptos que puedan ser de utilidad en trabajos futuros.

La herramienta fue desarrollada bajo el paradigma de programación orientado a objetos, y con el lenguaje de programación JAVA utilizando el IDE Eclipse [13]. Se siguió el patrón de diseño MVC (Model View Controller). La base de este desarrollo es RCP (Rich Client Plataform), una plataforma que permite crear aplicaciones Java multiplataforma según [18] y [19]. Es aquí donde se aplica el patrón MVC. De RCP se utilizó GEF (Grap Editing Framework), librería creada para facilitar la labor de construcción de aplicaciones para dibujar diagramas, en el caso particular de CasER los modelos conceptuales y lógicos. Para la implementación de las restantes partes se utilizaron otras herramientas que componen RCP, como son SWT (Standard Widget Toolkit) y JFace. En CasER se usaron para el editor de texto o el árbol que presenta cada uno de los objetos de los modelos.

GEF separa las distintas capas de un editor gráfico y es el soporte para la implementación del controlador. En este framework, el controlador es el encargado, entre otras cosas, de hacer de intermediario entre la vista y el modelo.

El Modelo consta de la información de base que interesa almacenar en un editor. La información que contenga el Modelo es la persistida al momento de guardar o cerrar un editor y con la que se cuenta al momento de abrirlo. Si bien GEF no pone restricciones en cuanto a las interfaces o clases abstractas que debe extender el Modelo, es necesario que el mismo cuente con un mecanismo para notificar adecuadamente sus cambios a terceras partes. Este mecanismo es necesario para que el Controlador pueda actuar al momento de registrar un cambio en el Modelo. Dado que GEF no suministra ninguna base para el armado del modelo, se confeccionó una clase abstracta denominada ObjetoCaser, la cual brinda las características principales comunes para todos los elementos del modelo. Una de las características más importantes, es la posibilidad de registrarse para ser notificado ante el cambio de cualquiera de las propiedades del elemento, como se verá más adelante.

Los cambios realizados sobre el modelo por parte del Controlador (generalmente a pedido del usuario) son representados por Comandos. Los comandos, sólo deben conocer la existencia del modelo (es por eso que se pueden considerar una extensión de éste), y deben contener la lógica para efectuar o deshacer los cambios requeridos por el Controlador.

La Vista compone la capa visual del editor. Dada la gran separación entre capas que propone GEF, la vista no puede conocer o hacer referencia a ningún objeto del modelo, ni tampoco tener lógica asociada al editor. Debe contener información propia de la representación visual de los elementos que se pretenden graficar. GEF utiliza Draw2D como capa visual base y en ella, el elemento atómico es la figura, representada por la interfaz IFigure. Draw2D provee una amplia gama de figuras elementales prediseñadas y la composición entre ellas permite un amplio espectro de posibilidades con un mínimo esfuerzo. Por lo general, un elemento del modelo tiene asociado una figura de la Vista, aunque no es una limitación del framework.

El Controlador tiene el rol de mediador entre el usuario, la vista y el modelo. En GEF, el controlador es llamado EditPart. Principalmente, actúa de observador del modelo, reflejando los cambios en la vista cuando es necesario. Por lo general existe un EditPart por cada elemento del modelo (aunque no es una limitación), y entre los distintos EditParts se forma una estructura de árbol, siendo el nodo raíz el correspondiente al diagrama. Los EditParts también son los encargados de administrar los pedidos de cambio del usuario, denominados Requests, y transformarlos en Comandos. Por ejemplo para el tipo de pedido “delete” (borrar), la política de edición deberá identificar el tipo elemento a borrar y crear el comando de borrado, correspondientemente. La administración también consta de aceptar, ignorar o rechazar los Requests, a través de clases que encapsulan las políticas de edición.

Hasta este punto, se describió en forma genérica las capas propuestas por GEF; se desarrolla aquí, cómo fueron implementadas para dar lugar a la implementación de la herramienta.

El modelo de CASER consta de una clase abstracta base, que contiene el comportamiento común a todos los elementos, como el soporte de propiedades. Luego existe un elemento raíz, que representa al diagrama y que contiene una lista de los elementos que lo componen, los que fueron llamados Nodos. El diagrama de estructura es el de la figura 66; se continúa con una descripción de los elementos principales:

- ObjetoCaser: Clase abstracta que contiene la funcionalidad común de todos los elementos del modelo como el soporte de propiedades y la capacidad de ser serializable para poder mantener un estado persistente. Los métodos principales que provee la clase son:

    - addPropertyChangeListener(), removePropertyChangeListener(): Permiten agregar o quitar un "Observador" de los cambios de valores de las propiedades del elemento, a través de la interfaz PropertyChangeListener, que contiene un método para informar dicho tipo de eventos.

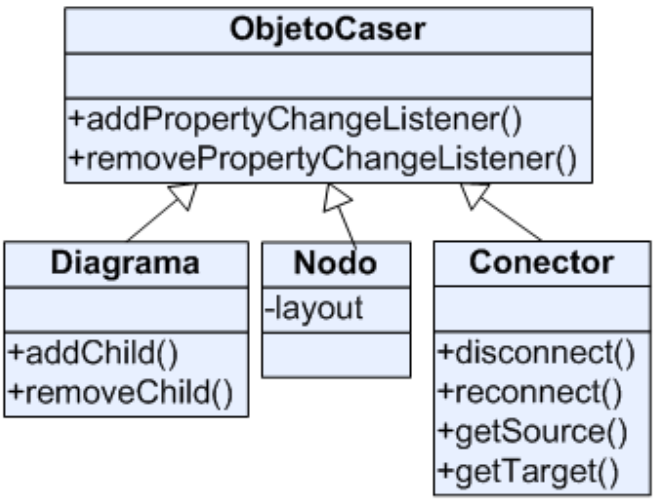

Figura 66. Elementos principales del modelo de CASER

- Diagrama: Elemento raíz del modelo que actúa de contenedor del resto de los elementos. Los métodos provistos para contener estos elementos son:

    - addChild(), removeChild(): Métodos de acceso a los elementos hijo del modelo. Permiten agregar y quitar los hijos, respectivamente.

- **Conector**: Representa una conexión dirigida entre dos elementos del modelo, el origen (source) y el destino (target). Posee además una propiedad que representa el punto medio de la conexión para poder mantener un orden visual. Los métodos principales son:

    - puntoMedio(): representa el punto medio de la conexión.

- conector(): La conexión debe construirse indicando los elementos origen y destino.

- disconnect(), reconnect(): Permiten indicar a los elementos que están asociados por esta conexión, que dicho vínculo debe eliminarse o volverse a establecer.

- getSource(), getTarget(): Métodos de acceso a los elementos origen y destino.

- **Nodo**: Clase abstracta que incorpora el comportamiento de un elemento del modelo visible gráficamente. Además posee propiedades y métodos de acceso para los siguientes atributos:

    - layout: Dimensiones del elemento, en alto y ancho y la posición del elemento dentro del diagrama (coordenadas X e Y).

Como principal componente de la figura 67, se encuentra la interfaz **ObjetoConceptual**; la que permite relacionar los elementos del diagrama con el texto de la especificación. No todos los elementos que se muestran en el editor gráfico mantienen una relación con elementos de la especificación; por esta razón, sólo aquellos que sí la mantienen (entidad, relación, atributo) son los que implementan la interfaz mencionada.

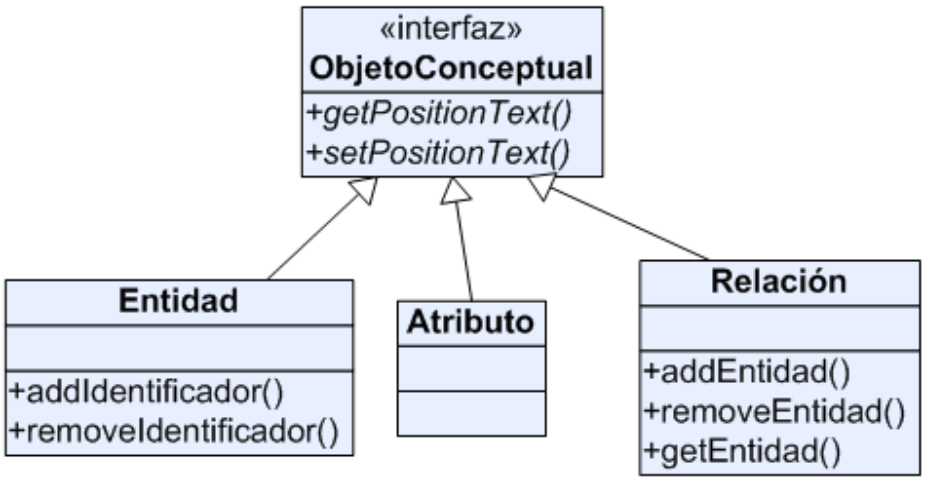

Figura 67. Interfaz que deben implementar objetos con relación con la especificación

La vista es la capa responsable de representar visualmente el estado del modelo. Debe ser actualizada cada vez que el modelo cambie, ya sea por acción de la interacción con el usuario o por cambios internos. La vista no debe almacenar información sobre el modelo y solamente debe poseer el comportamiento asociado a la visualización. Se compone para la implementación de CasER de dos elementos principales: los nodos y las conexiones entre ellos. Los nodos se representan con figuras según qué dato representen en el modelo conceptual en el enfoque de [1].

Las figuras que representan a los nodos del modelo son subclases de org.eclipse.draw2d.Figure. Mientras que los conectores heredan el comportamiento org.eclipse.draw2d.PolylineConnection. Los principales métodos de estas clases son:

- setLayout(): cambia la forma de los nodos según el usuario lo indique.
- getPuntoMedio(): indica el punto medio de la conexión según los nodos origen y destino son modificados.

La capa del controlador de CasER consta básicamente de un EditPart por cada elemento del modelo. Para poder actuar en caso de que alguna de las propiedades de los elementos del modelo cambie, los EditParts implementan la interface PropertyChange- Listener para poder registrarse como observadores de dichas propiedades. La interfaz PropertyChangeListener forma parte de la biblioteca integrada de Java y posee un método “propertyChange” que es llamado por el objeto observable cada vez que se produce un cambio sobre los atributos observados para poder notificar a los observadores. El diagrama de estructura y el detalle de los elementos se muestran en la figura 68.

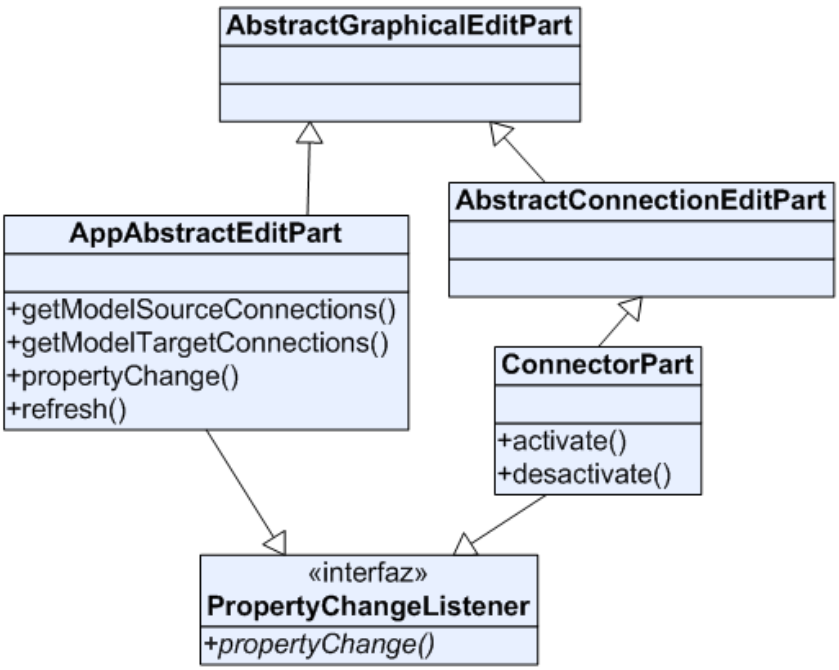

Figura 68. Elementos del controlador

Se describen a continuación otras clases que forma parte de esta capa y tienen un papel relevante en el desarrollo.

- **AbstractGraphicalEditPart**, **AbstractConnectionEditPart**: Clases abstractas que implementan la funcionalidad común de un EditPart gráfico en general y de conexión en particular, respectivamente. Estas clases son provistas por GEF y se encuentran en org.eclipse.gef.editparts. Los métodos principales que provee son:

    - activate(): indica que el controlador ha sido activado y permite llevar a cabo tareas de inicialización como por ejemplo el registro en calidad de observador al elemento del modelo correspondiente.

    - desactivate(): marca la finalización del ciclo de vida del controlador para poder realizar tareas de limpieza.

- createEditPolicies(): se crean las políticas de edición, que son básicamente las encargadas de convertir los pedidos del usuario (requests) en comandos entendibles por el modelo.

- createFigure(): método destinado a la construcción de la figura que representará gráficamente al nodo.

- **DiagramaPart**: Controlador que representa al diagrama en su totalidad. Juega el papel de nodo raíz en la jerarquía de controladores y brinda la política de edición que permite resolver los distintos tipos de elementos del diagrama. Agrega los siguientes métodos, aparte de la personalización de los provistos por las clases base:

  - getModelChildren(): Permite acceder a los elementos del modelo del primer nivel, por debajo del nodo raíz.

  - appAbstractEditPart: Controlador principal del framework, encargado de la mediación entre la capa de vista y modelo de cada nodo del diagrama. Los métodos adicionales provistos son:

  - getModelSourceConnections(), getModelTargetConnections(): Métodos para acceder a la lista de conexiones entrantes o salientes del nodo, respectivamente.

  - getSourceConnectionAnchor() y derivados: Especifica el tipo de terminación que tendrá la conexión.

  - propertyChange(), refresh(): Método que indica que una propiedad ha cambiado en el nodo asociado. Para actualizar la capa visual de los tipos de propiedades conocidos se invoca al método refresh correspondiente.

- **ConectorPart**: Controlador para las conexiones entre nodos.

Interacción de las distintas capas en la ejecución de un Nodo

En la figura 69 se muestra la interacción entre los distintos elementos de cada capa MVC, para el movimiento de un nodo dentro de un diagrama.

Los pasos se pueden dividir en dos grupos, por un lado la creación del comando, que se realiza al iniciar el movimiento del nodo dentro del diagrama (pasos a y b), y por otro lado la ejecución del comando, realizada al "soltar" el componente (pasos del c al h). A su vez, los elementos que intervienen en la interacción se dividen en tres grupos correspondientes a las distintas capas MVC, como lo indica el diagrama.

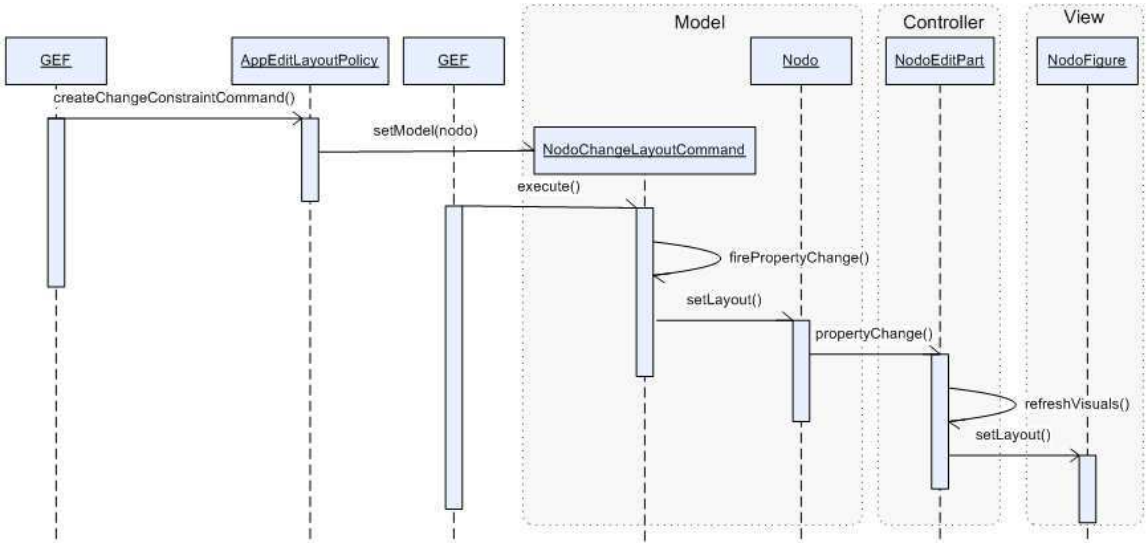

Figura 69. Elementos de las capas MVC

a) El framework GEF, invoca a la política de edición (EditPolicy) provista por CasER, requiriéndole la creación de un comando del tipo "ChangeConstraint", correspondiente al pedido del usuario de cambiar la posición del componente.

b) El EditPolicy crea el comando correspondiente, pasándole como parámetro una referencia del elemento en cuestión.

c) Una vez establecido el nuevo lugar del componente (cuando el usuario lo "suelta" dentro del diagrama) el framework GEF envía un mensaje al comando para que se ejecute.

d) El comando le informa el nuevo tamaño al elemento, a través del mensaje setLayout().

e) El cambio en el valor de la propiedad del elemento, dispara la notificación de todos los observadores registrados al elemento.

f) El EditPart asociado al elemento, al ser uno de los observadores, es notificado del nuevo valor a través del mensaje propertyChange().

g) El EditPart, al enterarse del cambio de valor, invoca al método que informará a la capa visual de dicho cambio, llamado refreshVisuals().

h) El controlador informa el cambio a la figura que representa el componente, enviando el mensaje setBounds().

Existen objetos que son los encargados de mantener configuraciones, controles comunes, y opciones por defecto que provee la herramienta y se describen a continuación.

- **AdministradorNodes**: esta clase es la que colabora con la parte de la interfaz, una de la operaciones fundamentales es la de establecer el lugar en el que un nuevo elemento agregado desde el texto debe ser ubicado, las operaciones que más relevantes son:

    - getX(), getY(): retornan los puntos X e Y respectivamente donde los elementos deben ser agregados en el editor gráfico.

    - getNombreEntidad(), getNombreRelacion(): cuando se agregan elementos desde la paleta los nombres de estos son consecutivos a medida que se van agregando, Entidad1 Entidad2… estos métodos son los encargados se generar un nuevo nombre.

- **AdminitradorModeloConceptual**: la característica principal de esta clase es la de tener los valores por defecto de los elementos del modelo conceptual, como cardinalidades mínimas y máximas de relaciones y atributos, coberturas en las jerarquías, ente otros.

- **AdministradorEditor**: es la encargada de abrir, guardar y cerrar un documento manteniendo reflejado los dos editores, el de texto y el gráfico.

Se describen a continuación otras clases que tienen un papel relevante en la etapa de pasaje a Modelo Lógico y Modelo Físico.

- **ProcesoLogico:** Controlador principal que representa el pasaje al Modelo Lógico.

Se encarga de ejecutar las tres reglas correspondientes a la eliminación de atributos compuestos, la eliminación de atributos polivalentes y la eliminación de jerarquías (figura 70).

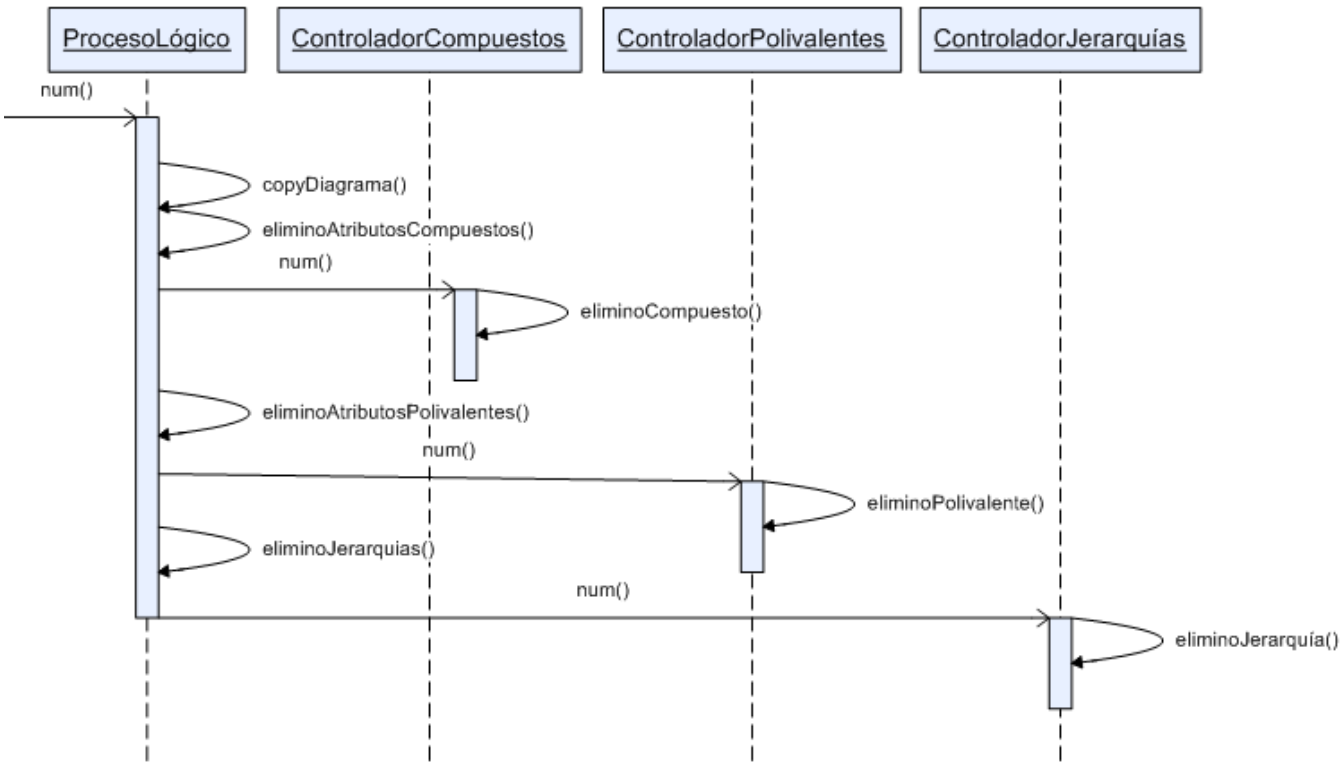

Figura 70. Proceso Lógico

Los métodos principales que provee son:

- copioDiagrama(): se encarga de realizar una copia del Diagrama Conceptual. A partir del nuevo diagrama, comienza la transformación al Modelo Lógico con la aplicación de las reglas correspondientes. Se realiza la copia para trabajar la conversión sin perder el diagrama original del Modelo Conceptual.
- run(): se encarga de ejecutar los métodos que van a realizar la conversión correspondiente a las reglas mencionadas anteriormente:
    1. eliminoAtributosCompuestos(): se encarga de invocar al controlador de atributos compuestos.

2. eliminoAtributosPolivalentes(): se encarga de invocar al controlador de atributos polivalentes.

3. eliminoJerarquias(): se encarga de invocar al controlador de jerarquías.

- **Controladoratributocompuesto**

  - inicioCompuesto(): indica que el controlador del proceso de atributos compuestos ha sido iniciado.

  - finCompuesto(): marca la finalización de la eliminación de atributos compuestos.

  - hayCompuesto(): indica si dentro de los elementos existe un atributo compuesto para eliminar.

  - ejecutoCompuesto(): se encarga de localizar todos los atributos compuestos del modelo conceptual y eliminarlos aplicándoles las reglas seleccionadas por el usuario.

  - logCompuesto(): se encarga de guardar en el log las opciones elegidas dentro de las alternativas propuestas al usuario.

- **Controladoratributopolivalente**

  - inicioPolivalente(): indica que el controlador del proceso de atributos polivalentes ha sido iniciado.

  - finPolivalente(): marca la finalización de la eliminación de atributos polivalentes.

  - hayPolivalente(): indica si dentro de los elementos, existe un atributo polivalente.

  - ejecutoPolivalente(): se encarga de localizar todos los atributos polivalentes del modelo conceptual y aplicarles las reglas elegidas por el usuario para la eliminación.

  - logPolivalente(): se encarga de guardar en el log, la opción elegida dentro de las alternativas propuestas al usuario.

- **Controladorjerarquia**

- inicioJerarquia(): indica que el controlador del proceso de jerarquías ha sido iniciado.

- finProceso(): marca la finalización de la eliminación de jerarquías.

- hayJerarquia(): indica si dentro de los elementos existe una jerarquía para eliminar..

- ejecutoJerarquia(): se encarga de localizar todas las jerarquías del modelo conceptual y aplicarles las reglas elegidas por el usuario.

- logJerarquia(): se encarga de guardar en el log, que opción fue elegida dentro de las alternativas propuestas al usuario.

- **AddOpcionesEliminarAtrCompuesto**: esta clase es la encargada de crear una ventana para mostrarle al usuario las posibles alternativas para la eliminación de atributos compuestos. Por cada opción posible, tiene un check para elegir. Las diferentes alternativas propuestas son:

  - Opción A: Generar un único atributo que se convierta en la concatenación de todos los atributos simples que contiene el atributo compuesto.

  - Opción B: Definir los atributos simples sin un atributo compuesto que los resuma.

  - Opción C: Generar una nueva entidad, la que representa el atributo compuesto, conformada por cada uno de los atributos simples que contiene.

- **AddopcionJerarquiaMHoDToMP**: esta clase es la encargada de crear una ventana para mostrarle al usuario las posibles alternativas para la eliminación de jerarquías. Por cada opción posible, tiene un check para elegir. Las diferentes alternativas propuestas son:

  - Opción A: Eliminar las entidades hijas, dejando solo la entidad padre la cual incorpora todos los atributos de sus hijos.

  - Opción B: Conservar todas las entidades.

  - Opción C: Eliminar la entidad padre.

- **DeleteAtributoPolivalenteWizard**: esta clase es la encargada de crear una ventana para que el usuario elija que cardinalidad mínima le desea poner a la relación creada por la eliminación del atributo polivalente.

Ventana informativa:

Ventana que muestra el listado de atributos compuestos, atributos polivalentes y jerarquías que fueron tratadas en el pasaje. Tiene la función de informar el estado del pasaje de cada uno de los elementos.

Clases que colaboran para la realización de esta ventana:

- **Progreso**: es la clase base para el manejo de la ventana. Le pide información al ContadorLogico y le encarga a la TableViewer la tarea de volcar toda la información en una tabla.

- **ContadorLogico**: la característica principal de esta clase es la de proporcionar información a la ventana de progreso en donde se muestra lo realizado hasta el momento. Guarda el estado de todos los atributos compuestos, polivalentes y jerarquías. Los métodos mas importantes son los siguientes:

    - guardoCompuestos(): se encarga de guardar los atributos compuestos que se van creando en el modelo conceptual. Los guarda con estado 'No realizado'.

    - guardoPolivalentes(): se encarga de guardar los atributos polivalentes que se van creando en el modelo conceptual. Los guarda con estado 'No realizado'.

    - guardoJerarquía(): se encarga de guardar las jerarquías que se van generando en el modelo conceptual. Los guarda con estado 'No realizado'.

    - eliminoCompuesto(), eliminoPolivalente(), eliminoJeraquia(): se encargan de marcar como eliminados los atributos y jerarquías, para que en la ventana de progreso, se muestren con una cruz de eliminados.

    - restauroCompuesto(), restauroPolivalente(), restauroJerarquía(): se encargan de cambiar el estado a los elementos en la operación de deshacer.

- **TableViewer**: es una clase que tiene toda la funcionalidad necesaria para armar una vista agrupando el contenido en columnas y filas (figura 71).

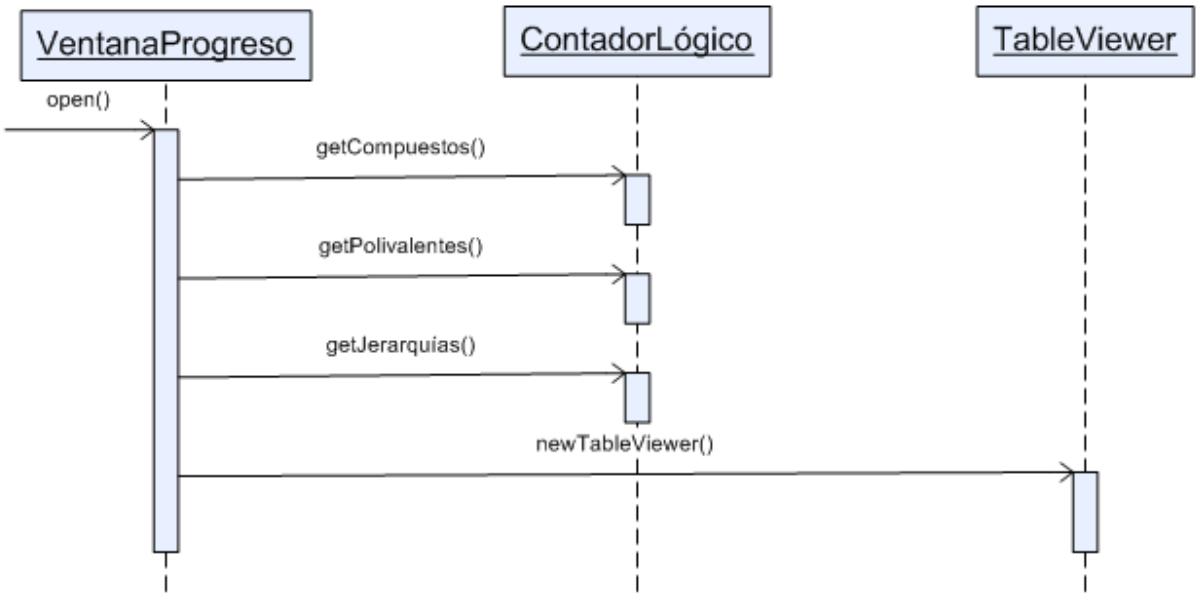

Figura 71. Ventana progreso

Proceso Deshacer:

- **ControladorDeshacer**: esta clase tiene como finalidad almacenar el estado de los atributos compuestos, polivalentes y jerarquías en un momento dado de manera que se pueda restaurar en ese punto de manera sencilla. Para ello se mantiene almacenado el estado del elemento para un instante de tiempo, de forma que ese recuerdo permita que el elemento sea modificado y pueda volver a su estado anterior. Teniendo esta clase, podemos restaurar el diagrama desde estados pasados. Además facilita el hacer y deshacer de las reglas de transformación al modelo Lógico. Siempre se respeta el orden en que se realizaron los cambios, es decir, se maneja la lista como una pila.

  Esta compuesto por una lista en donde se almacenan los estados de cada objeto a medida que sufren cambios.

Referenciamos los métodos más importantes para la administración de los cambios:

- agregarNuevo(): agrega un nuevo objeto con el estado antes de la modificación (esto permite recuperar el estado en cualquier momento).
- getUltimoDeshacer(): recupera el ultimo elemento modificado.
- hayUltimoDeshacer(): devuelve si hay alguna operación para deshacer.
- deshacerOperacion(): vuelve atrás la operación realizada, recuperando el último estado.

- **Deshacer**: esta clase representa a cada operación realizada. En ella se guarda el tipo de operación, la operación anterior y la operación siguiente, de esta manera se forma la lista encadenada de la que hablábamos anteriormente.
  - deshacerOperacion(): realiza la restauración de la operación a su estado anterior.
  - abrirModelo(): muestra el diagrama generado luego de la restauración.
  - loguear(): se encarga almacenar en el log, la descripción de lo que se deshizo para que quede registrado los cambios que se realizaron durante todo el pasaje.

- **ProcesoFisico**: es la clase encargada de llevar a cabo la transformación del Modelo Lógico al Modelo Físico. Para comenzar, primero verifica que la conversión al modelo lógico haya finalizado (figura 72). Sus principales métodos son los siguientes:
  - eliminoIdExternos(): invoca al ControladorIdentificadorExterno que es la clase que elimina los identificadores externos del modelo.
  - eleccionPasajeFisico(): abre una ventana con diferentes opciones de conversión para que el usuario elija. Las alternativas dadas son:
    1. Opción A: Generación automática de modelo físico (convención ideal).
    2. Opción B: Generación paso a paso de modelo físico.

Una vez elegida la opción, comienza la aplicación de reglas para la transformación. Recorre todos los elementos del modelo y solo trabaja con las relaciones encontradas. En base a las cardinalidades de las relaciones, aplica diferentes reglas.

A medida que se realiza el pasaje, se guarda en el log cada paso realizado, es decir, se describe la opción tomada y se muestra la tabla generada.

- pasoEntidadesAtabla(): se encarga de recorrer todos los elementos del modelo lógico buscando las entidades que no fueron tratadas durante el proceso anterior y las transforma en tabla. Cada una de las tablas generadas se guarda en el listado de tablas que posee la clase TablasFisicoDibujar.

- escriboTablas(): este método se encarga de hacer que se visualicen las tablas generadas en la nueva pestaña física y en el log físico. Para llevarlo a cabo invoca a dos metodos: armarArchivo() y armarArchivoLog();

- abroPestaña(): crea una nueva pestaña en la interfaz correspondiente al modelo físico y muestra las tablas resultantes. La nueva pestaña se crea utilizando un TextEditorFisico.

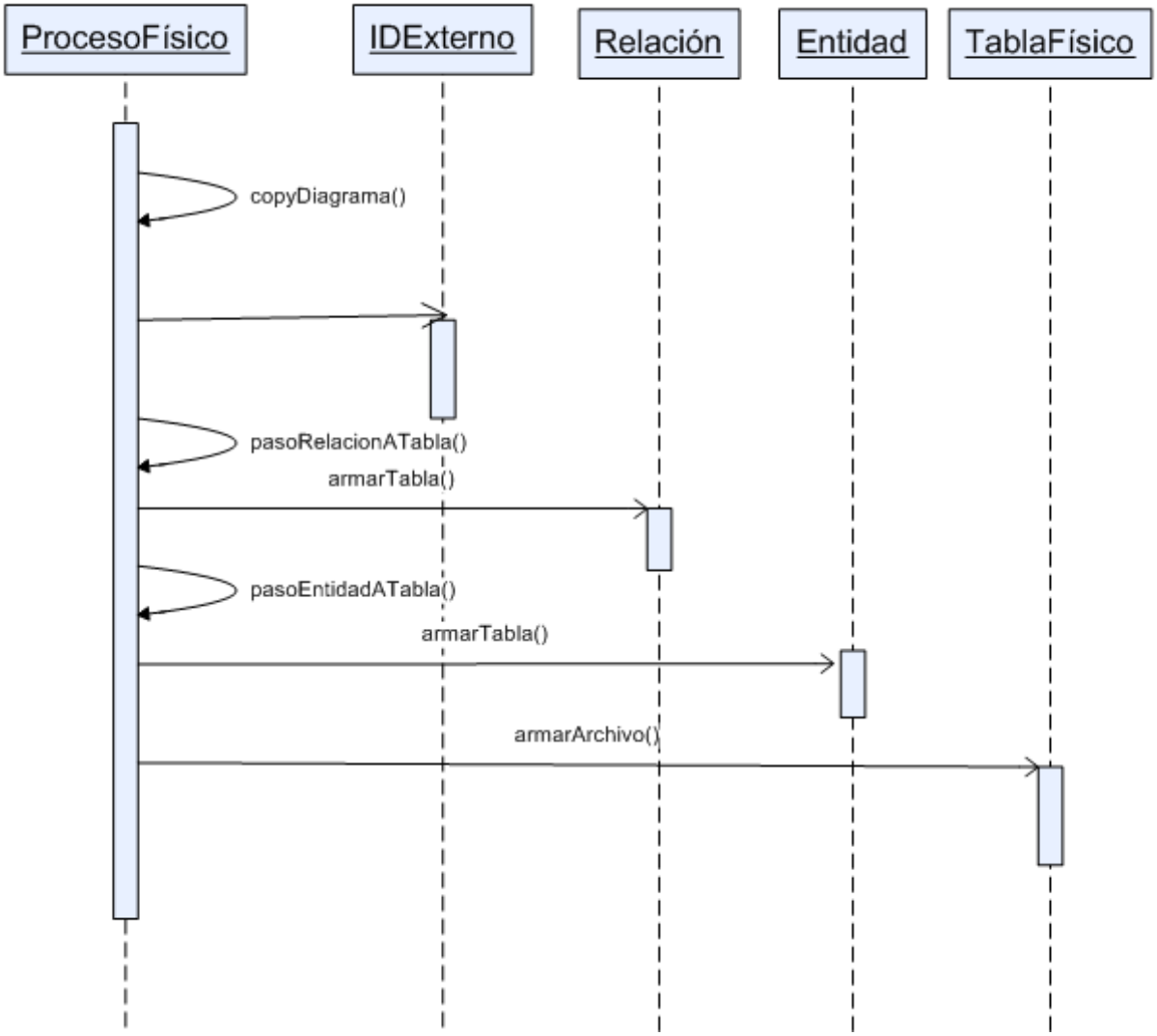

Figura 72. Proceso Físico

- **ControladorIdentificadorExterno**

  - inicioExterno(): marca el inicio del proceso de eliminación de identificadores externos.

  - ejecutoExterno(): recorre los elementos del modelo buscando identificadores externos. Toma cada identificador y lo pasa a otra entidad para que forme parte del nuevo identificador compuesto.

- inicioLogExterno(): escribe en el log el encabezado de la parte del proceso de eliminación de identificadores externos.

- logExterno(): escribe dentro del log, la operación realizada identificando las entidades que participan del identificador externo.

- finExterno: marca el fin del proceso.

- **ControladorPasajeTablas**

- **TablaFisico**: esta clase es un objeto que contiene todos los atributos necesarios para la generación de una tabla. Consta de un nombre, un conjunto de identificadores y un conjunto de atributos.

  - dibujartabla(): se encarga de la escritura de la tabla en el editor de texto que muestra las tablas generadas una vez finalizado el pasaje físico. Escribe las tablas de la siguiente manera:

    nombreTabla (IDENTIFICADOR, atributo1,atributo2, etc…)

- **TablaFisicoDibujar**: esta clase administra el almacenamiento y generación de las tablas creadas en la conversión al modelo físico.

  - guardoTabla(): este método se encarga de crear un objeto Tabla y guardar los atributos de la misma. Le llegan como parámetros el nombre de la tabla, una lista de identificadores, una lista de atributos. Una vez creada la tabla, la guarda en su lista de tablas.

  - armarArchivo(): recorre cada una de las tablas que se generaron y las ordena para que se escriban en el editor de texto con el método dibujoTablaPestanaFisica.

  - dibujoTablaPestanaFisica(): le ordena a la tabla elegida que se escriba invocando al método dibujoTabla().

- dibujoTablasLog(): recorre cada una de las tablas que se generaron y las escribe en el log referente al modelo físico.

- **FileOutHtml**: es la clase invocada para agregar información al log durante el proceso Lógico y Físico. Contiene la información agrupada de la siguiente manera:

  - textoCompuesto: guarda la información sobre la eliminación de atributos compuestos.
  - textoPolivalente: guarda la información sobre la eliminación de atributos polivalentes.
  - textoJerarquia: guarda la información sobre la eliminación de jerarquías.
  - textFisico: guarda información sobre la eliminación de identificadores externos, opciones tomadas y tablas resultantes.

Para finalizar, se presenta en la Figura 73 y 74 el diagrama de clases que corresponde al modelo de la implementación completo, con los atributos y métodos más relevantes de cada clase.

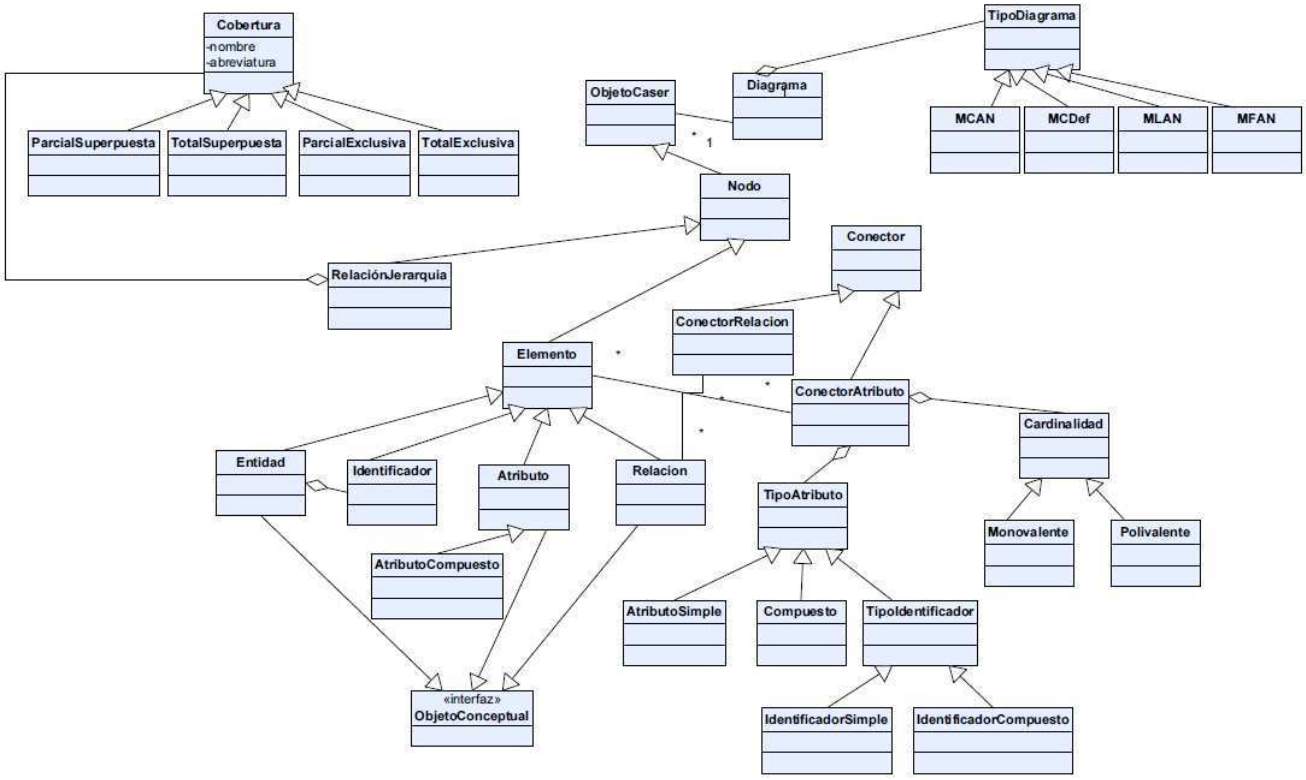

Figura 73. Diagrama de clases (Parte I)

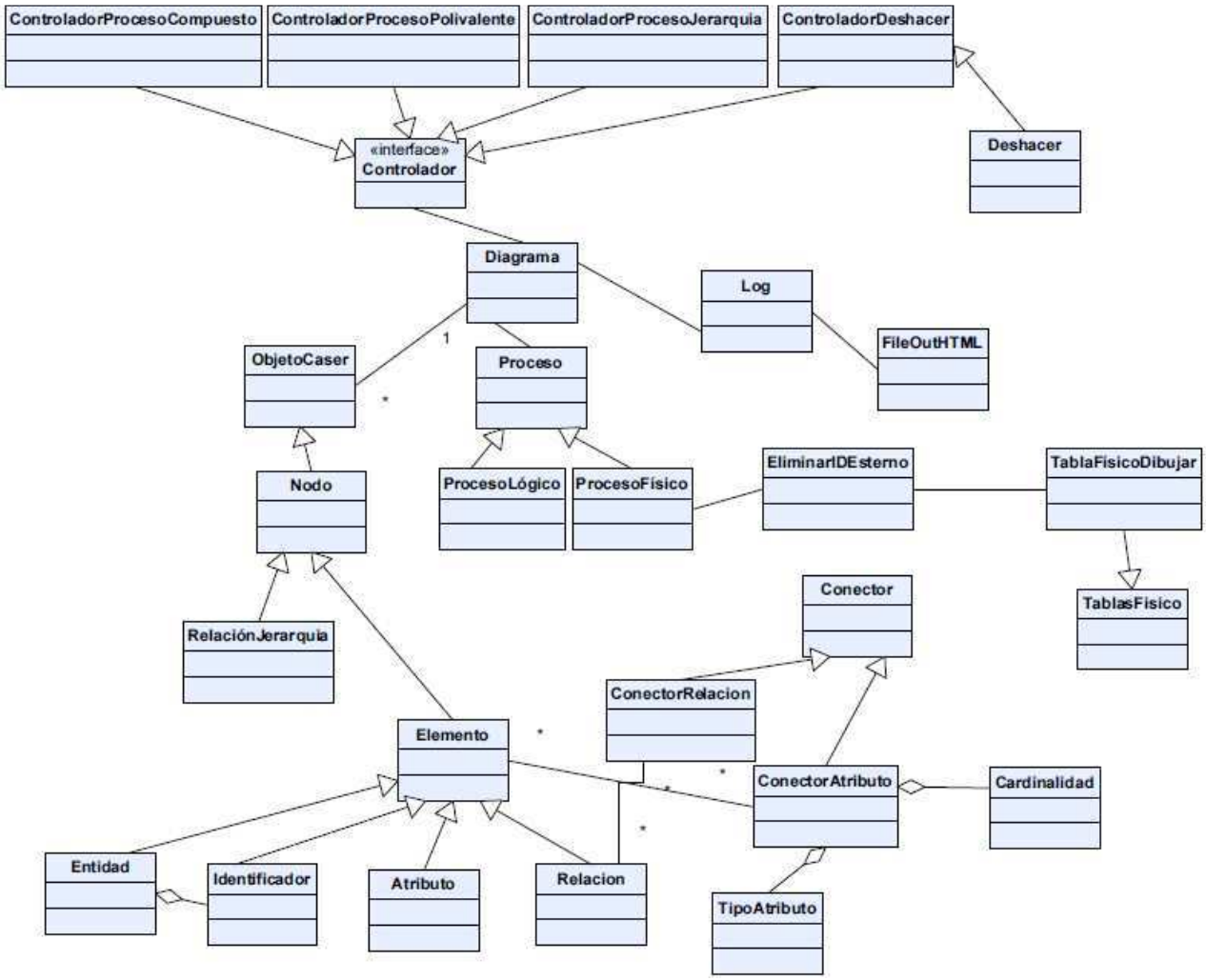

Figura 74. Diagrama de clases (Parte II)

CAPITULO 6

6.1 Conclusiones.

Las Bases de Datos actualmente constituyen un elemento imprescindible en cualquier organización. Por ende, es importante que el proceso de diseño se realice del mejor modo posible.

En la asignatura Introducción a las Bases de Datos de nuestra Unidad Académica, este tema es esencial, y los alumnos trabajan intensamente en casos prácticos a través de los cuales incorporan las pautas necesarias para diseñar Bases de Datos. Esta tarea se puede realizar de dos maneras; con lápiz y papel, iterando sobre gráficos que muchas veces se tienen que rehacer en forma completa; o utilizando alguna herramienta que actúe como un asistente en esta tarea.

En el año 2009 se creó (en una tesina de grado) la herramienta CASER 1.0 la cual asistía al alumno en el proceso de diseño conceptual. Esto constituyó un paso fundamental, a partir de una especificación de requerimientos y marcando palabras claves se puede generar automáticamente el modelo conceptual. Mientras que las tareas de diseño lógico y físico se completaban con el uso tradicional de papel y lápiz.

Esto motivó la creación de la versión 2.0 de la herramienta CASER, que permite completar el proceso de diseño en forma asistida, con la cual el diseño conceptual, el diseño lógico y el diseño físico se realiza sin utilizar papel y lápiz y en modo semiautomático.

En un entorno de modelado de datos, abstraer conceptos de la realidad, representarlos en un modelo gráfico, y luego obtener las tablas que representaran una base de datos es un desafío para cualquier estudiante.

En concordancia con su propósito educativo, la herramienta incorpora en todos los casos explicaciones a las tareas que realiza facilitando al alumno su entendimiento y haciendo hincapié en la incorporación de una metodología de trabajo para la solución de ejercicios de diseño de BD.

Todo esto conforma una ayuda al proceso de aprendizaje de un alumno.

Como conclusión podemos destacar las distintas ventajas que agregan potencialidad a esta nueva versión de la herramienta CasER:

- Asistencia al alumno en las distintas fases del modelado de datos brindando un esquema de trabajo ordenado para la transición entre los distintos modelos.
- La asistencia del software en las decisiones estructuradas del proceso de modelado de datos, jerarquiza los aportes que podrían realizar los docentes de clases prácticas a cuestiones de mayor peso.
- La herramienta también restringe al alumno limitando y brindando información sobre los flujos incorrectos que el alumno puede seleccionar durante cada una de las fases.
- El tiempo que demanda generar y editar esquemas de los modelos conceptual y lógico y tablas del modelo físico para los alumnos se ve disminuido de forma notable con la utilización de CasER 2.0, respecto de lo que demanda realizarlo manualmente con lápiz y papel.

Actualmente en el mercado (libre o comercial) no se encuentra un Software educativo con estas características, dado que todos los asistentes disponibles comienzan desde el diseño lógico, y evitan el diseño conceptual tan necesario para quienes intentan construir una Base de Datos sin experiencia previa, y tan aconsejable para quienes ya tienen experiencia.

6.2 Trabajos Futuros

Se pretende utilizar la herramienta para todo el proceso de diseño de BD, y obtener retroalimentación de los alumnos respecto a la usabilidad y las mejoras posibles en ese sentido.

Se prevé además la incorporación de dominios desde el modelo conceptual, lo que permitirá construir modelos físicos con tipos de datos incluidos.

Finalmente también se pretende disponer de la generación automática de los script en lenguaje SQL.

6.3 Bibliografía

[1] Batini, Navathe, Cieri. Diseño Conceptual de Bases de Datos: un enfoque de entidades-interrelaciones. Addison Wesley 1991.

[2] R Bertone, P Thomas, Introducción a las Bases de Datos, Fundamentos y Diseño. Editorial Pearson, ISBN 978-987615136-8. Mayo 2011

[3] Date. Introducción a los sistemas de Bases de Datos. Addison Wesley. 1994.

[4] Elmasri, Navathe. Fundamentos de Sistemas de Bases de Datos. Pearson-Addison Wesley. 2002.

[5] Korth-Silberschatz. Fundamentos de Bases de Datos. McGraw Hill. 1998.

[6] Kroenke. Procesamiento de Bases de Datos. Prentice Hall. 1996.

[7] Matthew Scarpino, Stephen Holder, Stanford Ng, Laurent Mihalkovic. SWT/JFace in action. Manning. 2005.

[8] Miguel Angel Abián. El archipiélago Eclipse (parte 3 de 4). 2005.

[9] Miguel Piattini. Fundamentos y modelos de Bases de Datos. Rama. 1998.

[10] Leite, J.C.S.P., Application Languages: A Product of Requirements Analysis, Departamento de Informática, PUC- /RJ, January 1989.

[11] Jacobson I. et al. Object Oriented Software Engineering: AUse-Case Driven Approach. Addison Wesley, 1992

[12] G. Booch, J. Rumbaugh, y I. Jacobson. The Unified Modeling Language User Guide. Addison–Wesley, 1999.

[13] Eclipse - http://www.eclipse.org.

[14] Erwin - http://www.ca.com/us/database-design.aspx

[15]POWERDESIGNER-http://www.sybase.com/products/modelingdevelopment/powerdesigner.

[16] Workbench - http://www.sql-workbench.net

[17] R. Bertone, P. Thomas, S. Antonetti, A. Miglio, “Herramienta para la enseñanza de Modelado Conceptual de Bases de Datos”, Congreso Argentina de Ciencias de la Computación. Octubre 2009, Jujuy

[18] S. Antonetti, A. Miglio, “CasER 1.0: Herramienta para la enseñanza de Modelado Conceptual de Bases de Datos”, Tesina de Grado 2009, Facultad de informática - UNLP

[19] P. Thomas, R. Bertone, A. Duran, M.F. Rius, “CasER 2.0: Herramienta para la enseñanza de Modelado de Bases de Datos”, Congreso Argentina de Ciencias de la Computación (CACIC) Octubre 2011, La Plata