Universidad Rey Juan Carlos

Escuela Superior de Ciencias Experimentales y Tecnología

Departamento de Informática, Estadística y Telemática

MIDAS/DB: Una Metodología basada en Modelos para el Desarrollo de la Dimensión Estructural de Sistemas de Información Web

TESIS DOCTORAL

Belén Vela Sánchez

2003

Agradecimientos

Quiero hacer un reconocimiento especial a mi directora de tesis, Esperanza Marcos Martínez. Sin tu ayuda, motivación y dedicación (esos numerosos fines de semana que te he hecho trabajar por esta tesis) no hubiera sido posible la realización de este trabajo. He aprendido muchísimo de ti y espero poder seguir haciéndolo. Por todo esto quiero darte las gracias y decirte que puedes contar conmigo ;-).

También quiero agradecer a todo el grupo de investigación Kybele (a Josemari, Paloma y Valeria) sus comentarios y sobre todo su gran ayuda en los momentos finales de la tesis.

A mis niños del laboratorio, Juancho, Elena, Cristina, Raúl y Javi, por ayudarme con esos “marroncillos” de última hora.

Quiero darle especialmente las gracias a mis padres, por estar siempre ahí, por darme tanto amor, por confiar en mí, por animarme en todo momento a seguir adelante y ”aguantarme” hasta cuando estaba inaguantable. Gracias, Papis.

Finalmente quiero agradecer a todos mis amigos su apoyo y confianza en mí. Especialmente quiero dar las gracias a Susana por su apoyo incondicional, por esas largas conversaciones llenas de quejas y lamentaciones y por animarme a alguna que otra salida. Gracias, Susa.

Gracias

Índices

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

1.1 Planteamiento y Justificación del Trabajo

En la última década la Web se ha consolidado como uno de los principales medios para compartir y difundir información a nivel mundial. La gran mayoría de las empresas y organizaciones han tenido que adaptarse a este nuevo entorno, inicialmente publicando su información en la Web, y en la actualidad extendiendo incluso su capacidad de negocio mediante la utilización de este medio. Este es el motivo por el que ha surgido la necesidad de técnicas y metodologías de Ingeniería del Software adaptadas al desarrollo específico de Sistemas de Información Web (SIW).

Las primeras propuestas metodológicas eran, en general, adaptaciones de metodologías clásicas y estaban enfocadas a un tipo concreto de desarrollo. Así, por ejemplo, los trabajos propuestos en Garzotto *et al.* (1993), Isakowitz *et al.* (1995), Isakowitz *et al.* (1998), Lowe y Hall (1999) y Schwabe y Rossi (1998) proporcionan modelos de datos y pasos metodológicos para guiar en el proceso de diseño de aplicaciones hipermedia y multimedia en la Web. Por otro lado, en Atzeni *et al.* (1998), Fraternali y Paolini (1998), Fraternali (1999) y Mecca *et al.* (1999) se proponen aproximaciones para el desarrollo de bases de datos (BD) accesibles a través de la Web. También han surgido extensiones de metodologías de desarrollo de propósito general como Conallen (2000) y Gómez *et al.* (2001), así como metodologías genéricas que permiten abordar diversos tipos de aplicaciones mediante la adaptación de su proceso de desarrollo (Bonifati *et al.* (2000); Castano *et al.* (2000)). En la actualidad, se habla ya de un nuevo área de trabajo que se denomina Ingeniería del Software para la Web (Web Engineering) y cuyas aportaciones surgen, directamente, con el objetivo de facilitar el desarrollo sistemático y (semi-) automático de SIW.

Las diferentes propuestas metodológicas que han ido surgiendo han estado también muy relacionadas con el tipo de aplicaciones que la Web soportaba en cada momento. Inicialmente, la Web se utilizaba básicamente como medio para publicar y consultar información que se organizaba en páginas HTML (HyperText Markup Language). Sin embargo, al aumentar la cantidad de información a publicar, esta

forma de organización comienza a dar problemas, debido fundamentalmente a los siguientes motivos:

- Por una parte, la actualización de los datos en las páginas Web resulta tediosa, ya que éstos se encuentran integrados con el propio hipertexto y con la presentación del mismo.
- Por otra parte, es habitual que, a fin de facilitar la legibilidad y navegación, haya información duplicada en distintas páginas, lo que origina los ya conocidos problemas de mantenibilidad: dificultad en encontrar y relacionar datos; las actualizaciones deben realizarse tantas veces como aparezca el dato en la Web; inconsistencias; información obsoleta, etc.

Un modo de solucionar este tipo de problemas consiste en la utilización de BD Web, permitiendo que las páginas Web generen su contenido dinámicamente a partir de información extraída de la BD. De este modo se eliminan las redundancias e inconsistencias anteriormente mencionadas. Además, la aparición de la tecnología XML (eXtensible Markup Language) ha permitido también separar los aspectos de presentación, de modo que un cambio en la presentación de las páginas no afecte ni al contenido, ni a la organización de las mismas. Todos estos aspectos relacionados con la estructuración, tanto interna como externa, de la información en la Web, es lo que se denomina *dimensión estructural* e incluye:

- El **contenido**, es decir, la organización interna de los datos a publicar.
- El **hipertexto**, es decir, la organización externa de este contenido: qué datos se muestran y en qué páginas, cómo se navega a través de ellos, etc.
- La **presentación**, es decir, la composición de cada página y la interacción con el usuario.

En la Figura 1 se muestra una clasificación de tres dimensiones ortogonales entre sí que, según Retschitzegger y Schwinger (2000), se deben considerar en el modelado de un SIW: *aspectos, niveles* y *fases*. Como se puede ver (eje z), es necesario tener en cuenta *aspectos* de estructura y de comportamiento. Para cada uno de estos aspectos hay que considerar tres *niveles* (eje y): el nivel de contenido, el nivel de hipertexto y el nivel de presentación. El nivel de contenido se

correspondería con el concepto de una base de datos tradicional (BD estructurada), mientras que el nivel de hipertexto representaría la forma en que los datos (semi-estructurados) son agrupados y enlazados. Finalmente, el nivel de presentación, se refiere a los aspectos de visualización de los datos y de interacción con el usuario. Para cada uno de estos aspectos y niveles se consideran distintas *fases* (eje x), desde modelado conceptual hasta implementación, según el nivel de abstracción que se desee representar.

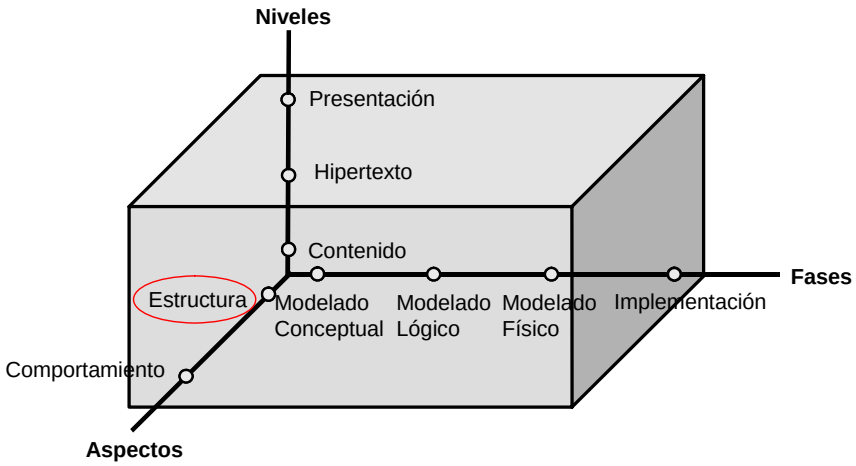


Figura 1. Dimensiones de Modelado

La tesis que se presenta (a la que en adelante nos referiremos como MIDAS/DB) se centra en el desarrollo de los aspectos estructurales de un SIW y, concretamente, en dos de ellos: contenido e hipertexto. MIDAS/DB se basa en el uso de estándares en el proceso de desarrollo del SIW. Dado que UML (Booch *et al.,* 1999b), es el lenguaje estándar para el desarrollo de sistemas orientados a objetos, se propone usar este lenguaje para modelar todo el sistema. Además, se propone la utilización de XML (Bray *et al.,* 2000; W3C, 2001a; W3C, 2001b), por los beneficios que esta tecnología presenta en cuanto a separación de estructura, presentación y contenido, intercambio de información e integración. En cuanto al modelo de persistencia de datos, se propone la utilización de BD (objeto-) relacionales, debido a su implantación y aceptación en el mercado de la tecnología relacional, así como a las facilidades proporcionadas por las extensiones de objetos. Además, también es necesario considerar la gestión de datos semi-estructurados, no sólo porque para la dimensión del hipertexto se está considerando tanto la utilización de tecnología XML como que los documentos XML tendrán que ser

almacenados y gestionados, sino también porque el propio contenido de la BD podría estar almacenado en una BD XML.

Hoy en día, los Sistemas de Gestión de Bases de Datos (SGBDs) soportan el almacenamiento y gestión de datos estructurados, semi-estructurados y no estructurados. De este modo, han aparecido BD XML nativas puras, como Tamino o eXist, Chaudhri *et al.* (2003), así como algunas propuestas que integran datos estructurados, semi-estructurados y no estructurados, como ToX (Barbosa *et al.,* 2001). Éste también es el caso de algunos SGBD (objeto)-relacionales como, por ejemplo, Oracle 9*i* Release 2 que incorpora nuevas características, agrupadas bajo el nombre de XML DB, que aúnan el contenido XML y los datos estructurados, Oracle Corporation (2003).

El modelado, así como el proceso de desarrollo, variará dependiendo de la tecnología de almacenamiento de datos seleccionada. Teniendo en cuenta los avances de la tecnología y centrándonos en los aspectos estructurales, es objetivo de este trabajo proporcionar técnicas para abordar la construcción de la dimensión estructural de un SIW de un modo sistemático. Este trabajo está centrado en dos aspectos estructurales: hipertexto y contenido.

1.2 Hipótesis y Objetivos

A continuación, se pasa a exponer la hipótesis y los objetivos que se han fijado al comienzo del trabajo de investigación que se ha llevado a cabo en esta tesis doctoral:

La **hipótesis** planteada al comienzo de este trabajo es:

"*Es factible la especificación de una metodología basada en modelos y apoyada en estándares que facilite el desarrollo sistemático de la dimensión estructural de Sistemas de Información Web (SIW)*".

El **objetivo principal** de este trabajo de investigación, derivado directamente de la hipótesis, es:

"*La especificación de una metodología orientada a modelos y basada en el uso de estándares, que facilite el desarrollo sistemático de la dimensión estructural de un SIW. La metodología incorporará las mejores prácticas de otras aproximaciones, adaptándolas si se considera oportuno, y definirá nuevas técnicas, notaciones, etc. cuando sea necesario*".

Para la consecución de este objetivo se han planteado los siguientes objetivos parciales:

1. Analizar las distintas metodologías existentes para el desarrollo de SIW, concretamente, las que se centran en los aspectos estructurales (BD Web e hipertexto), determinando sus aportaciones, así como sus limitaciones, con el fin de abordar problemas no resueltos e incorporar las mejores prácticas de otras metodologías.
2. Definir el marco metodológico para el desarrollo de la dimensión estructural de un SIW, especificando el proceso, las actividades, las tareas, técnicas y notaciones a utilizar.
3. Definir las extensiones[1] necesarias de UML para poder representar todo el sistema en una única notación.
4. Definir las guías para la transición entre los modelos propuestos para cada una de las fases del desarrollo.
5. Validar las extensiones de UML, así como las guías de transformación propuestas, mediante la implementación de las mismas en ADD-INs de Rational Rose y su aplicación a distintos casos de prueba.
6. Validar el marco metodológico mediante su aplicación a distintos casos de prueba.

Es importante resaltar que en ningún momento ha sido objetivo de esta tesis el buscar soluciones nuevas a problemas ya resueltos. Por ello, al comienzo de este trabajo se ha realizado un estudio de las metodologías para el desarrollo de SIW existentes detectando, tal y como se expone en el capítulo 2 que había problemas no resueltos. Por ello, MIDAS/DB se concibió como un marco que incluyera las mejores prácticas de otras metodologías y que aportara soluciones a problemas no resueltos. Los principales problemas detectados al comienzo del trabajo fueron: la integración contenido-hipertexto; UML como notación única de todo el sistema; utilización de tecnología OR y XML. A lo largo de la realización de esta tesis, han aparecido también algunas soluciones a estos problemas; cuando estas soluciones han sido satisfactorias han sido incorporadas, bien directamente, o bien con adaptaciones a MIDAS/DB.

[1] Lo que en este trabajo se ha denominado extensión de UML se corresponde con los conocidos *UML Profiles*. Se ha optado por utilizar un término en castellano que describiese este concepto, no considerándose adecuado el uso de la traducción literal del mismo.

1.3 Marco de Trabajo

La realización de este trabajo de investigación se está llevando a cabo dentro del grupo de investigación Kybele de la Universidad Rey Juan Carlos (URJC). Además, se ha realizado una estancia de cuatro meses para trabajar con el grupo de investigación DBTG (The DataBase Technology Group) del Instituto de Informática de Universidad de Zürich (Suiza), dirigido por el profesor Dr. Klaus R. Dittrich.

Dentro del grupo Kybele, la tesis que se presenta se integra dentro de distintos proyectos de investigación relacionados entre sí (ver Figura 2). La tesis comenzó, y se realizó casi en su totalidad, en el marco de los proyectos de investigación MIDAS y HERA. MIDAS (1999-2002) es un proyecto cofinanciado por el Ministerio de Ciencia y Tecnología (MCYT) y La Unión Europea [2FD97-2163], cuyo objetivo era la especificación de una metodología para el desarrollo de SIW. Este proyecto fue también financiado por la URJC [PGRAL-2001/05]. El proyecto HERA (2001-2002) se llevó a cabo de forma solapada en el tiempo con MIDAS. HERA consistía en la generación de una herramienta para el desarrollo rápido de portales WAP y se llevó a cabo junto con la empresa Intesys S.A., financiado por la Comunidad de Madrid a través de su convocatoria de ayudas a Pymes [CAM: 09/0194/2000]. También en el proyecto MIDAS existió una colaboración con la empresa Intesys S.A. La metodología resultante de MIDAS se complementa con la herramienta construida en HERA para el desarrollo de portales accesibles desde distintas plataformas.

Posteriormente, se continuó el trabajo en el marco de nuevos proyectos (DAWIS y EDAD) que son la continuación de MIDAS Y HERA. DAWIS (2002-2005) es un proyecto financiado por el Ministerio de Ciencia y Tecnología [TIC 2002-04050-C02-01] y en el que, además del grupo Kybele de la URJC donde reside la coordinación del proyecto, participan el grupo de BD de la UPM, ICM (Informática de la Comunidad de Madrid) y CRC Technologies. Dentro de DAWIS, el subproyecto de la URJC se centra en el *desarrollo sistemático y semi-automático de portales Web de acceso integrado a múltiples archivos digitales*, para lo que se está trabajando en la adaptación de MIDAS, y MIDAS/DB, por una parte para el domino de archivos digitales y, por otra, para el desarrollo de portales de acceso integrado mediante la instanciación de una arquitectura de referencia. Este proyecto fue también financiado por la URJC [PIGE 2002/05]. EDAD (2003-2004), *Entorno para*

el Desarrollo e Integración Automática de Archivos Digitales en la Web, pretende ofrecer el soporte de automatización a las propuestas metodológicas de DAWIS. Se trata de un proyecto cofinanciado por la Comunidad de Madrid y por la Unión Europea [07T/0056/2003 1] en el que participa, además de los grupos y entidades de DAWIS, el grupo de Archivos y Documentación de la UC3M.

En el marco de dichos proyectos se están llevando a cabo varias tesis doctorales. En concreto, este trabajo se centra en los aspectos metodológicos del desarrollo de la dimensión estructural del SIW. El marco metodológico, está siendo definido en la tesis doctoral de Dª Paloma Cáceres, y plantea una aproximación orientada a modelos y basada en MDA (*Model Driven Architecture*), Miller y Mukerji (2001), integrando en el proceso de desarrollo prácticas procedentes de metodologías ágiles, principalmente de eXtreme Programming, Beck (1999). El aspecto de comportamiento es objeto de la tesis doctoral de Dª Valeria de Castro y se está siguiendo una aproximación basada en portales y servicios Web.

Para finalizar, se explica la relación de estos proyectos con el trabajo realizado durante la estancia de la doctoranda en el grupo de investigación DBTG del Instituto de Informática de Universidad de Zürich (Suiza). Durante los cuatro meses de estancia la doctoranda trabajó en el proyecto europeo e-Parking: User-Friendly e-Commerce to Optimise Parking Space [IST-2000-25392], realizando la especificación de una federación de bases de datos para la Web y colaborando en la implementación de un prototipo. En la actualidad, la experiencia en este proyecto se está aplicando, en el marco de DAWIS y EDAD, para la adaptación de MIDAS/DB al desarrollo de portales de acceso integrado a múltiples archivos digitales.

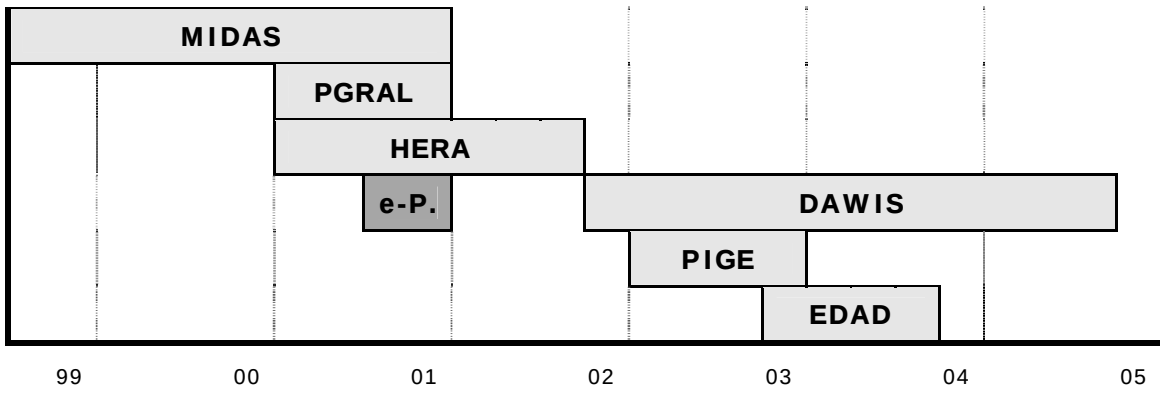


Figura 2. Marco de la tesis

1.4 Método de Investigación

El método de investigación utilizado para la realización de esta tesis doctoral se basa en el método para investigación en Ingeniería del Software propuesto, como una adaptación del método de Bunge (1976), en Marcos y Marcos (1998) y en el método denominado Investigación en Acción (*Action Research*), Avison *et al*. (1999). En Marcos y Marcos (1998) se propone un método con una serie de etapas en la investigación que por su generalidad, son aplicables, con ciertas modificaciones, a cualquier tipo de investigación (Figura 3).

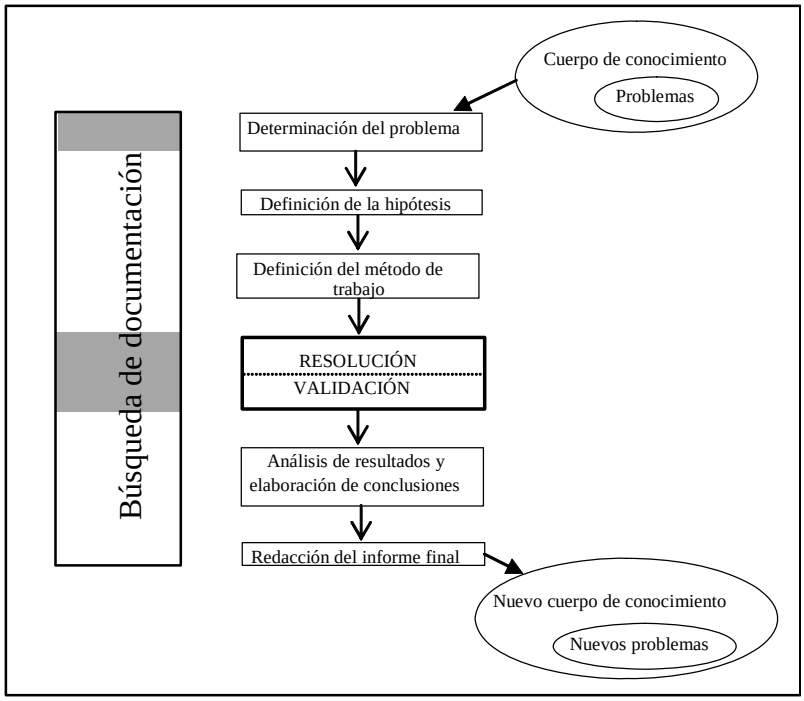


Figura 3. Método de Investigación

Tal y como se aprecia en la Figura 3, en el método de investigación se incluye una etapa que constituye la definición del propio método. Esto es debido a que cada investigación tiene sus propias características y esto hace que no exista un método universal que pueda ser aplicable, directamente y sin modificaciones, a cualquier trabajo de investigación. Dependiendo de la naturaleza del problema se utilizará un método deductivo, experimental, etc. De hecho, el método de resolución concreto de esta tesis es una combinación de Investigación en Acción, casos de estudio y casos de prueba.

1.4.1 Método de Resolución y Validación

La Investigación en Acción es un método de investigación cualitativo utilizado para la validación de los trabajos de investigación mediante su aplicación en proyectos reales. Este método, por su carácter de validación práctica, es especialmente apropiado para la investigación en Ingeniería y, específicamente, en Ingeniería del Software. Los casos reales se han utilizado en dos sentidos: como casos de estudio, a fin de encontrar las necesidades en el desarrollo de SIW; y como casos de prueba, a fin de validar y refinar tanto la metodología propuesta, como las técnicas. Todo ello en un proceso cíclico, tal y como se propone en Investigación en Acción.

El método de Investigación en Acción es "la forma que tienen los grupos de personas para preparar las condiciones necesarias para aprender de sus propias experiencias y hacer estas experiencias accesibles a otros", Mc Taggart (1991). Más concretamente, puede definirse como "el proceso de recopilar de forma sistemática datos de la investigación acerca de un sistema actual en relación con algún objetivo, meta o necesidad de ese sistema; de alimentar de nuevo con esos datos al sistema; de emprender acciones por medio de variables alternativas seleccionadas dentro del sistema, basándose tanto en los datos como en las hipótesis; y de evaluar los resultados de las acciones, recopilando datos adicionales", French y Bell (1996).

Una de las características más relevantes de este método es que la investigación se realiza a la vez que se aplican sus resultados, de modo que esta aplicación permite validar y refinar los resultados de la investigación en un proceso cíclico. Habitualmente la aplicación de los resultados se lleva a cabo en proyectos reales, en el marco de una empresa u organización que sirva de grupo crítico de referencia. Debido a la dificultad de encontrar empresas que permitan aplicar los resultados de una investigación en proyectos reales, se ha optado por una adaptación del método, aplicándolo a casos de estudio que se han definido dentro del grupo de investigación. Parte del grupo participa como grupo investigador, mientras que otra parte juega el rol de grupo crítico de referencia. Además, se han utilizado casos de prueba para validar las técnicas propuestas.

La Figura 4 representa, muy resumidamente, la aplicación del método de Investigación en Acción a la realización de la presente tesis doctoral, en un proceso iterativo que se va realimentando por medio de su aplicación a diferentes casos de

estudio. En la Figura 4 se reflejan los diferentes actores que, según Wadswroth (1998), intervienen en el proceso. Algunos de estos actores pueden coincidir en determinados casos:

- El **investigador**, que es aquel que impulsa como sujeto el proceso investigador (la doctoranda).
- El **objeto investigado**, que es el problema que se desea resolver (en este caso, la metodología objeto de la tesis).
- El **grupo crítico de referencia**, es decir, aquel para quien se investiga y que participa en la investigación, en nuestro caso el propio grupo de investigación Kybele, incluyendo a los desarrolladores (becarios y alumnos que realizan su proyecto fin de carrera). Un proceso cíclico refinará todos los resultados obtenidos de la aplicación a los distintos casos de estudio.

  A pesar de que la directora de la tesis y la doctoranda son investigadoras, y por este motivo quizás no deberían formar parte del grupo crítico de referencia, se ha decidido que participen en los dos grupos. Esto realmente no es contradictorio con el método de Investigación en Acción, en el que sí se permite que algunos de los actores coincidan. De este modo se mejora la comunicación entre ambos grupos y se consigue una mayor realimentación. Tal y como se propone en Marcos (2003), el proceso de investigación en Ingeniería del Software tiene muchos paralelismos y similitudes con el proceso de desarrollo software y, en concreto, esta modificación de Investigación en Acción se basa en una de las prácticas propuestas en eXtreme Programming (XP): el cliente en el equipo, Beck (2002).
- **Aquél para quien se investiga**, en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participe directamente en el proceso. En nuestro caso, cualquiera que pueda verse beneficiado por la aplicación de la metodología para el desarrollo de la dimensión estructural de SIW, así como los usuarios de los SIW desarrollados. Por ejemplo, uno de los casos de estudio desarrollados ha sido la Web del grupo de investigación Kybele; de este resultado se beneficia, tanto el propio grupo, como los alumnos de la URJC, usuarios de las páginas del grupo, etc.

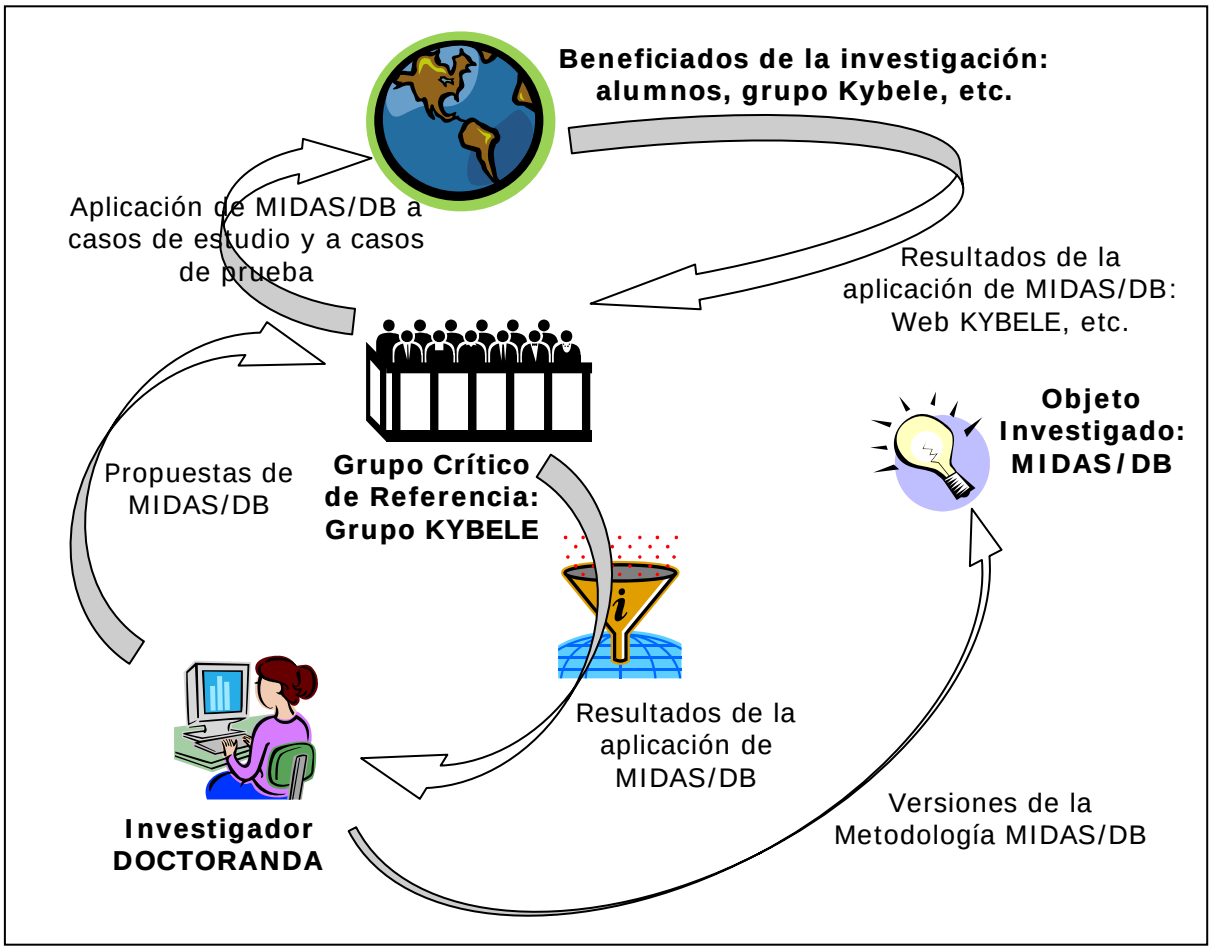


Figura 4. Aplicación del método de Investigación en Acción

Como ya se ha dicho, para la definición de la metodología MIDAS/DB se han utilizado distintos casos de estudio y de prueba.

Los **casos de estudio** han permitido mediante el desarrollo de casos reales, utilizando, cuando ha sido posible, prácticas de metodologías existentes, especificar el marco metodológico, proceso y técnicas requeridas. Los casos de estudio utilizados han sido los siguientes:

- CASO 1: Reserva de ordenadores y aulas informáticas
- CASO 4: Web del grupo de investigación Kybele
- CASO 6: Casas Rurales

Los **casos de prueba** han permitido realizar la validación de las distintas propuestas de la metodología y han sido los siguientes:

- CASO 1: Reserva de ordenadores y aulas informáticas
- CASO 2: Proyectos Arquitectónicos
- CASO 3: Libros de Recetas de Cocina
- CASO 4: Web del grupo de investigación Kybele
- CASO 5: Ejemplos Internet
- CASO 6: Casas Rurales
- CASO 7: Cine-Entradas

Todos estos casos se explican con más detalle en el capítulo 4 de esta tesis.

1.5 Organización de la Tesis

La organización de los restantes capítulos de esta tesis doctoral es la siguiente:

- **Capítulo segundo:** Se resume el estado actual de las metodologías de desarrollo de SIW, resaltando sus puntos fuertes y débiles.
- **Capítulo tercero:** En este capítulo se presenta la propuesta de metodología basada en modelos para el desarrollo de la dimensión estructural de un SIW que es objeto de esta tesis.
- **Capítulo cuarto:** Se presenta la validación de la metodología, mediante su aplicación a diferentes casos de estudio y casos de prueba, así como la implementación de las extensiones de UML como ADD-INs de Rational Rose.
- **Capítulo quinto**: En este capítulo se presenta el análisis de los resultados, las principales aportaciones del trabajo, la contrastación de resultados en distintos foros tanto nacionales como internacionales, así como algunas líneas de investigación abiertas.
- **Bibliografía, lugares de Internet**: Se incluye al final de la tesis una extensa bibliografía que contiene todas las referencias mencionadas en esta tesis, así como otro material y lugares de Internet utilizados durante la realización de la misma.

- **Apéndices:** Finalmente, se incluye un apéndice con la lista de siglas utilizadas, otro con un resumen de las extensiones de UML propuestas, un tercer apéndice con los casos de estudio y casos de prueba desarrollados y finalmente, otro apéndice con las reglas de transformación para pasar de un modelo conceptual a DTDs.

Estado del Arte

2.1 Visión Histórica

A principios de los años 90, la Web se desarrolló en el CERN (*Conseil Europeen pour la Reserche Nucleare*) como un sistema de información hipermedia que permitiera a los científicos compartir información. La primera utilidad de la Web fue la de publicar información que pudiera ser consultada por cualquier persona y desde cualquier lugar del mundo (Elmasri y Navathe, 2000). Esta información se publica habitualmente en forma de páginas HTML (HyperText Markup Language) que generalmente pueden contener datos en formato multimedia (audio, vídeo, etc.). Cada página puede tener enlaces a otras páginas (*hyperlinks*) permitiendo al usuario "navegar" de una página a otra. Cuando el número de páginas y de enlaces es elevado, el diseño de la estructura de navegación se convierte realmente en el diseño de una aplicación hipermedia, con la única diferencia de que esta aplicación se ejecutará en la Web.

Este es el motivo por el que las primeras propuestas metodológicas para el desarrollo de Sistemas de Información Web (SIW), proceden del campo de la ***hipermedia y multimedia***: HDM (Garzotto *et al.*, 1993), RMM (Isakowitz *et al.*, 1995; Isakowitz *et al.*, 1998), OOHDM (Schwabe y Rossi, 1998), la propuesta de Lowe y Hall (1999) son algunos ejemplos. HDM-lite (Fraternali y Paolini, 1998) es una adaptación de HDM para el desarrollo de aplicaciones hipermedia y multimedia en la Web. Debido a que UML se ha convertido en el lenguaje estándar de modelado para sistemas orientados a objetos, cabe también destacar algunos trabajos en el campo de la hipermedia y multimedia, basados en la representación del sistema con UML (Baresi *et al.*, 2001; Baumeister *et al.*, 1999; Koch *et al.*, 2000).

Sin embargo, estos trabajos proporcionan modelos de datos y pasos metodológicos para guiar en el proceso de diseño de aplicaciones hipermedia y multimedia, pero no tienen en cuenta el diseño y mantenimiento de los datos. Tal y como se indicó en la sección anterior, en la medida en que la cantidad de información publicada en la Web aumenta, se hace más difícil realizar un buen mantenimiento de la información contenida en las páginas HTML (Atzeni *et al.*, 1998). Por este motivo, surge la necesidad de crear aplicaciones de BD en la Web y

con ello surgen nuevas propuestas metodológicas, orientadas fundamentalmente al desarrollo y mantenimiento de ***BD Web***: Araneus (Atzeni *et al.*, 1998; Mecca *et al.*, 1999); Autoweb (Fraternali y Paolini, 1998).

Además de los trabajos anteriormente mencionados, cabe citar algunas propuestas metodológicas que aunque no son exclusivas para el desarrollo de la dimensión estructural de los SIW, pueden ser también útiles para este tipo de desarrollos. Estos trabajos podrían clasificarse en dos grandes grupos:

a) Marcos metodológicos que permiten abordar cualquier tipo de desarrollo (y entre ellos el de BD Web) mediante la adaptación del proceso propuesto. Este es el caso de WISDOM (Castano *et al.*, 2000), MIDAS (Marcos *et al.*, 2002b) o WebML, que considera aplicaciones Web y con acceso móvil (Bonifati *et al.*, 2000).

b) Extensiones de metodologías de desarrollo clásicas, entre las que podríamos destacar las extensiones de Conallen a UML (Conallen, 2000), OO-*H*method (Gómez *et al.*, 2001), etc.

Algunas de estas metodologías llevan asociadas herramientas que facilitan el desarrollo. En Fraternali (1999) puede encontrarse un estudio detallado de las principales herramientas para el desarrollo de aplicaciones Web.

Desde otra perspectiva, en Retschitzegger y Schwinger (2000) se presenta un estudio comparativo de los principales métodos de modelado de SIW existentes en la actualidad. En este trabajo, tal y como se aprecia en la Figura 5, se distinguen tres generaciones en las que aparecen enmarcadas varias de las metodologías que se han citado previamente. La primera generación corresponde a las metodologías para desarrollo de aplicaciones hipermedia (HDM, Garzotto *et al.* (1993)). En la segunda generación, se adaptan los métodos de hipermedia en tres sentidos: incorporando aspectos característicos del desarrollo Web (HDM-lite, Fraternali y Paolini (1998)); incorporando aspectos de metodologías clásicas de desarrollo software (OOHDM, Schwabe y Rossi (1998)); e incorporando aspectos de metodologías clásicas de desarrollo de BD (RMM, Isakowitz *et al.* (1995) e Isakowitz *et al.* (1998)). En la tercera generación, las metodologías contemplan ya aspectos de modelado del hipertexto y de la BD: Araneus (Atzeni *et al.*, 1998; Mecca *et al.*, 1999) ó W3I3, proyecto en el que se desarrolló WebML (Bonifati *et al.*, 2000). En esta tercera generación aparecen además extensiones de UML para el modelado de SIW, como por ejemplo, la extensión de Baumeister *et al.* (1999).

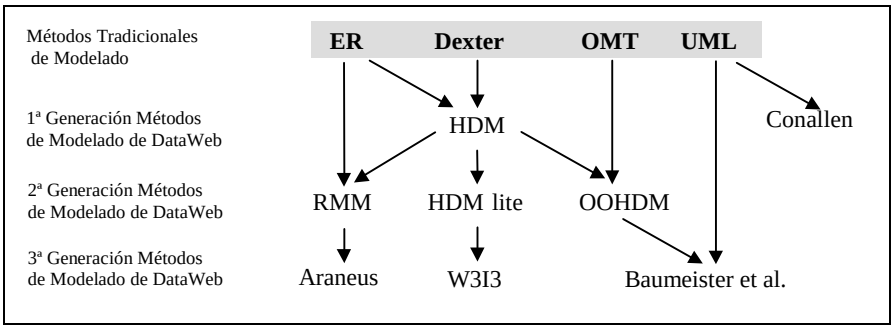


Figura 5. Orígenes de las Metodologías para el modelado de BD Web (Retschitzegger y Schwinger, 2000)

2.2 Resumen de las principales propuestas metodológicas

A continuación, se presenta un resumen del estudio sobre las distintas propuestas metodológicas para el desarrollo de SIW que se han considerado más relevantes. Este estudio se ha realizado con la finalidad de identificar las deficiencias existentes e intentar paliarlas mediante la definición de una nueva metodología que recoja las mejores técnicas de cada una de las propuestas y que defina nuevas técnicas siempre que sea necesario. El orden de presentación de las propuestas viene dado por la propia evolución de cómo han ido surgiendo las mismas. En primer lugar aparecieron los métodos para diseñar aplicaciones hipermedia, luego evolucionaron para poder diseñar además el sistema de persistencia de la información y por último, surgieron los marcos metodológicos genéricos que incorporaban aspectos cada vez más importantes en la definición de SIW.

2.2.1 HDM - Hypertext Design Model

HDM (Garzotto *et al.*, 1993) es una de las primeras propuestas que aparecieron para la definición de la estructura y la navegación de las aplicaciones hipertexto. Realmente lo que HDM propone es una técnica de modelado, que no puede considerarse una metodología, entre otros motivos porque no cuenta con un proceso de desarrollo. Dicha técnica se basa en el modelo entidad-interrelación (E/R) introduciendo elementos nuevos para la representación del hipertexto.

Una característica a resaltar de esta propuesta es que diferencia entre la información que se almacena y la que se muestra al usuario, siendo una de las pioneras en considerar estos aspectos. Sin embargo, a pesar de la creciente aceptación del paradigma de orientación a objetos de la época, HDM se basa en el

modelo E/R. Tampoco trata aspectos tan importantes como la interfaz o la recuperación de la información. Y como ya se ha indicado previamente, HDM propone únicamente una técnica de modelado, en ningún caso propone una metodología de desarrollo.

2.2.2 RMM - Relationship Management Methodology

RMM (Isakowitz *et al.*, 1995; Isakowitz *et al.*, 1998) es una propuesta metodológica para el diseño y desarrollo de aplicaciones hipermedia que se centra en las etapas de diseño, desarrollo y construcción.

La metodología se basa en el modelo E/R y en HDM. Permite representar los objetos del dominio, así como sus interrelaciones y los mecanismos de navegación entre los mismos en base a un modelo denominado modelo RMDM (Relationship Management Data Model). Por otra parte, RMM distingue claramente entre los niveles de contenido, presentación e hipertexto.

RMM contempla el aspecto navegacional, la interfaz y la realización de pruebas, en base a un proceso de siete pasos: Diseño E/R, Diseño de Fragmentos, Diseño Navegacional, Diseño del Protocolo de Conversión, Diseño de la Interfaz de Usuario, Diseño del Comportamiento Dinámico y Construcción y Pruebas.

Esta metodología ofrece la propuesta de un método estructurado y definido para el desarrollo de las aplicaciones hipermedia en base a un conjunto ordenado de pasos. Sin embargo, al igual que la propuesta del apartado anterior, se basa en el modelo E/R cuando la tendencia en su época, ya iba dirigida a la orientación a objetos. Tampoco contempla de forma completa el ciclo de desarrollo de un SIW.

2.2.3 HDM-lite

HDM-lite (Fraternali y Paolini, 1998) propone un modelado conceptual para la especificación de aplicaciones Web. Supone una evolución de HDM pero en el contexto de la Web.

Una aplicación se especifica en HDM-lite definiendo su contenido, navegación y presentación a través de esquemas de hiperbase, acceso y presentación, respectivamente. Un esquema hiperbase describe la estructura de la información de la aplicación, en base a modelos que representan entidades y sus relaciones; generalmente, este esquema se conoce como modelo conceptual de datos. Un esquema de acceso permite representar cómo acceder a las entidades del esquema

anterior, en definitiva, se corresponde con el modelo de navegación. El esquema de presentación, en base a páginas de estilo, se usa para indicar cómo se presenta la información y sus formas de acceso.

Los creadores de este método proponen además AUTOWEB, que es un conjunto de herramientas software para editar especificaciones conceptuales HDM-lite, transformarlas en esquemas lógicos y producir páginas Web automáticamente.

HDM-lite propone seguir el siguiente conjunto de pasos: construcción del esquema HDM-Lite con el apoyo de la herramienta Visual HDM Diagram Editor; transformación del modelo conceptual a un modelo lógico; introducción de datos en la BD, bien automáticamente mediante una aplicación producida por la herramienta Autoweb Data Entry Generator, o bien manualmente; por último la construcción dinámica de las páginas de la aplicación, a partir del contenido y el esquema de la BD, con la ayuda de la herramienta Autoweb Page Generator.

HDM-lite, además de tener las características de la propuesta de HDM, cuenta con una herramienta que permite la generación de las páginas Web de SIW. Hay que indicar que está centrada en el diseño e implementación de sitios Web, por lo que no soporta el ciclo de vida completo de un desarrollo software.

2.2.4 OOHDM – Object-Oriented Hypermedia Design Model

OOHDM (Schwabe y Rossi, 1998) es una metodología orientada a objetos para el diseño de aplicaciones hipermedia, que se basa en la propuesta de HDM, anteriormente mencionada.

Esta metodología contempla la posibilidad de tener diferentes vistas de navegación en base a los diferentes tipos de usuarios que pueden ejecutar las aplicaciones.

OOHDM propone un proceso de cuatro pasos siguiendo un modelo de proceso con prototipado, iterativo e incremental: diseño conceptual, diseño de navegación (en base a dos modelos, el de clases navegacionales y el de contextos navegacionales), diseño de la interfaz abstracta mediante ADVs (*Abstract Data Views*) e implementación. Cada paso se centra en un aspecto concreto del diseño y se construyen o enriquecen los modelos orientados a objetos obtenidos en iteraciones anteriores.

En esta metodología hay que destacar que se basa en el paradigma de orientación a objetos, haciendo uso de UML para la representación del diagrama de

clases. Ofrece además un conjunto de técnicas para representar los modelos de cada una de sus fases y permite la separación entre el nivel conceptual, de navegación y de presentación, lo que facilita el mantenimiento. Sin embargo, hay que señalar que sólo contempla el aspecto estructural de los sistemas, no el de comportamiento. También carece de pautas y de guías descriptivas para la realización de los modelos que propone.

2.2.5 Extensiones de UML para el desarrollo de aplicaciones Web

Dado que UML (Booch *et al*., 1999b) presentaba ciertas carencias para representar determinados aspectos Web, Conallen (2000) ha creado un conjunto de extensiones para representar elementos asociados al desarrollo Web, como páginas cliente, páginas servidor, etc.

Esta propuesta ha ampliado el modelo de UML para incorporar ciertos conceptos relacionados con la Web, pero aunque las extensiones ofrecen semántica para la representación de dichos conceptos (páginas cliente, páginas servidor, marcos, etc.), en ningún caso se ofrecen guías para representar los aspectos de navegación, presentación y de la información almacenada. Se trata, por tanto de una extensión muy cercana a la implementación del sistema.

2.2.6 UWE - UML-based Web Engineering approach

UWE (Koch *et al.*, 2000; Hennicker y Koch, 2000) es una metodología basada en UML para el diseño de aplicaciones hipermedia.

El modelo de proceso presentado en esta propuesta es un proceso iterativo, incremental y orientado a objetos basado en el Proceso Unificado (Jacobson *et al.* 2002). Plantea un conjunto de tres modelos: modelo espacial de navegación y modelo estructural de navegación, que se corresponden respectivamente con los modelos de fragmentos y de navegación propuestos en RMM y el modelo de presentación, que se corresponde con el modelo de interfaz abstracta de la propuesta de OOHDM. El proceso a seguir en UWE indica que a partir del modelo conceptual de la aplicación y del análisis de los casos de uso, y mediante las guías propuestas, se realice el modelo espacial de navegación. A partir de éste, se propone la realización del modelo estructural de navegación y en otro paso posterior se propone cómo obtener el modelo de presentación, en base a la información que hay que presentar al usuario.

UWE incorpora un aspecto muy importante que las propuestas anteriores no contemplan, que es la incorporación de UML para representar el diseño del hipertexto. También ofrece guías para la transformación entre modelos además de incluir una etapa de análisis de requisitos. Sin embargo, se centra exclusivamente en el aspecto de hipertexto.

Aunque inicialmente esta propuesta no contemplaba XML, los últimos trabajos de UWE han incorporado esta tecnología como parte de su trabajo (Kraus y Koch, 2002).

2.2.7 ARANEUS

La metodología de diseño Web ARANEUS (Atzeni *et al.*, 1998; Mecca *et al.*, 1999) aplica un proceso de diseño sistemático para organizar y mantener grandes cantidades de información con la finalidad de que esté accesible a través de la Web. La accesibilidad se plantea en base al almacenamiento de la información en una BD relacional, que bien puede existir previamente o bien, tiene que diseñarse paralelamente a la aplicación Web.

Esta metodología amplia y se basa en otras de las propuestas previamente mencionadas de diseño hipermedia (HDM (Garzotto *et al.*, 1993), RMM (Isakowitz *et al.*, 1995; Isakowitz *et al.*, 1998) y OOHDM (Schwabe y Rossi, 1998)), para representar el diseño de navegación mediante el modelo denominado NCM (Navigational Conceptual Model) y el diseño de fragmentos, que en ARANEUS se basa en el modelo ADM (Araneus Data Model).

ARANEUS propone un proceso de seis etapas, centrándose en el diseño del hipertexto y de la BD, tanto a nivel conceptual como a nivel lógico: diseño conceptual de la BD (E/R), diseño lógico de la BD (Relacional), diseño conceptual del hipertexto en base al modelo conceptual de navegación (NCM), diseño lógico del hipertexto en base al modelo de datos de ARANEUS (ADM), diseño de la presentación y, correspondencia BD-hipertexto y generación de páginas con un lenguaje específico denominado PENELOPE. Este lenguaje permite la generación automática de páginas con hipertexto complejo a partir de la información almacenada en la BD, con lo que por otra parte facilita el mantenimiento del sitio Web.

Para concluir, indicar que ARANEUS es una metodología sólida para el diseño, implementación y mantenimiento de sitios Web, con herramientas de apoyo

eficaces y soportada a partir de los modelos NCM y ADM. Permite la correspondencia directa desde la BD a estructuras hipertexto y viceversa, pero al mismo tiempo separa claramente el diseño del hipertexto del diseño de la BD. Diferencia además entre el diseño conceptual y el diseño lógico. En sus propuestas iniciales ARANEUS separaba la presentación del contenido mediante el lenguaje PENELOPE. Sin embargo, los últimos trabajos incorporan ya XML. Es una metodología que sólo contempla el aspecto estructural de SIW, obviando el aspecto de comportamiento.

2.2.8 OO-Hmethod – Object-Oriented Hypermedia method

OO-method (Pastor *et al.*, 1997) es un método orientado a objetos, centrado en la fase de diseño y de implementación, así como un lenguaje de especificación formal denominado OASIS (Pastor y Ramos, 1995). Esta propuesta diseña el sistema en base a tres modelos que son el modelo de objetos, el modelo dinámico y el modelo funcional, que permite representar tanto la vista estática como dinámica del SIW.

OO-*H*method (Gómez *et al.,* 2000; Gómez *et al.,* 2001) es una ampliación de OO-method, en la que se utiliza UML y donde se incorporan nuevos modelos para representar la interoperabilidad de los usuarios con el sistema Web: Diagrama de Acceso Navegacional (NAD) y Diagrama de Presentación Abstracta (APD). OO-*H*method propone también la utilización de XML. Para simplificar las tareas de diseño e implementación de SIW, dispone también de una herramienta CASE OO-*H*method que posee un enfoque orientado a objetos y que ofrece la posibilidad de realizar prototipos.

2.2.9 WISDOM – Web-based Information System Development with a cOmprehensive Methodology

WISDOM (Castano *et al.*, 2000) es un marco metodológico para el desarrollo de SIW, adaptable a los diferentes tipos de sistemas. La adaptación se realiza en base a una combinación adecuada de las actividades que se proponen.

WISDOM plantea dos actividades básicas que son la *importación* y la *exportación de información*. La primera de ellas contempla la colección, extracción e integración de toda la información relevante y existente tanto en sitios Web como en otras fuentes de datos propias de la organización. Generalmente requiere el desarrollo de una aplicación interna que permita la manipulación y adaptación de

dicha información. La segunda actividad, *exportación de información*, está relacionada con la necesidad de publicar e intercambiar información a través de la Web y consta de las subactividades de diseño del sitio Web y de diseño del flujo de trabajo (*workflow*). La primera subactividad está relacionada con los aspectos metodológicos del diseño hipermedia, ya comentados en propuestas anteriores (diseño de contenido, de hipertexto y de presentación); la segunda subactividad, el diseño del flujo de trabajo, permite dar soporte a la ejecución de los procesos de negocio de la organización sobre la Web.

Las principales características de WISDOM son: la incorporación de actividades para cubrir no sólo los aspectos estructurales de los SIW sino también los de comportamiento; el hecho de que sea un marco metodológico adaptable al desarrollo de diferentes tipos de SIW; y sin duda, la incorporación de las actividades de exportación e importación de información. Sin embargo, no ofrece guías de transformación entre etapas para el desarrollo del SIW.

2.3 Conclusiones

En la Tabla 1 se muestra una comparativa de las principales propuestas metodológicas presentadas, con respecto a los aspectos estructurales de los SIW (presentación, navegación y persistencia). Se incluye además, para las metodologías estudiadas, cuáles de ellas utilizan notación UML y tecnología XML y OR en su proceso de desarrollo. También se comparan otros aspectos, como las guías de transformación entre modelos y la existencia de herramientas asociadas.

Tras el estudio de las distintas propuestas metodológicas presentadas, MIDAS/DB ha recopilado las mejores soluciones y ha propuesto aquéllas que eran necesarias para cubrir los aspectos no contemplados hasta el momento por dichas metodologías. Quizá una de las propuestas más directamente relacionada, en determinados aspectos, con MIDAS/DB es ARANEUS. ARANEUS surge antes de la existencia de XML y define un lenguaje, PENELOPE, que separa los aspectos de estructura, navegación y presentación. Esta separación se propuso en las primeras especificaciones de MIDAS/DB, pero basándose en tecnología XML. Posteriormente, ARANEUS también incorpora XML a su propuesta.

Hay que señalar que la Tabla 1 muestra la situación actual de las metodologías para el desarrollo de SIW y que ésta es muy distinta del momento en que se comenzó a desarrollar MIDAS/DB. Así, por ejemplo, en el momento de

iniciarse los trabajos de MIDAS, las propuestas existentes no consideraban UML como lenguaje de modelado, ni tampoco se basaban en tecnología XML. No hay que olvidar que este trabajo se comenzó en el año 1999 y que UML fue estándar en 1997 (Booch *et al.* 1997), los DTDs en 1998 (Bray *et al.,* 1998; Sánchez y Delgado, 2002) y los esquemas XML no han sido estándar hasta el año 2001 (W3C, 2001b).

Tabla 1. Comparativa de distintas propuestas metodológicas en cuanto a los aspectos estructurales, tecnología empleada, guías de transformación y herramientas asociadas

	Aspectos estructurales			Tecnología			Guías de Trans-formación	Herra-mientas
Metodologías	**Presentación**	**Navegación**	**Persistencia**	**UML**	**XML**	**OR**		
HDM	✓	✓	×	×	×	×	×	×
RMM	✓	✓	✓	×	×	×	×	×
HDM-Lite	✓	✓	✓	×	×	×	×	Autoweb
OOHDM	✓	✓	✓	✓	✓	×	×	×
Ext. UML	✓	✓	✓	✓	×	×	×	Rational
UWE	✓	✓	×	✓	✓	×	✓	ArgoUML tool
ARANEUS	✓	✓	✓	×	✓	×	✓	PENELOPE
OO-*H*-Method	✓	✓	✓	✓	✓	×	✓	OO-Hmethod CASE
WISDOM	✓	✓	✓	×	×	×	×	×
MIDAS/DB	✓	✓	✓	✓	✓	✓	✓	✓

Notación Usada

✓ : se contempla un determinado aspecto en la metodología

× : no se contempla un aspecto en la metodología

Metodología MIDAS/DB

En este capítulo se presenta MIDAS/DB que, como ya se ha indicado previamente, es una metodología basada en modelos para el desarrollo de la dimensión estructural de Sistemas de Información Web (SIW). El capítulo se estructura del siguiente modo: en primer lugar se hace una breve descripción y justificación de las características de MIDAS/DB dentro del marco de MIDAS, que define un proceso completo para el desarrollo de SIW contemplando también la dimensión de comportamiento; a continuación se describe el proceso; por último, se presentan las técnicas nuevas que se han propuesto en MIDAS/DB.

3.1 Descripción General

MIDAS/DB se enmarca dentro de un trabajo más amplio denominado MIDAS, un marco metodológico que describe una arquitectura basada en modelos y un proceso de desarrollo ágil para el desarrollo de SIW. El marco de MIDAS, así como el modelado y desarrollo de los aspectos de comportamiento están siendo desarrollados en otras dos tesis doctorales muy estrechamente relacionadas con la tesis que aquí se presenta y cuya propuesta se centran en los aspectos estructurales.

3.1.1 Marco de trabajo: MIDAS

MIDAS es una metodología genérica que se basa en la utilización de modelos (Turk *et al.*, 2002), para el desarrollo de SIW. MIDAS propone un proceso iterativo e incremental (Sommerville, 2001), basado en prototipado, y utiliza prácticas extraídas de metodologías ágiles, como XP - eXtreme Programming (Beck, 1999). Por lo tanto, una característica diferenciadora de MIDAS es que es una metodología *ligera*, que se ha definido para satisfacer tanto las necesidades de los clientes como de los desarrolladores. Habitualmente, la introducción de una nueva metodología en una empresa supone que los analistas y desarrolladores tienen que cambiar sus hábitos de trabajo. Además, las necesidades más inmediatas que tiene el cliente que encarga la aplicación Web, es que su producto software esté disponible lo antes posible. El *modelo de proceso iterativo e incremental* aportará las ventajas de tener productos tangibles para el cliente en versiones sucesivas, además de permitir incorporar o modificar necesidades no detectadas o no planteadas en las primeras etapas de definición del producto. Diferentes autores (Beck, 1999; Overmyer,

2000) afinan aún más, indicando que el objetivo es realizar pequeños ciclos de vida en cortos espacios de tiempo. Esto garantiza sucesivas entregas al cliente hasta completar el producto final. Por otra parte, y gracias a los prototipos, el cliente tiene la posibilidad de validar el producto continuamente.

Por este motivo, MIDAS se ha definido con el fin de satisfacer los siguientes objetivos:

- Proporcionar a los desarrolladores una metodología basada en modelos que guíe su trabajo siguiendo su forma habitual de trabajar.
- Soportar un desarrollo de software rápido, con el fin de asegurar a los clientes una primera versión del software en el menor tiempo posible.
- Reducir la cantidad de documentación generada durante el desarrollo del SIW.

MIDAS proporcionará al cliente los productos en un tiempo corto, permitiendo introducir en cada iteración nuevos requisitos que no se identificaron en iteraciones anteriores del desarrollo. Otra ventaja será que las pruebas se harán para cada iteración con lo que se reducirá el riesgo de que se produzcan fallos.

Como ya se ha dicho, MIDAS propone distintas iteraciones y al final de cada una de ellas se obtiene una nueva versión del producto.

- En una primera iteración, MIDAS/SD, que constituye el núcleo del proceso, se definen los requisitos y la arquitectura del sistema.
- En la segunda iteración denominada MIDAS/HT, se desarrolla un primer prototipo del SIW, construyendo el hipertexto con páginas estáticas en HTML para proporcionar al cliente una primera versión del producto en un corto periodo de tiempo.
- En la tercera iteración, denominada MIDAS/DB, se implementará una nueva versión del hipertexto con páginas dinámicas en XML, recibiendo como entrada el prototipo definido en la iteración previa.
- En una iteración adicional, denominada MIDAS/FC, se desarrollan los servicios y la lógica del SIW.
- En otra iteración, MIDAS/TST, se probará el sistema.

Otros criterios, tales como seguridad, etc., pueden considerarse ortogonales a los anteriores. Así, por ejemplo, el nivel de seguridad requerido en una aplicación Web, no depende de que ésta tenga o no una BD detrás, ya que podrían requerirse los mismos niveles de seguridad para un sitio Web en el que sólo se publicara información estática; por ello, estos criterios no afectan al tipo de proceso de desarrollo software, sino que deberán ser tenidos en cuenta a lo largo del mismo, reflejándose en aquellas etapas del proceso que lo requieran.

En la Figura 6 se resume el proceso propuesto en MIDAS. En él, las iteraciones 2 y 3 corresponden a MIDAS/DB. Hay que señalar que aunque en esta figura las iteraciones se muestren una a continuación de la otra, esto no significa que hayan de realizarse necesariamente de un modo secuencial. En el apartado 3.2 se describe en detalle el proceso para la dimensión estructural que constituye el objeto de esta tesis.

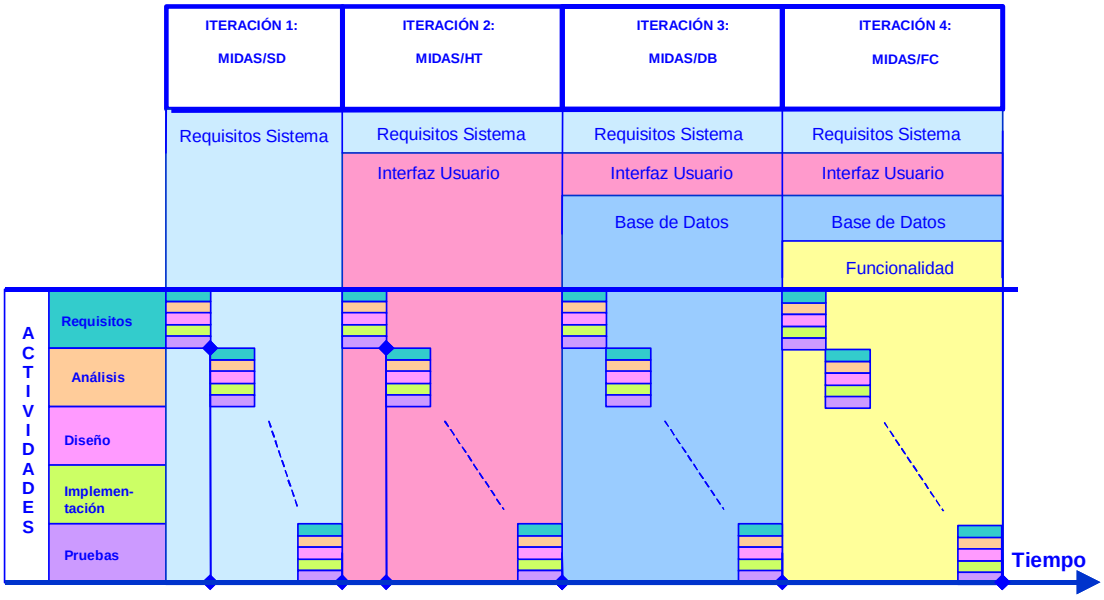


Figura 6. Proceso de MIDAS

Como se ha indicado, MIDAS es una metodología basada en modelos, donde cada modelo propuesto permite describir una vista del sistema a diferentes niveles de abstracción. Para ello, define una arquitectura basada en MDA (Model Driven Architecture), (Millar y Mukerji, 2001; Millar y Mukerji, 2003), con modelos independientes y dependientes de plataforma (PIMs y PSMs, respectivamente). La Figura 7 muestra una instanciación de la arquitectura MDA para el caso concreto que nos ocupa.

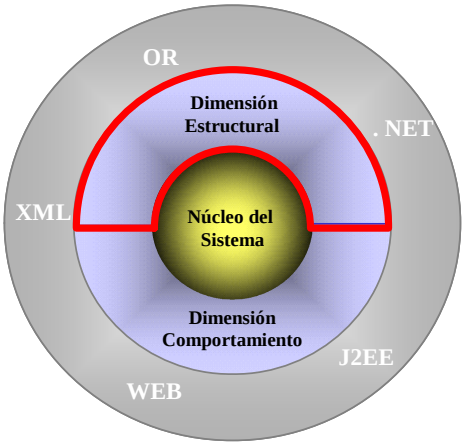


Figura 7. Arquitectura basada en Modelos de MIDAS

3.1.2 Características de MIDAS/DB

MIDAS/DB, para cada vista del sistema, selecciona, adapta e integra las técnicas o notaciones de las metodologías existentes, tomando las mejores prácticas de cada una de ellas. Para el diseño hipermedia se basa principalmente en las técnicas que se proponen en RMM (Isakowitz *et al.*, 1995; Isakowitz *et al.*, 1998) y en una adaptación de la notación propuesta en UWE (Baumeister *et al.*, 1999; Koch *et al.*, 2000) y para el diseño de bases de datos (BD) en los trabajos de Atzeni *et al.* (1998) y Mecca *et al.* (1999). Además, esta metodología define técnicas nuevas, siempre que sea necesario ya que, como se ha mencionado anteriormente, no pretende buscar soluciones nuevas a problemas ya resueltos, sino ofrecer soluciones a problemas abiertos.

Esta tesis doctoral se centra en la dimensión estructural del desarrollo de un SIW, que es la parte remarcada en la Figura 7. La dimensión estructural del sistema incluye, tal y como se verá en la sección siguiente, tanto los aspectos de contenido como los de hipertexto y presentación. Para ello, MIDAS/DB propone el uso de estándares en el proceso de desarrollo de un SIW, por lo que se basa en **UML** (Booch *et al.*, 1999), **XML** (Bray *et al.*, 2000) y **SQL:1999** (Eisenberg y Melton, 1999).

- UML como notación única

Debido a que UML se ha convertido en un lenguaje estándar de modelado orientado a objetos, en MIDAS/DB se propone utilizar este

lenguaje además de algunas extensiones del mismo, para modelar el sistema completo.

Todos los modelos del sistema se representarán en UML independientemente del nivel de abstracción del que se trate. Sin embargo, UML no tiene una notación específica para representar todas las técnicas necesarias para desarrollar un SIW. Por ejemplo, no tiene una notación específica para representar un modelo navegacional, un esquema de BDOR o esquemas XML (XML Schemas). Recientemente, han aparecido algunas extensiones UML para diseño Web, como por ejemplo, los trabajos de Baresi *et al.* (2001), Baumeister *et al.* (1999) y Koch *et al.* (2000). En MIDAS/DB se propone la utilización de los trabajos de Baumeister *et al.* (1999) y Koch *et al.* (2000). Además se definen nuevas extensiones para BDOR, incluyendo el modelo OR y las consultas, y para tecnología XML, incluyendo esquemas XML y XLink.

- Tecnología XML

Se utilizará XML (eXtensible Markup Language) (Bray *et al.*, 2000) debido a las ventajas que aporta que se indican a continuación: a) permite separar el contenido de la presentación, de modo que un cambio en la presentación de las páginas no afecte, ni al contenido, ni a la organización de las mismas (hipertexto); b) además, al separar el contenido de la presentación, permite que un mismo dato pueda representarse de la forma más conveniente en cada caso; c) permite facilidades de adaptación a múltiples plataformas, permitiendo visualizar la información en dispositivos móviles, PDAs, etc., lo que facilita el desarrollo de aplicaciones B2C (*Bussiness to Customer*); d) es particularmente útil para el intercambio de información entre empresas, para la comunicación entre aplicaciones, así como para integrar información procedente de distintas fuentes y plataformas, lo que facilita el desarrollo de aplicaciones B2B (*Bussiness to Bussiness*); e) al tratarse de un estándar abierto e implementado ya en los principales fabricantes de software (Oracle, IBM, Microsoft, etc.), minimiza el coste de migración de las aplicaciones a otras plataformas.

- Tecnología Objeto-Relacional

Pese al impacto que las bases de datos relacionales (BDR) han tenido en las últimas décadas, este tipo de BD tiene una serie de limitaciones para

soportar la persistencia de datos requerida por algunas de las aplicaciones actuales. Las mejoras continuas en el hardware han provocado que hayan surgido aplicaciones cada vez más sofisticadas, como CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing), CASE (Computer-Aided Software Engineering), GIS (Geographic Information System), etc. Este tipo de aplicaciones se caracteriza por requerir un sistema de objetos complejos relacionados por medio de interrelaciones también complejas. La representación de este tipo de objetos e interrelaciones por medio del modelo relacional implica que los objetos deben descomponerse en un elevado número de tuplas. De esta forma, para recuperar un objeto es necesario realizar un considerable número de combinaciones, lo que provoca que, cuando las tablas están profundamente anidadas, el rendimiento se ve afectado significativamente (Bertino y Marcos, 2000).

Para dar respuesta a estos problemas, surge la generación de BD orientadas a objetos, que incluyen tanto las BD objeto-relacionales (Stonebraker y Brown, 1999), como las bases de objetos puras (Bertino y Martino, 1993). Esta nueva tecnología se presenta como la más indicada para el almacenamiento de datos complejos, ya que soporta tipos de datos e interrelaciones complejas, datos multimedia, herencia, etc. La tecnología objeto-relacional presenta dos ventajas frente a las bases de objetos puras: proporciona un mejor soporte para aplicaciones voluminosas y es totalmente compatible con la tecnología relacional, ya que se trata de BD que se construyen extendiendo dicho modelo. Por ello, se espera que las BD objeto-relacionales sean las que mayor impacto tengan en el mercado en los próximos años. Según un informe de IDC (Leavit, 2000) en 1999 los ingresos por ventas de bases de datos relacionales y objeto-relacionales fueron de 11.100 millones de dólares y de 211 millones de dólares para bases de objetos. Hasta 2004, IDC predice un grado de crecimiento anual del 18’2% en el caso de las bases de datos relacionales y objeto-relacionales y del 12’5% para las bases de objetos. Parece claro, por tanto, que las bases de datos (objeto-)relacionales continuarán siendo el núcleo de la mayoría de los sistemas de información durante un largo período de tiempo.

Además, hay que destacar la importancia de definir métodos que guíen a los diseñadores en el proceso de diseño de las bases de datos (objeto-) relacionales, tal como se venía haciendo tradicionalmente con las

BDR. En los últimos años han surgido algunas aproximaciones para el diseño de bases de datos orientadas a objetos (Blaha y Premerlani, 1998; Kovács y Van Bommel, 1998; Muller, 1999; Silva y Carlson, 1995; Ullman y Widom, 1997), pero ninguna de estas propuestas puede considerarse "universal", ni para las bases de datos (objeto-)relacionales ni para las bases de objetos puras. Por una parte, las propuestas existentes no tienen en cuenta las últimas versiones de los estándares más representativos para ambas tecnologías: ODMG 3.0 en el caso de las bases de objetos (Cattell y Barry, 2000) y SQL: 1999 en el caso de las bases de datos (objeto-)relacionales (Eisenberg y Melton, 1999). Por otra parte, algunas de las propuestas se basan en técnicas antiguas como OMT (Blaha y Premerlani, 1998) o, incluso, el modelo E/R (Ullman y Widom, 1997). Por ello, las propuestas actuales deben actualizarse tomando como modelos de referencia UML, SQL: 1999 y ODMG 3.0.

Al igual que las metodologías tradicionales proporcionan técnicas gráficas que permiten representar un modelo relacional (Diagrama de Estructura de Datos, Grafo Relacional, etc.), un modelo objeto-relacional se puede representar, además de en SQL (SQL:1999 u Oracle8*i* o 9*i*), mediante una notación gráfica, que se propone sea UML. Por lo tanto, es importante resaltar la importancia de proporcionar guías metodológicas para el diseño de aplicaciones centradas en datos utilizando UML.

Aunque en los últimos años han surgido también gestores de datos XML nativos (Chaudhri et al., 2003), puede decirse que éstos son apropiados para el manejo y gestión de datos XML cuando el volumen de información a almacenar no es excesivo. En este sentido, y teniendo en cuenta que las BDOR ya incorporan capacidades para el tratamiento y gestión de datos XML, bien como documentos almacenados en una columna, bien como esquemas XML nativos, se propone el uso de esta tecnología que, por otra parte, es la más implantada en el mercado. Además, algunos de los inconvenientes que se le han atribuido a los gestores OR con soporte de datos XML, no son ciertas. En concreto, en Bertrand (2002) se dice que los modelos OR no son adecuados para almacenar documentos XML debido a la estructura jerárquica de los mismos. Sin embargo, este problema se resuelve en Oracle9*i*, que utiliza el propio modelo OR para soportar esquemas XML, mediante la utilización de *Varrays* y *nested tables* (tablas

anidadas) que permiten soportar de un modo natural el anidamiento de documentos XML.

3.2 Proceso

La Figura 8 muestra una visión global del proceso de desarrollo de la dimensión estructural de un SIW, incluyendo las iteraciones relativas a MIDAS/HT y MIDAS/DB y cómo éstas se relacionan entre sí.

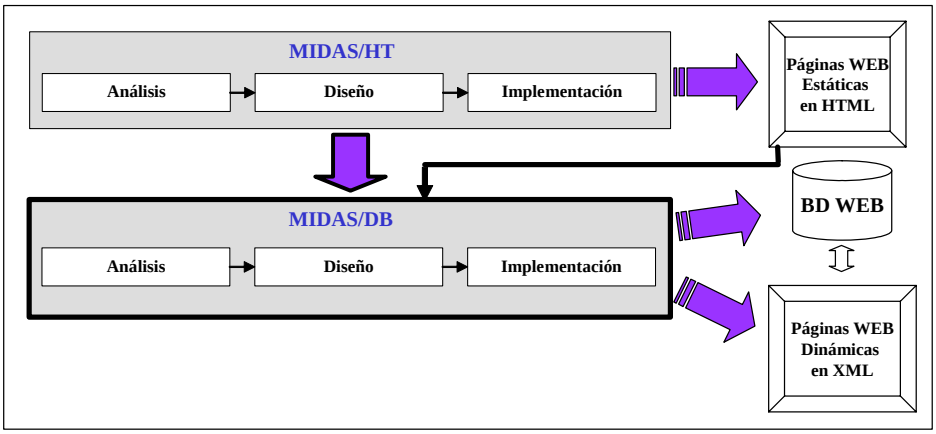


Figura 8. Proceso de desarrollo de la dimensión estructural de un SIW

En la iteración **MIDAS/HT** se desarrolla un primer prototipo del SIW, construyendo el hipertexto con páginas estáticas en HTML. Tanto el prototipo como los modelos generados en esta iteración servirán de entrada a la tercera iteración. En la tercera iteración, denominada **MIDAS/DB**, se desarrolla por un lado la BD Web y por otro lado, se implementa una nueva versión del hipertexto y de la presentación con páginas dinámicas en XML que extraen la información de la BD Web. Además, dado que la navegación entre páginas normalmente implica una consulta a la BD, el modelado de las consultas es una conexión clave entre el modelado del hipertexto y el modelado de la BD. La metodología que se define en este trabajo propone la integración del modelado de consultas en el modelado de la navegación, como un primer paso para integrar el diseño del hipertexto con el diseño del contenido en el desarrollo de un SIW.

Para cada una de estas actividades se definen un conjunto de tareas, técnicas y notaciones específicas que se detallan a continuación. Las actividades que contempla la metodología MIDAS/DB son: análisis, diseño e implementación para

cada una de las iteraciones. Otras actividades, como captura de requisitos, no son específicas de la dimensión estructural y, por ello, su estudio no es objeto de esta tesis.

3.2.1 MIDAS/HT

3.2.1.1 Descripción y Objetivos

Esta iteración tiene como **objetivo** principal obtener un primer prototipo del SIW mediante la construcción de una primera versión del hipertexto con páginas HTML o XML. Este prototipo sirve, por una parte, como una primera versión del producto, permitiendo que la aplicación pueda estar disponible lo antes posible en la Web y, por otra, para validar con el cliente las especificaciones obtenidas en la captura de requisitos.

Para cada actividad de la iteración MIDAS/HT se definen una serie de tareas, técnicas y notaciones que se resumen en la Tabla 2.

Tabla 2. Actividades para la iteración MIDAS/HT

	2ª. Iteración: **MIDAS/HT**		
Actividad	Tarea	Técnica	Notación
Análisis	Diseño Conceptual de Datos	Modelo Conceptual de Datos (OO)	Diagrama de Clases (UML)
	Diseño Conceptual del Hipertexto	Modelo Conceptual de Fragmentos (RMM)	Diagrama de Fragmentos (UWE)
		Modelo Conceptual de Navegación (RMM)	Diagrama de Navegación (UWE)
	Diseño Conceptual de la Presentación	Modelo Conceptual de Presentación (OOHDM)	Diagrama de Presentación (UWE)
Diseño	Diseño Lógico del Hipertexto	Prototipado con Herramientas de Diseño Gráfico (Dream Weaver, Front Page, XMLSpy, etc.)	
Implem.	Implementación de la Interfaz de Usuario		HTML/XML

3.2.1.2 Actividad de Análisis

En la actividad de **análisis** se deben obtener los modelos conceptuales del SIW.

En primer lugar, se realiza el **diseño conceptual de datos** para lo que se propone utilizar el *diagrama de clases de UML* (Jacobson *et al.,* 2001). Partiendo de este diseño conceptual de datos se realiza el diseño conceptual del hipertexto. El hipertexto representa la forma en que la información es agrupada y enlazada para navegar a través de ella.

Para realizar el **diseño conceptual del hipertexto** se propone utilizar dos técnicas de RMM (Isakowitz *et al.,* 1995; Isakowitz *et al.*, 1998): el *modelo de fragmentos* (Slice Model) y el *modelo de navegación*, que en RMM se denomina diagrama de aplicación (Application Diagram). En el *modelo de fragmentos*, la información que está relacionada entre sí, se agrupa en unidades significativas, denominadas fragmentos. Cada fragmento, por tanto, representa una unidad de información que se va a mostrar de forma agrupada. El *modelo de navegación* describe cómo navegar a través de los fragmentos utilizando elementos de acceso, por ejemplo, mediante un índice, un menú, etc. Aunque RMM propone partir de un modelo E/R enriquecido con un conjunto de primitivas de navegación (Relationship Management Data Model - RMDM), en MIDAS/DB se partirá del diagrama de clases en UML para obtener el modelo de fragmentos y de navegación. Además los diagramas de fragmentos y de navegación se representarán utilizando las extensiones de UML de UWE (Koch *et al.,* 2000). Las técnicas propuestas en RMM son más completas que las de UWE, y por ello se propone su utilización en MIDAS/DB, ampliando la notación UML de UWE para poder representar todos los constructores de RMM, como por ejemplo los recorridos guiados indexados o las precondiciones asociadas a los fragmentos.

En paralelo al modelo de navegación, se define el **diseño conceptual de presentación** que permitirá describir cómo se presentará la información en cada fragmento o página (inclusión de un botón, un menú desplegable, una figura, etc.). Se propone usar el modelo conceptual de presentación propuesto en OOHDM, Schwabe y Rossi (1998) en UML extendido siguiendo la propuesta definida en UWE, Koch *et al.* (2000).

3.2.1.3 Actividades de Diseño e Implementación

Una vez realizado el diseño conceptual y obtenidos los modelos conceptuales de datos, hipertexto y presentación, se pasa a las actividades de **diseño** e **implementación** del hipertexto, que se realizarán conjuntamente. Para ello se propone la técnica de *prototipado rápido*, utilizando alguna herramienta de diseño

gráfico (Dream Weaver, Front Page, XMLSpy, etc.). De este modo, a la vez que se realiza el diseño de la interfaz de usuario (IU), se genera la primera versión de las páginas Web, estáticas y en HTML o XML. Una vez realizadas las pruebas correspondientes, estas páginas podrán estar disponibles en la Web en un tiempo prudencial, sirviendo además de prototipo que permita validar con el usuario los requisitos iniciales de la aplicación Web. En función de esta validación con el usuario podrán modificarse, en la siguiente etapa, tanto el modelo conceptual de datos, como el del hipertexto.

3.2.2 MIDAS/DB

3.2.2.1 Descripción y Objetivos

En la tercera iteración, denominada **MIDAS/DB**, se desarrolla la dimensión estructural del sistema, que incluye tanto los aspectos de hipertexto como los de contenido y presentación. El **objetivo** de esta iteración es construir la BD Web e implementar una nueva versión del hipertexto con páginas dinámicas en XML. Por tanto, en esta iteración se lleva a cabo el desarrollo de: a) el hipertexto en XML con páginas dinámicas; b) la BD Web.

3.2.2.2 Actividad de Análisis

Se comienza con una etapa de captura de requisitos en la que, apoyándonos en el primer prototipo obtenido en la segunda iteración, MIDAS/HT, se revisan los requisitos iniciales con el usuario, haciendo especial hincapié en aquellos relativos a la BD. Con estos nuevos requisitos, en la actividad de **análisis** se refinan los modelos conceptuales elaborados en la iteración previa, tanto el de datos como el del hipertexto y la presentación, teniendo también en cuenta la realimentación proporcionada por el producto obtenido en la iteración previa. Tanto las técnicas como la notación empleada para representar los modelos conceptuales refinados serán las mismas que en el caso de la iteración MIDAS/HT (ver Tabla 3).

En esta actividad se incluye además el diseño conceptual de consultas, que puede constituir una forma de integración entre la BD Web y el hipertexto. Dado que una consulta es un sub-esquema, éstas pueden modelarse del mismo modo que los esquemas. Para ello, se propone utilizar como técnica el modelo conceptual de consultas y como notación, UML. El modelo conceptual de consultas será pues una consulta (expresada en UML) al modelo conceptual de datos. En el apartado 3.3.1.2 se presenta la extensión propuesta para el modelado de consultas en UML.

Tabla 3. Actividad de Análisis para la iteración MIDAS/DB

	3ª. Iteración: **MIDAS/DB**		
Actividad	Tarea	Técnica	Notación
Análisis	Diseño Conceptual de Datos Refinado	Modelo Conceptual de Datos (OO)	Diagrama de Clases (UML)
	Diseño Conceptual de Consultas	Modelo Conceptual de Consultas (MIDAS/DB)	Diagrama de Consultas (MIDAS/DB-UML)
	Diseño Conceptual del Hipertexto Refinado	Modelo Conceptual de Fragmentos (RMM)	Diagrama de Fragmentos (UWE)
		Modelo Conceptual de Navegación (RMM)	Diagrama de Navegación (UWE)
	Diseño Conceptual de la Presentación Refinado	Modelo Conceptual de Presentación (OOHDM)	Diagrama de Presentación (UWE)

3.2.2.3 Actividades de Diseño e Implementación

En la actividad de **diseño** se obtendrá el diseño lógico de la BD. Además se realizará un refinamiento del diseño lógico del hipertexto y de la presentación en base a las modificaciones realizadas en los correspondientes modelos conceptuales. El diseño lógico del hipertexto se realizará utilizando tecnología XML.

La actividad de **implementación** incluye el desarrollo de la BD Web en el producto final seleccionado, así como la integración de la BD con las páginas XML mediante una tecnología como ASP (Active Server Pages), JSP (Java Server Pages), etc. Aplicando la correspondiente hoja de estilo XSL se obtendrán las páginas HTML (o cualquier otro formato de presentación deseado, como por ejemplo WML, etc.) dinámicas que extraen la información de la BD.

Las tareas correspondientes a las actividades de diseño e implementación dependerán de factores como la tecnología de BD elegida, si la BD, o parte de ella ya existe, o bien si se parte de cero, etc. A continuación, se especifican las tareas a realizar, clasificadas según los siguientes casos:

Caso A: Partimos de una BD operativa y cuyos datos se quieren poner en la Web.

Caso B: Partimos de cero y la BD se implementa con tecnología OR.

Caso C: Partimos de cero y la BD se implementa en un gestor XML nativo (Sistema de Gestión de Bases de Datos XML –SGBDX).

A continuación, se describe el proceso de desarrollo y las técnicas propuestas para las actividades de diseño e implementación, para cada uno de los casos anteriormente mencionados. Como ya se ha dicho, las tareas de la actividad de análisis de esta iteración son comunes a los tres casos y se han descrito previamente en la sección 3.2.2.2.

3.2.2.3.1 Caso A: Integración del Hipertexto con una BD existente

En primer lugar será necesario averiguar si ya existe una BD en la organización; en caso afirmativo hay que ver si es necesario rediseñarla o basta con integrarla con el hipertexto desarrollado. Hay que tener en cuenta que, en general, una organización no suele rediseñar su BD simplemente con el fin de adecuarla a la Web, por ello, lo más común será diseñar el hipertexto tomando como entrada la BD existente.

En caso de rediseñar la BD, habrá que redefinir el modelo conceptual de datos. Si dicho modelo conceptual no existiese, sería necesario llevar a cabo un proceso de reingeniería para obtenerlo a partir de la BD existente. Así, se podrá redefinir la BD, añadiendo nuevos requisitos al modelo conceptual de la BD.

En caso de tener que integrar el hipertexto con la BD existente, hay que tener en cuenta dos casos: que la BD sea (objeto-)relacional o que sea una BD XML nativa. En ambos casos se procederá con el diseño e implementación del hipertexto, obviando las tareas específicas de diseño e implementación de la BD. En el primer supuesto, que será el más probable, se actuará de la misma forma en que se haría para el caso B, descrito a continuación. Y en el segundo supuesto, es decir, en el caso de que ya exista una BD XML nativa en la organización, se realizará la integración como se verá en el caso C.

3.2.2.3.2 Caso B: Desarrollo de una BD (Objeto-)Relacional

En este supuesto, se realizará el diseño de la BD partiendo de cero. Además, se desea tener una BD (objeto-)relacional que se integrará con el hipertexto en XML, extrayendo los datos dinámicamente para construir las páginas Web. Este caso es especialmente adecuado cuando la BD tiene otros usos además de la publicación de su información en las páginas Web, por ejemplo, una BD corporativa que incluye la gestión de empleados, la contabilidad de la empresa, etc. En este

caso, no toda la información almacenada en la BD tiene que estar accesible desde las páginas Web. Si el volumen de información es grande, también es más recomendable la utilización de tecnología más implantada.

La Figura 9 muestra el proceso de desarrollo de la BD Web para este caso. Como se observa, las tareas de diseño lógico de la BD y del hipertexto se realizan en paralelo y la implementación se hace de modo independiente.

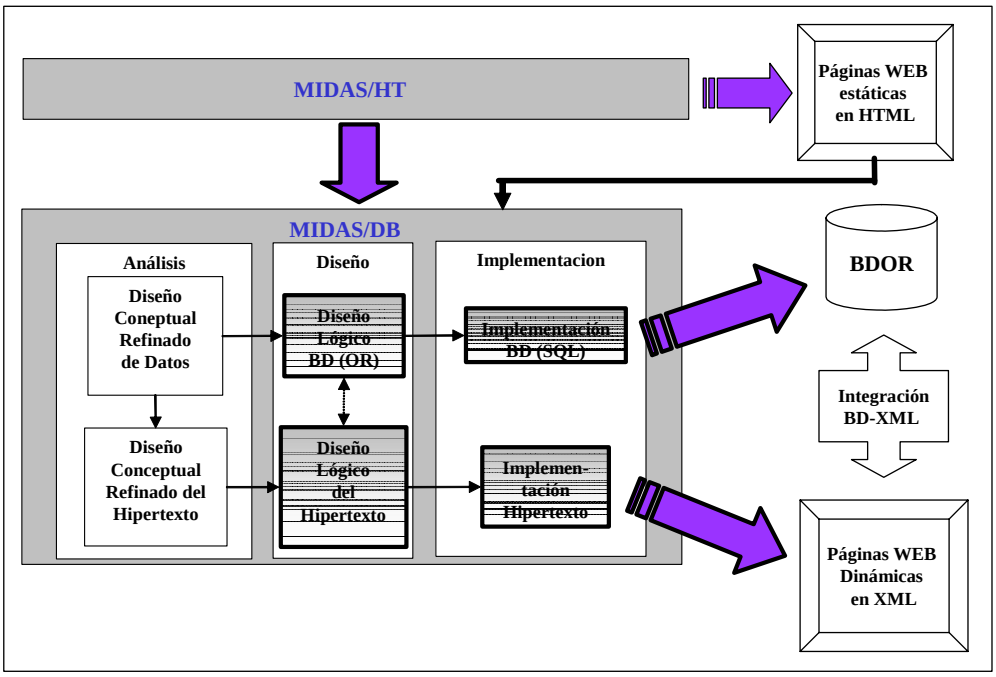


Figura 9. Proceso de desarrollo para el desarrollo de una BD OR Web

La Tabla 4 resume las tareas a realizar durante las actividades de análisis, diseño e implementación de la iteración MIDAS/DB para desarrollar una BD Web en el caso B.

Tabla 4. Actividades para la iteración MIDAS/DB (casos A y B)

	3ª Fase: Base de Datos Web (MIDAS/DB)		
Actividad	Tarea	Técnica	Notación
Análisis	*Diseño Conceptual Refinado*	*Modelos Conceptuales*	*UML*
Diseño	Diseño Lógico de los Datos Diseño Lógico de Consultas Diseño Lógico del Hipertexto Refinado	Modelo (Objeto-) Relacional Modelo Lógico de Consultas Modelo Lógico de Fragmentos (RMM) Modelo Lógico Navegacional (RMM)	Diagrama OR (MIDAS/DB-UML) Diagrama de Consultas (MIDAS/DB-UML) XML Schema (MIDAS-UML) XLink (MIDAS-UML)
Implemen.	Implementación BD Implementación Hipertexto + Implementación Refinada de la Interfaz de Usuario Integración XML-SQL		SQL del Producto Concreto Tecnología XML: XML/XML Schema/ XLink/XSL JSP/ASP/…

Diseño e Implementación de la BD

El desarrollo de la BD Web propone tres fases: análisis, diseño e implementación. La Figura 10 muestra el detalle de la tarea de ***diseño de la BD*** que se definirá partiendo del modelo conceptual de datos refinado, obtenido durante la actividad de análisis de esta iteración.

La fase de **diseño de la BD** se divide en dos pasos:

- Diseño Lógico **Estándar**, es decir, el diseño lógico independiente de cualquier producto.
- Diseño Lógico **Específico**, es decir, el diseño para un producto específico (por ejemplo, Oracle8*i* u Oracle9*i*), sin considerar en este punto aspectos de optimización.

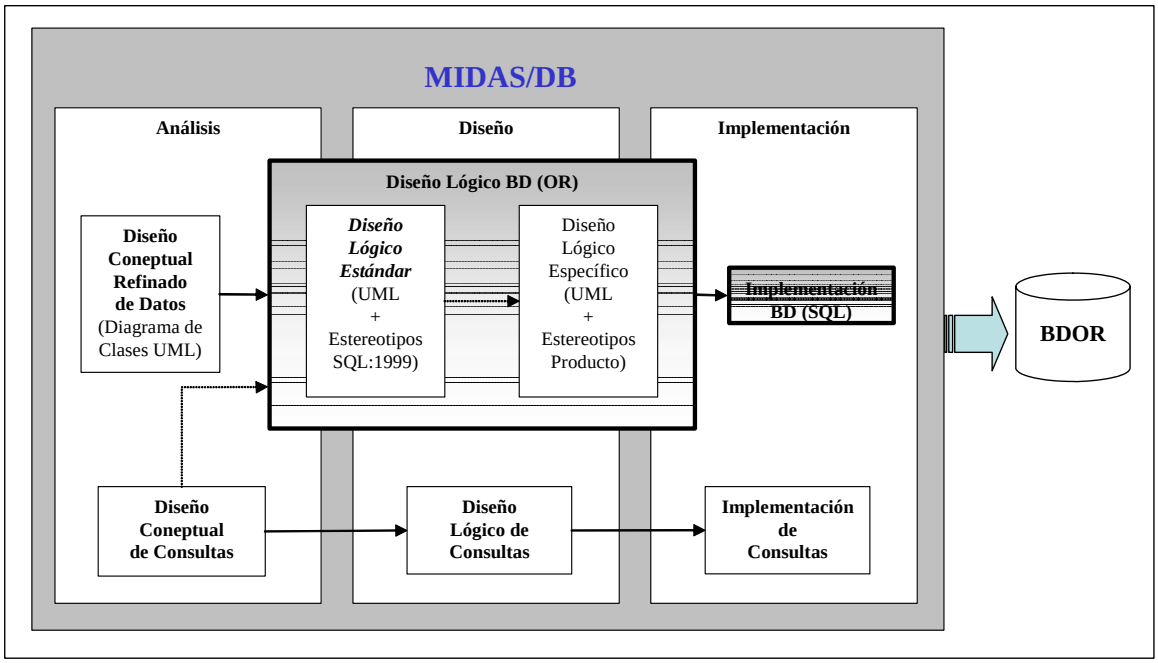


Figura 10. Diseño de Bases de Datos (Objeto-)Relacionales

El **diseño lógico estándar** es especialmente importante en el proceso de diseño de una base de datos (objeto-)relacionales porque cada producto implementa un modelo objeto-relacional diferente. Esta actividad proporciona una especificación (objeto-)relacional en SQL: 1999, que es independiente de producto, mejorando el mantenimiento de la base de datos y facilitando la migración entre productos. Se proponen dos técnicas alternativas: la definición del esquema en lenguaje SQL: 1999; y la utilización de un lenguaje que permita describir el esquema (objeto-)relacional estándar del SQL:1999 en notación gráfica. Esta notación gráfica se correspondería con el grafo relacional que representa el diseño lógico en el caso del tradicional diseño de bases de datos relacionales. Como notación gráfica se propone la utilización de las extensiones de UML definidas en el apartado 3.3.1.2.1 y resumidas en el apéndice B. Las dos notaciones, textual en SQL y gráfica en UML, son equivalentes; ambas representan un esquema (objeto-) relacional en SQL: 1999 y, por tanto, independiente de producto.

En el **diseño lógico específico** (etapa intermedia entre el diseño y la implementación) debe especificarse el esquema en el lenguaje SQL específico del producto elegido, que en nuestro caso es Oracle8*i* y 9*i*. Además, se propone de forma opcional el uso de una técnica gráfica que mejore la documentación y la comprensión del código SQL generado. Esta técnica, además, facilita la compilación del esquema de la BD mostrando el orden apropiado en el que deben compilarse los

nuevos tipos definidos por el usuario, así como las tablas. La notación gráfica propuesta es también UML, sustituyendo los estereotipos definidos para SQL: 1999 por los estereotipos para el producto elegido. En el apartado 3.3.1.2.2 se propone una extensión para Oracle (versiones 8*i* y 9*i*).

Finalmente, la fase de **implementación** incluye las tareas de diseño físico. En esta fase, el esquema obtenido en la fase anterior se refinará, realizando un ajuste que mejore los tiempos de respuesta y el espacio de almacenamiento, de acuerdo con las necesidades específicas de la aplicación.

Además, en esta actividad, se propone también realizar el diseño lógico de consultas. Si el diseño conceptual de consultas es una consulta al modelo conceptual de datos, el diseño lógico es una consulta al diseño lógico de datos; es decir, al modelo (objeto-)relacional de la BD. Para ello se propone, al igual que para el modelado conceptual de consultas, utilizar notación UML. El modelo lógico de consultas será pues una consulta (expresada en UML) al modelo lógico de datos. En el apartado 3.3.1.2 se presenta la extensión propuesta para el modelado de consultas en UML, y en el apartado 3.3.1.4 el modo en que estas modelos se integran con el modelo de hipertexto.

Diseño e Implementación del Hipertexto

El ***diseño el hipertexto*** incluye dos técnicas: el modelo de fragmentos y el modelo de navegación. Para construir el modelo lógico del hipertexto, se partirá de los modelos conceptuales de hipertexto refinados, incluyendo el de fragmentos y el de navegación. Los fragmentos se representan a nivel lógico con esquemas XML (W3C, 2001b) y los modelos de fragmentos y de navegación con XLink (W3C, 2001a). Como para todas las demás técnicas, se propone UML como notación, tanto para los XML esquemas, como para el XLink. Las correspondientes extensiones se describen en los apartados 3.3.2.2 y 3.3.2.4, respectivamente. Para hacer la transición desde el nivel conceptual al nivel lógico del hipertexto, cada uno de los fragmentos del diagrama de fragmentos se transforma en un esquema XML y los enlaces entre los fragmentos que aparecen en los diagramas de fragmentos y de navegación, se transforman usando XLink. En el apartado 3.3.2.5 se definen también unas guías básicas para la transformación entre modelos.

La implementación final incluye la implementación del hipertexto utilizando tecnología XML y su integración con la BD mediante tecnologías como ASP, JSP, etc. Los documentos XML obtenidos pueden almacenarse en un repositorio de XML

nativo, o bien, en un sistema de ficheros tradicional. Para finalizar (en la etapa de pruebas), se realizarían las pruebas del funcionamiento final de la BD Web.

Aplicando la correspondiente plantilla XSL a los documentos XML obtenidos, se obtendrán las páginas HTML (o WML, etc.) con la información recuperada dinámicamente de la BD.

3.2.2.3.3 Caso C: Base de Datos XML nativa

La tercera opción para el desarrollo de una BD Web es utilizar almacenamiento nativo de XML. Pueden ocurrir dos supuestos: a) que los datos y estructura del hipertexto coincidan con los datos y estructura de la BD; b) que no coincidan. En el supuesto a) es especialmente apropiado la utilización de un gestor XML nativo, especialmente si el volumen de información a almacenar no es excesivo. En este caso el *esquema de la BD es el esquema XML* y el diseño lógico de la BD y del hipertexto se realizan al mismo tiempo. La Figura 11 muestra cómo obtener directamente el esquema de la BD Web, a partir de la implementación del hipertexto en XML, empleando almacenamiento nativo de XML. En la Tabla 5 se resumen las tareas a llevar a cabo.

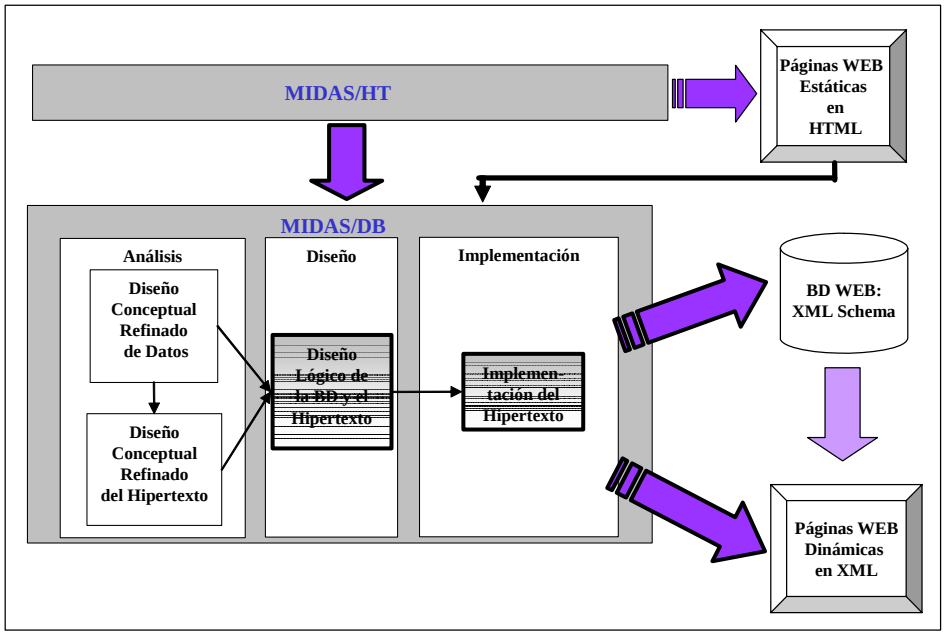


Figura 11. BD Web generada automáticamente a partir de los esquemas XML

En la actividad de diseño de esta iteración se obtiene simultáneamente el diseño lógico de los datos y del hipertexto. La técnica usada para obtener el diseño lógico de los datos es el modelo lógico de fragmentos representado con esquemas

XML en UML extendido. Partiendo del modelo lógico de fragmentos, se definirá el modelo lógico de navegación en XLink para obtener el diseño lógico del hipertexto en la extensión de UML definida en el apartado 3.3.2.4, de la misma forma que en el caso B.

La actividad de implementación incluye la implementación de la BD, que se generará a partir de los esquemas XML. La implementación del hipertexto y de la presentación se harán utilizando tecnología XML.

En caso de que los datos y organización de la BD no coincida con los del hipertexto, pero que ambos quieran ser implementados en XML, sería necesaria la utilización de plantillas de estilo XSL, que permitirían obtener diferentes vistas de los esquemas XML almacenados en la BD. Sin embargo, si existe mucha diferencia, se recomendaría utilizar almacenamiento (objeto-)relacional.

Tabla 5. Actividades para la iteración MIDAS/DB (caso C)

	3ª Iteración: Base de Datos Web (MIDAS/DB)		
Actividad	Tarea	Técnica	Notación
Análisis	*Diseño Conceptual Refinado*	*Modelos Conceptuales*	*UML*
Diseño	Diseño Lógico de Datos Diseño Lógico Refinado del Hipertexto	Modelo Lógico de Fragmentos (RMM) ⇩ Modelo Lógico Navegacional(RMM)	XML Schema (MIDAS-UML) ⇩ XLink (MIDAS-UML)
Implem.	Implementación BD Implementación del Hipertexto + Implementación Refinada de la Interfaz de Usuario		XML Schema Tecnología XML: XML/XML Schema/ XLink/XSL

3.3 Técnicas

A continuación, se van a pasar a describir las nuevas técnicas que se han propuesto en MIDAS/DB.

En primer lugar, en el apartado 3.3.1, se presentan las técnicas para la tecnología OR incluyendo una descripción de los metamodelos objeto-relacionales (el del estándar SQL:1999 y el de Oracle), las correspondientes extensiones de UML, así como la especificación de unas guías para pasar del nivel conceptual al lógico (estándar y específico) de una BDOR. También se describe, en este apartado, el modelado de consultas con UML. En este punto cabe destacar que inicialmente se comenzó trabajando con la versión 8*i* de Oracle, con lo que las extensiones se

definieron para esta versión. Sin embargo, cuando esta tesis ya estaba parcialmente realizada, apareció la nueva versión, Oracle9*i*, por lo que se comentarán también las mejoras incorporadas en esta versión de Oracle.

A continuación, en el apartado 3.3.2, se muestran las técnicas definidas para la tecnología XML, primero para el estándar XML Schema y luego para XLink. Para cada una de ellas se describe el metamodelo correspondiente y luego se define la extensión de UML. Para finalizar, se describen las reglas de transformación especificadas para pasar del nivel conceptual al nivel lógico en el diseño del hipertexto.

A la hora de especificar las técnicas de modelado en UML ha sido necesario decidir cómo extender este lenguaje. Una posible solución sería modificar el metamodelo de UML, pero esta es una solución demasiado drástica, ya que UML ha sido diseñado para que pueda extenderse de una forma controlada, mediante sus propios mecanismos de extensión. Dichos mecanismos permiten crear nuevos bloques de construcción por medio de estereotipos, valores etiquetados y restricciones. Esta es la solución que se ha adoptado en este trabajo para extender UML para nuestras necesidades.

3.3.1 Técnicas para OR

En este apartado se describen las técnicas para el diseño de bases de datos (objeto)-relacionales con UML. Aunque este trabajo, por las razones expuestas anteriormente, se centra en bases de datos (objeto-)relacionales, tanto las técnicas como las extensiones de UML propuestas podrían adaptarse fácilmente al diseño de bases de objetos puras.

3.3.1.1 Metamodelo Objeto-Relacional

3.3.1.1.1 Estándar SQL:1999

SQL: 1999 (Eisenberg y Melton, 1999) es el estándar actual para bases de datos (objeto-)relacionales. Su modelo de datos trata de integrar el modelo relacional con el modelo de objetos. Además de las extensiones para objetos, SQL: 1999 proporciona otras extensiones a SQL-92, tales como disparadores, extensiones OLAP, nuevos tipos de datos para el almacenamiento de datos multimedia, etc. Una de las diferencias principales entre el modelo relacional y el modelo objeto-relacional es que en éste ha sido eliminada la Primera Forma Normal

(1FN), la regla básica de un esquema relacional, al permitir que una columna de una tabla de objetos pueda contener un tipo de datos colección.

SQL: 1999 permite al usuario definir nuevos tipos de datos estructurados de acuerdo con los tipos de datos requeridos para cada aplicación. Los tipos de datos estructurados proporcionan a SQL: 1999 las características principales del modelo de objetos. Soporta el concepto de lenguaje fuertemente tipado, encapsulamiento, sustitabilidad, polimorfismo y vinculación dinámica.

Un tipo estructurado puede utilizarse como tipo de una tabla o como tipo de una columna. Un tipo estructurado utilizado como el tipo base en la definición de una columna permite la representación de atributos complejos; en este caso, los tipos estructurados representan un tipo valor. Un tipo estructurado utilizado como tipo base en la definición de una tabla se corresponde con la definición de un tipo de objeto (o una clase), siendo la tabla la extensión del tipo. En SQL: 1999 este tipo de tablas se llaman *tablas tipadas*. Un objeto en SQL: 1999 es una fila de una tabla tipada. Cuando se define una tabla tipada, el sistema añade una nueva columna que representa al identificador (OID) de cada objeto de la tabla (cada objeto se almacena como una fila de la tabla tipada). El valor de este atributo lo genera el sistema, es único para cada objeto y no puede ser modificado por el usuario.

Un tipo estructurado puede incluir métodos asociados que representan su comportamiento. Un método es una función SQL cuya signatura se define junto a la definición del tipo estructurado. La especificación del cuerpo del método se define de forma separada de su signatura.

SQL: 1999 soporta herencia simple para los tipos estructurados y para las tablas tipadas. Un subtipo hereda los atributos y el comportamiento del supertipo. Una subtabla hereda las columnas, restricciones, disparadores y métodos de la supertabla.

Una fila de una tabla tipada es un objeto, y se diferencia del resto de objetos por su OID. El valor del OID lo genera el sistema cuando se inserta un nuevo objeto en la tabla. El tipo de esta columna es un tipo referencia (REF). Hay diferentes tipos REF, uno por cada tipo de objeto, es decir, el tipo REF es un constructor de tipos más que un tipo en si mismo. Un atributo definido como un tipo referencia contiene el OID del objeto referenciado, por tanto, el tipo REF permite la implementación de relaciones sin necesidad de utilizar claves ajenas.

SQL: 1999 soporta otro tipo estructurado: el tipo ROW. El tipo ROW es un tipo estructurado definido por el usuario. No tiene extensión ni OID; por tanto, no puede utilizarse como un tipo de objeto.

SQL: 1999 únicamente soporta un tipo colección: el ARRAY. Un ARRAY puede utilizarse en cualquier parte donde se pueda usar cualquier otro tipo (como tipo de un atributo de un tipo estructurado, como el tipo de una columna, etc.). El tipo ARRAY permite la representación de atributos multivaluados, permitiendo que las tablas no estén en Primera Forma Normal.

La Figura 12 muestra el metamodelo de SQL: 1999 en UML.

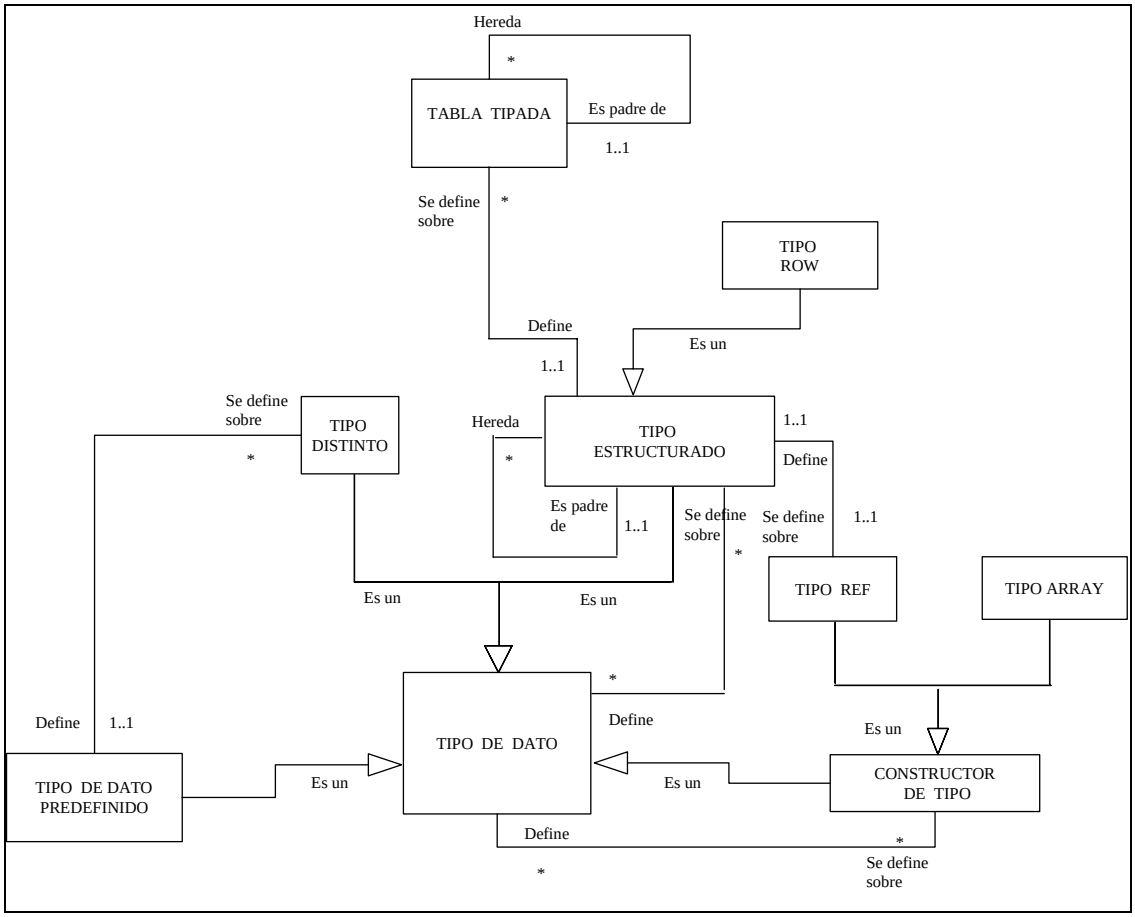


Figura 12. Metamodelo de SQL:1999

3.3.1.1.2 Producto: Oracle

Al igual que el estándar, Oracle8*i* (Oracle Corporation, 2000) soporta la definición por parte del usuario de tipos de datos estructurados (aunque con una sintaxis diferente). Un tipo estructurado puede utilizarse, al igual que en SQL:

1999, como un tipo de columna o como un tipo de una tabla. Un tipo estructurado utilizado como un tipo de columna representa un tipo valor y un tipo estructurado utilizado como el tipo de una tabla representa un tipo de objeto, siendo la tabla la extensión del tipo. Cada fila de este tipo de tablas es un objeto y, al igual que en SQL: 1999, posee una columna especial de tipo referencia (REF) que permite la identificación de cada objeto (OID). También es posible definir un atributo como una referencia a un tipo de objeto. Oracle8*i* permite asociar comportamiento a los tipos de objeto, definiendo la signatura de los métodos como parte de la definición del tipo. El cuerpo del método se define de forma separada.

A diferencia del estándar, Oracle8*i* soporta dos tipos de colección: VARRAYS, equivalente a los ARRAY de SQL:1999 y *Nested Tables*. Una Nested Table es una tabla anidada en otra tabla. Es posible definir un tipo de datos tabla y utilizar este tipo como un tipo de columna de otra tabla. Por tanto, esta columna contiene una tabla (llamada Nested Table) con una colección de valores, objetos o referencias. Un tipo colección en Oracle8*i* puede estar definido sobre cualquier tipo excepto sobre un tipo colección. Sin embargo, esta restricción desaparece en Oracle9*i*, permitiendo la definición de colecciones multinivel (Oracle Corporation, 2001).

Una de las principales diferencias entre los modelos objeto-relacional de Oracle8*i* y SQL: 1999 es que Oracle8*i* no soporta la herencia, ni de tipos ni de tablas. Sin embargo, la nueva versión, Oracle9*i*, soporta herencia simple de tipos, de tal forma que un subtipo puede crearse basado en otro tipo, del cual heredará todos los atributos y métodos. En el subtipo podrán añadirse nuevos atributos y métodos, y/o redefinir los métodos heredados (Oracle Corporation, 2001).

La Figura 13 muestra el metamodelo de Oracle 8*i* en UML. La figura incluye también la herencia (correspondiente a la versión 9*i*) con una línea punteada.

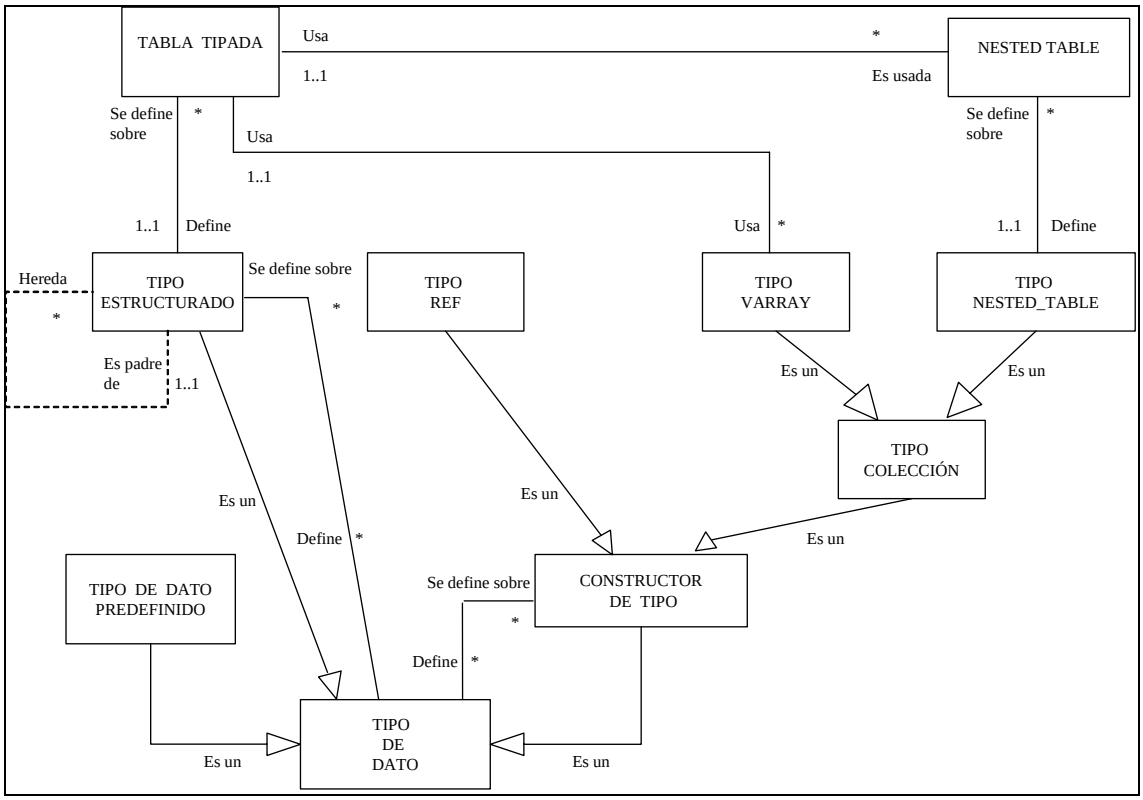


Figura 13. Metamodelo de Oracle

3.3.1.2 Extensión de UML para OR

Para poder utilizar UML como lenguaje estándar para el modelado de BD, hay que modificarlo, de tal forma que permita representar un modelo objeto-relacional. De esta forma, se podrán representar los constructores definidos anteriormente, tales como los ARRAYs, NESTED TABLEs o el tipo REF.

Esta extensión de UML define un conjunto de estereotipos, valores etiquetados y restricciones que permiten modelar aplicaciones que trabajen con bases de datos (objeto-)relacionales. Las extensiones se definen para ciertos componentes que son específicos de los modelos objeto-relacionales, permitiendo su representación en el mismo diagrama que describe el resto del sistema. La extensión se basa en los modelos objeto-relacionales de SQL: 1999 y Oracle8*i* (indicando las diferencias con respecto a la versión 9*i*). Cada elemento del modelo objeto-relacional debe tener una representación en UML. Para elegir el elemento estereotipado se utilizan los siguientes criterios:

Para SQL:1999 se han considerado tipos estructurados y tablas tipadas como clases estereotipadas porque son definidas explícitamente en el esquema en SQL.

El resto de los tipos (REF, ROW y ARRAY) se consideran como atributos estereotipados.

Para Oracle8*i* se han considerado tipos de objetos, tablas tipadas y tablas anidadas (nested tables) como clases estereotipadas por el mismo motivo que en el caso del SQL:1999. El tipo REF se ha considerado como un atributo estereotipado porque no puede ser definido explícitamente en el esquema en SQL. Puesto que un VARRAY puede, o no, ser definido explícitamente en el esquema, se permitirán las dos posibilidades: definirlo como un atributo estereotipado o como una clase estereotipada. Se utilizará la clase estereotipada cuando el tipo VARRAY se haya definido explícitamente en el esquema. De esta forma, el diagrama UML con estereotipos para Oracle8*i* ayudará al desarrollador en el proceso de compilación (la compilación de estos esquemas en Oracle8*i* es una tarea tediosa debido a que los tipos deben ser recompilados, por lo que esta técnica puede ser de ayuda para el usuario).

Un prerrequisito de las extensiones que se definen a continuación, es que ya estén definidas las extensiones requeridas para el modelo relacional (Ambler, 1999; Naiburg, 2000).

3.3.1.2.1 Estereotipos para SQL:1999

Tipo Estructurado

Clase del Metamodelo: Clase

Descripción: Un <<udt>> permite la representación de nuevos tipos de datos definidos por el usuario (User Data Types).

Icono: Ninguno

Restricciones: Únicamente puede utilizarse para definir tipos valor

Valores Etiquetados: Ninguno

Ejemplo:

```
CREATE TYPE TipoDireccion AS
( calle VARCHAR(20)
, numero NUMBER
, ciudad  VARCHAR(20)
, provincia  VARCHAR(10))
```

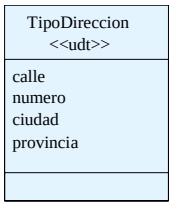
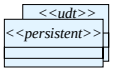


Tabla Tipada

Clase del Metamodelo: Clase

Descripción: Se define como <<persistent>>. Representa una clase del esquema de la BD, y se definirá como una tabla de un tipo de dato estructurado.

Icono: 

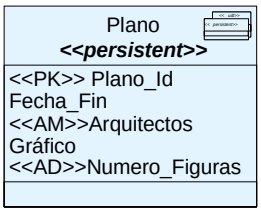
Restricciones: Una tabla tipada implica la definición de un tipo estructurado, que es el tipo de la tabla.

Valores Etiquetados: Ninguno

Ejemplo:

```
CREATE TYPE Plano AS
( Plano_Id INTEGER,
  Fecha_Fin DATE,
  Numero_Figuras INTEGER,
  Grafico BLOB)

CREATE TABLE Tabla_Plano OF Plano
```



Asociación Compose

Clase del Metamodelo: Asociación

Descripción: Una asociación <<compose>> es una interrelación especial que une un tipo de datos definido por el usuario <<udt>> o una clase persistente con la clase que lo utiliza. Es una interrelación unidireccional. La dirección de la asociación se representa por medio de una flecha que *apunta* a *la clase que utiliza* el tipo definido por el usuario o la clase persistente.

Icono: Ninguno

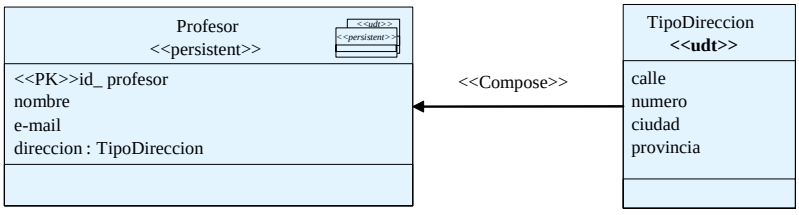
Restricciones: Únicamente puede utilizarse para unir una clase <<persistent>> con una clase <<udt>> o dos clases <<persistent>>, cuando una clase referencia a otra.

Valores Etiquetados: Ninguno

Ejemplo:

```
CREATE OR REPLACE TYPE TipoDireccion
( calle VARCHAR(20)
, numero NUMBER
, ciudad  VARCHAR(20)
, provincia  VARCHAR(10))

CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, direccion TipoDireccion))
```



Tipo REF

Clase del Metamodelo: Atributo

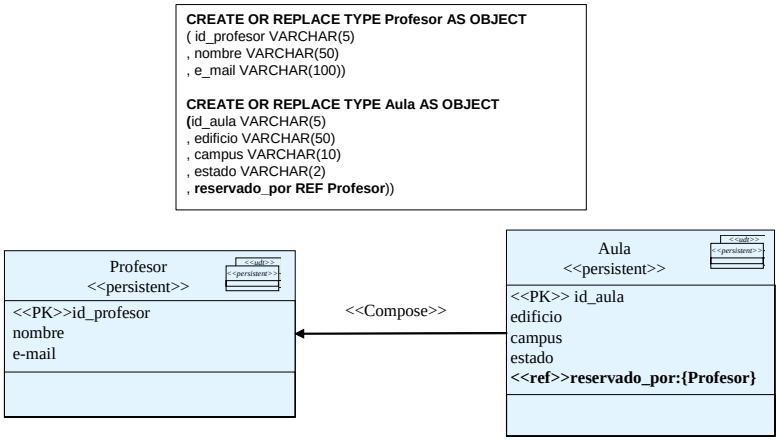
Descripción: Un tipo <<ref>> representa una referencia a una clase <<persistent>>.

Icono: ●→

Restricciones: Un atributo <<ref>> únicamente puede hacer referencia a una clase <<persistent>>

Valores Etiquetados: La clase <<persistent>> a la que hace referencia

Ejemplo:



ARRAY

Clase del Metamodelo: Atributo

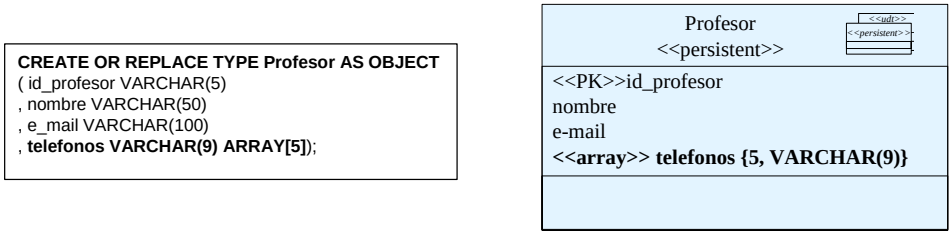
Descripción: Un <<array>> representa un tipo colección indexado y limitado.

Icono: ☐☐☐

Restricciones: Los elementos de un <<array>> pueden ser de cualquier tipo excepto de tipo <<array>>.

Valores Etiquetados: Los tipos básicos del array. El número de elementos.

Ejemplo:



Tipo ROW

Clase del Metamodelo: Atributo

Descripción: Un tipo <<row>> representa un atributo compuesto con un número fijo de elementos, cada uno de los cuales puede ser de distinto tipo de datos.

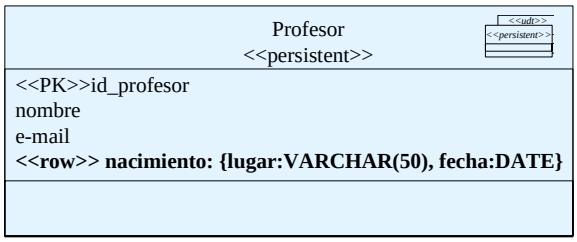
Icono: ☐

Restricciones: No puede tener métodos.

Valores Etiquetados: El nombre de cada elemento y su tipo de datos.

Ejemplo:

```
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, nacimiento ROW (lugar VARCHAR(50),fecha DATE))
```



Método Redefinido

Clase del Metamodelo: Método

Descripción: Un método <<redef>> es un método heredado que se implementa de nuevo en la clase hija.

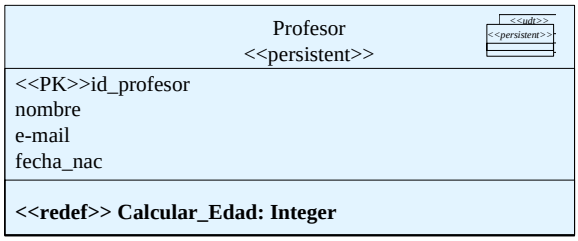
Icono: Ninguno

Restricciones: Ninguna

Valores Etiquetados: La lista de parámetros del método con sus tipos de datos. El tipo de datos devuelto por el método.

Ejemplo:

```
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, fecha_nac DATE
METHOD Calcular_Edad RETURNS INTEGER)
```



Método Diferido

Clase del Metamodelo: Método

Descripción: Un método <<def>> es un método en el que se difiere su implementación a su clase hija.

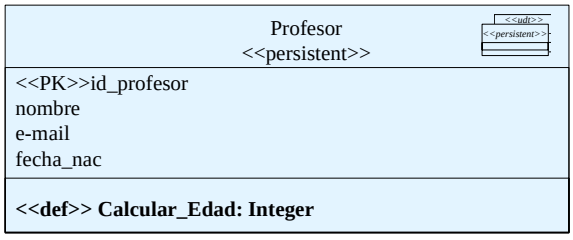
Icono: Ninguno

Restricciones: Se debe definir en una clase con clases hijas.

Valores Etiquetados: La lista de parámetros del método con sus tipos de datos. El tipo de datos devuelto por el método.

Ejemplo:

```
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, fecha_nac DATE
METHOD Calcular_Edad RETURNS INTEGER)
```



3.3.1.2.2 Estereotipos para Oracle

Tipo de Objeto

Clase del Metamodelo: Clase

Descripción: Un <<udt>> permite la representación de nuevos tipos de datos definidos por el usuario. Se corresponde con el tipo estructurado de SQL:1999.

Icono: Ninguno

Restricciones: Únicamente puede utilizarse para definir tipos valor.

Valores Etiquetados: Ninguno

Ejemplo:

```
CREATE OR REPLACE TYPE TipoDireccion AS OBJECT
( calle VARCHAR(20)
, numero NUMBER
, ciudad  VARCHAR(20)
, provincia  VARCHAR(10));
```

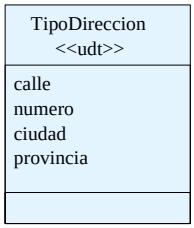
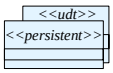


Tabla de Objeto

Clase del Metamodelo: Clase

Descripción: Se define como <<persistent>>. Representa una clase del esquema de la base de datos que deberá definirse como una tabla de un tipo de objeto. Se corresponde con las tablas tipadas de SQL:1999.

Icono: 

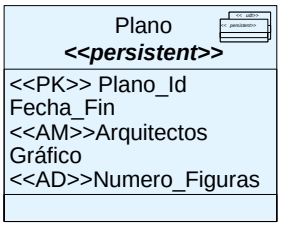
Restricciones: Una tabla tipada implica la definición de un tipo estructurado, que es el tipo de la tabla.

Valores Etiquetados: Ninguno

Ejemplo:

```
CREATE TYPE Plano AS OBJECT
( Plano_Id NUMBER,
  Fecha_Fin DATE,
  Numero_Figuras NUMBER,
  Grafico BFILE);

CREATE TABLE Tabla_Plano OF Plano;
```



Asociación Compose

Clase del Metamodelo: Asociación

Descripción: Una asociación <<compose>> es un tipo especial de interrelación que une un tipo definido por el usuario (<<udt>>, <<array>> o <<nt>>) o una clase persistente con la clase que lo utiliza. Es una interrelación unidireccional. La dirección de la asociación se representa mediante una flecha que *apunta* a la *clase que utiliza* el tipo de datos o la clase.

Icono: Ninguno

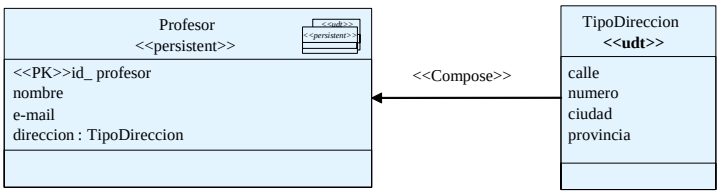
Restricciones: Únicamente puede utilizarse para unir una clase <<persistent>> con una clase <<udt>>, <<array>>, <<nt>> o <<persistent>>

Valores Etiquetados: Ninguno

Ejemplo:

```
CREATE OR REPLACE TYPE TipoDireccion AS OBJECT
( calle VARCHAR(20)
, numero NUMBER
, ciudad  VARCHAR(20)
, provincia  VARCHAR(10));

CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, direccion TipoDireccion);
```



Tipo REF

Clase del Metamodelo: Atributo

Descripción: Un <<ref>> representa una referencia a alguna clase <<persistent>>.

Icono: ●→

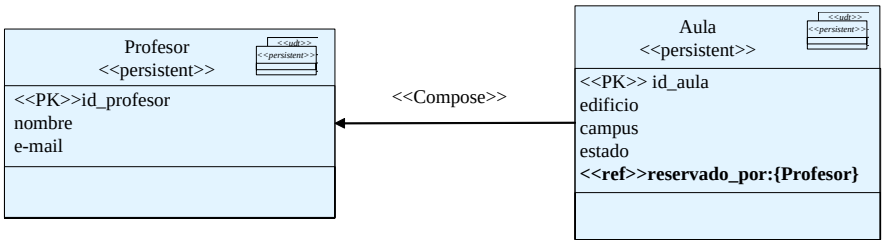
Restricciones: Un atributo <<ref>> únicamente puede hacer referencia a una clase <<persistent>>

Valores Etiquetados: La clase <<persistent>> a la que se refiere.

Ejemplo:

```
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100));

CREATE OR REPLACE TYPE Aula AS OBJECT
(id_aula VARCHAR(5)
, edificio VARCHAR(50)
, campus VARCHAR(10)
, estado VARCHAR(2)
, reservado_por REF Profesor));
```



VARRAY

Clase del Metamodelo: Atributo/Clase

Descripción: Un <<array>> representa un tipo colección indexado y limitado. Se corresponde con el tipo array en SQL:1999.

Icono: 

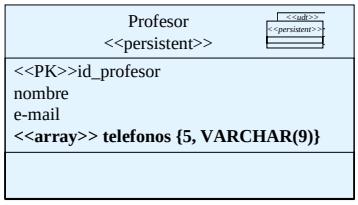
Restricciones: Los elementos de un <<array>> pueden ser de cualquier tipo de datos excepto de otro tipo colección (<<array>> o <<nt>>)

Valores Etiquetados: Los tipos básicos del array. El número de elementos.

Ejemplos:

Opción 1)

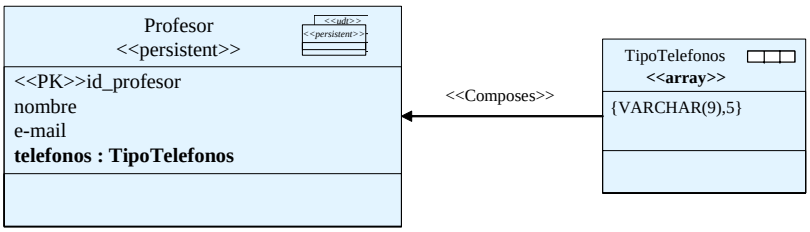
```
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, telefonos VARRAY(5) OF VARCHAR(9) );
```



Opción 2)

```
CREATE TYPE TipoTelefonos
            AS VARRAY(5) OF VARCHAR(9);

CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, telefonos Tipo_Telefonos );
```



Nested Table

Clase del Metamodelo: Clase

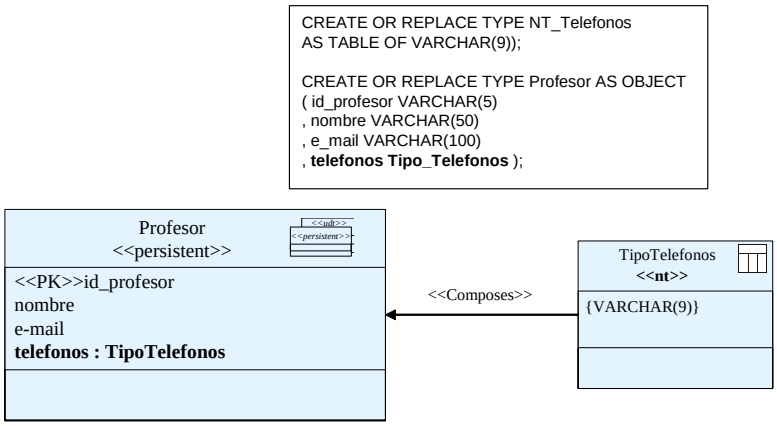
Descripción: Una <<nt>> representa un tipo colección no indexado ni limitado.

Icono: 

Restricciones: Los elementos de la <<nt>> pueden ser de cualquier otro tipo de datos excepto otro tipo colección (<<array>> o <<nt>>).

Valores Etiquetados: El tipo básico de la nested table.

Ejemplo:



Reglas que ha de cumplir un diseño bien formado:

- Cada clase <<udt>>, <<array>> o <<nt>> tiene que unirse por medio de una asociación <<compose>> con otra clase del esquema.
- Un atributo <<ref>> en una clase <<persistent>> implica una asociación con otra clase.
- Una clase <<persistent>> que contiene un atributo colección cuyos elementos sean objetos de una clase <<persistent>> o <<ref>> a estos objetos, implica una asociación entre ambas clases.
- Cada clase <<persistent>> corresponde en SQL:1999 a un tipo estructurado, con su correspondiente extensión. La extensión es la tabla tipada. Cada clase <<persistent>> en Oracle8*i* y 9i corresponde con un tipo de objeto con su correspondiente extensión. La extensión es la tabla de tipo de objetos. Es decir, el tipo de objeto y su extensión se representan en UML extendido como una clase <<persistent>>.
- Esta extensión tiene en cuenta los modelos objeto-relacionales de SQL:1999 y Oracle8*i*. Debería modificarse para su adaptación a las nuevas versiones de ambos modelos. Por ejemplo, en el caso de Oracle9*i*, (Oracle Corporation, 2001), se permiten las colecciones multinivel, por lo que los elementos de cualquier tipo colección (<<array>> o <<nt>>) pueden ser cualquier otro elemento de tipo colección (<<array>> o <<nt>>). Además, si se desean utilizar otros productos, la extensión debería adaptarse a los mismos.

3.3.1.3 Guías de Transformación

De forma similar a como las metodologías para bases de datos relacionales proponen algunas reglas para la transformación de un modelo conceptual en un modelo lógico estándar, en este método también se propone una técnica que determina la transformación de un modelo de un nivel de abstracción al siguiente. Esta técnica sugiere algunas reglas que deberían tenerse en cuenta, únicamente como guías.

A continuación, se presentan las reglas básicas de transformación de un modelo conceptual de datos en UML a un modelo Objeto-Relacional de SQL:1999 así como a uno de Oracle8*i* (ó 9*i*).

- **Transformación de Clases**

Para transformar una clase persistente de UML en una clase de SQL: 1999 o de Oracle, es necesario definir tanto el tipo de objeto, tipo estructurado en SQL: 1999, como la extensión del mismo, que será una tabla tipada. En el caso del SQL: 1999 la extensión es una tabla definida sobre el tipo estructurado y en Oracle es una tabla definida sobre el tipo de objeto.

Cada uno de los *atributos* de la clase UML pasa a ser un atributo del tipo. Debido a que ni SQL: 1999 ni Oracle soportan niveles de visibilidad, éstos desaparecen en las etapas de diseño e implementación. Si se quisieran implementar habría que recurrir a la definición de vistas, permisos, etc. Los *atributos multivaluados* se representan, tanto en SQL: 1999 como en Oracle, mediante un tipo colección. En SQL: 1999 se utilizará el tipo ARRAY, ya que es el único que soporta, mientras que en Oracle puede escogerse entre el VARRAY (si se conoce el número máximo valores posibles) o la tabla anidada, denominada NESTED TABLE (en el caso de que el número de valores posibles varíe mucho entre los elementos del tipo). La posibilidad de definir un atributo multivaluado sin necesidad de crear otra tabla asociada, rompe con la obligatoriedad de la 1FN del modelo relacional. Los *atributos derivados* se pueden implementar, tanto en SQL:1999 como en Oracle, de dos formas: mediante un disparador o mediante un método. Los *atributos compuestos* también pueden representarse en el modelo Objeto-Relacional sin necesidad de crear una tabla asociada, transformándose a SQL: 1999 mediante un tipo ROW o un tipo estructurado y a Oracle mediante un tipo objeto.

- **Transformación de Operaciones**

Las operaciones de cada clase UML se transformarán en SQL: 1999 y en Oracle especificando la cabecera en la definición del tipo de objeto, quedando así ligados al tipo que pertenecen. El cuerpo del método se especifica por separado.

- **Transformación de Asociaciones**

Las asociaciones de UML pueden representarse en SQL: 1999 y en Oracle bien como asociaciones unidireccionales, o bien como asociaciones bidireccionales. Cuando se sabe que las consultas van a recorrer la asociación en los dos sentidos puede ser recomendable definir asociaciones bidireccionales, ya que permiten mejorar los tiempos de respuesta. Sin embargo, este tipo de asociaciones tienen un mayor coste de mantenimiento. En el caso de que el diagrama UML muestre las relaciones de navegabilidad, éstas nos indicarán la dirección en la que ha de implementarse la asociación.

Según la cardinalidad máxima de la asociación se realizará la siguiente transformación:

1:1. Se implementa mediante un atributo de tipo REF en cada tipo de objeto que participa en la relación. Si la cardinalidad mínima es 1, sería también necesario imponer la restricción NOT NULL al atributo REF en la tabla tipada.

1:N. Se transforma, en SQL:1999, incluyendo un atributo de tipo REF en el tipo de objeto de cardinalidad N y un atributo de tipo ARRAY de REF en el tipo de objeto con cardinalidad 1. En Oracle en lugar de VARRAYs de referencias se pueden utilizar NESTED TABLEs, ya que este constructor permite mantener colecciones de elementos sin una dimensión predefinida. En el caso de que se conozca la cardinalidad máxima (por ejemplo, a una clase pueden asistir como máximo 10 alumnos), sería más conveniente utilizar también un VARRAY. Debido a que en SQL: 1999 no hay NESTED TABLEs, ni otro tipo de colección similar, aunque la relación sea de tamaño variable, es necesario utilizar un ARRAY.

N:M. Se transforma a SQL:1999 mediante la definición de un atributo ARRAY de referencias en cada tipo de objeto implicado en la relación. En Oracle se utilizará una NESTED TABLE, tal y como se ha visto en el caso de las relaciones 1:N.

- **Transformación de Generalizaciones**

SQL: 1999 soporta directamente el concepto de generalización, tanto de tipos como de tablas tipadas. La definición se realiza incluyendo una cláusula UNDER en la especificación de cada uno de los subtipos, indicando el supertipo del que heredan (sólo se permite herencia simple). Es necesario también, mediante la definición de cláusulas UNDER en las subtablas, especificar la correspondiente jerarquía de tablas.

Oracle8*i* no soporta herencia, por lo que las generalizaciones se implementarán, al igual que en el modelo relacional, mediante claves ajenas (o tipos REF) y restricciones (CHECK, aserciones y disparadores) u operaciones que permitan simular su semántica. A diferencia de Oracle8*i*, la versión 9*i* de Oracle incorpora el concepto de generalización, soportando herencia de tipos y de vistas.

Existen otros productos comerciales que también implementan ya la herencia. Este es, por ejemplo, el caso del *Universal Server de Informix* que aunque no soporta la semántica completa de la herencia (por ejemplo, la herencia de tablas es sólo de atributos), permite definirla y soportar algunas de sus características.

- **Transformación de Agregaciones**

Las agregaciones son un tipo de relación que se representan, tanto en SQL: 1999 como en Oracle, definiendo en el tipo de objeto compuesto (en el *todo*) un atributo de tipo colección. En SQL: 1999 la colección será un ARRAY de referencias. En Oracle la colección será una NESTED TABLE salvo que el número máximo de elementos componentes sea fijo en cuyo caso se podría representar mediante un VARRAY.

Sin embargo, con esta regla no se recoge la diferencia existente entre las agregaciones simples y las composiciones. La agregación simple distingue únicamente el *todo* de las *partes*. La composición, sin embargo, es un tipo especial de agregación, donde las *partes* mantienen una fuerte relación de pertenencia con respecto al *todo* y no pueden existir si no existe el *todo*. Conociendo esta diferencia se van a definir guías específicas para cada uno de los casos.

Agregaciones Simples. Para representar este tipo de agregación en un modelo OR se propone definir el *todo* como un atributo de tipo colección. Esta colección contendrá referencias a sus *partes*, es decir, referencias a los objetos que componen el *todo*. En SQL: 1999 la colección se puede definir con un ARRAY de

referencias en el *todo*, siendo necesario especificar un número máximo de *partes*. En Oracle la colección puede ser una NESTED TABLE, en el caso de no conocer el número de *partes* que puede tener un *todo*. Si se conoce el máximo número de componentes (cardinalidad máxima) sería más recomendable usar un VARRAY.

Se proponen definir la colección como un conjunto de referencias porque una agregación simple es una agregación en la que cada *parte* puede pertenecer a más de un *todo*, además no implica ningún tipo de restricción sobre el tiempo de vida con respecto al *todo*.

Composiciones. La composición, como ya se ha indicado previamente, es un tipo especial de agregación en la que las *partes* están físicamente ligadas al *todo*, es decir, su tiempo de vida depende de la del *todo*. Por lo tanto, una composición define tres restricciones con respecto al concepto de agregación simple:

Restricción 1: Una *parte* no puede pertenecer a más de un *todo* simultáneamente.

Restricción 2: Una vez creada una *parte*, vive y muere con el *todo*.

Restricción 3: Una *parte* se puede borrar explícitamente antes de borrar el *todo* asociado.

La transformación del concepto de composición definido en UML a un diseño objeto-relacional dependerá del modelo al que se va a transformar. Si se va a transformar al estándar SQL: 1999 se hará de la misma forma que con las agregaciones simples, es decir, incluyendo un atributo en la especificación del *todo*. Como SQL: 1999 proporciona un único tipo colección, que es el ARRAY, el atributo que se incluye en el *todo* será un ARRAY. Las restricciones anteriormente citadas deberán ser implementadas por medio de CHECKs, aserciones y/o disparadores.

Sin embargo, Oracle permite implementar directamente el concepto de composición manteniendo las diferencias con respecto al concepto de agregación. Esto se debe a que este producto además de soportar el tipo colección VARRAY, también proporciona las NESTED TABLEs. Una NESTED TABLE es un tipo colección que es una tabla. Al ser una tabla, se puede definir como una tabla de objetos. Por lo tanto, una NESTED TABLE puede contener las *partes* de un *todo* tanto como objetos como referencias. Al mismo tiempo la NESTED TABLE está embebida como una columna de otra tabla de objetos (el *todo*). Al implementar de esta forma las composiciones, se satisfacen las tres restricciones definidas previamente. Por lo

tanto, no se necesita recurrir a CHECKs, aserciones o disparadores para implementar la composición con la semántica que lleva asociada.

Las NESTED TABLE de Oracle permiten por lo tanto, implementar composiciones y agregaciones simples manteniendo las diferencias de semántica existentes entre ellas, de la misma forma que lo hace UML.

En la Tabla 6 se resumen las guías propuestas para el diseño de BDOR.

Tabla 6. Guías de diseño de bases de datos objeto-relacionales.

UML	SQL:1999	Oracle(8*i* y 9*i*)
Clase	Tipo Estructurado	Tipo de Objeto
Extensión de la Clase	Tabla Tipada	Tabla de Tipos de Objeto
Atributo	Atributo	Atributo
Multivaluado	ARRAY	VARRAY
Compuesto	ROW / Tipo Estructurado en columna	Tipo de Objeto en columna
Calculado	Trigger/Método	Trigger/Método
Asociación		
Uno-a-Uno	REF/REF	REF/REF
Uno-a-Muchos	REF/ARRAY	REF/Nested Table
Muchos-a-Muchos	ARRAY/ARRAY	Nested Table/Nested Table
Agregación	ARRAY	Nested Table
Generalización	Herencia simple de Tipos	Herencia simple de Tipos (*sólo en Oracle9i*)
Generalización	Herencia simple de Tablas Tipadas	Oracle no permite representar el concepto de generalización entre tablas.

3.3.1.4 Modelado de Consultas

En este apartado se describe el modelado de consultas mediante UML, así como su integración con el diseño del hipertexto.

3.3.1.4.1 Diseño de Consultas en UML

Habitualmente, las metodologías de desarrollo de Sistemas de Información, incluyendo aquellas que son específicas de BD, no tienen en cuenta el modelado y diseño de las consultas a la BD. Se parte de los requisitos de usuario y se pasa a la implementación de las mismas, directamente en SQL. Sin embargo, es perfectamente posible realizar el diseño de consultas a un nivel de abstracción que

sea independiente de la implementación, utilizando UML para facilitar su comprensión y mantenimiento y con herramientas que soporten su generación automática del mismo modo que con el resto del SIW.

En Akehurst y Bordbar (2001) se propone una extensión de OCL (Object Constraint Language) para el modelado de consultas SQL. Sin embargo, a nivel conceptual, es preferible utilizar una notación gráfica ya que ésta facilita la comprensión y legibilidad de las consultas. Además, el uso de OCL para la definición de consultas no aportaría ninguna ventaja sobre otros lenguajes existentes, como el álgebra relacional o el estándar SQL: 1999, que como el OCL, son independientes de plataforma. Por todo ello, en MIDAS/DB se propone la utilización de UML para definir consultas tanto a nivel conceptual como a nivel de diseño lógico. En este último caso se especificará el caso para Oracle y no para el estándar SQL: 1999 ya que, en lo que a consultas se refiere, éste puede considerarse un subconjunto de Oracle.

Una consulta se puede ver como un subesquema. Por tanto, este subesquema se puede representar usando la misma notación que se usa para representar el esquema, es decir, UML. Sólo las clases involucradas en la consulta aparecerán en el diagrama resultante, y, dentro del mismo, sólo los atributos que se desea que aparezca en el resultado de la consulta. Las restricciones involucradas en la consulta pueden ser especificadas mediante alguna de las opciones que proporciona UML (por ejemplo, restricciones, cardinalidades, etc.), incluyendo OCL o alguna extensión para este lenguaje.

Una consulta, al igual que cualquier otro esquema, podrá representarse a distintos niveles de abstracción. Si la consulta se representa a nivel conceptual, entonces ésta será un subesquema del esquema conceptual y si se trata de una consulta a nivel lógico, dicha consulta será un subesquema del esquema lógico. En MIDAS/DB, ambos esquemas se representan en UML con los estereotipos apropiados.

Por lo tanto, modelar consultas en UML es similar a modelar esquemas de bases de datos. Con el fin de poder modelar consultas en UML se ha definido un nuevo estereotipo: ***<<query>>***. Debido a que, como ya se ha dicho, las consultas se modelarán como subesquemas, esta extensión se aplica al mismo tipo de elemento UML al que se aplica el estereotipo <<Schema>>, al *Paquete*.

La Figura 14 muestra, representado mediante un diagrama de clases en UML, el esquema genérico de la extensión para consultas propuesta. Una base de datos, representada con el estereotipo <<Database>>), estará formada por un conjunto de esquemas, <<Schema>>. A su vez, una consulta, <<Query>>, será un subesquema de un esquema dado. Una consulta podrá afectar a distintos esquemas (en el caso más general, éstos formarán una base de datos distribuida). En esta tesis se estudia el caso de un único esquema, aunque es fácilmente extensible al caso de varios.

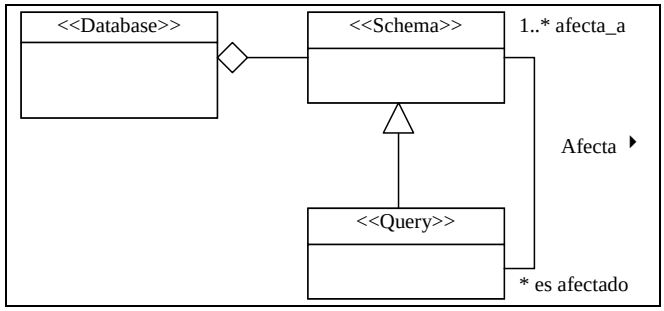


Figura 14. Metamodelo del sistema

Cada una de las clases que forman parte del subesquema (bien sea subesquema del esquema conceptual o del lógico) que representa la consulta, vendrá etiquetada con el identificador del subesquema correspondiente a la consulta (q1, q2, ...). Únicamente se reflejarán aquellas clases afectadas de alguna forma en la consulta y, dentro de éstas, aquellos atributos que deban aparecer en el resultado. Cada consulta implicará un conjunto de restricciones que se reflejarán en el subesquema UML correspondiente a dicha consulta, utilizando algunas de las posibilidades que permite UML para su especificación. En general, existen distintas posibilidades de expresar las restricciones asociadas a una consulta: restricciones junto al atributo afectado; restricciones en una clase; cardinalidades; restricciones especificadas como notas en OCL, etc.

El tratamiento de las consultas anidadas será diferente según la parte de la consulta que contenga a la subconsulta. Se pueden distinguir tres casos:

- La subconsulta está incluida en el select (i.e., SELECT (SELECT ...) FROM ... WHERE....). En este caso, la subconsulta es un subesquema que, a su vez, actúa como un atributo del subesquema que representa la consulta principal.

- La subconsulta está incluida en el from (i.e., SELECT... FROM (SELECT ...) WHERE...). En este caso, la propia subconsulta aparece en la consulta principal como una clase estereotipada, análoga a las utilizadas en las consultas anteriores. En el nombre de las clases debe aparecer el nombre del subesquema correspondiente a la subconsulta anidada.
- La subconsulta está incluida en el where (i.e., SELECT ... FROM ... WHERE X in (SELECT ...)). En este caso, la subconsulta podría verse como un subesquema de la consulta. Es decir, la consulta utilizaría el subesquema correspondiente a la subconsulta de modo análogo a como hacíamos en el caso de crear una consulta utilizando el esquema inicial.

Como ya se mencionó en el apartado 3.2.2., en el diseño del hipertexto están involucrados, además del modelo de fragmentos y el de navegación, el modelo de consultas.

3.3.1.4.2 Integrando el Modelo de Consultas en el Modelo Navegacional

En el modelo de fragmentos la información se agrupa y estructura en fragmentos que se corresponderán con páginas Web. El modelo de navegación completa el modelo de fragmentos incluyendo los elementos de acceso, con el fin de determinar cómo navegar de un fragmento a otro (mediante un índice, un menú, etc.). Cada una de las estructuras de acceso corresponde a una consulta, más o menos compleja, a la BD.

Por ejemplo, supongamos el sitio Web de Kybele; supongamos que se desea navegar de la página de docencia, donde aparecen todas las asignaturas impartidas por los miembros del grupo, a la página concreta de la asignatura de BD. Esta posibilidad de acceder desde la página de docencia a la página de la asignatura de BD se representará, a nivel conceptual, mediante un enlace entre los dos fragmentos implicados: docencia y asignatura. Dicho enlace se completará en el diagrama de navegación indicando la estructura de acceso que se desea; por ejemplo, un índice que contenga todas las asignaturas. Es claro que al seleccionar una asignatura del índice se produce una consulta a la BD para extraer la información relativa a la asignatura seleccionada.

En el modelado del hipertexto, se tienen en cuenta los fragmentos y las estructuras de navegación. Sin embargo, las estructuras de navegación se representan mediante un icono cuyo único significado es qué se desea obtener pero

no cómo. Es decir, la semántica implícita a toda estructura de navegación, la consulta, no se representa. En MIDAS/DB se propone representar esta semántica mediante diagramas UML tal como se ha explicado en la sección anterior.

El modelado de las consultas también se puede llevar a cabo en dos pasos, como el resto del diseño del hipertexto:

- Cada fragmento del modelo de fragmentos agrupa información que puede provenir de diferentes clases del modelo conceptual de datos. Por lo tanto, un fragmento implícitamente lleva asociado una consulta.
- Cada estructura de navegación tiene dos significados: por un lado, la forma visual de navegar de un fragmento a otro (a través de un índice, un recorrido guiado, etc.) y por otro lado, también representará una consulta.

De hecho la consulta que representa la semántica de un elemento de acceso se puede obtener en dos pasos:

- El primer paso se lleva a cabo de forma conjunta con el modelado de fragmentos. En este momento, la consulta representará cómo obtener del modelo conceptual de datos la información de cada fragmento. Es decir, para cada fragmento se definirá una consulta. Esta consulta representará las clases del modelo conceptual de datos involucradas para obtener la información que se muestra en el modelo de fragmentos, así como las restricciones necesarias
- El segundo paso se realizará junto con el modelado de navegación. En este punto, cada consulta que se haya definido previamente se completará añadiendo la restricción asociada al elemento de acceso incluido en el fragmento correspondiente. Además, si es posible navegar de un fragmento a otro a través de dos o más estructuras de navegación, la consulta asociada con el fragmento destino se desarrollará en dos o más consultas diferentes, una por cada estructura de navegación incluida.

La Figura 15 representa las relaciones entre los tres modelos conceptuales involucrados en el modelado del hipertexto.

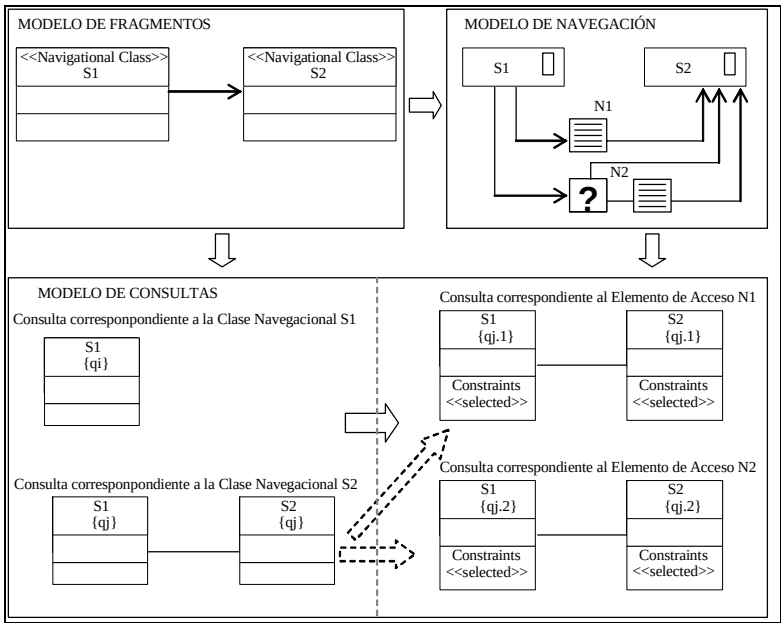


Figura 15. Relaciones entre los modelos de fragmentos, navegación y consultas

Como en la mayoría de los casos el *modelo de fragmentos* tendrá muchas diferencias con respecto al modelo conceptual de datos y además un modelo de fragmentos con un elevado número de expresiones OCL sería difícil de comprender, se propone diseñar un modelo de consultas para cada clase navegacional del modelo de fragmentos. Cada modelo de consultas estará basado en el modelo conceptual de datos y representará la forma en la que se obtienen los datos de los fragmentos.

El *modelo conceptual de navegación* representa cómo se realiza la navegación entre las clases navegacionales. Habitualmente sus atributos no se representan porque son los mismos que los de las clases de navegación del modelo conceptual de fragmentos. En el modelo conceptual de navegación se introducen nuevas clases estereotipadas: *<<index>>, <<guided tour>>, <<index guided tour>> <<query>> y <<menu>>.* La diferencia entre el elemento de acceso <<query>> propuesto en la UWE (Koch *et al.*, 2000) (que se representa por un signo de interrogación en el modelo de navegación) y el paquete <<query>> es que el segundo realmente representa una consulta, mientras que el primero representa tan sólo una búsqueda por “palabras” introducidas por el usuario. Este proceso obviamente también lleva asociado una consulta, pero de la misma forma que el resto de elementos de acceso, que también son consultas. Es decir, el paquete <<query>> puede representar cualquier tipo de consultas, tanto una consulta

"típica" a una BD, como las consultas derivadas de los elementos de acceso de las estructuras de navegación; incluyendo el elemento de acceso que habitualmente se conoce como consulta y que se representa, como ya se ha dicho, con un signo de interrogación en el modelo de navegación.

Quedan, en este punto, dos líneas de trabajo abiertas: por una parte, la utilización de las consultas más frecuentes para refinar el diseño de la BD; y por otra, la extensión del modelado del hipertexto para incluir estructuras de navegación que permitan representar consultas complejas.

3.3.2 Técnicas para XML

3.3.2.1 Metamodelo XML Schema

Un XML Schema (o esquema XML) es la definición de una estructura XML específica. El lenguaje XML Schema del W3C (W3C, 2001b) especifica cómo se define cada tipo de elemento en el esquema y qué tipo de datos tiene asociado dicho elemento. Un esquema XML siempre se almacena de forma separada del documento XML. El propio esquema XML es un tipo especial de documento XML, concretamente, es un documento que está escrito de acuerdo con las reglas dadas por la especificación del XML Schema del W3C. Estas reglas constituyen el lenguaje conocido como Lenguaje de Definición de XML Schema, que es una aplicación específica de XML.

El metamodelo de XML Schema, se muestra en la Figura 16 representado mediante un diagrama de clases UML.

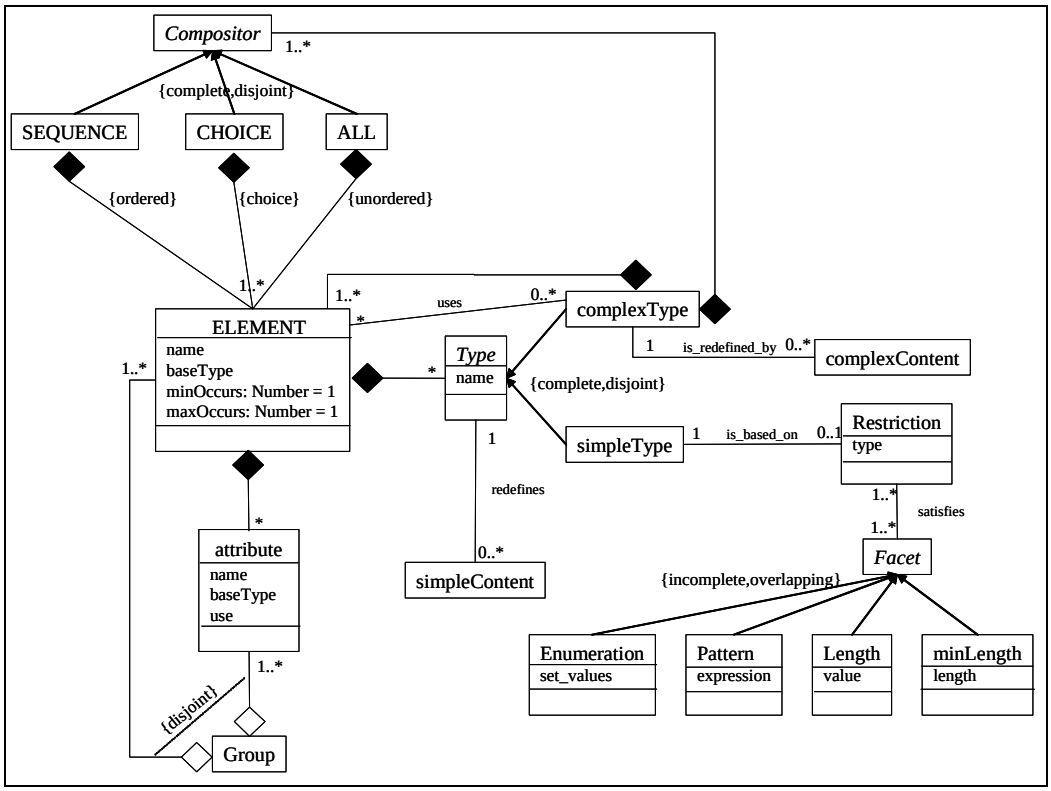


Figura 16. Metamodelo de XML Schema representado en UML

Cada esquema XML tiene un número de versión y un *namespace*; está compuesto por elementos, **ELEMENT,** que pueden tener atributos, **attributes**. Un elemento puede estar formado por otros tipos: **complexTypes**, que son tipos que pueden contener a su vez otros elementos, y **simpleTypes**, que son tipos que no pueden tener subelementos ni atributos.

Un esquema XML permite representar la cardinalidad de un elemento mediante los atributos **minOccurs** (número mínimo de ocurrencias) y **maxOccurs** (número máximo de ocurrencias). Para representar que un elemento es opcional, se pondrá el atributo **minOccurs** a 0. Para indicar que no hay un límite en el número máximo de ocurrencias), se pondrá el atributo **maxOccurs** a un valor indefinido (*unbounded)*. Si no se especifica el valor de estos atributos, cada atributo tomará el valor por defecto 1.

XML Schema proporciona un mecanismo para crear elementos y atributos definiendo nuevos tipos de datos. Se pueden definir **simpleTypes** para redefinir un tipo de datos existente y asignarselo a un elemento, a atributos o a tipos complejos para elementos. Los nuevos tipos tendrán un nombre y su definición estará situada

fuera de la definición de los elementos y atributos. Un tipo **simpleType** se puede definir como una restricción de otro tipo de datos.

XML Schema define 15 facetas, o características, incluyendo *length*, *minLength*, *minInclusive* y *maxInclusive*. Existen otras dos facetas que son particularmente útiles: *pattern* y *enumeration*. *Pattern* define una expresión regular a la que tiene que ajustarse el valor del tipo simple. La faceta *enumeration* limita un tipo simple a tomar un conjunto de valores predefinidos.

XML Schema proporciona mecanismos para redefinir un elemento existente, **complexContent**, que se usa para restringir o extender un tipo **complexType** existente. Un **simpleContent** se puede utilizar para indicar que un tipo nuevo sólo contiene datos de un determinado tipo existente, por ejemplo, de tipo carácter, pero sin contener ningún elemento

XML Schema permite la definición de grupos de elementos y atributos: **groups**. Un grupo no es un tipo de datos; actúa como un contenedor que contiene un conjunto de elementos o de atributos.

Existen tres tipos de compositores o contenedor: **sequence**, **choice** y **all**. **Sequence** es un compositor que define una secuencia ordenada de subelementos. El compositor **choice** define una elección entre varios posibles elementos o grupos de elementos. El compositor **all** define un conjunto no ordenado de elementos.

3.3.2.2 Extensión de UML para XML Schemas

A continuación, se propone la extensión de UML para representar XML Schemas (esquemas XML) en notación gráfica. Esta extensión de UML define un conjunto de estereotipos, valores etiquetados y restricciones que nos permite representar un XML Schema en notación gráfica de UML. Además se mostrará la representación gráfica de sencillos ejemplos de XML Schemas.

La extensión está definida para los componentes específicos de XML Schema, el actual estándar del W3C. Cada componente de un XML Schema se debe poder representar en notación gráfica con esta extensión de UML, conservando el orden y grado de anidamiento dado.

A la hora de elegir los estereotipos se ha seguido el siguiente criterio:

- Los elementos ELEMENT se han considerado clases estereotipadas porque están explícitamente definidos en el XML Schema.

- Los atributos de un ELEMENT se han considerado atributos estereotipados de las clases que representan al ELEMENT.
- Los tipos complexType se han considerado clases estereotipadas si tienen nombre. En este caso, el tipo complexType estará relacionado con el tipo ELEMENT o el tipo que lo use mediante una asociación <<uses>>. Si no tiene nombre, se representará de forma implícita mediante la composición que forma este tipo.
- Los tipos simpleType se han considerado clases estereotipadas con el mismo nombre que el elemento que los contiene. El tipo simpleType estará relacionado con su padre (ELEMENT) mediante una composición estereotipada con <<simpleType>>.
- Los tipos complexContent se han considerado como clases estereotipadas que deben estar relacionadas mediante una relación de herencia con el tipo padre (un complexType) que se está redefiniendo mediante el tipo complexContent.
- Los tipos simpleContent se han considerado clases estereotipadas que están relacionadas mediante una relación de herencia con el tipo padre (simpleType o complexType) que redefine el tipo simpleContent.
- Los compositors se consideran composiciones (tipo especial de asociación) estereotipadas. Sus estereotipos dependerán del tipo de compositor: <<Choice>>, <<Sequence>> o <<All>>.
- Para cada elemento, tipo o atributo se debe especificar, al lado del nombre de la clase, tipo o atributo el número de orden, incluyendo como un prefijo el número de orden del elemento o tipo padre.

Tipo ELEMENT

Clase del metamodelo: Clase

Descripción: Un tipo <<ELEMENT>> representa un elemento del XML Schema

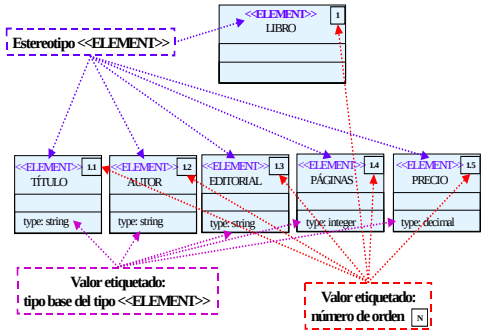
Icono: 

Restricciones: Sólo se puede usar para definir elementos de un XML Schema <<ELEMENT>>.

Valores Etiquetados: El nombre del elemento, el tipo base, el número mínimo y máximo de ocurrencias. Número de orden con un prefijo que será el número de orden del elemento al que pertenece.

Ejemplo:

```
<?xml version="1.0"?>
<!-- File Name: Esquema_Libro.xsd -->
<xsd: schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="LIBRO">
    <xsd:complexType>
      <xsd:sequence>
            <xsd:element name="TITULO" type="xsd:string"/>
            <xsd:element name="AUTOR" type="xsd:string"/>
            <xsd:element name="EDITORIAL" type="xsd:string"/>
            <xsd:element name="PAGINAS" type="xsd:integer"/>
            <xsd:element name="PRECIO" type="xsd:decimal"/>
      </xsd:sequence>
      <xsd: attribute name="Disponible" type="xsd:boolean" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



Atributo

Clase del metamodelo: Atributo

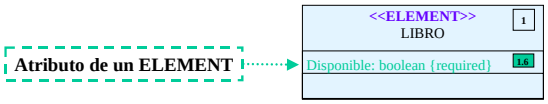
Descripción: Un atributo pertenece a un elemento <<ELEMENT>> del esquema

Icono: Ninguno

Restricciones: Un atributo sólo puede pertenecer a un único elemento <<ELEMENT>>.

Valores Etiquetados: El nombre del atributo, el tipo base, la restricción que debe satisfacer el atributo (required, optional) y el valor por defecto o fijado previamente. Otro valor etiquetado será su número de orden incluyendo como prefijo el número de orden del elemento al que pertenece el atributo.

Ejemplo:



Tipo ComplexType

Clase del metamodelo: Clase

Descripción: Un tipo <<complexType>> está compuesto por otros elementos u otro compositor.

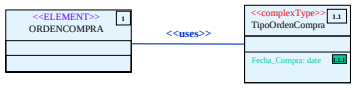
Icono: 

Restricciones: Debe estar relacionado por medio de una asociación <<uses>> con los elementos o tipos que usan el tipo <<complexType>>. Sólo se definirá como una clase si tiene nombre, sino se definirá de forma implícita.

Valores Etiquetados: Nombre

Ejemplo:

```
<?xml version="1.0"?>
<!-- File Name: OrdenCompra.xsd -->
<xsd: schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="OrdenCompra" type="TipoOrdenCompra">
    <xsd:complexType name= "TipoOrdenCompra">
      <xsd:attribute name="Fecha_Orden" type="xs:date" />
    </xsd:complexType>
</xsd:schema>
```



Tipo SimpleType

Clase del metamodelo: Clase

Descripción: Un tipo <<simpleType>> no tiene ni subelementos ni atributos.

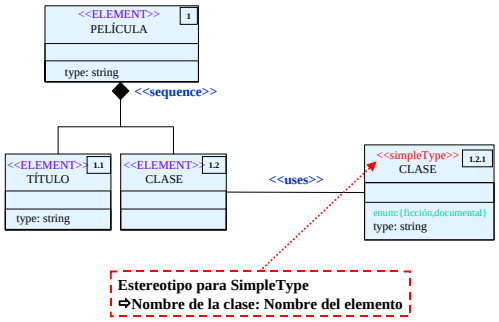
Icono: 

Restricciones: Debe estar asociado mediante una asociación <<uses>> con el elemento o atributo que lo usa.

Valores Etiquetados: Tipo base, restricciones del propio tipo base.

Ejemplo:

```
<?xml version="1.0"?>
<!-- File Name: Esquema_Película.xsd -->
<xsd: schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="PELÍCULA">
  <xsd:complexType>
   <xsd:sequence>
        <xsd:element name="TÍTULO" type="xsd:string"/>
        <xsd:element name="CLASE"
          <xsd:simpleType>
          <xsd:restriction base="xsd:string">
              <xsd:enumeration value="ficción">
              <xsd:enumeration value="documental">
          </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
    <xsd:choice>
      <xsd:element name="ACTOR" type="xsd:string"/>
      <xsd:element name="DIRECTOR" type="xsd:string"/>
    </xsd:choice>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```



Tipo ComplexContent

Clase del metamodelo: Clase

Descripción: Un <<complexContent>> es una subclase del tipo complexType que lo define.

Icono: Ninguno

Restricciones: Debe estar relacionado mediante una relación de herencia con los elementos o tipos <<complexType>> que redefine el tipo <<complexContent>>.

Valores Etiquetados: Nombre

Tipo SimpleContent

Clase del metamodelo: Clase

Descripción: Un <<simpleContent>> es una subclase del tipo <<complexType>> o <<simpleType>>.

Icono: Ninguno

Restricciones: Debe estar relacionado mediante una relación de herencia con el tipo que redefine el tipo <<simpleContent>>.

Valores Etiquetados: Nombre

Asociación Compositor

Clase del metamodelo: Asociación

Descripción: Una asociación compositor es un tipo especial de composición estereotipada con el tipo de compositor <<Choice>>, <<Sequence>> o <<All>>, que indica los elementos que componen al superelemento (padre).

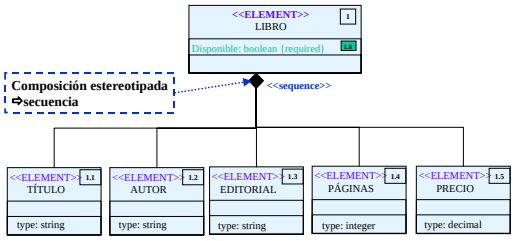
Icono: Ninguno

Restricciones: Sólo se puede usar para unir un <<ELEMENT>> con los <<ELEMENT>>que lo componen.

Valores Etiquetados: Ninguno

Ejemplo:

```
<?xml version="1.0"?>
<!-- File Name: Esquema_Libro.xsd -->
<xsd: schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="LIBRO">
  <xsd:complexType>
   <xsd:sequence>
        <xsd:element name="TITULO" type="xsd:string"/>
        <xsd:element name="AUTOR" type="xsd:string"/>
        <xsd:element name="EDITORIAL" type="xsd:string"/>
        <xsd:element name="PAGINAS" type="xsd:integer"/>
        <xsd:element name="PRECIO" type="xsd:decimal"/>
   </xsd:sequence>
   <xsd: attribute name="Disponible" type="xsd:boolean" use="required"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```



Asociación Uses

Clase del metamodelo: Asociación

Descripción: Una asociación <<uses>> es un tipo especial de asociación unidireccional que unirá un tipo <<complexType>> con nombre con el elemento <<ELEMENT>> o tipo que lo usa.

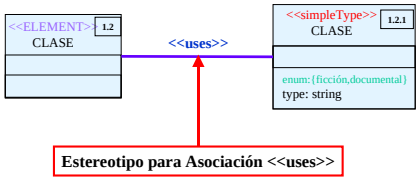
Icono: Ninguno

Restricciones: Sólo se puede usar para unir <<ELEMENT>> o tipos con un complexType con nombre.

Valores Etiquetados: Ninguno

Ejemplo:

```
<?xml version="1.0"?>
<!-- File Name: Esquema_Película.xsd -->
<xsd: schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="PELÍCULA">
  <xsd: complexType>
   <xsd:sequence>
        <xsd:element name="TÍTULO" type="xsd:string"/>
        <xsd:element name="CLASE">
          <xsd:simpleType>
          <xsd: restriction base="xsd:string">
                <xsd:enumeration value="ficción">
                <xsd:enumeration value="documental">
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:choice>
          <xsd:element name="ACTOR" type="xsd:string"/>
          <xsd:element name="DIRECTOR" type="xsd:string"/>
        </xsd:choice>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```



Reglas que ha de cumplir un diseño bien formado:

- Un tipo <<ELEMENT>> puede contener un conjunto de atributos.
- Cada tipo <<complexType>> que tenga nombre tiene que estar relacionado con el tipo <<ELEMENT>>, el tipo <<complexType>> o el tipo <<simpleType>> que lo usa mediante una asociación <<uses>>.

3.3.2.3 Metamodelo de XLink

El XML Linking Language (XLink) permite insertar elementos en documentos XML para crear y describir enlaces entre recursos. XLink proporciona un marco de trabajo para crear tanto enlaces unidireccionales, como estructuras de enlace más complejas. A continuación, se describirán brevemente los principales términos y conceptos de XLink. Para más detalles se puede consultar el estándar en W3C (2001a).

Un enlace XLink es una relación explícita entre recursos o partes de recursos. Un recurso es una unidad de información o servicio direccionable, es decir, que se puede referenciar. Un enlace se hace explícito por un elemento de enlace XLink.

Existen seis tipos de elementos XLink:

- Dos de ellos se consideran elementos de enlace: enlaces simples (simple links) y enlaces extendidos (extended links).
- Los demás elementos describen las características de un enlace mediante información adicional.

Recorrer un enlace (**"Traversal"**, como se especifica en el estándar) consiste en utilizar o seguir un enlace e implica una pareja de recursos: el origen, del que parte el recorrido y el destino, o recurso final. La información sobre cómo recorrer una pareja de recursos, incluyendo la dirección del recorrido e información sobre el comportamiento de la aplicación, se especifica mediante un arco (denominado **arc**). Si dos arcos de un enlace especifican el mismo par de recursos, pero intercambiando la posición del origen y el destino, entonces el enlace será multidireccional, que no es lo mismo que simplemente "volver atrás" tras recorrer un enlace.

Un recurso local es un elemento XML que participa en un enlace, siendo él o su padre un elemento de enlace. Cualquier recurso o parte de un recurso que participe en un enlace por ser referenciado por una URI se considera un recurso

remoto, incluso si es dentro del mismo documento XML. Un recurso local se especifica “por valor” y un recurso remoto “por referencia”.

Como se ha dicho previamente, XLink proporciona dos tipos de enlaces:

- Un enlace simple es un enlace que asocia exactamente dos recursos, uno local y otro remoto, con un arco que va del primero (local) al segundo (remoto).
- Un enlace extendido es un enlace que asocia un número arbitrario de recursos. Los recursos que participan pueden ser locales y remotos.

La Figura 17 muestra el metamodelo de XLink simplificado (sin atributos) representado con un diagrama de clases de UML.

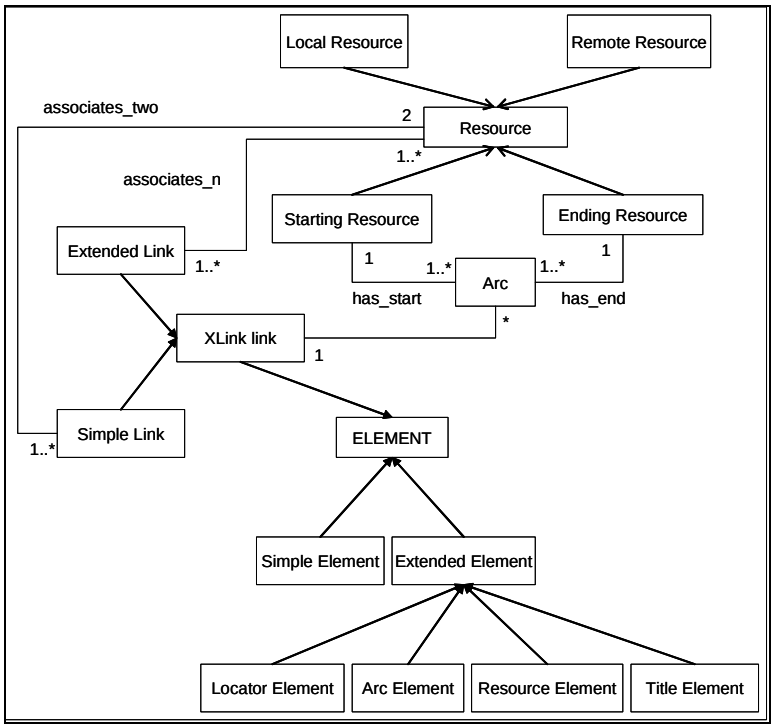


Figura 17. Metamodelo de XLink

3.3.2.4 Extensión de UML para XLink

Esta extensión de UML define un conjunto de estereotipos, valores etiquetados y restricciones que nos permite representar un documento en XML Linking Language (XLink) en notación gráfica mediante UML. La extensión está definida para los componentes específicos de XLink, de modo que cada componente de un XLink se debe poder representar mediante la esta extensión.

A la hora de elegir los estereotipos se ha seguido el siguiente criterio:

- Los elementos (ELEMENTs) de XLink, tanto los simples como los extendidos, se han considerado clases estereotipadas, porque son objetos de primera categoría en el metamodelo de XLink.
- Los atributos de los enlaces XLink se han considerado atributos estereotipados de las clases que representan a los elementos XLink. Los atributos que puede tener cada clase de un elemento XLink dependerán del tipo del elemento XLink.
- El valor proporcionado en los atributos 'from' o 'to' de una asociación del tipo "arc" se tiene que corresponder con el valor del atributo 'label' de algún elemento XLink de tipo 'locator' o 'resource'.

Siguiendo estos criterios, se ha definido la siguiente extension UML:

Elemento Simple XLink

Clase del metamodelo: Clase

Descripción: Un elemento <<Simple XLink>> representa un enlace simple. Este elemento puede tener un conjunto de atributos que se especificarán como atributos de la clase que representa el enlace simple.

Icono: Ninguno

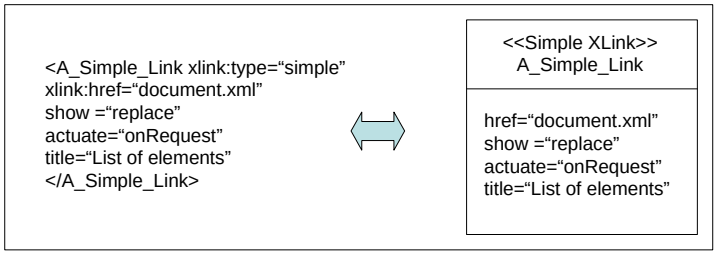
Restricciones: Sólo se puede usar para definir enlaces simples, es decir, para especificar un enlace entre un recurso local y uno remoto.

Valores Etiquetados: El nombre del elemento del XLink simple.

Atributos de la clase:

Opcionales: href, role, arcrole, title, show, actuate

Ejemplo:



Elemento Extended XLink

Clase del metamodelo: Clase

Descripción: Un elemento <<Extended XLink>> representa un tipo especial de elemento que puede contener un número no definido de recursos. Este elemento puede tener un conjunto de atributos que se especificarán como atributos de la clase que representa el enlace simple.

Icono: Ninguno

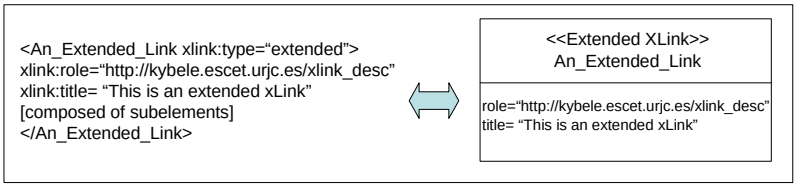
Restricciones: Sólo se puede usar para definir elementos del tipo enlaces extendidos (extended links), es decir, para especificar un enlace entre varios recursos.

Valores Etiquetados: El nombre del elemento Extended XLink.

Atributos de la clase:

Opcionales: role, title

Ejemplo:



Elemento XLink Remote Resource

Clase del metamodelo: Clase

Descripción: Un elemento << XLink Resource>> representa un elemento de recurso remoto.

Icono: Ninguno

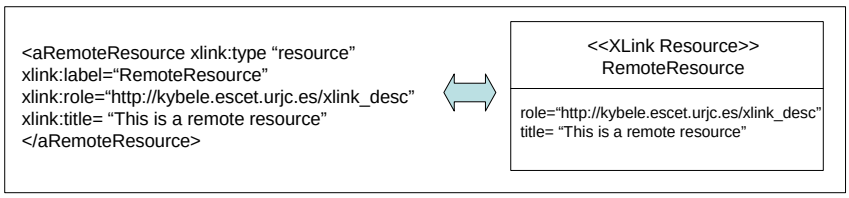
Restricciones: Sólo se puede usar para definir recursos remotos, es decir, un elemento con el valor "resource" en el atributo "type". El nombre de la clase que representa el recurso remoto será el valor del valor del atributo "label".

Valores Etiquetados: Ninguno

Atributos de la clase:

Opcionales: role, title, label

Ejemplo:



Elemento XLink Local Resource

Clase del metamodelo: Clase

Descripción: Un elemento << XLink Locator>> representa un elemento de recurso local.

Icono: Ninguno

Restricciones: Sólo se puede usar para definir recursos locales, es decir, un elemento con el valor "locator" en el atributo "type". El nombre de la clase que representa el recurso local será el valor del atributo "label".

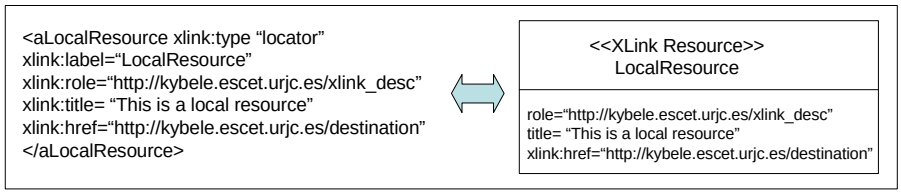
Valores Etiquetados: Ninguno

Atributos de la clase:

Requeridos: href

Opcionales: role, title, label

Ejemplo:



Composición Is_Composed

Clase del metamodelo: composición

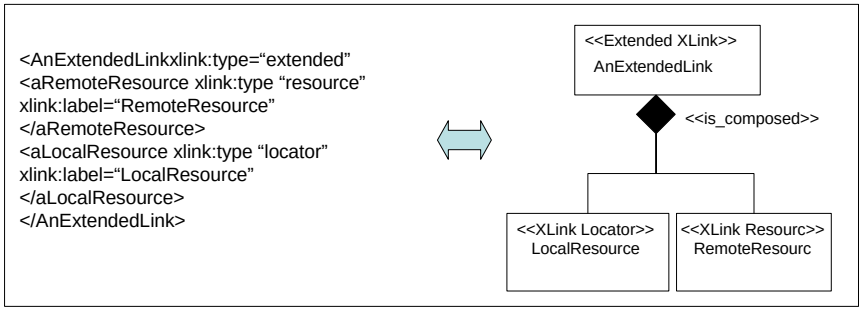
Descripción: Una composición <<is_composed>> es un tipo especial de asociación que enlaza un enlace XLink extendido con los recursos que componen el XLink extendido.

Icono: 

Restricciones: Sólo se puede usar para asociar el enlace extendido con los recursos que componen el enlace extendido.

Valores Etiquetados: Ninguno

Ejemplo:



Asociación Arc

Clase del metamodelo: Asociación

Descripción: Una asociación <<arc>> es un tipo especial de asociación unidireccional que enlaza dos recursos. Esta asociación se representa con un arco guiado que parte del recurso que representa el FROM (desde) y termina en el recurso que representa el TO (hacia). Ambos recursos pueden ser locales o remotos.

Icono: Ninguno

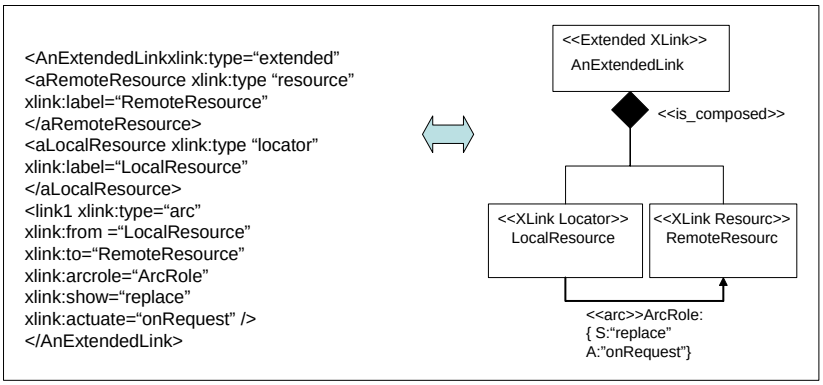
Restricciones: Sólo se puede usar para representar asociaciones entre dos recursos existentes y previamente definidos.

Valores Estiquetados: Los atributos show {S}, actuate {A}, arcrole y/o title (ambos directamente después del estereotipo <<arc>>).

Atributos de la clase:

Opcionales: arcrole, title, show, actuate, from, to.

Ejemplo:



3.3.2.5 Guías de Transformación

Como se ha visto, para realizar el diseño conceptual del hipertexto se utilizan dos técnicas: el modelo conceptual de fragmentos y el modelo conceptual de navegación. La transformación del diseño conceptual al lógico se lleva a cabo en dos pasos:

a) En primer lugar, se transforman cada uno de los fragmentos definidos en el modelo conceptual de fragmentos en un esquema XML.

b) Y, en segundo lugar, se transforma el modelo conceptual de fragmentos y el de navegación a un XLink, especificando así la navegación entre cada uno de los fragmentos y los elementos de acceso definidos.

3.3.2.5.1 Transformación de fragmentos a esquemas XML

Las guías para transformar los fragmentos definidos a nivel conceptual al nivel lógico están basadas en las reglas definidas en el estándar XMI (OMG, 2003). Sin embargo, hay que señalar que en MIDAS/DB caso se parte del modelo de fragmentos y *no* del modelo conceptual como se propone en el estándar anteriormente citado.

En MIDAS/DB se definieron inicialmente las reglas de transformación para guiar en el proceso de conversión de un modelo conceptual de datos a DTDs (ya que éste era el mecanismo de validación de documentos XML que en ese momento estaba estandarizado). Sin embargo, tras los sucesivos refinamientos realizados en la metodología, se consideró necesario partir del modelo de fragmentos y de navegación para pasar al diseño lógico del hipertexto definido en XML. Además, las reglas se adaptaron a esquemas XML y se incorporaron las propuestas de XMI. Las guías de transformación a DTDs pueden encontrarse en el apéndice D.

Para obtener el *modelo lógico de fragmentos* cada uno de los fragmentos del modelo conceptual de fragmentos se transformará en un esquema XML siguiendo las guías que se definen a continuación:

- Para cada fragmento se genera un esquema XML con un elemento raíz que incluye todos los elementos del fragmento. El elemento se deberá denominar con el nombre del fragmento del que se ha partido.
- Cada atributo del fragmento se transformará dentro del esquema XML correspondiente en un subelemento del elemento raíz previamente definido. Para cada elemento se puede especificar su tipo coincidiendo con el tipo que tuviese el atributo de la clase de modelo conceptual de la que se obtuvo el fragmento correspondiente.
- Dependiendo del tipo de atributo se realizará una transformación u otra, tal y como se indica en la Tabla 7:

Tabla 7. Transformación de Atributos según su Tipo

TIPO DEL ATRIBUTO		MÉTODO DE TRANSFORMACIÓN
SIMPLE	OBLIGATORIO	Como SubElemento del Elemento Raíz con el nombre del atributo
	OPCIONAL	Como SubElemento del Elemento Raíz con el nombre del atributo y minOccurs= '0'
MULTIVALUADO	OBLIGATORIO	Como SubElemento del Elemento Raíz de tipo ComplexType (choice, sequence, all)
	OPCIONAL	Como SubElemento del Elemento Raíz de tipo ComplexType (choice, sequence, all) y minOccurs='0'
COMPUESTO		Como SubElemento del Elemento Raíz de tipo ComplexType incluyendo los elementos del atributo compuesto
ENUMERADO		Como SubElemento del Elemento Raíz de tipo SimpleType (restricción del tipo enumeration, especificando los posibles valores)
ELECCIÓN		El subelemento será de tipo choice y sólo uno de los subelementos podrá aparecer
ÚNICO		El subelemento será del tipo unique y por lo tanto, su valor será único.
CLAVE		Se especifica

3.3.2.5.2 Transformación del modelo conceptual de navegación a XLink

Una vez obtenidos los esquemas XML para cada uno de los fragmentos, se pasa a definir el *modelo lógico de navegación* en XLink siguiendo las siguientes guías:

- El modelo conceptual de navegación completo se transforma, a nivel lógico, en un enlace extendido <<Extended Link>> que relaciona un conjunto de recursos. El enlace extendido se representa en XLink mediante una clase abstracta denominada 'Modelo Lógico de Navegación'.
- Esta clase abstracta se relaciona con todos los fragmentos que componen el modelo lógico de navegación mediante una composición estereotipada <<is_composed>>.
- Cada uno de los fragmentos del modelo conceptual de fragmentos se transforma, a nivel lógico, en un recurso local representado con un elemento <<XLink Locator>>, denominado con el nombre del fragmento correspondiente.
- Los elementos de acceso del modelo lógico de navegación: índices, visitas guiadas, consultas y menús, también se transforman a un recurso representado con un <<XLink Locator>>.
- La navegación entre dos fragmentos, o entre un fragmento y un elemento de acceso, se representa mediante una asociación <<arc>>.
- La información de cómo se realiza la transición através de dicha asocición, se incluye en el arc especificando los valores etiquetados correspondientes.

Validación

En el presente capítulo se muestra el proceso seguido para la validación del trabajo realizado en esta tesis doctoral. Para ello se describen brevemente los casos de estudio definidos con el fin de encontrar las necesidades en el desarrollo de SIW. Las lecciones aprendidas tras desarrollar cada uno de los casos han ayudado a ir definiendo y refinando la metodología en un proceso cíclico. También se presentan los casos de prueba aplicados a fin de validar tanto el proceso de la metodología propuesta, como las técnicas definidas en la misma.

Para ello, en el apartado 4.1 se muestra el proceso realizado para la validación resumiendo los casos de estudio y los casos de prueba utilizados, así como las lecciones aprendidas de cada uno de ellos. En el apartado 4.2 se presenta un caso de estudio completo que permite además ofrecer una visión global de MIDAS/DB. Los demás casos pueden encontrarse en el apéndice C. Para finalizar, el apartado 4.3 muestra la implementación de tres módulos de Rational Rose para el modelado de esquemas OR, de esquemas XML y XLink utilizando las extensiones de UML propuestas.

4.1 Proceso de Validación

En el capítulo 1 se presentó el método de investigación que se ha seguido para la realización de esta tesis doctoral. Se trata de un proceso cíclico, basado en el proceso del método Investigación en Acción, y cuya validación se realiza mediante la aplicación de los resultados obtenidos a casos reales. Estos casos han sido llevados a cabo en el laboratorio del grupo Kybele y se clasifican en dos tipos: a) casos de estudio y b) casos de prueba. Como ya se dijo en el capítulo 1, los casos de estudio y casos de prueba utilizados han sido los siguientes:

a) CASOS DE ESTUDIO: que mediante el desarrollo de casos reales, utilizando, cuando ha sido posible, prácticas de metodologías existentes, han permitido especificar el marco metodológico, proceso y técnicas requeridas.

- CASO 1: Reserva de ordenadores y aulas informáticas
- CASO 4: Web del grupo de investigación Kybele
- CASO 6: Casas Rurales

b) CASOS DE PRUEBA: que han permitido realizar la validación de las distintas propuestas de la metodología.

- La extensión OR:
  - CASO 1: Reserva de ordenadores y aulas informáticas
  - CASO 2: Proyectos Arquitectónicos
  - CASO 3: Libros de Recetas de Cocina
  - CASO 4: Web del grupo de investigación Kybele
- La extensión XML:
  - CASO 5: Ejemplos Internet
  - CASO 6: Casas Rurales
  - CASO 7: Cine-Entradas
- La metodología MIDAS/DB:
  - CASO 6: Casas Rurales
  - CASO 7: Cine-Entradas

La Figura 18 muestra la secuenciación en el tiempo de los casos de estudio y de prueba; al tratarse de un proceso cíclico, algunos de los casos de estudio han sido también casos de prueba para algunas técnicas de la metodología. La Figura 18 muestra además las iteraciones realizadas en la definición de MIDAS/DB, las necesidades detectadas en cada caso, así los resultados de las lecciones aprendidas.

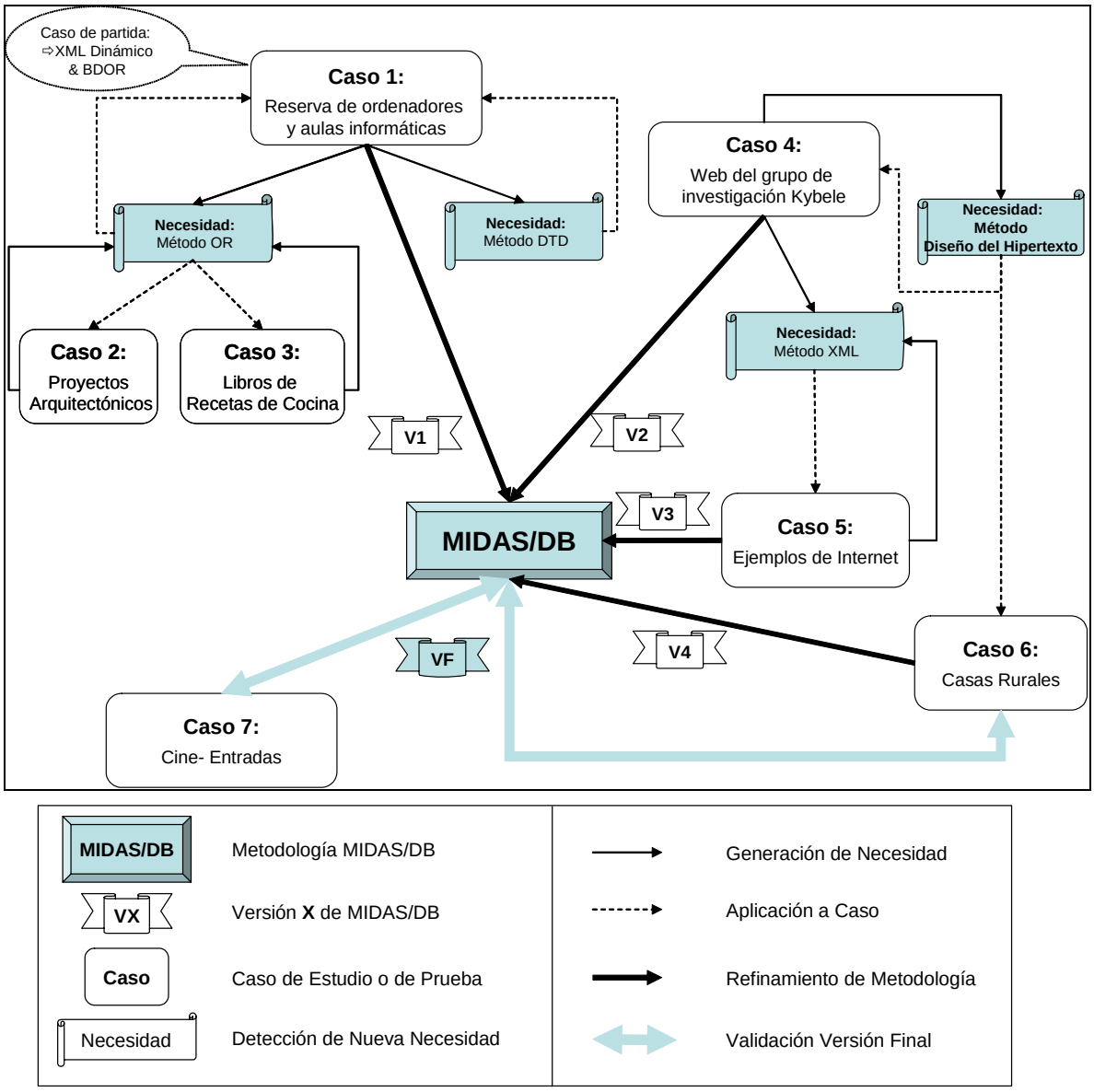


Figura 18. Proceso de validación de MIDAS/DB

4.1.1 Descripción de los Casos de Estudio y de Prueba

Al comienzo de este trabajo se realiza el estudio de las principales metodologías existentes en ese momento y se identifican las carencias de las mismas respecto a las necesidades de los SIW. A la vez, se comienzan a realizar diferentes casos de estudio para ver en qué medida las propuestas metodológicas cubrían todas las necesidades en el desarrollo de SIW. Algunos de los casos de prueba han servido para validar técnicas concretas y otros se han empleado para validar la metodología mediante un desarrollo completo. Los casos de estudio se han aplicado también a casos completos y algunos de ellos se han implementado, como, por ejemplo, el caso de la Web del grupo de investigación Kybele. De los

casos que están operativos en la Web se dará la dirección donde pueden encontrarse los mismos.

CASO1: Reserva de ordenadores y aulas informáticas

⇨ ***http://kybele.escet.urjc.es/ejemplos/reservas_aulas_PCs/***

Este caso consiste en la definición de un SIW que permite consultar y hacer reservas de ordenadores y aulas informáticas en la URJC. Este SIW tiene dos objetivos: por un lado, la información sobre las especificaciones de los ordenadores y de las aulas, así como el estado de las reservas debe almacenarse en una BDOR; por otro lado, las páginas deben ser dinámicas y definidas en XML, con el fin de permitir el acceso a las mismas desde múltiples plataformas; en concreto, se pretendía que los alumnos pudieran realizar sus reservas de ordenadores desde un teléfono móvil (en aquel momento, utilizando tecnología WAP).

Durante el desarrollo de este caso de estudio, se ve la necesidad de definir unas guías que faciliten la obtención del diseño lógico del hipertexto (utilizando DTDs) a partir del modelo conceptual; se especifica una primera versión de estas guías que puede verse resumida en el apéndice D. Una vez obtenido un DTD genérico, a partir del modelo conceptual de datos, se ve la necesidad de particionar el mismo de acuerdo al modelo de presentación previamente definido, siendo necesario un DTD parcial para cada una de las páginas del SIW.

Además, este caso de estudio sirve para detectar la conveniencia de un método para el desarrollo de BDOR. El método especificado se aplica a este mismo caso sirviendo así también como caso de prueba. Este método incluye una extensión de UML para representar esquemas de BDOR en notación gráfica, así como guías de transformación para la transición entre modelos. El diseño de la BDOR para este caso de estudio se puede ver en el apéndice C.

Al finalizar este caso, se dispone de una primera versión de MIDAS/DB que incluye: a) un método para el diseño de BDOR; b) una primera aproximación al diseño del hipertexto con tecnología XML, basada en DTDs.

Para validar el método de diseño de BDOR, éste se aplica además a otros dos *casos de prueba* adicionales: ***CASO2: Proyectos Arquitectónicos*** y **CASO3: *Libros de Recetas de Cocina***. Estos dos casos han permitido comprobar la validez de la notación y de las guías de transformación, así como realizar un refinamiento posterior del método para el diseño de BDOR. Estos dos casos se resumen también en el apéndice C.

CASO4: Web del grupo de investigación Kybele

⇨ ***http://kybele.escet.urjc.es/***

Casi de forma paralela al desarrollo del primer caso de estudio, se comenzó a trabajar en otro caso de estudio: la Web del grupo de investigación Kybele. MIDAS/DB propone, como ya se ha dicho, realizar en primera instancia un prototipo inicial con páginas estáticas. Posteriormente, y tras realizar un refinamiento con el cliente, que en este caso era el grupo de investigación Kybele, en otra iteración se desarrolla la BDOR y las páginas dinámicas en XML.

Tal y como se ha explicado anteriormente, el primer caso había permitido detectar la necesidad de especificar guías para la transformación del modelo conceptual de datos a un modelo lógico de hipertexto definido mediante DTDs, así como la necesidad de particionar estos DTDs de acuerdo al modelo conceptual de presentación. Así, se identifica la necesidad de establecer dos técnicas para el modelado conceptual del hipertexto: el modelo conceptual de fragmentos y el modelo conceptual de navegación. De igual modo se establece la relación existente entre el modelo de presentación y el de hipertexto con la tecnología XML: las páginas de estilo corresponden al modelo de presentación y los DTDs a los fragmentos.

En este momento se decide también pasar a utilizar esquemas XML (*XML Schemas*) en lugar de DTDs. Al comienzo de este trabajo la forma de comprobar que un documento XML es "válido" es mediante DTDs. Posteriormente, se estandarizan los esquemas XML y tras estudiar sus ventajas, se opta por sustituir los DTDs por los esquemas XML para realizar el diseño lógico del hipertexto y se decide adaptar la metodología a esta necesidad. Como MIDAS/DB propone realizar todo el sistema en una única notación UML, se decide realizar una extensión de UML con el fin de poder representar también el diseño lógico del hipertexto en este lenguaje.

Para validar la extensión de UML para el diseño de esquemas XML se eligen varios esquemas XML de Internet, que se utilizan como casos de prueba: ***CASO 5: Ejemplos de Internet***. Estos casos se representan con la extensión de UML con el fin de validar que entre la representación del esquema en XML y en UML existe una correspondencia unívoca. Por su extensión, estos casos se incluyen en el CD que se adjunta a esta tesis.

CASO 6: Casas Rurales

⇨ ***http://kybele.escet.urjc.es/ejemplos/casas_rurales/***

Se trata de un SIW para la consulta y reserva de casas rurales en la Comunidad de Madrid. Este caso permite especificar las técnicas que se utilizaran para el diseño del hipertexto de un SIW. Para ello se toman las técnicas de dos de las metodologías estudiadas (las que se han estimado más completas: RMM y UWE) y se aplican a este caso. Se realiza una comparativa entre ambas y se decide incorporar a MIDAS/DB como técnica para el diseño del hipertexto, el modelo de fragmentos y el modelo de navegación de la metodología RMM y adaptar la notación propuesta en la metodología UWE (ya que UWE se basa en UML) a los modelos de RMM.

Es aquí cuando surge la necesidad de utilizar XLink para representar el modelo lógico de navegación. Por ello, se especifica una nueva extensión para poder representar esquemas XLink en UML. También se definen unas guías de transformación del modelo conceptual de navegación al modelo lógico de navegación en XLink.

Finalmente, este caso se desarrolla de forma completa siguiendo el proceso propuesto en MIDAS/DB y las técnicas de navegación elegidas. Por este motivo, además sirve de caso de prueba de las nuevas técnicas definidas para incorporar XLink a MIDAS/DB.

En el apéndice C y en CD adjunto se puede ver una descripción más detallada de este caso de estudio.

Para terminar, se ha aplicado la metodología completa a un caso de prueba, **CASO 7: Cine-Entradas**, que se verá en el siguiente apartado. Este caso ha servido, por una parte, para refinar las distintas técnicas de la metodología y por otra, para validar el proceso completo de MIDAS/DB.

4.2 Aplicación de MIDAS/DB a un Caso de Prueba

El caso de prueba CINE-ENTRADAS consiste el desarrollo de un SIW para un *Consorcio de las Cadenas de Cine de la Comunidad de Madrid*. El sistema ofrece servicios tanto de consulta de información como de venta y reserva de entradas. Las páginas deben estar codificadas en XML y deben extraer la información de una BDOR que no existe previamente. La especificación de requisitos completa para este caso de estudio se encuentra en el CD que se adjunta.

La página Web donde se encuentra la aplicación completa es: ***http://kybele.escet.urjc.es/ejemplos/cine_entradas/***.

A continuación, se verán los resultados de aplicar la metodología para este caso de prueba, centrándonos en la iteración MIDAS/DB que es en la que se desarrolla la dimensión estructural del SIW.

4.2.1 Iteración: MIDAS/DB

Esta iteración recibe como entrada: a) los diagramas conceptuales de datos, de presentación y de hipertexto; b) el primer prototipo de las páginas en HTML desarrollado en la iteración MIDAS/HT.

4.2.1.1 Actividad de Análisis

En esta actividad se refinan los diagramas obtenidos en el modelado conceptual de la iteración MIDAS/HT. Tiene tres tareas: modelado conceptual de datos, modelado conceptual de hipertexto y modelado conceptual de presentación.

TAREA 1: Modelado Conceptual de Datos

En la Figura 19 se muestra el diagrama de clases en UML refinado a partir del diagrama de clases y el del prototipo realizados en la iteración previa.

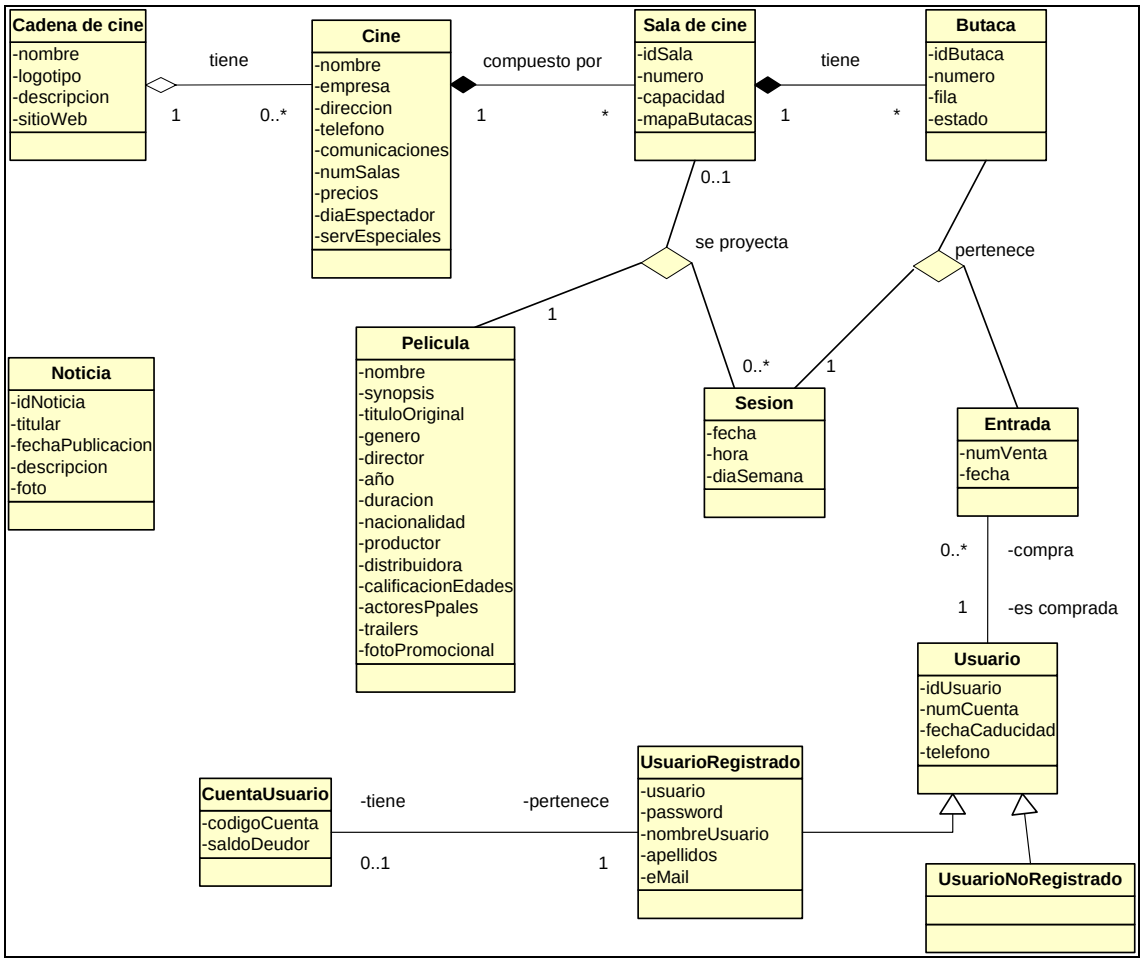


Figura 19. Modelo Conceptual de Datos en UML

TAREA 2: Modelado Conceptual de Hipertexto

Esta tarea lleva consigo la realización de dos nuevos diagramas, el de fragmentos y el de navegación.

A partir del diagrama conceptual de datos se obtiene el ***diagrama conceptual de fragmentos*** que se muestra en la Figura 20. Cada fragmento se representa con una clase *navegacional* denominada con el nombre de la clase del diagrama conceptual de datos que contiene la mayor parte de la información. Es importante señalar que, si bien el diagrama conceptual de datos debe recoger toda la información del sistema, el diagrama de fragmentos no, ya que éste representa tan sólo la información que se va a mostrar en la Web. La información presentada en cada uno de los fragmentos corresponde a la información que debería mostrarse en las páginas y puede pertenecer a una o más clases del diagrama conceptual de datos. Hay que tener en cuenta que pueden existir algunas clases en el diagrama conceptual de datos que no sea necesario incluir en el diagrama de fragmentos.

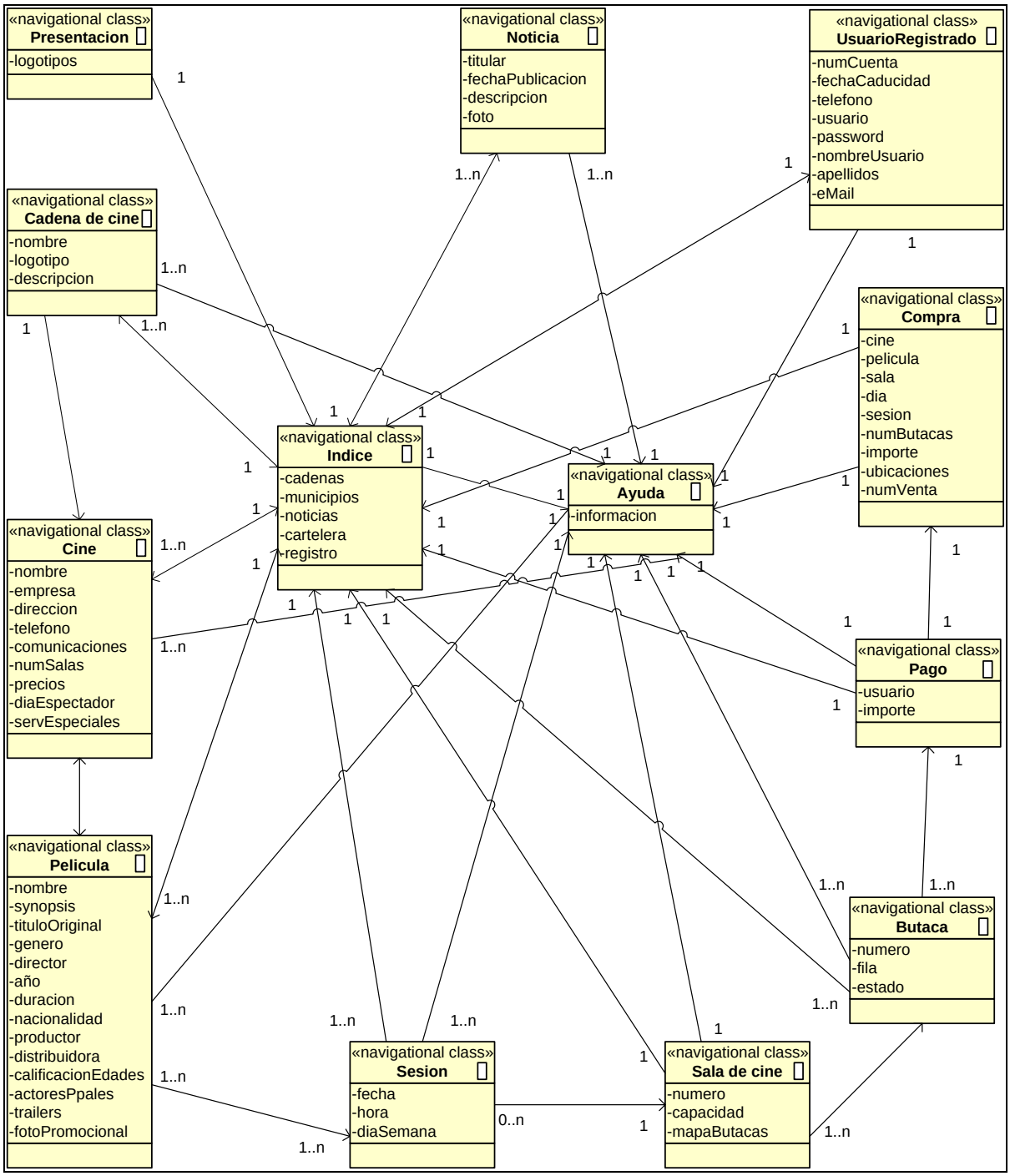


Figura 20. Diagrama Conceptual de Fragmentos

Partiendo del diagrama conceptual de fragmentos y añadiéndole los elementos de acceso requeridos, se construye el ***diagrama conceptual de navegación*** que se puede ver en la Figura 21.

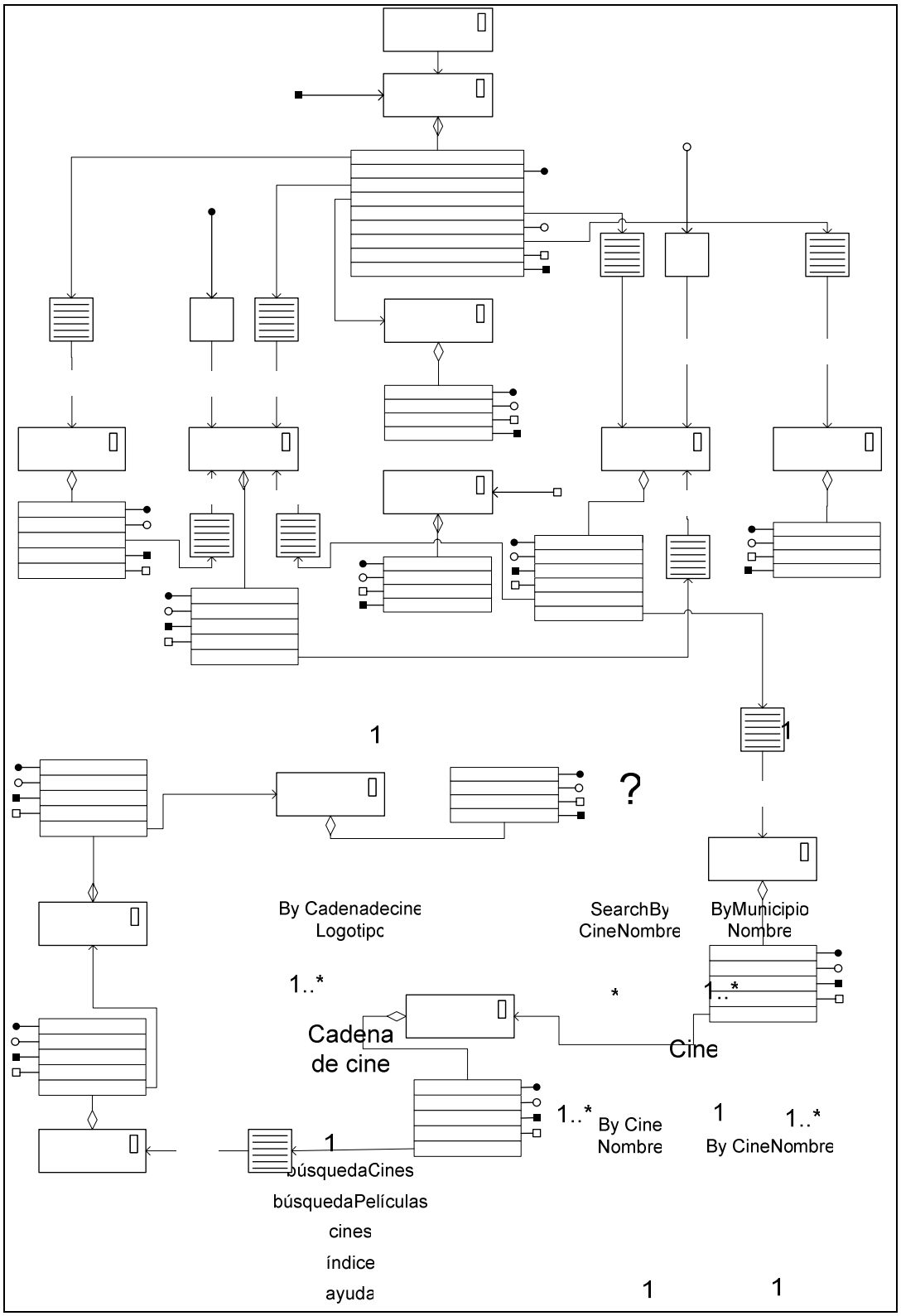


Figura 21. Diagrama Conceptual de Navegación

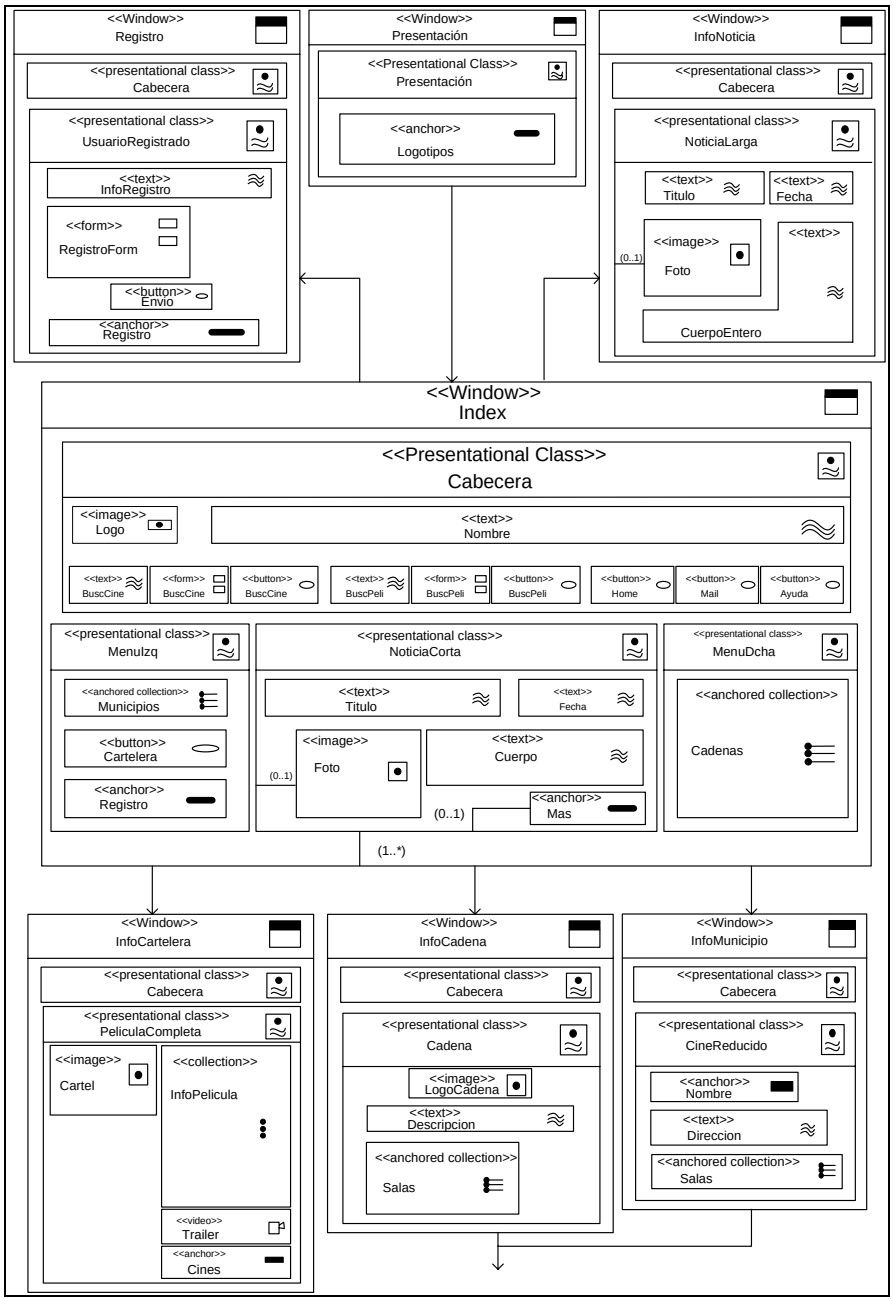
PagoMenú

búsquedaCines

TAREA 3: Modelado Conceptual de Presentación

Esta tarea consiste en representar, de un modo abstracto e independiente de la implementación, el diseño de las pantallas: qué información se mostrará en cada pantalla y cómo (mediante una lista desplegable, una imagen, un cuadro de texto, etc.). Por cada fragmento hay que diseñar un modelo de página. Las estructuras de navegación pueden, o no, dar lugar a nuevas páginas; esto es una decisión de diseño. Este diagrama muestra también la relación entre las distintas páginas que se obtiene de los enlaces del diagrama de navegación.

La Figura 22 muestra el diagrama conceptual de presentación siguiendo la notación propuesta en UWE.



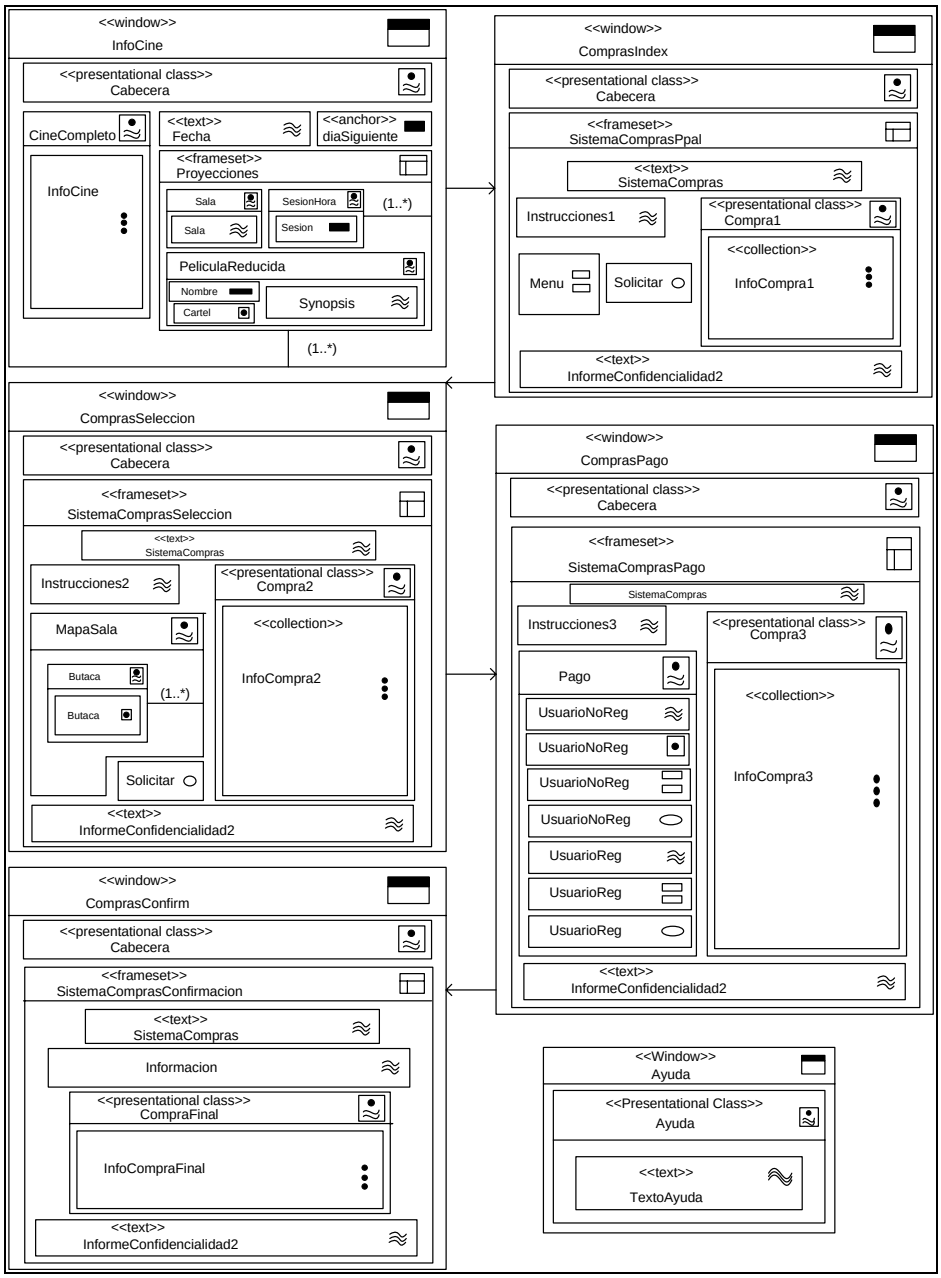


Figura 22. Diagrama Conceptual de Presentación

4.2.1.2 Actividades de Diseño e Implementación

A partir de los diagramas obtenidos como resultado de la actividad de análisis se pasará a realizar el diseño y la implementación. Aunque en realidad, el diseño y la implementación son dos actividades distintas, aquí se presentan de un modo conjunto por facilitar el seguimiento de la conversión entre modelos. Esta actividad tiene dos tareas: a) diseño e implementación de la BD y b) diseño e implementación del hipertexto.

TAREA 1: Diseño e Implementación de la BD

Esta tarea incluye el diseño lógico estándar y el diseño lógico específico.

El ***diseño lógico estándar*** consiste en un diseño independiente del producto que se vaya a utilizar para la implementación, por lo que utilizaremos el modelo de SQL: 1999. El diagrama UML del esquema SQL: 1999 se muestra en la Figura 23.

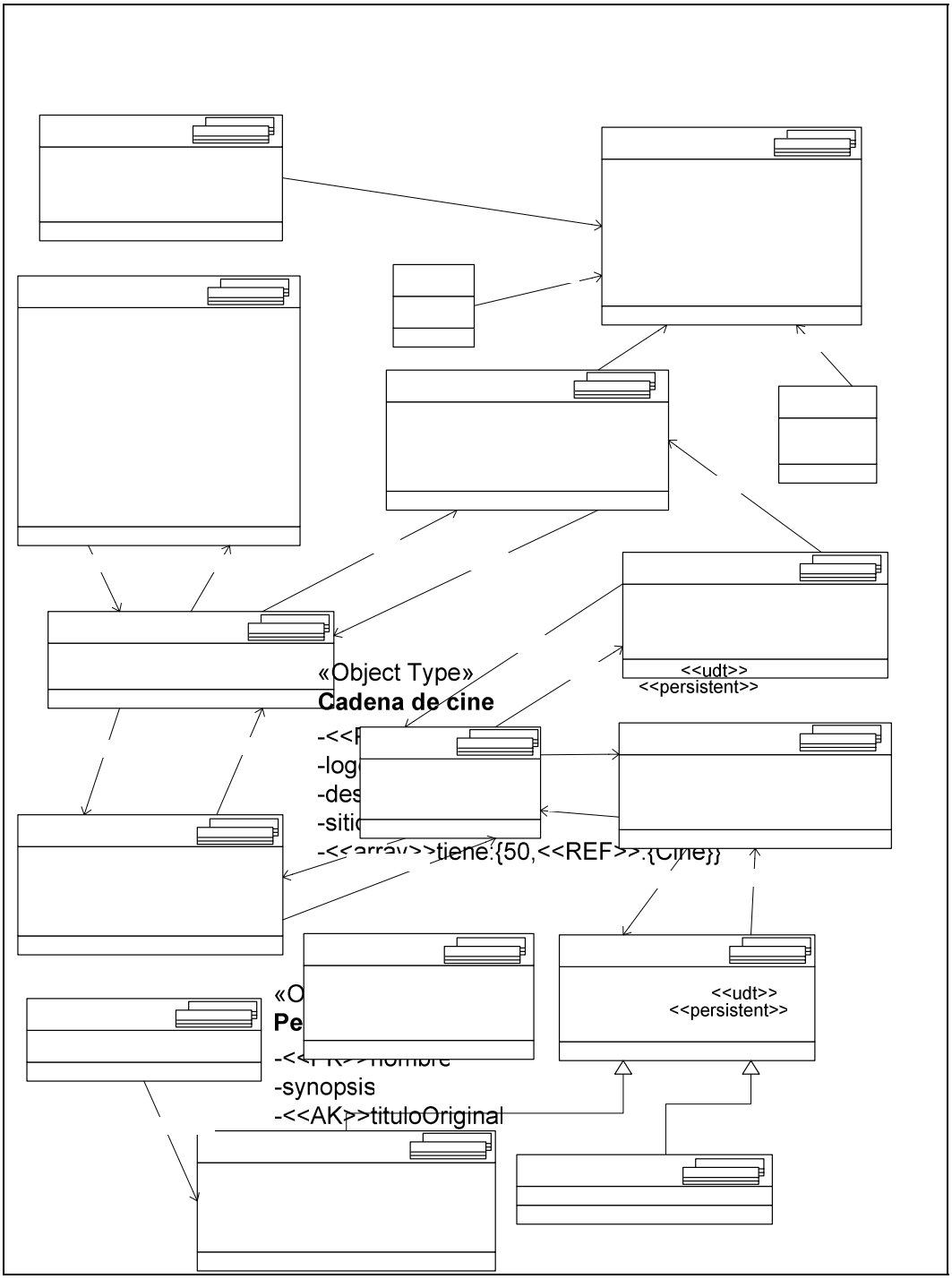


Figura 23. Diseño Lógico de la BD en SQL: 1999

A partir del diseño lógico estándar se pasa a definir el ***diseño lógico específico***, es decir el diseño para el producto concreto, pero sin tener en cuenta consideraciones de optimización de espacio y rendimiento. Como producto se ha elegido Oracle. Al comienzo de esta tesis se comenzó a trabajar con la versión 8*i*, la última disponible en aquel momento, por lo que la primera versión de las extensiones se realizó teniendo en cuenta el modelo soportado en aquel momento. Posteriormente, y tras la aparición de la versión 9*i*, se actualizaron las extensiones y las guías de transformación. En la Figura 24 se muestra el diagrama UML para Oracle8*i* (por ser bastante más diferente del diagrama del estándar, que el de Oracle9*i*) y en el apéndice C se puede ver el mismo diagrama para la versión 9*i*. La diferencia principal, aunque no la única, consiste en que esta nueva versión de Oracle incorpora herencia.

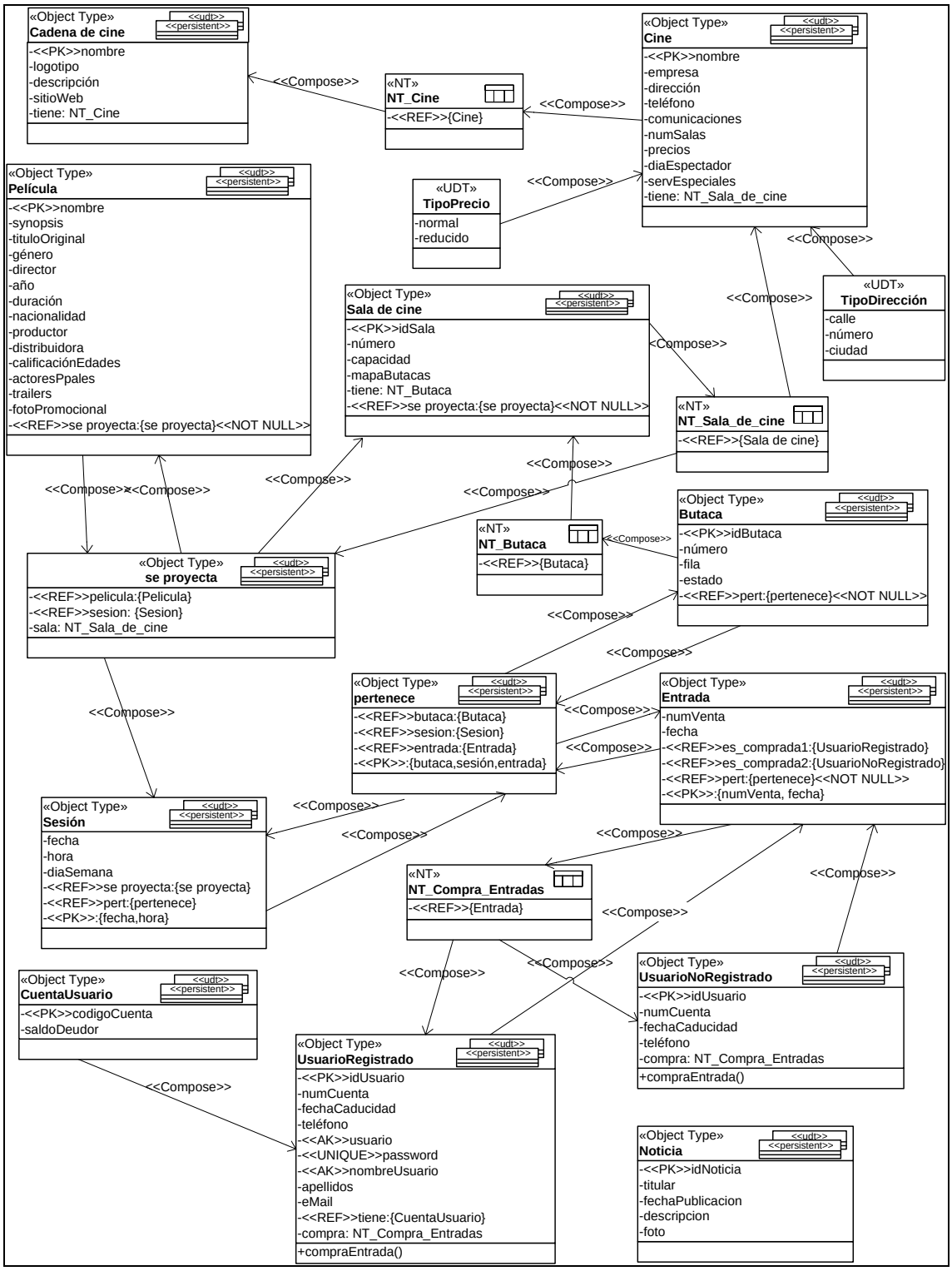


Figura 24. Diseño Lógico de la BD en Oracle 8*i*

A continuación, se pasa a realizar la ***implementación de la BD***. A partir del diseño lógico específico se pasa a especificar el correspondiente código SQL en el producto seleccionado. Como ya se ha dicho, este modo de representar el diseño lógico específico ayuda en el proceso de compilación del código SQL. La Figura 25

muestra el código SQL en Oracle 8*i*. El código para la versión 9*i* puede encontrarse en el apéndice C.

```
CREATE OR REPLACE TYPE TipoPrecio AS OBJECT
(normal NUMBER,
reducido NUMBER)
/
CREATE OR REPLACE TYPE TipoDireccion AS OBJECT
(calle VARCHAR(30),
numero NUMBER,
ciudad VARCHAR(20))
/
CREATE OR REPLACE TYPE Cine AS OBJECT
(nombre VARCHAR(30),
empresa VARCHAR(30),
direccion TipoDireccion,
telefono NUMBER,
comunicaciones VARCHAR2(200),
numSalas NUMBER,
precios TipoPrecio,
diaEspectador VARCHAR(9),
servEspeciales VARCHAR2(300),
tiene NT_Sala_de_cine)
/
CREATE OR REPLACE TYPE T_Ref_Cine AS OBJECT
(Ref_Cine REF Cine)
/
CREATE OR REPLACE TYPE NT_Cine AS TABLE OF T_Ref_Cine
/
CREATE OR REPLACE TYPE Cadena_de_cine AS OBJECT
(nombre VARCHAR(15),
logotipo BLOB,
descripcion VARCHAR2(4000),
sitioWeb VARCHAR(30),
tiene NT_Cine)
/
CREATE OR REPLACE TYPE Sala_de_cine AS OBJECT
(idSala NUMBER,
numero NUMBER,
capacidad NUMBER,
mapaButacas BLOB,
tiene NT_Butaca,
seproyecta REF Se_proyecta)
/
CREATE OR REPLACE TYPE T_Ref_Sala_de_cine AS OBJECT
(Ref_Sala_de_cine REF Sala_de_cine)
/
CREATE OR REPLACE TYPE NT_Sala_de_cine AS TABLE
                OF T_Ref_Sala_de_cine
/
CREATE OR REPLACE TYPE Butaca AS OBJECT
(idButaca NUMBER,
numero NUMBER,
fila NUMBER,
estado VARCHAR(10),
pert REF Pertenece)
/
CREATE OR REPLACE TYPE T_Ref_Butaca AS OBJECT
(Ref_Butaca REF Butaca)
/
CREATE OR REPLACE TYPE NT_Butaca AS TABLE
OF T_Ref_Butaca
/
CREATE OR REPLACE TYPE Se_proyecta AS OBJECT
(apelicula REF Pelicula,
asesion NT_Sesion,
asala NT_Sala_de_cine)
/
CREATE OR REPLACE TYPE Pelicula AS OBJECT
(Nombre VARCHAR(50),
synopsis VARCHAR2(2000),
tituloOriginal VARCHAR(50),
genero VARCHAR(15),
director VARCHAR(30),
anyo NUMBER,
duracion NUMBER,
nacionalidad VARCHAR(20),
productor VARCHAR(30),
distribuidora VARCHAR(20),
calificacionEdades VARCHAR(20),
actoresPpales VARCHAR(500),
trailers BLOB,
fotoPromocional BLOB,
seproyecta REF Se_proyecta)
/
CREATE OR REPLACE TYPE Sesion AS OBJECT
(fecha DATE,
hora NUMBER,
diaSemana VARCHAR(9),
seproyecta REF Se_proyecta,
pert REF Pertenece)
/
CREATE OR REPLACE TYPE T_Ref_Sesion AS OBJECT
(Ref_Sesion REF Sesion)
/
CREATE OR REPLACE TYPE NT_Sesion AS TABLE OF T_Ref_Sesion
/
CREATE OR REPLACE TYPE Entrada AS OBJECT
(numVenta NUMBER,
fecha DATE,
es_comprada1 REF UsuarioRegistrado,
es_comprada2 REF UsuarioNoRegistrado,
pert REF Pertenece)
/
CREATE OR REPLACE TYPE Pertenece AS OBJECT
(abutaca REF Butaca,
asesion REF Sesion,
aentrada REF Entrada)
/
CREATE OR REPLACE TYPE T_Ref_Entrada AS OBJECT
(Ref_Entrada REF Entrada)
/
CREATE OR REPLACE TYPE NT_Compra_Entradas
AS TABLE OF T_Ref_Entrada
/
CREATE OR REPLACE TYPE UsuarioNoRegistrado AS OBJECT
(idUsuario NUMBER,
numCuenta NUMBER,
fechaCaducidad DATE,
telefono NUMBER,
compra NT_Compra_Entradas)
/
CREATE OR REPLACE TYPE CuentaUsuario AS OBJECT
(codigoCuenta NUMBER,
saldoDeudor NUMBER,
pert REF UsuarioRegistrado)
/
CREATE OR REPLACE TYPE UsuarioRegistrado AS OBJECT
(idUsuario NUMBER,
numCuenta NUMBER,
fechaCaducidad DATE,
telefono NUMBER,
usuario VARCHAR(15),
password VARCHAR(15),
nombreUsuario VARCHAR(20),
apellidos VARCHAR(30),
eMail VARCHAR(30),
tiene REF CuentaUsuario,
compra NT_Compra_Entradas)
/
```

```
CREATE TABLE Tabla_Cadena_de_cine OF Cadena_de_cine
(PRIMARY KEY(nombre))
NESTED TABLE tiene STORE AS TCine
/
CREATE TABLE Tabla_Cine OF Cine
(PRIMARY KEY(nombre))
NESTED TABLE tiene STORE AS TSala1
/
CREATE TABLE Tabla_Pelicula OF Pelicula
(PRIMARY KEY(nombre),
tituloOriginal UNIQUE)
/
CREATE TABLE Tabla_Sala_de_cine OF Sala_de_cine
(PRIMARY KEY(idSala))
NESTED TABLE tiene STORE AS TButaca
/
CREATE TABLE Tabla_Butaca OF Butaca
(PRIMARY KEY(idButaca),
CHECK(estado='Libre' OR estado='Ocupada' OR
estado='Seleccionada'))
/
CREATE TABLE Tabla_Se_proyecta OF Se_proyecta
NESTED TABLE asesion STORE AS TSesion
NESTED TABLE asala STORE AS TSala2
/
CREATE TABLE Tabla_Entrada OF Entrada
(PRIMARY KEY(numVenta,fecha),
es_comprada1 NOT NULL,
es_comprada2 NOT NULL)
/
CREATE TABLE Tabla_Pertenece OF Pertenece
(abutaca NOT NULL,
asesion NOT NULL)
/
CREATE TABLE Tabla_Sesion OF Sesion
(PRIMARY KEY(fecha,hora))
/
CREATE TABLE Tabla_CuentaUsuario OF CuentaUsuario
(PRIMARY KEY(codigoCuenta),
pert NOT NULL)
/
CREATE TABLE Tabla_UsuarioRegistrado OF UsuarioRegistrado
(PRIMARY KEY(idUsuario),
tiene NOT NULL)
NESTED TABLE compra STORE AS TCompra1
/
CREATE TABLE Tabla_UsuarioNoRegistrado OF UsuarioNoRegistrado
(PRIMARY KEY(idUsuario))
NESTED TABLE compra STORE AS TCompra2
/
```

Figura 25. Código SQL en Oracle 8*i*

TAREA 2: Diseño e Implementación del Hipertexto

Una vez realizado el diseño lógico de la BDOR y la implementación en SQL, o en paralelo si así se desea, se pasará a realizar el diseño lógico del hipertexto, que consiste, por un lado en la especificación de cada uno de los fragmentos en esquemas XML y por otro lado, en la definición del modelo lógico de navegación

utilizando XLink. La representación, tanto de los esquemas XML como del XLink, se hará en notación UML.

Para obtener el ***diseño lógico de fragmentos***, cada fragmento del diagrama conceptual de fragmentos se transformará en un esquema XML, empleando para ello la notación UML propuesta y las guías especificadas en los apartados 3.3.2.2 y 3.3.2.5, respectivamente. Por lo tanto, se ha generado un esquema XML para cada uno de los fragmentos del diagrama conceptual de fragmentos. En la Figura 26 se presenta el esquema XML generado a partir del fragmento *Cadena de Cines* en notación UML, así como el correspondiente código XML.

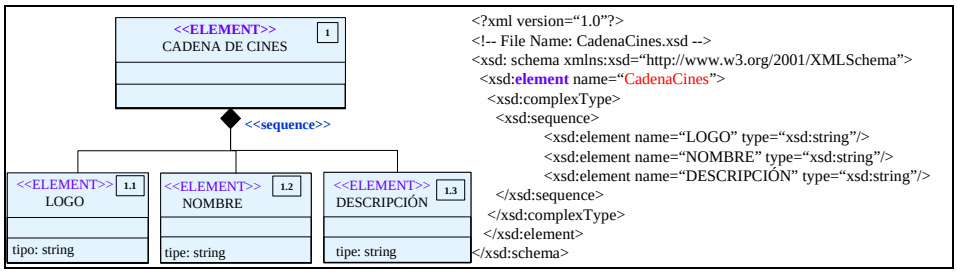


Figura 26. Un fragmento representado con un esquema XML con notación UML y el código correspondiente

Una vez obtenido los esquemas XML para cada uno de los fragmentos del diagrama conceptual de fragmentos, se pasa a realizar el ***diseño lógico de navegación***. Para pasar del diagrama conceptual de navegación al correspondiente en el nivel lógico, hay que transformarlo a XLink, siguiendo las guías de transformación y la notación UML extendida propuestas en los apartados 3.3.2.4 y 3.3.2.5, respectivamente. Tal y como se observa en la Figura 27, el diagrama conceptual de navegación completo se transforma en el nivel lógico, en un enlace extendido **<<Extended Link>>**, que relaciona un conjunto de recursos. Hay que hacer notar, que en la Figura 27 sólo se muestra una parte del modelo lógico de navegación generado, por motivos de claridad. En el apéndice C se incluye el modelo lógico de navegación completo XLink. El enlace extendido se representa en XLink mediante una clase abstracta denominada *'Modelo Lógico de Navegación'*. Esta clase se relaciona con todos los fragmentos que componen el diagrama lógico de navegación mediante una composición estereotipada <<is_composed>>. Cada uno de los fragmentos del diagrama conceptual de fragmentos se transforma, a nivel lógico, en un recurso local, representado con un elemento **<<XLink Locator>>** y denominado con el nombre del fragmento correspondiente. De esta forma, se ha creado un recurso para cada uno de los fragmentos *Cadena de Cines*, *Cine*, *Película* y *Sesión*. Los elementos de acceso del modelo lógico de navegación: índices, visitas guiadas, consultas y menús, también se transforman a un recurso

representado con un <<XLink Locator>>; en nuestro caso, tenemos cuatro índices que se definen con los correspondientes recursos locales. La navegación entre dos fragmentos, o entre un fragmento y un elemento de acceso, se representa medinte una asociación **<<arc>>**. La información de cómo se realiza dicha transición se incluye en la asociación *arc* especificando los valores etiquetados correspondientes. En la Figura 27 se muestra el XLink obtenido a partir del diagrama conceptual de navegación y en la Figura 28 el código XLink correspondiente.

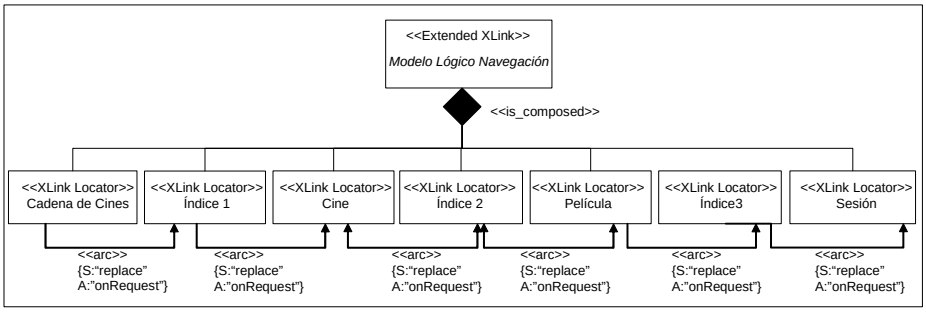


Figura 27. Diagrama Lógico de Fragmentos en XLink representado con UML

```
<Enlaces xmlns:xlink=http://www.w3.org/1999/xlink>
<ModeloLogicoNavegacion xlink:type="extended"
title="Consorcio Cines"
<!—definición local de todos
los recursos-->
<cadenaCines xlink:type "resource"
Xlink:label="fragmentoCadenaCines"
</cadenaCines>
<indice1 xlink:type "resource"
Xlink:label="fragmentoIndice1"
</indice1>
<cine xlink:type "resource"
Xlink:label="fragmentoCine"
</cine>
<indice2 xlink:type "resource"
Xlink:label="fragmentoIndice2"
</indice2>
<pelicula xlink:type "resource"
Xlink:label="fragmentoPelicula"
</pelicula>
<indice3 xlink:type "resource"
Xlink:label="fragmentoIndice3"
</indice3>
<sesion xlink:type "resource"
Xlink:label="fragmentoIndice"
</sesion>

<!-- definicion de los
enlaces entre fragmentos-->
<enlace1 xlink:type="arc"
xlink:from =
"fragmentoCadenaCines"
xlink:to="fragamentoIndice1"
xlink:show="replace"
xlink:actuate="onRequest" />
<enlace2 xlink:type="arc"
xlink:from ="fragmentoIndice1"
xlink:to="fragmentoIndice"
xlink:show="replace"
xlink:actuate="onRequest" />
<indice3 xlink:type="arc"
xlink:from ="fragmentoIndice"
xlink:to="fragmentoIndice2"
xlink:show="replace"
xlink:actuate="onRequest" />
<indice4 xlink:type="arc"
xlink:from ="fragmentoIndice2"
xlink:to="fragmentoPelicula"
xlink:show="replace"
xlink:actuate="onRequest" />

<enlace5 xlink:type="arc"
xlink:from ="fragmentoPelicula"
xlink:to="fragmentoIndice3"
xlink:show="replace"
xlink:actuate="onRequest" />
<indice6 xlink:type="arc"
xlink:from ="fragmentoIndice3"
xlink:to="fragmentoSesion"
xlink:show="replace"
xlink:actuate="onRequest" />
<enlace7 xlink:type="arc"
xlink:from ="fragmentoPelicula"
xlink:to="fragmentoIndice2"
xlink:show="replace"
xlink:actuate="onRequest" />
<enlace8 xlink:type="arc"
xlink:from ="fragmentoIndice2"
xlink:to="fragmentoCine"
xlink:show="replace"
xlink:actuate="onRequest" />
</ModeloLogicoNavegacion>
</Enlaces>
```

Figura 28. Código XLink correspondiente

Una vez implementados el hipertexto y la BD sólo queda la tarea de ***integración*** entre ambos, a fin de conseguir que las páginas Web se generen dinámicamente. En este caso la integración se ha realizado utilizando ASP, aunque existen, por supuesto otras posibilidades. Entre ellas, las aportadas por XML DB, la nueva versión de Oracle (9*i* release 2), que soporta directamente la actualización de documentos XML a partir de una BDOR que almacena el correspondiente esquema XML. También están apareciendo otras propuestas basadas en la

integración directa de los lenguajes de consulta de SQL y de XML, como XSQL o XQuery.

4.3 ADD-INs en Rational Rose

Además de los casos de estudio y de los casos de prueba realizados, se han implementado tres extensiones a Rational Rose (mediante Add-Ins). Estos módulos permiten: a) el modelado de esquemas OR, b) el modelado de esquemas XML en notación UML y c) el modelado de XLinks en notación UML.

Así, estos tres Add-Ins han permitido una segunda validación de las extensiones propuestas.

La Figura 29, Figura 30 y Figura 31 muestran respectivamente una pantalla del módulo para el modelado de esquemas OR, una del módulo para el modelado de esquemas XML y otra para del módulo para XLink, respectivamente.

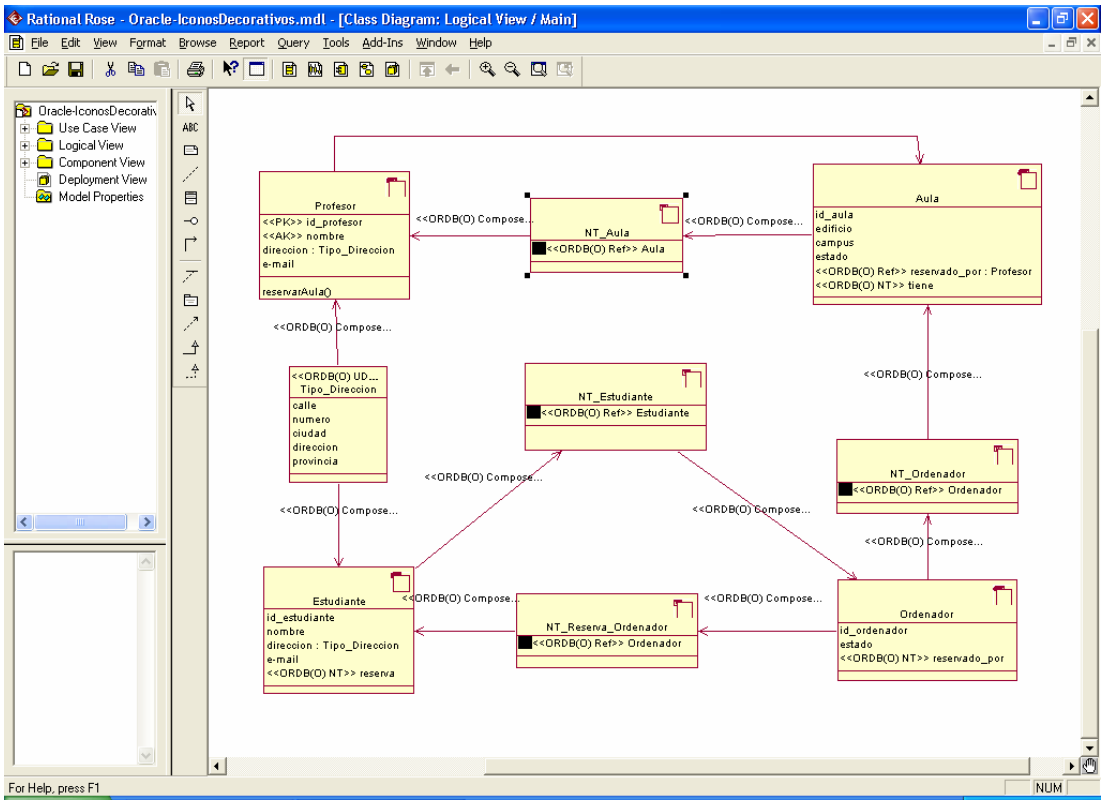


Figura 29. ADD-IN para módulo OR

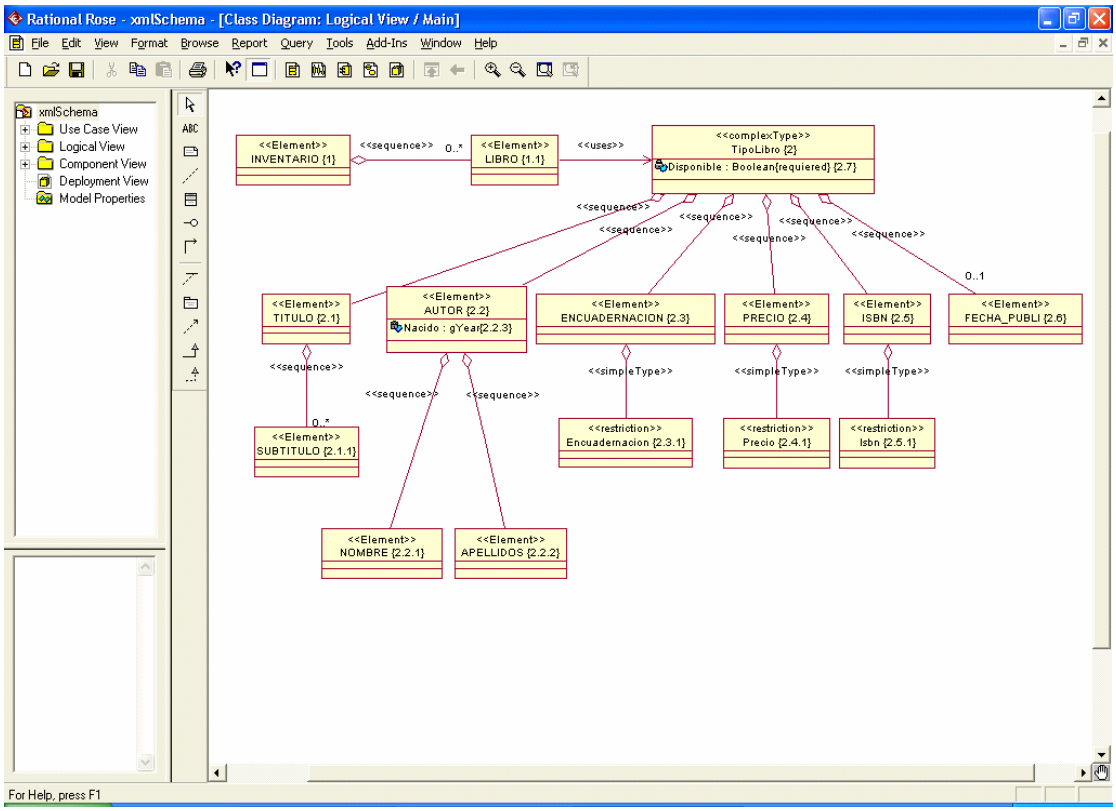


Figura 30. ADD-IN para módulo XML Schema

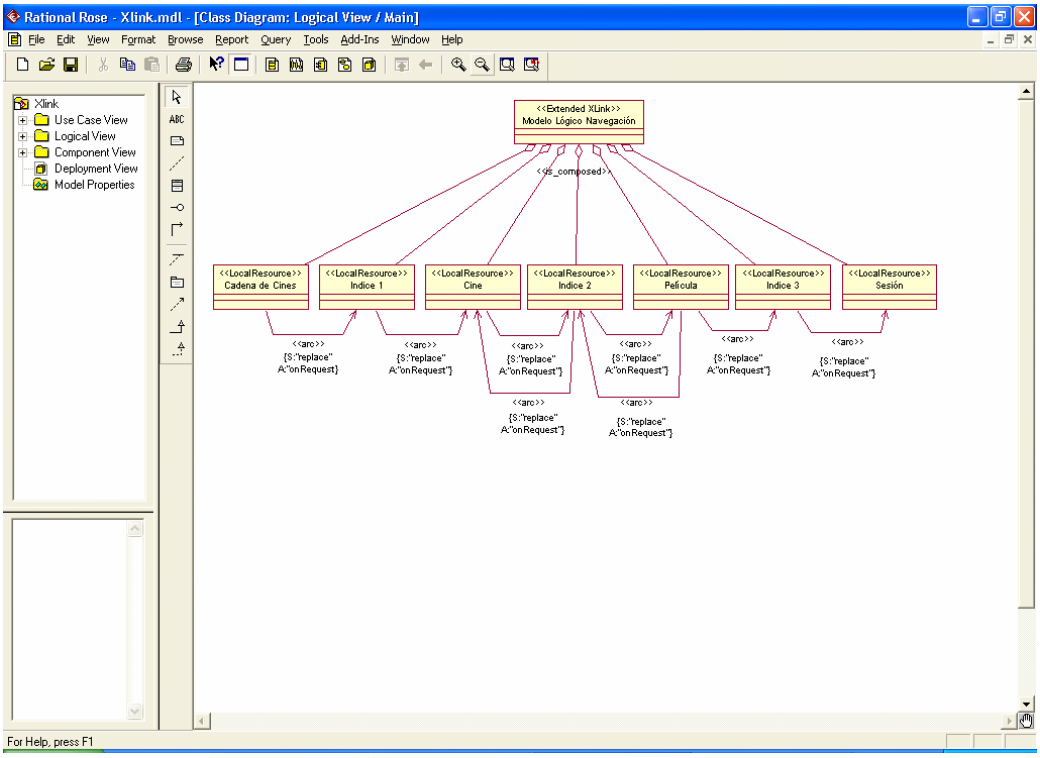


Figura 31. ADD-IN para módulo XLink

Conclusiones y Trabajos Futuros

En este capítulo se analiza cómo se han cumplido los objetivos indicados al principio de esta tesis, se mencionan las principales aportaciones del trabajo de investigación, se contrastan los resultados mediante las publicaciones realizadas y se exponen las líneas de trabajo que permanecen abiertas.

5.1 Análisis de la consecución de objetivos

En la sección 1.2 de este trabajo se plantearon una serie de objetivos parciales cuya finalidad era alcanzar el objetivo principal de esta investigación: la definición de la metodología presentada en el capítulo 0 de esta tesis.

A continuación, se analiza el resultado de cada uno de estos objetivos parciales.

Objetivo 1: *Analizar las distintas metodologías existentes para el desarrollo de SIW, concretamente, las que se centran en los aspectos estructurales (BD Web e hipertexto), determinando sus aportaciones, así como sus limitaciones, con el fin de abordar problemas no resueltos e incorporar las mejores prácticas de otras metodologías.*

Si bien se ha comprobado la ausencia de metodologías completas y comúnmente aceptadas para el desarrollo de la dimensión estructural de SIW, se ha realizado un análisis de las propuestas más importantes de los últimos años realizadas en este sentido. Este análisis se ha completado indicando las principales aportaciones y limitaciones de cada una de ellas. Tal y como se había planteado al comienzo de esta tesis, en el objetivo 1, MIDAS/DB ha incorporado técnicas, notaciones, etc. de otras propuestas, centrándose en la resolución de problemas no resueltos. Así, por ejemplo, MIDAS/DB incluye los modelos de fragmentos y navegación inicialmente propuestos en RMM pero utilizando una adaptación de la notación UML de UWE, que surgió cuando esta tesis ya estaba avanzada.

Hay que destacar que el análisis y estudio de estas metodologías, se ha llevado a cabo de modo continuado ya que, debido a la rapidez con la que cambia la tecnología implicada, las propuestas metodológicas varían necesariamente a la misma velocidad. Esto ha implicado que durante el tiempo que ha llevado la realización de la tesis, hayan variado los problemas a resolver en función de las necesidades específicas del momento. De hecho, cuando comenzó este trabajo no existían, por ejemplo, propuestas metodológicas que contemplaran el desarrollo de SIW con tecnología XML. Sin embargo, en la actualidad, prácticamente todas las

metodologías de desarrollo de SIW están orientando sus trabajos hacia el desarrollo basado en tecnología XML.

Objetivo 2: *Definir el marco metodológico para el desarrollo de la dimensión estructural de un SIW, especificando el proceso, las actividades, las tareas, técnicas y notaciones a utilizar.*

Se ha definido un marco metodológico, MIDAS/DB, incluyendo las actividades, tareas, técnicas y notaciones necesarias para desarrollar la dimensión estructural de un SIW, que incluye el desarrollo del hipertexto, de la BD y de la presentación. Para ello se han definido algunas técnicas nuevas y, como ya se ha mencionado, cuando ha sido posible se han incorporado técnicas de otras metodologías existentes.

Objetivo 3: *Definir las extensiones necesarias de UML para poder representar todo el sistema en una única notación*.

Cuando se planteó MIDAS/DB, una de las características diferenciadoras respecto a otras metodologías existentes en aquel momento, era que todo el sistema se representaría en una única notación, UML. Sin embargo, como ya se ha indicado anteriormente, durante la realización de este trabajo de tesis, comenzaron a aparecer otras propuestas de desarrollo basadas en UML. Algunas de estas propuestas han sido incorporadas en MIDAS/DB, como las de UWE para el modelado del hipertexto. Sin embargo, ha sido necesario definir nuevas extensiones para la representación de algunas técnicas propuestas en MIDAS/DB. En concreto, se han definido dos:

- Una para Objeto-Relacional (OR), incluyendo extensiones para el modelo OR y para el modelado de consultas.
- Otra para XML, incluyendo extensiones para XML esquemas y XLink.

Objetivo 4: *Definir las guías para la transición entre los modelos propuestos para cada una de las fases del desarrollo*.

Se han definido guías de transformación con el fin de pasar de un modelo a otro en el proceso de desarrollo de la dimensión estructural. En concreto se han especificado las siguientes guías de transformación:

- De un modelo conceptual de datos a un modelo OR estándar (SQL:1999); del modelo OR estándar a un modelo OR específico, en concreto el modelo de Oracle (versiones 8*i* y posteriormente, 9*i)*.
- De un modelo conceptual de datos a un modelo conceptual de navegación.

- De un modelo conceptual de navegación a un modelo lógico en XML: transformando los fragmentos a XML esquemas y el modelo conceptual de navegación a XLink.

Objetivo 5: *Validar las extensiones de UML, así como las guías de transformación propuestas, mediante la implementación de las mismas en ADD-INs de Rational Rose y su aplicación a distintos casos de prueba*.

En el capítulo 4 se ha descrito la implementación de las extensiones propuestas como ADD-INs de Rational Rose. Además, cada extensión se ha utilizado en distintos casos de prueba. De este modo, se han podido validar parcialmente las extensiones propuestas.

Objetivo 6: *Validar el marco metodológico mediante su aplicación a distintos casos de estudio y casos de prueba*.

MIDAS/DB se ha definido mediante un proceso iterativo e incremental, basado en las propuestas de Investigación en Acción, utilizando casos de estudio. Los casos de estudio han servido para determinar los problemas y necesidades, desde un punto de vista metodológico e ingenieril, en el desarrollo de SIW. Pero, además de los casos de estudio, una vez especificada la metodología, MIDAS/DB se ha aplicado a distintos casos de prueba, lo que ha permitido refinarla. En el capítulo 4 se han explicado estos casos, discutiendo algunas de las lecciones aprendidas.

5.2 Principales aportaciones

El trabajo realizado ha supuesto una serie de aportaciones relativas tanto al tema principal de la investigación (la definición de la propuesta metodológica para el desarrollo de la dimensión estructural de SIW), como a otros temas afines, en el contexto del marco del trabajo, y que se pasa a exponer a continuación:

Nueva metodología:

Se ha desarrollado una metodología basada en modelos para el desarrollo de la dimensión estructural de Sistemas de Información Web (SIW). Dicha metodología está ***basada en estándares***, UML, XML y SQL:1999, y propone el uso de ***UML*** para la definición de todo el SIW. Cuando se comenzó a trabajar en esta tesis no había ninguna metodología que cumpliera las características de MIDAS/DB. En la actualidad, aunque prácticamente todas las propuestas existentes consideran UML como el lenguaje de modelado, así como tecnología XML y OR, hasta donde la doctoranda conoce, ninguno de estos trabajos integran todas estas características.

Extensiones UML para el modelado OR:

Se ha propuesto una extensión de UML que permite especificar esquemas OR, tanto basados en el estándar SQL: 1999, como en un producto concreto, Oracle (se han considerado las versiones 8*i* y 9*i*). Dicha extensión es similar al conocido grafo relacional, utilizado en el diseño de BD relacionales, pero para el modelo OR, por lo que incluye también la posibilidad de especificar esquemas puramente relacionales en UML.

Se han propuesto también las extensiones necesarias para la especificación en UML de consultas, tanto a nivel conceptual (es decir, a un esquema conceptual) como lógico (es decir, a un esquema OR).

Extensiones UML para el modelado de XML:

Para el diseño lógico del hipertexto se ha propuesto la utilización de XML, definiendo cada fragmento mediante un XML esquema y los enlaces del diagrama de navegación con XLink. Para poder especificar el diseño lógico del hipertexto en UML, se han propuesto las extensiones necesarias para representar XML esquemas y XLink en notación gráfica con UML.

Implementación en Rational Rose:

Las extensiones de UML propuestas en este trabajo se han implementado mediante ADD-INs de Rational Rose.

5.3 Contrastación de resultados

5.3.1 Publicaciones

Algunos de los resultados presentados en esta tesis han sido contrastados mediante su publicación en distintos foros tanto nacionales como internacionales. Se han agrupado los artículos según el tipo de publicación:

Artículos en Revistas Internacionales

- E. Marcos, B. Vela y J. M. Cavero. “Methodological Approach for Object-Relational Database Design”. *SoSyM (Journal on Software and System Modeling)*. Ed. Springer Verlag, Heidelberg (Alemania). Editores B. Rumpe y R. France. Vol. 2, páginas 59-72, marzo 2003.

Artículos en Revistas Nacionales

- E. Marcos, B. Vela y J. M. Cavero. "Una Extensión de UML para el Diseño de Bases de Datos Objeto-Relacionales". *CUORE (Círculo de Usuarios Oracle de España)*. VIVAT ACADEMIA.Vol. Nº19, páginas 27-39, febrero 2002.
- E. Marcos y B. Vela. "El proceso de creación de una base de datos Web". *El profesional de la información*. Ed. Swets Blackwell, Barcelona (España).Vol. 11, Nº 4, páginas 248-255, julio-agosto 2002.

Congresos Internacionales

- E. Marcos, P. Cáceres, B.Vela y J.M. Cavero. "MIDAS/BD: a Methodological Framework for Web Database Design." 20th International Conference on Conceptual Modeling ER 2001: International Workshop on Data Semantics in Web Information Systems (DASWIS-2001). En *Conceptual Modeling for New Information Systems Technologies*. Ed.: Hiroshi Arisawa, Yahiko Kambayashi, Vijay Kumar, Heinrich C. Mayr, and Ingrid Hunt, Springer Verlag LNCS-2465, pp. 227-238, septiembre 2002 (aceptados 27,7%).
- E. Marcos, B. Vela, J.M. Cavero y P. Cáceres. "Aggregation and Composition in Object-Relational Database Design". *5th East-European Conference on Advances in Databases and Informations Systems* (ADBIS'01). Research Communications. Ed. Albertas Caplinskas y Johann Eder, pp. 195-209. Vilnius (Lithuania), 2001 (aceptados 50%).
- E. Marcos, B. Vela y J.M. Cavero. "Extending UML for Object-Relational Database Design". En: *UML 2001- The Unified Modelling Language. Modelling Languages, Concepts and Tooles***.** Springer Verlag, LNCS 2185, pp. 225-239, 2001 (aceptados 26%).
- J.M. Cavero, E. Marcos, B.Vela y C. Costilla. "Modelling ORDB Queries using UML", *Fifth Internacional Conference on Enterprise Information Systems*. ICEIS 2003. Proceedings of ICEIS. Ed. O. Camp, J. Filipe, S. Hammoudi, M. Piattini. Volumen 3, pp.535-539, Angers (Francia), 23-26 de abril de 2003.

- B. Vela y E. Marcos. “Extending UML to represent XML Schemas”. *The 15th Conference on Advanced Information Systems Engineering*. CAISE’03 Forum. Proceedings Short Papers. Ed. J. Eder y T. Welter. 2003, pp.97-100. Velden-Klagenfurt, Austria, 16-20 de junio de 2003 (aceptados 50%).
- P.Caceres, E. Marcos y B. Vela. “A MDA-Based Approach for Web Information System Development”. Workshop in Software Model Engineering (WiSME-2003). UML ’03 Forum. San Francisco, USA. 21 de octubre 2003. Aceptado.

Congresos Iberoamericanos

- B. Vela, J. M. Cavero y E. Marcos. “Diseño de Bases de Datos Objeto-Relacionales con UML.” 1ª Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC’2001). *Anales de las Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento*. Ed. Silvia Teresita Acuña y Cecilia María Lasserre. Editorial Universidad Nacional del Jujuy, 2001, pp. 59-68. Buenos Aires (Argentina), 13-15 junio 2001 (aceptados 35%).
- B. Vela y E. Marcos. “Una extensión de UML para representar XML Schemas”. *6º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software*. Memorias de Ideas 2003, Ed. M. Piattini, L. Cernuzzi y F. Ruíz. 2003, pp.109-119. Asunción (Paraguay), 30 de abril a 2 de mayo de 2003 (aceptados 50 %).
- E. Marcos, V. Castro y B. Vela. “Modelado de Servicios Web con UML: Un caso de estudio”. Workshop on Advances in Database and Information Retrieval. Apizaco-Tlaxcala (México), 8-9 septiembre 2003. Aceptado.

Congresos Nacionales

- J. M. Cavero, C. Costilla, E. Marcos y B.Vela. “Modelado de consultas a BDOR con UML”. JISBD 2002, VII Jornadas de Ingeniería del Software y Bases de Datos. Actas de las VII Jornadas de Ingeniería del Software y Bases de Datos. El Escorial (Madrid), 19-21 noviembre 2002.
- B.Vela y E. Marcos “Una extensión de UML para XML Schemas”. Taller de Ingeniería del Software Orientada al Web (WebE'02), dentro de las VII Jornadas de Ingeniería del Software y Bases de Datos. Actas

del taller Web de las JISBD 2002. El Escorial (Madrid), 19-21 noviembre 2002.

5.4 Líneas de investigación abiertas

Como ocurre en todo trabajo de investigación, son varios los frentes abiertos para continuar el trabajo presentado en la tesis. Algunos de ellos proceden de problemas que se conocían pero que, sin embargo, no han sido objeto de esta tesis y otros han surgido como nuevos problemas durante la realización de la misma. A continuación se señalan las principales líneas de trabajo abiertas:

- MIDAS/DB es una metodología que aborda el desarrollo de la dimensión estructuctural de SIW. Por tanto, la primera línea natural de trabajo es completar la metodología con el fin de ***contemplar también los aspectos de comportamiento***. En paralelo a esta tesis, se está realizando otra cuyo objetivo es la especificación de una instanciación de MDA para SIW; la arquitectura resultante de esta tesis, de carácter horizontal, contemplará aspectos tanto de estructura como de comportamiento. El modelado de los aspectos de comportamiento se están estudiando en el marco de otra tesis doctoral y como primer trabajo, se ha realizado ya una extensión de UML para el modelado de servicios Web basada en WSDL.
- Las técnicas propuestas en MIDAS/DB se han implementado como ADD-INS de Rational Rose. Otro aspecto que queda pendiente es la integración de todas las técnicas para el desarrollo completo de un SIW (tanto en su dimensión estructural como de comportamiento) en una ***herramienta*** que permita la generación (semi-)automática de SIW. Esta herramienta se está desarrollando en el marco de las tesis mencionadas.
- Un aspecto que deja abiertas algunas líneas de trabajo es el relativo a la integración de consultas en el diagrama de navegación. Por una parte, porque es necesario incluir elementos de acceso que permitan representar consultas más complejas. Por otra parte, además de representar consultas a BD objeto-relacionales, se extenderá este trabajo para incluir consultas a BD XML, incluyendo extensiones de UML para representar XQuery, XML SQL y XSQL. Además, y dado que las consultas determinan el tipo de accesos a la BD, se refinarán las guías de transformación a un esquema de BD en función de las consultas más comunes al mismo.

- En la actualidad, dentro del grupo de investigación Kybele y en el marco del proyecto DAWIS [TIC 2002-04050-C02-01], se está trabajando en la especialización de MIDAS para el desarrollo de portales Web de acceso integrado a archivos digitales. Este trabajo permitirá integrar los resultados de la tesis doctoral que se presenta, con la experiencia adquirida en el proyecto E-Parking durante la estancia de la doctoranda en el grupo de Bases de Datos (DBTG) de la Universidad de Zürich (Suiza).

Bibliografía

Ambler (1999). Persistence Modelling in the UML. Ambler, S. En: http://www.sdmagazine.com/articles/1999/0008/0008q/0008q.htm

Akehurst y Bordbar (2001). On Querying UML Data Models with OCL. Akehurst, D. H. y Bordbar, B. UML 2001 – The Unified Modeling Language 2001. Ed. Gogolla, M. y Kobryn, C. Springer-Verlag. LNCS 2185, pp. 91 – 103. Toronto, Canadá, 2001.

Atzeni *et al.* (1998). Design and Maintenance of Data-Intensive Web Sites. Atzeni, P., Mecca, G. y Merialdo, P. Advances in Database Technology. Ed. Sheck, Saltor, Ramos, Alonso. Proceedings of the EDBT'98. Springer-Verlag, Valencia, España, 1998.

Atzeni *et al.* (1999). Database Systems. Concepts, Languages and Architectures. Atzeni, P., Ceri, S., Paraboschi, S. y Torlone, R. McGraw-Hill, 1999.

Avison *et al.* (1999). Action Research. Avison, D., Lan, F., Myers, M. y Nielsen, A. Communications of the ACM, 42(1), pp. 94-97, 1999.

Barbosa *et al.* (2001). ToX - The Toronto XML Engine. Barbosa, D., Barta, A., Mendelzon, A., Mihaila, G., Rizzolo, F. y Rodriguez-Gianolli, P. International Workshop on Information Integration on the Web, Rio de Janeiro, Brasil, 2001.

Baresi *et al.* (2001). Extending UML for Modelling Web Applications. Baresi, L., Garzotto, F., Paolini, P. Proceedings of the 34th Hawaii International Conference on System Sciences, IEEE Computer Society, 2001.

Baumeister *et al*. (1999). Towards a UML Extension for Hypermedia Design. Baumeister, H., Koch, N. y Mandel, L. Proceedings of the UML'99. The Unified Modelling Language - Beyond the Standard. Ed. Robert France y Bernhard Rumpe. Springer-Verlag, LNCS 1723, pp. 614-629. Fort Collins, USA, 1999.

Beck (1999). Embracing Change with eXtreme Programming. Beck, K. Computer, Volumen 32, nº 10, pp. 70-77, 1999.

Beleña (2002). Desarrollo de aplicaciones de Gestión con PL/SQL, XML y XSLT. Beleña, C. Revista del Circulo de Usuarios de Oracle de España (CUORE). Febrero 2002, pp. 9-18, 2002.

Bertino y Guerrini (1998). Extending the ODMG Object Model with Composite Objects. Bertino, E. y Guerrini, G. Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages &

Applications (OOPSLA' 98), in ACM SIGPLAN Notices, Octubre 1998, Volumen 33, Nº 10, pp. 259-270, 1998.

Bertino y Marcos (2000). Object Oriented Database Systems. Bertino, E. y Marcos, E. En Advanced Databases: Technology and Design. Ed. O. Díaz y M. Piattini. Artech House.

Bertino y Martino (1993). Object-Oriented Database Systems. Concepts and Architectures. Bertino, E. y Martino, L. Addison-Wesley.

Bertrand (2002). XML: el desarrollo de nuevas aplicaciones empresariales y la industria del software. Bertrand López de Roda, E. Nóvatica, Volumen 158, Julio-Agosto 2002, pp. 13-16, 2002.

Blaha y Premerlani (1998). Object-Oriented Modeling and Design for Database Applications. Blaha, M. y Premerlani, W. Prentice Hall.

Bonifati *et al.* (2000). Building Multi-device, Content-Centric Applications Usign WebML and the W3I3 Tool Suite. Bonifati, A., Stefano, C., Fraternali, P. y Maurino, A. Conceptual Modeling for E-Bussiness and the Web. Ed. S. W. Liddle, H. C. Mayr, B. Thalheim. Springer-Verlag, pp. 64-75, 2000.

Booch *et al.* (1997). Unified Modelling Language, V1.0. G. Booch, J. Rumbaugh y. Jacobson. Rational Software Corporation. Enero 1997.

Booch *et al.* (1999a). UML for XML Schema Mapping Specification. Booch G., Christerson, M., Fuchs, M. y Koistinen, J. En: http://www.rational.com/media/uml/resources/media/

Booch *et al.* (1999b). The Unified Modelling Language User Guide. Booch G., Rumbaugh, J. y Jacobson, I. Addison Wesley, 1999.

Bray *et al.* (1998). Extensible Markup Language (XML) 1.0, W3C Recommendation. Bray T., Paoli, J, Sperberg-McQu4een, C. M. y Maler, E., En: http://www.w3.org/TR/1998/REC-xml-19980210

Bray *et al.* (2000). Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation. Bray, T., Paoli, J., Sperberg-McQu4een, C. M. y Maler, E. En: http://www.w3.org/TR/2000/REC-xml-20001006/

Bunge (1976). La Investigación Científica. Bunge, M. Ariel, S.A. Barcelona.

Cabot *et al.* (2001). UML en el Diseño de Bases de Datos Relaciones. Cabot, J., García Esteban, R., Almaleh, Z., Marcos, E., Cáceres, P. y Vázquez, M. J. NOVATICA, Marzo-Abril 2001, Volumen 150, pp. 62-65, 2001.

Cáceres y Marcos (2001). Las Metodologías de Desarrollo y la Mejora de Calidad en las Aplicaciones Web. Cáceres, P. y Marcos, E. Actas del 4º Encontro para a

Qualidade nas Tecnologias de Informaçao e Comunicaçoes, pp. 3-9. Lisboa, Portugal, 2001.

Campbell *et al.* (2003). XML Schema. Campbell, C., Eisenberg, A. y Melton, J. ACM SIGMOD Record. Junio 2003, Volumen 32, Nº 2, pp. 96 – 101, 2003.

Case *et al.* (1996). A generic object-oriented design methodology incorporating database considerations. Case, T., Henderson-Sellers, B. y Low, G. C. Annals of Software Engineering. Volumen 2, pp. 5-24, 1996.

Castano *et al.* (2000). A General Methodological Framework for the Development of Web-Based Information Systems. Castano, S., Palopoli, L. y Torlone, R. Conceptual Modelling for E-Bussines and the Web. Ed. S. W. Liddle, H. C. Mayr y B. Thalheim. Springer-Verlag. LNCS 1921, pp. 128-139. Berlin, Alemania, 2000.

Cattell y Barry (2000). The Object Data Standard: ODMG 3.0. Cattell, R. G. G. y Barry, D. K. Morgan Kaufmann, 2000.

Cavero *et al.* (2003). Modelling ORDB Queries using UML. Cavero, J. M., Marcos, E., Vela, B. y Costilla, C. Proceedings of ICEIS 2003. Ecola Superior de Tecnología de Setúbal. Ed. O. Camp, J. Filipe, S. Hammoudi y M. Piattini, pp. 535-539, 2003.

Chaudhri *et al.* (2003). XML Data Management. Native XML and XML-Enabled Database Systems. Ed. Chaudhri, A. B., Rashid, A. y Zicari, R. Addison Wesley, 2003.

Conallen (1999). Modeling Web Applications Architectures with UML. Conallen, J. Communications of the ACM, Octubre 1999, 42, pp. 63-70, 1999.

Conallen (2000). Building Web Applications with UML. Conallen J. Addison Wesley, 2000.

Eisenberg y Melton (1999). SQL:1999, formerly known as SQL3. Eisenberg, A. y Melton, J. ACM SIGMOD Record, Marzo 1999, Volumen 28, Nº. 1, pp. 131-138, 1999.

Eisenberg y Melton (2002). SQL/XML is Making Good Progress. Eisenberg A. y Melton J. ACM SIGMOD Record, Volumen 31, Nº. 2, pp. 101-108, Junio 2002.

Elmasri y Navathe (2000). Fundamentals of Database Systems. Elmasri, R. y Navathe, S. B. Tercera edición. Addison-Wesley, 2000.

Fournier (1999). A Methodology for Client/Server and Web Application Development. Fournier, R. Prentice Hall.

Fowler (2001). The New Methodology. Fowler, M. En: http://www.martinfowler.com/articles/newMethodology.html.

Fraternali y Paolini (1998). A conceptual Model and a Tool Environment for Developping more Scalable, Dynamic and Customizable Web Applications. Fraternali, P. y Paolini, P. Advances in Database Technology. Ed. Sheck, Saltor, Ramos, Alonso. Proceedings of the 6th. Conference on Extended Database Technology (EDBT'98). Springer-Verlag. Valencia, España, 1998.

Fraternali (1999). Tools and approaches for developing data-intensive Web applications: A survey. Fraternali, P. ACM Computing Surveys, Volumen 31, nº 3, 1999.

French y Bell (1996). Desarrollo organizacional (quinta edición). French, W. L. y Bell, C. H. Jr. Naucalpán de Juárez, México: Prentice-Hall, 1996.

Garzotto *et al*. (1993). HDM - a Model-Based Approach to Hypertext Application Design. Garzotto, F., Paolini, P. y Schwabe, D. ACM TODS, Enero 1993, 11(1), pp. 1-26, 1993.

Gómez et al. (2000). Extending a Conceptual Modelling Approach to Web Application Design. Gómez, J., Cachero, C. y Pastor, O. Proc. of the 12th International Conference on Advanced Information Systems (CaiSE'00). Springer-Verlag. LNCS 1789, pp. 79-93, 2000.

Gómez *et al.* (2001). Conceptual Modeling of Device-Independent Web Applications. Gómez J., Cachero, C. y Pastor, O. IEEE Multimedia, 8(2), pp. 26-39, 2001.

Hennicker y Koch (2000). A UML-based Methodology for Hypermedia Design. Hennicker R. y Koch N. Proceedings of the Unified Modeling Language Conference, UML´2000. Ed. Evans, A. y Kent, S. Springer-Verlag, LNCS 1939, pp. 410-424, 2000.

Informix Corporation (1999). Informix Guide to SQL: Reference. Electronic Documentation, Informix Press, 1999.

Isakowitz *et al.* (1995). RMM: A Methodology for Structured Hypermedia Design. Isakowitz T., Stohr, E. A.S. y Balasubramanian P. Communications of the ACM, Agosto, 58(8), pp. 34-43, 1995.

Isakowitz *et al.* (1998). The Extended RMM Methodology for Web Publishing. Isakowitz, T., Kamis, A. y Koufaris, M. Working Paper IS-98-18, Center for Research on Information System. En: http://rmm-java.stern.nyu.edu/rmm/

Jacobson *et al*. (2001). El Proceso Unificado de Desarrollo de Software. Jacobson, I., Booch, G. y Rumbaugh J. Addison Wesley, 2001.

Kim *et al*. (1989). Composite Object Revisted. Won Kim. Proc. of the SIGMOD Int. Conf. on the Management of Data, pp. 337-347, 1989.

Koch *et al*. (2000). Extending UML to Model Navigation and Presentation in Web Applications. Koch, N., Baumeister, H. y Mandel L. En Modeling Web Applications, Workshop of the UML´2000. Ed. Geri Winters y Jason Winters, York, England, 2000.

Kraus y Koch (2002). The expressive Power of UML-based Web Engineering. Kraus, A. y Koch, N. En Second International Workshop on Web-oriented Software Technology (IWWOST02). CYTED. Junio 2002. Ed. D. Schwabe, O. Pastor, G. Rossi y L. Olsina, 2002.

Kovács y Van Bommel (1998). Conceptual modeling-based design of object-oriented databases. Kovács, C. y Van Bommel, P. Information and Software Technology, Volumen 40, Nº 1, pp. 1-14, 1998.

Leavit (2000). Whatever Happened to Object-Oriented Databases? Leavit, N. Computer, pp. 16-19, Agosto 2000.

Lowe y Hall (1999). Hipermedia & the Web. An Engineering Approach. Ed. J. Wiley y Sons, 1999.

Marcos (2003). Investigación en Ingeniería del Software vs. Desarrollo Software. Marcos. E. Métodos de Investigación y Fundamentos Filosóficos e Ingeniería del Software y Sistemas de Información. Ed: E. Marcos. Dykinson, S. L., pp. 136-149, 2003.

Marcos y Cáceres (2001). Object Oriented Database Design. Marcos, E. y Cáceres, P. En: Developing Quality Complex Database Systems: Practices, Techniques, and Technologies. Ed. Shirley Becker. Idea Group.

Marcos *et al.* (2001a). Aggregation and Composition in Object-Relational Database Design. Marcos, E., Vela, B. y Cavero, J. M. Fifth East European Conference on Advances in Databases and Information Systems, ADBIS'2001, pp. 195-209. Vilnius, Lithuania, 2001.

Marcos *et al.* (2001b). Extending UML for Object-Relational Database Design. Marcos, E., Vela, B. y Cavero, J. M. UML 2001 – The Unified Modeling Language. Springer-Verlag. LNCS 2185, pp. 225-239. Toronto, Canadá, 2001.

Marcos *et al.* (2002a). Una Extensión de UML para el Diseño de Bases de Datos Objeto-Relacionales. Marcos, E., Vela, B. y Cavero, J.M. CUORE (Círculo de Usuarios Oracle de España). Vivat Academia. Febrero 2002, Nº 19, pp. 27-39, 2002.

Marcos *et al.* (2002b). MIDAS/DB: a Methodological Framework for Web Database Design. Marcos, E., Cáceres, P., Vela, B. y Cavero, J. M. Conceptual Modeling for New Information Systems Techonologies. Ed.: H. Arisawa, Y. Kambayashi,

V. Kumar, H. C. Mayr e I. Hunt. Springer-Verlag. LNCS-2465, pp. 227-238. Yokohama, Japón, 2002.

Marcos *et al.* (2003). Methodological Approach for Object-Relational Database Design using UML. Marcos, E., Vela, B. y Cavero, J. M. Journal on Software and Systems Modeling (SoSyM). Ed.: R. France y B. Rumpe. Springer-Verlag. Volumen SoSyM 2, pp. 59-72. Heidelberg, Alemania, 2003.

Marcos y Marcos (1998). An Aristotelian Approach to the Methodological Research: a Method for Data Models Construction. Marcos, E. y Marcos, A. En: Information Systems - The Next Generation. Ed. L. Brooks y C. Kimble. McGraw-Hill, pp. 532-543, 1998.

Marcos y Vela (2002). El proceso de creación de una base de datos Web. Marcos, E. y Vela, B. El profesional de la información. Julio-Agosto 2002, Volumen 11, Nº 4, 2002.

Mattos (1999). SQL:1999, SQL/MM and SQLJ: An Overview of the SQL Standards. Mattos, N. M. Tutorial, IBM Database Common Technology, 1999.

McTaggart (1991). Principles of Participatory Action Research. Mc Taggart, R. Adult Education Quarterly, 41(3), 1991.

Mecca *et al.* (1999). The ARANEUS guide to Web-site development. Mecca, G., Merialdo, P., Atzeni, P. y Crescenzi, V. En: http://poincare.inf.uniroma3.it/

Miller y Mukerji (2001). Model Driven Architecture. Document number: ormsc/2001-07-01. Ed. Miller, J. y Mukerji, J. En: http://www.omg.com/mda

Miller y Mukerji (2003). MDA Guide Version 1.0. Document number: omg/2003-05-01. Ed. Miller, J. y Mukerji, J. En: http://www.omg.com/mda

Muller (1999). Database Design for Smarties. Muller, R. Morgan Kaufmann.

Naiburg (2000). Database Modeling and Design Using Rational Rose 2000e. Naiburg, E. Rose Architect Volumen 2, Issue 3, pp. 48-51.

Pastor y Ramos (1995). OASIS 2.1.1: A Class -Definition Language to Model Information Systems Using an Object-Oriented Aproach. Pastor, O. y Ramos, I. Dpto. de Sistemas Informáticos y Computación, SPUPV-95.788. Universidad Politécnica de Valencia, Marzo 1995.

Pastor *et al.* (1997). OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods. Pastor O., Insfran E., Merseguer J., Romero, J. y Pelechano, V. 9th International Conference on Advanced Information Systems Engineering, (CAiSE'1997). Springer-Verlag. LNCS 1250. Barcelona, España, 1997.

OMG (2003). XML Metadata Interchange (XMI) Specification. Versión 2. Mayo 2003. Object Management Group. En: www.w3c.org

Oracle Corporation (1998). Objects and SQL in Oracle8. Oracle Technical White paper. En: Extended DataBase Technology conference (EDBT'98). Valencia, España, 1998.

Oracle Corporation (2000). Oracle8*i*. SQL Reference. Release 3 (8.1.7). En: www.oracle.com.

Oracle Corporation (2001). Simple Strategies for Complex Data: Oracle9*i* Object-Relational Technology. Oracle Technical White Paper. En: www.oracle.com

Oracle Corporation (2003). Oracle XML DB. Technical White Paper. En: www.otn.com

Overmyer (2000). What's Different about Requirements Engineering for Web Sites? Overmyer, S. P. Requirements Engineering, 5, pp. 62-65, Springer-Verlag, 2000.

Retschitzegger y Schwinger (2000). Towards Modeling of Data Web Applications - A Requirement's Perspective. Retschitzegger, W. y Schwinger, W. En Proceedings of the America's Conference on Information Systems. Volumen I, pp. 149-155, 2000.

Rodríguez *et al.* (2002). Moving Web Services Dependencies at the Front-end. Rodríguez, J. J., Díaz, O. e Ibáñez, F. Engineering Information Systems in the Internet Context 2002, pp. 221-237, 2002.

Routledge *et al.* (2002). UML and XML Schema. Routledge, N., Bird, L. y Goodchild, A. ACM International Conference Proceeding Series. Proceedings of the thirteenth Australian conference on Database technologies, Volumen 5, pp. 157-166, 2002.

Sánchez y Delgado (2002). XML: el ASCII del siglo XXI. Sánchez Fernández, L. y Delgado Kloos, C. Nóvatica. Volumen 158, Julio-Agosto 2002, pp.8-12, 2002.

Schwabe y Rossi (1995). The Object-Oriented Hypermedia Design Model. Schwabe, D. y Rossi, G. Communications ACM, Agosto 1995, 58(8), pp. 45-46, 1995.

Schwabe y Rossi (1998). An object-oriented approach to Web-based applications design. Schwabe, D. y Rossi, G. Theory and practice of object system, 4(4), pp. 207-225, 1998.

Silva y Carlson (1995). "MOODD, a method for object-oriented database design." Data & Knowledge Engineering, Volumen 17, pp. 159-181, 1995.

Sommerville (2001). Ingeniería de Software. Sommerville, I. 6ª Edición. Addison Wesley, 2001.

Stonebraker y Brown (1999). Object-Relational DBMSs. Traking the Next Great Wave. Stonebraker, M. y Brown, P. Morgan Kauffman, 1999.

Turk *et al.* (2002). Model-Driven Approaches to Software Development. Turk, D., France, R., Rumpe, B. y Georg G. In Advances in Object-Oriented Information Systems. Springer-Verlag. LNCS-2426. Montpellier, Francia, 2002.

Ullman y Widom (1997). A First Course in Database Systems. Prentice-Hall.

Vela *et al.* (2001). Diseño de Bases de Datos Objeto-Relacionales con UML. Vela, B., Cavero, J. M. y Marcos, E. Anales de las Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC 2001. Ed. Silvia Teresita Acuña y Cecilia María Lasserre. Editorial Universidad Nacional del Jujuy, Julio 2001, pp. 59-68. Buenos Aires, Argentina, 2001.

Vela y Marcos (2003a). Una extensión de UML para representar XML Schemas. Vela, B. y Marcos E. 6º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software. 30 de Abril a 2 de Mayo 2003. Ed. M. Piattini, L. Cerruzzi y F. Ruíz. Memorias de Ideas 2003, pp. 109-119. Asunción, Paraguay, 2003.

Vela y Marcos (2003b). Extending UML to represent XML Schemas. Vela, B. y Marcos, E. The 15th Conference On Advanced Information Systems Engineering (CAISE '03). CAISE'03 FORUM. 16-20 June 2003. Ed: J. Eder, T. Welzer. Short Paper Proceedings. Klagenfurt/Velden, Austria, 2003

W3C (2001a). XML Linking Language (XLink) Version 1.0. W3C Recommendation. En: http://www.w3.org/TR/xlink/

W3C (2001b). XML Schema Working Group. XML Schema Parts 0-2: [Primer, Structures, Datatypes]. W3C Recommendation. En: http://www.w3.org/TR/xmlschema-0/, http://www.w3.org/TR/xmlschema-1/ y http://www.w3.org/TR/xmlschema-2/

Wadswroth (1998). What is Participatory Action Research? Action Research International. Wadswroth, Y. En: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html

Winston *et. al* (1987). A taxonomy of part-whole relation. Cognitive Science Volumen 11, pp. 417-444, 1987.

Young (2001). XML. Step by Step. Young, M. J. Second Edition. Microsoft Press. 2001.

Lugares de Internet

- **Estándares**

  http://www.w3c.org/ es la página Web del World Wide Web Consortium (W3C). En esta página pueden encontrarse noticias y todo tipo de información relativa a dicho consorcio y enlaces a las tecnologías W3C (XML, HTML, XLink, XML Schemas, etc.).

  http://www.jcc.com/sql_stnd.html#CurrentStatus es el lugar del SQL:1999. En sus páginas pueden encontrarse enlaces a los últimos avances del estándar, así como enlaces a otros grupos de trabajo relacionados con el SQL:1999.

  http://www.rational.com/uml/ y **http://www.omg.org/uml/** en estas dos páginas (la primera de la empresa RATIONAL y la segunda del grupo OMG) se encuentra la información sobre el estándar UML.

- **Productos de Sistemas Gestores de Bases de Datos**

  http://www.oracle.com/ es el lugar de Oracle. En estas páginas se puede encontrar información relativa a este producto comercial, incluyendo información sobre todos sus productos y enlaces para poder descargarse la mayoría de los productos ofertados por esta empresa.

APÉNDICE A:
Siglas

En este apéndice se incluirán las siglas utilizadas en el presente documento.

ADBIS:	Advances in DataBases and Information Systems
ADM:	Araneus Data Model
ADV:	Abstract Data View
APD:	Abstract Presentation Diagram
ASP:	Active Server Pages
B2B:	Business to Business
B2C:	Business to Consumer
BD:	Base de Datos
BDR:	Base de Datos Relacional
BDOR:	Base de Datos Objeto-Relacional
CAISE:	Conference on Advanced Information Systems Engineering
CASE:	Computer Aided Software Engineering
CAD/CAM:	Computer Aided Design and Manufacturing
CERN:	Conseil Europeen pour la Reserche Nucleare
CICYT:	Comisión Interministerial de Ciencia Y Tecnología
CUORE:	Circulo de Usuarios de ORacle España
DASWIS:	DAta Semantics in Web Information Systems
DAWIS:	Digital Archives Web Integrated System
DBTG:	DataBase Technology Group
E/R:	Entidad/Interrelación
EDAD:	Entorno para el Desarrollo e integración automática de Archivos Digitales
GIS:	Geographic Information System
HDM:	Hypertext Design Model
HTML:	HyperText Markup Language
ICEIS:	Internacional Conference on Enterprise Information Systems
ICM:	Informática de Comunidad de Madrid
IDC:	International Data Corporation
IEEE:	Institute of Electrical and Electronics Engineers
ISO:	International Organization for Standardization
IU:	Interfaz de Usuario
JIISIC:	Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento
JISBD:	Jornadas de Ingeniería del Software y Bases de Datos

JSP:	Java Server Pages
LNCS:	Lecture Notes in Computer Science
MDA:	Model Driven Arquitecture
MIDAS:	MetodologÍa para el Desarrollo de Aplicaciones Web
MCYT:	Ministerio de Ciencia Y Tecnología
NAD:	Navigational Access Diagram
NCM:	Navigational Conceptual Model
OASIS:	Open and Active Specification of Information Systems
OCL:	Object Constraint Language
ODMG:	Object Data Management Group
OID:	Object IDentifier
OLAP:	Online Analytical Processing
OMG:	Object Management Group
OMT:	Object Modeling Technique
OO:	Orientado a Objetos
OO-Hmethod:	Object-Oriented Hypermedia method
OOHDM:	Object-Oriented Hypermedia Design Model
OR:	Objeto Relacional
ORDB:	Object Relational DataBase
PDA:	Personal Digital Assitant
PIGE:	Proyecto de Investigación GEneral
PIM:	Platform Independent Model
PSM:	Platform Specific Model
RMDM:	Relationship Management Data Model
RMM:	Relationship Management Methodology
SGBD:	Sistema de Gestión de Bases de Datos
SGBDX:	Sistema de Gestión de Bases de Datos XML
SI:	Sistema de Información
SIW:	Sistema de Información Web
SoSyM:	Software and System Modeling
SQL:	Structured Query Language
TIC:	Tecnología de la Información y de las Comunicaciones
UML:	Unified Modeling Language
UPM:	Universidad Politécnica de Madrid
URJC:	Universidad Rey Juan Carlos
UWE:	UML - based Web Engineering
XLink:	XML Linking Language
XMI:	XML Metadata Interchange

XML:	eXtended Markup Language
XSL:	eXtensible Stylesheet Language
XSLT:	eXtensible Stylesheet Language Transformations
XP:	eXtreme Programming
W3C:	World Wide Web Consortium
W3I3:	Web-based Intelligent Information Infrastructure
WebE:	Web Engineering
WebML:	Web Modeling Language
WISDOM:	Web based Information System Development with a cOmprehensive Methodology
WML:	Wireless Markup Language
WSDL:	Web Service Definition Language
WWW:	World Wide Web

APÉNDICE B:
Extensiones de UML

Como ya se ha dicho, MIDAS propone representar todos los modelos del SIW en UML independientemente del nivel de abstracción del que se trate. Sin embargo, UML no tiene una notación específica para representar todas las técnicas necesarias de un SIW, por lo que en algunos casos ha sido necesario realizar una extensión de UML para poder utilizar UML como notación única para todo el sistema.

Conceptos Previos

UML incorpora mecanismos que permiten extender el lenguaje de una manera controlada. Estos mecanismos consisten en estereotipos, valores etiquetados y restricciones permitiendo crear nuevos bloques de construcción (Conallen, 2000).

Estereotipo: un estereotipo extiende el vocabulario de UML, permitiendo crear nuevos tipos de bloques de construcción que deriven de los existentes pero sean específicos a un problema. Este nuevo bloque de construcción tendrá sus propias características (cada estereotipo puede proporcionar su propio conjunto de valores etiquetados), semántica (cada estereotipo puede proporcionar sus propias restricciones) y notación (cada estereotipo puede proporcionar su propio icono) especiales. En su forma más sencilla, un estereotipo se representa como un nombre entre comillas tipográficas (por ejemplo, <<nombre>>) y se coloca sobre el nombre de otro elemento. Como señal visual, se puede definir un icono para el estereotipo.

Valor etiquetado: un valor etiquetado extiende las propiedades de un bloque de construcción de UML, permitiendo añadir nueva información en la especificación de ese elemento. Mientras que con los estereotipos se pueden añadir nuevos elementos a UML, con los valores etiquetados se pueden añadir nuevas propiedades. En su forma más simple, un valor etiquetado se ve como una cadena de caracteres entre llaves que se coloca debajo del nombre de otro elemento.

Restricción: Una restricción extiende la semántica de un bloque de construcción de UML, permitiendo añadir nuevas reglas o modificar las existentes. Una restricción especifica condiciones que deben cumplirse para que el modelo esté bien formado. Una restricción se representa como una cadena de caracteres entre llaves junto al elemento asociado.

Según Conallen (2000), una extensión de UML debería contener: una breve descripción, la lista y descripción de los estereotipos, valores etiquetados y restricciones, y un conjunto de reglas que permitan determinar si un modelo está

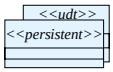
bien construido. Para cada estereotipo es necesario especificar la semántica y propiedades comunes que extienden el elemento básico que está siendo "estereotipado", definiendo un conjunto de valores etiquetados y restricciones para el estereotipo. Para poder identificar visualmente los estereotipos, pueden definirse para ellos nuevos iconos.

Extensión para BD Objeto-Relacionales

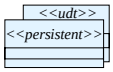
La Tabla 8 resume la propuesta de extensión de UML para el diseño de bases de datos objeto-relacionales.

Tabla 8. Extensión de UML para el diseño de bases de datos objeto-relacionales.

Descripción
Esta extensión de UML define un conjunto de estereotipos, valores etiquetados y restricciones que permiten modelar aplicaciones que trabajen con bases de datos objeto-relacionales. Las extensiones se definen para ciertos componentes que son específicos a modelos objeto-relacionales, permitiendo su representación en el mismo diagrama que describe el resto del sistema. La extensión se basa en los modelos objeto-relacionales de SQL:1999 y Oracle. Cada elemento del modelo objeto-relacional debe tener una representación en UML. Para elegir el elemento estereotipado se utiliza el siguiente criterio: Para SQL:1999 se han considerado *tipos estructurados y tablas tipadas* como clases estereotipadas porque son definidas explícitamente en el esquema en SQL. El resto de los tipos (REF, ROW y ARRAY) se consideran como atributos estereotipados. Para Oracle se han considerado *tipos de objetos, tablas tipadas* y *tablas anidadas (nested* tables) como clases estereotipadas por el mismo motivo que en el caso del SQL:1999. El tipo REF se ha considerado como un atributo estereotipado porque no puede ser definido explícitamente en el esquema en SQL. Puesto que un VARRAY puede, o no, ser definido explícitamente en el esquema, se permitirán las dos posibilidades: definirlo como un atributo estereotipado o como una clase estereotipada. Se utilizará la clase estereotipada cuando el tipo VARRAY se haya definido explícitamente en el esquema. De esta forma, el esquema UML con estereotipos para Oracle ayudará al desarrollador en el proceso de compilación (la compilación de estos esquemas en Oracle es una tarea tediosa debido a que los tipos deben ser recompilados, por lo que esta técnica puede ser de ayuda para el usuario).
Prerrequisitos para las Extensiones
Se considera que ya están definidas las extensiones requeridas para el modelo relacional (Ambler, 1999; Naiburg, 2000).

Estereotipos para SQL:1999	
Tipo Estructurado Clase del Metamodelo: Clase Descripción: Un <<*udt*>> permite la representación de nuevos tipos de datos definidos por el usuario (*User Data Types*). Icono: Ninguno Restricciones: Únicamente puede utilizarse para definir tipos valor. Valores Etiquetados: Ninguno	Tabla Tipada Clase del Metamodelo: Clase Descripción: Se define como <<*persistent*>>. Representa una clase del esquema de la base de datos, y se definirá como una tabla de un tipo de dato estructurado. Icono:  Restricciones: Una tabla tipada implica la definición de un tipo estructurado, que es el tipo de la tabla. Valores Etiquetados: Ninguno
Asociación *Compose* Clase del Metamodelo: Asociación Descripción: Una asociación <<*compose*>> es una interrelación especial que une un tipo de datos definido por el usuario <<*udt*>> o una clase <<persistent>>, con la clase que lo utiliza. Es una interrelación unidireccional. La dirección de la asociación se representa por medio de una flecha que apunta a la clase que utiliza el tipo definido por el usuario. Icono: Ninguno Restricciones: Únicamente puede utilizarse para unir una clase <<*persistent*>> con una clase <<*udt*>> *o con otra clase* <<*persistent*>>*.* Valores Etiquetados: Ninguno	Tipo *REF* Clase del Metamodelo: Atributo Descripción: Un tipo <<*ref*>> representa una referencia a una clase <<persistent>>. Icono: ●→ Restricciones: Un atributo <<*ref*>> únicamente puede hacer referencia a una clase <<*persistent*>>*.* Valores Etiquetados: La clase <<*persistent*>> a la que hace referencia.

Tipo *ARRAY* Clase del Metamodelo: Atributo Descripción: Un <<*array*>> representa un tipo colección indexado y limitado. Icono:  Restricciones: Los elementos de un <<*array*>> pueden ser de cualquier tipo excepto de tipo <<*array*>>. Valores Etiquetados: Los tipos básicos del array. El número de elementos.	Tipo *ROW* Clase del Metamodelo: Atributo Descripción: Un tipo <<*row*>> representa un atributo compuesto con un número fijo de elementos, cada uno de los cuales puede ser de distinto tipo de datos. Icono:  Restricciones: No puede tener métodos. Valores Etiquetados: El nombre de cada elemento y su tipo de datos.
Método Redefinido Clase del Metamodelo: Método Descripción: Un método <<*redef*>> es un método heredado que se implementa de nuevo en la clase hija. Icono: Ninguno Restricciones: Ninguna Valores Etiquetados: La lista de parámetros del método con sus tipos de datos. El tipo de datos devuelto por el método.	Método Diferido Clase del Metamodelo: Método Descripción: Un método <<*def*>> es un método en el que se difiere su implementación a su clase hija. Icono: Ninguno Restricciones: Se debe definir en una clase con clases hijas. Valores Etiquetados: La lista de parámetros del método con sus tipos de datos. El tipo de datos devuelto por el método.

Estereotipos para Oracle8*i*	
Tipo de Objeto Clase del Metamodelo: Clase Descripción: Un <<*udt*>> permite la representación de nuevos tipos de datos definidos por el usuario. Se corresponde con el tipo estructurado de SQL:1999. Icono: Ninguno Restricciones: Únicamente puede utilizarse para definir tipos valor. Valores Etiquetados: Ninguno	Tabla de Objeto Clase del Metamodelo: Clase Descripción: Se define como <<*persistent*>>. Representa una clase del esquema de la base de datos que deberá definirse como una tabla de un tipo de objeto. Se corresponde con las tablas tipadas de SQL:1999. Icono:  Restricciones: Una tabla tipada implica la definición de un tipo estructurado, que es el tipo de la tabla. Valores Etiquetados: Ninguno
Asociación *Compose* Clase del Metamodelo: Asociación Descripción: Una asociación <<*compose*>> es un tipo especial de interrelación que une un tipo definido por el usuario (<<*udt*>>, <<*array*>> o <<*nt*>>) o una clase, con la clase que lo utiliza. Es una interrelación unidireccional. La dirección de la asociación se representa mediante una flecha que apunta a la clase que utiliza el tipo de datos. Icono: Ninguno Restricciones: Únicamente puede utilizarse para unir una clase <<*persistent*>> con una clase <<*udt*>>, <<*array*>>, <<*nt*>> u otra clase <<persistent>>. Valores Etiquetados: Ninguno	Tipo *REF* Clase del Metamodelo: Atributo Descripción: Un <<*ref*>> representa una referencia a alguna clase <<persistent>>. Icono:  Restricciones: Un atributo <<*ref*>> únicamente puede hacer referencia a una clase <<*persistent*>>. Valores Etiquetados: La clase <<*persistent*>> a la que se refiere.

VARRAY	Nested Table
Clase del Metamodelo: Atributo/Clase Descripción: Un <<*array*>> representa un tipo colección indexado y limitado. Se corresponde con el tipo array en SQL:1999. Icono:  Restricciones: Los elementos de un <<*array*>> pueden ser de cualquier tipo de datos excepto de otro tipo colección (<<*array*>> o <<*nt*>>) Valores Etiquetados: Los tipos básicos del array. El número de elementos.	Clase del Metamodelo: Clase Descripción: Una <<nt>> representa un tipo colección no indexado ni limitado. Icono:  Restricciones: Los elementos de la <<nt>> pueden ser de cualquier otro tipo de datos excepto otro tipo colección (<<array>> o <<nt>>). Valores Etiquetados: El tipo básico de la nested table.

REGLAS QUE HA DE CUMPLIR UN DISEÑO BIEN FORMADO

Cada clase <<*udt*>>, <<*array*>> o <<*nt*>> tiene que unirse por medio de una asociación <<*compose*>> con otra clase del esquema.

Un atributo <<*ref*>> en una clase <<*persistent*>> implica una asociación con otra clase.

Una clase <<*persistent*>> que contiene un atributo colección cuyos elementos sean objetos de una clase <<*persistent*>> o <<*ref*>> a estos objetos, implica una asociación entre ambas clases.

Cada clase <<*persistent*>> corresponde en SQL:1999 a un tipo estructurado, con su correspondiente extensión. La extensión es la tabla tipada. Cada clase <<*persistent*>> en Oracle8*i* y 9*i* corresponde con un tipo de objeto con su correspondiente extensión. La extensión es la tabla de tipo de objetos. Es decir, el tipo de objeto y su extensión se representan en UML extendido como una clase <<*persistent*>>.

Comentarios

Esta extensión tiene en cuenta los modelos objeto-relacionales de SQL:1999 y Oracle8*i*. Debería modificarse para su adaptación a las nuevas versiones de ambos modelos. Por ejemplo, en el caso de Oracle9*i*, se permiten las colecciones multinivel, por lo que los elementos de cualquier tipo colección (<<array>> o <<nt>>) pueden ser cualquier otro elemento de tipo colección (<<array>> o <<nt>>). Además, si se desean utilizar otros productos, la extensión debería adaptarse a los mismos.

Estereotipos para Consultas
Consulta Clase del Metamodelo: Esquema Descripción: La extensión <<query>> se aplica al mismo tipo de elemento UML (Paquete) al que se aplica el estereotipo <<Schema>>. Esto es debido a que, tal y como se ha indicado anteriormente, se modelarán las consultas como subesquemas. Icono: Ninguno

Extensión para XML Schemas

La Tabla 9 resume la propuesta de extensión de UML para representar XML Schemas en notación gráfica.

Tabla 9. Extensión de UML para XML Schemas

Estereotipos para XML Schemas
Descripción
Esta extensión de UML define un conjunto de estereotipos, valores etiquetados y restricciones que nos permite representar un XML Schema en notación gráfica de UML. La extensión está definida para los componentes específicos de XML Schema, el actual estándar del W3C. Cada componente de un XML Schema se debe poder representar en notación gráfica con esta extensión de UML, conservando el orden y grado de anidamiento dado. A la hora de elegir los estereotipos se ha seguido el siguiente criterio: Los elementos ELEMENT se han considerado clases estereotipadas porque están explícitamente definidas en el XML Schema. Los atributos de un ELEMENT se han considerado atributos estereotipados de las clases que representan al ELEMENT. Los tipos complexType se han considerado clases estereotipadas si tienen nombre. En este caso, el tipo complexType estará relacionado con el tipo ELEMENT o el tipo que lo usa mediante una asociación <<*uses*>>. Si no tiene nombre, se representará de forma implícita mediante el *compositor* que compone. Los tipos simpleType se han considerado clases estereotipadas que en el caso de ser anónimos tendrán el mismo nombre que el elemento que los contiene, o bien con su propio nombre. Estará relacionado con su padre (ELEMENT) mediante una composición estereotipada con <<simpleType>>. Los tipos complexContent se han considerado como clases estereotipadas que deben estar relacionadas con una relación de herencia con el tipo padre, que debe ser un complexType, que se está redefiniendo mediante el tipo complexContent. Los tipos simpleContent se han considerado clases estereotipadas que están relacionadas mediante una relación de herencia con el tipo padre, que será un simpleType o un complexType, que será redefinido por el tipo simpleContent. Los compositores se consideran composiciones (tipo especial de asociación) estereotipadas. Sus estereotipos dependerán del tipo de *compositor*: <<Choice>>, <<Sequence>> o <<All>>. Para cada elemento, tipo y atributo se debe especificar al lado del nombre del elemento, tipo o atributo el número de orden, incluyendo como prefijo el número de orden de elemento o tipo padre del cual dependen.

Tipo ELEMENT Clase del metamodelo: Clase Descripción: Un tipo <<***ELEMENT***>> representa un elemento del XML Schema. Icono:  Restricciones: Sólo se puede usar para definir tipos <<ELEMENT>>. Valores Etiquetados: El nombre del elemento, el tipo base, el número mínimo y máximo de ocurrencias. Número de orden con un prefijo que será el número de orden del elemento al que pertenece.	Atributo Clase del metamodelo: Atributo Descripción: Un atributo pertenece a un tipo <<ELEMENT>>. Icono: Ninguno Restricciones: Un atributo sólo puede pertenecer a una única clase <<ELEMENT>>. Valores Etiquetados: El nombre del atributo, el tipo base, la restricción que debe satisfacer el atributo (*required*, *optional*) y el valor por defecto o fijo. Otro valor etiquetado será su número de orden incluyendo como prefijo el número de orden del elemento al que pertenece el atributo.
Tipo ComplexType Clase del metamodelo: Clase Descripción: Un tipo ***<<complexType>>*** está compuesto por otros elementos u otro compositor. Icono:  Restricciones: Debe estar relacionado por medio de una asociación <<uses>> con los elementos o tipos que usan el tipo complexType. Sólo se definirá como una clase si tiene nombre, sino se definirá de forma implícita. Valores Etiquetados: Nombre	Tipo SimpleType Clase del metamodelo: Clase Descripción: Un tipo <<***simpleType***>> no tiene ni subelementos ni atributos. Icono:  Restricciones: Debe estar asociado mediante una asociación <<uses>> con el elemento o atributo que lo usa. Valores Etiquetados: Tipo base, restricciones del propio tipo base.
Tipo ComplexContent Clase del metamodelo: Clase Descripción: Un ***<<complexContent>>*** es una subclase del tipo complexType que lo define. Icono: Ninguno Restricciones: Debe estar relacionado mediante una relación de herencia con los elementos o tipos complexType que redefine el tipo complexContent.	Tipo SimpleContent Clase del metamodelo: Clase Descripción: Un <<***simpleContent***>> es una subclase del tipo complexType o simpleType. Icono: Ninguno Restricciones: Debe estar relacionado mediante una relación de herencia con el tipo que redefine el tipo simpleContent.

Valores Etiquetados: Ninguno	Valores Etiquetados: Ninguno
Asociación Compositor Clase del metamodelo: Asociación Descripción: Una asociación compositor es un tipo especial de composición estereotipada con el tipo de compositor <<***choice***>>, <<***sequence***>> o <<***all***>>, que indica los elementos que componen al superelemento (padre). Icono: Ninguno Restricciones: Sólo se puede usar para unir un <<ELEMENT>> con los <<ELEMENT>>que lo componen. Valores Etiquetados: Ninguno	Asociación *Uses* Clase del metamodelo: Asociación Descripción: Una asociación <<***uses***>> es un tipo especial de asociación unidireccional que unirá un tipo <<complexType>> con nombre con el elemento <<ELEMENT>> o tipo que lo usa. Icono: Ninguno Restricciones: Sólo se puede usar para unir <<ELEMENT>> o tipos con un complexType con nombre. Valores Etiquetados: Ninguno

Reglas que ha de cumplir un diseño bien formado

Un tipo <<ELEMENT>> puede contener un conjunto de atributos.

Cada tipo <<complexType>> *con nombre* tiene que estar relacionado con el tipo <<ELEMENT>> que lo use mediante una asociación <<uses>>.

Extensión para XLink

La Tabla 10 resume la propuesta de extensión de UML para representar XLinks en notación gráfica.

Tabla 10. Extensión de UML para XLinks

<table>
<tr><th colspan="2">Estereotipos para XLink</th></tr>
<tr><td colspan="2">Descripción
Esta extensión de UML define un conjunto de estereotipos, valores etiquetados y restricciones que nos permite representar el XML Linking Language (XLink) en notación gráfica de UML. La extensión está definida para los componentes específicos de XLink. Cada componente de un XLink se debe poder representar en notación gráfica con esta extensión de UML.
A la hora de elegir los estereotipos se ha seguido el siguiente criterio:
Los elementos (ELEMENTs) de XLink, tanto los simples como los extendidos, se han considerado clases estereotipadas, porque son objetos de primera categoría en el metamodelo de XLink.
Los atributos de los enlaces XLink se han considerado atributos estereotipados de las clases que representan a los elementos XLink. Los atributos que puede tener cada clase de un elemento XLink dependerá del tipo del elmento XLink.
El valor proporcionado en los atributos ‘from’ o ‘to’ de una asociación del tipo “arc” se tiene que corresponder con el valor del atributo ‘label’ de algún elemento XLink de tipo ‘locator’ o ‘resource’.</td></tr>
<tr><td>Elemento Simple XLink
Clase del metamodelo: Clase
Descripción: Un elemento <<Simple XLink>> representa un enlace simple. Este elemento puede tener un conjunto de atributos que se especificarán como atributos de la clase que representa el enlace simple.

Icono: Ninguno
Restricciones: Sólo se puede usar para definir enlaces simples, es decir, para especificar un enlace entre un recurso local y uno remoto.

Valores Etiquetados: El nombre del</td><td>Elemento Extended XLink
Clase del metamodelo: Clase
Descripción: Un elemento <<Extended XLink>> representa un tipo especial de elemnto que puede contener un número no defindo de recursos. Este elemento puede tener un conjunto de atributos que se especificarán como atributos de la clase que representa el enlace simple
Icono: Ninguno
Restricciones: Sólo se puede usar para definir elementos del tipo enlaces extendidos (extended links), es decir, para especificar un enlace entre varios recursos.
Valores Etiquetados: El nombre del</td></tr>
</table>

elemento del XLink simple. Atributos de la clase: Opcionales: href, role, arcrole, title, show, actuate	elemnto Extended XLink Atributos de la clase: Opcionales: role, title
Elemento XLink Remote Resource Clase del metamodelo: Clase Descripción: Un elemento << XLink Resource>> representa un elemento de recurso remoto. Icono: Ninguno Restricciones: Sólo se puede usar para definir recursos remotos, es decir, un elemento con el valor “resource” en el atributo “type”. El nombre de la clase que representa el recurso remoto será el valor del valor del atributo “label”. Valores Etiquetados: Ninguno Atributos de la clase: Opcionales: role, title, label	Elemento XLink Local Resource Clase del metamodelo: Clase Descripción: Un elemento << XLink Locator>> representa un elemento de recurso local. Icono: Ninguno Restricciones: Sólo se puede usar para definir recursos locales, es decir, un elemento con el valor “locator” en el atributo “type”. El nombre de la clase que representa el recurso local será el valor del atributo “label”. Valores Etiquetados: Ninguno Atributos de la clase: Requeridos: href Opcionales: role, title, label
Composición *Is_Composed* Clase del metamodelo: composición Descripción: Una composición <<is_composed>> es un tipo especial de asociación que enlaza un XLink extendido con los recursos que componen el XLink extendido. Icono: ◆ Restricciones: Sólo se puede usar para asociar el enlace extendido con los recursos que componen el enlace extendido. Valores Etiquetados: Ninguno	Asociación *Arc* Clase del metamodelo: Asociación Descripción: Una asociación <<arc>> es un tipo especial de asociación unidireccional que enlaza dos recursos. Esta asociación se representa con una flecha que parte del recurso que representa el FROM (desde) y termina en el recurso que representa el TO. Ambos recursos pueden ser locales o remotos. Icono: Ninguno Restricciones: Sólo se puede usar para representar asociaciones entre dos recursos existentes. Valores Estiquetados: Los atributos show {S}, actuate {A}, arcrole y/o title (ambos directamente después del estereotipo <<arc>>). Atributos de la clase: Opcionales: arcrole, title, show, actuate,

	from, to.

Reglas que ha de cumplir un diseño bien formado

Las clases estereotipadas sólo pueden tener los atributos que se especifican para cada uno de los elementos.

APÉNDICE C:
Casos de Estudio y de Prueba

En este apéndice se muestran los distintos casos de estudio y de prueba que se han utilizado para validar y refinar la metodología propuesta en este trabajo: MIDAS/DB.

CASO DE ESTUDIO 1: Reserva de PCs y Aulas Informáticas

Este caso de estudio fue el primero que se utilizó para definir la metodología MIDAS/DB.

Se realizó el diseño de una BDOR con los siguientes requisitos: Los profesores pueden reservar aulas informáticas para impartir sus clases prácticas. Un aula se compone de un conjunto de ordenadores. Los estudiantes sólo pueden reservar ordenadores si un profesor u otro estudiante no los han reservado previamente.

La Figura 32 muestra el diagrama de clases de UML utilizado para diseñar el modelo conceptual de datos.

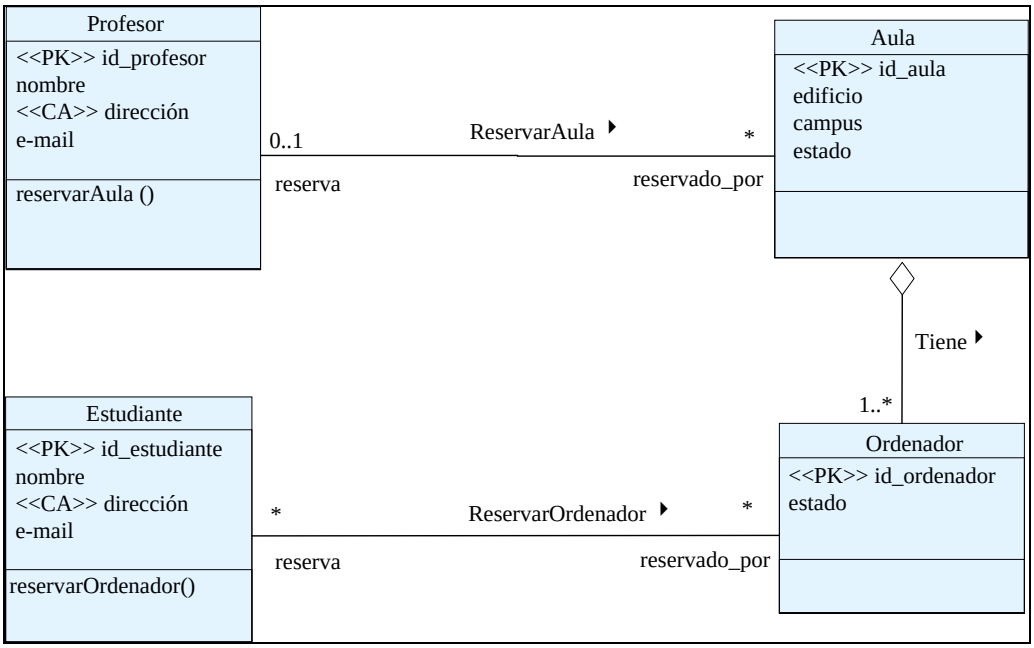


Figura 32. Diseño Conceptual en UML (CASO 1)

En la siguiente Figura 33 se muestra el diseño lógico estándar.

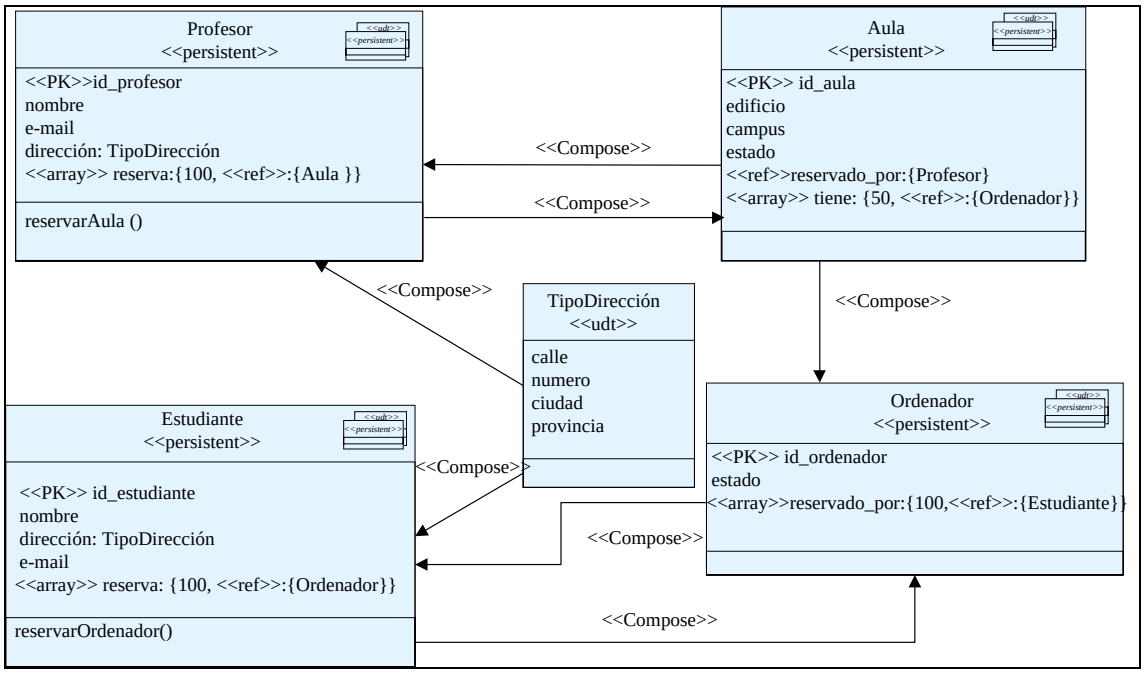


Figura 33. Diseño Lógico en SQL:1999 (CASO 1)

La Figura 34 muestra el diseño lógico específico para Oracle 8*i* en la notación gráfica de UML propuesta.

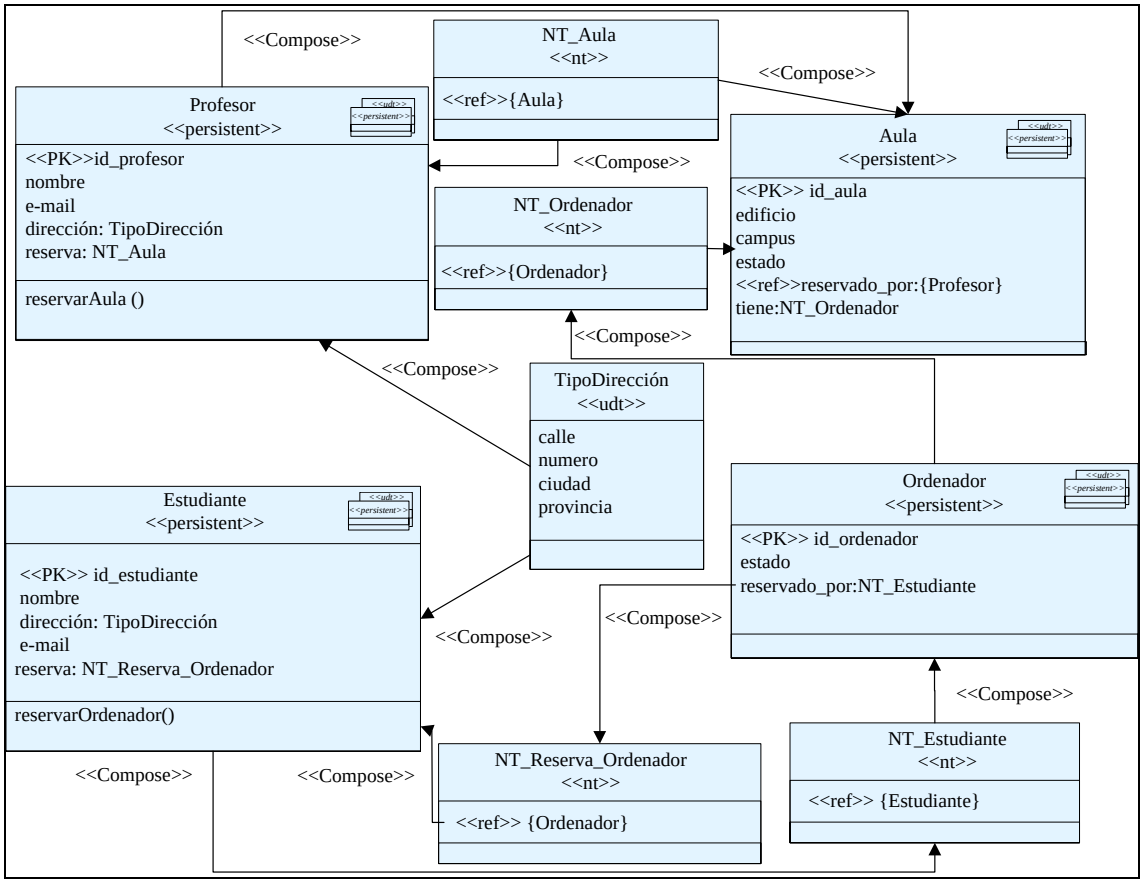


Figura 34. Diseño Lógico en Oracle8*i* (CASO1)

En la Figura 35 se muestra el código SQL correspondiente para Oracle8*i*.

```
CREATE OR REPLACE TYPE TipoDireccion AS OBJECT
( calle VARCHAR(20)
, numero NUMBER
, ciudad  VARCHAR(20)
, provincia  VARCHAR(10))
/
CREATE OR REPLACE TYPE Ordenador AS OBJECT
(id_ordenador VARCHAR(5)
, estado   VARCHAR(2)
, reservado_por NT_Estudiante)
/
CREATE OR REPLACE TYPE NT_Ordenador
AS TABLE OF REF Ordenador
/
CREATE OR REPLACE TYPE NT_Reserva_Ordenador
AS TABLE OF REF Ordenador
/
CREATE OR REPLACE TYPE Estudiante AS OBJECT
( id_estudiante VARCHAR(5)
, nombre VARCHAR(50)
, direccion TipoDireccion
, e_mail VARCHAR(100)
, reserva NT_Reserva_Ordenador)
/
CREATE OR REPLACE TYPE NT_Estudiante
AS TABLE OF REF Estudiante
/
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, direccion TipoDireccion
, reserva NT_Aula)
/
CREATE OR REPLACE TYPE Aula AS OBJECT
(id_aula VARCHAR(5)
, edificio VARCHAR(50)
, campus VARCHAR(10)
, estado VARCHAR(2)
, reservado_por REF Profesor
, tiene NT_Ordenador)
/
CREATE OR REPLACE TYPE NT_Aula
AS TABLE OF REF Aula
/

/* Recompilación*/
CREATE OR REPLACE TYPE Aula AS OBJECT
(id_aula VARCHAR(5)
, estado VARCHAR(2)
, reservado_por NT_Estudiante)
/
CREATE OR REPLACE TYPE Profesor AS OBJECT
( id_profesor VARCHAR(5)
, nombre VARCHAR(50)
, e_mail VARCHAR(100)
, direccion TipoDireccion
, reserva NT_Aula)
/

/* Creación de Tablas*/
CREATE TABLE TAula OF Aula
(PRIMARY KEY (id_aula))
NESTED TABLE tiene STORE AS tabla_ordenador_a;

CREATE TABLE TOrdenador OF Ordenador
(PRIMARY KEY (id_ordenador))
NESTED TABLE reservado_por STORE AS tabla_estudiante;

CREATE TABLE TEstudiante OF Estudiante
(PRIMARY KEY (id_estudiante))
NESTED TABLE reserva STORE AS tabla_ordenador_e;

CREATE TABLE TProfesor OF Profesor
(PRIMARY KEY (id_profesor))
NESTED TABLE reserva STORE AS tabla_aula;
```

Figura 35. Código en Oracle8*i* (CASO1)

CASO DE PRUEBA 2: Proyectos Arquitectónicos

El segundo caso de prueba se definió para validar el método de diseño OR, incluyendo la extensión UML y las guías de transformación.

Esta BD tiene los siguientes requisitos: Cada proyecto tiene un código y un nombre. De cada jefe de proyecto se desean recoger sus datos personales (código, nombre, dirección y teléfono). Un jefe de proyecto se identifica por un código. No hay dos nombres de jefe de proyecto con el mismo nombre. Un proyecto tiene un sólo jefe de proyecto y un jefe de proyecto sólo puede estar involucrado en un proyecto. En un momento determinado un jefe de proyecto puede no dirigir ningún proyecto. De los planos se desea guardar su número de identificación, la fecha de entrega, los arquitectos que trabajan en él y un dibujo del plano general con información acerca del número de figuras que contiene. Un proyecto se compone de una serie de planos, pero éstos se quieren guardar de modo independiente al

proyecto. Es decir, si en un momento dado se dejara de trabajar en un proyecto, se desea mantener la información de los planos asociados. Los planos tienen figuras. De cada figura se desea conocer, el identificador, el nombre, el color, el área y el perímetro. Además, de los polígonos se desea conocer el número de líneas que tienen, además de las líneas que lo forman. El perímetro se desea que sea un método diferido; el área se desea implementarlo como genérico para cualquier tipo de figura, pero además se desea un método específico para el cálculo del perímetro de los polígonos. De cada líneas que forma parte de un polígono se desea conocer el punto de origen y el de fin (según sus coordenadas, X e Y), así como la longitud. Cada línea tiene un identificador que permite diferenciarlo del resto. La longitud de la línea se puede calcular a partir de sus puntos origen y final.

La Figura 36 muestra el diagrama de clases de UML resultado de la tarea de modelado conceptual de datos.

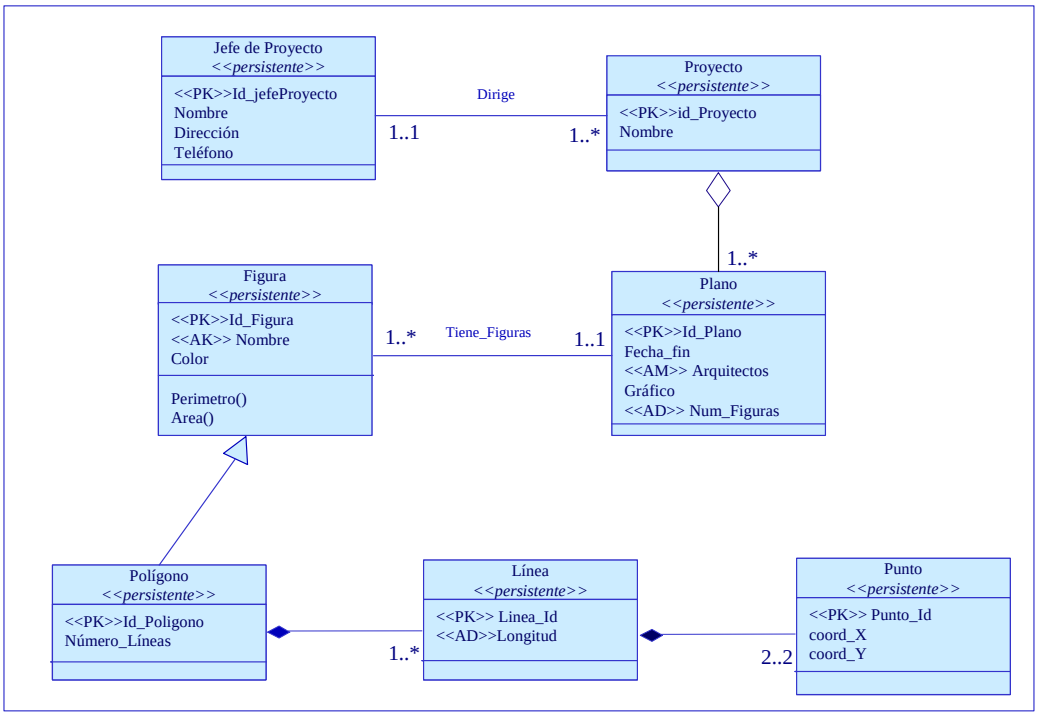


Figura 36. Diseño Conceptual en UML (CASO 2)

La Figura 37 muestra el diseño estándar con la notación gráfica propuesta.

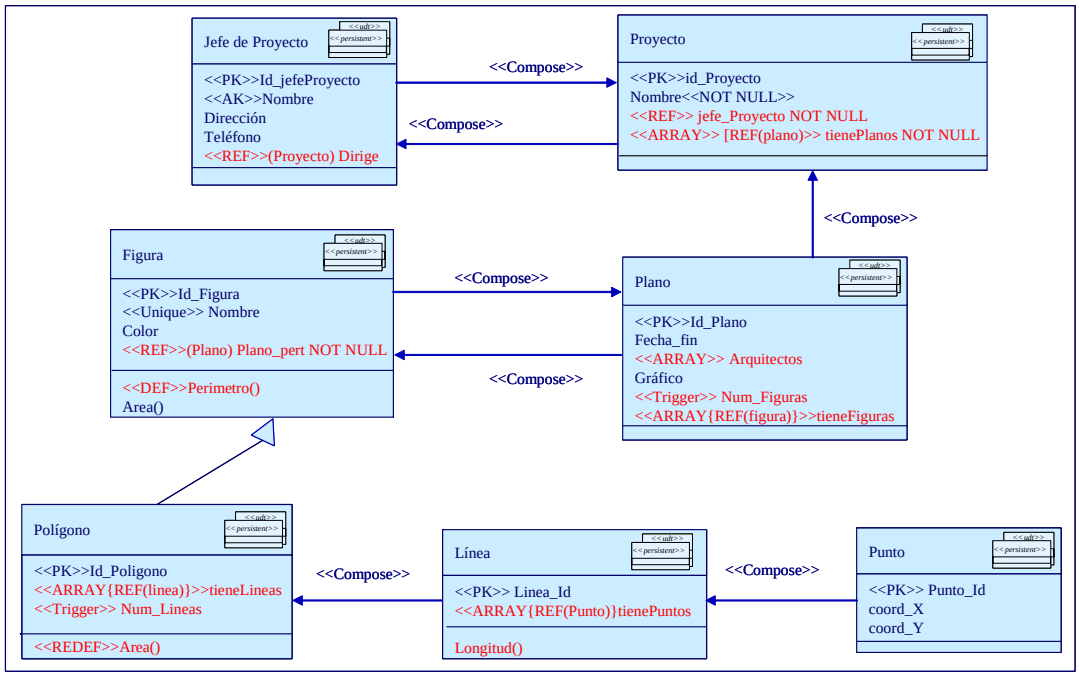


Figura 37. Diseño Lógico en SQL:1999 (CASO 2)

La Figura 38 muestra el diseño lógico específico para Oracle en la notación gráfica de UML propuesta. En esta figura vemos que los estereotipos son específicos para la nueva versión de Oracle (versión 9*i*), ya que la anterior no soporta la herencia.

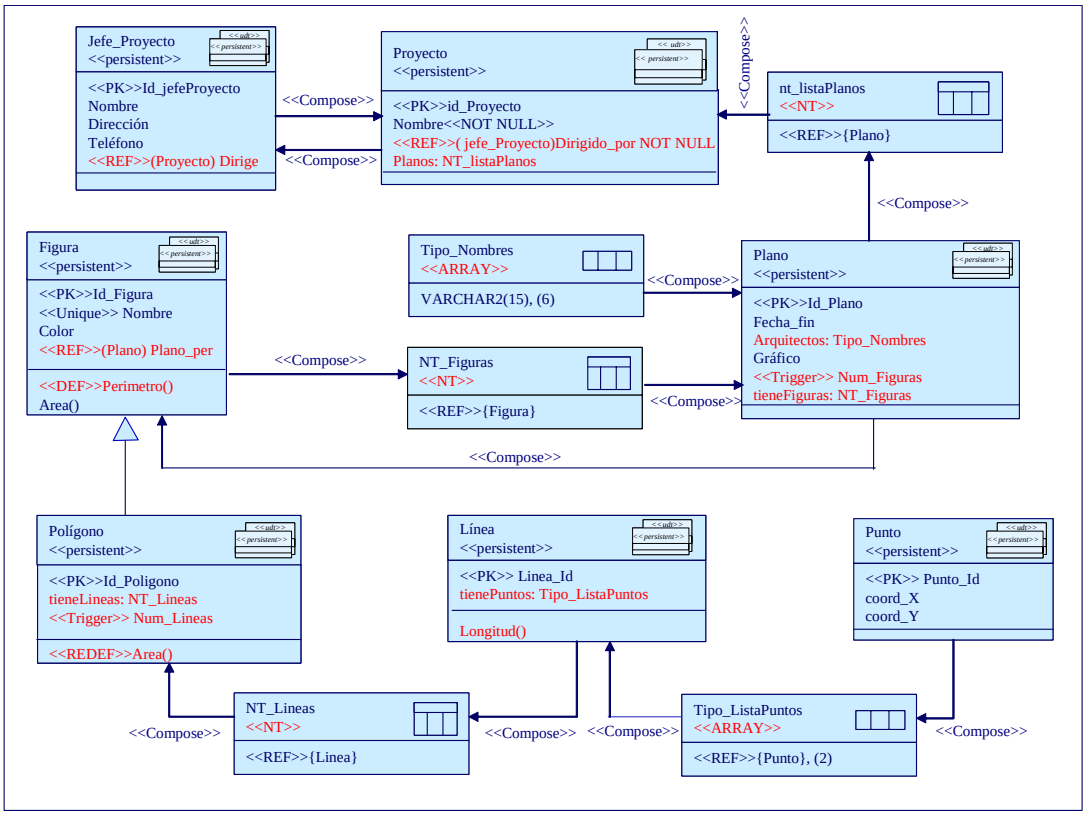


Figura 38. Diseño Lógico en Oracle9*i* (CASO 2)

```
-- Creación de tipos
CREATE OR REPLACE TYPE tipo_JefeProyecto AS OBJECT
 (id_jefeProyecto    NUMBER(4),
  Nombre             VARCHAR2(20),
  Direccion          VARCHAR2(30),
  Telefono           NUMBER(9),
  Dirige             REF tipo_Proyecto)
/
CREATE TYPE tipo_nombre
                     AS VARRAY(6) OF VARCHAR(15)
/
CREATE OR REPLACE TYPE tipo_Plano AS OBJECT
( id_Plano           NUMBER(4),
  Fecha_fin          DATE,
  Arquitectos        Tipo_Nombre,
  Grafico            BLOB,
  Num_Figuras        NUMBER(4),
  TieneFiguras       tipo_listaFiguras)
/
CREATE OR REPLACE TYPE tipo_RefPlano AS OBJECT
  (refPlano REF tipo_Plano)
/
CREATE OR REPLACE TYPE tipo_listaPlanos AS TABLE OF tipo_RefPlano
/
CREATE OR REPLACE TYPE tipo_Proyecto AS OBJECT
( id_Proyecto        NUMBER(4),
  nombre             VARCHAR2(30),
  Dirigido_Por       REF tipo_JefeProyecto,
  tienePlanos        tipo_listaPlanos)
/
CREATE OR REPLACE TYPE tipo_Figura AS OBJECT
 (id_Figura          NUMBER(4),
  Nombre             VARCHAR2(30),
  Color                                VARCHAR2(10),
  Plano_Pert         REF tipo_Plano)
NOT FINAL
/
CREATE OR REPLACE TYPE tipo_RefFigura AS OBJECT
(refFigura REF tipo_Figura)
/
CREATE OR REPLACE TYPE Tipo_ListaFiguras AS TABLE OF tipo_RefFigura
/
CREATE OR REPLACE TYPE tipo_punto AS OBJECT
(  id_Punto                          varchar(2),
   coord_X                           number(4),
   coord_Y                           number(4))
/
CREATE OR REPLACE TYPE tipo_RefPunto AS OBJECT
(  refPunto          REF tipo_punto)
/
CREATE OR REPLACE TYPE tipo_listaPuntos AS VARRAY(2) OF tipo_RefPunto
/
CREATE OR REPLACE TYPE tipo_Linea AS OBJECT
( id_Linea           NUMBER(4),
  tienePuntos        tipo_listaPuntos)
/
CREATE OR REPLACE TYPE tipo_RefLinea AS OBJECT
  (refLinea          REF tipo_linea)
/
CREATE OR REPLACE TYPE tipo_listaLineas AS TABLE OF tipo_RefLinea
/
CREATE OR REPLACE TYPE tipo_Poligono UNDER tipo_Figura
( Numero_Lineas      NUMBER(2),
  TieneLineas        tipo_listaLineas)
/
----------------- Recompilar tipo_jefeProyecto y tipo_plano ----------------------
CREATE OR REPLACE TYPE tipo_JefeProyecto AS OBJECT
 (id_jefeProyecto    NUMBER(4),
  Nombre             VARCHAR2(20),
  Direccion          VARCHAR2(30),
  Telefono           NUMBER(9),
  Dirige             REF tipo_Proyecto)
/
CREATE OR REPLACE TYPE tipo_Plano AS OBJECT
( id_Plano           NUMBER(4),
  Fecha_fin          DATE,
  Arquitectos        Tipo_Nombre,
  Grafico                              BLOB,
  Num_Figuras        NUMBER(4),
  TieneFiguras       tipo_listaFiguras)
/
------------------------------ Creación de Tablas ----------------------------------
CREATE TABLE tabla_JefeProyecto OF tipo_JefeProyecto
(id_jefeProyecto     PRIMARY KEY,
 Nombre                               UNIQUE);
CREATE TABLE tabla_Figura OF tipo_Figura
(id_Figura                            PRIMARY KEY,
 Nombre                               UNIQUE);
CREATE TABLE tabla_Plano OF tipo_Plano
(id_Plano                             PRIMARY KEY)
  NESTED TABLE tieneFiguras STORE AS nesTab_ListaFiguras;
CREATE TABLE tabla_Proyecto OF tipo_Proyecto
(id_Proyecto         PRIMARY KEY,
Nombre                                UNIQUE,
Dirigido_Por         NOT NULL)
NESTED TABLE tienePlanos STORE AS nesTab_listaPlanos;
CREATE TABLE tabla_Linea OF tipo_Linea
(PRIMARY KEY (id_Linea));
CREATE TABLE tabla_Poligono OF tipo_Poligono
(PRIMARY KEY (id_Figura))
  NESTED TABLE tieneLineas STORE AS nesTab_listaLineas;
CREATE TABLE tabla_punto OF tipo_punto
(PRIMARY KEY (id_Punto));
```

Figura 39. Código SQL para Oracle9*i* (CASO 2)

CASO DE PRUEBA 3: Libros de Recetas de Cocina

El tercer caso de prueba consiste en el diseño de una BDOR para gestionar libros de recetas de cocina, siguiendo MIDAS/DB

Esta BD tiene los siguientes requisitos: Cada receta tiene un identificador, además de un nombre y una descripción. Se debe guardar también los ingredientes de los que consta además de la cantidad necesaria para cada uno de ellos. Las recetas se publican en libros de cocina. Cada libro se identifica por un ISBN. No puede haber dos libros con el mismo título. Además, se desea conocer la fecha de edición del libro.De cada cocinero se desea conocer su nombre, su código de identificación así como su nacionalidad. No puede haber dos cocineros con el mismo nombre.Un cocinero puede escribir libros de recetas. De estos cocineros se desea conocer, además de los datos anteriormente descritos, el número de libros escritos. Un libro puede ser escrito por un máximo de cinco autores y un autor puede escribir varios libros. Un cocinero no tiene que ser necesariamente autor de libros. También hay cocineros que inventan recetas. Un cocinero puede o no ser creador

de recetas y en caso de serlo puede haber creado *varias. Cada receta corresponde a un sólo creador.Un cocinero que sea autor también puede ser creador de recetas y viceversa.*

La Figura 40 muestra el diagrama de clases de UML resultado del la tarea de modelado conceptual de datos.

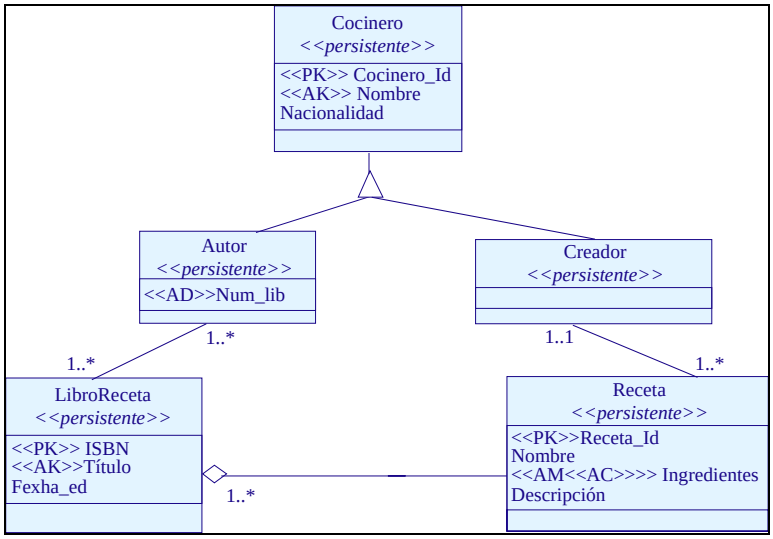


Figura 40. Diagrama Conceptual de Datos (CASO 3)

La Figura 41 muestra el diseño lógico estándar.

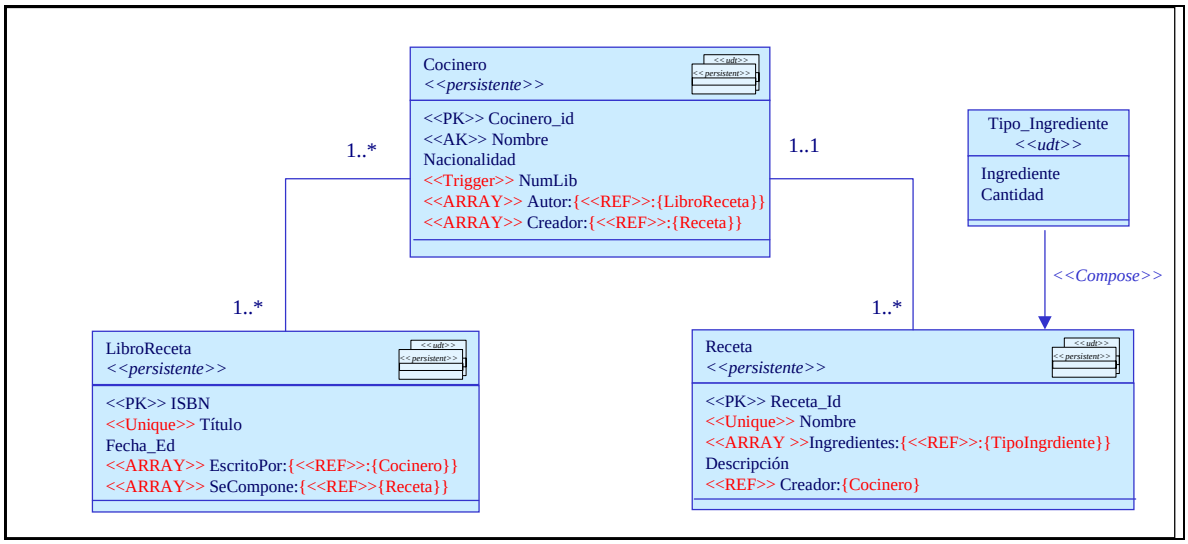


Figura 41. Diseño Lógico en SQL:1999 (CASO 3)

La Figura 42 muestra el diseño específico para Oracle 8*i*. En la Figura 43 se puede ver el código SQL correspondiente.

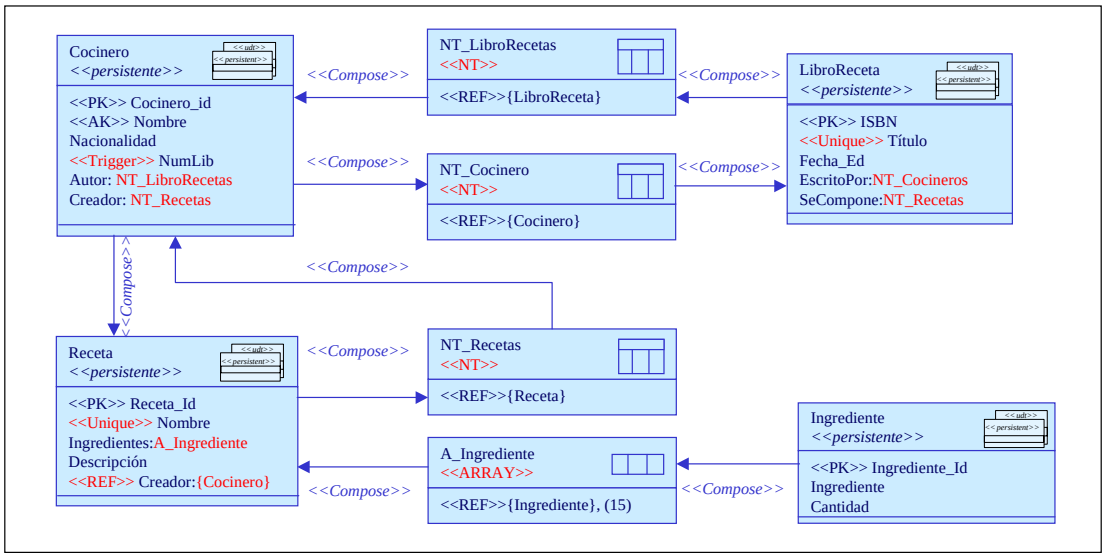


Figura 42. Diseño Lógico en Oracle 8*i* (CASO 3)

```
CREATE OR REPLACE TYPE Tipo_ingrediente AS OBJECT
(Ingrediente VARCHAR(15),
 Cantidad NUMBER)
/
CREATE OR REPLACE TYPE Tipo_lista_ingredientes
AS VARRAY(15) OF Tipo_ingrediente
/
CREATE OR REPLACE TYPE Receta AS OBJECT
(Receta_Id NUMBER,
Nombre VARCHAR(30),
Ingredientes Tipo_lista_ingredientes,
Descripcion CLOB,
CreadaPor REF Cocinero)
/
CREATE OR REPLACE TYPE Ref_Receta AS OBJECT
                (RReceta REF Receta)
/
CREATE OR REPLACE TYPE Lista_Receta
                AS TABLE OF Ref_Receta
/
CREATE OR REPLACE TYPE Cocinero AS OBJECT
(Cocinero_Id NUMBER,
Nombre VARCHAR(30),
Nacionalidad VARCHAR(15),
Num_libros NUMBER,
Autor Lista_LibroReceta,
Creador Lista_Receta)
/
CREATE OR REPLACE TYPE Ref_Cocinero AS OBJECT
                (RCocinero REF Cocinero)
/
CREATE OR REPLACE TYPE Lista_Cocinero
                AS TABLE OF Ref_Cocinero
/
CREATE OR REPLACE TYPE LibroReceta AS OBJECT
(ISBN CHAR(9),
Título VARCHAR(30),
Fecha_ed DATE,
EscritoPor Lista_Cocinero,
SeCompone Lista_Receta)
/
CREATE OR REPLACE TYPE Ref_LibroReceta AS OBJECT
                (RLibroReceta REF LibroReceta)
/
CREATE OR REPLACE TYPE Lista_LibroReceta
                AS VARRAY (5) OF Ref_LibroReceta
/
```

```
-------------------------------- Recompilar RECETA y COCINERO --------------------------------
CREATE OR REPLACE TYPE Receta AS OBJECT
(Receta_Id NUMBER,
Nombre VARCHAR(30),
Ingredientes Tipo_lista_ingredientes,
Descripcion CLOB,
CreadaPor REF Cocinero)
/
CREATE OR REPLACE TYPE Cocinero AS OBJECT
(Cocinero_Id NUMBER,
Nombre VARCHAR(30),
Nacionalidad VARCHAR(15),
Num_libros NUMBER,
Autor Lista_LibroReceta,
Creador Lista_Receta)
/
------------------------------ Creación de Tablas ---------------------------------
CREATE TABLE T_LibroReceta OF LibroReceta
(PRIMARY KEY (ISBN),
 UNIQUE (Título))
 NESTED TABLE EscritoPor STORE AS T_Escrito_Por
 NESTED TABLE SeCompone STORE AS T_Se_Compone;
CREATE TABLE T_Receta OF Receta
(PRIMARY KEY(Receta_Id),
 CreadaPor NOT NULL);
CREATE TABLE T_Cocinero OF Cocinero
(PRIMARY KEY (Cocinero_Id),
 UNIQUE (Nombre))
 NESTED TABLE Creador STORE AS T_Creador;
```

Figura 43. Código SQL para Oracle8*i* (CASO 3)

CASO DE ESTUDIO 4: Web del Grupo de Investigación KYBELE

El cuarto caso de estudio consistía en el desarrollo de un SIW para el grupo de investigación KYBELE.

Esta BD tiene los siguientes requisitos: La aplicación consiste en el desarrollo de un sitio Web con la información relativa a un grupo de investigación. Contendrá información sobre sus miembros (todos los componentes del grupo, que pueden ser profesores, colaboradores externos o alumnos colaboradores). Además recogerá la información relativa a la docencia que se imparte por los miembros de este grupo de investigación. Por lo tanto, tendrá la información sobre las asignaturas que se imparten en el grupo, tanto de doctorado como de primer y segundo ciclo, así como los proyectos Fin de Carrera que dirige alguno de los miembros del mismo. También se desea recoger información *sobre* los proyectos de investigación, publicaciones y tesis doctorales en los que participe algún miembro del grupo.

En la Figura 44 se muestra el diagrama de clases UML para este caso.

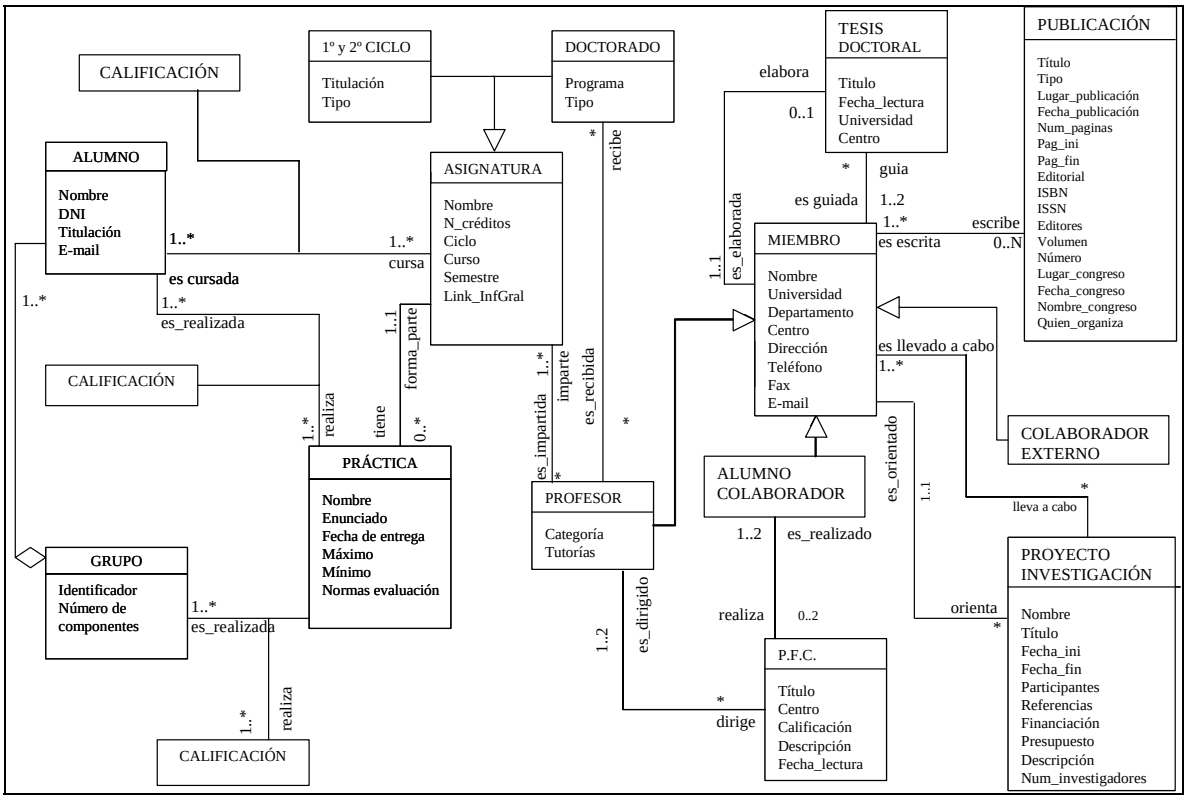


Figura 44. Diseño Conceptual en UML (CASO 4)

En la siguiente Figura 45 se muestra el diseño lógico estándar para este caso.

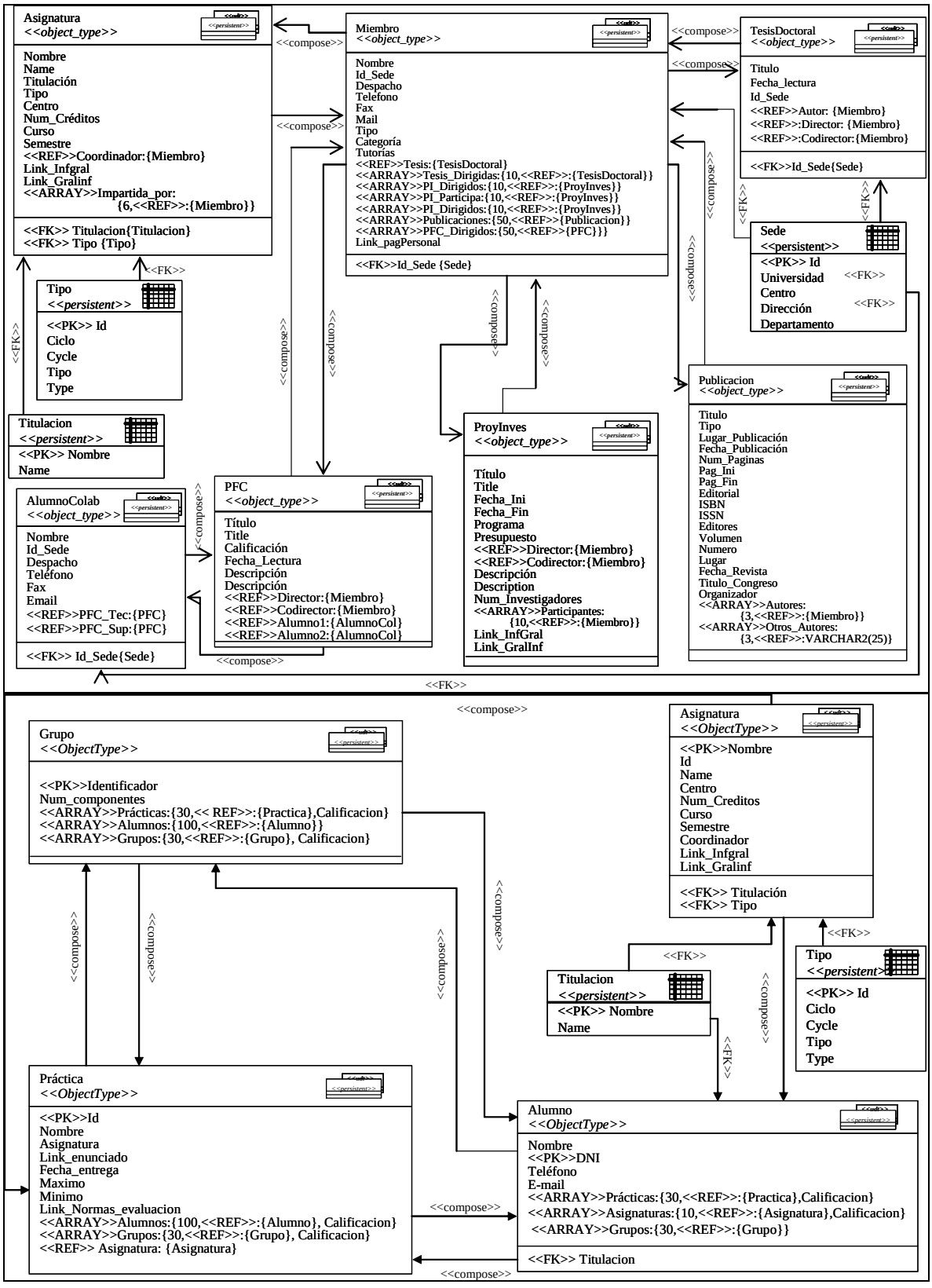


Figura 45. Diseño Lógico en SQL:1999 (CASO 4)

En la Figura 46 se muestra el diseño lógico específico para Oracle.

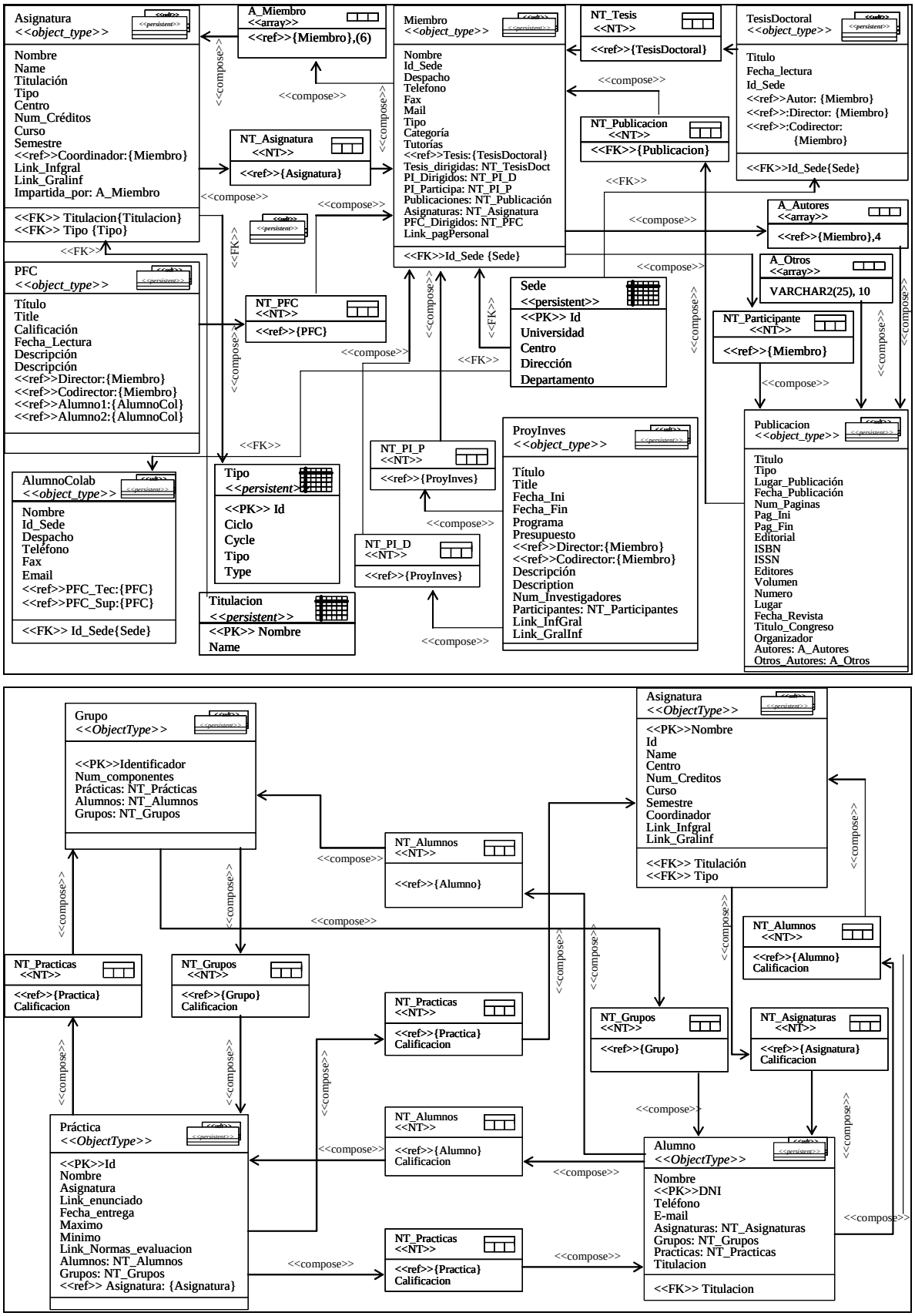


Figura 46. Diseño Lógico en Oracle 8*i* (CASO 4)

El código SQL asociado a este diseño lógico en Oracle8*i,* por su extensión se incluye en el CD que se adjunta.

CASO DE ESTUDIO 5: Ejemplos Internet

Los casos utilizados para validar y refinar la extensión de UML para representar los esquemas XML se incluyen en el CD que se adjunta a la memoria de la presente tesis doctoral.

CASO DE ESTUDIO 6: Casas Rurales

El caso de estudio 6 CASAS RURALES consiste en el desarrollo de un SIW para ofertar casas rurales en la Comunidad de Madrid. La aplicación permitirá realizar búsquedas complejas y realizar reservas de las mismas. El usuario podrá seleccionar diferentes requisitos para buscar la casa deseada, tales como la zona en la que se quiere que esté ubicada, la capacidad de la casa, un precio aproximado…y una vez hallada, podrá consultar toda la información existente sobre la casa elegida, pudiendo hacer una reserva online. Las páginas estarán codificadas en XML y deben extraer la información de una BD que no existe previamente. La especificación de requisitos completa se incluye también para este caso en el CD que se adjunta.

Este caso ha sido definido inicialmente con el fin de realizar una comparativa entre las metodología UWE y RMM, y extraer las mejores prácticas de las mismas y posteriormente, se utilizó como caso de prueba para validar la metodología MIDAS/DB.

A continuación, se mostrarán los modelos conceptuales obtenidos.

En la Figura 47 se muestra el diagrama de clases en UML.

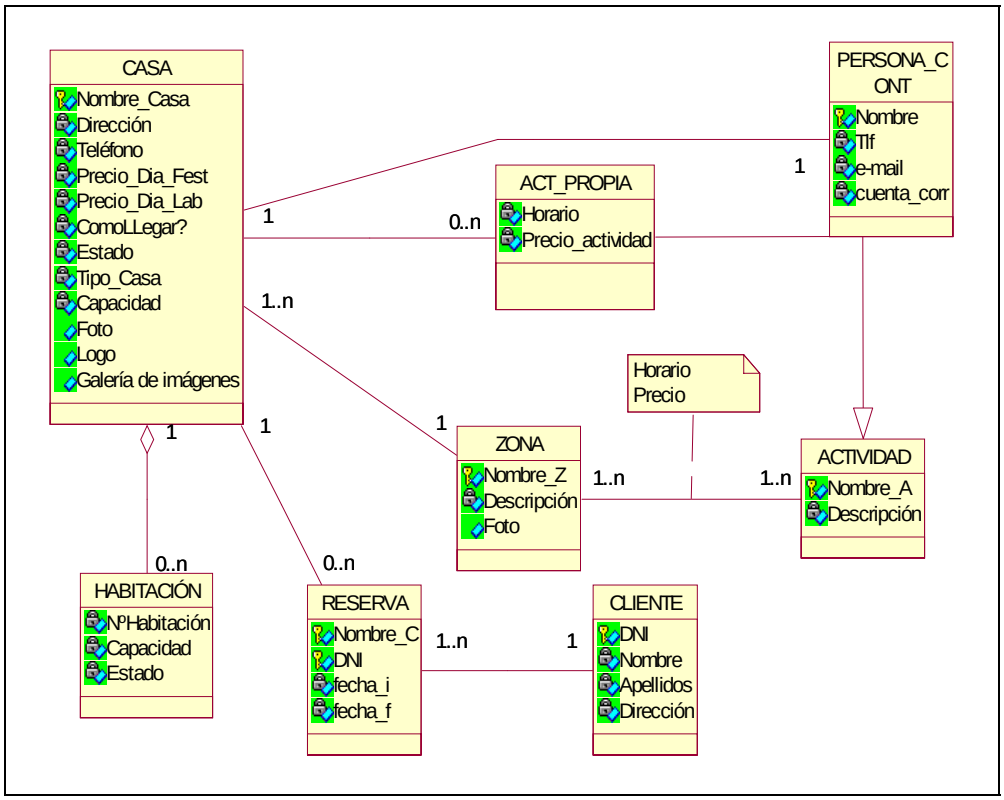


Figura 47. Diseño Conceptual en UML (CASO 6)

En la Figura 48 se muestra el modelo conceptual de fragmentos, que se representarán en la notación UML extendida basada en la propuesta de UWE.

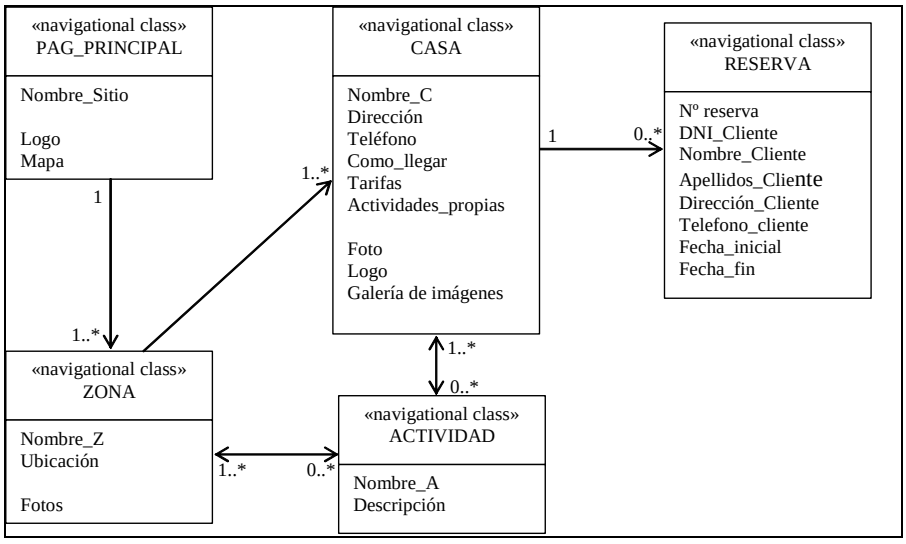


Figura 48. Modelo Conceptual de Fragmentos (CASO 6)

El modelo conceptual de navegación añade los elementos de acceso al modelo conceptual de fragmentos como se puede ver en la Figura 49.

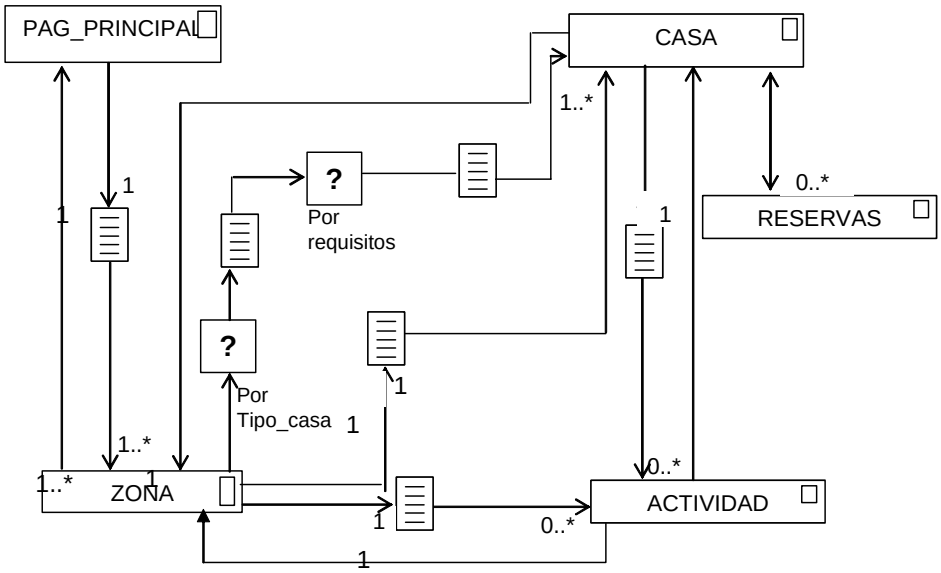


Figura 49. Modelo Conceptual de Navegación (CASO 6)

En la siguiente figura se muestra el modelo conceptual de la presentación siguiendo la notación de UWE.

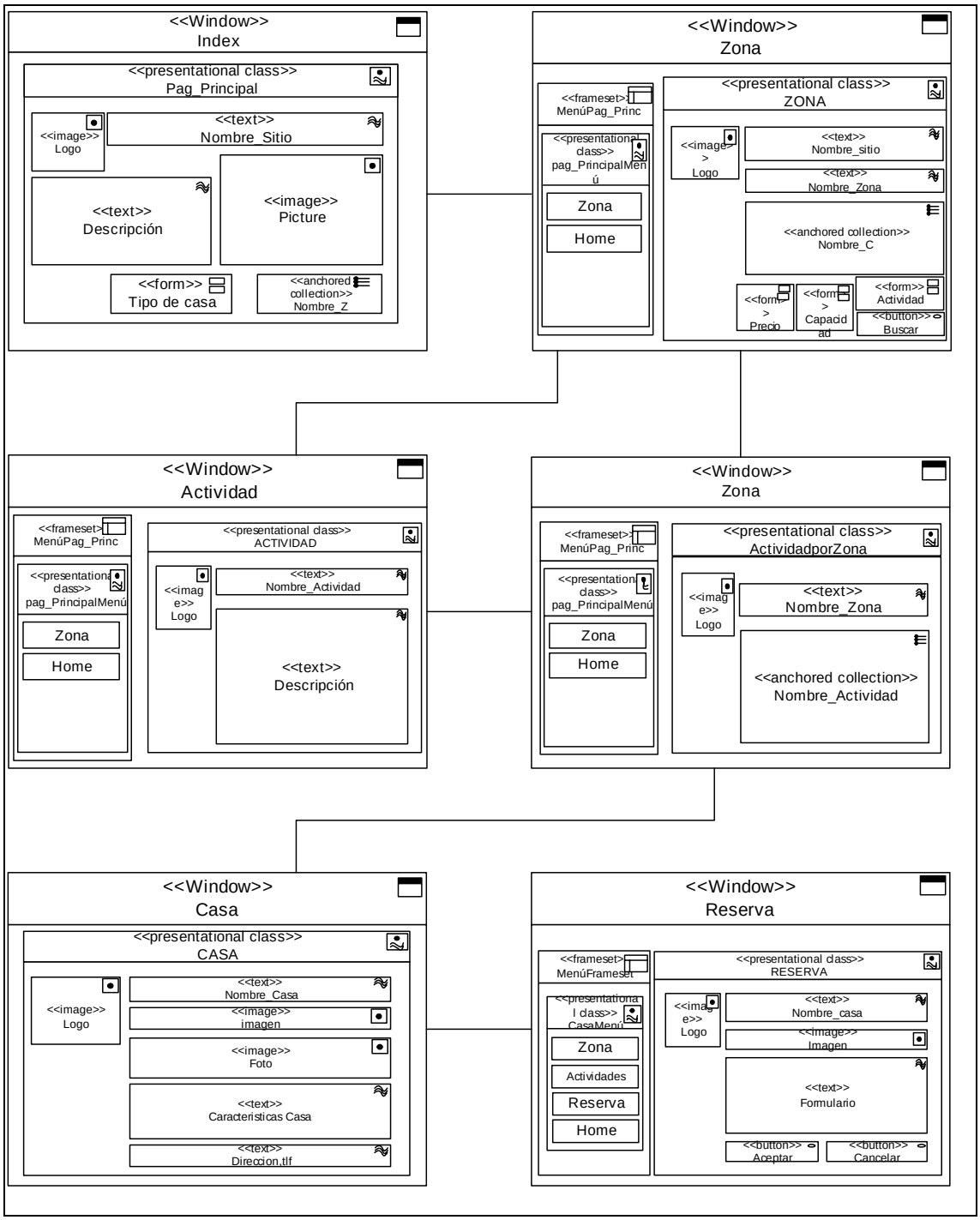


Figura 50. Modelo Conceptual de Presentación

A partir de estos modelos realizados en al actividad de análisis de la iteración MIDAS/DB se pasa a la actividad de diseño. En esta actividad por un lado se realizará el ***diseño lógico de la BD*** siguiendo las guías propuestas en la metodología y utilizando la notación UML. En la Figura 51 se muestra el diseño lógico de la BD en SQL:1999 siguiendo las guías de transformación propuestas en el apartado 3.3.1.3.

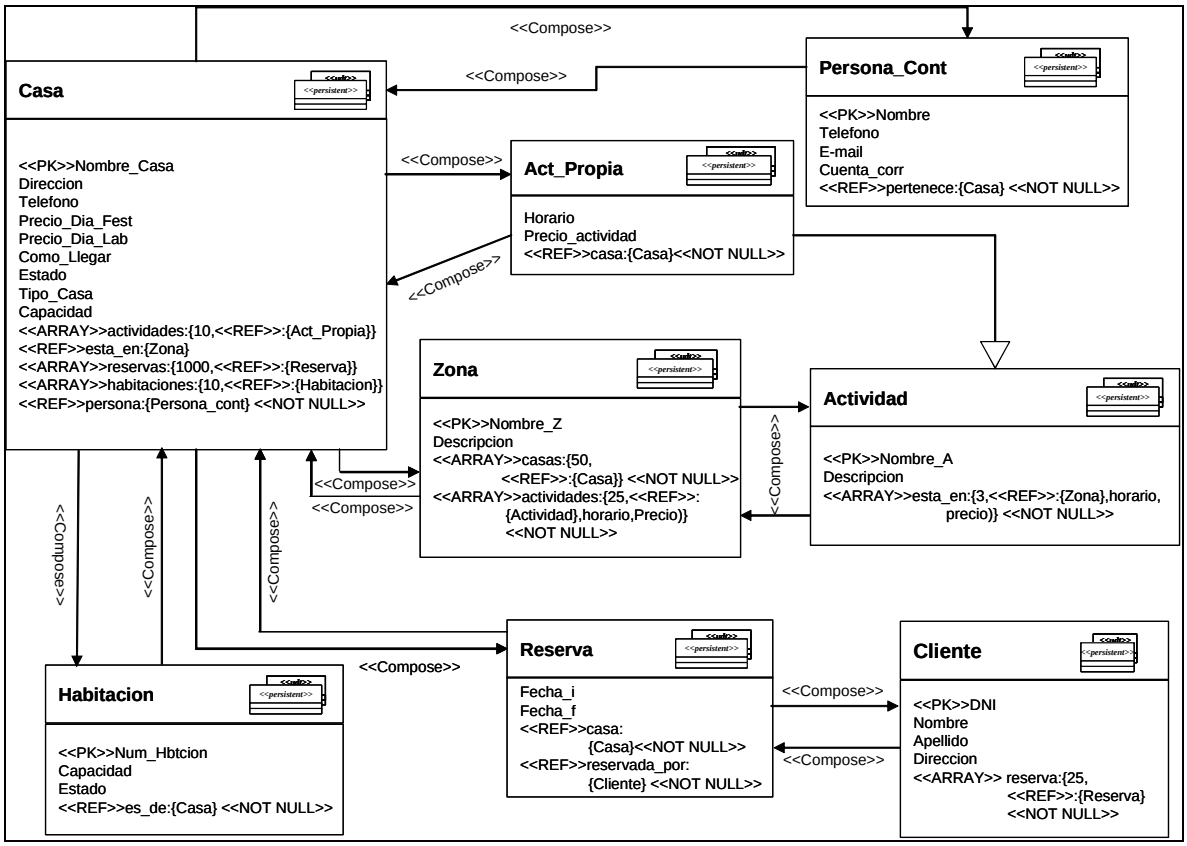


Figura 51. Diseño Lógico en SQL:1999 (CASO 6)

A partir del diseño lógico estándar se pasa a definir el diseño lógico específico en el producto Oracle9*i*, como se puede ver en la Figura 52.

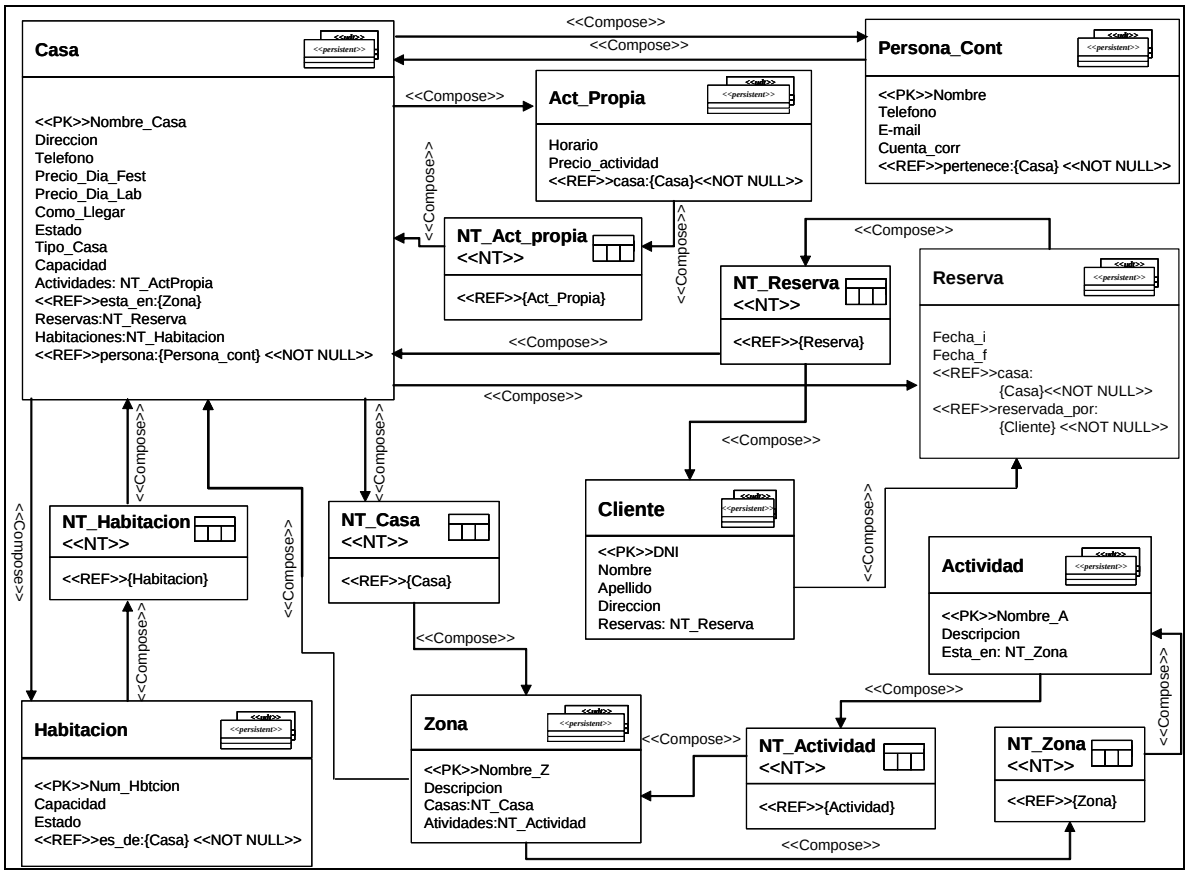


Figura 52. Diseño Lógico en Oracle9*i* (CASO 6)

A partir del diseño lógico específico se pasa a especificar el correspondiente código SQL en el producto seleccionado.

```
CREATE TYPE tipo_Casa
/
-- posteriormente se recompila el tipo

CREATE OR REPLACE TYPE tipo_ListaCasas
  AS TABLE OF tipo_Casa
/

CREATE TYPE tipo_Actividad
/
- posteriormente se recompila el tipo

CREATE OR REPLACE TYPE tipo_ListaActividades
  AS TABLE OF tipo_Actividad
/

CREATE OR REPLACE  TYPE tipo_Zona AS OBJECT (
 Nombre_z      VARCHAR2(20),
 Descripcion   VARCHAR2(255)
 Casas         tipo_ListaCasas
 Actividades   tipo_ListaActividades)
/

CREATE OR REPLACE TYPE tipo_ListaZonas
  AS TABLE OF REF tipo_Zona
/

CREATE OR REPLACE TYPE tipo_Actividad AS OBJECT (
 Nombre_Act    VARCHAR2(30),
 Descripcion   VARCHAR2(255)
 Zonas         tipo_ListaZonas)
/

CREATE OR REPLACE TYPE tipo_ListaActividades
  AS TABLE OF REF tipo_Actvidad
/

CREATE OR REPLACE tipo_ActPropia UNDER tipo_Actividad(
  Horario         VARCHAR2(25),
  Precio_Actividad NUMBER
  Casa REF        tipo_Casa)
/

CREATE OR REPLACE TYPE tipo_ListaActPropias
  AS TABLE OF REF tipo_ActPropia
/

CREATE OR REPLACE TYPE tipo_Habitacion AS OBJECT(
 Num_Hbtcion  NUMBER,
 Capacidad    NUMBER
 Estado       VARCHAR2(25),
 Es_de        REF tipo_Casa)
/

CREATE OR REPLACE TYPE tipo_ListaHabitaciones
  AS TABLE OF REF tipo_Habitacion
/

CREATE OR REPLACE TYPE tipo_Cliente
/
-- posteriormente se recompilará el tipo

CREATE OR REPLACE TYPE tipo_Reserva AS OBJECT(
 Cliente       REF tipo_Cliente VARCHAR2(8),
 Casa          REF tipo_Casa,
 fecha_e       DATE,
 fecha_s       DATE)
/

CREATE OR REPLACE TYPE tipo_ListaReserva
  AS TABLE OF REF tipo_Reserva
/
```

```
CREATE OR REPLACE TYPE tipo_Cliente AS OBJECT(
 DNI             VARCHAR2(25),
 Nombre_Cliente VARCHAR2(20),
 Apellidos       VARCHAR2(30),
 Calle           VARCHAR2(25),
 Num             NUMBER,
 Piso            NUMBER,
 Codigo          VARCHAR22(5),
 Localidad       VARCHAR2(30),
 Provincia       VARCHAR2(30),
 Telefono        VARCHAR2(9),
  Reservas       tipo_ListaReservas)
/

CREATE OR REPLACE TYPE tipo_PersonaCont AS OBJECT(
  Nombre         VARCHAR2(25),
  Telefono       VARCHAR2(9),
  E-mail         VARCHAR2(30),
  Cuenta_correo VARCHAR2(30),
  Pertenece      REF tipo_Casa)
/

CREATE OR REPLACE TYPE tipo_Casa (
 Nombre          VARCHAR2(20),
 Telefono        VARCHAR2(9),
 Calle           VARCHAR2(25),
 numero          NUMBER,
 CP              VARCHAR2(5),
 Poblacion       VARCHAR2(30),
 ComoLLegar      VARCHAR2(255),
 Precio_Dia_Fest NUMBER,
 Precio_Dia_Lab NUMBER,
 Tipo            VARCHAR2(20),
 Estado          VARCHAR2(10),
 Capacidad       NUMBER)
/

CREATE OR REPLACE TYPE tipo_ListaCasas
  AS TABLE OF REF tipo_Casa
/
```

```
CREATE TABLE Habitacion OF tipo_Habitacion
( PRIMARY KEY (Num_Hbtcion),
  Es_de          NOT NULL);

CREATE TABLE Zona OF tipo_Zona
( PRIMARY KEY (Nombre_Z))
  NESTED TABLE casas STORE AS NT_Casas,
  NESTED TABLE actividades STORE AS NT_Actividades;

CREATE TABLA Actividad OF tipo_Actividad
(  PRIMARY KEY (Nombre_A))
NESTED TABLE zonas STORE AS NT_Zonas;

CREATE TABLE Reserva OF  tipo_Reserva
(  PRIMARY KEY (cliente, casa));

CREATE TABLE Cliente OF tipo_Cliente
(  PRIMARY KEY(DNI))
NESTED TABLE reservas STORE AS NT_ReservasClte;

CREATE TABLE Persona_Cont OF tipo_Persona_Cont
(  PRIMARY KEY (Nombre)
   pertenece      NOT NULL);

CREATE TABLE ActPropia OF tipo_ActPropia
(  PRIMARY KEY (Nombre_A),
   casa           NOT NULL);
CREATE TABLE Casa OF tipo_Casa
( PRIMARY KEY (Nombre_Casa),
  persona        NOT NULL)
NESTED TABLE reservas STORE AS NT_ReservasCasa,
NESTED TABLE habitaciones STORE AS NT_Habitaciones,
NESTED TABLE NT_ActPropia STORE AS NT_ActPropia;
```

Figura 53. Código SQL en Oracle 9*i* (CASO6)

Para cada uno de los fragmentos del modelo conceptual de fragmentos se ha obtenido un esquema XML en notación UML (en el CD que se adjunta se muestran todos los esquemas obtenidos). En la se muestra el esquema XML generado a partir del fragmento CASA en notación UML, así como el correspondiente código XML.

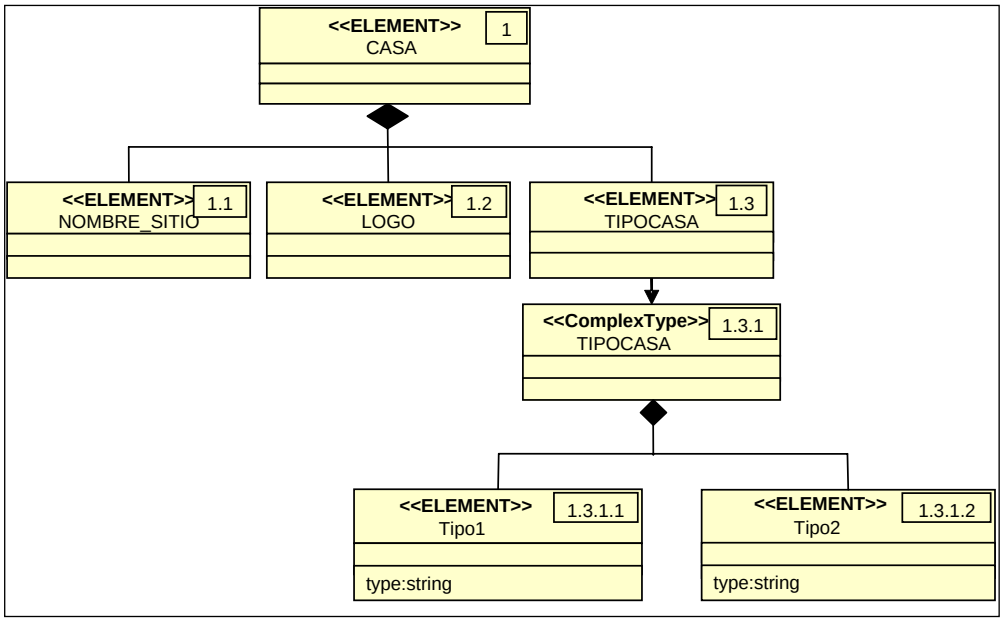


Figura 54. XML Schema para el fragmento CASA (CASO 6)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs= http://www.w3.org/2001/XMLSchema>
    <xs:element name="Buscador">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="nombresitio">
                    <xs:complexType>
                        <xs:attribute name="href" type="xs:anyURI"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="LogoCasa">
                    <xs:complexType>
                        <xs:attribute name="href" type="xs:anyURI"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="EncabezadoTipo" type="xs:string"/>
                <xs:element name="TipoCasa">
                    <xs:complexType name="TipoDeCasa">
                        <xs:sequence>
                            <xs:element name="tipo1" type="xs:string"/>
                            <xs:element name="tipo2" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        </xs:element>
    </xs:schema>
```

Figura 55. Código XML Schema para el fragmento CASA (CASO 6)

En la Figura 56 se muestra el XLink generado a partir del modelo conceptual de navegación y en la Figura 57 el código XLink correspondiente.

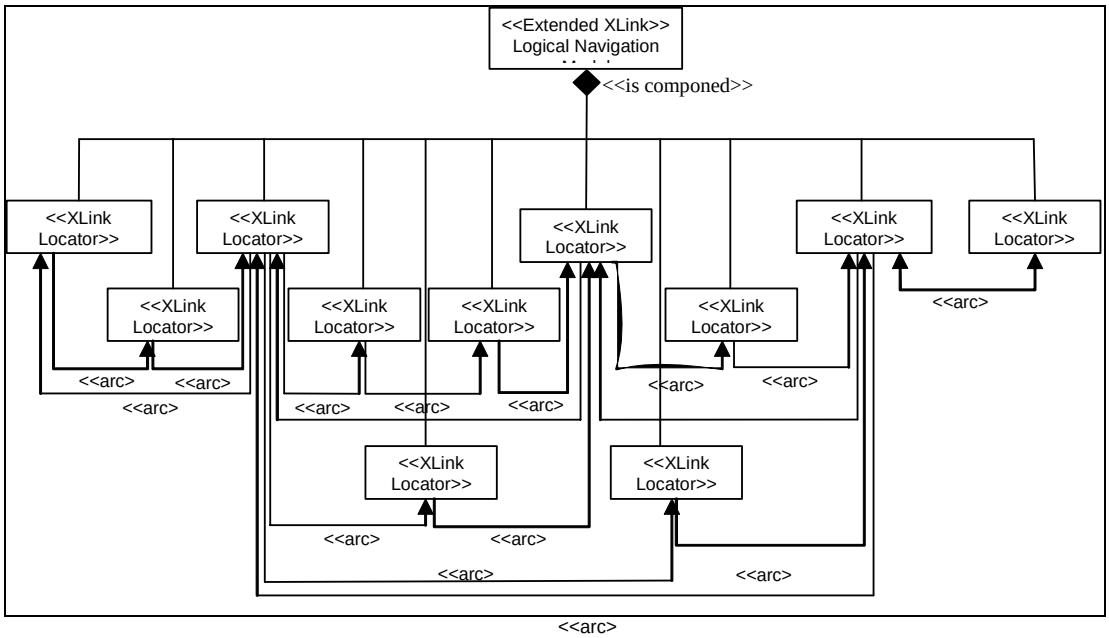


Figura 56. Modelo Lógico de Fragmentos representado XLink con UML (CASO 6)

```
<Links xmlns:xlink=http://www.w3.org/1999/xlink>
<ModeloLogicoNavegacion xlink:type="extended"
titulo="Consorcio Cines"
<!—definición de todos los recursos locales-->
<CadenaCines xlink:type "resource"
Xlink:label="sliceCadenaCines"
</CadenaCines>
<cine xlink:type "resource"
Xlink:label="sliceCine"
</cine>
<película xlink:type "resource"
Xlink:label="slicePelícula"
</película>
<sesion xlink:type "resource"
Xlink:label="sliceSesion"
</sesion>
<link1 xlink:type="arc"
xlink:from ="sliceCadenaCines"
xlink:to="sliceCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link2 xlink:type="arc"
xlink:from ="sliceCine"
xlink:to="slicePelícula"
xlink:show="replace"
xlink:actuate="onRequest" />
<link3 xlink:type="arc"
xlink:from ="slicePelicula"
xlink:to="sliceCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link4 xlink:type="arc"
xlink:from ="slicePelicula"
xlink:to="sliceSesion"
xlink:show="replace"
xlink:actuate="onRequest" />
</ModeloLogicoNavegacion>
</Links>
```

Figura 57. Código XLink correspondiente (CASO 6)

CASO DE PRUEBA 7: Cine-Entradas

Para este caso de prueba, descrito previamente en el capítulo 4, se incluye en esta sección el diseño lógico específico para el producto Oracle 9*i* (Figura 58) y el correspondiente código SQL (Figura 59).

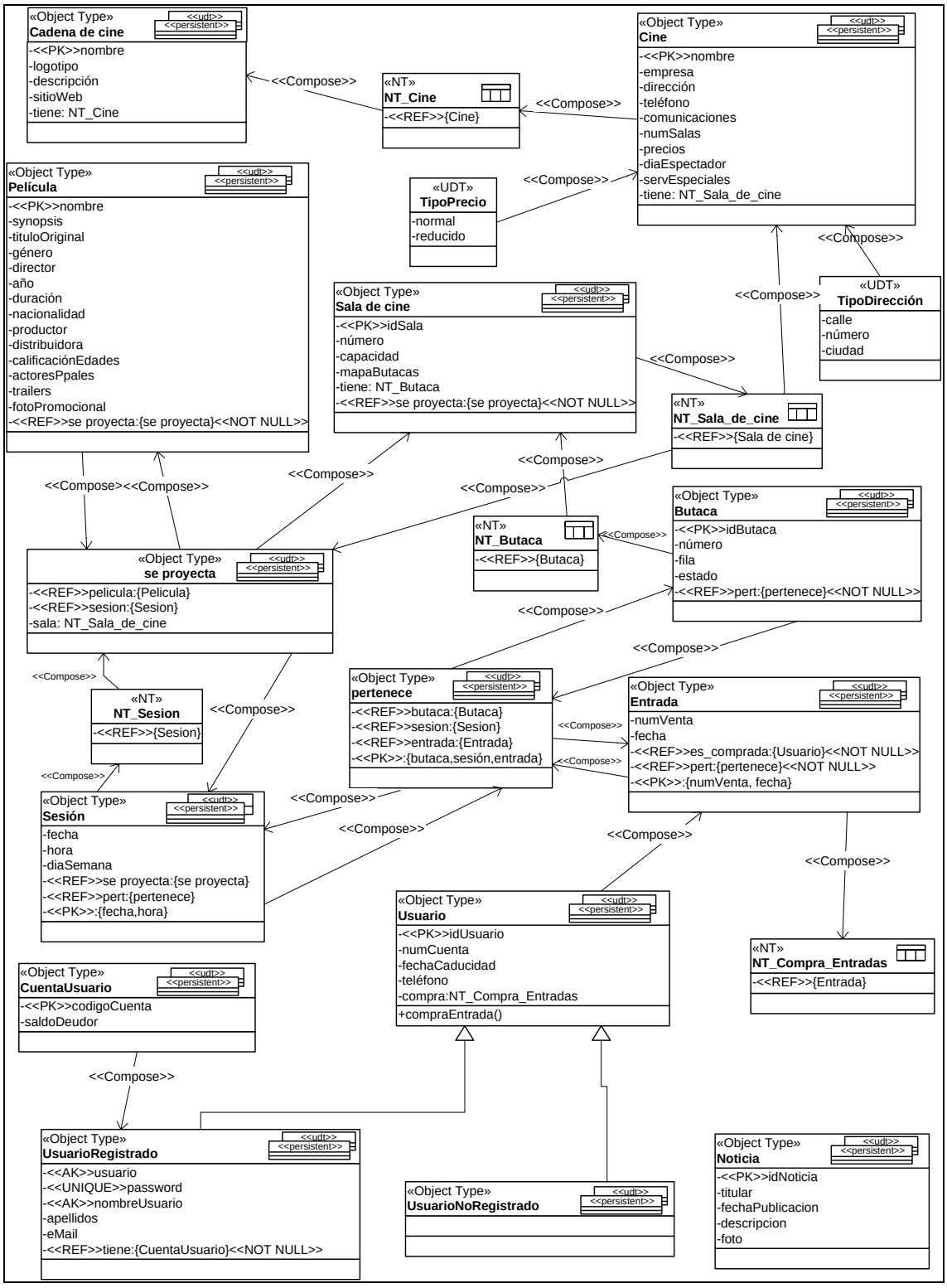


Figura 58. Diseño Lógico en Oracle 9*i* (CASO 7)

```
CREATE OR REPLACE TYPE TipoPrecio AS OBJECT
(normal NUMBER,
reducido NUMBER)
/
CREATE OR REPLACE TYPE TipoDireccion AS OBJECT
(calle VARCHAR(30),
numero NUMBER,
ciudad VARCHAR(20))
/
CREATE OR REPLACE TYPE Cine AS OBJECT
(nombre VARCHAR(30),
empresa VARCHAR(30),
direccion TipoDireccion,
telefono NUMBER,
comunicaciones VARCHAR2(200),
numSalas NUMBER,
precios TipoPrecio,
diaEspectador VARCHAR(9),
servEspeciales VARCHAR2(300),
tiene NT_Sala_de_cine)
/
CREATE OR REPLACE TYPE T_Ref_Cine AS OBJECT
(Ref_Cine REF Cine)
/
CREATE OR REPLACE TYPE NT_Cine AS TABLE OF T_Ref_Cine
/
CREATE OR REPLACE TYPE Cadena_de_cine AS OBJECT
(nombre VARCHAR(15),
logotipo BLOB,
descripcion VARCHAR2(4000),
sitioWeb VARCHAR(30),
tiene NT_Cine)
/
CREATE OR REPLACE TYPE Sala_de_cine AS OBJECT
(idSala NUMBER,
numero NUMBER,
capacidad NUMBER,
mapaButacas BLOB,
tiene NT_Butaca,
seproyecta REF Se_proyecta)
/
CREATE OR REPLACE TYPE T_Ref_Sala_de_cine AS OBJECT
(Ref_Sala_de_cine REF Sala_de_cine)
/
CREATE OR REPLACE TYPE NT_Sala_de_cine AS TABLE
        OF T_Ref_Sala_de_cine
/
CREATE OR REPLACE TYPE Butaca AS OBJECT
(idButaca NUMBER,
numero NUMBER,
fila NUMBER,
estado VARCHAR(10),
pert REF Pertenece)
/
CREATE OR REPLACE TYPE T_Ref_Butaca AS OBJECT
(Ref_Butaca REF Butaca)
/
CREATE OR REPLACE TYPE NT_Butaca AS TABLE
OF T_Ref_Butaca
/
CREATE OR REPLACE TYPE Se_proyecta AS OBJECT
(apelicula REF Pelicula,
asesion NT_Sesion,
asala NT_Sala_de_cine)
/
CREATE OR REPLACE TYPE Pelicula AS OBJECT
(Nombre VARCHAR(50),
synopsis VARCHAR2(2000),
tituloOriginal VARCHAR(50),
genero VARCHAR(15),
director VARCHAR(30),
anyo NUMBER,
duracion NUMBER,
nacionalidad VARCHAR(20),
productor VARCHAR(30),
distribuidora VARCHAR(20),
calificacionEdades VARCHAR(20),
actoresPpales VARCHAR(500),
trailers BLOB,
fotoPromocional BLOB,
seproyecta REF Se_proyecta)
/
CREATE OR REPLACE TYPE Sesion AS OBJECT
(fecha DATE,
hora NUMBER,
diaSemana VARCHAR(9),
seproyecta REF Se_proyecta,
pert REF Pertenece)
/
CREATE OR REPLACE TYPE T_Ref_Sesion AS OBJECT
(Ref_Sesion REF Sesion)
/
CREATE OR REPLACE TYPE NT_Sesion AS TABLE OF T_Ref_Sesion
/
CREATE OR REPLACE TYPE Entrada AS OBJECT
(numVenta NUMBER,
fecha DATE,
es_comprada1 REF UsuarioRegistrado,
es_comprada2 REF UsuarioNoRegistrado,
pert REF Pertenece)
/
CREATE OR REPLACE TYPE Pertenece AS OBJECT
(abutaca REF Butaca,
asesion REF Sesion,
aentrada REF Entrada)
/
CREATE OR REPLACE TYPE T_Ref_Entrada AS OBJECT
(Ref_Entrada REF Entrada)
/
CREATE OR REPLACE TYPE NT_Compra_Entradas
AS TABLE OF T_Ref_Entrada
/
CREATE OR REPLACE TYPE Usuario AS OBJECT
(idUsuario NUMBER,
numCuenta NUMBER,
fechaCaducidad DATE,
telefono NUMBER,
compra NT_Compra_Entradas)/

CREATE OR REPLACE TYPE UsuarioNoRegistrado UNDER Usuario
()
/
CREATE OR REPLACE TYPE UsuarioRegistrado UNDER Usuario
(usuario VARCHAR(15),
password VARCHAR(15),
nombreUsuario VARCHAR(20),
apellidos VARCHAR(30),
eMail VARCHAR(30),
tiene REF CuentaUsuario,
compra NT_Compra_Entradas)
/
CREATE OR REPLACE TYPE CuentaUsuario AS OBJECT
(codigoCuenta NUMBER,
saldoDeudor NUMBER,
pert REF UsuarioRegistrado)
/
```

```
CREATE TABLE Tabla_Cadena_de_cine OF Cadena_de_cine
(PRIMARY KEY(nombre))
NESTED TABLE tiene STORE AS TCine
/
CREATE TABLE Tabla_Cine OF Cine
(PRIMARY KEY(nombre))
NESTED TABLE tiene STORE AS TSala1
/
CREATE TABLE Tabla_Pelicula OF Pelicula
(PRIMARY KEY(nombre),
tituloOriginal UNIQUE)
/
CREATE TABLE Tabla_Sala_de_cine OF Sala_de_cine
(PRIMARY KEY(idSala))
NESTED TABLE tiene STORE AS TButaca
/
CREATE TABLE Tabla_Butaca OF Butaca
(PRIMARY KEY(idButaca),
CHECK(estado='Libre' OR estado='Ocupada' OR
estado='Seleccionada'))
/
CREATE TABLE Tabla_Se_proyecta OF Se_proyecta
NESTED TABLE asesion STORE AS TSesion
NESTED TABLE asala STORE AS TSala2
/
CREATE TABLE Tabla_Entrada OF Entrada
(PRIMARY KEY(numVenta,fecha),
es_comprada1 NOT NULL,
es_comprada2 NOT NULL)
/
CREATE TABLE Tabla_Pertenece OF Pertenece
(abutaca NOT NULL,
asesion NOT NULL)
/
CREATE TABLE Tabla_Sesion OF Sesion
(PRIMARY KEY(fecha,hora))
/
CREATE TABLE Tabla_CuentaUsuario OF CuentaUsuario
(PRIMARY KEY(codigoCuenta),
pert NOT NULL)
/
CREATE TABLE Tabla_UsuarioRegistrado OF UsuarioRegistrado
(PRIMARY KEY(idUsuario),
tiene NOT NULL)
NESTED TABLE compra STORE AS TCompra1
/
CREATE TABLE Tabla_UsuarioNoRegistrado OF UsuarioNoRegistrado
(PRIMARY KEY(idUsuario))
NESTED TABLE compra STORE AS TCompra2
/
```

Figura 59. Código SQL para Oracle 9*i* (CASO 7)

En la Figura 60 y en la Figura 61 se muestran respectivamente el modelo lógico de navegación completo en XLink con notación UML y el correspondiente código.

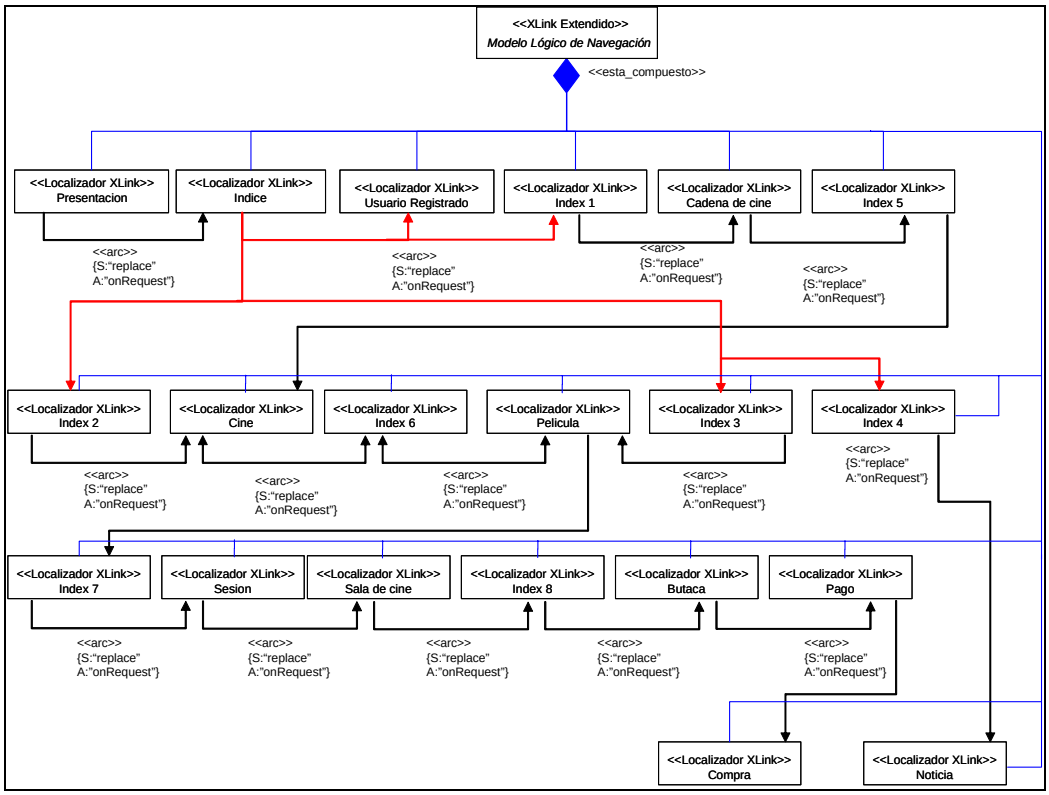


Figura 60. Modelo Lógico de Navegación representado en XLink con UML (CASO 7)

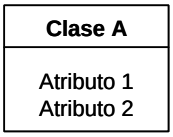
```
<Links xmlns:xlink="http://www.w3.org/1999/xlink">
<ModeloLogicoNavegacion xlink:type="extended"
titulo="Consorcio Cines">
<!—definición de todos los recursos locales-->
<Presentacion xlink:type "resource"
Xlink:label="slicePresentacion"/>
<Indice xlink:type "resource"
Xlink:label="sliceIndice"/>
<UsuarioRegistrado xlink:type "resource"
Xlink:label="sliceUsuarioRegistrado"/>
<Index1 xlink:type "resource"
Xlink:label="sliceIndex1"/>
<CadenaDeCine xlink:type "resource"
Xlink:label="sliceCadenaDeCine"/>
<Index5 xlink:type "resource"
Xlink:label="sliceIndex5"/>
<Index2 xlink:type "resource"
Xlink:label="sliceIndex2"/>
<Cine xlink:type "resource"
Xlink:label="sliceCine"/>
<Index6 xlink:type "resource"
Xlink:label="sliceIndex6"/>
<Pelicula xlink:type "resource"
Xlink:label="slicePelicula"/>
<Index3 xlink:type "resource"
Xlink:label="sliceIndex3"/>
<Index4 xlink:type "resource"
Xlink:label="sliceIndex4"/>
<Index7 xlink:type "resource"
Xlink:label="sliceIndex7"/>
<Sesion xlink:type "resource"
Xlink:label="sliceSesion"/>
<SalaDeCine xlink:type "resource"
Xlink:label="sliceSalaDeCine"/>
<Index8 xlink:type "resource"
Xlink:label="sliceIndex8"/>
<Butaca xlink:type "resource"
Xlink:label="sliceButaca"/>
<Pago xlink:type "resource"
Xlink:label="slicePago"/>
<Compra xlink:type "resource"
Xlink:label="sliceCompra"/>
<Noticia xlink:type "resource"
Xlink:label="sliceNoticia"/>
<link1 xlink:type="arc"
xlink:from ="slicePresentacion"
xlink:to="sliceIndice"
xlink:show="replace"
xlink:actuate="onRequest" />
<link2 xlink:type="arc"
xlink:from ="sliceIndice"
xlink:to="sliceUsuarioRegistrado"
xlink:show="replace"
xlink:actuate="onRequest" />
<link3 xlink:type="arc"
xlink:from ="sliceIndice"
xlink:to="sliceIndex1"
xlink:show="replace"
xlink:actuate="onRequest" />
<link4 xlink:type="arc"
xlink:from ="sliceIndice"
xlink:to="sliceIndex2"
xlink:show="replace"
xlink:actuate="onRequest" />
<link5 xlink:type="arc"
xlink:from ="sliceIndice"
xlink:to="sliceIndex3"
xlink:show="replace"
xlink:actuate="onRequest" />
<link6 xlink:type="arc"
xlink:from ="sliceIndice"
xlink:to="sliceIndex4"
xlink:show="replace"
xlink:actuate="onRequest" />
<link7 xlink:type="arc"
xlink:from ="sliceIndex1"
xlink:to="sliceCadenaDeCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link8 xlink:type="arc"
xlink:from ="sliceCadenaDeCine"
xlink:to="sliceIndex5"
xlink:show="replace"
xlink:actuate="onRequest" />
<link9 xlink:type="arc"
xlink:from ="sliceIndex5"
xlink:to="sliceCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link10 xlink:type="arc"
xlink:from ="sliceIndex2"
xlink:to="sliceCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link11 xlink:type="arc"
xlink:from ="sliceCine"
xlink:to="sliceIndex6"
xlink:show="replace"
xlink:actuate="onRequest" />
<link12 xlink:type="arc"
xlink:from ="sliceIndex6"
xlink:to="sliceCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link13 xlink:type="arc"
xlink:from ="sliceIndex6"
xlink:to="slicePelicula"
xlink:show="replace"
xlink:actuate="onRequest" />
<link14 xlink:type="arc"
xlink:from ="slicePelicula"
xlink:to="sliceIndex6"
xlink:show="replace"
xlink:actuate="onRequest" />
<link15 xlink:type="arc"
xlink:from ="slicePelicula"
xlink:to="sliceIndex7"
xlink:show="replace"
xlink:actuate="onRequest" />
<link16 xlink:type="arc"
xlink:from ="sliceIndex3"
xlink:to="slicePelicula"
xlink:show="replace"
xlink:actuate="onRequest" />
<link17 xlink:type="arc"
xlink:from ="sliceIndex4"
xlink:to="sliceNoticia"
xlink:show="replace"
xlink:actuate="onRequest" />
<link18 xlink:type="arc"
xlink:from ="sliceIndex7"
xlink:to="sliceSesion"
xlink:show="replace"
xlink:actuate="onRequest" />
<link19 xlink:type="arc"
xlink:from ="sliceSesion"
xlink:to="sliceSalaDeCine"
xlink:show="replace"
xlink:actuate="onRequest" />
<link20 xlink:type="arc"
xlink:from ="sliceSalaDeCine"
xlink:to="sliceIndex8"
xlink:show="replace"
xlink:actuate="onRequest" />
<link21 xlink:type="arc"
xlink:from ="sliceIndex8"
xlink:to="sliceButaca"
xlink:show="replace"
xlink:actuate="onRequest" />
<link22 xlink:type="arc"
xlink:from ="sliceButaca"
xlink:to="slicePago"
xlink:show="replace"
xlink:actuate="onRequest" />
<link23 xlink:type="arc"
xlink:from ="slicePago"
xlink:to="sliceCompra"
xlink:show="replace"
xlink:actuate="onRequest" />
</ModeloLogicoNavegacion>
</Links>
```

Figura 61. Código XLink correspondiente (CASO 7)

APÉNDICE D:
Reglas de Transformación del Modelo Conceptual a DTDs

TRANSFORMACIÓN DE UNA CLASE

Una clase de UML se transforma en un Elemento en el DTD.

<table>
<tr><th>UML</th><th colspan="2">DTD</th></tr>
<tr><td>CLASE UML</td><td colspan="2">ELEMENTO DTD</td></tr>
<tr><td rowspan="2">ATRIBUTOS DE LA CLASE
(regla general)
 
</td><td>SUBELEMENTOS DEL ELEMENTO</td><td><!ELEMENT ClaseA (at1, at2)>
<!ELEMENT at1(#PCDATA)>
<!ELEMENT at2(#PCDATA)></td></tr>
<tr><td>ATRIBUTOS DEL ELEMENTO</td><td><!ELEMENT ClaseA>
<!ATTLIST ClaseA
at1 CDATA
at2 CDATA)></td></tr>
</table>

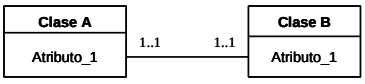
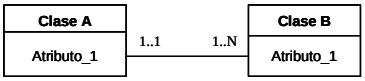
Consideraciones sobre la transformación de los Atributos de una Clase:

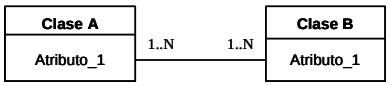
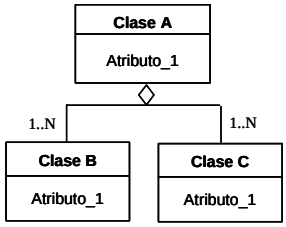
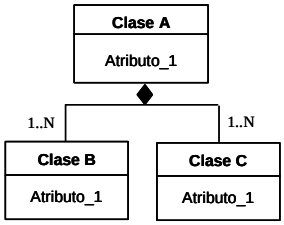
- Tipos de Datos para los atributos en un DTD:
  - CDATA – cadenas de caracteres
  - ID – atributos cuyos valores no se repiten, se emplean identificar unívocamente cada elemento
  - IDREF/IDREFS – sirven para referenciar otros elementos a partir de su ID (los utilizaremos a modo de punteros)
  - NMTOKEN/NMTOKENS – una cadena de texto de una o varias palabras

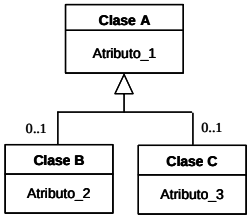
<table>
<tr><th colspan="2">TIPO DEL ATRIBUTO</th><th>MÉTODO DE TRANSFORMACIÓN</th><th>EJEMPLO</th></tr>
<tr><td rowspan="4">SIMPLE</td><td rowspan="2">OBLIGATORIO</td><td>Como Subelementos del Elemento</td><td><!ELEMENT ClaseA (atributo)>
<!ELEMENT atributo(#PCDATA)></td></tr>
<tr><td>Como Atributos del Elemento</td><td><!ELEMENT ClaseA>
<!ATTLIST ClaseA
atributo CDATA
#REQUIRED ></td></tr>
<tr><td rowspan="2">OPCIONAL</td><td>Como Subelementos del Elemento</td><td><!ELEMENT ClaseA (at1?)>
<!ELEMENT atributo (#PCDATA)></td></tr>
<tr><td>Como Atributos del Elemento</td><td><!ELEMENT ClaseA>
<!ATTLIST ClaseA
atributo CDATA #IMPLIED ></td></tr>
<tr><td rowspan="2">MULTI-VALUADO</td><td>OBLIGATORIO</td><td rowspan="2">Como Subelemento del Elemento
(sólo admite este método de transformación)</td><td><!ELEMENT ClaseA (at_multvldo)+>
<!ELEMENT atributo (#PCDATA)></td></tr>
<tr><td>OPCIONAL</td><td><!ELEMENT ClaseA (at_multvldo)*>
<!ELEMENT atributo (#PCDATA)></td></tr>
</table>

TIPO DEL ATRIBUTO	MÉTODO DE TRANSFORMACIÓN		EJEMPLO
COMPUESTO	Como Subelemento del Elemento que representa la clase	Subelemento	*<!ELEMENT ClaseA (atcomp)>* *<!ELEMENT atcomp(at1, at2)>* *<!ELEMENT at1(#PCDATA)>* *<!ELEMENT at2(#PCDATA)>*
		Atributos del Subelemento	*<!ELEMENT ClaseA (atcomp)>* *<!ELEMENT atcompEMPTY>* *<!ATTLIST atcomp at1 CDATA #REQUIRED at2 CDATA #REQUIRED>*
	Como nuevo Elemento referenciado como atributo en el Elemento que representa la clase		*<!ELEMENT ClaseA empty>* *<!ATTLIST ClaseA id_atcomp **IDREF #REQUIRED**>* *<!ELEMENT atcomp empty>* *<!ATTLIST atcomp clave **ID #REQUIRED**>*
ENUMERADO	Como atributo del Elemento que representa la Clase		*<!ELEMENT ClaseA empty>* *<!ATTLIST ClaseA at_enum **#FIXED** 'v1', 'v2' …>*

TRANSFORMACIÓN DE ASOCIACIONES

RELACIÓN	MÉTODO DE TRANSFORMACIÓN	EJEMPLO
RELACIÓN 1:1  	Unidireccional	*<!ELEMENT ClaseA (**ClaseB**)>* *<!ATTLIST ClaseA ….>* *<!ELEMENT ClaseB EMPTY)>* *<!ATTLIST ClaseB ….>*
	Bidireccional (si la relación es 1..1 se restringe con *REQUIRED*, si es 0..1 se utiliza *IMPLIED*)	*<!ELEMENT ClaseA EMPTY>* *<!ATTLIST ClaseA id_ClaseA **ID** ptr_ClaseB **IDREF #REQUIRED**>* *<!ELEMENT ClaseB EMPTY>* *<!ATTLIST ClaseB id_ClaseB **ID** ptr_ClaseA **IDREF #REQUIRED**>*
RELACIÓN 1:N  	Unidireccional (si la cardinalidad de la clase B es 1..N, se emplea el símbolo '+' – 1 ó más ocurrencias; si fuese 0..N se empleará '*' – 0 ó más ocurrencias)	*<!ELEMENT ClaseA (**ClaseB +**)>* *<!ATTLIST ClaseA ….>* *<!ELEMENT ClaseB EMPTY)>* *<!ATTLIST ClaseB ….>*
	Bidireccional	*<!ELEMENT ClaseA EMPTY>* *<!ATTLIST ClaseA id_ClaseA **ID** ptr_ClaseB **IDREFS** **#REQUIRED**>* *<!ELEMENT ClaseB EMPTY>* *<!ATTLIST ClaseB id_ClaseB **ID** ptr_ClaseA **IDREF #REQUIRED**>*

RELACIÓN	MÉTODO DE TRANSFORMACIÓN	EJEMPLO
RELACIÓN N:M  	Representar la relación con un nuevo elemento (que tiene como atributos referencias a las clases que participan en la relación)	*<!ELEMENT RelacionA_B EMPTY>* *<!ATTLIST RelacionA_B* *ptr_ClaseA* ***IDREF#REQUIRED*** *ptr_ClaseB* ***IDREF #REQUIRED****>*
	Incluir en el Elemento que representa cada clase atributos referenciadores a la otra clase	*<!ELEMENT ClaseA EMPTY>* *<!ATTLIST ClaseA* *id_ClaseA ID* *ptrs_ClaseB* ***IDREFS #REQUIRED>*** *<!ELEMENT ClaseB EMPTY>* *<!ATTLIST ClaseB* *id_ClaseB ID* *ptrs_ClaseA* ***IDREFS #REQUIRED>***
AGREGACIÓN  	La clases que representan las PARTES como Subelementos del Elemento que representa la clase del TODO	*<!ELEMENT ClaseA* ***(ClaseB +, ClaseC+)****>* *<!ATTLIST ClaseA id_ClaseA ID…>* *<!ELEMENT ClaseB EMPTY>* *<!ATTLIST ClaseB id_ClaseB ID …>* *<!ELEMENT ClaseC EMPTY>* *<!ATTLIST ClaseC* *id_ClaseC ID …>*
	La clases que representan las PARTES como Elementos independientes con una referencia a la clase que representa el TODO	*<!ELEMENT ClaseA EMPTY>* *<!ATTLIST ClaseA* *id_ClaseA ID* *ptrs_ClaseB* ***IDREFS #REQUIRED*** *ptrs_ClaseC* ***IDREFS #REQUIRED >*** *<!ELEMENT ClaseB EMPTY>* *<!ATTLIST ClaseB* *id_ClaseB ID* *ptr_ClaseA* ***IDREF #REQUIRED>*** *<!ELEMENT ClaseC EMPTY>* *<!ATTLIST ClaseC* *id_ClaseC ID* *ptr_ClaseA* ***IDREF #REQUIRED>***
COMPOSICIÓN  	Las clases que respresentan las PARTES como Subelementos del Elemento que representa la clase del TODO	*<!ELEMENT ClaseA* ***(ClaseB +, ClaseC+)****>* *<!ATTLIST ClaseA* *id_ClaseA ID…>* *<!ELEMENT ClaseB EMPTY>* *<!ATTLIST ClaseB* *id_ClaseB ID …>* *<!ELEMENT ClaseC EMPTY>* *<!ATTLIST ClaseC* *id_ClaseC ID …>*

JERARQUÍA		
	Las Subclases se representan como Subelementos de la Superclase	*<!ELEMENT ClaseA* ***(ClaseB ?, ClaseC?)*** *> <!ATTLIST ClaseA id_ClaseA ID...>* *<!ELEMENT ClaseB EMPTY> <!ATTLIST ClaseB id_ClaseB ID ...>* *<!ELEMENT ClaseC EMPTY> <!ATTLIST ClaseC id_ClaseC ID ...>*