LENGUAJE DE CONSULTAS PARA APLICACIONES HIPERMEDIAS BASADAS EN OOHDM

Autores
Pilar Panetta
Cecilia del Barrio
Gisela Gutierrez

Directores
Alicia Diaz
Silvia Gordillo

Facultad de Informática
Universidad Nacional de la Plata
1999

A nuestros Padres

Agradecemos a:

- Silvia Gordillo y Alicia Diaz por el apoyo recibido en el desarrollo de ésta Tésis.
- Damián Tolosa por su ayuda desinterasada.
- A nuestros seres queridos por la paciencia brindada en todos estos años.

Indice

Capítulo 1

INTRODUCCION

Entrada la década del 80, las aplicaciones hipermedias aparecen como una alternativa para crear sistemas de información complejos. La riqueza y versatilidad de sus conceptos han permitido que las mismas sean aplicadas en áreas tales como educación, ingeniería de software, arquitectura, diseño, etc.

Una hipermedia es una presentación de información que combina texto, imágenes, sonido, video, etc. y que no está restringida a ser lineal dado que no existe un orden simple que determine la secuencia a seguir. La estructura completa de una hipermedia muestra una red, la cual esta compuesta de dos tipos de elementos, nodos y links; donde los nodos son los que poseen la información y los links son los que nos permiten atravesar de un nodo hacia otro. Esta navegación entre nodos constituye el método de acceso a la información disponible en la hipermedia y es una de las características distintivas de este tipo de aplicaciones.

Uno de los principales conflictos que encuentra el usuario a la hora de interactuar con una hipermedia es la forma de obtener la información. Trabajar con hipermedias mal estructuradas o que contengan grandes volúmenes de información son algunas de las posibles causas de este problema. En el primer caso, el usuario podrá sentirse 'desorientado' dado que la información no se presenta en forma clara y ordenada. En tanto en el segundo caso el usuario deberá navegar a través de información no deseada hasta alcanzar finalmente la que es de su interés. Cualquiera de estas situaciones lo pueden llevar a desistir de la búsqueda, aun cuando, puede describir con exactitud lo que está buscando pero encuentra dificultades para lograrlo.

Una solución al problema presentado es agregar un mecanismo de acceso por consulta, es decir el usuario podrá formular una consulta que encapsule la información que desea buscar y una vez recuperada dicha información podrá especificar la forma de navegar sobre la misma. De esta manera el usuario accederá rápidamente a la información deseada para luego navegarla.

Persiguiendo este objetivo utilizamos como base el lenguaje de consultas OOHQL [1], [2]. Este lenguaje combina las características de las consultas de bases de datos orientadas a objetos y primitivas de navegación para hipermedias.

Modificando la notación y ampliando el lenguaje previamente mencionado definimos un nuevo lenguaje que nos permite realizar consultas sobre un hipertexto diseñado con el modelo OOHDM [6]. El lenguaje ofrece la posibilidad de consultar información del dominio de una aplicación y el esquema de la misma evitándose de esta forma tener que perder tiempo intentando encontrar la información deseada y contribuyendo así a resolver el problema de la desorientación.

Al elegir el modelo de diseño OOHDM tuvimos en cuenta entre otros factores su

completitud, destacando la potencialidad de poder definir distintas visiones navegacionales para un mismo dominio de información [8], [9], [10]. Algunas de estas características lo distinguen sobre otros modelos de diseño anteriores tales como HDM [3], RMM [4] o EORM [5].

En el capítulo 2 presentamos el método de diseño OOHDM (Object Oriented Hipermedia Design Model), analizando el modelo, los productos generados en cada actividad del proyecto y un breve análisis de cada etapa, haciendo hincapié en las primeras dos etapas del método.

El capítulo 3 presenta un Lenguaje de Consultas para aplicaciones hipermedias diseñadas con el modelo OOHDM, describiendo la sintaxis y la semántica de las consultas de información, de estructura y constructores navegacionales, acompañada de ejemplos ilustrativos.

En el capítulo 4 se exhibe un ejemplo práctico donde se modeliza bajo OOHDM una aplicación hipermedia existente "Art Gallery" y se muestra el uso del lenguaje generando un conjunto de consultas y definiendo la forma de navegar los resultados obtenidos.

Finalmente, hemos incluido dentro de un apéndice un conjunto de conceptos mencionados a lo largo del desarrollo de esta tesis, como así también una pequeña descripción de otros métodos de diseño para aplicaciones hipermediales.

Capítulo 2

OOHDM

(Object Oriented Hypermedia Design Model)

2.1 Introducción

Frecuentemente el desarrollo de sistemas hipermediales suele hacerse utilizando directamente herramientas de autor a nivel de implementación, descuidándose el importante proceso previo de análisis y diseño abstracto de los aspectos estructurales, de navegación y de interfaz con el usuario. Sin embargo, en los últimos años existe una tendencia a considerar el desarrollo hipermedial con un enfoque de proceso de ingeniería, por lo que ya se han propuesto diferentes metodologías, como HDM (Hypertext Design Model) o RMM (Relational Management Methodology) (*ver Apéndice A*), que establecen la necesidad de considerar un diseño previo a la construcción del sistema y ofrecen una serie de técnicas para recoger en diferentes modelos abstractos las especificaciones del sistema a desarrollar.

Luego surgieron metodologías que asumen la Orientación a Objetos como paradigma de diseño. En el caso de hipermedias, el enfoque orientado a objetos en los modelos es muy útil debido al gran nivel de abstracción que ofrece y a su mecanismo de composición que facilitan el modelado de la estructura hipermedial. Esto queda reflejado en dos de las metodologías más conocidas de desarrollo orientado a objetos de sistemas hipermediales como son EORM (Enhanced Object Relationship Model) (*ver Apéndice A*) y OOHDM [8].

La principal diferencia entre EORM y OOHDM radica en que OOHDM separa claramente el diseño conceptual de la aplicación del diseño navegacional, ofreciendo técnicas diferentes para ambos casos [6].

En este capítulo comentamos el modelo OOHDM en el que nos basamos para construir el Lenguaje de Consultas presentado en esta tesis.

2.2 Descripción del Modelo

OOHDM es un modelo para diseñar aplicaciones hipermedias. El proceso de diseño de

una aplicación hipermedia bajo este modelo comprende cuatro actividades fundamentales bien diferenciadas: en la primera etapa se realiza el modelado del dominio de la aplicación, obteniéndose un esquema conceptual de clases; en la segunda se expresa el diseño navegacional a través de dos tipos de esquemas: el esquema de clases navegacionales y el esquema de contexto navegacional; en la tercera etapa se especifica la estructura y el comportamiento de la interfaz del sistema hipermedia con el usuario; en la cuarta y última etapa se desarrolla el sistema real de hipermedia, se mapea el diseño de la interfaz abstracta en objetos de la interfaz concreta, es decir asocian los objetos de la interfaz abstracta con el lenguaje seleccionado para la implementación.

Estas actividades están directamente relacionadas unas con otras y existe entre ellas una fuerte dependencia al momento de su desarrollo. Durante las primeras tres actividades son construidos modelos orientados a objetos no siendo así en la etapa de implementación de la aplicación.

La siguiente figura muestra en forma resumida cada una de las etapas del proyecto especificando claramente los productos, mecanismos e interfaz involucrados en las mismas.

Productos	Mecanismos	Interfaz del proyecto
Clases, Subsistemas, relaciones y perspectivas de atributos	Clasificación, composición, generalización, especialización.	Modelaje de la interfase del dominio de la aplicación.
Nodos, links, estructuras de acceso, contextos de navegación, transformaciones navegacionales.	Mapeo de objetos conceptuales y de navegación. Padrones de navegación para la descripción de la estructura de la aplicación.	Tiene en cuenta el perfirl del usuario y la tarea. Enfatiza en aspectos cognitivos y arquitectónicos.
Objetos de interfase abstracta, relaciones a eventos externos, transformaciones de interfaz.	Mapeo entre objetos de navegación y objetos de interfaz.	Modelaje de objetos perceptibles. Implemetación de metáforas escogidas. Descripción de interfaz para objetos navegacionales.
Aplicación en ejecución	Aquellos provistos por el ambiente objetivo	Desempeño, completitud.

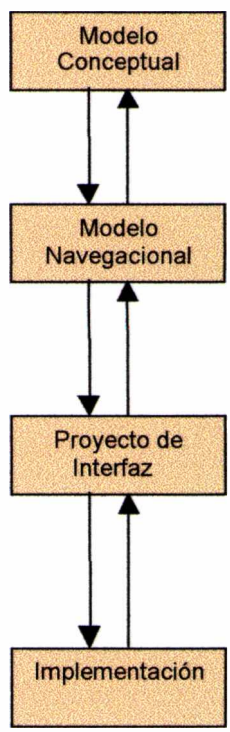


Figura 2.1: Actividades de OOHDM

A continuación, detallamos las distintas etapas antes mencionadas, haciendo hincapié en las correspondientes al Modelo Conceptual y el Modelo navegacional, dada su estrecha relación con el análisis de esta tesis. Se presentarán las etapas de Interfaz abstracta e implementación en forma breve y concisa, dejando a cargo del lector la profundización del tema [6], [10].

Hemos tomado a modo de ejemplo una aplicación de cine, ya existente, que será modelizada siguiendo las fases de diseño planteadas por OOHDM.

2.3 Modelo Conceptual

La tarea a realizar durante esta etapa es definir el dominio de la aplicación. Para ello se describe el conjunto de clases conceptuales identificando las relaciones que surgen entre ellas y organizándolas en jerarquías de clases en el caso de existir; determinar qué subsistemas se involucran, y construir con ello el modelo conceptual de clases.

Las clases se describen como un conjunto de atributos tipados y comportamiento. Para definir qué valores pueden asumir cada uno de los atributos de una clase se debe asignar un tipo o clase específica a cada uno de ellos.

Un tipo representa una relación implícita o la apariencia visual de este atributo, denominada también perspectiva del atributo.

Las clases pueden ser construidas usando jerarquía de generalización/especialización y de esta manera acrecentar el poder de abstracción del sistema.

Las relaciones expresan conexiones entre objetos del dominio, de este modo cuando una relación está entre clases ésta abstrae la relación objeto - objeto. Las relaciones en el modelo conceptual serán mapeadas a links en el modelo navegacional por lo tanto las relaciones no deben estar escondidas en los atributos de las clases. Esto significa que en el caso de que un atributo represente una entidad conceptual compleja a ser explorada en la hipermedia final, se debe especificar una relación.

Las relaciones son también definidas como clases por lo que incluyen atributos y comportamiento y son luego organizadas en jerarquías.

Las clases podrán relacionarse con subsistemas. Los subsistemas contienen puntos de entradas, que serán las clases, que podrán ser accedidas desde otras clases u otros subsistemas.

El modelo ofrece tres constructores de abstracción:

- Agregación: es útil para describir clases complejas como agregado de clases más simples.
- Generalización/especialización: es útil en la construcción de la jerarquía de clases y el uso de herencia como un mecanismo de reusabilidad.
- Subsistemas: es un mecanismo de agrupación para la abstracción de modelos de dominios complejos.

El objetivo de la construcción de un modelo conceptual es capturar la semántica del dominio de una aplicación.

A continuación mostramos una representación gráfica del esquema conceptual de una multimedia de Cine Argentino [11].

La siguiente figura exhibe las clases del dominio: *Persona, Actor, Director, Película, Premios*, y las relaciones entre dichas clases: *influenciado_por, es_influenciado, recibió, dirigió, dirigida_por, otorgado_a, premia_a, actuó y trabaja*.

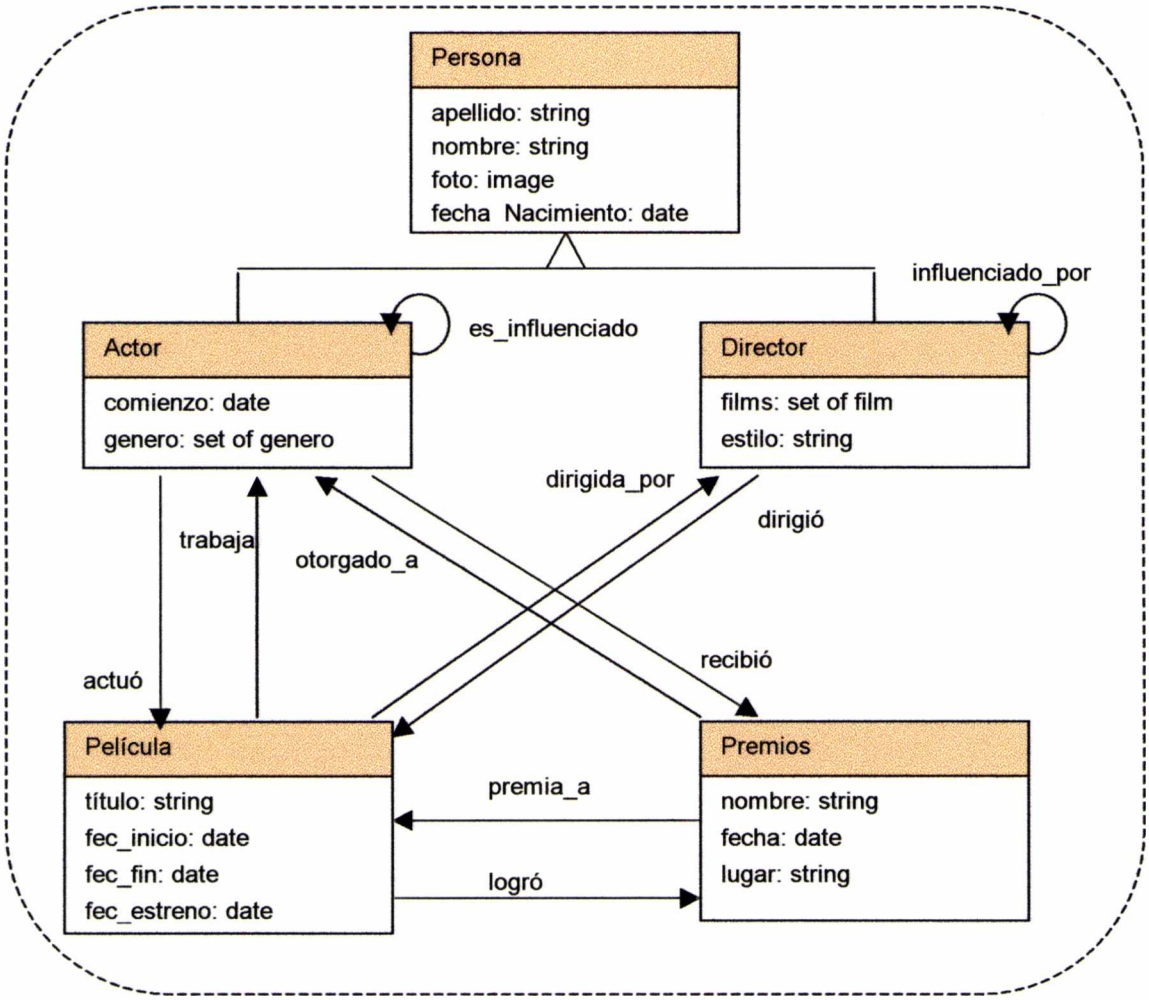


Figura 2.2: Esquema de clases conceptuales del Cine Argentino

En este punto se refleja sólo la semántica del dominio sin tener en cuenta las visiones de la aplicación.

OOHDM utiliza tarjetas para documentar cada una de las componentes del esquema conceptual. A continuación se presenta el formato de las tarjetas de clase y relación junto a las instancias de las mismas.

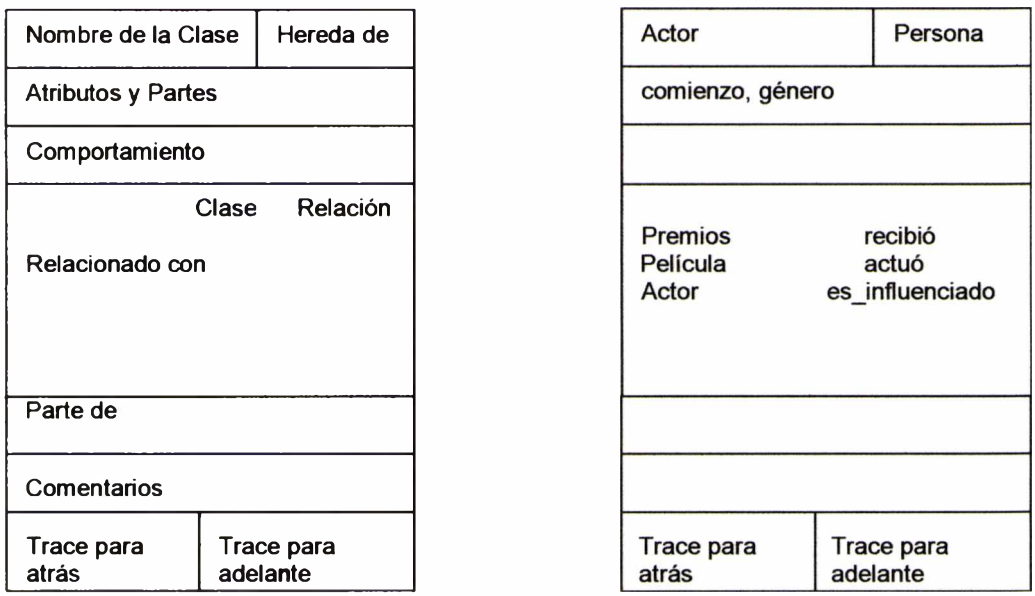


Figura 2.3: Tarjeta de Clase e instancia de la tarjeta de Clase.

Donde:

Nombre de la Clase: nombre identificador de la clase.

Hereda de: nombre de la superclase de la clase.

Atributos y partes: se mencionan los atributos simples y compuestos de la clase en definición.

Comportamiento: nombres de los métodos asociados a la clase.

Relacionado con: Clase-Relación: se detallan las distintas clases con las que se relaciona la clase y el nombre de relación correspondiente en cada caso.

Parte de: nombres de las clases de las cuales forma parte la clase mencionada.

Trace para atrás/ Trace para adelante: traza de navegación

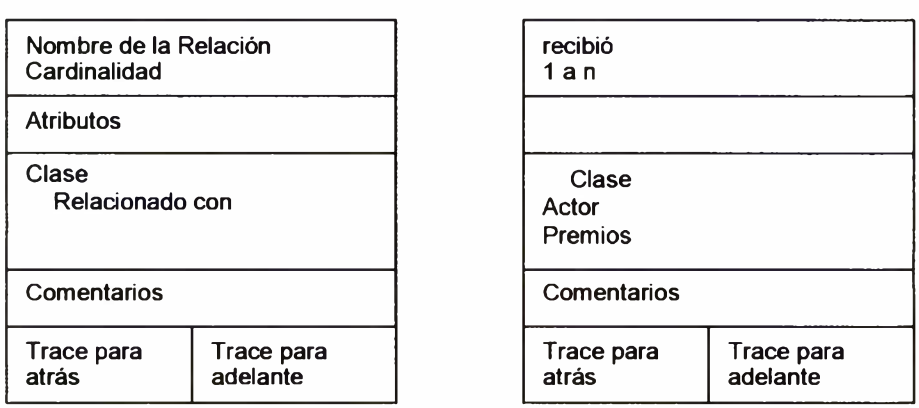


Figura 2.4: Tarjeta de una Relación e instancia de la Clase Relación.

Donde:

Nombre de la relación: nombre identificador de la clase relación.

Cardinalidad: Tipo de cardinalidad de la relación.

Atributos: se mencionan los atributos de la relación.

Relacionado con: se detalla el par de clases que se relacionan con la relación que está siendo especificada.

Trace para atrás / Trace para adelante: traza de navegación

2.4 Modelo Navegacional

En esta etapa la información del modelo conceptual es reorganizada adaptándola a las diferentes visiones navegacionales del dominio del problema de cada usuario de la aplicación. Utilizando el ejemplo del cine, la aplicación es vista en formas diferentes si el usuario es un *crítico de cine* o un *espectador.* Cada una de ellas construye un tipo distinto de aplicación hipermedia y define un conjunto de contextos y clases navegacionales. Los contextos navegacionales expresan una estructura navegacional general de la aplicación hipermedia, mientras que las clases navegacionales, como los nodos y links, especifican los objetos que serán vistos por el usuario. Las clases conceptuales son representadas por nodos en el modelo navegacional, y las relaciones por links.

El diseño navegacional está expresado en dos esquemas, el esquema de clases navegacionales y el esquema de contextos navegacionales.

2.4.1 Esquema de clases navegacionales

El esquema de clases navegacionales expresa las posibles vistas de la aplicación hipermedia a través de unos tipos predefinidos de clases, llamadas navegacionales, como son los nodos, los links y otras clases que representan estructuras o formas alternativas de acceso a los nodos como índices y tours guiados.

En OOHDM existe un conjunto predefinido de clases navegacionales:

Nodos: los nodos contienen la información que poseen las aplicaciones hipermedias, y son organizados en jerarquías "*parte_de*" y "*es_un*". Los nodos poseen atributos que pueden ser de diferentes tipos: aquellos que contienen información representada en las aplicaciones y anchors (otra clase básica).

Los nodos pueden ser accedidos en diferente contextos y sus atributos y anchors pueden mudarse de un contexto a otro.

Links: los links son definidos a partir de relaciones y se representan por medio de flechas en el esquema navegacional. Un link posee atributos (origen, destino y cardinalidad) y comportamiento. De acuerdo al comportamiento del link la aplicación reacciona de diferentes maneras. Cada vez que un link es atravesado entre objetos navegacionales existentes, un conjunto de objetos disponibles al usuario se modifica y otro puede o no desaparecer. Otro tipo de comportamiento que los links deben expresar es aquel para alcanzar el nodo objetivo (ya sea por medio de un algoritmo o consultando un banco de datos).

En OOHDM existen dos tipos de links: links de aplicación y links de contexto. Los primeros conectan nodos de diferentes contextos, los segundos conectan objetos navegacionales dentro del mismo contexto.

Anchors: los anchors se consideran disparadores de navegación en OOHDM. La selección de un anchor activa una navegación. La definición de un anchor recibe como argumento el tipo de link o estructura de acceso originada en ese anchor. Cuando un link recibe el mensaje "seleccionado" debe activar la instancia de link que corresponda.

Los anchors dependen siempre del contexto en el que el nodo correspondiente es activado. En el caso de que un mismo anchor sea utilizado en todo contexto navegacional definido para la aplicación hipermedia, éste deberá formar parte de la definición del nodo. Si el anchor se encuentra disponible sólo en algún contexto en particular la definición del mismo se hará en la clase EnContexto. Si un anchor opera como disparador de distintos links de acuerdo con el contexto en el cual se accede al nodo también será definido en la clase EnContexto.

Finalmente, los anchors pueden ser representados como objetos en la interfaz por medio de un botón o ícono, o simplemente no aparecer y ser activado el mensaje "seleccionado" por algún evento acontecido, como por ejemplo el resultado del pasar del tiempo.

Composiciones y Links estructurales: Describir objetos usando agregación de objetos más simples permite abstraer una compleja jerarquía de objetos en un objeto compuesto.

OOHDM permite la definición de diferentes tipos de objetos navegacionales compuestos. La definición de un objeto compuesto encierra también la definición de links entre las partes, que son conocidos como links de contexto o estructurales; y la especificación de cómo el usuario navegará a través de los componentes del compuesto, es decir la semántica navegacional asociada al objeto compuesto.

El tipo más sencillo de composición es el nodo compuesto. Navegar en el interior de un compuesto implica moverse en el interior de un contexto.

Indices y estructuras de acceso: las estructuras de acceso son modeladas como clases en el modelo navegacional y caracterizadas por un conjunto de selectores, un conjunto de objetos destino (generalmente objetos del contexto navegacional corriente) y un predicado en los objetos destino. Las estructuras de acceso pueden contener también otros atributos.

Las estructuras de accesos funcionan como índices o diccionarios, siendo ejemplo de éstas, las líneas de tiempo, índices jerárquicos, etc. Las mismas proveen caminos para facllitar el acceso a la información.

En la figura 2.5 se muestra el esquema de clases navegacionales de la aplicación del Cine Argentino. Las clases navegacionales *Persona, Actor, Director, Película y Premios* hacen referencia a la información general que se encuentra dentro de un de la aplicación del cine.

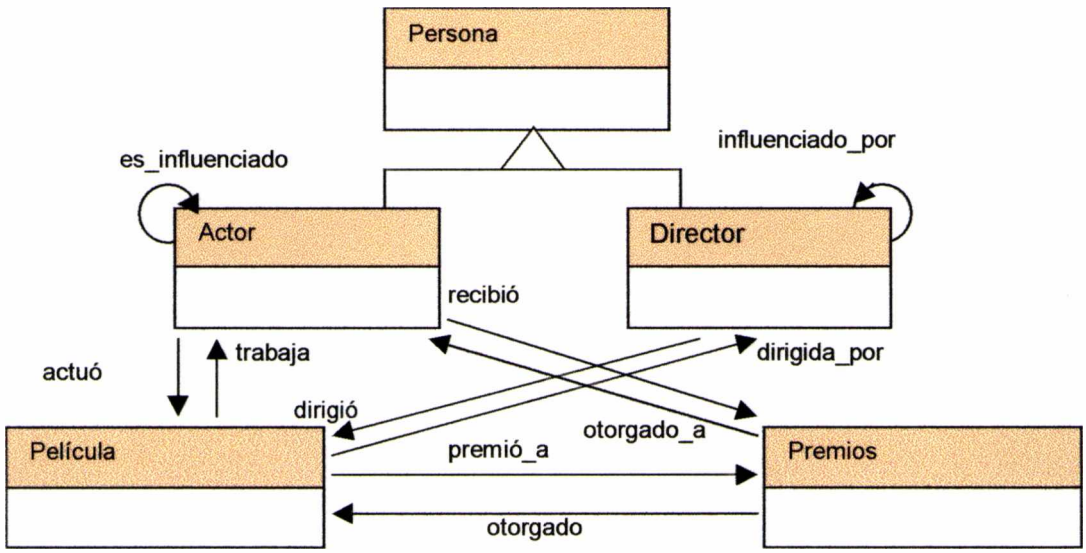


Figura 2.5. Esquema de Clases Navegacionales del Cine Argentino

2.4.2 Esquema de contextos navegacionales

El esquema de contextos navegacionales permite estructurar el espacio de navegación en subespacios, para lo que se indica la información que será mostrada al usuario y los enlaces que están disponibles cuando se acceda a un objeto (nodo) en un contexto determinado.

Un contexto navegacional es un conjunto de nodos, links y otros contextos navegacionales, que incluye también caminos predefinidos entre sus elementos.

En OOHDM existen diversos tipos de contextos navegacionales:

- **Contexto navegacional arbitrario:** es aquel en que el autor escoge cada uno de los miembros del contexto que pueden ser instancias de diferentes clases navegacionales. Se representan directamente por medio de Tours guiados.

- **Contexto navegacional anidado:** es aquel que contiene como elementos otros contextos de navegación. En este caso el usuario debe elegir uno de los elementos para proseguir la navegación.

- **Contexto navegacional derivado de link:** son contextos generados a partir de la definición de links entre dos clases. Son utilizados generalmente para representar la situación de que el usuario recorre un link que apunta a más de un nodo. En ese caso el usuario puede elegir explícitamente uno de los nodos destino o es llevado directamente a uno de ellos teniendo luego la posibilidad de visitar el resto de los

nodos destino de aquel link.

- **Contexto de navegación derivado de un compuesto:** los nodos compuestos generan un contexto de navegación que es el conjunto de las partes de un nodo. Las partes de un objeto agregado forman una estructura jerárquica que puede poseer un orden de navegación. Esa estructura jerárquica puede ser mapeada en un contexto navegacional. En general este tipo de contexto no es explícitamente especificado. Cuando estos contextos no son definidos, los objetos agregados son utilizados directamente en los aspectos de navegación de la hipermedia.

- **Contexto de navegación de índice:** este tipo de contexto tiene una semántica de navegación definida por medio de un índice para sus elementos. La navegación se realiza desde el índice hacia los elementos del contexto o desde un elemento hacia el índice. Un contexto de navegación de índice puede poseer otras semánticas de navegación como una navegación secuencial a través de los elementos del contexto o una navegación hacia otro contexto que tenga un elemento en común con el contexto actual.

- **Contexto de navegación de sesión:** son contextos construidos dinámicamente de acuerdo al usuario o a la sesión. Un ejemplo de estos contextos es la lista de los nodos visitados en una aplicación.

De acuerdo con el tipo de elemento de un contexto de navegación, OOHDM define dos tipos: contextos de navegación homogéneos y contextos de navegación heterogéneos. Un contexto navegacional es homogéneo cuando todos sus nodos pertenecen a la misma clase y heterogéneo cuando sus nodos pertenecen a clases diferentes.

Los contextos de navegación pueden definir caminos donde los nodos son presentados en secuencias predefinidas por el autor o donde el usuario puede escoger recorrer el camino explorando los nodos de alguna otra manera. Dichos contextos pueden incluir caminos secuenciales o condicionales.

Visto que un nodo puede pertenecer a más de un contexto navegacional, el usuario puede mudar de un contexto a otro que contenga el nodo actual y recorrer un link en ese nuevo contexto. Además de esto, el usuario puede recorrer un link que lo lleve a otro nodo de otro contexto de acuerdo a alguna relación entre las clases a las que pertenecen.

Al definir un contexto de navegación, el autor decide qué objetos estarán disponibles a partir de cada nodo dentro de cada contexto. En algunas situaciones el autor puede dar al usuario total libertad de navegación, es decir, el usuario posee todas las opciones de navegación disponibles en todo momento. Existen casos en los que el autor restringe esta posibilidad de navegación total para garantizar que el usuario pase por los nodos considerados más importantes en la hipermedia.

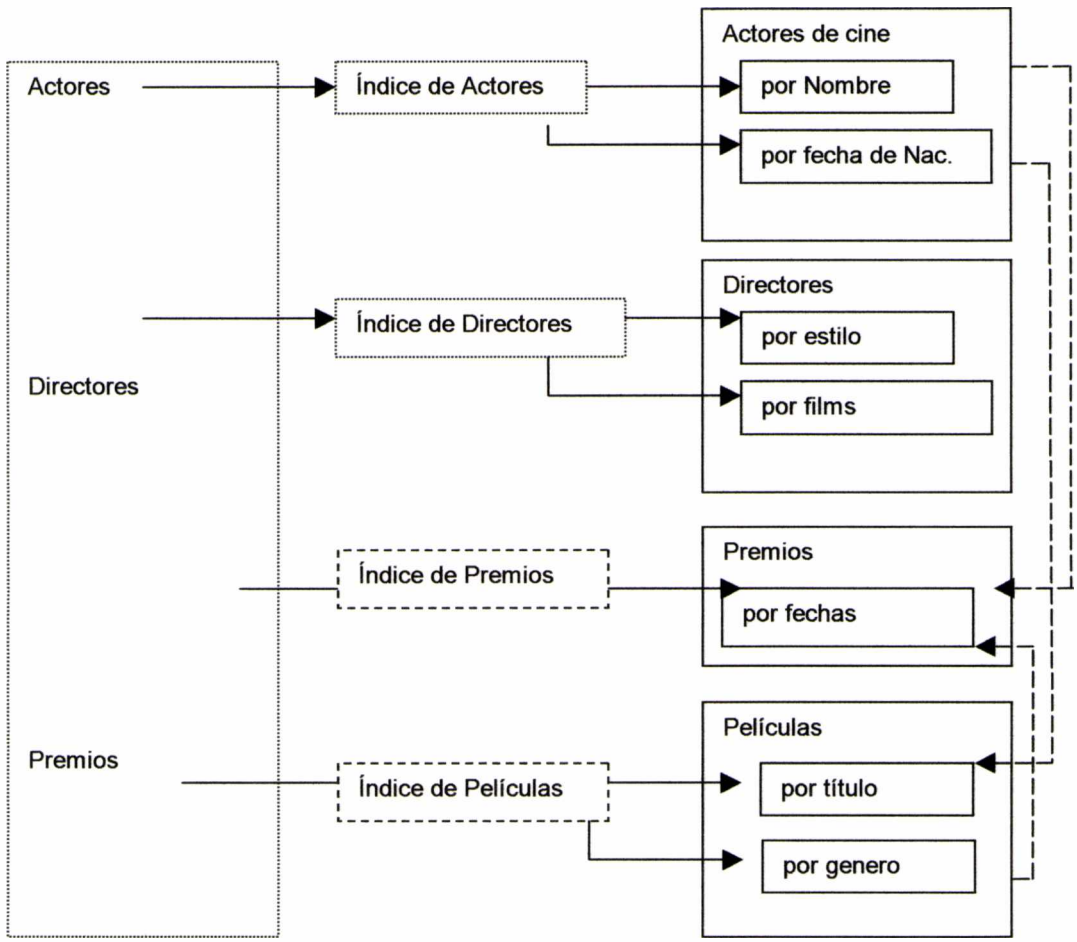


Figura 2.6: Esquema de contexto Navegacional del Cine Argentino.

En la figura 2.6 se representa un esquema de navegación para el ejemplo Cine Argentino, compuesto por índices (representados por cajas punteadas) y contextos (representados por cajas de líneas continuas). El índice de la izquierda representa el índice principal a partir del cual pueden ser accedidos índices intermedios. Los nodos pertenecientes a una misma clase son agrupados en cajas continuas etiquetadas: por ejemplo *Actores de Cine* encierra dos contextos cuyos nodos pertenecen a la clase navegacional *Actor*. Las líneas punteadas representan los cambios de un contexto a otro.

En OOHDM el esquema de clases navegacionales muestra las relaciones entre los objetos navegacionales, en cuanto el esquema de contexto navegacional muestra la navegación en general.

Formalmente, un contexto navegacional es documentado por medio de tarjetas exhibida en la figura 2.7.

Nombre del Contexto	Tipo
Atributos	
Incluye Clase EnContexto	
Punto de Entrada Punto de Entrada Inicial	
Mudanzas de contexto	
Estrategia	
Trace para atrás	Trace para adelante

Figura 2.7: Tarjeta de Contexto.

Los atributos son utilizados para representar propiedades de navegación así como información contextual. Dentro de los objetos que un contexto incluye tendremos clases de objetos y otros contextos navegacionales.

Los puntos de entrada en un contexto definen que miembros del contexto podrán ser accedidos desde un nodo fuera de ese contexto. En general, todos los elementos de un contexto pueden ser accedidos a partir de un nodo que no pertenece al contexto excepto cuando hay un punto de entrada obligatorio.

Algunos contextos de navegación de índice poseen un punto de entrada obligatorio para el propio nodo del contexto, que contiene el índice para sus miembros. Un ejemplo común de un contexto navegacional con un punto de entrada obligatorio es el tour guiado.

En la definición formal del contexto se debe especificar también su comportamiento, el cual define como será la navegación de acuerdo a dos aspectos: control y restricción.

2.4.3 Transformaciones navegacionales

Cuando se atraviesan los links se dice que ocurren transformaciones navegacionales. Estas definen en forma completa la estructura dinámica navegacional. En OOHDM se usa un modelo de estados de transición orientados a objetos, soportando estructura y comportamiento, anidamiento y permitiendo expresar modos de dinámica navegacional. A su vez, esta definición se complementa con un conjunto de predicados expresados en lógica temporal que permite consultar la especificación contra las propiedades deseadas tales como seguridad, etc.

2.5 Interfaz Abstracta

La metodología OOHDM contempla una tercera fase de diseño, denominada diseño de interfaz abstracta, en la que se realiza un modelo, para especificar la estructura y el comportamiento de la interfaz de la aplicación hipermedia con el usuario. Este modelo es abstracto y, por lo tanto, independiente de la implementación final del sistema.

El modelo de la intefaz abstracta se expresa a través de tres tipos de diagramas que se complementan entre sí. En primer lugar se deben crear los denominados diagramas de *Vistas de Datos Abstractos* (ADVs) que incluyen una vista (ADV) por cada clase navegacional (nodo, enlace o estructura de acceso) que fue establecida durante la etapa de diseño navegacional. Un diagrama de este tipo se compone de una serie de cajas (una caja es un ADV) que presentan las diferentes clases de objetos que aparecerán ante el usuario. Un segundo tipo de diagrama que compone el modelo de interfaz es el *Diagrama de Configuración*, donde se representan principalmente los eventos externos (provocados por el usuario, como el click o doble click del mouse) que maneja un ADV, los servicios que ofrece el ADV (como visualización) y las relaciones estáticas entre las ADV's. Por último, el modelo se completa con los denominados *Diagramas de Estados* que, representan el comportamiento dinámico de la aplicación hipermedia mediante un diagrama de transición de estados para cada ADV en el que se refleja los posibles estados por los que puede pasar cada objeto de la interfaz (oculto, desactivado, ampliado, reducido, normal, etc.) y los eventos que originan los cambios de estados.

2.6 Implementación

Para obtener una implementación ejecutable, el diseñador tiene que mapear los modelos de navegación y de interfaz abstracta en objetos concretos disponibles en la elección del ambiente de implementación. El modelo generado después de desarrollar los pasos 1-3 puede ser implementado en una forma directa utilizando muchas de las plataformas para generar hipermedia que están disponibles actualmente tales como Toolbook, Hipercard, Mac Web, etc. El uso de un conjunto uniforme de constructores de modelos (objetos y clases) en esta metodología permite una suave transición desde la modelización del dominio hasta el diseño navegacional y de interfaz.

Para el paso de implementación no se necesita de un ambiente orientado a objetos, sin embargo esto haría el trabajo más simple. En el caso en que el ambiente de ejecución no sea orientado a objetos, se pueden usar algunas técnicas que mapeen una especificación orientada a objetos a ambientes de ejecución no orientadas a objetos.

Capítulo 3

LENGUAJE DE CONSULTAS

3.1 Introducción

Actualmente, el método de acceso primario para alcanzar la información que contiene una aplicación hipermedia es la navegación. Existen aplicaciones hipermediales en las cuales el acceso navegacional es problemático debido a que conforman redes muy extensas y heterogéneas. En estos casos los usuarios se pierden caminando sin rumbo sobre la red para poder encontrar cierta información.

El método que OOHDM propone para acceder a la información es navegar sobre la red especificando contextos.

Supongamos que estamos navegando en una aplicación hipermedia diseñada bajo este modelo que posee gran cantidad de información. El alcanzar información concreta implica recorrer alguno o todos los contextos especificados en el momento del diseño, de este modo comenzaremos a recorrer el contexto que nos resulte más apropiado para alcanzarla. Durante la búsqueda puede suceder que nos encontremos con información no deseada o que simplemente nos desviemos de la búsqueda debido a que estamos recorriendo otro contexto, que sin saberlo comenzamos a recorrer como consecuencia de navegar por algún nodo que pertenecía a dos contextos.

Por lo tanto vemos que recuperar información de una hipermedia muy extensa puede resultar tedioso e interminable.

Para solucionar el problema antes mencionado proponemos un mecanismo de consulta como forma de acceso primaria junto con la navegación. Para ello, desarrollamos un lenguaje de consultas que nos permite acceder a la información, de aplicaciones hipermedias, seleccionándola de acuerdo al valor de sus atributos o a través de su semántica.

El lenguaje consta de dos niveles de consulta: consultas sobre la estructura de la hipermedia y consultas sobre la información contenida en la misma.

A su vez, provee constructores navegacionales que son utilizados para definir la semántica de navegación, aplicables al resultado de las consultas de información.

3.1.1 Esquema e instancia de esquema de la aplicación Portinari

La aplicación "Proyecto Portinari" es una aplicación hipermedia que muestra la colección de todos los trabajos producidos por el artista Cándido Portinari, así como documentos sobre su vida y la de sus contemporáneos, contando entre éstos a artistas, figuras políticas e intelectuales de la época.

Presentamos en primer lugar, el esquema navegacional de la aplicación que fue diseñada bajo el modelo OOHDM, el cual ha sido modificado para ser utilizado como ejemplo a lo largo de todo el capítulo. En segundo lugar se muestra una instanciación del mismo.

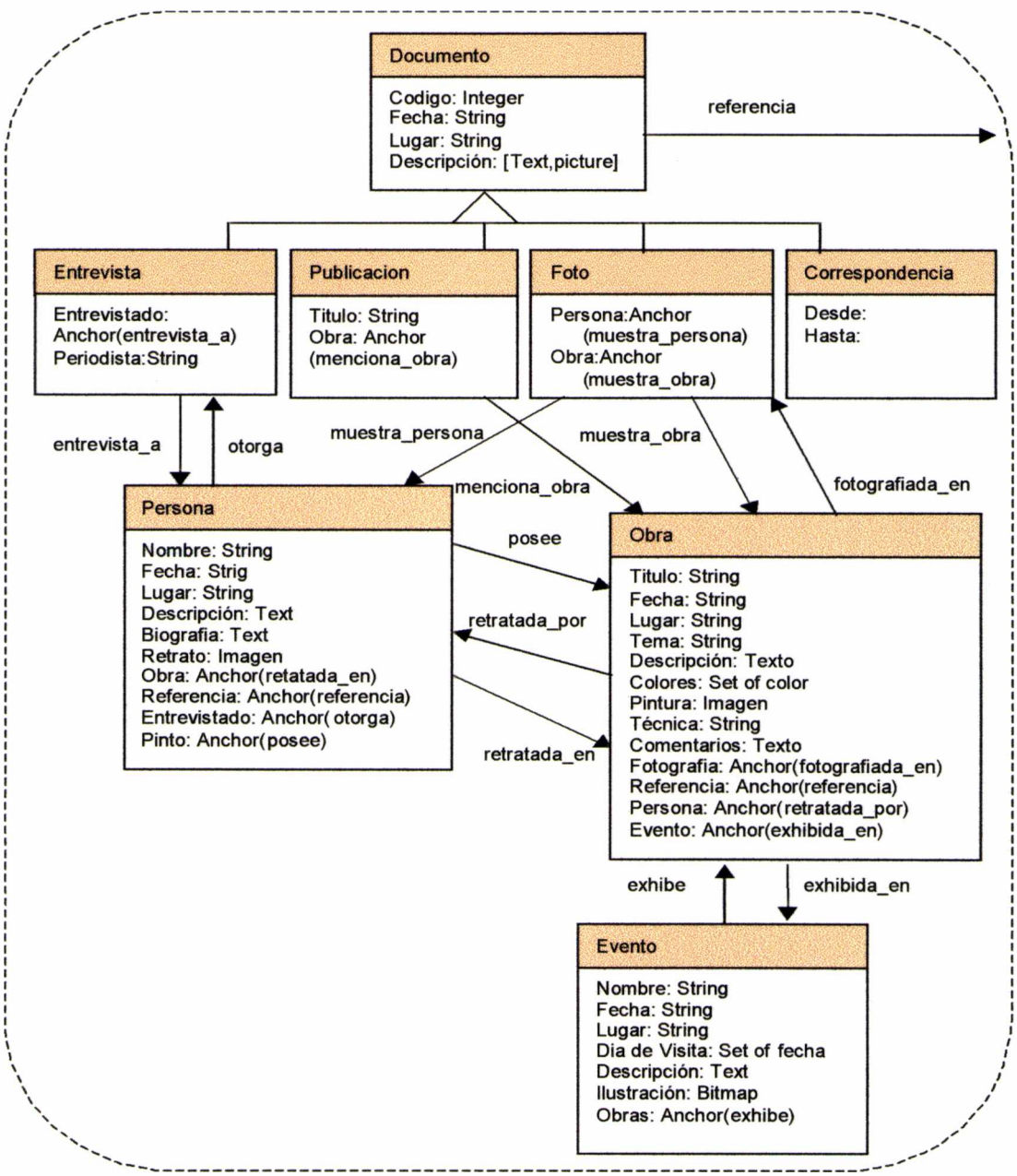


Figura 3.1: Esquema Navegacional de la aplicación Portinari

Como se puede apreciar en la figura 3.1, la aplicación Portinari está compuesta por las clases nodos: Evento, Foto, Obra, Persona, Entrevista, Correspondencia, Publicación, Documento, y las clases links: Describe, Retratada_en, Retratada_por, Otorga, Posee, Referencia, Muestra_persona, Exhibida_en, Fotografiada_en, Entrevista_a, Menciona_Obra, Muestra_obra, Exhibe.

A las instancias de las clases nodos las llamaremos nodos y formarán parte de una instancia del esquema navegacional que definimos en la figura 3.2. En la misma, los nodos se representan por medio de cajas redondeadas y los links por medio de flechas discontinuas así como las regiones sombreadas encierran los nodos de una misma clase.

Los nombres que referencian los links se corresponden con el nombre de la clase de los mismos.

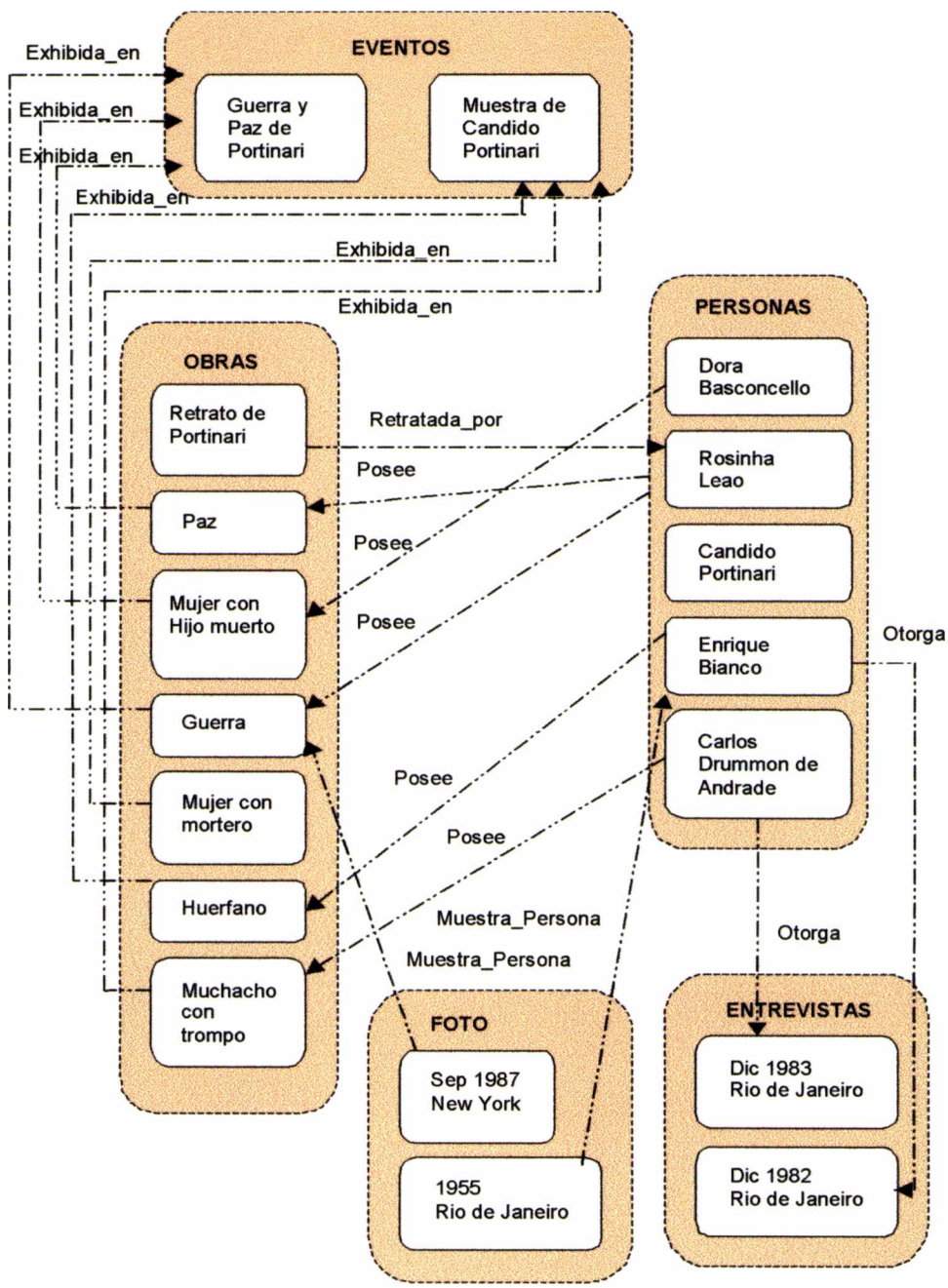


Figura 3.2: Instancia del Diagrama navegacional de la aplicación Portinari.

Cada nodo representa información de una *Persona*, una *Entrevista* u otro nodo concreto.

Durante el desarrollo de este capítulo enunciaremos ejemplos de consultas sobre los valores de los atributos de los nodos contenidos en la aplicación Portinari. En la figura 3.3, 3.4, 3.5 y 3.6 se detalla la información contenida en ellos, de modo tal que el lector pueda comprender las gráficas que acompañan a los ejemplos.

Nota: Para brindar una mayor claridad a la hora de interpretar la información se incluye en la representación de los nodos el nombre de sus atributos.

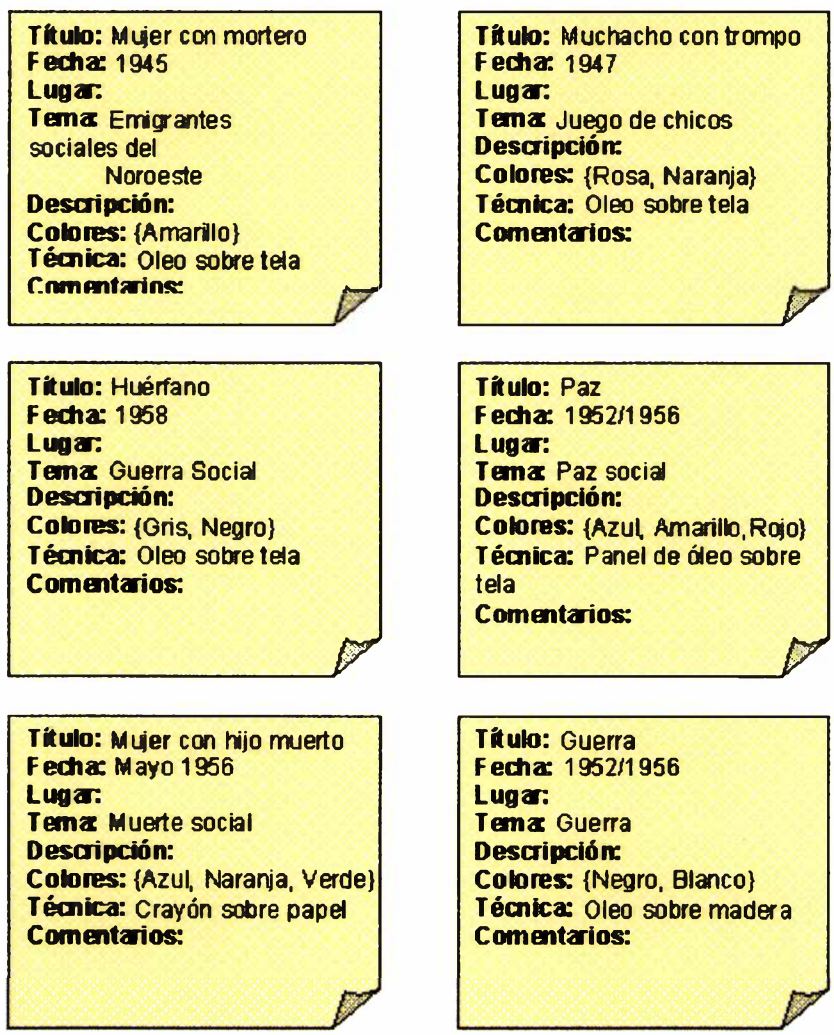


Figura 3.3: Nodos Obra

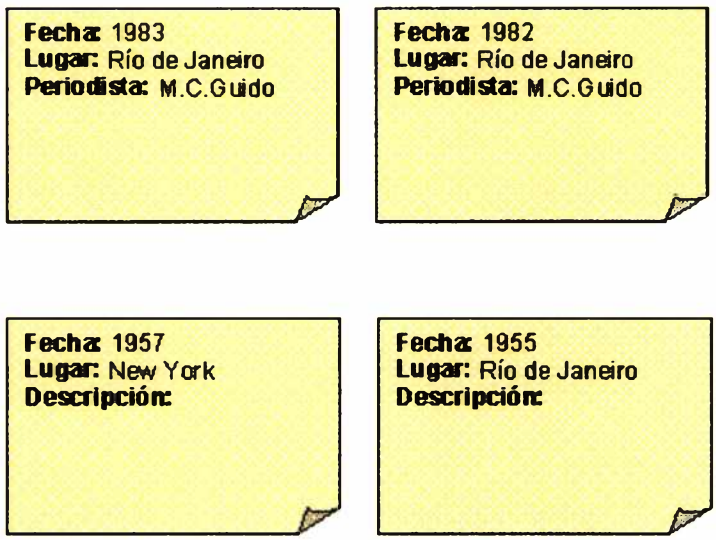


Figura 3.4: Entrevistas y Fotos

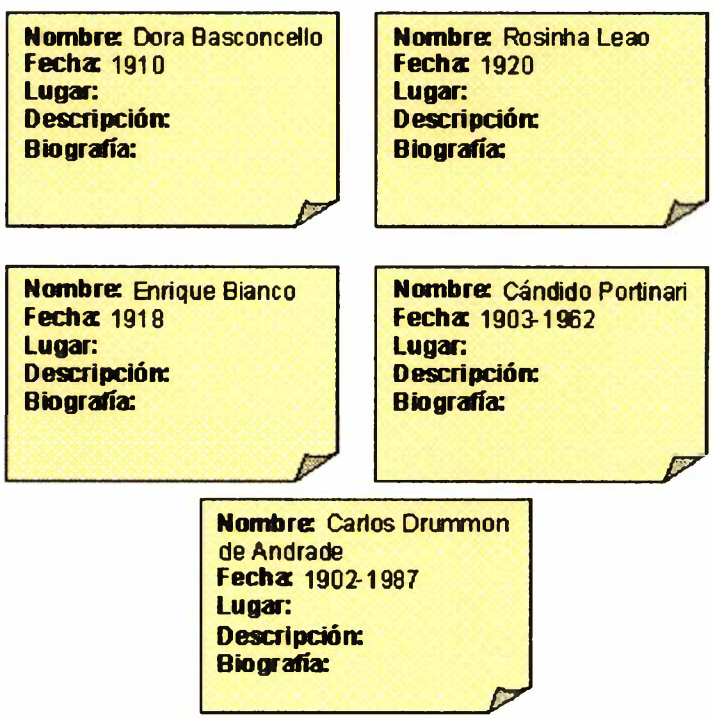


Figura 3.5: Nodos Persona

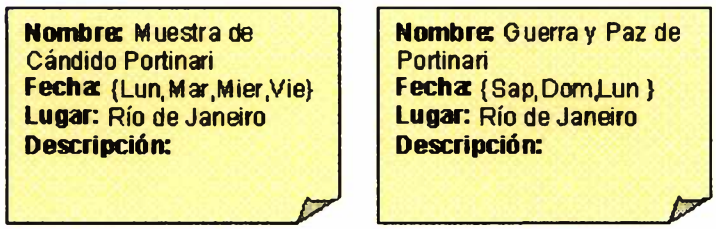


Figura 3.6: Nodos Evento

3.2 Consultas de información

Como hemos mencionado anteriormente, la información contenida dentro de una aplicación hipermedia está representada por las instancias de las clases nodos sobre las cuales el usuario navega.

Una consulta de información es aquella que se realiza a los nodos de un hipertexto, recuperando sólo aquellos que satisfacen condiciones específicas.

Selección

Esta operación permite seleccionar de un hipertexto nodos que cumplan una condición específica. La sintaxis de esta operación se define del siguiente modo:

SELECT ClaseNodo$_1$, ..., ClaseNodo$_n$

FROM *nodo$_1$*: ClaseNodo$_1$,..., *nodo$_m$*: ClaseNodo$_m$ **IN** hipertextin

[WHERE condición**]**

Donde:

ClaseNodo$_i$: son las clases de nodos pertenecientes al hipertexto *hipertextin* que conforman el hipertexto resultado de la consulta. 1< = i <= n.

nodo$_k$: variable sobre la cual se ligan en forma dinámica cada uno de los nodos pertenecientes a la clase *ClaseNodo$_k$* y sobre la cual se evalúa la condición de la cláusula WHERE. 1 < = k <= m, n <= m

hipertextin: es una variable que contiene al hipertexto de entrada sobre el cual se aplica la selección o simplemente otra selección.

condición: es una combinación de operandos y operadores que retornan un valor de verdad True o False. Donde: *operando* es un atributo de un objeto, un valor atómico o multivaluado, cualquier función que arroje resultado booleano u otra condición y *operador* es un operador de comparaciones escalares (>, <, >=, <=, =), o simplemente un operador relacional tal como NOT, AND, OR.

Los corchetes que encierran la cláusula WHERE indican que ésta es opcional.

El resultado de la operación de "Selección" devuelve aquellos nodos que cumplen con la

condición especificada y los links que hay entre ellos, formando un subhipertexto del hipertexto fuente consultado.

Enunciemos un ejemplo de selección:

Recuperemos las obras del hipertexto Portinari que fueron realizadas entre los años 1945 y 1956.

```
Q1 := SELECT Obra
      FROM o: Obra IN Portinari
      WHERE o.fecha >= "1945" and o.fecha <= "1956"
```

Observemos que Q1 es un subhipertexto del hipertexto *Portinari* que sólo posee nodos pertenecientes a la clase *Obra*. Cada nodo o del hipertexto de entrada es evaluado para determinar si su fecha de creación se encuentra dentro del intervalo especificado; si es así el nodo o estará incluido dentro del hipertexto Q1.

El hipertexto resultante está representado por las obras que componen la Figura 3.8 en la cual sólo hemos referenciado el nombre de la Obra y la fecha de realización de la misma con el fin de no distraer al lector repitiendo todos los valores de los atributos de cada instancia dado que los mismos fueron detallados en la Figura 3.3 a 3.6 inclusive.

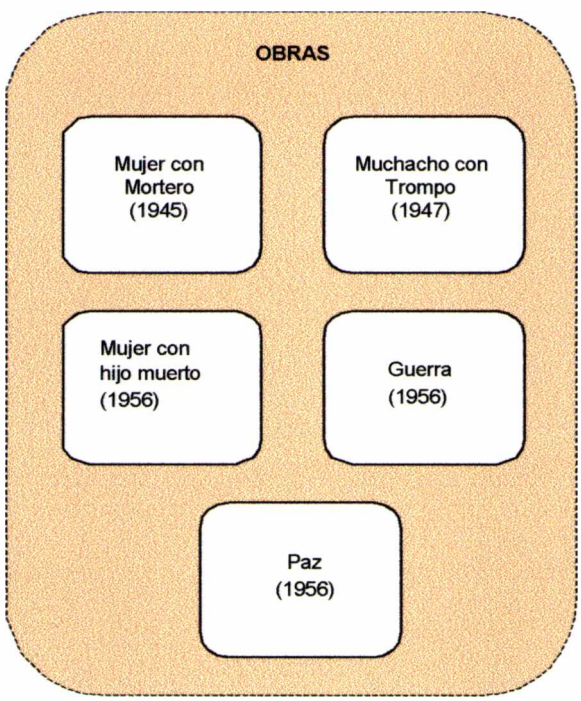


Figura 3.8: Obras realizadas entre 1945 y 1956

Consideremos otro ejemplo de selección: Obtener las obras realizadas entre los años 1945 y 1956, y los eventos que se realizaron durante el mismo período.

```
Q2 := SELECT Obra, Evento
      FROM o: Obra, e: Evento IN Portinari
      WHERE (o.fecha = > "1945" AND o.fecha = < "1956") OR
            (e.fecha = > "1945" AND e.fecha = < "1956")
```

El hipertexto Q2 se compone de nodos de la clase *Obra* y nodos de la clase *Evento*. Cada nodo obra o es evaluado independientemente de cada nodo evento e. Como mencionamos en la definición de la operación de selección, las relaciones existentes entre los nodos seleccionados se conservan dentro del hipertexto resultado.

Si observamos la figura 3.2 los nodos *Obra* que cumplen la condición de la cláusula WHERE son aquellos cuyo título es *"Paz"*, *"Mujer con hijo muerto"*, *"Guerra"*, *"Muchacho con trompo"* y *"Mujer con Mortero"* y el nodo evento es aquel que posee el nombre *"Guerra y Paz de Portinari"*.

Podemos notar también que entre los nodos *Obra y Evento* existe un link llamado *"Exhibida_en"* el cual forma parte del resultado de la consulta. Veamos la representación gráfica del resultado.

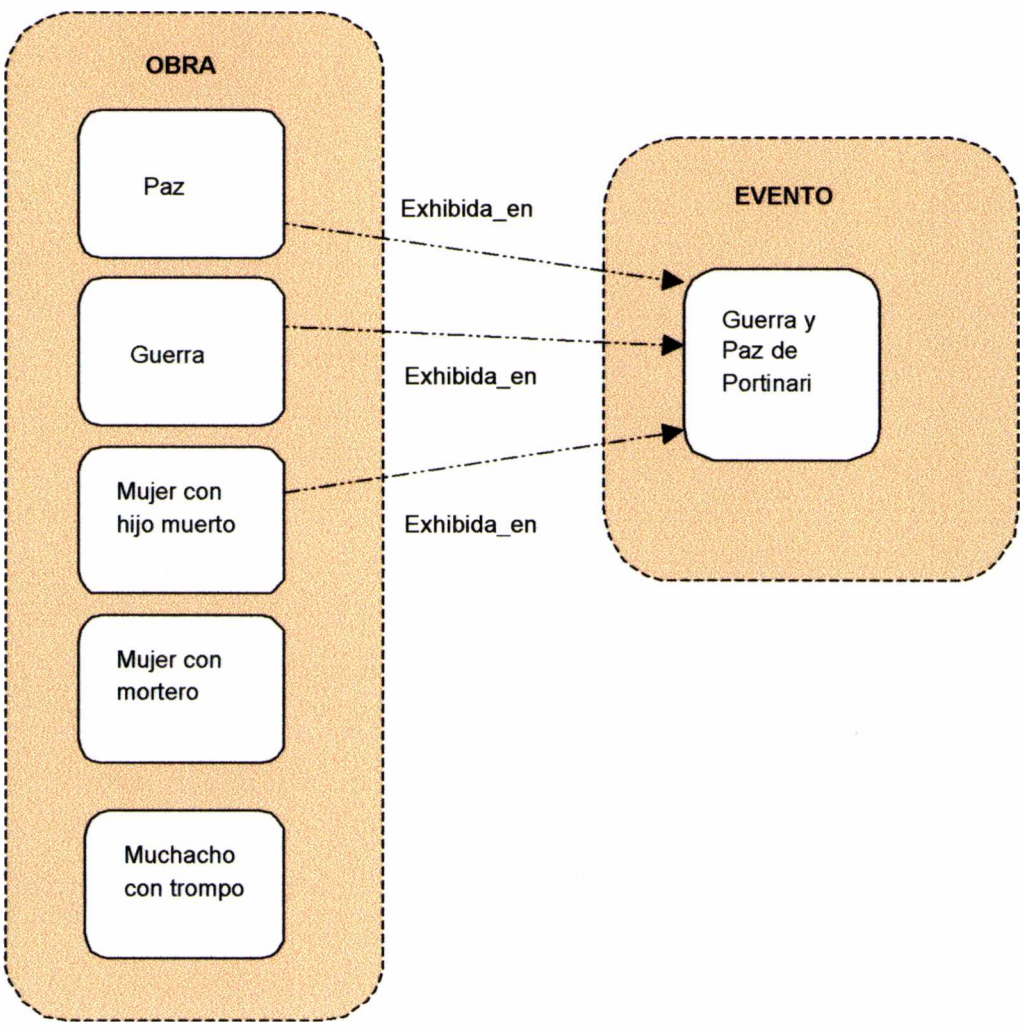


Figura 3.9: Hipertexto de Obras y Eventos realizados entre 1945 y 1956

El resultado de la consulta Q2 es un subhipertexto del hipertexto Portinari, compuesto de los nodos que cumplen la condición de la selección y las relaciones que hay entre los mismos. El lector podrá verificar que las figuras 3.8 y 3.9 están totalmente incluidas dentro de la gráfica de la figura 3.2, asegurando que el hipertexto original posee los subhipertexto generados anteriormente.

Funciones

El lenguaje provee un conjunto de funciones que pueden formar parte de las condiciones de la cláusula WHERE, con el fin de enriquecer la especificación de las consultas. Definimos funciones para manipular nodos y relaciones dentro de una hipermedia.

Hay_Relacion:

Esta función permite verificar la existencia de una relación específica entre dos nodos. Arroja un resultado booleano: True si la relación existe y False en caso contrario.

Hay_Relacion (nodo*1*, nombre_relación, nodo*2*)

Donde:

nodo1 y *nodo2* son nodos dentro de un hipertexto, *nodo1* es el nodo origen de una relación dada y *nodo2* es el nodo destino de la misma.

nombre_relación es el nombre de una relación o link entre dos nodos.

Ejemplificamos el uso de la función Hay_Relación en una consulta:

Queremos obtener las obras que fueron exhibidas en el evento *"Guerra y Paz, de Portinari"*.

```
Q3 := SELECT Obra
      FROM o: Obra, e: Evento IN Portinari
      WHERE  Hay_Relacion (o, "Exhibida_en", e) AND
                  (e.nombre = "Guerra y Paz de Portinari")
```

Si evaluamos la condición, la función *Hay_Relación* devolverá True o False de acuerdo a si existe una instancia de la relación *"Exhibida_en"* entre cada obra *o* y el evento e. Así, si la función devuelve True y el evento es el especificado (en este caso el evento es aquel cuyo atributo nombre es "Guerra y Paz de Portinari"), la obra estará incluida entre las obras del hipertexto resultado de la consulta.

Para una mayor comprensión del ejemplo, en la figura 3.2 podemos ver claramente los nodos *Obra* que están unidos a través de la relación *"Exhibida_en"* con el nodo *Evento* de nombre "Guerra y Paz de Portinari". Como resultado obtendremos entonces el hipertexto representado en la figura 3.10

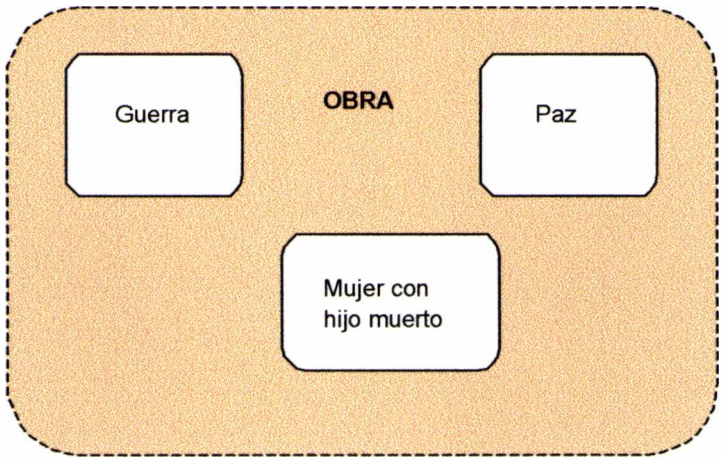


Figura 3.10: Obras exhibidas en el evento "Guerra y Paz, de Portinari"

Camino:

Así como la función *"Hay_Relacion"* nos permite saber si existe una relación entre dos nodos, la función *"Camino"* verifica la existencia de un camino entre dos nodos conectados o no en forma directa. De este modo, podemos saber si es posible llegar de un nodo a otro recorriendo las relaciones indicadas y atravesando los nodos intermedios ligados por las mismas. El resultado es booleano: True si algún camino existe y False si no existe ninguno.

Camino (nodo1, lista_relaciones, nodo2)

Donde:

nodo1 es el nodo origen del camino y *nodo2* es el nodo destino del mismo dentro de un hipertexto.

lista_relaciones: es una lista de *n* nombres de relaciones de la forma "R_1,R_2,..,R_n", donde R_1 es la primera relación a atravesar en la búsqueda del camino, R_2 es la segunda, y así sucesivamente.

Ejemplo del uso de la función Camino dentro de una consulta:

Recuperemos las personas que expusieron sus obras en el evento '*Guerra y Paz, de Portinari*'.

```
Q4 :=  SELECT Persona
       FROM p: Persona, e: Evento IN Portinari
       WHERE Camino(p,'Posee', 'Exhibida_en',e) AND
                    (e.nombre='Guerra y Paz de Portinari')
```

El resultado consta de los nodos Persona correspondientes a Dora Basconcello y Rosinha Leao, es decir que hemos efectuado el siguiente recorrido del hipertexto a través de la función Camino: posicionados en el nodo *Persona* cuyo nombre es "Rosinha Leao" atravesamos el link "Posee" y llegamos al nodo *Obra* titulado "Paz"; una vez allí recorremos el link "Exhibida_en" y alcanzamos el nodo *Evento* "Guerra y Paz de Portinari", por lo tanto concluimos que el nodo origen pertenece al resultado de la consulta. Este análisis se realiza sucesivamente con cada nodo *Persona* del hipertexto atravesando cada uno de sus links, quedando seleccionado además el nodo *Persona* de nombre "Dora Basconcello".

Veamos en forma gráfica la existencia del camino definido por las relaciones "Posee" y "Exhibida_en".

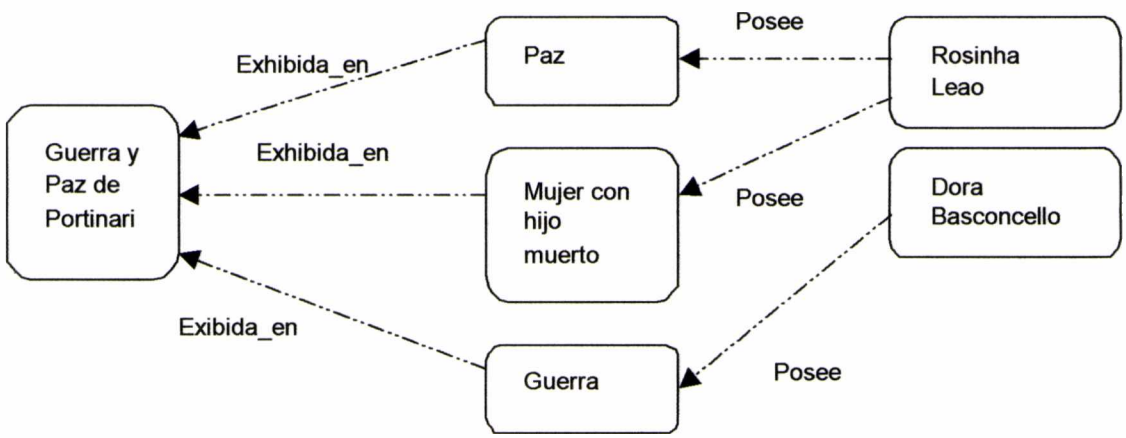


Figura 3.11: Nodos Persona, Obra y Evento involucrados en la consulta.

La función *Camino* es una simplificación de la función *Hay_Relación*.

Veamos con un ejemplo genérico como podemos verificar la existencia de un camino entre dos nodos usando la función *Hay_Relación*.

Hay_Relacion (nodo$_1$,"relacion$_1$",nodo$_2$) AND
Hay_Relacion (nodo$_2$,"relacion$_2$",nodo$_3$) AND
.....
Hay_Relacion (nodo$_{n-1}$,"relacion $_{n-1}$",nodo$_n$)

Si cada función *Hay_Relacion* retorna True entonces el resultado dará True, por lo tanto en ese caso se hallará un camino entre nodo$_1$ y nodo$_n$ definido por las relaciones relacion$_1$, relacion$_2$,...., relacion$_{n-1}$.

La función *Camino* nos retornará el mismo resultado escribiendo simplemente

Camino (nodo$_1$,"relacion$_1$, relacion$_2$, ..., relación $_{n-1}$",nodo$_n$), y sin tener que conocer ni mencionar los nodos intermedios que definan un camino.

Nota: *Camino*(nodo$_1$,R$_1$,nodo$_2$) = *Hay_Relación* (nodo$_1$,R$_1$,nodo$_2$)

Presentamos a continuación una función que permite comprobar si un nodo forma parte de un hipertexto.

Esta_en:

Esta función permite verificar la existencia de un nodo particular dentro de un hipertexto.

Si el nodo se encuentra dentro del hipertexto consultado retornará True y en caso contrario el resultado será False.

Esta_en (Nodo,Hipertext)

Donde :

Nodo: es un nodo cualquiera.

Hipertext : es una variable que referencia a un hipertexto o simplemente otra selección.

Ejemplo del uso de la función Esta_en dentro de una consulta:

Obtener las obras fotografiadas en el año 1810 y las personas retratadas en dichas obras.

```
Q5:= SELECT Obra, Persona
     FROM o: Obra, p: Persona IN Portinari
     WHERE  Hay_Relacion (p, "Retratada_en", o) AND
     Esta_en (o, (SELECT Obra
                  FROM o1: Obra, f: Foto IN Portinari
                  WHERE Hay_Relacion(o1,"Fotografiada_en",f)))
```

La consulta Q5 utiliza dentro de la cláusula WHERE la función Esta_en para verificar por cada nodo *o* la pertenencia de éste dentro de otro hipertexto obtenido por medio de una subconsulta.

El mismo resultado puede ser logrado de la siguiente forma:

```
Qaux :=  SELECT Obra
         FROM o1: Obra, f: Foto IN Portinari
         WHERE Hay_Relacion(o1,"Fotografiada_en",f)
```

```
Q7 :=  SELECT Obra, Persona
       FROM o: Obra, p: Persona IN Portinari
       WHERE  Hay_Relacion (p, "Retratada_en", o) AND
       Esta_en (o, QAux)
```

Notemos que el resultado de la consulta QAux es utilizado dentro de la función Esta_en, simplificando la escritura de la misma.

Con la misma modalidad podemos examinar si un nodo no pertenece a un hipertexto. En tal ocasión la función "*Esta_en*" se utilizará anteponiendo el operador de negación.

NOT (**Esta_en** (Nodoi,hipertext))

Debido a que la condición de la consulta puede incluir atributos multivaluados, se definen funciones para operar sobre ellos.

Existe:

Esta función verifica la existencia de un elemento dentro de un conjunto. Devuelve True si el elemento existe o False en caso contrario.

Existe (conjunto, elemento)

Donde:

conjunto : es un conjunto de valores.

elemento : elemento del conjunto que se quiere verificar.

Ejemplo de la utilización de la función Existe dentro de una consulta:

Obtener las obras que se pueden visitar los días "Lunes".

```
Q8 :=  SELECT Obra
       FROM o:Obra , e:Evento IN Portinari
       WHERE Hay_Relacion(o, "Exhibida_en",e) AND
       Existe  (e.dia_de_visita, "Lunes")
```

Si observamos las figura 3.2 y 3.5 simultaneamente, los nodos Obra que retornará la consulta Q8 serán todas las obras del hipertexto Portinari, ya que las dos instancias de eventos que la misma posee mencionan dentro de su atributo dia_de_visita al día "Lunes".

Incluido:

Esta función nos permite verificar si un conjunto está incluido en otro.

Incluido (conjunto1, conjunto2)

Donde:

conjunto1 es el conjunto a comparar y *conjunto2* es el conjunto sobre el cual se testea la condición de inclusión.

El resultado será un booleano, True si todos los elementos del *conjunto1* están contenidos en el *conjunto2*, en caso contrario retornará False.

Ejemplo del empleo de la función Incluido dentro de una consulta:

Obtener las obras donde los colores con que fueron pintadas se encuentran incluidos en el conjunto {azul, amarillo, rojo}.

```
Q9 := SELECT Obra
      FROM o: Obra IN Portinari
      WHERE Incluido (o.colores, {azul, amarillo, rojo})
```

El hipertexto resultado Q9 se muestra en la figura 3.12, el cual como podemos observar en la figura 3.3 contiene sólo los nodos *Obra* en cuya realización se han utilizado sólo los colores del conjunto especificado. Como sabemos, la obra "*Mujer con Mortero*" se confeccionó en color amarillo en tanto la obra "*Paz*" fue pintada utilizando una paleta de colores con los tres colores indicados: azul, amarillo y rojo.

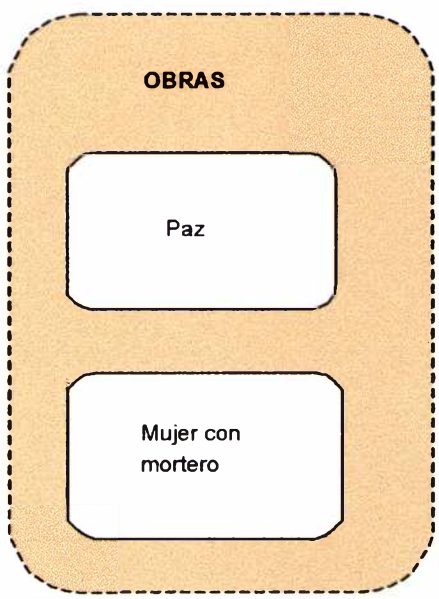


Figura 3.12: Obras en las que se usaron los colores azul, amarillo y rojo.

Proyección

Hasta el momento hemos recuperado nodos y links en las consultas realizadas conformando hipertextos. Para obtener sólo valores de algunos atributos de los nodos definimos la operación de proyección.

Formalmente la proyección se define del siguiente modo:

PROYECCION $a_1,...,a_n$ **IN** claseNodo1

$b_1,...,b_m$ **IN** claseNodo2

.....

$f_1,.....f_s$ **IN** claseNodoz

FROM hipertext

INTO lista

Donde:

a_i es el atributo *i* perteneciente a la clase nodo *claseNodoi*.

hipertext: puede ser una variable que contiene al hipertexto de entrada sobre el cual se aplica la proyección o simplemente otra selección.

lista: lista de valores de los atributos de los diferentes nodos pertenecientes a *hipertext*.

Por lo tanto, la información devuelta ya no serán nodos y links como en una SELECT sino que se reducirá a una lista con los valores de los atributos especificados de los nodos del hipertexto fuente.

Ejemplo del uso de la Proyección:

Obtener los nombres y el tema de las obras que se expusieron en el evento "*Guerra y Paz de Portinari*".

```
PROYECCION título, tema IN Obra
FROM (SELECT Obra
FROM o: Obra, e: Evento IN Portinari
WHERE Hay_Relación (o, "Exhibida_en",e) AND
        e.nombre="Guerra y Paz, de Porinari".)
INTO listaAtributos
```

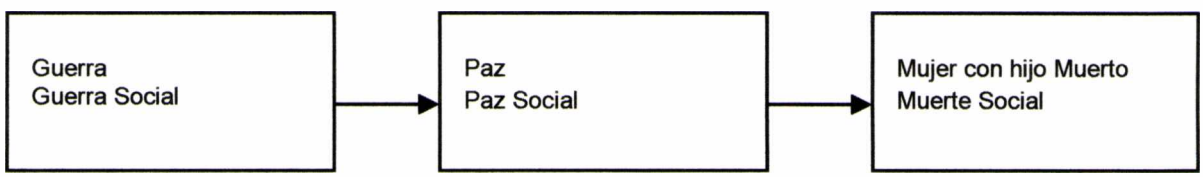


Figura 3.13: Resultado de la proyección

La selección encontrada luego de la palabra reservada FROM retorna un hipertexto con las obras que se expusieron en el evento "*Guerra y Paz de Portinari*".

La figura 3.13 exhibe los valores de los atributos proyectados título y tema de cada obra que fue exhibida en el evento "Guerra y Paz de Portinari" y los devuelve en forma de una lista de valores.

En algunos casos la información que necesitamos puede encontrarse distribuida en distintos hipertextos, o por el contrario, sólo comprende parte de un hipertexto. Así surge naturalmente la definición de las operaciones de unión y diferencia entre hipertextos.

Unión

Esta operación permite unir la información contenida en dos hipertextos que poseen esquemas idénticos. Dos hipertextos tienen esquemas idénticos cuando todas sus clases son compatibles, es decir poseen el mismo comportamiento, el mismo nombre y los mismos atributos; y estos atributos tienen el mismo dominio.

La sintaxis de la operación de unión entre hipertextos se define como sigue:

Hiper*1* **UNION** Hiper*2*

Donde:

Hiper1, Hiper2: variables que referencian los hipertextos a unir.

El hipertexto que resulta de unir dos hipertextos que poseen esquemas idénticos será un hipertexto con los nodos pertenecientes a cada uno de ellos, incluyendo las relaciones que existen en los hipertextos originales.

Confeccionemos un ejemplo del uso de la operación Unión:

Realizaremos dos consultas por separado y luego uniremos la información obtenida de ellas.

Consulta1: Obtener las personas que fueron fotografiadas en Río de Janeiro y las obras en las que dichas personas fueron retratadas.

```
Q10 :=  SELECT Persona, Obra
        FROM p: Persona, o: Obra, f: Foto IN Portinari
        WHERE (Hay_Relacion (p," Retratada_en", o) AND
                Hay_Relacion (f, "Muestra_Persona", p) AND
                f.lugar= "Río de Janeiro"
```

El resultado de esta consulta será el hipertexto compuesto por los siguientes nodos.

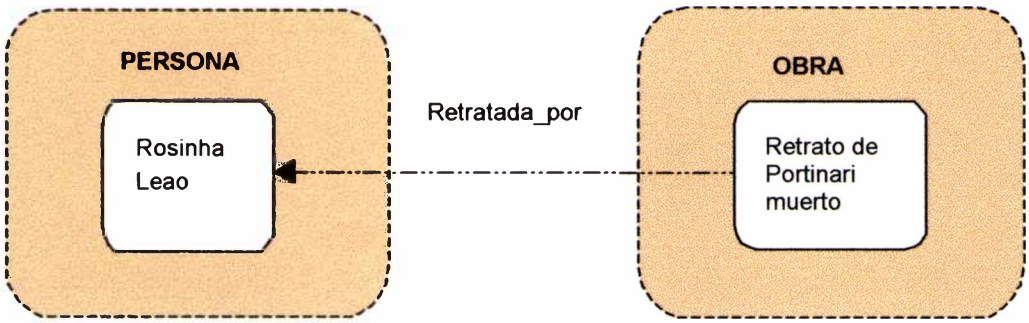


Figura 3.14: Hipertexto resultado de la consulta 1.

Consulta2: Obtener las personas que fueron entrevistadas en Río de Janeiro y las obras de dicha persona posee.

```
Q11:= SELECT Persona, Obra
      FROM p: Persona, o: Obra, e: Entrevista IN Portinari
      WHERE (Hay_Relacion (p, "Posee", o) AND
            Hay_Relacion (p, "Otorga", e) AND e.lugar="Rio de Janeiro"
```

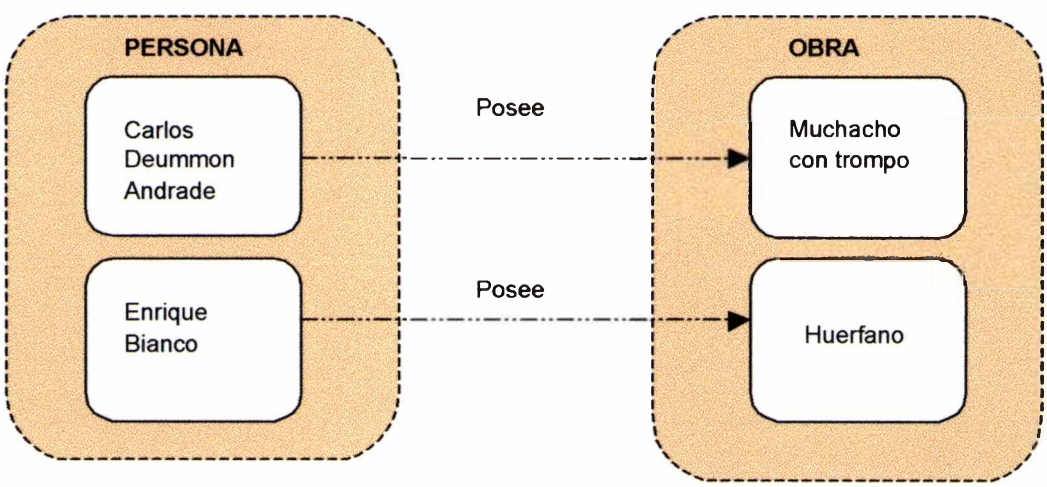


Figura 3.15: Hipertexto resultado de la consulta 2.

Si quisiéramos obtener las personas que fueron fotografiadas o entrevistadas en Río de Janeiro, podríamos utilizar la información de los hipertextos obtenidos en las consultas previas y unirlos. Por lo tanto la consulta resultaría de la forma:

QUnión:= Q10 UNION Q11

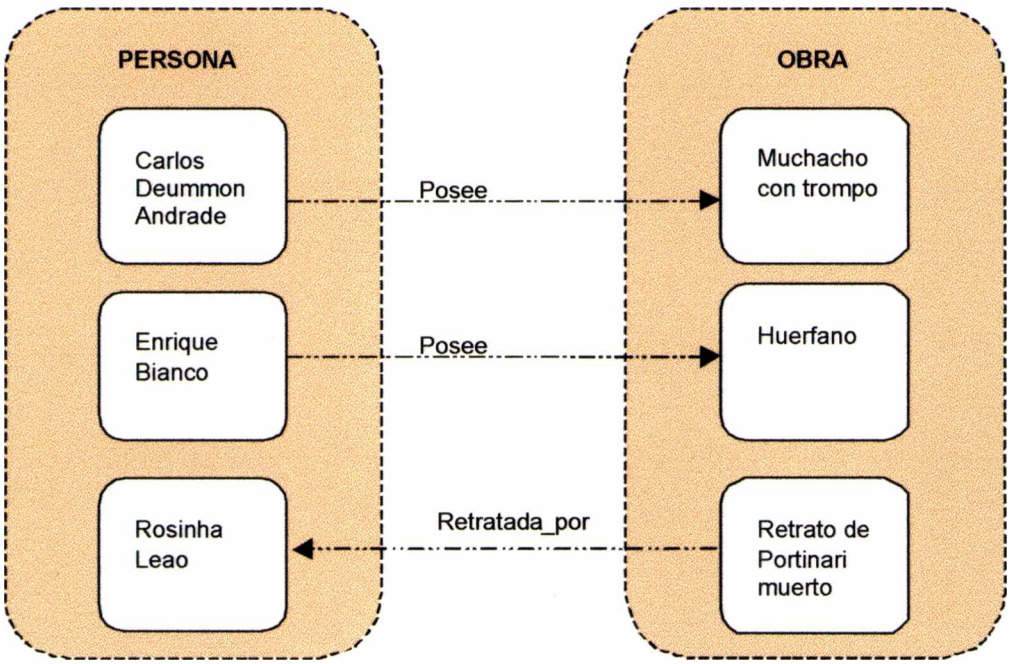


Figura 3.16: Unión de hipertextos.

Notemos que el resultado incluye los nodos de ambos hipertextos junto con las relaciones que poseen en los mismos.

Analicemos el resultado de la UNION cuando alguno de los hipertextos es vacío.

Hiper1 UNION Hiper2

Si *Hiper1* es vacío el resultado será *Hiper2*.

Si *Hiper2* es vacío el resultado será *Hiper1*.

Si *Hiper1* e *Hiper2* son vacíos el resultado será el hipertexto vacío.

Diferencia

Esta operación permite eliminar de un hipertexto aquellos nodos que se encuentran en otro hipertexto. Del mismo modo que en la unión los hipertextos que intervienen en esta operación deben poseer esquemas idénticos.

La forma de definir la operación de diferencia entre hipertextos es la siguiente:

Hiper$_1$ **DIFERENCIA** Hiper$_2$

Donde:

Hiper$_1$, Hiper$_2$: variables que referencian los hipertextos a diferenciar.

Ejemplo:

Para ejemplificar la Diferencia vamos a realizar dos consultas por separado, de forma tal de obtener cierta información, luego restaremos dicha información analizando el resultado.

Consulta1: Por un lado, nos interesa obtener todas las personas, obras y eventos del hipertexto Portinari. Para rescatar dicha información la consulta a realizar será:

Q12 := SELECT Persona, Obra, Evento

FROM p: Persona, o: Obra, e: Evento IN Portinari

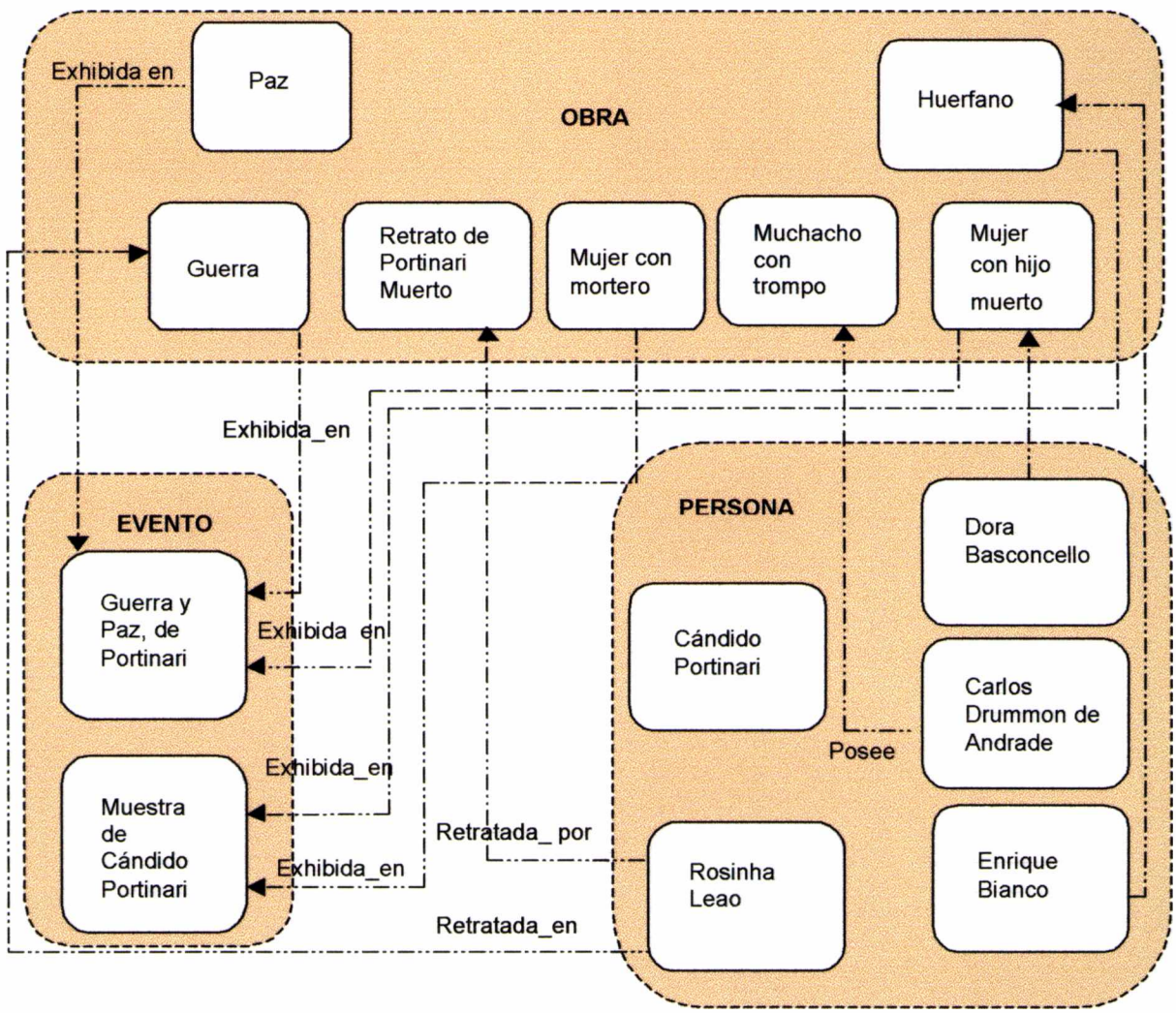


Figura 3.17: Hipertexto Q12

La figura 3.17 muestra un subhipertexto del hipertexto que contiene la figura 3.2, el cual posee sólo la información de las obras, eventos y personas rescatadas del hipertexto Portinari.

Consulta2: Obtener el evento *"Muestra de Cándido Portinari"*, las obras que fueron exhibidas en el mismo y, las personas que poseen dichas obras.

```
Q13 := SELECT Persona, Obra, Evento
       FROM p:Persona, o: Obra, e: Evento IN Portinari
       WHERE e.Nombre = "Muestra de Cándido Portinari"
             AND ((Hay_Relación (e, "Exhibida_en",o)
             AND Hay_Relación (o,"Posee",p))
             OR  (Hay_Relación (o,"Exhibida_en",e))
```

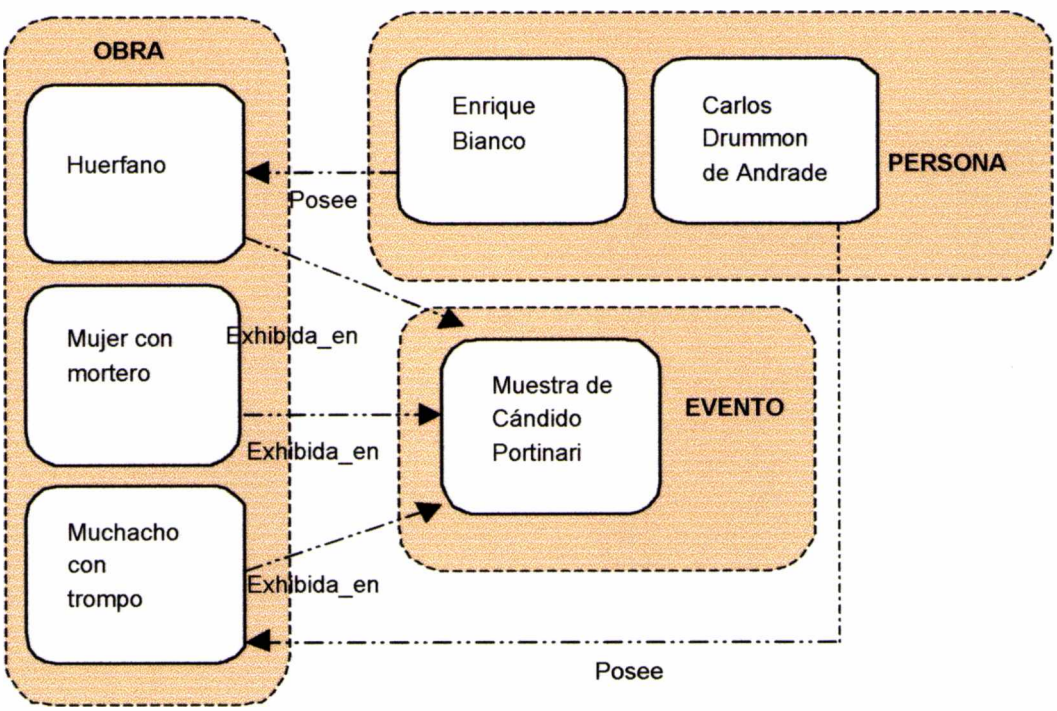


Figura 3.18: Hipertexto Q13

Veamos entonces cuál es el resultado al diferenciar los hipertextos previamente obtenidos:

```
QDiferencia:=   Q12 DIFERENCIA Q13
```

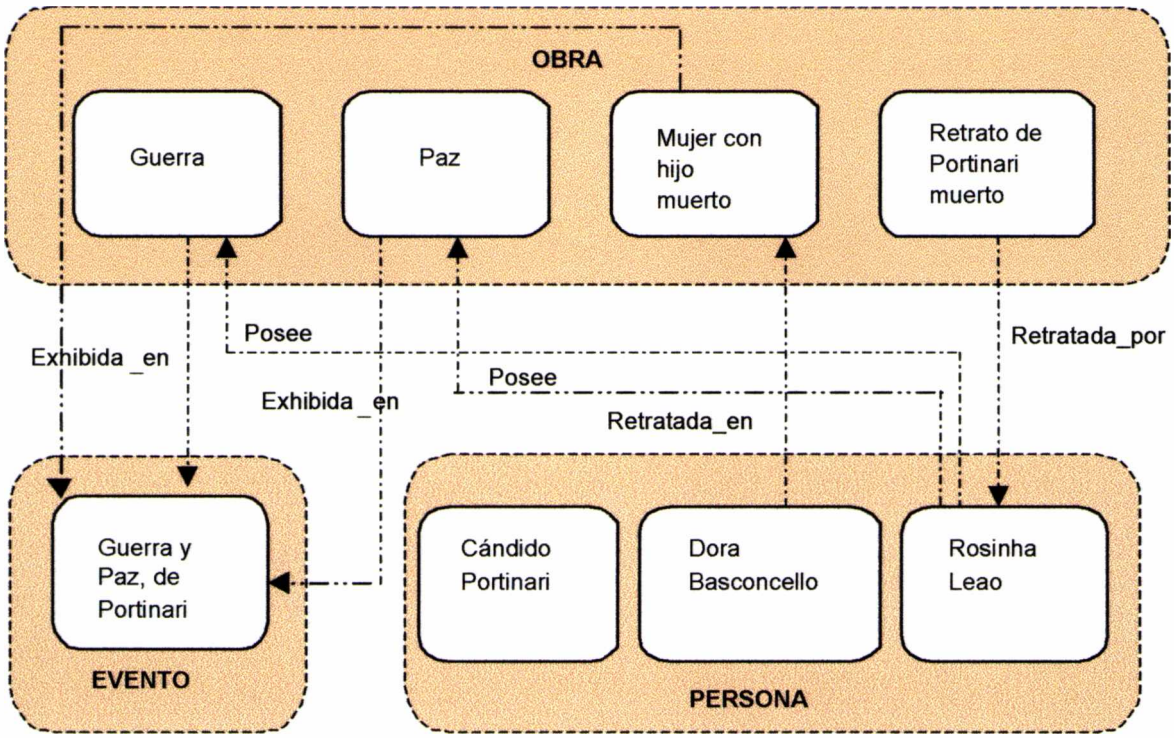


Figura 3.19: Diferencia entre dos hipertextos.

Siguiendo la definición de esta operación comprobamos que el hipertexto resultante de la operación DIFERENCIA está compuesto por aquellos nodos pertenecientes al hipertexto Q12 sin incluir los que forman parte del hipertexto Q13, como tampoco las relaciones que los mismos poseían.

Observemos que la obra titulada "Huérfano" no se encuentra en el hipertexto resultado, debido a que ésta forma parte de Q12 y Q13. Sin embargo la obra cuyo título es "Paz" sí se encuentra en el hipertexto Qdif dado que este se encuentra en Q12 y no en Q13

Si realizamos la diferencia inversa, es decir que al hipertexto Q13 le restamos el hipertexto Q12, el resultado en este caso será vacío.

Analicemos ahora el resultado de la operación DIFERENCIA cuando uno de los hipertextos es vacío:

Qdif := Hiper1 DIFERENCIA Hiper2

Si *Hiper1* es vacío el resultado será el hipertexto vacío.

Si *Hiper2* es vacío el resultado será el hipertexto *Hiper1*.

Si *Hiper1* e *Hiper2* son vacíos el resultado será el hipertexto vacío.

3.3 Constructores de Navegación

En la sección anterior definimos operaciones que permiten recuperar cierta información dentro de una hipermedia, siendo el resultado de casi todas ellas otra hipermedia.

Hasta el momento, el usuario puede recuperar la información deseada pero no tiene manera de recorrerla, debido a que las formas de navegar definidas en la hipermedia original ya no son aplicables dado que ha variado el contenido de la hipermedia.

Por lo tanto el usuario deberá definir como acceder al hipertexto recuperado, pudiendo detallar las clases participantes, el orden entre los objetos de esa clase, especificando el criterio exacto de navegación.

A continuación definimos constructores que permiten diseñar la navegación en el resultado de las consultas, cuando este resultado sea una hipermedia.

Contexto

Este constructor permite definir la forma de navegar la información de la hipermedia obtenida como resultado de una consulta. En particular los nodos de cada clase se recorrerán en forma completa.

CONTEXTO (Hipertexto: *HipertextIn*

Objetos que incluye: v_1: $clase_1, \ldots, v_n$: $clase_n$

Puntos de entrada: *nodo / estructura de acceso*

Estrategia: *Circular / Secuencial (lista de clases)*

Orden: v_1.*atributo orden*$_1$, ..., v_n.*atributo orden*$_n$

Cambio de contexto: $cont_1, \ldots, cont_k$ **)**

Donde:

HipertextIn: es un hipertexto representado a través de una variable o definido explícitamente por medio de una consulta, sobre el cual se crea el contexto navegacional.

v_k: variable sobre la cual se ligan en forma dinámica cada uno de los nodos pertenecientes a las clases de nodo $clase_k$, donde $1 <= k <= n$, siendo n el número de clases que contiene el hipertexto *HipertextIn*.

Puntos de entrada: puede ser único o múltiple. En el caso de ser único, este ítem contendrá el nodo por el que se accede al contexto y será obligatoria la entrada al mismo por el nodo especificado. En el caso de ser múltiple, se deberá especificar la estructura de acceso asociada.

Nodo: es especificado cuando el punto de entrada al contexto es único. Formalmente se especifica: v_k.*atributo* = valor, siendo *atributo* un atributo de la clase $clase_k$.

Estructura de acceso: referencia a la estructura de acceso (definida mas adelante) que contiene los diferentes puntos de entrada por medio de los cuales se puede acceder al contexto.

Estrategia : especifica cómo será la navegación dentro del contexto. La notación formal

es secuencial / circular (lista de clases, donde el orden de la enumeración de las mismas determina la estrategia entre clases).

Orden: especifica el criterio de ordenación de los objetos de cada clase. Donde:

$v_i.atributo$ es el atributo por el que serán ordenados los nodos de la clase $clase_i$. $Orden_i$ puede ser ascendente (ASC) o descendente (DESC).

$cont_k$: variable que representa un contexto. Por medio estas variables se especifica a qué contextos se puede cambiar desde el contexto actual. Los contextos $cont_k$ deben haber sido generados utilizando el hipertexto *HipertextIn*.

El cambio de contextos sólo podrá producirse cuando, dado un nodo *n* y dos contextos *c1* y *c2*, *n* pertenece a ambos y en la definición del contexto *c1* esta definido *c2* en el *Cambio de contexto* y/o viceversa.

Si nos encontramos navegando el contexto *c1* y alcanzamos dentro de éste al nodo *n*, tenemos la posibilidad de continuar navegando por el contexto *c2* a partir del nodo *n*, siguiendo la *Estrategia* y el *Orden* de *c2* sin tener en cuenta el *Punto de entrada* de c2.

Veamos ejemplos creando primero los distintos contextos y observando luego la manera de cambiar de contextos:

Ejemplo 1:

Recuperemos en un hipertexto las obras de la aplicación Portinari.

```
Q14 := SELECT Obra
       FROM o: Obra IN Portinari
```

Una vez rescatada la información en el hipertexto Q14 generemos un contexto sobre el mismo.

```
Cont1 := CONTEXTO (Hipertexto: Q14
                   Objetos que incluye: o:Obra
                   Punto de Entrada: o.título = "Mujer con mortero"
                   Estrategia: circular, (Obra)
                   Orden: o.fecha ASC
                   Cambio de contexto: )
```

Una vez creado el contexto, podremos navegar entre las obras como lo indica la figura 3.20.

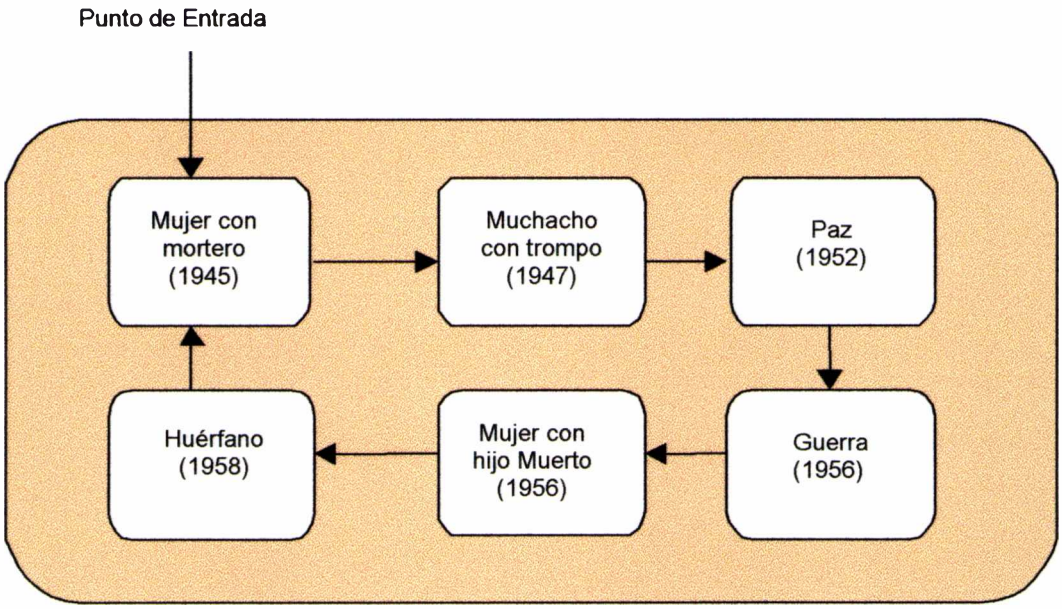


Figura 3.20: Contexto de Obras Cont1

Notemos que el punto de entrada es único y queda definido por el nodo *obra* cuyo título es "Mujer con mortero"; a partir de ésta la navegación es circular entre las obras las cuales se encuentran ordenadas según su fecha de creación ascendentemente.

Nótese que la definición del contexto Cont1 es una de las posibles formas de navegar sobre el hipertexto Q14.

Ejemplo 2:

Obtener las obras y los eventos en que dicha obra fue exhibida que pertenecen a la aplicación Portinari. La consulta a realizar será:

Q15 := SELECT Obra, Evento

FROM o: Obra, e: Evento IN Portinari

WHERE (*Hay_Relación* (o, "Exhibida_en", e)

Una vez recuperada del hipertexto Portinari la información deseada, definamos diferentes contextos que nos permitan navegar sobre dicha información.

Cont2 := CONTEXTO (Hipertexto: Q15

Objetos que incluye: o:Obra, e:Eventos

Punto de Entrada: o.título = "Guerra"

Estrategia: circular (Obra, Evento)

Orden: o.titulo ASC, e.nombre ASC

Cambio de contexto:)

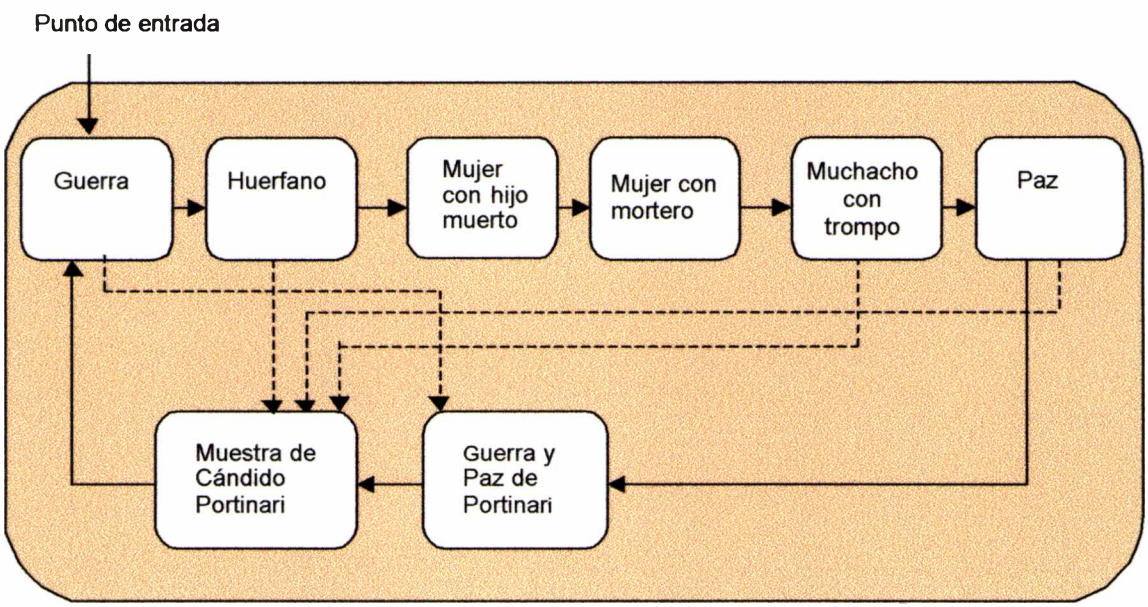


Figura 3.21: Representación gráfica del ejemplo 2

Como podemos observar, la Figura 3.21 muestra el resultado de la navegación especificada. Los nodos se recorren en forma circular, primero las obras, ordenadas por *título* ascendentemente, y luego los eventos, ordenados por *nombre*, como lo indica su estrategia. Veamos que resulta posible navegar desde una obra hacia el evento donde la misma fue exhibida, ya que ambos nodos pertenecen al contexto y el hipertexto sobre el cual está definido contiene estas relaciones.

Para diferenciar en la gráfica la estrategia y las relaciones hemos representado éstas últimas con líneas punteadas.

Ejemplo 3:

Definamos otro contexto sobre el mismo hipertexto que nos permita recorrer sólo las obras.

Cont3 := CONTEXTO (Hipertexto: Q15

Objetos que incluye: o:Obra

Punto de Entrada: o.titulo = "Guerra"

Estrategia: circular (Obra)

Orden: o.titulo ASC

Cambio de contexto:)

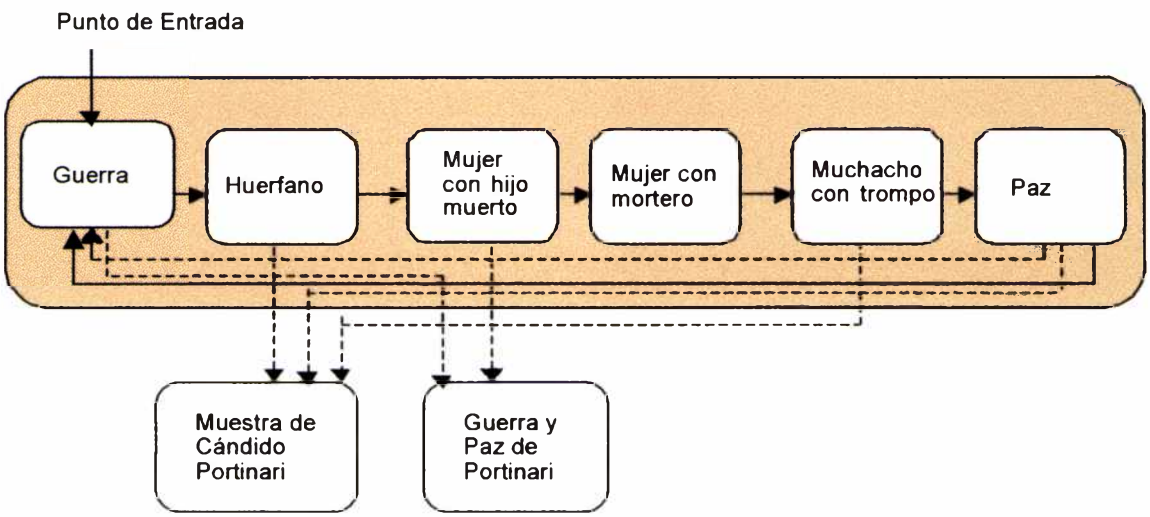


Figura 3.22: Representación gráfica del ejemplo 3.

Como podemos notar en la Figura 3.22 la navegación se realiza en forma circular sobre las obras del hipertexto Q15 cuyo punto de ingreso es la obra titulada "Guerra". Las relaciones entre las obras y los eventos se mantienen debido a que ambas clases pertenecen al hipertexto sobre el cual construimos este contexto, pero si navegamos a través de alguna de estas relaciones hacia los eventos saldremos del contexto, dado que el mismo está creado solamente sobre las obras, en consecuencia perdemos la información de navegación asociada a él.

Veamos que sucede si definimos un contexto Cont4 con las mismas características que Cont3 que nos permita realizar un cambio al contexto Cont2.

Cont4 := CONTEXTO (Hipertexto: Q15

Objetos que incluye: o:Obra

Punto de Entrada: o.titulo = "Guerra"

Estrategia: circular (Obra)

Orden: o.titulo ASC

Cambio de contexto: Cont2)

El contexto Cont4 nos permite navegar entre las obras del hipertexto Q15 ingresando por la obra titulada "Guerra" y recorriéndolas secuencialmente ordenadas por título. Como las obras de este contexto también pertenecen a Cont2 y hemos especificado en la definición de Cont4 la posibilidad de cambiar a este contexto, es posible cambiar de un contexto al otro. De esta forma podemos acceder a los eventos en que las mismas se encuentran exhibidas. Es importante destacar que una vez que nos encontramos navegando en Cont2 ya no podremos volver a Cont4 debido a que en la definición de dicho contexto no hemos indicado la posibilidad de mudanza.

La siguiente figura muestra gráficamente la posible navegación desde Cont4 a Cont2.

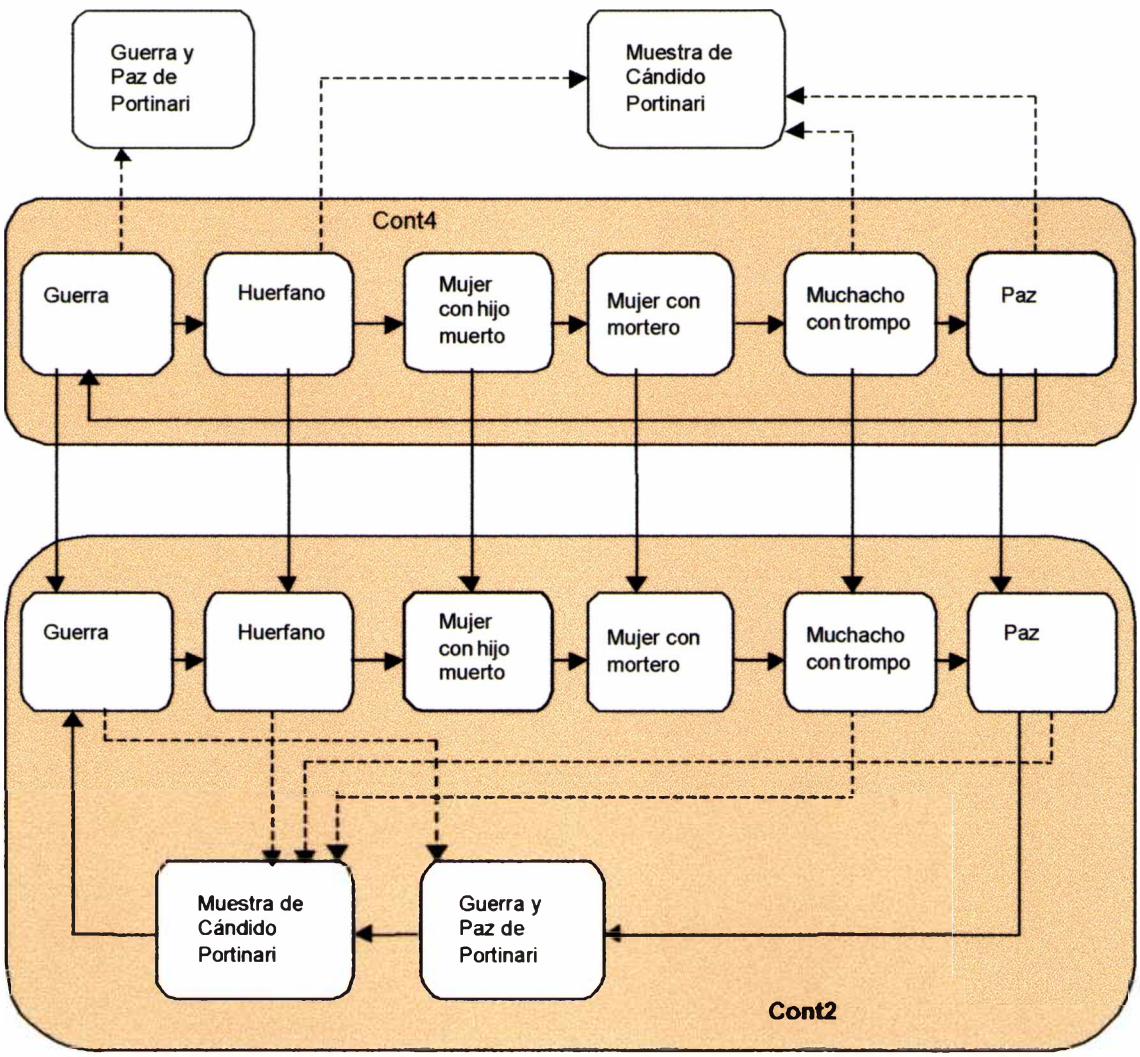


Figura 3.23: Esquema de un cambio de contexto

Tour Guiado

Es un constructor que permite diseñar el modo de navegar entre elementos arbitrarios de una hipermedia. Los elementos de un tour guiado pueden ser instancias de diferentes clases navegacionales.

TOUR_GUIADO (Objetos que incluye: $Hipertext_1$, $Hipertext_2$,.....,$Hipertext_n$
Estrategia: *Circular/Secuencial*)

Donde:

Objetos que incluye: $Hipertext_i$

$Hipertext_i$ ..$Hipertext_n$: son hipertextos representado por medio de variables o definidos explícitamente por medio de una consulta, que poseen un único elemento cada uno.

n: es la cantidad de elementos del tour, n>=1.

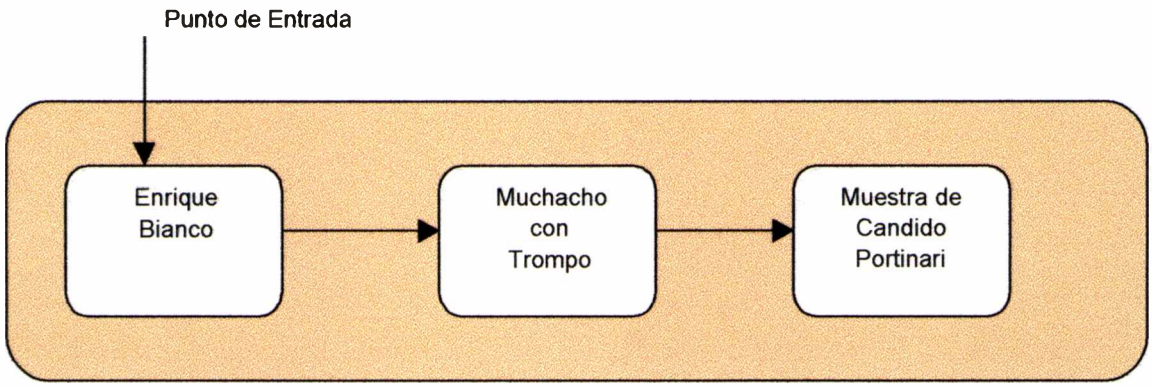
Estrategia: especifica cómo será la navegación entre los elementos. Si la estrategia es secuencial la navegación se realizará hasta el último elemento, mientras que si la estrategia es circular desde el último elemento se navega hacia el primer elemento del tour.

```
HT1 :=  SELECT Persona
        FROM  p: Persona IN Portinari
        WHERE p.nombre = "Enrique Bianco"

HT2: = SELECT Obra
        FROM o: Obra IN Portinari
        WHERE o.título = "Muchacho con Trompo"

T1:= TOUR_GUIADO (Objetos que incluye: HT1,
                                       HT2,
                                       SELECT Evento
                                       FROM e: Evento IN Portinari
                                       WHERE  e.nombre = "Muestra de Cándido
                                                         Portinari",
                  Estrategia: secuencial)
```

Los elementos del tour guiado serán los nodos contenidos en los hipertextos HT1 y HT2 y el nodo resultante de la consulta definida explícitamente dentro de los objetos que incluye T1.



Tour Guiado Ht1

Estructura_de_acceso

En los ejemplos anteriores construimos contextos navegacionales donde el punto de entrada era único, pero como sabemos es posible que este punto de entrada sea múltiple, es decir el usuario podrá seleccionar entre varias opciones para ingresar a un contexto.

La estructura de acceso nos permite organizar las diferentes posibilidades de acceso a un contexto cuando estas son múltiples.

ESTRUCTURA_DE_ACCESO (

Objetos que incluye: *v:clase$_v$ / contexto$_i$, ... ,contexto$_n$,*

estructura_de_acceso$_k$, ...estructura_de_acceso$_m$,

tour_guiado$_1$,....tour_guiado$_t$

)

Donde:

Objetos que incluye: compuesto por:

v: variable que referencia a los nodos correspondientes a la clase *clase$_v$*

contexto$_i$, estructura_de_acceso$_k$,, tour_guiado$_r$: contextos, estructuras de acceso y tours guiados que conforman la estructura de acceso en definición.

Cuando los objetos que incluye la estructura son los nodos correspondientes a una clase determinada la misma se define para ser utilizada como punto de entrada de un contexto, quedando definido este punto de entrada como múltiple. Por el contrario, el objetivo de crear estructuras cuyos componentes son contextos u otras estructuras de acceso es el de acceder directamente a los contextos especificados o a otras estructuras de acceso con un comportamiento similar a este último.

En los siguientes ejemplos veremos distintas aplicaciones de este constructor:

Ejemplo 1:

En este ejemplo definimos una estructura de acceso para acceder a través de ella a distintos contextos.

EA1:= ESTRUCTURA_DE_ACCESO (Objetos que incluye: Cont1, Cont4)

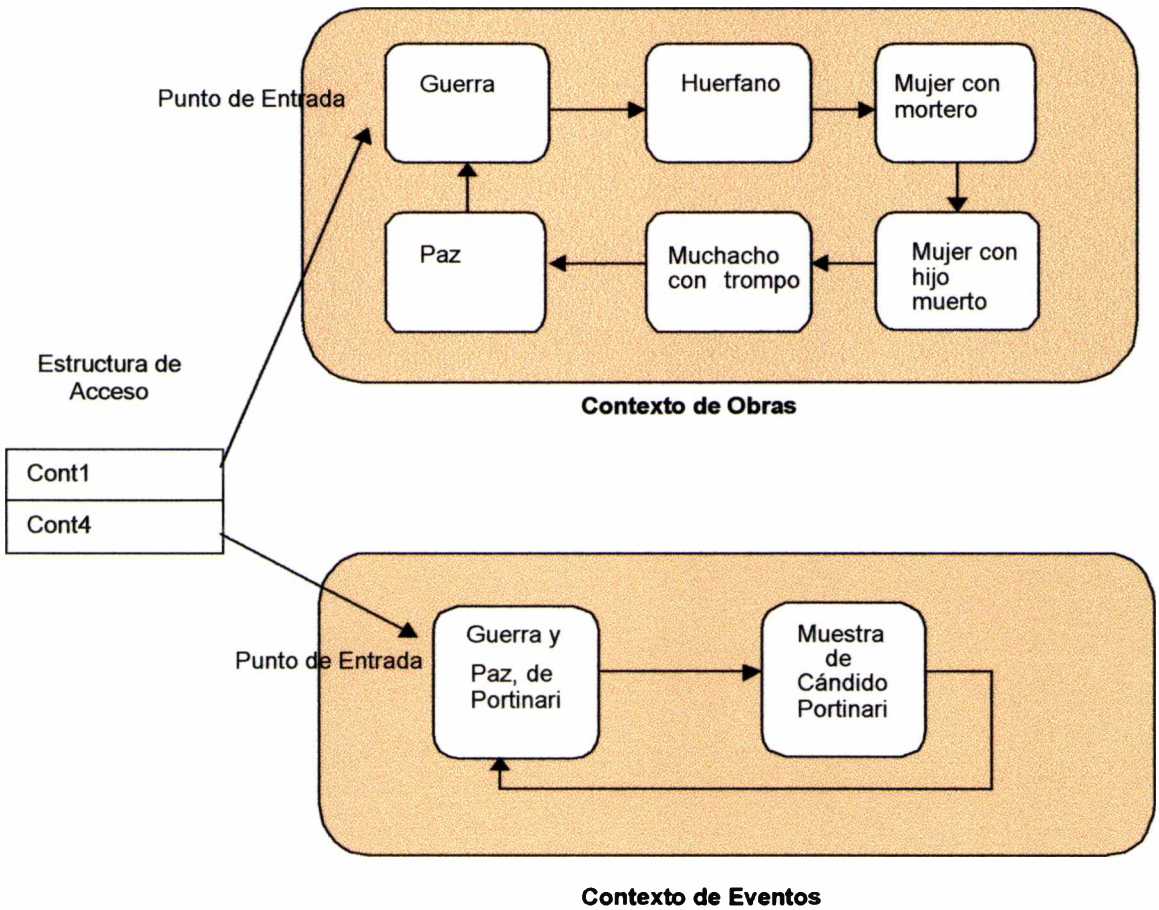


Figura 3.24: Representación gráfica de la estructura de acceso EA1

Una vez definida la estructura de acceso podemos elegir entonces qué contexto recorrer. Si seleccionamos *Cont1* podremos recorrer el contexto que nos permite ver las obras de Portinari, si seleccionamos *Cont4* podremos recorrer entonces los eventos que contiene el hipertexto especificado en *Cont4*. El acceso a cada uno de los contextos se realiza a través del punto de entrada definido en el mismo.

Ejemplo 2:

Este ejemplo utiliza la estructura de acceso para acceder directamente a elementos de un contexto compuesto por obras.

EA2 := ESTRUCTURA_DE_ACCESO (Objetos que incluye: o: Obra)

Una vez creada la estructura de acceso definimos el contexto sobre el cual la usaremos.

Recordemos que Q14 es un hipertexto que contiene todas las obras de la aplicación Portinari.

Cont5 := CONTEXTO (Hipertexto: Q14

Objetos que incluye: o: Obra

Punto de Entrada: EA2

Estrategia: circular

Orden: o. nombre ASC

Cambio de contexto:)

En la siguiente figura vemos como quedo definido el contexto Cont5 que contiene como punto de entrada a la estructura de acceso EA2.

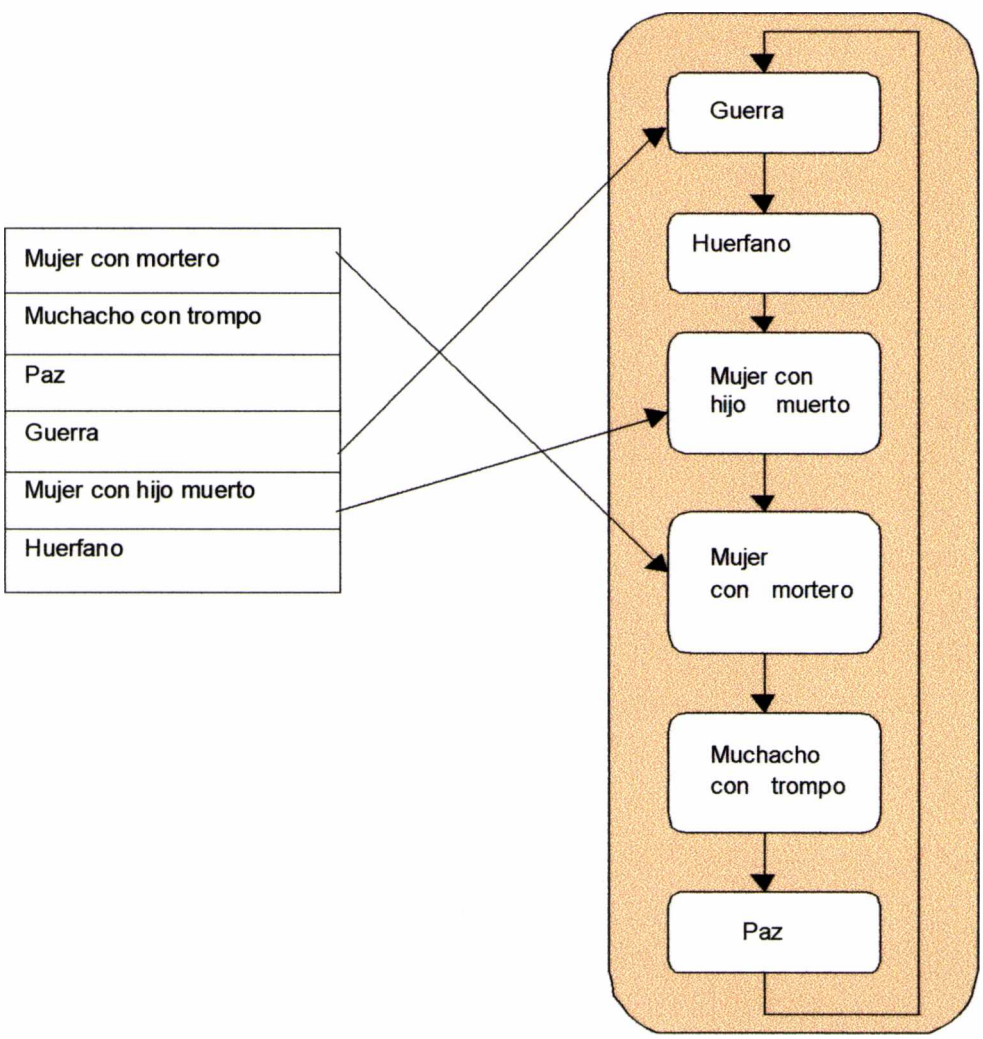


Figura 3.25: Contexto Cont5 accedido mediante una estructura de acceso EA2.

Notemos que se puede acceder a cualquier obra directamente y a partir de ésta seguir recorriendo el resto de las obras ordenadas de acuerdo a la estrategia especificada en el

contexto.

Como muestra la figura 3.25 cada entrada de esta estructura representa el acceso a un nodo del contexto, las cuales han sido representadas con el nombre de cada obra para que el ejemplo resulte claro, pero la forma en que se referencian las mismas en la estructura de acceso es una tarea del diseño de la interfaz.

La misma estructura de acceso puede ser utilizada en el contexto *Cont3*, en este caso se accede directamente a las obras, mientras que los eventos son alcanzados a través de las relaciones que los mismos tienen con cada obra.

3.4 Consultas de estructura

Las consultas de estructura están orientadas especialmente a los diseñadores de aplicaciones hipermedias. Por medio de ellas se podrá obtener información acerca de la estructura del esquema de la hipermedia. Tener conocimiento acerca de la definición de las clases y relaciones, así como también de los contextos navegacionales y estructuras de acceso es de significativa importancia a la hora de estudiar la estructura de una aplicación hipermedia en general.

3.4.1 Consultas de estructura sobre el esquema

Atributos

Esta consulta permite obtener los atributos definidos en una clase nodo perteneciente al esquema navegacional de un hipertexto.

El resultado de la operación es una lista, donde figuran los nombres de los atributos y la clase o dominio al cual pertenecen sus valores. En el caso de que el atributo sea un anchor, el componente de la lista corresponderá al nombre del atributo seguido del nombre de la clase relación a la cual referencia el anchor. La sintaxis de la operación es:

ATRIBUTOS (claseNodo, listAtributos)

Donde:

claseNodo: es la clase nodo consultada para obtener información de sus atributos.

listAtributos: lista de pares que corresponde al resultado. La estructura de la lista es [{atributo: dominio}], donde atributo es el mombre de un atributo de la clase *claseNodo* y dominio es el conjunto de valores posibles que dicho atributo puede poseer. Los elementos de la lista estarán separador por comas.

Ejemplo:

Obtener los atributos de la clase nodo Obra.

ATRIBUTOS (Obra, atributosObras)

El resultado de la consulta es atributosObras = {*título*: string, *fecha:* string, *lugar*: string, *descripción*: texto, *tema*: string, *colores*: set of color, *técnica*: string, *pintura*: Imagen, *comentarios*: texto, *fotografía*: anchor(Fotografiada_en), *referencia*: anchor(Referencia), *persona*: anchor(Retratada_por), e*vento*:anchor(Exhibida_en)}.

Es_superclase

Dadas dos clases pertenecientes al esquema, la función Es_superclase retorna un valor de verdad TRUE si una clase es superclase de otra, dentro de la jerarquía de clases, y FALSE en caso contrario.

La especialización de las clases dentro de un sistema va conformando una jerarquía de clases dentro del mismo. Por ello, la función Es_superclase podrá ser utilizada junto a otras funciones para poder descubrir la representación de la jerarquía mencionada

La función se escribe de la siguiente manera:

ES_SUPERCLASE (claseNodoA, claseNodoB)

Donde:

claseNodoA: es la clase nodo a verificar como superclase de la claseNodoB

claseNodoB: clase nodo.

Ejemplo:

ES_SUPERCLASE (Documento, Fotos)

Si observamos la figura 3.1 la operación ES_SUPERCLASE retornará TRUE dado que la clase *Fotos* es una especialización de la clase *Documento*.

Jerarquia

Esta consulta retorna una lista de nombres de clases que representa la jerarquía de superclases de una clase de nodo dada.

JERARQUIA (claseNodo, lista)

Donde :

claseNodo: es la clase de nodo a consultar.

lista: lista que corresponde al resultado.

Ejemplo:

Consultemos la jerarquía de clase de la clase *Entrevista*.

JERARQUIA (Entrevista, listaClases)

El resultado retornará la lista compuesta por {Documento}

Subclases

Para obtener un conocimiento mas detallado acerca de la estructura del esquema, como por ejemplo dada una clase nodo a qué otras clases puedo navegar, o por el contrario desde cuales la puedo alcanzar, el lenguaje cuenta con consultas que permiten trabajar con las relaciones.

La consulta *Subclases* permite obtener una lista de clases que conforman el primer nivel de subclases de una clase en particular.

SUBCLASES (claseNodo , listSubclases)

Donde:

claseNodo: es la clase padre a consultar.

listSubclases: lista de nombres de clase que corresponde al resultado.

Ejemplo:

Si quisieramos obtener las subclases de la clase *Documento*, la consulta será:

SUBCLASES (Documento, listSubclases)

Veamos que el contenido de listSubclases será: {Entrevista, Publicación, Fotos, Correspondencia}

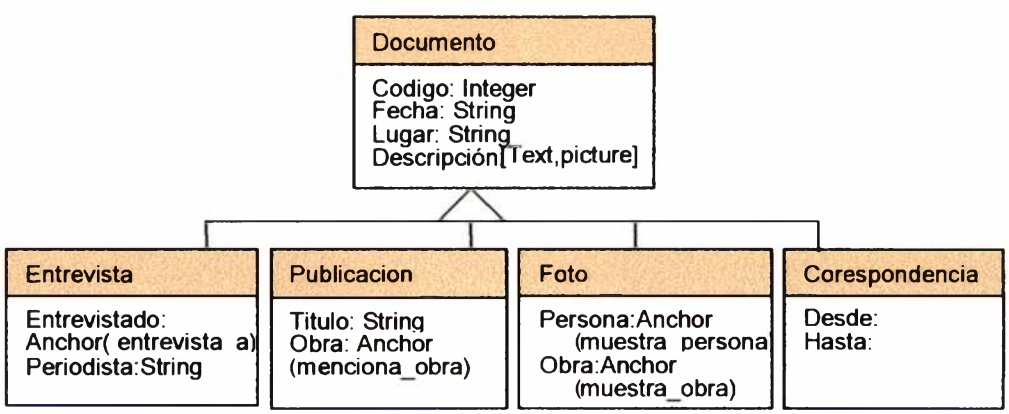


Figura 3.26: Subclases de la clase Documento.

Relacionada_por

Esta consulta nos devuelve las relaciones que existen entre dos clases nodos.

RELACIONADA_POR (claseNodo1, claseNodo2, listRelaciones)

Donde:

clasenodo1 y claseNodo2: son las clases nodos a consultar.

listRelaciones: lista de nombres de las relaciones que existen entre clasenodo1 y claseNodo2 y entre clasenodo2 y claseNodo1. En caso de que no exista alguna relación entre las clases especificadas, el resultado será una lista vacía.

Supongamos entonces que se quieren obtener las relaciones que unen las clases fotos y obras, la consulta será:

RELACIONADA_POR (Foto, Obra, listaRelaciones)

En este caso el resultado será la lista compuesta por:

{"Muestra_obra", "Fotografiada_en"}

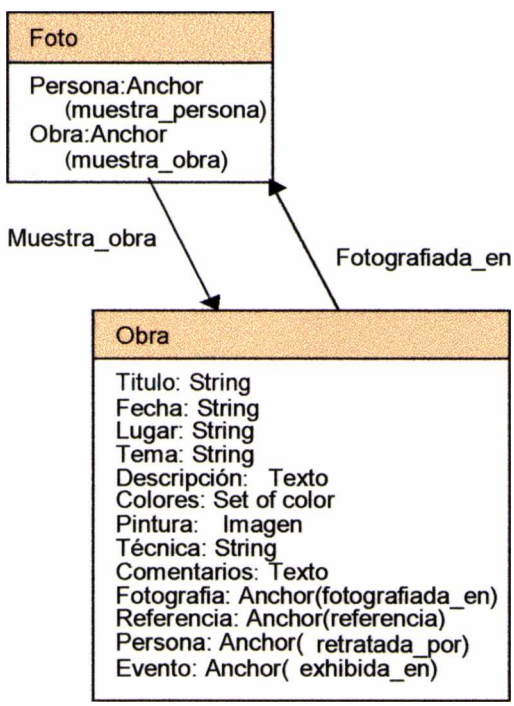


Figura 3.27: Relaciones entre las clases Fotos y Obra.

La consulta anterior nos permite, dentro de una hipermedia, reconocer cuáles son las relaciones que unen dos clases, indistintamente de qué clase sea el origen o el destino.

Relaciona

Esta consulta nos permite obtener los nombres de las clases nodo origen y destino de una clase relación.

RELACIONA (claseRelación, claseNodoOrigen, claseNodoDestino)

Donde:

claseRelación: es la clase relación a consultar

claseNodoOrigen: nombre de la clase origen de la relación *claseRelacion*.

claseNodoDestino: nombre de la clase destino de la relación *claseRelacion*.

Ejemplo:

Obtener el origen y el destino de la relación "*Exibida_en*".

RELACIONA ("Exhibida_en", origen, destino)

Como resultado de esta operación las variables origen y destino toman el valor *Obra* y *Evento* respectivamente.

Origen

Esta consulta devuelve la lista de los nombres de las relaciones en las cuales una clase nodo es el origen.

ORIGEN (claseNodo, listRelaciones)

Donde:

claseNodo: clase nodo a consultar.

listRelaciones: lista de las relaciones en las cuales *claseNodo* es el origen.

Ejemplo:

Obtener las relaciones donde la clase Persona es origen:

ORIGEN (Persona, relaPersona)

Luego de realizar la consulta, el contenido de *relaPersona* será: {*Otorga, Retratada_en, Posee*}.

Destino

Esta consulta devuelve la lista de los nombres de las relaciones en las cuales una clase nodo es el destino.

DESTINO (claseNodo, listRelaciones)

Donde:

claseNodo: clase nodo a consultar

listRelaciones: lista de las relaciones donde *claseNodo* es el destino.

Ejemplo:

Obtener las relaciones donde la clase *Obra* es destino:

DESTINO (Obra, relaObra)

y como resultado: relaObra = {*Exhibe, Menciona_obra, Muestra_obra*} .

Dado que dos clases pueden estar relacionadas también mediante la relación parte de, la cual nos indica que una clase esta compuesta por otras clases nodos, las operaciones definidas a continuación nos permiten obtener información acerca de la composición de clases.

Compuesta_por

Mediante esta consulta el diseñador puede obtener las clases que componen una clase especifica, retornando como resultado la lista de nombres de esas clases.

COMPUESTA_POR (claseNodo, ListNombreClase)

Donde:

claseNodo: clase nodo a consultar.

listNombreClase: lista de los nombres de las clases que componen la clase nodo indicada. En el caso de tratarse de una claseNodo que no es compuesta obtendremos como resultado una lista vacía.

Es_compuesta

A través de esta función podemos verificar si una clase es compuesta o no, es decir que devolverá True si tiene estructura de partes en su definición y False en caso contrario.

ES_COMPUESTA (claseNodo)

Donde:

claseNodo: clase a consultar

3.4.2 Consultas de estructura sobre Constructores Navegacionales.

Hipertexto

Esta operación devuelve el valor del atributo hipertexto de un contexto navegacional dado, es decir dado un contexto navegacional, el usuario o diseñador tiene la posibilidad de saber sobre que hipertexto fue creado.

HIPERTEXTO (ContextoNavegacional, Result)

Donde:

ContextoNavegacional: contexto navegacional a consultar.

Result: valor del atributo hipertexto de *ContextoNavegacional.*

Objetos

Esta operación devuelve una lista con los objetos que incluye un contexto navegacional dado, de esta manera podemos conocer la colección de objetos involucrados en un contexto.

OBJETOS (contextoNavegacional, lisNomObjetos)

Donde:

contextoNavegacional: contexto navegacional a consultar.

listNomObjetos: valor del atributo objetos que incluye del *contextoNavegacional.*

Punto_de_Entrada

Esta operación devuelve el punto de entrada de un contexto navegacional dado.

PUNTO_DE_ENTRADA (ContextoNavegacional, punto_entrada)

Donde:

ContextoNavegacional: contexto navegacional a consultar.

punto_entrada: valor del atributo punto de entrada del *ContextoNavegacional.*

Ejemplo:

Obtener el punto de entrada del contexto navegacional obtenido en Cont2 definido en la sección anterior.

PUNTO_DE_ENTRADA (Cont2, entrada)

El resultado de la consulta estará reflejado en la variable *entrada* cuyo valor será: entrada = o.titulo = "Guerra"

Estrategia

Esta operación permite obtener la strategia asociada a un contexto navegacional dado, y de esta manera saber cómo podrán ser recorridos los elementos de un contexto específico.

ESTRATEGIA (contextoNavegacional, estrategia)

Donde:

contextoNavegacional: contexto navegacional a consultar.

estrategia: valor del atributo estrategia del contexto navegacional *contextoNavegacional.*

Cambios_a_contexto

Esta operación nos permite conocer el nombre de los contextos a donde podemos mudar desde un contexto navegacional dado.

CAMBIOS_A_CONTEXTO (contextoNavegacional, listNombres)

Donde:

contextoNavegacional: contexto navegacional a consultar.

listNombres: valor del atributo cambio de contexto del contexto navegacional *contextoNavegacional.*

Objetos_est_acceso

Así como la operación *Objetos* nos permite obtener los objetos que incluye un contexto navegacional, la operación objetos_est_acceso nos permite obtener los objetos que incluye una estructura de acceso dada.

OBJETOS_EST_ACCESO (EstructuraAcceso, listNomObjetos)

Donde:

EstructuraAcceso: Estructura de acceso a consultar.

listNomObjetos: valor del atributo objetos que incluye de *EstructuraAcceso.*

Capítulo 4

EJEMPLO

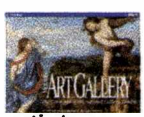
4.1 Introducción

En este capítulo, se presenta un ejemplo general en donde se muestra la utilización del Lenguaje de Consultas, generando hipertextos y definiendo el modo de navegación sobre los mismos. Para ello, hemos elegido la aplicación hipermedia Microsoft's Art Gallery la cual modelizaremos utilizando OOHDM para luego ejemplificar el uso del Lenguaje de Consultas.

EL resultado de este ejemplo muestra una *visión* particular de la Aplicación Microsoft's Art Gallery.

Es nuestro objetivo destacar que cada usuario podrá generar su propia vista basándose en un mismo dominio de información, organizándola y decidiendo la forma de recorrerla.

4.2 Microsoft's Art Gallery



Art Gallery es una aplicación hipermedia que presenta la Galería Nacional de Arte de Londres. En ella se encuentran colecciones de pinturas del oeste europeo comprendidas entre el siglo XIII y el siglo XX. La vida de pintores y artistas es presentada en esta aplicación por biografías detalladas mencionando fuertes influencias entre artistas, trayectoria y obras destacadas. Las colecciones de los artistas son organizadas por lugares y periodos de tiempos agrupando de este modo las obras de las distintas partes de Europa.

4.3 Modelización en OOHDM

En primera instancia, vamos a presentar el esquema conceptual de la aplicación modelando el dominio de la misma, mostrando las clases conceptuales, relaciones y atributos; luego, el esquema de clases navegacionales; y por último, el esquema de contextos navegacionales que será acompañado de una secuencia de navegación reflejando el objetivo de una búsqueda específica. Los esquemas fueron tomados de la

referencia bibliográfica [7].

La modelización de esta aplicación hipermedia no incluye el diseño de interfase abstracta ni la implementación debido a que para ejemplificar sobre la misma un conjunto de consultas no es necesario el modelado de estas etapas.

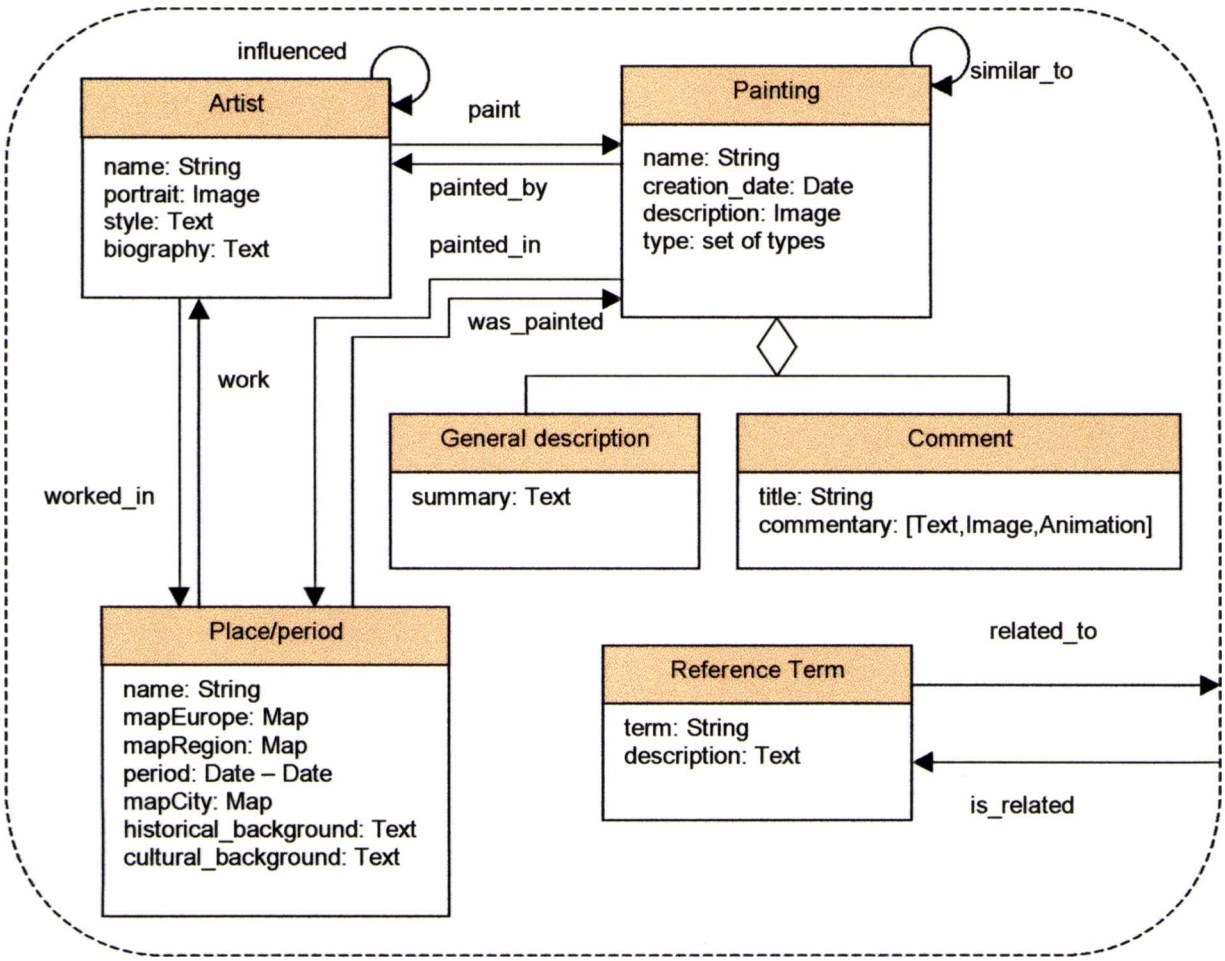


Figura 4.1: Esquema conceptual de Art Gallery.

Como podemos observar en la figura anterior las clases identificadas son: *Artist*, *Painting*, *Place/Period* y *Reference term*.

La clase *Artist* consiste de los datos personales del artista como así tambien de información acerca del estilo del pintor, características generales de su trabajo, etc. La clase *Painting* contiene información general de las obras, notas acerca de su composición y detalles. La clase *Place/period* consiste de información geográfica acerca de una ciudad o región en un cierto periodo de tiempo. La clase *Reference Term* representa el glosario de terminos de la aplicación.

En el esquema conceptual se representan las relaciones por medio de flechas entre las clases. Se identifican en el las siguientes relaciones: *worked_in*, *painted_in*, *painted*, *influenced*, *similar_to* y *related_to*.

El siguiente paso en la modelización consiste en definir el modelo navegacional de la aplicación, en el cual se determinan las clases navegacionales.

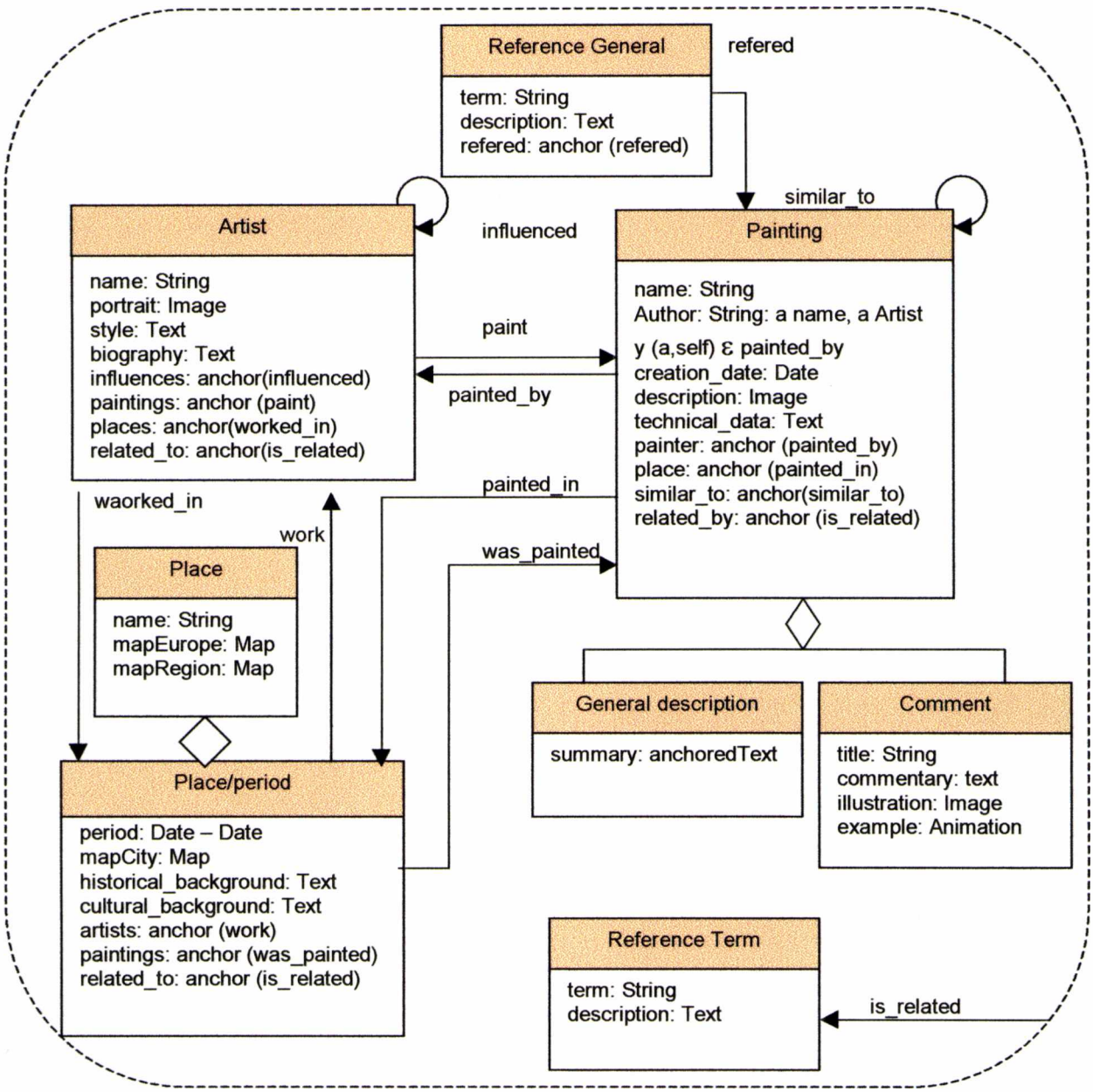


Figura 4.2: Esquema de clases navegacionales de Art Gallery.

En la Figura 4.2 vemos que las clases navegacionales del ejemplo son: *Artist, Painting, Place/Period, General Description, Comment, Reference term y Reference General,* por lo que podemos observar la analogía entre las clases definidas en el modelo conceptual y el navegacional. La clase *Place/period* ha sido mapeada en una clase compuesta debido a que en la aplicación un *Place/period* es siempre visto como parte de un *Place*, a pesar del esquema conceptual, donde cada *Place/period* es especificado independientemente uno de otro. Por ejemplo, si el usuario este leyendo acerca de Picasso y desea ver las ciudades y periodos donde trabajó, le serán presentadas las opciones "París 1900- 1950" y "Rest of France 1900-1925". Cuando el usuario elija navegar a "París 1900-1950" y luego selecciones el botón Next page, el debería alcanzar otro *Place/period* de París relacionado con Picasso. En cambio, este navega por todos los periodos en los cuales París fue una ciudad de alguna relevancia. Esto nos indica que cada periodo de París es parte del lugar

París, desde el punto de vista navegacional.

Notemos que, Place es usada solamente para agrupar los atributos comunes: *name*, *mapEurope*, *mapRegion*.

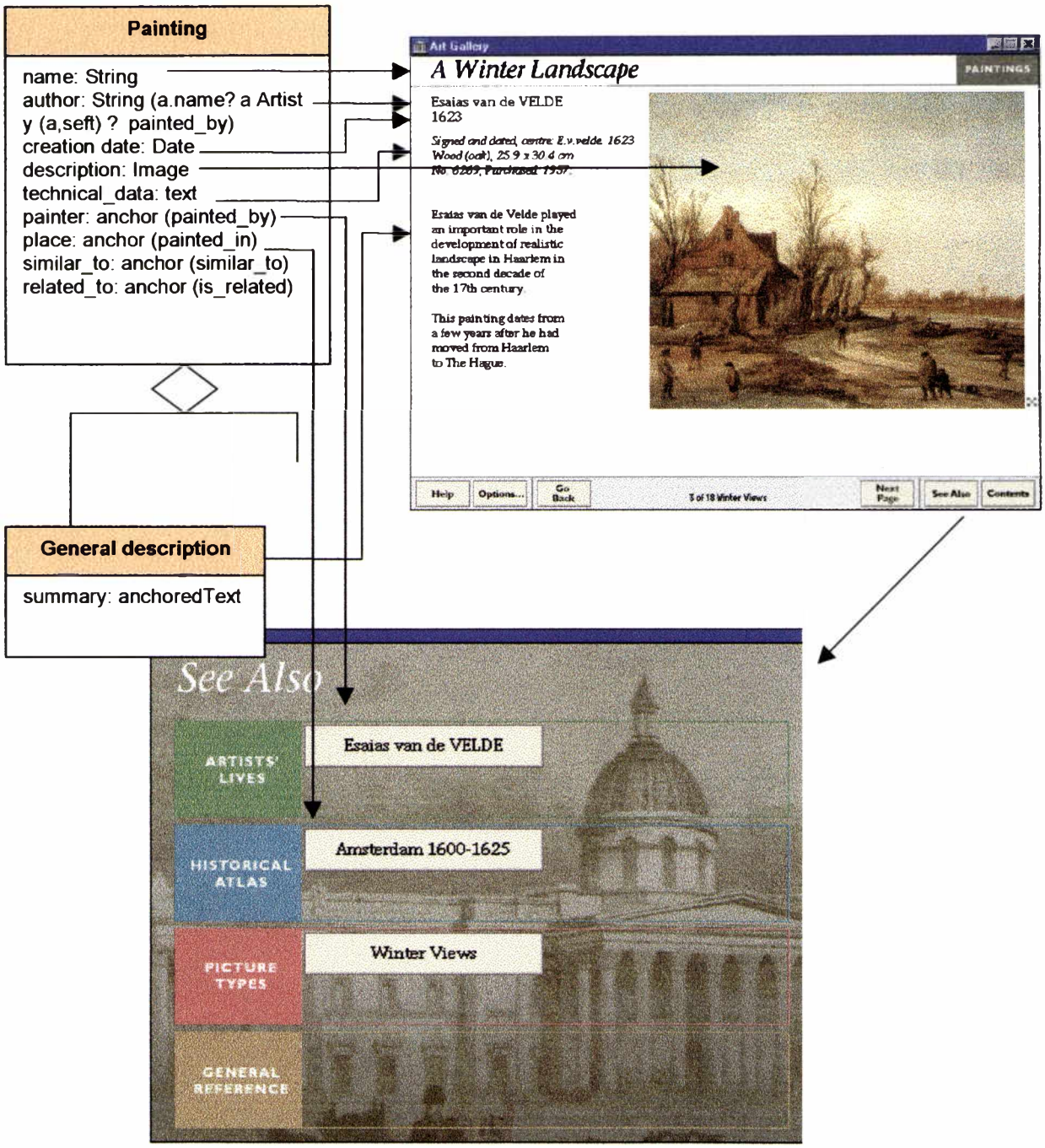


Figura 4.3: Ejemplo de la clase navegacional *Painting* y una instancia de la misma.

La Figura 4.3 muestra un ejemplo de la clase nodo *Painting* junto con una instancia de la misma "A Winter LandScape". Esta clase está derivada de la correspondiente clase en el esquema conceptual mostrada en la figura 4.1. Vemos como en el esquema navegacional el atributo A*utor* es agregado a la definición de la clase navegacional *Painting*, el cual es mapeado desde la clase conceptual *Artist*.

Como hemos visto en el capítulo 2, la modelización de la fase navegacional de OOHDM implica además de la definición de las clases navegacionales, la descripción de las

diferentes formas de navegación, dentro de la aplicación, representada por el esquema de contextos navegacionales.

El modelo de contextos navegacionales presentado a continuación muestra los contextos encontrados en la aplicación Microsoft's Art Gallery.

Una vez presentado el esquema de contextos navegacionales, navegaremos por un contexto específico mostrando a cada instante las alternativas existentes para continuar la navegación.

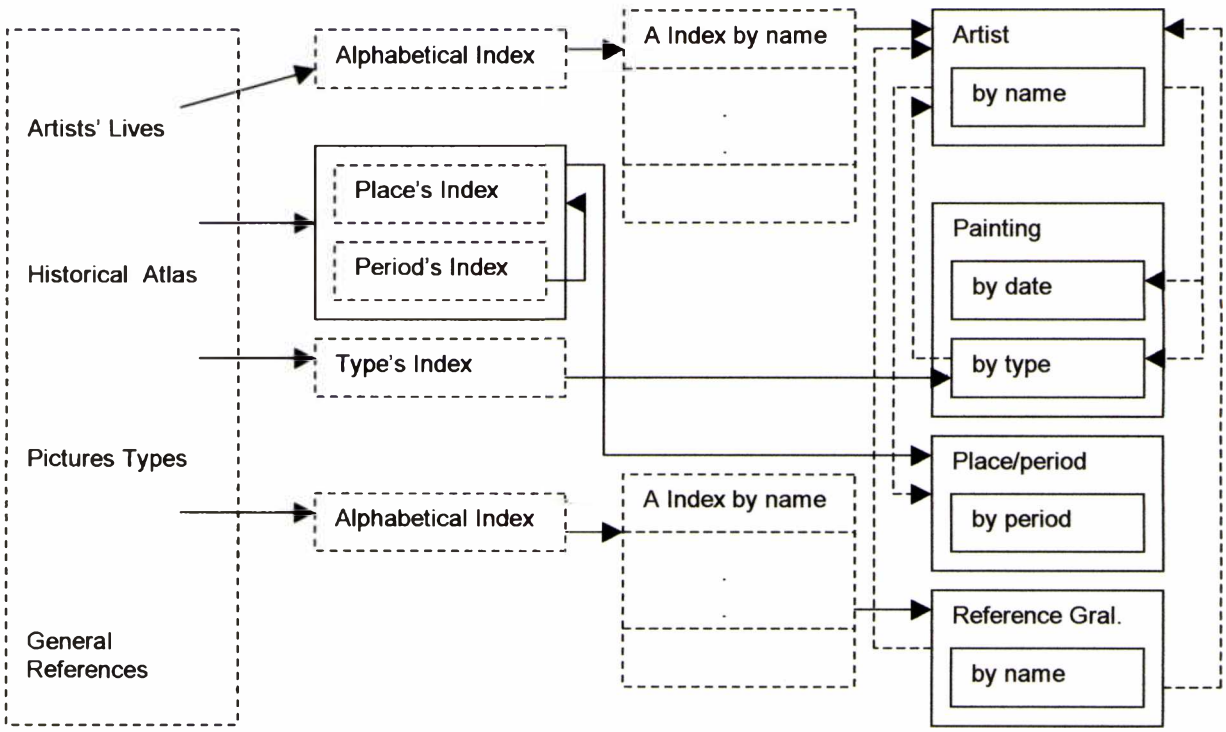
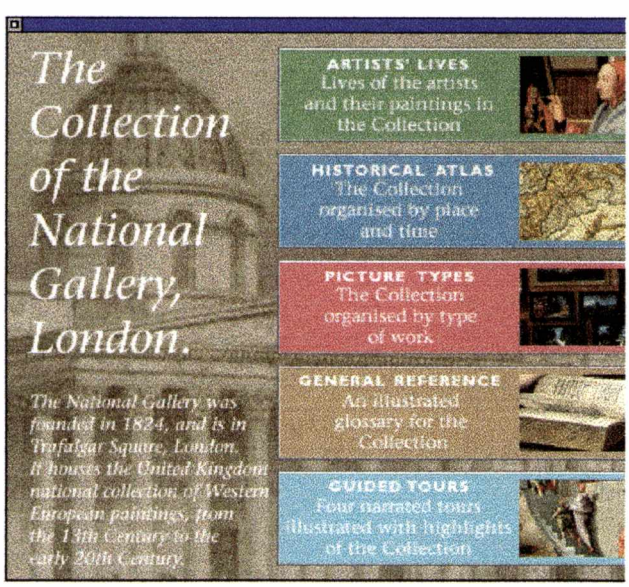
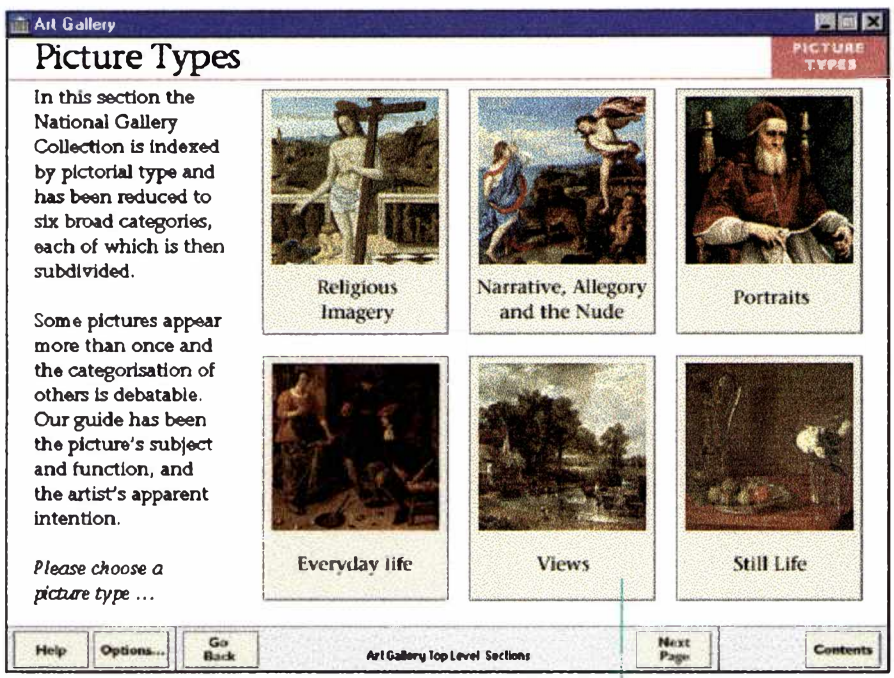


Figura 4.4: Esquema de Contextos Navegacionales.

El punto de entrada a la aplicación exhibe un índice general o menú principal donde el usuario puede elegir entre un índice alfabético de artistas, ver la colección por un periodo o lugar, las pinturas clasificadas por tipo, acceder al glosario general de términos o bien a alguno de los tours guiados ya definidos.

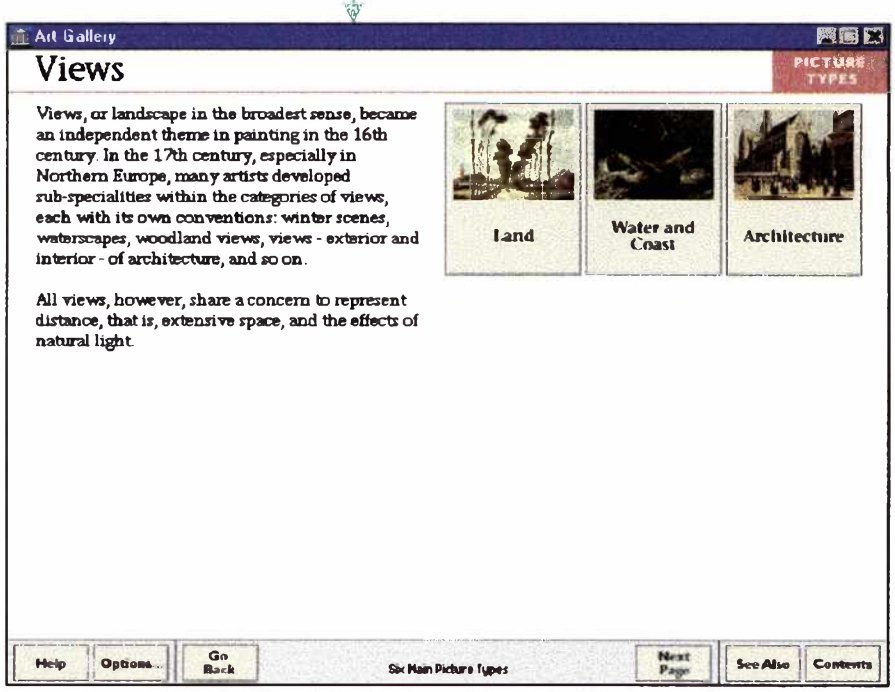


Supongamos que el usuario elige comenzar a navegar dentro de la hipermedia a traves de la clasificación de pinturas por tipo ya que está interesado en visitar el conjunto de pinturas de la colección, que exhiban paisajes de inviernos. Mostramos la secuencia de pantallas visitadas:



En esta sección la colección es organizada en seis categorías cada una de las cuales se encuentra a su vez subclasidicada.

Los paisajes (views) son subclasidicados en "Land", "Water and coast" and "Architecture".



Nuevamente, el usuario debe escoger entre nuevas alternativas (Naturalistic, Topographical, Idealiced Landscape, Animals in Landscape, Battle Scenes y Winter), las cuales derivarán finalmente en un conjunto de obras con las características elegidas (Land, Water and Coast, Architecture) a lo largo de todo el camino recorrido. Notemos que la elección de alguna de ellas nos muestra la pintura propiamente dicha dejando atrás la clasificación por tipos.

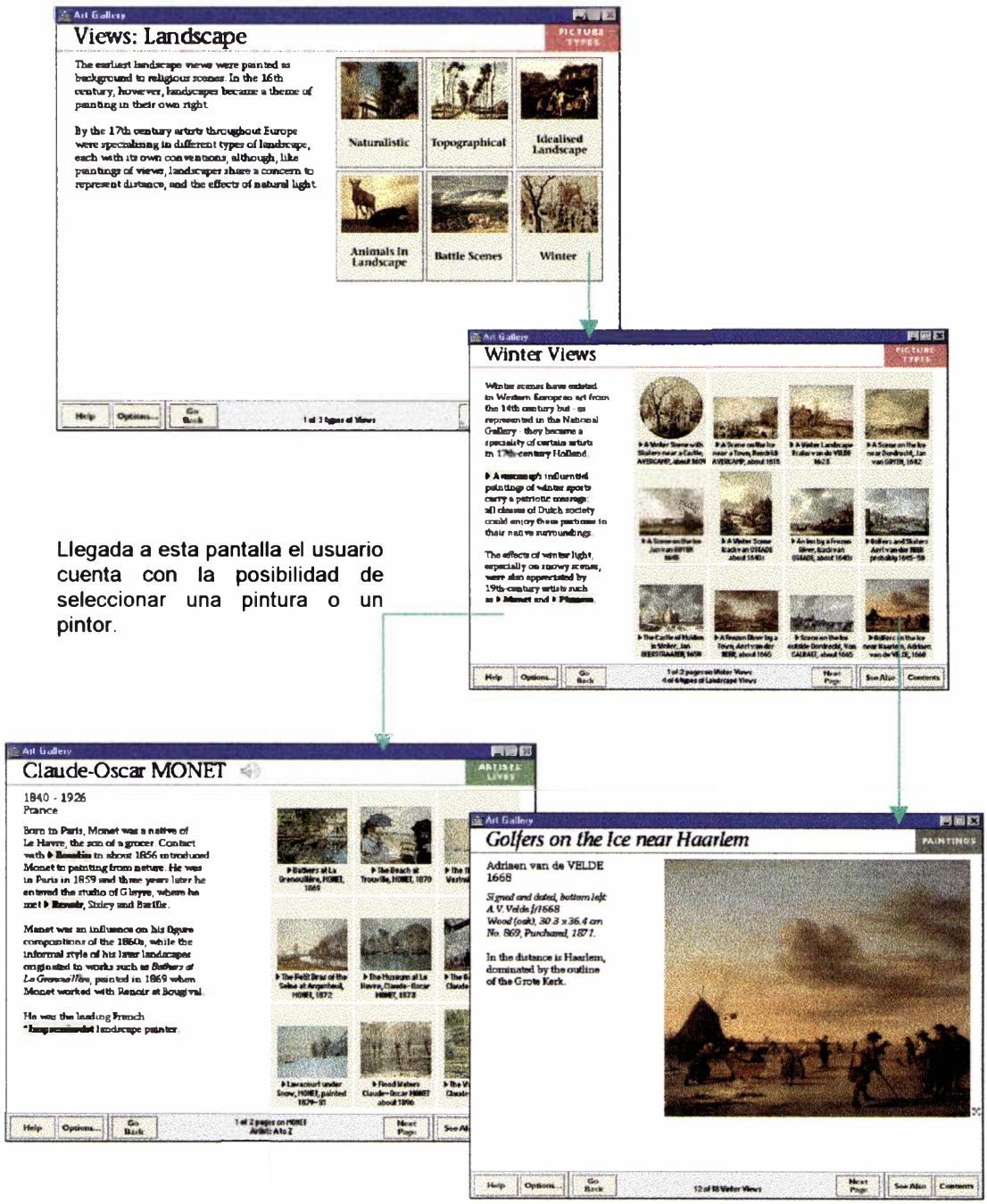


Figura 4.5: Camino recorrido para llegar a obtener pinturas invernales.

La elección de cualquiera de las dos alternativas mostradas en la figura 4.5 lleva al usuario a recorrer dos contextos diferentes. La elección de un *vínculo* (palabra resaltada dentro del texto) que contiene el nombre de un pintor, lo llevará a navegar por el contexto de *Artists' Lives* que reúne a los pintores ordenados alfabeticamente. En cambio, la selección de alguna de las pinturas le permite navegar, presionando en botón *Next Page*, el contexto conformado por las pinturas clasificadas en *Winter views* rescatando, de esta manera, la información pretendida desde un principio.

Veamos a continuación cómo, utilizando el Lenguaje de Consultas desarrollado como objetivo de esta tesis, un usuario puede rescatar directamente por medio de una consulta, información específica

4.4 Aplicación del Lenguaje de Consultas

Esta sección tiene como objetivo principal resolver distintas inquietudes de consultas realizadas por un usuario.

Comenzaremos presentado la respuesta a la consulta planteada en la sección anterior, luego realizaremos otro conjunto de consultas sobre la aplicación Art Gallery y por último confeccionaremos una visión particular de la galería de arte.

4.4.1 Consulta: Pinturas Invernales

Vamos a resolver la consulta planteada en la sección 4.3 utilizando el Lenguaje de Consultas. Veamos entonces, cómo podemos obtener en forma directa las pinturas que exhiben paisajes de invierno contenidas en el hipertexto ArtGallery.

```
Q1:= SELECT Painting
      FROM p: Painting IN ArtGallery
     WHERE p.technical_data = "Winter"
```

El resultado del hipertexto Q1 está formado por las pinturas: "A Winter Scene with Skaters", "A Scene on the Ice near a Town", "A winter Landscape", "A Scene on the Ice near Dordrecht", "A Scene on the Ice", "A winter scene", "An Inn by a Frozen River", "Golfers and Skaters near Village", "The Castle of Muiden in winter", "A frozen river by a town", "Scene on the Ice outside Dordrecht", "Golfers on the Ice near Haarlem","A scene on the Ice", "Winter Landscape", "The diligence in the snow", "Lavacourt under snow", "Skating in Holland" y "The louvre under snow", que son las accedidas en la aplicación siguiendo la secuencia de navegación detallada en la sección anterior. Notemos, entonces, lo directo que se hace ahora acceder a esta información dentro de una aplicación hipermedia utilizando el Lenguaje de consulta.

Una vez obtenidos los datos, definimos la forma de navegar, preservando el orden que contiene la hipermedia original.

En primera instancia, dado que en la aplicación se puede acceder a cada obra en forma directa, definimos la estructura de acceso que nos permitirá realizar la misma selección.

```
EA1:= ESTRUCTURA_DE_ACCESO (Objetos que incluye: p: Painting)
```

Confeccionamos entonces el contexto que nos permite navegar por la información obtenida en Q1.

```
C1:=CONTEXTO ( Hipertexto: Q1
               Objetos que incluye: p: Painting
               Punto de Entrada: EA1
               Estrategia: circular (Painting)
               Orden: p.creation_date ASC
               Cambio de contexto:      )
```

La figura 4.6 exhibe el resultado de las sentencias formuladas anteriormente.

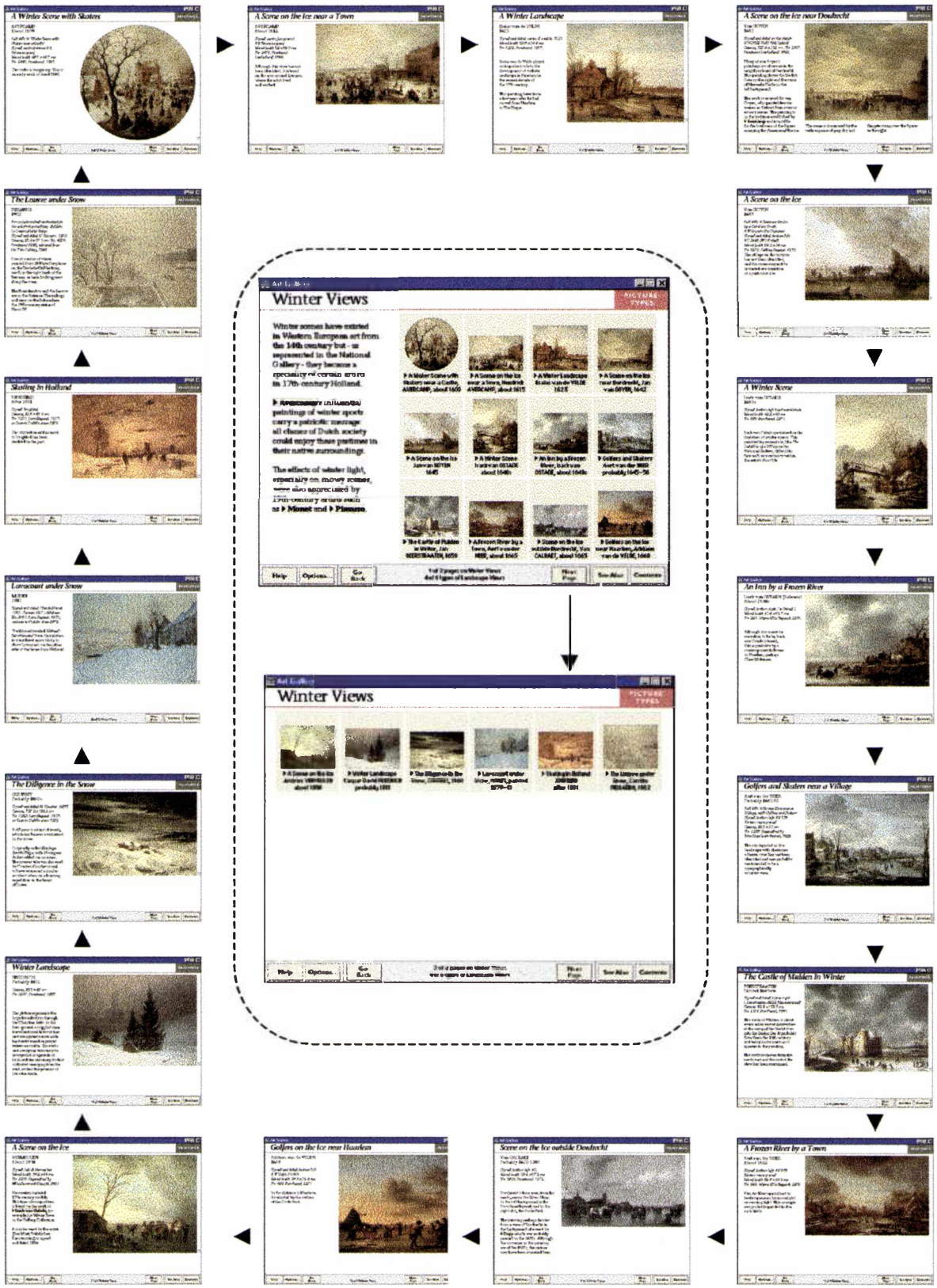


Figura 4.6: Representación gráfica del resultado de la consulta en Microsoft' Art Gallery.

La representación de la estructura de acceso fue enmarcada por líneas punteadas en la figura 4.6, ya que en la hipermedia original se encuentra contenida en dos pantallas. La segunda pantalla de la estructura de acceso es alcanzada por medio del botón *Next Page* a partir de la primera pantalla.

La selección de cualquiera de las pinturas visualizadas en la estructura de acceso nos sumerge en el contexto de pinturas invernales (C1), las cuales podrán ser navegadas a través de este por medio del botón *Next Page*, siguiendo un orden ascendente por fecha de creación como fue especificado.

4.4.2 Potencialidad del uso del Lenguaje de Consultas.

Cada una de las consultas que formularemos a continuación posee algún grado de dificultad a la hora de recuperar la información pretendida cuando se navega por la aplicación Art Gallery, ya sea, recorriendo nodos no deseados o simplemente no hallando caminos directos que nos orienten hacia el destino final.

- Algunos artistas mencionados en la aplicación Art Gallery han sido influenciados en sus carreras por otros artistas que dedicaron tiempo para entrenarlos, asesorarlos y llevarlos a formar parte de los *Grandes Artistas* que trascendieron en los últimos siglos.

Si intentamos rescatar los artistas que fueron influenciados por otro artista en particular, por ejemplo "Albrecht Dürer", debemos navegar por todos los artistas y verificar en cada uno de ellos si dicha influencia existe. Resolvamos esta inquietud por medio de una consultas:

```
Q2 :=  SELECT Artist
       FROM  a: Artist, b: Artist IN ArtGallery
       WHERE  HayRelación (a, "influenced", b) AND
                  b.name = "Albrecht Dürer"
```

El hipertexto resultante Q2, esta formado por los artistas "Hans Baldung Grien", "Jacopo de' Barbari", "Jacopo Bassano" y "Giovanni Bellini" entre otros, los cuales fueron obtenidos directamente especificando una condición de búsqueda y evitando que se navegue sobre información irrelevante en la búsqueda de datos concretos.

Definimos la navegación sobre el hipertexto Q2 por medio del contexto C2.

```
C2 := CONTEXTO  (Hipertexto: Q2
                    Objetos que incluye: a: Artist
                    Punto de Entrada: a.name ="Giovanni Bellini"
                    Estrategia: circular (Artist)
                    Orden: a.name ASC
                    Cambio de contexto:    )
```

- La colección de pinturas encontradas en la Galería Nacional de Londres ha sido organizada por tipo de trabajo en seis grandes categorías: "Imágenes Religiosas", "Narrativa, Alegoría y Desnudo", "Retratos", "Vida Cotidiana", "Paisajes" y "Naturaleza Muerta". Cada una de estas a su vez posee subcategorías, lo cual permite desglosar el tema original en subtemas relacionados.

Si pretendemos hallar una pintura específica (por ejemplo: "The Introduction of the Cult of Cybele") dentro de la colección y no contamos con datos acerca de su creador, deberíamos conocer con anterioridad la posible clasificación a la que pertenece para reducir los caminos de búsqueda.

Q3 := SELECT Painting

FROM p: Painting IN ArtGallery

WHERE p.name = "The Introduction of the Cult of Cybele"

En el hipertexto Q3 está incluida la pintura buscada en la aplicación Art Gallery, para poder navegarla debemos definir el siguiente contexto.

C3 :=CONTEXTO (Hipertexto: Q3

Objetos que incluye: p: Painting

Punto de Entrada: p.name ="The Introduction of the Cult of Cybele"

Estrategia: circular (Painting)

Orden: p.name ASC

Cambio de contexto:)

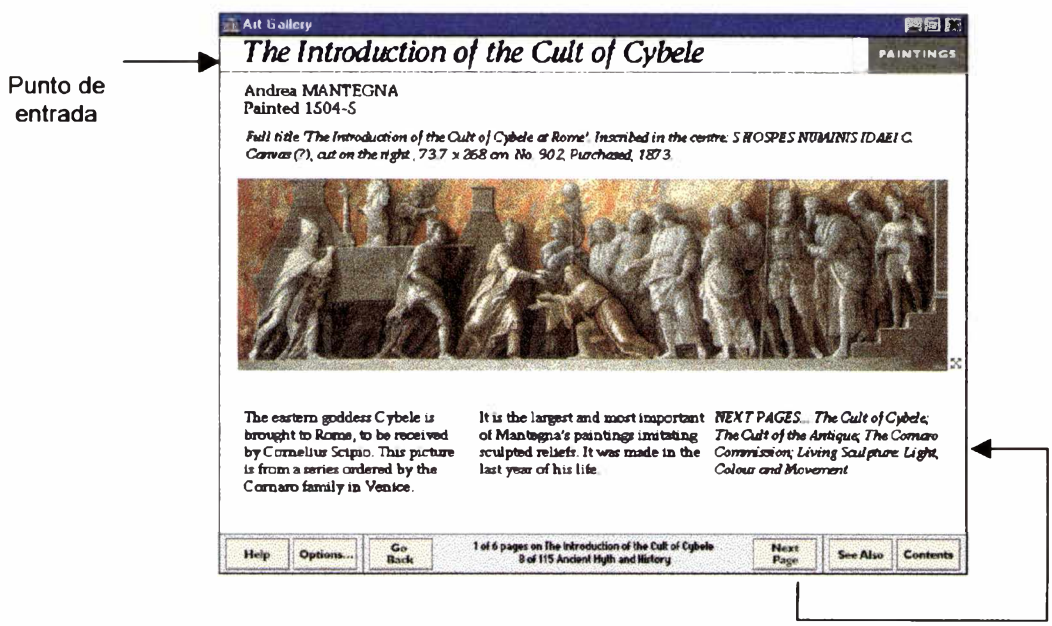


Figura 4.7: Contexto de navegación del hipertexto Q3.

Sobre el hipertexto Q3 fue confeccionado un contexto que tiene como punto de entrada la pintura "The Introduction of the Cult of Cybele". Por ser esta el único nodo existente en el hipertexto, la estrategia indica que el siguiente nodo dentro del contexto es la misma pintura.

4.4.3 Una visión de la Aplicación Art Gallery

Cuándo es necesaria la construcción de una visión particular de la hipermedia? En síntesis, cuando se combina la necesidad de rescatar información puntual y a su vez recorrerla bajo un criterio determinado. Las visiones posibles a construir podrían ser incontables a la hora de pedirle a un usuario que defina la más interesante o necesaria sobre la aplicación Art Gallery, por lo que hemos decidido tomar el rol del mismo y describir paso a paso su generación.

Estamos interesados en rescatar todas las pinturas de la Galería de Arte que hayan sido realizadas en Francia o en Italia y sus creadores, pero nuestra idea es manipular la información como dos conjuntos claramente separados y, organizarla de la siguiente forma:

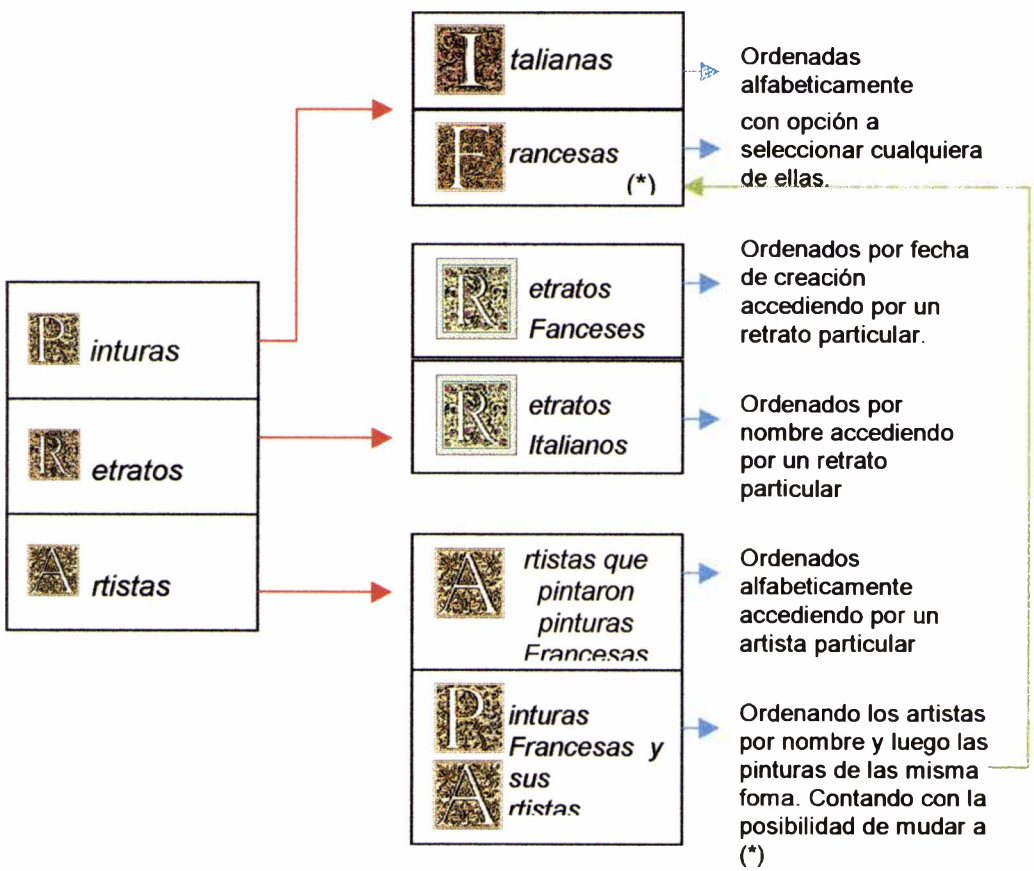


Figura 4.8: Representación gráfica de la visión a construir

Así, comenzamos con aquellas localizadas en Francia y posteriormente haremos lo propio con las de Italia.

Nuestra primera tarea es construir el hipertexto sobre el cual vamos a trabajar.

H1:= SELECT Painting, Artist

FROM p: Painting, a: Artist, pp: Place/Period IN ArtGallery

WHERE HayRelación (p, "painted_in", pp) AND

((pp.name = "París" or pp.name = "Rest of France") AND

HayRelación (p," painted_by", a)

De esta manera en el hipertexto H1 hemos recuperado todas las instancias de la clase *Painting* vinculadas por medio de la relación *Painted_in* con las instancias de la clase *Place/Period* cuyo atributo *name* es igual a "París" o "Rest of France"; y, las instancias de la clase *Artist,* que se relacionan con las antes recuperadas a traves de la relación *Painted_by*.

Una vez obtenido el hipertexto nos concentramos en la creación del contexto navegacional que defina el modo de recorrer sólo las pinturas, para ello primero definimos una estructura de acceso que nos presente las pinturas ordenadas alfabéticamente.

EAPaintings:= ESTRUCTURA_DE_ACCESO (Objetos que incluye: p: Painting)

CN1 := CONTEXTO (Hipertexto: H1

Objetos que incluye: p: Painting

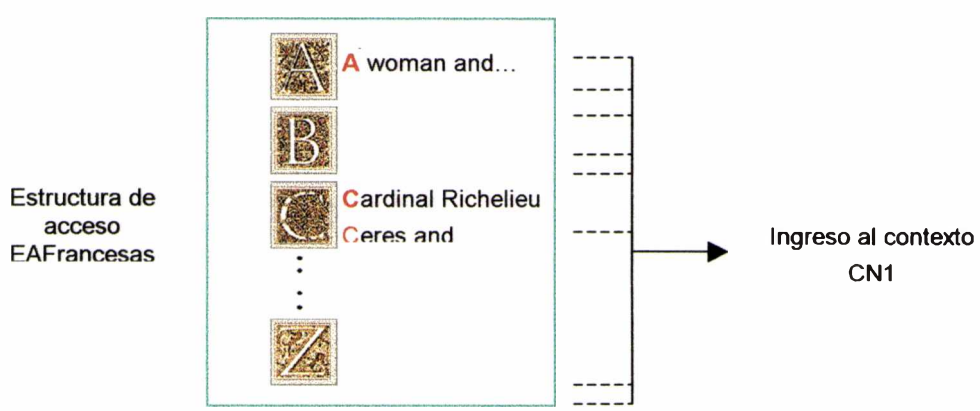
Punto de entrada: EAPaintings

Estrategia: secuencial (Painting)

Orden: p.name ASC

Cambio de contexto:)

La siguiente figura muestra una posible representación gráfica de la estructura de acceso que se utiliza como punto de entrada al contexto CN1, exhibiendo todas las pinturas ordenadas alfabéticamente por su nombre.



Continuemos desarrollando el ejemplo recuperando las pinturas y artistas de las distintas ciudades y regiones de Italia presentes en el Art Gallery.

En consecuencia, el hipertexto sobre el que vamos a trabajar se genera a partir de la siguiente consulta:

```
H2 :=  SELECT Painting, Artist
       FROM p: Painting, a: Artist, pp: Place/Period IN ArtGallery
      WHERE HayRelación (p, "painted_in", pp) AND
     ((pp.name = "Rome" OR pp.name = "Milan" OR
      pp.name = "Brescia" OR pp.name = "Venice" or pp.name = "Ferrara"  OR
      pp.name = "Bologna" OR pp.name = " Florence" OR
       pp.name = "Siena"  OR pp.name = "Naples" OR
       pp.name = "Rest of Italy"))   AND
       HayRelación  (p," painted_by", a)
```

Como vemos la especificación de H2 responde al patrón ya utilizado en la creación de H1.

Esta similitud se manifiesta también al momento de generar su contexto navegacional en el cual es posible reutilizar la estructura de acceso ya definida dado que deseamos que ambos contextos posean el mismo tipo de punto de entrada. Así, la definición del contexto creado sobre H2 es la siguiente:

```
CN2 := CONTEXTO  (Hipertexto: H2
                  Objetos que incluye: p: Painting
                  Punto de entrada: EAPaintings
                  Estrategia: secuencial (Painting)
                  Orden: p.name ASC
                  Cambio de contexto: )
```

Los contextos CN1 y CN2 previamente definidos nos posibilitan navegar entre las pinturas de Francia e Italia respectivamente. La estructura de acceso creada a continuación es la que nos permite seleccionar entre uno u otro.

```
EAPinturas:= ESTRUCTURA_DE_ACCESO (Objetos que incluye: CN2, CN1)
```

De acuerdo a la Figura 4.8, continuamos con la confección de esta visión recuperando las pinturas de Fancia e Italia respectivamente que son clasificadas en la aplicación Art Gallery como "Retratos" (Portraits).

```
H3 :=  SELECT Painting
        FROM  p: Painting IN H1
       WHERE Existe(p.type, "Portraits")
```

Ya que contábamos con el hipertexto H1 conteniendo entre sus nodos las pinturas realizadas en Francia, utilizaremos al mismo para filtrar de este sólo los retratos. Del mismo modo lo haremos partiendo de H2 para seleccionar los retratos de Italia.

```
H4 := SELECT Painting
       FROM  p: Painting IN H2
       WHERE Existe(p.type, "Portraits")
```

```
CN3 := CONTEXTO  (Hipertexto: H3
                  Objetos que incluye: p: Painting
                  Punto de entrada: p.name = "Bona of Savoy"
                  Estrategia: circular (Painting)
                  Orden: p.creation_date ASC
                  Cambio de contexto: )
```

CN4 := CONTEXTO (Hipertexto: H4

Objetos que incluye: p: Painting

Punto de entrada: p.name = "Triple Portrait of Cardinal Richelieu"

Estrategia: secuencial (Painting)

Orden: p.name ASC

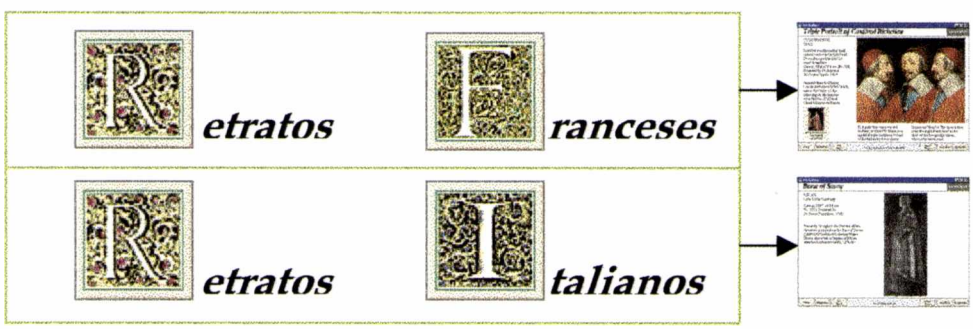
Cambio de contexto:)

Los contextos CN3 y CN4 describen la navegación de H3 y H4 respectivamente y, poseen puntos de entrada obligatorios a diferencia de los contextos anteriores CN1 y CN2 los cuales emplearon estructuras de accesos.

Definamos ahora una estructura de acceso que abarque la invocación a cada uno de los contextos.

EARetratos:= ESTRUCTURA_DE_ACCESO (Objetos que incluye: CN3, CN4)

Exponemos en la siguiente figura el acceso a los contextos CN3 y CN4 en la cual se exhiben las instancias de la clase *Painting* que son los puntos de entradas respectivos a los mismos.



Ahora nos enfocaremos en los artistas que pintaron las pinturas organizadas CN1 y CN2. Por un lado agruparemos a todos los artistas creadores de las pinturas italianas en un contexto de navegación, y por otro, nos interesamos en creadores de pinturas francesas y en las pinturas propiamente dichas.

CN5 := CONTEXTO (Hipertexto: H2

Objetos que incluye: a: Artist

Punto de entrada: a.name = "Phillippe de CHAMPAIGNE"

Estrategia: secuencial (Artist)

Orden: a.name ASC

Cambio de contexto:)

CN6 := CONTEXTO (Hipertexto: H1

Objetos que incluye: a: Artist, p: Painting

Punto de entrada: a.name = "The Blood of the Redeemer"

Estrategia: circular (Artist, Painting)

Orden: a.name ASC, p.name ASC

Cambio de contexto: CN1)

CN5 nos permite recorrer cada uno de los artistas, viendo y teniendo la posibilidad de navegar hacia una pintura debido a que la misma pertenece al hipertexto sobre el cual se creo el contexto. Una vez accedida la pintura nos encontraremos fuera del contexto, sin contar con la posibilidad de retornar al mismo. Este mismo caso se presenta en CN1 y en CN2.

Distinto es el caso en CN6 debido a que el mismo incorpora un cambio de contexto, por lo tanto el navegar desde un artista a la pintura que el mismo realizó no nos cambia de contexto, salvo que el navegante lo decida, ya que esa pintura está incluida en ambos contextos. La figura 4.9 exhibida en la página siguiente ilustra el cambio de contexto especificado en CN6

La estructura de acceso EAartistas agrupa los contextos definidos.

EAartistas := ESTRUCURA_DE_ACCESO (objetos que incluye: CN5,CN6)

Como último punto, confeccionamos una estructura de acceso que reúna a las estructuras EAPinturas, EARetratos, Eaartistas y que nos represente el menú principal de nuestra visión.

EAartistas := ESTRUCURA_DE_ACCESO

(Objetos que incluye : EAPinturas, EARetratos, EAartistas)

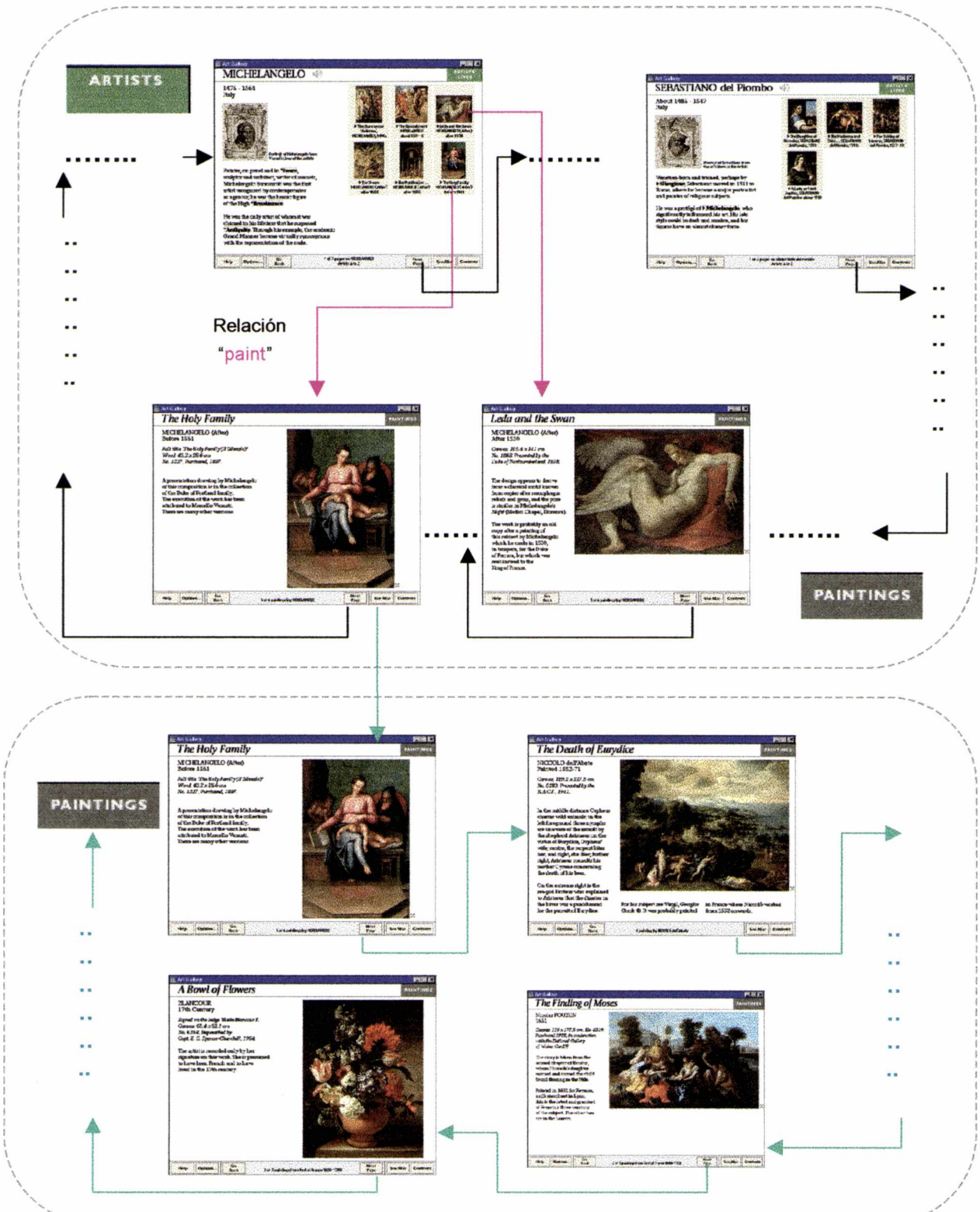


Figura 4.9: Representación gráfica del cambio de contexto

Conclusiones

En esta tesis hemos especificado un lenguaje de consultas, basado en OOHQL, para aplicaciones hipermediales diseñadas con el método OOHDM, permitiendo de esta manera al usuario o diseñador de dichas aplicaciones contar con una herramienta adicional a la hora de interactuar con las mismas. Este lenguaje brinda la posibilidad de recuperar información específica en forma simple y directa, atacando así la problemática de la desorientación con la que se enfrentan los usuarios al navegar sobre aplicaciones hipermedias mal diseñadas o aquellas que manejan un amplio dominio de información.

Esta tesis presenta una solución al problema antes mencionado. Dicha solución esta basada en un mecanismo de consulta de información a través del lenguaje de consultas definido en el Capítulo 3.

Una de las ventajas de contar con un nuevo mecanismo de consulta de información dentro de una aplicación hipermedia, es que las exploraciones de los usuarios se tornan más interesantes, al saber que cuentan con la posibilidad de reducir espacios de navegación a través de una consulta sencilla donde puedan especificar sólo los objetos pretendidos y las restricciones que éstos deben cumplir. De este modo, por medio de una consulta se reduce el espectro de información al dominio deseado y la navegación se realiza solo a través de los objetos que forman parte del resultado, evitando accesos navegacionales no deseados.

Específicamente, para recuperar los datos deseados se utilizarán las consultas de información definidas en el lenguaje, ya que las mismas se realizan sobre los nodos de la hipermedia donde se encuentra almacenada la información. Así mismo se cuenta con distintas funciones (Hay_relación, Existe, Esta_en, etc) que brindan una mayor flexibilidad al especificar las condiciones de búsqueda dentro de la consulta.

Luego de recuperar la información, el usuario podrá especificar la forma de recorrer la misma utilizando los constructores de navegación.

Otra de las ventajas, es que el diseñador obtiene un mayor conocimiento de la aplicación ya que se provee un conjunto de consultas que le permiten descubrir el esquema de la misma. Estas consultas, denominadas consultas de estructuras permiten conocer la estructura de la hipermedia, es decir verificar que clases están relacionadas, cuales son las relaciones entre las mismas y los atributos que forman parte de cada clase entre otros.

En resumen notemos que los dos niveles de consultas provistos por el lenguaje, orientados a la información y a la estructura de la aplicación, permiten construir visiones completas del sistema y, decidir luego como navegarlas utilizando a tal efecto los constructores de navegación aquí definidos.

Queda planteada, a partir de esta Tesis, la posibilidad de implementar el lenguaje de consultas aquí presentado. Debemos destacar que la forma en que se encuentre implementado OOHDM incidirá directamente sobre la implementación de este lenguaje.

A continuación analizamos en forma general dos alternativas de implementación de OOHDM. En el primer caso asumimos que la misma esta desarrollada en una base de datos relacional y, en el segundo, en una base de datos orientada a objetos.

Si la aplicación se encuentra implementada bajo un esquema relacional dicho esquema estará compuesto por tablas o entidades que contengan información de las clases definidas en el esquema conceptual OODHM y de las relaciones entre las mismas las cuales pueden obtenerse utilizando algún método de transformación como el deducido en Elmasri/Navathe [21].

Para ejecutar una consulta en este esquema, podría utilizarse el propio SQL relacional para resolverla, para lo cual es necesario un proceso capaz de identificar qué tablas de la base de datos relacional se corresponden con las clases especificadas en la consulta, para poder así reescribirlas en SQL. Dada la semejanza del lenguaje definido con el SQL estándar este proceso no presenta un alto nivel de complejidad.

Las consultas que contienen funciones tales como Existe, Hay_relación, etc., si bien son más complejas no agregarían gran dificultad a los procesos de traducción dado que las mismas pueden ser evaluadas a través del uso de las claves foráneas de las tablas.

Por otro lado, si la aplicación se encuentra sobre un esquema orientado a objetos, como OOHDM es un método de diseño orientado a objetos cada clase conceptual o relaciones entre clases estará representada en forma similar en la base de datos orientada a objetos, acompañadas por métodos que permitan recuperar información de las mismas.

A diferencia del esquema relacional, la forma de recuperar información dentro de una base de datos orientada a objetos debería ser directa, ya que un objeto en la hipermedia es un objeto dentro de la base de datos.

Por lo anterior, podemos decir que la implementación del lenguaje de consultas aquí definido, podría abarcar mas de un trabajo de investigación.

Apéndice A

1 Hipertexto e hipermedia

1.1 Definición

La forma más simple de definir hipertextos es compararlos con textos tradicionales, los cuales son leídos en forma secuencial, es decir sólo hay una secuencia lineal que define el orden en el cual el texto debe ser leído. Primero debemos leer la página uno, luego la página dos, y así sucesivamente hasta concluir la lectura del mismo.

El concepto de *hipertexto* es simple: permitir la posibilidad de unir documentos por medio de links y dejar al usuario saltar desde una pieza de información a otra.

Un sistema de hipertexto es un conjunto de documentos donde cada uno de ellos muestra en pantalla links visibles a otros documentos. Los links son representados por texto en diferente color, estilo de letra, etc. y se destacan dentro del documento. El usuario navega por el hipertexto seleccionando links, los cuales lo llevan a otros documentos que poseen links a documentos adicionales y así sucesivamente.

La figura A1 muestra un ejemplo. Asumimos que comenzamos a leer el documento A, en lugar de tener un solo lugar donde ir, la estructura del hipertexto nos da tres opciones para seguir B,D y E. Suponemos que decidimos ir al documento B, luego entonces podremos pasar a C o a E y desde E llegar a D o a F. También es posible ir desde el documento A al documento D atravesando sólo un link. Este ejemplo nos muestra que pueden existir varios caminos diferentes que conectan dos elementos en la estructura de un hipertexto.

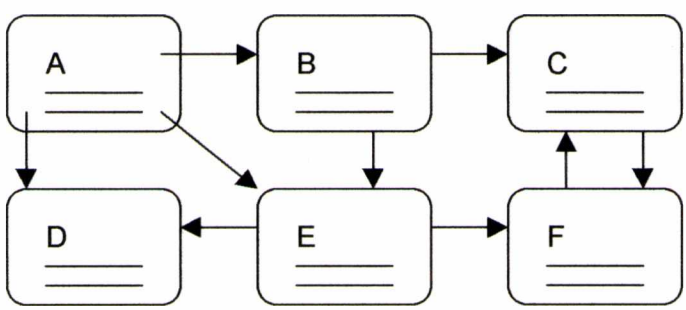


Figura A1. Visión genérica de la estructura de un hipertexto.

Un hipertexto presenta distintas opciones a los lectores, y es cada lector en forma individual quien determina cuál de todas esas opciones seguir en el momento mismo en que está leyendo. Esto significa que el autor ha establecido un número de alternativas a los lectores para explorar más que para seguir una cadena de información.

El termino *hipermedia* combina los conceptos de hipertexto y multimedia y se diferencia de los hipertextos sólo en la información que la misma posee en sus nodos, ya que en una hipermedia el contenido de los nodos puede ser además de texto, imágenes, sonido, video, etc.

1.2 Componentes de una Hipermedia

La estructura completa de una hipermedia muestra una red la cual esta compuesta, en principio de dos tipos de elementos, nodos y links, donde los *nodos* son los que poseen la información y están organizados implícita o explícitamente en una o más estructuras. Los *links* son los que nos permiten, dentro de la red, atravesar de un nodo hacia otro. El número del links normalmente no es fijo en primera instancia, pero dependerá del contenido de cada nodo. Algunos nodos están relacionados con muchos otros nodos y por lo tanto poseen muchos links, otros sólo son nodos de destino por lo que solo tienen links entrantes.

Un link conecta dos nodos y es dirigido, en el sentido que sale de un nodo origen a otro denominado nodo destino. Los links, pueden ser parte de un texto, un gráfico , un botón, etc.

Existe también un tercer componente denominado *anchor*. Se trata de un mecanismo para señalar puntos incluidos en el interior de los nodos que sirven de origen o de destino a un determinado enlace o link entre nodos. Por ejemplo, en un nodo con contenido textual puede haber palabras o frases marcadas que al ser seleccionadas por el usuario activen un enlace originando el acceso a otro nodo (o a otra parte del mismo nodo).

1.3 Método para recorrer una Hipermedia

La *navegación* es el proceso de moverse desde un nodo a otro en una aplicación diseñada bajo este paradigma. La navegación se produce al activar, dentro de la hipermedia, una serie de links. El atravesar distintos links dentro de una hipermedia, permite al usuario navegar dentro de la misma.

Los *sistemas de hipermedia* son entornos que ofrecen a los usuarios todos los mecanismos para crear, manipular y consultar hipertextos.

2 Métodos de diseño para aplicaciones hipermedias

Existen modelos para diseñar aplicaciones hipermedias, los cuales proveen un vocabulario de conceptos y primitivas predefinidas que pueden ser utilizados para especificar el contenido de los nodos. El uso de un modelo (o grupos de modelos) ayuda a describir la aplicación independientemente de su implementación.

La mayoría de estos modelos de diseño hace una clara distinción entre la noción de esquema y la noción de instancia de esquema. El *esquema* está compuesto por un

conjunto de clases, jerarquías de clases y, relaciones entre ellas, las cuales constituyen el dominio de la aplicación.

Los métodos de diseño de aplicaciones hipermedias procuran organizar el proceso de desenvolvimiento en una serie de actividades o tareas que describan la estructura del dominio de las mismas.

De esta forma la definición de un esquema conceptual ayuda a dar una visión del dominio de la aplicación hipermedia más abstracta de la obtenida por la inspección del código, o por la tratativa de extraer una semántica de las estructuras de los nodos y de los links.

Una aplicación particular en un dominio dado, es especificada por la *instanciación del esquema*. La noción de esquema y de instancia de esquema permite la reutilización del mismo esquema para aplicaciones diferentes en el mismo dominio. Estas aplicaciones comparten la misma estructura global, pero difieren en las instancias particulares que están presentes.

Luego de haber definido la instancia de un esquema se puede entonces especificar la semántica de navegación del mismo.

Durante la construcción del modelo navegacional de una aplicación hipermedia los objetos son mapeados en nodos y las relaciones entre ellos en links. En esta etapa se establecen qué nodos están relacionados y a qué nodos se podrá acceder desde un punto cualquiera de la aplicación.

La mayoría de los modelos permite definir la forma en que se puede navegar o sea permite definir como la información es visualizada por el usuario. En cada estructura se especifica qué nodos pueden ser accedidos dentro de la misma. Una estructura de navegación puede ser por ejemplo poder navegar por toda la red hipermedia o simplemente por una sub-red que posea información especifica.

HDM y EORM son métodos que se preocupan básicamente por la producción de un modelo conceptual de aplicación, mientras que, RMM y OOHDM son métodos de proyecto que definen diversas etapas donde son confeccionados modelos que no sólo detallan los aspectos conceptuales sino que también tienen en cuenta otras cuestiones como los aspectos de navegación e interface con o el usuario.

Vamos a describir cada uno de los métodos mencionados anteriormente exceptuando OOHDM que ha sido estudiado y comentado con detalle en el capítulo 2 de ésta tesis.

2.1 HDM (Hypermedia Design Model)

HDM [3] es un método de proyecto que propone un modelo para el desarrollo de aplicaciones hipermedias preocupándose de modelar el dominio de la aplicación con el objetivo de definir su estructura semántica. HDM no detalla otras etapas del ciclo de desarrollo de la hipermedia.

De acuerdo con HDM, el dominio de una aplicación puede ser modelado a través de un esquema compuesto por un conjunto de *tipos de entidades y relaciones*.

Cada entidad es constituida por una jerarquía de componentes estructurados que pueden ser vistos sobre diferentes perspectivas utilizándose unidades (menor elemento de información que puede ser visualizado en una aplicación HDM que se asemejan al concepto de nodo en un hipertexto).

El concepto de *entidad* en HDM posee diferencias con relación al del modelo entidades y relaciones. Las entidades en HDM pueden poseer estructuras internas complejas, semántica de navegación asociada a ellas, etc., en tanto, en el modelo de entidades y relaciones las entidades son planas.

Estructuras de información pueden ser interconectadas a través de *relaciones,* organizando a éstas en tres categorías:

- *Relaciones estructurales* que conectan componentes pertenecientes a la misma entidad intentando reflejar su estructura jerárquica.
- *Relaciones de aplicación* que son usadas cuando se desea conectar entidades o componentes a otras entidades o componentes.
- *Relaciones de perspectiva* hacen la conexión de unidades que pertenecen al mismo componente.

Un esquema HDM es un conjunto de definiciones de entidades y links de aplicación.

Una instancia de esquema HDM es un conjunto particular de entidades y links definido acorde al esquema dado. La especificación de una *semántica de browsing* en particular hace operativo al esquema de instancia. HDM no provee primitivas para especificar tal semántica de browsing, el mismo provee una semantica de browsing por default que es compatible con la mayoría de los sistemas de hipertextos card – oriented. Sin embargo deben ser definidas diferentes semánticas de browsing para describir más sofisticados efectos de visualización para aplicaciones especificadas con HDM.

En HDM, una aplicación hipermedia puede ser dividida en dos porciones: una base de datos hipertextual y un conjunto de estructuras de acceso.

- La base de dados representa el núcleo da aplicación. El usuario explora la base a través de las relaciones definidas. Pero antes de comenzar esta navegación, la aplicación debe presentar al usuario un conjunto de puntos de entrada que exhiban una visión de la base y dejar escoger a éste el punto de entrada mas conveniente.
- Las estructuras de acceso tienen como función permitirle al usuario que pueda seleccionar los puntos de entrada para nuevas incursiones a la base. Índices y tours guiados son ejemplos de estructuras de acceso.

Existen extensiones al modelo, como o HDM2, que acrecentan algunas nuevas primitivas de modelización.

2.2 EORM (Enhanced Object Relationship Model)

EORM [5] es un método para la modelización de aplicaciones en general que utiliza conceptos de la orientación a objeto para expresar los objetos del dominio y sus relaciones. A pesar de poder ser utilizado para el proyecto de aplicaciones en general, el método presenta características interesantes para modelización de aplicaciones hipermedia por utilizar conceptos de orientación a objetos y por proveer construcciones mas complejas para especificar la semántica de las relaciones entre los objetos.

En esta metodología propuesta por Lange [5], el proceso de desarrollo de un sistema de información hipermedia comprendería una fase de *Análisis* orientado a objetos de sistemas, sin considerar los aspectos hipermediales del mismo, obteniendo un modelo de objetos con la misma notación utilizada en OMT (Object Modeling Technique), que refleje la estructura de la información (mediante clases de objetos con atributos y relaciones entre clases) y el comportamiento del sistema (a través de los métodos asociados a las clases de objetos).

La idea fundamental de esta metodología es considerar una segunda fase, de *Diseño*, durante la cual se proceda a modificar el modelo de objetos obtenido durante el análisis añadiendo la semántica apropiada a las relaciones entre clases de objetos para convertirlas en enlaces hipermedias, obteniendo finalmente un modelo enriquecido, EORM, en el que se refleje tanto la estructura de la información (modelo abstracto hipermedial compuesto de nodos y enlaces) como las posibilidades de navegación ofrecidas por el sistema, sobre dicha estructura.

El desarrollo del sistema concluiría con una última fase de *Construcción,* durante la que se generaría el código fuente y se prepararía la interface gráfica de usuario. Esta etapa consiste de una biblioteca reutilizable de definiciones de presentación, donde una presentación consiste de uno o mas campos en los cuales un objetos puede exhibir su contenido.

EORM no separa explícitamente el proyecto conceptual del proyecto navegacional.

2.3 RMM (Relationship Management Methodology)

RMM [4] es un método de proyecto para el desarrollo de aplicaciones hipermedias cuyos principales conceptos fueron inspirados por el modelo entidades –relaciones. El método focaliza las etapas de modelización e implementación, principalmente las etapas que son destinadas a la especificación del dominio y de los mecanismos de acceso de la aplicación.

RMM posee un modelo de datos llamado Relationship Management Data Model (RMDM) que es usado para la construcción de modelos que describen los objetos de información y mecanismos de navegación de aplicaciones hipermedia. Las primitivas de RMDM están clasificadas en tres grupos.

- Las primitivas del dominio de entidades - relaciones describen cómo las informaciones están estructuradas en el dominio de la aplicación.
- Las primitivas del dominio RMD (Relationship Management Data) describen el modelo de *slices* (agregaciones de atributos de una entidad) de las entidades.
- Las primitivas de acceso describen los mecanismos de navegación presentes en la aplicación.

En el tercer grupo vemos las primitivas de acceso, los links unidireccionales y bidireccionales son utilizados para especificar los accesos entre *slices*. Los agrupamientos son mecanismos similares a los menúes que son utilizados para permitir el acceso a otros elementos del hiperdocumento o de otros mecanismos de acceso. Los índices condicionales son casos especiales de agrupamientos cuya lista de elementos es formada por instancias de una entidad que cumplan con un predicado.

[13] Franca Garzotto, Paolo Paolini y Daniel Schwabe. "*HDM – A Model for the Design of Hypertext Applications.* Hypertext'91 Proceedings, December 1991.

[14] Franca Garzotto, Luca Mainetti, Paolo Paolini. "*Hypermedia Desing, Analysis, and Evaluation Issues*". Communications of the Acm. August 1995/Vol 8,Nro 8

[15] T. Isakowitz, E. A. Stohr, P.B.Amann, M.Scholl. Milano. "*Gram: A graph data model and query language*". November 30 –December 4 1992.

[16] VI EBAI Laboratorio de Multimidia "*Tópicos em Multimidia*". Embalse Argentina 05-19 de Julio 1993

[17] Jacob Nielsen Bellcore. "*Hypertext and Hypermedia*". Morristown, New Jersey 1993.

[18] Won Kim. "*Introduction to Object Oriented Databases*". 1990 Massachusetts.

[19] Won Kim, " *Modern Database System* ".

[20] D. Lange. "*A Object Oriented design method for hypermedia information sysyem*". Proceeding of the 27th. Annual Hawaii Intenational Conference on System Science, Enero 1994.

[21] Ramez Elmasri. "*Sistemas de Bases de Datos, Conceptos Fundamentales*". Shamkant B. Navathe. Addison-Wesley Iberoamericana 1997

[22] Balasubramaniman " *A metodology for Structured Hypermedia Design* "