HERRAMIENTA PARA DISEÑAR Y CONSULTAR APLICACIONES HIPERMEDIAS

EXPOSION 5/12/98
Nota 8 (ocho)

MaBRodriguez

TRABAJO DE GRADO

HERRAMIENTA PARA DISEÑAR Y CONSULTAR APLICACIONES HIPERMEDIAS

Directores: Lic. Silvia Gordillo
Lic. Alicia Díaz

Alumnos: Sebastián López Brusa (30261/3)
Sebastián Marcón (29828/6)
Ceferino Alvaro Felipe Morales (30299/9)

Año: 1998

Agradecimientos:

- *Un agradecimiento especial a nuestras profesoras guías Alicia y Silvia, que en forma incondicional nos dedicaron todo de sí.*
- *A nuestros padres, por su apoyo permanente.*
- *A nuestras parejas: Gaby, Melina y Pitina (por orden alfabético) que soportaron nuestras constantes discusiones y peleas.*
- *Al Ruso y Viviana, que también las sufrieron por formar parte de nuestro entorno.*
- *A Martín, por su generosidad en facilitarnos materiales.*
- *A todos los que de alguna manera participaron en forma directa o indirecta para que lleguemos a esta instancia.*

Tabla de contenidos

RESUMEN

En este trabajo presentamos una herramienta para diseñar y consultar aplicaciones hipermedias basada en un modelo orientado a Objetos [Gordillo, Díaz]. El paradigma orientado a Objetos posee un conjunto de características, las cuales aprovechamos no solo para modelizar la información y facilitar el desarrollo de hipermedias, sino también para proveer diferentes alternativas de acceso a la información.

También presentamos un lenguaje de consulta basado en un lenguaje de consulta para un Sistema de Base de Datos Orientado a Objetos. Este combina las características de consultas a Bases de Datos Orientadas a Objetos y las primitivas para la navegación en una hipermedia. El lenguaje ofrece la posibilidad de consultar la información del dominio de la aplicación, y le permite a los diseñadores obtener información sobre el esquema de la aplicación.

1. INTRODUCCIÓN

La complejidad de los dominios de las aplicaciones hipermedias, la navegación a través de un espacio de información y construir la interface de la hipermedia para grandes aplicaciones, constituyen los principales problemas con los que uno se enfrenta al momento de construir aplicaciones hipermedias. Estas desventajas pueden ser superadas con un buen modelo de diseño.

Las aplicaciones hipermedias están caracterizadas por la manipulación de información, organizada como un grafo de nodos y links, donde el usuario navega con el objetivo de encontrar información específica. Modelar la información usando el modelo de nodos y links, y el acceso a éstos a través de la navegación nos permite obtener aplicaciones con gran flexibilidad. Sin embargo, el modelo de nodos y links con acceso navegacional no es suficiente para desarrollar y consultar aplicaciones hipermedias de gran tamaño. Desarrollar una metodología para diseñar hipermedias se ha convertido en una tarea muy importante porque existe una necesidad real de obtener aplicaciones en la que los conceptos tales como la reusabilidad, fácil mantenimiento, modularidad, etc. puedan ser usados para obtener un buen nivel de calidad en el producto final [Gordillo y Díaz].

Un problema importante en este campo es el de la "desorientación". El método primario para acceder a la información es a través de la navegación. Cuando la red (conjunto de nodos y links), es pequeña y se encuentra bien estructurada, la navegación probablemente será suficiente para accederla. Pero esto no siempre ocurre si la hipermedia tiene un conjunto grande de nodos y relaciones de diferentes tipos; o si el usuario quiere recuperar alguna información especifica. En ambos casos, es probable que el usuario tenga problemas para navegar, porque se podría encontrar perdido en el hiperespacio.

Si se agrega la posibilidad de consultar la información se obtiene una manera de

solucionar este problema. Esto se hace a través de herramientas adicionales que permitan la recuperación de la misma sobre la aplicación hipermedia. Por otro lado los diseñadores podrían necesitar acceder no solo a la información almacenada en la aplicación hipermedia, sino que también a la información estructural en cualquiera de las etapas del proceso de diseño. Esto es muy provechoso ya que le permite al diseñador entender la estructura de diseño de la aplicación que él mismo esta desarrollando.

"Hay algunas propuestas que definen un modelo para diseñar hipermedias, como [Garzotto 1993], [Isakowitz 1995], [Nanard y Nanard 1991] y [Schwabe 1995] las cuales describen una metodología para el diseño de la hipermedia, pero no definen como consultar la aplicación.

Por otro lado se ha puesto gran empeño en agregar la capacidad de consultar la información de las aplicaciones hipermedias como una posibilidad adicional al acceso navegacional [Amann y Scholl 1992], [Amann y Scholl 1994], [Concens y Mendelzon 1990]. Algunos de estos utilizaron bases de datos relacionales como una manera de proveer conocimiento sobre la estructura del grafo y los tipos de nodo, pero la información específica almacenada en el sistema no podía ser alcanzada a través de consultas. Por ejemplo, Parunak (1991) combina una ingeniería hipermedia con un Sistema de Bases de Datos Relacional, y de esta forma representa la información con atributos atómicos que pueden ser consultados. [Marmann y Schlargeter 1992] proponen una Base de Datos Orientada a Objetos como el soporte para aplicaciones hipermedias, en este caso, el acceso a la información es provisto usando el lenguaje de consulta específico de la base de datos." [Gordillo, Díaz]

Los modelos de diseño proveen un ambiente de trabajo en el cual los autores de aplicaciones hipermedias pueden desarrollar y analizar la organización conceptual de una aplicación en un alto nivel de abstracción.

Una ventaja importante de las aplicaciones desarrolladas de acuerdo a un modelo

radica en que resultarán en una estructura de representación muy predecible. Por lo tanto, el ambiente de navegación para los lectores debería ser predecible también, disminuyendo el problema de desorientación y sobrecarga de información.

Nosotros presentamos una herramienta para diseñar y consultar aplicaciones hipermedias basada en un modelo orientado a Objetos [Gordillo, Díaz], en la cual aprovechamos las características de este paradigma (herencia, modularidad, fácil mantenimiento, etc.).

A continuación haremos una descripción de dicho modelo, detallando la metodología y el lenguaje de consultas definido. Luego en las secciones siguientes explicaremos las características de cada una de las partes de la herramienta propuesta; enunciaremos algunos de los criterios de diseño aplicados en la navegación y en las consultas; brindaremos detalles de la implementación; y mostraremos una pequeña aplicación ejemplificando el modo de uso. Para finalizar indicaremos los inconvenientes suscitados durante el desarrollo y las soluciones adoptadas; analizaremos las posibles extensiones de la herramienta; y expondremos las conclusiones.

2. EL MODELO

Design and Query Strategies to Hypermedia Applications

Lo que sigue es una descripción del modelo orientado a objetos, en el que nos hemos basado para desarrollar la herramienta.

2.1. La Metodología

Básicamente, el modelo consiste de dos niveles de diseño. El primero, el *Diseño de Alto Nivel,* consiste en modelar los principales componentes de la aplicación descriptos en forma abstracta mediante la utilización de los conceptos de Modelización Orientada a Objetos como objetos, relaciones, jerarquías, composición y atributos. El Diseño de *Bajo Nivel* permite a los diseñadores describir características específicas de los sistemas de hipermedia como navegación, perspectivas, etc.

- **Diseño de Alto Nivel**

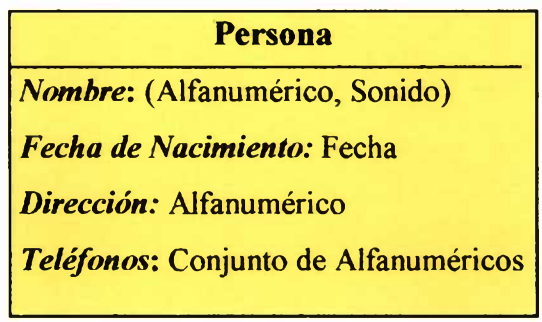
El objetivo de este nivel es pensar en el dominio de la aplicación de una manera conceptual obteniendo una representación de alto nivel de la aplicación.

Para desarrollar el modelo del dominio de la aplicación usaremos los conceptos de "objetos" y "relaciones" entre estos objetos.

Una clase describe un grupo de objetos con similares propiedades, común comportamiento, relaciones similares hacia otros objetos y una semántica común [Rumbaugh]. Los ***Objetos*** son representados en términos de sus atributos y comportamiento y son descriptos por clases llamadas *clases nodo.* Los tipos de atributos que se permiten son los atómicos, multivaluados (los que pueden tomar más de un valor) en la forma de lista o conjunto, y los atributos multidominio que pueden tomar valores de varios dominios permitiéndonos visualizar la

información de diferentes formas de acuerdo a la perspectiva que queramos.

La siguiente figura nos muestra un ejemplo de una clase nodo donde se destacan los tres tipos de atributos descriptos anteriormente:



En esta clase (**Persona**) tenemos un atributo multidominio denominado ***Nombre***, el cual puede tomar valores de los dominios Alfanumérico y Sonido; dos atributos atómicos (Fecha de Nacimiento y Dirección); y un atributo multivaluado definido como un Conjunto de Alfanuméricos (***Teléfonos***).

Las ***Relaciones*** entre clases nodo son también clases y por lo tanto están definidos a través de su comportamiento y atributos. Estas son llamadas *clases link* y siempre definen dos atributos: *from* y *to* que representan las clases origen y destino respectivamente, es decir desde qué clase nodo sale la relación y a cuál clase nodo llega.

Desde el punto de vista de la hipermedia, las relaciones definen vínculos para navegar a través de la información de la aplicación.

La relación **realizado por** es un ejemplo de vínculo entre una clase **Monumento** y la clase **Escultor:**

En este caso el atributo *From* se definió con la clase Monumento y el atributo *To*

con la clase Escultor.

Como ya dijimos las relaciones al ser clases llevan atributos que la describen. En el ejemplo anterior la relación **realizado por** podría contener el atributo *Fecha de Realización.*

Desde el punto de vista hipermedial es importante asegurarse que, cuando es posible, las relaciones no deben estar escondidas en atributos de las clases. Esto significa que cuando un atributo representa una entidad conceptual compleja con la intención de ser explorada en la hipermedia final, una relación debe ser especificada.

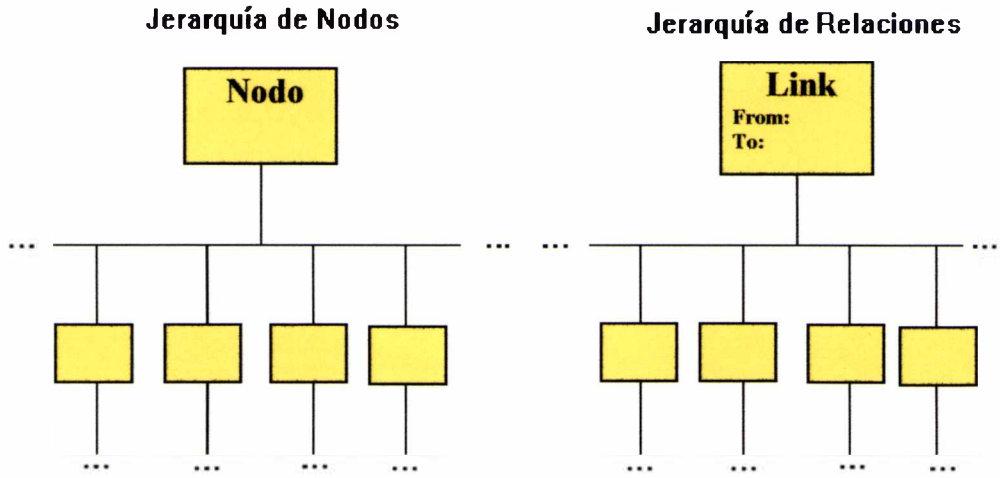
Por ejemplo, **Escultor** podría haber sido definida como un atributo de **Monumento.** Esto podría ser una correcta decisión de modelaje solamente si uno no quiere darle énfasis a la relación para navegar desde **Monumento** hacia **Escultor**.

La cardinalidad de la relación es también especificada en el esquema. Una relación puede tener cardinalidad **1:1** o **1:N**. O sea que un determinado objeto se puede relacionar sólo con un objeto o con más de uno según la cardinalidad elegida para la relación.

Existen dos tipos de relaciones especiales que tienen una semántica ya definida: las relaciones **es_un** y **partes_de.**

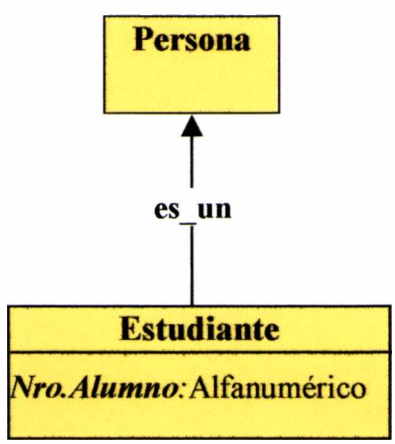
Relación es_un:

Al estar en un modelo orientado a objetos tanto las clases *nodo* como *link* son organizadas jerárquicamente por medio de la relación **es_un** donde la clase **Nodo** y la clase **Link** son consideradas las clases raíz respectivamente. Esto significa que cualquier otra clase que se defina (dependiendo de la jerarquía en que se lo haga) **es_ un** Nodo o **es_ un** Link



Obviamente la relación **es_un** tiene la misma semántica definida en el paradigma Orientado a Objetos. Esto significa que una clase (llamada subclase) hereda todos los atributos, comportamiento y relaciones de otra clase (su superclase) [Booch, 1994].

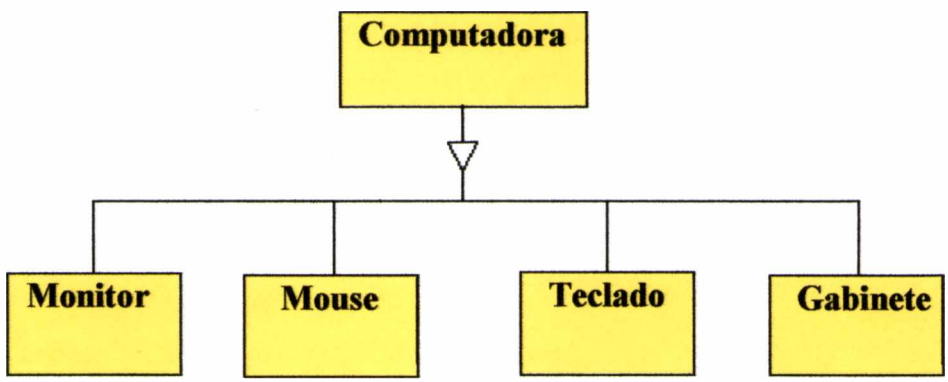
El que sigue es un ejemplo de relación **es_un** entre la clase **Persona** definida anteriormente y la clase **Estudiante**.



De esta manera los atributos de la clase **Estudiante** son el Nro. Alumno, más los que hereda de la clase **Persona** (Nombre, Fecha de Nacimiento, Dirección, Teléfonos).

Relación Partes_De:

Esta relación permite diseñar objetos compuestos. La composición semántica nos permite definir y manipular el objeto entero (incluyendo sus partes conceptuales) o uno de sus componentes específicos. Desde el punto de vista de la hipermedia, los objetos compuestos definen un contexto navegacional donde el usuario navega dentro del mismo objeto. El contexto navegacional es un conjunto de nodos con una estrategia de vinculación entre los miembros del conjunto [Gordillo-Díaz-Rossi]. El vínculo que se establece al definir la relación **partes_de** es similar al que se establece con cualquier relación, es decir desde un objeto se podrá navegar a cualquiera de los objetos que lo componen, de la misma manera que se navega a cualquier otra clase de objeto con el que se definió una relación conceptual (no especial). A continuación vemos un ejemplo de definición de **partes_de** donde se establece que la entidad **Computadora** está compuesta por la entidades **Monitor, Mouse, Teclado y Gabinete.**



Tanto las relaciones conceptuales como la relación **Partes_de** permiten definir al atributo *to* como un multidominio. Esto significa que una relación puede 'llegar' a más de una clase nodo, significando desde el punto de vista de la hipermedia que desde un nodo a través del mismo vínculo se puede llegar a nodos de distinta clase.

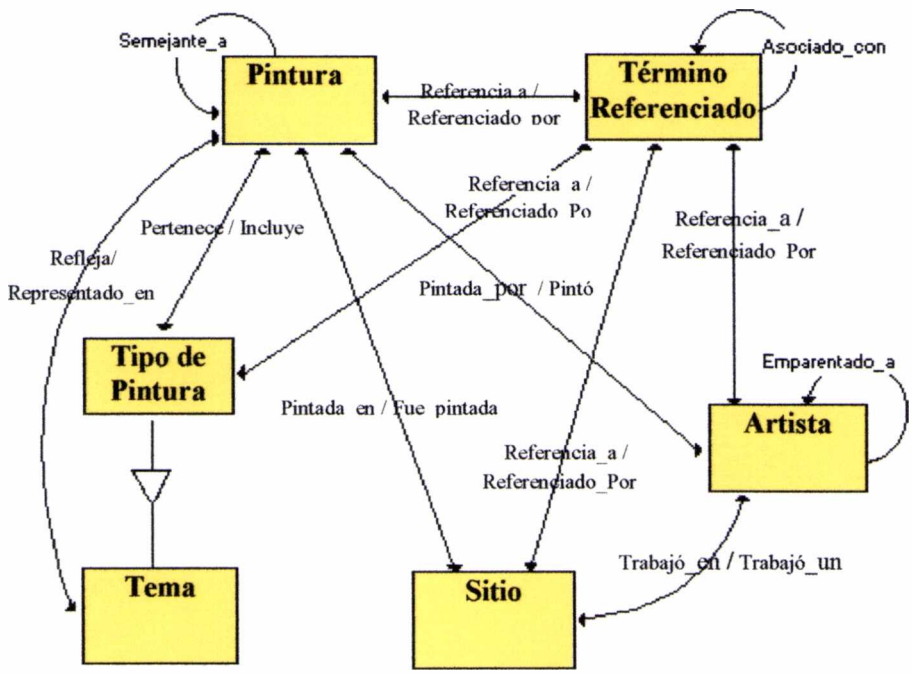
Descriptas las principales características de esta etapa nos queda por decir que el resultado de la misma no es muy diferente a la de los modelos construidos por diseñadores de Sistemas de Bases de Datos Orientadas a Objetos. En efecto, los elementos utilizados para modelar la aplicación son los mismos que aquellos

usados en cualquier Modelo Orientado a Objetos.

El esquema resultante está compuesto de un conjunto de clases de nodos que capturan la información a ser manipulada en la aplicación hipermedia, y las clases link que representan las relaciones entre esas clases nodo.

Un ejemplo: Galería de Arte

El diagrama que se muestra a continuación es un conjunto de clases de nodos y relaciones correspondientes al MICROSOFT ART GALLERY CD-ROM. Esta es una aplicación hipermedia que representa la célebre colección de arte de la National Gallery, en Londres y será utilizada como base para ejemplificar diferentes aspectos de este modelo.



El esquema que se ve en la figura anterior es parte del dominio de la aplicación, analizado en esta primer etapa del Diseño.

Las clases nodos graficadas en el esquema anterior se ven representadas de la siguiente manera:

Es importante notar que la clase **Artista** tiene un atributo multidominio que puede tomar valores en los dominios Alfanumérico y Sonido.

Además de las clases nodo, podemos ver la representación de algunas de las clases link del esquema:

Referenciado _Por
from: Término de Referencia
to: (Pintura,Artista,Tipo de Pintura,Sitio)

Pintado_Por
from: Pintura
to: Artista

Trabajo_en
from: Artista
to: Sitio

Pertenece
from: Pintura
to: Tipo de Pintura

En estas clases se ve la definición de los atributos ***from*** y ***to*** que siempre deben existir ya que indican la clase nodo origen y la clase nodo destino.

En la clase **Referenciado_Por** se puede observar el ***to*** definido como un multidominio (Los nodos pueden ser de las clases Artista, Pintura, Tipo de Pintura o Sitio). Este es un interesante caso donde se describe una relación general entre una clase nodo particular con otras.

- **Diseño de Bajo Nivel**

Las aplicaciones hipermedias tienen características adicionales que las hacen diferentes de las aplicaciones de bases de datos. Estas características aún no se han visto en este modelo ya que no han sido reflejadas en el nivel anterior. La intención es definir estas facilidades durante el Diseño de Bajo Nivel, en el cual incorporaremos tres aspectos importantes concernientes con los siguientes puntos:

- *Estructura Navegacional:* Se debe definir para mantener a la navegación como método primario de acceso a la información de la aplicación.
- *Perspectivas:* Muchas veces es necesario observar los objetos que se definieron desde distintos puntos de vista, ya que el total de la información que brindan puede no ser relevante en un momento dado. Para ello lo que se permite es mostrarlos desde diferentes perspectivas que inclusive pueden ser definidas por el diseñador.

- *Interfaces al Usuario:* Se pueden definir diferentes interfaces para visualizar cada una de las perspectivas de un objeto.

Para implementar las características enunciadas, definimos un concepto de diseño muy importante al modelo: El ***Ejemplar***.

Un ***Ejemplar*** establece diferentes disposiciones para la información descripta en el mismo nodo de una manera abstracta [Gordillo-Díaz]. Por lo tanto mientras las clases nodo representan la información conceptual de la aplicación especificando los objetos en la hiperbase, los ejemplares son usados para **navegar** porque proveen los posibles caminos desde un nodo y la información que será visualizada. Es decir que al permitir definir más de un ejemplar para un mismo nodo se logra que éste sea visualizado de diferentes maneras en la hipermedia, ya que cada ejemplar establece una perspectiva diferente para dicho nodo, constituyéndose así en el componente principal de la estructura navegacional

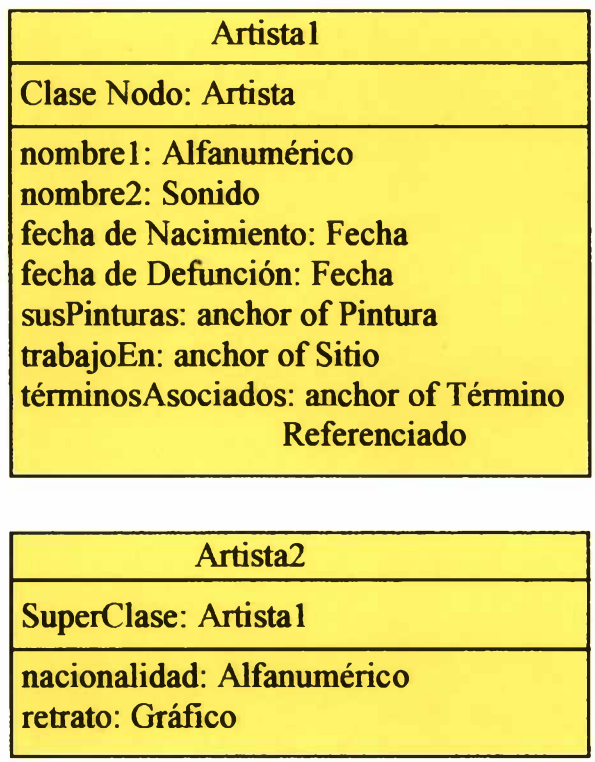
Un ejemplar es definido a través de una lista de atributos, una lista de comportamientos, y un conjunto de anchors. Los atributos que lo componen son seleccionados de los atributos que existen en la clase nodo asociada con el ejemplar (esta lista contendrá aquellos atributos definidos en la clase nodo que son de un interés particular para este ejemplar), y al elegirse se considera que la lista de comportamientos esperado es también seleccionada.

Cuando se diseña la navegación también se debe indicar el origen de los vínculos (links) dentro de los nodos. Aparece así el concepto de anchor. El ***anchor*** es la visualización del origen de un vínculo donde la clase nodo del ejemplar es el dominio del atributo *from*. Cada ejemplar tendrá los ***anchors*** seleccionados que van a ser mostrados en una perspectiva. Dicha selección se realiza sobre una lista conformada por todos los vínculos posibles, definidos en el nivel anterior, cuyo origen será representado por el anchor. El dominio del *from* de estos vínculos obviamente coincidirá con la clase nodo del ejemplar.

A través de la definición de varios ejemplares para la misma clase nodo, se pueden obtener diferentes perspectivas de la misma información. En definitiva, cada ejemplar constituye una perspectiva de la información donde el diseñador decide que información es relevante, como mostrarla, y que anchors definen la estructura navegacional.

Los atributos multidominio nos permiten definir diferentes puntos de vista para la misma información. Los diseñadores tienen que determinar que dominios serán utilizados en una perspectiva en particular y para cada uno de los seleccionados deben definir un atributo atómico como objeto visual, que tomará valores en dicho dominio.

La siguiente figura nos muestra dos ejemplares definidos para la clase Artista:



En el ejemplar Artista1 podemos observar el atributo *nombre* multidominio de la clase Artista el cual ha sido dividido en dos atributos *nombre1*, con dominio Alfanumérico y *nombre2* al que le corresponde el dominio Sonido. Se debe tener

en cuenta que no es necesario involucrar todos los dominios en un ejemplar. El diseñador podría haber seleccionado sólo un dominio para mostrar el atributo multidominio.

Al igual que las clases nodo y las clases link, los ejemplares son organizados en una jerarquía de especialización/generalización donde la clase raíz es la clase a la que se denominó **Ejemplar**. En la figura anterior se puede observar que el ejemplar Artista2 ha sido definido como una subclase del ejemplar Artista1, por lo tanto la información visualizada estará compuesta por todos los atributos y anchors de Artista1 que se heredan más los atributos *nacionalidad* y *retrato* que se añaden a la definición.

Por último analizaremos como las perspectivas son visualizadas. Mientras los ejemplares describen de una manera abstracta cual es la información manipulada, su representación gráfica nos permite determinar como será vista dicha información. Por cada elemento perteneciente al ejemplar, sea atributo o anchor, se indica un elemento visual que lo representa. Está definición es una tarea donde se utilizan técnicas para la interacción hombre/computadora, que le permiten al diseñador elegir la forma en que se visualizará cada ejemplar en la navegación que están fuera del alcance del punto de vista de la ingeniería de software.

Así concluimos con el análisis de dos aspectos muy importantes del modelo. El diseño de Alto y Bajo Nivel. Primero vimos como a través de las técnicas de modelización orientadas a objetos se definen todas las clases (nodos y links) que componen el esquema de la aplicación, sus propiedades y características. Luego a partir de dicho esquema, y a través de la definición de los ejemplares, con sus atributos, anchors y su representación gráfica vimos la manera en que el modelo permite representar las diferentes perspectivas con las que se desea visualizar la información de los nodos definidos en el esquema, logrando así la estructura navegacional. Con está estructura se logra mantener la navegación como método primario de acceso a la información de la aplicación.

Ahora veremos una descripción del lenguaje de consulta el cual permite acceder a la información de una manera directa, sin la necesidad de la navegación.

2.2. El lenguaje de Consulta

La existencia de un esquema de la aplicación nos lleva a incrementar las capacidades del modelo mediante la definición de un lenguaje de consultas que permite acceder a determinada información específica cuando la información es innecesaria o se vuelve un proceso dificultoso. Por ejemplo, teniendo en cuenta el ejemplo de la galería de arte, si el usuario quiere obtener todas las pinturas que fueron pintadas por artistas de nacionalidad italiana es mejor realizar una consulta que obtenerlas por medio de la navegación.

El lenguaje de consultas que vamos a describir nos permite recuperar dos tipos de información:

- La información sobre el esquema y está orientada a ayudar a los diseñadores en el proceso de diseño.
- La información sobre la hipermedia resultante y es provechoso para quienes lo utilizan como método de acceso, alternativo a la navegación.

El lenguaje de consultas especificado por [Gordillo-Díaz] en el modelo está basado en el definido por [Kim,1990], al cual se le han realizado algunos cambios y agregados para trabajar con aplicaciones hipermedias. En particular del lenguaje de consulta de Kim se ha adoptado el operador *Select*, los cuantificadores *each* y *exist* y los operadores de comparación. Se han agregado algunas funciones especiales para recuperar información concerniente con las relaciones entre objetos y sobre el esquema de la aplicación.

La siguiente tabla muestra todos los operadores del lenguaje de consulta. La tabla está organizada en dos partes. Una muestra los operadores utilizados para consultar la información de la aplicación y la otra los operadores usados en las

consultas sobre el esquema de la aplicación.

	Operación	Operador
Consultas sobre la Información	selección	Select *
	Proyección	Select *
	Cuantificadores	each,exist *
	Comparadores	=, = =, <, >, <=, >= *
	Funciones	isPartOf, related_By, path
Consultas sobre el Esquema		hierarchy
		properties
		source, target
		exemplar
		relatedTo, relatedFrom

Los operadores marcados con el asterisco (*) son los tomados del lenguaje de consultas de Kim.

Consultas sobre la información de la aplicación

La operación esencial para realizar consultas sobre la información de la aplicación es el operador selección, el cual es similar a la selección en Sistemas de Base de Datos Orientadas a Objetos. El significado de cada uno de las operaciones es el siguiente;

Selección: La selección de objetos de una o más clases es similar a la de bases relacionales.

Sintaxis: Select *targetClause*
from *rangeClauses*
[where *qualificationClause*]

targetClause: es la especificación de las clases nodo de salida.

rangeClause: indica el enlace de las variables con los correspondientes conjunto de instancias de las clases.

qualificationClause: es una combinación booleana de predicados.

[..] indica parámetro opcional

El siguiente ejemplo recupera las pinturas, y sus artistas, que hayan sido pintadas antes de 1870.

Select *Pintura, Artista*
from *Pintura:p*
where (p.fechaDeCreación<1870)

El resultado de una consulta es un subconjunto de la hipermedia original compuesto por todas las instancias de las clases nodo que figuran en *targetClause* y que satisfagan la cláusula *where.*

Cuantificadores: Las cláusulas de calificación del where pueden incluir atributos multivaluados. Por está razón el modelo incluye dos cuantificadores EACH y EXIST, que corresponden a los cuantificadores universal y existencial respectivamente.

Proyección: El operador de proyección permite la recuperación de algunos atributos de objetos pertenecientes a una o más clases. La operación es similar a los sistemas de bases de datos relacionales.

Sintaxis: Select *targetClause*
from *rangeClauses*
[where *qualificationClause*]

La única diferencia con la selección es que en targeClause se especifican los atributos de las clases nodo de salida.

La respuesta a este tipo de consultas es un conjunto de listas de los atributos especificados que cumplan la condición dada en el where.

En el siguiente ejemplo el resultado es un conjunto de lista de pares representando el nombre de los artistas y el nombre de sus pinturas que hayan sido pintadas antes de 1870.

Select *Artista.nombre, Pintura.nombre*
from *Pintura:p, Artista:a*
where (p.fechaDeCreación<1870) and related_by(a,pintó,p)

Related by: Esta operación permite establecer si hay una relación entre dos

objetos de dos clases nodo. El resultado de esta operación es un valor booleano que devuelve verdadero si existe la relación o falso en caso contrario. Es parte de la condición del where.

Sintaxis: related_by(*fromObject, relationshipName, toObject*)

El siguiente ejemplo muestra como recuperar los Artistas de pinturas de la que Tipo de Pintura es "retratos".

```
Select Artista
   from: Artista:a, Pintura:p, Tipos de Pintura:ps
   where (ps.nombre="retratos") and Related_by (p, pintado_por, a)
         and Related_by(ps, incluye, p))
```

IsPartOf: Es similar a la anterior pero en particular para la relación *partes_de*.

Sintaxis: IsPartOf(*fromObject, relationshipName, toObject*).

Path: Esta operación permite establecer si hay un camino de relaciones entre objetos de clases nodo.

El resultado de esta operación es un valor booleano que es verdadero si el camino existe, o falso en caso contrario.

Sintaxis: path(*fromObject, relationshipsList, toObject*)

donde *relationshipsList* es una lista de clases link.

El ejemplo que sigue recupera sitios donde trabajaron artistas que pintaron pinturas cuyo TipoDePintura es "retratos"

```
Select Sitio
   from Artista:a, Pintura:p, Tipo de Pintura:ps, Sitio:s
   where((ps.nombre="retratos") and
         path(p, pintado_por, trabajo_en, s)))
```

Consultas sobre el esquema de la aplicación

Esta clase de consultas es muy provechosa para el diseñador, ya que le permite obtener información sobre cualquiera de las jerarquías del esquema.
Las consultas posibles son:

Hierarchy: Esta operación se utiliza para establecer cuales son las superclases y subclases de una clase dada.

Sintaxis: *hierarchy className* From *Hierarchy_Type*

hierarchy indica si es ascendente o descendente, es decir si se recuperan las superclases o subclases.
className es el nombre de una clase de una jerarquía y *Hierarchy_Type* indica que jerarquía es: Nodo, Link o Ejemplar.

Properties: Recupera las propiedades definidas en una clase.

Sintaxis: properties *className* from *hierarchy_type*

hierarchy_type, y *classname* son las mismas de la operación anterior.

Source and target: Recupera las clases nodo origen y destino de una especifica relación.

Sintaxis: source *className* source

className es el nombre de una clase en la jerarquía de links.

Exemplar: Esta consulta recupera los nombres de los ejemplares que están asociados con una clase nodo específica.

Sintaxis: exemplar *className*

nodeClass es el nombre de una clase nodo

Related to and related from: La operación *related_to* determina que clases nodos son alcanzables desde una clase específica, y *related from* define todas las clases nodos desde los cuales esa clase específica puede ser alcanzable.

Sintaxis: related_to *nodeClass*

related_from *nodeClass*

nodeClass es el nombre de una clase nodo.

3. LA HERRAMIENTA PROPUESTA

Como hemos dicho la idea básica es brindar una herramienta que permita desarrollar aplicaciones hipermedias, basada en el modelo descripto anteriormente. Con la misma, el usuario podrá diseñar un esquema en el que quedarán definidos los componentes de la aplicación, navegar a través de la hipermedia resultante y realizar consultas sobre el contenido, tanto del esquema de la aplicación como de la información.

Lo que sigue a continuación es una descripción de la herramienta enunciada:

3.1. Características Generales

Para el desarrollo de la herramienta nos hemos basado en tres aspectos fundamentales como lo son: el diseño del esquema de la aplicación, la navegación y las consultas.

Para el primero, la herramienta consta de un **Editor del esquema de aplicación**, donde el usuario puede diseñar su hipermedia utilizando todos los elementos que conforman el modelo. Es decir puede definir clases de nodos y de relaciones, sus atributos, los ejemplares, la representación visual, indicar la jerarquía entre clases, consultar el esquema y establecer otras características concernientes al mismo.

Por otra parte la herramienta provee un **Navegador**, que será el elemento con el cual, a través de la navegación propiamente dicha, se obtiene una de las maneras de acceder a la información de la hipermedia resultante del diseño. La otra forma es por medio de las consultas que, basándose en el lenguaje establecido, se definen desde el **Editor de Consultas.**

La herramienta posee también un **Editor de Instancias** el cual le brinda al

usuario la posibilidad de definir las instancias de cualquiera de las clases definidas en el esquema, dándole valores a sus atributos.

En la siguiente tabla se pueden observar los módulos anteriormente citados, destacándose sus principales funciones.

Módulo	*Funciones*
Editor del esquema de la aplicación	• Definición de Clases de nodos y links. • Definición de los ejemplares y su representación gráfica. • Representación jerárquica de las clases. • Consultas sobre el esquema.
Navegador	• Acceso a la información de la aplicación mediante la navegación
Editor de Consultas	• Formulación de las consultas sobre la información de la aplicación • Acceso directo a la información.
Editor de Instancias	• Instanciación de nodos y links.

3.2. Editor del esquema de aplicación:

Este editor es el elemento con que cuenta el usuario para definir su esquema conceptual. Esta conformado por un diagrama en el que se reflejan las clases nodo (entidades) y las relaciones, que se hayan definido; y por un esquema jerárquico en el que se muestran las mismas clases pero con una visión en la que se le da énfasis a la jerarquía existente entre ellas.

En ambos casos las operaciones que se pueden realizar son las mismas. Esto es agregar una entidad, una relación conceptual, una subclase, definir la relación Partes_De (considerada como Compuesta_Por en nuestra herramienta), definir y

visualizar los ejemplares y su representación gráfica, etc.

Agregado de una Entidad:

La creación de una **Entidad** implica agregar una nueva clase nodo conceptual al esquema que se está diseñando. La herramienta provee la facilidad no solo de identificar a la entidad con un nombre unívoco dentro del esquema, sino también la de ir incorporando los **Atributos** que caracterizarán a la clase nodo que se está por crear. Dicha clase, puede ser, o bien agregada simplemente como una nueva entidad al esquema (que será considerada subclase de la entidad conceptual raíz 'Nodo'), o bien como una subclase de alguna entidad creada previamente. Es importante aclarar que al agregar una entidad o una subclase de una entidad, la herramienta generará automáticamente la relación 'es_un' que vincula la nueva clase con su entidad superclase considerando siempre, en el primero de los casos, la clase 'Nodo' como la clase 'padre'. Es decir la definición de la relación 'es_un' se genera de forma implícita al agregar una subclase y no de forma explícita.

Definición de Atributos:

Cuando se agrega una entidad (o como veremos luego una relación) la herramienta brinda la posibilidad de agregar los atributos que la caracterizan.

Cuando se definen los **Atributos**, se deberán agregar todos aquellos que sean 'propios' a la clase en cuestión. Recordemos que por tratarse de un modelo orientado a objetos se heredan todos los atributos que hayan sido definidos en las clases antecesoras.

La definición de los atributos consiste en indicar el nombre, el tipo de dominio y el tipo del atributo. Los tipos de dominio con los que se puede definir un atributo son los siguientes: **simple** (o sea atómico), **múltiple** (varios tipos de datos declarados simultáneamente para un mismo atributo) o **multivaluado** (varios valores para un mismo tipo). En este último, se debe definir si estarán

organizados en forma de lista o de conjunto.

Los tipos validos que permite la herramienta para un atributo simple son: alfanumérico, entero, real, fecha, texto, gráfico, sonido o animación.

Los alfanuméricos pueden tener tamaño 1,2, ...80. Por su parte, la manera de definir los atributos que sean tipos textos, gráficos, sonidos o animación, es indicando el camino del archivo físico cuya extensión será 'TXT' en el caso de ser texto, 'BMP' si es un gráfico, o 'WAV' y 'AVI' para el sonido y para la animación respectivamente (se eligieron estos formatos de archivos como representativos de cada uno de los tipos, pero cualquier otro podría ser permitido con una simple modificación en la herramienta). La simbología utilizada para identificar el tipo de atributo es la siguiente:

A[n]	Atributo atómico Alfanumérico de tamaño n
E	Atributo atómico Entero
R	Atributo atómico Real
F	Atributo atómico Fecha
T	Atributo atómico Texto
G	Atributo atómico Gráfico
S	Atributo atómico Sonido
N	Atributo atómico Animación
ta1, ta2,...tan	Atributo multidominio de los tipos atómicos ta1, ta2, ...tan
L(ta1)	Atributo multivaluado Lista del Tipo atómico ta1
C(ta1)	Atributo multivaluado Conjunto del Tipo atómico ta1

Es importante señalar que la herramienta fue realizada considerando **herencia restrictiva**. Esto significa que se permitirá la redefinición de un atributo sólo si el conjunto de valores admisibles para el tipo del atributo que redefine es un subconjunto de los valores permitidos para el tipo del atributo redefinido, aclarando que si un atributo es multidominio en una redefinición se debe respetar la herencia en todos los tipos que lo componen, o sea para cada dominio del

atributo redefinido debe existir en el nuevo atributo al menos un dominio que sea un subconjunto. Dados los tipos de atributos que se permiten usar en la herramienta, ocurre que ningún dominio de valores de un atributo es subconjunto de otro (excepto en los atributos especiales From y To de una relación). De esta manera, en la práctica, el usuario nunca tendrá la posibilidad de redefinir un atributo, aunque esto en teoría es valido por tratarse de un modelo orientado a objetos.

Dentro de una misma clase no se permite que haya más de un atributo con el mismo nombre. Obviamente tampoco se podrá declarar un atributo con el mismo nombre que uno perteneciente a una clase antecesora, puesto que ya se lo hereda.

Agregado de una Relación:

La herramienta provee la manera de agregarle al esquema **Relaciones conceptuales** y la relación especial **'Compuesta_por'**. Como vimos anteriormente, a la relación **'es_un'** no se la debe indicar explícitamente, ya que el sistema las crea en forma implícita cuando se define una subclase.

Cuando se añade una relación conceptual, al igual que con las entidades, dicha clase puede ser, o bien agregada simplemente como una nueva relación del esquema (que será siempre considerada subclase de la relación conceptual raíz 'Link'), o bien como una subclase de alguna relación creada previamente. Se debe tener en cuenta que para la definición de los atributos **From** y **To**, el modelo permite definir mas de una entidad en ambos casos. Sin embargo en la herramienta, para un mayor entendimiento, permitimos que solo el atributo **To** pueda ser multidominio, limitando a 1 (una) la cantidad de entidades posibles a definir en el atributo **From**.

Si la relación que se esta agregando es una subclase de una ya existente, se heredan los atributos **From** y **To** de la clase antecesora pudiendo ambos ser redefinidos. En este caso por la característica de herencia restrictiva que posee la herramienta se pueden indicar solo entidades que sean subclases de las entidades

definidas en los atributos From y To heredados, considerándose heredadas aquellas entidades que no sean redefinidas.

Una aclaración importante es que en el atributo To (el único que puede ser multidominio) no se permite que haya dos clases que pertenezcan a la misma cadena de herencia. Es decir que una de ellas sea antecesora de la otra.

A cada Relación se le debe definir la cardinalidad que posee, la misma puede ser **1:1** o **1:N**, es decir si una única instancia de entidad definida para el From esta relacionada solo con una o con varias instancias de una entidad del To.

Si la relación que se está definiendo es una subclase de otra ya definida, se hereda la cardinalidad de la antecesora, pudiéndose redefinir solamente si la heredada no es de cardinalidad 1:1.

Al igual que en la definición de las entidades, cuando agregamos una relación conceptual al esquema, se debe identificar unívocamente el nombre y los atributos que la componen, y para ello se procede de forma análoga a la descripta para los atributos de las entidades, ya que estos últimos poseen las mismas características que los anteriores.

Con la relación 'Compuesta_Por' se procede de la misma manera que con una relación conceptual. La diferencia radica en que no existe una relación de jerarquía entre este tipo de relación especial, y además no se permite la definición de atributos.

Definición de ejemplares:

Al agregarle un ejemplar a una entidad determinada se debe indicar, además de su nombre, cual es su antecesor inmediato en la jerarquía de ejemplares (tengamos en cuenta que todos los ejemplares también conforman una estructura jerárquica cuya raíz esta dada por el ejemplar abstracto denominado **Ejemplar**), y cual es el conjunto de atributos y anchors que lo conforman.

El ejemplar padre a seleccionar se obtiene de la jerarquía pudiendo ser cualquiera de los ejemplares definidos para la entidad en cuestión, o para las entidades que

son ancestras de dicha entidad.

Para el agregado de atributos y anchors se debe tener en cuenta que al ser un modelo orientado a objeto se heredarán todos aquellos que estén definidos en los ejemplares antecesores.

El conjunto de atributos factibles a ser definidos en el nuevo ejemplar esta compuesto por todos los atributos pertenecientes a la entidad a la que se le esta agregando el ejemplar (obviamente se incluyen los que esta hereda), menos todos aquellos heredados (los que pertenecen a los ejemplares ancestros). Vale aclarar que si el atributo es multidominio se puede seleccionar, si se quiere, solo alguno de los dominios.

Un anchor en un ejemplar surge a partir de una relación definida en la que la entidad, o nodo, del nuevo ejemplar es el origen de esa relación. De esta manera, el conjunto de anchors posibles a ser definidos en el nuevo ejemplar surge de todas las relaciones en las que el dominio del From es la entidad a la que se le esta agregando el ejemplar, o una de sus entidades antecesoras, exceptuando las que ya hayan sido elegidas para definir alguno de los anchors que se heredan. Si el atributo To de la relación es multidominio, se determina un anchor por cada una de las entidades que forman parte de ese dominio, pudiendo, si se quiere, seleccionar solo alguno de ellos. Cuando se elige el anchor se debe indicar también cual es el ejemplar con el que se quiere visualizar el nodo al que conduce el anchor desde el ejemplar que se está agregando.

Una aclaración importante es que cada vez que se da de alta una nueva entidad, la herramienta genera automáticamente un **ejemplar por defecto**. Esto significa que cada nodo tendrá garantizada en la aplicación una perspectiva básica dada por dicho ejemplar (el cual contendrá **todos** los atributos y anchors posibles), como forma de visualizar su información. El nombre dado al ejemplar por defecto será el del nombre de la entidad, y el número '0' concatenado al final. Por ejemplo, el ejemplar que se genera al definir la clase conceptual 'Persona' será 'Persona0'.

Representación gráfica de un ejemplar:

El usuario tiene la posibilidad de definir la representación gráfica de cada uno de los ejemplares. Esto significa que puede manipular los componentes (tanto a los atributos como los anchors) de forma tal que determine cual será la ubicación de visualización dentro de la perspectiva, definiendo así como se verán durante la navegación. Además de su ubicación, podrá definir otras características tales como el color, el tamaño, el tipo y tamaño de la letra (en los componentes que tengan), borde, transparencia, imagen de fondo, etc. Todas ellas pueden ser modificadas, pudiendo el usuario generar la perspectiva deseada para cada nodo. También es de definición del usuario el tamaño de la ventana en la que se visualizará la información del nodo. Esta propiedad también forma parte de la representación visual de los ejemplares y para la misma se usaron dos criterios diferentes. Uno consiste en que la ventana ocupe un espacio fijo conformado por el máximo tamaño posible de muestreo, y el otro es considerar que el tamaño y ubicación sean determinados por el diseñador. Como veremos más adelante ambos criterios son utilizados para decidir si se podrá navegar desde el nodo, o si sólo se mostrará su información.

Consultas al Esquema:

Nuestra herramienta permite que en cualquier momento de la etapa de diseño, el usuario pueda realizar todas las **Consultas del Esquema** que el modelo define, las que están relacionadas con los siguientes puntos:

- Jerarquía del esquema, ya sea de las entidades, de las relaciones o de los ejemplares.
- Propiedades de las clases nodo, como por ejemplo atributos, relaciones que parten de él y relaciones que llegan a él.
- Propiedades de las clases link como por ejemplo atributos, cuál es la clase nodo origen y cuáles son las clases nodos destino.
- Ejemplares que están asociados a una clase nodo dada.

No existe un módulo dedicado a realizar las consultas. Las mismas son posibles de formularse en cualquier momento en el que se esté manipulando a cualquiera de las clases del esquema

3.3. Navegador:

El navegador, es la parte de la herramienta que le permite al usuario recorrer la hipermedia diseñada. El mismo se nutre del esquema generado previamente y de la instanciación de las entidades y relaciones definidas. Posteriormente veremos como se realiza dicha instanciación.

A través de la navegación el usuario podrá 'moverse' entre los nodos definidos, para acceder a la información que está dada por la instancia que se muestra y visualizada según una perspectiva que es determinada por el ejemplar.

El navegador tiene un **Nodo inicial** que es configurado por el usuario, quien debe indicar cual es la entidad, ejemplar e instancia que conforman el nodo inicial de la hipermedia. A partir de él podrá ir accediendo a la información de los nodos con los que esté vinculado a través de los anchors, los cuales conducen hacia un nuevo nodo si es que se ha instanciado la relación asociada con el anchor elegido, caso contrario la navegación no tendrá efecto. Si la relación asociada al anchor es de cardinalidad 1:N, para acceder al nuevo nodo, el usuario deberá previamente seleccionar una instancia definida de una lista que se despliega al activar el anchor, para poder obtener la información que desee.

El usuario solamente podrá navegar desde aquellos nodos que se muestran en forma maximizada. Los otros nodos se proporcionan para casos en los que el diseñador de la hipermedia desee brindar más información que la que se visualiza a través de ventanas, que también se activan por medio de un anchor, quedando visualizadas por encima del nodo en el que se estaba parado. Son de un tamaño menor a las que se muestran en forma maximizada y deberán ser cerradas, antes de continuar navegando. En el caso que el ejemplar elegido para mostrar un nodo

en forma no maximizada contenga anchors, éstos no serán visualizados ya que no se puede navegar.

Vale aclarar que la herramienta cuenta con un mecanismo que va armando automáticamente una estructura navegacional por defecto, la misma permite que aquellos nodos a los que aún no se le haya definido un ejemplar, igualmente podrán ser accedidos dado que se utilizará el ejemplar generado por defecto.

Nuestra herramienta brinda también a los usuarios la posibilidad de navegar a través de ella, no solo accediendo a la información de los nodos haciendo uso de los anchors, sino también mediante la **historia de nodos visitados**.

La otra manera de recuperar información de la hipermedia es a través de consultas las cuales permiten el acceso a la información de un nodo en forma directa. Estas consultas se generan como se explica a continuación.

3.4. Editor de Consultas:

La herramienta provee un editor, mediante el cual el usuario podrá formular, en cualquier momento de la navegación, las consultas que desee tanto para obtener información de la hipermedia, como para acceder rápidamente al nodo que tiene la información recuperada . Las consultas que se pueden realizar se basan en el lenguaje descripto en el modelo.

En la edición de la consulta, se deben indicar las entidades involucradas en el **Select** y en el **From**, los atributos que se quieren visualizar si se realiza una **Proyección**, y se debe armar el **Where** que condicionará la consulta. La definición de cada consulta permite aplicar cuatro tipos de condiciones a la información que se desea seleccionar: atributos, path, related_by y composed_by.

- Atributos: La herramienta permite realizar filtrado de información haciendo uso de los operadores lógicos con los atributos de las entidades seleccionadas en el From, siendo estos atributos del tipo alfanumérico, reales, enteros y fechas. Las condiciones se pueden armar entre elementos del mismo tipo, pudiendo ser el

segundo operando un valor editado por el usuario. Vale aclarar que también se permiten comparar dos atributos alfanuméricos, aún en los casos en que tengan diferentes tamaños en su definición. También se pueden realizar comparaciones entre un atributo simple del tipo de los enunciados y un atributo multivaluado que contenga elementos de dicho tipo, mediante las cláusulas **Each** y **Exist**. Una comparación donde intervenga la cláusula Each dará afirmativa si todos los elementos del atributo multivaluado cumplen la condición, y cuando intervenga la cláusula Exist, dará afirmativa si existe al menos un elemento que la cumpla.

- Path: El usuario debe seleccionar la entidad origen, la entidad destino y uno de los posibles caminos entre ellas. Un camino es una sucesión de vínculos que pasando por nodos intermedios, permite llegar de la entidad o nodo origen al destino. Esta condición será afirmativa si existe al menos una sucesión de instancias que verifiquen el path.
- Related_by: Tiene las mismas características de funcionamiento que el path, con la diferencia que el camino es de un solo vínculo.
- Composed_by: El usuario deberá seleccionar la relación Compuesta_Por deseada, entre todas las que se definieron en el esquema. Está condición, de la misma manera que las dos anteriores, será afirmativa si existe al menos una instancia que la verifique.

Como el modelo no permite consultas de naturaleza recursiva no es posible seleccionar dos veces la misma entidad en el **From**.

La respuesta del sistema a las consultas, puede ser de dos tipos, dependiendo del tipo de selección realizada por el usuario. Si realiza una selección que sea una **proyección**, el resultado será la muestra de los atributos que eligió en la consulta (solamente se permite consultas sobre los atributos de tipo alfanumérico, real, entero o fecha), cuyas instancias cumplan las condiciones establecidas en la consulta. Si realiza en cambio una selección de entidades, el sistema proveerá una estructura de acceso a la información de los nodos, mostrando una lista de

anchors que tienen como etiqueta el nombre de las instancias que cumplan con las condiciones definidas en la consulta. De esta manera el usuario podrá navegar hacia alguno de ellos, y luego regresar a la resultante de la consulta para acceder a otro.

3.5. Editor de Instancias:

Hasta aquí hemos descripto las partes más importantes que componen nuestra herramienta. Además de ellas existen otras que son de menor relevancia, pero que son provistas para que el usuario pueda hacer uso de la aplicación.

Una vez que se haya definido la estructura general de las clases conceptuales, tendrá que realizar la instanciación de sus entidades y links para obtener así la información de los nodos y los vínculos que los relacionan entre sí. Para ello cuenta con el **Editor de Instancias**, que le permite ir seleccionando la entidad deseada y luego agregarles todas las instancias de la misma con los valores de los atributos. Cada nueva instancia debe tener un nombre identificatorio dentro de una misma entidad.

Los atributos de tipo alfanumérico, entero, real y fecha son editados por el usuario, mientras que a los textos, gráficos, sonidos y animaciones se les debe seleccionar la dirección física que poseen en el la unidad periférica de almacenamiento. Los atributos multidominio son 'divididos' como si se trataran de atributos simples, y se los instancia como tales. Los atributos multivaluados pueden ser o bien del tipo lista, cuyos datos deben ser ingresados en el orden en que aparecerán en la hipermedia, o bien del tipo conjunto, los cuales obviamente no permiten la repetición de sus valores.

Este editor posibilita la generación de las instancias de las relaciones, es decir, cuando se está posicionado en una instancia de una entidad, se puede acceder a la instanciación de las relaciones conceptuales de las cuales la entidad con la que se está trabajando, actúa como punto de origen de la relación (From). Esto hace que

cuando el usuario de la hipermedia acceda al nodo de la instancia de esta entidad, encontrará un anchor representativo de la relación conceptual que al activarlo servirá como medio para navegar hacia la instancia elegida de la entidad destino.

4. CRITERIOS DE DISEÑO

Antes de explicar aspectos relacionados con la implementación de la herramienta, enunciaremos algunos criterios adoptados en el diseño de la Navegación y de las Consultas.

4.1. Navegación:

Los aspectos que se comentan a continuación están relacionados con la visualización de la información a la que se accede durante la navegación y el significado de las componentes que se observan.

Navegando el usuario irá accediendo a la información de los distintos nodos definidos en la hipermedia. El espacio de visualización de dicha información, cuya distribución y forma de mostrarse queda determinada por el ejemplar con el que se muestra el nodo, tiene algunas características. El mismo es de tamaño fijo (salvo que sea el caso de un nodo que se ha decidido mostrar en forma no maximizada) y ocupa el máximo posible de visualización de la pantalla. Sin embargo esto no significa que la información que se ve sea la única disponible ya que también existe un mecanismo de scroll a través del cual se puede acceder a otras componentes (atributos o anchors) que no están visibles, no porque esa perspectiva no los muestre, sino porque no están en ese momento en el espacio de muestreo. Además, el sistema cuenta con componentes que poseen desplazamiento interno propio, como por ejemplo los atributos del tipo 'Texto', que pueden tener un tamaño que no necesariamente alcance para mostrar el total del contenido. Lo mismo ocurre con los atributos multivaluados (conjuntos y listas) que en un espacio de visualización con tamaño definido por el usuario, se pueden observar uno a uno los valores instanciados, logrando así mostrar variada información en un mismo espacio.

Con respecto a como se verán los anchors durante la navegación, podemos decir que los mismos aparecerán, por defecto con el nombre del nodo al que conducen en forma subrayada. Gracias a la facilidad de manejo de la representación visual de los ejemplares, el mismo puede verse sobre otro objeto, e incluso transparente. Un caso particular del tratamiento de los anchors, fue dado a las ventanas que no estén maximizadas, puesto que no aparecerán en ellas. Dichas ventanas tendrán un comportamiento al estilo ventana pop-up, ya que la única forma de poder seguir navegando hacia otro nodo será cerrando dicha ventana y regresando al nodo que la activó. Cabe aclarar que por defecto los textos que identifican los anchors serán de color azul y cuando el usuario (durante el recorrido de la hipermedia) pase por un nodo en el que hay un anchor que ya ha sido activado, el mismo será identificado ya que su texto será de color violeta.

Nuestra herramienta brinda a los usuarios la posibilidad de navegar a través de ella, no solo accediendo a la información de los nodos a través de los anchors, sino también mediante la **historia de nodos visitados**. El criterio utilizado para guardar dicha historia fue el siguiente: cada nodo accedido es incorporado en el último lugar a la lista de la historia de nodos visitados, guardándose en la misma el nombre del nodo accedido, el nombre del ejemplar con el que se está visualizando, y la instancia que es la que determina la información mostrada. Con estos datos es posible que cuando el usuario lo desee, pueda acceder nuevamente a la información que ya ha visto, con la simple utilización de la historia. Además, haciendo uso de las componentes 'Anterior' y 'Siguiente', podrá retroceder o avanzar respectivamente a cada nodo en el camino armado por el navegador. De esta manera, podrá regresar hacia ellos por el mismo camino que lo condujo hasta el nodo activo. En caso de estar 'de regreso' en uno de ellos y de tomarse 'un camino alternativo' de navegación a través de un anchor, el navegador descarta la historia de los nodos recorridos a partir del nodo actual, y considerará al nodo activo como el anteúltimo visitado y al que se viaja como el último.

4.2. Acceso a través Consultas:

Un tema de suma importancia a la hora de implementar una hipermedia es la metodología que brinda la misma para recuperar la información. Si se tratara de un hipermedia pequeña, la búsqueda de ella podría lograrse a través de la navegación. Si en cambio el espacio hipermedial es grande, se requerirá del uso de 'consultas' para acceder a ella. Una forma de implementar las consultas sería que la hipermedia posicione al usuario en el primer nodo que encuentre que satisfaga las condiciones estipuladas en ellas. Esta forma de implementar las consultas trae aparejado el hecho de que el usuario desconoce qué cantidad de nodos satisfacen su petición. Más apropiado es resolver la consulta con el despliegue de un menú que ofrezca la lista de las identificaciones de aquellos nodos que satisfagan las condiciones establecidas en ella, pudiendo el usuario acceder al nodo que seleccione en dicho menú.

Nuestra herramienta brinda una estructura de acceso a la información basada en la metodología recientemente descripta. La misma utiliza el lenguaje de consultas estandarizado que se describe en el modelo y que les permite a los usuarios definir sus consultas (queries) y obtener en forma fácil y eficiente la manera de acceder a la información deseada. Si lo que se realiza es una selección de entidades, el sistema proveerá una estructura de acceso a la información de los nodos, mostrando una lista de anchors, pudiendo de esta manera el usuario, navegar hacia alguno de ellos. Cada vez que un anchor es utilizado para acceder al nodo, el navegador lo marca como un acceso a un nodo ya visitado.

5. IMPLEMENTACION

5.1. Características generales:

La herramienta fue implementada en su totalidad con **Delphi 3.0,** utilizándose una base de datos relacional con archivos en formato **Paradox.**

La misma consta de un ambiente principal desde el cual se puede acceder a los módulos que ejecutan las principales funciones tales como el editor del esquema de la aplicación (tanto en su visión gráfica o jerárquica), al navegador, al editor de consultas como asi también al editor de instancias. Cada uno de éstos fueron desarrollados en forma independiente, realizando prácticamente toda su comunicación a través de los archivos de datos.

Para la interacción con el usuario se utilizaron interfaces orientadas a menúes, tanto horizontales como pull-down y pop-up; gráficas e icónicas. La entrada de datos se realizó principalmente a través de ventanas de selección simple o múltiple, ventanas de lista, de datos y cajas de diálogo. Los componentes más usados para implementar dichas interfaces fueron los forms, botones, labels, edits, listbox, imagenes, etc.

A continuación se describen los aspectos más relevantes de la implementación de la herramienta, algunos de los inconvenientes surgidos, y la solución que se les ha brindado.

5.2. Estructura de los archivos:

Representación de la jerarquía de clases:

Como ya se ha descripto, la herramienta se basa en un modelo orientado a objetos en la que se definen tres tipos de jerarquías: la de las clases Nodo (entidades), la de las clases Link (relaciones) y la de ejemplares.

Al no disponer de una base de datos orientada a objetos surgió la necesidad de

implementar las jerarquías nombradas, y la herencia existente entre las clases, mediante archivos relacionales.

Se definieron tres archivos principales, uno para las entidades, uno para las relaciones y otro para los ejemplares. Cada uno de ellos además de llevar la información propia que es característica de los tres tipos de clases debe contener, para mantener las jerarquías, cierta información de control. Para ello lo que se hizo es definir un campo en la estructura de los archivos que relaciona a cada una de las clases que se define con su antecesor inmediato (el padre) logrando así tener toda la cadena completa:

En detalle los archivos nombrados tienen la siguiente estructura:

Archivo de Entidades:

CodigoEntidad

NombreEntidad

CodigoEntidadPadre

Con estos tres campos se tiene la información necesaria como para identificar unívocamente una entidad, y conocer toda la estructura jerárquica de las Clases nodo. Esta última se puede armar rápidamente ya que dado el Código de una entidad se determina inmediatamente cual es la Entidad que es su superclase a través del *CodigoEntidadPadre*. Pero estos campos no son los únicos que el archivo tiene definido pues existen algunos que llevan la información necesaria para que la entidad se represente gráficamente en el diagrama del esquema de la aplicación (*CoordenadaX, CoordenadaY, Ancho, Alto*), y otros que almacenan diversa información de control (*UltimoNro.AtributoDefinido, CantidadDeAtributos, UltimoEjemplarDefinido, UltimoNroInstanciaDefinido*).

La única entidad que está siempre definida es la clase abstracta 'Nodo' la cual es la raíz del árbol jerárquico de entidades. O sea que toda nueva entidad, que no se agregue como subclase de otra ya agregada, con su Código de Entidad Padre la referenciará.

Archivo de Relaciones:

CodigoRelacion

NombreRelacion

CodigoRelacionPadre

Cardinalidad

EntidadDelFrom

EntidadesDelTo

TipoRelacion

Con esta estructura se identifica unívocamente a las relaciones y se mantiene la estructura jerárquica de las clases Link en forma análoga a las entidades. La diferencia esta en los atributos que son sólo característicos de las relaciones: uno que indica la cardinalidad, *TipoRelacion* que indica si es una relación conceptual o la especial 'Compuesta_Por', *EntidadDelFrom* que tiene el código de la Entidad que es origen de la relación y *EntidadesDelTo* que lleva los códigos de las entidades a las cuales llega la relación (recordemos que puede ser multidominio). Estos dos campos últimos, en el caso que la relación sea una subclase de una ya agregada al esquema, tendrán algún valor definido sólo si existe una redefinición con respecto a los valores que hereda. Es decir que *EntidadesDelFrom* aparecerá en blanco si se mantiene el mismo origen que su superclase y *EntidadesDelTo* solo tendrá los códigos de las entidades nuevas (que serán subclase de alguna de las que se haya definido como destino en su superclase), considerándose las no redefinidas como heredadas. Los códigos de las entidades que se heredan se pueden obtener fácilmente ya que como vimos, si se tiene la información de una relación sus ancestros son rápidamente accesibles.
También como en el archivo de entidades existen atributos que sirve para graficar a la relación en el diagrama del esquema (CoordenadaOrigenX, CoordenadaOrigenY, CoordenadaDestinoX, CoordenadaDestinoY), y otros que llevan información de control (UltimoNroAtributo, CantidadAtributos, CantidadInstancias) .

La única relación que está siempre definida es la clase abstracta 'Link' la cual es la raíz del árbol jerárquico de relaciones y tanto en su origen como en su destino tiene definida la entidad abstracta Nodo.

Archivo de Ejemplares:

CodigoEntidadDelEjemplar

CodigoEjemplar

CodigoEntidadPadre

CodigoEjemplarPadre

NombreEjemplar

Para identificar unívocamente al ejemplar, a diferencia que en los dos archivos anteriores, se utilizan dos atributos. Uno que indica la entidad para la cual se definió el ejemplar y otro para distinguirlo de los otros ejemplares de la misma entidad. Para mantener la estructura jerárquica además del ejemplar que es padre se debe almacenar a que entidad pertenece este último ya que puede ser de una entidad que sea superclase de la entidad del ejemplar en cuestión.

Existen otros atributos (*UltimaPosicionGráfico, Maximizado, CoordenadaX, CoordenadaY, Alto, Ancho*) que son necesarios para llevar información sobre la representación gráfica del ejemplar.

El ejemplar abstracto, denominado de la misma manera 'Ejemplar', que es la raíz del árbol jerárquico de los ejemplares, es el único que está siempre definido en este archivo y se considera como perteneciente a la clase 'Nodo'.

Hasta ahora hemos visto aquellos archivos que mantienen la información característica de los tres tipos de clases que se pueden definir, entidades o nodos, relaciones o links y ejemplares. Hay otro aspecto muy importante que es la representación de la estructura de almacenamiento para los atributos de las clases nombradas y de los anchors (sólo en el caso de los ejemplares). Hay que tener en cuenta que entre las clases existe herencia, de modo tal que si se quieren saber los atributos (o anchors) de una clase determinada debe existir una estructura y una

metodología que me permita recuperar todos los heredados.

Se utiliza un único **Archivo de Atributos**, en el que se almacenan tanto los atributos de las entidades como los de las relaciones, cuya composición es la siguiente:

CodigoClase

Nro.Atributo

TipoDominio

TipoAtributo

NombreAtributo

Tamaño

Los valores de los códigos utilizados para identificar a las clases nodos son distintos a los valores de los códigos de las clases links, por lo que sólo con el *CodigoClase* (que coincidirá con el código de una entidad o de una relación) más un número propio para el atributo es suficiente para identificarlos unívocamente.

En caso de ser un atributo con múltiples dominios, **todos** los dominios se indican en el campo *TipoAtributo* separados por una coma por lo que para un mismo atributo habrá una sola ocurrencia en el archivo.

La estructura por sí sola no refleja la herencia de atributos entre las clases. Para obtener los atributos heredados de una clase se utilizan las jerarquías representadas tanto en el archivo de entidades como en el de relaciones. Es decir, a partir del Código de entidad (o de relación) se obtienen todas las clases que son antecesoras de la clase en cuestión, siendo el conjunto de atributos de esta clase todos los de sus antecesoras más los propios.

Existe un **Archivo de atributos de un ejemplar** cuya estructura es muy simple:

CodigoEntidadDelEjemplar

CodigoEjemplar

CodigoEntidadDelAtributo.

NroAtributo

TipoAtributo

En este archivo se mantienen todos los atributos que fueron seleccionados para cada ejemplar. Es una referencia al Archivo de Atributos visto anteriormente. Se indica para cada ejemplar sólo los atributos propios. Para obtener los heredados se procede de la misma manera que con las relaciones y las entidades, mediante la jerarquía de ejemplares. Se pueden observar dos códigos que identifican entidades porque algunos de los atributos de un ejemplar pueden ser de clases antecesoras a la entidad a la que pertenece (los que quedaron definidos por la herencia). El Tipo del atributo es necesario ya que si es de dominio múltiple se puede elegir sólo alguno de ellos.

El **Archivo de Anchors** de un ejemplar es similar a este. Su estructura, como se ve a continuación, es suficiente para llevar la información sobre las relaciones seleccionadas que constituyen los anchors de cada ejemplar.

CodigoEntidadEjemplar

CodigoEjemplar

CodigoRelacion

CodigoEntidadDestino (a la que llega la relación elegida)

CodigoEntidadEjemplarDestino

CodigoEjemplarDestino

Simplemente se indican los anchors propios que se seleccionan por ejemplar ya que los heredados se obtienen en forma análoga a lo ya explicado. Además del código de relación de la cual surge el anchor se tiene el código de la entidad a la que conduce. Este es necesario para unificar a la relación porque el atributo To de la relación puede ser de múltiple dominio. Para identificar el ejemplar con el que se representa el nodo destino también es necesario almacenar el código de entidad al que pertenece ya que esta no tiene porque ser la misma que el destino de la relación, pues puede ser una antecesora de ella.

Con los archivos que hemos analizado hasta el momento la herramienta está en

condiciones de mantener toda la información referida a las clases del esquema (sus propiedades, estructura jerárquica y herencia). Ahora nos referiremos a otro cariz muy importante que fue tenido en cuenta en el momento de diseñar las estructuras de los archivos.

Representación de las instancias:

Cuando el usuario instancia una clase, sea una entidad o una relación, todos los valores que le asigna al objeto deben ser guardados de modo tal que cuando se recuperen sea identificable a que clase pertenece. Tomemos el caso de los atributos de una entidad. A priori, no se sabe cuales van a ser las clases nodo que se van a definir en el esquema, y menos aún los atributos que ellas pueden tener. Como todo esto el usuario lo determina con el uso de la herramienta, es decir dinámicamente, fue imposible definir previamente una estructura de archivos adecuada que permita almacenar los objetos de todas las entidades que se definan. Lo que se hace es crear un archivo a medida que una nueva clase se agrega al esquema. La estructura de dicho archivo además de permitir distinguir a cada una de las instancias que se definan de esa clase, debe guardar un lugar para cada atributo que se pueda instanciar. Por lo tanto una vez que se sabe cuales son los atributos que componen una entidad se crea el archivo cuya estructura consta de un *CodigoDeInstancia* y *NombreDeLaInstancia*, que lógicamente permiten identificar las instancias, y de un campo para cada atributo definido. La cuestión era como llamar a cada campo de modo tal que sea unívoco dentro del mismo archivo y fácilmente determinable. Con el nombre con el que se define al atributo no resultaba adecuado ya que podía ser muy extenso, había que transformarlo para que sea aceptable por el manejador de archivos(eliminar acentos, eñes y otros cambios) y sobre todo no bastaba para identificar el atributo porque se necesita saber también la entidad para la cual se definió y de que tipo era, pues podía ser heredado o bien tener múltiple dominio. En definitiva se decidió utilizar un caracter para identificar el dominio, cuatro para indicar la clase a la que pertenece y cuatro más para reconocerlo dentro de su clase,

quedando así un nombre de 9 caracteres cuyo formato se ve de la siguiente manera:

D E E E E N N N N. La 'D' denota la identificación del dominio, la 'E' indica el código de la entidad y la 'N' el número de atributo.

La denominación dada a cada archivo es un nombre de ocho caracteres que resulta de la concatenación del término 'ENTI' más el *código de la entidad* definida.

Por ejemplo si se agrega al esquema la clase Estudiante (a la que se le asigna el código '0006') con los atributos *Nro.Alumno(Alfanumérico)* y *edad(entero)*, y a su vez hereda *Nombre(Alfanumérico)* de la clase persona(Código '0002'), se crea un nuevo archivo para guardar las instancias que se definan de Empleado, denominado **ENTI0006**, con la siguiente estructura:

CodigoDeInstancia

NombreDeInstancia

A00060001 (Atributo *Nro.Alumno* con nro. 0001)

E00060002 (Atributo *edad* con nro. 0002)

A00020001 (Atributo heredado *Nombre* con nro. 0001)

Si la entidad tiene un atributo que es multivaluado, también se agrega un campo que lo representa en el archivo que se crea, aunque todos los valores se almacenan en un archivo que si fue creado previamente con este objetivo. Existe un archivo por cada tipo permitido para un multivaluado(alfanuméricos, enteros, fechas y reales) que almacena instancias de listas o conjuntos de todas las clases.

En cuanto al formato de los valores que se almacenan, podemos decir que los mismos se mantienen tal cual se definen, sin ninguna transformación. La diferencia está en los atributos del tipo texto, gráfico, sonido y animación ya que lo que se hace es guardar el camino físico del archivo que es elegido como instancia de estos.

Con referencia a las relaciones, la instanciación de sus atributos se almacenan en

estructuras idénticas a las descriptas, es decir se crea un archivo por cada clase link que se agrega al esquema. La diferencia con respecto a las entidades es que también se instancia la relación propiamente dicha, la cual es determinada por la instancia de la entidad origen y la de destino (puede ser más de una si es de cardinalidad N). Para esto se utiliza un archivo, cuya estructura si pudo ser definida en el diseño de la herramienta, el cual guarda las instancias de todas las relaciones. El formato es el siguiente:

CodigoRelacion

CodigoEntidadOrigen

CodigoInstanciaOrigen

CodigoEntidadDestino

CodigoInstanciaDestino

CodigoInstanciaRelacion

Es importante notar que para la misma relación puede haber más de una instancia no sólo por ser de cardinalidad N, sino porque el To puede ser de multidominio. Este sería el caso en el que se tendría más de un *CodigoEntidadDestino* distinto para el mismo *CodigoRelacion*

5.3. Implementación de Consultas

La implementación del Editor de Consultas sobre la información de la aplicación con el que dispone la herramienta se ha realizado teniendo en cuenta dos aspectos. Uno se refiere a la edición de las consultas en sí y el otro a la resolución de las mismas.

Con respecto a la edición se buscó que la formulación de una consulta se realice de modo tal que el costo por garantizar su validez sintáctica sea mínimo. A su vez se trató de evitar que el usuario necesite tener presente con exactitud la sintaxis del lenguaje de consultas.

La solución adoptada fue utilizar una interface en la cual la confección total de la consulta, ya sea la elección de las cláusulas, operadores, Clases (Entidades o Relaciones) se efectúe sin la necesidad de su tipeo. Para ello se destinaron distintas estructuras de selección a través de las cuales el usuario determina la composición de las cláusulas **Select**, **From** y **Where**.

En cuanto a la resolución de las consultas nos hemos visto en la necesidad de representarlas en el Modelo Relacional, mediante tablas y SQL. Para la transformación de sentencias del lenguaje de consultas definidas en el modelo a sentencias SQL relacionales (interpretables por el lenguaje de desarrollo, Delphi 3.0) se diseño un módulo a tal efecto.

Módulo de Transformación de Consultas

Tras la realización de cada Consulta la herramienta genera un archivo de texto, el cual contiene las sentencias sintácticamente validas en el lenguaje de consultas definido en el modelo (garantizado por el Editor de Consultas), y teniendo como salida del mismo otro archivo de texto, ahora con sentencias validas en el SQL relacional.

En el proceso de transformación intervienen todas aquellas tablas que almacenan las instancias de las clases referenciadas en el archivo de entrada, como así también las de sus subclases (característica inherente a la orientación a objetos). Es decir las tablas que contienen los atributos instanciados (para entidades y relaciones) y la que contiene las instancias de las relaciones. Además se hace uso de tablas auxiliares generadas dinámicamente.

Las tablas de las Entidades, siguiendo la nomenclatura propuesta en ***Estructuras de Archivos*** son las de la forma '***Entixxxx***', donde xxxx denota un número interno dado por la herramienta para identificar a la entidad. Asociadas a estas tablas se generan dinámicamente, tablas de la forma: '***_ntixxxx***' las cuales contienen todas las instancias de las subclases propias de la entidad.

Por otro lado para referirnos a las instancias de relaciones (involucradas

indirectamente en la consulta), además de utilizar la tabla que posee todas las instancias, a la cual a partir de ahora llamaremos **'InstRela'**, se crean tablas auxiliares asociadas a está de la forma **'_InstRel1'**, **'_InstRel2'**, etc. (se genera una por cada cláusula **Related_By**), o de la forma **'_InstRP1'**, **'_InstRP2',** etc. (una por cada cláusula **Path).**

Por ultimo tras la ejecución de la consulta se genera una nueva tabla donde se almacenan los resultados de esta cuya estructura es de la forma:

Código de Entidad (CodigoEntidad)
Código de Instancia (CodigoDeInstancia)
Nombre de la Instancia (NombreInstancia)
Indicación de Visitado (Visitado. Para controles posteriores)

La tarea fundamental de este módulo radica en la transformación de cada sentencia de su entrada (pertenecientes al lenguaje de consultas del modelo) en una equivalente para su salida (con la sintaxis SQL ejecutable en el lenguaje de desarrollo), que permite obtener el resultado buscado al formular la consulta.

En el **Apéndice A** se enuncian una serie de casos que ejemplifican la transformación.

Inconvenientes presentados en la Implementación del Módulo de Transformación

Uno de los principales inconvenientes que se nos presentó al momento de implementar este módulo radicó en el tiempo que insumía el proceso de transformación de las sentencias de la consulta del modelo en sentencias SQL relacionales, y la obtención de aquellas instancias que verificaban la consulta propiamente dicha. El problema en sí residía en la gran cantidad de tablas que intervenían en la consulta agravado por la cantidad de instancias de estas. Por ejemplo, si se tiene definido conceptualmente la Clase Persona, con las subclases Estudiante y Profesor y la Clase País con las subclases País Americano y País

Europeo y la Relación Habita_en definida desde Persona a País (ejemplo análogo al del Caso 6 del **Apéndice A**), deberían participar las tablas con instancias de Persona, Estudiante, Profesor; País, País Americano y País Europeo, más la tabla de instancias de relaciones entre clases. La solución adoptada fue disminuir la cantidad de tablas que intervenían en la consulta, y esto se logró con la generación de una sola tabla que engloba a todas las instancias de sus subclases, aplicándose este concepto tanto para las entidades como para las relaciones.

En el ejemplo anterior entonces aparecerán tres tablas, una de ellas contiene instancias de la entidad Persona y sus subclases, otra tabla con instancias de la entidad País y sus subclases y por último la tabla que información sobre las instancias de la relación Habita_en y eventualmente sus subrelaciones. De esta manera se logra una notable reducción de la cantidad de tablas (en este simple ejemplo quedan solamente tres).

Otro de los inconvenientes a solucionar consistió en la gran cantidad de instancias que participan en la consulta. Esto obviamente no se puede acotar porque no se pueden dejar de considerar todas las instancias de las entidades. Analizando el conjunto de tablas que intervienen en la consulta encontramos que una tabla de ellas, la de instancias de relaciones esta sobredimensionada ya que contiene información de todas las instancias de relaciones del modelo, y en la consulta probablemente aparezcan pocas relaciones, o generalmente una. Basándonos en este criterio, decidimos definir una de las tablas internas (llamada _nstRel), la cual para cada cláusula de la consulta sólo contiene información sobre la/s relaciones de esta.

Como hemos explicado anteriormente, estas tablas intermedias se generan en el módulo de transformación, surgiendo así un concepto ampliado del módulo, el cual no solo realiza tareas meramente sintácticas sino también del 'tipo semántico' ya que en algunos casos analiza el contenido y significado de la consulta.

Estos inconvenientes, los de la cantidad de tablas e instancias, fueron resueltos

con el objetivo de bajar los tiempos de respuesta, y se solucionaron partiendo de la base de que los SQL relaciones (al menos los interpretados en el lenguaje de desarrollo), primero realizan el producto cartesiano de las tablas definidas en la cláusula From y luego aplican los predicados enunciados en el Where. La mayor cantidad de instancias incrementaba el tamaño de las tablas que junto con la gran cantidad de tablas intervinientes acrecentaba el tiempo de respuesta considerablemente.

5.4. Implementación del Navegador

Dada la característica que poseen las hipermedias de permitirle al usuario recorrerlas con total libertad, existe una gran variedad de nodos que pueden ser accedidos, los cuales difícilmente tengan características comunes entre sí. Por ende, resulta prácticamente imposible utilizar una estructura esquemática de visualización que sea adaptable a todos los nodo. Esto significa que se necesitaría al menos una estructura propia para cada uno de ellos. El navegador ha sido implementado mediante el uso de un formulario al que solamente se le ha definido estáticamente la etiqueta que contendrá la identificación del nodo activo y las estructuras que le permitirán a los usuarios recorrer la historia de la hipermedia. Todo el resto de los componentes que serán visualizados en un nodo se crean dinámicamente. Cuando el usuario ejecuta una acción para acceder a un nuevo nodo, el proceso identifica las características que este posee. Determina el ejemplar con el que se visualiza, para poder conocer cuales son los atributos y anchors que deberán mostrar y de esta forma, poder crear dinámicamente todas las componentes que los representan. A estas componentes se las representa con las características determinadas por el usuario en la etapa de la definición de la representación visual de ellas, como ser el color, alto, ancho, transparencia, modo de cursor, tipo y tamaño de letra (si tiene), etc. Además algunas de ellas tienen comportamientos diferentes que el resto, como por ejemplo los anchors, a los que

se les asigna el evento 'On Click' del mouse, el cual permite a los usuarios acceder al nodo que describe su etiqueta. Así, cuando uno de estos anchors es disparado, el navegador procede a la destrucción de todas las componentes representativas de los atributos del nodo que se abandona, para crear posteriormente las correspondientes del nodo al que se accede.

A su vez, además del ejemplar, se debe determinar la instancia a ser visualizada en el nodo al que se navega. Para esto se tiene en cuenta la instancia del nodo en la que se activó el anchor, y a partir de conocer las instancias de la relación de la que surgió dicho anchor, podemos saber cual es la instancia del nuevo nodo.

Un tratamiento especial fue dado a la hora de implementar los nodos **no maximizados**, es decir aquellos nodos que por lo general no son tan relevantes en la hipermedia y que se utilizan para brindar algún detalle adicional del nodo que se encuentra activo. O que si son relevantes pero simplemente el usuario no desea usar la totalidad del espacio de muestra reservado para un nodo. Este tipo de nodo fue implementado en otro formulario análogo al principal que se despliega delante de éste, pero con la característica principal de que sus anchors no tienen el efecto de navegabilidad. El usuario lo visualiza y luego lo debe cerrar para volver al nodo que lo llamó, y de esta manera poder seguir navegando.

Algunas de las componentes de representación utilizadas pertenecen al lenguaje con el que se implementó la herramienta (campos de edición para atributos alfanuméricos, campos memos para los textos, bits de mapas para gráficos, etc.). Otras, fueron diseñadas específicamente con el objetivo de brindar estructura de soporte para la representación y visualización para otro tipo de atributos como en el caso de los multivaluados, ya sean listas o conjuntos. Si el tipo de atributo multivaluado (sea lista o conjunto), es alfanumérico, entero, real o fecha, la representación dada es una caja que contiene una lista con los valores correspondientes. En cambio si el tipo de atributo multivaluado es texto, gráfico, sonido o animación, la forma de representación varía dependiendo la misma si se

trata de una lista o de un conjunto. En el caso de tratarse de una lista, el navegador mostrará la primera ocurrencia de ella, y para acceder a las otras, deberá requerir la siguiente o la anterior mediante el uso de los indicadores de desplazamiento ordenado dentro del atributo. En el segundo caso, se presentará una ocurrencia cualquiera, y para acceder al resto, podrá elegir otra de ellas de una lista que contiene el nombre que las identifica.

Para implementar la **historia** de la navegación, se fue llevando una lista ordenada de las características identificatorias de los nodos visitados (entidad que representa, ejemplar elegido como perspectiva, e instancia visualizada) y cada vez que el usuario retrocede o avanza en la hipermedia, la información a ser visualizada es obtenida de ella. Cuando el usuario retrocede en la historia y toma un camino alternativo a través de un anchor, se descarta de la lista todos aquellos nodos incorporados a partir del nodo que se encuentra activo y se agrega a la lista la información del nodo a ser accedido.
Cabe aclarar que para navegar se utilizaron métodos comunmente usado en muchos navegadores como son los botones para avanzar, retroceder y para regresar al nodo inicial de la hipermedia.

5.5. Implementación del Instanciador

La implementación del instanciador de clases nodo, tiene características muy similares a las del navegador, ya que en función de la entidad a la que se quiere instanciar, se deben generar automáticamente las componentes que la componen, y en caso de cambiar de entidad, se deben destruir las componentes de la anterior y luego generar las correspondientes a los atributos de la nueva entidad a ser instanciada.
Lo que se hizo es una especie de navegación entre las instancias de la entidad. De

esta manera además de poder definir nuevas instancias, el usuario se encuentra con algunos botones similares a los utilizados en la navegación con los que puede visualizar las instancias ya definidas,

Para el ingreso de datos se utilizaron componentes predefinidas del lenguaje en la instanciación de algunos de los atributos, y para otros casos (como los atributos multivaluados) se crearon estructuras que facilitan su definición.

La definición de las instancias de una relación se hace desde este mismo módulo. La misma se realiza basándose en la instancia de la clase nodo definida en el origen (From). Es decir que cuando se está visualizando la instancia de una entidad, considerándosela como el origen, se puede acceder a instanciar las relaciones. Simplemente se deben elegir las instancias de las entidades que están definidas como destino (dominio del To) de las relaciones en las que la entidad correspondiente a la instancia origen es dominio del From.

6. EJEMPLO: GALERIA DE ARTE

En este punto desarrollaremos una hipermedia de la Galería de Arte, cuyas caractarísticas son similares a la utilizada previamente como ejemplo en la etapa de la descripción del modelo.

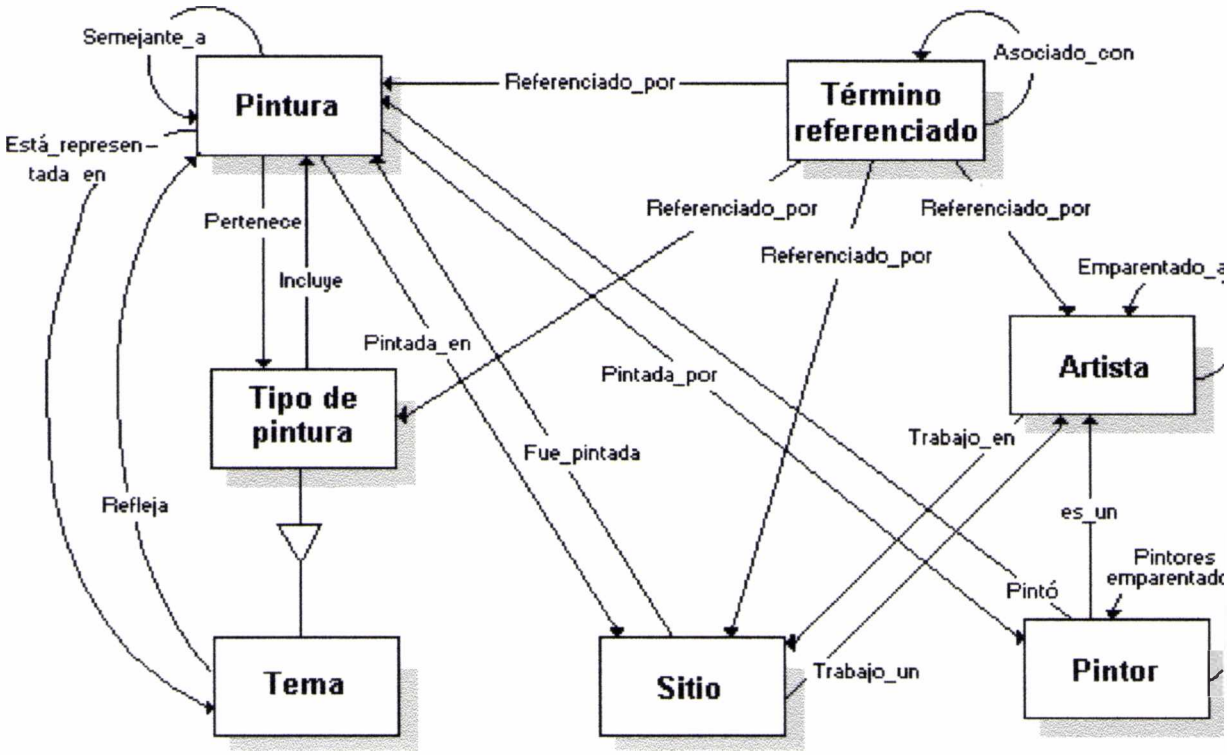
Si bien la herramienta no le exige al usuario realizar las diferentes etapas del diseño de la hipermedia en forma secuencial, para describir el ejemplo, hemos optado por detallar los pasos de una manera arbitraria siguiendo cierta lógica de continuidad a la hora de operar con ella.

Para comenzar, describiremos la manera en que se da de alta algunas de las clases que intervienen en esta aplicación, es decir la **definición de clases nodo y link**. A continuación agregaremos los **ejemplares de las clases nodo** que intervengan y le daremos una representación visual a cada uno de ellos. Luego mostraremos la manera de proceder para realizar las **instancias de los componentes**, ya sean las entidades como de las relaciones. Mostraremos algún **nodo** y el efecto producido al ser activado un anchor, tanto los de navegación hacia otro nodo maximizado, como los que se muestran en una ventana más pequeña. Para finalizar, veremos la forma de realizar los **'Queries'** sobre la aplicación, la respuesta del sistema a la ejecución del mismo y la posterior utilización de las estructuras de acceso a la hipermedia dada por dicha respuesta.

En el ***APENDICE B*** se da una breve explicación sobre el uso de la herramienta.

Esquema general de la Galería de Arte:

A continuación veremos el diagrama de la galería de Arte diseñado para ser modelizado en la herramienta.



Como se ve en el esquema de la Galería de Arte, hemos definido las siguiententes clases conceptuales 'Nodo': 'Pintura', 'Tipo de pintura', 'Tema', 'Sitio', 'Artista', 'Pintor' y 'Término referenciado'.

Además, han sido definidas las siguientes clases conceptuales 'Link':

Nom.Relación (Cardinalidad)	Atributo From	Atributo To
'Semejante_a' (1:N)	Pintura	Pintura
'Asociado_con' (1:N)	Término referenciado	Término referenciado
'Emparentado_a' (1:N)	Artista	Artista
'Pintores emparentados' (1:N)	Pintor	Pintor

'Está_representada_en' (1:N)	Pintura	Tema
'Refleja' (1:N)	Tema	Pintura
'Pertenece' (1:1)	Pintura	Tipo de pintura
'Incluye' (1:N)	Tipo de pintura	Pintura
'Pintada_en' (1:1)	Pintura	Sitio
'Fue_pintada' (1:N)	Sitio	Pintura
'Pintada_por' (1:1)	Pintura	Pintor
'Pintó' (1:N)	Pintor	Pintura
'Trabajó_en' (1:N)	Artista	Sitio
'Trabajó_un' (1:N)	Sitio	Artista
'Referenciado_por' (1:N)	Término referenciado	Pintura, Artista, Sitio, Tipo de Pintura
'Compuesta_por' (1:N)	Tipo de pintura	Tema
'Es_un'	Pintor	Artista

Definición de nodos y links:

Comenzaremos a definir en la aplicación las entidades que intervienen en la Galería de Arte, cada una de ellas con los atributos (y sus tipos) que las conforman.

'Pintura': Nombre (A[20]) ; Pintura (G) ; Descripción general (T) ; Descripción técnica (T) ; Fecha de creación (D)

'Tipo de pintura': Nombre (A[20]) ; Descripción (T)

'Tema': Nombre (A[20]) ; Descripción (T)

'Sitio': Nombre (A[20]) ; Mapa (G) ; Descripción (T) ; Año inicial del período

(E) ; Año final del período (E)

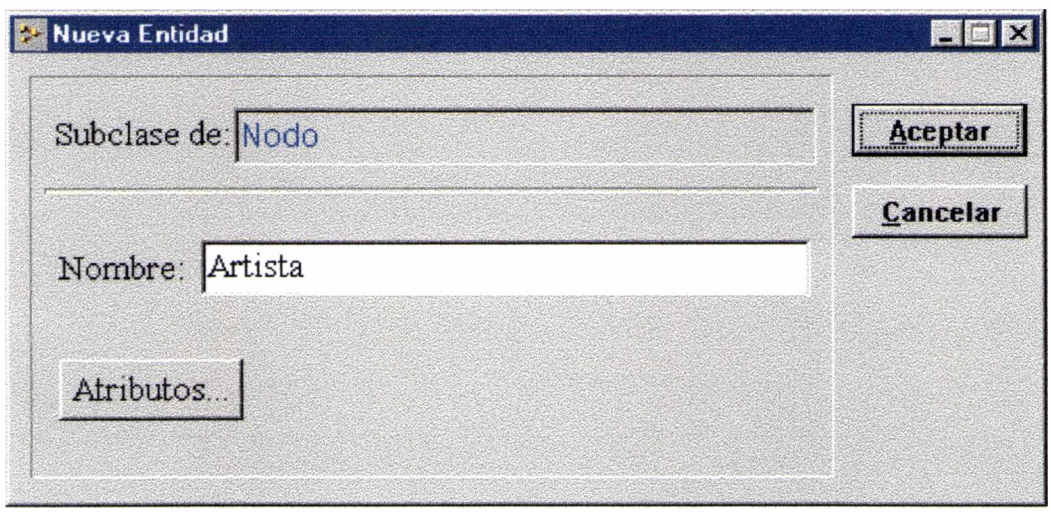
'Artista': Nombre (A[20] y S) ; Año de nacimiento (E) ; Año de defunción (E) ; Nacionalidad (A[20]) ; Retrato (G)

'Pintor': Biografía (T)

'Término referenciado': Nombre (A[20]) ; Descripción (T)

Comenzaremos explicando la manera de agregar entidades en el diagrama. Para ello tomaremos como ejemplo a la entidad 'Artista', de la que luego le crearemos a continuación una subclase de ella llamada 'Pintor'.

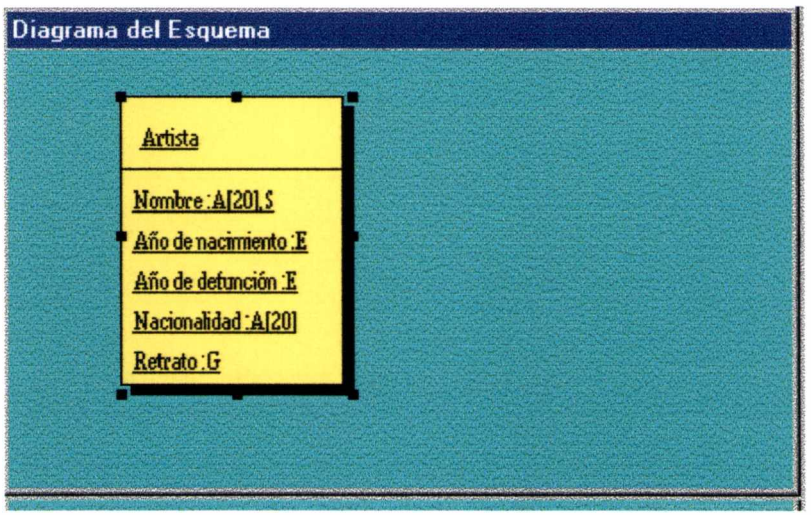
Mediante la selección del ícono de 'Agregar Entidad' la herramienta despliega una interface como detallamos a continuación, que le solicita al usuario el nombre de la nueva entidad, que en nuestro caso será 'Artista'.



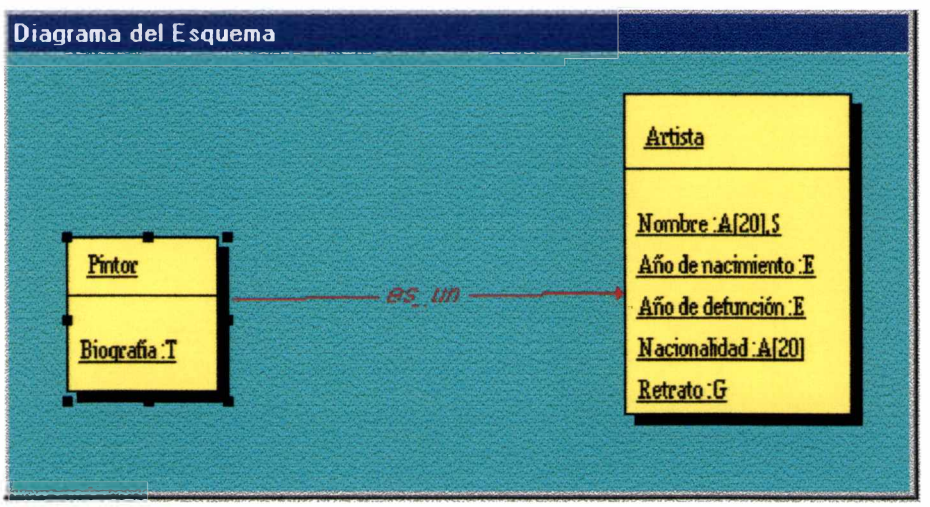
Luego mediante el botón de 'Atributos...' accedemos a una ventana que nos permite observar los atributos propios de la clase que estamos definiendo (o modificando). Por tratarse de la definición de una nueva entidad , veremos que su contenido está vacío. Si lo que se estuviéramos definiendo fuese una subclase de

una entidad ya creada, la herramienta permite visualizar los atributos que hereda la clase que estamos definiendo. En nuestro ejemplo, por tratarse de una clase que tiene como única antecesora la clase conceptual 'Nodo', encontraremos que no tenemos acceso a la visualización de los 'Atributos Heredados'. Mediante el botón de 'Agregar' accedemos a la interface de definición de atributos propios de la clase que estamos definiendo. En ella declaramos los atributos detallados previamente para la clase 'Artista'. A continuación vemos la interface en el momento de definir el primer atributo de dicha entidad llamado 'Nombre' que es de dominio 'Múltiple' y sus tipos son 'Alfanumérico de tamaño 20' y 'Sonido'.

Finalizada la definición de los atributos de la clase, el diagrama se modificará mostrando un interface como la siguiente:

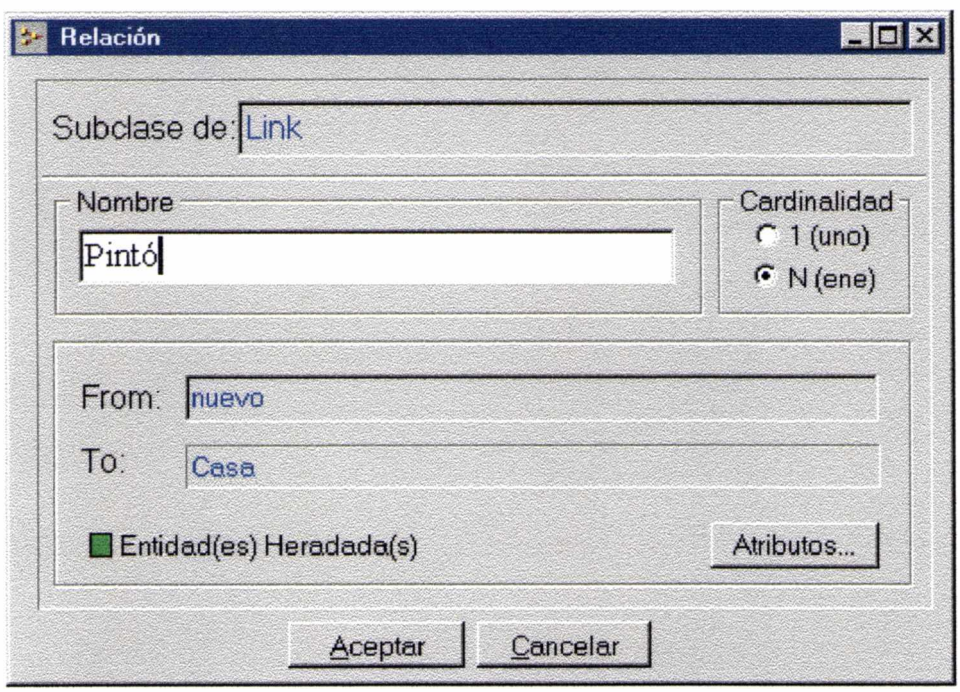


Ahora crearemos la clase 'Pintor', la cual será subclase de la clase 'Artista' recientemente definida. El procedimiento es análogo al descripto recientemente, con la salvedad de que ahora solo debemos definir aquellos atributos que son propios de la clase 'Pintor'. La interface 'Atributos' muestra los atributos heredados por la nueva clase que se está creando, y son los definidos previamente en la entidad 'Artista'. Una vez generada esta subclase, el diagrama agrega automáticamente la relación 'es_un' entre la clase 'Pintor' y 'Artista'.



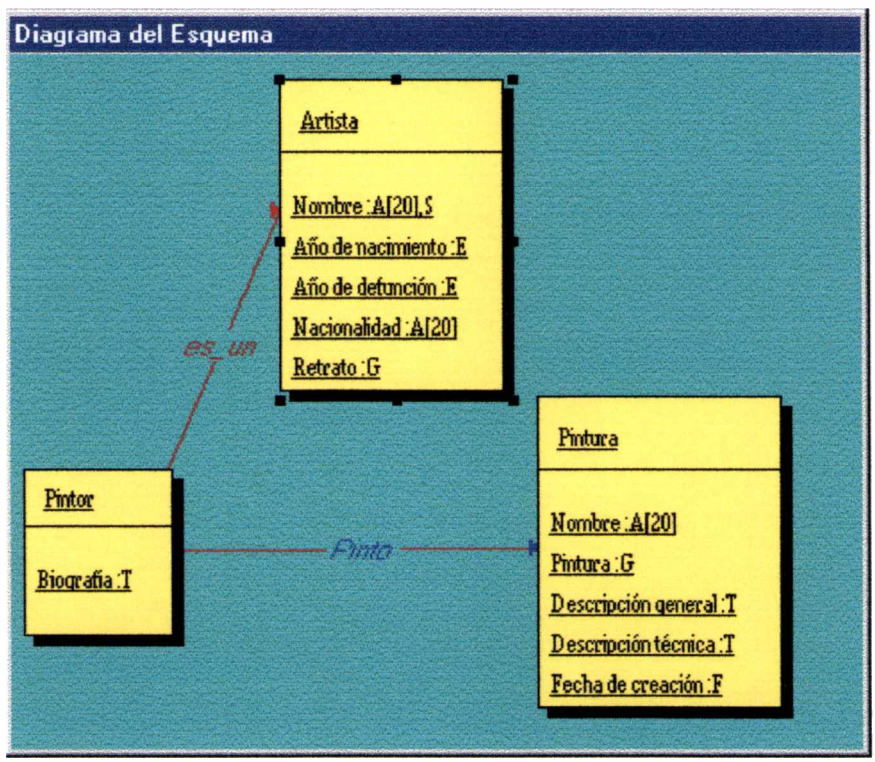
Ahora continuaremos por la definición de las relaciones entre las entidades.

Consideremos que además de las clases 'Artista' y 'Pintor' tengamos definida la clase 'Pintura'. Tomemos para el ejemplo la relación 'Pintó', cuyo atributo 'From' es la entidad 'Pintor' y la entidad del 'To' es la entidad 'Pintura'. Mediante el ícono de 'Agregar Relación', el sistema solicita al usuario que defina la entidad que será instancia del atributo 'From' y las entidades que serán instancias del atributo 'To' en la nueva relación. Luego de ligar la relación, le herramienta muestra una interface en la que detalla el nombre de la clase 'Link' antecesora (que en nuestro caso será la misma clase 'Link') y los nombre de las entidades elejidas como instancias 'From' y 'To' de la nueva relación. El usuario debe ingresar el nombre y la cardinalidad (1 o N) que poseerán las instancias de ella. La interface que se genera en este procedimiento será la siguiente:

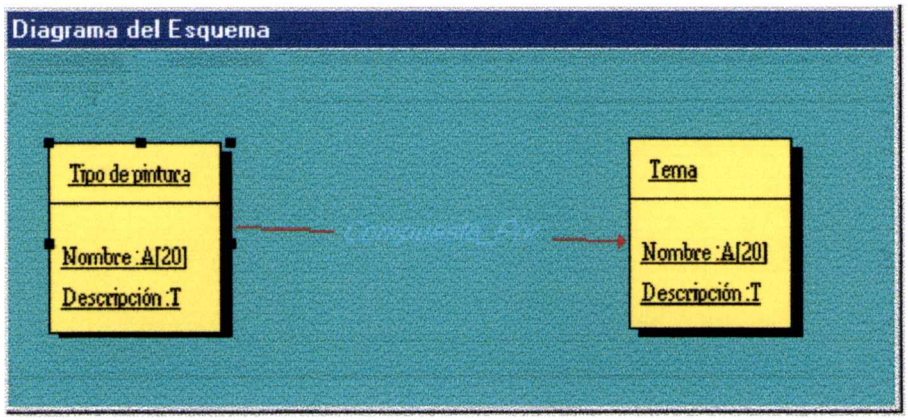


Al igual que para la definición de las entidades, activando el botón de 'Atributos' se puede acceder a la pantalla de detalle de los atributos heredados y a partir de esta a la de 'Agregado' de nuevos atributos para la relación. Ambas interfaces son análogas a las descriptas previamente para la definición de los atributos de

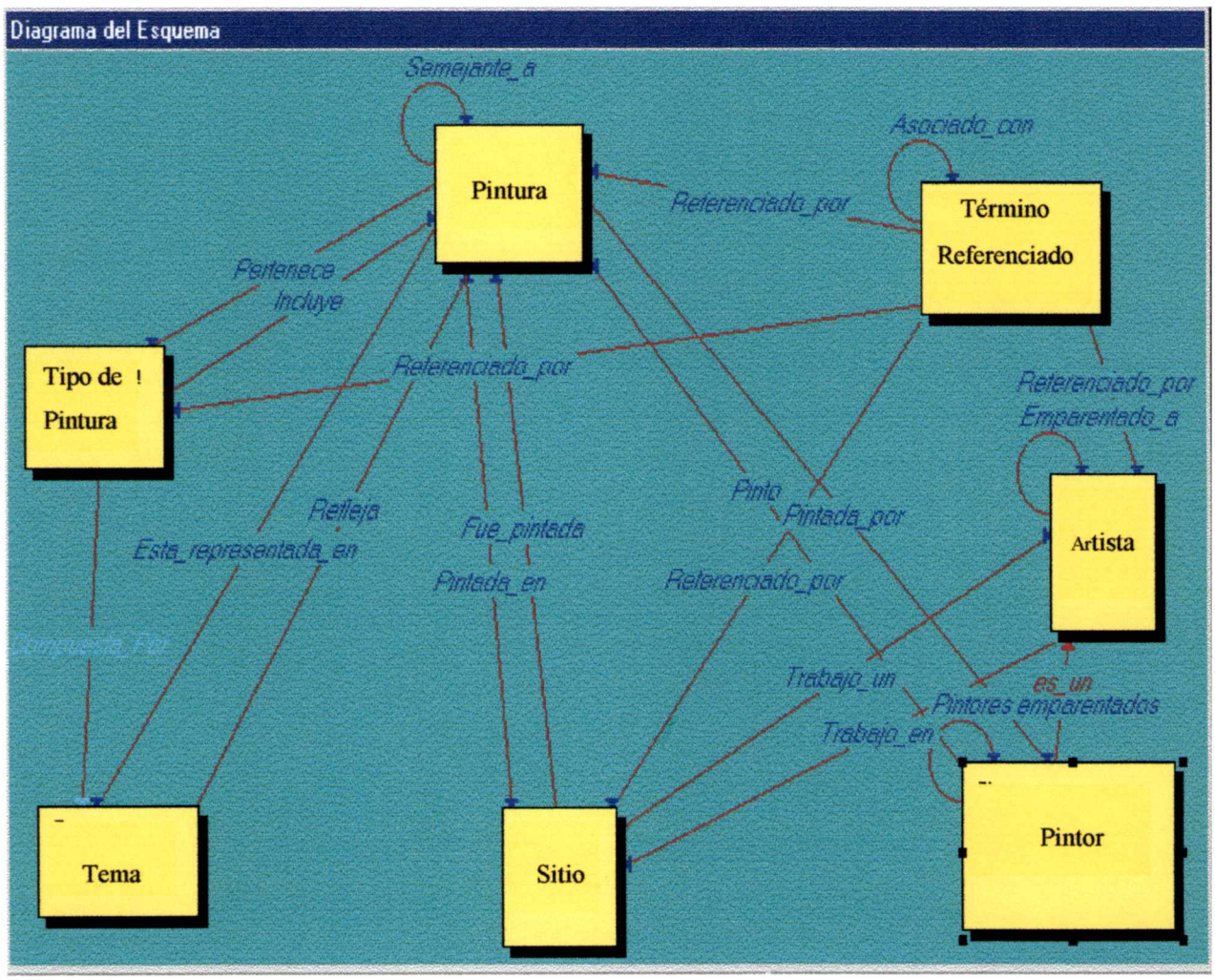
las entidades. En nuestro ejemplo, la relación que estamos definiendo no posee atributos, con lo cual no es necesario acceder a ellas. Al aceptar la definición de la relación, el esquema quedará definido de la siguiente manera:



El proceso de creación de la relación 'Compuesta_por' es similar a la descripta para las relaciones conceptuales 'Link', con la diferencia que la interface de definición de ella necesita solamente la especificación de la cardinalidad de la relación (1 o N), ya que su nombre está tácito ('Compuesta_por') y no permite la declaración de atributos propios. Supongamos que ya tenemos definidas en nuestro esquema las entidades 'Tipo de Pintura' y 'Tema'. Al crear la relación 'Compuesta_por' entre la primera con la segunda, este sector del esquema quedará de la siguiente manera:

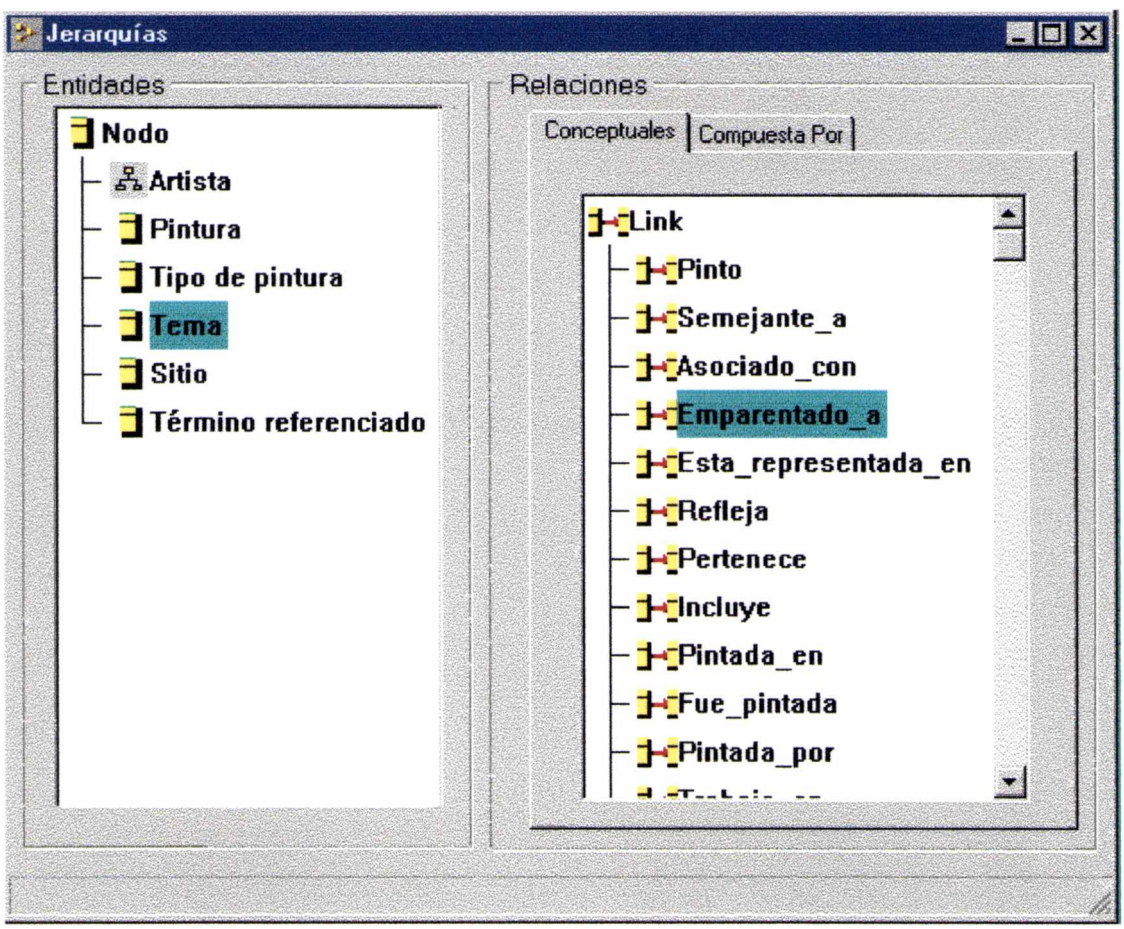


Una vez que tengamos definidas todas las entidades y relaciones del esquema, el diagrama de la herramienta quedará definido de la siguiente manera:

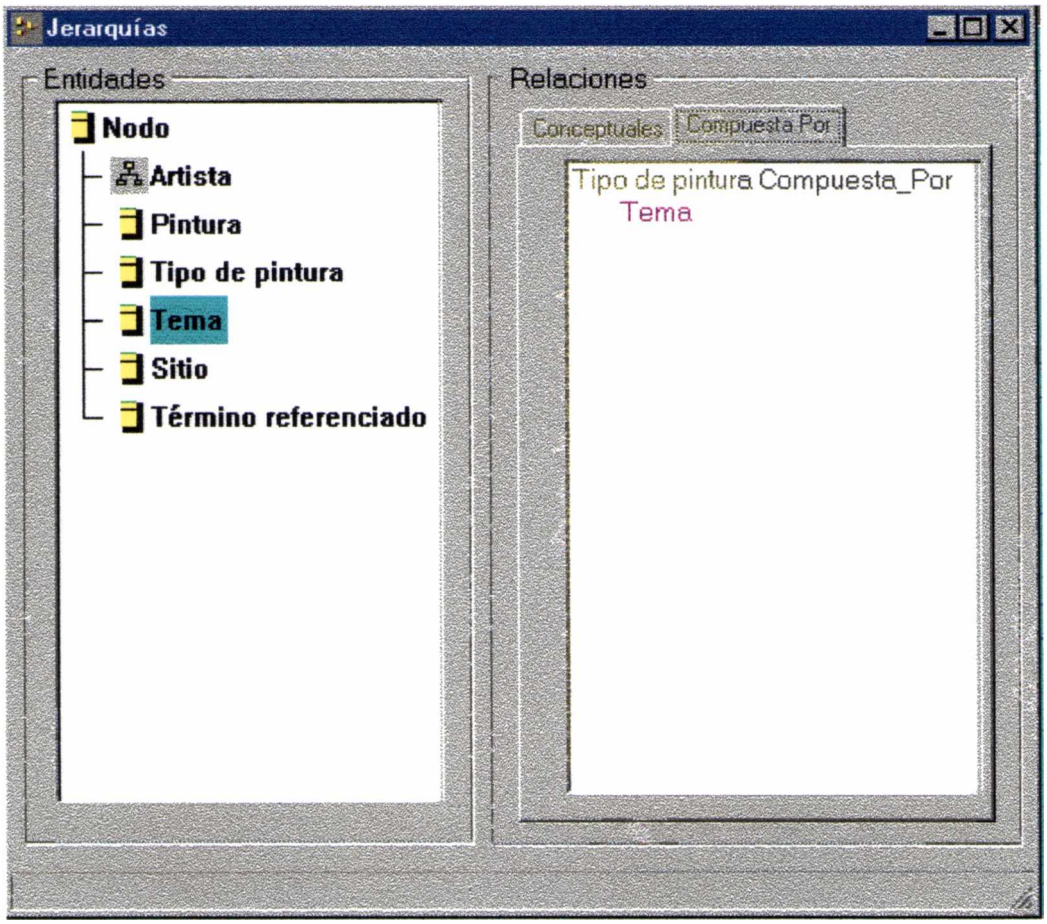


Estas definiciones de clases se ven también reflejadas en la interface de las 'Jerarquías', la cual presenta el árbol de jerarquía de las entidades y el árbol de jerarquía de las relaciones (las conceptuales y las compuestas_por). En nuestro ejemplo, el diagrama visto recientemente tendrá las representaciones en dichos árboles de la siguiente manera:

Jerarquías con las Relaciones conceptuales:



Jerarquías con las Relaciones 'Compuestas Por':



Ejemplares de las clases nodo y sus representaciones visuales:

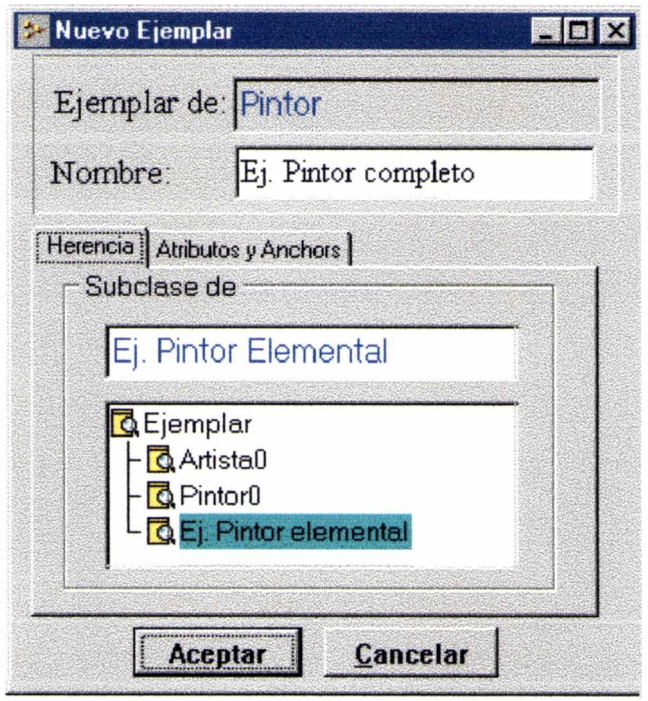
Una vez que tengamos definidas las entidades y relaciones del esquema de la aplicación, comenzamos a definir los ejemplares y sus representaciones visuales. Para ello, accedemos a la interface que nos permite llevar esto a cabo, habiendo previamente seleccionado una entidad, que en este caso será 'Pintor'. Cuando accedemos a la interface de 'Ejemplares de Pintor' seleccionamos la opción 'Nuevo' para definir un nuevo ejemplar de la clase. Nos encontramos así con una interface que nos pide el nombre del nuevo ejemplar para la clase en cuestión, al que llamaremos 'Ej. Pintor Elemental', el cual tendrá como único antecesor a la clase 'Ejemplar'. Le definimos entonces los atributos que contendrá, que serán: Nombre (A[20]) del atributo multidominio y Biografía (T). Este ejemplar que

cuenta solamente con dos atributos y ningún anchor, será utilizado por alguna instancia en la que obviamente no tendremos intenciones de navegar hacia otro nodo. Decidimos entonces definirle a los nodos representados con este ejemplar una representación visual que no se despliegue en forma maximizada cuando sea accedido en la navegación.

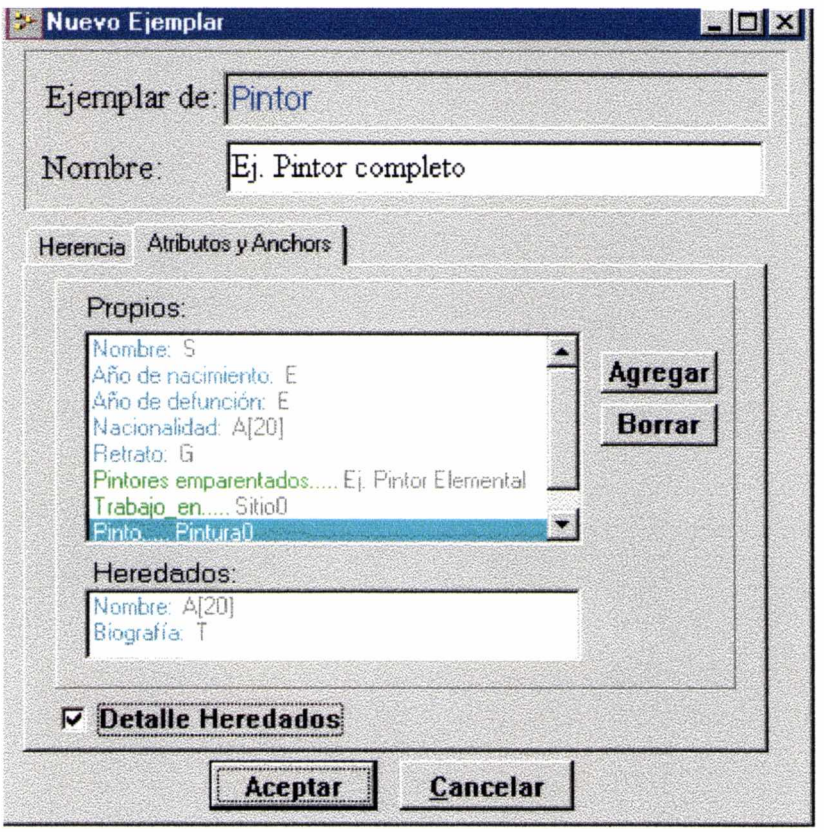
Crearemos a continuación otro ejemplar con mayor cantidad de elementos, el cual no solo tendrá los atributos definidos en el ejemplar anteriormente creado, sino que también tendrá los atributos Nombre (S), Año de nacimiento (E), Año de defunción (E), Nacionalidad (A[20]) y Retrato (G); además estarán los anchors 'Emparentado_a', 'Trabajó_en' y 'Pintó', con lo cual formaremos un ejemplar que tenga el total de atributos y anchors posibles. Como este ejemplar tiene los mismos elementos del ejemplar anterior más algunos propios, podemos generarlo como subclase de él, debiendo definirle solamente los atributos y anchors recientemente descriptos. A este nuevo ejemplar lo llamaremos 'Ej. Pintor Completo'. Como este ejemplar ha sido definido con componentes navegacionales (anchors), la intención es la de permitirle al nodo que haga uso de ellas, brindando la posibilidad de navegar hacia otros nodos. Por lo tanto, cuando definamos su representación visual, haremos que el nodo se vea en forma maximizada.

Tenemos ahora creados dos ejemplares para la entidad 'Pintor'. Si accedemos a la pantalla de los ejemplares definidos para dicha entidad y seleccionamos uno de ellos, como por ejemplo 'Ej. Pintor Completo', podemos investigar sus propiedades (nombre del ejemplar, atributos y anchors definidos, jerarquía dentro de la clase 'Ejemplar', etc.).

La interface de la herencia del ejemplar 'Ej. Pintor Completo' que describe su herencia será la siguiente:

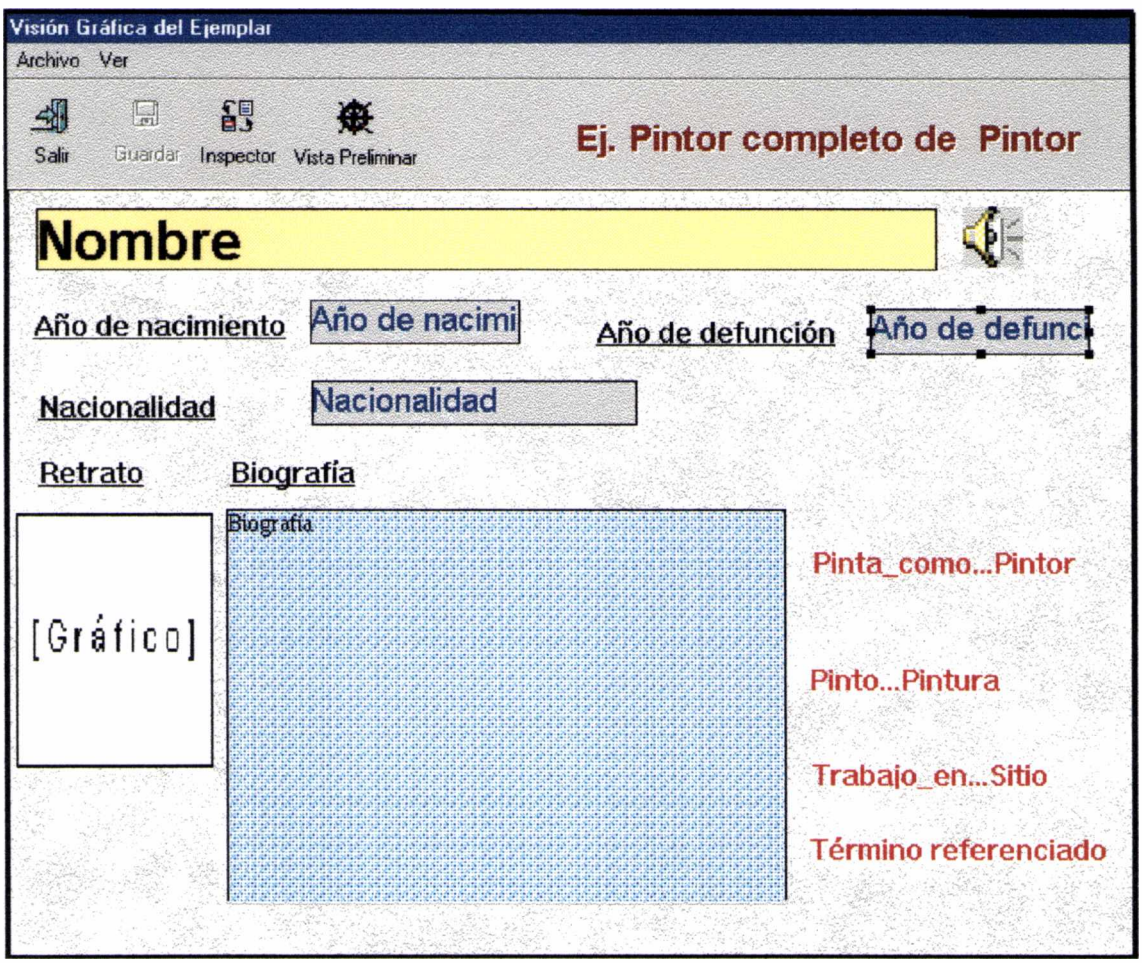


La interface de la herencia del ejemplar 'Ej. Pintor Completo' que describen sus anchors y atributos será la siguiente:

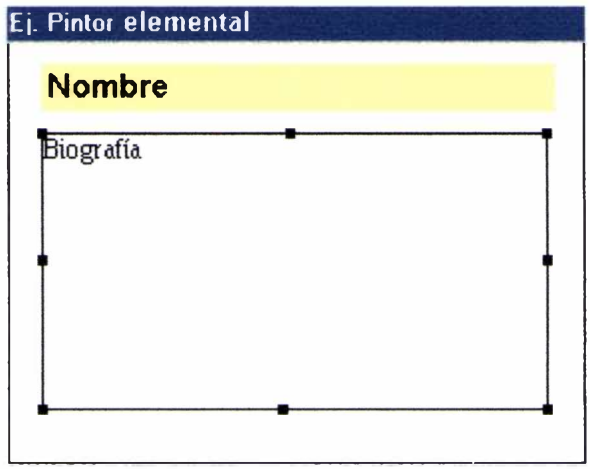


Como dijimos previamente, la representación visual de los ejemplares debe ser

definido por el usuario. Tomemos el ejemplar 'Ej. Pintor Completo', cuya visualización se hará mediante un nodo maximizado. El posicionamiento y las características de sus componentes las hemos diseñado en forma arbitraria de la siguiente manera:



A su vez, definimos la representación visual del ejemplar 'Ej. Pintor Elemental', cuya visualización se hará mediante un nodo no maximizado. El posicionamiento y las características de sus componentes será la siguiente manera:



Una vez que tengamos definidas todas las entidades, sus respectivos ejemplares, la representación visual de éstos y las relaciones intervinientes en nuestro ejemplo, estamos en condiciones de instanciar sus componentes.

Instancias de los componentes:

Comenzaremos por la instanciación de las entidades. Tomemos como ejemplo la entidad 'Pintor' y como instancia a ingresar al pintor 'Leonardo da Vinvi'. Seleccionamos en el diagrama del esquema dicha entidad y luego accedemos a la interface de 'Instanciador de Entidades'. Dado que la entidad ha sido creada recientemente, no contiene ninguna instancia, con lo cual nos aparecerán sus atributos vacíos. Cambiamos el 'Nombre de la Instancia' que aparece por defecto por 'Leonardo', que es una forma más representativa para identificar a dicho pintor del resto de los pintores que vamos a instanciar. A continuación ingresamos las instancias de cada atributo, como ser:

Nombre (A[20]): 'Leonardo da Vinci'

Nombre (S): Camino físico del archivo 'leonardo.wav'

Año de nacimiento (E): '1452'

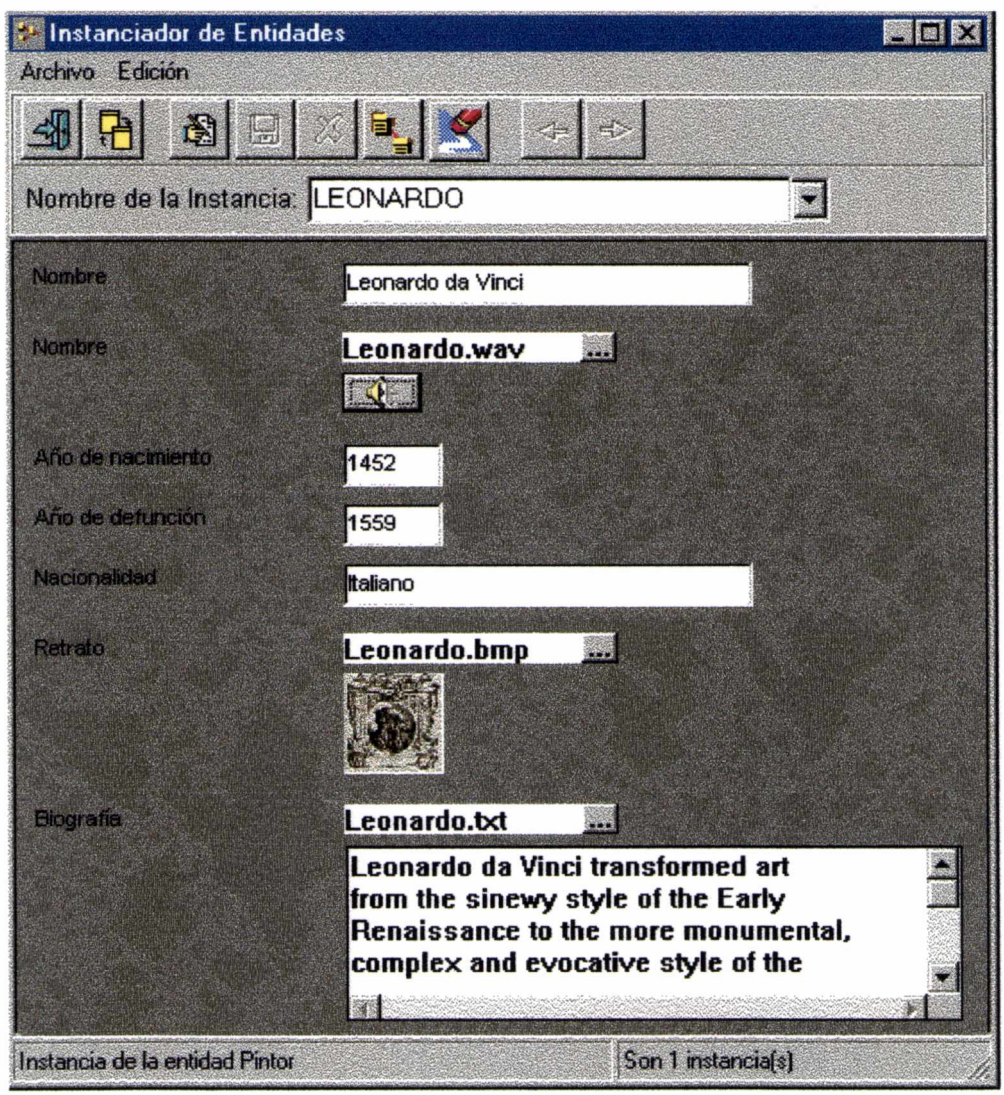
Año de defunción (E): '1519'

Nacionalidad (A[20]): 'Italia'

Retrato (G): Camino físico del archivo 'leonardo.bmp'

Biografía (T): Camino físico del archivo 'leonardo.txt'

Una vez instanciados todos los atributos del pintor 'Leonardo', la interface tendrá el siguiente aspecto:

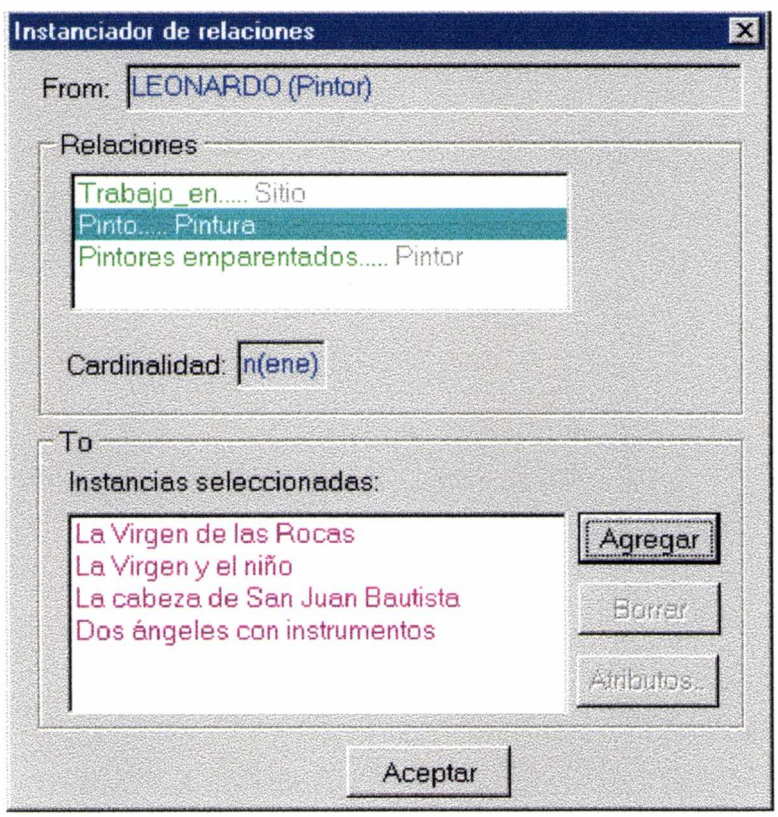


Ahora podemos agregar un nuevo pintor, modificar alguno ya existente, ingresar instancias de otra entidad o instanciar las relaciones que tengan definida en el atributo 'From' a la clase 'Pintor' o a alguna superclase de ella, que en nuestro caso ser'a 'Artista', y por intermedio de las cuales se podrá navegar desde 'Leonardo da Vinci' hacia otros nodos. Supongamos que ya están instanciadas

todas las entidades restantes del esquema que estamos ejemplifiicando. Optamos entonces por la instanciación de las relaciones, las que en este caso serán:

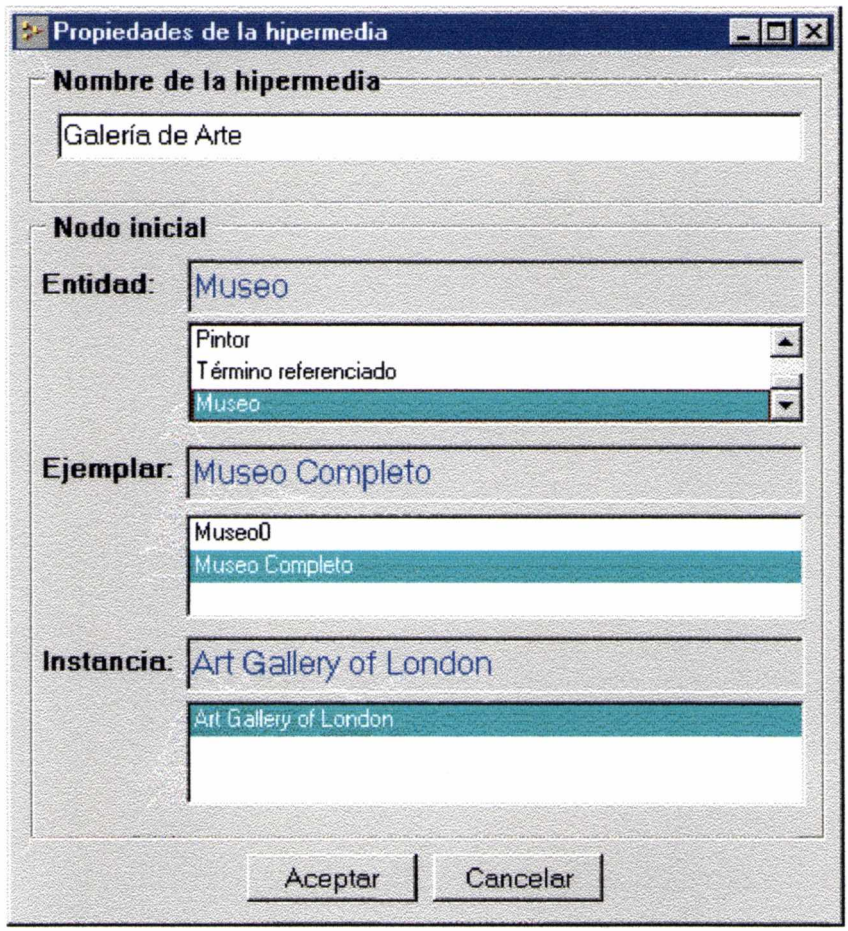
'Emparentado_a' (1:N)	Artista	Artista
'Pintó' (1:N)	Pintor	Pintura
'Trabajó_en' (1:N)	Artista	Sitio

Veamos la relación 'Pintó', cuyas instancias tendrán cardinalidad N. En la hipermedia queremos representar que el artista 'Leonardo' 'Pintó' las siguientes pinturas: 'La virgen y el niño', 'La virgen de las rocas', 'La cabeza de San Juan Bautista' y 'Dos ángeles con instrumentos'. Vemos a continuación la interface de estas instanciaciones.



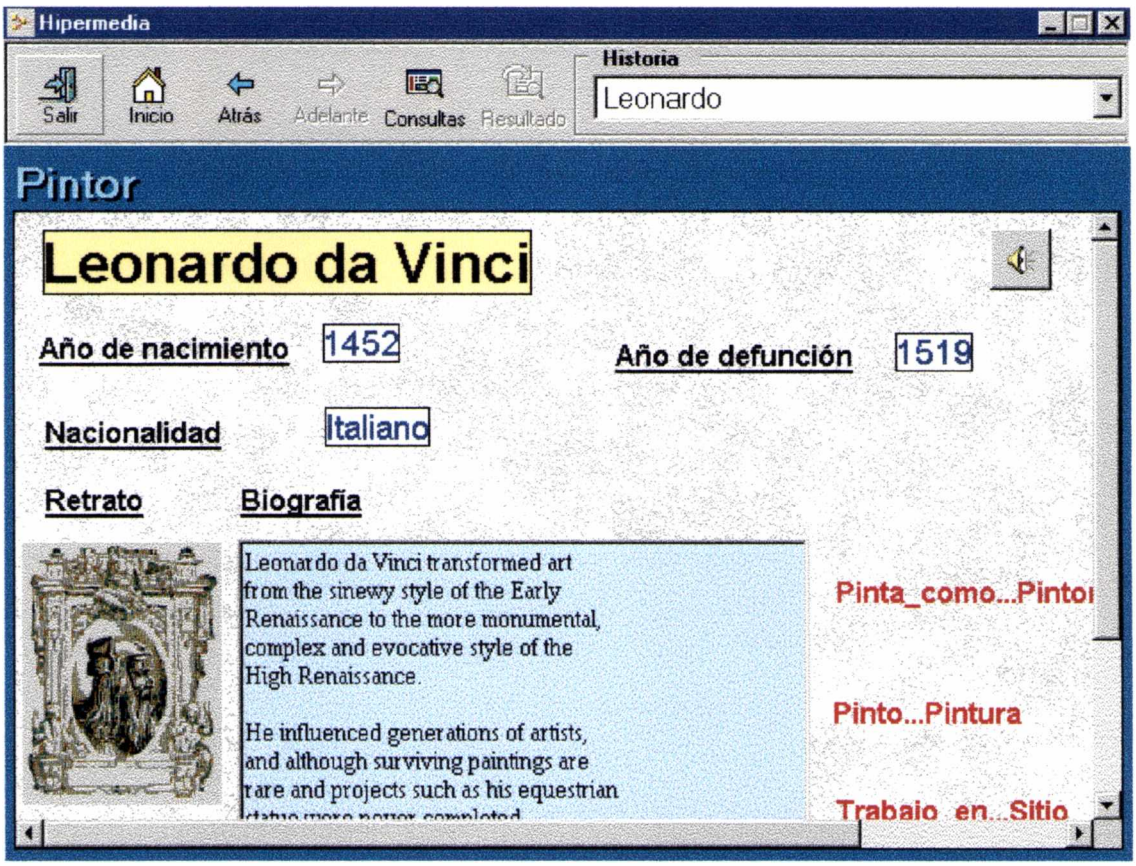
Una vez realizadas todas las instancias de las entidades y de las relaciones, debemos indicarle a la herramienta cuál será el **Nodo Inicial** de la hipermedia.

Para ello elegimos al nodo de la entidad 'Pintor', con el ejemplar 'Ej. Pintor completo' y por la instancia de 'Pintor' 'Leonardo'.



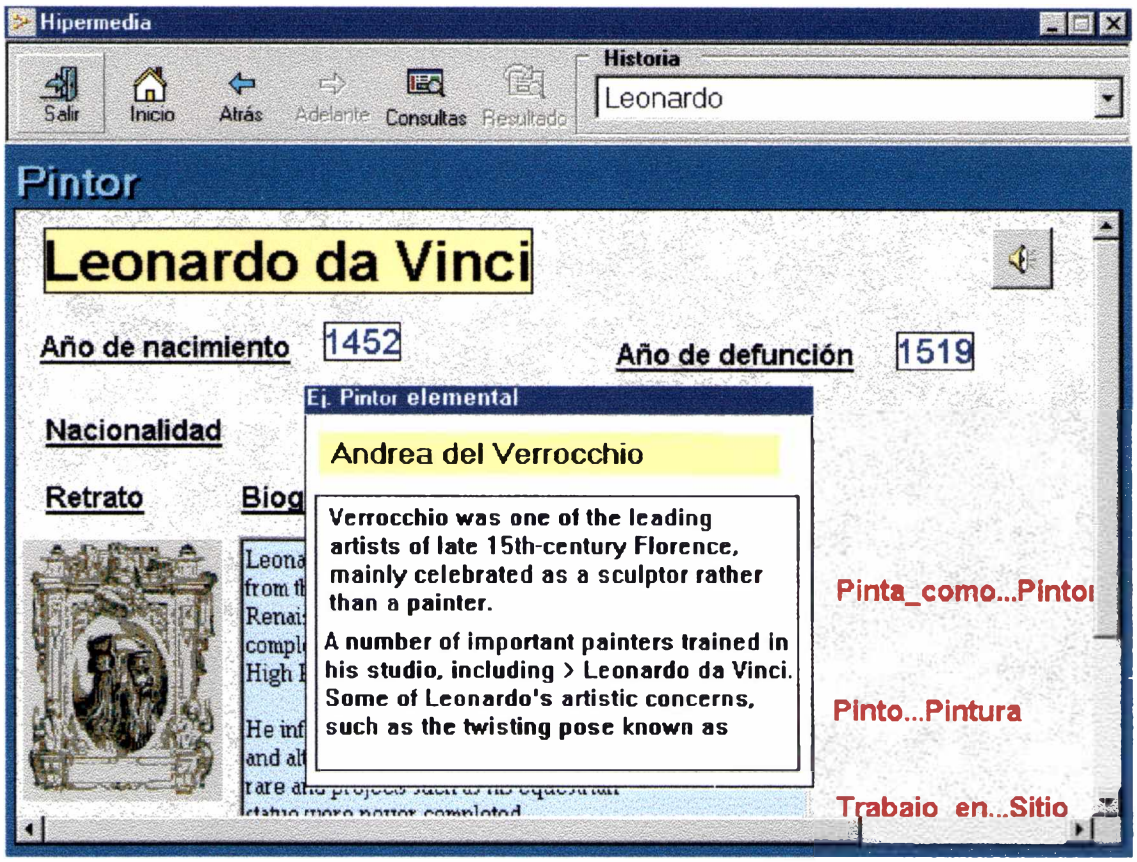
Nodo:

Cuando accedamos a algún nodo, veremos que el mismo estará mostrando una instancia, y que los atributos y anchors que aparezcan en ella (con sus respectivas características), serán los que se hayan definido en el ejemplar (y en la representación visual) con la que haya sido definido dicho nodo. Tomemos para nuestro ejemplo al nodo definido como inicial. El nodo que aparece en la hipermedia será:



El ejemplar utilizado para visualizar este último nodo, tiene la característica de estar maximizado, por ello dicho nodo ocupa el máximo posible dentro del espacio de visualización.

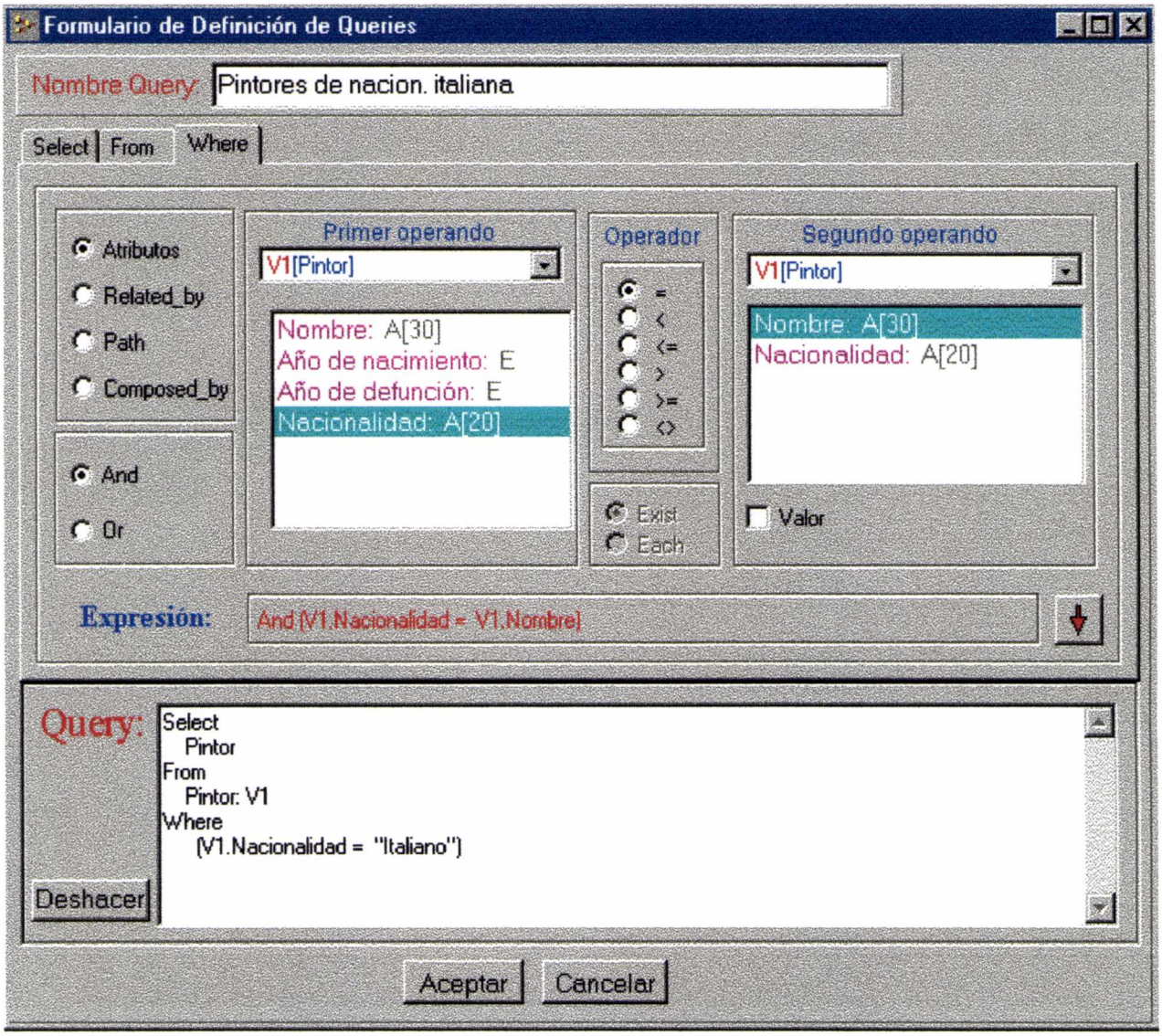
Supongamos ahora que la instancia recientemente definida (Leonardo da Vinci) que está representado con el ejemplar 'Ej. Pintor completo', está vinculado con el pintor 'Andrea del Verrocchio' mediante la relación 'Pintores Relacionados' y que ya hemos instanciado dicha reación. El ejemplar al que se accede mediante ella desde el nodo con el que se está visualizando al 'Pintor' Leonardo da Vinci, es el ejemplar 'Ej. Pintor elemental', el cual tiene la representación visual de nodo no maximizado. Por lo tanto cuando se navegue desde el primero hacia el segundo, veremos que el nodo mostrando la información del pintor 'Andrea del Verrocchio' aparece de la siguiente manera:



Queries:

Para finalizar nuestro ejemplo veremos la manera en que se definen las consultas que se desea realizar acerca de la información que contiene la hipermedia. Supongamos que nuestra intención es la de obtener un acceso rápido a aquellos 'Pintores' cuya nacionalidad sea 'Italia'. Al acceder al formulario de 'Definición de Queries', tendremos que darle un nombre a nuestra nueva consulta: 'Pintores italianos'. Luego, en la definición del 'Select' elegimos la entidad 'Pintor', y en la opción 'Where' seleccionamos el atributo 'Nacionalidad' '=' 'Italia'.

El query llamado **'Pintores de nacion. italiano'** quedará definido de la siguiente manera:



Al ser activado este query, la herramienta devolverá una estructura de acceso (como se muestra a continuación) a todos aquellos nodos de los Pintores cuya nacionalidad sea 'Italia'. Entre otros, obtendremos un acceso directo al nodo con información sobre 'Leonardo' (da Vinci) que es el mismo que definimos en el ejemplo previamente detallado.

7. PROBLEMAS Y SOLUCIONES

Durante el desarrollo del proyecto nos hemos encontrado con diferentes problemas relacionados tanto con el diseño de algunos aspectos de la herramienta como con la implementación misma.

En este sentido podemos decir que la modificación del esquema fue uno de los puntos en el que mayor cantidad de consideraciones hubo que tener en cuenta. Esto se debe a que un cambio en cualquiera de las clases no solo repercute en ellas si no que también lo hace sobre todas las subclases que tengan, sobre los ejemplares, su representación gráfica e instancias definidas.

Básicamente los principales problemas y soluciones adoptadas con respecto a la modificación del esquema están referidas a la eliminación de clases e instancias y a la modificación propiamente dicha de estas (consistente en el agregado, edición y eliminación de atributos o anchors). Para el agregado de una clase simplemente hay que tener en cuenta que sólo se puede hacer si se la define como una subclase de alguna de las ya existentes, o de las clases abstractas de cada jerarquía (**Nodo, Link** o **Ejemplar**). Recordemos que la relación 'es_un' entre dos clases no se la puede definir explícitamente como tal sino que queda determinada con la definición de una subclase.

Modificación del esquema

Lo que sigue es el detalle de los problemas encontrados relacionados con la alteración del esquema y la resolución que se le ha dado, denotándose las restricciones generadas.

- La eliminación de una clase implicaba una reestructuración de todas sus subclases. Esto consistía no solo en lograr que se refleje en forma adecuada sus nuevas estructuras que quedaban determinadas por la herencia, sino que

además se debían modificar todos los ejemplares de todas las subclases ya que podían quedar inconsistentes con referencias a atributos o anchors que ya no existían. Lo que se hizo es permitir el borrado sólo de aquellas clases que no tienen subclases, disminuyendo de esta manera el costo de la operación, y la desorientación producida hacia el usuario. De esta manera para eliminar una clase, previamente se deben eliminar todas sus subclases.

- Si se quisiera eliminar una clase nodo la cual interviene en alguna relación, éstas también deberían ser reestructuradas o eliminadas ya que de lo contrario el *from* como el *to* podrían quedar referenciando a una entidad que ya no existe. Análogamente con el punto anterior se determinó que solamente se pueden eliminar aquellas clases nodos de las cuales no sale o no llega ninguna relación. Es decir que para borrar una entidad se deben borrar previamente todos los vínculos en los que interviene.
- Si se eliminaba un ejemplar podría ocurrir que en el momento de la navegación la hipermedia intente acceder a un nodo al que le fue borrada la estructura de visualización. Lo que se hizo es asociarle al nodo, inmediatamente después a la eliminación del ejemplar con el que se muestra, el ejemplar por defecto (el que se genera automáticamente con el agregado de la clase nodo). De está forma y no permitiendo que el ejemplar por defecto sea eliminado o modificado se evita que un anchor conduzca a un nodo que carezca de un ejemplar para su visualización.
- Si se agregaba un atributo o un anchor a un ejemplar con subejemplares ya definidos, naturalmente esta modificación se reflejaba también en tales subejemplares, afectando directamente las representaciones visuales de cada uno de ellos. Para evitar este inconveniente decidimos no permitir modificar un ejemplar si tiene subejemplares. A los ejemplares que se permiten modificar, agregando un atributo (o un anchor), la herramienta le da un formato de muestra, por defecto.

Consideraciones sobre la Herramienta

Por otro lado existieron otros problemas, algunos de los cuales también están relacionados con la modificación del esquema pero con consecuencias ajenas al mismo. A continuación los enunciamos.

- En principio el único ambiente para visualizar las clases nodo y links del esquema era a través del diagrama del esquema de la aplicación. Como este hace hincapié en las relaciones conceptuales no visualiza la jerarquía de las clases link e inclusive muchas veces imposibilita observar con claridad y precisión la jerarquía de clases nodo (algo muy necesario para interpretar la herencia), decidimos agregar una ventana donde se muestra la jerarquía de links y las entidades organizadas taxonómicamente.
- Si se agregaba una clase utilizando la interface de jerarquías, el diagrama quedaba inconsistente al no reflejar dicha clase. Lo que hubo que hacer es darle un posicionamiento por defecto en el diagrama pudiendo el usuario reorganizarlo cuando desee.
- Cuando se agrega una relación al esquema como una subclase de una relación que es multidominio (más de un tipo de entidad en el to), la nueva relación debe tener un vinculo definido hacia cada una de las entidades que conforman el *to* de su superclase, o hacia una de sus subclases. El inconveniente que existía era que durante la definición de la nueva relación se podía indicar solamente alguno de los vínculos y no todos. La solución fue asumir que los no definidos se consideraban heredados con lo que se reflejan en el diagrama automática e inmediatamente después de la definición de la relación, evitando exigirle al usuario que defina todos explícitamente. Supongamos la relación multidominio 'viven_en', definida desde Persona hasta Vivienda y País. También tenemos la clase Indígena definida como especialización de Persona y la clase Choza como especialización de Vivienda. Ahora si agregamos la relación 'viven_en_1', como subclase de 'vive_en', indicando el origen

Indígena, con solo definir un posible destino, por ejemplo Choza, se asume automáticamente la otra parte del dominio, en este caso País.

- Cuando la relación es multidominio puede ser que, en el momento de la elección de los anchors en un ejemplar se quiera seleccionar sólo uno de los vínculos, de la misma manera que si un atributo tiene más de un dominio se puede querer mostrarlo sólo en alguno de sus formatos. Para permitir esto, cuando se está diseñando un ejemplar, los atributos y las relaciones multidominio se muestran en forma separada, una vez para cada dominio, con el mismo nombre de atributo o relación y distinguiéndolas por medio del dominio o de la entidad de destino respectivamente. La misma solución fue adoptada para las condiciones de las consultas que se formulan.
- En un comienzo con respecto a las consultas sobre la información de la aplicación, la idea era que se pudieran recuperar las que ya se habían realizado para reformularlas sin la necesidad de volver a definir. Con las modificaciones del esquema pasaba que algunas de las consultas hechas quedaban con referencias a clases o atributos que ya no existían. De esta manera se decidió que cada vez que se quiera realizar una consulta sobre la información, la misma debe ser definidas para asegurar la consistencia con las clases del esquema ya que el editor siempre está actualizado.
- Como ya vimos, la herramienta permite que el ejemplar utilizado para visualizar un nodo sea definido de dos maneras diferentes. De una forma el nodo se muestra de forma tal que ocupa todo el espacio de muestreo del navegador y se puede ir hacia otros nodos. En cambio si el ejemplar no está maximizado, se muestra arriba del que se estaba visualizando. Es un nodo sólo para mostrar información, sin anchors visualizados, cuyo tamaño lo define el usuario en diseño, y que debe ser cerrado para poder seguir navegando. El problema que había surgido es que estos ejemplares pueden ser elegidos, tanto para visualizar el nodo inicial de la hipermedia, como para visualizar los nodos resultantes de una consulta (cada uno un nuevo punto de

partida para navegar). En ambos casos un ejemplar contextual no tiene sentido ya que justamente no hay hasta ese momento otro nodo relacionado visualizado. En lugar de evitar que un ejemplar de este tipo sea elegido ya sea como nodo inicial o para mostrar una consulta, decidimos, en el caso que ocurra, obviar el formato definido y mostrar al nodo en forma maximizada, con sus anchors, para que se pueda comenzar a navegar. Si luego, durante la navegación, se vuelve a acceder a él, el formato definido en diseño es respetado.

- En el momento del diseño de un ejemplar, muchas veces se agregaba un anchor con una referencia a un ejemplar que todavía no había sido creado. Esto traía como consecuencia que se podía intentar navegar hacia un nodo cuya perspectiva estaba definida con un ejemplar inexistente. La solución adoptada consiste en limitar dicha referencia a un ejemplar existente (tengamos en cuenta que al menos existe el ejemplar por defecto).
- Cuando se intentaba navegar a través de un anchor, cuya instancia de la relación que representa no había sido definida o había sido eliminada, se estaba queriendo acceder un nodo que carecía de información para visualizar. Las soluciones planteadas consistían en informar la situación producida a través de un mensaje que se emita en el momento de selección del anchor sin permitir acceder al nuevo nodo; o bien brindar el acceso al nuevo nodo y mostrando en este una representación de un nodo sin información. Al considerarlas soluciones funcionalmente similares adoptamos indistintamente una de ellas, quedándonos con la primera.
- Otro de los problemas que hubo que solucionar fue con respecto a las relaciones de cardinalidad N. Había que encontrar la forma de que durante la navegación, en el momento de seleccionar un anchor que representa a dicha relación, se pudiese determinar a cual instancia se intenta acceder. En el momento de diseño de un ejemplar solamente se puede elegir un único anchor por relación, ya que éstos representan relaciones y no instancias, y además

sería imposible determinar la cantidad de instancias de una relación de cardinalidad N que puede haber definidas en el momento de la navegación. Lo que se resolvió consiste en utilizar una ventana del estilo pop-up. Esta se arma dinámicamente y se despliega en el momento de selección del anchor, mostrando una lista de nuevos anchors (uno para cada instancia) de los cuales se puede seleccionar sólo uno de ellos para de esta manera, finalmente, determinar la instancia a la que se quiere acceder. Si la cardinalidad de la relación es N, pero se definió una sola instancia, la ventana se omite accediendo directamente a la instancia mencionada.

- Al no poseer Delphi una componente adecuada para la visualización y edición de los atributos multivaluados (en particular si contienen sonidos, animaciones, texto o gráficos) surgió la necesidad de crearla. La misma es de tamaño variable, y además de mostrar a los elementos que componen el multivaluado, está conformada de componentes primitivas del Delphi que permite movernos entre ellos.

8. POSIBLES EXTENSIONES

A la herramienta presentada se le podría, en un futuro, realizar una serie de ampliaciones orientadas a facilitar el uso de la misma ya sea desde el punto de vista del diseño, como del uso de la hipermedia resultante. Algunas de las extensiones posibles son:

- Lograr un ambiente de desarrollo en el que se pueda diseñar más de una hipermedia, cada una con su esquema, y juego de datos, posibilitando además que en un esquema se definan clases importadas de otro.
- La herencia existente entre las clases ya sea en la jerarquía de Nodos, Links o Ejemplares es simple, es decir una subclase tiene un solo 'padre'. La posibilidad de definir herencia múltiple sería una ampliación importante para el diseño del esquema en la herramienta.
- Brindar la posibilidad de diseñar más de una representación gráfica a un mismo ejemplar.
- Agregar predicados de mayor complejidad a las expresiones de las consultas sobre la información de la aplicación, como por ejemplo, obtener el camino más directo entre dos nodos.
- Lograr la importación y exportación de los datos de las instancias, a formatos que puedan ser interpretados y aprovechados por otras aplicaciones, que permitan la manipulación de los datos de una menera diferente.

9. CONCLUSIONES

Hemos presentado una herramienta para diseñar y consultar aplicaciones hipermedias basada en un modelo orientado a Objetos (Design and Query Strategies to Hypermedia Applications).

Lo que se buscó es brindar un ambiente integrado de desarrollo que además de aplicar el modelo, con las sabidas ventajas que esto implica, le permita al usuario realizarlo de manera sencilla y natural.

Con la utilización de la herramienta se puede diseñar una hipermedia estructurando la información a través de los conceptos de nodos, relaciones y ejemplares. La aplicación se puede modelar en diferentes niveles de abstracción sin perder las facilidades de una hipermedia tradicional. A su vez el mantenimiento de las aplicaciones diseñadas resulta más fácil ya que existe un mayor entendimiento de su dominio, el cual se conforma siguiendo las características del paradigma orientado a objetos.

Por otro lado con la herramienta se puede recuperar la información de la hipermedia resultante. Esto se puede hacer mediante los mecanismos primitivos de navegación y a través de un método alternativo. Dicho método, a partir del esquema definido y basándose en el lenguaje de consultas enunciado en el modelo, permite el acceso a la información que se requiera, de una manera directa e inmediata, evitando así la eventual 'desorientación' con la que se puede encontrar el usuario cuando navega. La capacidad de consultar no sólo se limita a la información de la aplicación, ya que también se aplica sobre el esquema. Esta cualidad es aprovechada por el diseñador de la hipermedia al permitirle estar en todo momento en conocimiento del dominio de la aplicación.

Con lo expuesto hasta aquí, podemos asegurar la viabilidad de la implementación del modelo.

APENDICE A

Transformación de las Consultas para su ejecución

Para un mejor entendimiento del proceso de transformación de cada sentencia analizaremos varios casos de consultas que se pueden formular. En esos ejemplos veremos el significado de la consulta, su representación en el lenguaje del modelo, y a través de un ejemplo su equivalente en el modelo relacional característico del lenguaje de desarrollo de la herramienta.

Caso 1: Selección de instancias de una Clase Nodo definida en el esquema conceptual.

Lenguaje de consulta:

Select *NombreClaseNodo*
From *NombreClaseNodo: variable*

Ejemplo:

Select *Persona*
From *Persona: v1*

Transformación:

Supongamos el caso que en el esquema conceptual se hayan definido dos especializaciones para la Clase Persona: Estudiante y Profesor, representados internamente con tablas denominadas Enti0002 para Persona, Enti0003 para Estudiante y Enti0004 para Profesor.

```
Select Distinct
  v1.CodigoEntidad, v1.CodigoDeInstancia, v1.Visitado,
  v1.NombreInstancia E_Persona
From "_nti0002" V1
```

La tabla '_nti0002', contiene instancias de las clases Persona, Estudiante y Profesor. NombreInstancia es renombrado como E_Persona, de esta forma se puede determinar a

posteriori la entidad principal a la cual se le aplico la consulta.

Caso 2: Selección de instancias de una Clase Nodo definida en el esquema conceptual que cumplan cierta condición.

Lenguaje de Consulta:

Select *NombreClaseNodo*
From *NombreClaseNodo : variable*
Where *variable.NombreAtributo = valor*

La palabra ***valor*** denota a un literal valido en el dominio del NombreAtributo, y el operador (=) fue puesto como ejemplo de uno de los operadores relacionales definidos en el modelo

Ejemplo:

Select *Persona*
From *Persona : v1*
Where *(v1.Nombre='Juan')*

Transformación:

Bajo la suposición del Caso 1, pero con el atributo Nombre (definido en Persona) internamente de la forma A00020001(siguiendo la nomenclatura de **Estructuras de Archivos**) queda:

```
Select Distinct
  v1.CodigoEntidad, v1.CodigoDeInstancia, v1.Visitado,
  v1.NombreInstancia E_Persona
From "_nti0002" v1
Where
  (v1.A00020001 = 'Juan')
```

Caso 3: Selección de atributos de instancias de una Clase Nodo (proyección), con una condición.

Lenguaje de Consulta:

Select
NombreClaseNodo.NombreAtributo1,

NombreClaseNodo.NombreAtributo2,

From *NombreClaseNodo : variable*

Where *(variable.NombreAtributo3 = valor)*

Ejemplo:

Select *Persona.Nombre, Persona.Edad*

From *Persona : v1*

Where *(v1.Ciudad = 'La Plata')*

Transformación:

Bajo la suposición del Caso 1, y considerando los atributo Nombre, Edad y Ciudad definidos en Persona e internamente de la forma A00020001, E00020002 y A00020003, respectivamente.

```
Select Distinct
  v1.CodigoEntidad, v1.CodigoDeInstancia, v1.Visitado,
  v1.A00020001 A_Persona_Nombre,
  v1.E00020002 A_Persona_Edad
From "_nti0002" v1
Where
  (v1.A00020003 = 'La Plata')
```

Se renombran los campos que representan los atributos de la entidad Persona para usarlos posteriormente (por ejemplo A00020001 pasa a ser 'A_Persona_Nombre').

Caso 4: Selección de las instancias de una Clase Nodo, que estén relacionadas con las instancias de otra Clase Nodo.

Lenguaje de Consulta:

Select *NombreClaseNodo1*

From

NombreClaseNodo1 : variable1

NombreClaseNodo2 : variable2

Where ***(Related_By****(variable1, NombreClaseLink, variable2)*

Ejemplo:

Select *Persona*

From

Persona : v1

País : v2

***Where (Related_by** (v1, habita_en, v2)*

Transformación:

Bajo la suposición de los Casos anteriores y considerando *País* y *habita_en* definidos internamente como Enti0005 y Enti0006, respectivamente.

```
Select Distinct
  v1.CodigoEntidad, v1.CodigoDeInstancia, v1.Visitado,
  v1.NombreInstancia E_Persona
From
  "_INSTR1" i1, "_nti0002" V1, "_nti0005" V2
Where
  (((i1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (V2.CodigoEntidad="0005") and (i1.CodigoEntidadDestino="0005") and
  (i1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)  and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
  or
  ((i1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (V2.CodigoEntidad="0005") and (i1.CodigoEntidadDestino="0005") and
  (i1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)  and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
  or
  ((i1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
  (V2.CodigoEntidad="0005") and (i1.CodigoEntidadDestino="0005") and
  (i1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)  and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino)))
```

La tabla '_INSTREL1' aparece al existir en el Where la cláusula Related_by. Obsérvese que el tamaño de la sección del Where a aumentado considerablemente, esto es debido a la necesidad de tener que expresar por un lado el vínculo entre las instancias de Personas y País, a través de la tabla interna '_INSTREL1', y de reflejar el **efecto de propagación** a las subclases de la Entidad Persona. En definitiva analizando la composición de esta sección, podemos identificar tres partes: la primera establece si

hay vínculos entre Persona y País, la segunda entre Estudiante y País, mientras que la ultima entre Profesor y País. El resultado de esta consulta es de similares características a las antes mencionadas.

Caso 5: Selección de las instancias de una Clase Nodo, que estén relacionadas con las instancias de otra Clase Nodo (similar al anterior caso), pero ahora se consideran especializaciones en ambas Clases.

Lenguaje de Consulta:

Select *NombreClaseNodo1*
From
NombreClaseNodo1 : variable1
NombreClaseNodo2 : variable2
Where (Related_By*(variable1, NombreClaseLink, varible2)*

Ejemplo:

Select *Persona*
From
Persona : v1
País : v2
Where (Related_by *(v1, habita_en, v2)*

Transformación:

Bajo la suposición de los Casos anteriores, pero País teniendo dos especializaciones: País Americano y País Europeo, definidos internamente como Enti0007 y Enti0008, respectivamente.

```
Select Distint
  v1.CodigoEntidad, v1.CodigoDeInstancia, v1.Visitado,
  v1.NombreInstancia E_Persona
From
  "_INSTR1" i1, "_nti0002" V1, "_nti0005" V2
Where
  (((i1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (V2.CodigoEntidad="0005") and (i1.CodigoEntidadDestino="0005") and
  (i1.CodigoRelacion="0006") and
```

```
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
                    (V2.CodigoEntidad="0007") and (i1.CodigoEntidadDestino="0007") and
                    (i1.CodigoRelacion="0006") and
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
                    (V2.CodigoEntidad="0008") and (i1.CodigoEntidadDestino="0008") and
                    (i1.CodigoRelacion="0006") and
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
                    (V2.CodigoEntidad="0005") and (i1.CodigoEntidadDestino="0005") and
                    (i1.CodigoRelacion="0006") and
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
                    (V2.CodigoEntidad="0007") and (i1.CodigoEntidadDestino="0007") and
                    (i1.CodigoRelacion="0006") and
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
                    (V2.CodigoEntidad="0008") and (i1.CodigoEntidadDestino="0008") and
                    (i1.CodigoRelacion="0006") and
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
                    (V2.CodigoEntidad="0005") and (i1.CodigoEntidadDestino="0005") and
                    (i1.CodigoRelacion="0006") and
                                        (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))
                    or
                    ((i1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
                    (V2.CodigoEntidad="0007") and (i1.CodigoEntidadDestino="0007") and
                    (i1.CodigoRelacion="0006") and
```

(V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen) and (V2.CodigoDeInstancia=i1.CodigoInstanciaDestino))

or

((i1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and (V2.CodigoEntidad="0008") and (i1.CodigoEntidadDestino="0008") and (i1.CodigoRelacion="0006") and (V1.CodigoDeInstancia=i1.CodigoInstanciaOrigen) and (V2.CodigoDeInstancia=i1.CodigoInstanciaDestino)))

Ahora aparecen nueve partes claramente delineadas, cada una de esta corresponde a la comparación entre cada subclase propia de la primer entidad con la una de las subclases propias de la segunda entidad que interviene en el **Related_by**. En el caso anterior como Persona tiene tres subclases propias, al igual que País, aparecen nueve partes (producto cartesiano entre ambas).

Caso 6: Selección de instancias de una Clase Nodo, para las cuales existe al menos un camino posible (**Path**) con alguna instancia de otra Clase Nodo. Mostraremos el caso con una sola Clase Link en el camino para mostrar su similitud con el **Related_by**.

Lenguaje de Consulta:

Select *NombreClaseNodo1*
From
NombreEntidad1 : variable1
NombreEntidad2 : variable2
Where (Path*(v1, habita_en ,v2))*

Ejemplo:

Select *Persona*
From
Persona : v1,
País : v2
Where ***(Path*** *(v1, habita_en, v2)*

Transformación:

Bajo la suposición de los casos anteriores.

```
Select Distinct
  v1.CodigoEntidad,  v1.CodigoDeInstancia,  v1.Visitado,
  v1.NombreInstancia E_Persona
From
  "_InstRP1" ii1,  "_nti0002"  V1,  "_nti0005"  V2
Where
  ((((ii1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (ii1.CodigoEntidadDestino="0005") and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (ii1.CodigoEntidadDestino="0007")and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (ii1.CodigoEntidadDestino="0008") and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (ii1.CodigoEntidadDestino="0005") and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (ii1.CodigoEntidadDestino="0007") and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (ii1.CodigoEntidadDestino="0008") and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
  (ii1.CodigoEntidadDestino="0005") and (ii1.CodigoRelacion="0006") and
                    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen)      and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
```

((ii1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
(ii1.CodigoEntidadDestino="0007") and (ii1.CodigoRelacion="0006") and
(V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
or
((ii1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
(ii1.CodigoEntidadDestino="0008") and (ii1.CodigoRelacion="0006") and
(V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
(V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino))))

Se observa gran similitud sintáctica con el Related_by, en cuanto a la composición de sus partes. Aparece una nueva tabla interna, '_InstRP1', la cual es análoga a la tabla '_InstRel1', usada en la cláusula Related_by, y se generaran tantas como cláusulas Path aparezcan en la consulta.

Caso 7: Selección de instancias de una Clase Nodo, para las cuales existe al menos un camino posible (**Path**) con alguna instancia de otra Clase Nodo. Mostraremos el caso con varias Clases Link en el camino.

Lenguaje de Consulta:

Select *NombreClaseNodo1*
From
NombreClaseNodo1 : variable1
NombreClaseNodo2 : variable2
Where
*(**Path**(variable1,NombreClaseLink1,.....,NombreClaseLinkN,variable2)*

Ejemplo:

Select *Estudiante*
From
Estudiante : v1,
Océanos : v2
Where (Path *(v1, habita_en, ubicado_en, rodeado_por, v2)*

La consulta intenta conocer a aquellos estudiantes que habitan en países que pertenecen a continentes rodeados por Océanos.

Transformación:

Bajo la suposición de los Casos anteriores, pero con nuevas Entidades como Continente y Océanos, representados internamente por las tablas Enti0010 y Enti0012, y con las relaciones Ubicado en (de País a Continente) representado por Enti0011 y Rodeado por (de Continente a Océanos) esta dado por la tabla Enti0013.

```
Select Distinct
  v1.CodigoEntidad, v1.CodigoDeInstancia, v1.Visitado,
  v1.NombreInstancia E_Estudiante
From
  "_InstRP1" ii1, "_nti0005" A1, "_nti0010" A2, "_nti0003" V1, "_nti0012" V2
Where
  ((((ii1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (ii1.CodigoEntidadDestino="0005") and (ii1.CodigoRelacion="0006") and
  (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
  (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (ii1.CodigoEntidadDestino="0007") and (ii1.CodigoRelacion="0006") and
  (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
  (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0002") and (V1.CodigoEntidad="0002") and
  (ii1.CodigoEntidadDestino="0008") and (ii1.CodigoRelacion="0006") and
  (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
 (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (ii1.CodigoEntidadDestino="0005") and (ii1.CodigoRelacion="0006") and
  (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
  (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (ii1.CodigoEntidadDestino="0007") and (ii1.CodigoRelacion="0006") and
  (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
  (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
  or
  ((ii1.CodigoEntidadOrigen ="0003") and (V1.CodigoEntidad="0003") and
  (ii1.CodigoEntidadDestino="0008") and (ii1.CodigoRelacion="0006") and
  (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
  (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
```

```
    or
    ((ii1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
    (ii1.CodigoEntidadDestino="0005") and (ii1.CodigoRelacion="0006") and
    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
   (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
    or
    ((ii1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
    (ii1.CodigoEntidadDestino="0007") and (ii1.CodigoRelacion="0006") and
    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
    (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino))
    or
    ((ii1.CodigoEntidadOrigen ="0004") and (V1.CodigoEntidad="0004") and
    (ii1.CodigoEntidadDestino="0008") and (ii1.CodigoRelacion="0006") and
    (V1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen) and
    (A1.CodigoDeInstancia=ii1.CodigoInstanciaDestino)))
  and
    (((ii1.CodigoEntidadOrigen_1="0005") and (A1.CodigoEntidad="0005") and
    (ii1.CodigoEntidadDestino_1="0010") and (ii1.CodigoRelacion_1="0011") and
    (A1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen_1) and
   (A2.CodigoDeInstancia=ii1.CodigoInstanciaDestino_1))
    or
    ((ii1.CodigoEntidadOrigen_1="0007") and (A1.CodigoEntidad="0007") and
    (ii1.CodigoEntidadDestino_1="0010") and (ii1.CodigoRelacion_1="0011") and
    (A1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen_1) and
   (A2.CodigoDeInstancia=ii1.CodigoInstanciaDestino_1))
    or
    ((ii1.CodigoEntidadOrigen_1="0008") and (A1.CodigoEntidad="0008") and
    (ii1.CodigoEntidadDestino_1="0010") and (ii1.CodigoRelacion_1="0011") and
    (A1.CodigoDeInstancia=ii1.CodigoInstanciaOrigen_1) and
   (A2.CodigoDeInstancia=ii1.CodigoInstanciaDestino_1)))
  and
    (((ii1.CodigoEntidadOrigen_2="0010") and (A2.CodigoEntidad="0010") and
    (ii1.CodigoEntidadDestino_2="0012") and (ii1.CodigoRelacion_2="0013") and
    (A2.CodigoDeInstancia=ii1.CodigoInstanciaOrigen_2) and
   (V2.CodigoDeInstancia=ii1.CodigoInstanciaDestino_2))))
```

Aquí aparecen tablas auxiliares como son "_nti0005" que contienen información referente a la clase País y "_nti0010" que denota a Continente. Estas tablas son utilizadas como intermediarias entre las relaciones, y obviamente representan a todas las instancias propias de la entidad a la que pertenecen. Por otro lado, podemos distinguir tres grupos, cada uno de ellos separados por las cláusulas **and** (habrá tantos grupos

como relaciones participen de la consulta) donde se analizan las instancias de las relaciones entre las entidades en origen y destino de cada relación. Luego de la conjunción de estas surgen aquellas instancias que verifican el predicado de la consulta.

Caso 8: Selección de instancias de una Clase Nodo, con una condición aplicada sobre un atributo multivaluado, mediante la cláusula **Exist**.

Lenguaje de Consulta:

Select NombreClaseNodo1
From NombreClaseNodo1 : variable1
Where (Exist v1.NombreAtributo1 = valor)

Ejemplo:

Select Persona
From Persona : v1
Where (**Exist** v1.Telefonos = '01-2345678')

Transformación:

```
Select Distinct
  v1.CodigoEntidad, v1.CodigoDeInstancia,  v1.Visitado,
  v1.NombreInstancia E_Persona
From
  "_nti0002"  V1
```

La tabla '_nti0002', contiene instancias de las clases Personas, Estudiante y Profesor, que cumplen la condición explicitada en el Where del ejemplo. Notemos que esta equivalencia no hace referencia a la cláusula Exist, es decir que el testeo de verificación del predicado de la consulta lo realizó el Módulo de Transformación, previo a la transformación en sentencias de salida.

Caso 9: Selección de instancias de una Clase Nodo, con una condición aplicada sobre un atributo multivaluado, mediante la cláusula **Each**.

Este caso no lo ejemplificamos ya que es análogo a lo anterior.

APENDICE B

Manual del ususario

Al ingresar a la herramienta generadora de Hipermedias, el usuario se encontrará con una interface que le permite al usuario hacer uso de la misma mediante los botones que se encuentran en la barra de herramientas, o a través del menú.

Barra de herramientas principal.

La barra de herramientas principal se encuentra disponible para su uso en forma permanente. La misma cuenta con las funciones más elementales para la utilización de la herramienta. Estas funciones son:

Salir: abandona la herramienta.

Diagrama: permite visualizar la representación del esquema de la aplicación mediante el diagrama correspondiente. La selección de esta opción habilita el uso de una nueva barra de herramientas.

Jerarquía: permite visualizar la representación del esquema de la aplicación mediante los dos árboles de jerarquías (Entidades y Relaciones).

Cambiar: posibilita alternar entre el modo de uso del diagrama o el de los árboles de jerarquías.

Hipermedia: es la función a través del cual el usuario puede acceder a la hipermedia resultante de la aplicación generada.

Instancia: posibilita acceder a la interface de instanciación de la Entidad seleccionada y de sus correspondientes relaciones.

Barra de herramientas del diagrama.

Esta barra permite realizar diferentes funciones sobre el diagrama. Ellas son:

Manejador: cuando este botón es seleccionado, posibilita a continuación que cualquier Entidad o Relación existente en el diagrama sea "elegida" para su posterior manipuleo. En el caso de las Entidades permite reposicionarlas dentro del diagrama, operar con sus atributos (agregar, borrar y/o modificarlos), acceder a la interfaces de los Ejemplares y sus formas de trabajarlos, investigar las subclases o superclases que contengan, las clases alcanzables y las clases de entrada (a partir de las cuales es posible llegar hasta la Entidad), eliminar una Entidad y ver sus propiedades. En el caso de las Relaciones, permite reposicionarlas, investigar las subclases o superclases que contengan, eliminar una Relación y ver sus propiedades.

Entidad: la selección de este botón, permite al usuario crear una nueva Entidad que será subclase de la clase conceptual "Nodo", accediendo a la interface que le pedirá la definición del nombre y los atributos que contenga.

Subentidad: este botón permite a los usuarios acceder a una interface que posibilita la creación de una subclase de la entidad que se encuentre seleccionada con el "Manejador". Dicha interface es la misma que se utiliza para la definición de las Entidades.

Relación: este botón permite generar una nueva relación entre una o más entidades. La selección del mismo despliega una paleta para que el usuario seleccione cuál es la Entidad que actuará como atributo "From" en la nueva relación y cuáles las Entidades que actuarán como atributo "To" (mediante la selección de otro botón). Para finalizar, seleccionando el botón "Vincular" se accede a una interface para la definición del nombre, los atributos y la cardinalidad de la nueva relación.

Subrelación: permite a los usuarios crear una subrelación de la relación que esté seleccionada. El método de selección de las Entidades "From" y "To" es análogo al decripto para las Relaciones. Luego, mediante el botón "Vincular" se accede a la interface que permite la definición del nombre, los atributos y la cardinalidad

de la nueva subrelación.

Compuesta: este botón permite al usuario definir una nueva "Composición" de Entidades. El procedimiento es similar al de definición de las Relaciones, por lo cual se despliega una paleta análoga a la de dicho proceso. Al seleccionar el botón "Vincular, se despliega una interface que le pide al usuario la definición de la cardinalidad de la nueva Composición.

Menú principal del Diagrama del Esquema.

El menú principal del Diagrama del Esquema contiene las siguientes funciones:

Archivos: Permite seleccionar entre las formas de operar la herramienta, es decir con el modo Diagrama o con el modo de Jerarquías. También permite cerrar la aplicación.

Ventana: Permite cambiar de ventanas mediante las opciones Siguiente y Anterior.

Herramientas: Mediante la opción Navegador el usuario puede acceder al primer nodo de la Hipermedia generada por la herramienta. La opción Instanciador posibilita el acceso a la interface que permite cargar todas las instancias deseadas de cada entidad, y también las instancias de las relaciones de las cuales dichas entidades actúan como origen. El submenú de Consultas permite acceder a las interfaces que permiten generar nuevas consultas sobre la aplicación y la que permite investigar consultas ya realizadas. En la opción Nodo Inicial, se permite que el usuario seleccione la Entidad, el ejemplar y la instancia a la que se accederá como el primer nodo de la Hipermedia.

Arboles de Jerarquía.

Los árboles de Jerarquía son una manera alternativa de operar con la herramienta. El árbol de Jerarquía de Entidades permite visualizar todas las Entidades con sus

respectivas subclases y superclases definidas en el diseño de la hipermedia. Los elementos pueden ser seleccionados, y mediante el botón derecho del mouse se podrá acceder a operar con sus atributos (agregar, borrar y/o modificarlos), acceder a la interfaces de los Ejemplares y sus formas de trabajarlos, las clases alcanzables y las clases de entrada (a partir de las cuales es posible llegar hasta la Entidad), eliminar una Entidad, ver sus propiedades y agregar subclases.

Por su parte el árbol de Jerarquía de las Relaciones posee dos vistas diferentes: la de las clases Relaciones Conceptuales y las Relaciones Compuestas_por. Al igual que para las Entidades, los árboles jerárquicos de las Relaciones pueden ser operados mediante el uso del botón derecho del mouse. Si el árbol en cuestión es el de Relaciones Conceptuales, el evento brindará las opciones de agregar una subclase a la clase seleccionada, de ver o modificar sus propiedades y la de eliminar una Relación Conceptual. Si por otro lado operamos el árbol de Relaciones Compuestas_por, se puede anexar a la aplicación una nueva relación de este tipo, se pueden ver las propiedades, o bien se puede eliminar una de ellas.

Menú principal de la Jerarquía.

Además de las opciones expuestas recientemente para el Menú principal del Diagrama del Esquema, el menú de la jerarquía cuenta con otras tres opciones:

Entidades: El submenú que despliega esta opción es idéntico al descripto cuando se selecciona una opción en el árbol de Entidades y se cliquea el botón derecho del mouse.

Relaciones: Este submenú permite, teniendo seleccionada una Relación Conceptual en el árbol correspondiente, agregar una relación subclase de ella, eliminarla o modificarla.

Ejemplares: La opción Agregar accede a una interface que permite agregar un nuevo Ejemplar a la Entidad que se encuentra seleccionada en el árbol jerárquico de Entidades.

REFERENCIAS

[Gordillo,Díaz] Design and Query Strategies to Hypermedia Applications. 1995

Nielsen, Hipertext and Hypermedia 1993

An Object oriented model for querying hypermedia applications , S.Gordillo y A. Díaz 1996.

Building hypermedia application by querying design models, G. Rossi, S.Gordillo y A.Díaz 1996.

[Rumbaugh] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen: "Object Oriented Modeling and Design", 1991.

[Schwabe95] D. Schwabe and G. Rossi: "Defining Hypermedia Applications as Navigational Views of Hyperbases". Int. Conf. on System Science, Hawaii, 1995.

[Kim 95] W.Kim "Modern Database System"

Delphi 2 Unleashed, Charles Calvert. 1996

Local SQL Help, Object Pascal Help, Visual Component Library Help. Borland Delphi 3.