Trabajo de Grado

Título:

CASE para el Diseño de Aplicaciones Hipermediales Estructuradas.

Autores:

Mascena, Mario Enzo.
Montanaro, Carlos José.

Directora:

Licenciada Díaz, Alicia.

Codirector:

Licenciado Rossi, Gustavo.

Propósito:

Optar al Grado de Licenciado en Informática.

Lugar:

Universidad Nacional de La Plata.

Mayo de 1996.

TES
96/8
DIF-01934
SALA

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMATICA
Biblioteca
50 y 120 La Plata
catalogo.info.unlp.edu.ar
biblioteca@info.unlp.edu.ar

DIF-01934

Dedicatoria.

- A muestros padres y abuelos, por su paciencia.
- A nuestros compañeros y amigos.
- A 'Los Malditos'.

Indice.

Prólogo.

1.- Introducción.

1.1.- Descripción del problema. 2
1.2.- Objetivo del proyecto. 3
1.3.- Alcance del proyecto. 4

Sección 1: Fundamentos Básicos de Hipermedia.

1.- Que es Hipertexto ? 5

Sección 2: Modelos y Metodologías de Diseño.

1.- Generalidades. 8
2.- HDM: Un modelo para el diseño de Aplicaciones Hipertextuales. 9
2.1.- Características. 9
2.2.- Introducción. 9
2.3.- El modelo de diseño. 10
2.4.- Primitivas de HDM. 10
2.4.1.- Entidades y Componentes. 10
2.4.2.- Unidades y Perspectivas. 11
2.4.3.- Tipos de Entidades. 11
2.4.4.- Links y Tipos de Links. 11
2.4.5.- Derivaciones de Links. 12
2.4.6.- Esquema e Instancia de Esquema. 12
2.4.7.- Semánticas de Browsing. 13
2.5.- Ejemplo. 14
3.- OOHDM: Modelo y Metodología de Diseño de Hipermedia orientado a Objetos. 16
3.1.- Introducción. 16
3.2.- Proceso de Desarrollo. 17
3.2.1.- Paso 1: Análisis del Dominio (Modelización). 17
3.2.2.- Paso 2: Diseño Navegacional. 17
3.2.3.- Paso 3: Diseño de Interface Abstracta. 18
3.2.4.- Paso 4: Implementación. 18
3.3.- Ejemplo 18
4.- RM: Una Metodología para el diseño de Aplicaciones Hipermedias Estructuradas 20
4.1.- Introducción. 20
4.2.- Modelo de Datos. 21
4.3.- Metodología de Diseño. 24
4.3.1.- Paso 1: Diseño de Entidades y Relaciones. 25
4.3.2.- Paso 2: Diseño de Entidad. 26
4.3.3.- Paso 3: Diseño Navegacional. 27
4.3.4.- Diseño y construcción de la interface con el usuario. 29

Sección 3: Por qué se elige la Metodología RM ?

1.- Diferencias entre RM y HDM. 30
2.- Diferencias principales entre OOHDM y RM. 30
3.- Ventajas de RM. 31

Sección 4: RMCASE. Generalidades.

1.- Introducción. 32
2.- Proyecto RMCASE: Case para el diseño de aplicaciones Hipermedias. 33
2.1.- Modelo de Datos. 33
2.2.- Metodología de Diseño. 35
2.3.- Características de RMCASE. 36
2.3.1.- Proceso de Diseño. 36
2.3.2.- Contextos de Trabajo. 37
2.3.3.- Feedbacks Loops. 37
2.3.4.- Prototipación. 37
2.3.5.- Proceso de Diseño: Ejemplo. 38
2.4. Contextos de Trabajo. 39
2.4.1.- Contexto de Diseño de Entidades y Relaciones. 39
2.4.2.- Contexto de Diseño de Entidad. 39
2.4.3.- Contexto de Diseño Navegacional. 40
2.4.4.- Contexto de Población de la Hiperbase. 41
2.4.5.- Contexto para Establecer Relaciones a Nivel de Instancias. 42
2.4.6.- Contexto de Construcción del Prototipo. 42
2.4.6.1.- Consideraciones acerca del Protitipo. 43

Sección 5: RMCASE: Construyendo una Aplicación.

1.- Introducción. 44
1.1.- Que es un proyecto en RMCASE ? 44
2.- Cómo desarrollar una aplicación Hipermedia ? 44
2.1.- Generalidades de RMCASE. 44
2.1.1.- Proyectos. 45
2.1.2.- Aplicación. 45
3.- Trabajando con Proyectos. 46
3.1.- Crear un Proyecto nuevo. 46
3.1.1.- Editor de Diseño E-R. 46
3.1.1.1.- Crear una Entidad. 47
3.1.1.2.- Borrar una Entidad. 48
3.1.1.3.- Crear una Relación Asociativa. 48
3.1.1.4.- Borrar una Relación Asociativa. 49
3.1.1.5.- Propiedades de las Entidades. 49
3.1.1.6.- Propiedades de las Relaciones Asociativas. 50
3.1.2.- Editor de Diseño de Entidad. 51
3.1.2.1.- Crear un Slice. 52
3.1.2.2.- Borrar un Slice. 53
3.1.2.3.- Crear una Relación Estructural. 53

3.1.2.4.- Borrar una Relación Estructural. 53
3.1.2.5.- Propiedades de los Slices. 53
3.1.2.6.- Propiedades de las Relaciones Estructurales. 54
3.1.3.- Editor de Diseño Navegacional. 55
3.1.3.1.- Crear un Grouping. 56
3.1.3.2.- Borrar un Grouping. 57
3.1.3.3.- Crear un Link Navegacional. 57
3.1.3.4.- Borrar un Link Navegacional. 57
3.1.3.5.- Propiedades del Grouping. 57
3.1.3.6.- Propiedades de los Links Navegacionales. 58
3.1.3.7.- Propiedades de los Links Asociativos. 59
3.2.- Abrir un Proyecto. 60
3.3.- Carga de Datos. 60
3.3.1.- Carga de Datos de las Entidades. 61
3.3.1.1.- Carga de Datos Nuevos de las Entidades. 62
3.3.1.1.1.- Carga de Imágenes. 63
3.3.1.1.2.- Carga de Memos. 64
3.3.2.- Carga de Datos de los Groupings. 65
3.3.2.1.- Datos Nuevos de los Groupings. 66
3.4.- Establecer Relaciones entre las Instancias de las Entidades. 67
3.5.- Construcción del Prototipo. 69
3.6.- Ejecución. 70
4.- Conclusiónes. 72
5.- Futuras Extensiones. 74

Sección 6: Arquitectura de RMCASE.

1.- Introducción. 75
2.- Arquitectura. 75
3.- Utilización de las Tablas. 77

Apéndice.

Bibliografía.

Agradecimientos.

BIBLIOTECA
FAC. DE INFORMATICA
U.N.L.P

Prólogo.

En la actualidad las herramientas para el desarrollo de aplicaciones Hipermedia como HyperCard, ToolBook y otras, no ofrecen la posibilidad de "diseñar" la aplicación a desarrollar, solo permiten definir pantallas que se conectan entre sí y pueden ser recorridas en forma secuencial. Se puede observar que una aplicación simple, como por ejemplo mostrar un catálogo de productos, desarrollada con estas herramientas, en cuanto más extenso sea el catálogo, más tedioso se torna el desarrollo de la aplicación. Se debe definir una pantalla para cada producto del catálogo, con lo cual se deduce que la cantidad de pantallas a definir va a depender exclusivamente de la cantidad de productos del catálogo y luego se deben conectar las pantallas en el orden que se desee recorrer el catálogo en la aplicación (por rubro, por descripción, por código, etc.). Esta manera de trabajar es ineficiente e impráctica. Los problemas se agudizan en el caso de tener que agregar un dato más a los productos (por ejemplo presentación), lo cual significa que se debe modificar cada una de las pantallas de los productos, ya que no existe la posibilidad de agrupar las pantallas de un mismo tipo y manejarlas como tal; con lo cual la modificación y actualización de la aplicación se transforma en una tarea ardua y complicada.

Tampoco permiten trabajar con bases de datos ya existentes ni se pueden generar dentro de ellas; con lo cual siguiendo con el ejemplo anterior, si se dispone de una base de datos con la información de los productos del catálogo, no sería útil como ayuda para generar las pantallas de la aplicación. En resumen, este tipo de herramientas limitan la posibilidad que el desarrollador se valga de información adicional a partir de bases de datos externas para utilizarla en el desarrollo de la aplicación.

Todos estos problemas insúmen mayores esfuerzos desde el punto de vista económico y del tiempo utilizado, que luego tendrán incidencia directa en costo final de la aplicación.

Para resolver los problemas mencionados anteriormente proponemos una herramienta más potente que los soluciona ampliamente y agrega facilidades para que el diseñador pueda "diseñar" y desarrollar la aplicación rápidamente, proveyendo el mantenimiento de la documentación del diseño para facilitar también las futuras modificaciones y actualizaciones de la misma, con la consecuente reducción de los costos.

Nuestra herramienta es un ambiente computarizado que asiste al diseño y construcción de Aplicaciones Hipermediales con las características de un CASE.

Ahora es posible diseñar más rápida y fácilmente aplicaciones cada vez más complejas y extensas, utilizando una herramienta nueva que puede ser destinada a fines comerciales, con mejoras en la calidad del diseño y una considerable reducción de los costos en la aplicación Hipermedia final.

1.- Introducción.

1.1.- Descripción del problema.

Una aplicación Hipermedia, desde un punto de vista estructural, consiste de porciones de información en forma de texto, imágenes, sonido y video. Cada porción de información es llamada nodo. Cada nodo está conectado con otros nodos a través de *links*. Los *links* están asociados a una parte del nodo para poder navegar entre estos. Las asociaciones entre los nodos y *links* forman una estructura semejante a un grafo.

Las aplicaciones Hipermedia que buscamos modelar, con las características estructurales anteriormente mencionadas, pueden dividirse en dos conjuntos principales. El primer conjunto se refiere a las aplicaciones cuya estructura es relativamente fija pero los datos de la misma varían a través del tiempo como por ejemplo un catálogo de productos; el segundo gran conjunto es el de aplicaciones tipo presentaciones. Estos dos conjuntos tiene dos características en común. Primero ambos exhiben una fuerte estructura para organizar la información y segundo la información en ambos conjuntos es volátil en el sentido de que necesita ser constantemente actualizada y mantenida.

Las herramientas comerciales que se utilizan para el desarrollo de aplicaciones Hipermedia como por ejemplo: ToolBook, HiperCard y otras poseen un modelo de datos secuencial propio tipo libro, organizado en páginas apto para el desarrollo de presentaciones.

El problema que poseen estas herramientas es que están orientadas a la implementación de una aplicación Hipermedia y no al diseño, y en el caso de una aplicación que posea muchos nodos del mismo tipo no tienen la posibilidad de agruparlos y de manejarlos como tal, sino que debe implementar cada uno por separado sin la posibilidad de capturar la estructura en común. En el caso de que haya que modificar la estructura de un tipo de nodo, se debe hacer en todos y cada uno de los nodos de ese tipo.

Comercialmente no existen herramientas para el diseño de aplicaciones Hipermedia, por lo tanto esta etapa del desarrollo no se podría realizar bajo un ambiente computarizado.

Las desventajas mencionadas anteriormente hacen necesario la construcción de una herramienta que permita diseñar e implementar una aplicación Hipermedia con facilidades para mantener, actualizar y modificar una estructura de diseño la cual servirá también como documentación de la aplicación.

El mejoramiento de la calidad del diseño de Hipermedias y la reducción de su costo es un desafío importante para la industria de la información. Una manera de atacar el problema es proveer a los diseñadores de Hipermedia con ambientes de desarrollo apropiados. Los ambientes de ingeniería de Hipermedia que proveen conjuntos de herramientas integradas aumentan la eficiencia y efectividad de los diseñadores [2].

El diseño de aplicaciones Hipermedias es una tarea compleja, que difiere del diseño de otras aplicaciones en que las primeras involucra navegación, aspectos computacionales y de interface con el usuario.

Los desarrollos de Hipermedia, especialmente en escala comercial, frecuentemente involucran equipos de desarrolladores que necesitan ser coordinados y supervisados por un período de tiempo extenso. Las técnicas del desarrollo de sistemas formales y de administración de proyectos, son necesarias para asegurar que el producto de Hipermedia alcance sus objetivos y se termine dentro del tiempo y costos preestablecidos. Sin embargo, las técnicas de la industria del software tradicional deben ser modificadas para adaptarse al nuevo medio. Los proyectos de Hipermedia difieren de los proyectos de desarrollo de software tradicional en distintas características críticas. Primero, ellos pueden involucrar gente con diferentes habilidades: autores, músicos, artistas, escritores, diseñadores de contextos y también programadores. En segundo lugar, el diseño de aplicaciones Hipermedia involucra la captura y organización de la estructura de un dominio complejo y hacer este claro y accesible a los usuarios. Este proceso es un desafío que actualmente se considera más un arte que una ciencia debido al trabajo artístico y la estética de la aplicación Hipermedia final. Finalmente, la necesidad de prototipar e intensificar el testeo con los usuarios es aún más pronunciada en los desarrollos de Hipermedia que con el software tradicional porque la tolerancia a errores de los usuarios en esta clase de aplicaciones es muy baja [7].

Las nuevas áreas de aplicación necesita la flexibilidad de Hipermedia combinada con una rica variedad de datos Multimediales. Desafortunadamente la construcción de Grandes Aplicaciones de Hipermedia es difícil y además una vez que ésta ha sido construida, el mantenimiento es más complicado aún.

1.2.- Objetivo del proyecto.

El objetivo fundamental de este proyecto es la implementación de un ambiente computarizado para asistir al diseño y construcción de aplicaciones Hipermediales, que se construirá como un CASE.

Para esto es necesario contar con un modelo conceptual, que utilizaremos como fundamento para el desarrollo del CASE que cumpla con las características deseadas. Analizaremos algunos modelos conceptuales y elegiremos el que presente más utilidad para nuestro proyecto.

Las características deseadas del CASE son:

1. Debe contar con un estado de diseño y un estado de implementación.

2. Facilitar el acceso entre los diferentes pasos de la metodología que soporta el CASE.

3. Los pasos del CASE estarán interconectados y diseñados de una manera consistente que facilite el trabajo de los desarrolladores.

4. Manejo de objetos a nivel instancia.

5. Se obtendrá como salida un prototipo de la aplicación Hipermedia. El prototipo permitirá a los desarrolladores experimentar con la aplicación implementada.

6. Una facilidad que flexibiliza el diseño para su posterior implementación, es la creación de objetos a partir de una Base de Datos existente pudiendo usar sus atributos en los niveles inferiores de diseño.

1.3.- Alcance del proyecto.

El proyecto que se desarrolló involucró el estudio de numerosos temas debido a la gran cantidad de información que se manejó. Cuando hablamos de información nos referimos a técnicas de diseño, lenguajes de programación y objetos multimediales (datos de tipo sonido, video, imagen y texto). El estudio se realizó en etapas que pasamos a describir:

- Análisis de los problemas en el desarrollo de aplicaciones de Hipermedia que se resolverían posteriormente con la implementación del CASE.

- Investigar los distintos modelos y metodologías de diseño para la elección de la más apropiada, en la cual se basará el CASE, de acuerdo al tipo de aplicaciones a las que se apunta diseñar (ver 1.1).

- Análisis en profundidad de la metodología elegida para explotar sus ventajas al máximo.

- Análisis de las herramientas utilizadas para la implementación del CASE:
  - Lenguaje de programación.
  - Manejador de base de datos.
  - Lenguaje de query asociado.
  - Formatos de los objetos multimediales (video, sonido, imagen y texto).
  - Técnicas para facilitar el diseño gráfico (*drag and drop*, etc.).
  - Técnicas de navegación de documentos de Hipermedia.
  - Técnicas de generación de prototipos.

- La implementación de un prototipo que demuestre la factibilidad de la herramienta.

A continuación daremos una breve reseña de las secciones con que cuenta este informe: la sección 1 se refiere a los fundamentos básicos de Hipermedia; en la sección 2 analizamos brevemente los modelos y metodologías de diseño más difundidas; en la sección 3 nos referimos al porque de la metodología elegida; la sección 4 explica el CASE implementado al que llamamos RMCASE; la sección 5 muestra como construir una aplicación Hipermedia con RMCASE y por último en la sección 6 se describen detalle de la implementación de RMCASE.

Sección 1: Fundamentos Básicos de Hipermedia.

1.- Qué es Hipertexto ?

Podemos comenzar a definir qué es Hipertexto, contrastándolo con la idea tradicional de texto que conocemos ya sea en los textos tradicionales o aún en los procesadores de texto. En los mismos el texto se presenta en forma lineal, y la información solo puede accederse usando ese paradigma, avanzando o retrocediendo páginas (en el mejor de los casos seleccionando a que página ir, aunque solo por su número) [1].

Dos excepciones interesantes son las notas al pie de página en los libros (podemos elegir si leerlas o no) y los procesadores de textos con "*outlines*". En realidad, ya los índices en los libros o en un capítulo permiten el acceso más ágil a la información.

Un Hipertexto en cambio es esencialmente no secuencial. La información está organizada como una red, cuyos nodos contienen información (textual o gráfica al principio) y relacionados por "*links*", originados en el "interior" de los nodos, cuyo origen como muestra la Figura 1.1 sobresale (ya sea remarcando la palabra, con un gráfico, etc.). El sentido es permitirle al lector un acceso no secuencial a la información. Cuando estamos leyendo el nodo A y seleccionamos el *link* 1, "navegamos" hacia el nodo B, el cual se presentará frente al lector. Si en dicho nodo seleccionamos el *link* 2, navegamos al nodo C, etc.. Los Hipertextos poseen además un mecanismo de "*backtrack*" o retorno que permite desandar el camino recorrido.

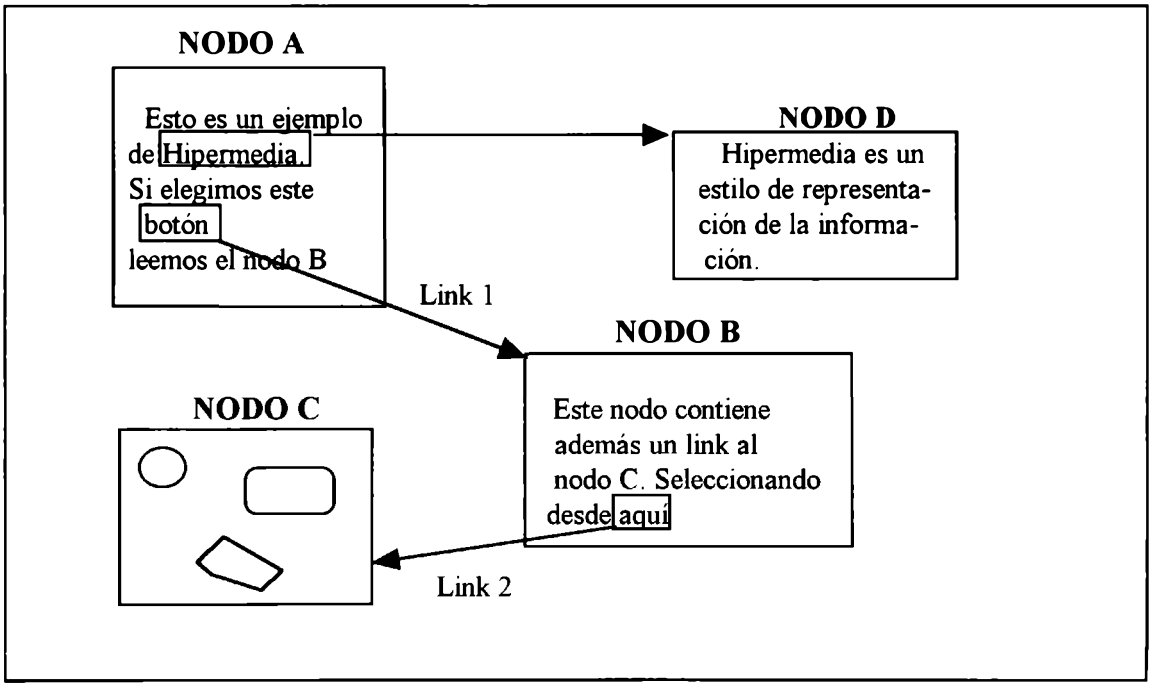


Figura 1.1: Estructura básica de un Hipertexto.

Podemos enunciar entonces la siguiente definición:

"Hipermedia es un estilo de construcción de sistemas para la creación, manipulación, presentación y representación de la información en la cual:

- la información se almacena en una colección de nodos multimedia

- los nodos se encuentran almacenados en forma implícita o explícita en una o más estructuras (habitualmente una red de nodos conectados por *links*)

- los usuarios pueden acceder a la información navegando a través de las estructuras disponibles (por ejemplo en un grafo como el de la Figura 1.1)"

En un Hipertexto las ventanas en la pantalla se asocian a uno o mas nodos en la base de información; y los orígenes de los *links* se presentan como íconos en la representación gráfica, como se muestra en la Figura 1.2.

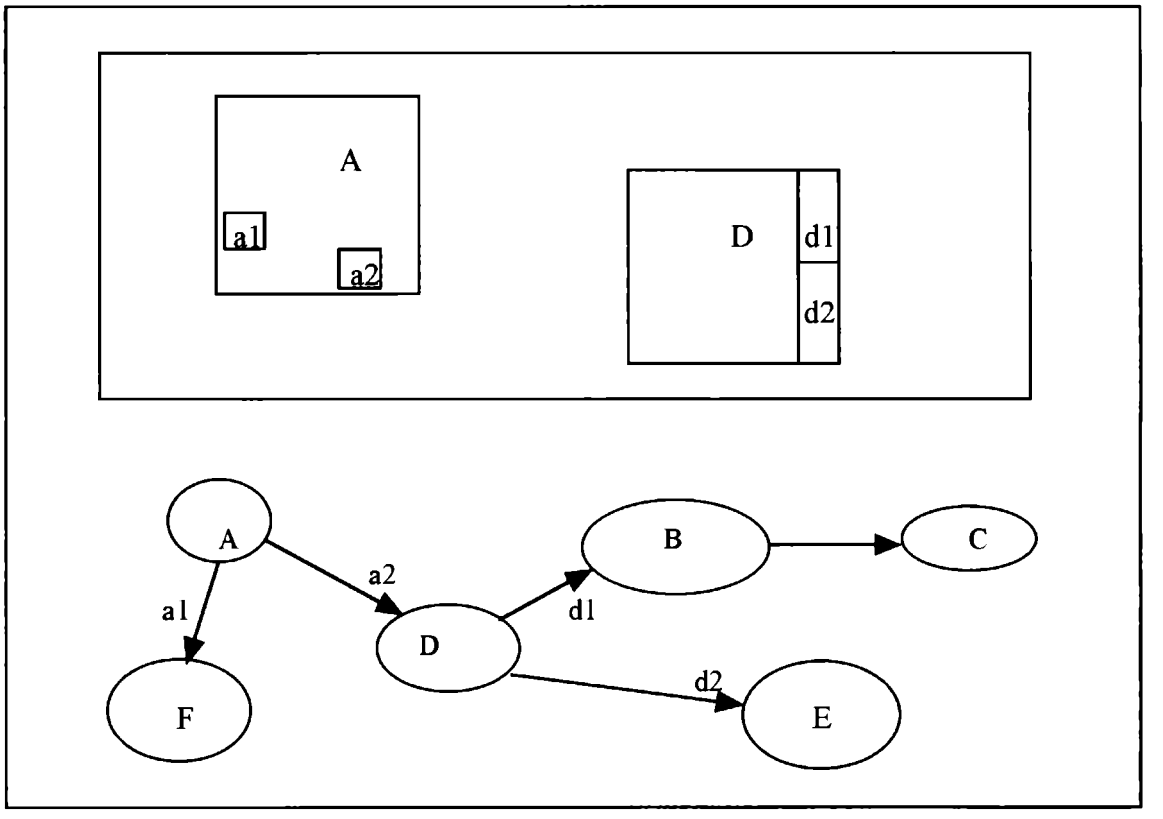


Figura 1.2: Exhibición del Hipertexto.

En la figura anterior, cada nodo del Hipertexto está asociado a una o más ventanas en la pantalla de la computadora, y los orígenes de los *links* (*anchors*) se identifican como botones, o zonas especiales, o eventualmente tipografías especiales. Seleccionar un "*anchor*" provoca la navegación hacia el nodo destino correspondiente.

Existen numerosos sistemas de Hipermedia comerciales: HyperCard [17] (en el entorno Macintosh), Guide y ToolBook [18] (en el entorno Windows), etc..

En resumen, una aplicación Hipermedia está caracterizada por la representación de la información mediante nodos multimediales y el acceso a dichos nodos en forma navegacional (recorriendo *links* que vinculan a los nodos).

Los conceptos esenciales son: nodos, *links* y *anchors*. Como herramientas adicionales de acceso a un Hipermedia deben mencionarse los *browsers*. Un *browser* es una herramienta que presenta una representación abstracta de un Hipermedia, permitiendo seleccionar el nodo deseado a partir de dicha representación, cuya información puede presentarse en distintos niveles de abstracción.

Finalmente, un sistema de Hipermedia es un sistema de software que permite crear y acceder a Hipertextos, proveyendo interfaces con ventanas, facilidades para la creación de nodos, *links* y *anchors* y para navegar a través de los nodos. También permite la construcción de bases de información de Hipermedia, proveyendo herramientas para editar nodos multimediales, indicar la presencia de un *anchor* y vincular nodos con relaciones con semánticas diversas.

Para más información del lector acerca de Hipermedia puede consultar otros trabajos de grado como por ejemplo [Una herramienta Hipermedia para la Enseñanza de la Tecnología Orientada a Objetos de Leonardi Carmen y Lopez Jorgelina].

Sección 2: Modelos y Metodologías de Diseño.

1. Generalidades.

La mayoría de las aplicaciones Hipermedias tratan con dominios complejos, con lo cual se torna difícil organizar la información en estos casos, además se necesita contar con una buena documentación para poder actualizarla y mantenerla.

Es necesario poder describir los aspectos globales, toda la estructura Hipermedia y sus semánticas navegacionales, independientemente de la implementación. Para esto se debe proveer constructores de diseño de Alto-Nivel y mecanismos de abstracción, que deben proveer una clara transición desde las descripciones del modelo del dominio abstracto al diseño de la aplicación de Hipermedia concreta, acortando el gap entre la ingeniería de software moderna y las peculiaridades del campo de Hipermedia. Un modelo de datos resuelve estos problemas para facilitar el diseño de este tipo de aplicaciones. Un modelo de datos es un conjunto de objetos lógicos usados para proveer una abstracción de una porción del "mundo real".

Los modelos de diseño proveen un lenguaje en el cual el analista de la aplicación puede especificar una aplicación dada ayudando a la comunicación entre el analista y el usuario final; entre el analista y el diseñador del sistema y entre el diseñador del sistema y el implementador, cuando ellos son diferentes personas. Esto también es usado para documentar la aplicación, proveyendo soporte para los usuarios de la aplicación y para el mantenimiento del sistema; también sirve como un lenguaje común para comparar aplicaciones (discutiendo diferencias y similitudes).

Los modelos de diseño proveen una estructura en la cual los diseñadores de aplicaciones de Hipermedia pueden desarrollar, analizar y comparar diseños y estilos retóricos de diseño, a un alto nivel de abstracción. Este análisis puede ser hecho sin tener que considerar una visualización particular (formatos de pantalla, funcionalidad de botones, etc.) ni a los contenidos detallados de las unidades de información.

Los modelos de diseño pueden ser usados por herramientas de diseño, de la misma manera como los generadores de aplicación están basados sobre lenguajes para especificar clases de aplicaciones, o como las herramientas CASE que tienen un lenguaje de especificación para describir software.

El seguimiento del modelo debe ser claro, de manera que se pueda mapear cambios en el modelo a la aplicación final y viceversa, soportando así un completo mantenimiento. Finalmente este debe ser un ambiente independiente; esto significa que debe ser posible construir un diseño abstracto y luego una implementación usando diferentes plataformas de Hipermedia.

La mayoría del material referente al diseño de Hipermedia propone una técnica formal para una cierta clase de dominio de aplicación. Una técnica formal es una fórmula que guía a los diseñadores a lo largo de una tarea especificando una serie ordenada de etapas de diseño.

Un modelo de diseño formal ofrece conceptos y notaciones convenientes para representar los resultados de cada etapa de diseño. Por ejemplo los modelos HDM (Hypertext Design Model [4]), OOHDM (Object Oriented Hypermedia Design Model, [5] y [6]) y RMD (Relationship Management Design, [7] y [8]) proveen conceptos para representar cada etapa de diseño (un modelo de datos conceptual, un modelo de navegación abstracta, un modelo de interfaz). El rango de las aplicaciones de Hipertexto es tan amplio que no hay una única técnica de diseño formal que sea apropiada para diseñar a todo tipo de aplicaciones.

Refiriéndonos a los modelos y metodologías de diseño en particular, el material encontrado sobre estos fue muy escaso dado que hay muy pocos trabajos en desarrollo y en la actualidad solo unos pocos son conocidos, todos ellos se analizarán luego en esta sección.

Los modelos y metodologías de diseño estudiados son: HDM, OOHDM y RM que fueron tenidos en cuenta para posteriormente elegir el más adecuado para utilizarlo en la construcción de la herramienta propuesta. A continuación describiremos brevemente cada uno de los modelos. Si el lector desea más detalle puede remitirse a las referencias.

2.- HDM : Un Modelo para el diseño de Aplicaciones Hipertextuales.

2.1.- Características.

Las características básicas de HDM son la representación de aplicaciones a través de distintas primitivas de diseño: *entidades tipadas* compuestas de jerarquías de componentes; diferentes *perspectivas* para cada componente; *unidades* correspondientes a pares componente-perspectiva; *cuerpos* representando el actual contenido de las unidades; *links estructurales*, uniendo las componentes o subentidades de la misma entidad; *links de aplicaciones* tipadas, interconectando componentes pertenecientes a diferentes entidades; y una específica *semántica de browsing* basada en *anchors*, como una forma para activar diferentes tipos de *links* desde dentro de una unidad.

2.2.- Introducción.

HDM pone énfasis en el modelo de *design-in-the-large*, el cual provee las primitivas que permiten al diseñador describir la aplicación de Hipertexto en una manera más clara y concisa que simplemente la construcción de un esqueleto de una aplicación en un sistema de Hipertexto dado; esta descripción debe ser lo más independiente posible del sistema.

El *design-in-the-small* (ej. llenado del contenido de los nodos y determinación de su apariencia para el lector) es también muy importante, pero esta es una actividad diferente que puede ser tratada separadamente.

Las especificaciones en el modelo de diseño HDM pueden ser traducidas automáticamente a un nivel más bajo a una especificación Nodo-Link dando una estructura de la actual implementación. Esta especificación puede estar destinada a herramientas de

Hipertexto estándar, aplicando *design-in-the-small* en los pasos de diseño. Este proceso esencialmente provee una forma de describir una aplicación de Hipertexto completa en un sistema de manera independiente.

2.3.- El Modelo de Diseño.

De acuerdo con HDM, un dominio de aplicación está compuesto de *entidades,* las cuales están compuestas por una jerarquía de *componentes*. Las entidades pertenecen a un *tipo*. Las entidades pueden conectarse a otras entidades o componentes por *links* que pueden ser *estructurales* o de *aplicación*. Los *links* estructurales reflejan la estructura jerárquica de las entidades; los *links* de aplicación conectan entidades o componentes a otras entidades o componentes para reflejar las relaciones del dominio de la aplicación. Las componentes pueden ser instanciadas por una o más *perspectivas* dentro de *unidades*. Las unidades proveen un contexto de referencia para la información contenida en sus *bodies* (Cuerpos). Un *esquema* HDM es entonces un conjunto de definiciones de tipos de entidades y de *links* de aplicaciones. Una *instancia* de un *esquema* HDM es un conjunto particular de entidades y *links* definidos de acuerdo a un esquema dado.

Una vez que una instancia del esquema ha sido definida, podría operar vía la especificación de una *semántica de browsing* particular (ver Sección 2: 2.4.7) . HDM no provee las primitivas que especifican tal semántica de *browsing*, pero hay una semántica de *browsing* por defecto que es compatible con la mayoría de los sistemas de Hipertexto.

2.4.- Primitivas de HDM.

2.4.1.- Entidades y Componentes.

Se refiere a una *entidad* como un objeto concreto conceptual del mundo real en el dominio que es relevante para la aplicación. Generalmente, una entidad denotará algo complejo, cuya estructura interna puede ser descompuesta. Una entidad en su integridad puede ser representada a través de distintas piezas de información individuales más pequeñas que son llamadas *componentes.*

Una componente es una pieza de información que describe una parte de una entidad. Las componentes están agrupadas dentro de una jerarquía arbitraria, dependiendo de la aplicación, para formar la correspondiente entidad.

Las entidades en HDM cumplen el rol de *"objetos compuestos"* (con la relación de inclusión), con la restricción de que todas las componentes de una entidad son homogéneas con respecto a sus perspectivas. HDM usa jerarquías como las estructuras de las entidades porque este tipo de estructuras es muy utilizado en contextos específicos. La jerarquía es muy útil para ayudar al usuario cuando navega en hiperdocumentos.

2.4.2.- Unidades y Perspectivas.

Las componentes describen piezas de información. Para enriquecer la mayoría de los ambientes de Hipertexto la información puede ser representada de diferentes formas. Una abstracción que puede ser hecha en estas situaciones es permitir al diseñador referirse (y pensar) acerca del concepto que está siendo representado por una componente, independientemente de la forma que es descripto. En otras palabras el concepto puede ser pensado como tener distintas "perspectivas".

HDM ayuda a esta abstracción teniendo *perspectivas* para las componentes. El término perspectiva significa la apariencia de una pieza de información. La descripción de una componente de acuerdo a una perspectiva es llamada *unidad*, la cual tiene un *body* conteniendo la información de sí misma. HDM no aclara nada acerca de *bodies*.

Una componente puede tener una o más unidades (correspondiendo a Perspectivas).

2.4.3.- Tipos de Entidades.

Una abstracción común encontrada en la mayoría de los modelos es la noción de *tipo*. Un *tipo de entidad* agrupa entidades que tienen propiedades comunes, principalmente usando el mismo conjunto de perspectivas, estando descompuestas en componentes de acuerdo al mismo criterio, y que se puedan relacionar a entidades de otros tipos.

En HDM el conjunto de perspectivas asociadas a un tipo de entidad es indicativo solamente de posibles perspectivas para sus entidades, por esto es necesario que tenga la noción de una *perspectiva default*. El sentido que se quiere dar a la perspectiva por defecto es que todas la entidades de un tipo tengan al menos esta perspectiva; además este concepto es importante en el proceso de mapeo en estructuras Nodo-Link.

2.4.4.- Links y Tipos de Links.

La mayor ventaja del modelo de Hipertexto es la organización de una información base en un estilo no lineal. Esto significa, que las piezas de información pueden ser relacionadas a otras a través de los *links* asociados a ellas.

HDM diferencia tres clases de *links*: *estructural, aplicación y perspectiva*. Los *links* estructurales conectan componentes que pertenecen a la misma jerarquía. Las entidades proveen un contexto de navegación, tal que siguiendo los *links* estructurales permiten al lector examinar diferentes partes de la misma "cosa".

Los *links* de aplicación dan forma a las relaciones semánticas en el dominio. Es decir, ellos representan algunas relaciones que el diseñador consideró importantes, en el sentido que estas relaciones evocan algunas asociaciones entre conceptos útiles para el usuario del Hiperdocumento. Proveyendo un *link* de aplicación, el diseñador estará haciendo "natural" al usuario, acceder a otros datos relacionados con la información que está siendo leída en ese momento.

Dado que las componentes están para una abstracción de distintas perspectivas del mismo sujeto, los *links* de perspectiva permiten al lector moverse entre las diferentes perspectivas de la misma componente.

Para ser consistente con la noción de entidad y tipos de entidad, HDM definió tipos de *links* de aplicación como un conjunto de instancias de *links* de una misma relación.

2.4.5.- Derivaciones de Links.

Al tener HDM las jerarquías como conceptos primitivos considera que, a nivel de diseño, el diseñador necesite especificar solo un conjunto mínimo de *links* estructurales, para lo cual es necesario definir la estructura de una entidad, desde la cual cualquier otro *link* estructural implícito puede ser derivado si se desea.

Las *reglas de derivación* pueden existir para los *links* de aplicación también, si se consideran a estos como relaciones entre entidades (componentes). Las propiedades de estas relaciones, tales como simetría y transitividad, cuando se presentan, pueden ser usadas para derivar a otros *links*.

Hasta aquí HDM mismo no incluye un lenguaje para especificar las reglas de derivación (para los *links* de aplicación), pero es posible tener tal lenguaje en un formalismo fuera de HDM. Dado tal lenguaje, el uso de HDM provee una gran cantidad de concisiones para la especificación del modelo, el diseñador necesita proveer una cantidad mucho menor de *links* que la cantidad de *links* que se presentaban tanto a nivel conceptual como a nivel concreto.

2.4.6.- Esquema e Instancia de Esquema.

Una clase de aplicación de Hipertexto en un dominio dado puede ser caracterizada vía la noción de *esquema*, definidos como un conjunto de definiciones de tipo de entidad y tipo de *link*. Un esquema caracteriza clases de aplicación porque no especifica entidades y *links* reales, pero bastantes propiedades generales de entidades potenciales, y *links* potenciales entre ellas. Las definiciones de tipo de entidad en el esquema incluye reglas de derivación para *links* estructurales; los tipos de *links* de aplicación permitirán la inclusión de reglas de derivación cuando haya un lenguaje para que tales reglas sean definidas.

Una aplicación particular en un dominio dado es especificada instanciando un esquema, por ej., dando entidades y *links* como instancias de los tipos definidos en el esquema. La noción de esquema e instancia de esquema permite, algunas veces, la reutilización del *mismo* esquema para diferentes aplicaciones en el mismo dominio. Estas aplicaciones compartirán la misma estructura *global*, pero diferirán en las instancias particulares que estén actualmente presente. Un nivel de reutilización más amplio se puede obtener cuando las nuevas aplicaciones preserven la estructura local (los *links* estructurales dentro de las entidades y los *links* de aplicación entre ellas) en una *instancia* del esquema, pero redefine los *bodies* (cuerpos) de las unidades.

Una instancia del esquema caracteriza los aspectos "estáticos" de una aplicación específica. Para especificar la conducta dinámica (a tiempo de ejecución) de una aplicación, se debe agregar una semántica de *browsing* particular. Esto agrega otra característica de rehuso,

en la cual se mantiene la misma especificación estática, pero se usa una semántica de *browsing* diferente, generando una aplicación diferente a tiempo de ejecución.

2.4.7.- Semánticas de Browsing.

La visualización y las propiedades dinámicas de los Hipertextos son definidas por sus *semánticas de browsing*. En HDM las semánticas de *browsing* determinarán además tres clases de alternativas de *diseño* para el *design-in-the-large*:

1) Cuáles son los objetos, en términos de HDM, que puede percibir el usuario: entidades, componentes, o unidades? Cómo son percibidos estos objetos?

2) Cuáles son los *links* percibidos entre los objetos; en términos de HDM, cuáles *links*, y cómo, están visibles los *links* (navegables)?

3) Cuál es la conducta cuando los *links* son activados; en términos de HDM, que sucede cuando uno activa un *link* de uno a varios (1 a N)? Cuál es el estado de los objetos fuente y destino?

HDM no especifica una semántica de *browsing* para los Hipertextos especificados con él. Trabaja proveyendo primitivas para la definición de tales semánticas de *browsing* en un estado preliminar, pero semánticas de *browsing* simples pueden ser especificadas con formalismos tales como Redes de Petri. Otros aspectos de las actividad de *browsing*, tales como la apariencia de pantallas e íconos, y otros aspectos del *design-in-the-small* no son tratados.

En esta clase de semántica de *browsing*, asumimos que los objetos abstractos (entidades y componentes) no están directamente visibles, solo las unidades (correspondientes a la noción de nodo en Hipertexto) pueden ser percibidas por los lectores como objetos concretos. En consecuencia, los lectores pueden ver *links* solamente entre unidades y así, al final, las conexiones deben ser establecidas entre unidades. También se asume que solo un nodo está activo a la vez y solo un *link* puede ser atravesado a la vez. Se llaman *links* concretos a las conexiones entre unidades, que se distinguen de los *links* abstractos los cuales son definidos entre Entidades y Componentes. Es necesario entonces especificar que *links* concretos pueden ser obtenidos a partir de los *links* abstractos.

En Hipertexto las conexiones entre piezas de información están generalmente visualizados a través de "*anchors*" (o botones). Los *anchors* son áreas bien identificables en la pantalla que señalan la existencia de una conexión, y pueden ser seleccionados por los lectores para atravesar un *link*. Los tipos de *anchors* proveen un mecanismo para presentar grupos de tipos de *links* juntos, también permite renombrarlos u ocultarlos como se desee. Asignando un conjunto de tipo de *link* a cada tipo de *anchor* el diseñador puede controlar la visibilidad de los *links*.

Activando un *anchor* HDM se pueden activar distintos nodos destino posibles, aunque el resultado de esta activación debe ser definido. Claramente, esto es parte de una semántica de *browsing* particular que está siento usada, la cual se torna parcialmente dependiente del sistema que se está usando para implementar el Hipertexto. Si se soportan múltiples ventanas activas, por ejemplo, una posibilidad puede ser simplemente mostrar todos los destinos, cada uno en una ventana separada. Aún esta solución no es aceptable o no es posible en varios sistemas.

Una aproximación alternativa a tratar con esta situación es introducir la noción de *chooser*. Un *chooser* es un mecanismo asociado con un *anchor* de un único origen y múltiples destinos, que permite la selección de uno de los múltiples destinos del *anchor*. Los *choosers* son estructuras que pueden ser implementadas automáticamente desde la especificación del Hipertexto, usando cualquier mecanismo disponible presente en el ambiente de implementación (ej. menúes).

2.5.- Ejemplo.

Este ejemplo describe una aplicación acerca de las Bandas Nacionales. El objetivo es tener una forma organizada de acceder y referirse a un gran conjunto de datos de diferentes naturalezas (pero interrelacionados). Esta organización maneja Músicos, los cuales tocan en diferentes Bandas, pueden tocar más de un Instrumento y las Bandas editan Discografía, esta información está capturada en el esquema descripto en la Figura 2.1, donde los tipos de entidades y tipos de *link* de aplicación han sido especificados (como algunas relaciones en este ejemplo son simétricas se representan con líneas bidireccionales, los nombres indican las relaciones y sus inversas).

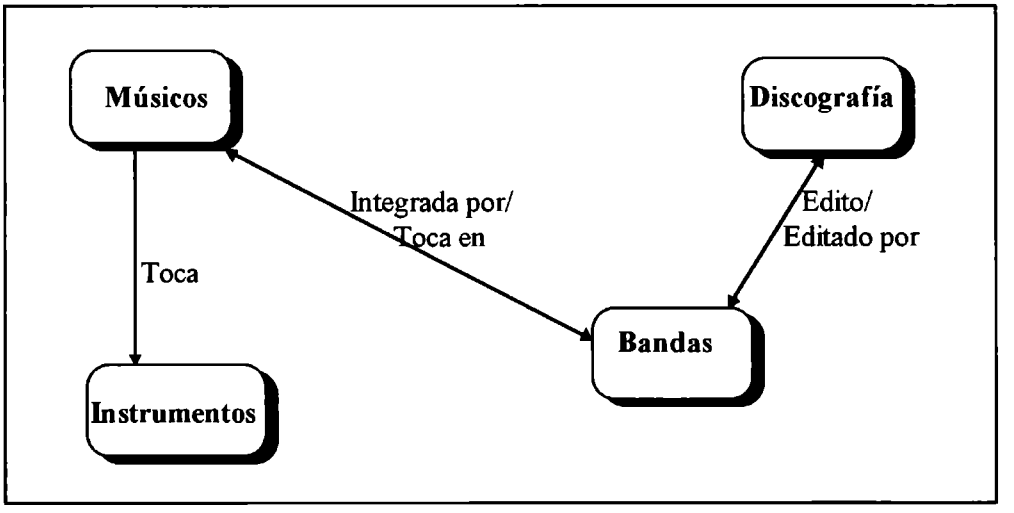


Figura 2.1: Ejemplo en HDM del Esquema de tipos de entidades y tipos de links de aplicación para Bandas Nacionales.

En la Figura 2.2, cada tipo de Entidad tiene un conjunto de Perspectivas asociadas. Nosotros no enumeramos todas aquí, pero damos un par de ejemplos (la marcada con "*" es la perspectiva por defecto).

- Los Músicos tienen Biografía* y Estructura.
- Las Bandas tienen Historia* y Estructura.

Las áreas elipsoidales denotan instancias de entidad las cuales tienen una estructura interna más detallada. La entidad Músicos está compuesta por cuatro componentes: *fecha de nacimiento, nombre, domicilio y foto*; la entidad Bandas tiene como componentes a: *año de inicio, nombre, foto y mejor video.*

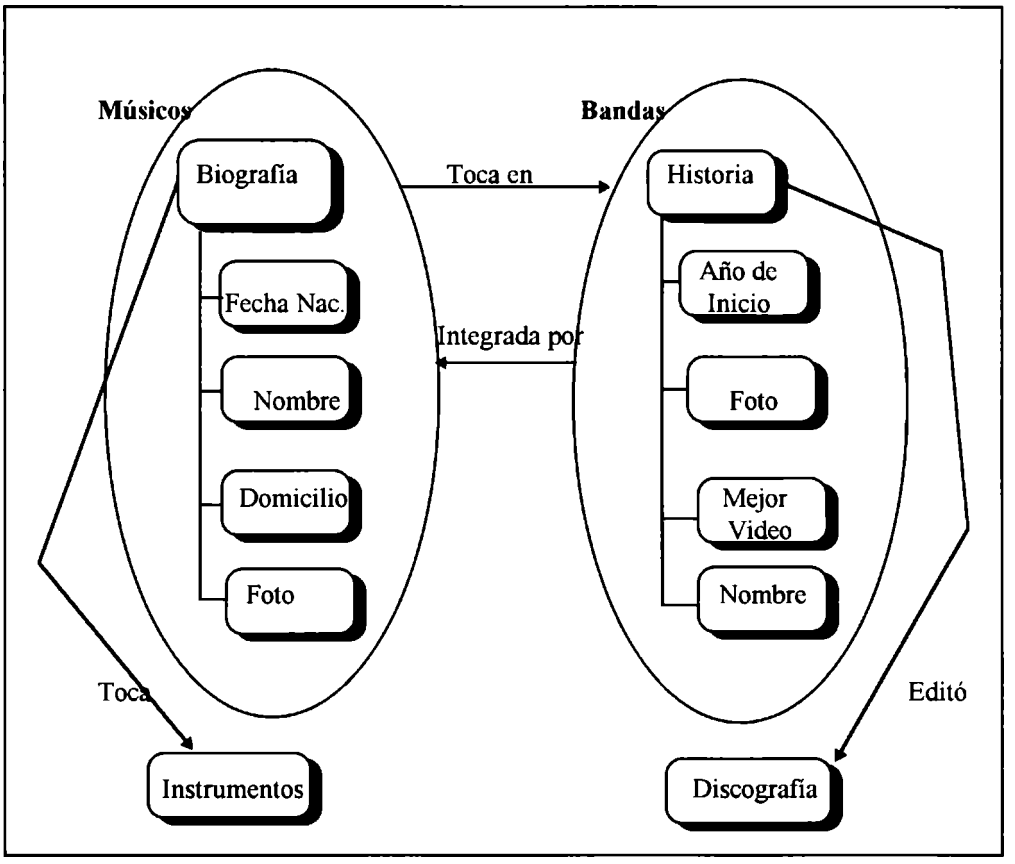


Figura 2.2: Ejemplo en HDM del Esquema de la aplicación Bandas Nacionales con detalle del diseño de las entidades Músicos y Bandas.

En la Figura 2.3 se instancia el esquema, la instancia de la entidad Músicos corresponde a Charly García y la de Bandas a Serú Girán; todos sus componentes están instanciados. Charly García "Toca" la Guitarra (que es una instancia de entidad de tipo Instrumentos no detallada). También se representan los *links* de aplicación entre Charly García y Serú Girán como "Toca en" y su inversa (o el *link* de vuelta) "Integrada por". Con respecto a Serú Girán "Editó" como trabajo discográfico a Serú Girán 92 (que es una instancia del tipo de entidad Discografía).

Con respecto a las perspectivas, el usuario puede ver a la perspectiva "Estructura" de la instancia de Músicos "Charly García" donde se mostrarán sus componentes o puede pasar rápidamente a la otra perspectiva "Biografía" que mostrará en este caso los datos de las componentes pero en forma más general.

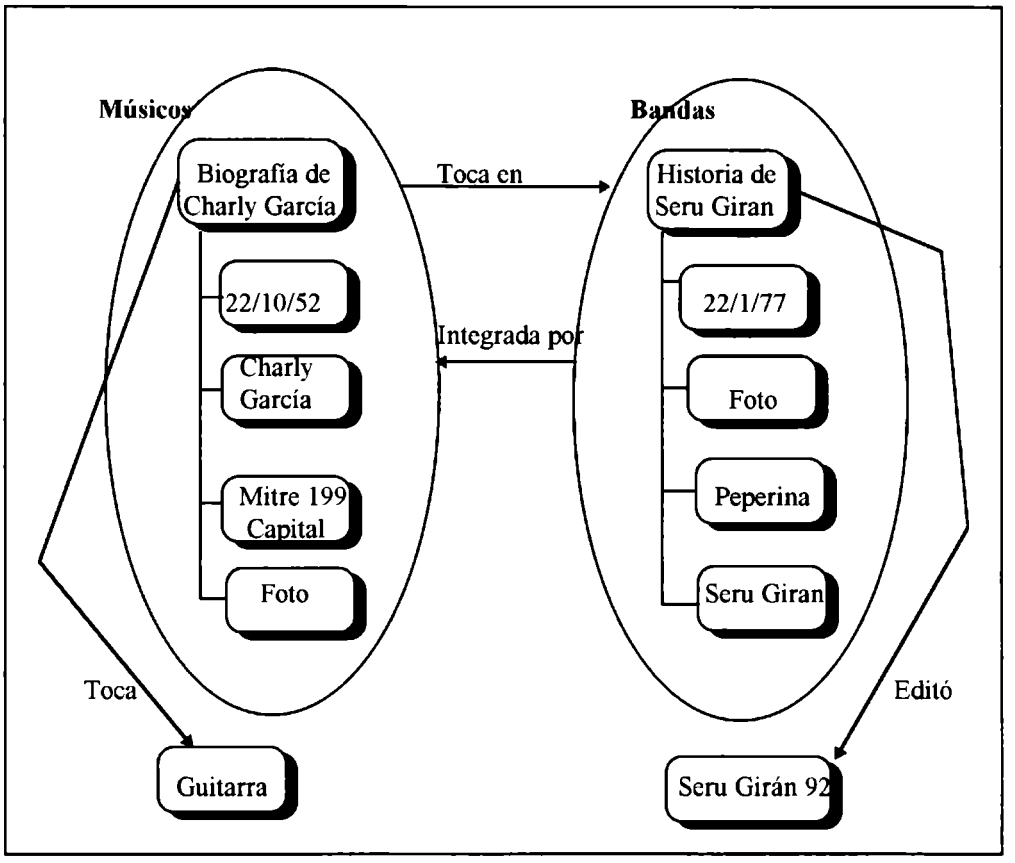


Figura 2.3: Ejemplo en HDM de una instancia del Esquema de la Figura 2.2.

3.- OOHDM: Modelo y Metodología de Diseño de Hipermedia Orientado a Objetos.

3.1.- Introducción.

El OOHDM está orientado a la construcción de grandes aplicaciones de Hipermedia. Permite la descripción del dominio de la aplicación usando constructores de alto nivel y mecanismos de composición. El modelo realiza una evolución desde los modelos de dominio abstracto a descripciones de aplicaciones de Hipermedia y luego a aplicaciones concretas, las cuales no necesariamente han sido implementadas en un sistema orientado a objetos. Siendo este modelo orientado a objetos puede ser usado como una estructura potente para rehusar componentes de Hipermedia.

Los modelos de datos orientados a objetos pueden solucionar los problemas mencionados anteriormente (ver Sección 2: 2.1), no sólo como artefactos de implementación sino también como herramientas de modelización durante todo el ciclo de vida del desarrollo de una aplicación Hipermedia.

El modelo provee constructores para la descripción de las relaciones conceptuales entre los objetos del dominio y también para la definición de su estructura y comportamiento. Combinando las clases con las descripciones basadas en instancias, el modelo preserva una potente abstracción sin perder flexibilidad.

3.2.- Proceso de Desarrollo.

Los autores de OOHDM no solo proponen un modelo sino también tienen en cuenta todo el proceso de desarrollo de aplicaciones de Hipermedia. Provee constructores de alto nivel para la especificación de aplicaciones de Hipermedia ayudando así a la comunicación entre el analista, el usuario final y el programador, lo cual puede ser usado para documentar la Aplicación proveyendo así soporte para mantenimiento. El modelo sostiene un proceso en cuatro pasos basados en estilos de desarrollo iterativos, incremental y prototípicos en lugar del tradicional.

3.2.1- Paso 1: Análisis del Dominio (Modelización).

En este paso un modelo conceptual del dominio de la aplicación es construido usando principios de la modelización orientada a objetos que son conocidos [19], aumentado con algunas primitivas como perspectivas de atributos (atributos multi-valuados, similar a las perspectivas HDM, ver Sección 2: 2.4.2). Las clases conceptuales pueden ser construidas usando agregación y las jerarquías de especialización-generalización. En este paso no se le da importancia a los tipos de usuarios y tareas, solo a las semánticas para el dominio de la aplicación. El resultado de este paso es un esquema conceptual, subsistemas, clases y relaciones.

3.2.2- Paso 2: Diseño Navegacional.

Aquí se describe la estructura navegacional de una aplicación en términos de contextos navegacionales, los cuales son inducidos a clases navegacionales tales como nodos, *links*, índices y tours guiados. Las clases y contextos navegacionales tienen en cuenta los tipos de usuarios deseados y sus tareas.

Los nodos en OOHDM representan ventanas lógicas (o vistas) sobre clases conceptuales definidas durante el análisis del dominio (Paso 1). Diferentes modelos navegacionales pueden ser construidos para el mismo esquema conceptual para expresar diferentes vistas sobre el mismo dominio.

Los *links* son derivados de las relaciones conceptuales definidas en el primer paso. Las semánticas navegacionales están definidas en función de nodos y *links*, pudiendo modelar movimientos en el espacio de navegación (ej. el subconjunto de nodos con el cual los usuarios pueden interactuar en un momento dado.) independientemente del modelo conceptual. El modelo navegacional puede evolucionar independientemente del modelo conceptual, simplificando el mantenimiento.

3.2.3- Paso 3: Diseño de Interface Abstracta.

El modelo de interface es construido definiendo objetos perceptibles (ej. una imagen, un mapa de una ciudad , etc.) en términos de clases de interfaces. Las clases de interfaces están definidas como agregaciones de clases primitivas (tales como campos texto y botones) y recursivamente como clases de interfaces. Mapea objetos de interface a objetos navegacionales, provee un contexto perceptible. El comportamiento de la interface es declarado especificando como manejar los eventos externos y los generados por el usuario y como toma lugar la comunicación entre los objetos de interface y de navegación.

Después que este paso ha sido realizado se cuenta con la suficiente información para implementar la aplicación usando un sistema de Hipermedia particular. En otras palabras, el diseño de interface abstracta provee el mapeo entre lo abstracto, el diseño de aplicación de alto nivel y la aplicación concreta en un ambiente de hardware y software dado.

3.2.4- Paso 4: Implementación.

La implementación mapea objetos de interface a objetos de implementación y puede incluir arquitecturas elaboradas (ej. cliente - servidor), en la cual las aplicaciones son clientes para un servidor de base de datos compartida que contenga los objetos conceptuales. Al ser un modelo orientado a objetos provee un modelo de seguimiento natural permitiendo así un mejor entendimiento y mantenimiento de la aplicación de Hipermedia. Esto puede también ser la base para la teoría de reusabilidad de componentes de Hipermedia.

3.3- Ejemplo.

Como se especifica en el paso 1, primero se describe el dominio de aplicación utilizando las primitivas de OOHDM (definición de clases, herencia, relación parte-de, etc.) para crear la hiperbase que contenga la información organizada. Continuamos utilizando el ejemplo Bandas Nacionales para mostrar como se define un esquema en OOHDM.

En la Figura 2.4 se muestra la clase Músico que es una agregación de Foto Músico y la clase Banda es una agregación de Foto y Video. "Toca en" es un ejemplo de una relación entre dos clases, en este caso Músico y Banda.

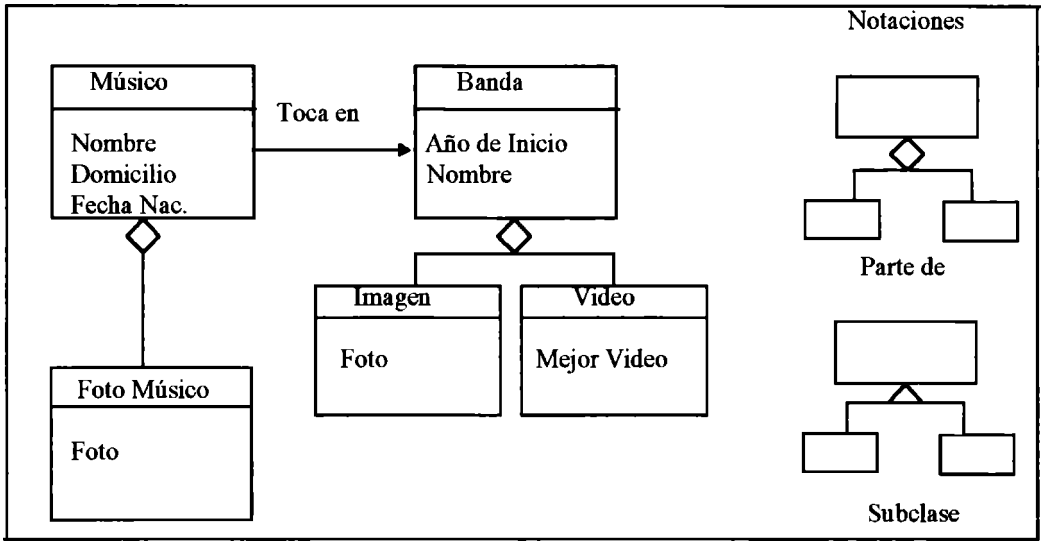


Figura 2.4: Parte del esquema de la aplicación Bandas Nacionales.

Ilustraremos la noción de clases navegacionales a partir de la clase Músico.

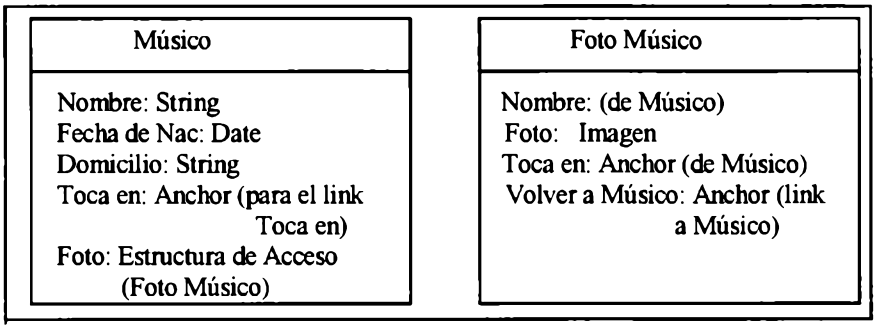


Figura 2.5: Clases Navegacionales.

Los nodos de la clase navegacional Músico tienen sus atributos inducidos desde los atributos y relaciones de la clase conceptual Músico: Nombre, Fecha de Nacimiento y Domicilio son copiados directamente; "Toca en" se deriva de la relación correspondiente y Foto se deduce de la información estructural (Músico es una agregación de Foto Músico). El atributo Foto contiene la estructura de acceso al nodo que describe la foto de Músico.

El nodo de la clase navegacional Foto Músicos ha sido definido en forma similar, donde el atributo Foto ha sido inducido de la clase conceptual Foto Músico. Sin embargo el atributo Nombre y "Toca en" son heredados del nodo de la clase navegacional Músico (y no de la clase conceptual). Se incluye un atributo adicional de tipo *link* que permite retornar al nodo Músico.

4.- RM: Una Metodología para el diseño de Aplicaciones Hipermedias Estructuradas.

4.1.- Introducción.

Los desarrolladores y diseñadores de aplicaciones Hipermedia necesitan un conjunto de convenciones para la representación de los diseños de Hipermedia. La metodología de "Relationship Management" (RM) es una propuesta para resolver los problemas del diseño y construcción de aplicaciones Hipermedia. El nombre "Manejo de Relaciones" desde nuestro punto de vista de Hipermedia es como un vehículo para el manejo de relaciones entre objetos de información.

Las clases de aplicaciones para las cuales RM es más adecuado exhiben una estructura regular del dominio de interés, por ejemplo, hay clases de objetos, relaciones definibles entre estas clases y múltiples instancias de objetos dentro de cada clase. Varias aplicaciones Hipermedias satisfacen estos requerimientos. Como varias aplicaciones Hipermedias (por ejemplo catálogos de productos, presentaciones, etc.) poseen la característica de que sus datos requieren frecuente actualización (volatilidad de los datos), algunos medios para automatizar el desarrollo inicial y el subsiguiente proceso de actualización son necesarios. Esto trata con la idea de que los datos y relaciones importantes puedan ser mantenidos usando una base de datos. Agregando una nueva instancia de objeto, por ejemplo, debe entonces simplemente ser una cuestión de agregar un nuevo registro a la base de datos. Todas las relaciones importantes que involucren el nuevo objeto deben automáticamente ser construidas por un sistema que interactúe con la base de datos.

Estas dos características, estructura y la necesidad de actualizar datos, son la razón de ser para el sistema de manejo de base de datos, y la metodología RM confía plenamente en las técnicas de modelización de datos durante la fase de diseño y en los sistemas de administración de base de datos en las fases de construcción y mantenimiento.

Volatibilidad de Información

Estructura	**Bajo**	**Alto**
	Alto Mediana utilidad	Alta utilidad
	Bajo No es útil	Baja utilidad

Figura 2.6: Utilidad de RM.

La Figura 2.6 ilustra la utilidad de RM en el diseño y desarrollo de aplicaciones Hipermedia. Los dos ejes representan la Estructura y Volatibilidad de la información. Las aplicaciones mencionadas anteriormente tienen alta estructura y alta volatibilidad de información y la metodología RM es apropiada. Como contrapartida, un trabajo artístico puede no tener una estructura legible y generalmente se mantiene sin cambios a través del tiempo, en

este caso RM no es apropiado. Las aplicaciones que son altamente estructuradas y se mantienen sin cambios durante un largo período de tiempo pueden hacer uso de la metodología RM durante las fases de diseño y construcción, pero, no requiere mucho mantenimiento así que el problema de actualización es relativamente sin importancia y la ventaja obtenida por el uso de RM no es pronunciada. Finalmente, las aplicaciones que tienen estructuras regulares o dinámicas y alta volatilidad pueden obtener poca ventaja del uso de RM. En este caso, sin embargo, juzgamos que es posible que algunas partes del dominio puedan ser estructuradas y que el problema de alta volatilidad pueda al menos ser parcialmente manejado.

4.2.- Modelo de Datos.

El modelo de datos de RM (RMD) es la pieza fundamental de la metodología RM. RMD provee un lenguaje para describir los objetos de información y los mecanismos de navegación en las aplicaciones Hipermedia.

La Figura 2.7 muestra las primitivas de modelización de RMD. En la parte superior de la figura están las *primitivas de dominio*, las cuales modelan información acerca del dominio de la aplicación. Los tipos de entidad y sus atributos representan objetos abstractos o físicos, tales como personas o cuentas bancarias. Las relaciones asociativas, las cuales pueden ser uno a uno (1:1) ó uno a varios (1:N), describen asociaciones entre diferentes tipos de entidad. En RM las relaciones varios a varios (N:N) se dividen en dos relaciones 1:N. Esta no es la manera de modelar las relaciones N:N en la modelización de base de datos [20] pero es la adecuada desde el punto de vista navegacional.

Las relaciones varios a varios (N:N) pueden existir, como en la modelización de base de datos, pero en la modelización de Hipermedias ésta se divide en dos relaciones (1:N) porque lo único que interesa es modelar la navegación.

Ya que las entidades pueden contener un gran número de atributos de diferentes naturalezas (por ejemplo: información de salarios, datos biográficos, fotografías, etc.), puede ser ineficiente o no deseado mostrar todos los atributos de una instancia de entidad a la vez. Por esto, los atributos son agrupados en *slices*. Por ejemplo una entidad Músico con atributos nombre, edad, foto y biografía, puede tener un *slice* General, que contenga nombre, edad y foto y un *slice* Biografía con el nombre y la biografía. Entonces, cada instancia de una entidad Músico será mostrada por dos *slices*, y si la aplicación soporta esto, un usuario puede elegir cual de los dos *slices* ver.

La navegación está soportada en RMD por las seis primitivas de acceso mostradas al final de la Figura 2.7. Los *links* unidireccionales y bidireccionales son usados para especificar el acceso entre los *slices* de una entidad. Las estructuras de acceso más importantes soportadas por RMD son los índices, los tours guiados y los groupings. Un índice actúa como una lista de ítems de información proveyendo acceso directo a cada ítem de la misma. Un tour guiado implementa un camino lineal a través de un conjunto de ítems permitiendo al usuario moverse hacia atrás o hacia adelante en el camino. El mecanismo de *grouping* sirve como un punto de acceso a otras partes del documento de Hipermedia. Por ejemplo, la pantalla inicial de la mayoría de las aplicaciones contiene un menú o un conjunto de botones que proveen acceso a diferentes funciones o clases de información.

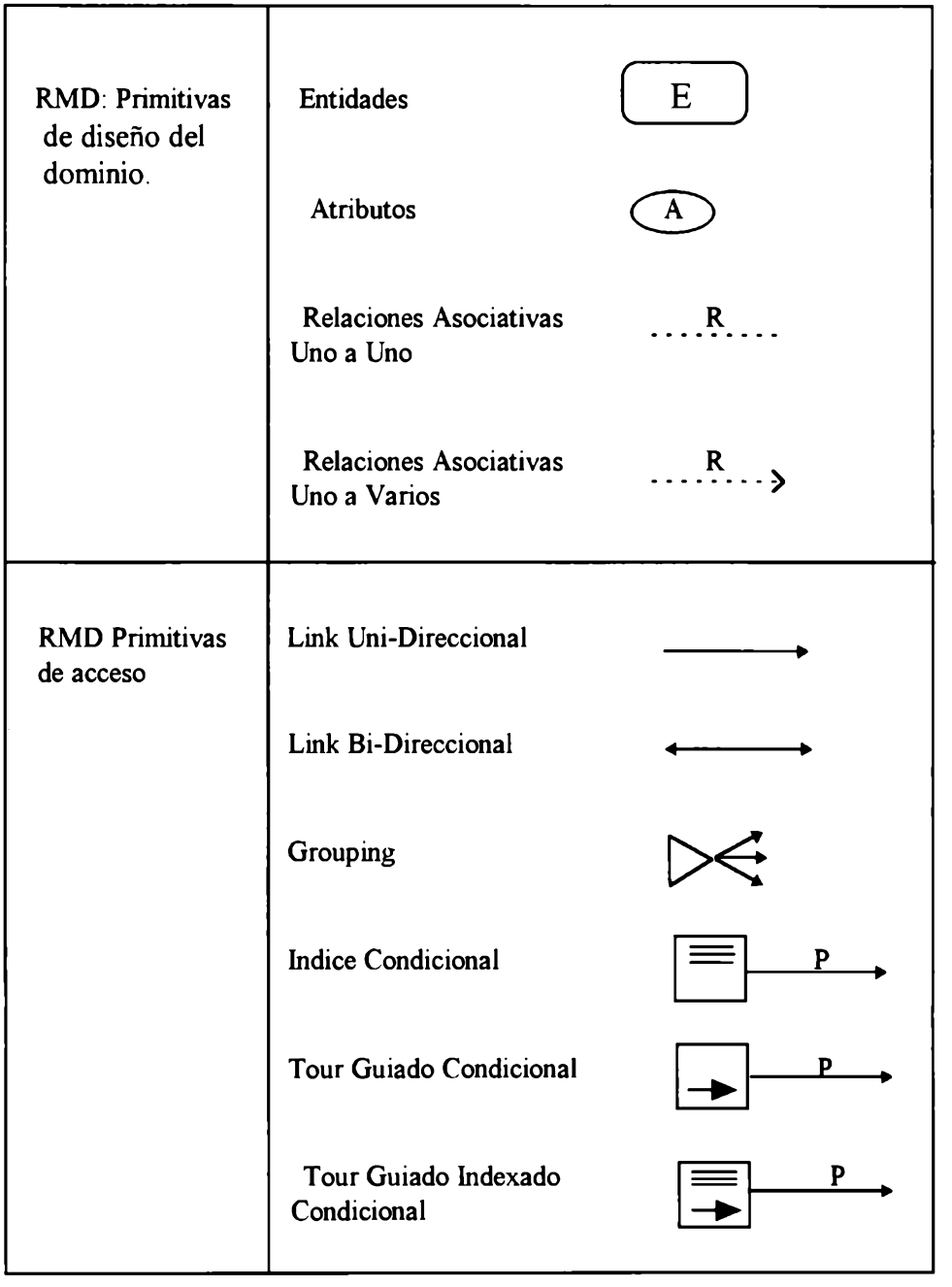


Figura 2.7: Primitivas de RMD.

Las condiciones o predicados lógicos cualifican índices y tours guiados determinando que instancias de una entidad son accesibles desde el constructor. Por ejemplo, la Figura 2.8-a muestra un tour guiado condicional de todos los Instrumentos electrónicos. El predicado Instrumentos(Clase="electrónico") indica que solo las instancias de la entidad Instrumentos cuya clase es electrónico, participa en el tour guiado. La parte derecha de la figura muestra una instancia de tal tour guiado. La Figura 2.8-b es un ejemplo de un índice condicional. Aquí, el acceso es permitido vía un constructor tipo índice. Hay también *links* de retorno desde cada nodo participante en el índice, como se muestra en la parte derecha de la figura. Por último el tour guiado indexado condicional combina índices y tour guiados para proveer una estructura de acceso más completa.

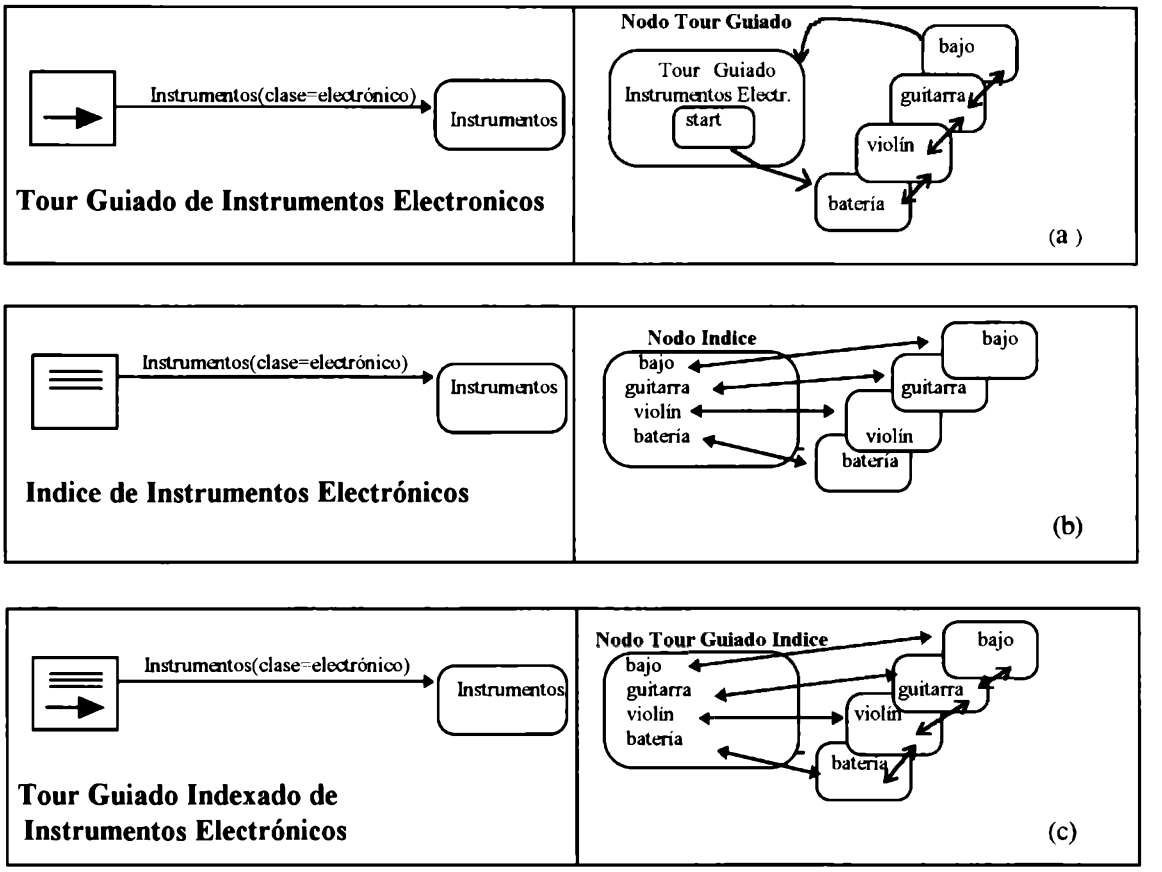


Figura 2.8: Ejemplos de constructores de acceso de RMD.

En la Figura 2.9 se muestra un ejemplo de un diagrama RMD con un *grouping*, cuatro entidades y estructuras de acceso.

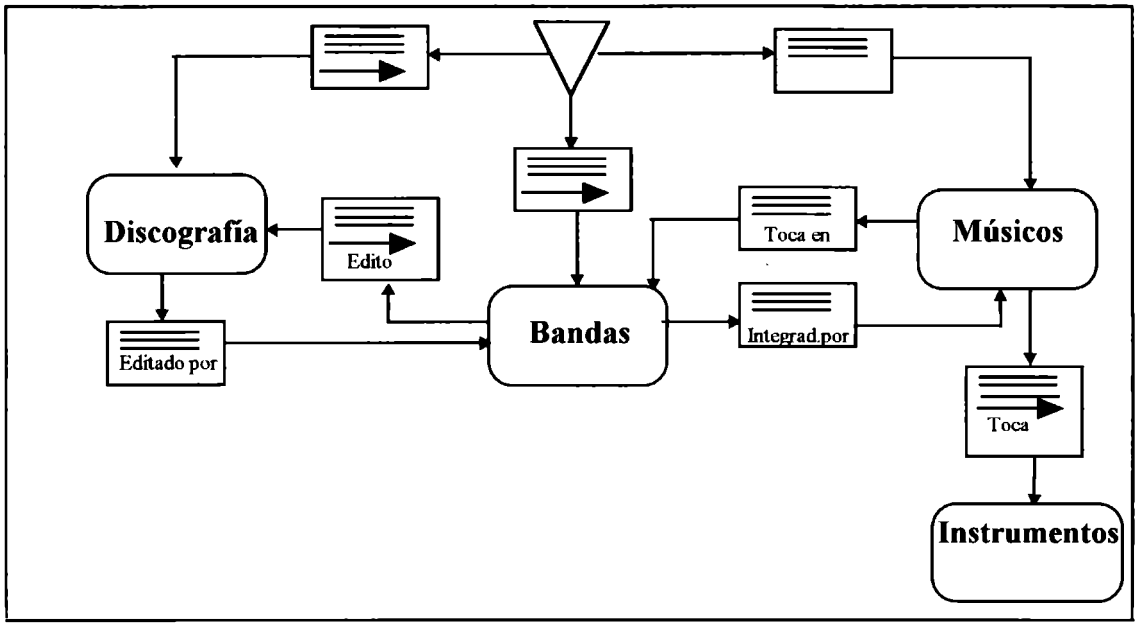


Figura 2.9: Diagrama RMD para la Aplicación Bandas Nacionales.

4.3.- Metodología de Diseño.

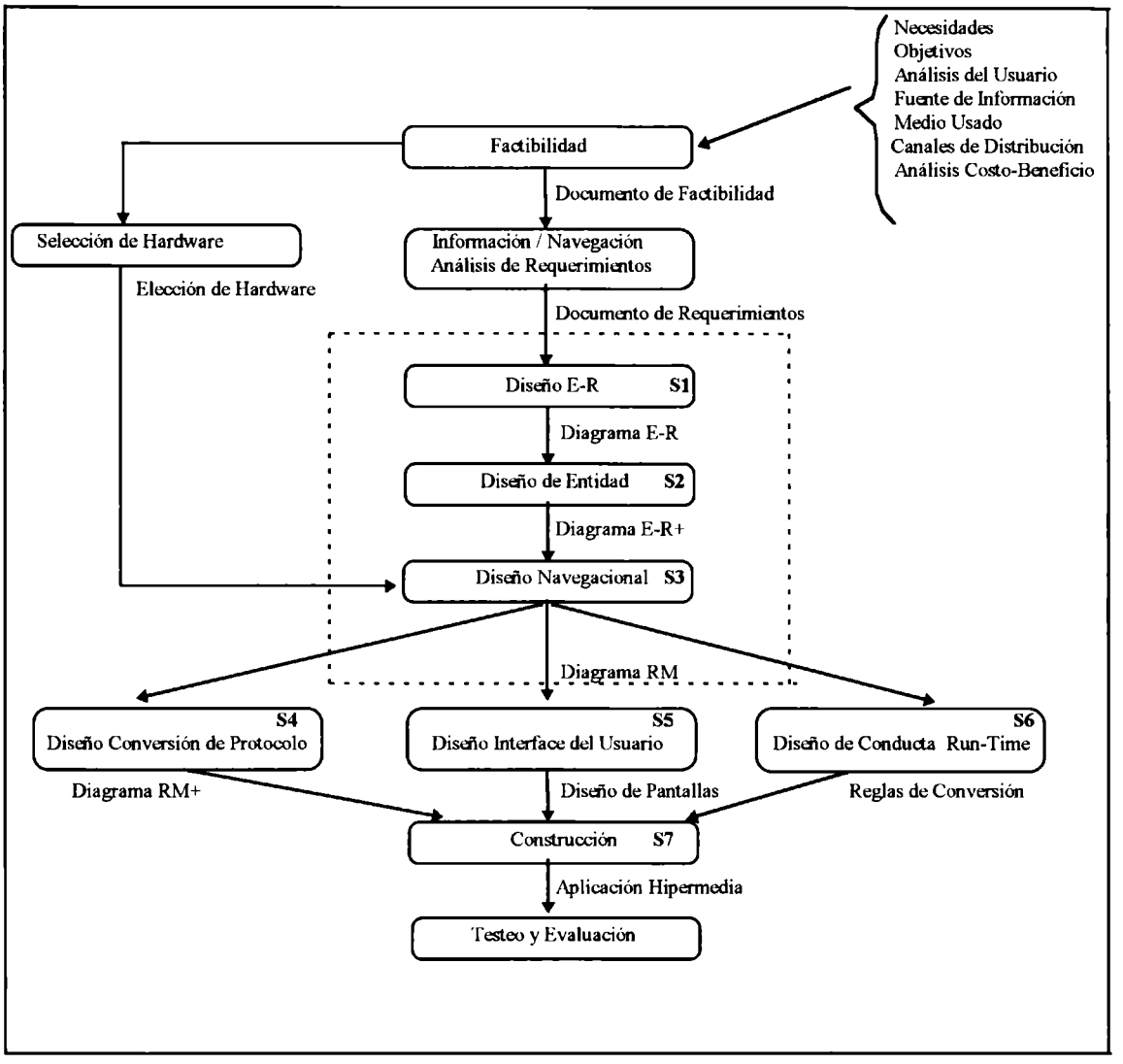


Figura 2.10: Metodología de Diseño de RM.

Las flechas que conectan las distintas etapas están nombradas con los objetos para ser usados como entrada del próximo paso. Los loops feedbacks no son mostrados. La Metodología RM se enfoca más en los pasos contenidos en el área punteada.

La Figura 2.10 muestra gráficamente la metodología RM dentro del contexto del ciclo del desarrollo del software completo. RM pone énfasis en las faces de diseño, desarrollo y construcción, dejando de lado la factibilidad y el análisis de requerimientos. La metodología consta de siete pasos que proveen una guía para el proceso de diseño. Fundamentalmente apunta al diseño de mecanismos de acceso, lo cual es llevado a cabo a través de los tres primeros pasos de la metodología.

Los nombres de las flechas son los distintos "diseños intermedios" generados a través del uso de la metodología. Ellos sirven para monitorear y documentar el sistema y para facilitar las tareas de mantenimiento. No se muestra en la figura los numerosos *loops feedback* que interconectan los distintos pasos. Un ambiente RM soportado por computadora puede proveer

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

un fuerte soporte de estos mecanismos *feedback* facilitando las actualizaciones de estos "diseños intermedios".

El modelo de datos provee una fuerte prescripción para la elección de los nodos y *links* en la aplicación Hipermedia. Sin embargo, hay muchos diseños que deben ser decididos independientemente por el diseñador. Además de explicar la metodología de diseño, también se provee una guía de diseño para cada paso.

4.3.1.- Paso 1: Diseño de Entidades y Relaciones.

El primer paso de diseño es para representar el dominio de información de la aplicación, a través de un diagrama de Entidades y Relaciones (E-R) [21][22][23]. La representación E-R es familiar para los analistas de sistemas, es una buena documentación, y puede modelar dependencias de información en numerosos dominios de aplicación. Esta etapa del proceso de diseño representa un estudio de las entidades y relaciones relevantes del dominio de la aplicación. Estas entidades y relaciones forman la base de la aplicación Hipermedia y muchos de ellos se verán en la aplicación final como nodos y *links* en la red Hipermedia. En muchas situaciones el diagrama E-R puede ya estar disponible, por ejemplo si la aplicación es una interface Hipermedia para una base de datos existente. En este caso este puede ser rehusado directamente en este paso.

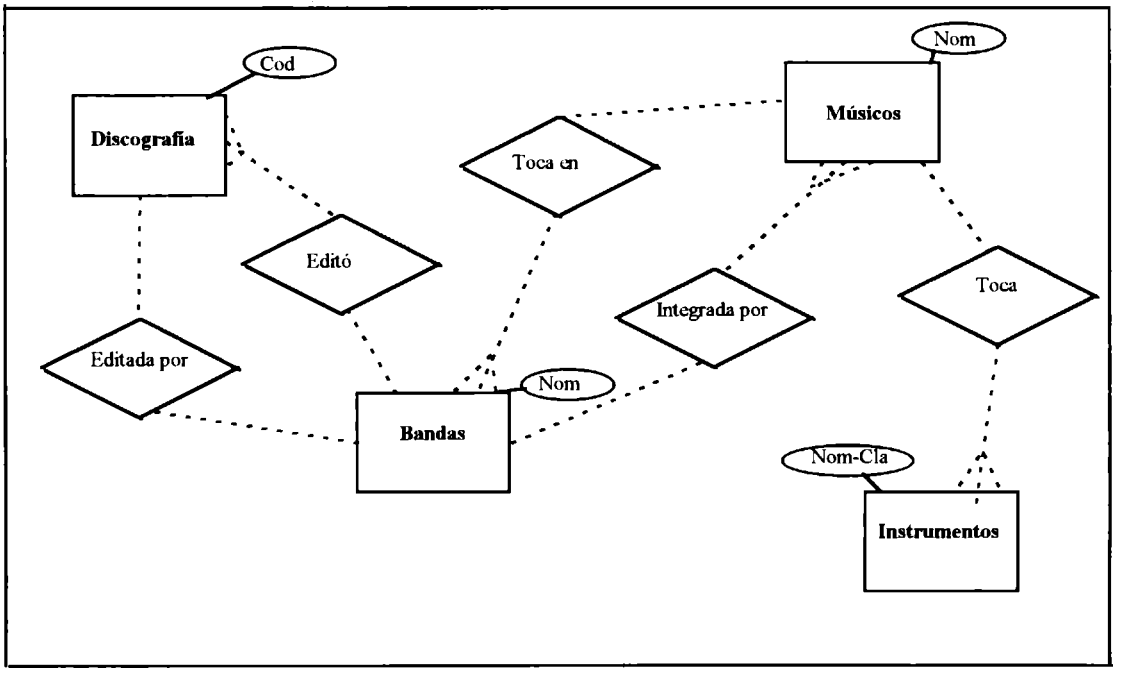


Figura 2.11: Diagrama E-R de la Aplicación Bandas Nacionales.

Las Entidades aparecen dentro de rectángulos y las relaciones con líneas punteadas. Los nombres de las relaciones aparecen dentro de rombos. En los Elipses se especifican las claves de la Entidad.

La Figura 2.11 es un diagrama E-R. Las entidades son mostradas como rectángulos y las relaciones como líneas punteadas. En RMD las relaciones que aparecen en el diagrama E-R

son llamadas *relaciones asociativas*, ya que ellas representan asociaciones entre instancias de entidad. La notación gráfica también muestra la aridad de cada relación. Cuando un arco se abre en tres líneas, la aridad de ese lado de la relación es N (varias). Las posibles aridades de las relaciones, de acuerdo con la estructura de E-R son uno a uno y uno a varios.

Sólo los atributos claves de la entidad se muestran en la figura. Estos aparecen subrayados, dentro de elipses. Todos los atributos son por defecto atómicos y univaluados.

- **Guía para el diseño de E-R.**

El objetivo en el diseño de aplicaciones Hipermedia es trazar los *links* entre los objetos explícitos ya que estos son los caminos principales vía los cuales los usuarios recorren los ítems individuales de información. Un análisis del dominio usando E-R ayuda a identificar relaciones importantes a través de las cuales la navegación puede ser soportada. Contrariamente, si un camino navegacional entre entidades es un requerimiento para la aplicación, una correspondiente relación asociativa tiene que ser incluida en el diseño E-R. Así, un diseñador no sólo observa el dominio sino que también le impone estructura.

Una vez que el E-R para la aplicación ha sido dibujado, es necesario identificar las entidades y relaciones que participarán en la aplicación Hipermedia final. Esta selección tendrá lugar en los dos pasos siguientes, Diseño de Entidad y Diseño Navegacional.

4.3.2.- Paso 2: Diseño de Entidad.

Este paso involucra la determinación de cómo la información en las entidades elegidas será presentada a los usuarios, y cómo ellos pueden acceder a esta. Este paso es único para aplicaciones Hipermedia, involucra explotar una entidad en *slices* o partes importantes y organizarlos en una red de Hipertexto. En una forma más simple, toda la información en una entidad puede ser mostrada dentro de una ventana con barras de *scroll*. Además esta idea es simple para los desarrolladores, pero esto puede ser inadecuado para los usuarios, quienes pueden desorientarse cuando escrolean grandes ventanas. Alternativamente la información puede ser dividida en unidades importantes que puedan ser presentadas como separadas pero todas interrelacionadas.

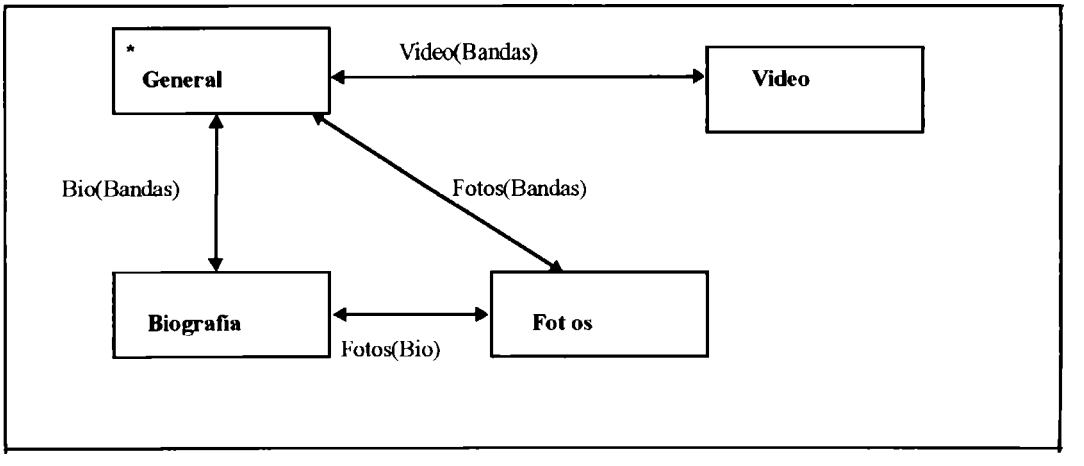


Figura 2.12: Diagrama de Entidad para Bandas.

La organización de entidades en *slices* se llama etapa de Diseño de Entidad, y de esto resulta un diagrama de entidad como el de la Figura 2.12. Cada *slice* agrupa uno o más atributos de la entidad. Cada entidad tiene un *slice* distinguido, su cabecera, el cuál es usado como *anchor* para los *links* que llegan a la entidad. En el diagrama, la cabeza de la entidad es marcada con un asterisco.

El diagrama de entidad también modela la navegación entre *slices* a través de *links* unidireccionales y bidireccionales. Los nombres en los *links* bidireccionales identifican ambas direcciones. Estos *links* representan conexiones entre *slices*. Sin embargo, estas conexiones difieren de las relaciones asociativas que aparecen en el diagrama E-R en que conectan piezas de información de una instancia de entidad, mientras que las relaciones asociativas interconectan diferentes instancias de entidad las cuales, en la mayoría de los casos, pertenecerán a diferentes clases de entidad. Gráficamente, las conexiones entre *slices* se muestran como líneas llenas para diferenciarlas de las relaciones asociativas las cuales son representadas como líneas punteadas.

Hay una razón importante para diferenciar estas dos clases de conexiones. Cuando un usuario atraviesa un *link* asociativo, la información cambia de contexto. Sin embargo, cuando un *link* de *slice* es atravesado, el contexto de la información se mantiene dentro de la misma entidad.

La salida de la etapa de Diseño de Entidad enriquece el diagrama E-R, llamado E-R+, el cual es obtenido explotando cada entidad en sus *slices*.

- **Guía del Diseño de Entidad.**

Hay cuatro consideraciones principales: (1) dividir una entidad en *slices*, (2) elegir el *slice* cabecera de la entidad, (3) interconectar los distintos *slices* y (4) poner nombre a los *links*. Es importante recordar que cada *slice* representará un todo para el usuario del sistema. Así, los *slices* deben agrupar sólo ítems de información relacionada, pero no debe contener demasiada información. Lo ideal sería no utilizar barras de *scroll*, ya que los usuarios tienden a perder el foco cuando las usan.

La elección del *slice* cabecera y la decisión sobre las interconexiones entre *slices* requieren un análisis del dominio. Generalmente se elige como *slice* cabecera al más representativo de los *slices*. Los *links* representan la necesidad de conectar las unidades más generales a otras más específicas y se proveen *links* bidireccionales para poder navegar hacia atrás desde ellos. Finalmente se deben elegir nombres de *links* apropiados y en lo posible que ayuden a identificar el *slice* destino.

4.3.3.- Paso 3: Diseño Navegacional.

En este paso se diseñan los caminos que permitirán la navegación del Hipertexto. Se analiza cada relación asociativa que aparece en el diagrama E-R+ (E-R + Diseño de Entidad). Si de acuerdo al análisis de requerimientos, una relación asociativa debe estar accesible en la navegación, ésta es remplazada por una o más estructuras de accesos de RMD. Como la metodología RM sirve para dominios que son actualizados frecuentemente, todos los caminos navegacionales son especificados en términos genéricos. Esto significa que no hay *links ad-hoc*

entre instancias de entidades; además los *links* son especificados haciendo referencia a las propiedades de entidades y relaciones. Los tres elementos navegacionales de RMD que hacen esto posible son los índices condicionales, los tours guiados condicionales y los tours guiados indexados condicionales.

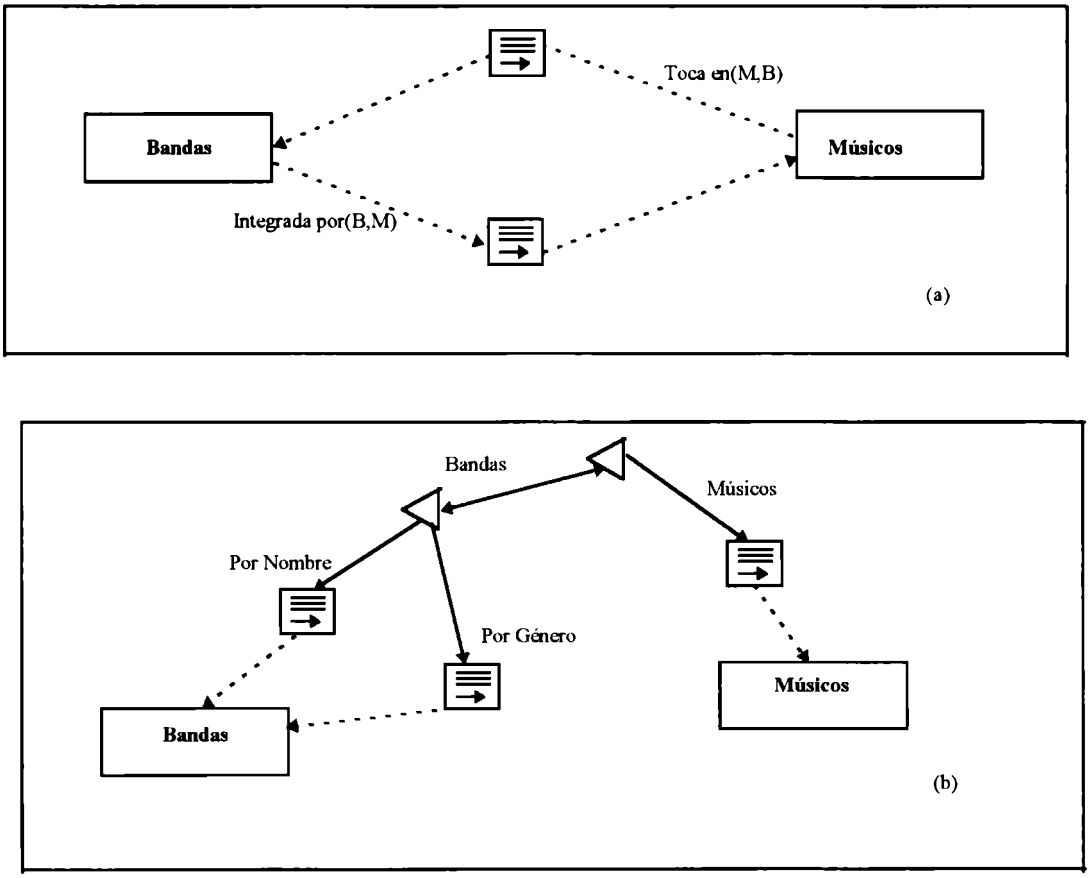


Figura 2.13: Ejemplo de un Diagrama RMD.

Primero, se diseña la navegación entre entidades, la cual está basada en las relaciones asociativas. El nombre de la relación es usado como una condición en estas estructuras de acceso para indicar que instancias de la entidad van a estar interconectadas.

En segundo lugar se diseñan las estructuras de acceso de alto nivel. El *grouping* (agrupamiento) adicional como se muestra en la parte superior de la Figura 2.13-b, provee una estructura jerárquica para acceder al sistema, similar a un menú estándar. Por defecto, las estructuras de acceso permiten entradas a una entidad a través de su *slice* cabecera. Sin embargo, los diseñadores pueden especificar un punto de entrada diferente. Esto puede ser indicado en la estructura de acceso señalando el nombre del *slice* destino.

Al finalizar el diseño navegacional, el diagrama E-R+ se transforma en un diagrama RMD (modelo de datos de RM) como el que se muestra en la Figura 2.9, la cual describe todas las estructuras de acceso que se pueden implementar en el sistema.

- **Guía del Diseño Navegacional.**

El diseño puede ser realizado *bottom-up*, comenzando primero por cada entidad y luego en las estructuras de accesos más generales, o el proceso inverso puede ser adoptado obteniendo así un diseño *top-down*. El RMD aconseja *bottom-up* porque se concentra en la información requerida necesaria para cada componente de información.

Durante la etapa de diseño, los diseñadores tienen que identificar (a) las componentes de información y las relaciones que serán accedidas, (b) qué *groupings* serán mostrados y (c) que estructuras de acceso usar en cada caso. Las decisiones en (a) y (b) se deben basar en los requerimientos del sistema obtenidos durante la etapa de análisis de requerimientos. Los *groupings* proveen puntos de entrada jerárquico; se recomienda reducir la profundidad de accesos jerárquicos para evitar la desorientación del usuario. Las relaciones uno a uno son implementadas con *links* bidireccionales. Para las relaciones uno a varios, la elección es más compleja: entre un tour guiado, un índice y un tour guiado indexado.

Se prefiere un tour guiado a un índice cuando el número de instancias de la entidad participante es relativamente pequeño (menos que diez) y cuando no hay claves en los índices que puedan ayudar a los usuarios. Por otro lado cuando hay un gran número de instancias y hay una clave importante, un índice sería lo mejor. El tour guiado indexado se usa generalmente cuando hay una clave para el índice y se desea hacer alguna navegación local.

4.3.4.- Diseño y construcción de la interface con el usuario.

A continuación describiremos brevemente, ya que no tratan directamente con el diseño, los cuatro pasos restantes de la metodología.

Como las plataformas de desarrollo de Hipermedia existente no soportan directamente las estructuras navegacionales RMD, estas necesitan ser implementadas usando los bloques de construcción disponibles en cada ambiente. Tal implementación es especificada en el Paso 4, *diseño del protocolo de conversión,* vía un conjunto de reglas de conversión. Paso 5, *diseño de la interface con el usuario,* involucra el diseño de la disposición de la pantalla para cada objeto que aparece en el diagrama nodo *link* obtenido en los pasos previos. Esto incluye ubicación de botones, la apariencia de nodos e índices y la ubicación de las ayudas navegacionales. Las decisiones acerca de cómo atravesar los *links*, historial, *backtracking* y mecanismos navegacional serán implementadas durante el Paso 6, *diseño del comportamiento a tiempo de ejecución.* Finalmente el Paso 7 consiste en la *construcción y testeo* como en los proyectos de ingeniería de software tradicional. En las aplicaciones Hipermedias se debe tener un especial cuidado en testear todos los caminos navegacionales.

Sección 3: Por qué se elige la metodología RM ?

1.- Diferencias entre RM y HDM.

Los modelos de base de datos son abstracciones útiles en aplicaciones de base de datos, pero las características de Hipermedia, en particular los aspectos de navegación, requieren nuevos modelos de diseño. El modelo de datos de RM está basado en HDM que es apropiado para describir la estructura del dominio de aplicación. HDM describe esquemas de representación pero provee poca información sobre los procedimientos para usar estas representaciones en el proceso de diseño, por ejemplo, este no describe una metodología de desarrollo.

El modelo de datos de RM (RMD) extiende y mejora el modelo de datos de HDM con el agregado de nuevos constructores, se retienen los conceptos de entidades, *grouping*, *links* estructurales y asociativos. Sin embargo las perspectivas de HDM no son soportadas porque no se les encontró utilidad en la práctica; por esto no se hace diferencia entre componentes y unidades. Los constructores de HDM como los índices y los tour guiados son reemplazados por otros más potentes como *índices condicionales*, *tours guiados condicionales* y *tours guiados indexados condicionales* que es una combinación de los anteriores; cada constructor es calificado con condiciones o predicados lógicos que determinan las instancias de componentes que serán accesibles.

Sobre el modelo de datos RMD a diferencia de HDM, se crea una metodología que incluye una secuencia recomendada de pasos, formalismos adicionales de estructuras de acceso, incrementa el énfasis en representaciones gráficas y un procedimiento detallado paso por paso de diseño y desarrollo de Hipermedia.

2.- Diferencias Principales entre OOHDM y RM.

OOHDM provee los principios de modelización orientada a objetos enriqueciéndolo con algunas primitivas como las perspectivas y RM provee un modelo de datos basado en el modelo de Entidades y Relaciones (con primitivas de HDM) y una metodología detallada paso por paso. Ambos modelos de datos se basan en HDM.

El particular énfasis de OOHDM está puesto en el diseño de interface abstracta. OOHDM usa los objetos como constructores de modelización por todo el proceso de diseño, favoreciendo la construcción de nuevas aplicaciones de Hipermedia por el sistemático rehuso de componentes existentes.

El modelo de RM puede ya estar especificado si se utiliza una base de datos existente que ya ha sido modelada utilizando el diseño de Entidades y Relaciones.

Con respecto a la metodología de diseño RM detalla los pasos con mayor profundidad y brinda mayor claridad ya que provee guías para facilitar la tarea a los diseñadores.

Los diseñadores que utilicen OOHDM deben tener amplios conocimientos del diseño orientado a objetos; en cambio RM al estar basado en el modelo de Entidades y Relaciones es más simple de manejar y por lo tanto puede ser utilizado por diseñadores no expertos. Para desarrollar aplicaciones simples, utilizando RM sería suficiente, no obstante diseñando con OOHDM no aprovecharíamos todo su potencial porque este modelo está orientado a diseñar aplicaciones más complejas.

3.- Ventajas de RM.

- La metodología RM propone la etapa de diseño en tres pasos. La salida que se obtiene de cada uno de estos pasos es un "diseño intermedio" bien definido (E-R, E-R+ y RMD. Ver Figura 2.10) que sirve para monitorear y documentar el sistema y para facilitar las tareas de mantenimiento. RM también permite moverse hacia atrás y hacia adelante entre los distintos pasos del diseño facilitando las actualizaciones de los "diseños intermedios". Esta facilidad que permite moverse entre los pasos de diseño se denomina *feedback loop*.
- El primer paso de diseño de RM que se utiliza para representar el dominio de información de la aplicación, se basa en el modelo de Entidades y Relaciones; esto es una ventaja ya que esta representación es familiar para los analistas de sistemas y programadores y además puede modelar información en numerosos dominios de aplicación. También es una ventaja que en muchas situaciones el modelo de Entidades y Relaciones puede ya estar disponible, por ejemplo si la aplicación es una interface Hipermedia para una base de datos existente. En este caso este puede ser rehusado directamente. Esta ventaja en particular es muy importante para el desarrollo de nuestro proyecto, ya que uno de nuestros objetivos es proveer la facilidad de trabajar con bases de datos externas ya existentes y estas pueden estar diseñadas con el modelo de Entidades y Relaciones.
- Provee claramente los mecanismos de acceso en forma completa y detallada.
- Es adecuado para diseñar el tipo de aplicaciones al que nosotros apuntamos en este trabajo (ver Introducción).
- Promueve una construcción incremental del diseño de la aplicación Hipermedia.

Por lo descripto anteriormente la metodología RM es la que encontramos más adecuada para utilizarla como base en el desarrollo de nuestro trabajo: RMCASE, pues al igual que OOHDM propone una metodología de diseño similar pero trabaja sobre un modelo de datos más simple.

Sección 4: RMCASE. Generalidades.

1.- Introducción.

Presentamos en esta sección nuestro trabajo, el cual consiste en el desarrollo de una herramienta CASE, basada en la metodología RM (ver Sección 2), para el diseño y construcción de aplicaciones Hipermedia estructuradas, que llamaremos RMCASE.

Las clases de aplicaciones para las cuales RMCASE es más adecuado, exhiben una estructura regular del dominio de interés, por ejemplo, hay entidades, relaciones definibles entre estas entidades y múltiples instancias de entidades dentro de cada una. Es adecuado también para aplicaciones Hipermedias que requieren frecuente actualización de los datos, porque provee medios para automatizar el desarrollo inicial y el subsiguiente proceso de actualización. Los datos y relaciones importantes se mantienen usando una base de datos. Agregar una nueva instancia a una entidad, por ejemplo, es simplemente una cuestión de incorporar un nuevo registro a la base de datos.

El modelo de datos que utilizamos, que provee un lenguaje para la descripción de objetos de información y los mecanismos de navegación en aplicaciones Hipermedia, es similar al modelo de datos de RM con algunas diferencias que especificaremos luego en esta sección.

La metodología utilizada está basada en la metodología de RM fundamentalmente en los pasos de diseño (los tres primeros), en los pasos subsiguientes se agregaron modificaciones para agilizar la utilización de RMCASE, que aclararemos luego en el punto 2.2 de esta sección.

RMCASE permite construir con facilidad prototipos de manera inmediata pudiendo testear y evaluar lo realizado durante el diseño, con la posibilidad de volver al diseño para modificarlo y construir nuevamente el prototipo, y así aproximándose cada vez más a la aplicación Hipermedia final deseada.

Nosotros denominamos pasos a las distintas etapas que el diseñador de la aplicación Hipermedia debe atravesar para obtener el prototipo, lo cual no implica que estos deban realizarse en el orden preestablecido.

2.- Proyecto RMCASE: Case para el diseño de aplicaciones Hipermedias.

A continuación en el punto 2.1 definiremos el modelo de datos, en 2.2 la metodología utilizada, en 2.3 describiremos las características que incluyen el proceso de diseño soportado y en 2.4 los contextos de trabajo que se implementan en RMCASE.

2.1.- Modelo de Datos.

Como señalamos anteriormente nuestro modelo de datos está basado íntegramente en el modelo de datos de RM. Al respecto introducimos algunas diferencias por motivos de notación gráfica en la implementación y eficiencia en la estructuras de acceso, pero los conceptos se mantienen.

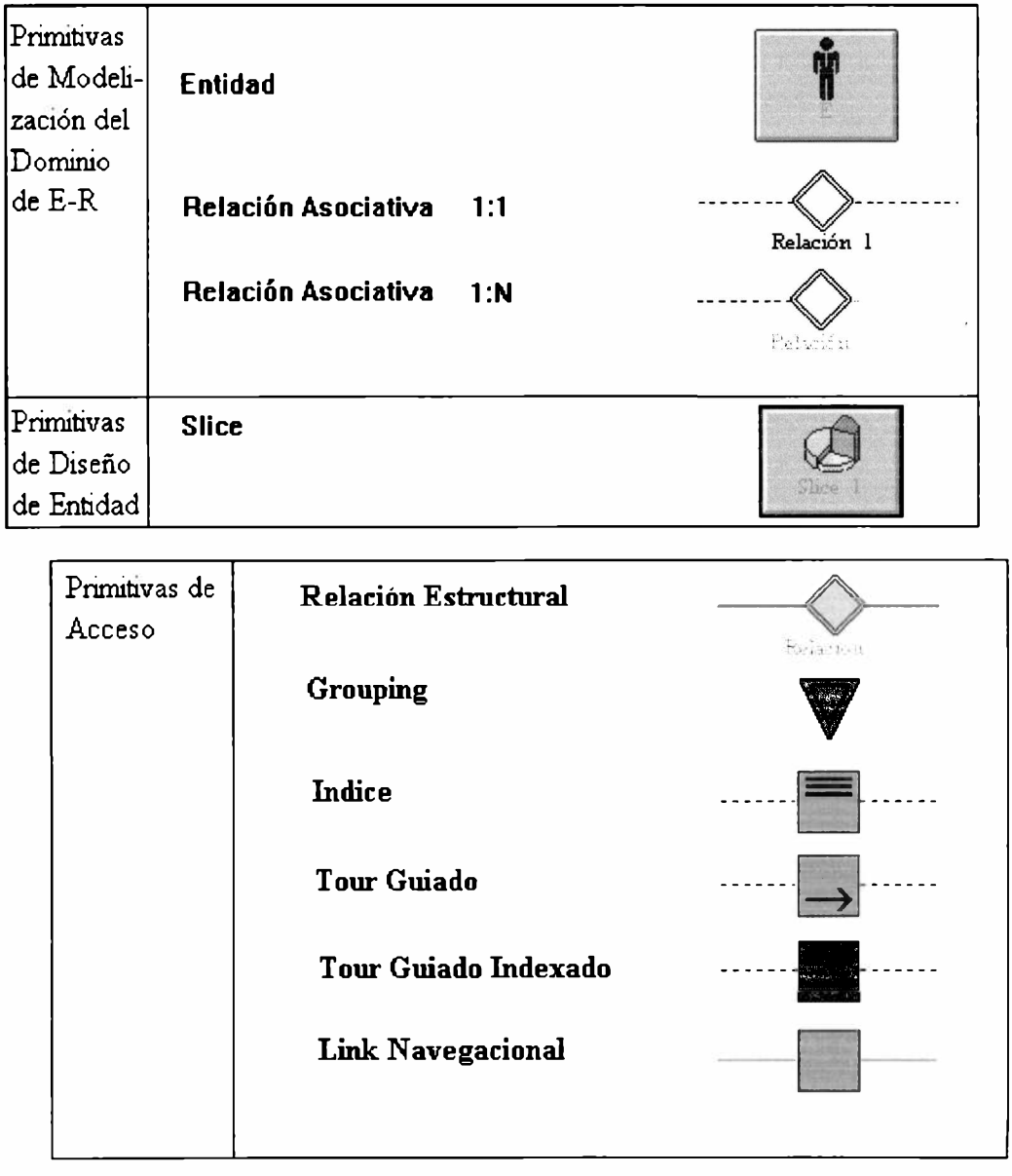


Figura 4.1: Primitivas de RMCASE.

La Figura 4.1 muestra en la parte superior las primitivas de modelización del dominio con las cuales se modela la información que interviene en la aplicación.

Las entidades y sus atributos representan objetos abstractos o físicos; en RMCASE las entidades se representan gráficamente como botones conteniendo el nombre de la entidad y una imagen que identifica a la misma. Los atributos de las entidades no tienen representación gráfica.

Las *relaciones asociativas* describen asociaciones entre entidades y se representan gráficamente con un rombo y líneas punteadas que llegan a las entidades involucradas. El nombre de la relación aparece debajo de los rombos. La aridad de las relaciones pueden ser uno a uno (1:1) ó uno a varios (1:N). Nosotros diferenciamos la aridad según el color de la línea punteada que llega a la entidad: si es negro la aridad en esa entidad es 1, si es rojo la aridad es N (ver Figura 4.2).

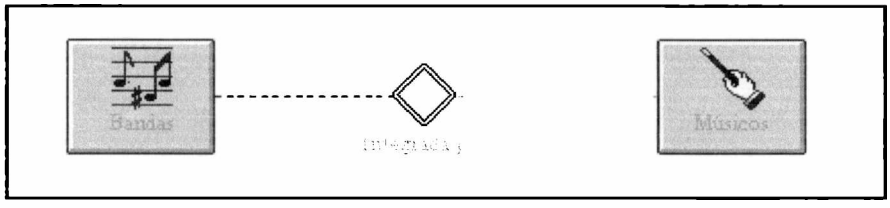


Figura 4.2: Relación Asociativa entre dos Entidades.

Las Entidades intervinientes son Bandas y Músicos. La relación asociativa se llama "Integrada por" y la aridad es 1:N desde Bandas, es decir que la línea que llega a Músicos es roja. Las entidades Músicos y Bandas tienen una imagen que las identifican además del nombre.

Los *slices* agrupan uno o más atributos de diferente naturaleza (ver Sección 2: 4.2), los atributos pueden formar parte de más de un *slice*. El usuario elige cual de los *slices* de una entidad quiere ver. Cada *slice* se representa como un botón con el nombre y una imagen por defecto, como se ve en la mitad de la Figura 4.1.

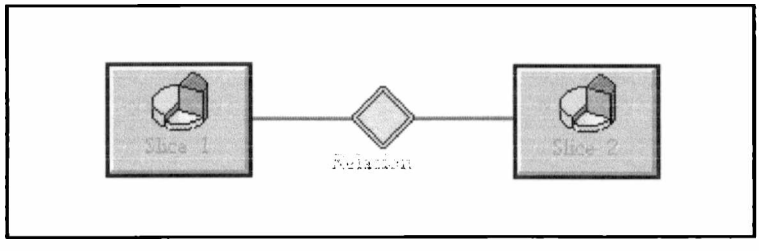


Figura 4.3: Relación Estructural entre dos Slices.

En RMCASE se soportan las seis primitivas de acceso que se muestran en la parte inferior de la Figura 4.1. Las *relaciones estructurales* se representan de la misma manera que las relaciones asociativas pero las líneas son llenas de color azul y son bidireccionales.

Las estructuras de acceso más importantes son los índices, los tours guiados y los *groupings*. Un índice actúa de la misma manera como lo indica RM (ver Sección 2: 4.2). En cambio introducimos una modificación en los *tours guiados* permitiendo al usuario salir del camino sin tener que llegar al último ítem. Los *groupings* se asocian a otros *groupings* o entidades a través de los *links* navegacionales. En la Figura 4.1 el *link* navegacional está representado con líneas llenas de color rojo, en este caso está representado sin estructura de acceso. Solamente los *links* navegacionales entre un *grouping* y una entidad pueden contener

una estructura de acceso, en estos se pueden definir condiciones del acceso que indicarán qué instancias de una entidad se podrán acceder.

Las relaciones asociativas en la navegación se llaman *links* asociativos y tienen asociada una estructura de acceso excepto en las que la aridad es 1 a 1. Los *links* asociativos se representan de la misma manera que una relación asociativa con una estructura de acceso, por ejemplo en la Figura 4.1, el tour guiado, el índice y el tour guiado indexado representan *links* asociativos.

2.2.- Metodología de Diseño.

La metodología utilizada está basada en la de RM fundamentalmente en los pasos de diseño (Paso1: Diseño de Entidades y Relaciones, Paso 2: Diseño de Entidad y Paso 3: Diseño Navegacional); luego a diferencia de RM se agregó un paso (Paso 4) que consiste en la carga de los datos que serán presentados al usuario en la ejecución del prototipo; para el Paso 5 de RM, que consiste en el diseño de la interface con el usuario, se utilizó una interface por defecto para demostrar la factibilidad del prototipo; en el Paso 5 de la metodología de RMCASE se establecen las relaciones a nivel de instancias y el Paso 6 consiste en la construcción del prototipo de acuerdo al diseño realizado en los pasos anteriores.

A continuación describiremos los seis pasos de la metodología de RMCASE para el diseño y construcción de aplicaciones Hipermedia (ver Figura 4.4). Nuestro énfasis está puesto en los pasos de diseño (los tres primeros). Cada uno de estos pasos representa un contexto de diseño que se implementa en un editor de diseño.

Los tres pasos siguientes realizan las tareas de poblar la Hiperbase, establecer las relaciones a nivel de instancias y construcción del prototipo. En una etapa posterior se puede ejecutar y evaluar el prototipo dejando la posibilidad de rediseñarlo y construirlo nuevamente.

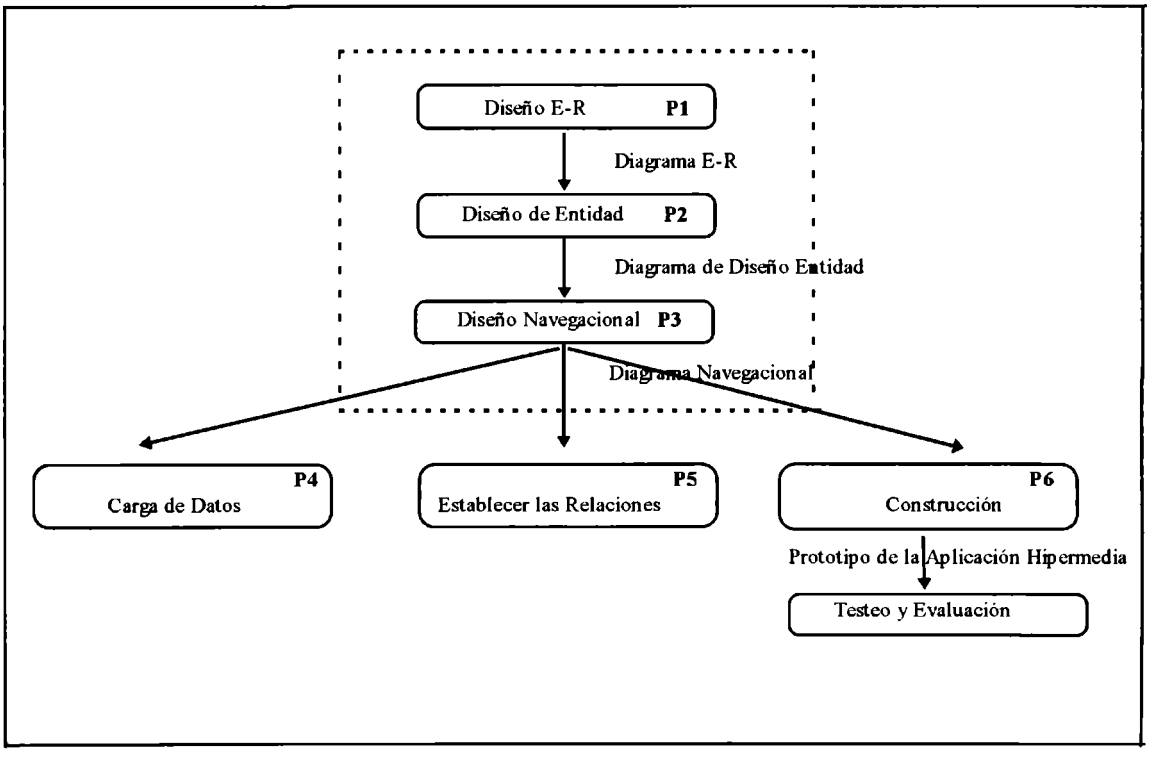


Figura 4.4: Metodología de RMCASE.

En la Figura 4.4, los tres primeros pasos (encerrados en la zona punteada) se refieren a los pasos de diseño. La construcción (paso P6) no necesariamente se debe realizar luego de haber finalizado P4 y P5. Para una mejor ejecución y testeo del prototipo se aconseja construir luego de haber modificado el diseño en los pasos P1, P2 ó P3.

2.3.- Características de RMCASE.

RMCASE fue diseñado no solo como una herramienta *Case* que soporta la metodología RM, sino que ayuda a resolver los siguientes problemas comunes:

- la rigidez de las metodologías del desarrollo del software tienden a retardar el desarrollo del software, para esto se provee la facilidad *feedback loops* que permiten a los desarrolladores seguir su proceso cognitivo natural durante el diseño y el desarrollo.
- la dificultosa comunicación entre los usuarios, diseñadores y desarrolladores que existe mientras se desarrolla la aplicación Hipermedia, se facilita a) con diagramas de diseño intermedio fáciles de comprender que son utilizados como documentación de la aplicación y b) con la posibilidad de experimentar con prototipos de la aplicación.
- para la dificultad para los usuarios de visualizar las aplicaciones Hipermedias, se provee la posibilidad de un prototipado rápido.

2.3.1.- Proceso de Diseño.

Se puede ver al diseño de Hipermedia como cualquier otra actividad de diseño, desde dos puntos de vistas:

- las técnicas de diseño formal prescritas para producir el diseño.
- las observaciones sobre cómo los diseñadores actualmente conducen el proceso de diseño

La mayoría del material referente al diseño de Hipermedia se ocupa principalmente del primer punto, proponiendo una técnica formal para una cierta clase de dominio de aplicación. Pero el rango de las aplicaciones de Hipertexto es tan amplio que no hay una única técnica de diseño formal que sea apropiada para diseñar a todas las aplicaciones.

Por otro lado, los diseñadores (como humanos) tienden a realizar aspectos específicos del proceso de diseño de la misma manera, descuidando las técnicas formales de diseño usadas. Este proceso cognitivo involucra a la secuencia actual de estados mentales y acciones que toman lugar mientras se sigue la técnica de diseño. Sin embargo, observando a los diseñadores de hipertexto mientras trabajan, nos muestran que el esfuerzo y los pasos detallados del proceso de diseño en sí son importantes, las técnicas formales generalmente ignoran el proceso de diseño [2].

En resumen, diseñar no es simplemente seguir una técnica de diseño formal. Esto es una actividad humana incremental y oportunística que involucra los estados identificados en los métodos de diseño formal y las categorías de las actividades mentales de los diseñadores. El rigor de las técnicas de diseño formal y la aparentemente errática organización de los actuales procesos de diseño no son contradictorias; son complementarias. Ellos son dos aspectos igualmente importantes e indisociables del diseño de aplicaciones Hipermedia.

RMCASE soporta el proceso cognitivo de los diseñadores, facilita moverse entre los pasos de la metodología manteniendo el foco en el mismo objeto de diseño. Este tipo de

navegación permite a los diseñadores alternar fácilmente entre los niveles conceptual y concreto.

Este proceso de diseño RMCASE lo implementa a través de Contextos de trabajo, *Feedback loops* y Prototipación. A continuación describiremos de cada uno de estos conceptos.

2.3.2.- Contextos de Trabajo.

RMCASE es una herramienta Case para el diseño y desarrollo de aplicaciones Hipermedia que desde el punto de vista metodológico soporta todo el ciclo de vida del software de una aplicación (como el descripto en 2.2) mediante un conjunto de contextos, uno por cada estado o paso, llamado *contexto de trabajo*.

Cada contexto de trabajo en RMCASE maneja un conjunto de objetos de diseño (por ejemplo entidades, relaciones, atributos, *slices*, estructuras de acceso, etc.) de manera particular. Sin embargo los objetos de diseño no son exclusivos de un contexto en particular, estos siguen una trayectoria evolutiva como parte del proceso del desarrollo del software. Por lo tanto al mismo objeto de diseño se puede acceder con diferente grado de transformación según el contexto a través del cual lo estoy visualizando.

En 2.4 describiremos en detalle cada contexto de trabajo.

2.3.3.- Feedback loops.

RMCASE posibilita la rápida transición entre los estados (pasos) de la metodología vía la "navegación" entre contextos. Como los objetos de diseños son compartidos entre diferentes contextos, esta clase de "navegación" permite a los desarrolladores enfocarse en uno o más objetos de diseño mientras se mueven hacia atrás y hacia adelante entre los distintos estados de la metodología. Esta capacidad de cambiar hacia atrás y hacia adelante entre los pasos de la metodología es un componente fundamental en el proceso de diseño y desarrollo. Con RMCASE se puede comenzar a diseñar cualquier entidad, definir sus *slices*, su estructura de navegación, crear un prototipo experimental y luego pasar a otra entidad. Los *feedbacks loops* se implementan para la "navegación" entre los diferentes contextos de trabajo.

El concepto de navegación aquí es distinto al utilizado en Hipermedia, nosotros lo utilizamos para cambiar de contextos de trabajo.

2.3.4.- Prototipación.

Un prototipo ofrece a los diseñadores la capacidad de experimentar la aplicación como eventualmente sería ejecutada. El contexto de prototipo puede ser usado para instanciar una aplicación desde un diseño. Aquí los desarrolladores de software pueden testear los diferentes aspectos del diseño de una aplicación Hipermedia, tal como su estructura de información y los patrones de navegación.

Los componentes de la aplicación pueden ser objetivamente evaluados para luego realizar las actividades de rediseño y reconstrucción. Por esto los diseñadores no solo necesitan herramientas que soporten *feedbacks* entre los contextos de trabajo de la metodología, sino también necesitan mecanismos para moverse fácil y rápidamente entre los niveles abstractos y de instancia [2].

RMCASE posibilita la construcción de prototipos desde cualquier estado del diseño y desarrollo del software, aún con diseños incompletos.

2.3.5.- Proceso de Diseño: Ejemplo.

A continuación explicaremos como se puede construir una aplicación Hipermedia con RMCASE mostrando como se utilizan las características mencionadas anteriormente y como el diseñador puede seguir su proceso cognitivo de diseño.

Queremos diseñar una aplicación Hipermedia que resuma la historia del Rock Nacional en los últimos años para ser expuesto en un negocio de venta de instrumentos musicales. La idea intuitiva es utilizar información de las Bandas Nacionales surgidas últimamente y la discografía producida por ellas.

La información que interviene en la aplicación es acerca de: bandas, la discografía de estas e instrumentos musicales, pero en las bandas intervienen los músicos que tocan instrumentos. En resumen tenemos las siguientes entidades: bandas, músicos, discografías e instrumentos; las relaciones serían: un músico toca en una o más bandas, una banda está integrada por uno o más músicos y además tiene un líder, un músico toca varios instrumentos, una banda edita uno o más discos y un disco puede ser editado por una banda. En primera instancia vamos a diseñar la entidad bandas, le agregamos los atributos: nombre de la banda, cantidad de integrantes, año de inicio, foto de la banda, mejor video y mejor tema. Luego defino los *slices* de la entidad agrupando los atributos nombre de la banda, cantidad de integrantes, año de inicio y foto de la banda en un *slice* "Datos Generales" y en otro *slice* "Video" los atributos: nombre de la banda, mejor video y mejor tema. Después cargo algunas instancias de Bandas y voy al diseño navegacional para crear el *grouping* cabecera de la aplicación y un *link* navegacional que me conecte a la entidad Bandas y así puedo construir el prototipo para visualizar la aplicación de bandas nacionales con solamente la entidad Bandas. Una vez que ejecuté y analicé el prototipo vuelvo al diseño y agrego la entidad Músicos y las relaciones asociativas "Toca en", "Integrada por" y "Líder" entre las entidades Músicos y Bandas. En este momento me doy cuenta de que podría hacer más completa la aplicación si agrego la información de los temas de las bandas creando la entidad Temas, entonces elimino el atributo mejor tema de la entidad Bandas y creo la relación "Mejor Tema" entre las entidades Temas y Bandas. Luego diseño Músicos y Temas de la misma manera que Bandas y cargo algunas instancias de estas entidades. Para cada relación asociativa establezco las relaciones a nivel instancia y construyo nuevamente el prototipo para experimentar nuevamente una visualización de la aplicación final. Posteriormente, sigo diseñando el resto de la aplicación pasando de un contexto de trabajo a otro ("*feedback loops*"), modificando y agregando objetos de diseño en cada "contexto de trabajo" y "prototipando" cuando lo necesito.

2.4.- Contextos de Trabajo de RMCASE.

A continuación describiremos en profundidad los contextos de trabajos utilizados en RMCASE. La salida de cada uno de estos contextos es un diagrama (en especial los contextos de diseño que son los tres primeros) que es utilizado como una importante documentación de la aplicación que se está implementando.

2.4.1.- Contexto de Diseño de Entidades y Relaciones.

Este contexto representa el primer paso de la metodología que es el diseño de Entidades y Relaciones (E-R). El diseño E-R puede ya existir si la aplicación está basada en una base de datos ya diseñada con E-R. Los tres objetos básicos de diseño en este contexto son las entidades, los atributos y las relaciones.

- Una entidad es un elemento conceptual del dominio de la aplicación, caracterizada por un conjunto de atributos. Todas las entidades deben poseer un clave, que corresponde a un conjunto de atributos.
- Un atributo representa una unidad de información. Los atributos tienen un nombre y un tipo y siempre están asociados a una única entidad. Se debe especificar qué atributos formarán la clave de la entidad.
- Una relación asociativa es una unión conceptual entre dos entidades. La aridad o cardinalidad puede ser uno a uno ó uno a varios. Las relaciones varios a varios se deben dividir en dos, uno a varios. Las relaciones asociativas deben diseñarse teniendo en cuenta que luego derivarán en los caminos que el usuario utilizará para navegar entre los distintos ítems de información.

Aquí en este paso el diseñador no solo observa el dominio, sino que también le impone una estructura.

2.4.2.- Contexto de Diseño de Entidad.

Este contexto se corresponde con el segundo paso de la metodología: el diseño de entidad. En este contexto se determina cómo la información de una entidad elegida será presentada al usuario y cómo puede acceder a ésta. Esto involucra explotar una entidad en partes o *slices* significativos y organizarlos en una red de Hipertexto. Los objetos de diseño que se utilizan aquí son: los atributos, los *slices* y las relaciones estructurales.

- Un *slice* es un conjunto de atributos que pertenecen a la entidad elegida. Cada entidad tiene un *slice* distinguido, que se lo llama *cabecera*, y se utilizará como punto de entrada por defecto para los *links* que llegan a la entidad.
- Un atributo puede formar parte de más de un *slice*. Un *slice* debe tener como mínimo un atributo.

- El diagrama de diseño de entidad también modela la navegación entre *slices* a través de relaciones estructurales bidireccionales. Estas relaciones representan conexiones entre *slices*.

Las tareas que debe realizar el diseñador en este paso son:

- Dividir la entidad en *slices* teniendo en cuenta que cada *slice* representará un todo para el usuario del sistema. Los *slices* deben agrupar solamente ítems de información relacionada pero no deben contener demasiada información, en este caso se sugiere crear un nuevo *slice*.

- Elegir un *slice* cabecera que se verá como una propiedad del *slice*.

- Interconectar los distintos *slices* a través de las relaciones estructurales. En estos dos últimos puntos se sugiere un análisis del dominio.

- Construir apropiadamente el nombre de las relaciones estructurales. Nosotros sugerimos lo siguiente: por ej. desde un *slice* Datos a otro Videos el nombre de la relación estructural sea "Datos[Videos]".

- Asignar a cada *slice* los atributos que incluirá de la entidad elegida (teniendo en cuenta lo dicho en el primer punto).

2.4.3.- Contexto de Diseño Navegacional.

Este contexto de diseño se refiere al tercer paso de la metodología, se especifican las características navegacionales de la aplicación. Cada relación asociativa del diagrama E-R será analizada y se le asignará una única estructura de acceso. En el caso de las relaciones 1:1 no se asigna una estructura de acceso.

En el diseño navegacional los objetos de diseño son:

- Los *groupings* son estructuras de selección que proveen una estructura jerárquica para acceder a la información similar a un menú. Se debe definir un *grouping* cabecera que será el punto de comienzo en la aplicación Hipermedia.

- Los *links* navegacionales (ver Figura 4.5) se utilizan para establecer que partes de la aplicación Hipermedia estarán asociadas a un *grouping*. Estos se establecen entre dos *groupings* o entre un *grouping* y una entidad. Se pueden definir condiciones de acceso que indicarán qué instancias de una entidad se podrán acceder. En particular en el *link* entre *groupings* no se define ninguna condición de acceso. Las condiciones involucran un atributo de la entidad y una constante para su comparación. Por ej. Músicos(Sexo) = "Masculino". En este caso en la ejecución de la aplicación solo se visitarán los Músicos varones.

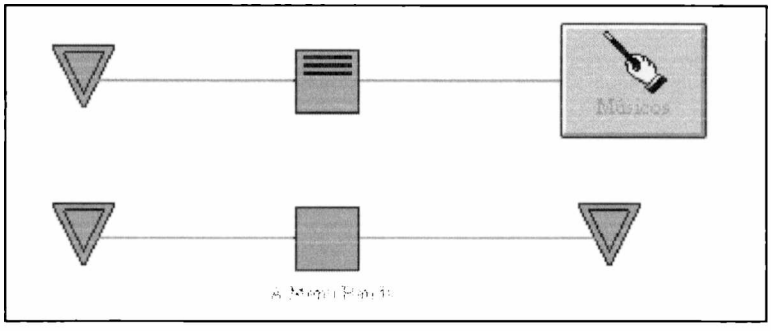


Figura 4.5.

En la parte superior se muestra un ejemplo de un link navegacional entre un grouping y una entidad. En la parte inferior mostramos un link navegacional entre dos groupings.

- Las estructuras de acceso: índices, tours guiados y tours guiados indexados, que fueron definidos anteriormente, se aplican solamente a los *links* navegacionales que conectan un *groupings* y una entidad y los *links* asociativos que son los que derivan de las relaciones asociativas del contexto de diseño E-R.

Una vez que el diagrama E-R ha sido dibujado, es conveniente que el grafo resultante quede conexo para poder visitar todas las entidades. No hay una navegación parcial del diseño, todas las entidades interconectadas serán visitadas durante la ejecución de la aplicación.

Durante esta etapa de diseño las tareas que el diseñador debe realizar son:

- Crear los *groupings* correspondientes para organizar la navegación. Esta tarea debe estar basada en los requerimientos que presenta la aplicación .

- Identificar el *grouping* cabecera. Este será el punto de partida en la ejecución de la aplicación Hipermedia. De esto se deduce que debe haber por lo menos un *grouping* en el diseño.

- Asignar que estructura de acceso se utilizará en cada *link* asociativo y en cada *link* navegacional, índice, tour guiado o tour guiado indexado.

- Cuando se utilizan los índices o los tours guiados indexados se debe especificar el atributo que aparecerá en la lista de índices que provee el acceso directo. Se recomienda elegir un tour guiado cuando el número de instancias que se visitarán es menor que 10; se debe utilizar un índice cuando hay un gran número de instancias y además la entidad tiene una clave significativa; los tours guiados indexados se utilizan cuando hay una clave que se pueda ubicar en un índice y además se desea alguna navegación local.

- En los *links* navegacionales entre un *grouping* y una entidad se puede especificar una condición de navegación como mencionamos anteriormente.

2.4.4.- Contexto de Población de la Hiperbase.

Este es el cuarto paso de la metodología. Los datos que se cargan aquí corresponden a las instancias de las entidades y los atributos de la pantalla de presentación de los *groupings*.

Estos datos se preservan de manera que estén organizados por entidad y por *grouping*. Los datos de las instancias de la entidad, se pueden cargar aquí directamente desde una base de datos externa. Hablando a nivel de instancias los datos pueden ser modificados, borrados o agregados.

Se provee la facilidad para manejar los distintos tipos de objetos multimediales. Los distintos formatos con los que se puede trabajar son: video, imagen, sonido, memo y texto.

Las tareas que se realizan en este paso son:

- Carga de los datos de *grouping*. Se proveen atributos por defecto que serán utilizados en la pantalla de presentación del *grouping* cuando se ejecute la aplicación Hipermedia. Permitimos que se cargue un atributo de cada tipo (sonido, video, texto, imagen y memo) por *grouping*.

- Carga de las instancias de entidad. El poblado de los datos de una entidad es mucho más simple cuando la misma se basa en una base externa; de otra manera la tarea es más tediosa porque se deben cargar todos los atributos de todas las instancias de la entidad.

2.4.5.- Contexto para establecer Relaciones a Nivel de Instancia.

El usuario deberá indicar, por cada relación asociativa del contexto de diseño E-R, las asociaciones de una instancia de la entidad origen de la relación que se conectarán con una instancia de la entidad destino si la aridad de la relación es 1:1, o con varias instancias de la entidad destino si la aridad es 1:N. Cuando se navega en ejecución y se atraviesa una relación asociativa solo se visitan las instancias de la entidad destino que fueron conectadas en este paso con la instancia de la entidad origen.

2.4.6.- Contexto de Construcción del Prototipo.

En este contexto la aplicación Hipermedia es construida en base a lo establecido en las etapas de diseño. El diseño de la inteface con el usuario no se realiza porque obedece a un diseño por defecto que genera RMCASE. Antes de la construcción se testeará que la información que se generó en los pasos previos de diseño sea consistente. Por ejemplo:

- todas las entidades deben tener por lo menos un *slice*.
- todas las entidades y *slices* deben tener por lo menos un atributo.
- debe existir un *grouping* cabecera y un *link* navegacional en el diseño.
- las condiciones de los *links* navegacionales deben ser con atributos de tipo comparable.
- las estructuras de acceso indexadas deben tener especificado el atributo índice.
- todas las entidades deben tener cargadas instancias y todas sus relaciones establecidas.

La construcción solo se llevará a cabo si los testeos previos son satisfactorios.

Después de finalizado este paso el usuario puede ejecutar el prototipo para testearlo y evaluarlo y volverlo a rediseñar y construir si así lo desea (ver 2.2).

2.4.6.1.- Consideraciones acerca del prototipo.

El prototipo que genera RMCASE provee una distinción entre los distintos tipos de *links*, agrupándolos por separado según sean: (a) *links* navegacionales entre *groupings*, (b) *links* asociativos y navegacionales entre *grouping* y entidad ó (c) *links* estructurales (que son los que derivan de las relaciones estructurales del diseño de entidad). De la misma manera se agrupan los atributos según el tipo, mostrando solo un atributo por tipo por vez. Por ejemplo supongamos que tenemos en la entidad Músicos los atributos de tipo texto: nombre artístico, nombre real, domicilio y edad entonces generamos una lista visualizando solo uno de ellos, pudiendo el usuario de la aplicación elegir de la lista el que desea ver. De la misma manera agrupamos cada tipo de *link* y cuando el usuario elige uno de la lista ese es el que atravesamos.

Efecto que produce atravesar un *link* según el grupo:

a) permiten pasar de un *grouping* a otro quedando más organizado para el usuario la forma de ver las opciones de navegación que se le presentan desde donde está parado actualmente. La información que se muestra en la pantalla de *grouping* son los atributos del mismo que fueron cargados en el paso 4.

b) cambian el contexto de la información cuando se atraviesan, visitando otro tipo de entidad diferente al que estaba mostrándose anteriormente.

c) cuando se atraviesan no cambia la instancia de la entidad, solo se muestra diferente información de la misma instancia de entidad contenida en otro *slice*. De esto se deduce que el prototipo visualiza pantallas correspondientes a *groupings* o *slices*.

El prototipo provee un mecanismo de *backtracking* muy potente, que permite volver al mismo contexto del que se partió en un paso anterior.

Cuando se atraviesa un *link* con una estructura de acceso indexada, el prototipo despliega una lista de las posibles instancias de la entidad destino a las que el usuario podrá acceder directamente. Cuando se vuelve de una instancia visitada a través de un índice, el prototipo mantiene el contexto de información pudiendo el usuario visitar otra instancia de la lista de índices que ya está nuevamente desplegada.

Si se atraviesa un *link* con una estructura de acceso con tour guiado, el prototipo permite la navegación local entre instancias de la entidad destino.

El prototipo también provee un fácil acceso a la estructura de la red para permitir al usuario ubicarse dentro de la aplicación.

Sección 5: RMCASE: Construyendo una aplicación.

1.- Introducción.

En esta sección describiremos en profundidad como construir con RMCASE una Aplicación Hipermedia. Cada Aplicación Hipermedia se maneja como un proyecto dentro de RMCASE.

1.1.- Qué es un proyecto en RMCASE?

Un proyecto en RMCASE involucra toda la documentación referente a una aplicación Hipermedia, desde el diseño de la misma hasta el prototipo generado. Está claro que pueden existir varios proyectos aunque en RMCASE solo se trabaja con uno a al vez. En RMCASE se puede crear un proyecto nuevo o abrir uno existente para ser modificado o consultado.

2.- Cómo desarrollar una aplicación Hipermedia?

Existe más de una manera de desarrollar un proyecto, nosotros lo presentaremos en orden acorde a los pasos de la metodología RMCASE especificada en la Sección 4. Para ello utilizaremos el ejemplo de Bandas Nacionales. Este ejemplo contiene información de músicos, bandas nacionales, temas musicales, discografía de las bandas e instrumentos musicales. Los músicos pueden tocar en varios instrumentos, ser autores de varios temas y pueden tocar en más de una banda. Las bandas están integradas por músicos, uno de ellos puede ser el líder de la banda y además pueden editar discografía. Los temas son compuestos por uno o varios músicos, pertenecen a una banda y están editados en alguna discografía.

2.1.- Generalidades de RMCASE.

RMCASE consta de las siguientes opciones:

1.- Proyectos.

2.- Aplicación.

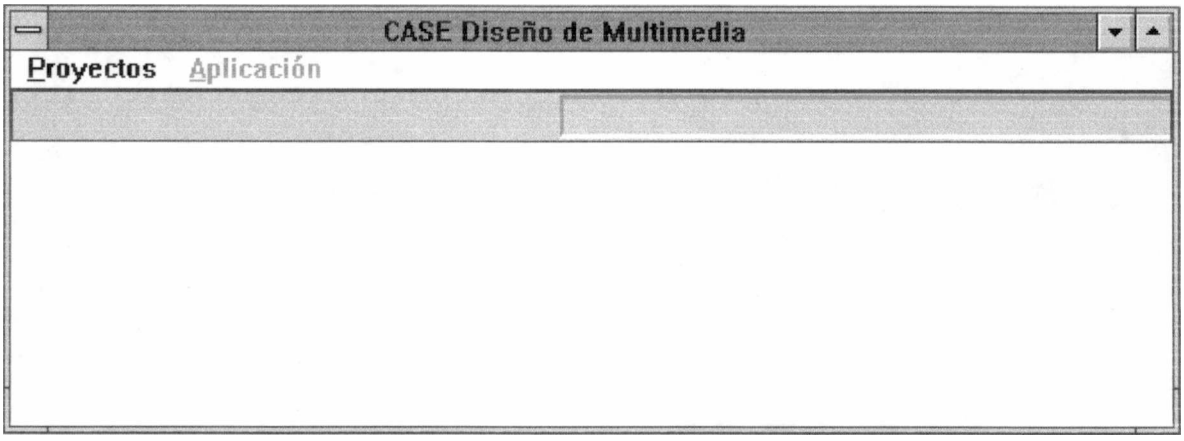


Figura 5.1: Pantalla Inicial de RMCASE

2.1.1.- Proyectos.

Eligiendo el menú "Proyectos" usted puede:

a.- Crear un Proyecto nuevo.

b.- Abrir un Proyecto existente.

c.- Guardar un Proyecto.

d.- Guardar un Proyecto con otro nombre.

e.- Ir al Diseño Navegacional.

f.- Acerca de RMCASE.

g.- Salir de RMCASE y regresar a Windows.

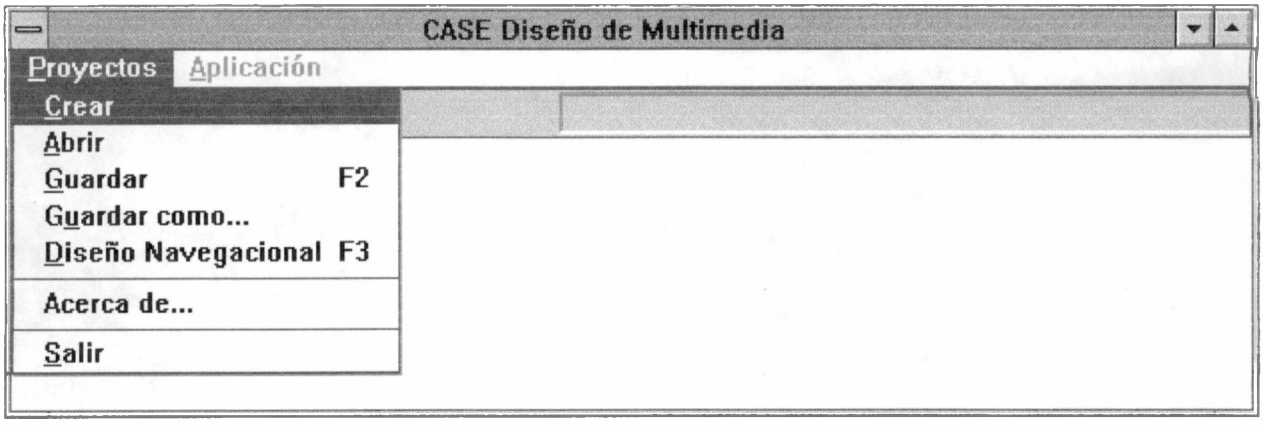


Figura 5.2: Menú Proyectos.

2.1.2.- Aplicación.

Esta opción inicialmente está deshabilitada hasta que se abra un proyecto. Eligiendo el menú "Aplicación" usted puede:

a.- Cargar datos

a.1.- Datos de las Entidades.

a.2.- Datos de los Groupings.

b.- Establecer relaciones entre instancias de entidades.

c.- Construcción del Prototipo.

d.- Ejecución del Prototipo.

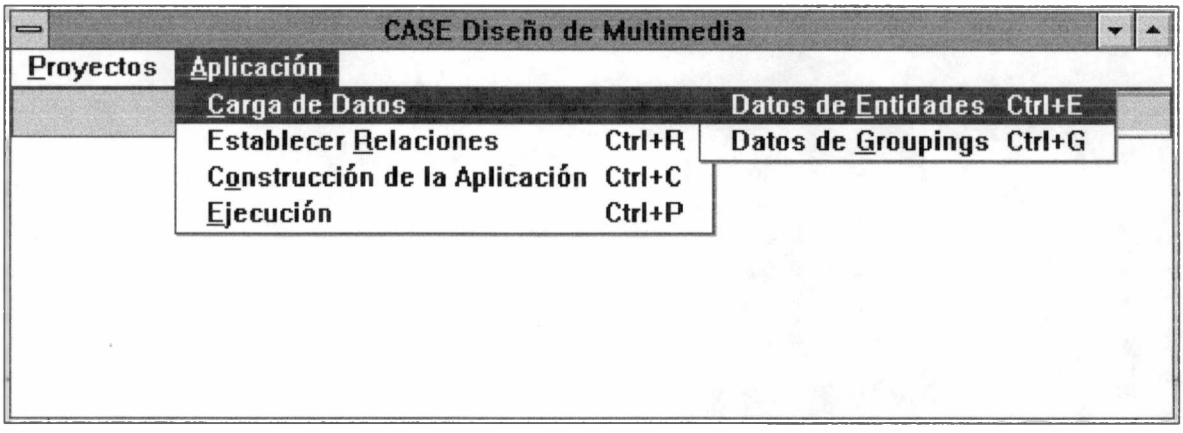


Figura 5.3: Menú Aplicación

3.- Trabajando con Proyectos.

3.1.- Crear un Proyecto nuevo.

Cuando se crea un proyecto nuevo, RMCASE le presenta al diseñador el editor de Diseño E-R vacío listo para crear entidades y relaciones asociativas. A partir de este diseño se pueden activar los editores de Diseño de Entidad y de Diseño Navegacional (en los puntos siguientes 3.1.1., 3.1.2. y 3.1.3. se especifican los tres editores de diseño).

3.1.1.- Editor de Diseño E-R.

Las principales operaciones que se pueden realizar en este editor son crear y borrar entidades y relaciones asociativas.

Con respecto a las entidades:

- Se puede asignar una base de datos externa a una entidad. Cuando hablamos de una base de datos externa nos referimos a una tabla de una base de datos Access. Es decir que se debe especificar primero del nombre de la base de datos y luego elegir la tabla a utilizar de la lista de tablas de la base de datos que se presentará en pantalla. Las bases de datos Access tiene extensión ".MDB".

- Los atributos se eligen a partir de los campos de la tabla, si ésta se especificó. Se pueden agregar atributos nuevos aunque tenga una base de datos externa asignada. Los atributos también pueden ser borrados.
- Se puede asignar un ícono que identifique a la entidad.
- El nombre de la entidad puede ser modificado.
- Las entidades deben poseer una clave. Las claves se definen eligiendo un grupo de atributos, los cuales deben ser de tipos comparables.

Observación:

Refiriéndonos al primer ítem explicado anteriormente con respecto a las bases de datos externas, debemos hacer una aclaración dado que se mezclan distintos niveles de abstracción cuando se manejan entidades a nivel diseño y base de datos a nivel de implementación.

Inicialmente trabajábamos a un nivel de diseño la creación de entidades y sus atributos, y a nivel de población de la hiperbase asignábamos una base de datos externa a la entidad, para después relacionar univocamente los atributos de la entidad con los campos de la base de datos externa. Es decir, que de esta manera no se mezclaban los niveles de abstracción pero surgían posteriormente demasiados inconvenientes y obstáculos que hacían que el trabajo del diseñador se tornara muy pesado y tedioso. El mayor inconveniente que se presentaba era que el diseñador antes de asignar la base de datos externa podía utilizar algunos atributos de la entidad para diseñar estructuras de acceso, condiciones de navegación e inclusive determinar claves de la entidad, pero cuando asignaba los campos a los atributos se encontraba con que los tipos de los campos no eran tipos válidos para realizar estas operaciones entonces debía volver a un estado de diseño anterior para poder realizar nuevamente estas operaciones.

Por este motivo y para facilitar la tarea al diseñador transgredimos en este caso particular los niveles de abstracción, permitiendo que la base externa y sus atributos se asignen directamente en esta etapa.

Con respecto a las relaciones asociativas:

- El nombre de la relación puede ser modificado.
- La aridad de la relación puede ser modificado (1:1, 1:N).

3.1.1.1.- Crear una Entidad.

Cuando se desea crear una entidad se debe clickear en el botón que indica la figura del editor de Diseño E-R, el puntero del mouse cambiará de forma (tomará forma de un cuadrado lleno de color negro), y aquí el diseñador deberá clickear dentro del editor en la posición que desee ubicar a la entidad creada. Antes de ubicar la entidad dentro del editor, el diseñador podrá anular la operación clickeando el botón derecho del mouse.

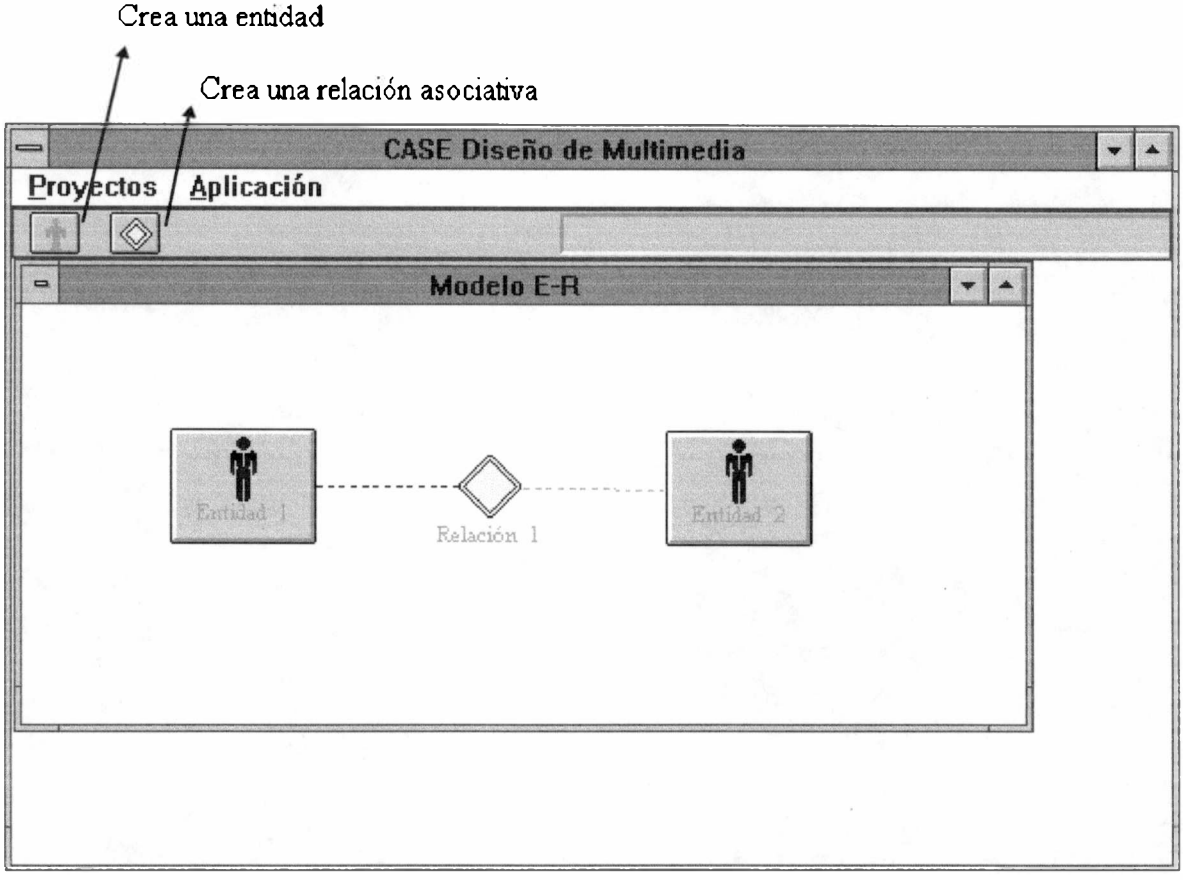


Figura 5.4: Editor de Diseño E-R.

Nota: Para observar el editor de Diseño E-R con los detalles en color ver el Apéndice.

3.1.1.2.- Borrar una Entidad.

Para borrar una entidad se debe clickear en la misma y luego presionar la tecla "Delete", la entidad desaparecerá junto con sus atributos y las relaciones asociativas que la involucraban. También se borrará todo lo que a ella respecta en el resto del proyecto. Por lo tanto, por todas las acciones que se producen cuando se ejecuta esta operación se debe tener sumo cuidado al realizarla.

3.1.1.3.- Crear una Relación Asociativa.

Cuando se desea crear una relación asociativa, al igual que cuando se crea una entidad, se debe clickear en el botón que indica la figura del editor de Diseño E-R, el puntero del mouse cambiará de forma (tomará forma de una flecha con doble sentido), y aquí el diseñador deberá clickear primero la entidad origen de la relación a crear y luego la entidad destino. Antes de clickear en la entidad destino el diseñador podrá anular la operación clickeando el botón derecho del mouse.

3.1.1.4.- Borrar una Relación Asociativa.

Para borrar una relación asociativa se debe clickear en la misma y luego presionar la tecla "Delete", la relación asociativa desaparecerá junto con todo lo que a ella respecta en el resto del proyecto. Por lo tanto, por todas las acciones que se producen cuando se ejecuta esta operación se debe tener sumo cuidado al realizarla.

3.1.1.5.- Propiedades de las Entidades.

Las propiedades de una entidad son: el nombre, la base de datos externa, la tabla de la base de datos, los atributos, la clave y el ícono que representa la entidad.

El nombre de la entidad por defecto se genera automáticamente con el nombre "Entidad N" siendo N la cantidad de entidades existentes. En un principio no hay base de datos externa asignada, por lo tanto no hay tabla, ni atributos, ni clave. El ícono por defecto es siempre el mismo para todas las entidades.

Para actualizar las propiedades de una entidad se debe clickear en la entidad con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.5. En esta pantalla el diseñador puede modificar el nombre de la entidad en (1), para que luego este aparezca en el botón de la entidad en el editor de diseño E-R. Nosotros previamente hicimos este trabajo con las entidades de la Figura 5.4, la Entidad 1 se renombró con Músicos y la Entidad 2 con Instrumentos.

Haciendo doble click en (2) se puede examinar el disco para elegir una base de datos Access para basar a la entidad. Recordemos que estos archivos tienen extensión ".MDB". Una vez elegida esta, se carga automáticamente la lista de tablas de la base de datos en (4); a su vez cuando se elige la tabla en (4) se cargan automáticamente los campos de esta en (5). Luego el diseñador indica con el botón Agregar que campos quiere como atributos de la entidad, éstos campos se van cargando en (6). También el diseñador puede agregar atributos nuevos, es decir que no sean campos de la tabla, clickeando el botón Nuevo e indicando el nombre del nuevo atributo. Cuando no se especifica una base de datos externa en (2) todos los atributos van a ser nuevos. Notar que los atributos "nombre" y "clase" de (6) tienen un número como prefijo, esto significa que son claves. Para hacer un atributo clave se debe hacer doble click en el mismo, si se procede en forma similar el atributo deja de ser clave.

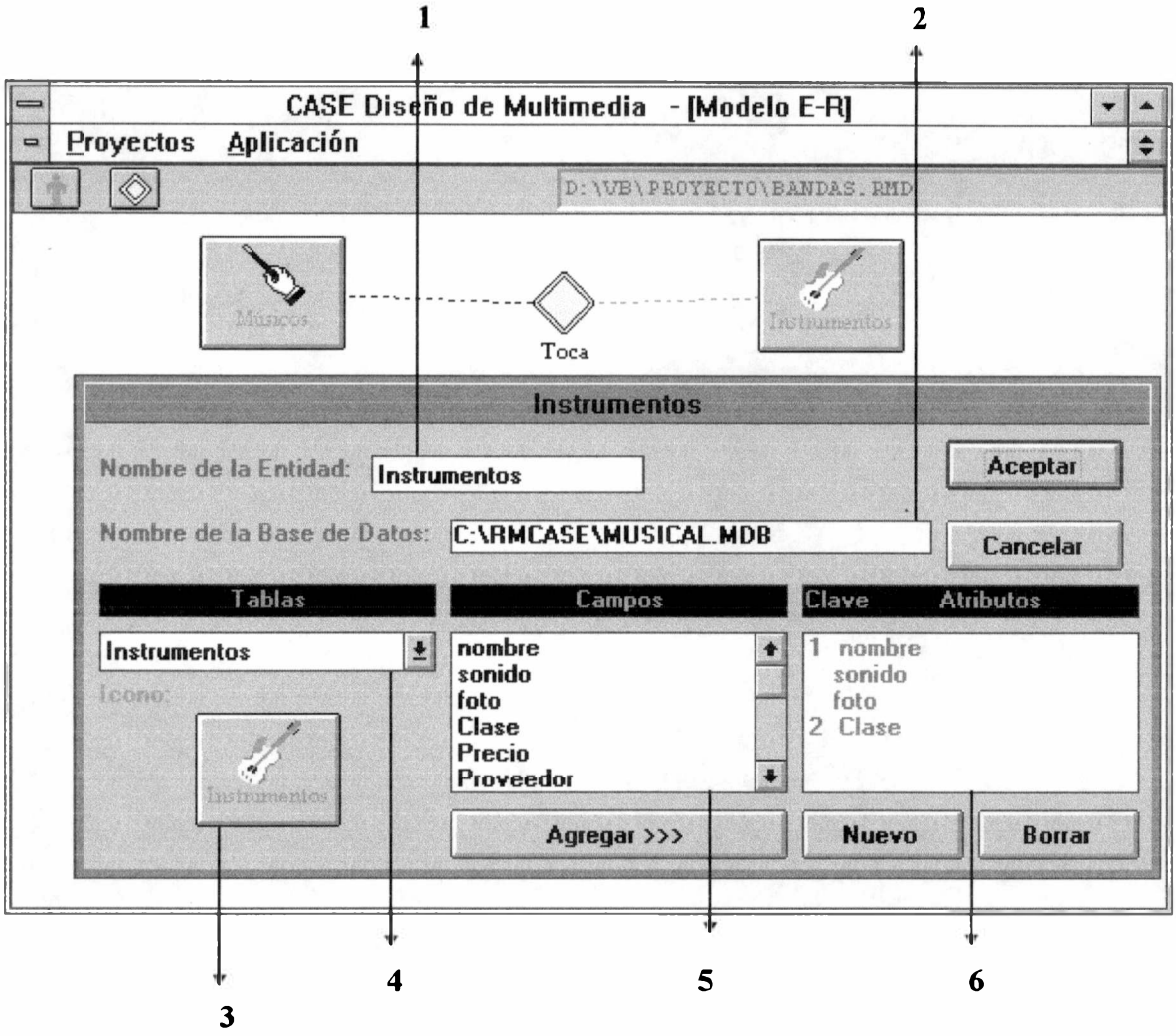


Figura 5.5: Pantalla de Propiedades de la Entidad.

1- Nombre de la entidad. 2- Nombre de la base de datos Access Externa. 3- Botón para elegir el ícono. 4- Listas de tablas de la base de datos Access elegida en 2. 5- Lista de campos de la tabla elegida en 4. 6.- Lista de atributos elegidos desde 5 y con el Botón Nuevo.

Botón Aceptar: Acepta las modificaciones realizadas.

Botón Cancelar: Cancela la operación.

Botón Agregar: Agrega el campo elegido en 5 a 6.

Botón Nuevo: Agrega a 6 un atributo nuevo que especifica el diseñador.

Botón Borrar: Borra el atributo especificado en 6.

3.1.1.6.- Propiedades de las Relaciones Asociativas.

Las propiedades de una relación asociativa son: el nombre y la aridad. El nombre por defecto se genera automáticamente con el nombre "Relación N" siendo N la cantidad de relaciones asociativas existentes y la aridad en un principio se define como 1 a 1.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P

Para actualizarlas se debe clickear en ella con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.6. En esta pantalla el diseñador puede modificar el nombre de la relación asociativa en (1), para que luego este aparezca en el editor de diseño E-R como se ve en la Figura 5.4.

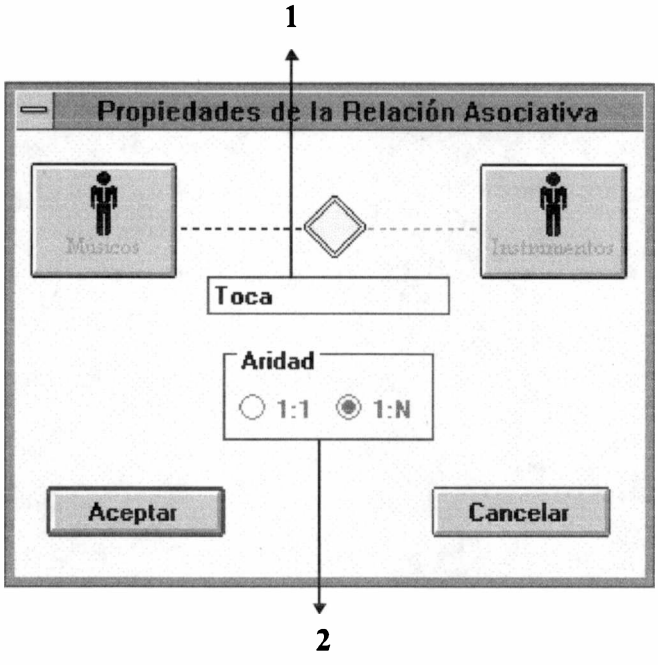


Figura 5.6: Pantalla de Propiedades de las Relaciones Asociativas.

1- Nombre de la relación asociativa. 2- Aridad de la relación.

Botón Aceptar: Acepta las modificaciones realizadas.

Botón Cancelar: Cancela la operación.

En este caso la entidad origen de la relación corresponde a Músicos y la entidad destino a Instrumentos. La aridad es 1:N (un músico puede tocar varios instrumentos) como se ve en (2). El color rojo de la línea que llega a instrumentos indica que son N en esta relación. Para modificar la aridad se debe clickear en el botón 1:1.

3.1.2.- Editor de Diseño de Entidad.

Para que el diseñador pueda realizar el Diseño de una Entidad este debe hacer doble click en la entidad deseada en el editor de Diseño E-R.

Las principales operaciones que se pueden realizar en este editor son crear y borrar *slices* y relaciones estructurales.

Con respecto a los *slices*:

- Los atributos del *slice* se eligen a partir de los atributos de la entidad que se está diseñando. Los atributos también pueden ser borrados del *slice*.

- El nombre del *slice* puede ser modificado.
- Se determina cuál va a ser el *slice* cabecera de la entidad que se está diseñando.

Con respecto a las relaciones estructurales:

- El nombre de la relación puede ser modificado.

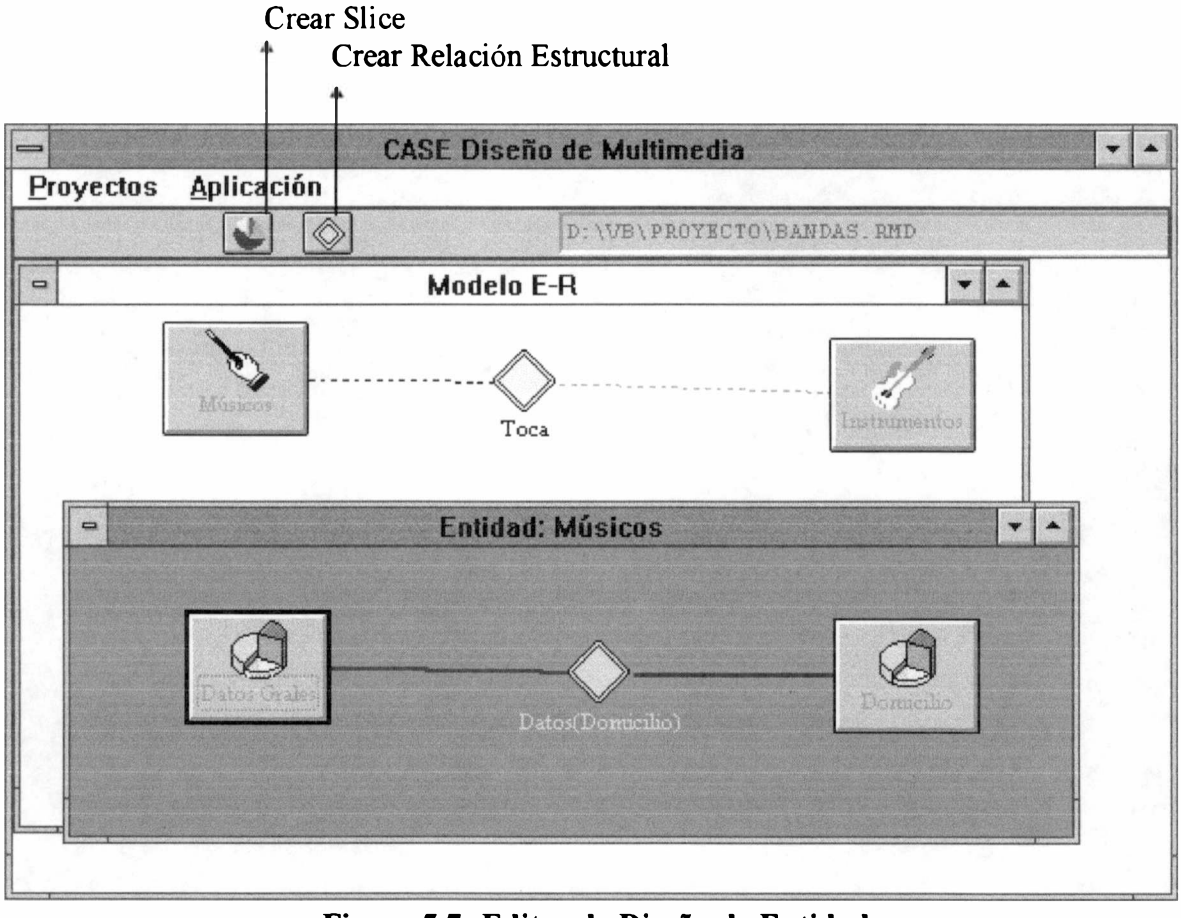


Figura 5.7: Editor de Diseño de Entidad.

Nota: Para apreciar el editor de Diseño de Entidad con los detalles en color ver el Apéndice.

3.1.2.1.- Crear un Slice.

Cuando se desea crear un *slice* se debe clickear en el botón que indica la Figura 5.7 del editor de Diseño de Entidad, el puntero del mouse cambiará de forma (tomará forma de un cuadrado lleno de color negro), y aquí el diseñador deberá clickear dentro del editor en la posición que desee ubicar al *slice* creado. Antes de ubicar el *slice* dentro del editor, el diseñador podrá anular la operación clickeando el botón derecho del mouse.

3.1.2.2.- Borrar un Slice.

Para borrar un *slice* se debe clickear en el mismo y luego presionar la tecla "Delete", el *slice* desaparecerá junto con sus atributos y las relaciones estructurales que lo involucraban. También se borrará todo lo que a él respecta en el resto del proyecto.

3.1.2.3.- Crear una Relación Estructural.

Cuando se desea crear una relación estructural, se debe clickear en el botón que indica la Figura 5.7 del editor de Diseño de Entidad, el puntero del mouse cambiará de forma (tomará forma de una flecha con doble sentido), y aquí el diseñador deberá clickear primero el *slice* origen de la relación a crear y luego el *slice* destino. Antes de clickear en el *slice* destino, el diseñador podrá anular la operación clickeando el botón derecho del mouse.

3.1.2.4.- Borrar una Relación Estructural.

Para borrar una relación estructural se debe clickear en la misma y luego presionar la tecla "Delete", la relación estructural se borrará.

3.1.2.5.- Propiedades de los Slices.

Las propiedades de los *slices* son: el nombre, los atributos y el nombre del *slice* distinguido que llamamos cabecera. El nombre por defecto se genera de la misma manera que para la entidad (por ejemplo "Slice 1"). Inicialmente no tiene atributos y el *slice* cabecera por defecto es el primer *slice* que se creó.

Para actualizarlas se debe clickear en él con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.8. En esta pantalla el diseñador puede modificar el nombre del *slice* en (1), para que luego este aparezca en el botón del *slice* en el editor de diseño de Entidad.

Automáticamente se cargan en (2) los atributos de la entidad que se está diseñando. Luego el diseñador indica con el botón Agregar que atributos de (2) quiere como atributos del *slice*, éstos atributos se van cargando en (3).

En (4) el diseñador elige de los *slice* de la entidad, el que va a ser cabecera, es decir el que servirá como punto de entrada a la entidad durante la navegación.

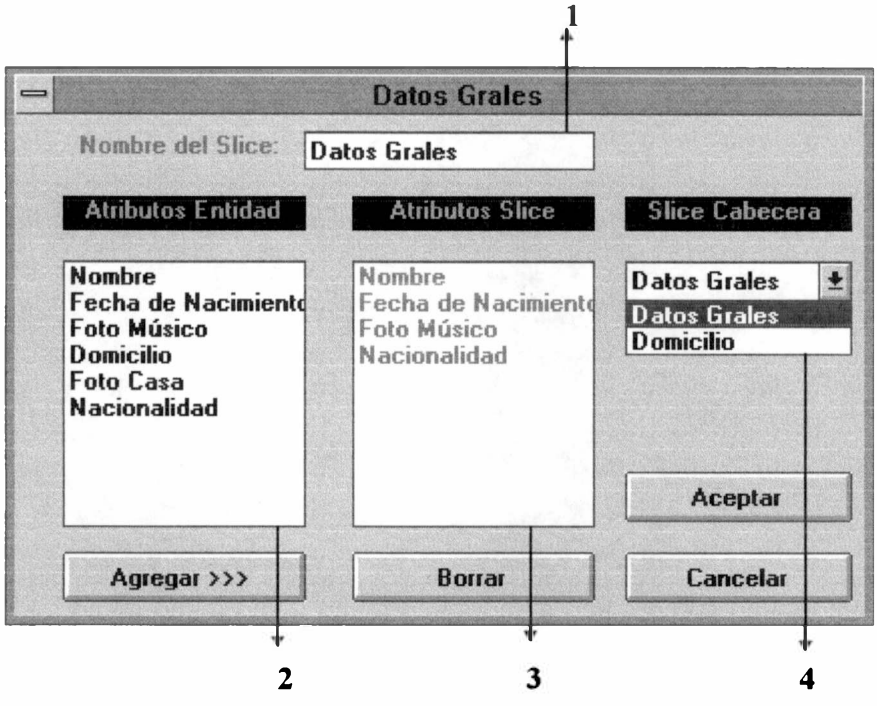


Figura 5.8: Pantalla de Propiedades del Slice.

1- Nombre del slice. 2- Lista de atributos de la entidad. 3- Lista de atributos del slice. 4- Lista de slice para elegir el cabecera.

Botón Aceptar: Acepta las modificaciones realizadas.

Botón Cancelar: Cancela la operación.

Botón Agregar: Agrega el atributo elegido en 2 a 3.

Botón Borrar: Borra el atributo especificado en 3.

3.1.2.6.- Propiedades de las Relaciones Estructurales.

Las única propiedad es el nombre que por defecto se define de la misma manera que en las relaciones asociativas (por ejemplo "Relación 1").

Para actualizar las propiedades de una relación estructural se debe clickear en ella con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.9. En esta pantalla el diseñador puede modificar el nombre de la relación estructural en (1), para que luego este aparezca en el editor de diseño de Entidad como se ve en la Figura 5.7.

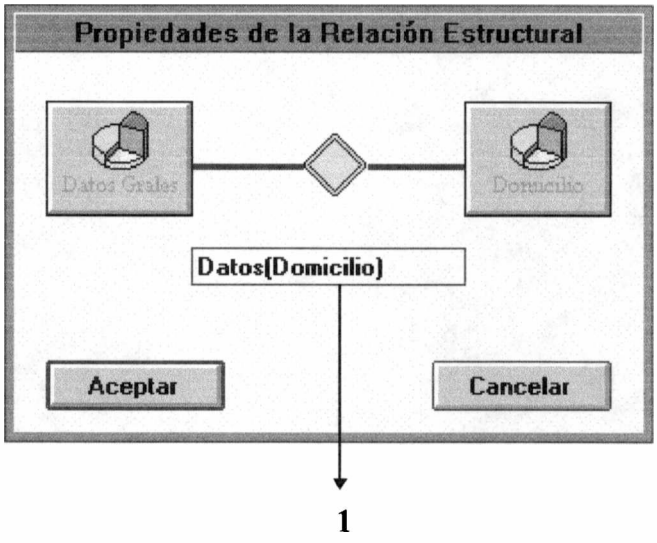


Figura 5.9: Pantalla de Propiedades de la Relación Estructural.

Botón Aceptar: Acepta las modificaciones realizadas.

Botón Cancelar: Cancela la operación.

3.1.3.- Editor de Diseño Navegacional.

Para realizar el Diseño Navegacional se debe hacer doble click en cualquier relación asociativa en el editor de Diseño E-R ó en el menú Proyectos clickear el submenú Diseño Navegacional. Este se construye a partir del Diseño E-R, tal es así que las modificaciones en la estructura del Diseño E-R ,como agregar o borrar una entidad o una relación asociativa, se refleja en el editor de Diseño Navegacional (no se refleja el cambio de posición de estos objetos). Inicialmente la ubicación de las entidades y las relaciones asociativas coinciden con la posición de creación, pero luego el diseñador puede reubicar los objetos porque se pueden crear *groupings* y *links* navegacionales debiéndose modificar el grafo.

Las principales operaciones que se pueden realizar en este editor son crear y borrar *groupings* y *links* navegaciones así como también asignar las estructuras de acceso a los *links* asociativos y navegaciones.

Con respecto a los *groupings*:

- El nombre del *grouping* puede ser modificado.
- Se determina cual va a ser el *grouping* cabecera de la aplicación.

Con respecto a los *links* navegaciones:

- El nombre del *link* puede ser modificado.
- Se asigna la estructura de acceso cuando este es entre *grouping* y entidad.
- En caso de que la estructura de acceso involucre un índice, se debe indicar que atributo significativo se mostrará en la lista de índices en la ejecución de la aplicación.
- Se puede asignar una condición que involucre a un atributo y a una constante cuando el *link* es entre un *grouping* y una entidad.

Con respecto a los *links* asociativos:

- Se asigna la estructura de acceso excepto que la aridad de la relación asociativa en la que se basa el *link* sea 1:1.
- En caso de que la estructura de acceso involucre un índice se debe indicar que atributo significativo se mostrará en la lista de índices en la ejecución de la aplicación

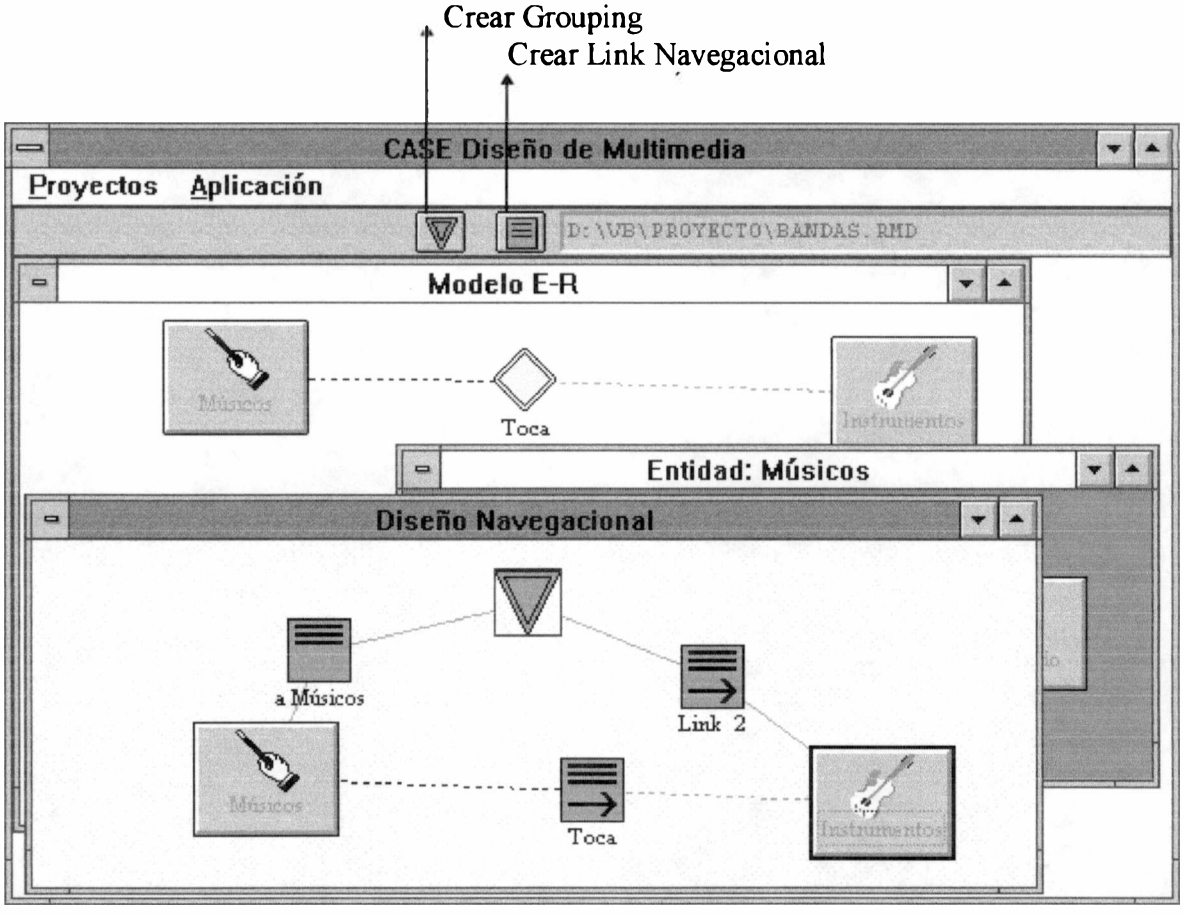


Figura 5.10: Editor de Diseño Navegacional.

En la Figura 5.10 observamos que el editor de Diseño Navegacional es la ventana que está activa, además los editores E-R y Diseño de Entidad están abiertos y pueden activarse sin inconveniente clickeando en cualquier porción de los mismos pasando a ser la ventana activa y los demás solo se verán abiertos. Esta es una gran facilidad que presenta RMCASE para que los diseñadores puedan observar conjuntamente los tres editores de diseño.

Nota: Para observar el editor de Diseño Navegacional con los detalles en color ver el Apéndice.

3.1.3.1.- Crear un Grouping.

Cuando se crea un *grouping* se debe clickear en el botón que indica la figura del editor de Diseño Navegacional, el puntero del mouse cambiará de forma (tomará forma de un cuadrado lleno de color negro), y aquí el diseñador deberá clickear dentro del editor en la

posición que desee ubicar al *grouping* creado. Antes de ubicar el *grouping* dentro del editor el diseñador podrá anular la operación clickeando el botón derecho del mouse.

3.1.3.2.- Borrar un Grouping.

Para borrar un *grouping* se debe clickear en el mismo y luego presionar la tecla "Delete", el *grouping* desaparecerá junto con los *links* navegaciones que lo involucraban.

3.1.3.3.- Crear un Link Navegacional.

Cuando se crea un *link* navegacional, se debe clickear en el botón que indica la figura del editor de Diseño Navegacional, el puntero del mouse cambiará de forma (tomará forma de una flecha con doble sentido), y el diseñador deberá clickear primero el *grouping* origen de la relación a crear y luego la entidad o *grouping* destino. Antes de clickear en el destino, el diseñador podrá anular la operación con el botón derecho del mouse.

3.1.3.4.- Borrar un Link Navegacional.

Para borrar un *link* navegacional se debe clickear en él y luego presionar la tecla "Delete", el *link* se borrará.

3.1.3.5.- Propiedades del Grouping.

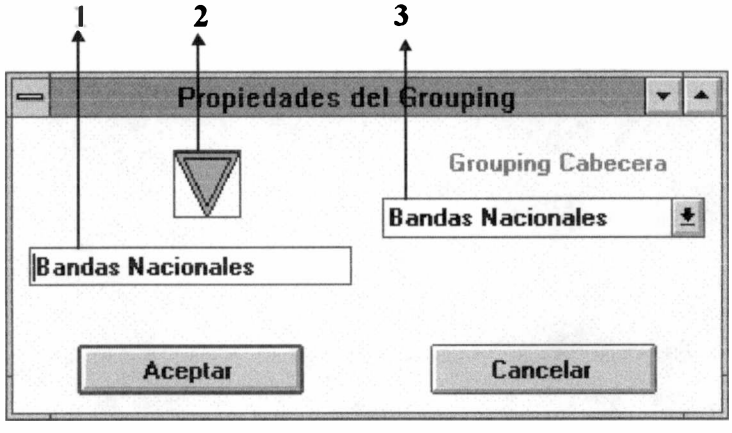


Figura 5.11: Pantalla de Propiedades de Grouping.

1- Nombre del Grouping. 2- Marca de grouping cabecera. 3- Lista de grouping para elegir el cabecera.

Para actualizar las propiedades de un *grouping* se debe clickear en él con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.11. En esta pantalla el diseñador puede modificar el nombre del *grouping* en (1).

En (2) el recuadro en el *grouping* indica que es cabecera. Por defecto el primer *grouping* que se crea es el cabecera.

En (3) el diseñador elige de los *grouping* de la aplicación que va a ser cabecera, es decir el que servirá como punto de inicio de la aplicación Hipermedia.

3.1.3.6.- Propiedades de los Links Navegaciones.

Las propiedades de los *links* navegaciones son: el nombre, y en el caso de *links* entre *grouping* y entidad también la estructura de acceso, la condición y el atributo índice solo para estructura de acceso indexada. Por defecto el nombre se construye de la misma manera que para las relaciones (por ejemplo "Link 1"); la estructura de acceso que se eligió por defecto es el tour guiado sin condición, porque de haber elegido alguna de las otras dos estructuras de acceso debíamos haber elegido un atributo índice por defecto y tal vez el atributo no hubiera sido el correcto.

Para actualizar las propiedades de un *link* navegacional se debe clickear en él con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.12. En esta pantalla el diseñador puede modificar el nombre del *link* en (4). Para establecer una condición en el *link* el diseñador debe indicarlo en (7) y luego llenar (1), (2) y (3). Para indicar que estructura de acceso va a tener el *link* debe elegirla en (5), una vez elegida esta será recuadrada. Si la estructura de acceso contiene un índice, en (6) se debe indicar cual será el atributo para la lista de índices.

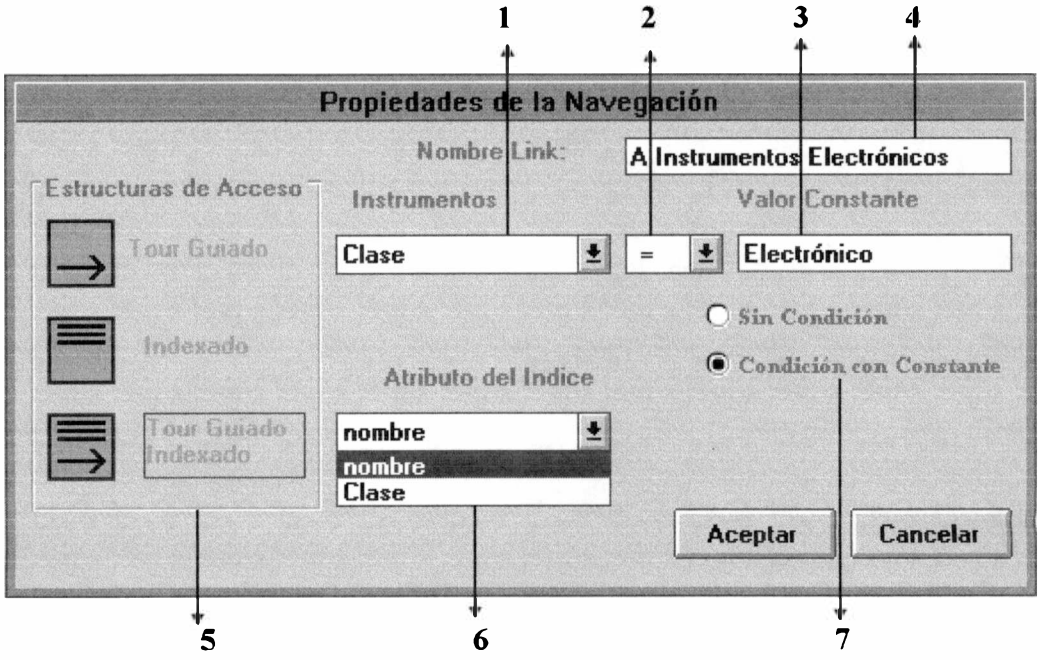


Figura 5.12: Pantalla de Propiedades de los Links Navegaciones.

1,2 y 3 Conforman la condición. 1- Atributo de la entidad destino. 2- Condicional. 3- Constante. 4- Nombre del link. 5- Lista de estructuras de acceso. 6- Lista de Atributos para elegir el atributo del índice. 7- Opción para elegir si el link contiene o no condición.

Botón Aceptar: Acepta las modificaciones realizadas.

Botón Cancelar: Cancela la operación.

3.1.3.7.- Propiedades de los Links Asociativos.

Las propiedades de los *links* asociativos son: el nombre, la estructura de acceso y el atributo índice si esta es indexada. Recordemos que los *links* asociativos derivaban de las relaciones asociativas entonces solo en el caso de que la aridad sea 1:N se permite la estructura de acceso; por defecto, igual que en los *links* navegaciones, la estructura es tour guiado. El nombre no se puede modificar porque es el de la relación asociativa.

Para actualizar las propiedades de un *link* asociativo se debe clickear en él con el botón derecho del mouse y se mostrará una pantalla como la de la Figura 5.13. Para indicar qué estructura de acceso va a tener el *link* debe elegirla en (1), una vez elegida esta será recuadrada. Si la estructura de acceso contiene un índice, en (2) se debe indicar cual será el atributo para la lista de índices.

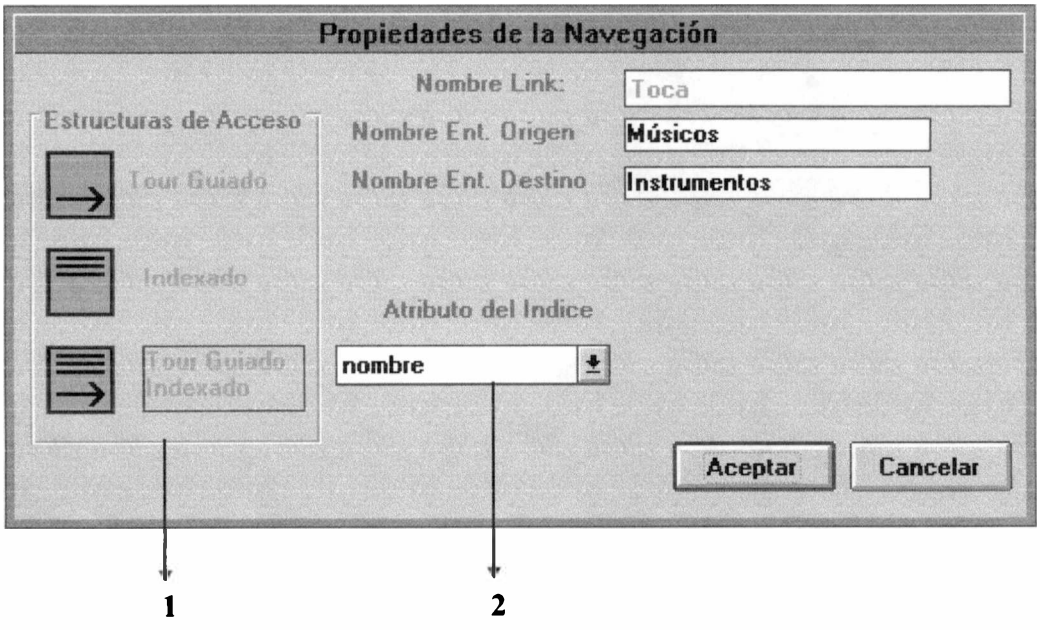


Figura 5.13: Pantalla de Propiedades de los Links Asociativos.

1- Lista de estructuras de acceso. 2- Lista de Atributos para elegir el atributo del índice.

Botón Aceptar: Acepta las modificaciones realizadas.

Botón Cancelar: Cancela la operación.

Todos los objetos que aparecen en los editores de diseño pueden ser ubicados en una nueva posición (*drag and drop*) dentro del editor al que pertenecen.

3.2.- Abrir un Proyecto.

El proyecto creado en 3.1. fue guardado con la opción Guardar del menú Proyectos con nombre Bandas.rmd, todos los proyectos se guardarán con la misma extensión ".RMD". Las opciones Guardar y Guardar Como del menú Proyectos son cajas de diálogos de Windows que no merecen mayor explicación ya que son usadas frecuentemente por los usuarios de Windows.

Cuando el diseñador activa la opción Abrir del menú Proyectos, eligiendo un archivo .RMD, automáticamente se mostrará el editor de Diseño E-R correspondiente al proyecto elegido como se muestra en la Figura 5.14.

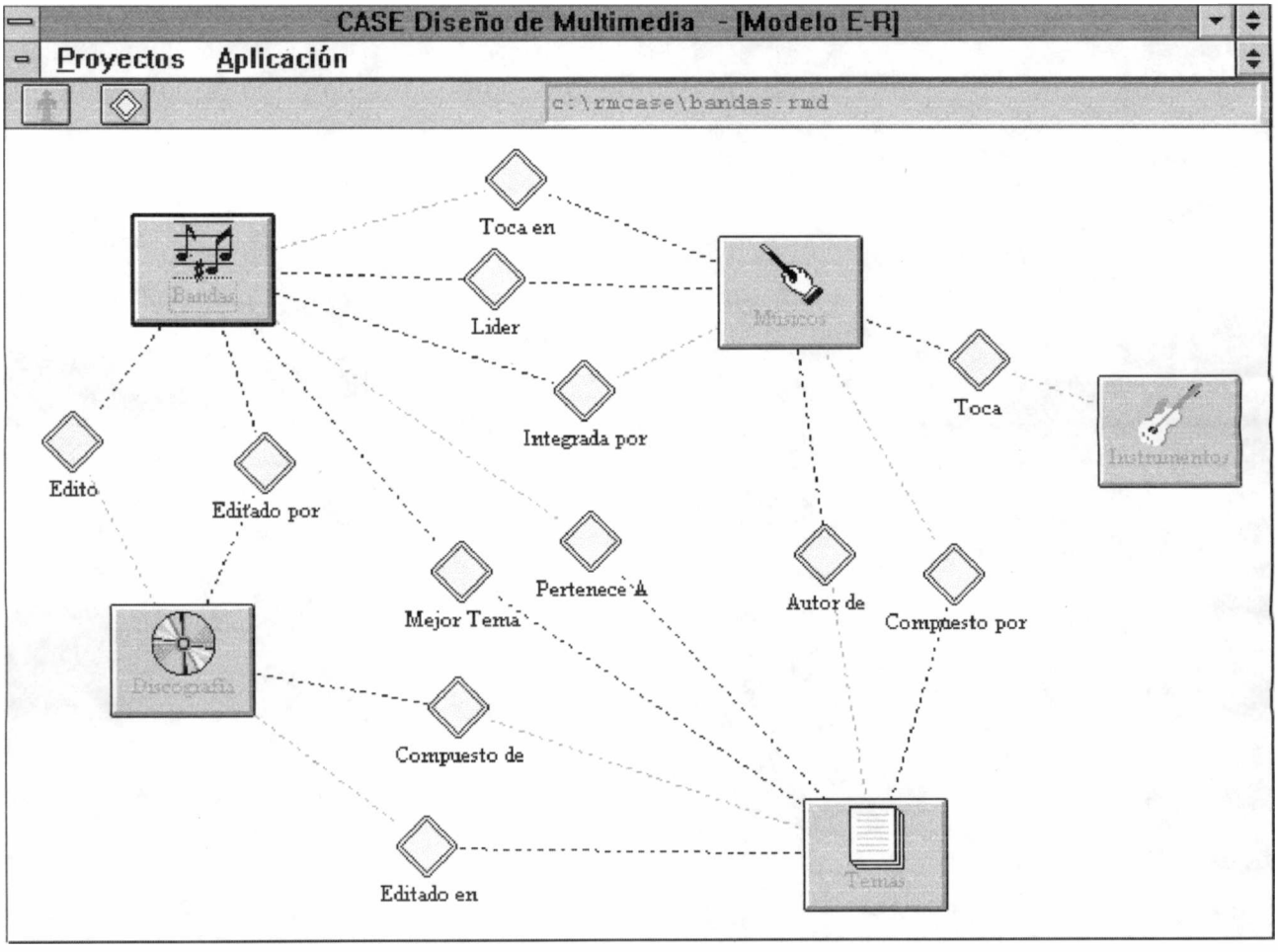


Figura 5.14: Pantalla del Diseño E-R del Proyecto Bandas Nacionales.

En la parte superior derecha de la figura se muestra el nombre del archivo que corresponde al proyecto, en este caso a bandas.rmd.

3.3.- Carga de Datos.

La carga de datos corresponde al menú Aplicación que consta de la carga de los datos de las instancias de las entidades y de los *groupings*.

3.3.1.- Carga de Datos de las Entidades.

En este punto el diseñador carga las instancias de las entidades, completando cada atributo. Esta pantalla como se ve en la Figura 5.15 tiene como subtítulo "Tipado de Atributos Nuevos", dado que aquí se les asigna el tipo a los atributos nuevos de la entidad antes de comenzar la carga de los mismos, ya que sino estos se asumen por defecto como de tipo Texto.

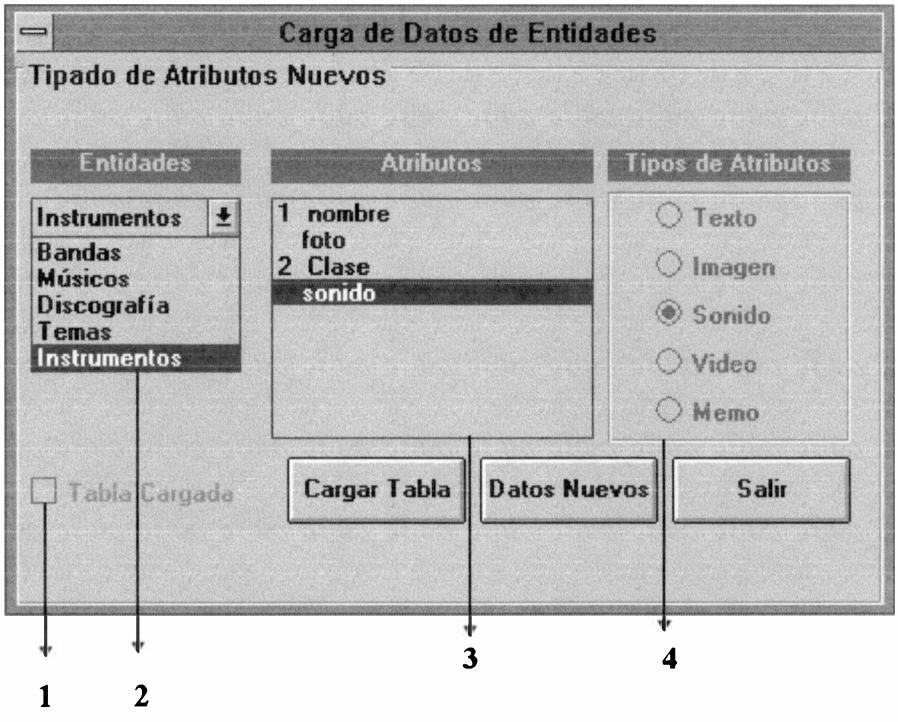


Figura 5.15: Pantalla de Carga de Datos de las Entidades.

1- Indica si la tabla de la entidad elegida en (2) está cargada. 2- Lista de entidades para elegir la entidad a cargar. 3- Lista de atributos de la entidad elegida. 4- Indica el tipo del atributo elegido, y permite asignarle el tipo a un atributo nuevo.

Botón Salir: Sale de la operación.

Botón Datos Nuevos: Se cargan los atributos especificados en (3) para la entidad elegida en (2).

Botón Cargar Tabla: Se cargan los atributos de la entidad que pertenecen a una tabla de una base de datos externa directamente.

Para realizar la carga de los datos de las entidades se elige primero la entidad a cargar en (2), inmediatamente en (3) se listarán los atributos de la misma y en (1) se indicará si la tabla ya está cargada. Si la entidad elegida tiene atributos nuevos el diseñador deberá indicar a cada atributo de que tipo es en (4), si en cambio el atributo está asociado a una tabla de una base de datos externa el tipo del mismo es el que indica la tabla y no se puede cambiar.

Cuando la entidad elegida tiene atributos asociados a una tabla de una base de datos externa las instancias se cargan directamente desde la tabla clickeando el botón Cargar Tabla. Si la entidad tiene algún atributo que no está asociado a una tabla de base de datos externa para cargarlos se debe clickear el botón Datos Nuevos.

3.3.1.1.- Carga de Datos Nuevos de las Entidades.

Para la carga de los datos nuevos se presentará una pantalla con los atributos de la entidad como se muestra en la Figura 5.16 con la entidad "Instrumentos".

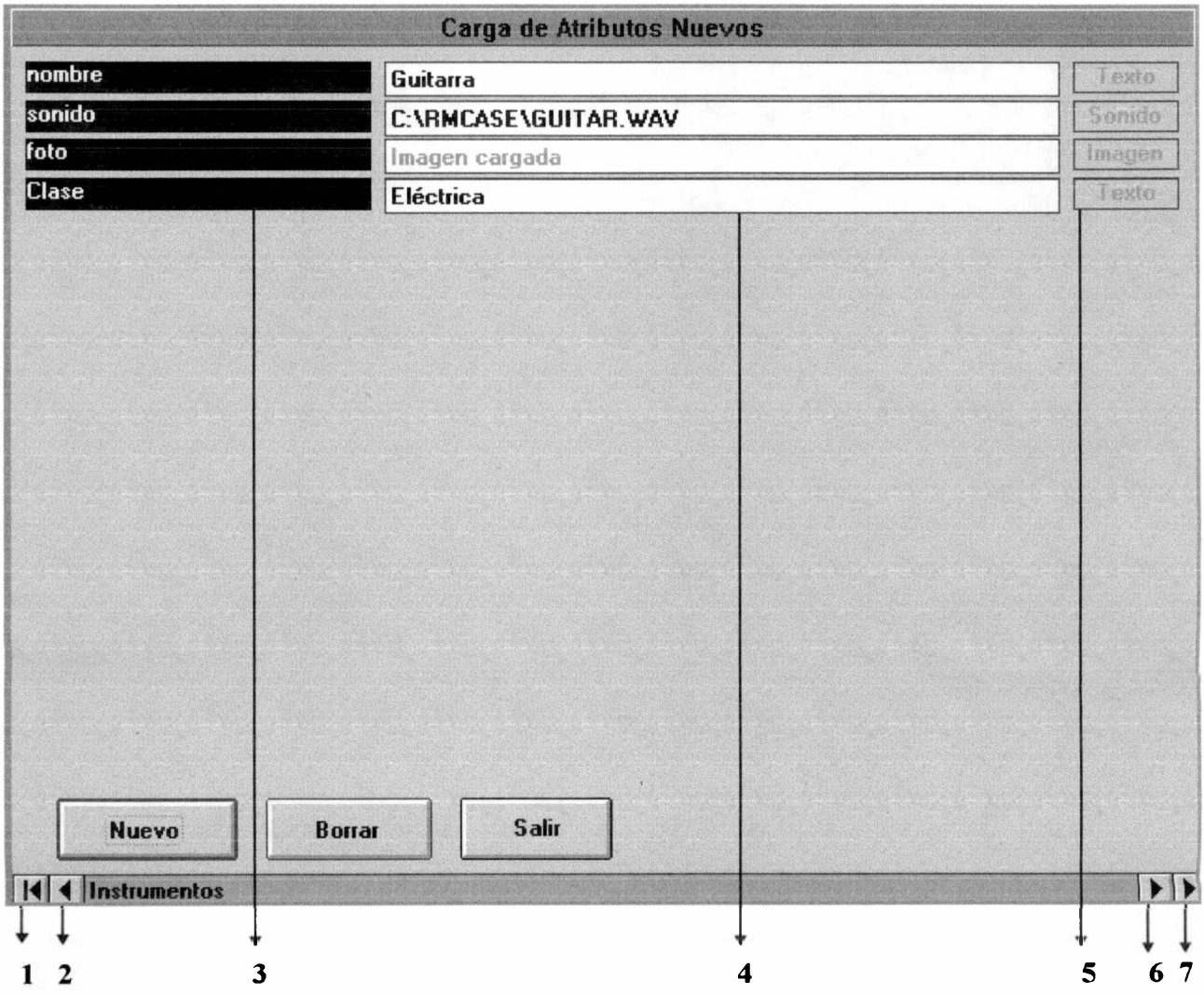


Figura 5.16: Pantalla de Carga de los Atributos Nuevos de las Entidades

1- Permite ubicarse en la primer instancia de la entidad. 2- Permite ubicarse en la instancia anterior a la actual (si la actual no es la primera). 3- Nombre de los atributos. 4- Contenido de los atributos. 5- Tipo de los atributos. 6- Permite ubicarse en la instancia siguiente a la actual (si la actual no es la última). 7- Permite ubicarse en la última instancia de la entidad.

Botón Nuevo: Permite agregar una instancia con los atributos vacíos a la entidad.

Botón Borrar: Permite borrar una instancia.

Botón Salir: Sale de la carga de datos nuevos.

Para entender como se cargan los distintos tipos de atributos a continuación explicamos que operaciones realizar con cada tipo de atributo.

a.- Tipo Memo: En (4) se especifican las primeras letras del memo. Con un click en (5) se abre una pantalla donde se trata el atributo memo. (Ver Figura 5.18).

b.- Tipo Imagen: En (4) se especifica si la imagen está cargada o no. Con un click en (5) se abre una pantalla donde se trata el atributo imagen. (Ver Figura 5.17).

c.- Tipo Texto: Con un click en (4) se puede modificar.

d.- Tipo Sonido: Con un click en (4) se puede entrar el nombre completo del archivo sonido. El formato de sonido que maneja RMCASE es el correspondiente a los archivos con extensión ".WAV". Con un Doble click en (4) se abre una caja de diálogo para elegir el archivo con las características anteriores. Con un click en (5) se ejecuta el archivo de sonido cargado en (4).

e.- Tipo Video: Con un click en (4) se puede entrar el nombre completo del archivo video. El formato de video que maneja RMCASE es el correspondiente a los archivos con extensión ".AVI" (que manejan audio y video simultáneamente). Con un Doble click en (4) se abre una caja de diálogo para elegir el archivo con las características anteriores. Con un click en (5) se ejecuta el archivo de video cargado en (4).

El diseñador puede agregar nuevas instancias de la entidad clickeando el botón Nuevo y se mostrara la nueva instancia con los atributos vacíos; también se puede borrar la instancia actual clickeando el botón Borrar. Llamamos instancia actual a la instancia que se esta viendo actualmente en pantalla. El diseñador puede moverse a la instancia anterior a la actual clickeando en (2), si la actual no es la primer instancia de la entidad; a la instancia siguiente de la actual clickeando en (6), si ésta no es la última instancia de la entidad; a la primer instancia clickeando en (1) y a la última clickeando en (7).

3.3.1.1.1.- Carga de Imágenes.

Luego de clickear en (5), en un atributo imagen de la pantalla de la Figura 5.16, se muestra la pantalla de la Figura 5.17 para cargar o borrar imágenes. Aquí el diseñador puede cargar una imagen desde un archivo .BMP clickeando el botón "Traer Archivo" o desde el portapapeles de Windows clickeando el botón "Pegar Imagen"; también se puede borrar una imagen ya cargada clickeando el botón "Borrar".

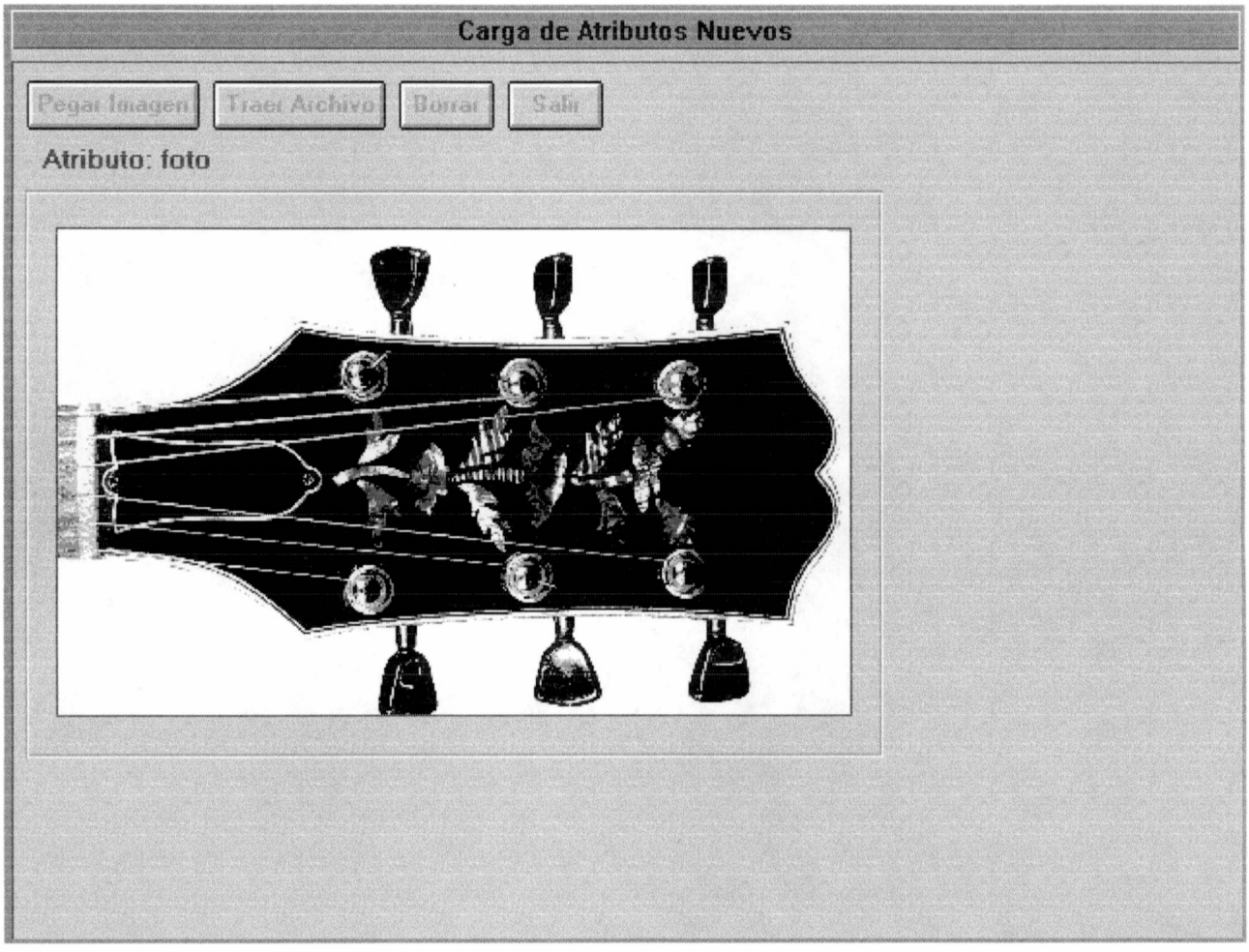


Figura 5.17: Pantalla de Carga de Imágenes.

Botón Pegar Imagen: Permite cargar una imagen desde el portapapeles de Windows.

Botón Traer Archivo: Permite cargar una imagen desde un archivo .BMP.

Botón Borrar: Borra la imagen que está cargada.

Botón Salir: Vuelve a pantalla de los atributos.

3.3.1.1.2.- Carga de Memos.

Luego de clickear en (5), en un atributo memo de la pantalla de la Figura 5.16, se muestra la pantalla de la Figura 5.18 para cargar o borrar el texto de los campos memos. Aquí el diseñador puede cargar el texto de un memo desde un archivo .TXT clickeando el botón "Traer Archivo" o desde el portapapeles de Windows clickeando el botón "Pegar Texto"; también se puede borrar un campo memo ya cargado clickeando el botón "Borrar".

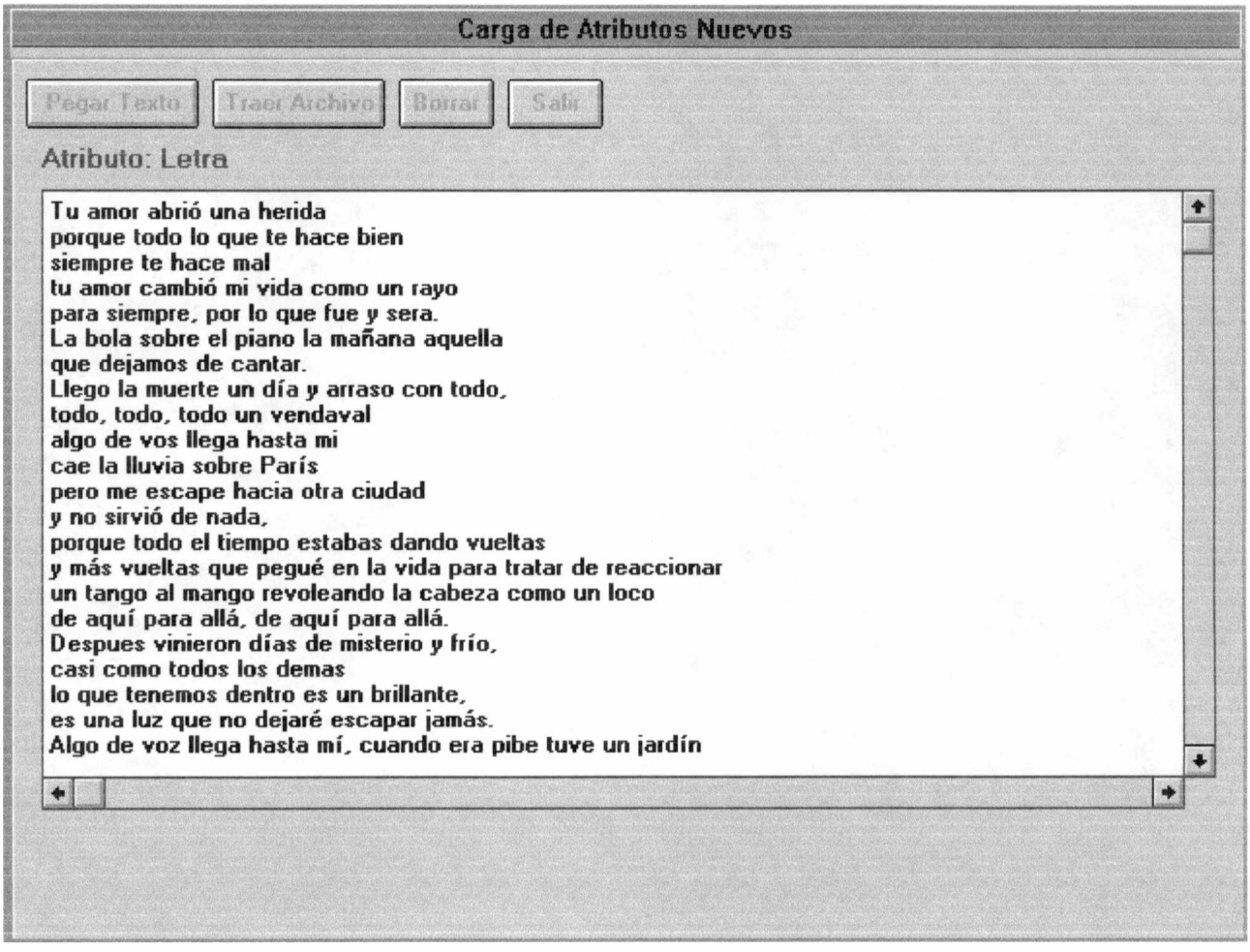


Figura 5.18: Pantalla de Carga de Memos.

Botón Pegar Texto: Permite cargar un texto desde el portapapeles de Windows.

Botón Traer Archivo: Permite cargar un texto desde un archivo .TXT.

Botón Borrar: Borra el texto que está cargado.

Botón Salir: Vuelve a pantalla de los atributos.

3.3.2.- Carga de Datos de los Groupings.

Los datos de los *groupings* son los atributos que definimos por defecto en la sección 4. Dado que un *grouping* es un menú donde se eligen opciones, corresponde entonces a una pantalla de presentación visualizando opciones y sus atributos. Nosotros definimos un atributo de cada tipo como se describe en el punto (2) de la Figura 5.19; obviamente para todos los *groupings* definidos en el proyecto se tienen los mismos atributos que pueden no utilizarse. En el punto (1) de la figura aparece la lista de *groupings* que hay en el Diseño Navegacional, en este caso hay dos *groupings*.

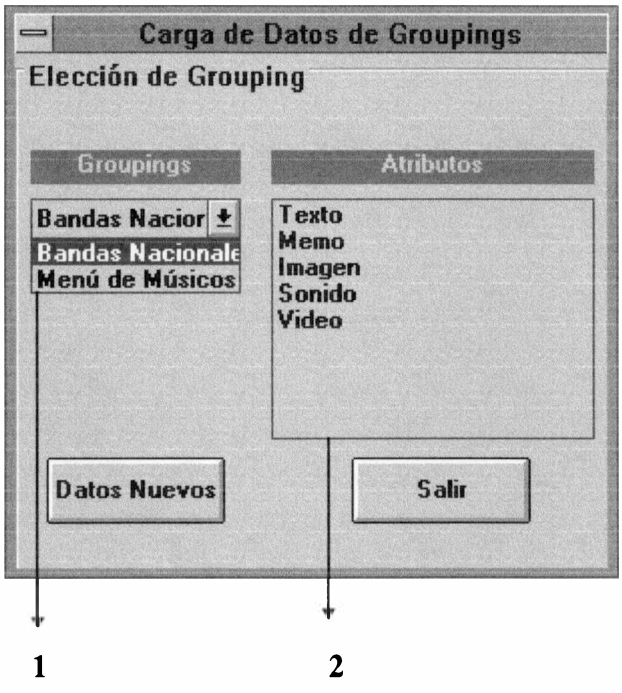


Figura 5.19: Pantalla de Carga de Datos de Groupings.

Botón Salir: Sale de la operación.

Botón Datos Nuevos: Se cargan los atributos de (2) para el grouping elegido en (1).

Esta pantalla se utiliza para elegir el *grouping* donde se cargarán los atributos correspondientes luego de presionar el botón Datos Nuevos.

3.3.2.1- Datos Nuevos de los Groupings.

Luego de clickear en el botón Datos Nuevos de la pantalla de la Figura 5.19 se presentará la pantalla de la Figura 5.20 para cargar los datos de los atributos.

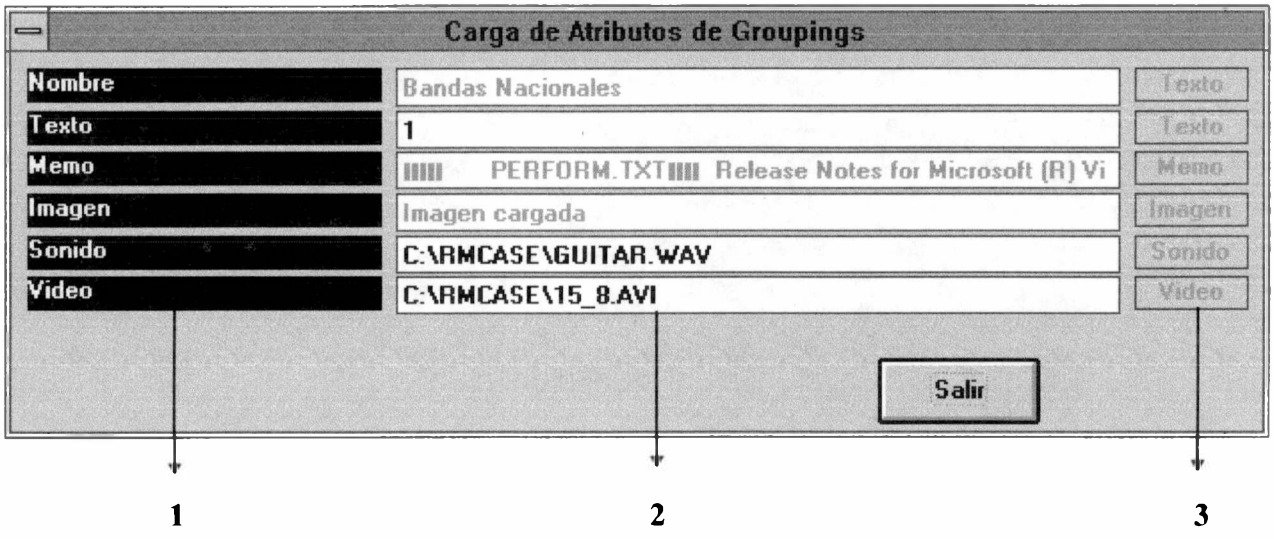


Figura 5.20: Pantalla de Atributos de Groupings.

1- Nombre de los atributos del grouping excepto el primero que es el nombre del Grouping. 2- Contenido de los atributos. 3- Tipo de los atributos.

En (1) se presentan los nombres de los atributos, en (2) el contenido y en (3) el tipo de los mismos. La primer fila corresponde al nombre del *grouping* y las siguientes a los atributos. Los atributos tienen distinto tratamiento según el tipo especificado:

a.- Tipo Memo: En (2) se especifican las primeras letras del memo. Con un click en (3) se abre una pantalla donde se trata el atributo memo. (Ver Figura 5.18).

b.- Tipo Imagen: En (2) se especifica si la imagen está cargada o no. Con un click en (3) se abre una pantalla donde se trata el atributo imagen. (Ver Figura 5.17).

c.- Tipo Texto: Con un click en (2) se puede modificar.

d.- Tipo Sonido: Con un click en (2) se puede entrar el nombre completo del archivo sonido. El formato de sonido que maneja RMCASE es el correspondiente a los archivos con extensión ".WAV". Con un Doble click en (2) se abre una caja de diálogo para elegir el archivo con las características anteriores. Con un click en (3) se ejecuta el archivo de sonido cargado en (2).

e.- Tipo Video: Con un click en (2) se puede entrar el nombre completo del archivo video. El formato de video que maneja RMCASE es el correspondiente a los archivos con extensión ".AVI" (que manejan audio y video simultáneamente). Con un Doble click en (2) se abre una caja de diálogo para elegir el archivo con las características anteriores. Con un click en (3) se ejecuta el archivo de video cargado en (2).

3.4.- Establecer Relaciones entre las Instancias de las Entidades.

Esta operación se realiza accediendo a la opción Establecer Relaciones del menú Aplicación, y consiste en indicar por cada relación asociativa que instancias de la entidad destino van a ser accedidas en ejecución desde cada instancia de la entidad origen al atravesar la relación asociativa en cuestión.

Para establecer que instancias estarán asociadas a tiempo de ejecución se debe elegir primero la relación a instanciar de la lista (2) de la Figura 5.21 donde están todas las relaciones asociativas de la aplicación Hipermedia, inmediatamente se mostrará en (4) la aridad de la relación elegida, en (1) y (3) se mostrarán los nombres de las entidades origen y destino de la relación respectivamente, en (5) y (7) se mostrarán las claves de las instancias cargadas en el punto Carga de Datos de Entidades de la entidad origen y destino respectivamente y en (6) se mostrarán las relaciones entre instancias que ya se han establecido.

Una vez elegida la relación, para establecer que instancias estarán conectadas:

- Si la aridad es 1:N se elige una instancia de la entidad origen en (5), inmediatamente en (7) se marcan las instancias que ya están conectadas a la instancia origen y clickeando en una instancia sin conectar en (7) se siguen agregando conexiones, si la instancia ya estaba conectada al volverla a clickear se desconecta.

- Si la aridad es 1:1 solo se podrá elegir una instancia destino por cada instancia origen.

Cada vez que se clickea en una instancia origen, se pedirá confirmación de las conexiones establecidas con la instancia origen anterior, si se confirman, estas conexiones se agregan en la lista de conexiones resultantes en (6).

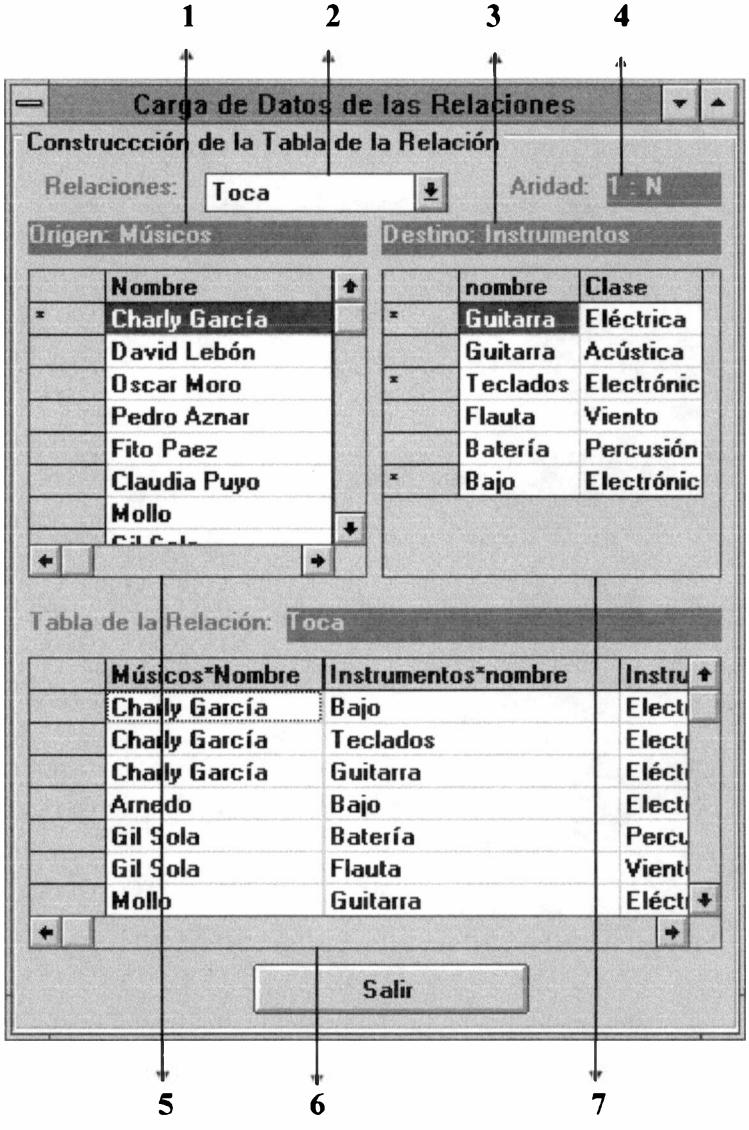


Figura 5.21: Pantalla para establecer las Relaciones Asociativas a Nivel de Instancias.

1- Entidad origen de la relación. 2- Lista de relaciones asociativas para elegir la que se va a cargar. 3- Entidad destino de la relación. 4- Aridad de la relación asociativa elegida en (2). 5- Lista de las claves de las instancias de la entidad origen de la relación. 6- Lista resultante de las relaciones establecidas entre la lista (5) y la lista (7) de instancias. 7- Lista de las claves de las instancias de la entidad destino de la relación.

Las marcas en las instancias conectadas como se ve en la Figura 5.21 se distinguen con un asterisco (*) en la primer columna de la lista de instancias correspondiente.

3.5.- Construcción del Prototipo.

En esta opción del menú Aplicación el diseñador construye los caminos navegaciones diseñados en los pasos descriptos anteriormente. Previamente a la construcción se realizan determinados testeos como para verificar que lo diseñado está en condiciones de ser ejecutado (por ejemplo controla que cada entidad tenga al menos un (1) *slice*, que la aplicación tenga al menos un *grouping*, que las entidades tengan atributos, que exista al menos una relación asociativa, etc.), en caso de que algunos de los testeos no sean satisfactorios en la pantalla de la Figura 5.22 se mostrará el error que guiará al diseñador a corregirlo, si todos los testeos son satisfactorios la construcción finalizará con la pantalla de la Figura 5.22.

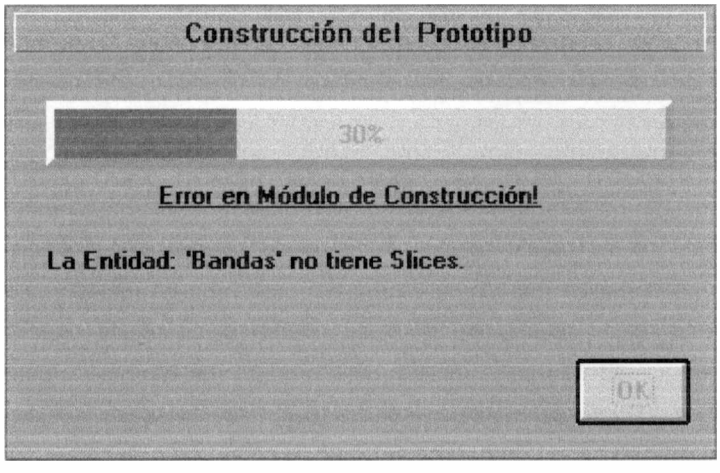


Figura 5.22-a: Pantalla de Construcción del Prototipo con error.

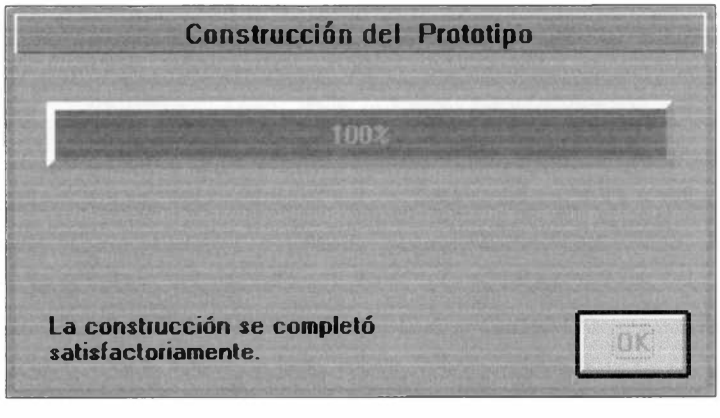


Figura 5.22-b: Pantalla de Construcción del Prototipo.

3.6.- Ejecución.

En esta opción el diseñador testea y evalúa la ejecución del prototipo según lo desarrollado en las etapas anteriores. Para mostrar como manejar el prototipo especificaremos dos pantallas características del mismo, que muestran la información de todas las opciones que se manejan. La presentación y distribución de los datos en la pantalla está predefinido (salida estándar) ya que RMCASE no facilita el diseño de la interface.

Cuando se ejecuta el prototipo, la primer pantalla que se muestra es la del *grouping* cabecera definido en pasos anteriores; en la Figura 5.23 se muestra la misma según el ejemplo que presentamos. En (1) se muestra la lista de los posibles *grouping* (menúes) que pueden ser alcanzados desde este punto y en (2) se muestra la lista de las entidades que pueden ser alcanzadas. La definición de un *grouping* constaba de atributos predefinidos de tipo estándar: texto, memo, imagen, sonido y video que se ven en los puntos (3), (9), (8), (4) y (5) respectivamente.

El punto (6) indica que estamos parados en una pantalla de *grouping* cuyo nombre se especifica en (7).

Cliqueando en (1) ó (2) el usuario de la aplicación puede navegar hacia otro contexto (*grouping* o entidad).

En la Figura 5.24 se muestra otra pantalla del prototipo que corresponde a un *slice* de una entidad, donde se pueden observar otras características diferentes de la Figura 5.23.

En (1) y (4) existe una correspondencia, ya que en (4) se muestra una lista de todos los atributos de tipo texto de esta pantalla (esto ocurre con todos los tipos de atributos cuando existe más de un atributo del mismo tipo en una misma pantalla) y en (1) se muestra el contenido del atributo de tipo texto elegido en (4). En (2) se muestra un atributo de tipo imagen, en este caso como existe un solo atributo de este tipo no es necesario una lista asociada al tipo imagen.

En (3) se muestra una lista de todas las relaciones asociativas que tienen como origen a la entidad actual (es decir la que se muestra en ese momento) y cuando se elige una relación de la lista (3) cambiamos el contexto a la entidad destino de la relación elegida, la entrada a una entidad es siempre el *slice* elegido como cabecera durante el diseño. Si la relación elegida en (3) tiene una estructura de acceso indexada (índice o tour guiado indexado), en (7) se mostrarán todas las posibles instancias de la entidad destino que podemos visitar para que el usuario elija la que desee; en (6) se indica el tipo de estructura de acceso indexada.

En (5) se muestra la lista de las relaciones estructurales de la entidad accesibles desde el *slice* actual. Al elegir una relación de (5) no se cambia de entidad pero sí de *slice*, cambiando los atributos que se muestran en la pantalla.

En la parte inferior de la pantalla se muestran los datos de ubicación del nodo dentro la red de Hipertexto. En (8) y (9) se muestran el ícono y el nombre respectivamente que se le asignó en el diseño a la entidad actual; en (10) se indica el nombre del *slice*.

Con (11) retrocedemos a la instancia anterior y con (12) avanzamos a la siguiente cuando la estructura de acceso es un tour guiado o un tour guiado indexado.

A medida que el usuario navega por la aplicación Hipermedia se van guardando en forma ordenada los puntos visitados, luego con (13) el usuario puede desandar el camino recorrido (*backtracking*) volviendo al contexto anterior.

Cuando el usuario se encuentra desorientado con respecto a su ubicación dentro de la red de Hipertexto le ofrecemos mediante (14) consultar la red de Hipertexto (el diseño navegacional del proyecto) y de ese modo saber en que lugar de la red está ubicado, tomando como referencia a (8) que es el ícono que identifica a la entidad o *grouping*; como en la red solo se muestran las entidades y *groupings*, clickeando en una entidad se muestra el diseño de entidad que incluye a los *slices* y las relaciones estructurales.

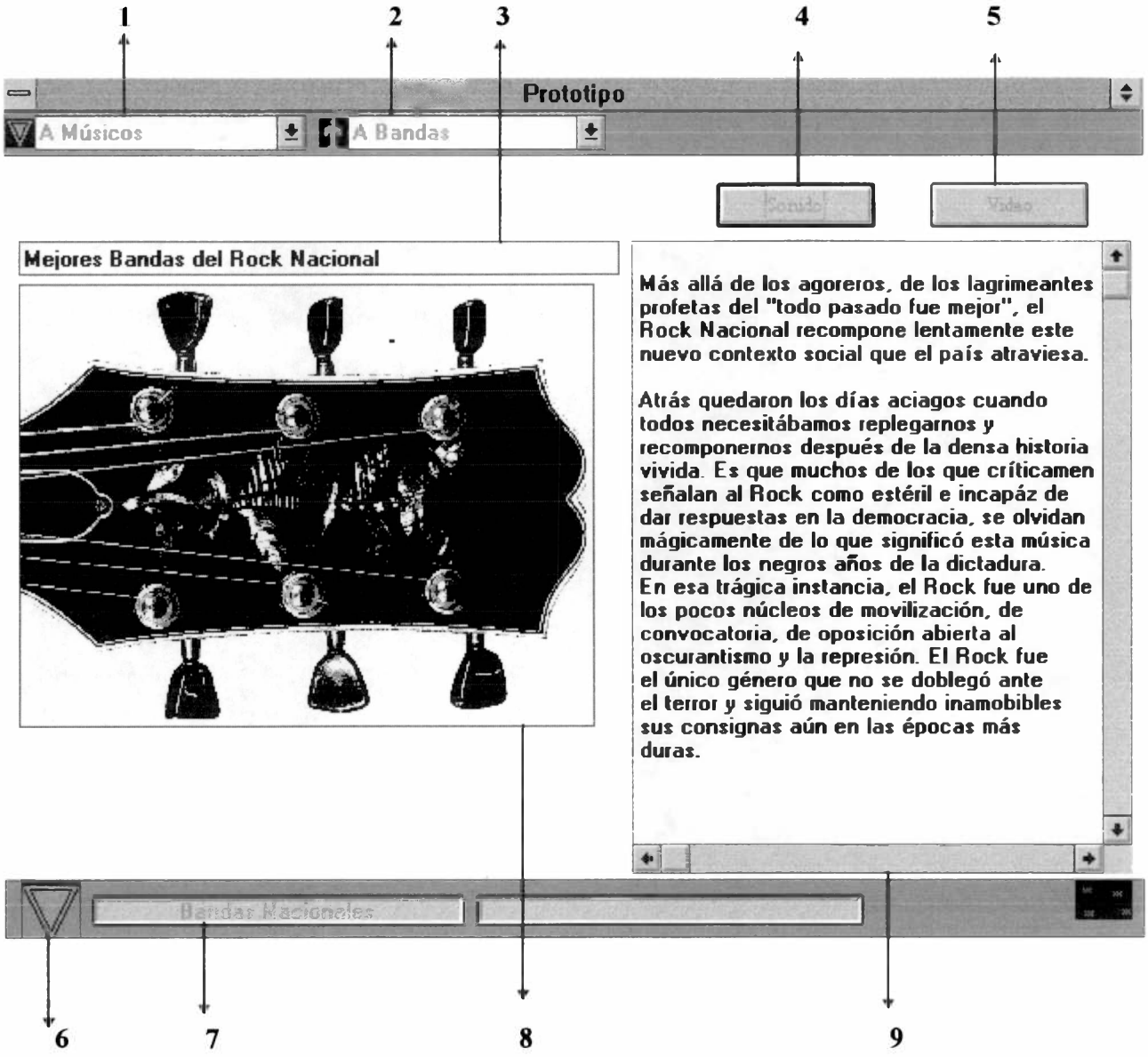


Figura 5.23: Pantalla Grouping.

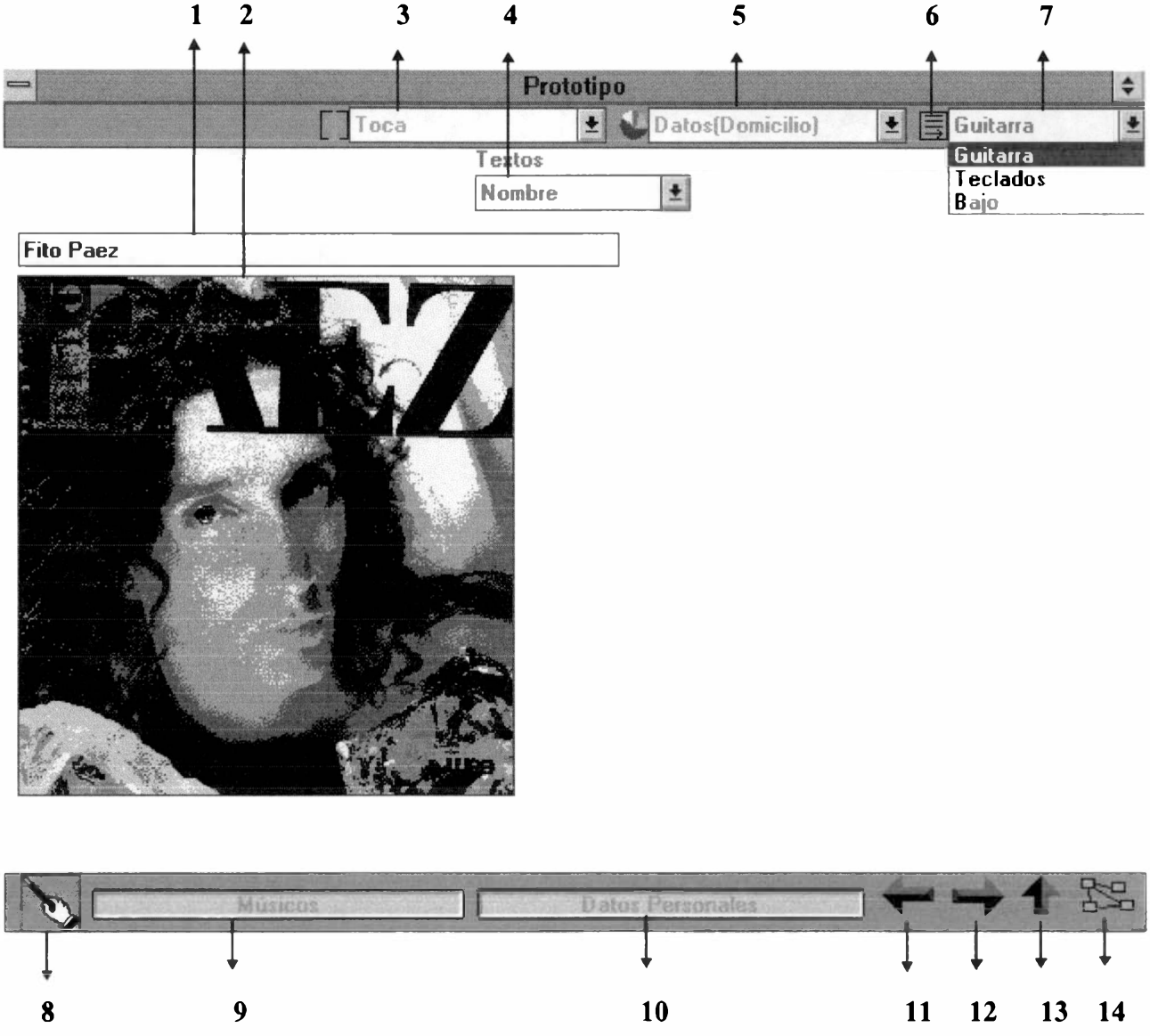


Figura 5.24: Pantalla Entidad.

Nota: Para observar el prototipo con los detalles en color ver el Apéndice.

4.- Conclusiones.

Inicialmente, se nos presentó el problema de desarrollar una aplicación Hipermedia cuyo dominio de la información sea altamente estructurado. Los datos que se requieren para tal aplicación pueden ser generados en ella misma, pueden ser transferidos desde bases de datos externas a la aplicación, o puede requerirse una forma combinada de ambos procesos. Se puede desarrollar utilizando las herramientas tradicionales para implementar aplicaciones Hipermedias (nótese que no hablamos del diseño de la aplicación sino de la implementación de la misma) tales como ToolBook en ambiente Windows ó HiperCard para el ambiente Macintosh, por citar las más difundidas, que trabajan perfectamente con datos u objetos multimediales organizándolos en una estructura secuencial tipo libro (página tras página), estas

no satisfacen netamente las necesidades de la implementación, ya que los objetos multimediales que pueden ser manipulados, además de los que se generan, deben ser traídos uno por uno para que puedan ser trabajados y ubicados correctamente en una página determinada.

Estas herramientas no son suficientes para desarrollar aplicaciones Hipermedias que trabajen o se basen en información externa (base de datos fuera de la herramienta)

La discusión se presentó de esta manera porque primero analizamos los datos que iba a requerir la aplicación y nos topamos de inmediato con el primer problema: las limitaciones para incorporar información ya existente en bases de datos. Sin embargo esto no es relevante teniendo en cuenta el problema principal que es el de diseñar la aplicación Hipermedia estructurada.

Sin reparar en el primer obstáculo nos encontramos luego con uno mayor ¿Cómo puedo diseñar una aplicación Hipermedia que esté estructurada de una manera no secuencial? La respuesta es que no se puede con las herramientas que se encuentran actualmente en el mercado. Nosotros tal vez pudiéramos desarrollar una aplicación Hipermedia estructurada específica que resuelve los problemas anteriores según los requerimientos de datos y la estructura de la aplicación. Pero volvemos con otro problema por que cada aplicación Hipermedia tiene sus propios requerimientos y su propio dominio estructurado en menor o mayor grado, por lo tanto no es suficiente el desarrollo individualizado sino que el tratamiento debería ser general para toda aplicación Hipermedia. Por lo expuesto, tuvimos que pensar en proveer un ambiente de desarrollo que me permita diseñar e implementar aplicaciones Hipermedias estructuradas. A esta altura se deduce que la respuesta al problema de desarrollar aplicaciones Hipermedias estructuradas es la construcción de un CASE que me permita diseñar e implementar este tipo de aplicaciones.

Antes de comenzar la construcción del CASE propiamente dicho, debimos pensar que modelo de datos y que metodología de diseño era la más adecuada para diseñar e implementar el tipo de aplicaciones con las características que nosotros deseábamos [ver Introducción]. Analizamos un modelo de datos HDM que carecía de metodología pero era apropiado para describir la estructura del dominio de aplicación. En el estudio de otros modelos encontramos que están basados fuertemente en HDM como OOHDM y RM y estos sí proveían una metodología de diseño. Las diferencias entre OOHDM y RM son substanciales porque el primero está íntegramente basado en diseño orientado a objetos y el segundo en el diseño de entidades y relaciones. Para modelar aplicaciones complejas el OOHDM sería más acertado porque el diseño se realiza de una manera natural, pero el modelo no está orientado para ser utilizado por diseñadores inexpertos y no cualquiera lo podría utilizar. Igualmente nuestra sensación es que el modelo OOHDM es mucho más de lo que pretendemos, en cambio RM que utiliza conceptos bien conocidos como entidades y relaciones que pueden modelar el tipo de aplicaciones que deseamos, cubre nuestras espectativas con respecto al modelo de datos y al especificar una metodología de diseño en forma detallada.

El diseño de hipertexto es una actividad humana compleja. Las técnicas de diseño formales mejoran la consistencia de los documentos hipermedias y provee guías precisas. Pero los factores humanos en el diseño de hipertexto son muy importantes también. Las guías de diseño son muy esenciales pero no suficientes, los *feedback loops* experimentales son vitales. RM soporta el proceso cognitivo de diseño completamente incluyendo los *feedback loops*, ventajas que nos hicieron considerar que el modelo RM es el más adecuado para basar el CASE a desarrollar.

Ya refiriéndonos al CASE desarrollado, cumplimos con las características fundamentales que propusimos: facilitamos un estado de diseño con diferentes niveles de abstracción: a nivel entidad (Diseño E-R), a nivel de *slices* (Diseño de Entidad) y a nivel de navegación (Diseño Navegacional). Cada uno de estos niveles cuenta con su propio editor especialmente diseñado para facilitar la creación y manejo de los objetos que en ellos se manejan. El cambio de contexto de diseño se realiza en forma más que fácil por la interconexión de los editores.

Con el prototipo terminado demostramos que es factible construir una única herramienta que permita diseñar y desarrollar una aplicación Hipermedia completa con las ventajas que presenta mantenerla, contando ya con el diseño y pudiéndolo actualizar con facilidad. La prototipación, que proveemos en RMCASE, posibilita experimentar la aplicación como eventualmente sería ejecutada y se pueden testear los diferentes aspectos del diseño de la aplicación Hipermedia, tal como su estructura de información y los patrones de navegación.

5.- Futuras Extensiones.

Las extensiones que se pueden realizar las dividimos en dos tipos: las conceptuales y las extensiones a la herramienta. Las futuras extensiones conceptuales son:

1) Proveer la posibilidad de navegar el prototipo según el tipo de usuario (por ejemplo si la aplicación Bandas Nacionales la navega un músico podemos incluir en la entidad Temas las notas correspondiente a cada instancia, esta información solo es importante para usuarios que entienden de música y no es relevante para otro tipo de usuario).
2) Construir un cuarto contexto de diseño de aplicación que se implemente en un editor de entidades y relaciones que permita diseñar un grafo donde cada entidad represente una aplicación RMCASE y cada relación un acceso entre dos aplicaciones (por ejemplo, la aplicación Bandas Nacionales puede constituir la entidad Música de una aplicación más general llamada Arte Nacional que incluirá también a las otras aplicaciones del arte argentino como Cine, Pintura, Escultura, Literatura y Oratoria. Luego se decidirá los accesos para navegar a través de las distintas aplicaciones).

Las futuras extensiones a la herramienta son:

1) Profundizar el manejo de base de datos como por ejemplo trabajar con distintos formatos (Dbase, Foxpro, Paradox, Btrieve, etc.) y ofrecer más opciones y facilidades para manejarlas.
2) Proveer un diseño de la interface para el prototipo, manteniendo un diseño por defecto como el que se provee actualmente que permite la construcción del prototipo sin la necesidad de elaborar un diseño. Además proveer la posibilidad de contar con varios diseños de interface para la misma aplicación, eligiendo el usuario el que desea.
3) Analizar la posibilidad de incluir condiciones en las estructuras de acceso de los *links* asociativos.
4) Incorporar la posibilidad de implementar una aplicación concreta, como por ejemplo generar código HTML a partir de un diseño de interface para ejecutar la aplicación directamente en World Wide Web.
5) Proveer un módulo de ejecución de la base de datos del proyecto (ver Sección 6) para facilitar la distribución comercial de la aplicación.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Sección 6: Arquitectura de RMCASE.

1.- Introducción.

En esta sección describiremos como es la arquitectura de RMCASE y algunos detalles de implementación presentes en el desarrollo de la herramienta.

RMCASE fue desarrollado bajo el lenguaje de programación Visual Basic for Windows. Fundamentalmente Visual Basic es un lenguaje orientado a eventos que provee facilidades para programar con características y objetos de Windows.

El manejador de base de datos que utilizamos es Microsoft Access. La característica principal de Access es que maneja a la base de datos como un archivo compuesto por tablas. También utilizamos el lenguaje de Query que provee Access para consultar más rápidamente las tablas.

2.- Arquitectura de RMCASE.

RMCASE maneja a cada aplicación Hipermedia como un proyecto. Un proyecto se implementa como una base de datos Access como se muestra la Figura 6.1. Las tablas que componen un proyecto se pueden dividir en dos grupos bien diferenciados; el primer grupo se refiere al de las tablas básicas que debe contener todo proyecto y el segundo grupo son las tablas de datos propias de cada proyecto en particular. Las tablas que componen el primer grupo son las siguientes:

1) Entidades: información de las entidades del diseño de Entidades y Relaciones (nombre, posición en el editor, clave, tabla externa asociada e ícono).
2) Relaciones Asociativas: información de las relaciones asociativas del diseño de Entidades y Relaciones (nombre, entidad origen y entidad destino involucradas en la relación, posición en el editor y aridad de la relación).
3) Atributos: información de los atributos de las entidades del diseño de Entidades y Relaciones (entidad y *slice* al que pertenece, nombre del atributo, si es un atributo clave, tipo del atributo y si es un atributo nuevo o de una tabla externa).
4) *Slices*: información de los *slices* de las entidades en el diseño de Entidad (entidad a la que pertenece, nombre del *slice*, posición en el editor y si es el *slice* cabecera de la entidad).
5) Relaciones Estructurales: información de las relaciones estructurales del diseño de Entidad (entidad a la que pertenece, *slices* que involucra, nombre, posición en el editor).
6) *Groupings*: información de los *groupings* del diseño Navegacional (nombre, posición en el editor, si es el *grouping* cabecera de la aplicación y los atributos de la pantalla de los *groupings*).

7) *Links* Navegacionales: información de los *links* navegacionales del diseño Navegacional (nombre, posición en el editor, *grouping* origen, *grouping* o entidad destino, tipo de estructura de acceso, condición y atributo de la estructura de acceso indexada).

8) *Links* Asociativos: información de los *links* asociativos del diseño Navegacional (nombre, posición en el editor, condición y atributo de la estructura de acceso indexada).

9) Entidades Navegacionales: información de las entidades en el diseño Navegacional (nombre y posición en el editor).

10) Navegación del Prototipo: esta tabla se genera en el módulo de construcción del prototipo del menú "Aplicación" y contiene la información de los todos los caminos posibles de navegación entre *groupings*, entidades y *slices* a través de los *links* correspondientes.

11) *Backtracking*: esta tabla se genera cuando se navega en el prototipo almacenando los estados del camino para luego contar con la información suficiente para poder desandarlo.

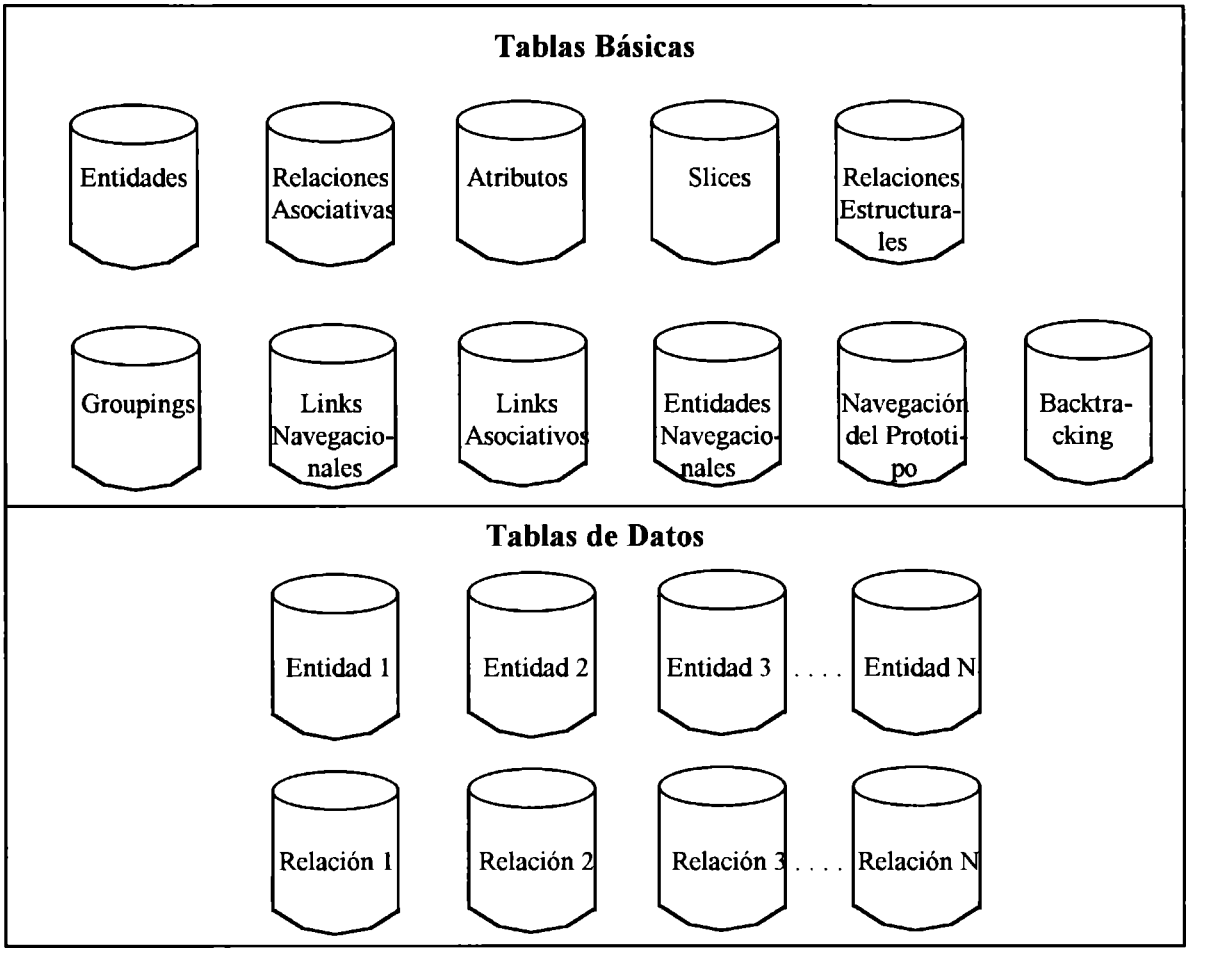


Figura 6.1: Estructura de la base de datos Access que se utiliza en cada proyecto.

El segundo grupo contiene la información de las tablas de datos correspondientes a las entidades y relaciones asociativas que se cargan en el contexto de población de hiperbase.

1) Cada tabla de datos de entidad contiene las instancias de una entidad del diseño de Entidades y Relaciones. Todas las operaciones de carga de datos de entidad se realizan sobre estas tablas (ver Sección 5: 3.3.1).

2) Las tablas de datos de las relaciones asociativas almacenan en cada registro las claves de la entidad origen y destino involucradas en la relación. Todas las operaciones de establecer relaciones a nivel de instancias se realizan sobre estas tablas (ver Sección 5: 3.4).

3.-Utilización de las tablas.

Se detallará a continuación las tablas que se utilizan en cada contexto de trabajo y el editor correspondiente:

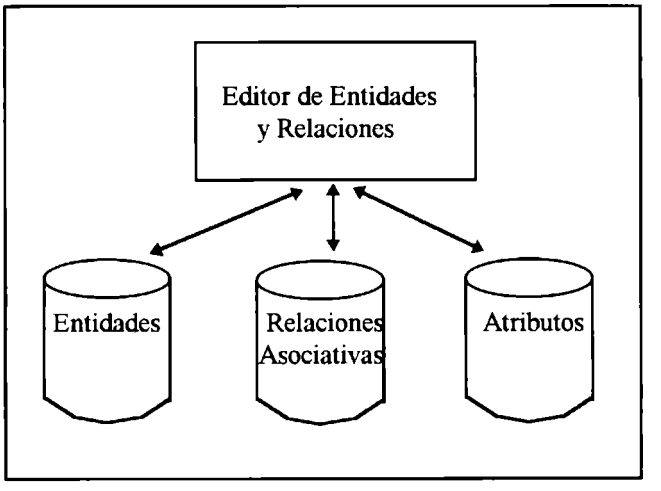


Figura 6.2: Contexto de diseño de Entidades y Relaciones.

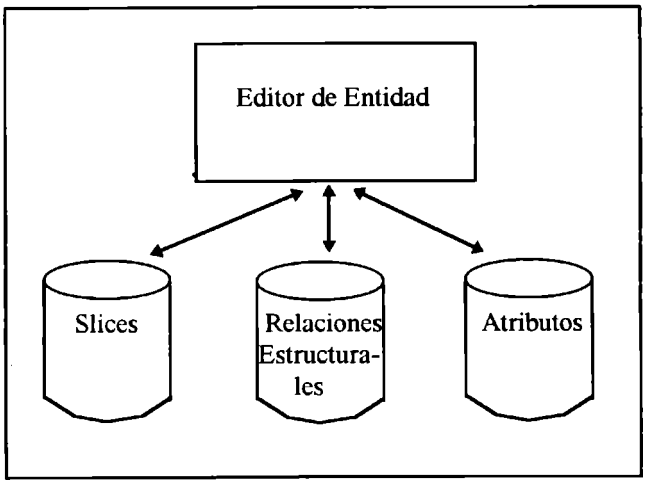


Figura 6.3: Contexto de diseño de Entidad.

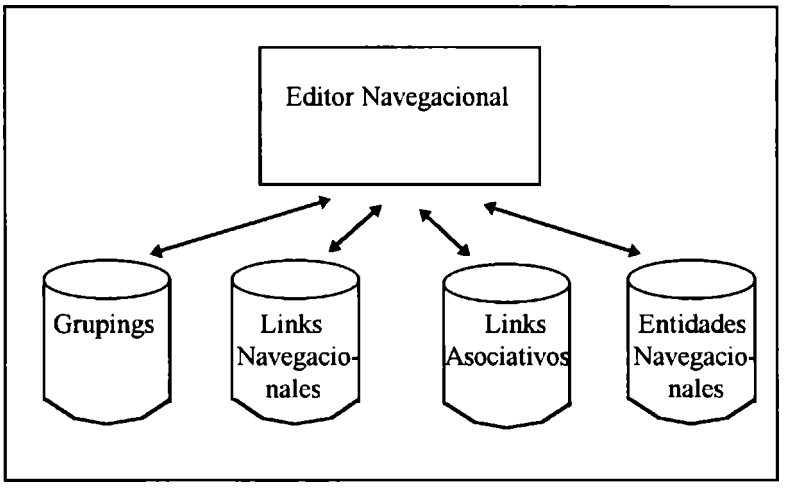


Figura 6.4: Contexto de diseño Navegacional.

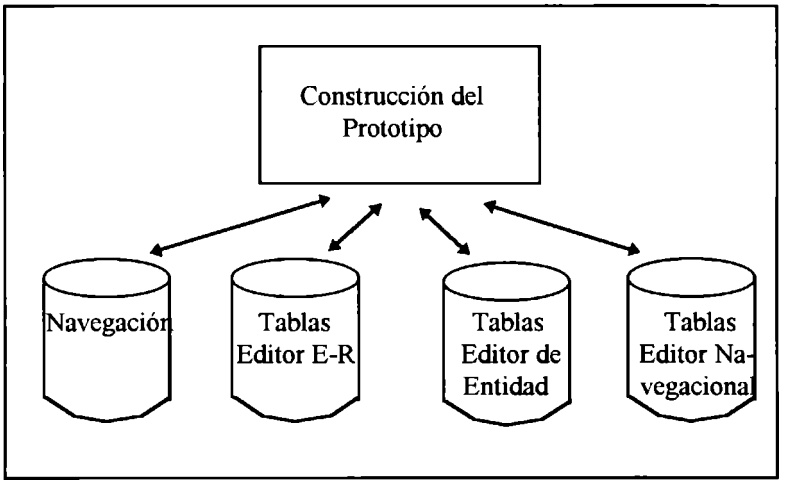


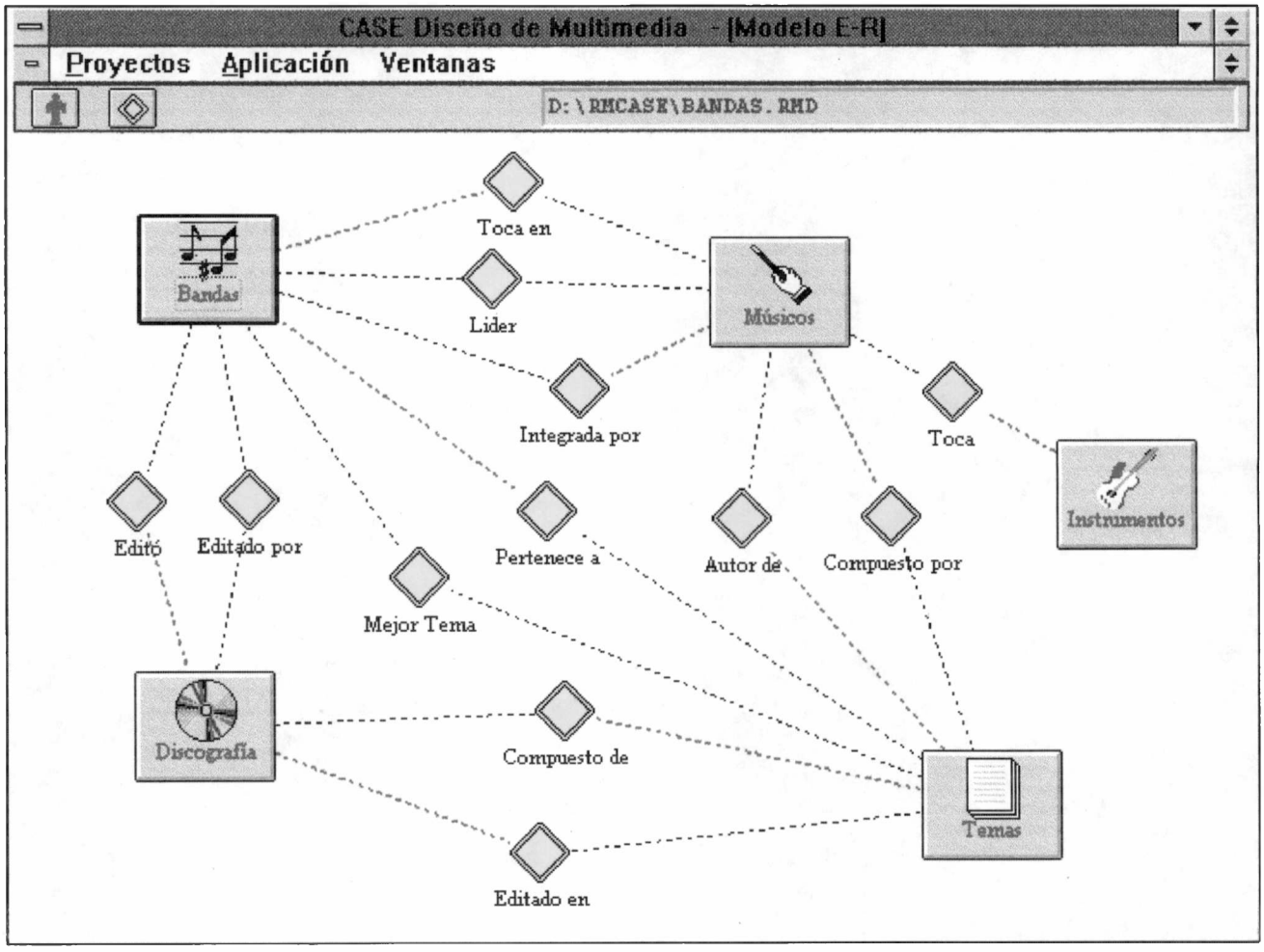
Figura 6.5: Contexto de Construcción del Prototipo.

Se debe destacar que cuando se ejecuta el prototipo y se navega por los distintos estados se genera la tabla de *Backtracking*. Este es el único momento en que se utiliza esta tabla y es usada para obtener la información de los estados anteriores cuando se quiera volver atrás en la navegación.

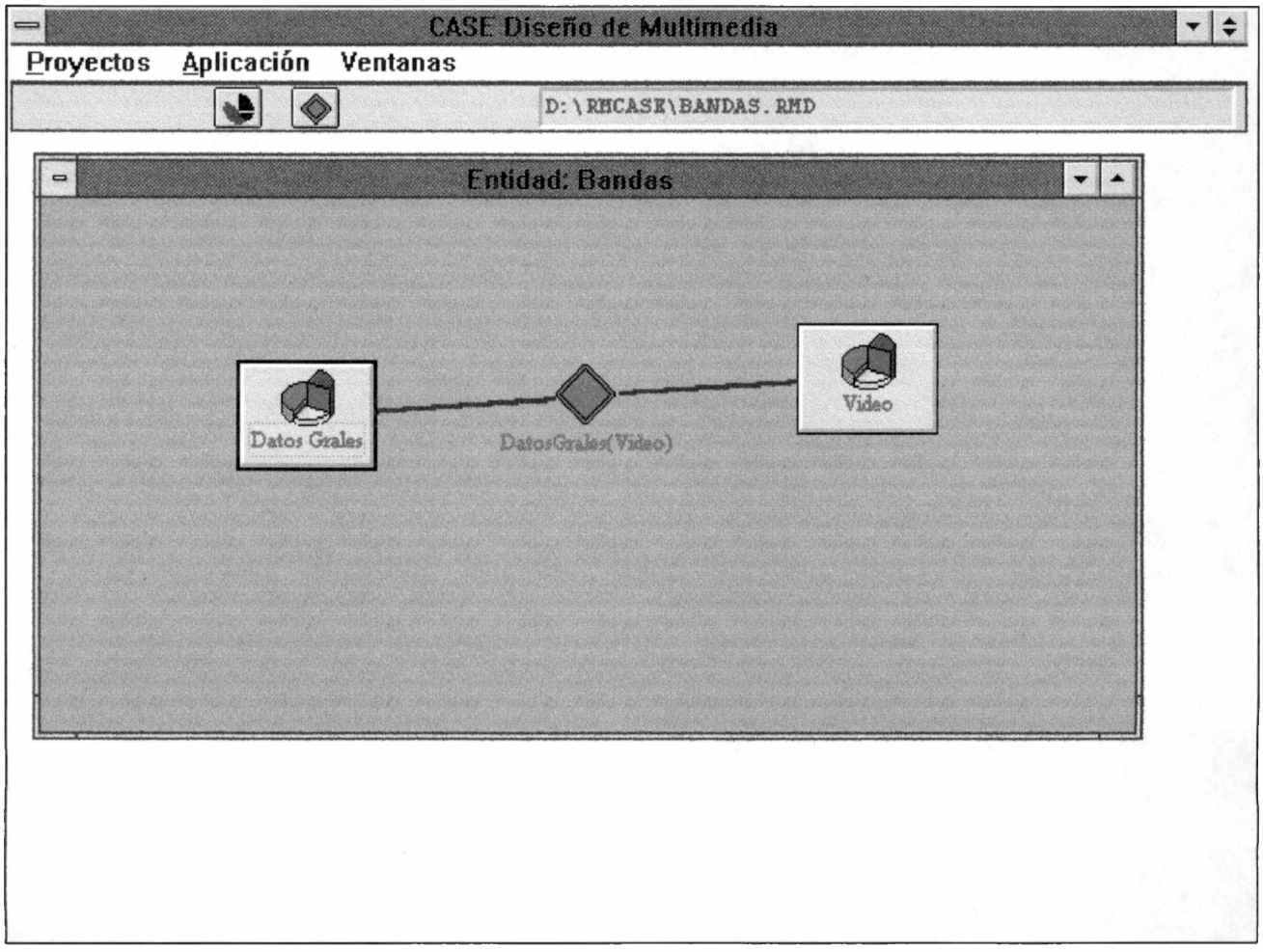
Apéndice:

Pantallas de RMCASE en color.

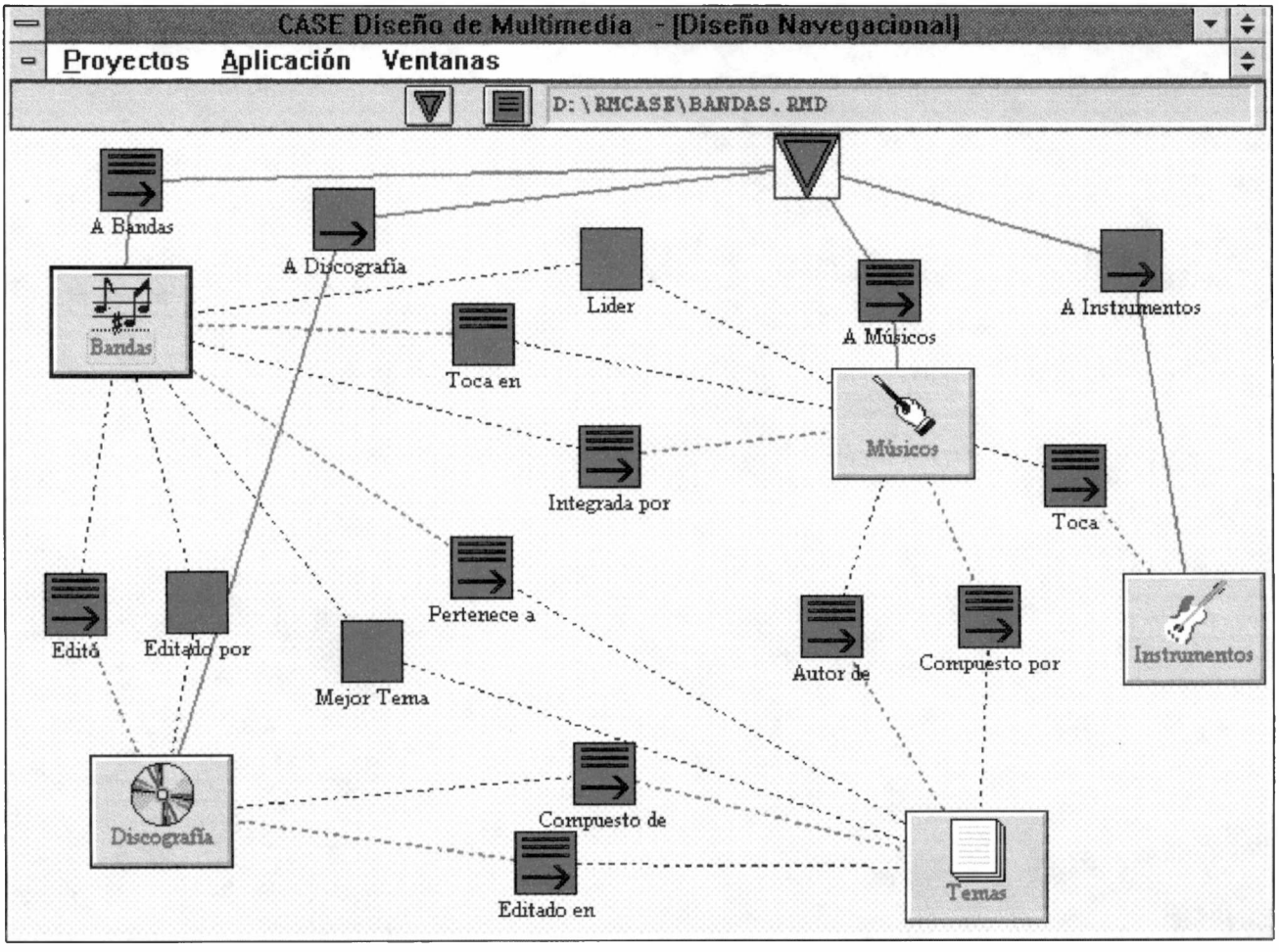
1.- Ejemplo del Editor E-R.



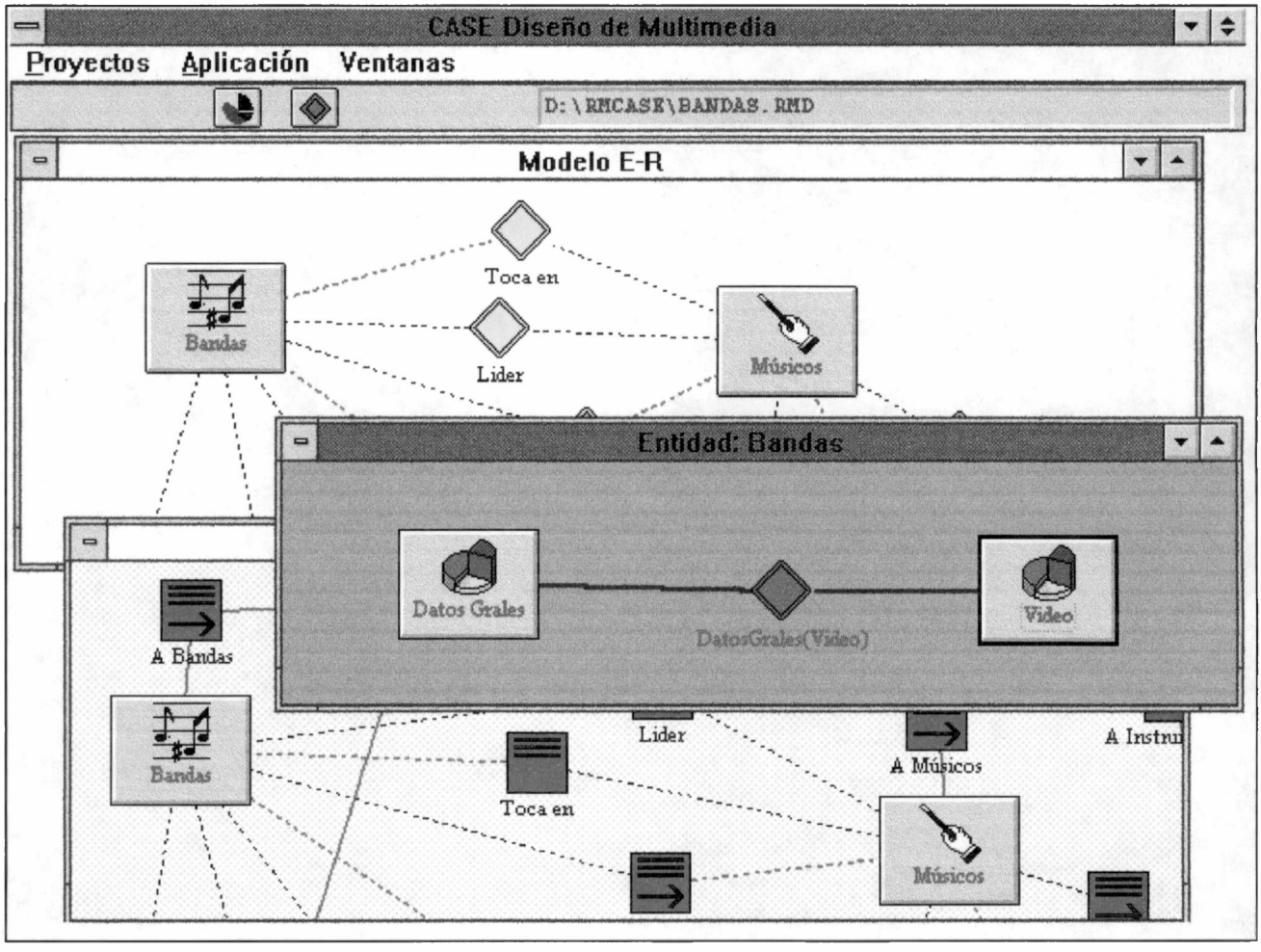
2.- Ejemplo del Editor de Diseño de Entidad.



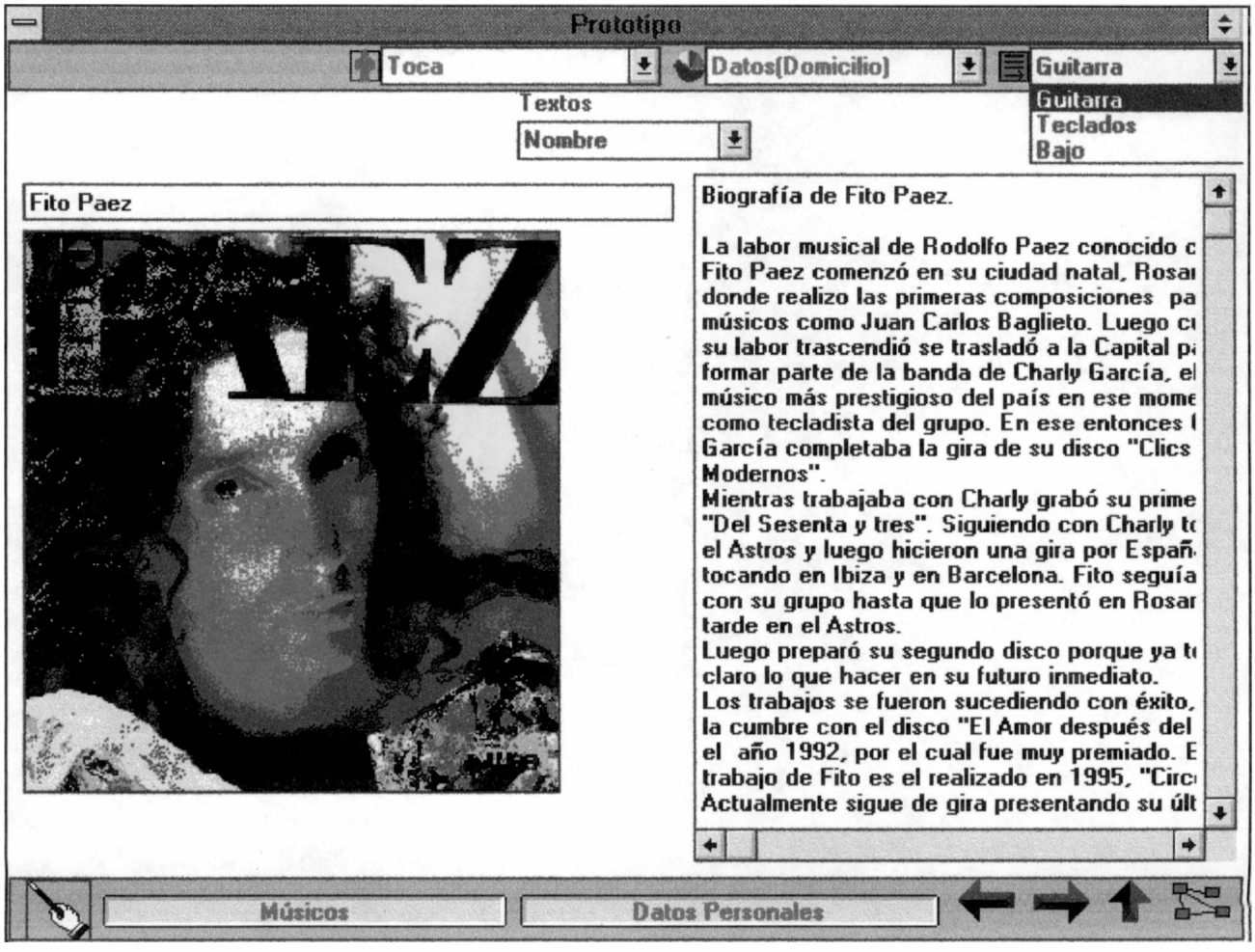
3.- Ejemplo del Editor de Diseño Navegacional.



4.- Ejemplo con los tres editores de diseño activos.



5.- Ejemplo de la ejecución del prototipo.



Bibliografía

1) Hypertext and Hypermedia.
Jacob Nielsen. Academic Press Professional. 1993.

2) Hypertext Design Environments and the Hypertext Design Process.
Jaselyne Nanard and Marc Nanard. Comunications of the ACM. 1995.

3) Introdução aos Sistemas e à Autoria Hipermídia.
Daniel Schwabe, Gustavo Rossi.

4) HDM: A Model for the Design of Hypertext Aplications.
Franca Garzotto, Paolo Paolini, Daniel Schwabe. Hipertext '91 Proceedings.

5) OOHDM: An Object Oriented Hypermedia Design Model.
Daniel Schwabe, Gustavo Rossi. 1994.

6) The Object-Oriented Hypermedia Design Model.
Daniel Schwabe, Gustavo Rossi. 1995.

7) RM: A Methodology for the Design of Structured Hypermedia Aplications.
Tomás Isakowitz, Eduard A. Stohr, P. Balasubramanian. 1994.

8) Designing Hypermedia Applications.
Tomás Isakowitz, Eduard A. Stohr, P. Balasubramanian. IEEE Computer Society Press. 1994.

9) Hypermedia Design, Analysis, and Evaluation Issues.
Franca Garzotto, Luca Mainetti and Paolo Paolini. Comunications of the ACM. 1995.

10) Computer Support for Designing Structured Hypermedia Aplications.
Tomás Isakowitz, Alicia Díaz, Vanesa Maiorana and Gabriel Gilabert. 1995.

11) Computer-Aided Support for Hypermedia Design and Development.
Alicia Díaz and Tomás Isakowitz. Proceedings of the Second International Workshop on Hypermedia Design. Montpeller 1995.

12) Microsoft Visual Basic. Programmer's Guide.
Microsoft Corporation.

13) The Visual Basic for Windows Handbook.
Microsoft Corporation.

14) Microsoft Visual Basic. Professional Features Book 2.
Microsoft Corporation.

15) Microsoft Access. Manual del usuario.
Microsoft Corporation.

16) Microsoft Windows. Manual del usuario.
Microsoft Corporation.

17) Hypercard.
Apple Computers.

18) Multimedia Toolbook 3.0.
Asymetrix.

19) Object Oriented Modeling and Design.
J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen.

20) Principles of Databases.
J. Ullman. Computers Science Press. 1982.

21) Data Modeling.
Lawrence Sanders.

22) Fundamental of Data Base.
R. Elmasri, S. Navethe. Addison - Wesley. 1993.

23) Diseño Conceptual de Base de Datos.
Batini, Ceri, S. Navethe. Addison - Wesley. 1994.


BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Agradecimientos.

- A la Lic. Alicia Díaz. Por ayudarmos y saber orientarnos cada vez que lo necesitamos.

- A L.B.S. Informática y a R.C.M. Computación y Sistemas. Por facilitarnos recursos para la realización del proyecto.

Por cualquier consulta, sobre la herramienta implementada comunicarse a los siguientes teléfonos:

Mascena Mario Enzo 021-523543.
Montanaro Carlos José 021-245091.