UNIVERSIDAD DE LA REPÚBLICA – FACULTAD DE INGENIERÍA

INSTITUTO DE COMPUTACIÓN

AÑO 2005

PROYECTO DE GRADO

VISUAL REPORTING SERVICES

ESTUDIANTES: GUSTAVO FERNÁNDEZ, ROLANDO LARRAINCI

TUTOR: FERNANDO RODRÍGUEZ

EMPRESA RESPONSABLE: MAGMA TOOLS

USUARIO RESPONSABLE: GUSTAVO LARRIERA

RESUMEN

En este trabajo se presenta una solución al problema planteado por la empresa Magma para permitir la edición y ejecución de reportes por parte de un usuario final (UF). Para resolver el problema presentado se proponen: una definición de representación de modelos de negocio (Meta-Modelo de Negocio, MMN); un lenguaje de edición de la representación propuesta (Lenguaje de Edición del Meta-modelo, LEM); un integrador de fuentes de información; y una representación genérica de los reportes.

El meta-modelo propuesto es capaz de capturar y representar a alto nivel los conceptos (entidades) existentes en el modelo de negocio de un usuario, incluyendo sus atributos y relaciones, además de vincular los conceptos con los datos manejados en uno o más sistemas de información utilizados, y al mismo tiempo abstraerlos de su representación en ellos.

En base a estos elementos se construyó una aplicación prototipo que permite seleccionar información, diseñar y ejecutar reportes, para las distintas instancias del modelo, validando de esta forma los conceptos desarrollados.

Palabras Clave:
Metadata, Servicios, Macroservicios, Adaptador, Modelo de Negocio, Reporte, Reporteador, Reporting Services, RDL, Fuentes de Información, Web Services, Remoting.

ÍNDICE

1 INTRODUCCIÓN

Una parte importante de las estrategias de Business Intelligence (BI) de las empresas hoy en día está formada por los reportes, o informes, los cuales son utilizados por sus mandos de niveles medios y altos para la toma de decisiones.

El diseño de los reportes es una tarea que implica seleccionar los datos que puedan ser de interés, para luego presentarlos y agruparlos de forma que permitan descubrir nueva información.

Resulta de interés que estos reportes puedan ser diseñados y ejecutados por quienes conocen el negocio, que son en ultima instancia quienes van a utilizarlos. Ellos serán los usuarios finales de nuestro sistema.

Al mismo tiempo, los datos con los que se componen estos reportes tienden a surgir de varios sistemas de información y no de uno solo. Por esto se debe trabajar con varias fuentes de información.

En este trabajo se presenta una representación de modelos de negocio, un meta-modelo de negocio, que permite presentar de una forma entendible, a los mandos de las empresas, los datos operacionales (estos son, los datos generados a partir de la operación diaria de las empresas). En el MMN también se puede indicar de qué forma se obtienen los datos para las entidades que se definan, pudiendo estos datos provenir de distintas fuentes de información utilizadas por la empresa, lo que implica manejar la correspondencia de los datos de los sistemas utilizados con el modelo que se quiera utilizar.

A partir de este modelo se desarrolló una aplicación prototipo que permite a quienes conocen el modelo del negocio, diseñar y ejecutar nuevos reportes, permitiendo además integrar a través de MMN los datos provenientes de distintos sistemas de información. El modelo presenta los datos provenientes de estos sistemas según el "modelo de negocio" de los mandos de las empresas. También se deja abierta la posibilidad de utilizar distintos reporteadores, de forma de no limitar la herramienta a un único reporteador.

Se provee un prototipo de la solución desarrollada en C#.Net. En la cual se permite integrar fuentes de información como Web Services, servicios de Remoting y bibliotecas de ensamblados, y se utiliza MS Reporting Services como reporteador.

1.1 MOTIVACIÓN

Este trabajo fue presentado por la empresa Magma, con el objetivo de poder brindar a sus clientes una herramienta de diseño y ejecución de reportes que no implique la intervención de personal técnico. Una herramienta de estas características permite que el cliente disponga de los reportes que desee, sin que se deba dedicar personal y equipamiento a la implementación de los mismos.

Magma cuenta con un conjunto de herramientas de software a partir de las cuales se genera el producto solicitado por un cliente, siendo éste una solución de ERP, CRM o similar. Es decir que tiene gran parte del proceso de desarrollo asistido por herramientas automatizadas.

En la actualidad la empresa no cuenta con herramientas que faciliten la personalización de los reportes solicitados por los clientes, identificándose este como un punto débil en el proceso de desarrollo. Esta situación mejoraría, si se contara con una herramienta que permita que los clientes diseñen sus propios reportes, con esto se daría una gran libertad para obtener los reportes que se deseen.

En el desarrollo de este trabajo se diseñaron un modelo de representación de modelos de negocio (MMN), un lenguaje de edición de este modelo (LEM) y un motor de integración de datos que permite la ejecución de los reportes. A partir de estos elementos se planea implementar una herramienta gráfica que permita que usuarios sin conocimientos técnicos de computación puedan diseñar y ejecutar nuevos reportes.

1.2 COMPLEJIDADES INVOLUCRADAS

Para el desarrollo de este trabajo se encontraron las siguientes complejidades:

- **Generación de reportes**

  Hoy en día los reportes que utiliza una empresa pueden marcar la diferencia, tanto en los

costos como en la competencia con otras empresas y el apoyo en la toma de decisiones de operación de las empresas.

Para que los reportes estén disponibles se debe realizar un proceso que básicamente cuenta con los siguientes pasos:

- **análisis de los datos** presentes en los sistemas de información de la empresa,
- **diseño** de los reportes, procesando y presentando los datos de una forma que facilite el análisis de la información,
- **implementación** de los reportes diseñados,
- **ejecución** de los reportes (habitualmente se querrá ejecutar los reportes en períodos regulares).

El análisis y el diseño conviene que sean llevados a cabo por quienes conocen el negocio y usan los reportes (estos serán nuestros *usuarios finales*, o simplemente *usuarios*), en base a la información disponible en los sistemas. Dado que ellos son quienes pueden decidir que reportes les serán de utilidad y que información necesitaran. A partir del diseño que surja los reportes deberán ser implementados y luego ejecutados por personas con los conocimientos adecuados de informática.

Este proceso tiene complicaciones que pueden llevar a que dure más de lo deseable. Además, no es un proceso que ocurra una única vez, si no que debe repetirse para que los reportes evolucionen según la experiencia que se adquiere al utilizarlos. Por otra parte el proceso de ejecución, en general, se lleva a cabo en forma semanal, mensual u otros intervalos de conveniencia, y esto suele quedar a cargo de personal técnico.

La elaboración de los reportes implica un proceso de intercambio de información, el cual puede extenderse en el tiempo y llevar a que el reporte no esté disponible cuando se lo necesita. Este retraso se intenta eliminar a través del uso de nuestro modelo.

Mediante el modelo de negocio planteado aquí se trata de simplificar el proceso de análisis, diseño, implementación y ejecución de los reportes. Esta simplificación se logra presentando al usuario final los datos que existen en los sistemas de información de su empresa con una representación que se adapte a los conceptos del negocio que maneja habitualmente, y permitiéndole crear y ejecutar sus propios reportes. Para lograr esta presentación de los datos se propone el modelo MMN, que se presentará a lo largo del artículo. Los usuarios programadores serán los encargados de crear esquemas de MMN adecuados para representar el negocio de quienes quieran utilizar el sistema.

- **Integración de datos**

Otro punto importante a tener en cuenta en la generación de reportes hoy en día, es la variedad de sistemas de información que suelen utilizarse dentro de las empresas. Esto lleva a que los reportes a generar tomen sus datos de varios sistemas, y no de uno solo.

A partir de esto surgen diversos problemas de integración y proceso de datos.

Para solucionar este problema se propone, junto con la integración de datos que se realiza a través de la especificación del modelo de negocio antes mencionado, una herramienta que permite procesar los datos obtenidos de diversos sistemas de información.

Estos sistemas de información son los que nos proveen los datos para los reportes, los cuales deben estar en un formato común para poder procesarlos. Dadas las herramientas utilizadas por Magma resulta de especial interés integrar Web Services, servicios de Remoting (formato propietario de MS para el acceso remoto a objetos) y datos obtenidos de bibliotecas de ensamblados de .Net.

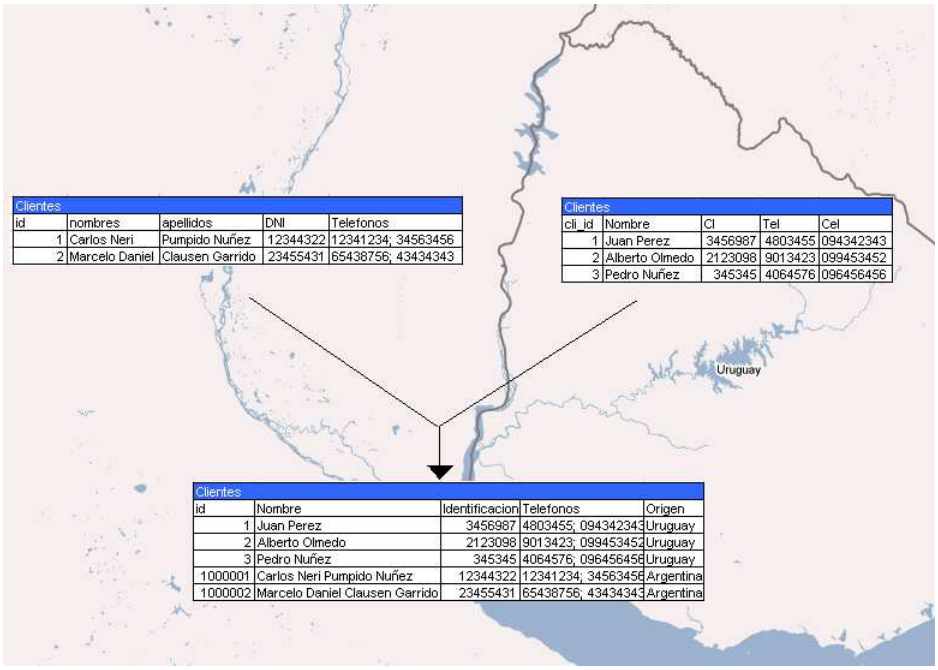
1.3 EJEMPLO.

A continuación se presenta un ejemplo que plantea las dificultades mencionadas en el punto anterior.

Supongamos que se quiere crear un reporte sobre los clientes de una empresa para un sistema de Customer Relationship Management (CRM), los datos sobre los clientes provienen de dos sistemas de información heterogéneos, uno utilizado en Uruguay y otro en Argentina. El sistema utilizado en Uruguay tiene como documento de identidad la cédula y el de Argentina el DNI, además se quiere que los clientes de uno y otro país estén identificados de acuerdo a su

sistema de origen. También se debe resolver el hecho de que las compras en Argentina se almacenan en pesos argentinos y en Uruguay en pesos uruguayos, pero se quieren las compras en dólares y de acuerdo al cambio del momento.

Otro requerimiento podría ser que en los dos países se utilicen reporteadores distintos. La solución que se presenta permite utilizar reporteadores distintos en cada país.



Para que un usuario final pueda crear el reporte deseado, se deben presentar estos datos al usuario de una forma tal que éste los pueda manipular. Por esto se deben crear nuevos modelos por sobre las representaciones utilizadas en los sistemas de información, que no siempre corresponden a los conceptos manejados por los usuarios, si no a representaciones convenientes para los sistemas informáticos. Por ejemplo, las compras de los clientes en Argentina podrían estar almacenadas en una tabla común de movimientos. Esta tabla incluiría las compras de los clientes, los pagos en cuotas, las compras a proveedores, con sus respectivos pagos.

La solución habitual consistiría en que el usuario final analice los datos disponibles a través de los sistemas que se utilizan en la empresa, diseñe el reporte deseado y pida a un programador que cree el reporte para que luego este pueda ser utilizado. Este proceso se debe repetir cada vez que se requiera el diseño de un nuevo reporte, y esto se agrava porque es bastante habitual en este tipo de desarrollos que el usuario varíe los requerimientos luego de creado el reporte.

También se deben tener en cuenta valores que se desee que cambien de una ejecución de un reporte a otra. En nuestro ejemplo podría ser necesario que el tipo de cambio con el que se realizan las cuentas se pueda indicar cuando se ejecuta el reporte.

El meta-modelo propuesto permite que un programador, defina qué datos corresponden a qué conceptos del modelo de negocio, indicando también como se integran los datos y que valores asumen aquellos datos que existen en una fuente y no en la otra. Sobre este modelo el usuario final puede trabajar para analizar los datos disponibles y crear aquellos reportes que considere útiles. El meta-modelo le permite trabajar sobre datos homogéneos integrados dentro del modelo, y con una estructura que se adapta a su visión de la realidad, en oposición a la representación de las fuentes de información que se adapta al procesamiento de los datos.

1.4 OBJETIVOS

En este trabajo se busca proveer un marco conceptual a partir del cual usuarios no técnicos sean capaces de generar los reportes que deseen, con una intervención reducida de personal técnico.

Para alcanzar esto planteamos el siguiente esquema de trabajo:

- Generar un ***estudio sobre el estado de arte*** de las opciones tecnológicas disponibles para generación de reportes, analizando también las opciones de arquitecturas, modelos y tecnologías que ayuden en la integración de los datos en dichos reportes.
- **Evaluar las representaciones de reportes existentes** (principalmente las que sean independientes del reporteador), determinando cuales son las posibilidades de utilizar distintos reporteadores con nuestra herramienta.
- Plantear un modelo de meta-datos que permita tener una ***representación común de los objetos de alto nivel*** utilizados para crear reportes, sin importar cual es su representación en los sistemas subyacentes, o los procesos que deben ejecutarse para instanciar los objetos del modelo.
- Proponer un lenguaje ***de edición de la representación común de los objetos de alto nivel***, que permita crear y modificar instancias de la representación propuesta.
- ***Sugerir una representación genérica para los reportes***, en formato XML, teniendo en cuenta la representación utilizada por Microsoft.
- ***Implementar un prototipo de la arquitectura planteada*** y de la herramienta generadora de reportes, de forma de poder probar los conceptos analizados y expuestos.
- ***Implementar dos Editores del Modelo de Negocio***. Uno orientado a los programadores y otro a usuarios finales.

Como representación común de los objetos de alto nivel se propone un modelo de representación del modelo del negocio de los clientes, al que llamamos Meta-Modelo de Negocio, y el Lenguaje de Edición del Meta-Modelo. Este modelo y el lenguaje se utilizan en los Editores del Modelo de Negocio que permiten crear y modificar instancias (esquemas) de MMN y, generar los reportes definidos en el modelo.

El lenguaje LEM propuesto, es un lenguaje de comandos, del estilo de las consolas de comando de Linux o el Command Prompt de Windows. LEM no es adecuado para que lo utilicen usuarios finales, pero se optó por un lenguaje de este tipo para adecuar el alcance del trabajo al tiempo y los recursos disponibles, dándole prioridad a las funcionalidades frente a la usabilidad o amigabilidad. Este luego será complementado por el trabajo de Magma, con el cual se potenciará la usabilidad y amigabilidad de todo el entorno a través de un editor gráfico, en base a las funcionalidades brindadas.

Dentro de los esquemas del modelo propuesto (MMN) se pueden definir entidades, las cuales representan los conceptos del modelo de negocio de los clientes. También se pueden representar relaciones entre las entidades, manteniendo restricciones, agrupamientos, aplicación de filtros, proyecciones y cálculos. Además, dentro del modelo se incluye la información sobre los sistemas de información y las operaciones en estos (a las que llamaremos servicios) a partir de las cuales se obtienen los datos para las entidades del modelo.

La ejecución de los nuevos reportes podría implicar un procesamiento e integración de datos, con los cuales se pueden descubrir aspectos de los datos analizados que estaban ocultos con la información original. El procesamiento de datos se realiza dentro de la herramienta provista, y la integración de los datos es indicada en MMN y llevada a cabo por la herramienta.

Los esquemas del modelo pueden ser extendidos por el cliente mismo, para crear nuevas entidades basadas en las entidades ya existentes, o para definir sus nuevos reportes. Estas extensiones se realizan indicando operaciones sobre las entidades ya existentes, seleccionando las entidades a partir de las cuales se creará la nueva entidad e indicando restricciones en las relaciones y atributos de las entidades.

Se presentan ejemplos de las posibilidades que brinda el modelo desarrollado, con los cuales se considera que se alcanza al objetivo planteado.

2 ESTADO DEL ARTE

El trabajo aquí presentado tiene como objetivo lograr que usuarios finales sean capaces de diseñar y ejecutar sus propios reportes. Analizando algunas de las diferentes herramientas existentes en el mercado, es que nos introduciremos en el tema para comprender los conceptos manejados y determinar cuales son los temas relacionados, además se analizan cuales son los estándares propuestos que participan en cualquier etapa de la generación de reporte.

Como el producto está pensado para usuarios finales se investigó sobre la representación de **modelos de negocios** y la **metadata** requerida para vinculación estos modelos con las **fuentes de información heterogéneas,** que soportan la información de la organización.

Dos puntos que están fuertemente unidos a la generación de reportes con datos provenientes de múltiples fuentes de información son el **procesamiento de datos** y la **integración de fuentes de información.**

Dado que se estableció que, en el prototipo, la interfase del sistema con el usuario sería en modo comando es que se presenta el análisis de **compiladores y traductores**.

Los puntos a analizar en el capitulo estado del arte, entonces son: generación de reportes y análisis de diferentes reporteadores; representación de modelos de datos y metadata; procesamiento de datos; integración de fuentes de información; sistemas multifuentes, y; lenguajes de programación y compiladores.

Aquí se presenta un resumen del documento principal de estado del arte. En él se podrá encontrar más información sobre la mayoría de los puntos.

2.1 GENERACIÓN DE REPORTES

La generación de reportes consiste en combinar una instancia de los datos que surgen de la operación de una organización (datos operacionales), los cuales son procesados y formateados según una definición del reporte. Esta actividad constituye actualmente una parte fundamental de una estrategia general en Business Intelligence (BI).

Sobre la teoría de generación de reportes no se han encontraron estándares, más allá de un par de propuestas que son:

- RDL(Report Definition Language) es un lenguaje de **diseño de reportes** desarrollado por MS[RDL2003],
- ROM (Report Object Model) esta representación de los reportes es utilizada por la herramienta BIRT de Eclipse.

2.1.1 Diseño de Reportes

El diseño de un reporte contiene un conjunto de instrucciones que describen una forma de obtener los datos procedentes de las fuentes de información, el lugar donde serán colocados esos datos, el formato que tendrá tanto el texto como el área donde se mostrarán. Es decir que el diseño del reporte mantiene una descripción un reporte, tanto su presentación como los datos que se muestran.

En esta área se encontró un proyecto de estandarización realizado por MS, este consiste en el desarrollo de un lenguaje de definición de reportes, llamado RDL (por Report Definition Language). Éste tiene como objetivo promover la interoperabilidad entre reporteadores comerciales, mediante la definición de una representación común que permita el intercambio de definiciones de reportes, es decir que cualquier reporteador pueda generar un reporte teniendo la definición del reporte y los datos.

Este es un importante aporte para la implementación de nuestro proyecto, debido a la independencia que da para la generación del diseño de reporte, puesto que partiendo de cero se puede generar el diseño de un reporte y por medio de un Web Service ofrecido por RS se puede generar el reporte. El resto de los reporteadores investigados exigen que alguna etapa del diseño del reporte sea realizada desde su propio entorno, lo cual dificulta la integración de reporteadores con un proyecto de terceros.

Algunas otras organizaciones han establecido sus propios lenguajes de definición de reportes, como la herramienta BIRT (Business Intelligence and Reporting Tool) de Eclipse para Java, donde se utiliza una representación a la que llama **XML Report Design Schema**, pero no se da a conocer la estructura de estos esquemas.

2.1.2 Reporteadores

Son muchas las herramientas que permiten generar un diseño del reporte y combinarlos con datos provenientes de sistemas de información, para generar una instancia de reporte (generalmente llamado reporte únicamente), la cual se pueda imprimir o guardar como un archivo; algunas de estas herramientas son:

- Business Objects Enterprise (de Business Objects) [boe2005]
- MicroStrategy Architect (de MicroStrategy) [msa]
- Cognos Impromptu (Cognos) [cgi]
- SQL Server Reporting Services (Microsoft) [msrs]
- Eclipse Business Intelligence and Reporting Tools (Actuate Corporation y Eclipse Fundation) [ecbi2005].

También se provee un resumen de las características de estos reporteadores en la versión extendida del Estado del Arte.

2.1.3 Conclusiones

Existe una gran variedad de reporteadores. El inconveniente en la mayor parte de estos sistemas es que el diseño del reporte solo se puede lograr a partir del propio reporteador y no a partir de sistemas independientes.

Dos reporteadores que se diferencian del resto por su forma de representar sus reportes son MS Reporting Services (RDL) y Business Intelligence and Reporting Tools (ROM, Report Object Model), para los cuales se pueden crear reportes sin tener que utilizar ambientes de diseño propietarios.

BIRT no será utilizado por no haber estado la información disponible a tiempo, pero queda planteado como trabajo a futuro integrar este reporteador y analizar la representación de reportes que propone.

Se presentará una representación genérica de los reportes basada en el RDL de Microsoft y se integrará MS RS como reporteador en nuestro prototipo.

2.2 MODELO DE DATOS

La información generalmente es registrada procesada y luego comunicada a otras personas en una forma simbólica que pueda ser entendida. Para estos fines resulta de fundamental importancia contar con una adecuada representación simbólica de la información.

El modelo de datos debe ser una fiel representación del problema a modelar, donde el objetivo principal es la obtención de estructuras no redundantes, consistentes, e íntegras; por lo cual el lenguaje natural no es el más adecuado y menos en sistemas computacionales.

2.2.1 Modelo de Negocio

El Modelo de Negocio consiste en capturar el esquema general y los procedimientos que gobiernan el negocio, es decir que representa los conceptos, tareas y las relaciones, que habitualmente se manejan en la temática a modelar.

Si deseamos que un usuario inexperto interactué con un sistema y que pueda sacarle el mayor provecho lo antes posible, se debe manejar un contexto familiar, es decir su Modelo de Negocio. Este debe independizarse del modelo utilizado en los sistemas de información que es donde se representan los nombres de fuentes de información, modelos relacionales, esquemas y consultas de base de datos, servidores, servicios web, etc.

Presentar el MN de una forma familiar a un usuario final requiere de una representación distinta a la que habitualmente se utiliza en los Sistemas de Información, que está orientada a la operación con los datos y no a la interacción con el usuario (la presentación para el usuario solo se utiliza en la interfaz del sistema).

Para lograr presentar estos datos en forma adecuada debe existir una forma que permita relacionarlos, la solución a este problema es agregar información la cual mapee entre los dos modelos, a esta información se le llama metadatos, Y deberemos desarrollar un meta-modelo para representar nuestros metadatos.

Los metadatos deberán ser definidos por alguien que conozca el modelo utilizado en los sistemas de información y que logre presentarlo al usuario inexperto, en una forma que sea entendible y acorde con sus conceptos.

Representación del Modelo de Negocio

No existen estándares los cuales permitan representar Modelos de Negocios, pero si existen algunos que permiten modelar datos y procesos. Si tenemos en cuenta la probada capacidad de algunos de estos modelos para vincular a expertos en sistemas con usuarios finales, se podrán presentar adaptaciones para representar Modelos de Negocios.

Se han explorado y desarrollado diferentes formas de modelar datos y procesos, existiendo modelos jerárquicos, de red, semánticos, relacionales, etc. Dentro de estos veremos MER y UML:

- **UML**(Unifed Modeling Languaje) es un lenguaje para la especificación, visualización, construcción y documentación de las diferentes etapas del ciclo de vida del software. Fue originalmente concebido por la Corporación Rational Software, y ha sido presentado al Object Management Group (OMG) y aprobado por éste como un estándar (17 de noviembre de 1997).
- **MER** (Modelo Entidad Relación) usado para la especificación conceptual de Base de Datos, para tener un manejo más amigable tiene asociada una representación gráfica llamada DER (diagramas entidad-relación) estos diagramas fueron propuestos por Peter P. Chen en 1976.

Estos modelos tienen como objetivo permitir la abstracción de la realidad y reflejar las necesidades del negocio. En ellos se definen una serie de conceptos, como Entidades, Atributos y Relaciones, y para una mejor fidelidad y comprensión de la realidad estos modelos presentan una representación gráfica.
Algunas de estas representaciones resultan ser un buen punto de contacto entre expertos en sistemas y usuarios finales, es esperable que los conceptos definidos en estos modelos puedan servir como punto de partida para la definición de nuestro modelo de negocio.
Otra de las ventajas de incluir los conceptos usados en UML o en el MER, es facilitarle la comprensión y adaptación a los usuarios que ya están habituados al manejo de estos modelos, estos usuarios serían aquellos que deberán definir las instancias del meta-modelo y la interacción de este con las fuentes de información.

2.2.2 Conclusiones

El modelo planteado estará orientado a representar la información presente en las fuentes de información de forma entendible para los usuarios finales.

Existen modelos que presentan conceptos útiles para nuestros objetivos (como el Modelo relacional o el Orientado a Objetos), pero que no cubren todas nuestras necesidades, por lo que serán utilizados como base.

2.3 METADATOS

Los metadatos son "datos sobre datos" y describen el contenido, calidad, condiciones, origen y otras características de otros datos.

Son varios los motivos para usar metadatos, entre los que encontramos: optimizar búsquedas, permiten la interoperación de sistemas, disminuir el tráfico mediante el intercambio de descripciones en lugar de información, realizar controles, determinar restricciones, entre otras. Algunos ejemplos particulares de su utilización son: catálogos, resumen de documentos, la información necesaria para poder vincular dos modelos.

Debido a la necesidad de integración de información que ha generado el desarrollo de la web en los últimos años se han creado y popularizado varias formas de representar y utilizar metadatos. A continuación resumimos las más destacadas.

2.3.1 XML (Extensible Markup Language)

XML (e**X**tensible **M**arkup **L**anguage) es un lenguaje de etiquetas extensible desarrollado por la W3C que ofrece un formato para la descripción de datos estructurados. Lo cual permite declaraciones de contenido más precisas, resultados de búsquedas más exactos y fácil de distribuir. XML garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes.
Desde su creación en 1998, XML ha servido como base para la construcción de otros lenguajes según diversos aspectos y con distintas orientaciones:

- para el intercambio y extracción de información: SOAP, WSDL, XQuery, XPath, SAX, DOM;
- formación "vocabularios" específicos de negocio: MathML, MusicML, OTA, HL7, XBRL;
- formateo o presentación de la información: XHTML, XForms, WML, SVG;
- transformación y procesamiento del propio XML: XSL, XSLT, DTD, XSL-FO, XML-Schema, RelaxNG, XLink, XPointer.

2.3.2 Consultas sobre XML

Una vez obtenida la información, en un cierto formato, es deseable aplicarle algún tipo de procesamiento. Uno de los problemas que tiene mantener información en XML es que el volumen de la información crece rápidamente, y las herramientas que existen para el manejo de los datos no son prácticas, entre estas herramientas tenemos: los parsers SAX y DOM, o la familia de herramientas bindings. Otra solución existente es XSLT que permite transformar datos de XML a otros formatos como HTML o PDF, pero solo son útiles para hacer presentaciones diferentes de los datos no para aplicar operaciones o buscar información.

Las herramientas que están empezando a aparecer en el mercado son implementaciones del lenguaje xQuery. Muchas veces se encuentran prototipos o primeras versiones considerablemente inestables de estas herramientas.

2.3.2.1 XQuery

XQuery es un lenguaje de consultas que se aplica sobre información que tiene formato XML, este lenguaje está siendo desarrollado por el grupo de trabajo XQuery de la W3C y tiene varias implementaciones independientes.

Se puede decir que XQuery es a XML como SQL es a las bases de datos relacionales [gut2005] [rui2004] [min2004]. Su principal función es extraer información de un conjunto de datos organizados como un árbol n-ário de etiquetas XML. En este sentido XQuery es independiente del origen de los datos.

En este lenguaje funcional cada consulta es una expresión que es evaluada y devuelve un resultado, el cual también se encuentra en XML. Una de sus principales características es poder combinar diversas expresiones, que cumplan ciertas reglas, para crear nuevas expresiones más complejas y de mayor potencia semántica.

XQuery ha sido construido sobre la base de XPath, siendo este un lenguaje declarativo para la localización de nodos y fragmentos de información en árboles XML. Se basa en este lenguaje para realizar la selección de información y la iteración a través del conjunto de datos. La nueva versión de XPath (2.0) está siendo desarrollada por el mismo grupo de trabajo de la W3C y de manera conjunta a la versión 1.0 de XQuery, la última versión salió a fines del 2005.

Las principales características definidas por el grupo de trabajo de XQuery son:

- Es un lenguaje declarativo. Al igual que SQL hay que indicar que se quiere, no la manera de obtenerlo.
- Independiente del protocolo de acceso a la colección de datos.
- Una consulta en XQuery funciona en forma independiente de las fuentes de información.
- Las consultas y los resultados respetan el modelo de datos XML.
- Las consultas y los resultados ofrecen soporte para los namespace.
- Es capaz de soportar XML-Schemas y DTDs y también de trabajar sin ninguno de ellos.
- Trabaja con independencia de la estructura del documento, esto es, sin necesidad de conocerla.
- Soporta tipos simples, como enteros y cadenas, y tipos complejos, como un nodo compuesto por varios nodos hijos.
- Las consultas soportan cuantificadores universales (para todo) y existenciales (existe).
- Las consultas soportan operaciones sobre jerarquías de nodos y secuencias de nodos.
- Es posible en una consulta combinar información de múltiples fuentes.
- Las consultas son capaces de manipular los datos independientemente del origen de estos.
- Es posible definir consultas que transformen las estructuras de información originales y crear nuevas estructuras de datos.
- El lenguaje de consulta es independiente de la sintaxis, dicho de otra forma, es posible que existan varias sintaxis distintas para expresar una misma consulta en XQuery.

Todas las consultas de XQuery están formadas por la combinación de las expresiones For, Let, Where, Order y Return, generalmente en ese orden, por eso se dice que siguen la regla FLOWOR. Pasamos a analizar cada expresión:

- **For**: Vincula una o más variables a expresiones escritas en XPath, creando un flujo de tuplas en el que cada tupla está vinculada a una de las variables.
- **Let**: Vincula una variable al resultado completo de una expresión añadiendo esos vínculos a las tuplas generadas por una cláusula for o, si no existe ninguna cláusula for, creando una única tupla que contenga esos vínculos.

- **Where**: Filtra las tuplas eliminando todos los valores que no cumplan las condiciones dadas.
- **Order by**: Ordena las tuplas según el criterio dado.
- **Return**: Construye el resultado de la consulta para una tupla dada, después de haber sido filtrada por la cláusula where y ordenada por la cláusula order by.

La W3C estipula las características del lenguaje, pero no desarrolló una implementación del mismo, sino que deja a cargo de terceros esta tarea, publicando en su sitio alguno de los productos que implementan este lenguaje, entre los que se destacan:

- BEA's Liquid Data.
- GNU's Qexo (Kawa-Query).
- Compiles XQuery on-the-fly to Java bytecodes. Based on and part of the Kawa framework. An online sandbox is available too. Open-source.
- Microsoft's SQL Server 2005 Express, with XQuery support.
- También existe una DLL provisto en algunos de los documentos comentados llamada Microsoft.Xml.XQuery.dll, el documento es [Min2004].
- Oracle's Xquery Technology – Preview.
- PHP. Open source.

2.3.2.2 Inconvenientes

La idea de usar este lenguaje de consultas sobre datos formateados en XML parece ser una buena opción, pero en la etapa de análisis de los productos que implementan XQuery se encontraron algunos inconvenientes, entre los que destacamos:

- Falta de funcionalidades que aparecen en la definición de XQuery;
- No todas las herramientas ofrecían una API que se pudiera operar en forma independiente del producto;
- En algunos casos podía ser un problema el intentar generar las consultas en tiempo de ejecución, sin la participación de un usuario y en caso de lograrse la generación de la consulta esta tal vez no fuera la optima demorando la ejecución de la misma;
- Otro problema es el consumo de recursos que implica el realizar consultas sobre XML usando XQuery, aunque no se pudo establecer si es problema de las implementaciones probadas, de XQuery, o de XML.

2.3.3 RDF (Resource Description Framework)

RDF fue diseñado por Ramanathan V. Guha y se transformó en una recomendación de la W3C. Este modelo se basa en la idea de convertir las declaraciones de los recursos en expresiones con la forma sujeto-predicado-objeto.

- El sujeto es el recurso, es decir aquello que se está describiendo.
- El predicado es la propiedad o relación que se desea establecer acerca del recurso.
- El objeto es el valor de la propiedad o el otro recurso con el que se establece la relación.

La RDF junto con OWL es una de las tecnologías esenciales de la Web semántica, donde las representaciones basadas en las estructuras de árbol no son suficientes. Entendiéndose por Web Semántica a la idea de tomar a la Web como una base de datos global, donde tanto las máquinas como los humanos puedan hacer deducciones y organizar la información, lo cual implican que cada elemento de la Web tenga: semántica (significado de los elementos), estructura (organización de los elementos) y sintaxis (comunicación).

Otro punto importante es que cualquier especificación de declaraciones RDF puede serializarse en XML, aunque también pueden usarse otros lenguajes.

2.3.4 Clasificación de la Información

Tener la información clasificada optimiza mucho su procesamiento y más en la actualidad que los volúmenes de información son muy grandes, puesto que las búsquedas navegación y filtrados son más rápidas. Además, al problema de la gran cantidad de información se pueden

sumar problemas como que la información puede provenir de diferentes fuentes, por lo cual expresiones diferentes pueden tener el mismo significado. Por ejemplo los conceptos proveedor y acreedor, o los conceptos deudor y cliente, los conceptos pueden ser iguales pero las expresiones son diferentes.

Para solucionar este tipo de problemas es adecuado que la información esté estructurada, clasificada y se puedan establecer relaciones entre los conceptos, de esta forma todas las operaciones serán más exactas y más rápidas.

Las taxonomías y ontologías sirven para tener una definición formal y precisas de los términos, poder clasificarlos y relacionarlos.

2.3.4.1 Taxonomías

Podemos definir a la taxonomía como la ciencia que estudia las clasificaciones, en las taxonomías se definen las clasificaciones y relaciones jerárquicas entre los elementos a analizar, basándose en un conjunto de atributos de cada elemento clasificado.

Dicha organización tiene la estructura de un árbol dirigido, por lo cual cada elemento solo puede pertenecer a una única rama, lo cual puede ser un grave problema, pues no en todo los casos un concepto puede ser clasificado de manera tajante en un sólo lugar. Es en este punto donde entran las ontologías, la cual se analiza en la próxima sección.

2.3.4.1.1 Taxonomía XBRL (Extensible Business Reporting Language)

XBRL nace en el seno de AICPA http://www.aicpa.org (La asociación americana de censores y auditores de cuentas), para simplificar la automatización e intercambio de información financiera y empresarial.

XBRL es una especificación en XML que permite aplicar etiquetas identificativas únicas a los distintos elementos que componen la información financiera y contable, como, por ejemplo <beneficios netos>, estas etiquetas proporcionan una amplia gama de información sobre dicho elemento, por ejemplo, si es monetario, porcentaje o fracción.

Esta formalización de cada dato, indicando sus características, su denominación en distintos idiomas, las Normas Internacionales de Contabilidad que lo amparan, cómo se representa toda la metadata que le es aplicable, es lo que se denomina una taxonomía XBRL. Como ejemplo, una de las taxonomías más destacadas en España es la realizada por la DGI (Datos generales de identificación) que recoge todos los elementos de identificación contenidos en los documentos financieros. Para descargarla http://xbrl.org.es/gp/2005-03-10/dgi-2005-03-10.zip

2.3.4.1.2 Inconvenientes

Los principales problemas en la utilización de XBRL en nuestro sistema son tres.

- Las taxonomías solo abarcan el área financiera y contable.
- Hace falta una taxonomía regional o nacional, como la citada anteriormente.
- Sería necesario un manejo en forma profunda de los conceptos financiero y contable puesto que son cientos las definiciones manejadas en estas taxonomías y algunas de ellas pueden tener interpretaciones diferentes en distintas regiones o ser no tener interpretación en otra región.

2.3.4.2 Ontologías

Una ontología es una descripción formal de los conceptos y las relaciones (semánticas) entre conceptos. (GRUBER, 1993) Se usa para crear vocabularios estructurados en la recuperación o el intercambio de información. Los participantes del intercambio o búsqueda de información pueden ser humanos y/o agentes automáticos.

Una de las ventajas que tiene la ontología sobre la taxonomía es que una ontología es un grafo sin ninguna jerarquía y la taxonomía un árbol dirigido. Esto permite representar aquellas clasificaciones donde un elemento puede pertenecer a dos ramas a la vez, dando una mayor adaptabilidad al modelo.

Si bien, otros enfoques, como el de XML, han logrado cierta aproximación a este manejo de la información, no ha resultado suficiente, por lo que han surgido otras herramientas como RDF, con el que es posible crear ontologías limitadas pero útiles. En cambio si se desea una clasificación más extensa y compleja existe un lenguaje creado específicamente para definir ontologías que es OWL y está acompañada de una serie de editores para simplificar su generación.

OWL es un lenguaje de Ontologías para la Web que surge como una extensión de RDFS para permitir expresiones de relaciones complejas y de mayor precisión que RDF.

2.3.5 Conclusión

Con la llegada de la Web se amplia el uso de los metadatos que participan de las más variadas operaciones, como ser: describen el contenido, calidad, condiciones y origen de la información; permiten acelerar la búsqueda; facilitan las operaciones y la integración de datos; permiten y ayudan al entendimiento de la información almacenada, etc.

No existe un estándar para persistir la información mantenida en los metadatos pero los métodos más usados son los esquemas relacionales y en formato XML. Cada uno de ellos tiene sus pros y sus contras. Las ventajas del Esquema Relacional son la velocidad y la seguridad, aunque si la información es de poco volumen el tiempo de procesamiento es despreciable y además existen técnicas de encriptación que pueden asegurar la información. Las desventajas del esquema relacional es que está muy atado al ambiente de ejecución aunque existen esquemas relacionales con sus respectivitos lenguajes estándar para hacer consultas sobre ellos en todos los ambiente. Las ventajas de XML sobre los esquemas relacionales son la interoperabilidad, simplicidad y extensibilidad que permite, y su principal desventaja es que el volumen de información en XML crece rápidamente, por lo cual es costoso operar con información almacenada en este formato. Por otro lado, en cuanto a las características de las tecnologías y herramientas que soportan y permiten operar sobre estas dos estructuras, es que para los modelos relacionales están muy estandarizadas, y en cambio XML promete una mayor evolución.

Desde el inicio uno de los principales objetivos de XML es permitir el intercambio de información heterogénea a través de la Web basándose en una especificación simple y extensible, y para ello es imprescindible permitir y favorecer tanto la descripción de los datos a intercambiar como su manipulación.

También existen muchas especificaciones basadas en XML, para describir clasificaciones, relaciones y conceptos en las más diversas áreas, que pueden ir desde el sector de la salud hasta las actividades contables, a estas especificaciones formales se les llama Taxonomías, y aunque existe una taxonomía para la descripción de información contables (XBRL) está muy centrada en reportes financieros y maneja clasificaciones regionales según las regulaciones legales y conceptos financieros de cada región.

En cuanto a las herramientas de manipulación de datos que están bajo el formato de XML también se tiene varias especificaciones de herramientas, cada una de ellas con varias implementaciones, entre las que se destacan XPath o XQuery las cuales permiten realizar consultas sobre datos almacenados en formato XML.

En definitiva, para representar el meta-modelo se utilizará XML, esta decisión se tomó por:

- Posibilidades de interoperabilidad que brinda.
- Se estimó que el tamaño de las instancias del meta-modelo es manejable por las herramientas disponibles para el manejo de XML.
- La gran difusión de su uso.
- La adaptabilidad que brinda.

2.4 PROCESAMIENTOS DE DATOS

Uno de los problemas que tienen los esquemas relacionales, cuando se manejan volúmenes grandes de datos, es que están optimizados para las operaciones inserción, eliminación o actualización, pero la normalización demora la recuperación de datos debido a la combinación de muchos join. Es así que en los últimos tiempos se han desarrollado nuevas tecnologías para optimizar las operaciones de recuperación de datos que se encuentran en modelos relacionales, entre las cuales tenemos a OLAP.

2.4.1 OLAP

OLAP (procesamiento analítico en línea, **On-Line Analitycal Processing**) es una tecnología que utiliza estructuras multidimensionales para proporcionar un acceso rápido a los datos con el

fin de analizarlos. Los datos de origen de OLAP se almacenan habitualmente en almacenes de datos en una base de datos relacional.

Los datos que podrían provenir de esquemas relacionales o data warehouse y estar distribuidos en varios repositorios, por lo cual es necesario llevarlos a un formato común, luego son procesados según algún indicador llamado medida y en un contexto llamado dimensiones. Una vez que los datos se procesan, en función de las medidas y de las dimensiones, son guardados en una base multidimensional llamada cubo, la cual permite a los usuarios navegarlo utilizando distintas herramientas.

2.4.2 Meta-modelo para el cálculo de vistas

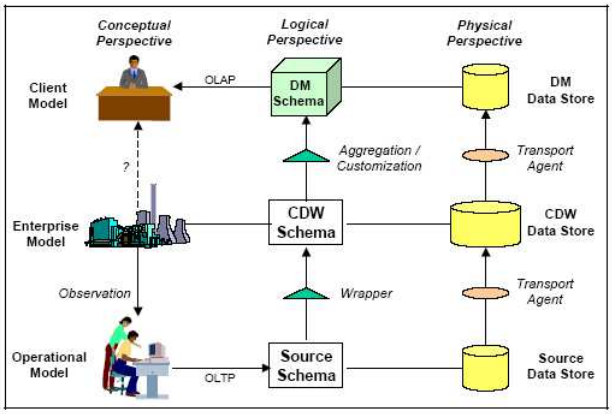
En [per2002] se utiliza una simplificación del meta-modelo desarrollado en el contexto del proyecto de Data Warehouse Quality (DWQ), y este modelo simplificado es extendido para permitir el cálculo de vistas a partir del modelo.

2.4.2.1 Marco de Metadata de DW

En el proyecto DWQ se proponen tres perspectivas para describir un sistema de DW:

- una *conceptual* de la corporación,
- una *lógica* de los modelos de datos, y
- una perspectiva *física* del flujo de datos.

Estas perspectivas están relacionadas en forma ortogonal con las tres capas tradicionales de un sistema de DW, que son las bases de datos fuentes, el DW corporativo y los data marts, como se muestra en la siguiente figura.



La *perspectiva conceptual* concierne a los modelos de negocio de los sistemas de información de la empresa. El rol central corresponde al *modelo corporativo*, el cual brinda una visión corporativa de los *objetos* conceptuales de la corporación. También le conciernen los diferentes modelos *operacionales* y los modelos *clientes* de cada sección de la empresa. Tanto los modelos operacionales como los modelos clientes son vistas parciales del modelo corporativo.

2.4.2.2 Meta-modelo para el cálculo de vistas y sus componentes

Para representar una vista se utiliza la clase *ViewExpression*, la cual tiene cuatro componentes, con los siguientes roles:

from. Un conjunto de relaciones que conforman la entrada de la expresión de cálculo. Una vista puede calcularse tanto a partir de relaciones fuentes como de otras vistas.

select. Una expresión de proyección para cada atributo de la vista. La siguiente sección presenta una gramática para construir las expresiones.

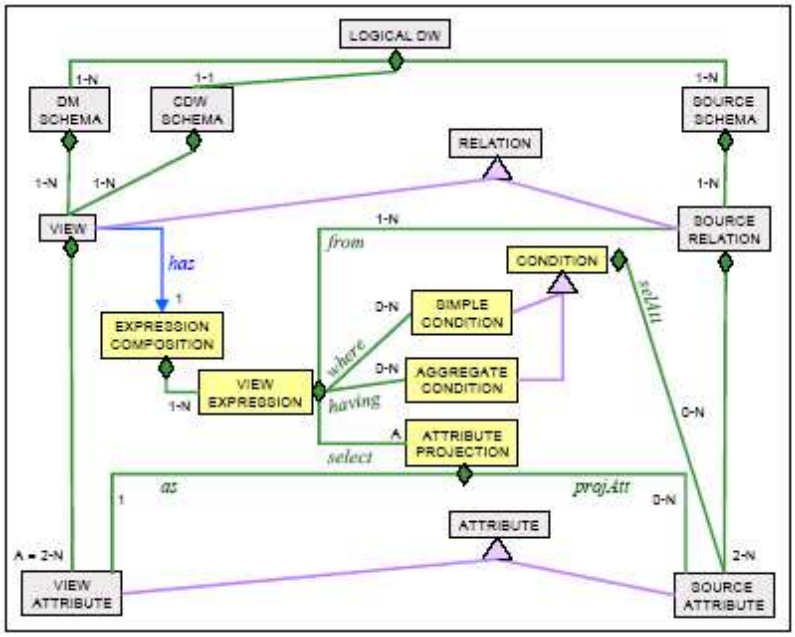
where. Un conjunto de condiciones o predicados que deben cumplir los datos de las relaciones.

having. Un conjunto de condiciones o predicados que deben cumplir los datos una vez agrupados o resumidos.

Una vista puede ser calculada con varias expresiones de cálculo, pero debe existir un mecanismo para componer dichas expresiones al calcular la vista, por ejemplo, realizando la unión de las instancias. La clase *Expression-Composition* expresa la manera de combinar varias expresiones para calcular la vista.

El mismo tipo de jerarquías puede definirse para las clases *Condition* y *ExpressionComposition*.

En la figura siguiente se muestra una representación de la metadata requerida para permitir el cálculo de vistas.



2.4.3 Conclusión

Como se comentó anteriormente, la principal utilidad de OLAP es que provee acceso rápido e interactivo a grandes volúmenes de datos en esquemas relacionales o data warehouse (data source). Estos datos podrían estar distribuidos en varios repositorios, con lo que sería necesario compatibilizar los datos.

El análisis de estas tecnologías (OLAP y Data Warehouse) nos provee de una base para el diseño del metadata requerido para la integración de fuentes de información, mediante la utilización de estructuras multidimensionales y para procesar la información a mostrar a través del cálculo de vistas.

Una de las principales ventajas de OLAP, como lo es el acceso a los datos en tiempo real, queda descartada por razones de tiempo, que nos impide implementar una solución que utilice estas técnicas.

2.5 SISTEMAS MULTIFUENTES

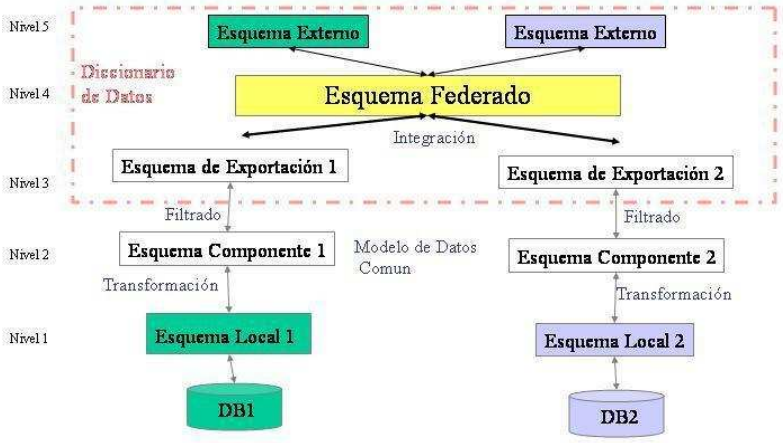
Los Sistemas Multifuentes son sistemas que integran datos provenientes de diferentes fuentes de información y donde además los datos pueden estar en diferentes formatos o no.
Existen problemas críticos al construir estos tipos de sistemas y distintas propuestas de solución a los mismos. Al analizar esta temática es importante que nos preguntemos cual es el objetivo de la construcción de estos sistemas y encontramos tres posibles puntos [mot2002], que son:

- **Interconexión** (conectar a nivel de hardware distintos sistemas entre sí);
- **Integración** (se refiere a unificar la información que se encuentre replicada, resolviendo conflictos de heterogeneidad);
- **Interoperabilidad** (se refiere no sólo a integrar información, sino que también a integrar funcionalidades, como por ejemplo ABM de los datos en la fuente indicada).

En el caso de la generación de reportes con datos que provienen de diferentes fuentes el concepto de sistemas multifuentes que mejor se adapta son los sistemas de integración, puesto que nuestro sistema requerirá integrar información que proviene de diferentes fuentes no operaciones o mantenimientos de los datos.
Las arquitecturas propuestas para la integración de fuentes de información son muy variadas, acá comentaremos algunas de ellas

2.5.1 Integración a Nivel de Bases de Datos: Base de Datos Federadas

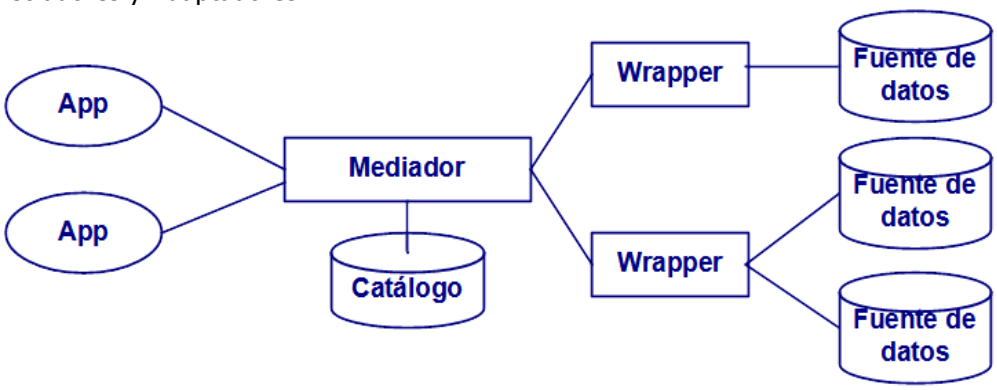
Las Bases de datos federadas son una colección de sistemas de bases de datos independientes, cooperativos, heterogéneos, que son autónomos y que permiten compartir todos o algunos de sus datos.
Se puede tener varios niveles de integración en la Arquitectura de un Sistema Federado según Sheth y Larson [sl90].
En el nivel más simple (Nivel 1) cada sistema aporta alguna parte al sistema global sin distinguir objetos comunes entre los sistemas, el primer paso consiste en estandarizar todos los esquemas en uno común.
El Nivel 2 de integración, se integran subesquemas que tienen correspondencia de equivalencia, en un único subesquema homogéneo con los datos funcionados.
En el Nivel 3, tenemos los sistemas que no solo comparten datos sino también funcionalidades.
En el nivel 4 se encuentra el **Esquema Federado** que corresponde al **esquema integrado** de la federación.
Por último, en el nivel 5, los **Esquemas Externos,** el resultado de transformar el esquema federado a los modelos de datos que usa cada uno de los sistemas participantes.



Arquitectura de 5 niveles de Sheth y Larson.

2.5.2 Arquitectura de Wrappers y Mediadores

Otra arquitectura que apunta a resolver el problema de integración de fuentes de información y repositorios distribuidos es la arquitectura de Wrappers y Mediadores, o también llamada de Administradores y Adaptadores.



En esta figura se muestra la arquitectura de Wrappers y Mediadores, donde tenemos varias fuentes de información heterogéneas y cada fuente tiene su propio wrapper.
Los wrappers se encargan de proveer acceso a las fuentes y convirtiendo esos datos en un formato común.
El mediador puede procesar, integrar y almacenar la información aportada por los wrappers, en una vista unificada de todos los datos disponibles. Cuando una aplicación realiza una consulta, el mediador descompone la misma en consultas más pequeñas, las cuales son enviadas a los wrappers que correspondan para que estos las ejecuten en sus propias fuentes de datos, luego

se obtienen los resultados de las consultas y se unen estos resultados de forma de satisfacer la consulta original de la aplicación.

2.5.3 Arquitectura orientada a servicios (SOA)

Las exigencias de tiempo, presupuesto y procesamiento cada vez más limitado, nos llevan a la necesidad de optimizar los recursos, una de las optimizaciones posibles está en la implementación de sistemas re-usables. Si muchas de las funcionalidades ya están implementadas, las alternativas para agregar nuevas operaciones son [par2005]:

1. **Reutilizar** la funcionalidad ya implementada en otros sistemas. Si el sistema no está pensado para este fin es una labor difícil de realizar, además si le agregamos el problema de la falta de documentación o lo poco actualizada que ésta se encuentra, en muchos casos es más complicado entender lo que se hace para reutilizarlo que hacerlo de nuevo.
2. **Reimplementar** la funcionalidad requerida ("reinventar la rueda"). Aunque implica más tiempo de desarrollo, es la más fácil segura y generalmente la escogida. Esto trae como resultado funcionalidad replicada en la aplicación por no haber una estrategia de integración de aplicaciones, obteniéndose así un sistema con pobre respuesta al cambio.

Por lo tanto la mejor opción es desarrollar los sistemas pensando en base a integración tanto de datos como de operaciones. Una buena estrategia para ello es motivar la construcción de servicios que proveen las funcionalidades definidas para la aplicación, más que en el desarrollo de aplicaciones. De esta manera, una aplicación final simplemente orquesta la ejecución de un conjunto de estos servicios, añade su lógica particular y le presenta una interfaz al usuario final. Esta Arquitectura requiere la definición de los servicios que componen el sistema, sus interacciones, y con que tecnologías serán implementados. Las interfaces son gobernadas por contratos que indican claramente el conjunto de mensajes soportados, su contenido y las políticas aplicables.

Cada servicio es una aplicación completamente autónoma e independiente, teniendo una interfaz de llamadas basada en mensajes. Generalmente, los servicios incluyen tanto lógica de negocios como manejo de estado de los datos. De esta forma si se genera un servicio que obtenga información de diferentes fuentes, este servicio puede ser usado tanto para la generación de reportes como para mostrar esos datos directamente en pantalla.

El manejo de varios servicios integrados como si fuera uno solo, llamado servicio de negocio o **macroservicios,** podría dejarse a cargo de otro servicio más que normalmente se llama orquestador, encargado de organizar todos los macroservicios, en vez de tener piezas del proceso entre todos los servicios. Esto disminuye las dependencias entre servicios y las aplicaciones clientes, con lo que se hace más fácil la administración del sistema.

2.5.3.1 Estrategias de Implementación de SOA

El primer gran problema en la implementación de los servicios son los protocolos de comunicación, la idea es que estos protocolos sean los más flexibles y que permitan la integración de sistemas [sch2005]. Existen diferentes formas de comunicación entre los servicios, alguna de ellas son:

- **Web Services XML (WS)** tiene como ventajas que es un protocolo abierto, independiente de ubicación, fácil de implementar y promueve una arquitectura desacoplada; la desventaja es que su rendimiento es bajo.
- **Web Services XML con Web Services Extensions (WSE)** el cual permite configurar un canal binario sobre el que va toda la mensajería SOAP, además provee muchas de las características requeridas por SOA implementando las especificaciones WS-Security, WS-Routing, WS-ReliableMessaging, entre otras.
- **Remoting** las ventajas que tiene son muy buen rendimiento si se utiliza un serializador binario sobre TCP, pero como desventajas que utiliza un protocolo propietario.

Los Web Services Extensions es una estrategia a que involucra la implementación de la versión 1.2 de SOAP y de las especificaciones conocidas como WS-*. Las cuales se encargaran de darle la seguridad, confiabilidad y demás requisitos de los sistemas basados en Web Services. Estas especificaciones están dadas por la organización WS-I.

2.5.3.1.1 Web Services

En la actualidad, los Servicios Web es uno de los mecanismos más exitosos en la implementación de procesos distribuidos o integración de sistemas, y la que mejor se adapta a la arquitectura SOA.
Las aplicaciones se construyen utilizando múltiples servicios Web XML desde diversas fuentes que trabajan conjuntamente con independencia de dónde residen o cómo hayan sido implementadas.
Probablemente existan tantas definiciones de los Servicios Web XML como compañías que los desarrollan, pero casi todas las definiciones tienen estos aspectos comunes, como lo muestra la siguiente imagen:

Es así que podemos definir un servicio Web XML como un servicio de software expuesto en la Web mediante SOAP, descrito con un archivo WSDL y registrado en UDDI.
Simple Object Access Protocol (SOAP) es el protocolo utilizado en la mayoría de los Servicios Web para exponer sus funcionalidad.
Web Services Description Language (WSDL) es un documento XML que proporciona un modo de describir sus interfaces, presentando suficiente detalle como para permitir a un usuario construir una aplicación cliente para comunicarse con ellos.
Universal Discovery Description and Integration (UDDI) permite registrar los Servicios Web de modo que los potenciales usuarios puedan encontrarlos.
Una de las ventajas principales de la arquitectura de Servicios Web es que permite a los programas escritos en diferentes lenguajes y sobre diferentes plataformas comunicarse entre sí de un modo basado en estándares, con una complejidad bastante menor a otras opciones. La otra ventaja significativa que tienen los servicios Web XML sobre anteriores iniciativas es que trabajan con protocolos Web estándar: XML, HTTP y TCP/IP.

2.5.3.1.2 SOAP

SOAP (Simple Object Access Protocol) es el protocolo de acceso a objetos simple. La versión actual es la 1.1 y se puede encontrar en www.w3.org/tr/soap. Este es el protocolo de comunicaciones para los Servicios Web, basado en XML y describe un formato de mensajería para la comunicación entre equipos. Contiene también varias secciones opcionales que describen las llamadas a métodos (RPC) y proporcionan detalles sobre el envío de mensajes SOAP a través de HTTP.

Seguridad en SOAP

En las primeras etapas de su desarrollo SOAP se percibía como un protocolo basado en HTTP, y por ello se supuso que la seguridad HTTP sería adecuada para SOAP. Pero actualmente existen especificaciones en trabajos basados en SOAP para proporcionar funcionalidades de seguridad adicionales para los servicios Web. La especificación WS-Security define un sistema de encriptación completo.

2.5.3.1.3 WSDL

WSDL significa Web Services Description Language. Podemos decir que un archivo WSDL es un documento XML que describe un conjunto de mensajes SOAP y cómo se realiza el intercambio de mensajes. El archivo WSDL define todo lo necesario para escribir un programa que interactúe con un servicio Web.

2.5.3.1.4 UDDI

Universal Discovery Description and Integration permite obtener catálogos de negocios de Internet, obtener información sobre sus servicios ofrecidos y contactar con alguien para más información. Es decir que UDDI permite encontrar negocios los cuales podemos obtener Servicios Web.
Naturalmente, también podemos ofrecer un Servicio Web sin registrarlo en UDDI, pero si deseamos llegar a un mercado significativo, necesitamos UDDI para que los clientes puedan encontrarnos.
UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. El directorio UDDI incluye varias formas de buscar los servicios que necesitamos para construir aplicaciones, proporciona información, contactos, enlaces e información técnica para permitirnos evaluar qué servicios satisfacen nuestros requerimientos.

2.5.4 Conclusión

Se analizaron una serie de arquitecturas e implementaciones desarrolladas, fundamentalmente con el fin de solucionar el problema de integrar sistemas de información heterogéneos. Estas arquitecturas no son excluyentes una de otra, si no que en un determinado sistema se puede aplicar una de ellas y en otro entorno usar otra arquitectura diferente, también podrían combinarse para solucionar un problema común.
La ventaja que tienen los Web Services sobre las otras tecnologías es que utiliza protocolos abiertos, lo cual le da una gran compatibilidad con productos de distintas empresas, además de tener solucionado una amplia gama de problemas como seguridad, comunicación, interoperabilidad etc. Las desventajas es que su rendimiento es más bajo que el de un protocolo binario y en algunos caso esto puede ser un punto critico en un sistema. De todas formas, al llevar la teoría a la práctica, las implementaciones de WS no se apegan al estándar, por lo que la integración de los WS que utilizan distintas implementaciones suelen ser dificultosas.
Para la implementación de estas arquitecturas existen varias tecnologías, entre éstas resultan de especial interés Web Services, Remoting, COM, COM+ entre otras, dada la orientación de cliente a las tecnologías de Microsoft y en especial .Net.
También se utilizará la arquitectura de wrappers y mediadores, que permite la integración de sistemas existentes.

2.6 LENGUAJES

Un lenguaje está formado por un conjunto de símbolos básicos (alfabeto) y un conjunto de reglas que especifican como manipularlos [bre2003]. También se debe definir un significado a las cadenas formadas por los símbolos básicos.

2.6.1 Traductores y Compiladores

Cuando pensamos en crear un lenguaje de programación para interactuar con un cierto ambiente, este se debe traducir a datos que el sistema está preparado para almacenar o en órdenes las cuales el sistema está preparado para ejecutar.
Según el conjunto de órdenes los lenguajes de programación pueden clasificarse en lenguajes de:

- **bajo nivel**, ordenes más parecidas al lenguaje de la maquina, y;
- **alto nivel**, ordenes más parecidas al lenguaje humano y en particular ingles.

Otra clasificación se realiza según sean interpretados y compilados.
Un traductor es un programa que toma el texto escrito en un lenguaje (el lenguaje fuente) y lo convierte en el texto equivalente en un segundo lenguaje (el lenguaje destino u objeto), generalmente en los lenguaje de programación el de entrada es de más alto nivel que el de salida.
Un ensamblador es un programa que traduce de un lenguaje ensamblador a lenguaje máquina, mientras que un compilador es un programa que traduce de un lenguaje de alto nivel a un lenguaje de bajo nivel o a lenguaje máquina.

2.6.2 Generación de Compiladores

La traducción puede ser realizada en forma "manual" por un programador, el cual implementa un sistema que lee cada comando y una serie de parámetros, y selecciona de una lista una

tarea a ejecutar; otra técnica es crear un traductor o un compilador en base a la sintaxis y la semántica del lenguaje, este sistema será quien convierta un programa escrito en ese lenguaje en una serie de ordenes o información a operar.
A estos programas, que permiten crear traductores, se les suele llamar compiladores de compiladores o Meta-compiladores y son programas que reciben como entrada las especificaciones del lenguaje para el que se desea obtener un compilador y generan como salida el compilador para ese lenguaje.

2.6.3 Diseño de Lenguajes de Programación

El lenguaje de programación puede definirse describiendo:

- Las expresiones que pueden aparecer en sus programas (la sintaxis del lenguaje)
- Lo que significan sus programas (la semántica del lenguaje).
- La sintaxis del lenguaje se presenta ampliamente con una notación denominada gramática libre de contexto o BNF. (Backus-Naur Form).
- La semántica del lenguaje es más difícil de expresar que la sintaxis y generalmente se decide por especificarla usando descripciones informales y ejemplos.

2.6.3.1 Compilador.

Cualquier compilador debe realizar dos tareas principales: análisis del programa a compilar y síntesis de un programa en lenguaje maquina que, cuando se ejecute, realizará las actividades descritas en el programa fuente.
Para el estudio de un compilador, es necesario dividir su trabajo en fases. Cada fase representa una transformación al código fuente para obtener una representación intermedia o el código objeto, las etapas más comunes son:

- **Análisis Léxico** en esta fase de análisis léxico se leen los caracteres del programa fuente y se agrupan en cadenas que representan los componentes léxicos. Cada componente léxico es una secuencia lógicamente coherente de caracteres relativa a un identificador, una palabra reservada, un operador o un carácter de puntuación. A la secuencia de caracteres que representa un componente léxico se le llama lexema o token.
- **Análisis Sintáctico:** aquí los componentes léxicos se agrupan en frases gramaticales que el compilador utiliza para sintetizar la salida.
- **Análisis Semántico:** la fase de análisis semántico se intenta detectar instrucciones que tengan la estructura sintáctica correcta, pero que no tengan significado para la operación implicada. **Generación de código Intermedio**: algunos compiladores generan una representación intermedia explícita del programa fuente, una vez que se han realizado las fases de análisis. Se puede considerar esta operación intermedia como un subprograma para una máquina abstracta. Esta representación intermedia debe tener dos propiedades importantes: debe ser fácil de producir y fácil de traducir al programa objeto.
- **Optimización de Código**: se trata de mejorar el código intermedio, de modo que resulte un código de máquina más rápido de ejecutar.
- **Generación Automática de Código:** esta constituye la fase final de un compilador. En ella se genera el código objeto.

2.6.4 Conclusiones

Para el desarrollo del lenguaje se hace necesaria la utilización de un generador de compiladores. De las herramientas analizadas se eligió **Coco/R** (http://www.ssw.uni-linz.ac.at/Coco/), por la similitud de esta con herramientas utilizadas anteriormente y la estabilidad que presento la herramienta en las pruebas iniciales que se realizaron.

3 SOLUCIÓN PROPUESTA

El objetivo final es que un usuario final pueda generar sus propios reportes a través de nuestro sistema. Para lograrlo nos propusimos, como se menciona más arriba, crear una representación de modelos de negocio, es decir un meta-modelo de negocio (**MMN**), esta solución también permite integrar los datos provenientes de distintos sistemas de información. Sobre MMN implementamos las aplicaciones que habilitan a crear y editar instancias (**esquemas**) del meta-modelo, y a partir de éstas ejecutar los reportes definidos dentro del modelo. En las siguientes secciones se describirá el meta-modelo propuesto y su funcionamiento.

Uno de los objetivos primordiales es que los datos para estos reportes se puedan obtener de fuentes de información variadas, para esto es que se propone una arquitectura de "Adaptadores" y "Administradores", a partir de los cuales se integran los datos de la forma que se indique en un esquema de MMN, esto también se describe en las siguientes secciones.

Además, la solución propuesta debe permitir que se utilicen distintos generadores de reportes, para esto se propone una solución análoga, en la cual la interfaz con el reporteador es realizada por el adaptador.

Los componentes de la solución son los siguientes:

- meta-modelo de negocio (definición del meta-modelo y su representación en XML)
- lenguaje de edición del meta-modelo (definición de los comandos utilizados),
- representación genérica de los reportes (definición de la representación),
- editor del meta-modelo (aplicación mediante la cual se edita el meta-modelo),
- integrador de fuentes de información (permite establecer la comunicación con los sistemas de información, y obtener y procesar los datos para los reportes),
- adaptador de reportador para MS Reporting Services (se encarga de la generación de los reportes utilizando MS RS).

Primero describiremos en forma general como funciona la solución propuesta, para luego pasar a desglosar las características y funcionamiento de cada componente:

- un programador debe codificar los adaptadores de las **fuentes de información**, en caso de que estas los requieran;
- el programador debe crear el **esquema** del MMN;
- definir como se accede a las fuentes de información que se utilizarán en el esquema recién creado;
- definir las entidades servidas que serán la base del resto del modelo. Los datos para estas entidades se obtienen a partir de los **servicios** que el programador define;
- definir las entidades relacionales, las cuales obtienen sus datos a partir de **relaciones de expresión** con otras entidades;

La forma en que se definen los servicios y relaciones de expresión se detalla más adelante.

Una vez que un usuario programador definió el esquema que se quiere utilizar como base, éste está pronto para ser utilizado por el usuario final, que puede:

- definir nuevas entidades, tanto relacionales como servidas;
- generar reportes a partir de las entidades definidas por él o las ya creadas por el programador.

Los servicios mencionados son *operaciones de las fuentes de información*. Por ejemplo, un sistema puede tener una operación ObtenerClientes, que reciba algunos parámetros, la cual puede ser utilizada para obtener datos de la entidad que representa a los clientes.

El programador también puede definir las entidades en base a sus relaciones con otras entidades por ejemplo, podría definir una entidad "clientes mayores de 50" en base a una restricción sobre el atributo edad de la entidad "clientes", la cual se establecería en una expresión de relación.

El programador debe tener conocimiento de las fuentes de información que se utilizarán en el modelo y como se integran sus datos, además de saber como es que deben presentar los datos al usuario final. Así, el esquema de meta-modelo creado representa el modelo del negocio de un cliente, y será utilizado por el cliente para analizar los datos, extender el modelo y crear

nuevos reportes.

En el siguiente esquema se presentan los componentes que participan en la solución presentada.

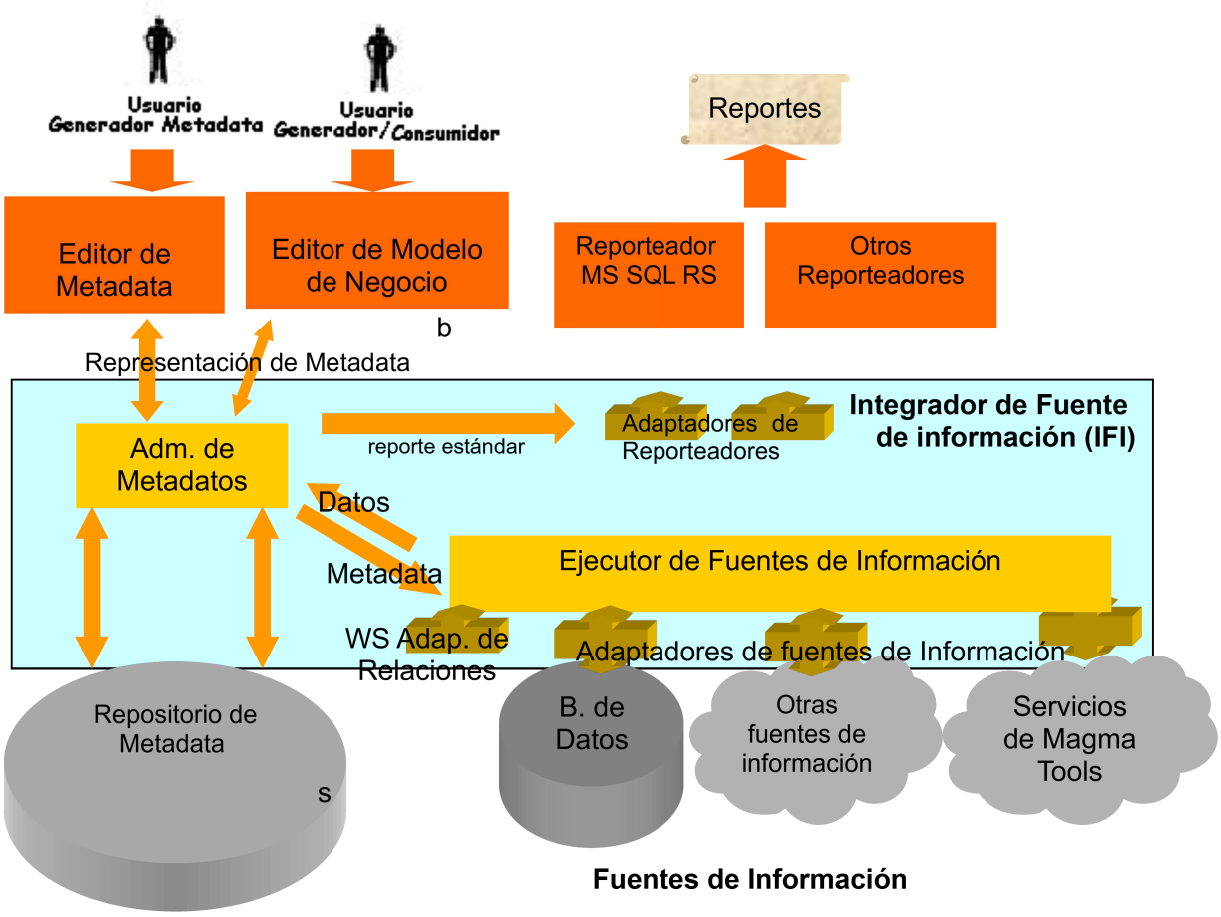


Diagrama de la Solución

Cuando la definición del MMN por parte del programador está concluida, entonces el modelo está pronto para ser utilizado por un usuario final en la creación y ejecución de nuevos reportes. La herramienta utilizada por el usuario final es el Editor del Modelo de Negocio. Esta herramienta le permite extender el meta-modelo creado por el programador, y ejecutar los reportes en base a las estructuras definidas en el meta-modelo, y en base a estas estructuras se obtienen datos según los servicios definidos, se los integra y procesa para luego obtener los datos de las entidades que se presentaran en los reportes y crear la instancia del reporte según los formatos de presentación definidos en la definición de éstos.

Cuando el programador o el usuario final modifican los esquemas (o instancias) del meta-modelo, los editores que utilizan (Editor de Metadata y Editor del Modelo de Negocio) envían los comandos al Administrador de Metadata (AM). Éste se encarga de procesar las instrucciones recibidas para reflejarlas en las representaciones de los esquemas.

Un caso particularmente importante de los comandos es cuando se solicita ejecutar un reporte. En este caso el AM envía una solicitud al Ejecutor de Fuentes de Información (EFI), el cual se encarga de obtener los datos necesarios para el reporte. Cuando estos datos están disponibles, el AM los envía junto con la representación del reporte al Adaptador de Reporteador, que se encarga de generar el diseño del reporte y ejecutar el reporte en el reporteador relacionado.

La obtención de los datos por parte del EFI, implica la ejecución de los servicios o relaciones definidas para las entidades del modelo. Ejecutar los servicios lleva a invocar las operaciones de las fuentes de información y realizar el mapeo de los datos obtenidos, con la información que se encuentra en el modelo, presentándose los datos de las distintas fuentes de información en forma homogénea. Cuando los datos a presentar en el reporte surjan de entidades relacionales

el procesamiento también es llevado a cabo por el EFI.

No solo es importante la selección y obtención de la información a ser mostrada en los reportes, la presentación que se le da a la información también es fundamental, dentro de la representación de reporte es posible indicar el formato en que se presenta la información, y existe la posibilidad de crear plantillas con los formatos deseados para aplicarlas a reportes creados posteriormente.

3.1 META-MODELO DE NEGOCIO

El meta-modelo **MMN** tiene como objetivo poder mostrar al usuario el modelo como él lo maneja habitualmente, ocultando la representación que se utiliza en los sistemas de información, y a partir de esto definir los reportes que desee.

Por otra parte MMN utiliza una representación en XML, esto permite una mayor interoperabilidad y un manejo del modelo más abierto, dada la gran cantidad de herramientas que permiten procesar XML en los diferentes ambientes.

Dentro del meta-modelo, los conceptos del MN se representan como **entidades** del meta-modelo. Estas entidades tienen **atributos** (nombre, teléfono del cliente, etc.) y existirán **relaciones** entre ellas. A partir de estos objetos se representan los conceptos del MN y como éstos se vinculan.

Las entidades se pueden ver como conceptos análogos a los objetos de UML, al igual que sus atributos y relaciones, corresponden a las mismas estructuras en UML. MMN agrega características a estos conceptos, que permiten determinar de qué forma y de que sistemas de información se obtienen los datos relacionados a las entidades.

Desde el punto de vista de un data warehouse el modelo permite representar dimensiones, con sus niveles y medidas, permitiendo a quien edita el meta-modelo crear sus propios niveles y medidas, y generar nuevos reportes. Para esto el modelo de metadata utilizado representa la forma que se calculan los niveles y medidas definidos por el usuario.

Para indicar los sistemas de información utilizados dentro del meta-modelo se utilizan las definiciones de **fuentes de información (FI)**. Estas definiciones están formadas por una serie de parámetros (que se describen más adelante) que dependen del tipo de la fuente (Servicio Web, Message Queue, Remoting, etc.) y que permiten acceder a ella. Más adelante se presentan algunos ejemplos de estas definiciones.

Dentro del modelo existen tres tipos de entidades:

- **Servidas**, cuando obtenga sus datos de una FI. Para estas entidades se deben definir **servicios**, los que indican de que FI y como se obtienen los datos para la entidad.
- **Relacionales**, aquellas cuyos datos se obtengan a partir del procesamiento de los datos de otras entidades. La forma en que se obtienen los datos de este tipo de entidades se indicará a través de **expresiones de relación**.
- **de reporte**, las cuales permiten la declaración de variables dentro de las expresiones de relación o servicios, y también en los vínculos. Además, éstas no podrán ser referenciadas desde otras entidades.

También se utilizaran **vínculos**, éstos permitirán asociar los resultados, tanto de servicios como de expresiones de relación con los atributos de las entidades, es decir que definen el cálculo a realizar para obtener los datos relacionados al atributo.

Los reportes se generarán a partir de los distintos tipos de entidades. A partir de la definición de éstas se obtendrán sus datos, y se les podrá asociar el formato con el que se desea que se ejecute el reporte.

Volviendo al ejemplo que se plantea al comienzo, se definirían las entidades "clientes" y "clientes mayores de 50", para ambas entidades se tendrían los atributos: nombre, doc. de identidad, compras, edad y teléfono. También se establecería entre ambas una relación de especialización que indica que "clientes mayores de 50" es un subconjunto de los datos de "clientes". Tanto atributos como relaciones y entidades tienen otras características que se discuten en los siguientes párrafos.

Por otro lado, el MMN debe establecer un vínculo entre los datos que están presentes en los sistemas de información y la forma en que estos datos quieren ser presentados al usuario. Esto

se resuelve mediante la definición de **servicios**, **expresiones de relación** y **vínculos**.

Los sistemas de información que se usan en un determinado esquema de MMN se declaran a través de la definición de las **fuentes de información**. En la versión actual las fuentes de información pueden ser sistemas del tipo de Web Services, servicios de Remoting o bibliotecas de ensamblados. Para definir el acceso a una fuente de información se deben dar los parámetros indicados según el tipo de fuente que esta sea. Por ejemplo, si se utiliza un Web Service como fuente de información, una definición en MMN sería similar a la siguiente:

```
<FuenteDeInformacion IdFuente="1" Nombre="clientes_ARG_ws" Tipo="webservice">
  <Parametro
Nombre="url">http://localhost/WS_CRM_Arg_Uru/WS_CRM.asmx</Parametro>
</FuenteDeInformacion>
```

En el caso de la utilización de un ensamblado, la definición es como se muestra a continuación:

```
<FuenteDeInformacion IdFuente="0" Nombre="clientes_BRA_dll" Tipo="dllfile">
  <Parametro Nombre="dll-name">CD_CRM_Arg_Uru.dll</Parametro>
  <Parametro Nombre="clase">CapaDatos.AccesoADatos</Parametro>
  <Parametro Nombre="path">D:\CRM_Arg_Uru\CD_CRM_Arg_Uru</Parametro>
</FuenteDeInformacion>
```

Los parámetros que definen la FI dependen de su tipo, dado que la forma en que se accede una FI depende de su tipo. Estos parámetros son interpretados por el proveedor de adaptadores según el tipo de la fuente. Para extender los tipos de FI utilizados se debe extender el proveedor de adaptadores (esto se comenta en los trabajos a futuro más abajo).

Los **servicios** son *operaciones de los sistemas de información* que se referencian en el esquema. Para utilizarlos se deben haber definido previamente las FI en el esquema. En ellos se especifica como invocar una operación en una fuente de información, y como se pueden utilizar parámetros que modifiquen los datos que se obtienen. Para una entidad se pueden definir tantos servicios como se deseen, de esta forma cada servicio obtiene los datos de la entidad que se encuentran en los distintos sistemas.

Siguiendo con el ejemplo de los clientes, supongamos que se tienen las operaciones ObtenerClientes, en la fuente en Argentina que no recibe parámetros y GetClientes en Uruguay, que recibe dos parámetros que corresponden filtros por las edades máximas y mínimas. Así, la entidad "clientes" se define con dos servicios uno obtiene los datos de la fuente de información en Argentina y el otro en Uruguay. Para el servicios de la FI uruguaya se deben indicar que los parámetros que se pasan al invocar el servicio son -1, asumiendo que de esta forma no se aplican filtros a los datos, en el caso de la fuente de Argentina, la invocación de la operación no lleva parámetros, solo debe indicar sobre que FI se realiza. Las definiciones de los servicios, dentro de la definición de la entidad cliente, serían de la siguiente forma:

```
<ColServicios Autogenerado="2">
  <Servicio IdServicio="1" Nombre="ObtenerClientes" IdFuente="4">
    <ColParametros Autogenerado="0" />
    <ColVinculos Autogenerado="0" />
  </Servicio>
  <Servicio IdServicio="0" Nombre="GetClientes" IdFuente="5">
    <ColParametros Autogenerado="2">
      <Parametro IdParametro="1" Nombre="EdadMinima" Valor="-1" />
      <Parametro IdParametro="0" Nombre="EdadMaxima" Valor="-1" />
    </ColParametros>
    <ColVinculos Autogenerado="0" />
  </Servicio>
</ColServicios>
```

En estas definiciones faltan los vínculos de los resultados obtenidos de las FI con los atributos de las entidades. Más adelante veremos como se establecen estos vínculos.

Por otro lado las **expresiones de relación** se utilizan cuando las entidades son **relacionales**, en este caso los datos surgen a partir de estas expresiones de relación. Ellas indican como se procesan los datos de las entidades con las cuales la entidad está relacionada. Mediante estas entidades se declaran nuevos conceptos que no se representan directamente en los sistemas de información, o aquellas cuyos datos impliquen un procesamiento, como puede ser el filtrado según algún criterio (por ejemplo la edad, en el caso que venimos manejando) o el resumen, por sumas, promedios, valores mínimos o máximos.

Un ejemplo de entidad relacional podría ser "clientes mayores de 50", esta entidad se definiría de la siguiente forma:

```
<Entidad IdEntidad="1" Nombre="clientes_mayores_de_50"
Tipo="EntidadRelacional">
  <ColExpresionesRelaciones Autogenerado="2">
    <ExpresionRelacion IdExpresionRelacion="0" Nombre="condidcion">
      <CondicionIndividual Valor="RELclientes.Edad&gt;=50" />
      <Ordenar />
      <Agrupar>
      <CondicionGrupal Valor="" />
      </Agrupar>
      <ColVinculos Autogenerado="9" />
    </ExpresionRelacion>
  </ColExpresionesRelaciones>
  <ColRelaciones Autogenerado="1">
    <Relacion IdRelacion="0" Nombre="RELclientes" IdEntidadForanea="0"
IdAtributoLocal="6" IdAtributoForaneo="0" Cardinalidad="1a1"
Tipo="especializacion" />
  </ColRelaciones>
  <ColAtributos Autogenerado="12">
    <Atributo IdAtributo="10" Tipo="int" Nombre="Edad" Multiplicidad="False"
Oculto="False" Horizontal="False" Clave="False" />
    <Atributo IdAtributo="9" Tipo="string" Nombre="Telefonos"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False" />
    <Atributo IdAtributo="8" Tipo="string" Nombre="Identificacion"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True" />
    <Atributo IdAtributo="7" Tipo="string" Nombre="Nombre"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False" />
    <Atributo IdAtributo="6" Tipo="long" Nombre="Id" Multiplicidad="False"
Oculto="True" Horizontal="False" Clave="True" />
    <Atributo IdAtributo="11" Tipo="string" Nombre="Origen"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False" />
  </ColAtributos>
</Entidad>
```

En esta definición se distinguen las colecciones de:

- *atributos*,
- *relaciones,* y;
- *expresiones de relación*.

Se tiene una relación la cual indica que esta entidad es una especialización de la entidad "clientes". Existe una expresión de relación, a partir de la cual se calculan los datos de la entidad. En este caso se establece la restricción ya mencionada sobre la edad de los clientes con una condición individual. Cuando se quieran agrupar u ordenar los datos se podrían establecer condiciones grupales o de orden en la expresión de relación.

Los atributos de las entidades tienen varias características como ser:

- nombre,
- tipo,
- multiplicidad,
- oculto,
- horizontal, o,
- clave.

Estas características dan la posibilidad de mejorar la presentación y desempeño del esquema, ocultando algunos atributos de las entidades, como ser los identificadores internos de los sistemas, pero de todas formas utilizarlos para procesar los datos. Una descripción exhaustiva de su uso se encuentra en el documento de "Discusión de la metadata".

Por otro lado, las expresiones de relación tienen condiciones individuales, grupales y de orden las que permiten operar sobre los datos y modelar las entidades que se deseen. Los posibles usos de estas estructuras también se describen en el documento de "Discusión de la metadata".

Uno de los problemas centrales del proyecto es la generación de reportes por un usuario final

teniendo en cuenta que el sistema no debe quedar atado a un reporteador en particular.

El primer aporte para solucionar estos problemas en la meta-modelo de negocios es mantener una instancia de representación genérica de reportes por cada entidad, es decir que cada entidad tiene una y solo una instancia de representación genérica de reporte y en el momento de generar el reporte esta información debe ser pasada al Adaptador de Reportes, para permitir la generación del diseño del reporte.

En segundo lugar el meta-modelo de negocio también mantiene información sobre las propiedades de los atributos de cada entidad, estas propiedades permiten indicar si un atributo va a ser mostrado o no en un reporte (Oculto) o si los datos de un determinado atributo se mostraran en filas o columnas (Horizontal), permitiendo generar variantes en los reportes.

En tercer lugar el meta-modelo de negocio también mantiene toda la información necesaria para calcular los datos de una entidad, y esta información es la que posteriormente va a ser mostrada en los datos del reporte.

En cuarto lugar el meta-modelo de negocio mantiene los datos de variables y los cálculos donde son referenciadas, generando así una **entidad de reporte**, estas entidades se caracterizan por permitir la definición de **variables** y además por no poder ser utilizadas en expresiones de relación de otras entidades. La utilización de variables da la posibilidad de reutilizar reportes con estructuras parecidas, lo que puede ser útil en casos como los siguientes:

- cuando se quieren generar reportes con estructuras iguales, pero con diferentes valores. Ejemplos de esto podrían ser la generación de un reporte semanal, donde la variable sería la fecha para la que se ejecuta el reporte.
- Al tener entidades de estructuras similares o derivadas unas de otras, la entidad puede ser variable dentro del reporte. Un ejemplo de esto sería un reporte que usa indistintamente la entidad "cliente" o la entidad "clientes mayores de 50", en este caso se define un único reporte con una variable a la que se le asignará el nombre de la entidad deseada.

Más arriba se mencionan algunas características de los atributos de las entidades, describiremos brevemente como pueden ser utilizadas estas características. Por una descripción más especifica ver el documento de "4 - Meta-Modelo de Negocio".

Los atributos claves y los ocultos pueden ser utilizados en conjunto para no mostrar a los usuarios finales identificadores internos que no son de interés para él, pero si permitir que el modelo utilice estos identificadores para el procesamiento de los datos. De esta forma se muestra al usuario solo la información que interesa, y se utilizan índices en forma interna para obtener un mejor rendimiento.

La propiedad **Horizontal** permite indicar cuando se quiere que los valores de un atributo aparezcan en las columnas del reporte que se genera a partir de la entidad a la que pertenece.

3.2 Lenguaje de Edición del Meta-Modelo de Negocio (LEM)

Como forma de edición del MMN, se propone un lenguaje de comandos, al que llamamos Lenguaje de Edición del Meta-modelo. El lenguaje funciona al estilo de las consolas de comando de Linux o el Command Prompt de Windows.

Se optó por un lenguaje de este estilo planeando que sobre LEM se pueda desarrollar una interfaz gráfica que presente el MMN con la amigabilidad y usabilidad necesarias para un usuario final, este también podría ser utilizado por programadores expertos si así lo prefirieren. Es claro que este lenguaje no brinda la amigabilidad necesaria para que un usuario final interactúe con el modelo. Otro punto que llevo a esa decisión es el tener que limitar la complejidad de la solución al tiempo y los recursos disponibles.

El lenguaje se compone de varios comandos, los que no serán descritos aquí en su totalidad, una descripción detallada de cada comando se encuentra en el Documento de *Lenguaje de Edición del Meta-Modelo*.

La mayoría de los comandos corresponden a las altas, bajas y modificaciones de los distintos objetos que conforman el modelo, estos serían: esquema, fuente de información, entidad, relación, expresión de relación, servicio, atributo, formato de reporte. Para cada uno de estos objetos se tienen los comandos Crear*Objeto*, Modificar*Objecto*, Eliminar*Objeto*. Los parámetros para cada uno varían según el tipo de objeto sobre el que actúan.

También se destacan otros comandos como:

- vincular,
- condición,
- agrupar,
- ordenar,
- EjecutarEntidad, o
- GenerarReporte.

Por otro lado los comandos como **seleccionar**, **liberar** u **ObtenerDatosObjetoSeleccionado** se utilizan al operar sobre el modelo pero no lo modifican.

Presentamos la definición del esquema que venimos manejando como ejemplo, para mostrar la forma en que se utilizan los comandos. En algunos casos utilizamos las abreviaciones de los comandos, las cuales permiten utilizar el lenguaje de forma más cómoda.

Creamos un esquema "CRM_Arg_Uru" y dentro de este definimos los accesos a las fuentes de información que utilizaremos:

```
crearesquema c:\CRM_Arg_Uru
crearFuente clientes_BRA_dll, tipo:dllfile, path:D:\CRM_Arg_Uru\
CD_CRM_Arg_Uru, dll-name:CD_CRM_Arg_Uru.dll, clase:CapaDatos.AccesoADatos
crfi clientes_ARG_ws, tipo:webservice,
url:http://localhost/WS_CRM_Arg_Uru/WS_CRM.asmx
crfi clientes_URU_remoting, tipo:remoting, path:D:\CRM_Arg_Uru\
Remoting_CRM_Arg_Uru, rem-name:Remoting_CRM_Arg_Uru.dll,
proxy:RemotingProxy.RemotingProxy, metodoacceso:ObtenerObjeto,
interface:Comun.IAccesoADAtos
```

crfi es la abreviación de crearFuente. A cada una de las fuentes definidas se accede de forma distinta, por lo que para cada definición se tiene los datos adecuados que son necesarios para acceder a la fuente. La primera es un ensamblado de .Net, en el comando se indica esto mediante el tipo (`tipo:dllfile`) y se proveen los datos necesarios para utilizarla, es decir, el path en el que se encuentra, su nombre y la clase dentro de ella que se utilizará. La segunda es un Web Service, por lo que se da su url para acceder a él. En el caso de la tercera se trata de un servicio de remoting, para el cual se debe dar una forma de acceder a un Proxy para el servicio.

Luego definimos la entidad "clientes", con los atributos Id, Nombre, Identificación, Telefono y Origen, también se definen los servicios a partir de los cuales la entidad obtendrá sus datos. Estos servicios acceden a las fuentes `clientes_ARG_ws` y `clientes_URU_remoting`.

```
crearentidad clientes
seleccionar entidad, clientes
crearatributo Id, t:long, c:true, o:true
crat Nombre, t: string, o:false, c:false
crat Identificacion, t:string, o:false, c:true
crat Telefonos, t:string
crat Origen, t:string
```

Todos los atributos deben tener un nombre y tipo. El resto de las propiedades de los atributos asumen los valores por defecto en caso de que no se indique un valor.

Los posibles tipos de los atributos son:

- `string`
- `boolean`
- `decimal`
- `float`
- `duration`
- `datetime`
- `time`

- date
- char
- int
- double
- long

se permiten estos tipos de datos por ser generales y posibles de ser manejados al integrar datos de sistemas desarrollados en distintas tecnologías.

El comando *seleccionar* es utilizado para indicar sobre que objeto del modelo se aplicarán los siguientes comandos, mediante este comando y el comando *liberar* se puede ir recorriendo la estructura del meta-modelo para trabajar sobre el objeto que se desee.

Los servicios y vínculos, que relacionan los datos provenientes de la fuente de información con atributos de la entidad, se definen de la siguiente manera:

```
crearservicio GetClientes, clientes_URU_remoting
seleccionar servicio, GetClientes
vincular Id, cli_id
vincular Identificacion, CI
vincular Telefonos, Tel + '; ' + Cel
vincular Origen, 'Uruguay'
liberar
crse ObtenerClientes, clientes_ARG_ws
seleccionar servicio, ObtenerClientes
vincular Id, 1000000 + id
vincular Nombre, nombres + ' ' + apellidos
vincular Identificacion, DNI
vincular Telefonos, Telefonos
vincular Origen, 'Argentina'
liberar
```

En estas definiciones se realiza el mapeo de los datos obtenidos de las fuentes con los atributos de la entidad. En el caso de Nombre, para el primer servicio no se define un vínculo porque el dato tiene el mismo nombre en la fuente, en el caso del segundo el Nombre se compone de la concatenación de nombres y apellidos.

Al realizarse la definición de los vínculos para los servicios se debe conocer la estructura de los datos provistos por los adaptadores de las fuentes de información. Es por esto que quien debe definir las entidades servidas de un esquema debe ser un usuario con conocimiento de los adaptadores.

Luego de la ejecución de los comandos presentados, se tendrá la entidad `clientes` seleccionada, por esto ahora debemos ejecutar el comando `liberar`, con lo que pasaremos a tener seleccionado el esquema.

```
Liberar
```

Ahora podemos crear la entidad `clientes_mayores_de_50` que será una entidad de relación. Primero definimos la entidad con sus atributos.

```
crearentidad clientes_mayores_de_50
seleccionar entidad, clientes_mayores_de_50
crearatributo Id, t:long, c:true, o:true
crat Nombre, t: string, o:false, c:false
crat Identificacion, t:string, o:false, c:true
crat Telefonos, t:string
crat Edad, t:int
crat Origen, t:string
```

Para después crear la expresión de relación con la que se calculan los datos de la entidad.

```
crearrelacion RELclientes, n:clientes, al:Id, af:Id, c:1a1, t:especializacion
crearexpresionrelacion condidcion, c: RELclientes.Edad >= 50
```

También se pueden utilizar otros comandos para modificar el formato o generar reportes a partir de entidades:

```
crearPlantillaFormatoReporte miNuevoFormato, c:\miNuevoFormato.txt
crearPlantillaFormatoReporte miFormatoABorrar, c:\ miFormatoABorrar.txt
EliminarPlantillaFormatoReporte miFormatoABorrar
ModificarFormatoReporte miNuevoFormato, 29, 21, 3,4, true, true, true,
'Listado de Clientes', 2, 2, 2, 2, 'c:\logo.bmp'
ModificarAnchoColumna Nombre, 5
EliminarFormatoReporte

generarreporte clientes
ejecutarentidad clientes
```

Con el comando **generarreporte** se obtiene un reporte completo con los formatos indicados en la plantilla que tiene asociada la entidad. El comando **ejecutarentidad** obtiene los datos pero son formateados.

3.3 INTEGRADOR DE FUENTES DE INFORMACIÓN

La integración de los datos se realiza en el **Integrador de Fuentes de Información (IFI)**, para lograr integrar los datos nos basamos en la definición de un formato común de los datos, y en una arquitectura de "administradores" y "adaptadores".

El IFI se compone del **Ejecutor de Fuentes de Información (EFI)** y los adaptadores, a los adaptadores se accede a través de un **proveedor de adaptadores**, los cuales pueden ser de tres tipos: Web Services, objetos publicados por Remoting o bibliotecas de ensamblados. Cuando se pide la ejecución de un servicio el EFI levanta el adaptador correspondiente, mediante el proveedor de adaptadores, y le solicita la ejecución de la operación indicada en el servicio. A su vez el adaptador accede a la fuente de información que adapta, obtiene los datos y en caso de ser necesario los transforma al formato utilizado en el IFI.

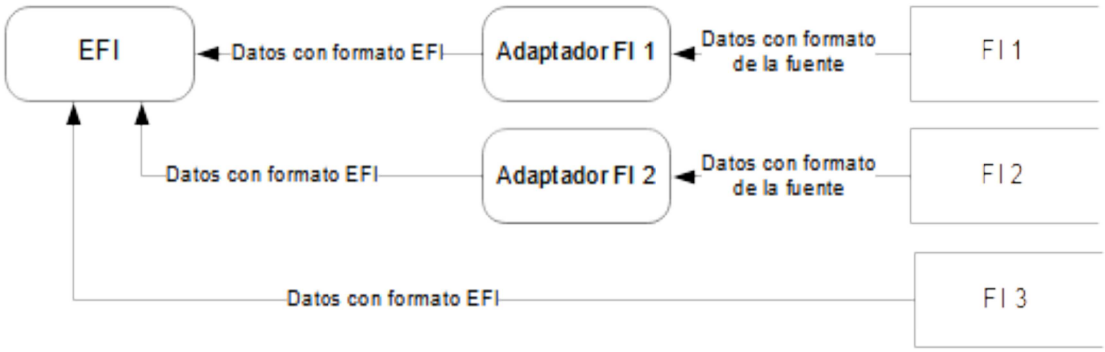
Con la arquitectura de "administradores" y "adaptadores" elegida se hace posible la utilización de nuevas fuentes de información mediante:

- la implementación de un adaptador, que transforme la información provista por la fuente al formato utilizado por el sistema, y;
- la definición de la forma de acceso a la fuente de información en el esquema de MMN desde el que se quiera acceder a ella.

En general los sistemas utilizados como fuentes de información para los reportes utilizan distintos formatos para representar sus dato. Aún dentro de una misma tecnología la representación de los datos puede variar, y son mayores aún las variaciones cuando se consideran sistemas implementados en tecnologías diferentes. Los adaptadores se encargan de convertir los datos del formato utilizado en las fuentes de información al utilizado en nuestro modelo.

Otro punto que se resuelve a través de los adaptadores es la forma de acceso a las fuentes. Las formas en que los sistemas hacen disponibles sus datos son muy variadas, y por esto la comunicación con ellos se debe establecer de distintas formas. La forma en que se realiza esta comunicación también se resuelve dentro del adaptador.

El formato utilizado en el IFI es la representación en XML de los DataSet de .Net con la opción WriteSchema, este formato es compatible con muchas tecnologías y puede ser utilizado con Web Services lo que brinda una gran flexibilidad para interactuar con distintos sistemas. Cada vez son más comunes las arquitecturas orientadas a servicios que utilizan mensajes de XML para comunicarse.



El **proveedor de adaptadores** da la posibilidad de acceder directamente a las fuentes de información que provean los datos en el formato utilizado por el EFI, o tener adaptadores de distintos tipos. Con esto se permite acceder a los adaptadores o las fuentes, cuando éstos están implementados como Web Services, objetos publicados por remoting o bibliotecas de ensamblados locales. Otras formas de acceso a las fuentes de información podrían ser incorporadas en el futuro, mediante la extensión del proveedor de servicios brindado.

Para decidir que servicios de que adaptador se van a requerir se centra la toma de decisiones en un "Administrador". Nuestro administrador es el Ejecutor que se encuentra dentro del EFI, este obtiene los datos de las fuentes de información a través de los adaptadores y luego los procesa de acuerdo a la definición del MMN. Es aquí que luego de obtener los datos el Ejecutor realiza el procesamiento de la información.

3.4 Representación Genérica de los Reportes

También se presenta la posibilidad de usar diferentes reporteadores, esto se soluciona mediante un manejo de "Adaptadores" y "Administradores" similar al que se utiliza con las fuentes de información, donde el Administrador de Metadata invoca los servicios del Adaptador de Reporte que tenga configurado, pasando los datos necesarios, y este llama a su reporteador relacionado, con el fin de ejecutar el reporte con los datos y el formato indicados por el Adaptador.

Para permitir independizar el sistema del reporteador se debió generar una representación genérica de los reportes, esta información es mantenida en el meta-modelo de negocio y más particularmente en cada entidad del esquema, es decir que cada concepto de negocio, además de tener la información propia del concepto que modela y de cómo calcular sus dato, también tiene asociado una representación genérica del reporte, la cual describe como se mostrará cuando se requiera generar un reporte de ese concepto.

Las características más importantes de esta representación son:

- Mantiene los conceptos más importantes en la definición de cualquier reporte.
- Es lo suficientemente genérica como para poder ser usada por cualquier reporteador.
- Tener un formato fácil de comprender para poder ser adatada a cualquier reporteador.

La información manejada en esta representación genérica se puede dividir en dos áreas, que son:

- Formato de la hoja del reporte.
- Formato del cuerpo del reporte.

El **Formato de la hoja del reporte** representa la configuración de la hoja, donde se indican el tamaño de la hoja a imprimir, los márgenes y la información que va en el encabezado y pie de página.

En el **Formato del cuerpo del reporte** se indica: los nombres, ancho, formato de texto y colores de fondo de cada columna.

La implementación de esta representación genérica de reportes se realizo en base a la plantilla **NormalValoresFormato.xml,** la cual aporta el modelo para generar el formato del reporte que es mantenido en el MMN. Esta estrategia de usar plantillas para mantener información es comentada en la siguiente sección.

3.5 GENERACIÓN DE DISEÑO DE REPORTES

El diseño del reporte es donde se indica que datos se van a mostrar, como se van a mostrar y que presentación tendrán esos datos, la tarea fundamental de los reporteadores es combinar el diseño del reporte con los datos del reporte y generar una instancia de reporte.

En la actualidad no existe un estándar para la generación del diseño de reportes, es decir que cada reporteador tiene su propio modelo de diseño de reporte, teniendo en cuenta esto y el requerimiento de que el producto no tiene que estar atado a un único reporteador, se diseño una representación genérica de reportes (comentada en el punto anterior) y el Adaptador de Reporte el cual partiendo de la Representación Genérica de un reporte crea en un diseño de reporte para un reporteador en particular.

El Adaptador de Reporte que se implementó es para RS y la estrategia usada para generar cada diseño de reportes fue basarse en plantillas.

Para nuestra implementación, una plantilla de reporte es un archivo con formato XML y más particularmente RDL, el cual sirve como modelo para la generación de un reporte o un área particular del reporte, la mayor parte de la información de las plantilla no cambia solo cambian algunos valores o se agrega información la cual debe mantener el formato; la plantilla aporta estructura y valores de inicialización a la generación del diseño del reporte.

Por ejemplo en el caso de los atributos, la plantilla mantiene la siguiente información:

```
<TableCells>
	<TableCell>
		<ReportItems>
			<Textbox Name="Id">
				+<Style>
				<Value>=Fields!Id.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
</TableCells>
```

En cambio al generarse el diseño del reporte vemos la información siguiente, donde se mantiene el formato pero se agregaron tanto nodos TableCell como atributos de la entidad existen, colocando el nombre de cada atributo como nombre de un cuadro de texto y de un campo

```
<TableCells>
	<TableCell>
		<ReportItems>
			<Textbox Name="Cedula">
				+<Style>
				<Value>=Fields!Cedula.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
		<TableCell>
		<ReportItems>
			<Textbox Name="Nombre">
				+<Style>
				<Value>=Fields!Nombre.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
		<TableCell>
		<ReportItems>
			<Textbox Name="Direccion">
				+<Style>
				<Value>=Fields!Direccion.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
</TableCells>
```

Existen otras dos plantillas, además de **NormalValoresFormato.xml** que fue comentada en

la sección anterior, esta son la Plantillas Normales RDL y las Plantillas de Formato de Cuerpo.

- **Plantillas Normales de RDL** es en si un diseño de reporte, el cual aporta la estructura básica para generar el nuevo diseño de reporte (*.rdl). Esta plantilla es adaptada para cada nuevo diseño de reporte a generar agregando la configuración de la hoja y el formato del cuerpo, datos indicados en la representación genérica del reporte.

  Existen dos de estas plantillas una para generar reportes en formato matriz y otra para generar reportes en formato tablas (ver documento **Manual de Experto.Doc**).
- **Plantillas de Formato de Cuerpo** son plantilla que determinan como serán mostrados los datos en cuanto a colores, tipos, de letras bordes, sombreados, etc., es decir que permiten cambiar la presentación estética del reporte, si en la instancia de la representación genérica de reporte no se indica ningún formato se asume el formato por defecto que está definido en la Plantilla Normal de RDL.

3.6 ADAPTADOR DE REPORTES

Como se comento anteriormente, el adaptador de reporte es la pieza de software encargada de:

- Generar un diseño de repote para un reporteador en particular,
- Hacer lo necesario para publicar ese diseño de reporte en el reporteador,
- Ejecutar el reporte con el formato y datos especificados.

Para realizar esta tarea el Adaptador de reporteador debe recibir y procesar los datos del reporte y la representación intermedia con la entidad a reportear, esta representación intermedia contiene los atributos a mostrar con sus propiedades y la representación genérica del reporte.

Es por esto que si se quiere generar un adaptador para integrar un reporteador con nuestro producto este debe proveer de una API la cual permita:

- Generar un diseño de reporte desde cero, sin la participación de ningún tipo de entorno o asistente, es decir que se pudiera generar todo el diseño en forma programática,
- Ejecutar un diseño de reporte con los datos propuestos,
- Permita mostrar el reporte ejecutado en un formulario.

Una de las soluciones halladas fue RS que es un reporteador el cual basa el diseño de sus reportes en archivos con formato RDL, los cuales son archivos de texto plano con formato XML, por lo cual pueden ser generados por cualquier aplicación. Además, a través de una interfase con Web Services permite publicar, ejecutar y mostrar los reportes.

Para la primera etapa de generación del diseño del reporte ya se comento anteriormente, en ella se trabajo con el método de plantillas, las cuales se iban adaptando a cada entidad a reportear según la información aportada por la representación intermedia.

Sobre los datos que son mostrados en el reporte se realiza una tarea especial, la cual consiste en extender los orígenes de datos para que acepte DataSet como un nuevo origen de datos, puesto que los orígenes aceptados por RS son limitados.

Esto ultimo no es un requerimiento que deba cumplir un reporteador para que sea adaptado a este sistema, pero si tiene esta posibilidad (aceptar DataSet como origen de datos) simplifica mucho la generación de su Adaptador.

4 TECNOLOGÍAS UTILIZADAS

La aplicación prototipo se utilizó el **.Net Framework**. El uso de esta fue propuesto por Magma como un requerimiento del proyecto de grado, en particular se utilizó el lenguaje C#.NET.

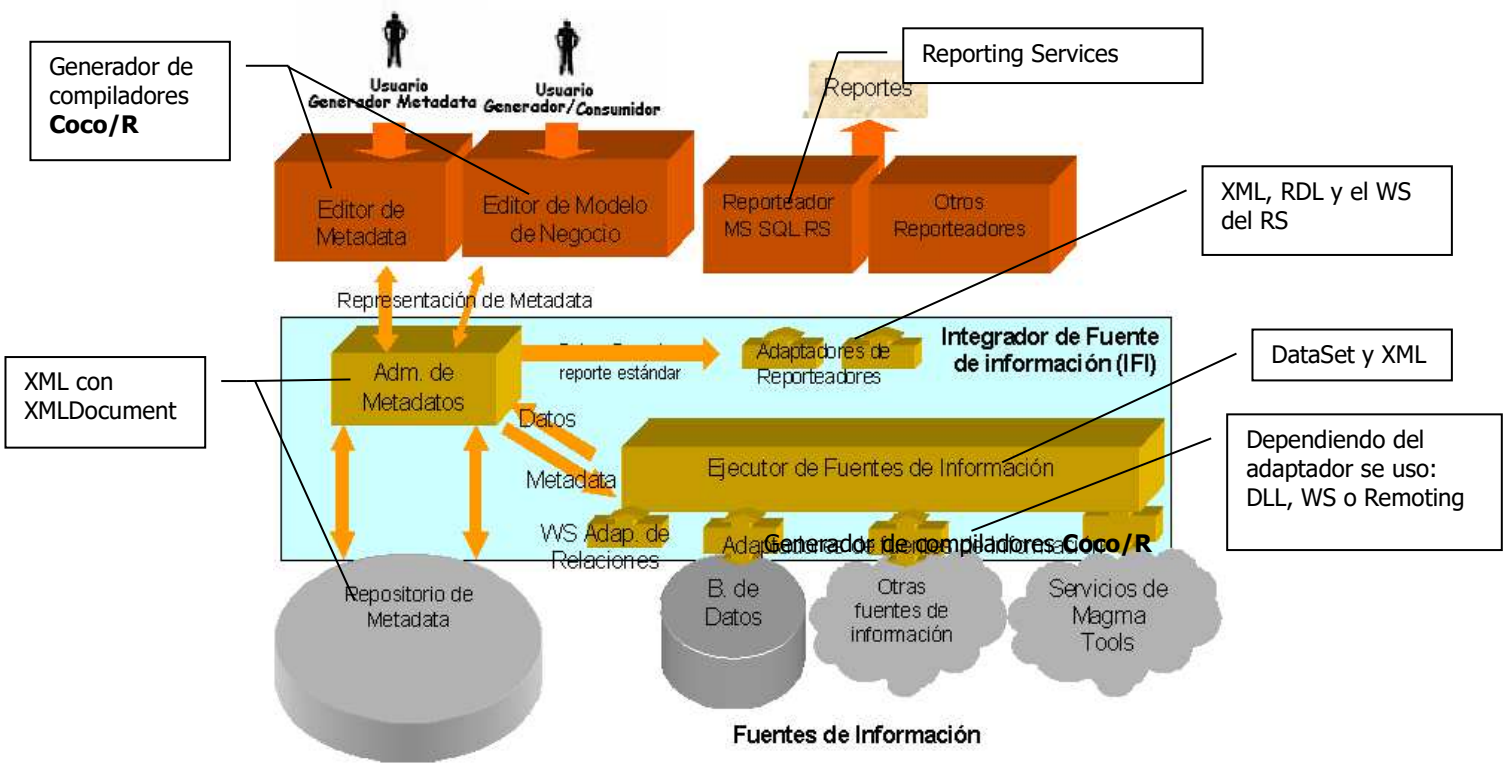
Como generador de reportes se desarrollo un adaptador para **Reporting Services**. Esta herramienta es de particular interés principalmente por las siguientes razones:

- Presentar una representación de los reportes definida y abierta, dando la posibilidad de generar un reporte a partir de los datos y del diseño de reporte en formato RDL sin importar como se generó tal diseño;
- Ofrece una forma de extender los orígenes de datos, lo cual permite agregar diferentes orígenes, como DataSet o un archivo XML;
- Permite publicar y ejecutar el reporte a través de una API abierta, esto lo hace muy ventajoso y simple de integrar con productos independientes, y;
- Por el interés comercial generado por el hecho de que su licencia forma parte de la de SQL Server, por lo que teniendo SQL Server, Reporting Services es gratis.

Para el desarrollo del parser de LEM se utilizó la herramienta **Coco/R** (http://www.ssw.uni-linz.ac.at/Coco/), esto se hizo por la similitud de esta con herramientas utilizadas anteriormente y la estabilidad que presento la herramienta en las pruebas iniciales que se realizaron.

En la versión extendida del estado del arte se pueden encontrar los estudios realizados sobre MS Reporting Services y Coco/R.

En el siguiente diagrama se muestra cual es la solución desarrollada y donde participan las diferentes tecnologías y/o herramientas antes comentadas



5 DETALLES DE IMPLEMENTACIÓN

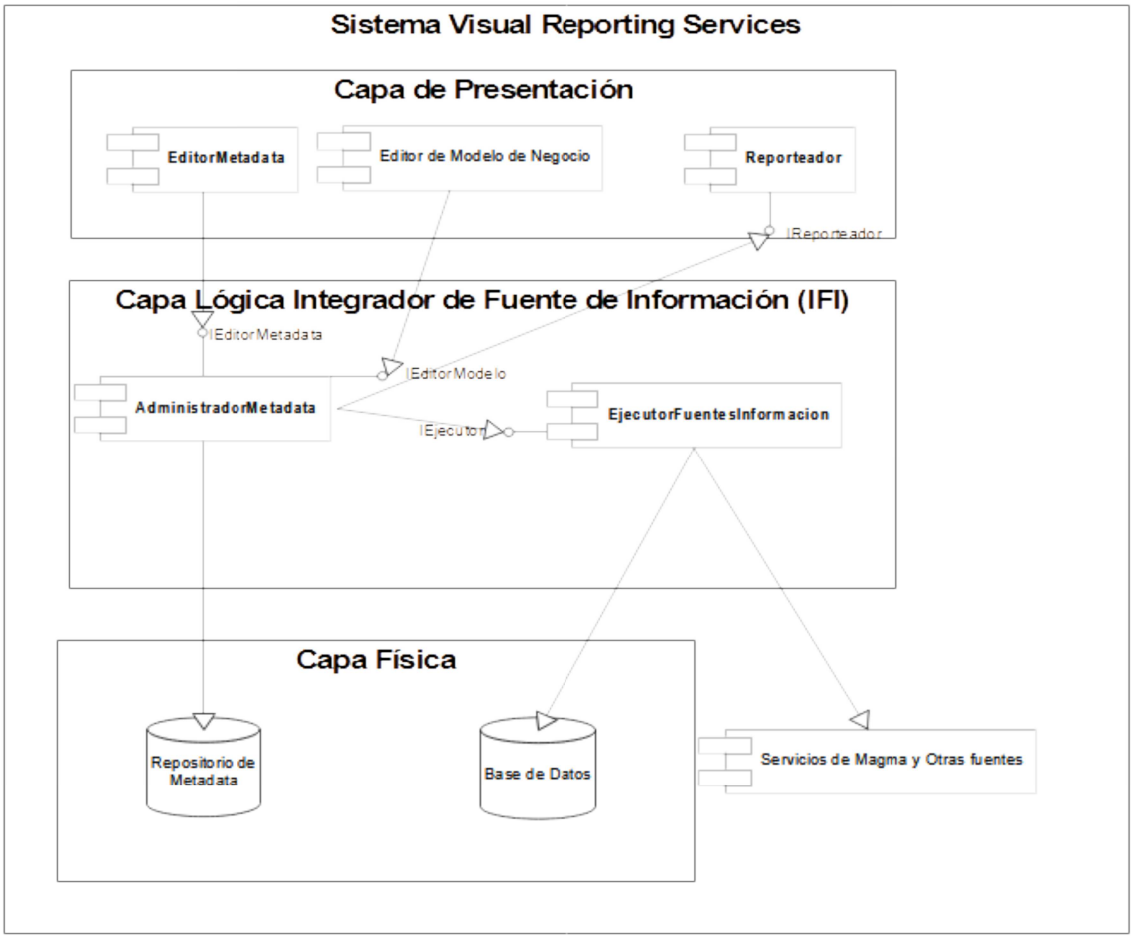
El sistema tiene una arquitectura en tres capas (presentación, lógica y persistencia), que se presentan en el diagrama más abajo junto con los componentes que existen en cada una de ellas.

La capa de presentación se compone por el Editor de Metadata, Editor de Modelo de Negocio y Reporteadores.

La capa lógica, a la que llamamos Integrador de Fuentes de Información (IFI), está formada por el Administrador de Metadata (AM) y el Ejecutor de Fuentes de Información (EFI).

La capa de persistencia se compone por las Fuentes de Información y la persistencia del meta-modelo. Las fuentes de información no son parte de nuestro sistema, si no que son los sistemas de los cuales se obtienen los datos de las entidades, los cuales podrían ser Bases de Datos, servicios de Magma, servicios de MS Remoting, Servicios Web u otras fuentes.

En la siguiente figura se presenta una vista general de las capas y sus componentes.



5.1.1 Capa Presentación

La capa de presentación se compone del Editor de Modelo de Negocio y el Editor de Metadata.

El primero permitirá al usuario final (usuario Generador/Consumidor de reportes) acceder al modelo, interactuar con éste, introduciendo y modificando nuevas entidades y reportes, y generando los reportes que desee. También permite indicar los formatos de los reportes a

generar. Esto se hace mediante un subconjunto de LEM, el que se especifica en el documento **5 - Lenguaje de Edición de Meta-Modelo**.

El segundo editor (Editor de Metadata) permite, al programador, crear y modificar esquemas del MMN, y tiene acceso a todos los comandos de LEM.

Para la implementación de estos dos editores se utilizó un generar de parsers llamado Coco-R.

5.1.1.1 Capa Presentación – Reporteador

Al reporteador utilizado se accede mediante el adaptador del reporteador. De esta forma se independiza el reporteador del resto del modelo, mediante la utilización de una representación de datos y una especificación del formato de los reportes independiente de los reporteadores. Por esto, el Reporteador recibe los datos a presentar en los reportes (calculados a partir de Entidades del modelo, a través de la interacción con las fuentes de información) y el formato del reporte. Con estos datos y a través de un adaptador se crea el reporte en el reporteador utilizado.

En nuestro caso usamos Reporting Services como reporteador, que permite, a través del RDL, indicar los datos y el formato que se desee. Por lo que el adaptador convierte la especificación del reporte que surge MMN al formato RDL y luego produce el reporte mediante Reporting Services. La integración con distintos generadores de reportes depende de las APIs que presente cada reporteador que se desee utilizar.

La integración con otros reporteadores se deja como trabajo a futuro, debido a que habitualmente los reporteadores comerciales utilizan tecnologías y representaciones propietarias, que no están orientadas a la integración con sistemas de este tipo (ver Estado del Arte).

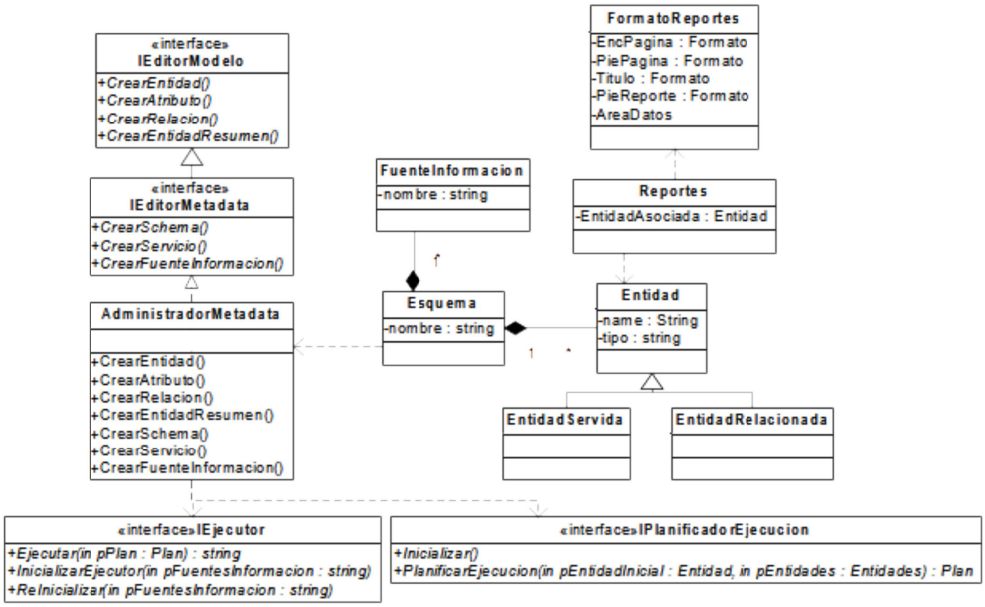
5.1.2 Capa Lógica - Integrador de Fuentes de información (IFI)

El IFI es la capa que vincula a los usuarios con la metadata y las fuentes de información. Está formado por varios componentes que nos permiten procesar las solicitudes provenientes de la interfaz, obteniendo las operaciones a realizar en las fuentes de información a partir del repositorio de metadata.

5.1.2.1 Capa Lógica - Administrador de Metadata

El Administrador de Metadata, encapsula la representación del Modelo de Negocio, encargándose de editar la representación del modelo según las operaciones que se realicen desde la capa de presentación. También actúa en la generación de un reporte cuando esta sea solicitada. Para esto utiliza el Ejecutor de Fuentes de Información (EFI), al que se le solicitan los datos para el reporte a generar. Luego estos datos son enviados al reporteador, donde se formatean según la definición del reporte en MMN.

Se utiliza una estructura intermedia para la representación del Modelo de Negocio, en el siguiente diagrama se muestra solo parte de esta estructura, junto con la estructura del Administrador de Metadata. Cabe aclarar que en las interfaces IEditorMetadata e IEditorModelo no se presentan todos los métodos que en ellas se definen.



5.1.2.2 Capa Lógica - Ejecutor de Fuentes de Información (EFI)

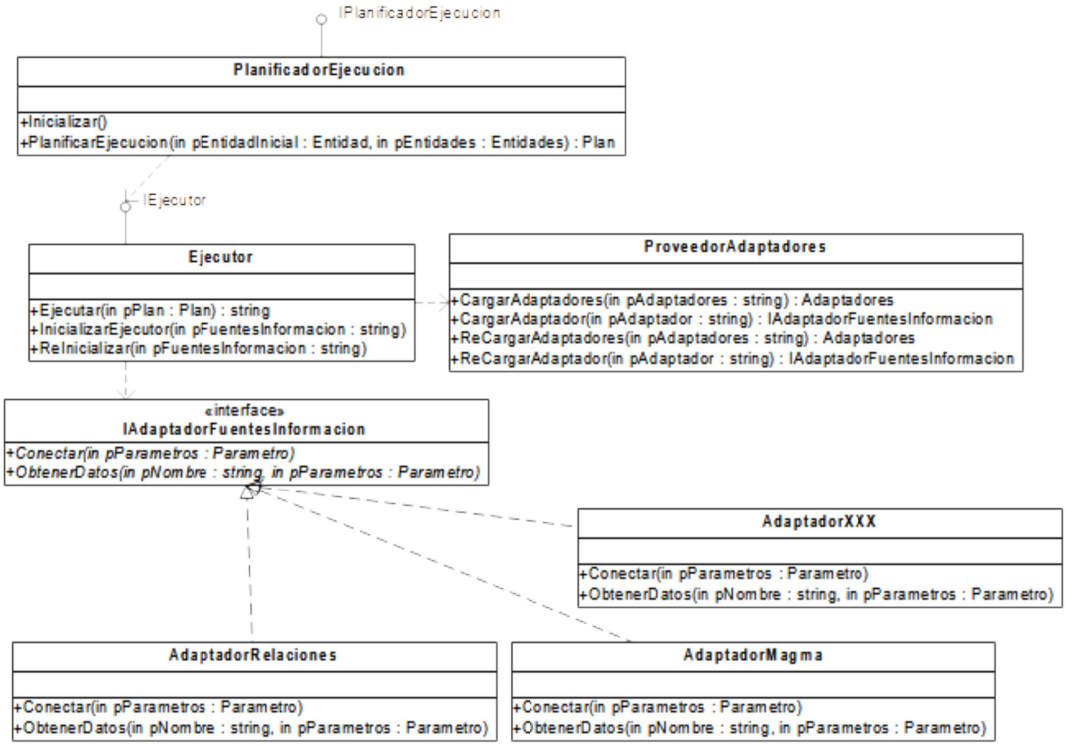
El Ejecutor de Fuentes de Información se encarga de la ejecución de los servicios a través de la invocación de los adaptadores correspondientes.

Para la ejecución de un servicio se utiliza el adaptador específico para el tipo de fuente de información en la que se debe ejecutar el servicio. Al ejecutar un macroservicio, se utilizan los adaptadores correspondientes para cada servicio y luego las relaciones entre estos se ejecutan con el Adaptador de Relaciones.

Agregar una nueva Fuente de Información requiere agregar un nuevo Adaptador, el que permite: acceder a la fuente de información de la misma forma que al resto, y obtener los datos en el formato común a todas las fuentes. Para utilizar esta nueva Fuente de Información solo es necesario agregarla a través de la interfaz del IEditorMetadata con los parámetros adecuados.

El Ejecutor accede a los adaptadores mediante el ProveedorAdaptadores, este proveedor permite acceder a los adaptadores sin importar la forma en que estos sean implementados.

El ProveedorAdaptadores se encarga de obtener instancias o referencias (ya sean referencias web, a objetos remotos o cualquier otro tipo) a los adaptadores de las fuentes de información existentes.



5.1.2.2.1 Adaptadores de Fuentes de Información

Cada adaptador permite abstraer la ejecución de los servicios de su implementación, retornando los datos en un formato común.

5.1.2.2.2 Adaptador de relaciones

Un adaptador particular que se utiliza es el Adaptador de Relaciones. Este se encarga de procesar las entidades cuyos datos no surjan directamente de un servicio, y que puedan implicar operaciones con los datos generados por otros adaptadores.

5.1.2.3 Capa Lógica - Adaptadores para los Reporteadores

Estos módulos permiten utilizar los generadores de reportes a partir de la especificación del reporte provista por el Administrador de Metadata, adaptando los datos al reporteador que corresponda.

Los datos son provistos por el Administrador de Metadatos con un formato común para todas las Fuentes de Información, junto con el formato con el que debe mostrar el reporte.

5.1.3 Capa de persistencia

5.1.3.1 Repositorio de Metadata

En el repositorio de metadatos se tiene la información de los Modelos de Negocio de los clientes. Estos modelos mapearán el esquema lógico del negocio con los servicios de las fuentes de información. Cada modelo estará representado por un Esquema, el que está compuesto por las **fuentes de información, entidades, reportes y formatos de reportes**. Por más detalles de la estructura del esquema ver el documento de Estructura Intermedia.

5.1.3.2 Sistemas Externos

Los sistemas externos son las fuentes de información de los modelos que se representan. Como se dice más arriba a estos sistemas se accede mediante los respectivos adaptadores. Estos sistemas podrán ser distintos tipos: componentes COM+, servicios web, servicios de remoting, etc.

5.1.3.3 Base de Datos

El sistema presentado no maneja directamente bases de datos, los datos son manipulados a través de archivos XML. Los sistemas utilizados como fuentes de información seguramente accedan a bases de datos, pero estos no son accedidos en forma directa por nuestra solución.

6 CONCLUSIONES

Se considera que se plantea una solución que cumple los objetivos establecidos al comienzo del trabajo. A lo largo del mismo se propusieron soluciones a los problemas planteados, en base al **estudio sobre el estado de arte** de las tecnologías relacionadas.

Durante el desarrollo del trabajo se logró:

- Generar un completo documento del **Estado de Arte** en las tecnologías relacionadas.
- Proponer y utilizar una ***representación común de los objetos de alto nivel*** utilizados para crear reportes, estos se representan en el **Meta-Modelo de Negocio**, en base al cual se pueden moldear datos integrados de distintas fuentes de información.
- En base al MMN se tienen **representaciones de reportes independientes del reporteador** con el que éstos se vayan a generar y de los sistemas de información de los que provienen los datos. Se propone una representación de los reportes que se orienta a la representación del reporte y no al procesamiento de esta representación que haga el reporteador.
- se presenta una **arquitectura de integración de datos** para la utilización de distintos sistemas de información.
- Se implementó un prototipo de la arquitectura planteada que utiliza el MMN y la representación de reportes, logrando producir reportes en base a los modelos logrados y la integración de datos que en ellos se indican.
- Con este prototipo se presentan ***dos Editores del Modelo de Negocio***, uno orientado a los programadores y otro a usuarios finales.

Como se mencionó el **prototipo implementado** como prueba de los conceptos se compone de dos aplicaciones independientes que son el *Editor de Metadata* y el *Editor de Modelo de Negocios*. El segundo es para Usuarios Finales, permite la gestión de formatos de reportes, además de la gestión de conceptos de negocios partiendo de otros conceptos anteriormente definidos, y también la definición y ejecución de reportes. El primero está enfocado a usuarios Programadores de Metadata y brinda todas las funcionalidades del Editor de Modelo de Negocios, a las que suma el mantenimiento de fuentes de información y de entidades servidas, las que están directamente ligadas a las fuentes de información.

Estos dos editores generan los reportes con los datos que se obtienen a partir del *Ejecutor de Fuentes de Información*, el cual interactúa con los sistemas de información, e integra los datos de acuerdo con lo especificado en el esquema del meta-modelo con el que se esté trabajando.

Estos dos editores se apoyan sobre un *lenguaje (LEM)* que permite modelar y operar la realidad manejada por el usuario final, integrar diferentes fuentes de información y utilizar diferentes reporteadores para generar los reportes representados. Para el manejo del lenguaje se implementó el Compilador LEM.

El MMN y el prototipo presentados brindan la posibilidad de crear reportes, en base a una representación del modelo de negocio que puede presentar datos que se encuentran en distintos sistemas de información con una estructura más amigable a los usuarios finales.

Pero quedan planteados varios puntos a mejorar de cara al futuro. El área de la generación de reportes tiene una gran complejidad, a lo que se suma el hecho de que, en general, quienes producen estas herramientas tratan de no compartir sus avances para no perder ventajas frente a la competencia. De todas formas, a partir de la propuesta del RDL hecha por Microsoft, se nota una tendencia al desarrollo de herramientas abiertas para la generación de reportes.

El desarrollo de una herramienta gráficamente amigable para ser usada por parte de un usuario final excede en duración el alcance del proyecto de grado. También hay otros puntos a mejorar en el prototipo, tanto en el área de generación de reportes como en la de integración de datos, que se plantearan como trabajos a futuro en la sección siguiente.

6.1 APORTES TECNOLÓGICOS

A lo largo del desarrollo del prototipo se desarrollaron algunas aplicaciones o bibliotecas que de por sí tienen un valor para destacar. Estas son:

- extensión de las fuentes de datos de MS Reporting Services,
- biblioteca para la realización de consultas sobre DataSets,
- generador de proxies para WS.

6.1.1 Extensión de Orígenes de datos de MS RS

El reporteador utilizado, Reporting Services, permite la generación de reportes a partir de una definición del reporte en formato RDL. Dentro de esta especificación se indica un origen de datos (data source) los que luego se incluirán en el reporte.
Estas fuentes están limitadas a unos pocos orígenes de datos, como ser SQL Server, Oracle, etc.
Para lograr utilizar como fuente de datos un archivo que contenga los datos en formato XML, que surge a partir del EFI, se debe agregar un origen de datos específico para esto.

En el desarrollo de nuestro trabajo se provee una biblioteca la cual implementa este origen de datos (llamada RSExtensionOrigenDatosDataSet) y un instalador de la misma, que configura esta fuente de datos en una instalación de MS RS (ModificarConfiguracionRSExtenderDataSet), tanto para el servidor como para el Diseñador de MS RS.

6.1.2 Biblioteca para consultas sobre DataSets

Esta biblioteca consta de una clase que permite realizar consultas del estilo de las consultas SQL sobre objetos de la clase DataSet de .Net.

Por defecto las consultas en los DataSet están limitadas a operaciones sobre una tabla, para la cual se deben definir relaciones con otras tablas, para así operar con ellas. Además estas operaciones devuelven conjuntos de datos.

A través de nuestra biblioteca se pueden realizar consultas sobre todas las tablas del DataSet sin importar las relaciones que entre estas existan. Estas consultas se realizan de la forma en que se realizaría en un DBMS, se unen todas las tablas indicadas en la cláusula FROM y luego se aplican las condiciones sobre estos datos. También se agregan nuevas operaciones que no están dentro de las funcionalidades brindadas por los DataSet. Estas operaciones son operaciones de agrupación de datos, como conteo, sumas, promedios, mínimos o máximos de un conjunto de datos.

6.1.3 Generador de proxies para WS

Para establecer la comunicación con un Web Service de forma dinámica, solo a partir de una URL se desarrollo un **Proveedor de Proxies** para Web Services. Este proveedor se comunica con el WS en la URL indicada y crea una clase para el proxy mediante la herramienta WSDL del framework de .Net, y luego compila esta clase para poder utilizarla como forma de acceso al WS.

7 TRABAJOS FUTUROS

Dada la extensión del problema a resolver quedan una serie de puntos pendientes en nuestro trabajo.

El primer punto pendiente a resolver es el **desarrollo de una interfaz gráfica** que permite la aplicación para un usuario final de los conceptos representados a través del MMN, aunque si se realizó una pequeña incursión al implementarse el Visor de Esquemas. Es a través de esta aplicación que se podría validar de mejor forma los conceptos desarrollados a lo largo del trabajo.

Otro punto que ha quedado fuera del alcance de este trabajo es **el desarrollar adaptadores para otros reporteadores**. Esto ocurrió por razones de tiempo y como fue mencionado anteriormente, porque para tener acceso a las interfaces de los reporteadores comerciales se deben adquirir licencias costosas, por esto no se puede verificar cuales son los reporteadores que brindan interfaces de software que permitan la integración del reporteador a nuestro modelo.

Quedó pendiente un mayor análisis del modelo ROM utilizado por BIRT. Esto se debe a que la mayor parte de la información existente sobre este modelo apareció después del desarrollo de este trabajo.

Otra idea con mayor alcance podría ser la generación de una capa visual, por ejemplo con formularios, la cual permita realizar edición, es decir que se puedan realizar altas, bajas y modificaciones, de los datos almacenados en las fuentes de información.

8 REFERENCIAS

[cer2004] Certia "Microsoft SQL Server 2000 Reporting Services vs. Business Objects Crystal Reports / Crystal Enterprise" Business Intelligence Team of Certia 2004

[RDL2003] "Report Definition Language Specification" Mircosoft 2003

[RS2003] "Libro en pantalla de Reporting Services" Microsoft 2003

(http://msdn.microsoft.com/library/en-us/RSPORTAL/HTM/rs_gts_portal_3vqd.asp y http://www.microsoft.com/sql/reporting/default.asp

[boe2005] Producto: Business Objects Enterprise Empresa: Business Objects

[msa2005] Producto: MicroStrategy ArchitectPara Empresa: MicroStrategy Página http://www.microstrategy.com/

[cgi2005] Producto: Cognos Impromptu Empresa: Cognos

[msrs2005] Producto: SQL Server Reporting Services Empresa: Microsoft

[ecbi2005] Eclipse BIRT Home (http://www.eclipse.org/birt/phoenix/)

[gar2000] Jesús García Molina, M. José Ortín, Begoña Moros, Joaquín Nicolás, Ambrosio Toval, Departamento de Informática y Sistemas Facultad de Informática. Universidad de Murcia "De los Procesos del Negocio a los Casos de Uso" 2000

[Gut2005] J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres, D.Villadiego "XQuery" Universidad de Sevilla, Lenguajes y Sistemas Informáticos 2005

[rui2004] Luis Ruiz Pavón "Consultas sobre documentos XML" www.elguille.info 2004

[Min2004] Mariano Minoli "Presentando XQuery en sociedad" Microsoft 2004

[XBRL2005] XBRL España"Libro blanco xbrl" XBRL España, Miembro de XBRL Internacional http://www.xbrl.org/

[cab2004.] Ismael Caballero "XBRL, el lenguaje estándar para entornos financieros y contables", 2004

[per2003] V. Peralta, "A Framework For Multi-Source Information Systems Development"

[per2002] V. Peralta, Z. Kedad: Una plataforma basada en Metadata para Cálculo de Vistas en Sistemas de Data Warehousing". Instituto de Computación - Universidad de la República Montevideo, Uruguay. 2002

[mot2003] Regina Motz, "Multifuentes", Instituto de Computación de la UDELAR, Montevideo, Uruguay. 2003

[SL90] A. P. Sheth and J. A. Larson, "Federated Database Systems and managing distributed heterogeneous, and autonomous databases" ACM Computing Surveys, 1990.

[sch2005] Ariel Schwindt, "Construcción de Sistemas Multiplataforma basados en Servicios", Microsoft 2005

[par2005] José David Parra, "Hacia una Arquitectura Empresarial basada en Servicios", Microsoft 2005

[dav2000] José David Parra, SOA

[bre2003] Ramon Brena "Autómatas y Lenguajes", Tecnológico de Monterrey 2003

VISUAL REPORTING SERVICES

GLOSARIO

ÁREA DE DATOS DEL REPORTE O ÁREA DEL REPORTE	Es donde están los datos de las entidades en forma detallada, generalmente se disponen en columna con los nombre del atributo en el título de la columna.
ATRIBUTOS	Es la información relevante que se quiere conocer de cada Entidad de Negocio.
BUSINESS INTELLIGENCE (BI)	Es el proceso consistente en recopilar, analizar y utilizar datos del negocio para mejorar el rendimiento empresarial. La tecnología de BI son las herramientas sobre la que se apoya este proceso.
CÁLCULOS	Llamaremos cálculos a las expresiones que contienen llamadas a funciones, utilizan operadores, atributos de las entidades, constantes, variables, o cualquier otra manipulación de datos que se pueda realizar.
CAMPO	Se le llamará campo a los valores atómicos en los resultados de las ejecuciones de servicios o relaciones de expresión.
CONCEPTOS DE NEGOCIO	Alias de Entidades de Negocio.
COREOGRAFÍA DE SERVICIOS	Se refiere al flujo de ejecución de los servicios en el curso de ejecución de una macro operación; es decir que se refiere a los servicios, su ejecución y condiciones en que lo hacen (pre-condiciones y post-condiciones).
CRM	La Gestión de Relación con Clientes (Customer Relationship Management, CRM) es una estrategia que permite a las empresas identificar, atraer, retener y ayudarles a incrementar la satisfacción de los clientes. Esta estrategia, implica disponer del software adecuado que te permita gestionar las relaciones con los clientes.
DATA WAREHOUSE DWH	Los almacenes de datos (Data Warehouse) son bases de datos estructuradas específicamente para consultas y análisis. Los datos que se manejan en el DWH son datos que representan el historial de la organización y son datos informacionales, esto significa que son datos resumidos y periódicos, en contraposición a los datos operacionales.
DATOS DE UNA ENTIDAD	Al hablar de datos de una entidad nos referiremos a las instancias de datos de una entidad que están en las fuentes de información a integrar. Ejemplo los datos correspondiente al atributo teléfono son: 711 71 42, 411 23 23, 458 78 78.
DATOS OPERACIONALES	Información, con poca o ninguna "elaboración", que se va ingresando y guardando en el sistema mientras se cumple cada tarea en la organización.
DESREGISTRAR FUENTES DE INFORMACIÓN	Borrar el registro de una fuente de información de un modelo en la que había sido registrada. Esto implica una acción sobre la configuración del modelo, pero no sobre la fuente de información misma.
DISEÑO DEL REPORTE	Es donde se determina de donde proviene la información, que datos se van a mostrar y el formato en el que se presentaran esos datos.
EDITAR	Se utilizará editar para hacer referencia indistintamente a la creación o modificación de un elemento.
ENCABEZADO Y PIE DEL CORTE DE CONTROL DEL REPORTE O GRUPO	Por cada corte de control (agrupamiento) el reporte puede tener un encabezado que muestra los Nombre de los atributos y en el pie van algunas operaciones agregadas sobre el grupo.
ENTIDADES DE NEGOCIO O ENTIDADES	Es el modelado de los conceptos que el usuario final maneja habitualmente en su negocio. Un ejemplo de estos conceptos podrían ser los clientes, dentro de estos podrían estar los clientes en Argentina, Paraguay y Brasil, además de tenerse clientes deudores y dentro de estos los clientes morosos.

ERP	La Planeación de Recursos Empresariales (Enterprise Resource Planning, ERP) es una forma de utilizar la información a través de la organización de forma más innovadora, en áreas claves, como fabricación, compras, administración, etc. El objetivo es unir la información dispersa en diferentes formatos y diferentes áreas de la organización, y proporcionar una forma común de acceder ver y utilizar estos datos.
ESQUEMA MMN	Lugar físico que permite guardar un conjunto de Entidades de Negocio que modelan un Negocio.
FUENTE DE INFORMACIÓN	En general hablaremos de fuentes de información haciendo referencia a un sistema operacional que provee información, más la forma de acceder a esta, que se definirá en el archivo de configuración.
EJECUCIÓN DE REPORTES O GENERACIÓN DEL REPORTE	Consiste en la creación del reporte o listado, es decir que muestra la información guardada en las fuentes de información en un momento dado, con la presentación indicada en el diseño del reporte.
IMPLEMENTADOR MAGMA	Desarrolladores que trabajan apoyándose en Magma Tools (IDE de la empresa Magma), con el fin de generar nuevos productos o adaptar algunos existentes cono ser Magma ERP.
INSTANCIA DE UNA PLANTILLA	Llamaremos instancia de la plantilla a la copia de una plantilla con nuevos valores, en algunos casos pude agregarse información pero manteniendo el formato y la mayor parte de la información de la plantilla original.
MACROSERVICIO	Es la operación resultante de integrar un conjunto de operaciones atómicas.
METADATA	Se le llama metadata a la información acerca de la información. Por ejemplo un conjunto de propiedades, versión, formato, restricciones sobre la información.
METADATOS DE UNA ENTIDAD	Describen (definen) la estructura de una entidad. Por ejemplo: el metadato para el Atributo teléfono es: nombre=teléfono, tipo=texto, multiplicidad=true, oculto=false , horizontal=false.
MODELO DE NEGOCIO	Es la representación de los conceptos que manejan los usuarios de una organización en un área específica.
OLAP	Procesamiento analítico en línea (OLAP, Online Analytical Processing). Tecnología que utiliza estructuras multidimensionales para proporcionar un acceso rápido a los datos con el fin de analizarlos. Los datos de origen de OLAP se almacenan habitualmente en almacenes de datos o en una base de datos relacional.
ORIGEN DE DATOS	Son los sistemas operacionales de los clientes y es de donde proviene la información para ser mostrada en los repotes, podrán ser DBMS, COM+, Web Services.
ORM	(Object Relational Mapping) Estudia las relaciones entre modelos jerárquicos de objetos y esquemas relacionales de bases de datos, las herramientas de este tipo permiten manejar la persistencia de los objetos.
ORQUESTADOR	Se le llama así a un instrumento de software necesario para cumplir actividades de coordinación entre procesos que pertenecen a sistemas diversos, los cuales interactúan entre sí para cumplir con una tarea específica.
PLANTILLAS	Una plantilla es un modelo a seguir, mantenida en forma persistente que generalmente no cambia; con una estructura y valores por defecto, la estructura permite estandarizar la forma que se mantiene la información además de determinar que información es requerida, los valores son utilizados para inicializar variables requeridas.

PLANTILLAS DE REPORTE RDL	Son documentos con formato XML, más precisamente RDL o bastante similar, que sirve como modelo para la generación de diseños de reportes. Este modelo es copiado y adaptados sus valores para cada diseño de reporte a generar.
PROGRAMADOR DE METADATA	El Programador de Metadata usuario del sistema que conoce las diferentes fuente de información, además sabe como se mapean estas Fuentes de Información con cada uno de las Entidades de Negocio.
PROGRAMADOR MAGMA	Esta es una definición usada en forma interna en Magma y se trata de los desarrolladores que trabajan modificando el código de Magma Tool.
RDL	El lenguaje de definición de reportes (Report Definition Language, RDL) es un esquema XML para la definición de informes, creado por Microsoft; una reporteador de este tipo de documentos es SQL Server 2000 Reporting Services.
REGISTRAR FUENTES DE INFORMACIÓN	Hacer una entrada en el archivo de configuración de Fuentes de Información.
REPORTE, INFORME O INSTANCIA DEL INFORME	Es un modo de presentar la información en formato de sólo lectura, por lo general ofrecen un mayor nivel de procesamiento sobre la información. Los reportes suelen obtener información de dos fuentes: primero diseño del reporte que le indica de donde se sacan los datos y su presentación; segundo los datos en si que provienen de una fuente de información. Por ejemplo, un documento que muestre las ventas por productos, con el total correspondiente a cada producto y presentar el porcentaje sobre el total global.
REPORTEADORES	Sistemas que dados un conjunto de datos y un diseño de reporte permiten generar reportes. Estos sistemas por lo general también proveen formas de editar los diseños de reportes. Ejemplos: Crystal Report o Microsoft Reporting Services.
SERVICIO	Una operación a aplicar que varía según la Fuente de Información. Por ejemplo, un servicio para un DBMS es un Stored Precedure o el texto de una consulta, y para un web service los servicios serian sus métodos. También hablaremos de servicio al referirnos a la especificación de cómo ejecutar un servicio (y con que parámetros) dentro de nuestro modelo.
SOA	La Arquitectura Orientada a Servicios (SOA) posibilita la rápida creación de aplicaciones basadas en comunidades de servicios interoperables; esta enfocada a la reutilización e integración de los sistemas y tecnologías para construir servicios que reduzcan el tiempo de desarrollo y cumplan con los requisitos de negocio.
USUARIO FINAL	Para nuestro sistema son usuarios que pueden tener poco o ningún conocimiento sobre los Orígenes de Datos y desean generar reportes con los datos almacenados en ellos.
WRAPPER O ADAPTADOR	La función de un adaptador de software, o wrapper, es permitir adaptar una interfaz a otra, con el fin de que el objeto que es adaptado, pueda colaborar con otro que requiere una interfaz diferente. Un adaptador también puede cambiar los formatos de salida de aquello que adapta, al formato del sistema al que provee el servicio.

VISUAL REPORTING SERVICES

GLOSARIO

ÁREA DE DATOS DEL REPORTE O ÁREA DEL REPORTE	Es donde están los datos de las entidades en forma detallada, generalmente se disponen en columna con los nombre del atributo en el título de la columna.
ATRIBUTOS	Es la información relevante que se quiere conocer de cada Entidad de Negocio.
BUSINESS INTELLIGENCE (BI)	Es el proceso consistente en recopilar, analizar y utilizar datos del negocio para mejorar el rendimiento empresarial. La tecnología de BI son las herramientas sobre la que se apoya este proceso.
CÁLCULOS	Llamaremos cálculos a las expresiones que contienen llamadas a funciones, utilizan operadores, atributos de las entidades, constantes, variables, o cualquier otra manipulación de datos que se pueda realizar.
CAMPO	Se le llamará campo a los valores atómicos en los resultados de las ejecuciones de servicios o relaciones de expresión.
CONCEPTOS DE NEGOCIO	Alias de Entidades de Negocio.
COREOGRAFÍA DE SERVICIOS	Se refiere al flujo de ejecución de los servicios en el curso de ejecución de una macro operación; es decir que se refiere a los servicios, su ejecución y condiciones en que lo hacen (pre-condiciones y post-condiciones).
CRM	La Gestión de Relación con Clientes (Customer Relationship Management, CRM) es una estrategia que permite a las empresas identificar, atraer, retener y ayudarles a incrementar la satisfacción de los clientes. Esta estrategia, implica disponer del software adecuado que te permita gestionar las relaciones con los clientes.
DATA WAREHOUSE DWH	Los almacenes de datos (Data Warehouse) son bases de datos estructuradas específicamente para consultas y análisis. Los datos que se manejan en el DWH son datos que representan el historial de la organización y son datos informacionales, esto significa que son datos resumidos y periódicos, en contraposición a los datos operacionales.
DATOS DE UNA ENTIDAD	Al hablar de datos de una entidad nos referiremos a las instancias de datos de una entidad que están en las fuentes de información a integrar. Ejemplo los datos correspondiente al atributo teléfono son: 711 71 42, 411 23 23, 458 78 78.
DATOS OPERACIONALES	Información, con poca o ninguna "elaboración", que se va ingresando y guardando en el sistema mientras se cumple cada tarea en la organización.
DESREGISTRAR FUENTES DE INFORMACIÓN	Borrar el registro de una fuente de información de un modelo en la que había sido registrada. Esto implica una acción sobre la configuración del modelo, pero no sobre la fuente de información misma.
DISEÑO DEL REPORTE	Es donde se determina de donde proviene la información, que datos se van a mostrar y el formato en el que se presentaran esos datos.
EDITAR	Se utilizará editar para hacer referencia indistintamente a la creación o modificación de un elemento.
ENCABEZADO Y PIE DEL CORTE DE CONTROL DEL REPORTE O GRUPO	Por cada corte de control (agrupamiento) el reporte puede tener un encabezado que muestra los Nombre de los atributos y en el pie van algunas operaciones agregadas sobre el grupo.
ENTIDADES DE NEGOCIO O ENTIDADES	Es el modelado de los conceptos que el usuario final maneja habitualmente en su negocio. Un ejemplo de estos conceptos podrían ser los clientes, dentro de estos podrían estar los clientes en Argentina, Paraguay y Brasil, además de tenerse clientes deudores y dentro de estos los clientes morosos.

ERP	La Planeación de Recursos Empresariales (Enterprise Resource Planning, ERP) es una forma de utilizar la información a través de la organización de forma más innovadora, en áreas claves, como fabricación, compras, administración, etc. El objetivo es unir la información dispersa en diferentes formatos y diferentes áreas de la organización, y proporcionar una forma común de acceder ver y utilizar estos datos.
ESQUEMA MMN	Lugar físico que permite guardar un conjunto de Entidades de Negocio que modelan un Negocio.
FUENTE DE INFORMACIÓN	En general hablaremos de fuentes de información haciendo referencia a un sistema operacional que provee información, más la forma de acceder a esta, que se definirá en el archivo de configuración.
EJECUCIÓN DE REPORTES O GENERACIÓN DEL REPORTE	Consiste en la creación del reporte o listado, es decir que muestra la información guardada en las fuentes de información en un momento dado, con la presentación indicada en el diseño del reporte.
IMPLEMENTADOR MAGMA	Desarrolladores que trabajan apoyándose en Magma Tools (IDE de la empresa Magma), con el fin de generar nuevos productos o adaptar algunos existentes cono ser Magma ERP.
INSTANCIA DE UNA PLANTILLA	Llamaremos instancia de la plantilla a la copia de una plantilla con nuevos valores, en algunos casos pude agregarse información pero manteniendo el formato y la mayor parte de la información de la plantilla original.
MACROSERVICIO	Es la operación resultante de integrar un conjunto de operaciones atómicas.
METADATA	Se le llama metadata a la información acerca de la información. Por ejemplo un conjunto de propiedades, versión, formato, restricciones sobre la información.
METADATOS DE UNA ENTIDAD	Describen (definen) la estructura de una entidad. Por ejemplo: el metadato para el Atributo teléfono es: nombre=teléfono, tipo=texto, multiplicidad=true, oculto=false , horizontal=false.
MODELO DE NEGOCIO	Es la representación de los conceptos que manejan los usuarios de una organización en un área específica.
OLAP	Procesamiento analítico en línea (OLAP, Online Analytical Processing). Tecnología que utiliza estructuras multidimensionales para proporcionar un acceso rápido a los datos con el fin de analizarlos. Los datos de origen de OLAP se almacenan habitualmente en almacenes de datos o en una base de datos relacional.
ORIGEN DE DATOS	Son los sistemas operacionales de los clientes y es de donde proviene la información para ser mostrada en los repotes, podrán ser DBMS, COM+, Web Services.
ORM	(Object Relational Mapping) Estudia las relaciones entre modelos jerárquicos de objetos y esquemas relacionales de bases de datos, las herramientas de este tipo permiten manejar la persistencia de los objetos.
ORQUESTADOR	Se le llama así a un instrumento de software necesario para cumplir actividades de coordinación entre procesos que pertenecen a sistemas diversos, los cuales interactúan entre sí para cumplir con una tarea específica.
PLANTILLAS	Una plantilla es un modelo a seguir, mantenida en forma persistente que generalmente no cambia; con una estructura y valores por defecto, la estructura permite estandarizar la forma que se mantiene la información además de determinar que información es requerida, los valores son utilizados para inicializar variables requeridas.

PLANTILLAS DE REPORTE RDL	Son documentos con formato XML, más precisamente RDL o bastante similar, que sirve como modelo para la generación de diseños de reportes. Este modelo es copiado y adaptados sus valores para cada diseño de reporte a generar.
PROGRAMADOR DE METADATA	El Programador de Metadata usuario del sistema que conoce las diferentes fuente de información, además sabe como se mapean estas Fuentes de Información con cada uno de las Entidades de Negocio.
PROGRAMADOR MAGMA	Esta es una definición usada en forma interna en Magma y se trata de los desarrolladores que trabajan modificando el código de Magma Tool.
RDL	El lenguaje de definición de reportes (Report Definition Language, RDL) es un esquema XML para la definición de informes, creado por Microsoft; una reporteador de este tipo de documentos es SQL Server 2000 Reporting Services.
REGISTRAR FUENTES DE INFORMACIÓN	Hacer una entrada en el archivo de configuración de Fuentes de Información.
REPORTE, INFORME O INSTANCIA DEL INFORME	Es un modo de presentar la información en formato de sólo lectura, por lo general ofrecen un mayor nivel de procesamiento sobre la información. Los reportes suelen obtener información de dos fuentes: primero diseño del reporte que le indica de donde se sacan los datos y su presentación; segundo los datos en si que provienen de una fuente de información. Por ejemplo, un documento que muestre las ventas por productos, con el total correspondiente a cada producto y presentar el porcentaje sobre el total global.
REPORTEADORES	Sistemas que dados un conjunto de datos y un diseño de reporte permiten generar reportes. Estos sistemas por lo general también proveen formas de editar los diseños de reportes. Ejemplos: Crystal Report o Microsoft Reporting Services.
SERVICIO	Una operación a aplicar que varía según la Fuente de Información. Por ejemplo, un servicio para un DBMS es un Stored Precedure o el texto de una consulta, y para un web service los servicios serian sus métodos. También hablaremos de servicio al referirnos a la especificación de cómo ejecutar un servicio (y con que parámetros) dentro de nuestro modelo.
SOA	La Arquitectura Orientada a Servicios (SOA) posibilita la rápida creación de aplicaciones basadas en comunidades de servicios interoperables; esta enfocada a la reutilización e integración de los sistemas y tecnologías para construir servicios que reduzcan el tiempo de desarrollo y cumplan con los requisitos de negocio.
USUARIO FINAL	Para nuestro sistema son usuarios que pueden tener poco o ningún conocimiento sobre los Orígenes de Datos y desean generar reportes con los datos almacenados en ellos.
WRAPPER O ADAPTADOR	La función de un adaptador de software, o wrapper, es permitir adaptar una interfaz a otra, con el fin de que el objeto que es adaptado, pueda colaborar con otro que requiere una interfaz diferente. Un adaptador también puede cambiar los formatos de salida de aquello que adapta, al formato del sistema al que provee el servicio.

VISUAL REPORTING SERVICES

ANEXO I

ESTADO DEL ARTE

ÍNDICE

1 INTRODUCCIÓN

Este trabajo se sitúa en el área de la **generación de reportes** por usuarios finales. Para ello se debe proveer un **modelo de datos** que permita vincular conceptos manejados comúnmente por los usuarios con los sistemas operacionales de la organización. Esta vinculación se debe realizar a través de una **metadata**, la cual debe de tener la capacidad de **integrar** diferentes fuentes de información; como lo establecen las estrategias como **business intelligence**. La interacción del usuario con el sistema y la persistencia del modelo de datos se debe realizar a través de diferentes **lenguajes**, uno que sea amigable y el otro que permita gestionar toda la información ingresada.

Dado los objetivos del proyecto, la generación de los reportes deberá realizarse a partir de un modelo de más alto nivel que el modelo relacional utilizado en las bases de datos operacionales, es aquí que entran en juego las técnicas de diseño de data warehouse, que analizaremos.

Así mismo, en el mundo de hoy en día, es necesario integrar datos provenientes de distintas fuentes lo que nos lleva a la integración de sistemas también tratada aquí. Para lograr esta integración de sistemas se plantea la necesidad de tener una representación de reportes que sea estándar e independiente del generador de reportes que se utilice.

Otro punto importante en este desarrollo es el lenguaje que le permitirá a los diferentes tipos de usuarios interactuar con el sistema, y el lenguaje que se usará para persistir la información ingresada.

Los puntos a analizar en este documento son: en la sección 2 generación de reportes y análisis de diferentes reporteadores, en la sección 3 representación de modelos de datos y metadata, en la sección 4 procesamiento de datos, en la sección 5 integración de fuentes de información, en la sección 6 sistemas multifuentes y en la sección 7 lenguajes de programación y compiladores.

2 GENERACIÓN DE REPORTES

La generación de reportes consiste en combinar una instancia de los datos operacionales los cuales son procesados y formateados según una definición del reporte. Esta función constituye una parte fundamental de una estrategia general en business intelligence (BI).

La problemática a resolver nos lleva como primer punto a investigar los conceptos manejados, los estándares propuestos y las herramientas existentes en el mercado, para la generación de reportes.

En cuanto a los conceptos y sus definiciones están plasmados en el documento Glosario.doc.

A pesar de la existencia de muchas herramientas dedicadas a la generación de reportes estas herramientas son muy propietarias, por lo cual no se han encontrado presentación de estándares de generación de reportes. Si existen algunas propuestas como **RDL** (*Report Definition Language*) realizada por MS para la definición de lenguaje de definición de reportes, y de **ROM** (*Report Object Model*) el modelo utilizado por la herramienta BIRT para Eclipse, estas dos propuestas se analizan en los capítulos siguientes de este documento.

Del análisis de las herramientas, que permiten la generación de reporte, se distingue la participación de tres procesos fundamentales que son la base para la solución propuesta, estos procesos son:

- **Administración de las Fuentes de Información**: en este punto se analiza como integrar las múltiples fuentes de información, de la organización, a la generación de todos los posibles reportes.
- **Diseño del reporte**: en esta etapa se determina cuales son los datos que se van a mostrar y el formato que se presentaran esos datos en la generación de un reporte en particular.
- **Generación del reporte**: esta etapa consiste en combinar las dos etapas previas, es decir que se generan listados con la información guardada en las fuentes de información, hasta el momento de la generación del reporte, con la presentación indicada en el diseño del reporte.

Pasaremos a analizar las dos últimos temas, puesto que la administración de los datos esta analizada en un capitulo aparte llamado Sistemas Multifuentes.

2.1 DISEÑO DE REPORTES

En diseño de reportes contienen un conjunto de instrucciones que describen una forma de obtener los datos procedentes de las fuentes de información, el lugar donde serán colocados esos datos, el formato que tendrá tanto el texto como el área donde se mostrarán, es decir que el diseño del reporte mantiene una descripción de un reporte.

En esta área se analizó un proyectos de estandarización realizado por MS que es el lenguaje RDL [RDL2003] el cual tiene como objetivo promover la interoperabilidad entre los reporteadores comerciales mediante la definición de una representación común que permita el intercambio de definiciones de reportes, es decir que cualquier reporteador pueda generar un listado teniendo la definición del reporte y los datos.

Para MS una definición de reporte es conjunto de datos en formato XML que siguen una gramática definida en el lenguaje RDL (Report Definition Language), en este lenguaje se describen todas las variantes posibles que puede tener una definición de reporte.

RDL es una definición del esquema, no un lenguaje de programación o protocolo como HTTP u ODBC. RDL no especifica cómo deben generarse o cómo se deben procesarse los reportes, además es importante destacar que está totalmente encapsulado, esto quiere decir que la interpretación de un documento RDL es independiente del generador del documento y de la generación de reporte.

Este es un importante aporte para la implementación de nuestro proyecto, debido a la independencia que da para la generación del diseño de reporte, puesto que partiendo de cero se puede generar el diseño de un reporte y por medio del un servicios ofrecido por RS se puede generar el reporte. Los otros reporteadores exigen que alguna etapa del diseño del reporte sea realizada desde el propio entorno del reporteador, lo cual no lo hace viable de ser integrado a un sistema independiente.

Otros punto importante que aporta la definición de RDL es una definición y enumeración de cada concepto manejado en la generación de reportes.

Algunas otras organizaciones han establecido sus propios lenguajes de definición de reportes, como la herramienta BIRT (Business Intelligence and Reporting Tool) de Eclipse para Java, menciona que se utiliza una representación a la que llama **XML Report Design Schema**, pero al momento de realizar este trabajo no se había dado a conocer la estructura de estos esquemas. Actualmente se encuentra disponible una especificación del modelo utilizado (Report Object Model, ROM), y se indica que el Report Design Schema corresponde a un trabajo previo realizado por Actuate.

BIRT se compone de un diseñador de reportes para eclipse (Eclipse Report Designer) junto con un motor de diseño de reportes (Elipse Report Engine), los que generan como salida un XML Report Design. Esta salida la toma un motor de reportes (Report Engine) el cual genera la salida deseada.

2.2 REPORTEADORES

Son muchas las herramientas que permiten generar un diseño del reporte y combinarlos con datos provenientes de las fuentes de información para generar una instancia de reporte (generalmente llamado reporte únicamente) la cual se pueda imprimir o guardar como un archivo; algunas de estas herramientas son:

- Business Objects Enterprise (de Business Objects) [boe2005].
- MicroStrategy Architect (de MicroStrategy) [msa2005].
- Cognos Impromptu (Cognos) [cgi2005].
- SQL Server Reporting Services (Microsoft) [msrs2005].
- Eclipse Business Intelligence and Reporting Tools (Actuate Corporation y Eclipse Fundation) [ecbi2005].

2.2.1 Crystal Reports (de Business Objects)

La empresa Business Objects tiene en el mercado el producto con el nombre Business Objects Enterprise y el sub producto Crystal Reports. Este producto es una plataforma de business intelligence (BI), que ofrece un repertorio completo de funcionalidades de BI para: generación de reportes (para esto se basa en la conocida herramienta Crystal Reports), consulta y análisis, gestión del rendimiento empresarial, e integración de datos. También ofrece muchos de estos servicios a través de la tecnología de Web Services. Por más información [boe2005].

2.2.2 MicroStrategy Architect (de MicroStrategy)

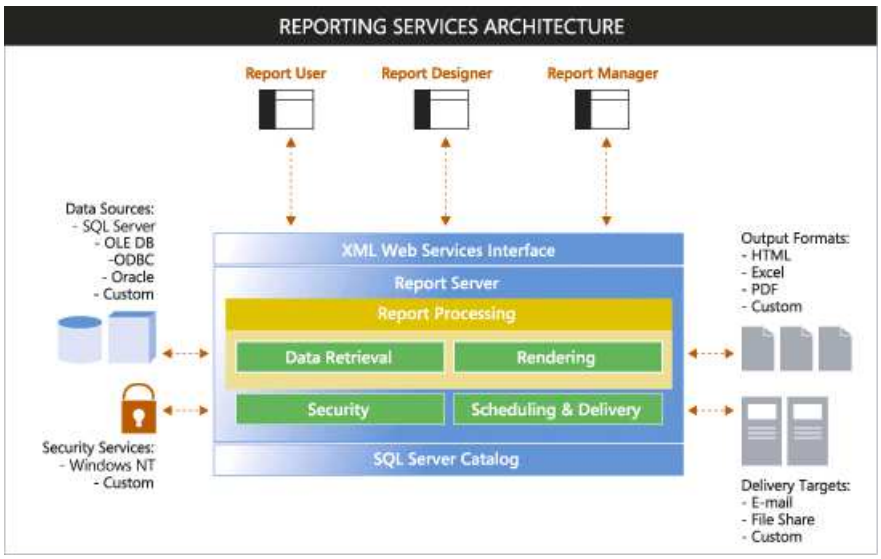
MicroStrategy Architect es un herramienta que permite crear aplicaciones de Business Intelligence contra data warehouses o sobre los datos operacionales. Para ello mantiene capas de abstracción que separan los requerimientos del negocio de la definición física de las tablas en la base de datos. Además de incluir vistas integradas de los datos obtenidos de repositorios de datos heterogéneos. Al definir las dimensiones de las diferentes fuentes dentro del modelo de objeto puede, automáticamente, combinar datos provenientes de diferentes fuentes en el mismo documento de reporte. Los datos pueden provenir de cualquier fuente a la que pueda acceder MicroStrategy 8, incluyendo data warehouse, data marts, SAP BW, y cualquier base de datos de sistema operacional. Por más información [msa2005].

2.2.3 Cognos Impromptu (Cognos)

Cognos es un producto que abarca todas las áreas de Business Intelligence, como ser: generación de reportes, análisis, análisis de evolución, integración de múltiples fuentes de información basado en una arquitectura simple con Web Services. Por más información [cgi2005].

2.2.4 Reporting Services (Microsoft).

Es una herramienta que permite crear, administrar y entregar tanto informes tradicionales en papel como informes Web interactivos. Reporting Services se integra en el marco de trabajo de Microsoft Business Intelligence y combina las funciones de administración de datos de SQL Server y Microsoft Windows Server™. SQL Server Reporting Services admite una amplia gama de orígenes de datos comunes, como OLE DB y Conectividad abierta de bases de datos (ODBC), así como varios formatos de salida tanto para exploradores Web conocidos como para aplicaciones de Microsoft Office System. Gracias a Microsoft Visual Studio® .NET y Microsoft .NET Framework, los programadores pueden aprovechar las funciones de los sistemas de informes existentes y conectarse a orígenes de datos personalizados, crear otros formatos de salida e incluso realizar entregas en diferentes dispositivos. Los programadores pueden generar los informes mediante herramientas de diseño de Microsoft o generarlo utilizando el lenguaje RDL (Report Definition Language), que fue propuesto por Microsoft como un estándar de la industria. Este lenguaje esta en formato XML, y se utiliza para definir los datos y los formatos de los informes.



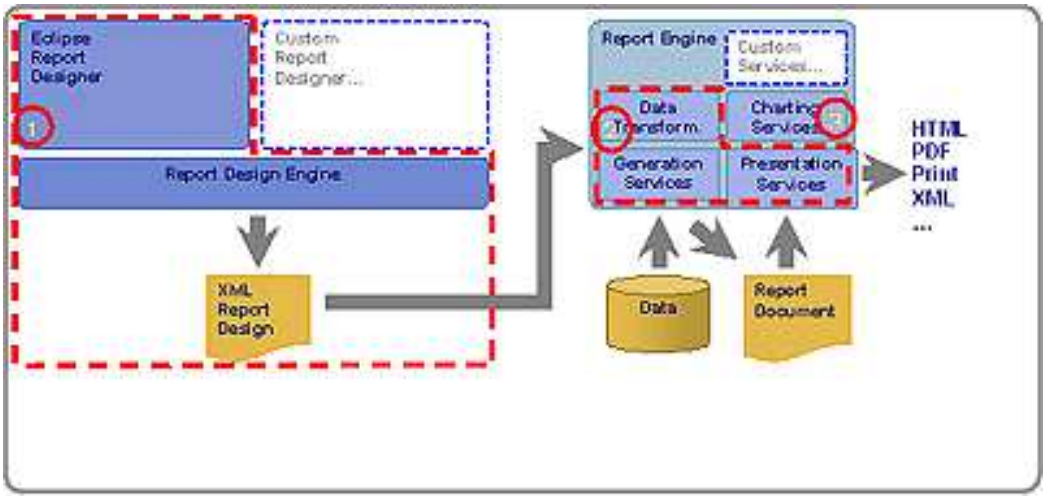
Por más información [msrs2005].

2.2.5 Business Intelligence and Reporting Tools (BIRT)

BIRT es una herramienta para la generación de reportes en Java, desarrollada por Actuate Corporation y Eclipse Fundation, se anuncia como el primer proyecto de Business Intelligence y Reporting Technology que es Open Source. Las primeras versiones de la Herramienta se liberaron en marzo del 2005. Los elementos disponibles son:

- **Report Object Model** (ROM): un esquema que describe los elementos que conforman el diseño de un reporte de BIRT
- **Eclipse Report Designer**: una herramienta que se integra con Eclipse que genera reportes basada en un diseño según el ROM.
- **Eclipse Report Engine**: que permite que los reportes sean generados usando diseños de ROM creados en el Designer.
- **Eclipse Charting Engine**: que brinda la posibilidad de generar una amplia variedad de gráficos dentro de BIRT.

Al comienzo de este trabajo el material encontrado sobre este proyecto fue bastante limitado, en particular no se encontraron especificaciones del Report Object Model. A pesar de esto, actualmente se encuentran disponibles especificaciones con fecha de mayo del 2005. Es por esta razón que no se analizó este modelo para ser incluido dentro de nuestro trabajo, pero sin duda ROM es un punto de interés dentro del tema tratado.



Más información sobre BIRT y ROM puede ser encontrada en [ecbi2005].

2.3 CONCLUSIÓN

Son muchos los sistemas que permiten el diseño y la generación de reportes (Reporteadores), en muchos de ellos una vez diseñado el reporte, permiten generar reportes tanto en el propio sistema como en sistemas generados por terceros. El inconveniente en la mayor parte de esto sistemas es que el diseño del reporte solo se puede lograr a partir del propio reporteador y no a partir de sistemas independientes, por lo tanto el usuario final de los sistemas independientes solo podrá generar los reportes prediseñados.

Esta es la principal ventaja que tiene MS Reporting Services en combinación RDL, puesto que partiendo de cero se puede generar un archivo de diseño de reporte con una instancia de RDL adecuado y usando MS Reporting Services generar un reporte, esto quiere decir que usando este sistema se podrá diseñar reportes en tiempo de ejecución en sistemas independientes al reporteador.

Otra característica importante que tiene MS Reporting Services es la de ser un sistema gratuito, puesto que muchos de los otros reporteadores pueden llegar a costar decenas de miles de dólares y hasta cientos de miles de dólares.

Para los objetivos planteados resultan de particular interés MS Reporting Services y BIRT, por sus representaciones de reportes independientes de los reporteadores. BIRT no será utilizado por no haber estado la información disponible a tiempo, pero queda planteado como trabajo a futuro integrar este reporteador. Se presentará una representación genérica de los reportes basada en el RDL de Microsoft y se integrará MS RS como reporteador en nuestro prototipo.

3 MODELO DE DATOS

La información, generalmente, es registrada procesada y luego comunicada a otras personas en una forma simbólica que pueda ser entendida. Para estos fines resulta de importancia fundamental contar con una adecuada representación simbólica de la información.

La pieza más elemental de información es el "dato" (por ejemplo una cifra de sueldo). Un dato elemental no es de utilidad por sí mismo, es útil sólo cuando está vinculado con otros datos (por ejemplo la identificación de un empleado), por lo que el almacenamiento y la comunicación de información requieren la construcción de "estructuras de datos".

El modelo de datos debe ser una fiel representación del sistema de información objeto de estudio y debe aparecer representada toda parte lógica de la información necesaria, donde el objetivo fundamental es la obtención de estructuras no redundantes, sin inconsistencias, seguras e íntegras; por lo cual el lenguaje natural no es el mas adecuado y menos en sistemas computacionales.

Se han explorado y desarrollado diferentes "modelos de datos" como ser: Relacionales, Jerárquicos, de Red, Semánticos, o inclusive UML define diferentes formas de modelar la realidad para las diferentes etapas del ciclo de vida del software.

Los modelos de datos poseen diferentes niveles de abstracción a la hora de representar datos. Los de mayor nivel de abstracción y potencia semántica se orientan al diseño de las estructuras de datos. Este tipo de modelo de datos recibe la denominación de Modelo Semántico o Modelo de Negocio.

Estos dos últimos puntos son los que analizaremos en las próximas secciones.

3.1 MODELO DE NEGOCIO

La generación de reportes o consultas es una tarea común, lo que se pretende es dejar esa tarea a la persona que utiliza la información, no a quien la genera, si se logra salvar esta diferencia, el usuario final podrá consultar y crear sus reportes de modo personalizado y sin intermediarios.

Aprender a crear reportes por un usuario inexperto se puede lograr en tiempos mínimos si este usuario se maneja en el contexto del negocio y en todas las partes que lo componen. El modelo del negocio o también llamado capa semántica son reglas de acceso a los datos para reflejar la estructura de la organización. Esta capa mantiene al usuario al margen de los conocimientos técnicos. El usuario puede olvidarse de nombres, lenguajes de consulta SQL, uniones de tablas, etc., e inclusive este modelo puede integrar diferentes fuentes de información; es decir que puede ver la información justamente como la percibe en su negocio, no de acuerdo a la estructura de la base de datos operativa.

Para realizar esta tarea de independizar el Modelo de Negocio del Modelo Operativo (Relacional), debe de existir en el Modelo de Negocio información sobre los datos que permita mapearlos al Modelo Operativo. Esta información serán los Metadatos del Modelos de Negocio.

Los metadatos deberán ser definidos por alguien que conozca el modelo operativo y que logre presentarlo al usuario inexperto, en una forma que sea entendible y acorde con sus conceptos.

3.1.1 Representación del Modelo de Negocio

UML es un enfoque de notaciones, que se ha generalizado su utilización en todo el ciclo de vida del software, fundamentalmente para la descripción de soluciones orientadas a objetos. Organizaciones como OMG han propuesto y adoptado UML como el estándar de la notación orientada a objetos, que es el paradigma de desarrollo más usado en la actualidad.

Aunque no existe una definición directa de modelo de negocio en UML; si existen varias vinculaciones entre los diferentes modelos y la realidad a representar, por ejemplo los casos de uso, que son la base de este modelo, o el modelo conceptual o el diagrama de actividades, se

pueden vincular casi directamente con los conceptos y las actividades que habitualmente se manejan en el negocio, ver [gar2000].

Ya que UML, y más particularmente los casos de uso el modelo conceptual o el modelo clases, nos permiten representar una realidad y algunas de las representaciones relacionados con estos modelos resultan ser un buen punto de contacto entre expertos en sistemas y usuarios finales, entonces es esperable que los conceptos y las representaciones incluidos en estos modelos los podamos extender para nuestra definición de representación de modelos de negocio. Es así que surgen términos como Entidades para definir los conceptos del negocio, o Atributos o Relaciones los cuales provienen de estos modelos.

Otra de las ventajas de incluir los conceptos usados en el modelo de conceptual es que esto facilitaría el aprendizaje, tanto de las definiciones como de las representaciones, a aquellos usuarios que ya están empapados en el manejo de UML, que serían todos aquellos usuarios dedicados al desarrollo de las fuentes de información.

3.2 CONCLUSIONES

El modelo planteado debe ser claro y orientado a nuestros objetivos. Existen modelos que presentan conceptos útiles para nuestros objetivos (como el Modelo relacional o el Orientado a Objetos), pero que no cubren todas las nuestras necesidades, por lo que serán utilizados como base para el desarrollo de nuestro modelo.

4 METADATOS

Los metadatos son "datos sobre datos" y describen el contenido, calidad, condiciones, origen, información para obtenerlo y otras características de datos. Permiten y ayudan al entendimiento de la información almacenada. Un registro de metadatos consiste en un conjunto de atributos, o elementos, necesario para describir la información en cuestión. Como ejemplo de metadatos se podrían nombrar: el catálogo de una biblioteca, la lista de canciones de un CD de música, resumen de un documento, la información necesaria para saber como obtener los datos de los clientes en el modelo operativo, etc.

Para mantener la información contenida en los registros de metadatos se pueden usar diferentes opciones como ser: tener en lugares separados los metadatos de la información relacionada, como los que manejan algunos adaptadores con el fin de permitir interoperación de sistemas heterogéneos; o embebida dentro de la misma información como en las páginas web.

Las razones para usar metadatos son de lo más variadas, a continuación se describen algunas:

- Aportar una descripción sobre la información,
- Permite clasificarla posibilitando realizar una búsqueda centrada en un área requerida y no sobre todo el conjunto de datos, lo cual permite acelerar el proceso de de búsqueda.
- Otro aporte interesante de los metadatos sobre las búsquedas de información, es que la búsqueda se convierte en un proceso de comparación entre los términos de la consulta y la información guardada por el sistema. Por lo tanto, utilizando los metadatos combinados con el lenguaje de búsqueda se aumenta la posibilidad de encontrar la información requerida.
- Permitirá ínteroperación de sistemas los cuales pueden tener fuentes de datos comunes o heterogéneos, indicando por ejemplo, una descripción del sistema que los mantiene o una descripción de los datos de cada fuente o un mapeo entre los datos de las diferentes fuentes o describir funcionalidades de cada sistema. Estas descripciones de los datos, sus fuentes y sus operaciones se podrá tener en un mismo formato o no y si no están en un mismo formato también se podrían utilizar metadatos sobre los metadatos. De esta forma se amplía, no solo el conjunto de datos donde se realizan las operaciones, sino también las operaciones a realizar, debido a que teniendo datos integrados puedo realizar operaciones que vinculen las diferentes fuentes de datos.
- Algunas otras razones podrían ser: disminuir el tráfico de la red, pudiéndose intercambiar descripciones en lugar de información; permitir realizar controles ante de producir errores, determinar restricciones, entre otras.

Varias tecnologías aplicadas a la Web han expandido recientemente las posibilidades y capacidades de los metadatos. Estas tecnologías se han adaptado para trabajar con sistemas interoperables, aumentando su riqueza en la descripción y facilitando el acceso y la posibilidad de acceso a la información. Estas herramientas suministran una mayor semántica y estructuración a los documentos de metadatos, permitiendo más opciones de trabajo con los objetos (datos) y los metadatos. Entre los metalenguajes que permiten implementar medatatos tenemos SGML (Standard Generalized Markup Language) y XML (eXtensible Markup Language)

SGML es de los primeros metalenguajes de metadatos que permite la creación de diferentes lenguajes de etiquetado a partir de una DTD (Document Type Definition).

XML es una versión resumida de SGML, pero tal vez más potente. Uno de sus principales objetivos es permitir el intercambio de información a través de la Web.

Basándonos en XML existen especificaciones de descripción de recursos como RDF (Resource Description Framework) que permite describir los datos de cada documento y definir las relaciones que hay entre los datos y XML o CDF (Channel Definition Format) desarrollado por Microsoft.

Además de los metalenguajes y esquemas descriptivos, tenemos diferentes estándares de especificaciones para la generación de metadato como ser:

- MOF (Meta-Object Facility),
- Dublin Core, o
- IEEE LTSC LOM (Learning Object Metadata).

Éstos modelan un conjunto de elementos para describir una amplia gama de recursos de Internet o sistemas con datos provenientes de fuentes diversas.

4.1 XML (EXTENSIBLE MARKUP LANGUAGE)

XML es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados. Permite declaraciones de contenido más precisas y resultados de búsquedas más exactos y fáciles de distribuir. Basándose en XML surgieron aplicaciones para ver y manipular datos basadas en el Web, intranet o extranet, es por esta razón que se torna un lenguaje muy potente para la integración de sistemas, otra de las características que aporta el uso de este lenguaje es la amplia variedad de herramientas diseñadas para el manejo de la información en este formato.

XML garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes. La interoperabilidad resultante de esta tecnología, está creando rápidamente una serie de aplicaciones relacionadas con el comercio electrónico.

XML proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos.

Se apoya en diferentes especificaciones, que permiten mejorar la interfase de XML con el usuario, entre otras tenemos:

- **DTD** (*Document Type Definition*) esta especificación que encierra una definición formal de un tipo de documento y especifica su estructura lógica. Si el documento tiene un DTD relacionado se le llama "bien formado" (well-formed) y sino se le llama "validado" (valid).
- **XSL** (*eXtensible Stylesheet Language*): Define o implementa el lenguaje de estilo de los documentos escritos para XML.
- **XLL** (*eXtensible Linking Language*): Define el modo de establecer conexiones entre diferentes enlaces. Se considera que es un subconjunto de HyTime.
- **XUA** (*XML User Agent*): Estandarización de navegadores XML. Se aplica a los navegadores para que compartan todas las especificaciones XML.

Una serie de aplicaciones impulsan el desarrollo de XML en diferentes áreas, algunas de ellas son:

- Aplicaciones que utilizan la Web para conectar dos o más bases de datos permitiendo la interoperación de sistemas distribuidos, pudiéndose modificar el contenido y la estructura de esta.
- Aplicaciones que requieran presentar diferentes versiones de los mismos datos a diferentes usuarios.
- Aplicaciones donde diferentes agentes intentan adaptar la búsqueda de información a las necesidades de los usuarios individualmente, basándose en XML se permitirá una mayor precisión de la búsqueda requerida, debido a los metadata que aporta.
- Aplicaciones que deben integrar datos procedentes de fuentes heterogéneas, gracias a la capacidad de buscar en varias bases de datos no compatibles entre sí es, utilizando la capacidad de ampliación y la flexibilidad de XML le permiten describir los datos contenidos en una gran variedad de aplicaciones, además dado que los datos en XML son autodescriptivos, se pueden intercambiar y procesar información sin necesidad de una

descripción incorporada de los datos entrantes, simplificando la tarea de integración de datos.

4.2 RDF (RESOURCE DESCRIPTION FRAMEWORK)

Es muy difícil automatizar cualquier tarea en la Web, debido al volumen de información que contiene no es posible gestionarla manualmente. La solución que se propone aquí es el uso de metadatos para describir los datos contenidos en la Web; si esta metodología aporta soluciones a algunos problemas de la Web, entonces puede ser muy factible de ser usada en sistemas de menor dimensión pero con las mismas características de integración de sistemas heterogéneo que tiene la Web.

RDF (Resource Description Framework) es una recomendación del W3C, que proporciona la tecnología para administrar metadatos los cuales describen recursos, proporciona interoperatividad entre aplicaciones que intercambian información, y proporciona las facilidades para el procesamiento automático de esos recursos.

RDF tiene un modelo muy simple consistente en arcos y nodos etiquetados lo cual lo hace más flexible que XML para expresar metadatos, pero cualquier especificación de declaraciones RDF puede serializarse en XML

RDF tiene principalmente una interpretación semántica y se utiliza para modelar conocimiento, donde las representaciones basadas en las estructuras de árbol no son suficientes.

El modelo RDF puede considerarse como un recurso representado por registros con tres campos (recurso, propiedad y valor), en particular los recursos o nodos pueden ser todas las cosas descritas por expresiones RDF, por ejemplo una página Web u objetos no directamente accesibles vía Web, o la representación de un libro impreso, o el modelado de un cliente. Los recursos se designan siempre por URI's y no se encuentran dentro de la información a modelar por lo cual se pensó para ser un modelo de metadatos.

El lenguaje particular utilizado es XML que es un lenguaje para modelar datos, pero pueden usarse otros lenguajes.

4.3 CONSULTAS SOBRE XML

XML es la tecnología de uso casi obligatorio en el intercambio e integración de información en los entornos de desarrollo de sistemas. Pero a medida que el volumen de datos almacenados en XML aumenta, se comprueba que, las herramientas más habituales para manipular datos en XML, los parsers SAX y DOM, no son prácticas para manejar grandes y complejas colecciones datos. La principal contra para procesar datos en XML mediante DOM y SAX es la necesidad de escribir una gran cantidad de código para realizar el procesamiento.

También existe una familia de herramientas conocidas como bindings, las cuales son útiles y ahorran una gran cantidad de trabajo solo en algunos casos.

Otra solución existente como XSLT permite transformar datos de XML a otros formatos como HTML o PDF, pero solo son útiles para hacer presentaciones diferentes de los datos no para aplicar operaciones o buscar información. Las herramientas de parsers, binding y XSLT no son prácticas cuando queremos consultar grandes volúmenes de información proveniente de los servidores de bases de datos que pueden retornar información en XML nativas (Native XML Database)

Usando SAX este tipo de tareas es rápido y sencillo si la consulta y la estructura de la información no tienen una gran complejidad. Alguno de los inconvenientes es que el código está fuertemente ligado a la estructura de los datos en XML, por lo que cualquier cambio en la consulta o en la estructura de los datos obliga a volver a escribir el código. Además el código es poco parametrizable y por lo cual es poco reutilizable por lo cual consultas similar habría que volverla a escribir desde el principio.

Las herramientas de bindings simplifican el cargar en memoria los datos, pero, aplicadas a base de datos voluminosas no es la técnica más adecuada, además no es la tarea de escribir una

consulta lo complicado sino que muchos de ellos requieren los documentos de definición de XML (DTDs o XML-Schemas), por lo que se simplifica en escribir las consultas se complica para escribir el DTD o XML-Schemas correspondiente.

Como el manejo e intercambio de grandes volúmenes de información se esta haciendo cotidiano, y uno de los estándares más usado hoy en día para el intercambio de información es XML, es que se requiere definir un lenguaje declarativo (no dependiente del recorrido ni del origen de datos), de consulta, simple, que permita la recorrida fácil sobre XML, que permita devolver un conjunto de nodos que cumplen ciertas condiciones. Este lenguaje podría ser XQuery.

4.3.1 XQuery

Se puede decir que XQuery es a XML como SQL es a las bases de datos relacionales [Gut2005] [Rui2004] [Min2004]. XQuery es un lenguaje de consulta diseñado para escribir consultas sobre colecciones de datos expresadas en XML. Abarca desde archivos XML hasta bases de datos relacionales con funciones de conversión de registros a XML. Su principal función es extraer información de un conjunto de datos organizados como un árbol n-ário de etiquetas XML. En este sentido XQuery es independiente del origen de los datos.

XQuery es un lenguaje funcional, lo que significa que, en vez de ejecutar una lista de comandos como un lenguaje procedural clásico, cada consulta es una expresión que es evaluada y devuelve un resultado, al igual que en SQL. Diversas expresiones pueden combinarse de una manera muy flexible con otras expresiones para crear nuevas expresiones más complejas y de mayor potencia semántica.

XQuery está llamado a ser el estándar de consultas sobre documentos XML.

El grupo W3C Actualmente está trabajando en XQuery y existen o están en proceso numerosas implementaciones de motores y herramientas que lo soportan.

El grupo de trabajo en XQuery del W3C definió y logró alcanzar un conjunto de metas para este lenguaje, alguna de ellas son:

- XQuery es un lenguaje declarativo. Al igual que SQL hay que indicar que se quiere, no la manera de obtenerlo.
- XQuery es independiente del protocolo de acceso a la colección de datos.
- Una consulta en XQuery funciona en forma independiente de las fuentes de información.
- Las consultas y los resultados son respetar el modelo de datos XML
- Las consultas y los resultados ofrecen soporte para los namespace.
- Es capaz de soportar XML-Schemas y DTDs y también es capaz de trabajar sin ninguno de ellos.
- XQuery puede trabajar con independencia de la estructura del documento, sin necesidad de conocerla.
- XQuery soporta tipos simples, como enteros y cadenas, y tipos complejos, como un nodo compuesto por varios nodos hijos.
- Las consultan soporta cuantificadores universales (para todo) y existenciales (existe).
- Las consultas soportan operaciones sobre jerarquías de nodos y secuencias de nodos.
- Es posible en una consulta combinar información de múltiples fuentes.
- Las consultas deben ser capaces de manipular los datos independientemente del origen de estos.
- Es posible definir consultas que transformen las estructuras de información originales y es posible crear nuevas estructuras de datos.

- El lenguaje de consulta es independiente de la sintaxis, esto es, es posible que existan varias sintaxis distintas para expresar una misma consulta en XQuery.

XQuery ha sido construido sobre la base de Xpath. Este es un lenguaje declarativo para la localización de nodos y fragmentos de información en árboles XML. XQuery se basa en este lenguaje para realizar la selección de información y la iteración a través del conjunto de datos. La nueva versión de Xpath (2.0) está siendo desarrollada por el mismo grupo de trabajo de la W3C y de manera conjunta a la versión 1.0 de Xquery, la siguiente versión esta propuesta para salir a fines del 2005.

Una consulta en XQuery es una expresión que lee una secuencia de datos en XML y devuelve como resultado otra secuencia de datos en XML.

En XQuery las consultas pueden estar compuestas por cláusulas de hasta cinco tipos distintos. Las consultas siguen la norma FLWOR (leído como flower), siendo FLWOR las siglas de For, Let, Where, Order y Return. A continuación, se describe la función de cada bloque:

- **For**: Vincula una o más variables a expresiones escritas en XPath, creando un flujo de tuplas en el que cada tupla está vinculada a una de las variables.
- **Let**: Vincula una variable al resultado completo de una expresión añadiendo esos vínculos a las tuplas generadas por una cláusula for o, si no existe ninguna cláusula for, creando una única tupla que contenga esos vínculos.
- **Where**: Filtra las tuplas eliminando todos los valores que no cumplan las condiciones dadas.
- **Order by**: Ordena las tuplas según el criterio dado.
- **Return**: Construye el resultado de la consulta para una tupla dada, después de haber sido filtrada por la cláusula where y ordenada por la cláusula order by.

En la página del Consorcio W3C aparece una lista de fabricantes de software que ya implementan soluciones basadas en Xquery. Algunos de estos fabricantes:

- BEA's Liquid Data
- GNU's Qexo (Kawa-Query)
- Compiles XQuery on-the-fly to Java bytecodes. Based on and part of the Kawa framework. An online sandbox is available too. Open-source.
- Microsoft's SQL Server 2005 Express, with XQuery support
- También existe una DLL provisto en algunos de los documentos comentados llamada Microsoft.Xml.XQuery.dll, el documento es [Min2004]
- Oracle's Xquery Technology - Preview
- PHP. Open source.

4.3.2 Inconvenientes

En un principio parece una buena solución al problema de integrar y fundamentalmente de procesar datos que estuvieran en formato XML, debido a: la amplia gama de operaciones que se pueden aplicar, mucha similitud entre este lenguaje y SQL lo cual lo hace más simple de comprender. Pero, en la etapa de investigación de alguno de los productos que implementan xQuery, se notó: falta de funcionalidades que aparecen en la definición de xQuery, no todos ofrecían una API que se pudiera operar en forma independiente del producto, además que podía llegar a ser todo un problema el intentar generar las consultas en tiempo de ejecución, sin la participación directa del usuario, lo cual no era el objetivo en este paso, pues se pretendía tener una herramienta que asistiera a un usuario final el cual no iba a conocer la información acá requerida; otro problema no trivial es el consumo de recurso que tiene el realizar consultas sobre XML usando xQuery.

4.4 CLASIFICACIÓN DE LA INFORMACIÓN

El dominio de la información en las organizaciones es conceptualmente muy amplio, complejo, de gran valor para la organización y puede ser muy voluminoso, en algunos casos es tan voluminoso que no es viable procesarlo por personas, se requiere de mecanismos eficientes de clasificación filtrado, búsqueda y navegación; con el fin que se pueda acceder a los contenidos más relevantes y más exacto. Además esto se ve agravado por el hecho de que la información pude provenir de diferentes fuentes, de la misma organización o externa, por lo cual expresiones diferentes pueden tener el mismo significado, por ejemplo los conceptos proveedor y acreedor, o los conceptos deudor y cliente.

La solución a estos problemas esta dada por sistemas que comprendan nuestras preguntas y nos aporten la respuestas adecuadas, para ello es adecuado que la información esté estructurada, clasificada y se puedan establecer relaciones entre los conceptos, de esta forma las búsquedas y filtros serán más exactos.

Las taxonomías y ontologías sirven para tener una definición formal y precisas de los términos, poder clasificarlos y relacionarlos.

4.4.1 Taxonomías

En su sentido más general, es la ciencia de la clasificación, en las taxonomías se definen las clasificación y relaciones jerárquica entre los elementos a analizar.

Una taxonomía es una organización jerárquica basada en un conjunto de atributos de los elementos a clasificar y relacionar. Dicha organización tiene la estructura de un árbol dirigido, por lo que no sólo no tiene ciclos, sino que un elemento no puede pertenecer al mismo tiempo a más de una clasificación (rama) dentro de la organización.

Sin embargo, el hecho de que las relaciones se traten de árboles dirigidos es un grave problema, pues difícilmente un concepto puede ser clasificado de manera tajante en un sólo lugar. Es en este punto donde entran las ontologías.

4.4.1.1 XBRL (Extensible Business Reporting Language)

La necesidad de un estándar digital para el intercambio de información contable entre aplicaciones de software se acentúa si se trata de integrar múltiples datos procedentes en diversos formatos (pdf, xls, html, doc, etc.). XBRL [XBRL2005] es un estándar en la generación de reportes, ampliamente aceptado por la comunidad contable y financiera internacional desarrollado por XBRL.org, subgrupo de la W3C.

Un consorcio internacional de empresas y organizaciones, patrocinado por el AICPA (American Institute of Certified Public Accountants), entre las que se encuentran las grandes de la informática, de la contabilidad y la consultoría, e instituciones como el IASB (Internacional Accounting Standards Board), el IMA (Institute of Management Accountants), el CICA (Canadian Institute of Chartered Accountants) o el ICAEW (Institute of Chartered Accountants in England and Wales) han apoyado este emprendimiento estableciendo exigencias sobre la generación de reportes en este estandar.

Es cierto que XBRL ha nacido en el seno de AICPA http://www.aicpa.org (La asociación americana de censores y auditores de cuentas) y que el mayor avance en XBRL existe hoy en día para información financiera. Sin embargo ya en el año 1998 y 1999 XBRL se llamó XFRML (Extensible Financial Reporting Markup Language) el nombre actual (Extensible Business Reporting Language) se adoptó posteriormente, cuando se comprobó que XBRL se puede utilizar no sólo para la información financiera sino para cualquier otro tipo de informes empresariales que se deseen modelar e intercambiar

XBRL desde 1998 es un lenguaje cuya sintaxis ha sido diseñada para el intercambio de informes empresariales, basado en XML y con algunas extensiones como especificación de espacios de nombres (Namespaces), la definición de esquemas de datos en XML (XMLSchema) y la definición de recursos enlazados mediante XML (XLink) y basado en algunos principios como:

- Las definiciones de los metadatos a intercambiar fuesen definiciones estándar

- Las taxonomías fácilmente extensibles de forma que diversas organizaciones fueran capaces de publicar definiciones a medida.

4.4.1.2 Taxonomía XBRL

Según la teoría de la comunicación, para que se pueda intercambiar un mensaje entre un emisor y un receptor, debe existir un código que sea conocido por los participantes. Este es el papel de las Taxonomías XBRL.

Toda Taxonomía XBRL está basada en un esquema XML. Las reglas y limitaciones de los esquemas XML también se aplican a las taxonomías XBRL.

Una Taxonomía XBRL contiene un esquema que es el conjunto de elementos que pueden aparecer en los informes y la estructura de los mismos. Este conjunto lo denominaremos diccionario de términos definidos.

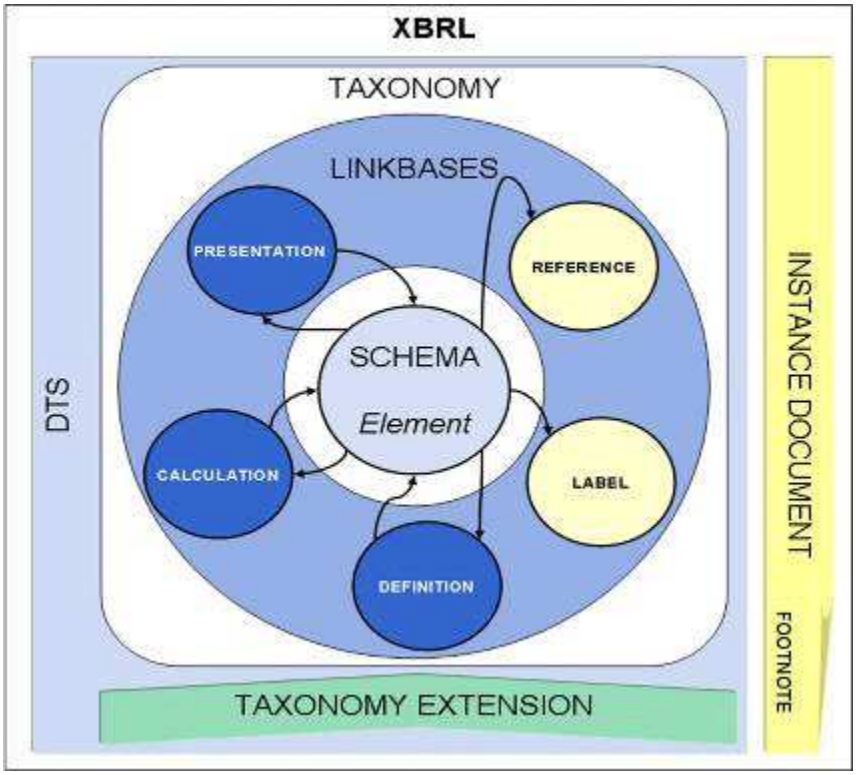
Linkbase de etiquetas: Las etiquetas o textos asociados a los elementos del diccionario que pueden utilizarse en distintos idiomas y con distintos propósitos a la hora de construir representaciones de los informes.

Linkbase de referencias: Las referencias a textos legales o normativas que fundamentan la base legal del concepto a modelar. Estas referencias juegan un papel muy importante a la hora de aclarar la utilización de los conceptos cuando se van a crear los informes.

Linkbase de presentación: Las reglas para construir una representación del informe que se pretende modelar.

Linkbase de cálculo: Las reglas de cálculo (sumas y restas) entre elementos de la taxonomía que permiten validar los informes XBRL.

Linkbase de definición: Reglas adicionales que permiten documentar relaciones entre elementos de la taxonomía y que se utilizarán para validar los informes.



Una de las taxonomías más destacadas en España es la realizada por la DGI (Datos generales de identificación) que va a recoger todos los elementos de identificación contenidos en los documentos financieros utilizados por la CNMV, la Central de Balances del Banco de España, los Registros Mercantiles y otras entidades o empresas de agregación y distribución de información

financiera, la IPP (Información pública periódica de empresas con valores admitidos a cotización oficial) que va a incorporar todos los elementos de información que las empresas cotizadas remiten periódicamente a la CNMV, de acuerdo con los anexos de su Circular 2/2002, y la PGC90 que contendrá todos los elementos de las cuentas anuales del Plan General de Contabilidad de 1990.

Para descargar la taxonomía se puede ir a http://xbrl.org.es/gp/2005-03-10/dgi-2005-03-10.zip

4.4.1.3 Inconvenientes

Los principal problema en la utilización de XBRL en nuestro sistema son dos por un lado la falta de una taxonomía regional o nacional, como citada anteriormente, y en segundo la necesidad de un manejo en forma profunda de los conceptos financiero y contable puesto que son cientos las definiciones manejadas en estas taxonomías y algunas de ellas pueden tener interpretaciones diferentes en distintas regiones.

4.4.2 Ontologías

Una ontología es una descripción formal de los conceptos y las relaciones (semánticas) entre conceptos. (GRUBER, 1993) Se usa para crear vocabularios estructurados para la recuperación o el intercambio de información. Estos miembros pueden ser humanos o agentes automáticos.

Las ontologías pueden contar con un alto grado de estructuración para especificar descripciones de clases (las cosas), relaciones entre clases y las propiedades que éstas tienen.

Son útiles para diversas aplicaciones inteligentes como es el caso de recuperación de información que puede provenir de fuentes heterogéneas, tratamiento de lenguaje natural o comercio electrónico.

A diferencia de una taxonomía, una ontología es un grafo sin jerarquía en particular, y definitivamente no es un árbol, por lo que el problema mencionado anteriormente deja de existir.

Es importante señalar que no se trata de generar conocimiento, el conocimiento ya está en el, se trata de clasificarlo para poder hallarlo de manera más eficiente.

Si bien, otros enfoques, como el de XML, han logrado cierta aproximación a este manejo de la información, no ha resultado suficiente, por lo que han surgido otras herramientas como RDF, con el que es posible crear ontologías limitadas pero útiles. En cambio si se desea una clasificación más extensa y compleja existe un lenguaje creado específicamente para definir ontologías que es Web Ontology Language (OWL) y esta acompañada de una serie de editores para simplificar su generación.

4.4.3 Web Ontology Language (OWL)

OWL es un lenguaje de Ontologías para la Web que surge como una extensión de RDFS para permitir expresiones de relaciones complejas y de mayor precisión que RDFS. OWL proporciona un lenguaje que utiliza las características definidas por RDFs para añadir las siguientes capacidades a las ontologías:

- Capacidad de ser distribuidas a través de varios sistemas
- Escalable a las necesidades de la Web
- Compatible con los estándares Web de accesibilidad e internacionalización
- Abierto y extensible

OWL extiende RDFS para permitir la expresión de relaciones complejas entre diferentes clases RDFS, y mayor precisión en las restricciones de clases y propiedades específicas. Esto incluye por ejemplo:

- Los recursos para limitar las propiedades de clases con respecto a número y tipo
- Los recursos para inferir qué elementos que tienen varias propiedades son miembros de una clase en particular
- Los recursos para determinar si todos los miembros de una clase tendrán una propiedad en particular, o si puede ser que sólo algunos la tengan
- Los recursos para distinguir entre relaciones uno-a-uno, varios-a-uno o uno-a-varios, permitiendo que las "claves externas" de las bases de datos puedan representarse en una ontología
- Los recursos para expresar relaciones entre clases definidas en diferentes documentos en la Web
- Los recursos para construir nuevas clases a partir de uniones, intersecciones y complementos
- Los recursos para restringir rangos y dominios para especificar combinaciones de clases y propiedades.

4.5 CONCLUSIÓN

Los metadatos son "datos sobre datos" y describen el contenido, calidad, condiciones, origen, aportan información para acelerar la búsqueda, facilitan las operaciones y la integración de datos etc. Permiten y ayudan al entendimiento de la información almacenada.

Para la persistencia de estos metadatos no existe un único estándar pero básicamente podemos definir dos formatos que son XML y en esquemas relacionales, cada uno de estos sistemas tiene sus pro y sus contra, por ejemplo las estructuras XML crecen rápidamente lo que lleva a un mayor consumo de recursos por lo cual las operaciones sobre estas estructuras son más lentas que en los esquemas relacionales, pero también es cierto que si las estructuras almacenadas son reducidas este tiempo es totalmente despreciable, la ventaja que ofrece XML sobre los esquemas relacionales es la interoperabilidad puesto que los sistemas relacionales están muy sujetos al ambiente donde se ejecutan, además el pasaje del modelo relacional a modelo XML es mucho mas simple que del modelo XML al relacional; en cuanto a las tecnologías y herramientas que soportan y permiten manipular las dos estructuras, como los modelos relacionales, están muy estandarizadas, en cambio XML promete una mayor evolución.

En cuanto al uso de metadata son muchas las aplicaciones que tienen, entre las cuales se destaca la habilidad que permite la interoperación de sistemas, los cuales pueden estar basados en fuentes de datos comunes o heterogéneas. Al tener una descripción de los datos de cada fuente y forma de relacionarlos, sin importar en que formato estén, se podrían llevar a un formato común y realizar operaciones sobre esos datos, colaborando así en la integración de fuentes de información.

El desarrollo de la Web han expandido las posibilidades y capacidades de los metadatos. Estas tecnologías se han adaptado para trabajar con sistemas interoperables, aumentando su riqueza en la descripción, facilitando y posibilitan el acceso a la información.

Entre las representaciones de medatatos sobresale XML, que es el estándar más usado en la actualidad. Uno de los principales objetivos de XML es permitir el intercambio de información heterogénea, basándose en una especificación simple y extensible, y para ello es imprescindible permitir y favorecer tanto la descripción de los datos a intercambiar como su manipulación.

Basándonos en XML existen especificaciones de descripción de recursos como RDF, el cual permite describir los datos de cada documento y definir las relaciones que hay entre los datos y XML.

Además de los metalenguajes y esquemas descriptivos, tenemos diferentes estándares de especificaciones para la generación de metadato como ser MOF (Meta-Object Facility), Dublin Core o IEEE LTSC LOM (Learning Object Metadata), los cuales modelan un conjunto de elementos para describir una amplia gama de recursos de Internet o sistemas con datos provenientes de fuentes diversas.

También existen muchas especificaciones basadas en XML, para describir clasificaciones, relaciones y conceptos en las más diversas áreas que pueden ir desde el área de la salud hasta las actividades contables, a estas especificaciones formales se les llama taxonomías, y aunque existe una taxonomía para la descripción de reporte contables (XBRL) está muy centrada en reportes financieros y maneja clasificaciones regionales según las regulaciones legales y conceptos financieros de cada región.

En cuanto a las herramientas de manipulación de datos que están bajo el formato de XML también se tiene barias especificaciones de herramientas, cada una de ellas con varias implementaciones, entre las que se destacan XPath o XQuery las cuales permiten realizar consultas sobre datos almacenados en formato XML.

Para representar el meta-modelo se utilizará XML, esta decisión se tomó por:

- Posibilidades de interoperabilidad que brinda.
- Se estimó que el tamaño de las instancias del meta-modelo es manejable por las herramientas disponibles para el manejo de XML.

- La gran difusión de su uso.
- La adaptabilidad de XML.

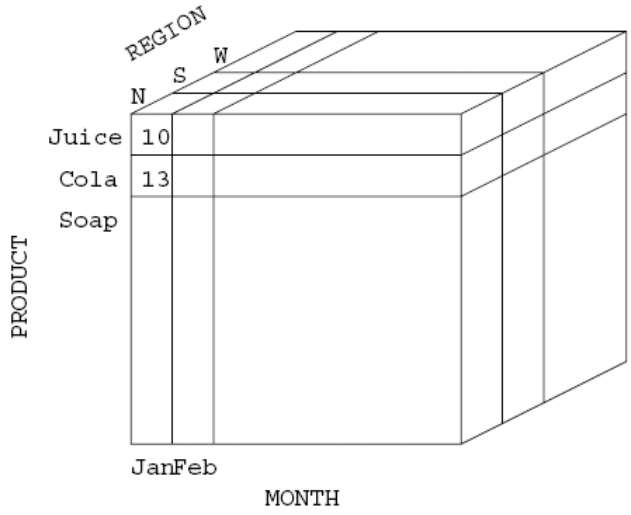
5 PROCESAMIENTOS DE DATOS

En la actualidad los volúmenes de información manejada, ya sea en esquemas relacionales o modelos multidimensionales, son muy grandes, es por ellos que últimamente se han impulsado una variedad de modelos y herramientas que optimicen los recursos para el análisis de la información en estas situaciones, entre los modelos más actuales tenemos a OLAP [per2002] [per2003]

5.1 MODELO OLAP

La principal utilidad de OLAP (procesamiento analítico en línea, **On-Line Analitycal Processing**) es que provee acceso rápido e interactivo a grandes volúmenes de datos preagregados. La información necesaria para tomar decisiones estratégicas se pone a la mano de quienes tienen que tomarlas, accediéndose a ésta mediante indicadores de desempeño claves.

OLAP toma los datos existentes en esquemas relacionales o data warehouse (data source), resumiéndolos en indicadores de desempeño claves (medidas) en un contexto (dimensiones). Los datos podrían estar distribuidos en varios repositorios, con lo que sería necesario compatibilizar los datos. Una vez que los datos se procesan a una base multidimensional (cubo) todas las medidas son preagregadas, lo que hace más rápido el acceso a la información. Así, el cubo procesado se puede hacer disponible a los usuarios quienes pueden navegarlo utilizando distintas herramientas.



5.2 ELEMENTOS DE LOS MODELOS MULTIDIMENCIONALES

Los elementos básicos de los cubos multidimensionales son: medidas, dimensiones y esquemas.

Medidas

Son los indicadores de performance claves que se quieren evaluar. Como "rule of thumb" se puede decir que: todo número que tenga sentido al agregarlo, es una medida. Por ejemplo, tiene sentido agregar las ventas por día, mes o año, pero no tendría sentido agregar números de teléfono.

Dimensiones

Son las categorías del análisis de datos. La "rule of thumb" sería: cuando un reporte se pide "por" algo, ese algo es una dimensión. Por ejemplo: productos, tiempo y región. Las dimensiones se ordenan por **niveles jerárquicos** (por ej.: año, trimestre, mes, día), los valores en los niveles se llaman **miembros**, por ejemplo 2005 y 2004 son miembros de el nivel año en la dimensión tiempo.

Cuando un cubo tiene demasiadas dimensiones (más de 12 por ejemplo) se hace difícil de entender y navegar, además se debe tener en cuenta que la cantidad de datos crece en forma exponencial al aumentar las dimensiones y niveles.

Esquemas

Las dimensiones y medidas se representan físicamente mediante esquemas estrella. La forma básica del esquema estrella consiste en tablas de dimensiones colocadas alrededor de una tabla de hechos (**fact table**). La tabla de hechos tiene como columnas a las medidas y referencias a las tablas de las dimensiones.

Históricamente, hay tres formas de almacenamiento: OLAP multidimensional, OLAP relacional y OLAP híbrido. El OLAP relacional (ROLAP) utiliza RDBMS, utilizando la tecnología tradicional de DBMS en general es más escalable, lento y mantenible que el MOLAP. Con MOLAP se utilizan estructuras especializadas para el almacenamiento y consulta de datos multidimensionales, el problema principal con MOLAP es que no es muy escalable.

5.3 META-MODELO PARA EL CÁLCULO DE VISTAS

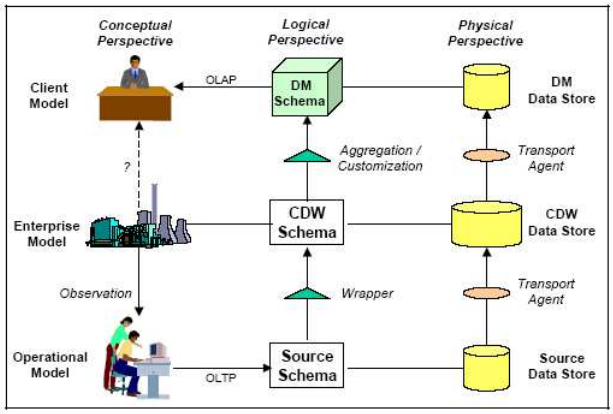
En [per2002] se utiliza una simplificación del meta-modelo desarrollado en el contexto del proyecto de Data Warehouse Quality (DWQ), y este modelo simplificado es extendido para permitir el cálculo de vistas a partir del modelo.

5.3.1 Marco de Metadata de DW

El en el proyecto DWQ se proponen, para describir un sistema de DW, tres perspectivas:

- una *conceptual* de la corporación,
- una *lógica* de los modelos de datos, y
- una perspectiva *física* del flujo de datos.

Estas perspectivas están relacionadas en forma ortogonal con las tres capas tradicionales de un sistema de DW, que son las bases de datos fuentes, el DW corporativo y los data marts, como se muestra en la siguiente figura.



La *perspectiva conceptual* concierne a los modelos de negocio de los sistemas de información de la empresa. El rol central corresponde al *modelo corporativo*, el cual brinda una visión corporativa de los *objetos* conceptuales de la corporación. También le conciernen los diferentes modelos *operacionales* y los modelos *clientes* de cada sección de la empresa. Tanto los modelos operacionales como los modelos clientes son vistas parciales del modelo corporativo.

La *perspectiva lógica* concibe al DW desde el punto de vista de los modelos de datos concretos, implementados por los *esquemas* lógicos. Los modelos conceptuales tienen una representación lógica en los esquemas clientes o data marts (DM), el esquema de empresa o data warehouse corporativo (CDW) y los esquemas fuentes, respectivamente. La perspectiva lógica también

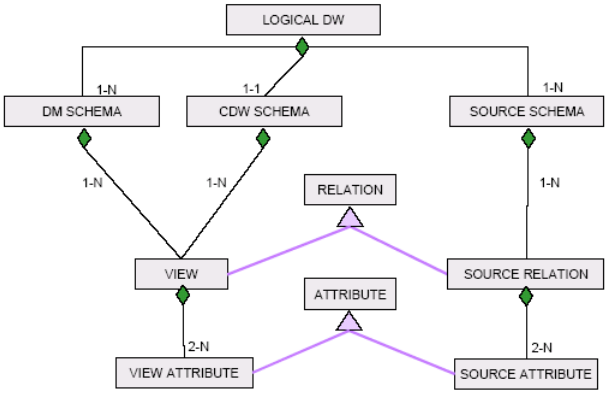
describe la manera de calcular los objetos de cada DM a partir de los objetos del CDW y éstos a partir de los objetos fuentes.

La *perspectiva física* representa los depósitos de datos correspondientes a cada esquema y el flujo de datos entre ellos.

5.3.2 Meta-modelo lógico de DW

A continuación se desarrolla la perspectiva lógica del modelo de DWQ, es decir, en los esquemas de DMs, CDW y fuentes, y en la manera de calcular los objetos de un esquema en función de los de otro.

El meta-modelo de DWQ describe las propiedades y componentes de los diferentes esquemas. La figura siguiente se muestra la versión simplificada del meta-modelo, donde se excluyen algunos de los objetos como restricciones, factores de calidad, esquemas históricos, etc.



El DW se compone de tres conjuntos de esquemas:

- los esquemas de DMs que abstraen los modelos clientes,
- el esquema del CDW que representa el modelo corporativo, y
- los esquemas fuentes que corresponden a los modelos operacionales.

Los esquemas están compuestos de relaciones que representan los objetos conceptuales, pero la naturaleza de esas relaciones es diferente. Los esquemas fuentes contienen relaciones fuentes con datos materializados y posiblemente heterogéneos, pero los esquemas de DMs y del CDW contienen vistas, posiblemente no materializadas que representan transformaciones de los datos de las relaciones fuentes. Para representar las relaciones fuentes y las vistas, se especializa la clase *Relation* en dos sub-clases *SourceRelation* y *View* respectivamente.

El mismo razonamiento se puede hacer para los atributos, obteniendo dos sub-clases *View-Attribute* y *SourceAttribute*, ambas especializaciones de la clase *Attribute*.

Este meta-modelo enfatiza la representación de los esquemas. Por ejemplo, describe el esquema de las vistas, es decir, indica los atributos de los que está compuesta. Pero para definir una vista no es suficiente con dar el esquema se necesita definir la consulta de cálculo, la cual indica cómo calcular los datos a partir de relaciones fuentes.

Dichos cálculos pueden representarse como operaciones, cuya entrada son las relaciones fuentes y sus atributos. En la siguiente sección se presenta la extensión del modelo para expresiones de cálculo de vistas.

5.3.3 Meta-modelo para el cálculo de vistas y sus componentes

Para representar una vista utilizamos la clase *ViewExpression*, la cual tiene cuatro componentes, con los siguientes roles:

- *from*. Un conjunto de relaciones que conforman la entrada de la expresión de cálculo. Una vista puede calcularse tanto a partir de relaciones fuentes como de otras vistas.
- *select*. Una expresión de proyección para cada atributo de la vista. La siguiente sección presenta una gramática para construir las expresiones.
- *where.* Un conjunto de condiciones o predicados que deben cumplir los datos de las relaciones.
- *having.* Un conjunto de condiciones o predicados que deben cumplir los datos una vez agrupados o resumidos.

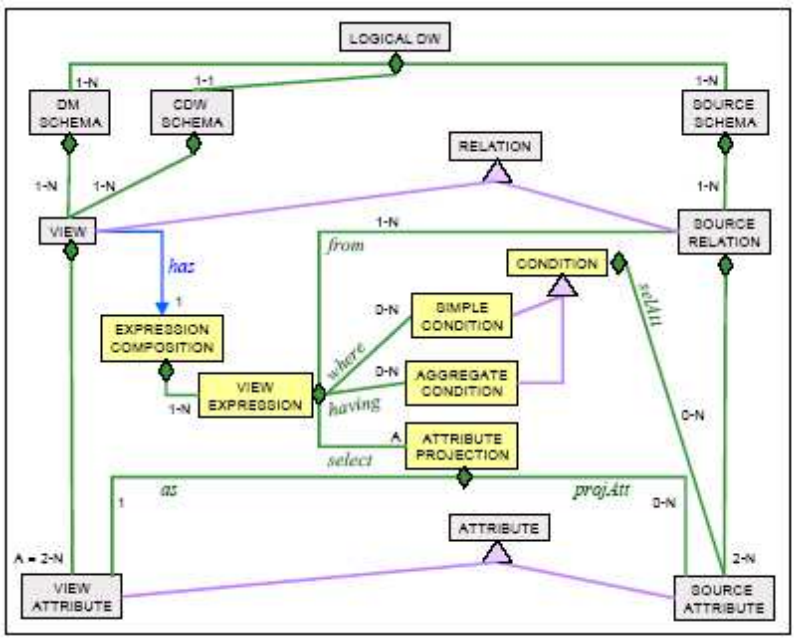
El componente *from* indica el conjunto de relaciones fuentes necesarias para calcular la vista.

El componente *select* indica el conjunto de expresiones de proyección usadas, cada una correspondiendo a un atributo de la vista. La clase *AttributeProjection* establece un mapeo entre un atributo de una vista a ser calculado (*as*) y los atributos fuentes usados para construir la expresión (*projAtt*) siguiendo las reglas de la gramática. Sólo se permite construir expresiones de proyección a partir de los atributos fuentes de las relaciones indicadas en el componente *from*.

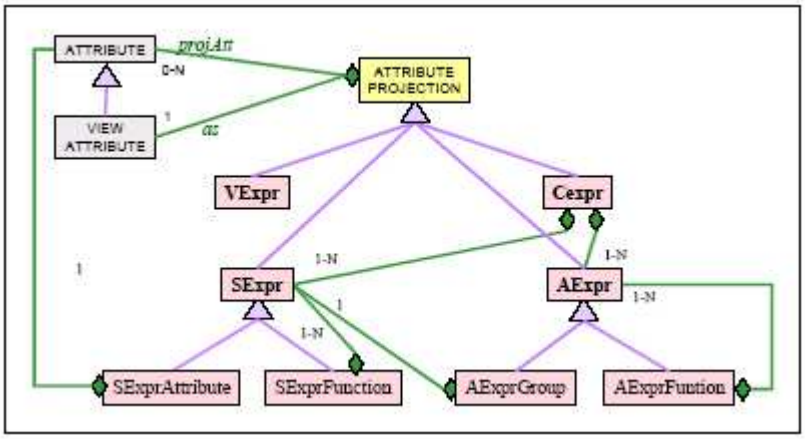
Los componentes *where* y *having* indican las condiciones sin y con agregados respectivamente. Las clases *SimpleCondition* y *AggregateCondition* son especializaciones de la clase *Condition* y siguen las reglas de la gramática de condiciones. El rol *selAtt* indica qué atributos fuentes se usan para construir una condición, que deben ser atributos de las relaciones indicadas en el componente *from*.

Una vista puede ser calculada con varias expresiones de cálculo, pero debe existir un mecanismo para componer dichas expresiones al calcular la vista, por ejemplo, realizando la unión de las instancias. La clase *Expression-Composition* expresa la manera de combinar varias expresiones para calcular la vista.

Para permitir el anidamiento de expresiones de cálculo de vistas, una vista debe poder calcularse tanto a partir de relaciones fuentes como de otras vistas. Por este motivo generalizamos el componente *from* para representar un conjunto de relaciones. La Figura siguiente muestra dicha generalización. Se debe controlar la consistencia de las instancias, impidiendo que una vista pueda calcularse a partir de si misma, o que existan ciclos en la dependencia de cálculos (componente *from)*.



Como una vista puede calcularse a partir de cualquier relación, también las expresiones de proyección y las condiciones pueden construirse a partir de atributos de esas relaciones. Entonces, los componentes *projAtt* y *selAtt* se generalizan también.



La Figura anterior muestra la jerarquía de clases para las expresiones de proyección.

Las sub-clases de la clase *AttributeProyection* representan los cuatro tipos de expresiones de proyección definidas en la gramática: VExpr, SExpr, AExpr y CExpr. Cada una puede ser extendida para representar una regla de cálculo específica. Por ejemplo, una expresión simple (SExpr) puede ser un atributo o una función sobre otras expresiones simples. Tanto el conjunto de funciones válidas cómo su cardinalidad pueden ser restringidas por cada aplicación.

El mismo tipo de jerarquías puede definirse para las clases *Condition* y *ExpressionComposition*.

5.4 CONCLUSIÓN

Como se comentó anteriormente, la principal utilidad de OLAP es que provee acceso rápido e interactivo a grandes volúmenes de datos en esquemas relacionales o data warehouse (data source). Estos datos podrían estar distribuidos en varios repositorios, con lo que sería necesario compatibilizar los datos.

El análisis de estas tecnologías (OLAP y Data Warehouse) nos provee de una base para el diseño del metadata requerido para la integración de fuentes de información, mediante la utilización de estructuras multidimensionales y para procesar la información a mostrar a través del cálculo de vistas.

Una de las principales ventajas de OLAP, como lo es el acceso a los datos en tiempo real queda descartado por razones de tiempo, que nos priva de implementar una solución que utilice estas técnicas.

6 SISTEMAS MULTIFUENTES

Existen problemas críticos al construir Sistemas Multifuentes y distintas propuestas de solución a los mismos. Al analizar esta temática es importante que nos preguntemos cual es el objetivo de la construcción de estos sistemas y encontramos tres posibles puntos [mot2002], que son:

- **Interconexión** de sistemas. El mismo refiere a poder conectar a nivel de hardware distintos sistemas entre sí. El objetivo de una interconexión es intercambiar información y para que la interconexión sea exitosa es suficiente con disponer al menos de un protocolo común de comunicación.
- **Integración** esta se refiere no sólo a intercambiar información sino también a unificar la información que se encuentre replicada, resolviendo posibles conflictos de heterogeneidad entre las diferentes fuentes de origen de la información.
- **Interoperación** se refiere no sólo a integrar información sino también a integrar funcionalidades. Hablamos entonces en forma general de sistemas interoperables cuando nos referimos a sistemas que cooperan intercambiando datos y sincronizando las ejecuciones de sus aplicaciones.

Distintos sistemas de información son interoperables cuando poseen la habilidad de dar y recibir servicios y operar como un único sistema en la resolución de un problema común.

- Las características principales de los sistemas interoperables son:
- Que exista distribución en un ambiente de Cliente/Servidor
- Que exista comunicación a pesar de heterogeneidad
- Que cada uno de los sistema usa las facilidades y funcionalidades del otro

En nuestro caso, que consiste a un usuario final permitirle la generación de reportes basándose en el modelo conceptual, el tipo de sistemas multifuentes que mejor se adapta son los sistemas integración, puesto que nuestro sistema requiere combinar información que proviene de diferentes fuentes y luego que está integrada poder realizar algunas operaciones.

6.1 INTEGRACIÓN DE SISTEMAS

Se puede tener varios niveles de integración en la arquitectura de un sistema federado según Sheth y Larson [sl90], en el nivel más simple (Nivel 0) de integración, la integración realizada es una generalización de clases sin identificar objetos comunes, es decir que cada sistema aporta alguna parte al sistema global sin distinguir objetos comunes entre los sistemas, por ejemplo si para la generación de cada reporte solo se tomara la información de un único sistemas de los diferentes sistemas integrados.

El Nivel 1 de integración se da cuando se integran subesquemas que tienen correspondencia de equivalencia, en un único subesquema homogéneo con los datos funcionados. Un ejemplo sería si para la generación de un reporte, uno de los sub sistema aporta los clientes y otro los proveedores en el sub sistema integrado se podría tener personas que no estaban en ningún sub sistema y que tiene los datos de los dos sistemas integrados.

En el Nivel 2, tenemos los sistemas que no solo comparten datos sino también funcionalidades. Por ejemplo, siguiendo el sistema anterior, podría tener personas con preferencias las cuales serian los clientes con un monto de cuota superior a uno dado, y los proveedores que permiten pagos diferidos con mas de 60 días, generando así una funcionalidad que no existía. Al lograr este nivel de integración es que hablamos de sistemas interoperables.

En el cuarto nivel se encuentra el **Esquema Federado** que corresponde al **esquema integrado** de la federación. Por último, en el nivel cinco, los **Esquemas Externos,** el resultado de transformar el esquema federado a los modelos de datos que usa cada uno de los sistemas participantes. Los esquemas externos permiten que cada participante pueda acceder a la federación en el mismo modelo en el cual están acostumbrados a trabajar.

6.1.1 Arquitectura orientada a servicios (SOA)

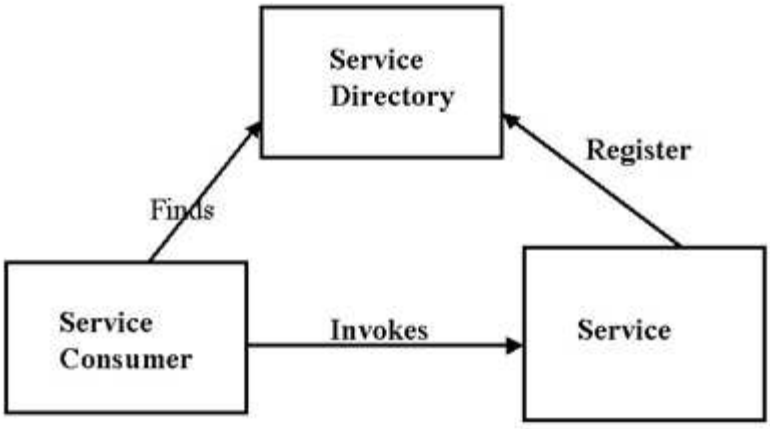
La exigencia de crear sistemas más complejos y con el mínimo tiempo y presupuesto lleva a que se use de forma óptima nuestros recursos y que se piense desde un principio la implementación de sistemas reusables. Crear estas aplicaciones, requiere en muchos casos de funcionalidades ya antes implementadas como parte de otros sistema. En este punto los arquitectos de soluciones tienen dos opciones [par2005]:

- Tratar de **reutilizar** la funcionalidad ya implementada en otros sistemas. Una labor difícil de realizar, debido a que muchos de los sistemas heredados no fueron diseñados para ese fin.
- **Re implementar** la funcionalidad requerida ("reinventar la rueda"). Aunque implica más tiempo de desarrollo, es la más fácil segura y la más escogida.

Esto trae como resultado: funcionalidad replicada en la aplicación por no haber una estrategia de integración de aplicaciones, se generan múltiples fallas, y por lo general estos modelos no escalan muy bien, el inconveniente final es una pobre respuesta al cambio.

Al diseñar sistemas empresariales debe pensarse en su integración, una buena estrategia para ello es motivar la construcción de servicios, más que de aplicaciones. Estos servicios se encargarían de elaborar las funcionalidades definidas para la aplicación. De esta manera, una aplicación final simplemente orquesta la ejecución de un conjunto de estos servicios, añade su lógica particular y le presenta una interfaz al usuario final. Por ejemplo una aplicación pensada para generar reportes, tendría que pedir a algún servicio la información a mostrar, a otros servicios el formato en que se desea presentar, y el reporteador mostrarla lo generado al usuario final.

La Arquitectura Orientada a Servicios (SOA), define los servicios de los cuales estará compuesto el sistema, sus interacciones, y con que tecnologías serán implementados. Las interfaces que utiliza cada servicio para exponer su funcionalidad son gobernadas por contratos, que definen claramente el conjunto de mensajes soportados, su contenido y las políticas aplicables. El servicio deberá registrarse en un directorio, y en éste deberán buscarlo quienes quieran consumirlo.



Un servicio debe ser una aplicación completamente autónoma e independiente y tener una interfaz de llamadas basada en mensajes. Generalmente, los servicios incluyen tanto lógica de negocios como manejo de estado de los datos. De esta forma si se genera un servicio que obtenga información de diferentes fuentes para la generación de reportes, luego también se podrá usar este servicio para mostrar los datos por pantalla.

Detectar un problema de rendimiento o funcionalidad en la integración de los servicios se puede volver muy complicado. Una solución a este problema es manejar los aspectos de procedimiento de varios servicios integrados como si fuera un solo, llamado servicio de negocio

o **macroservicios**. Un macroservicio controla las acciones paso a paso en la ejecución de algún trabajo, moviendo el sistema de un estado a otro. En cada paso, este llamará una operación de negocio provista por un servicio.

Así, un servicio de negocio centraliza la definición del proceso, en vez de tener piezas del proceso entre todos los servicios. Esto disminuye las dependencias entre servicios y las aplicaciones clientes. Ayudando a facilitar la administración del sistema.

6.1.1.1 Estrategias de Implementación de SOA

Un tema no menor son los protocolos de comunicación, el objetivo es que estos protocolos sean los más flexibles y que permita la integración de sistemas [sch2005]. Si bien la forma más habitual de implementar este tipo de arquitecturas es mediante Web Services, se pueden utilizar diferentes formas de comunicación entre los servicios, que a su vez son implementados por distintos, alguna de ellas son:

- Web Services XML tiene como ventajas que es un protocolo abierto (independiente de plataforma), independiente de ubicación, fácil de implementar y promueven una arquitectura desacoplada; las desventajas es que su rendimiento es más bajo que el de un protocolo binario, además no tiene todas las capacidades de mensajería requeridas por SOA.
- Web Services XML con Web Services Extensions (WSE) el cual permite configurar un canal binario sobre el que va toda la mensajería SOAP, además provee muchas de las características requeridas por SOA implementando las especificaciones WS-Security, WS-Routing, WS-ReliableMessaging, entre otras. Las ventajas que tiene son que a pesar de ser basado en estándares, las implementaciones no son muy diferentes a la de Microsoft, dificultando la interoperabilidad. Como la independencia de plataforma es muy importante en una arquitectura basada en SOA, esta parece ser la mejor opción pensada en la escalabilidad y la interoperabilidad del sistema. Los Web Services Extensions es una estrategia a que involucra la implementación de la versión 1.2 de SOAP y de las especificaciones conocidas como WS-*. Las cuales se encargaran de darle la seguridad, confiabilidad y demás requisitos de los sistemas basados en Web Services. Estas especificaciones están dadas por la organización WS-I.
- Remoting las ventajas que tiene son el muy buen rendimiento si se utiliza un serializador binario sobre TCP, pero como desventajas que utiliza un protocolo propietario. Aunque existen productos que permiten interoperabilidad con Java 1.2+. Además no tiene una mensajería confiable y el enrutamiento de mensajes requeridas por SOA deben ser implementados manualmente.
- Java RMI (Remote Message Interface), representa una tecnología propietaria que brinda un mejor rendimiento que los Web Services. Permite el desarrollo de aplicaciones cliente-servidor.

6.1.1.2 Web Services

Los servicios Web es la tecnología de punto en la implementación de proceso distribuido o integración de sistemas en la actualidad y la que mejor se adapta a la arquitectura SOA.

Las aplicaciones que se construyen utilizando múltiples servicios Web XML desde diversas fuentes que trabajan conjuntamente con independencia de dónde residen o cómo hayan sido implementadas.

Probablemente existan tantas definiciones de los Servicios Web XML como compañías que los desarrollan, pero casi todas las definiciones tienen estos aspectos comunes, Como lo muestra la siguiente imagen

Los Servicios Web XML exponen funcionalidad útil a los usuarios mediante un protocolo Web estándar. En la mayoría de casos, el protocolo utilizado es Simple Object Access Protocol (SOAP).

Los Servicios Web XML proporcionan un modo de describir sus interfaces con suficiente detalle para permitir a un usuario construir una aplicación cliente para hablar con ellos. Esta descripción se proporciona generalmente en un documento XML que responde al nombre de documento Web Services Description Language (WSDL).

Los Servicios Web XML se registran de modo que los potenciales usuarios puedan encontrarlos. Esto se realiza mediante Universal Discovery Description and Integration (UDDI).

Una de las ventajas principales de la arquitectura de servicios Web XML es que permite a los programas escritos en diferentes lenguajes sobre diferentes plataformas comunicarse entre sí de un modo basado en estándares. Pero muchas otras tecnologías como CORBA y antes sobre DCE también aseguraron esas facilidades. Pero la primera diferencia es que SOAP es bastante menos complejo que las anteriores aproximaciones, de modo que la barrera de entrada para una implementación SOAP compatible con los estándares es significativamente menor. La otra ventaja significativa que tienen los servicios Web XML sobre anteriores iniciativas es que trabajan con protocolos Web estándares: XML, HTTP y TCP/IP.

Entonces definimos un servicio Web XML como un servicio software expuesto en la Web mediante SOAP, descrito con un archivo WSDL y registrado en UDDI.

Exponer las aplicaciones existentes como servicios Web XML permitirá a los usuarios construir nuevas y más potentes aplicaciones que utilicen servicios Web XML como bloques de construcción. Por ejemplo, un usuario podría desarrollar una aplicación de compras para obtener automáticamente la información de precios de varios fabricantes, que permitiera seleccionar un fabricante, enviar el pedido y a continuación realizar seguimiento del envío hasta que sea recibido. La aplicación del fabricante, además de exponer sus servicios en la Web, podría a su vez utilizar servicios Web XML para verificar el crédito del cliente, realizar un cargo en su cuenta y realizar el envío con una empresa de transporte.

6.1.1.3 SOAP (Simple Object Access Protocol)

SOAP es el protocolo de acceso a objetos simple. La versión actual es la 1.1 y la especificación se puede encontrar en www.w3.org/tr/soap. Se basa en XML y describe un formato de mensajería para la comunicación entre equipos. Contiene también varias secciones opcionales que describen las llamadas a métodos (RPC) y proporcionan detalles sobre el envío de mensajes SOAP a través de HTTP.

SOAP es el protocolo de comunicaciones para los servicios Web XML. Al estar descrito como protocolo de comunicaciones se piensa como implementa la invocación de objetos o los cuales

son los servicios de nombre. Pero SOAP define el formato XML para los mensajes. Si tenemos un fragmento XML correctamente definido e incluido en un par de elementos SOAP, ya tenemos un mensaje SOAP.

Existen otras partes de la especificación SOAP que describen cómo representar datos utilizando XML y cómo utilizar SOAP para realizar llamadas a procedimientos remotos. La mayoría de las implementaciones actuales de SOAP soportan aplicaciones RPC porque los programadores que están acostumbrados a crear aplicaciones COM o CORBA ya conocen el estilo RPC. SOAP también soporta aplicaciones de tipo documental, en las que el mensaje SOAP es simplemente un envoltorio sobre un documento XML. Éstas son muy flexibles y muchos servicios Web XML lo aprovechan para construir servicios que serían difíciles de implementar utilizando RPC.

Otra parte opcional de la especificación SOAP define el aspecto de un mensaje HTTP que contiene un mensaje SOAP. Este enlace con HTTP es importante porque HTTP está soportado por casi todos los Sistemas Operativos actuales (y muchos otros no tan actuales). El enlace HTTP es opcional, pero casi todas las implementaciones SOAP lo soportan, porque es el único protocolo estandarizado para SOAP. Por esta razón, existe la idea generalizada y equivocada de que SOAP requiere HTTP.

Una fuente importante de confusión cuando nos iniciamos con SOAP es la diferencia entre la especificación SOAP y las múltiples implementaciones de la especificación SOAP. La mayor parte de personas que utilizan SOAP no escriben mensajes SOAP directamente, pero utilizan un kit de herramientas SOAP para crear y parsear los mensajes SOAP. Estos kits de herramientas generalmente traducen llamadas a funciones con un cierto lenguaje a un mensaje SOAP. Por ejemplo, el kit de herramientas Microsoft SOAP Toolkit 2.0 traduce llamadas de funciones COM a SOAP y el kit de herramientas Apache Toolkit traduce llamadas de funciones Java a SOAP.

6.1.1.4 Seguridad

Una de las primeras preguntas que se formulan los recién llegados a SOAP es cómo trata SOAP la seguridad. En las primeras etapas de su desarrollo, SOAP se percibía como un protocolo basado en HTTP, y por ello se supuso que la seguridad HTTP sería adecuada para SOAP. Por esta razón, el estándar SOAP actual asume que la seguridad es una cuestión de transporte y no dice nada sobre los aspectos de seguridad.

Cuando SOAP se expandió hasta convertirse en un protocolo de propósito general ejecutándose sobre diversos transportes, la seguridad se convirtió en un aspecto más importante. Actualmente existen especificaciones de trabajos basados en SOAP para proporcionar funcionalidades de seguridad adicionales para los servicios Web. La especificación WS-Security define un sistema de encriptación completo.

6.1.1.5 WSDL

Las siglas WSDL significan Web Services Description Language. Para nuestro propósito, podemos decir que un archivo WSDL es un documento XML que describe un conjunto de mensajes SOAP y cómo se realiza el intercambio de mensajes. Como WSDL es XML, es legible y editable pero en la mayoría de casos, se genera y se consume por parte de software.

Para ver el valor de WSDL, imaginemos que queremos empezar invocando un método SOAP que proporciona uno de nuestros socios de negocio. Podríamos pedirle algunos mensajes, podríamos ver un ID de cliente de 2837 y asumir que es un entero cuando de hecho es una cadena. WSDL especifica el contenido de un mensaje de petición y el aspecto del mensaje de respuesta en una notación inequívoca.

La notación que utiliza un archivo WSDL para describir formatos de mensajes está basada en el estándar XML Schema lo cual significa que es neutral respecto del lenguaje de programación y que está basado en estándares, lo que lo hace apropiado para describir interfaces de servicios Web XML accesibles desde una amplia variedad de plataformas y lenguajes de programación.

Además de describir el contenido de los mensajes, WSDL define dónde está disponible el servicio y qué protocolo de comunicaciones utilizar para hablar con el servicio. Esto significa

que el archivo WSDL define todo lo necesario para escribir un programa que interactúe con un servicio Web XML.

En resumen WSDL es un estándar que surgió para cubrir las siguientes necesidades:

- Tipos de datos reconocidos por el servicio Web
- Esquema de mensajes
- Método de intercambio utilizado por el servicio Web (solicitud-respuesta, unidireccional, multidifusión, etc.)
- Ubicación del servicio Web
- Información de errores
- Información de encabezado

Existen varias herramientas disponibles para leer un archivo WSDL y generar el código necesario para comunicarse con un servicio Web XML. Alguna de las herramientas las podemos encontrar en Microsoft Visual Studio® .NET.

6.1.1.6 UDDI (Universal Discovery Description and Integration)

UDDI ofrece los servicios que necesitamos, obtener información sobre el servicio ofrecido y contactar con alguien para más información. Se puede ofrecer un servicio Web sin registrarlo en UDDI, pero si deseamos llegar a un mercado significativo, necesitamos UDDI para que los clientes puedan encontrarlo.

Una entrada de directorio UDDI es un archivo XML que describe un negocio y los servicios que ofrece. Una entrada en el directorio UDDI está formada por tres partes:

- Las "páginas blancas" describen la compañía que ofrece el servicio: nombre, dirección, contactos, etc.
- Las "páginas amarillas" incluyen categorías industriales basadas en taxonomías estándares como el North American Industry Classification System y el Standard Industrial Classification.
- Las "páginas verdes" describen el interfaz al servicio con suficiente detalle para alguien que desarrolle una aplicación que consuma el servicio Web.

El modo en que se definen los servicios es mediante un documento UDDI llamado Type Model o tModel. En muchos casos, tModel contiene un archivo WSDL que describe un interfaz SOAP a un servicio Web XML, pero tModel es suficientemente flexible para describir prácticamente cualquier tipo de servicio.

El directorio UDDI también incluye varias formas de buscar los servicios que necesitamos para construir aplicaciones. El directorio UDDI proporcionará información, contactos, enlaces e información técnica para permitirnos evaluar qué servicios satisfacen nuestros requerimientos. UDDI nos permite encontrar negocios de los que podemos obtener servicios Web.

Si ya conocemos con quién queremos hacer negocio pero no sabemos qué servicios ofrece, la especificación WS-Inspection nos permite navegar por una serie de servicios Web XML ofrecidos en un determinado servidor para encontrar los que satisfacen nuestras necesidades.

Hasta ahora hemos hablado sobre cómo hablar con los servicios Web XML (SOAP), cómo se describen los servicios Web XML (WSDL) y cómo encontrar los servicios Web XML (UDDI). Estos protocolos constituyen un conjunto de especificaciones que proporcionan la base para la integración y agregación de aplicaciones.

6.2 ARQUITECTURA DE WRAPPERS Y MEDIADORES

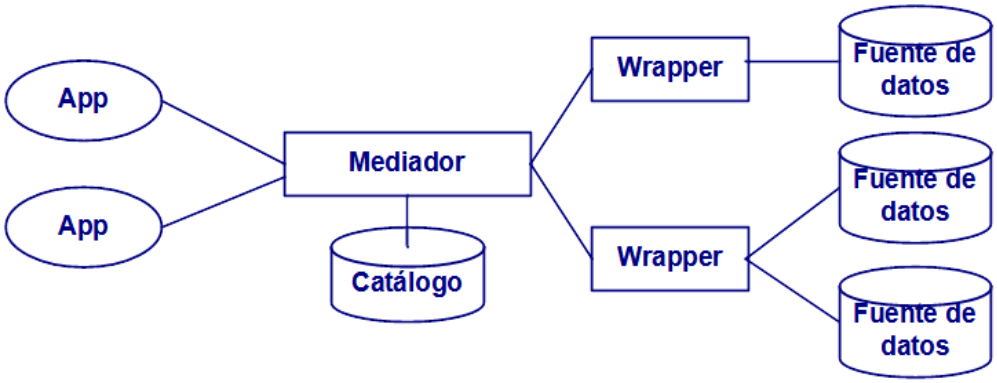
Un problema común que deben enfrentar las organizaciones en el día de hoy es la diversidad de fuentes de información y repositorios distribuidos con las que deben tratar. Alguno de estos ejemplos son: bases de datos, sistemas de archivos planos, documentos estructurados, etc.

El proceso de toma de decisiones necesita información de fuentes de datos múltiples, heterogéneas y distribuidas, por lo que la integración de fuentes de datos heterogéneos tiene una importancia considerable. La integración de múltiples fuentes requiere crear una vista integrada que permita realizar consultas en forma distribuida.

Las características para una integración de múltiples fuentes de datos pueden resumirse como:

- El número de fuentes de datos puede ser muy alto y hacer de la integración en una vista y la resolución de conflictos un gran desafío.
- Agregar y quitar fuentes de datos del sistema debería realizarse con mínimos impactos en la vista integrada.
- Las fuentes de datos varían en capacidades de computación. Por ej; archivos de texto plano, hojas de cálculo, sistemas de bases de datos.
- La naturaleza de las fuentes de datos estructuradas, semi-estructuradas y nada estructuradas puede no aportar información para la vista integrada.

Como forma de atacar estas características se plantean varias arquitecturas, dentro de las cuales encontramos la arquitectura de wrappers y mediadores.



Aquí se muestra la arquitectura de wrappers y mediadores, donde los wrappers se utilizan para proveer acceso a las fuentes de datos heterogéneas. Para cada fuente de datos, un wrapper exporta información a cerca de su esquema fuente, sus datos y las capacidades de consultas. Un mediador almacena la información aportada por los wrappers en una vista unificada de todos los datos disponibles en un diccionario o catálogo de datos central. Cuando una aplicación realiza una consulta, el mediador descompone la misma en consultas más pequeñas que se envían a los wrappers que corresponda para que estos las ejecuten en sus propias fuentes de datos, luego se obtienen los resultados de dichas consultas desde los wrappers implicados y se unen estos resultados de forma de satisfacer la consulta original de la aplicación.

6.3 CONCLUSIÓN

En este punto vemos una serie de arquitecturas e implementaciones desarrolladas con el fin de solucionar el problema de integrar diferentes sistemas de información.

Estas arquitecturas no son excluyentes una de otra, sino que en un entorno del sistema se puede aplicar una de ellas, en otro contexto otra de las arquitecturas, o también combinarse para solucionar un problema común.

Un ejemplo de uso de dos arquitecturas sería que para cada fuente de información se creara un Wrapper. Este expondría el servicio entregando los datos solicitados en un formato preestablecido adaptando la información a un formato común. Entre los consumidores de la información y el conjunto de adaptadores de las fuentes se tendría un mediador el cual se encargue de unir la información y ofrecerlo como si fuera una única fuente de información.

La ventaja que tiene Web Services sobre las otras tecnologías es ser un protocolo abierto e independiente de plataforma, lo cual le da una gran compatibilidad con productos de otras

empresas, además de tener solucionado una amplia gama de problemas como seguridad, comunicación, interoperabilidad etc. Las desventajas es que su rendimiento es más bajo que el de un protocolo binario y en algunos caso esto puede ser un punto critico en un sistema. Por otro lado las implementaciones existentes de WS no se apegan al estándar por lo que la integración de los WS que utilizan distintas implementaciones suelen ser dificultosas.

Para la implementación de estas arquitecturas existen varias tecnologías, entre éstas resultan de especial interés Web Services, Remoting, COM, COM+ entre otras, dada la orientación de cliente a las tecnologías .Net.

También se utilizará la arquitectura de wrappers y mediadores, que permite la integración de sistemas existentes.

7 LENGUAJES

Un lenguaje está formado por un conjunto de símbolos básicos (alfabeto) y un conjunto de reglas que especifican como manipularlos [bre2003]. También se debe definir un significado a las cadenas formadas por los símbolos básicos.

7.1 TRADUCTORES Y COMPILADORES

Cuando se define un lenguaje (lenguaje de programación) para interactuar con un sistema informático, entonces este lenguaje se debe de traducirse a datos que el sistema esta preparado para almacenar o en órdenes las cuales el sistema está preparado para ejecutar.

Los lenguajes de programación pueden clasificarse de acuerdo a su semejanza con el lenguajes maquina o a su semejanza con el lenguaje humano (generalmente inglés) en lenguajes de bajo nivel y lenguajes de alto nivel.

También pueden clasificarse en cuanto a su forma de ejecución en interpretados y en compilados. Los lenguajes compilados son lenguajes de alto nivel en los que las instrucciones del, código fuente, se traducen del lenguaje utilizado a código máquina propio de un sistema. Por el contrario un lenguaje interpretado las líneas del código fuente se traducen o interpretan una a una siendo más lentos que los programas compilados. Eventualmente puede haber un traductor del código fuente a un lenguaje intermedio el cual es interpretado línea a línea, por ejemplo Java y algo similar hacen los lenguajes .Net.

Un traductor es un programa que recibe como entrada código escrito en un cierto lenguaje y produce como salida código en otro lenguaje. Generalmente el lenguaje de entrada es de más alto nivel que el de salida. Ejemplos de traductores son los ensambladores y los compiladores.

Un ensamblador es un programa que traduce de un lenguaje ensamblador a lenguaje máquina, mientras que un compilador es un programa que traduce de un lenguaje de alto nivel a un lenguaje de bajo nivel o a lenguaje máquina.

Un traductor es un programa que toma el texto escrito en un lenguaje (el lenguaje fuente) y lo convierte en el texto equivalente en un segundo lenguaje (el lenguaje destino u objeto).

7.2 GENERACIÓN DE COMPILADORES

La traducción puede ser realizada en forma "manual" por un programador, el cual implementa un sistema que lee cada comando y una serie de parámetros, y selecciona de una lista una tarea a ejecutar; otra técnica es crear un traductor o un compilador en base a la sintaxis y la semántica del lenguaje, este sistema será quien convierta un programa escrito en ese lenguaje en una serie de ordenes o información a operar.

A estos programas, que permiten crear traductores, se les suele llamar compiladores de compiladores o Metacompiladores y son programa que recibe como entrada las especificaciones del lenguaje para el que se desea obtener un compilador y genera como salida el compilador para ese lenguaje.

7.3 DISEÑO DE LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación puede definirse describiendo:

- Las expresiones que pueden aparecer en sus programas (la sintaxis del lenguaje)
- Lo que significan sus programas (la semántica del lenguaje).

La sintaxis del lenguaje se presenta con una notación denominada gramática libre de contexto o BNF. (Backus-Naur Form).

La semántica del lenguaje es más difícil de expresar que la sintaxis y generalmente se decide por especificarla usando descripciones informales y ejemplos.

7.3.1 Compilador.

Cualquier compilador debe realizar dos tareas principales: análisis del programa a compilar y síntesis del código fuente en lenguaje maquina que, cuando se ejecute, realizara correctamente las actividades descritas en el programa fuente.

Para el estudio de un compilador, es necesario dividir su trabajo en fases. Cada fase representa una transformación al código fuente para obtener una representación intermedia o el código objeto, las etapas más comunes son:

- **Análisis Léxico**: en la fase de análisis léxico se leen los caracteres del programa fuente y se agrupan en cadenas que representan los componentes léxicos. Cada componente léxico es una secuencia lógicamente coherente de caracteres relativa a un identificador, una palabra reservada, un operador o un carácter de puntuación. A la secuencia de caracteres que representa un componente léxico se le llama lexema o token.
- **Análisis Sintáctico:** en esta fase, los componentes léxicos se agrupan en frases gramaticales que el compilador utiliza para sintetizar la salida.
- **Análisis Semántico**: la fase de análisis semántico se intenta detectar instrucciones que tengan la estructura sintáctica correcta, pero que no tengan significado para la operación implicada.
- **Generación de código Intermedio**: algunos compiladores generan una representación intermedia explícita del programa fuente, una vez que se han realizado las fases de análisis. Se puede considerar esta operación intermedia como un subprograma para una máquina abstracta. Esta representación intermedia debe tener dos propiedades importantes: debe ser fácil de producir y fácil de traducir al programa objeto.
- **Optimización de Código**: en esta fase se trata de mejorar el código intermedio, de modo que resulte un código de máquina más rápido de ejecutar.
- **Generación de Código**: esta constituye la fase final de un compilador. En ella se genera el código objeto

7.4 EJEMPLO DE GENERADORES DE COMPILADORES

Lex y Yacc

Descripción: los generadores más populares de analizadores léxicos y sintácticos LALR(1).

Lenguaje: Pascal - C

ParseGenerator

Descripción: es una IDE (Entorno Integrado de Desarrollo), bajo Windows32, para los generadores AYACC y ALEX, clones de Yacc y Lex respectivamente.

Lenguaje: C - C++

Flex y Bison

Descripción: versiones mejoradas (generan analizadores más rápidos) de Lex y Yacc.

Lenguaje: C

JLex y Cup

Descripción: los generadores más populares de analizadores léxicos y sintácticos para java.

Lenguaje: para java

Coco/R

Descripción: es un generador de parsers descendentes.

Lenguaje: c# entre otros

PCCTS

Descripción: es un conjunto de herramientas para la construcción de traductores y reconocedores de lenguajes. Comprende tres herramientas: ANTLR un generador de parsers LL(k), DLG un analizador de analizadores léxicos y SORCERER un generador de parsers.

Lenguaje: c# entre otros

7.5 CONCLUCIONES

Para el desarrollo del lenguaje se hace necesaria la utilización de un generador de compiladores. De las herramientas analizadas se eligió **Coco/R** (http://www.ssw.uni-linz.ac.at/Coco/), por la similitud de esta con herramientas utilizadas anteriormente y la estabilidad que presento la herramienta en las pruebas iniciales que se realizaron.

8 REFERENCIAS

[cer2004] Certia "Microsoft SQL Server 2000 Reporting Services vs. Business Objects Crystal Reports / Crystal Enterprise" Business Intelligence Team of Certia 2004

[RDL2003] "Report Definition Language Specification" Mircosoft 2003

[RS2003] "Libro en pantalla de Reporting Services", Microsoft 2003 (http://msdn.microsoft.com/library/en-us/RSPORTAL/HTM/rs_gts_portal_3vqd.asp y http://www.microsoft.com/sql/reporting/default.asp

[boe2005] Producto: Business Objects Enterprise Empresa: Business Objects

[msa2005] Producto: MicroStrategy ArchitectPara Empresa: MicroStrategy Página http://www.microstrategy.com/

[cgi2005] Producto: Cognos Impromptu Empresa: Cognos

[msrs2005] Producto: SQL Server Reporting Services Empresa: Microsoft

[ecbi2005] Eclipse BIRT Home (http://www.eclipse.org/birt/phoenix/)

[gar2000] Jesús García Molina, M. José Ortín, Begoña Moros, Joaquín Nicolás, Ambrosio Toval, Departamento de Informática y Sistemas Facultad de Informática. Universidad de Murcia "De los Procesos del Negocio a los Casos de Uso" 2000

[Gut2005] J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres, D.Villadiego "XQuery" Universidad de Sevilla, Lenguajes y Sistemas Informáticos 2005

[rui2004] Luis Ruiz Pavón "Consultas sobre documentos XML" www.elguille.info 2004

[Min2004] Mariano Minoli "Presentando XQuery en sociedad" Microsoft 2004

[XBRL2005] XBRL España"Libro blanco xbrl" XBRL España, Miembro de XBRL Internacional http://www.xbrl.org/

[cab2004.] Ismael Caballero "XBRL, el lenguaje estándar para entornos financieros y contables", 2004

[per2003] V. Peralta, "A Framework For Multi-Source Information Systems Development"

[per2002] V. Peralta, Z. Kedad: Una plataforma basada en Metadata para Cálculo de Vistas en Sistemas de Data Warehousing". Instituto de Computación - Universidad de la República Montevideo, Uruguay. 2002

[mot2003] Regina Motz, "Multifuentes", Instituto de Computación de la UDELAR, Montevideo, Uruguay. 2003

[SL90] A. P. Sheth and J. A. Larson, "Federated Database Systems and managing distributed heterogeneous, and autonomous databases" ACM Computing Surveys, 1990.

[sch2005] Ariel Schwindt, "Construcción de Sistemas Multiplataforma basados en Servicios", Microsoft 2005

[par2005] José David Parra, "Hacia una Arquitectura Empresarial basada en Servicios", Microsoft 2005

[dav2000] José David Parra, SOA BIRT BIRT, 2000

[bre2003] Ramon Brena, "Autómatas y Lenguajes", Tecnológico de Monterrey 2003

VISUAL REPORTING SERVICES

ANEXO II

META-MODELO DE NEGOCIO

ÍNDICE

1 INTRODUCCIÓN

El principal objetivo del metadata es la representación del meta-modelo de negocio del cliente, el mapeo de los datos con las fuentes de información, y la creación de nuevas entidades a partir de entidades existentes o vínculos con las fuentes de información, también está presente el formato de como se va a presentar la información para cada entidad.

Todo el modelo de negocio será representado en un **esquema** el cual tendrá un nombre y una colección fuentes de información y otra de entidades.

Cada **fuente de información** tendrá un nombre, un tipo y una colección de parámetros con los que se indicará como se puede acceder a los servicios ofrecidos por dicha fuente.

Las **entidades** tendrán un nombre y estarán formadas por atributos, relaciones y una forma de obtener los datos.

Los **atributos** tendrán un nombre, un tipo, además podrán ser clave, múltiples, ocultos u horizontales, y se mantendrán dentro de una colección de atributos.

Las **relaciones** tendrán una entidad local, que es aquella en la que se declara la relación, un campo local, otra entidad (foránea) con la cual se establece la relación y un campo foráneo que pertenece a esta última. Los tipos de relación serán: relaciones simples, especialización o generalización. Cuando una entidad sea una especialización de otra tendrá todos los atributos de ésta, pudiendo agregar otros que puedan ser calculados a partir de las relaciones con otras entidades. Otro tipo de relación será generalización en donde una entidad tiene algunos atributos de otra y agrega nuevos atributos. También se podrá mantener la cardinalidad de la relación entre dos entidades pudiendo ser esta N a N (por defecto), 1 a N, N a 1 y 1 a 1.

Los datos podrán surgir de los servicios ofrecidos por las fuentes de información o de restricciones y cálculos a partir de las relaciones con otras entidades. Según la forma en que se obtengan sus datos las entidades son **servidas**, **relacionales** o **de reporte**.

Cuando los datos de una entidad surgan de las fuentes de información, a esta entidad le llamamos **entidad servida**. Estas tendrán un conjunto de servicios, los cuales se compondrán de una referencia a la fuente de información que ofrece dicho servicio y un conjunto de parámetros. Los parámetros podrán ser constantes, variables o funciones, los cuales serán aplicados cuando se invoque al servicio de la fuente de información.

Si los datos surgen de una relación de expresión en la cual se combinan los datos de otras entidades, entonces la entidad será una **entidad relacional**. En este caso para obtener los datos de una entidad se aplican cálculos, agrupaciones y ordenamientos por uno varios atributos y/o restricciones sobre otras entidades las cuales deben estar relacionadas con esta, se deberá seleccionar que atributos se relacionarán con que cálculos o atributo de las otras entidades.

Un tercer tipo de entidades son las **entidades de reporte**, el origen de los datos de estas pueden ser tanto servicios como expresiones de relación, la particularidad de estas entidades es que permiten la definición de variables. Estas variables pueden ser utilizadas:

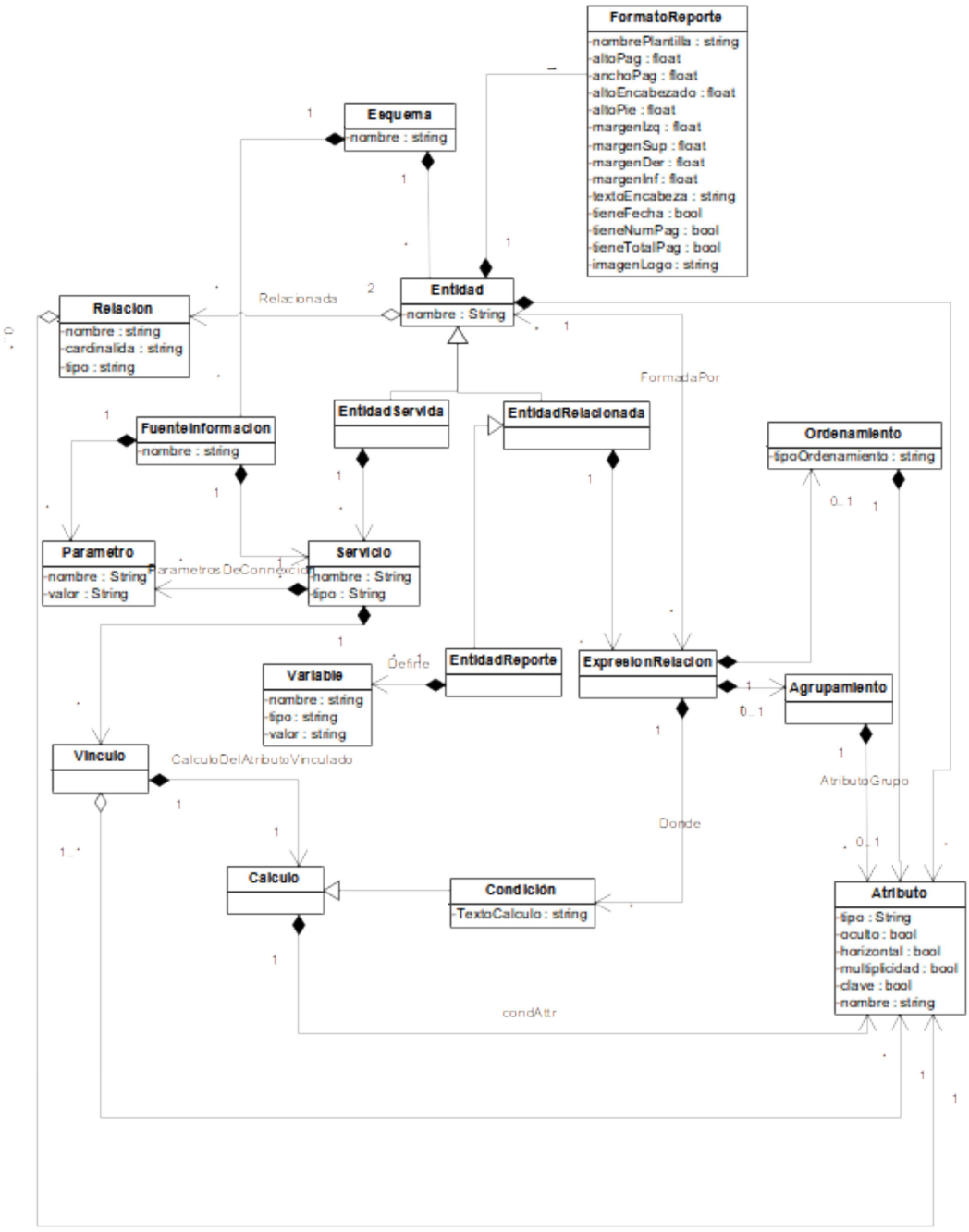
- En los parámetros de los servicios,
- En las condiciones de las expresiones de relación,
- en los vínculos de servicios o expresiones de relación (que veremos a continuación).

Por otro lado, este tipo de entidades no podrá ser referenciado desde otras entidades.

Cuando se solicita la generación de un reporte, se ejecuta un servicio de una fuente de información, con lo cual se obtiene un resultado que será un conjunto de datos los cuales deberán ser mapeados con los atributos, este mapeo se resuelve a través de los vínculos.

Tanto para los servicios como para las expresiones de relación se deberan definir **vínculos**, estos vínculos mapean los resultados de las fuentes de información o de las relaciones con los atributos de las entidades. Es aquí, en los vínculos, que se indican cuales son los calculos y funciones a aplicar sobre los datos que se obtienen de servicios y expresiones de relación.

En el siguiente diagrama se muestra como están relacionados los conceptos más importantes presentados anteriormente y que información mantiene cada uno de estos conceptos. Por mayor detalle se puede ver el documento **7 - Representacion Intermedia.doc,** donde se describen cada una de las clases que implementan de la representación intermedia, las cuales se corresponden con los conceptos manejados en el Meta-Modelo de Negocio.



2 REPRESENTACIÓN DE LA METADATA

La persistencia de la metadata se realiza en formato XML, con el fin de impulsar la interoperabilidad del sistema, la independencia con el ambiente a ejecutar y por ser más abierto.

3 ESQUEMA

El primer nodo del XML será el nodo esquema, que tendrá como único atributo el nombre el cual coincide con el nombre del archivo donde esta guardado.

Por ejemplo:

```
<Esquema Nombre="c:\00VAR">
```

Este nodo contendrá dos nodos más, que son ColFuentes y ColEntidades.

Cabe destacar que todos los nodos que tengan el prefijo "Col" significa que contendrán una colección de nodos de cierto tipo. Estos nodos tendrán un atributo llamado Autonumerico, el cual permite generar un id único para cada elemento de la colección, el valor del este atributo es el mismo para todos los nodos del mismo tipo dentro del mismo esquema; este valor es incrementado cuando se agrega un elemento a una colección de este esquema y no se decrementa cuando se elimina el elemento. Por ejemplo si se agrega un nodo Atributo en cualquier Entidad el Autonumerico de todos los ColAtributos de todas las Entidades se incrementara en uno. Además cada elemento de una colección tiene un atributo llamado Id el cual tiene un valor único para cada elemento de la colección existente en el esquema.

Pro ejemplo:

```
<ColAtributos Autogenerado="33">
```

3.1 COLFUENTES

El nodo ColFuentes contendrá una serie de nodos cada uno relacionado con un origen de datos, cada uno de los nodos tendrá un nombre único, un tipo que podrá tener los valores: **Webservices**, **DLL** o **Remoting** y una serie de parámetros con sus nombres, estos parámetros varían según el servicio y las necesidades de integración. Según el tipo de la fuente el orígen de datos puede ser un Web Services, una DLL o un servicio de Remoting.

El siguiente es un ejemplo de la definición de una fuente de información del tipo `webservice`, en ella solo se utiliza una parámetro el cual indica la URL para acceder a él:

```
<ColFuentes Autogenerado="1">
  <FuenteDeInformacion IdFuente="0" Nombre="Magma" Tipo="webservice">
    <Parametro Nombre="url">
      http://localhost/WSMagma/MagmaServicioWeb.asmx
    </Parametro>
  </FuenteDeInformacion>
</ColFuentes>
```

En el ejemplo I tenemos una colección de tres fuentes de información; cada uno de estos orígenes tiene asociada una fuente de información de distinto tipo, ahí se pueden ver los distintos parámetros utilizados para acceder a cada fuente.

3.2 COLENTIDADES

El nodo ColEntidades tendrá una colección de nodos Entidad, donde cada uno de estos últimos tendrá los atributos Nombre y tipo. El tipo indicará la forma de obtener los datos asociados a la entidad; siendo sus posibles valores:

- EntidadRelacional, para indicar que los datos se calculan a partir de los datos de otras entidades,
- EntidadServida, para indicar que los datos van a ser obtenidos de algún servicio ofrecido por las fuentes de información, o,

- EntidadReporte, para indicar que los datos los obtiene de la relacion con otras entidades pero que usa al menos una variable en el nodo ColExpresionesRelaciones.

Por ejemplo:

```
<Entidad IdEntidad="6" Nombre="entradaVendida" Tipo="EntidadServida">
<Entidad IdEntidad="5" Nombre="UnTitular" Tipo="EntidadRelacional">
<Entidad IdEntidad="3" Nombre="MovimientosDeTitular"
Tipo="EntidadReporte">
```

Dentro de cada nodo Entidad tenemos los nodos: ColRelaciones y FormatoReporte; además se agregarán los nodos ColExpresiónRelaciones y ColVariables para las entidades relacionales y de reporte; y ColServicios para las entidades servidas.

3.2.1 ColAtributos

El nodo ColAtributos contiene un conjunto de nodos Atributos los cuales tiene una serie de atributos que son:

- Nombre,
- Tipo, que corresponde al tipo de dato del atributo,
- Oculto, que determina si el atributo se muestra o no en los reportes,
- Horizontal, que permite colocar los datos asociados con el Atributo como títulos de columna del reporte,
- Clave, que determina que los datos asociados a este atributo no tendrán repetición.

Ejemplo:

```
<ColAtributos Autogenerado="33">
  <Atributo IdAtributo="32" Tipo="string" Nombre="funcion"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
  <Atributo IdAtributo="31" Tipo="int" Nombre="precio"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
  <Atributo IdAtributo="30" Tipo="long" Nombre="funcionProgramada"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
  <Atributo IdAtributo="29" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
</ColAtributos>
```

3.2.2 ColRelaciones

Los nodos ColRelaciones están formados por un conjunto de nodosRelacion, los cuales tienen como atributos:

- `Nombre`, el cual identifica la relación dentro de la entidad, por lo que debe ser único dentro de la colección de relación de esta entidad,
- `IdRelacion`, que es un número que identifica a la relación en todo el esquema.
- `IdEntidadForanea`, el identificador de la entidad foránea con la cual se define la relación,
- `IdAtributoForaneo`, con el cual se define la relación el id del atributo local con el cual se relacionan,
- `Tipo` de relación, el cual puede tener los valores: generalización, especialización, composicion o relacion,
- `Cardinalidad` con la cual se relacionan las entidades, siendo los posibles valores 1aN, Na1 NaN y 1a1 donde el primer valor el correspondiente a la entidad local y el segundo el de la entidad foránea.

Cabe destacar que si una entidad esta invocada en un cálculo entonces debe de aparecer en una relación.

Ejemplo:

```
<ColRelaciones Autogenerado="4">
```

```
<!—Entidades con las cuales se relaciona la entidad actual-->
        <Relacion IdRelacion="3" Nombre="RelacionMovimientos"
IdEntidadForanea="1" IdAtributoLocal="15" IdAtributoForaneo="3"
Cardinalidad="1aN" Tipo="relacion" />
        <Relacion IdRelacion="2" Nombre="RelacionTitulares"
IdEntidadForanea="0" IdAtributoLocal="12" IdAtributoForaneo="0"
Cardinalidad="1aN" Tipo="relacion" />
      </ColRelaciones>
```

3.2.3 FormatoReporte

El nodo FormatoReporte permite mantener información correspondiente a como será presentada la información de la entidad cuando se desee generar un reporte con su definición. Es así que este nodo contiene dos nodos que son FormatoHojaReporte y FormatoCuerpoReporte.

3.2.3.1 FormatoHojaReporte

El nodo **FormatoHojaReporte** permite mantener el formato de la hoja donde se va a imprimir el reporte generado con los datos del nodo Entidad que contiene a este nodo. Entre los nodos que se definieron para mantener la configuración de la hoja tenemos los nodos:

- PageHeader que en el encabezado de página permite configurar: alto, si se coloca un texto y que texto se coloca y si se coloca una imagen para el logo y guarda la imagen en formato Base64 en el encabezado
- PageFooter que en el pie de página permite configurar: el alto, si se muestran textos con el número de página, total de páginas y fecha de generado el reporte.
- PageWidth que determina el ancho de la hoja
- PageHeight que determina el alto de la hoja
- LeftMargin que determina el margen izquierdo de la hoja
- RightMargin que determina el margen derecho de la hoja
- TopMargin que determina el margen superior de la hoja
- BottomMargin que determina el margen inferior de la hoja

3.2.3.2 FormatoCuerpoReporte

En el nodo **FormatoCuerpoReporte** se encuentra: en el nodo Plantilla con el nombre de la plantilla relacionada con el reporte, la cual permite indicar el color de relleno de las diferentes áreas del reporte y el formato de las letras; y el nodo ColColumna el cual contiene una colección de nodos AnchoColumna donde cada uno tiene nombre del Atributo y el Ancho que tiene su columna; en caso de no aparecer el nombre de un Atributo o tener Ancho menor a 0 entonces toma el ancho por defercto.

Ejemplo del nodo FormatoReporte:

```
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>5</Height>
              <ReportItems>
                <Image Name="image1">
                    <Value>
Qk1y/wAAAAAAADYAAAAoAAAAgwAAAKUAAAABABgAAAAAAAAAAAAAAjLgAAIy4AAAAAAAAAAAA
AAJTi/
Iza9HC+4Gy63JTvDMkjQLUTPITjDFCm0IDXAMkTRM0TRKznHMT3LOkTSMT7KFyq1HjS8NU
rVL0TPLkLQGS27KzzNNkfh0gnl1g3p2g3p2gnl1gHdzfnVxfnVxf3Zyf3Zyf3Zyf3Zyf3Z
yf3dwfnZvgntyg351gXxzfHdue3Ztf3pxg351fHpwe3lvenhuenhue3lvfHpwfXxyf31zf
3lygXlygXlygXlygXlygXlygXlygXlygnpzgXlygXlygHhxgHhxgXlygXlygntyfXhvfnp
vgHxxgHxxgHxxgHxxfnpvfXlugn5zgn5zgX1ygHxxf3twf3twfnpvfXluhHpzhHpzg3lyf
WdX
```

```
                    </Value>
                  </Image>
                  <TextoEncabezado>Magma VRS</TextoEncabezado>
                </ReportItems>
              </PageHeader>
              <PageFooter>
                <Height>6</Height>
                <ReportItems>
                  <TextoNumeroDePagina>True</TextoNumeroDePagina>
                  <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
                </ReportItems>
              </PageFooter>
              <PageHeight>30</PageHeight>
              <PageWidth>20</PageWidth>
              <LeftMargin>1</LeftMargin>
              <RightMargin>3</RightMargin>
              <TopMargin>2</TopMargin>
              <BottomMargin>4</BottomMargin>
              <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
            </FormatoHojaReporte>
            <FormatoCuerpoReporte>
              <Plantilla Nombre="CuerpoTabla2" />
              <ColColumnas>
                <AnchoColumna Nombre="proximoEstreno">
                  <Width>5</Width>
                </AnchoColumna>
                <AnchoColumna Nombre="nombreCartelera">
                  <Width>5</Width>
                </AnchoColumna>
              </ColColumnas>
            </FormatoCuerpoReporte>
          </DisenioReporte>
        </FormatoReporte>
```

3.2.4 ColExpresionesRelaciones

Si un nodo Entidad contiene un nodo ColExpresionesRelaciones entonces la entidad pasa a ser del tipo EntidadRelacional, cada elemento de la colección será un nodo ExpresionRelacion los cuales tendrán el atributo nombre el cual es único para la colección de expresiones actual, además puede contener cuatro nodos más, el primer nodo permite determinar la condición que deben cumplir cada dato para pertenecer a la Entidad este nodo se llama CondicionIndividual, el segundo nodo puede indicar los Atributos de la Entidad por lo cual se va a Agrupar y el tercero permite indicar porque campos se va a Ordenar.

3.2.4.1 ColVariables

El nodo ColVariable contiene un conjunto de nodos Variable, cada uno de estos nodos guarda la definición de una variable que luego será invocada en cálculos o condiciones.

Cada uno de estos nodos contiene los atributos Tipo que determinan el tipo de dato de la variable, Nombre que corresponde al nombre de la variable sin el @ y Valor para determinar si tiene algún valor relacionado aunque este dato solo será agregado cuando se vaya a generar el reporte.

Por Ejemplo:

```
<ColVariables Autogenerado="4">
  <Variable IdVariable="2" Tipo="datetime" Nombre="FechaMinima"
Valor="" />
  <Variable IdVariable="3" Tipo="decimal"
Nombre="PorcentajeImpuestoMov" Valor="" />
  <Variable IdVariable="1" Tipo="long" Nombre="NroTitular" Valor="" />
```

```
</ColVariables>
```

3.2.4.2 CondicionIndividual

El nodo CondicionIndividual, en el atributo Valor, permite mantener una expresión booleana , la cual puede estar formada por funciones con o sin parámetros, variables y/o constantes; las cuales pueden estar vinculadas por algunos operadores lógicos o relacionales.

Por ejemplo:

```
<CondicionIndividual Valor="(RelacionMovimientos.cod_tita=@NroTitular)
and (RelacionTitulares.cod_tit=RelacionMovimientos.cod_tita) and
(RelacionMovimientos.fec_doc&gt;@FechaMinima)" />
```

3.2.4.3 Agrupamientos

El nodo Agrupar permite mantener una colección con id de atributos por los que se va a agrupar los Atributos por los cuales se va a agrupar.

Por ejemplo:

```
<Agrupar>
  <Atributo IdAtributo="45" />
  <Atributo IdAtributo="46" />
</Agrupar>
```

3.2.4.4 Ordenamientos

El nodo Ordenar permite indicar el ID del Atributo por el cual se va a ordenar y el tipo de ordenamiento que puede ser ASC o DESC, si se tienen muchos campos para ordenar se toma en cuenta el orden que esta definido, primero ordena por el Atributo que esta nombrado primero.

Por ejemplo:

```
<Ordenar>
  <Atributo IdAtributo="59" TipoOrden="ASC" />
  <Atributo IdAtributo="60" TipoOrden="DESC" />
</Ordenar>
```

3.2.4.5 ColVinculos

En este nodo se selecciona un conjunto de vínculos, cada vínculo determina como se calculan los datos relacionados con cada Atributo de la Entidad actual.

Cada nodo Vinculo contendrá un atributo con el identificador del Atributo al cual se le va a vincular con un calculo y un nodo Calculo.

En cada Entidad, se podrá tener por cada Atributo un Vinculo para cada ExpresiónRelacion.

3.2.4.5.1 Calculo

Este nodo determina la operación a realizar para obtener los datos vinculados con el Atributo. En este cálculo podrán participar constantes, variables, o funciones, operadores, o atributos pertenecientes a una entidad que está relacionada con la entidad actual. Si este fuera el caso, debe aparecer el nombre de la relación (no el de la entidad) y el nombre del Atributo separados por un punto.

Las siguiente es una lista de las funciones y operadores reconocidos:

- Operaciones aritméticas
  `+, -, *, /.`
- Operadores relacionales
  `=, !=, <, >, <=, >=, not()`
- Operadores lógicos
  `&&, ||, not()`
- Funciones de cadena

```
+ (concatena), length(string), substring(string ,
ini, fin), string(Numero)
```

- Funciones agregadas

```
count(), min(), max(), avg(), sum()
```

- Uso general

```
iff(condición,resultado_si_verdadero,
resultado_si_verdadero)
```

Ejemplo:

```
<ColVinculos Autogenerado="13">
  <Vinculo IdVinculo="12" IdAtributo="19">
    <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.MovimientoConSigno -
RelacionMovimientos.Movimiento * (@PorcentajeImpuestoMov / 100)" />
  </Vinculo>
    <Vinculo IdVinculo="11" IdAtributo="17">
      <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.imp_mov_mn * @PorcentajeImpuestoMov / 100"
/>
  </Vinculo>
    <Vinculo IdVinculo="10" IdAtributo="20">
      <Calculo Tipo="CalculoServicio"
Valor="substring(iif(RelacionMovimientos.MovimientoConSigno&gt;=0,'Dep
osito','Retiro'),1,3)" />
  </Vinculo>
    <Vinculo IdVinculo="9" IdAtributo="18">
      <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.MovimientoConSigno" />
  </Vinculo>
    <Vinculo IdVinculo="8" IdAtributo="16">
      <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.fec_doc" />
  </Vinculo>
    <Vinculo IdVinculo="7" IdAtributo="15">
      <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.nro_trans" />
  </Vinculo>
    <Vinculo IdVinculo="6" IdAtributo="14">
      <Calculo Tipo="CalculoServicio"
Valor="substring(RelacionTitulares.razon_social,len(RelacionTitulares.
razon_social),2)" />
  </Vinculo>
  <Vinculo IdVinculo="5" IdAtributo="13">
    <Calculo Tipo="CalculoServicio" Valor="RelacionTitulares.nom_tit"
/>
  </Vinculo>
  <Vinculo IdVinculo="4" IdAtributo="12">
    <Calculo Tipo="CalculoServicio" Valor="RelacionTitulares.cod_tit"
/>
          </Vinculo>
</ColVinculos>
```

3.2.5 ColServicios

Si existe el nodo ColServicios en una entidad entonces esta entidad pasa a ser una entidad servida (`Tipo="EntidadServida"`).

El nodo ColServicios permite mantener un conjunto de nodos Servicio, donde cada nodo Servicio determina que servicio de que fuente de información se debe ejecutar para obtener los datos relacionados con la entidad. Al existir varios servicios los datos de la entidad provendrán de varias fuentes de información.

Cada nodo Servicio tendrá los atributos: Nombre el cual es único para cada colección de servicios, el identificador del nodo FuenteInformacion que ofrece dicho servicio. Además cada nodo Servicio, contiene dos nodos más, que son: **ColParametros** que corresponde a la información necesarias para la ejecución del servicios en la fuentes de información y **ColVinculos** que determina que cálculos se deben realizar para obtener los datos relacionados con los Atributos de la Entidad, en caso que no exista un vinculo para un Atributo de la entidad se le vincula con campo resultado de aplicar el servicio indicado de la fuente de información.

Por ejemplo, en este caso tenemos una entidad con dos servicios en su colección de servicios. Además, al servicio `"ObtenerPeliculas"`, que se ofrece en la fuente de información con `IdFuente="5"`, se le deben pasar los parámetros:

- pProximoEstreno,
- pNombre,
- pId,
- pDuracionMinima.

```
<ColServicios Autogenerado="19">
  <Servicio IdServicio="12" Nombre="ObtenerPeliculasGenero"
dFuente="6">
    <ColParametros Autogenerado="13">
      <Parametro IdParametro="12" Nombre="pGenero" Valor="xxx" />
    </ColParametros>
    <ColVinculos Autogenerado="44">
      <Vinculo IdVinculo="43" IdAtributo="67">
        <Calculo Tipo="CalculoServicio" Valor="duracion +
tiempoTrailers" />
      </Vinculo>
      <Vinculo IdVinculo="42" IdAtributo="63">
        <Calculo Tipo="CalculoServicio" Valor="id" />
      </Vinculo>
      <Vinculo IdVinculo="41" IdAtributo="65">
        <Calculo Tipo="CalculoServicio"
Valor="substring(nombreCartelera,1,10)" />
      </Vinculo>
    </ColVinculos>
  </Servicio>
  <Servicio IdServicio="11" Nombre="ObtenerPeliculas" IdFuente="5">
    <ColParametros Autogenerado="13">
      <Parametro IdParametro="10" Nombre="pFechaEstreno"
Valor="#09/12/2000#" />
      <Parametro IdParametro="9" Nombre="pProximoEstreno" Valor="True"
/>
      <Parametro IdParametro="8" Nombre="pNombre" Valor="a" />
      <Parametro IdParametro="7" Nombre="pId" Valor="-1" />
      <Parametro IdParametro="11" Nombre="pDuracionMinima" Valor="1"
/>
    </ColParametros>
    <ColVinculos Autogenerado="44">
      <Vinculo IdVinculo="40" IdAtributo="67">
        <Calculo Tipo="CalculoServicio" Valor="duracion + tiempoCorto
+ tiempoTrailers" />
      </Vinculo>
      <Vinculo IdVinculo="39" IdAtributo="63">
        <Calculo Tipo="CalculoServicio" Valor="id" />
      </Vinculo>
      <Vinculo IdVinculo="38" IdAtributo="65">
        <Calculo Tipo="CalculoServicio"
Valor="substring(nombreCartelera,1,10)" />
      </Vinculo>
    </ColVinculos>
  </Servicio>
</ColServicios>
```

En este caso si existiera el atributo Fecha_Movimiento el la entidad actual, entonces debe de existir el campo Fecha_Movimiento como resultado de aplicar el servicios ObetenerServicio en el nodo FuenteInformación con Id "0".

4 EJEMPLOS COMPLETOS

Presentaremos aquí 4 ejemplos de esquemas que se enfocan en distintas capacidades que brinda el Meta-modelo de negocio. Los comandos con los que generan los esquemas que aquí se presentan pueden ser encontrados en el documento "5 - Lenguaje de Edición de Meta-Modelo".

El *Ejemplo I* (Entidad Servida Con Fuentes Heterogéneas) muestra una forma de integrar los datos heterogéneos provenientes de distintas fuentes de información.

El *Ejemplo II* (Venta de Entradas de Cine) introduce a la definición de entidades relacionales.

El *Ejemplo III* (Estado de Cuenta) muestra el uso de las variables para obtener reportes para los cuales los valores varían de una ejecución a otra.

El *Ejempo IV* (Productora por Género) permite ver el uso de los atributos horizontales y el formato matriz para obtener reportes en los cuales los atributos de las entidades aparecen como columnas del reporte.

4.1 EJEMPLO I: UNA ENTIDAD SERVIDA CON FUENTES HETEROGÉNEAS

```
<Esquema Nombre="c:\Ej1-titularesMagma">
  <ColFuentes Autogenerado="3">
    <FuenteDeInformacion IdFuente="2" Nombre="titulares_remoting"
Tipo="remoting">
      <Parametro
Nombre="proxy">RemotingProxy.RemotingProxy</Parametro>
      <Parametro Nombre="interface">Comun.IAccesoADAtos</Parametro>
      <Parametro Nombre="path">D:\ProyectoDeGradoFI\
WebServicesEjemplo\RemotingProxy\bin\Debug</Parametro>
      <Parametro Nombre="rem-name">RemotingProxy.dll</Parametro>
      <Parametro Nombre="metodoacceso">ObtenerObjeto</Parametro>
    </FuenteDeInformacion>
    <FuenteDeInformacion IdFuente="1" Nombre="titulares_ws"
Tipo="webservice">
      <Parametro
Nombre="url">http://localhost/WebServiceCines/WSCines.asmx</Parametro>
    </FuenteDeInformacion>
    <FuenteDeInformacion IdFuente="0" Nombre="titulares_dll"
Tipo="dllfile">
      <Parametro Nombre="dll-name">CapaDatos.dll</Parametro>
      <Parametro Nombre="clase">CapaDatos.AccesoADatos</Parametro>
      <Parametro Nombre="path">D:\ProyectoDeGradoFI\
WebServicesEjemplo\CapaDatos\bin\Debug</Parametro>
    </FuenteDeInformacion>
  </ColFuentes>
  <ColEntidades Autogenerado="1">
    <Entidad IdEntidad="0" Nombre="titulares" Tipo="EntidadServida">
      <ColServicios Autogenerado="3">
        <Servicio IdServicio="2" Nombre="ObtenerClientesPorNombre"
IdFuente="1">
          <ColParametros Autogenerado="1">
            <Parametro IdParametro="0" Nombre="pNombre" Valor="Ana" />
          </ColParametros>
          <ColVinculos Autogenerado="9">
            <Vinculo IdVinculo="8" IdAtributo="3">
              <Calculo Tipo="CalculoServicio" Valor="'----'" />
            </Vinculo>
            <Vinculo IdVinculo="7" IdAtributo="2">
              <Calculo Tipo="CalculoServicio" Valor="Nombre" />
            </Vinculo>
            <Vinculo IdVinculo="6" IdAtributo="1">
```

```
              <Calculo Tipo="CalculoServicio"
Valor="convert(Cedula_De_Identidad,'System.Int64')" />
            </Vinculo>
            <Vinculo IdVinculo="5" IdAtributo="0">
              <Calculo Tipo="CalculoServicio"
Valor="'ObtenerClientesPorNombre' + Cedula_De_Identidad" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
        <Servicio IdServicio="1" Nombre="ObtenerClientes"
IdFuente="0">
          <ColParametros Autogenerado="1" />
          <ColVinculos Autogenerado="9">
            <Vinculo IdVinculo="4" IdAtributo="3">
              <Calculo Tipo="CalculoServicio" Valor="'----'" />
            </Vinculo>
            <Vinculo IdVinculo="3" IdAtributo="2">
              <Calculo Tipo="CalculoServicio" Valor="Nombre" />
            </Vinculo>
            <Vinculo IdVinculo="2" IdAtributo="1">
              <Calculo Tipo="CalculoServicio"
Valor="convert(Cedula_De_Identidad,'System.Int64')" />
            </Vinculo>
            <Vinculo IdVinculo="1" IdAtributo="0">
              <Calculo Tipo="CalculoServicio" Valor="'ObtenerClientes'
+ Cedula_De_Identidad" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
        <Servicio IdServicio="0" Nombre="ObtenerTitulares"
IdFuente="2">
          <ColParametros Autogenerado="1" />
          <ColVinculos Autogenerado="9">
            <Vinculo IdVinculo="0" IdAtributo="0">
              <Calculo Tipo="CalculoServicio"
Valor="'ObtenerTitulares' + convert(cod_tit,'System.String')" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="4">
        <Atributo IdAtributo="3" Tipo="string" Nombre="razon_social"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="2" Tipo="string" Nombre="nom_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="1" Tipo="long" Nombre="cod_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="0" Tipo="string" Nombre="id"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="0" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
```

```
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
  </ColEntidades>
</Esquema>
```

4.2 EJEMPLO II: VENTA DE ENTRADAS DE CINE

```
<Esquema Nombre="c:\Ej2-Cines">
  <ColFuentes Autogenerado="2">
    <FuenteDeInformacion IdFuente="1" Nombre="cines_dll"
Tipo="dllfile">
      <Parametro Nombre="dll-name">CapaDatos.dll</Parametro>
      <Parametro Nombre="clase">CapaDatos.AccesoADatos</Parametro>
      <Parametro Nombre="path">d:\ProyectoDeGradoFI\
WebServicesEjemplo\CapaDatos\bin\Debug</Parametro>
    </FuenteDeInformacion>
    <FuenteDeInformacion IdFuente="0" Nombre="cines_ws"
Tipo="webservice">
      <Parametro
Nombre="url">http://localhost/WebServiceCines/WSCines.asmx</Parametro>
    </FuenteDeInformacion>
  </ColFuentes>
  <ColEntidades Autogenerado="9">
    <Entidad IdEntidad="8" Nombre="recaudacionXsalaXfecha"
Tipo="EntidadRelacional">
      <ColExpresionesRelaciones Autogenerado="6">
        <ExpresionRelacion IdExpresionRelacion="1"
Nombre="condidcion">
```

```
          <CondicionIndividual
Valor="(RELfuncionProgramada.fecha&gt;='05/03/2005') and
(RELfuncionProgramada.fecha&lt;='12/09/2006') and
(RELfuncionProgramada.id=RELentradaVendida.funcion_Programada_id)" />
          <Ordenar />
          <Agrupar>
            <Atributo IdAtributo="49" />
            <Atributo IdAtributo="50" />
            <CondicionGrupal Valor="" />
          </Agrupar>
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="21" IdAtributo="48">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.sala_id" />
            </Vinculo>
            <Vinculo IdVinculo="27" IdAtributo="54">
              <Calculo Tipo="CalculoServicio"
Valor="min(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="26" IdAtributo="53">
              <Calculo Tipo="CalculoServicio"
Valor="max(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="25" IdAtributo="52">
              <Calculo Tipo="CalculoServicio"
Valor="sum(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="24" IdAtributo="51">
              <Calculo Tipo="CalculoServicio"
Valor="count(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="23" IdAtributo="50">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.fecha" />
            </Vinculo>
            <Vinculo IdVinculo="22" IdAtributo="49">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.sala" />
            </Vinculo>
          </ColVinculos>
        </ExpresionRelacion>
      </ColExpresionesRelaciones>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="56" Tipo="long"
Nombre="funcion_Programada_id_Entrada" Multiplicidad="False"
Oculto="True" Horizontal="False" Clave="False" />
        <Atributo IdAtributo="55" Tipo="long"
Nombre="funcion_Programada_id" Multiplicidad="False" Oculto="True"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="54" Tipo="int" Nombre="prcioMinimo"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="53" Tipo="int" Nombre="precioMaximo"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="52" Tipo="long"
Nombre="cantidadRecaudada" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="51" Tipo="long"
Nombre="cantidadEntradas" Multiplicidad="False" Oculto="True"
Horizontal="False" Clave="False" />
```

```
        <Atributo IdAtributo="50" Tipo="date" Nombre="fecha"
Multiplicidad="False" Oculto="False" Horizontal="True" Clave="False"
/>
        <Atributo IdAtributo="49" Tipo="string" Nombre="sala"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="48" Tipo="long" Nombre="sala_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="10">
        <Relacion IdRelacion="9" Nombre="RELentradaVendida"
IdEntidadForanea="6" IdAtributoLocal="56" IdAtributoForaneo="35"
Cardinalidad="na1" Tipo="especializacion" />
        <Relacion IdRelacion="8" Nombre="RELfuncionProgramada"
IdEntidadForanea="5" IdAtributoLocal="55" IdAtributoForaneo="23"
Cardinalidad="na1" Tipo="especializacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="7" Nombre="EntradasPorPeliculaPorFecha"
Tipo="EntidadRelacional">
```

```
      <ColExpresionesRelaciones Autogenerado="6">
        <ExpresionRelacion IdExpresionRelacion="0"
Nombre="condidcion">
          <CondicionIndividual
Valor="(RELfuncionProgramada.fecha&gt;='05/03/2005') and
(RELfuncionProgramada.fecha&lt;='12/09/2006') and
(RELfuncionProgramada.id=RELentradaVendida.funcion_Programada_id)" />
          <Ordenar />
          <Agrupar>
            <Atributo IdAtributo="40" />
            <Atributo IdAtributo="41" />
            <CondicionGrupal Valor="" />
          </Agrupar>
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="20" IdAtributo="45">
              <Calculo Tipo="CalculoServicio"
Valor="min(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="19" IdAtributo="44">
              <Calculo Tipo="CalculoServicio"
Valor="max(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="18" IdAtributo="43">
              <Calculo Tipo="CalculoServicio"
Valor="sum(RELentradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="17" IdAtributo="42">
              <Calculo Tipo="CalculoServicio"
Valor="count(entradaVendida.precio)" />
            </Vinculo>
            <Vinculo IdVinculo="16" IdAtributo="41">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.fecha" />
            </Vinculo>
            <Vinculo IdVinculo="15" IdAtributo="40">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.pelicula" />
            </Vinculo>
            <Vinculo IdVinculo="14" IdAtributo="39">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.pelicula_id" />
            </Vinculo>
          </ColVinculos>
        </ExpresionRelacion>
      </ColExpresionesRelaciones>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="45" Tipo="int" Nombre="prcioMinimo"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="44" Tipo="int" Nombre="precioMaximo"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="43" Tipo="long"
Nombre="cantidadRecaudada" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="42" Tipo="int" Nombre="cantidadEntradas"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="41" Tipo="date" Nombre="fecha"
Multiplicidad="False" Oculto="False" Horizontal="True" Clave="False"
/>
```

```
        <Atributo IdAtributo="40" Tipo="string" Nombre="pelicula"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="39" Tipo="long" Nombre="pelicula_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="47" Tipo="long"
Nombre="funcion_Programada_id_Entrada" Multiplicidad="False"
Oculto="True" Horizontal="False" Clave="False" />
        <Atributo IdAtributo="46" Tipo="long"
Nombre="funcion_Programada_id" Multiplicidad="False" Oculto="True"
Horizontal="False" Clave="False" />
      </ColAtributos>
      <ColRelaciones Autogenerado="10">
        <Relacion IdRelacion="7" Nombre="RELentradaVendida"
IdEntidadForanea="6" IdAtributoLocal="47" IdAtributoForaneo="35"
Cardinalidad="na1" Tipo="especializacion" />
        <Relacion IdRelacion="6" Nombre="RELfuncionProgramada"
IdEntidadForanea="5" IdAtributoLocal="46" IdAtributoForaneo="23"
Cardinalidad="na1" Tipo="especializacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
```

```
    </Entidad>
    <Entidad IdEntidad="6" Nombre="entradaVendida"
Tipo="EntidadServida">
      <ColServicios Autogenerado="7">
        <Servicio IdServicio="6" Nombre="ObtenerEntradasVendidas"
IdFuente="0">
          <ColParametros Autogenerado="6" />
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="13" IdAtributo="38">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.pelicula" />
            </Vinculo>
            <Vinculo IdVinculo="12" IdAtributo="37">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.funcion" />
            </Vinculo>
            <Vinculo IdVinculo="11" IdAtributo="36">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncionProgramada.sala" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="38" Tipo="string" Nombre="pelicula"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="37" Tipo="string" Nombre="funcion"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="36" Tipo="string" Nombre="sala"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="35" Tipo="long"
Nombre="funcion_Programada_id" Multiplicidad="False" Oculto="True"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="34" Tipo="int" Nombre="precio"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="33" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
      </ColAtributos>
      <ColRelaciones Autogenerado="10">
        <Relacion IdRelacion="5" Nombre="RELfuncionProgramada"
IdEntidadForanea="5" IdAtributoLocal="35" IdAtributoForaneo="23"
Cardinalidad="na1" Tipo="especializacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
```

```
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="5" Nombre="funcionProgramada"
Tipo="EntidadServida">
      <ColServicios Autogenerado="7">
        <Servicio IdServicio="5" Nombre="ObtenerFuncionesProgramadas"
IdFuente="0">
          <ColParametros Autogenerado="6" />
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="10" IdAtributo="32">
              <Calculo Tipo="CalculoServicio"
Valor="RELpelicula.nombreCartelera" />
            </Vinculo>
            <Vinculo IdVinculo="9" IdAtributo="31">
              <Calculo Tipo="CalculoServicio"
Valor="RELfuncion.nombre" />
            </Vinculo>
            <Vinculo IdVinculo="8" IdAtributo="30">
              <Calculo Tipo="CalculoServicio" Valor="RELsala.nombre"
/>
            </Vinculo>
            <Vinculo IdVinculo="7" IdAtributo="25">
              <Calculo Tipo="CalculoServicio" Valor="hora_comienzo" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="32" Tipo="string" Nombre="pelicula"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="31" Tipo="string" Nombre="funcion"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="30" Tipo="string" Nombre="sala"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
```

```
        <Atributo IdAtributo="29" Tipo="long" Nombre="pelicula_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="28" Tipo="long" Nombre="funcion_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="27" Tipo="long" Nombre="sala_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
        <Atributo IdAtributo="26" Tipo="long" Nombre="programacion_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="25" Tipo="time" Nombre="hora"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
        <Atributo IdAtributo="24" Tipo="date" Nombre="fecha"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
        <Atributo IdAtributo="23" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
      </ColAtributos>
      <ColRelaciones Autogenerado="10">
        <Relacion IdRelacion="4" Nombre="RELpelicula"
IdEntidadForanea="0" IdAtributoLocal="29" IdAtributoForaneo="1"
Cardinalidad="na1" Tipo="relacion" />
        <Relacion IdRelacion="3" Nombre="RELfuncion"
IdEntidadForanea="3" IdAtributoLocal="28" IdAtributoForaneo="16"
Cardinalidad="na1" Tipo="relacion" />
        <Relacion IdRelacion="2" Nombre="RELsala" IdEntidadForanea="2"
IdAtributoLocal="27" IdAtributoForaneo="12" Cardinalidad="na1"
Tipo="relacion" />
        <Relacion IdRelacion="1" Nombre="RELprogramacion"
IdEntidadForanea="4" IdAtributoLocal="26" IdAtributoForaneo="18"
Cardinalidad="na1" Tipo="relacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
```

```
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="4" Nombre="programacion"
Tipo="EntidadServida">
      <ColServicios Autogenerado="7">
        <Servicio IdServicio="4" Nombre="ObtenerProgramaciones"
IdFuente="0">
          <ColParametros Autogenerado="6" />
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="6" IdAtributo="21">
              <Calculo Tipo="CalculoServicio"
Valor="RELcomplejo.nombre" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="21" Tipo="string" Nombre="complejo"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="20" Tipo="date" Nombre="fecha_fin"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="19" Tipo="date" Nombre="fecha_inicio"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="18" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
        <Atributo IdAtributo="22" Tipo="long" Nombre="complejo_id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="10">
        <Relacion IdRelacion="0" Nombre="RELcomplejo"
IdEntidadForanea="1" IdAtributoLocal="22" IdAtributoForaneo="8"
Cardinalidad="na1" Tipo="relacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
```

```
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="3" Nombre="funcion" Tipo="EntidadServida">
      <ColServicios Autogenerado="7">
        <Servicio IdServicio="3" Nombre="ObtenerFunciones"
IdFuente="1">
          <ColParametros Autogenerado="6" />
          <ColVinculos Autogenerado="28" />
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="17" Tipo="string" Nombre="nombre"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
        <Atributo IdAtributo="16" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
      </ColAtributos>
      <ColRelaciones Autogenerado="10" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
```

```
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="2" Nombre="sala" Tipo="Entidad">
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="15" Tipo="string" Nombre="complejo"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="14" Tipo="long" Nombre="idcomplejo"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="13" Tipo="string" Nombre="nombre"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="12" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
      </ColAtributos>
      <ColRelaciones Autogenerado="10" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
```

```
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="1" Nombre="complejo" Tipo="EntidadServida">
      <ColServicios Autogenerado="7">
        <Servicio IdServicio="2" Nombre="ObtenerComplejos"
IdFuente="0">
          <ColParametros Autogenerado="6" />
          <ColVinculos Autogenerado="28" />
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="10" Tipo="string" Nombre="ciudad"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="9" Tipo="string" Nombre="nombre"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
        <Atributo IdAtributo="8" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
        <Atributo IdAtributo="11" Tipo="string" Nombre="direccion"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="10" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>
```

```
<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="0" Nombre="pelicula" Tipo="EntidadServida">
      <ColServicios Autogenerado="7">
        <Servicio IdServicio="1" Nombre="ObtenerPeliculasGenero"
IdFuente="1">
          <ColParametros Autogenerado="6">
            <Parametro IdParametro="5" Nombre="pGenero" Valor="xxx" />
          </ColParametros>
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="5" IdAtributo="5">
              <Calculo Tipo="CalculoServicio" Valor="duracion +
tiempoTrailers" />
            </Vinculo>
            <Vinculo IdVinculo="4" IdAtributo="1">
              <Calculo Tipo="CalculoServicio" Valor="id" />
            </Vinculo>
            <Vinculo IdVinculo="3" IdAtributo="3">
              <Calculo Tipo="CalculoServicio"
Valor="substring(nombreCartelera,1,10)" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
        <Servicio IdServicio="0" Nombre="ObtenerPeliculas"
IdFuente="0">
          <ColParametros Autogenerado="6">
            <Parametro IdParametro="4" Nombre="pDuracionMinima"
Valor="1" />
            <Parametro IdParametro="3" Nombre="pFechaEstreno"
Valor="#09/12/2000#" />
            <Parametro IdParametro="2" Nombre="pProximoEstreno"
Valor="True" />
            <Parametro IdParametro="1" Nombre="pNombre" Valor="a" />
            <Parametro IdParametro="0" Nombre="pId" Valor="-1" />
          </ColParametros>
          <ColVinculos Autogenerado="28">
            <Vinculo IdVinculo="2" IdAtributo="5">
              <Calculo Tipo="CalculoServicio" Valor="duracion +
tiempoCorto + tiempoTrailers" />
            </Vinculo>
            <Vinculo IdVinculo="1" IdAtributo="1">
              <Calculo Tipo="CalculoServicio" Valor="id" />
```

```
            </Vinculo>
            <Vinculo IdVinculo="0" IdAtributo="3">
              <Calculo Tipo="CalculoServicio"
Valor="substring(nombreCartelera,1,10)" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="57">
        <Atributo IdAtributo="7" Tipo="boolean"
Nombre="proximoEstreno" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="6" Tipo="date" Nombre="fechaEstreno"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="5" Tipo="int" Nombre="duracionTotal"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="4" Tipo="int" Nombre="duracion"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="3" Tipo="string" Nombre="nombreEntrada"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="2" Tipo="string"
Nombre="nombreCartelera" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="1" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
        <Atributo IdAtributo="0" Tipo="string" Nombre="nombreOriginal"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="10" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
```

```
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
  </ColEntidades>
</Esquema>
```

4.3 EJEMPLO III: ESTADO DE CUENTA (CON VARIABLES)

```
<Esquema Nombre="c:\Ej3-MovimientosDeUnCliente">
  <ColFuentes Autogenerado="1">
    <FuenteDeInformacion IdFuente="0" Nombre="Magma"
Tipo="webservice">
      <Parametro
Nombre="url">http://localhost/WSMagma/MagmaServicioWeb.asmx</Parametro
>
    </FuenteDeInformacion>
  </ColFuentes>
  <ColEntidades Autogenerado="4">
    <Entidad IdEntidad="3" Nombre="MovimientosDeTitular"
Tipo="EntidadReporte">
      <ColExpresionesRelaciones Autogenerado="0">
        <ExpresionRelacion IdExpresionRelacion="1"
Nombre="ExpresionMovimientoDeTitular">
          <CondicionIndividual
Valor="(RelacionTitulares.cod_tit=@NroTitular) and
(RelacionTitulares.cod_tit=RelacionMovimientos.cod_tita)" />
          <Ordenar />
          <Agrupar>
            <CondicionGrupal Valor="" />
          </Agrupar>
          <ColVinculos Autogenerado="13">
            <Vinculo IdVinculo="12" IdAtributo="19">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.MovimientoConSigno -
RelacionMovimientos.Movimiento * (@PorcentajeImpuestoMov / 100)" />
            </Vinculo>
            <Vinculo IdVinculo="11" IdAtributo="17">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.imp_mov_mn * @PorcentajeImpuestoMov / 100"
/>
            </Vinculo>
            <Vinculo IdVinculo="10" IdAtributo="20">
              <Calculo Tipo="CalculoServicio"
Valor="substring(iif(RelacionMovimientos.MovimientoConSigno&gt;=0,'Dep
osito','Retiro'),1,3)" />
            </Vinculo>
            <Vinculo IdVinculo="9" IdAtributo="18">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.MovimientoConSigno" />
            </Vinculo>
```

```
            <Vinculo IdVinculo="8" IdAtributo="16">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.fec_doc" />
            </Vinculo>
            <Vinculo IdVinculo="7" IdAtributo="15">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionMovimientos.nro_trans" />
            </Vinculo>
            <Vinculo IdVinculo="6" IdAtributo="14">
              <Calculo Tipo="CalculoServicio"
Valor="substring(RelacionTitulares.razon_social,len(RelacionTitulares.
razon_social),2)" />
            </Vinculo>
            <Vinculo IdVinculo="5" IdAtributo="13">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionTitulares.nom_tit" />
            </Vinculo>
            <Vinculo IdVinculo="4" IdAtributo="12">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionTitulares.cod_tit" />
            </Vinculo>
          </ColVinculos>
        </ExpresionRelacion>
      </ColExpresionesRelaciones>
      <ColVariables Autogenerado="4">
        <Variable IdVariable="3" Tipo="decimal"
Nombre="PorcentajeImpuestoMov" Valor="" />
        <Variable IdVariable="1" Tipo="long" Nombre="NroTitular"
Valor="" />
      </ColVariables>
      <ColAtributos Autogenerado="21">
        <Atributo IdAtributo="20" Tipo="string"
Nombre="TipoMovimiento" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="19" Tipo="decimal" Nombre="MontoFinal"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="18" Tipo="decimal"
Nombre="MovimientoConSigno" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="17" Tipo="decimal"
Nombre="MontoImpuestoAlMovimiento" Multiplicidad="False"
Oculto="False" Horizontal="False" Clave="False" />
        <Atributo IdAtributo="16" Tipo="datetime" Nombre="fec_doc"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="15" Tipo="long" Nombre="nro_trans"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="14" Tipo="string" Nombre="razon_social"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="13" Tipo="string" Nombre="nom_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="12" Tipo="long" Nombre="cod_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="4">
        <Relacion IdRelacion="3" Nombre="RelacionMovimientos"
IdEntidadForanea="1" IdAtributoLocal="15" IdAtributoForaneo="3"
Cardinalidad="1aN" Tipo="relacion" />
```

```
        <Relacion IdRelacion="2" Nombre="RelacionTitulares"
IdEntidadForanea="0" IdAtributoLocal="12" IdAtributoForaneo="0"
Cardinalidad="1aN" Tipo="relacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="2" Nombre="DatosUnTitular"
Tipo="EntidadReporte">
      <ColExpresionesRelaciones Autogenerado="0">
        <ExpresionRelacion IdExpresionRelacion="0"
Nombre="ExpresionDatosUnTitular">
          <CondicionIndividual
Valor="RelacionDatosUnTitular.cod_tit=@NroTitular" />
          <Ordenar />
          <Agrupar>
            <CondicionGrupal Valor="" />
          </Agrupar>
          <ColVinculos Autogenerado="13">
            <Vinculo IdVinculo="3" IdAtributo="11">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionDatosUnTitular.razon_social" />
            </Vinculo>
```

```
            <Vinculo IdVinculo="2" IdAtributo="10">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionDatosUnTitular.nom_tit" />
            </Vinculo>
            <Vinculo IdVinculo="1" IdAtributo="9">
              <Calculo Tipo="CalculoServicio"
Valor="RelacionDatosUnTitular.cod_tit" />
            </Vinculo>
          </ColVinculos>
        </ExpresionRelacion>
      </ColExpresionesRelaciones>
      <ColVariables Autogenerado="4">
        <Variable IdVariable="0" Tipo="long" Nombre="NroTitular"
Valor="" />
      </ColVariables>
      <ColAtributos Autogenerado="21">
        <Atributo IdAtributo="10" Tipo="string" Nombre="nom_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="9" Tipo="long" Nombre="cod_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="11" Tipo="string" Nombre="razon_social"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="4">
        <Relacion IdRelacion="1" Nombre="RelacionDatosUnTitular"
IdEntidadForanea="0" IdAtributoLocal="9" IdAtributoForaneo="0"
Cardinalidad="Na1" Tipo="relacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
```

```
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="1" Nombre="movimientos" Tipo="EntidadServida">
      <ColServicios Autogenerado="2">
        <Servicio IdServicio="1" Nombre="ObtenerMovimientos"
IdFuente="0">
          <ColParametros Autogenerado="0" />
          <ColVinculos Autogenerado="13">
            <Vinculo IdVinculo="0" IdAtributo="5">
              <Calculo Tipo="CalculoServicio" Valor="imp_mov_mn" />
            </Vinculo>
          </ColVinculos>
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="21">
        <Atributo IdAtributo="8" Tipo="decimal"
Nombre="MovimientoConSigno" Multiplicidad="False" Oculto="False"
Horizontal="False" Clave="False" />
        <Atributo IdAtributo="7" Tipo="decimal" Nombre="imp_mov_mn"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="6" Tipo="long" Nombre="cod_tita"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="5" Tipo="decimal" Nombre="Movimiento"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="4" Tipo="datetime" Nombre="fec_doc"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="3" Tipo="long" Nombre="nro_trans"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="4">
        <Relacion IdRelacion="0" Nombre="RelacionMovimientoTitulares"
IdEntidadForanea="0" IdAtributoLocal="6" IdAtributoForaneo="0"
Cardinalidad="Na1" Tipo="relacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
```

```
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="0" Nombre="titulares" Tipo="EntidadServida">
      <ColServicios Autogenerado="2">
        <Servicio IdServicio="0" Nombre="ObtenerTitulares"
IdFuente="0">
          <ColParametros Autogenerado="0" />
          <ColVinculos Autogenerado="13" />
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="21">
        <Atributo IdAtributo="2" Tipo="string" Nombre="razon_social"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="1" Tipo="string" Nombre="nom_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="0" Tipo="long" Nombre="cod_tit"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="True"
/>
      </ColAtributos>
      <ColRelaciones Autogenerado="4" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
                  </Value>
                </Image>
                <TextoEncabezado>
                </TextoEncabezado>
              </ReportItems>
            </PageHeader>
```

```
            <PageFooter>
              <Height>-1</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>-1</PageHeight>
            <PageWidth>-1</PageWidth>
            <LeftMargin>-1</LeftMargin>
            <RightMargin>-1</RightMargin>
            <TopMargin>-1</TopMargin>
            <BottomMargin>-1</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="Normal" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
  </ColEntidades>
</Esquema>
```

4.4 Ejemplo IV: Productora por Genero (Formato Matriz)

```
<Esquema Nombre="c:\Ej4-pruebasRSMatrizYAgrupar">
  <ColFuentes Autogenerado="2">
    <FuenteDeInformacion IdFuente="1" Nombre="otros_cines"
Tipo="webservice">
      <Parametro
Nombre="url">http://localhost/WebServiceCines/WSCines.asmx</Parametro>
    </FuenteDeInformacion>
    <FuenteDeInformacion IdFuente="0" Nombre="algunos_cines"
Tipo="webservice">
      <Parametro
Nombre="url">http://localhost/WebServiceCines/WSCines.asmx</Parametro>
    </FuenteDeInformacion>
  </ColFuentes>
  <ColEntidades Autogenerado="2">
    <Entidad IdEntidad="1" Nombre="GrupoProductoraGeneroCantidadID"
Tipo="EntidadRelacional">
      <ColExpresionesRelaciones Autogenerado="0">
        <ExpresionRelacion IdExpresionRelacion="0"
Nombre="agruparProductoraGenero">
          <CondicionIndividual Valor="" />
          <Ordenar />
          <Agrupar>
            <Atributo IdAtributo="5" />
            <Atributo IdAtributo="6" />
            <CondicionGrupal Valor="" />
          </Agrupar>
          <ColVinculos Autogenerado="4">
            <Vinculo IdVinculo="3" IdAtributo="4">
```

```
              <Calculo Tipo="CalculoServicio"
Valor="count(agrupamientoProdGeneroCountID.id)" />
            </Vinculo>
            <Vinculo IdVinculo="2" IdAtributo="6">
              <Calculo Tipo="CalculoServicio"
Valor="agrupamientoProdGeneroCountID.productora" />
            </Vinculo>
            <Vinculo IdVinculo="1" IdAtributo="5">
              <Calculo Tipo="CalculoServicio"
Valor="agrupamientoProdGeneroCountID.genero" />
            </Vinculo>
            <Vinculo IdVinculo="0" IdAtributo="3">
              <Calculo Tipo="CalculoServicio"
Valor="agrupamientoProdGeneroCountID.id" />
            </Vinculo>
          </ColVinculos>
        </ExpresionRelacion>
      </ColExpresionesRelaciones>
      <ColAtributos Autogenerado="7">
        <Atributo IdAtributo="6" Tipo="string" Nombre="productora"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="5" Tipo="string" Nombre="genero"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="4" Tipo="long" Nombre="cantidad_id"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="3" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
      </ColAtributos>
      <ColRelaciones Autogenerado="1">
        <Relacion IdRelacion="0"
Nombre="agrupamientoProdGeneroCountID" IdEntidadForanea="0"
IdAtributoLocal="3" IdAtributoForaneo="0" Cardinalidad="1aN"
Tipo="relacion" />
      </ColRelaciones>
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>5</Height>
              <ReportItems>
                <Image Name="image1">

<Value>Qk02pwEAAAAAADYAAAAoAAAAdwEAAGAAAAABABgAAAAAAAAAAAAAAjLgAAIy4AAAA
AAAAAAAAAts7EwNjOx9/
Vxd3TwNjOs8vBlq6ke5OJWnJoSGBWPVVLQFhOQVlPPVVLQlpQTmZceZGHr8e9hp6UdY2Dg
5uReZGHjKSakKiek6uhkKieiaGXgpqQfZWLe5OJfJSKfZaMbYh+c5GGcYqAXndtZHlwW3B
nVmdef5CHlaadl6qhmayjkamfi6OZhaCWiKOZjqedna+om6ukqLixqr+3m7KqmLOqmLOqh
KGYgaCXZYR7WnlwX351XXhvU25lUGhgSmJaXHRsb4d/
Vm5mS2NbSWFZQVlRS2NbQFhQRFxUL0c/
Rl5WTGRcPFRMO1NLN09HPVVNSWBYY3pye5KKepGJg5qSgJePbYV9gJiQfJeOdZCHcYyDbI
d+Z4R7co+GdJGIZ4J4ZX1zVWtfWG1kXnNqYXhweI+HaYBzQnIzQnITIlIDEkHi8kHS4jHC
0iHC0iGi4iGi4iGCkeGCkeGSofGywhHC4hHS8iHjEiHzIjGSwdGy4fHTAhHzIjIDIlIDIl
Hi8kHS8iJDcmHzMgFy0bFiwaFy8dGzMhGTQgGDMfFTIbFjMcFDQbFTYbFTYbFDYYEjcXEz
UXFjIeFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEfFzEf
FzEfFzEfFzEfFzEfFzEfFzEfAAAA</Value>
                </Image>
                <TextoEncabezado>Mis Peliculas</TextoEncabezado>
              </ReportItems>
            </PageHeader>
```

```
            <PageFooter>
              <Height>6</Height>
              <ReportItems>
                <TextoNumeroDePagina>True</TextoNumeroDePagina>
                <TextoTotalDePaginas>False</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>False</TextoFechaGeneradoReporte>
              </ReportItems>
            </PageFooter>
            <PageHeight>30</PageHeight>
            <PageWidth>20</PageWidth>
            <LeftMargin>1</LeftMargin>
            <RightMargin>3</RightMargin>
            <TopMargin>2</TopMargin>
            <BottomMargin>4</BottomMargin>
            <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
          </FormatoHojaReporte>
          <FormatoCuerpoReporte>
            <Plantilla Nombre="matrizCompacto" />
            <ColColumnas>
              <AnchoColumna Nombre="xxx">
                <Width>-1</Width>
              </AnchoColumna>
              <AnchoColumna Nombre="genero">
                <Width>5</Width>
              </AnchoColumna>
              <AnchoColumna Nombre="productora">
                <Width>5</Width>
              </AnchoColumna>
            </ColColumnas>
          </FormatoCuerpoReporte>
        </DisenioReporte>
      </FormatoReporte>
    </Entidad>
    <Entidad IdEntidad="0" Nombre="ProductoraIDGenero"
Tipo="EntidadServida">
      <ColServicios Autogenerado="1">
        <Servicio IdServicio="0" Nombre="ObtenerPeliculaIdGenero"
IdFuente="0">
          <ColParametros Autogenerado="0" />
          <ColVinculos Autogenerado="4" />
        </Servicio>
      </ColServicios>
      <ColAtributos Autogenerado="7">
        <Atributo IdAtributo="2" Tipo="string" Nombre="productora"
Multiplicidad="False" Oculto="False" Horizontal="True" Clave="False"
/>
        <Atributo IdAtributo="1" Tipo="string" Nombre="genero"
Multiplicidad="False" Oculto="False" Horizontal="False" Clave="False"
/>
        <Atributo IdAtributo="0" Tipo="long" Nombre="id"
Multiplicidad="False" Oculto="True" Horizontal="False" Clave="True" />
      </ColAtributos>
      <ColRelaciones Autogenerado="1" />
      <FormatoReporte>
        <DisenioReporte>
          <FormatoHojaReporte>
            <Width>-1</Width>
            <PageHeader>
              <Height>-1</Height>
              <ReportItems>
                <Image Name="image1">
                  <Value>
```

```
                    </Value>
                  </Image>
                  <TextoEncabezado>
                  </TextoEncabezado>
                </ReportItems>
              </PageHeader>
              <PageFooter>
                <Height>-1</Height>
                <ReportItems>
                  <TextoNumeroDePagina>True</TextoNumeroDePagina>
                  <TextoTotalDePaginas>True</TextoTotalDePaginas>

<TextoFechaGeneradoReporte>True</TextoFechaGeneradoReporte>
                </ReportItems>
              </PageFooter>
              <PageHeight>-1</PageHeight>
              <PageWidth>-1</PageWidth>
              <LeftMargin>-1</LeftMargin>
              <RightMargin>-1</RightMargin>
              <TopMargin>-1</TopMargin>
              <BottomMargin>-1</BottomMargin>
              <ImagenEncabezado>Logo de la empresa</ImagenEncabezado>
            </FormatoHojaReporte>
            <FormatoCuerpoReporte>
              <Plantilla Nombre="Normal" />
              <ColColumnas>
                <AnchoColumna Nombre="xxx">
                  <Width>-1</Width>
                </AnchoColumna>
              </ColColumnas>
            </FormatoCuerpoReporte>
          </DisenioReporte>
        </FormatoReporte>
      </Entidad>
    </ColEntidades>
</Esquema>
```

VISUAL REPORTING SERVICES

ANEXO III

LENGUAJE DE EDICIÓN DE META-MODELO

INDICE

1 INTRODUCCIÓN

En este documento se presenta el **Lenguaje de Edición del Meta-Modelo (LEM)** y las instrucciones que lo componen con las que se puede editar el *Meta-modelo de Negocio* (MMN), también se explica su forma de uso. Todas estas instrucciones pueden ser utilizadas en el *Editor de Metadata*, y un subconjunto de ellas en el *Editor de Modelo de Negocio* que es usado por el usuario final. Los comandos que no se pueden usar en el Editor de Modelo de Negocio son indicados explícitamente con "*" (un asterisco) en el título, el resto se podrá usar en ambos editores.

Durante la edición de un esquema se trabaja con estado, por lo que se debe seleccionar aquellos elementos sobre los que se quiera aplicar un comando (mediante el comando "seleccionar"). A lo largo del documento se hará referencia al objeto seleccionado, es en función del tipo de este elemento que se puede ejecutar uno u otro comando, sobre los objetos que contiene. Estos comandos se presentan agrupados por tipo de objetos que modifican. Cuando sea necesario se indicará que entidades deben estar seleccionadas para poder ejecutar los comandos.

1.1 CONSIDERACIONES

El lenguaje presenta las siguientes características generales:

- Los comandos **no** son case sensitive.
- No existen palabras reservadas, dado que se trata de un lenguaje de comandos.
- Cada línea se considera como una unidad independiente, que deberá comenzar con un comando. Luego del comando se encontrarán los parámetros, los parámetros estarán separados por coma y habrá parámetros opcionales que en este documento se indicaran entre paréntesis recto "[]" pero no se debe escribir con el comando.
- El fin de los comandos será indicado por un carácter de fin de línea.

Los comandos se presentan ordenados en capítulos según los objetos sobre los que actúan. Al principio de cada capítulo se explicarán parámetros que sean comunes a todos los comandos del capítulo. Los nombres de los objetos del modelo se ajustarán a la definición de identificadores, salvo que explícitamente se indique otra cosa.

2 EJEMPLOS

A continuación se presentarán algunos ejemplos del lenguaje, de forma de mostrar el uso del mismo. A medida que se va definiendo el esquema se explica como funcionan los comandos.

Se sugiere que los pasos, a grandes rasgos, para crear la representación de un modelo de negocio sean:

1) Crear un esquema nuevo.
   a) Crear las Fuentes de Información.
   b) Crear las Entidades, que se corresponde con los conceptos que habitualmente se usan en el negocio. Luego para cada entidad creada se debe:
      i) Agregarle los atributos
      ii) Crear las relaciones con otras entidades
      iii) Si el concepto a representar obtiene sus datos directamente de las fuentes de información crea un servicio, sino crear una expresión de relación.
         (1) Agregar los cálculos que permitan vincular cada atributo con los datos de la entidad.
         (2) Si el concepto lo requiere, agregar operaciones de Agrupación, Ordenación, Criterios o Variables.
   c) Se puede configurar el diseño del reporte asociado a una entidad o dejar el diseño por defecto
   d) Una vez creada una entidad que representa un concepto del negocio, se puede generar un reporte con los datos de la entidad.
   e) Cuando sea necesario se puede guardar el un modelo de negocio para una posterior edición.

Algunos de estos pasos se pueden cambiar de orden, por ejemplo se pueden crear algunas fuentes de información y luego crear entidades y posteriormente crear más fuentes de información.

Para ejecutar estos comandos de debe utilizar el Editor del Metadata, una explicación de cómo utilizar esta herramienta puede ser encontrada en los documentos **Manual de Usuario Final** y en el **Manual de Usuario Experto**, en ellos se detalla más sobre los pasos y requerimientos para instalar los editores y como usarlos.

2.1 EJEMPLO I: UNA ENTIDAD SERVIDA CON FUENTES HETEROGÉNEAS

En este ejemplo se define un esquema, y dentro de este se crea una entidad servida, la cual utiliza servicios de tres **fuentes de información heterogéneas** para obtener sus datos. La entidad creada representará los titulares de las cuentas existentes en los sistemas de información accedidos.

Para comenzar creamos el esquema de metadata, esto se hace con el siguiente comando:

```
cres c:\titularesMagma
```

el nombre del esquema es "titularesMagma" y el archivo representando el esquema se encontrará en c:. El comando utilizado es crearesquema, para el cual también se tiene la abreviación cres. Luego creamos las tres fuentes de información, que son titulares_dll, titulares_ws y titulares_remoting:

```
crfi titulares_dll, tipo:dllfile, path:D:\ProyectoDeGradoFI\
WebServicesEjemplo\CapaDatos\bin\Debug, dll-name:CapaDatos.dll,
clase:CapaDatos.AccesoADatos

crfi titulares_ws, tipo:webservice,
url:http://localhost/WebServiceCines/WSCines.asmx

crfi titulares_remoting, tipo:remoting, path:D:\ProyectoDeGradoFI\
WebServicesEjemplo\RemotingProxy\bin\Debug, rem-name:RemotingProxy.dll,
proxy:RemotingProxy.RemotingProxy, metodoacceso:ObtenerObjeto,
interface:Comun.IAccesoADAtos
```

`crfi` es la abreviación de `crearFuente`. Cada una de estas fuentes es de un tipo distinto, la primera es una biblioteca de ensamblados (`tipo:dllfile`), la segunda un servicio web (`tipo:webservice`) y la tercera es un servicio de remoting (`tipo:remoting`). Por esto a cada una se accede de forma distinta y los parámetros utilizados para hacerlo son distintos.

Ahora se puede crear la Entidad llamada "titulares", con los atributos: Id, cod_tit, nom_tit, razon_social; los datos surgen de tres servicios que definimos más adelante, uno de cada fuente de información titulares_dll, titulares_ws y titulares_remoting.

```
cren titulares
seleccionar entidad, titulares
crearatributo id, t:string, c:false
crat cod_tit, t:long, o:false, c:false
crat nom_tit, t:string, o:false
crat razon_social, t:string
```

Con estos comandos se crea la entidad y luego se la selecciona para crear los atributos en ella, esto se hace mediante el comando seleccionar (la forma abreviada es "sl") indicando el tipo de objeto que se quiere seleccionar (`entidad`) y su nombre (`titulares`). Para cada atributo se da un nombre y un tipo, notar que id es el atributo clave. El resto de las características de los atributos toman los valores por defecto (ver 10.1).

Ahora agregamos los servicios, los cuales indican la forma de obtener los datos relacionados con esta entidad. Además, se vincula cada atributo con un campo proveniente del servicio o con un cálculo puede o no utilizar atributos provenientes del servicio.

Para el primer servicio definiremos solo un vínculo, para el atributo id, el resto de los atributos quedan automáticamente vinculados con un campo en el servicio que tiene exactamente el mismo nombre que el atributo. El vínculo para id se obtiene de concatenar la palabra `'ObtenerTitulares'` con `cod_tit`.

```
crse ObtenerTitulares, titulares_remoting
seleccionar servicio, ObtenerTitulares
vincular id, 'ObtenerTitulares' + convert(cod_tit,'System.String')
liberar
```

Notar que para definir este servicio se tiene seleccionada la entidad `titulares` (que fue seleccionada para definir los atributos), luego se selecciona el servicio para definir el vínculo y después se libera el servicio quedando seleccionada la entidad.

Para el siguiente servicio se definen vínculos para 4 de los atributos. Un caso particular es el de `razon_social` que no tiene correspondencia con la información preveniente de la fuente por lo cual se le vincula con un valor fijo como '- - - -'.

```
crse ObtenerClientes, titulares_dll
seleccionar servicio, ObtenerClientes
vincular id, 'ObtenerClientes' + Cedula_De_Identidad
vincular cod_tit, convert(Cedula_De_Identidad,'System.Int64')
vincular nom_tit, Nombre
vincular razon_social, '----'
liberar
```

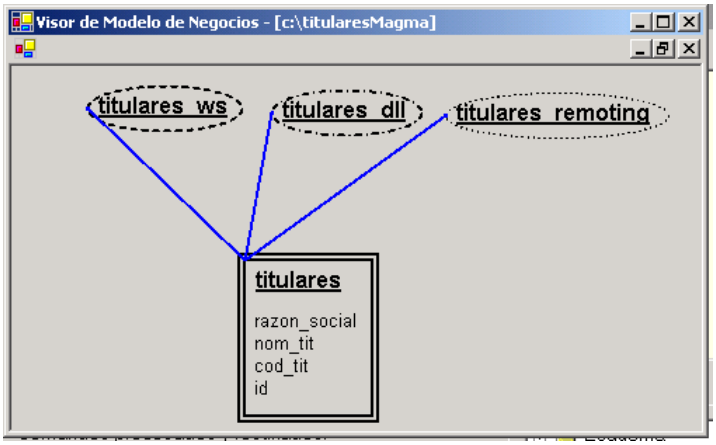
Se crea el servicio `ObtenerClientesPorNombre` el cual requiere del pasaje de un parámetro con el nombre del titular, para este caso es 'Ana'.

```
crse ObtenerClientesPorNombre, titulares_ws, pNombre:Ana
seleccionar servicio, ObtenerClientesPorNombre
vincular id, 'ObtenerClientesPorNombre' + Cedula_De_Identidad
vincular cod_tit, convert(Cedula_De_Identidad,'System.Int64')
vincular nom_tit, Nombre
vincular razon_social, '----'
liberar
```

Para guardar el esquema creado se puede utilizar el comando `guardaresquema` (o `gues`). En el último renglón se genera un reporte con todos los datos de la entidad, que provienen de diferentes fuentes de información:

```
generarreporte titulares
```

El esquema generado se puede ver en el visor de la siguiente forma:



Aquí las fuentes de información son los óvalos y las entidades los rectángulos. Las relaciones entre las entidades y las relaciones entre las entidades y las fuentes de información se indican mediante las líneas.

2.2 EJEMPLO II: VENTA DE ENTRADAS DE CINE

En el siguiente ejemplo se definirán un modelo relacionado con una cadena de cines. En el se definen las entidades película, complejo, sala, función, **programación, funcionProgramada,** entrada vendida, **EntradasPorPeliculaPorFecha, recaudacionXsalaXfecha**.

Primero se crea el esquema y las fuentes de información utilizadas, y se guarda el esquema generado con estos comandos:

```
cres c:\pruebasFI00ConRelacionesN
crfi cines_ws, tipo:webservice,
url:http://localhost/WebServiceCines/WSCines.asmx
crfi cines_dll, tipo:dllfile, path:d:\ProyectoDeGradoFI\WebServicesEjemplo\
CapaDatos\bin\Debug, dll-name:CapaDatos.dll, clase:CapaDatos.AccesoADatos
gues
```

Se crea la entidad película, con los atributos nombre, id y nombreCartelera. Luego se crea un servicio para esta entidad llamado "ObtenerPeliculas". Este servicio se encuentra en la fuente de información recién creada y no tiene parámetros (más adelante se definirán servicios con parámetros):

```
cren pelicula
seleccionar entidad, pelicula
crat nombreOriginal, t:string, o:false, c:true
crat id, t:long, o:true, c:true
crat nombreCartelera, t:string
crat nombreEntrada, t:string
crat duracion, t:int
crat duracionTotal, t:int
crat fechaEstreno, t:date
crat proximoEstreno, t:boolean
```

A continuación se define el servicio y luego los vínculos entre los resultados del servicio y los atributos de la entidad, notar que el servicio `ObtenerPeliculas` requiere de varios parámetros:

```
crse ObtenerPeliculas, cines_ws, pId:-1, pNombre:a, pProximoEstreno:true,
pFechaEstreno:09/12/2000, pDuracionMinima:1

seleccionar servicio, ObtenerPeliculas
vincular nombreEntrada, substring(nombreCartelera, 1, 10)
vincular id, id
vincular duracionTotal, duracion + tiempoCorto + tiempoTrailers
liberar
```

A continuación se define otro servicio para la entidad el cual tiene como parámetro del campo genero, cuyo valor será xxx.

```
crse ObtenerPeliculasGenero, cines_dll, pGenero:xxx
seleccionar servicio, ObtenerPeliculasGenero
vincular nombreEntrada, substring(nombreCartelera, 1, 10)
vincular id, id
vincular duracionTotal, duracion + tiempoTrailers
liberar
```

Ahora pasamos a definir la entidad complejo, pero después del último liberar quedó seleccionada la entidad películas, se debe liberar esta para tener seleccionado el esquema y poder crear la nueva entidad. Los complejos tienen id, nombre, ciudad, direccion, en particular el atributo id es el campo clave (c:true) y también tiene este atributo como oculto (o:true), también se crea un servicio para la entidad:

```
liberar

cren complejo
seleccionar entidad, complejo
crat id, t:long, o:true, c:true
crat nombre, t:string, o:false, c:true
crat ciudad, t:string
crat direccion, t:string
/* creo los servicios para la entidad pelicula */
crse ObtenerComplejos, cines_ws
liberar
```

Luego continua la definición de la entidad `sala`:

```
/*creo la entidad sala*/
cren sala
seleccionar entidad, sala
crat id, t:long, o:true, c:true
crat nombre, t:string
crat idcomplejo, t:long, o:true
crat complejo, t:string
/* creo los servicios para la entidad sala */
crse ObtenerSalas, cines_dll
seleccionar servicio, ObtenerSalas
vincular complejo, RELcomplejo.nombre
liberar
liberar
```

Continúa con la definición de la entidad servida `funcion`:

```
cren funcion
seleccionar entidad, funcion
crat id, t:long, o:true, c:true
crat nombre, t:string, c:true
/* creo los servicios para la entidad funcion */
crse ObtenerFunciones, cines_dll

liberar
```

Crea la entidad relacional `programacion`:

```
cren programacion
```

```
seleccionar entidad, programacion
crat id, t:long, o:true, c:true
crat fecha_inicio, t:date
crat fecha_fin, t:date
crat complejo, t:string
crat complejo_id, t:long, o:true
```

Agrega a la entidad `programacion` una relación con la entidad `complejo` a través del atributo id

```
crre RELcomplejo, n:complejo, al:complejo_id, af:id, c:na1, t:relacion
```

Creo los servicios para la entidad `programacion`:

```
crse ObtenerProgramaciones, cines_ws
seleccionar servicio, ObtenerProgramaciones
vincular complejo, RELcomplejo.nombre
/* libero el servicio y la entidad */
liberar
liberar
```

Creo la entidad servida `funcionProgramada`:

```
cren funcionProgramada
seleccionar entidad, funcionProgramada
/* creo los atributos */
crat id, t:long, o:true, c:true
crat fecha, t:date, c:true
crat hora, t:time, c:true
crat programacion_id, t:long, o:true
crat sala_id, t:long, o:true, c:true
crat funcion_id, t:long, o:true
crat pelicula_id, t:long, o:true
crat sala, t:string
crat funcion, t:string
crat pelicula, t:string
```

Creo relaciones con múltiples entidades:

```
crre RELprogramacion, n:programacion, al:programacion_id, af:id, c:na1,
t:relacion
crre RELsala, n:sala, al:sala_id, af:id, c:na1, t:relacion
crre RELfuncion, n:funcion, al:funcion_id, af:id, c:na1, t:relacion
crre RELpelicula, n:pelicula, al:pelicula_id, af:id, c:na1, t:relacion
```

Crea los servicios para la entidad `funcionProgramada`:

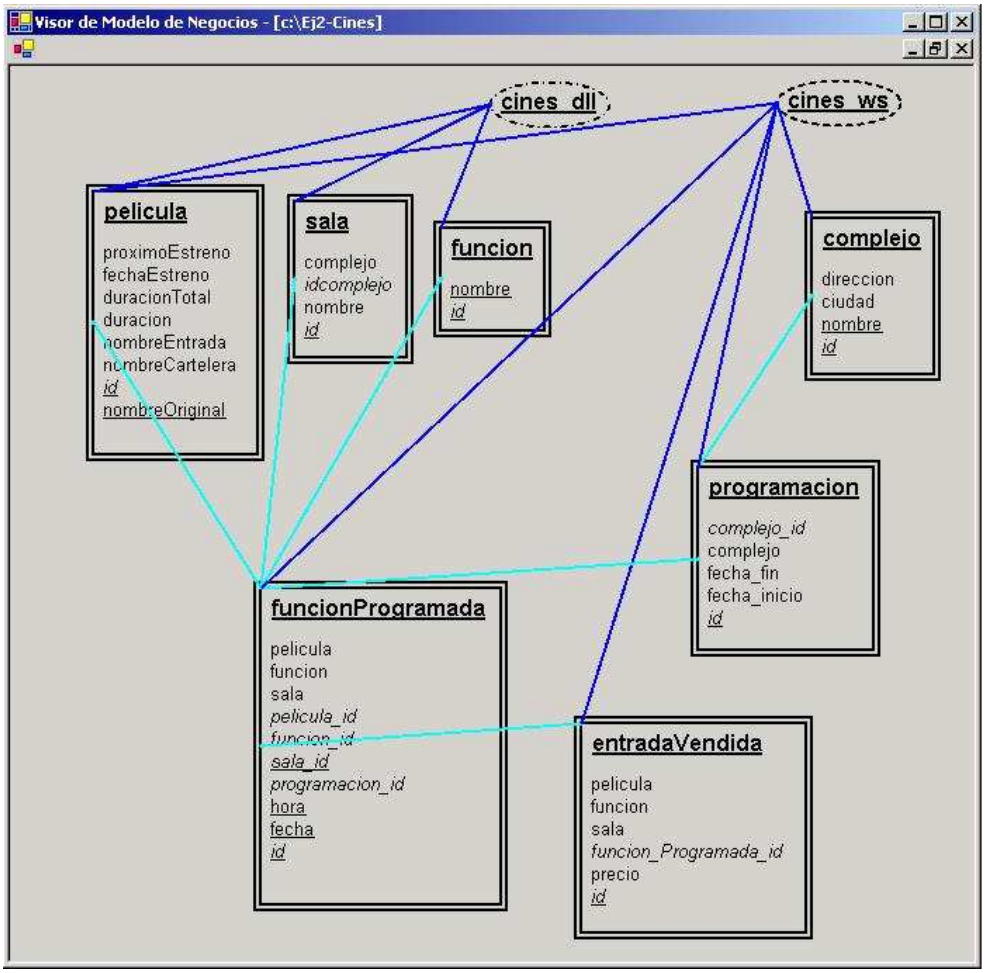
```
crse ObtenerFuncionesProgramadas, cines_ws
sl servicio, ObtenerFuncionesProgramadas
vincular hora, hora_comienzo
vincular sala, RELsala.nombre
vincular funcion, RELfuncion.nombre
vincular pelicula, RELpelicula.nombreCartelera
```

Creo la entidad servida `entradaVendida`:

```
/* libero el servicio y la entidad */
Liberar
Liberar
cren entradaVendida
seleccionar entidad, entradaVendida
crat id, t:long, o:true, c:true
crat precio, t:int
crat funcion_Programada_id, t:long, o:true
crat sala, t:string
crat funcion, t:string
crat pelicula, t:string
```

```
crre   RELfuncionProgramada,   n:funcionProgramada,   al:funcion_Programada_id,
af:id, c:na1, t:especializacion
/* creo los servicios para la entidad entradaVendida */
crse ObtenerEntradasVendidas, cines_ws
sl servicio, ObtenerEntradasVendidas
vincular sala, RELfuncionProgramada.sala
vincular funcion, RELfuncionProgramada.funcion
vincular pelicula, RELfuncionProgramada.pelicula
liberar
liberar
```

Ahora tenemos definidas las entidades necesarias para empezar el análisis de los datos. En el visor se puede ver el meta-modelo representado hasta el momento de la siguiente forma:



Todas las entidades definidas hasta el momento son servidas, y a partir de estas creamos las entidades relacionales, ha partir de las cueles se puede obtener, por ejemplo, un reporte de las entradas vendidas por película por fecha

```
cren EntradasPorPeliculaPorFecha
seleccionar entidad, EntradasPorPeliculaPorFecha
crat pelicula_id, t:long, o:true
crat pelicula, t:string
crat fecha, t:date, h:true
crat cantidadEntradas, t:int
crat cantidadRecaudada, t:long
crat precioMaximo, t:int
crat prcioMinimo, t:int
crat funcion_Programada_id, t:long, o:true
crat funcion_Programada_id_Entrada, t:long, o:true
```

Ya creamos la entidad con sus atributos, y ahora la relacionamos con `funcionProgramada y entradaVendida`:

```
crre RELfuncionProgramada, n:funcionProgramada, al:funcion_Programada_id,
af:id, c:na1, t:especializacion
crre RELentradaVendida, n:entradaVendida, al:funcion_Programada_id_Entrada,
af:funcion_Programada_id, c:na1, t:especializacion
```

Ahora creamos la expresión de relación con una condición simple, y vincula los atributos:

```
crex condidcion, c: (RELfuncionProgramada.fecha >= '05/03/2005') and
(RELfuncionProgramada.fecha <= '12/09/2006') and (RELfuncionProgramada.id =
RELentradaVendida.funcion_Programada_id)
seleccionar expresionRelacion, condidcion
vincular pelicula_id, RELfuncionProgramada.pelicula_id
vincular pelicula, RELfuncionProgramada.pelicula
vincular fecha, RELfuncionProgramada.fecha
vincular cantidadEntradas, count(entradaVendida.precio)
vincular cantidadRecaudada, sum(RELentradaVendida.precio)
vincular precioMaximo, max(RELentradaVendida.precio)
vincular prcioMinimo, min(RELentradaVendida.precio)
agrupar pelicula, fecha
liberar
liberar
```

En la declaración de los vínculos se especifican algunos cálculos, como el conteo, la suma o el precio máximo y mínimo de las entradas.

En el caso de esta entidad las fechas para las cuales se obtiene el reporte son fijas, pero mediante el uso de variables los valores de las fechas podrían ser diferentes según la instancia del reporte que se desee generar. Ejemplos de entidades con variables no se muestran en este ejemplo, pero se verán más adelante en el documento.

A partir de esta entidad se puede generar un reporte mediante el comando:

```
generarreporte EntradasPorPeliculaPorFecha
```

Ahora creamos una entidad relacional similar, a la que llamamos `recaudacionXsalaXfecha` con sus atributos relaciones vínculos y expresión de relación con condiciones individuales:

```
cren recaudacionXsalaXfecha
seleccionar entidad, recaudacionXsalaXfecha
crat sala_id, t:long, o:true
crat sala, t:string
crat fecha, t:date, h:true
crat cantidadEntradas, t:long, o:true
crat cantidadRecaudada, t:long
crat precioMaximo, t:int, o:true
crat prcioMinimo, t:int, o:true
crat funcion_Programada_id, t:long, o:true
crat funcion_Programada_id_Entrada, t:long, o:true

/* relaciona la entidad actual con funcionProgramada */
crre RELfuncionProgramada, n:funcionProgramada, al:funcion_Programada_id,
af:id, c:na1, t:especializacion
crre RELentradaVendida, n:entradaVendida, al:funcion_Programada_id_Entrada,
af:funcion_Programada_id, c:na1, t:especializacion
```

Creo una expresión de relación con una condición que tiene fechas y atributos de otras entidades, notar que para este último caso se pone el nombre de la relación no de la entidad.

```
crex condidcion, c: (RELfuncionProgramada.fecha >= '05/03/2005') and
(RELfuncionProgramada.fecha <= '12/09/2006') and (RELfuncionProgramada.id =
RELentradaVendida.funcion_Programada_id)

seleccionar expresionRelacion, condidcion
vincular sala_id, RELfuncionProgramada.sala_id
```

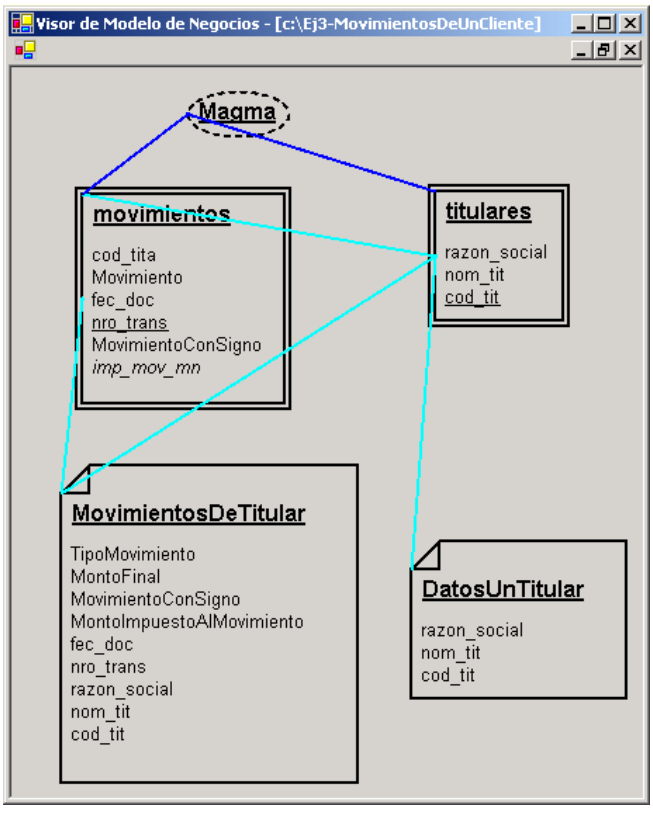
```
vincular sala, RELfuncionProgramada.sala
vincular fecha, RELfuncionProgramada.fecha
vincular cantidadEntradas, count(RELentradaVendida.precio)
vincular cantidadRecaudada, sum(RELentradaVendida.precio)
vincular precioMaximo, max(RELentradaVendida.precio)
vincular prcioMinimo, min(RELentradaVendida.precio)
agrupar sala, fecha
```

2.3 EJEMPLO III: ESTADO DE CUENTA (CON VARIABLES)

En este ejemplo se muestra el uso de las variables, las que nos permiten instanciar reportes con datos de entrada variables. Definiremos un reporte en el que se muestra el estado cuenta de un cliente y los movimientos que ha realizado, cual es el cliente para el que se quiere el reporte se indicará a través de una variable en el momento de generarse el reporte, en este esquema se maneja la información de los Clientes y los Movimientos de la cuenta corriente.

Para este modelo de negocio se maneja la Entidad Servida Cliente la cual muestra algunos datos de los clientes (titulares de la cuenta), la Entidad Servida Movimientos que maneja algunos datos de todas las cuenta corriente de los clientes, la Entidad Reporte Datos de un Cliente que a través de una variable selecciona los datos de un cliente y por último la Entidad Reporte MovimientosDeTitular que muestra los movimientos de un cliente desde una fecha indicada.

El esquema que se generará se podrá ver en el visor representado de la siguiente forma:



Se crea el esquema y se definen las fuentes de información:

```
cres c:\MovimientosDeUnCliente
gues
crfi                    Magma,                    tipo:webservice,
url:http://localhost/WSMagma/MagmaServicioWeb.asmx
```

Creo la entidad titulares con los servicios:

```
cren titulares
seleccionar entidad, titulares
```

```
crat cod_tit,              t:long, o:false, c:true
crat nom_tit,              t:string, o:false
crat razon_social, t:string
crse ObtenerTitulares, Magma
liberar
```

Creo la entidad `movimientos` con las relaciones a la entidad `titulares` además define los servicios y vínculos:

```
cren movimientos
seleccionar entidad, movimientos
crat nro_trans,     t:long, o:false, c:true
crat fec_doc,              t:DateTime
crat Movimiento,    t:decimal
crat cod_tita,      t:long, o:false
crat imp_mov_mn,    t:decimal,o:true
crat MovimientoConSigno, t:decimal

crearRelacion     RelacionMovimientoTitulares,     n:titulares,     al:cod_tita,
af:cod_tit, c:Na1
crse ObtenerMovimientos, Magma
seleccionar servicio, ObtenerMovimientos
vincular Movimiento, imp_mov_mn
liberar
liberar
```

Crea la Entidad Reporte `DatosUnTitular` relacionada con la entidad `titulares`:

```
cren DatosUnTitular
seleccionar entidad, DatosUnTitular
crat cod_tit, t:long, o:false, c:false
crat nom_tit, t:string, o:false
crat razon_social, t:string

crearRelacion RelacionDatosUnTitular, n:titulares, al:cod_tit,c:Na1
```

Ahora creamos una expresión de relación y en ella una variable llamada `@NroTitular`:

```
crearExpresionRelacion ExpresionDatosUnTitular
crva @NroTitular, t:long
```

Agregamos a la expresión de relación una serie de vinculos a los atributos de la entidad a la que se hace referencia en la relación `RelacionDatosUnTitular`:

```
seleccionar EXPRESIONRELACION, ExpresionDatosUnTitular
vincular cod_tit,          RelacionDatosUnTitular.cod_tit
vincular nom_tit,       RelacionDatosUnTitular.nom_tit
vincular razon_social,     RelacionDatosUnTitular.razon_social
```

Agrega a la expresión de relación una condición donde participa la variable que definimos:

```
condicion individual,      RelacionDatosUnTitular.cod_tit = @NroTitular
```

Genera un reporte con la entidad `DatosUnTitular` y asignando a la variable `@NroTitular` el valor 10, mostrando solo los datos del cliente con `cod_tit` igual a 10.

```
generarreporte DatosUnTitular, @NroTitular:10
```

Ahora creamos la segunda Entidad de Reporte `MovimientosDeTitular`, a la cual relacionamos con las entidades `titulares` y `movimientos`, además los datos para esta nueva entidad se obtienen a partir de una expresión de relación en la que participan tres variables. Por otro lado, en esta entidad se invocan una serie de funciones, tanto en los vínculos de los atributos como en las condiciones.

Primero se deben debo liberar la expresión y la entidad que estaban seleccionadas.

```
liberar
liberar
```

Y ahora si podemos definir la entidad `MovimientosDeTitular`:

```
cren MovimientosDeTitular
seleccionar entidad, MovimientosDeTitular
crat cod_tit, t:long, o:false, c:false
crat nom_tit, t:string, o:false
crat razon_social, t:string
crat nro_trans, t:long, o:false
crat fec_doc, t:Date
crat hora_doc, t:Time
crat MontoImpuestoAlMovimiento, t:decimal
crat MovimientoConSigno, t:decimal
crat MontoFinal, t:decimal
crat TipoMovimiento, t:string

crearRelacion RelacionTitulares, n:titulares, al:cod_tit
crearRelacion RelacionMovimientos, n:movimientos, al:nro_trans

crearExpresionRelacion ExpresionMovimientoDeTitular
```

Define tres variables de la entidad:

```
crva @NroTitular, t:long
crva @FechaMinima, t:DateTime
crva @PorcentajeImpuestoMov, t:decimal

seleccionar EXPRESIONRELACION, ExpresionMovimientoDeTitular

vincular cod_tit, RelacionTitulares.cod_tit
vincular nom_tit, RelacionTitulares.nom_tit
```

Agregamos un vinculo para el atributo `razon_social`, en el cual se aplica la función `substring` para obtener los 2 últimos caracteres de la razón social, para obtener el largo utiliza la función `len`:

```
vincular razon_social, substring(RelacionTitulares.razon_social,
len(RelacionTitulares.razon_social), 2)

vincular nro_trans, RelacionMovimientos.nro_trans
vincular fec_doc, RelacionMovimientos.fec_doc
vincular hora_doc, RelacionMovimientos.fec_doc
vincular MovimientoConSigno, RelacionMovimientos.MovimientoConSigno
```

Otra función muy importante es aplicada acá en el vínculo con el atributo TipoMovimiento y es la función `iif` la cual dependiendo de la condición muestra el valor que se encuentra en el segundo parámetro o el valor del tercer parámetro.

```
vincular TipoMovimiento, substring
(iif(RelacionMovimientos.MovimientoConSigno>=0,'Deposito','Retiro'),1,3)

vincular MontoImpuestoAlMovimiento,
       RelacionMovimientos.imp_mov_mn*@PorcentajeImpuestoMov/100
vincular MontoFinal,RelacionMovimientos.MovimientoConSigno -
RelacionMovimientos.Movimiento * (@PorcentajeImpuestoMov/100)
```

En este caso se define una condición individual con variables:

```
condicion individual, (RelacionMovimientos.cod_tita= @NroTitular) and
                    (RelacionTitulares.cod_tit=RelacionMovimientos.cod_tita)
                    and (RelacionMovimientos.fec_doc > @FechaMinima)
```

Por último se genera el reporte con la entidad MovimientosDeTitular pasando los valores para las tres variables:

```
generarreporte MovimientosDeTitular,
@FechaMinima:'15/12/2001' ,@NroTitular:10, @PorcentajeImpuestoMov:1
```

Se aplican otras operaciones sobre las variables como eliminar una variable de la entidad actual:

```
Condicion individual, (RelacionTitulares.cod_tit= @NroTitular) and
(RelacionTitulares.cod_tit=RelacionMovimientos.cod_tita)
liberar

elVA @FechaMinima
```

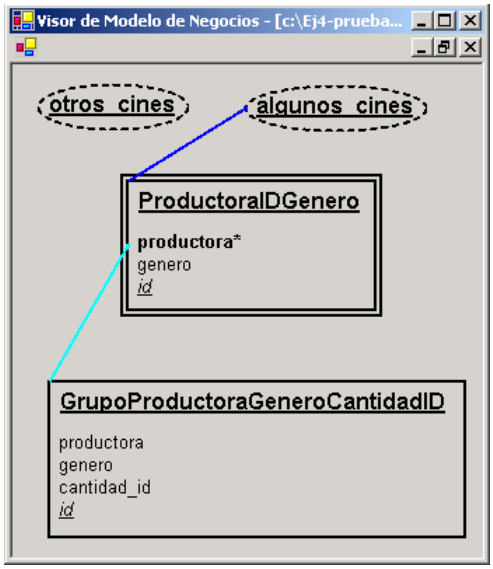
Notar que en este caso solo se asignan dos variables, porque la otra fue eliminada:

```
generarreporte MovimientosDeTitular, @NroTitular:10, @PorcentajeImpuestoMov:1
```

2.4 EJEMPLO IV: PRODUCTORA POR GÉNERO (FORMATO MATRIZ)

Este esquema mantiene las productoras películas y su género por una productora lo que se quiere mostrar es la cantidad de películas que produjo cada productora discriminando las películas por genero.

Aquí se muestra el manejo de operaciones de agrupación de datos, operaciones agregadas sobre los grupos y el uso de los atributos horizontales para la generación de reportes de tipo Matriz. El esquema utilizado se puede ver de la siguiente forma:



Comenzamos por crear el esquema y definir las fuentes de información.

```
cres c:\pruebasRSMatrizYAgrupar00
gues
crfi algunos_cines, tipo:webservice,
url:http://localhost/WebServiceCines/WSCines.asmx
crfi otros_cines, tipo:webservice,
url:http://localhost/WebServiceCines/WSCines.asmx
```

Crea una entidad servida con los el Id de las películas su género y productora:

```
cren ProductoraIDGenero
seleccionar entidad, ProductoraIDGenero
crat id, t:long, o:true, c:true
crat genero, t:string
crat productora, t:string, h:true
```

Creo los servicios para la entidad `ProductoraIDGenero`

```
crse ObtenerPeliculaIdGenero, algunos_cines
```

Crea la Entidad Relacional `GrupoProductoraGeneroCantidadID` que será la entidad que contenga los agrupamientos y las funciones agregadas, además define la relación con la otra entidad (primero se libera la entidad que se tenía seleccionada).

```
liberar
cren GrupoProductoraGeneroCantidadID
seleccionar entidad, GrupoProductoraGeneroCantidadID
crat id, t:long, o:true, c:true
crat cantidad_id, t:long
crat genero, t:string
```

Notar que el siguiente atributo es horizontal (h:true), esto junto con la definición del agrupamiento harán que el reporte tipo matricial, siendo los datos de este atributo colocados como título de columna

```
crat productora, t:string, h:true

crearRelacion agrupamientoProdGeneroCountID, n:ProductoraIDGenero, al:id
crearExpresionRelacion agruparProductoraGenero
seleccionar EXPRESIONRELACION, agruparProductoraGenero
```

Define un agrupamiento con dos atributos, lo cual es el segundo punto para que muestre los datos en tipo matriz.

```
agrupar genero, productora
```

Vincula cada atributo con un la forma de calcular sus datos

```
vincular id,agrupamientoProdGeneroCountID.id
vincular genero,agrupamientoProdGeneroCountID.genero
vincular productora,agrupamientoProdGeneroCountID.productora
vincular cantidad_id , count(agrupamientoProdGeneroCountID.id)
liberar
liberar
```

Ahora genera el reporte con el tipo de matriz

```
generarreporte GrupoProductoraGeneroCantidadID
```

Notar que modificando el valor del atributo productora para que no sea horizontal (h:false) el reporte vuelve a ser del tipo tabular.

```
seleccionar entidad, GrupoProductoraGeneroCantidadID
modificarAtributo productora , h:false
generarreporte GrupoProductoraGeneroCantidadID
```

Cambia el orden en que se presentan los atributos del reporte tabular.

```
seleccionar EXPRESIONRELACION, agruparProductoraGenero
ordenar productora ASCENDENTE, genero DESCENDENTE
generarreporte GrupoProductoraGeneroCantidadID
```

Volvemos al atributo productora a horizontal para seguir trabajando con el reporte tipo Matriz

```
modificarAtributo productora , h:true
```

También se puede configurar otra serie de características en cuanto al formato del reporte del tipo Matriz, como ser:

- Ancho de columnas

```
ModificarAnchoColumna productora , 5
ModificarAnchoColumna genero, 5
```

- Crea una plantilla de formato de reporte tipo matriz, partiendo de un reporte previamente generado, del cual toma los colores y las fuentes de las diferentes áreas. Notar que se tiene que indicar la ubicación del archivo RDL.

```
CrearPlantillaCuerpoReporte FormatoCuerpo12Matriz, 'c:\RDL\
Reporte11Matriz.rdl'
```

- Cambia el formato del reporte para el indicado en la plantilla `FormatoCuerpo12Matriz` generada anteriormente, también cambia tamaño de la hoja y del los márgenes, indicar si se muestra un encabezado y un pie y el logo de la empresa (`c:\Logo.jpg`):

```
ModificarFormatoReporte FormatoCuerpo12Matriz, 30, 20, 5, 6, false ,true ,
false , 'Mis Peliculas - Nuevo Formato Matriz', 1, 2, 3, 4,'c:\Logo.jpg'
```

Generar el reporte con el nuevo formato establecido
generarreporte GrupoProductoraGeneroCantidadID

Al elimina la plantilla que tiene el reporte actualmente queda el formato por defecto, generando así una forma rápida de cambiar todos los formatos de todos los reportes a un nuevo formato que es el formato por defecto.

```
ModificarFormatoReporte FormatoCuerpo12Matriz, 30, 20, 5, 6, false ,true ,
false , 'Mis Peliculas - Formato Matriz por defecto', 1, 2, 3, 4,'c:\Logo.jpg'
EliminarPlantillaCuerpoReporte FormatoCuerpo12Matriz
generarreporte GrupoProductoraGeneroCantidadID
```

COMANDOS SOBRE ESQUEMAS

Los nombres de los esquemas deberán identificar al sistema, y pueden ser un identificador o varios identificadores separados por "\", a los que llamaremos paths.

2.5 COMANDO CREARESQUEMA (CRES)*

crearEsquema [*nombre*]

Descripción:
Crea un esquema nuevo y vacío, con el *nombre* indicado, el esquema podría no tener *nombre al ser creado*. En caso de que no se indique un nombre este deberá indicarse al guardar el esquema.

2.6 COMANDO RENOMBRARESQUEMA (RENES)

renombrarEsquema *nuevo_nombre*

Descripción:
Renombra el esquema actual.

2.7 COMANDO ELIMINARESQUEMA(ELES)

eliminarEsquema [*nombre*]

Descripción:
Elimina el esquema indicado en el *nombre* y en caso de no indicarse borra el esquema actual.

2.8 COMANDO ABRIRESQUEMA (ABES)

abrirEsquema *nombre*

Descripción:
Abre el esquema identificado por el *nombre* pasado como parámetro.

2.9 COMANDO GUARDARESQUEMA (GUES)

guardarEsquema [*nombre*]

Descripción:
Guarda el esquema. En caso de ser nuevo se deberá indicar el nombre, en caso de no ser nuevo y no se indicó un nombre se guarda el esquema con el mismo nombre con el que se abrió.

2.10 COMANDO CERRARESQUEMA (CEES)

cerrarEsquema

Descripción:
Cierra el esquema abierto.

3 COMANDOS SOBRE FUENTES DE INFORMACIÓN

Los nombres de las fuentes de información son identificadores. Las url corresponden a la definición habitual de la sintaxis de una URL.

Para aplicar cualquiera de estos comandos se debe tener seleccionado un Esquema.

3.1 COMANDO CREARFUENTE (CRFI)*

crearFuente *nombre*, url

Descripción:
Crea una nueva asociación a una fuente de información. Se deberá indicar la url en la que se acceda a la fuente de información.

3.2 COMANDO MODIFICARFUENTE (MODFI)*

modificarFuente *nombre* [*,nuevoNombre*] [, url]

Descripción:
Permite modificar la fuente de información, su nombre y su url.

3.3 COMANDO ELIMINARFUENTE (ELFI)*

eliminarFuente *nombre*

Descripción:
Permite eliminar la fuente de información. **No** se verifica si esta fuente es usada dentro de algún esquema o no.

3.4 COMANDO CHECKEARFUENTE (CHFI)*

checkearFuente *nombre*

Descripción:
Verifica si se puede establecer una conexión con la fuente de información.

4 COMANDO GENERALES

Los comandos agrupados bajo este punto no requerirán de tener un único tipo objeto seleccionado, es decir que se podrán aplicar cuando se tienen objetos de diferentes tipos o se aplican al modelo (esquema) que se está editando actualmente. Los tipos de objetos de un modelo son: esquema, entidad, entidadServida, entidadRelacional, entidadReporte, relaciones, expresionRelacion, vinculo, condicion, ordenación, agrupación, atributos, fuenteInformacion, servicios o parámetros.

4.1 COMANDO SELECCIONAR (SL)

seleccionar *tipo_objetivo, identificador*

Descripción:
Permite seleccionar un objeto, que pertenezca al objeto seleccionado actualmente, indicando su tipo e identificador.

4.2 COMANDO LIBERAR (LB)

liberar

Descripción:
Libera el objeto seleccionado actualmente, quedando con el objeto contenedor del objeto seleccionado como actual.

4.3 COMANDO VINCULAR (VI)

Este comando se puede ejecutar cuando se tienen seleccionados un servicio o una expresión de relación.

vincular *atributo_local* [, *calculo*]

Descripción:
Permitirá vincular el ***atributo_local*** de la entidad seleccionada con el ***calculo***, este cálculo será la forma de obtener el valor del atributo a partir de los resultados de las relaciones o los servicios de la entidad a la que pertenezca el atributo. El formato del cálculo está definido más arriba.

Para cada atributo de la entidad seleccionada existirá como máximo un vínculo. En caso de no existir ningún vínculo, se asumirá un vínculo con un atributo del mismo nombre en la entidad o en el servicio de la fuente de información relacionada. Si dos o más entidades tienen atributos con el mismo nombre será obligatoria la definición del vínculo.

Los operadores y funciones que se podrán usar están definidos más abajo en **Aclaración sobre las Funciones y Operadores**.

4.4 COMANDO CONDICION (CO)

Este comando se puede ejecutar cuando se tiene seleccionada una expresión de relación.

Condición *tipo*, *condicion*

Descripción:
Las condiciones serán sentencias que se puedan evaluar como verdaderas o falsas, y el *tipo* podrán ser *INDIVIDUAL* o *GRUPAL*. Más arriba se presentó una definición formal de las condiciones.

4.5 COMANDO AGRUPAR (AG)

Este comando se puede ejecutar cuando se tiene seleccionada una expresión de relación.

agrupar *atributo+*
Descripción:
Permite agrupar las entidades por uno o una lista de atributos, los que deben pertenecer a la entidad que actualmente se esta definiendo.

4.6 COMANDO ORDENAR (OR)

Este comando se puede ejecutar cuando se tiene seleccionada una expresión de relación.

ordenar, (*atributo*, *TipoDeOrdenamiento*)*+*

Descripción:
Permite indicar el ordenamiento que se quiere para los datos de una entidad o de un reporte, se deberá indicar un conjunto de parejas *atributo TipoDeOrdenamiento*, donde *atributo* será el nombre de un atributo de la entidad actual y *TipoDeOrdenamiento* serán los valores ASCENDENTE o DESENDENTE para indicar el tipo de orden que se tomara en cuenta para ese atributo.

Si se indican dos o más atributos para ordenar, se ordenará primero por el primer nombre de atributo que aparezca y solo en el caso de haber repetición en los datos de la Entidad asociado a ese atributo, se toma en cuenta el segundo atributo y así sucesivamente para el resto de los atributos de la lista.

4.7 COMANDO EJECUTARENTIDAD

ordenar, (*atributo*, *TipoDeOrdenamiento*)*+*
EjecutarEntidad [*entidad*]

Descripción:
Genera una nueva ventana con los datos asociados a una entidad, pero estos datos solo son mostrados no se pueden modificar ni guardar ni imprimir. Si no se indica una *entidad* se muestra los datos de la última entidad seleccionada.

5 COMANDOS SOBRE ENTIDADES

5.1 COMANDO CREARENTIDAD (CREN)

crearEntidad *identificador*

Descripción:
Permite crear una entidad con el nombre indicado en *identificador*.

5.2 COMANDO MODIFICARENTIDAD (MODEN)

modificarEntidad *nombre , n:nuevo_nombre*

Descripción:
Permite renombre la entidad.

5.3 COMANDO ELIMINARENTIDAD (ELEN)

eliminarEntidad [*nombre*]

Descripción:
Eliminar la entidad indicada o la seleccionada actualmente quedando el esquema como objeto seleccionado.

6 COMANDOS SOBRE RELACIONES

Todos los comandos sobre relaciones se deben ejecutar teniendo seleccionada una entidad.

La *cardinalidad* podrá ser uno de los valores: *1a1*, *1aN*, *Na1* o *NaN*. El parámetro *tipo_relacion* podrá asumir solamente alguno de los siguientes valores: **generalización**, **especializacion, composicion** o **relacion** (*por defecto*). En la generalización la entidad seleccionada debe tener un subconjunto de los atributos de aquella entidad a la que generaliza, la cardinalidad deberá ser 1a1. En la especialización, la entidad seleccionada tendrá todos los atributos de la entidad a la que especializa más otros que surjan de relaciones con otras entidades, la cardinalidad también será 1a1. En el caso de la composición la relación será Na1 o 1a1 y no habrá restricciones en los atributos. En el caso de una relación común no existirán restricciones.

6.1 COMANDO CREARRELACION (CRRE)

crearRelacion *nombre, n:entidad_foranea,* al:*atributo_local* [, af: *atributo_foraneo*] [, c: *cardinalidad*] [, t: *tipo_relacion*]

Descripción:
Se podrán crear **relaciones** cuando el objeto seleccionado actualmente sea una entidad. La *entidad_foranea* indicará la entidad que se esté asociando a la entidad seleccionada. Los parámetros *atributo_local* y *atributo_foraneo* indicarán los atributos a través de los cuales se relacionan las entidades, en caso de que solo se indique un atributo este deberá existir en las dos entidades. Los parámetros *cardinalidad* y *tipo_relación* se explican más arriba.

6.2 COMANDO MODIFICARRELACION (CRDRE)

modificarRelacion *nombre*[*, n:entidad_foranea*][*,* al:*atributo_local*]] [, af: *atributo_foraneo*] [, c: *cardinalidad*] [, t: *tipo_relacion*]

Descripción:
Permite modificar la relación, los parámetros que no se incluyan no serán modificados. Los parámetros son similares a los de **crearRelacion**.

6.3 COMANDO ELIMINARRELACION (ELRE)

eliminarRelacion [*nombre_relacion*]

Descripción:
Elimina la relación indicada o la que esté seleccionada.

7 COMANDOS SOBRE EXPRESIONRELACION

Todos los comandos sobre expresionRelaciones se deben ejecutar teniendo seleccionada una entidad.

7.1 COMANDO CREAREXPRESIONRELACION (CREX)

crearExpresionRelacion *nombre* [*,* c:*condicion*]

Descripción:
Permite indicar la forma en que se calculan los datos de una entidad a partir de relaciones con otras entidades. Se indica un *nombre* que identificará a la expresión de relación y una condición a aplicar sobre cada registro de los datos de las entidades donde está definida la expresión de relación (condición individual).

7.2 COMANDO MODIFICAREXPRESIONRELACION (MODEX)

modificarExpresionRelacion *nombre* [, n:*nuevo_nombre*] [*,* c:*condicion*]

Descripción:
Permite modificar el nombre de la expresión de relación y la condición sino se indica ninguna condición se conservará la anterior, si se indica una condición vacía se eliminará la que exista, esta condición también se puede modificar a través del comando Condicion en el punto 3.5.

7.3 COMANDO ELIMINAREXPRESIONRELACION (ELEX)

eliminarExpresionRelacion [*nombre*]

Descripción:
Elimina la expresión de relación indicada o la que esté seleccionada, junto con todos los vínculos agrupaciones, ordenaciones o condiciones en ella definido.

8 COMANDOS SOBRE SERVICIOS

Todos los comandos sobre servicios se deben ejecutar teniendo seleccionada una entidad.

8.1 COMANDO CREARSERVICIO (CRSE)*

crearServicio *nombre* , *fuente_info* [, nombre_parametro:(calculo | condicion)]+

Descripción:
Crea un servicio con el nombre indicado, que se ejecuta en la fuente de datos *fuente_info*. Se puede indicar una lista de parámetros, de la forma: "nombre_parametro: (calculo | condicion)", con el valor que resulte de este cálculo o condición se invocará una operación de la fuente de datos.

8.2 COMANDO MODIFICARSERVICIO (MODSE)*

modificarServicio *nombre*[, n:*nuevo_nombre*][, f:*fuente_info*][, nombre_parametro:(calculo | condicion)]+

Descripción:
Permite modificar el nombre, la fuente de información o el tipo del servicio. Si ya existen parámetros se modifica su valor, si no existen se crean, el resto de los parámetros queda como está. Para eliminar un parámetro se debe pasar un valor nulo para ese parámetro, de la siguiente forma: "***,nombre_parametro:***", para el caso del *nuevo_nombre* o la *fuente_info* si no se ingresan queda el valor anterior.

8.3 COMANDO ELIMINARSERVICIO (ELSE)*

eliminarServicio *servicio*

Descripción:
Elimina el servicio de la entidad que se encuentre seleccionada.

9 COMANDOS SOBRE ATRIBUTOS

Estos comandos solo se pueden aplicar cuando se tiene seleccionada una entidad.

9.1 COMANDO CREARATRIBUTO (CRAT)

crearAtributo *nombre* , t:*tipo* [, m:*multiplicidad*] [, o:*oculto*] [, h:*horizontal*][, c:clave]

Descripción:
Crea un nuevo atributo en la entidad que se está trabajando. Se deberá indicar, además del nombre, el tipo del atributo, si el atributo es múltiple (por defecto no es múltiple), si es oculto (por defecto son visibles) y si el atributo es horizontal o no (por defecto no lo es). También se podrá indicar si el atributo es clave de la entidad (si es así, el valor del atributo debe ser único e identificatorio de la entidad).

Más abajo se listan los **tipos** con los cuales se podrán definir los atributos.

9.2 COMANDO MODIFICARATRIBUTO (MODAT)

modificarAtributo *nombre* [, n:*nombre_nuevo*] [, t:*tipo*] [, m:*multiplicidad*] [, o:*oculto*] [, h:horizontal][c:clave]

Descrpción:

Modifica un atributo de la entidad en la que se está trabajando. Se pueden cambiar el nombre, el tipo del atributo, si el atributo es múltiple (por defecto no es múltiple), si es oculto, y la horizontalidad. Los atributos que no sean indicados no se modificaran.

9.3 COMANDO ELIMINARATRIBUTO (ELAT)

eliminarAtributo *nombre*

Descripción:
Elimina el atributo indicado en la entidad actualmente seleccionada.

10 COMANDOS SOBRE REPORTES

10.1 COMANDO GENERARREPORTE (GERE)

generarReporte [*entidad*]

Descripción:
Genera un repote nuevo con la *entidad* seleccionada. Si no se indica una *entidad* se genera el reporte con la última entidad seleccionada.

11 COMANDO SOBRE PLANTILLAS DE FORMATO DE REPORTE

Las plantillas de formato de reporte nos permiten modificar la presentación del reportes, variando colores, letras, bordes, sombreados, etc.; el formato cambiando forma de mostrar los atributos y el ancho de las columnas; el estilo de reporte ya sea presentándolo en forma de tabla o matriz, o cambiar la configuración de la hoja donde se va a imprimir modificando el tamaño papel, encabezados y pie o márgenes.

Por defecto tendrá un plantilla (Normal.xml) la cual no se podrá eliminar además estará la posibilidad de seleccionar otras plantillas.

Las plantillas no estarán asociadas a un esquema en particular.

En cualquier momento podrá aplicar los siguientes comandos sin necesidad de tener un esquema abierto.

11.1 CREARPLANTILLAFORMATOREPORTE (CRPR)

crearPlantillaFormatoReporte *nombre, pathNuevoFormato*

Descripción:
Define una nueva plantilla de formato de reporte, con el *nombre* indicado y el formato de cada área del reporte lo obtiene *pathNuevoFormato* el cual indica el camino y nombre del archivo RDL que tiene el formato a copiar.

11.2 ELIMINARPLANTILLAFORMATOREPORTE (ELPR)

EliminarPlantillaFormatoReporte *nombre*

Descripción:
Elimina una plantilla de formato de reporte con el *nombre* indicado.

Si en una entidad se referencia a una plantilla que no esta presente se usará la plantilla Normal pero el resto de los valores (*altoHoja, anchoHoja, altoEncabezado, altoPie, fechaAlPie, numeroPaginaAlPie, altoHojasAlPie, textoEnEncabezado, margenIzquierdo, margenSuperior, margenDerecho, margenInferior*) no se modificarán.

12 COMANDO SOBRE FORMATO DE REPORTE

Estos comando se podrán aplicar cuando este seleccionado el formato de reporte de una entidad, y solo cambiará el diseño de reporte generado para esa entidad.

Toda entidad tendrá un formato asociado y si no se modificar el formato correspondiente será el formato definido como Normal.

Si un valor del formato no es modificado también asume el valor que se encuentra en la Plantilla Normal

Las unidades que se tomaran para los datos numéricos serán los centímetros (cm)

12.1 MODIFICARFORMATOREPORTE (MOFR)

ModificarFormatoReporte *nombrePlantilla, altoHoja, anchoHoja, altoEmcabezado,altoPie, fechaAlPie, numeroPaginaAlPie, totalHojasAlPie, textoEnEncabezado, margenIzquierdo, margenSuperior, margenDerecho, margenInferior, pPathImagenEncabezado*

Descripción:
Modifica el formato del reporte para la entidad actual, los valores que se pueden ingresar son: *nombrePlantilla* para modificar colores y letras, además de *altoHoja, anchoHoja,*

altoEncabezado, altoPie, margenIzquierdo, margenSuperior, margenDerecho, margenInferior para modificar los valores numéricos de estos datos, si uno de estos valor es menor a 0 toma el valor de la plantilla Normal.xml, *fechaAlPie* (muestra, en el pie de página, la fecha de generado del reporte)*, numeroPaginaAlPie* (muestra , en el pie de página, el número de cada una de las páginas del reporte)*, totalHojasAlPie* (muestra, en el pie de página, el total de páginas que tiene el reporte); estos son valores boléanos que determinan si se muestra esta información en el reporte o no; *textoEnEncabezado* es el texto que se mostrará en el encabezado de página.

Si los valores numéricos son menores a 0 no son modificados
Si los textos con null o "" no son modificados y para borrarlos se coloca un infragion "_" u otro valor.
Los valores booleanos se modifican siempre.

12.2 ModificarAnochoColumna (moAC)

ModificarAnchoColumna *nombreAtributo, anchoColumna*
Descripción:
Modifica el la presentación de la columna del reporte, para el atributo indicado en *nombreAtributo*, modificando el ancho de su columna con el valor indicado en *anchoColumna*

12.3 EliminarFormatoReporte (elFR)

EliminarFormatoReporte

Descripción:
Elimina el formato de reporte en la entidad actual, colocando como nuevo formato del reporte a la plantilla Normal.

13 Comando sobre Variables

Para aplicar estos comandos debe tener seleccionado una entidad que tenga definida alguna expresión de relación.

Solo se podrán agregar variables en las entidades relacionales que no estén referenciadas desde otras entidades, ya sea desde vínculos, cálculos, relaciones etc.

13.1 Comando crearVariable (crVa)*

crearVariable *nombre, tipo*

Descripción:
En la entidad seleccionada define una variable con el *nombre* indicado y del *tipo* indicado, el tipo podrá ser cualquiera de los tipos siguientes: "string", "boolean", "decimal", "float", "duration", "datetime", "time" , "date", "char", "int", "double" y "long"

13.2 Comando eliminarVariable (elVa)*

eliminarVariable *nombre*

Descripción:
En la entidad seleccionada elimina la variable con el nombre dado.

14 Definiciones Generales

En este capítulo se definirán varios elementos utilizados en la mayoría de los comandos. Como ser: los identificadores, urls, paths, cálculos, condiciones, entre otros. Muchas de las definiciones que aquí se presentan también se encuentran informalmente dentro de las

descripciones de los comandos en las que se usan. Pero se agrupan aquí para mayor claridad y formalismo.

14.1 IDENTIFICADORES

Los identificadores podrán contener letras, números, guiones y guiones bajos, empezando siempre con una letra:

```
ident = letter {letter | digit | '_' | '-'}
```

14.2 PATHS

Los paths deberán cumplir con la siguiente regla:

```
path =  ["./" | "../" | ident] {"/" ("./" | "../" | ident) }.
```

14.3 URLS

Las urls cumplirán con la siguiente regla:

```
url-spec = "http://" [ host [path]]
host = (ident | entero) [.(ident | entero) [.(ident | entero) [.(ident
<| entero)]]]
entero = {digit}
```

14.4 CÁLCULOS

14.5 CONDICIONES

La base de la siguiente gramática de condiciones es la gramática de Pascal

```
expression = simple-expression [ relational-operator simple-expression
].

simple-expression = [ +|- ] term { addition-operator term }.

term = factor { multiplication-operator factor }.

factor =
      variable | number | string | set | function-designator | "("
expression ")" | not factor.

relational-operator = "=" | "<>" | "<" | "<=" | ">" | ">=" | "in".

addition-operator = "+" | "-" | or.

multiplication-operator = "*" | "/" | div | mod | and.

variable = entire-variable | component-variable.

entire-variable = identifier | identifier.

component-variable = indexed-variable | field-designator.

indexed-variable = array-variable "[ " expression-list " ]".

field-designator = record-variable "." field-identifier.

set = "[ " expression-list " ]".

expression-list = [ expression { "," expression } ].

function-designator = function-identifier [ actual-parameter-list ].
```

number = integer-number | real-number.

integer-number = ["+" | "-"] digito { digito }.

real-number = integer-number "." {digito} [scale-factor] | integer-number [scale-factor] .

scale-factor = ("E" | "e") ["+" | "-"] { digito }.

string = "'" {string-char} "'"

function-identifier = div | idiv | mod | floor | ceiling | round | concat | len | starts-with *string | ends-with string*| contenido like *string*|substring|Trim(expresion) | upper-case | lower-case | Convert(expression, type) | count | min | max | avg | sum | datediff | dateadd | distinct-values <>| ISNULL(expression, replacementvalue) | exists| IIF(expr, truepart, falsepart).

actual-parameter-list = "(" actual-parameter { "," actual-parameter } ")".

actual-parameter = expression.

14.6 CARDINALIDAD

Las cardinalidades de las relaciones podrán tener la siguiente forma:
Cardinalidad = entero "a" entero | entero "aN" | "Na" entero | "NaN"

14.7 TIPO DE RELACIÓN

Los tipos de relación serán: "generalización", "especialización", "composición" o "relación"

14.8 TIPO OBJETIVO

Los tipos sobre los que se podrán aplicar los comandos serán:
"esquema" | "entidad"| "entidadservida" | "entidadreporte" | "relacion" | "atributo" | "expRelacion" | "servicio" | "vinculo" | "fuenteinformación" | "variables"

14.9 OTRAS OPERACIONES OFRECIDAS

14.9.1 ObtenerDatosObjetoSeleccionado

Esta función permite obtener el nombre y el tipo de datos del objeto seleccionado actualmente

14.9.2 Actualizado

Permite determinar si la persistencia del esquema esta actualizada o si se han realizado cambios los cuales no han sido guardados.

14.9.3 ObtenerXMLEsquema

Permite obtener un esquema XML representación del modelo de negocio que esta actualmente abierto.

14.9.4 SetLogOn

Permite indicar que se debe mantener un log con los errores que surgen de la ejecución de los comandos.

15 ACLARACIÓN SOBRE LAS FUNCIONES Y OPERADORES

A todas las funciones se les debe pasar los parámetros a operar entre paréntesis y separado con comas entre un parámetro y otro.
Las siguiente es una lista de las funciones reconocidas por el editor del modelo de negocio:

- Operaciones aritméticas
  `+, -, *, mod.`
- Operadores relacionales
  `=, !=, <, >, <=, >=, not()`
- Operadores lógicos
  `and, or, not()`
- Funciones de cadena
  `+ (concatenación), substring(), string()`
- Funciones agregadas
  `count(), min(), max(), avg(), sum()`
- Funciones de fecha:
  `datediff()`
  `dateadd()`
- Uso general
  `distinct-values(), empty(), exists()`

VISUAL REPORTING SERVICES

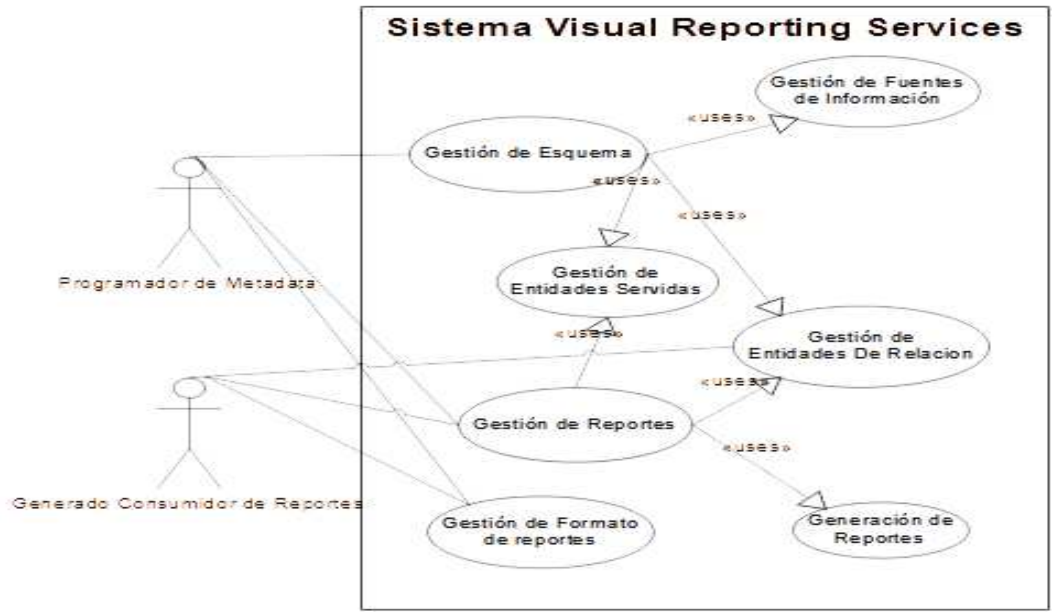
ANEXO IV

DESCRIPCIÓN DE LA ARQUITECTURA

ÍNDICE

1 VISTA DEL MODELO DE CASOS DE USO

1.1 DIAGRAMA DE LOS CASOS DE USO RELEVANTES A LA ARQUITECTURA:



1.2 CASOS DE USO RELEVANTES A LA ARQUITECTURA

1.2.1 Gestión de Esquema

1.2.1.1 Actor:

Programador de Metadata (Iniciador)

1.2.1.2 Descripción:

El programador de metadata puede agregar Esquemas nuevos, ingresando la información requerida, el sistema debe validar que el nombre no este repetido. También es posible modificar sus entidades o eliminarlo.

1.2.2 Gestión de Fuente de Información

1.2.2.1 Actor:

Programador de Metadata (Iniciador)

1.2.2.2 Descripción:

El programador de metadata puede agregar fuentes de información nuevas, ingresando la información requerida, el sistema debe validar que el nombre no este repetido. También es posible modificar algunas de sus características o eliminar una fuente de información si aún no tiene ninguna entidad relacionada.

1.2.3 Gestión de Entidades Servidas (serviced)

1.2.3.1 Actor:

Programador de Metadata (Iniciador)

1.2.3.2 Descripción:

El Programador de Metadata puede agregar nuevas entidades servidas, ingresando la información requerida, el sistema debe validar que el nombre no este repetido así como el tipo de datos de los atributos y la existencia de las fuentes de información. También es posible modificar algunas de sus características o eliminar una Entidad si aún no tiene ninguna Entidad relacionada.

1.2.4 Gestión de Entidades Relacionales

1.2.4.1 Actor:

Programador de Metadata, Generador/Consumidor Reportes (Iniciadores)

1.2.4.2 Descripción:

El actor puede agregar entidades nuevas, ingresando la información requerida, el sistema debe validar que el nombre no este repetido así como el tipo de datos de los atributos y la existencia de las fuentes de información. También es posible modificar algunas de sus características o eliminar una Entidad si aún no tiene ninguna Entidad relacionada.

1.2.5 Gestión de Entidades Reportes

1.2.5.1 Actor:

Generador/Consumidor de Reportes (Iniciador)

1.2.5.2 Descripción:

El Generador/Consumidor de Reportes puede agregar Entidades Reportes nuevos, ingresando la información requerida, el sistema debe validar que el nombre no este repetido. También es posible modificar sus reportes o eliminarlo.

1.2.6 Generación de reportes

1.2.6.1 Actor:

Generador/Consumidor de Reportes (Iniciador)

1.2.6.2 Descripción:

El usuario puede generar reportes a partir de entidades de su esquema de metadata. Luego de haber generado una entidad puede crear un reporte para visualizar los datos que están asociados a esa entidad.

1.2.7 Gestión de Formato de Reportes

1.2.7.1 Actor:

Generador/Consumidor de Reportes (Iniciador)

1.2.7.2 Descripción:

En un Esquema abierto el usuario Generador/Consumidor de Reportes puede administrar los Formatos de Reportes, usando el Editor de Modelo de Negocio. Los Formatos de Reportes nuevos requieren el ingreso de un nombre único y un archivo que contendrá el formato del reporte. También podrá cambiar algunos valores que están por defecto en el formato de reporte, como ser ancho de las columnas tamaño de los márgenes y encabezado y pie o incluso podrá ingresarse una imagen para usarse como logotipo de los reportes. Además se podrá realizar bajas y modificaciones de los Formatos de los Reportes ya existente en el Esquema abierto.

1.2.7.3 Pre-Condiciones:

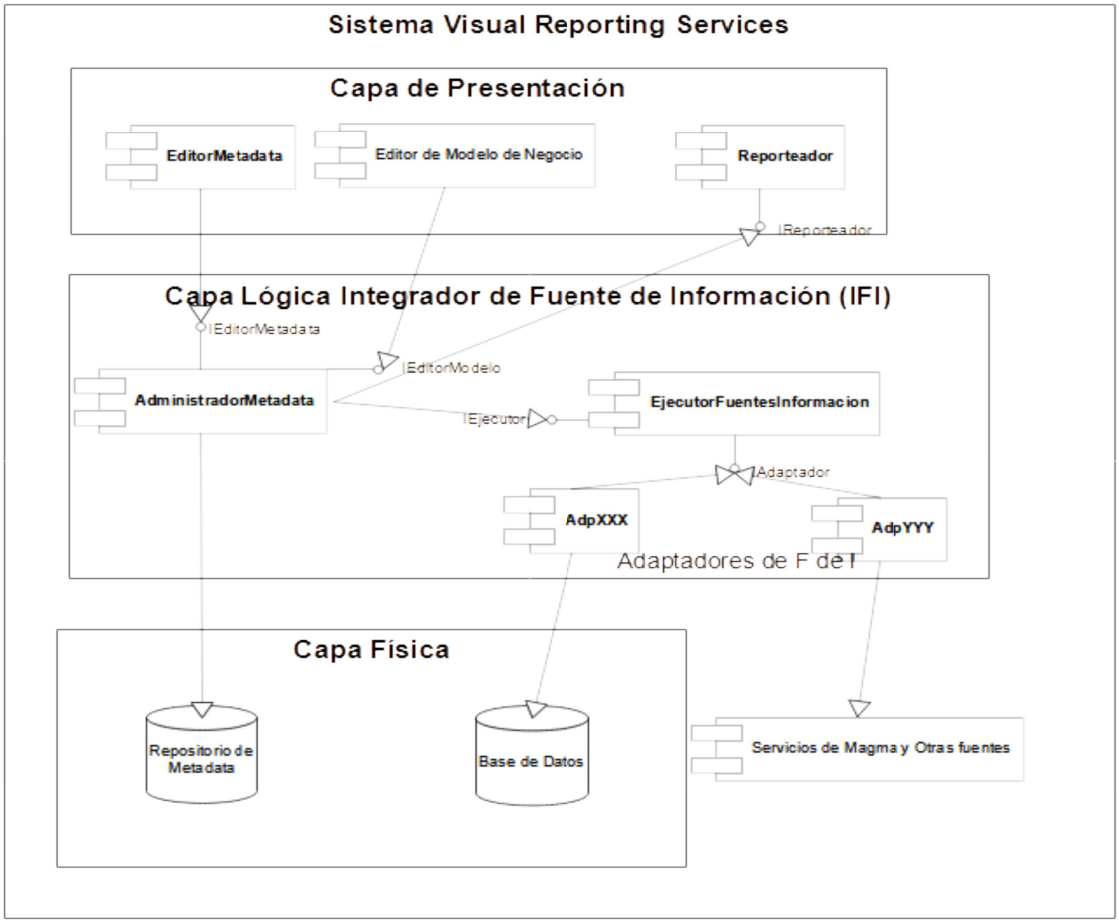
Tener un Esquema abierto.

1.2.7.4 Pos-Condiciones:

Permitir la administración del Formato del Reportes

2 VISTA DEL MODELO DE DISEÑO

2.1 DESCOMPOSICIÓN EN SUBSISTEMAS:



2.1.1 Descripción del Sistema Visual Reporting Services

El sistema está diseñado en tres capas (presentación, lógica y persistencia) y se presentan en el diagrama de más arriba, junto con los componentes que existen en cada una de las capas.

La capa de presentación se compondrá por el Editor de Metadata, Editor de Modelo de Negocio y Reporteadores.

La capa lógica estará formada por el Integrador de Fuentes de Información, la cual estará formada por el Administrador de Metadata y el Ejecutor de Fuentes de Información.

En la capa de persistencia se presentan, las fuentes de información, es decir los sistemas de los cuales se obtienen los datos de las Entidades que podrían ser Bases de Datos, archivos con datos en formato XML, servicios de Magma u otras fuentes. Por otro lado, se tienen el Repositorio de Metadata que permite mantener la persistencia de cada modelo de negocio.

2.2 CAPA PRESENTACIÓN

2.2.1 Capa Presentación - Editor de Modelo de Negocio

Esta interfaz se encargará de permitir editar el modelo de negocio al usuario Generador/Consumidor de Reportes. Sobre un modelo creado anteriormente puede editar algunos conceptos del negocio o configurar y generando reportes sobre cualquier concepto definido previamente.

Este editor trabajará sobre un modelo del negocio establecido por el Programador de Metadata (utilizando el Editor de Metadata que se describe más abajo). Esto se hará mediante un subconjunto del Lenguaje de Edición de Metadata (LEM), el que se especifica en un documento aparte (5b - Lenguaje de Edición de Metadata.doc). Este editor se comunicará con el Administrador de Metadata por mediante la interfaz IEditorModelo.

2.2.2 Capa Presentación - Editor de Metadata

Este componente permitirá crear, al Programador de Metadata, repositorios de metadata, modificar el contenido de los repositorios. El contenido que podrá ingresar el Programador de Metadata serán las formas en que se mapean las fuentes de información, los conceptos de negocio con sus descripciones, los formatos de los reportes y definir las formas de acceder a las Fuentes de Información.
En este editor se utilizará el Lenguaje de Edición de Metadata (LEM) en su totalidad, y se comunica con el Administrador de Metadata mediante la interfaz IEditorMetadata.

2.2.3 Capa Presentación – Reporteador

Se independiza el reporteador utilizado del resto del sistema mediante la utilización de una representación de datos y una especificación del formato de los reportes independiente de los reporteadores. Con la especificación del formato del reporte y la definición del concepto a reportear, a través de un adaptador de reporteadores se creará la definición del reporte para un reporteador en particular. El reporteador recibe los datos a presentar y la definición del reporte, generando el reporte especificado.

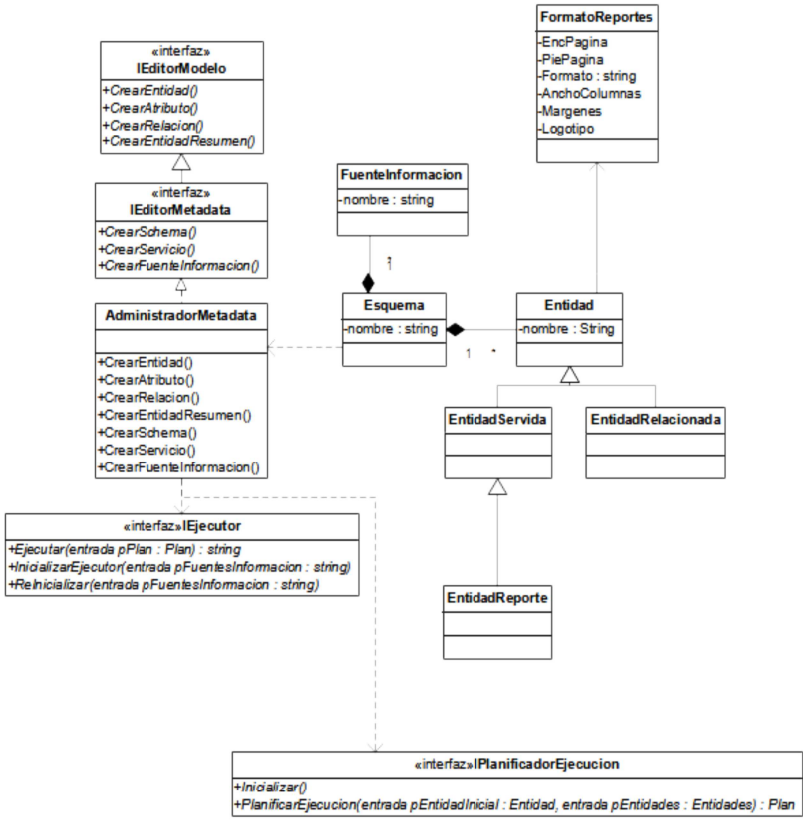
En nuestro caso utilizaremos Reporting Services como reporteador que permite, a través del RDL, definir el diseño del reporte. La integración con otros generadores de reportes dependerá de las APIs que el reporteador a utilizar presente.

2.3 CAPA LÓGICA - INTEGRADOR DE FUENTES DE INFORMACIÓN (IFI)

El IFI es la capa que vincula a los usuarios con la persistencia de los modelos de negocio y las fuentes de información. Está formado por varios componentes que nos permite procesar las solicitudes provenientes de la interfaz, cada solicitud llevará a la ejecución de operaciones sobre la metadata y en el caso que la solicitud fuere la generación de un reporte estas operaciones se extenderá hasta las mismas fuentes de información y al reporteador.

2.3.1 Capa Lógica - Administrador de Metadata

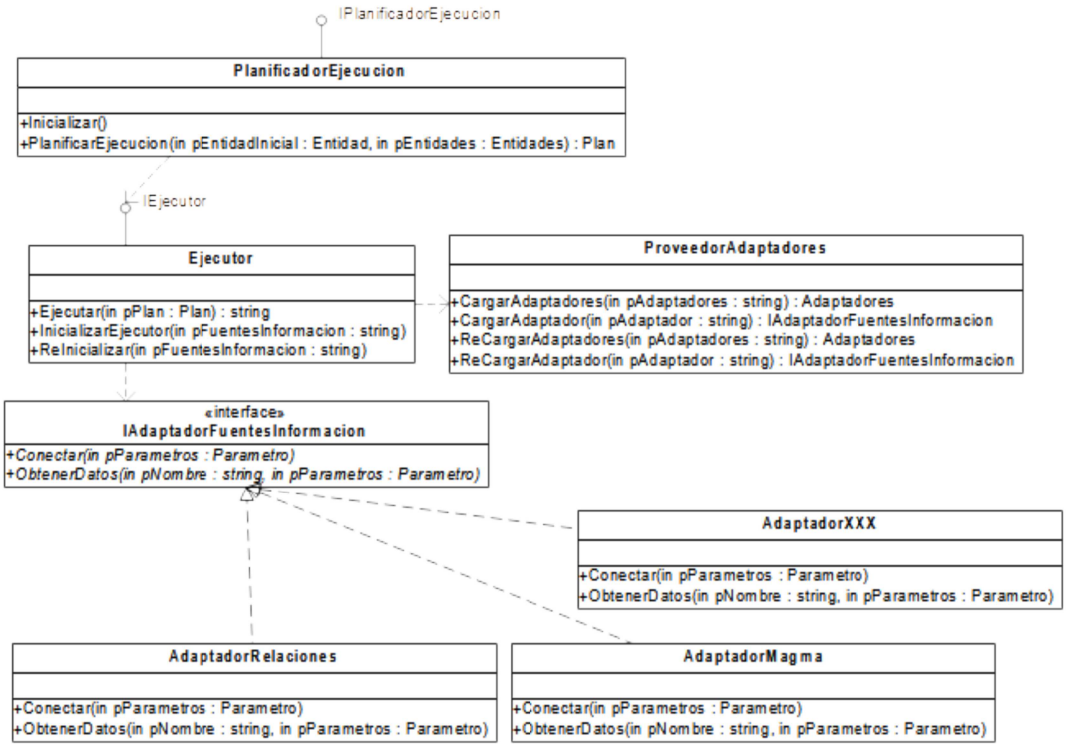
El Administrador de Metadata, encapsulará la representación del Modelo de Negocio, encargándose de editar la representación del modelo según las operaciones que se realicen desde la capa de presentación. También actúa en la generación de un reporte cuando esta sea solicitada. Para esto utilizará el Ejecutor de Fuentes de Información (EFI), al que se le solicitarán los datos para el reporte a generar. Luego estos datos serán enviados al reporteador junto con el diseño del reporte el cual proviene del adaptador de reporte.
Se utilizará una estructura intermedia para la representación del Modelo de Negocio, en el siguiente diagrama se muestra solo parte de esta estructura, junto con el la estructura del Administrador de Metadata Cabe aclarar que en las interfaces IEditorMetadata e IEditorModelo no se presentan todos los métodos que en ellas se definen.



2.3.2 Capa Lógica - Ejecutor de Fuentes de Información (EFI)

El Ejecutor de Fuentes de Información se encarga de la ejecución de los *servicios* y *macroservicios* a través de la invocación de los adaptadores correspondientes.
Para la ejecución de un servicio se utilizará el adaptador específico para la fuente de información en la que se debe ejecutar el servicio. Al ejecutar un macroservicio, se utilizarán los

adaptadores correspondientes para cada servicio y luego las relaciones entre estos se ejecutarán con el *Adaptador de Relaciones*.
Agregar una nueva Fuente de Información requiere agregar un nuevo Adaptador, el que permitirá: acceder a la fuente de información de la misma forma que al resto, y; obtener los datos en el formato común a todas las fuentes. Para utilizar esta nueva Fuente de Información solo será necesario agregarla a través de la interfaz del IEditorMetadata con los parámetros adecuados.
El Ejecutor accederá a los adaptadores mediante el ProveedorAdaptadores, este proveedor permitirá acceder a los adaptadores sin importar la forma en que estos sean implementados.
El ProveedorAdaptadores se encarga de obtener instancias o referencias (ya sean referencias web, a objetos remotos o a DLL o a cualquier otro tipo) a los adaptadores de las fuentes de información existentes.



2.3.2.1 Adaptadores de Fuentes de Información

Cada adaptador permitirá abstraer la ejecución de los servicios de su implementación, retornando los datos en un formato común.

2.3.2.2 Adaptador de relaciones

Un adaptador particular que se utilizará será el Adaptador de Relaciones. Este se encargará de procesar las entidades cuyos datos no surjan directamente de un servicio, y que puedan implicar operaciones con los datos generados por otros adaptadores, este adaptador estará definido en el Ejecutor.

2.3.3 Capa Lógica - Adaptadores para los Reporteadores

Estos módulos permitirán utilizar los generadores de reportes a partir de la especificación del reporte provista por el Administrador de Metadata, generando un diseño de reporte para el reporteador que corresponda.
Los datos serán provistos por el administrador de metadatos con un formato común para todas las Fuentes de Información, junto con la especificación que deberá mostrar el reporte.

2.4 CAPA DE PERSISTENCIA

2.4.1 Repositorio de Metadata

En el repositorio de metadatos se tendrá la información de los Modelos de Negocio de los clientes. Cada modelo estará representado por un Esquema, el que estará compuesto por las ***fuentes de información, entidades y formatos de reportes***. Por más detalles de la estructura del esquema ver el documento de Estructura Intermedia (6 - Representacion Intermedia.doc) o el documento que hace referencia a la persistencia de la representación intermedia (7 - Discusión de la metadata.doc).

2.4.2 Sistemas Externos

Los sistemas externos serán las fuentes de información de los modelos que se representan. Como se dice más arriba a estos sistemas se accede mediante los respectivos adaptadores. Estos sistemas podrán ser e distintos tipos: componentes COM+, servicios web, servicios de remoting, etc.

2.4.3 Base de Datos

Dentro de los sistemas externos antes mencionados podrían existir bases de datos. El acceso a estos servidores de bases de datos se maneja en los adaptadores.

3 JUSTIFICACIÓN DEL DISEÑO

Para diseñar esta solución nos apoyamos en las entrevistas con el cliente y algunos trabajos similares. De las entrevistas surgió la necesidad de poder acceder a distintas fuentes de información, y además que el acceso a ellas sea configurable. También interesaba poder utilizar distintos generadores de reportes.
Ante estas necesidades nos planteamos una arquitectura de "Adaptadores" y "Mediadores", en donde los Adaptadores cumplen la funcionalidad de expones información en un cierto formato y los Mediadores se encargan de realizar la integración y manipulan los datos.

Para el caso de las fuentes de información se tendrá un adaptador para cada una de estas fuentes, donde el adaptador será el encargado de interactuar con la fuente que adapte. El adaptador será el que, ante un pedido del Ejecutor, solicite la información a la fuente de datos. Los datos que se obtengan en respuesta a estas solicitudes deberán ser en el formato común utilizado. El formato utilizado será el de un DataSet de .NET escrito en XML con la opción WriteSchema.
Además se pensó en un adaptador particular, llamado Adaptador de Relaciones, el cual está integrado al Ejecutor de Fuentes de Información. Este adaptador será utilizado para calcular las relaciones entre las entidades del modelo, así como las operaciones que halla con los datos obtenidos de otros adaptadores.
Para permitir tanta libertad como fuese posible, a los adaptadores se accederá mediante un proveedor de adaptadores. Mediante este proveedor de adaptadores el adaptador será accedido de igual manera, ya sea que está implementado en un Web Service, un objeto publicado por remoting, una dll local u otra forma.
Para poder decidir que servicios de que adaptador se va a requerir, es que se centra la toma de decisiones en un "Mediador", nuestro mediador será el Ejecutor que se encuentra dentro del EFI. Este Ejecutor deberá ser configurable durante la ejecución, de forma que se puedan agregar o quitar fuentes de información en tiempo de ejecución.
Como también está la posibilidad de usar diferentes reporteadores, es que se pensó en un manejo similar, al anterior, la generación de reportes. En este caso se debe agregar un Adaptado para cada reporteador que se quiera integrar al sistema, este adaptador recibirá del Administrador de Metadata los datos y algunas especificaciones del formato del reporte, el Adaptadores de Reportes con la información recibida genera un diseño de reporte el cual se lo pasa al Reporteador junto con los datos, este ultimo deberá genera el reporte con el diseño indicado y los datos indicado.

Mediante lo antes descrito se solucionan a alto nivel las principales dificultades que se presentan para la realización de los objetivos. Los detalles de más bajo nivel se describirán en documentos específicos para cada punto importante, como ser: el lenguaje de edición de metadata, la metadata, la representación intermedia y el acceso e integración de las fuentes de información.

VISUAL REPORTING SERVICES

ANEXO V

REPRESENTACIÓN INTERMEDIA

ÍNDICE

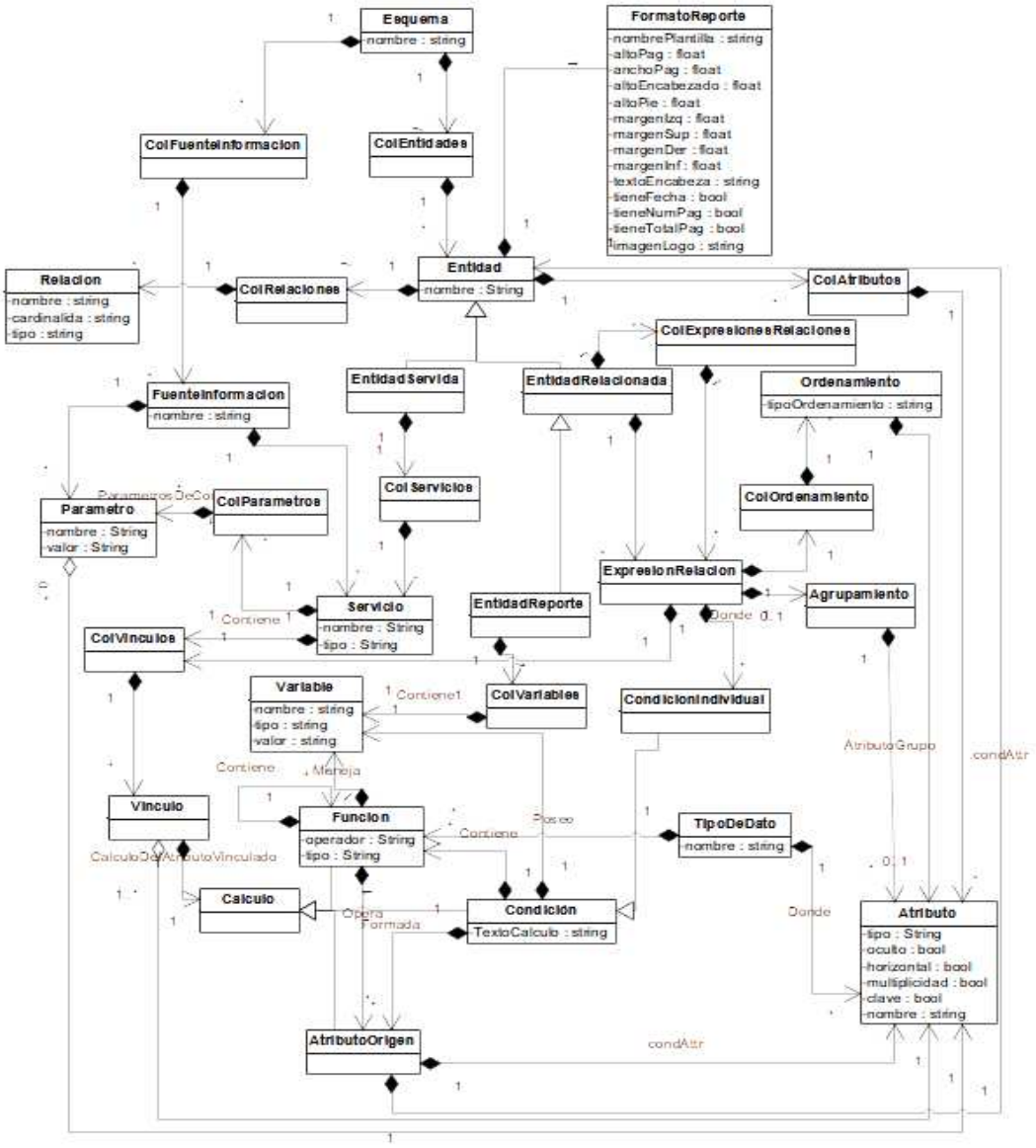
1 Introducción

Este documento tiene como objetivo describir la estructura que permite almacenar la metadata para realizar la representación del negocio y la generación de reportes.

A esta metadata le llamamos representación intermedia y su persistencia se realizara en un archivo XML el cual es descripción en el documento **4 - Meta-Modelo de Negocio.doc**.

Un punto importante en las descripción de cada una de estas clase es la existencia del atributo ID, el cual es de uso interno, es decir que ningún usuario percibe la existencia de este atributo. Este atributo es el único que identifica a cada objeto de una clase y no puede ser modificado, de esta forma se simplifica mantener relaciones entre objetos y modificar el valor de cualquier atributo.

El siguiente diagrama muestra como se relacionan las diferentes clases y cuales son sus principales atributos, más adelante se describe que modela cada clase y que atributos contiene.



2 Clase Esquema

Esta es la clase que engloba toda la estructura, representa el modelo del negocio de usuario. Estará compuesta de una colección de entidades una colección de fuentes de información y por los tipos de datos que se podrán usar. Los esquemas serán identificados por su nombre que coincide con el nombre del archivo más la extensión XML donde es persistido.

3 Clase FuentesInformación

Cada FuenteInformacion contendrán los datos necesarios para interoperar con la fuente misma de información, la metadata que esta mantendrá será: el nombre de la fuente de información el cual será único y una serie de parámetros necesarios para que el adaptador establezca la conexión, estos parámetros variarán si la fuente de información es un Web Services o un servicio de Remoting o una DLL.

4 Clase Entidad

La clase Entidad modela los conceptos de negocios que el usuario usa habitualmente en su organización, tendrá un nombre que lo identifica, una lista de atributos y otra de relaciones con otras entidades del modelo. Las clases EntidadRelacion, EntidadReporte y EntidadServida son subclases de ésta. Las entidades tendrán una forma de obtener sus datos, las subclases se encargaran de vincular la forma de obtener los valores con cada atributo. Además en la entidad se puede mantener el formato del reporte es decir algunas características de cómo se van a mostrar esos datos a la hora de generar el reporte con esa entidad.

5 Clase EntidadRelacional

Esta clase hereda de Entidad, representará aquellas entidades que obtengan sus datos a partir de relaciones con otras entidades del modelo. Por esto contendrán una lista de objetos ExpresionRelacion que serán los encargados de definir la forma de obtener los datos de la Entidad.

6 Clase EntidadReporte

Estas clases modelan aquellos conceptos del negocio que requieren de un valor diferente cada vez que se quiera generar un reporte, como por ejemplo el código del cliente para ver los movimientos de un cliente.

Esta clase hereda de EntidadRelacional, el aporte de estas clase a la clase EntidadRelacional es el la posibilidad de definir un conjunto de variables.

7 Clase EntidadServida

Esta clase representará a las entidades cuyos datos se obtengan mediante la ínteroperación con alguna fuente de información. Por lo que agregaran a la entidad básica uno o varios objetos Servicio.

8 Clase Atributo

Los atributos representan la información relevante de cada concepto del negocio, por lo cual una colección de la clase Atributo será parte de una Entidad. Tendrá: un nombre, que la identificara dentro de la entidad en la que se encuentre; un tipo de dato en el cual se encuentran todos los valores asociados a este atributo; además tendrá indicadores de: visibilidad del atributo para indicar si será mostrado o no en el reporte; multiplicidad que indicará si el atributo puede tener varios valores o no, horizontal determina si el los

datos del atributo van a ser mostrados en una columna o una fila y un indicador que determina si el atributo es clave en esa entidad o no.

9 Clase Relación

La clase Relación representara las relaciones entre los conceptos del negocio es decir entre las entidades. Por lo que contendrá un nombre de relación, que es único dentro de la entidad; una referencia de la entidad foránea relacionada con la entidad local, los atributos por los cuales están relacionados tanto el local como el foráneo, un tipo que podrá ser RelacionSimple, especialización, generalización y una cardinalidad que podrá ser 1a1 1aN Na1 y NaN.

10 Clase ExpresionRelacion

Con esta clase se representarán las expresiones que determinan como se calcularán los datos asociados a la entidad, estos cálculos pueden contener Ordenamientos Agrupaciones y Condiciones Individuales además de vínculos de cada atributo con la forma de calcular los datos asociados a cada atributo.

11 Clase Agrupamiento

Permite modelar la forma en que se van agrupar los dato de la entidad local, esta clase define una colección de referencia de Atributos locales manteniendo el orden con que fueron ingresados

12 Clase Ordenamiento

Modela la forma en que se van a ordenar los dato de una entidad, al igual que la clase Agrupamiento esta clase tiene una colección de referencia de Atributos de la entidad actual manteniendo el orden con que fueron ingresados.

13 Clase Condición

Esta clase participan de la definición de aquellos conceptos que quieren mostrar algunos datos (dependiendo de una condición) de otro concepto definido previamente.
La clase Condición será una clase que compondrá las ExpresionRelacion.

14 Clase Servicio

Esta clase representara una forma de acceder a un servicio de la fuente de información, esto se hará con un identificador de la fuente de información el nombre del servicios y los parámetros necesarios para ejecutar la invocación al servicio.

15 Clase Parámetro

Esta clase representa un parámetro pasado a un servicio. Se tendrá un nombre y un valor asociado con el nombre del parámetro.

16 Clase Vinculo

La clase Vínculo permitirá mapear un atributo de una entidad con la forma de calcular los datos relacionados a ese atributo, este cálculo puede ser el resultado de la ejecución de un Servicio o una ExpresionRelacion.

17 Interfase ICalculo

Determina cuales son las operaciones necesarias para cualquier clase que vaya a implementar o participar de un cálculo. Estas operaciones son Tipo que determina el tipo de dato relacionado con el calculo, ToString que retorna un texto con el calculo a

realizar y TipoCalculo que indica que operación se esta realizando si es un atributo una función o una condición.

18 Clase Calculo

La clase Calculo es una clase base para todas aquellas clases que sean o participen de un cálculo, es decir aquellas clases que puedan participar de una operación o función definida en un vínculo o condición. Las clases que heredan de Calculo son: Función, AtributoOrigen, y Condición.

19 Clase Función

Representa una función u operador relacionado con un cálculo para el vínculo de un atributo o una condición de la Expresión.

20 Clase AtributoOrigen

En una entidad puede haber una serie de cálculos donde participan atributos de la entidad local o de entidades foránea, AtributoOrigen permite mantener un Atributo y la Entidad foránea que es originario el atributo.

21 Clase Variable

La clase Variable permite modelar información que se va a ingresar en el momento de generar el Reporte y que puede variar de un reporte a otro, como por ejemplo fechas limites o el código de un cliente.

22 Clase FormatoReporte

Esta clase permite modelar algunos datos de cómo se va a visualizar la información en el momento de generar un reporte con los datos relacionado a cierta entidad.

23 Clases Colecciones

También existen una serie de clases que comienzan con la palabra Col, estas clases permiten mantener colecciones de algún tipo de objetos y realizar las operaciones elementales de agrega eliminar buscar modificar y recorrer.
Un ejemplo de este tipo de clase es ColEntidades la cual es una colección de objetos del tipo Entidad.

VISUAL REPORTING SERVICES

ANEXO VI

MANUAL DE PROGRAMADOR

ÍNDICE

1 INTRODUCCIÓN

Visual Reporting Services (VRS), a través de sus editores, permite la generación de modelos de negocios por medio de la representación de conceptos, y vinculándolos con los datos operacionales de la organización da la posibilidad de generar reportes con el fin de apoyar la toma de decisiones.

Cabe destacar que el objetivo de este documento es indicar la forma de instalar y configurar los editores, además de mostrar como utilizarlos; no es el objetivo detallar los comandos que se pueden aplicar. Ejemplos y detalles del lenguaje de comandos se pueden encontrar en el documento **5 - Lenguaje de Edición de Metadata.doc.**

Este manual presenta las tareas que puede realizar el usuario programador o experto (UE) a través de VRS, que son:

- Instalación de los Editores,
- Configuración de los Editores y el Entorno,
- Instalación y configuración de los ejemplos disponibles,
- Uso del Editor de Metadata y puntos a tener en cuenta,
- Generación de Reportes.

1.1 USUARIO DE VRS

Para la generación de estos modelos de negocio se requiere la participación de dos tipos de usuarios uno que conozca a fondo todos los conceptos del negocio y otro usuario que maneje algunos conceptos básicos del negocio pero que tenga amplios conocimientos de los datos operacionales de la organización, estos usuarios son:

- Usuario Generador y Consumidor de Reportes o Usuario Final (UF),
- Usuario Programador de Metadata o Usuario Experto (UE).

El UF es quien conoce todos los conceptos del negocio que quiere modelar y es el mayor consumidor de reportes, pero no tiene todos los conocimientos para instalar los editores, ni la información de donde provienen los datos que el desea incluir en sus reportes.

El UE es quien debe saber donde se encuentran instalados los productos que permiten la ejecución de los editores, además conoce el acceso a los adaptadores de las fuentes de información (FI) y las funcionalidades que estos ofrecen, también tiene conocimientos de algunos de los conceptos de negocio fuertemente ligados a las FI.

Las tareas fundamentales de este usuario son:

- Instalar el o los editores a utilizar.
- Configurar los editores para el correcto funcionamiento.
- Definir los conceptos base para que el UF pueda interactuar con el sistema sin tener conocimientos de las FI de donde provienen los datos.

Cualquiera de los dos usuarios podrá generar reportes o modificar la representación de un modelo de negocio en particular, pero el usuario final tendrá más restricciones debido al poco conocimiento que este tiene de las FI y sus adaptadores.

Es por ello que cada usuario tendrá una interfase diferente con el sistema, a estas interfaces las llamaremos *Editor de Metadata* que será el ambiente usado por el UE y *Editor de Modelo de Negocio* que será la interfase del UF.

La interacción de cualquiera de los dos usuarios con el sistema se hará a través de comandos ingresados en su respectivo editor, donde los comandos aceptados por el Editor de Modelo de Negocio son un subconjunto de comandos aceptados por el Editor de Metadata.

2 INSTALACIÓN DE LOS EDITORES

En este punto se describen cuales son los requerimientos para la instalación de VRS, juntos con los pasos que se deben seguir para la instalación y la ejecución de los ejemplos entregados. También se describe la instalación de una extensión de las fuentes de datos de MS Reporting Services (MS RS), la cual es requerida para la correcta generación de reportes a través de los Editores.

Los puntos a tratar son:
- **Prerrequisitos**: donde se indica cuales son los requerimientos de hardware y de software para instalar el sistema.
- **Pasos de Instalación**: donde se comenta como se deben instalar los editores, se comenta como se instala y la extensión de RS requerida y los ejemplos.
- **Configuración**: donde se indica cuales son las variables de configuración de los editores y que información se requiere para poder configurar correctamente el ambiente de trabajo.

2.1 PRERREQUISITOS

Los requerimientos aquí establecidos para la ejecución del sistema corresponden al equipo de menor potencia en el que fue probado, no son los requerimientos mínimos necesarios, pero con ellos se asegura un correcto funcionamiento de la solución. Además se debe notar que alguno de los productos de software nombrados pueden tener mayores requerimientos de hardware o software según la versión utilizada. Otro punto que se debe tener en cuenta para el funcionamiento adecuado son los adaptadores de las fuentes de información y las fuentes mismas, que afectarán el funcionamiento del sistema según el volumen de datos que deban manejar y las tecnologías de los adaptadores. Estas variables pueden afectar el desempeño de la aplicación. No es lo mismo manejar 10.000 que 100.000 registros, o acceder a una FI a través de una DLL o de un WS.

2.1.1 Hardware

El sistema fue probado, con un funcionamiento aceptable en una PC con:
- 256 MB de RAM,
- Procesador de 1000 MHz
- Espacio en el disco solo para la aplicación Visual Reporting Services 5 MB

2.1.2 Software

En cuanto al software requerido para el correcto funcionamiento es
- MS Windows 2000 Server o posterior.
- .Net Framework 1.1
- MS SQL Server 2000.
- MS SQL Server 2000 Reporting Services
- ModificarConfiguracionRSExtenderDataSet, este producto forma parte del paquete entregado con Visual Reporting Services, y se describe su instalación en el punto Modificar Configuración de RS para Extender DataSet
- Adaptadores que permitan acceder a los datos de las Fuentes de Información

2.2 PASOS DE INSTALACIÓN

Como ya se indicó en los requerimientos los editores de VRS se ejecutan sobre una versión de MS Windows 2000 Server, pero además requiere tener actualizado los Service Pack y tener una versión actualizada de Internet Explorer.

Como el producto fue desarrollado con MS Visual Studio 2003 el primer paso es instalar .Net Framework 1.1 y el último Service Pack de este producto.

Luego de tener el Framework instalado se debe agregar el generador de reportes (reporteador), para este proyecto como reporteador se utilizo MS SQL Server 2000 Reporting Services con el Service Pack 2, pero para el funcionamiento de este producto se requiere tener instalado una versión de MS SQL Server 2000 con el último Service Pack 4.

El último paso es instalar los editores de Visual Reporting Services, utilizando el **instalador** provisto. Esto también instala un producto exclusivamente generado para trabajar con RS llamado ModificarConfiguracionRSExtenderDataSet (o ExtenderOrigenDataSet), que es una extensión de RS para permitir la utilización de DataSet como origen de datos.

En caso de no tener toda la información requerida en el momento de la instalación se puede ejecutar posteriormente el siguiente archivo:

ModificarConfiguracionRSExtenderDataSet.exe

En el siguiente punto **Modificar La Configuración De RS Para Extender DataSet** se indica cuales son los requerimientos necesarios para instalar este producto.

En resumen los pasos a realizar para instalar el producto y ejecutar los ejemplos son:

a. En una maquina con una versión de Windows 2000 o posterior, actualizar los Services Pack de Windows y de Internet Explorer
b. Instalar .net Framework 1.1 y su último Service Pack
c. Instalar MS SQL Services 2000 y su último Service Pack
d. Instalar MS SQL Services 2000 Reporting Services y su último Service Pack
e. Ejecutar el instalador entregado el cual instala los editores
f. Modificar la configuración de RS para extender DataSet usando la última ventana de la instalación
g. Si es necesario modificar la configuración del Editor.
h. Sino se hizo en el punto “f”, instalar la extensión de los orígenes de datos de MS RS, ejecutando el archivo ModificarConfiguracionRSExtenderDataSet.exe.

Estos son todos los puntos para la instalación de los editores de Visual Reporting Services, en este caso es importante seguir el orden de la instalación, e instalar los Service Pack de cada producto de MS.

2.2.1 Modificar La Configuración De RS Para Extender DataSet

ModificarConfiguracionRSExtenderDataSet es el primer producto que se tiene que instalar que no es de MS, para instalarlo se debe tener instalado MS RS. La utilidad de este producto es extender MS RS para que acepte un DataSet como origen de datos, puesto que los únicos tipos de orígenes de datos de RS son: SQL Server, Oracle, ODBC y OLE DB.

Esta extensión se debe realizar tanto en el Diseñador de RS como en el Servidor de RS, para ello se debe:

- Modificar los siguientes archivos de configuración:

rssrvpolicy.config y

RSReportServer.config que están ubicados por defecto en la carpeta del Servidor de RS (C:\Archivos de programa\Microsoft SQL Server\MSSQL\



rspreviewpolicy.config RSReportDesigner.config

Reporting Services\ReportServer) y ubicados por defecto en la capeta de herramientas de diseño de RS (C:\Archivos de programa\Microsoft SQL Server\80\Tools\Report Designer), con esto último se incluye el nuevo origen de datos en el diseñador de RS al utilizar Visual Studio .Net.
Estas parejas de archivos son muy similares pero no idénticas.


RSExtensionOrigenDatosDataSet.dll

- Copiar el archivo en la carpeta donde se encuentra el diseñador y el servidor de RS, este archivo es la implementación de la extensión de RS para aceptar el origen de datos DataSet.

Como los archivos de configuración de RS son archivos XML y es difícil su modificación, se generó un ejecutable el cual verifica la existencia de los archivo en la ubicación indicada y si existen los modifica. Basta que uno de los cuatro archivos no se encuentre para que no realice ningún cambio.

También, para prevenir posibles errores, los archivos originales no son borrados sino que se realiza un respaldo en el mismo directorio donde estén ubicados cambiando ".config" por "_VRS_N.BAK", donde "N" es un número que se va incrementando cada vez que se ejecute el programa, esto es con el fin de no sobre escribir archivos.

Por ejemplo el archivo original de **RSReportServer.config** luego de ejecutar este producto y si pueden realizar los cambios en los cuatro archivos pasara a llamarse **RSReportServer_VRS_0.BAK.** Es muy importante tener en cuenta esto por si se quiere volver a dejar la versión original de los archivos.

Advertencia 1

No remplazar los archivos *.config con los archivos de otra instalación, puesto que cada uno tiene un número de seguridad diferentes. Tampoco borrar los *._VRS_**0**.BAK, pues estos guardan un respaldo de los archivos originales.

Advertencia 2

Si ejecuta 2 veces la aplicación se agregará dos veces las entradas en los archivo de configuración (y se generan un total de 8 archivos). Aunque esta acción fue probada y el funcionamiento de RS fue correcto tanto en el servidor como en el diseñador, se recomienda antes de ejecutar ModificarConfiguracionRSExtenderDataSet por segunda vez, restaurar los archivos originales. Es decir eliminar los cuatro archivos .config y restaurarlos desde los respaldos, que son los archivos con los nombre ***_VRS_0.BAK** (solo los que tienen el número 0) ponerles el nombre ***.config**

En resumen las tareas que debe realizar esta aplicación son:

- Modificar los archivos de configuración, tanto para el diseñador como para el servidos, de MS RS
- Copiar o remplazar el archivo `RSExtensionOrigenDatosDataSet.dll` en las carpetas:

```
C:\Archivos de programa\Microsoft SQL Server\MSSQL\Reporting\bin
C:\Archivos de programa\Microsoft SQL Server\80\Tools\Report Designer
```

2.2.1.1 Prerrequisitos de ModificarConfiguracionRSExtenderDataSet:

Los prerrequisitos para la instalación de ModificarConfiguracionRSExtenderDataSet son:

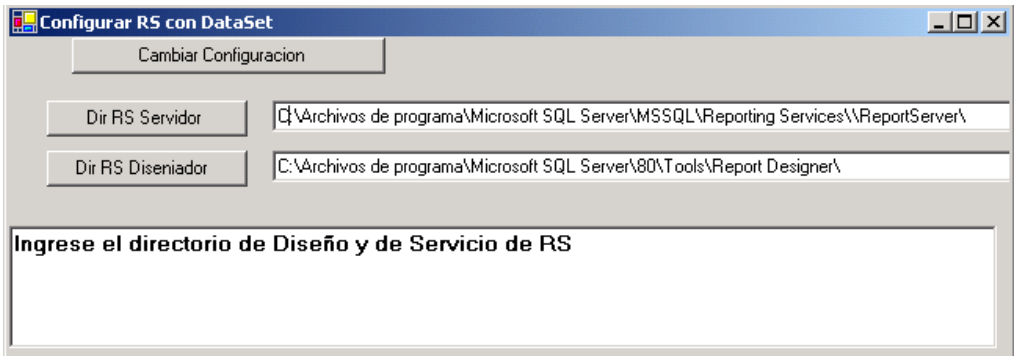
- Verifique que se realizaron los pasos previos comentados en el punto **Pasos de Instalación.**

- Verifique que este **cerrado** MS Visual Estudio y el Administrador de Informes de Reporting Services.
- **Verifique la ubicación exacta de los cuatro archivos nombrados anteriormente** (su ubicación podrá ser indicada en el instalador de la herramienta).
- Verificar que en el App.config del producto esté indicado en forma correcta la ubicación del archivo RSExtensionOrigenDatosDataSet.DLL

2.2.1.2 Instalación

Este producto se puede instalar junto con el editor de VRS o en forma posterior, la ventana que se muestra en cualquiera de los dos casos es la de la siguiente imagen.
Los pasos para instalar este producto en forma independiente a la instalación del editor son:

1. Verificar que se cumplan los prerrequisitos para la ejecución de ModificarConfiguracionRSExtenderDataSet.
2. Tener la ubicación exacta de los archivos de configuración de RS.
3. Si es necesario puede modificar el archivo `App.config` para indicar donde se encuentra RSExtencionOrigenDatosDataSet.dll, por defecto están en el mismo directorio que los editores.
4. Luego ejecutar el archivo `ModificarConfiguracionRSExtenderDataSet` el cual abre la ventana



5. Ingresar la carpeta donde se encuentran los dos primeros archivos en el primer renglón.
6. Ingresar la carpeta donde se encuentran los dos segundos archivos en el segundo renglón
7. Seleccionar el botón Cambiar Configuración.

En caso de mal funcionamiento de RS recupere los archivos de configuración, en caso de no tenerlos reinstale RS (esto hará que se reescriban los archivo de configuración) y luego ejecute la aplicación ModificarConfiguracionRSExtenderDataSet nuevamente.

3 CONFIGURACIÓN DEL ENTORNO

Previo a la descripción de la configuración del editor se hará una descripción de algunos archivos e información requerida, que se puede configurar.

Los **Archivos de Configuración** `app.config` y `vrs.config` contienen datos para el funcionamiento del editor, los valores en este último archivo pueden ser editados desde la ventana de configuración de la aplicación, como se indica más adelante.

La generación de reportes se basa en el uso tres **plantillas** para las cuales se puede configurar su ubicación, más aclaraciones sobre plantillas se darán en el punto **Diseño y Generación de Reportes**.

La generación de un reporte resulta de la combinación de dos datos importantes, el **diseño del reporte** y los **datos de la entidad** que se mostrarán en el reporte (datos del

reporte), esta información se guarda en archivos temporales que luego son eliminados. Los dos archivos que mantienen estos datos son DatosReporte.xml y DiseñoReporte.xml. Estos dos archivos son creados por defecto en el directorio `C:\VisualReportingServices\Temp\Reportes`, siendo `C:\VisualReportingServices` el directorio donde esta instalado el editor.

El archivo DatosReporte.xml es un archivo con formato xml, que resulta de persistir un XML Schema del DataSet que contiene el conjunto de los registros a mostrar.

El archivo DiseñoReporte.xml es un archivo con formato RDL que contiene la definición del reporte para RS.

Los puntos que se tocarán en esta sección son dos archivos que participan en la generación de un reporte y descripción de la información que se puede configurar en los editores

3.1 CONFIGURACIÓN DEL EDITOR

En ambos editores seleccionando la opción de menú Configurar/ Opciones se permite configurar la ubicación u operaciones que se realizarán sobre diferentes archivos. Al seleccionar esta opción se abrirá la ventana de configuración la cual se encuentra dividida en 4 fichas.

Para modificar cualquier opción basta ingresar el dato y seleccionar el botón Aceptar.

Importante: el botón Aceptar guarda los cambios realizados en todas las ficha, no solo en la ficha actual, y además cierra la ventana.

3.1.1 Configuración del Repositorio de Proxies

Mediante el menú de configuración en el Editor de Metadata se puede editar la configuración del Repositorio de Proxies. Este componente se encarga del almacenamiento de los proxies para las fuentes de información.

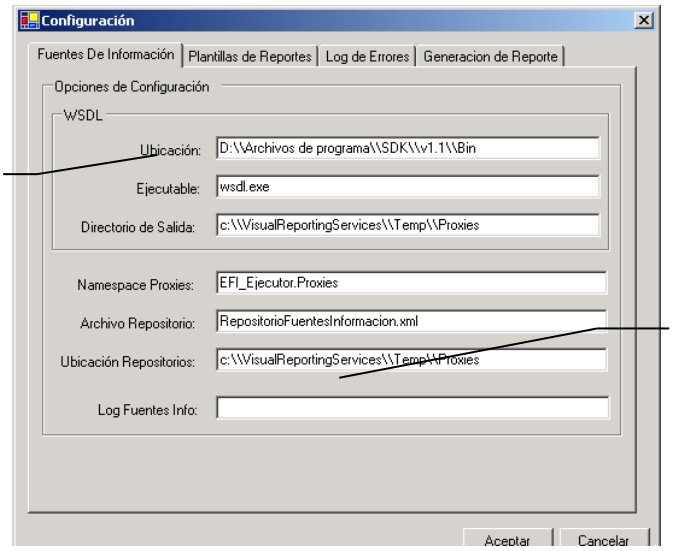
Los parámetros configurables son los siguientes:

- Ubicación del WSDL, que indica donde se encuentra el ejecutable del aplicativo WSDL
- Ejecutable del WSDL, que indica el nombre del ejecutable del aplicativo WSDL.
- Directorio de salida, indica donde se ubican los archivos de salida del WSDL.
- Namespace de los proxies, indica en que namespace se generan los proxies.
- Indica la ubicación del Repositorio de Adaptadores para las FI.
- Archivo del repositorio en el que se almacenan sus datos.
- Ubicación del repositorio, ubicación del archivo del repositorio.
- Log Fuentes Info, archivo de logueo del generador de proxies.

En la siguiente imagen se muestra la pantalla con la información por defecto, siempre y cuando se tenga instalada una versión de Windows en español.

3.1.

Del WSDL se puede configurar:
Ubicación, Ejecutable y Directorio de salida



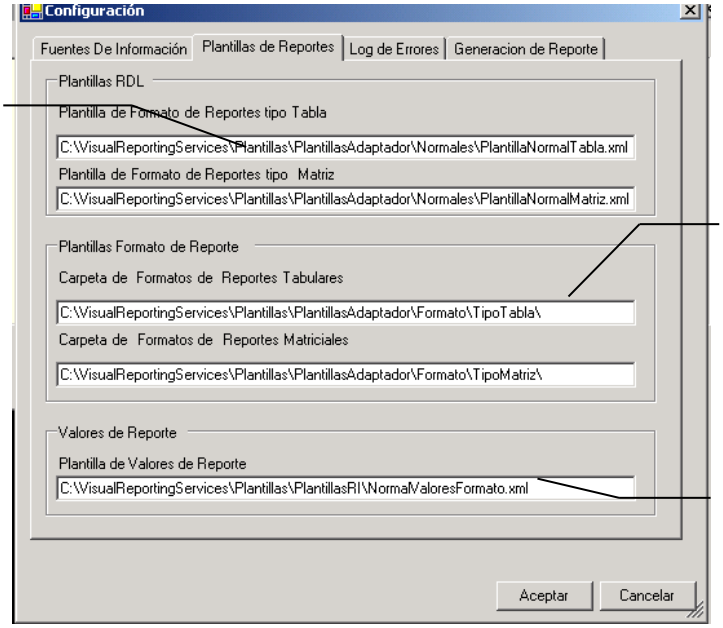
Configuración del Repositorio de Adaptadores

Configuración de Diseño del Reporte

Cada editor podrá tener diferentes plantillas o las mismas pero ubicadas en diferentes carpetas, con el fin de modificar la ubicación de las plantillas se debe ir a la ficha de Plantillas de Reporte en la ventana de Configuración, esto se hace seleccionando Configuración / Opciones / Plantillas de Reportes, luego se ingresa la ubicación de las plantillas. También desde esta ficha se puede indicar donde están instaladas esas carpetas sino se realizo la instalación por defecto.

En la imagen se puede ver la ficha con los valores por defecto.

Dirección y nombre de las dos plantillas Normales, con TODA la estructura del RDL,
PlantillaNormalTabla.xml
PlantillaNormalMatriz.xml

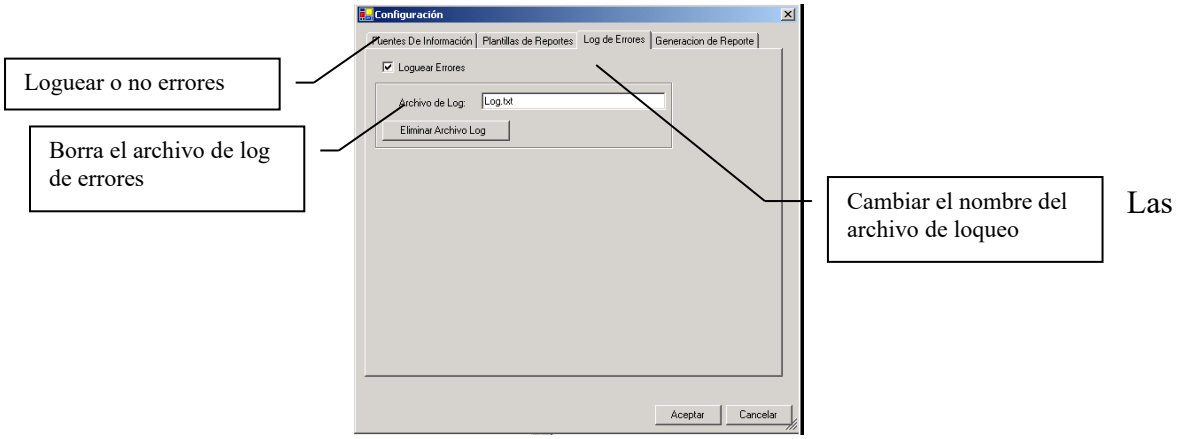


Carpetas de las múltiples plantillas de cuerpo de reporte.
Eje CuerpoTabla2.xml o CuerpoMatriz2.xml

Ubicación de la plantilla con los valores del formato NormalValoresFormato.xml

Configuración del logueo de errores

Mediante el menú de configuración (Configurar/ Opciones) en el Editor se puede configurar el logueo de errores al ejecutar un conjunto de comandos, esto permite guardar los errores para un posterior análisis.



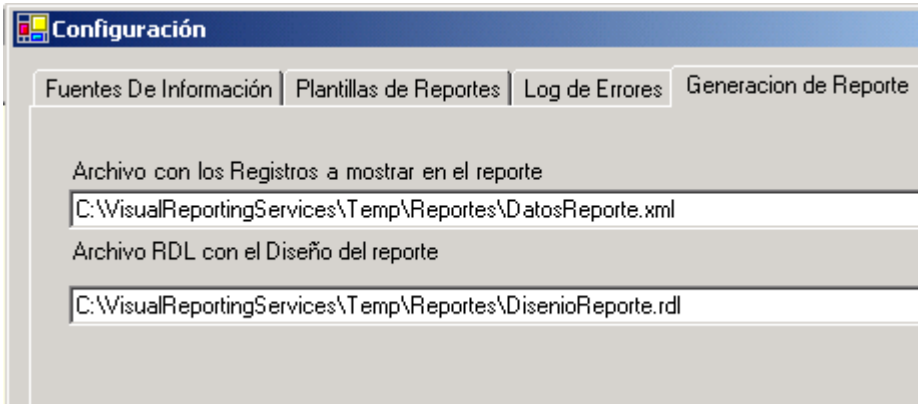
Las opciones son:

- Loguear o no errores,
- Indicar el nombre de archivo de log
- Eliminar el archivo de log de errores.

3.1.4 Configuración de la Generación del Reporte

Esta ventana permite configurar la carpeta y nombre donde se mantienen temporalmente los dos archivos necesarios para la generación del reporte, normalmente el primer nombre de archivo DatosReportes.xml y el segundo nombre de archivo es DisenioReporte.xml.

La siguiente imagen muestra la ubicación por defecto de estos archivos.



4 Instalación y Configuración de Ejemplos

Los archivos de ejemplo son copiados, cuando se instalan los editores de VRS, en el mismo lugar que el producto pero en la sub carpeta EjemplosDeScript
Se tienen los siguientes archivos de ejemplo para la el Editor de Metadata:

- Ej1-titularesMagma.txt
- Ej2-Cines.txt
- Ej3-MovimientosDeUnCliente.txt
- Ej4-pruebasRSMatrizYAgrupar.txt

Dentro de estos ejemplos se utilizan las siguientes fuentes de información, las cuales tiene que haber sido instaladas previamente para el buen funcionamiento de los script:

- WSMagma
- WebServiceCines
- CapaDatos

- AccesoASQLServer
- RemotingCines.

Para ejecutar los scripts se deberán tener instaladas las fuentes de información, abrir el Editor del Meta-modelo, y de ahí abrir el archivo correspondiente al ejemplo que se quiera ejecutar.
Estas fuentes de información no son instaladas con el producto, pero pueden ser encontradas con el mismo CD, bajo la carpeta WebServicesEjemplo. Utilizan bases de datos de MS Access y SQL Server. Como bases de datos de SQL Server se tienes las bases Magma_Demo y Cines, que debe estar disponible en MS SQL Server local. La instalación de las FI y los servicios que proveen se describen a continuación.
Cuando la FI sea un servicio Web se deberá copiar la carpeta del ejemplo (dentro del wwwroot o compartirla con el nombre del ejemplo) y dar acceso a la cuenta del Internet Information Server para que pueda acceder a los archivos dentro de la carpeta del servicio web.

El **WSMagma** accede a la base de datos de `Magma_Demo`, para utilizar este WS se debe ubicar en la url `http://localhost/WSMagma` y presenta los siguientes métodos:

```
public DataSet ObtenerTitulares()
public DataSet ObtenerMovimientos()
public DataSet ObtenerSaldoHastaFecha(string fecha, long
cod_tit)
public DataSet
ObtenerMovimientoTitularDespuesDEFecha(string fecha, long
cod_tit)
```

El WS **WebServiceCines** ubicado en la url `http://localhost/WebServiceCines` el cual permite acceder a la base de datos `Sabor.MDB`, los métodos que ofrece son:

```
public DataSet ObtenerClientes()
public DataSet ObtenerClientesPorNombre(string pNombre)
public DataSet ObtenerClientesConFiltros(string pNombre,
string pIngresoDespuesDe, string pIngresoAntesDe)
public DataSet ObtenerPeliculas(long pId, string pNombre,
bool pProximoEstreno, DateTime pFechaEstreno, decimal
pDuracionMinima)
public DataSet ObtenerPeliculasGenero(string pGenero)
public DataSet ObtenerSalas()
public DataSet ObtenerComplejos()
public DataSet ObtenerProgramaciones()
public DataSet ObtenerPeliculaIdGenero()
```

La biblioteca **CapaDatos** permite acceder también a la base de datos Sabor.MDB pero por medio de una DLL, para ello se tiene que ubicar el archivo CapaDatos.dll en el directorio `D:\ProyectoDeGradoFI\WebServicesEjemplo\CapaDatos\bin\Debug`

Las funcionalidades ofrecidas por este servicio son:

```
public static DataSet ObtenerClientes()
public static DataSet ObtenerClientesPorNombre(string
pNombre)
public static DataSet ObtenerClientesPorNombre(string
pNombre, string pIngresoDespuesDe, string pIngresoAntesDe)
public static DataSet ObtenerPeliculas()
public static DataSet ObtenerPeliculas(long pId, string
pNombre, bool pProximoEstreno, DateTime pFechaEstreno, decimal
pDuracionMinima)
```

```
      public static DataSet ObtenerPeliculasGenero(string
pGenero)
      public static DataSet ObtenerSalas()
      public static DataSet ObtenerProgramaciones()
      public static DataSet ObtenerPeliculaIdGenero()
```

Para utilizar las fuentes de información RemotingCines y AccesoASQLServer se deberá:

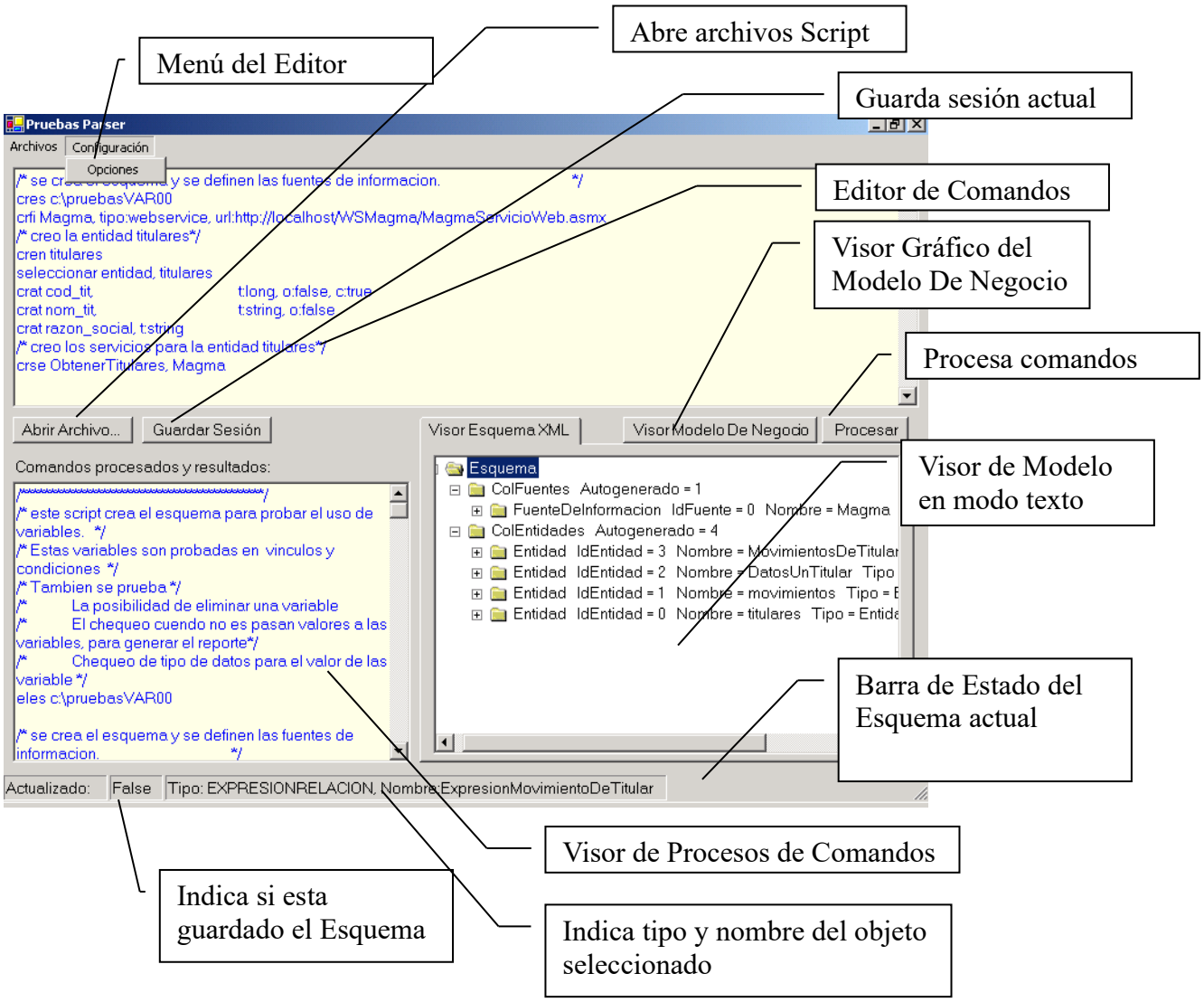
- restaurar la base de datos “Cines”, para la que se puede encontrar un respaldo en la carpeta “fuentes de informacion\WebServicesEjemplo\ AccesoASQLServer” dentro de los ejemplos en el CD entregado. La biblioteca utiliza el usuario ConsultorCines contraseña en blanco, para acceder a la BD.
- Instalar el servicio de remoting RemotingCines, lo que se puede hacer con el archivo “instalar RemotingCines.bat” que podrá encontrar en “fuentes de informacion\WebServicesEjemplo\RemotingCines” dentro de ejemplos en el mismo cd. Para que este servicio se instale correctamente es probable que debe editar el archivo para indicar la ubicación de los ejecutables del servicio de remoting y del aplicativo installutil.

Los servicios provistos por estas fuentes de información tienen la misma firma que los que provee CapaDatos, diferenciándose de estos por el origen de los datos, los datos y la tecnología mediante la cual acceden a ellos.

5 UTILIZAR EL EDITOR DE METADATA

Luego de tener configurado el ambiente de trabajo y generados los adaptadores de Fuentes de Información se pasa a trabajar con uno de los editores de Visual Reporting Services, para el caso del UE el editor adecuado sería el Editor de Metadata.

Este editor consta de: tres áreas de trabajo, un menú, cuatro botones y una barra de estado.



Las áreas de trabajo son: Editor de Comandos, Visor de Proceso de Comando y Visor de Esquema, las cuales pasamos a detallar.

El área **Editor de Comandos** es donde se ingresan los comandos a ejecutar, para editar el modelo de negocios o generar reportes, estos comandos están descriptos en el documento **5 - Lenguaje de Edición de Metadata.doc**

El área **Visor de Procesador de Comandos**, muestra el resultado de la ejecución de los comandos, en caso que la ejecución del comando no se pueda realizar en esta área se mostrará un mensaje de error.

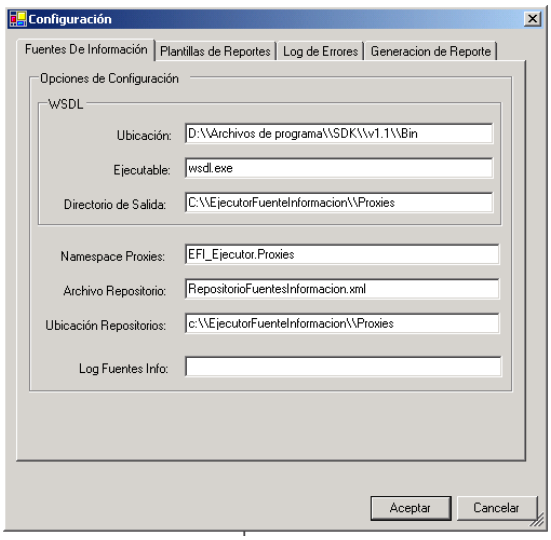
Área **Visor de Esquema** muestra en modo de texto los datos completos del esquema que se esta editando actualmente, este visor muestra la misma información que se persiste cuando se guarda el esquema. Para ver la descripción de esta información ver el archivo **Análisis de la metadata y su Persistencia.doc**

Las opciones de menú son dos: Archivo y Configuración.

La opción Archivo se divide en Archivo / Guardar Sesión Como, la cual es equivalente al botón Guardar Sesión y la tarea que realiza es guardar el resultado de la sesión actual, cada sesión comienza cuando se abre por ultima vez el editor y finaliza cuando se cierra, la información que guarda es todo lo mostrado en el área Visor de Procesador de Comandos.

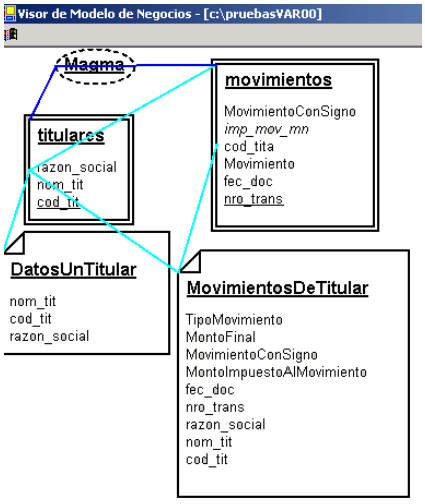
La opción Archivo/Abrir es equivalente al botón Abrir Archivo, en este caso permite abrir un archivo con comandos para ser ejecutados en el editor, todo el contenido del archivo abierto será colocado en el área Editor de Comandos.

La opción Configuración/ Opciones abre una nueva ventana que permite configurar las variables de entorno de trabajo. Esta ventana, que fue descrita en profundidad más arriba, esta dividida en las fichas Fuente de Información, Plantillas de Reportes, Log de Errores y Generación de Reportes.



Los botones restantes en la ventana principal son: Visor de Modelo de Negocio y Procesar.

El botón Visor Modelo de Negocios, abre una ventana en la cual se muestra una representación gráfica del modelo de negocio que se esta editando actualmente, en esta representación gráfica se visualiza: el esquema con sus fuentes de información, las entidades representadas de diferente formas según el tipos entidades que sea, además se representa sus atributos, la relación entre las entidades y las fuentes de información.



También representa los atributos de las entidades con diferente formato según sus propiedades de clave oculto horizontal o múltiples.

El Botón Procesar permite ejecutar todos los comandos que se encuentran en el área Editor de comandos, en caso que los comandos no sean correctos se interrumpe la ejecución de los comandos y se muestra un error en el área Visor de Procesador de Comandos.

La barra de estado, que se encuentra en la parte inferior de la pantalla, indica si el esquema esta guardado o no, además indica el nombre y tipo del objeto seleccionado actualmente.

5.1 Forma de Trabajar

La forma de trabajar con este editor es:

a. Si lo requiero configuro el ambiente de trabajo usando la opción de menú Configuración/Opciones.
b. Ingresar uno o muchos comandos, en el editor de comandos, pudiendo seleccionar un archivo con los comandos a ejecutar o ingresándolos a mano.
c. Agregar enter (nuevo renglón) al final del último comando
d. Seleccionar el botón Procesar para ejecutar los comandos.
e. Verificar si los comandos se ejecutaron correctamente en el área del visor de Procesador de Comandos.
f. Se puede verificar el estado del esquema actual usando
   El Visor de Esquema que muestra en forma detallada pero en modo texto o
   El Visor de Modelo de Negocio que muestra en forma gráfica pero más global.
g. Se puede guardar el esquema generado para una posterior edición.
h. Se puede generar reportes con alguno de los conceptos representados.

5.2 Representación de Modelos de Negocio

Una vez instalado y configurado el ambiente de trabajo como se indica más arriba, se pueden comenzar a utilizar el Editor de Metadata.

Se sugiere que los pasos, a grandes rasgos, para crear la representación de un modelo de negocios con este editor sean:

1. Crear un esquema nuevo.
2. Crear en el esquema las Fuentes de Información.
3. Crear las Entidades, que se correspondan con los conceptos que habitualmente se usan en el negocio, y en cada entidad.
   a. Agregarle los atributos
   b. Crear las relaciones entre las entidades
   c. Si el concepto a representar obtiene sus datos directamente de las fuentes de información crea un servicio sino crear una expresión de relación.
   d. Agregar los cálculos que permitan vincular cada atributos con los datos de la entidad
   e. Si el concepto lo requiere agregar operaciones de: Agrupación, Ordenación, Criterios o Variables.
4. En cualquier momento se puede configurar el diseño del reporte asociado a una entidad definida o dejar el diseño por defecto
5. En cualquier momento se puede generar un reporte con una entidad ya creada que representa un concepto del negocio.

6. Cuando esta definido un modelo de negocio se puede guardar para una posterior edición.

Algunos de estos pasos se pueden cambiar de orden, por ejemplo se pueden crear algunas fuentes de información y luego crear entidades y posteriormente crear más fuentes de información.

A continuación se detallaran más cada uno de estos pasos y otras operaciones alternativas que se puedan realizar, indicando cuales son los comandos que se pueden aplicar en cada momento

5.2.1 Editar Modelo de Negocio (Esquema)

Un modelo de negocio se va a representar por medio de un esquema, por lo tanto la edición del modelo de negocio se hace por medio de los comandos que afectan al esquema, estos comandos son

- crearEsquema o crES
- renombrarEsquema o renES
- eliminarEsquema o elES
- abrirEsquema o abES
- guardarEsquema o guES
- cerrarEsquema o ceES

Una alternativa a crear un esquema nuevo, obviamente, es abrir un esquema persistido anteriormente usando el comando abES

5.2.2 Representación Fuentes de Información

Una vez creado los adaptadores que permiten obtener los datos operacionales de las diferentes fuentes de información, el esquema debe mantener la forma de acceder a los servicios ofrecidos por esos adaptadores, esta información es manejada al definirse las fuentes de información en el esquema.

Los comandos que se pueden aplicar a las instancias de las fuentes de información definidas en un esquema son:

- crearFuente (crFI)
- modificarFuente (modFI)
- eliminarFuente (elFI)

5.2.3 Representación de los Conceptos (Entidades)

Luego de creadas las fuentes de información en el esquema, se debe crear los diferentes conceptos que se manejan en la organización, cada concepto va a estar representado por una entidad en el esquema.

Sino se definió la forma de obtener los datos relacionados con un concepto, diremos que este concepto esta representado por una entidad; después de definida la forma de obtener los datos relacionados con el concepto la representación pasa a ser uno de los siguientes tipos EntidadesServidas o EntidadesRelacionales o EntidadesReportes.

Los comandos que se pueden aplicar una vez seleccionada una entidad son los comandos de Atributos los de Relaciones y dependiendo el tipo de entidad se podrán aplicar los comandos de Servicio o Expresiones de Relación y/o los de Variables

5.2.3.1 Definición de Entidades servidas

Una Entidad Servida es la representación un concepto del negocio que obtiene sus datos directamente de ejecutar uno o varios servicios de los adaptadores de las fuentes de información.

Es por esto que estas entidades manejan la representación de servicios relacionado con las fuentes de información.

Al crearse un servicio en una entidad, el concepto pasa a ser representado por una EntidadServida. Las características de los servicios serán descriptas más adelante

5.2.3.2 Definición de entidades Relacionales

Un concepto pasa a representarse por una Entidad Relacional cuando se crea una Expresión de Relación, en esta expresión se indica que la forma de obtener los datos de la entidad es por relaciones con otras entidades definidas anteriormente.

Las características de las expresiones de relación se indicarán más adelante.

5.2.3.3 Definición de entidades Reportes

Un concepto pasa a representarse por una Entidad Relacional cuando se crea una variable en la Entidad Relacional.

Las características de las variables se indicarán más adelante.

5.2.4 Representación de Datos Relevantes de los Conceptos (Atributos)

Los datos más relevantes de cada concepto son representados por los atributos, es por ello que una vez creada una entidad se debe crear los atributos de esa entidad.

Los comandos que se pueden aplicar sobre los atributos son:

- crearAtributo (crAt)
- modificarAtributo (modAt)
- eliminarAtributo (elAt)

5.2.5 Representación de Conexiones entre Conceptos (Relaciones)

Una vez representado un concepto y sus datos relevantes, también podemos representar la conexión que existe entre dos conceptos por medio de la definición de Relaciones. Puede ser que una entidad no tenga relaciones con otras entidades por lo cual no es obligatoria la creación de una relación en cada entidad.

Los comandos que se pueden aplicar sobre las relaciones son:

- crearRelacion (crRe)
- modificarRelacion (crdRe)
- eliminarRelacion (elRe)

5.2.6 Representación de Servicios

Un servicio representa una operación ofrecida por un adaptador de una fuente de información, al ejecutarse esa operación se obtendrán algunos datos operacionales de la organización.

Los servicios solo podrán ser creados en entidades o en entidades servidas.

Los comandos que se pueden aplicar sobre un servicio son:

- crearServicio (crSe)*
- modificarServicio (modSe)*
- eliminarServicio (elSe)*

5.2.7 Representación de Operaciones entre los Conceptos (Expresiones)

Una expresión de relación representa las diferentes operaciones que se pueden realizar entre dos o más conceptos del negocio, es así que en ella vamos a encontrar las operaciones de agrupamiento y/ o de ordenación y/ o condiciones que deben de cumplir los datos de la entidad, además de estas operaciones también permite definir los cálculos vinculados con cada atributo de la entidad actual.

Los comandos que se pueden aplicar sobre las expresiones son:

- crearExpresionRelacion (crEx)*
- modificarExpresionRelacion (modEx)*
- eliminarExpresionRelacion (elEx)*

5.2.8 Representación de Cálculos de los Atributos (Vínculos)

Los datos relacionados con un atributo se podrán obtener de varias formas diferentes, las cuales son: una constante, una variable, directamente de un servicio o directamente de la relación con un atributo de otra entidad, o por aplicar una función o una operación a los puntos antes comentados. Es decir que el vínculo es lo que permite representar como se calcularan los datos para cada atributo.

La única operación posible sobre los vínculos es:

- vincular

5.2.9 Definición de Variables

Es común que un concepto requiera de información externa para determinar los datos de la entidad que lo representa, se sabe que operaciones se realizará con esta información pero solo se conoce su valor en el momento de generar el reporte, es para la representación de este tipo de conceptos que se requiere definir una variable en la entidad que los representa.

Los comandos sobre las variables son:

- crearVariable (crVa)
- eliminarVariable (elVa)

5.2.10 Otras Operaciones

Como ya se adelanto en las expresiones de relación otras operaciones comunes en los conceptos de negocio son las operaciones de ordenamiento de datos, o la agrupación y aplicar alguna operación sobre cada grupo, y las condiciones.

Los comandos que se pueden aplicar sobre una expresión de relación para realizar estas operaciones son:

- condicion (co)
- agrupar (ag)
- ordenar (or)

5.3 EJEMPLO

El siguiente es un ejemplo simple, el cual muestra todos los pasos a realizar y comandos a ejecutar para la generación de un Esquema y un reporte, este ejemplo también está en forma más extendida, junto con otros ejemplos, en el documento **Lenguaje de Edición de Metadata.doc**.

Los pasos sugeridos en puntos anteriores para la generación del modelo de negocio y en particular un esquema son:

1. Crear los adaptadores que permiten obtener los datos de las entidades
2. Crear un esquema nuevo.
3. Crear en el esquema las Fuentes de Información.
4. Crear las Entidades, que se correspondan con los conceptos que habitualmente se usan en el negocio, y en cada entidad.
    a. Agregarle los atributos
    b. Crear las relaciones entre las entidades
    c. Si el concepto a representar obtiene sus datos directamente de las fuentes de información crea un servicio sino crear una expresión de relación.
    d. Agregar los cálculos que permitan vincular cada atributos con los datos de la entidad
    e. Si el concepto lo requiere agregar operaciones de: Agrupación, Ordenación, Criterios o Variables.

5. Se puede configurar el diseño del reporte asociado a una entidad o dejar el diseño por defecto
6. Una vez creada una entidad que representa un concepto del negocio, se puede generar un reporte con los datos de la entidad.
7. Cuando esta definido un modelo de negocio se puede guardar para una posterior edición.

Pasamos a detallar cada uno de estos pasos, con los respectivos comandos.

1. Crear los adaptadores que permiten obtener los datos de las entidades

Para este esquema se requiere tener implementado un Web Services ubicados en la dirección `url:http://localhost/WebServiceCines/WSCines.asmx`, la cual ofrece los servicios

2. Crear un esquema nuevo y guardarlo

```
cres c:\pruebasRSMatrizYAgrupar00
gues
```

3. Crear en el esquema las Fuentes de Información que permita ejecutar el servicio definido en el puntos 1

```
crfi algunos_cines, tipo:webservice,
url:http://localhost/WebServiceCines/WSCines.asmx
crfi otros_cines, tipo:webservice,
url:http://localhost/WebServiceCines/WSCines.asmx
```

4. Crear las Entidades, que se corresponde con los conceptos que habitualmente se usan en el negocio.

```
cren ProductoraIDGenero
```

5. Agregarle a la entidad los atributos, previa selección de la entidad a modificar.

```
seleccionar entidad, ProductoraIDGenero
crat id, t:long, o:true, c:true
crat genero, t:string
crat productora, t:string, h:true
```

6. Crear las relaciones entre las entidades, para este caso debido a la simplicidad del ejemplo no se agregaron relaciones
7. Si el concepto a representar obtiene sus datos directamente de las fuentes de información crea un servicio sino crear una expresión de relación.

Para este ejemplo se crea un servicio llamado `ObtenerPeliculaIdGenero`

```
crse ObtenerPeliculaIdGenero, algunos_cines
```

8. Agregar los cálculos que permitan vincular cada atributos con los datos de la entidad

```
seleccionar servicios, algunos_cines
vincular id,id
vincular genero, genero
vincular productora, productora
liberar
liberar
```

9. Si el concepto lo requiere agregar operaciones de Agrupación, Ordenación, Criterios o Variables, en este caso no se hará.
10. Se puede configurar el diseño del reporte asociado a una entidad o dejar el diseño por defecto, también se puede configurar otra serie de características en cuanto al formato del reporte, como ser:

- Ancho de columnas

```
ModificarAnchoColumna productora , 5
ModificarAnchoColumna genero, 5
```

- Formato del reporte para cambiar el formato del reporte (`matrizCompacto`), tamaño de la hoja y de los márgenes, indicar si se muestra un encabezado y un pie y el logo de la empresa (`c:\Logo.jpg`):

```
ModificarFormatoReporte matrizCompacto, 30, 20, 5, 6, false ,true , false ,
'Mis Peliculas', 1, 2, 3, 4,'c:\Logo.jpg'
```

11. Una vez creada una entidad que representa un concepto del negocio, y realizadas las configuraciones requeridas, se puede generar un reporte con los datos de la entidad.

```
generarreporte ProductoraIDGenero
```

6 Diseño y Generación de Reportes

En la generación de los reportes hay varios puntos que son de interés entre los que destacamos:

- Plantillas de Reportes: muestra como cambiar la presentación de las diferentes áreas del reporte
- Metodología para crear el reporte, donde se describa los pasos a seguir para la generación del mismo.
- Tipos de reportes: donde se describen los diferentes tipos de reportes y como pasar de uno a otro.
- Como son mantenidos tanto los reportes como sus diseños.

6.1 Plantillas de Reportes

Una plantilla de reporte es un archivo con formato RDL el cual es una especificación de XML, el cual sirve como modelo para la generación de un reporte o un área particular del reporte, la mayor parte de la información de las plantilla no cambia solo cambian algunos valores y/o alguna información menor, la cual debe mantener el formato de la plantilla; la plantilla aporta estructura y valores de inicialización a la generación del diseño del reporte. Por ejemplo en el caso de los atributos, la plantilla mantiene la siguiente información:

```
<TableCells>
	<TableCell>
		<ReportItems>
			<Textbox Name="Id">
				+<Style>
				<Value>=Fields!Id.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
</TableCells>
```

En cambio al generarse el diseño del reporte vemos la información siguiente, donde se mantiene el formato pero se agregaron tanto nodos TableCell como atributos de la entidad existen, colocando el nombre de cada atributo como nombre de un cuadro de texto y de un campo

```
<TableCells>
	<TableCell>
		<ReportItems>
			<Textbox Name="Cedula">
				+<Style>
				<Value>=Fields!Cedula.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
		<TableCell>
		<ReportItems>
			<Textbox Name="Nombre">
```

```
                        +<Style>
                        <Value>=Fields!Nombre.Value</Value>
                    </Textbox>
                </ReportItems>
            </TableCell>
                <TableCell>
                <ReportItems>
                    <Textbox Name="Direccion">
                        +<Style>
                        <Value>=Fields!Direccion.Value</Value>
                    </Textbox>
                </ReportItems>
            </TableCell>
</TableCells>
```

Previa a la generación de un reporte se debe crear las plantillas de formato de reporte; excepto que se desee usar las plantillas prediseñadas que vienen con el sistema.

Cada entidad puede tener un formato de reporte diferente; para estandarizar la estructura de la información y determinar que valores se pueden modificar, se parte de una plantilla común y se genera una instancia de esa plantilla para cada entidad. Es decir que cada entidad tendrá una instancia de la plantilla de formato.

En la generación del formato del reporte se utilizan tres tipos de plantillas diferentes, que son Plantillas Normales de RDL, Plantillas de Representación Intermedia y Plantillas de Formato de Cuerpo.

6.1.1 Plantillas Normales de RDL

Son las plantillas que tiene la estructura básica para generar el diseño de reporte (*.rdl). Existen dos de esta plantillas una para generar reportes en formato matriz y otra para generar reportes en formato tablas, si la Entidad a reportear tiene atributos horizontales se utiliza la plantilla de matriz sino se utiliza la plantilla de tablas.

Esta plantilla dará el formato de reporte por defecto, por lo tanto todas aquellas entidades que no lo modificaron tendrán el formato que aporta esta plantilla.

Aunque no es necesario generar estas plantillas nuevamente, puesto que se puede cambiar el formato del reporte asociado a una entidad a través del comando ModificarFormatoReporte, se describe un método para generarlas, para ello se utiliza MS Visual Estudio 2003, los pasos para generar esta plantilla son:

1. En MS VS 2003 crear un proyecto nuevo del tipo Business Intelligense con la plantilla Proyectos de Informes
2. En la solución creada agrega un origen de datos nuevo, con las siguientes características
    a. El nombre del origen de datos será “DataSource1”
    b. El origen de datos será “DataSet Data Extensión”, que es una extensión de Reporting Services para aceptar como origen de datos los DataSet (ver este documento más arriba).
    c. Cadena de conexión cualquier cosa (por ejemplo “yyy”), puesto que para este origen de datos no se usará, pero si generará un nodo que permita mantener esta información, además este dato es requerido por el asistente.
    d. Para la autenticación utilizar autenticación de Windows (seguridad integrada)

    Asegurarse que el Origen de datos tenga datos en los valores:
    Nombre: DataSource1
    Tipo: DataSet Data Extensión
    Cadena de conexión: yyy

Credenciales: Autenticación de Windows

3. Crea un reporte nuevo seleccionando Agregar nuevo Informe
    a. Seleccionar el origen de datos DataSource1 creado en el paso 1 a
    b. Ingreso una dirección de archivo que tenga cualquier persistencia de un DataSet, por ejemplo PeliculaGeneroYProductora.xml, que esta incluido en el producto
    c. Seleccionar Tabla o Matriz según el tipo de Plantilla a crear.
    d. Si es una plantilla en Tabla agrego un único campo en Detalle, si es una plantilla en matriz agregar un campo en fila uno en columna y uno en detalle. Los nombres de los campos solo podrán tener caracteres americanos y sin espacios.
    e. Seleccionar un estilo, este se podrá cambiar más adelante usando el diseñador.

Verificar que en el RDL del reporte aparezcan los tags:

DataSourceName, CommandType, CommandText, Timeout
Por ejemplo:

```
<DataSourceName>DataSource1</DataSourceName>
<CommandType>Text</CommandType>
<CommandText>
        C:\PelicualaGeneroYProductora.xml
</CommandText>
<Timeout>0</Timeout>
```

4. Usando el diseñador agregue los siguientes controles, los cuales deberán tener el nombre indicado
    a. Un encabezado con:
        i. Un control text box llamado txtTextoEncabezado
        ii. Una control imagen de nombre “image1”
    b. Deberá tener un pie de página con cuatro controles TextBox llamados:
        i. txtNumeroDePagina,
        ii. txtSeparador,
        iii. txtTotalHojas,
        iv. txtFechaGeneradoReporte
    c. El reporte deberá tener cualquier dato en las propiedades:
        i. Description
        ii. Autor
        iii. AutoRefresh
5. Modificar cualquier parte del formato, tanto de las diferentes áreas como de los cuadros de texto
6. Guardar el archivo de reporte generado

Colocar la nueva plantilla como plantilla normal

    a. Editar el archivo RDL generado y quitar todas las referencias a espacios de nombre, en particular a los espacios de nombre **xmlns** y **xmlns:rd**, de acá en más no se podrá usar el diseñador para modificar estos archivos
    b. Guardarlo con el nombre PlantillaNormalMatriz.xml o PlantillaNormalTabla.xml, según el tipo de reporte generado.
    c. En el Editor de Modelo de Negocios cambie el directorio de las plantillas RDL de tablas y/o de matriz, según corresponda; para ello seleccione Configuración/ Opciones/ Plantillas de Reportes. Otra opción es remplazar los archivos existentes indicados en la ventana de configuración.

El formato que tenga este diseño de reporte será el formato que mostrarán los reportes generados por medio de los Editores de VRS, cuando se tome los valores por defecto.

Se entregan 4 archivos con estas plantillas que son:

OriginalPlantillaNormalMatriz.xml OriginalPlantillaNormalMatriz.xml que son los archivos RDL generados con el editor de reportes.
PlantillaNormalMatriz.xml PlantillaNormalMatriz.xml: que son los archivos RDL sin los espacios de nombre.

6.1.2 Plantillas de Representación Intermedia

Esta plantilla es usada para mantener una representación genérica e independiente de los reportes, sin interesar si el formato del reporte es tabla o matriz. Las plantillas normales adaptan sus valores según el contenido de las plantillas de Representación Intermedia la cual da independencia del formato del reporte. Esta plantilla mantendrá información para la configuración de la hoja y el nombre del la Plantilla de Formato de Cuerpo de Reporte.

No es conveniente editar esta plantilla y en caso de hacerlo muy probablemente se tenga que editar el código de la aplicación. La ubicación de esta plantilla se configura en cada Editor seleccionando Configuración/ Opciones/ Plantillas de Reportes/Plantillas de Valores de Reporte.

6.1.3 Plantillas de Formato de Cuerpo

Son plantillas que determinan como serán mostrados los datos en cuanto a colores, tipos, de letras bordes, sombreados, etc., es decir que permiten cambiar la presentación estética del reporte.

Es muy común crear este tipo de plantillas, es por ello que se buscó un método que simplifique el hecho de generar estas plantillas, los pasos son:

a. Se debe crear un reporte usando MS VS 2003 y un proyecto de Business Intelligense
b. Para este caso el nombre del origen de datos debe ser “DataSource1” el tipo y la conexión puede ser cualquier texto.
c. Si se trata de una plantilla para reportes tipo Tablas debe agregarse un único campo en Detalles.
   Si se trata de una plantilla para reportes tipo matriz debe agregarse 3 campos uno en fila otro en columna y otro en detalle
   En cualquiera de los dos casos los nombres solo tendrán caracteres americanos y sin espacios.
d. Modificar el formato del cuerpo del reporte, en la vista diseño, colocando la prestación deseada.
e. Esto generará un archivo RDL el cual contiene la plantilla con el formato de cuerpo de reporte; si ejecutamos el comando `CrearPlantillaCuerpoReporte` genera una plantilla de cuerpo de reporte nueva y la guarda en el lugar indicado por el Editor en el menú Configuración/ Opciones/ Plantillas de Reportes/ Plantillas de Formatos de Reportes
f. Una vez definida la plantilla de formato de reporte ejecutando el comando `ModificarFormatoReporte` se puede relacionar ese formato de reporte con una entidad

6.2 GENERACIÓN DE REPORTES

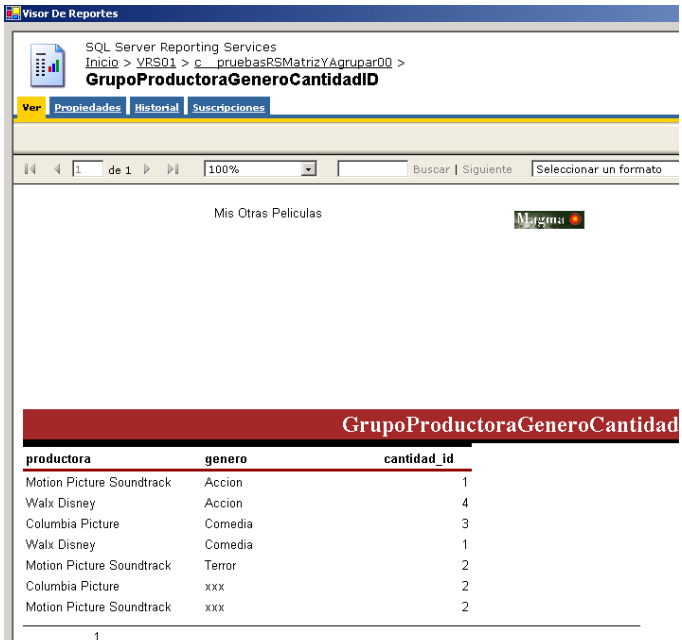
Una vez definido la representación de un modelo de negocios y generados los formatos de reportes, podemos generar un reporte con el formato deseado, y configurar algunos datos a mostrar en ese reporte. Los pasos son:

a. Modificar el formato del reporte de una entidad con el comando ModificarFormatoReporte
b. Generar el reporte con el comando generarreporte
c. Guardar el reporte en los formatos que tiene preestablecidos RS o imprimir el reporte.

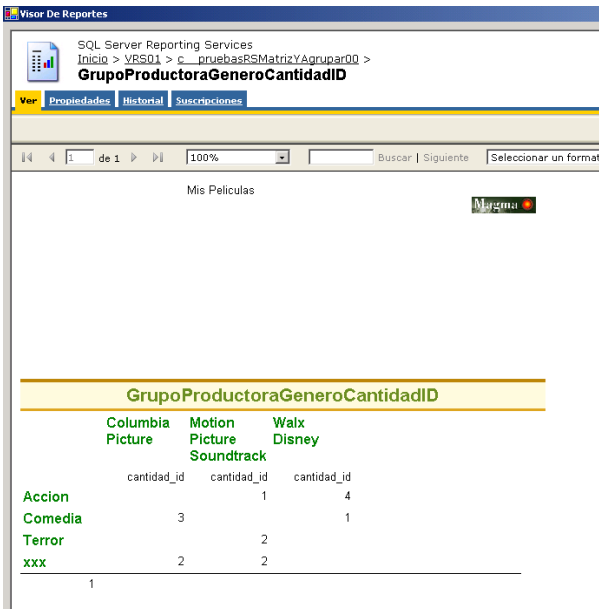
6.2.1 Tipo de Reportes

Existen dos tipos de reportes que son los reportes tipo matriz y los reportes tipo tabla, los cuales mostramos en los ejemplos siguientes:

Reporte tipo Tabla



Reporte tipo Matriz



Generalmente tendremos reportes tipo tabla, pero si se desea generar reportes tipo matriz entonces la entidad a reportear debe cumplir tres condiciones que son excluyentes:

- Por lo menos debe existir un atributo horizontal, los datos asociados a estos atributos se mostrarán como títulos de columna.
- Debe existir una operación de agrupamiento con los atributos horizontales y otros atributos, los datos a asociados a estos otros atributos estarán como títulos de fila.
- Los atributos que no estén en la operación de agrupamiento, se colocaran sus datos asociados en el área de detalle.

En conclusión para ser un reporte tipo matriz deberá tener atributos horizontales y agrupamientos donde se incluyan los atributos horizontales, sin se realizara un repote tipo matriz.

6.2.2 Ubicación del Diseño de Reporte en RS

Cada vez que se genera un reporte, usando uno de los editores de VRS el diseño del reporte es guardado en el servidor de Reporting Services en la carpeta VRS01 y dentro de la carpeta que tiene igual nombre que el esquema, pero por seguridad los datos relacionados con este diseño de reporte no son guardados, es decir que el único dato que se guarda de reportes anteriores y su diseño. Por lo tanto si queremos ver el reporte directamente desde el servidor nos indica que no se puede ejecutar el comando c:\...\ DatosReporte.xml.

7 EJECUCIÓN DESDE VISUAL ESTUDIO

En esta sección se pasa a describir un punto que se tiene que tener en cuenta para la ejecución, desde MS Visual Estudio, del producto implementado.

Los dos editores basan su configuración del ambiente en un archivo llamado el cual persiste en formato XML toda esta configuración, por lo tanto al momento de iniciarse la aplicación desde el entorno de MS Visual Estudio, debe de existir el archivo anteriormente comentado en la capeta bin/Debug o bin/Release, según el modo de ejecución.

Cabe destacar que en la misma solución se tienen los dos proyectos correspondientes a los dos Editores, se debe tener el archivo **vrs.config** en la carpeta correspondiente al editor y al modo de inicio (Debug o Release) que se seleccione.

VISUAL REPORTING SERVICES

ANEXO VII

MANUAL DEL USUARIO FINAL

ÍNDICE

1 INTRODUCCIÓN

Visual Reporting Services (VRS) le permitira la creación, edición y ejecución de reportes. Esto se hará a partir de un Meta-modelo de Negocio, mediante el cual integrarán los datos provenientes de los distintos sistemas que se quieran utilizar en el esquema. A través de los editores, se podrá generar modelos de negocios, representando los conceptos del mismo, y vinculando estos conceptos con los datos operacionales de la organización se podrán generar reportes con el fin de apoyar el análisis de los datos y la toma de decisiones.

El objetivo de este documento es describir el sistema, en particular el uso del *Editor del Modelo de Negocio*, presentando la forma de trabajar del usuario final. No se pretende detallar los comandos existentes para la edición de los esquemas. Ejemplos y detalles de estos se pueden encontrar en el documento **Lenguaje de Edición de Metadata.doc**.

Este manual presenta las tareas que puede realizar un usuario final usando el Editor del Modelo de Negocio, estas son:

- instalación del Editor del Modelo de Negocios,
- configuración de los ejemplos,
- como utilizar el Editor de Modelo de Negocios y puntos a tener en cuenta,
- como diseñar y generar reportes.

1.1 USUARIOS DE VRS

Se deberá disponer de un esquema del modelo de negocio (MN) que haya sido generado por un Usuario Programador de Metadata (o Usuario Experto, UE). Este esquema presenta los conceptos básicos del MN e integra los datos provistos por distintos sistemas de información utilizados por la organización para la que se crea el esquema. Este esquema lo toma el Usuario Generador y Consumidor de Reportes (o Usuario Final, UF) para extenderlo, agregando nuevos conceptos basados en los anteriores y generando reportes que reflejen los datos de estos conceptos.

El UF es quien conoce los conceptos del negocio, quien querrá modelar reportes para consumirlos, pero no tiene todos los conocimientos para instalar los editores, ni la información de donde provienen los datos que desea incluir en sus reportes.

El UE debe saber como acceder a los productos que permiten la ejecución de los editores, además de conocer la forma de acceder a los adaptadores de las FI y las funcionalidades y datos que estos ofrecen. También tiene conocimientos de los conceptos de negocio fuertemente ligados a las FI, pero su conocimiento es tan extenso como el del UF.

Las tareas fundamentales del usuario final son:

- Edición de Modelos de Negocios agregando nuevos conceptos.
- Generación de reportes.

Ambos usuarios podrán generar reportes o modificar la representación de un modelo de negocio en particular, pero el Usuario Final tendrá algunas restricciones debido al poco conocimiento que este tiene de las fuentes de información, sus adaptadores y como estos proveen la información. En particular no podrá crear entidades (conceptos) que accedan directamente a las fuentes de información (entidades servidas), sino que deberá definir nuevas entidades en base a las ya existentes (entidades relacionales).

La interfase del UF con el sistema las llamaremos Editor del Modelo de Negocios, la interacción con este editor se hará a través de comandos, los que conformaran un subconjunto de los aceptados para UE.

2 INSTALACIÓN DEL EDITOR

Aquí se describen los requerimientos para la instalación del sistema, los pasos que se deben seguir para la instalación del sistema, y como configurar los ejemplos disponibles. También se describe la instalación de una extensión de Reporting Services, la cual es requerida para poder generar reportes correctamente utilizando Reporting Services a través de los Editores.
Los puntos a tratar son:

- **Prerrequisitos**: donde se indica cuales son lo requerimientos de hardware y de software donde se probó el sistema.
- **Pasos de Instalación**: donde se comenta como se debe instalar el editor.

2.1 PRERREQUISITOS

Los requerimientos acá establecidos para la ejecución del sistema conforman las características del equipo con menor potencia en el que fue probado, no son estrictamente los requerimientos mínimos de instalación, pero si aseguran el correcto funcionamiento del mismo. Cabe destacar que alguno de los productos de software necesarios puede tener mayores requerimientos de hardware o software según edición y versión. Otro punto que se debe tener en cuenta son los adaptadores y las fuentes de información, tanto por los requerimientos de procesamiento, software o volumen de datos que manejen, características que pueden impactar negativa o positivamente en el desempeño del VRS.
Claramente no es lo mismo manejar 10.000 que 100.000 registros, o usar adaptadores o fuentes de información a los que se accedan a través de DLLs que utilizar otras que sean accedidos por Web Services.

2.1.1 Hardware

El sistema fue probado, con un funcionamiento aceptable en una PC con:

- 256 MB de RAM,
- Procesador de 1000 MHz
- Espacio en el disco solo para la aplicación Visual Reporting Services 5 MB

2.1.2 Software

En cuanto al software requerido para el correcto funcionamiento es

- MS Windows 2000 Server o posterior.
- .Net Framework 1.1
- MS SQL Server 2000.
- MS SQL Server 2000 Reporting Services
- ModificarConfiguracionRSExtenderDataSet, este producto forma parte del paquete entregado con Visual Reporting Services, y se describe su instalación en el punto Modificar Configuración de RS para Extender DataSet
- Adaptadores que permitan acceder a los datos de las Fuentes de Información

2.2 PASOS DE INSTALACIÓN

Como se indicó en los prerrequisitos los editores de VRS se ejecutan sobre una versión de MS Windows 2000 Server, también se requiere tener actualizados los Service Pack y tener una versión actualizada de Internet Explorer.

Como el producto fue desarrollado con MS Visual Studio 2003 el primer paso es instalar .Net Framework 1.1 y el último Service Pack de este producto, los cuales también se encuentran en CD de VRS.

Luego de instalar el .Net Framework se debe agregar el generador de reportes (reporteador), para la versión actual se utiliza MS SQL Server 2000 Reporting Services con el Service Pack 2, para el funcionamiento de este producto se requiere tener instalado MS SQL Server 2000 con el último Service Pack 4.

El último paso es instalar los editores de Visual Reporting Services, mediante el **instalador** provisto. Este instalador también nos permitirá instalar ModificarConfiguracionRSExtenderDataSet (o ExtenderOrigenDataSet), el cual es una extensión de RS al que se le agrega un nuevo origen de datos con el cual se interactúa con VRS.

En caso de no tener toda la información requerida en el momento de la instalación se puede ejecutar posteriormente el programa  ModificarConfiguracionRSExtenderDataSet.exe .
En el siguiente punto se detalla como instalar esta extensión y cuales son los requerimientos de software necesarios para instalarla.

En resumen los pasos para instalar el producto son:

a. Disponer de una maquina con una versión de Windows 2000 o posterior, con los Service Pack de Windows y de Internet Explorer.
b. Instalar .net Framework 1.1 y su último Service Pack.
c. Instalar MS SQL Services 2000 y su último Service Pack.
d. Instalar MS SQL Services 2000 Reporting Services y su último Service Pack
e. Ejecutar el instalador entregado el cual instala los editores.
f. Modificar la configuración de RS para extender DataSet usando la última ventana de la instalación.
g. Si es necesario modificar la configuración del Editor.
h. Sino se hizo en el punto “f”, se deberá instalar la extensión de MS RS mediante la aplicación ModificarConfiguracionRSExtenderDataSet.

Estos son los puntos para la instalación de los editores de Visual Reporting Services, en este caso es importante seguir el orden de la instalación, e instalar los Service Pack de cada producto de MS.

2.2.1 Modificar La Configuracion De RS Para Extender DataSet

ModificarConfiguracionRSExtenderDataSet debe ser instalado antes de intentar ejecutar reportes a través de los editores. Sino se ha hecho ocurrirá un error que no permitirá generar los reportes. La utilidad de este producto es extender MS Reporting Services para aceptar DataSets como origen de datos, puesto que los únicos tipos de orígenes de datos de RS son: SQL Server, Oracle, ODBC y OLE DB.

Esta extensión se debe realizar tanto en el Diseñador de RS como en el Servidor de RS, para ello se debe:

- Modificar 4 archivos de configuración, llamados:

 rssrvpolicy.config y  RSReportServer.config que por defecto están en la carpeta del Servidor de RS (C:\Archivos de programa\Microsoft SQL Server\MSSQL\Reporting Services\ ReportServer) y  rspreviewpolicy.config  RSReportDesigner.config ubicados por defecto en la capeta de herramientas de diseño de RS (C:\Archivos de programa\Microsoft SQL Server\80\Tools\Report Designer), con estos se incluye el nuevo origen de datos en el diseñador de RS el cual es utilizado en Visual Studio.

- copiar el archivo 

en la carpeta donde se encuentra el diseñador y el servidor de RS, este archivo es la implementación de la extensión de RS para aceptar el origen de datos DataSet.

Como los archivos de configuración de RS son archivos XML y es difícil su modificación, se generó un ejecutable el cual verifica la existencia de los archivo en la ubicación indicada y si existen los modifica. Basta que uno de los cuatro archivos no se encuentre para que no realice ningún cambio.

También, para prevenir posibles errores, los archivos originales no son borrados sino que se realiza un respaldo en el mismo directorio donde estén ubicados cambiando ".config" por "_VRS_N.BAK", donde "N" es un número que se va incrementando cada vez que se ejecute el programa, esto es con el fin de no sobre escribir archivos.

Por ejemplo el archivo original de **RSReportServer.config** luego de ejecutar este producto y si pueden realizar los cambios en los cuatro archivos pasara a llamarse **RSReportServer_VRS_0.BAK.** Es muy importante tener en cuenta esto por si se quiere volver a dejar la versión original de los archivos.

Advertencia 1

No remplazar los archivos *.config con los archivos de otra instalación, puesto que cada uno tiene un número de seguridad diferente, ni borrar los *._VRS_**0**.BAK, pues estos se guardan como respaldo de los archivos originales.

Advertencia 2

Si ejecuta 2 veces la aplicación se agregará dos veces las entradas en los archivo de configuración y se generan un total de 8 archivos (aunque esta acción fue probada y el funcionamiento de RS fue correcto tanto en el servidor como en el diseñador) se recomienda antes de ejecutar ModificarConfiguracionRSExtenderDataSet por segunda vez, restaurar los archivos originales; es decir eliminar los cuatro archivos .config y restaurarlos desde los respaldos, que son los archivos con los nombre ***_VRS_0.BAK** (solo los que tiene el número 0) ponerles el nombre ***.config**

En resumen lo que se hace para habilitar este nuevo origen de datos es:

- Modificar los archivos de configuración del diseñador y del servidor de MS RS,
- Copiar del archivo `RSExtensionOrigenDatosDataSet.dll` en las carpetas: `C:\Archivos de programa\Microsoft SQL Server\MSSQL\Reporting\bin`, `C:\Archivos de programa\Microsoft SQL Server\80\Tools\Report Designer`, o las que correspondan según su instalación de MS RS.

2.2.1.1 Prerrequisitos de ModificarConfiguracionRSExtenderDataSet:

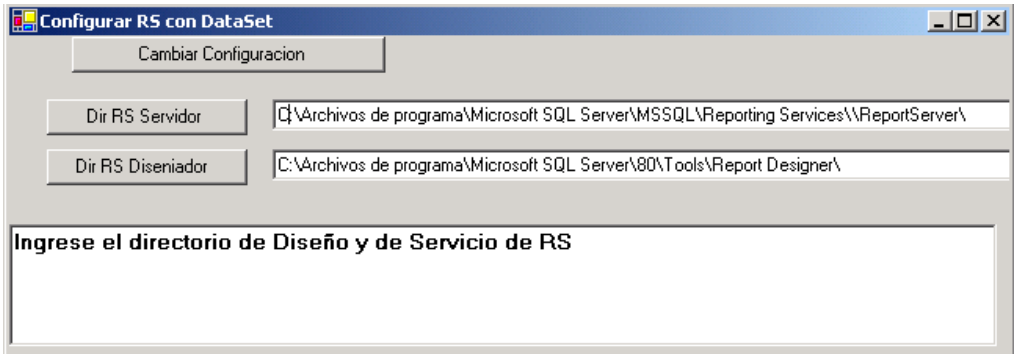
Los prerrequisitos para la instalación de ModificarConfiguracionRSExtenderDataSet son:

- Verifique que se realizaron los pasos previos comentados en el punto **Pasos de Instalación.**
- Verifique que este **cerrado** MS Visual Estudio y el Administrador de Informes de Reporting Services.
- **Verifique la ubicación exacta de los cuatro archivos nombrados anteriormente**.
- Verificar que en el App.config del producto esté indicado en forma correcta la ubicación del archivo RSExtensionOrigenDatosDataSet.DLL

2.2.1.2 Instalación

Este producto se puede instalar junto con el editor de VRS o en forma posterior, la ventana que se muestra en cualquiera de los dos casos es la misma que se muestra en la imagen siguiente.
Los pasos para instala este producto sino fue instalado junto con el editor son:

1. Verificar que se cumplan los prerrequisitos para la ejecución de ModificarConfiguracionRSExtenderDataSet.
2. Tener la ubicación exacta de los archivos de configuración de RS.
3. Si es necesario puede modificar el archivo `App.config` para indicar donde se encuentra RSExtencionOrigenDatosDataSet.dll, por defecto están en el mismo directorio que los editores.
4. Luego ejecutar el archivo `ModificarConfiguracionRSExtenderDataSet` el cual abre la ventana



5. Ingresar la carpeta donde se encuentran los dos primeros archivos en el primer cuadro de texto.
6. Ingresar la carpeta donde se encuentran los dos segundos archivos en el segundo cuadro de texto.
7. Seleccionar el botón Cambiar Configuración.

En caso de mal funcionamiento de RS recupere los archivos de configuración, en caso de no tenerlos reinstale RS (esto hará que se reescriban los archivo de configuración) y luego ejecute la aplicación ModificarConfiguracionRSExtenderDataSet nuevamente.

3 CONFIGURACIÓN DEL ENTORNO

Previo a la descripción de la configuración del editor se hará una descripción de algunos archivos o información requerida que se puede configurar.

Los **Archivos de Configuración** `app.config` y `vrs.config` contienen datos para el funcionamiento del editor, este último archivo pueden ser editados desde la ventana de configuración de la aplicación, como se indica en esta sección en el siguiente punto.

La generación de reportes se basa en el uso tres **plantillas** las cuales se puede configurar su ubicación, más aclaraciones sobre plantillas se darán en el punto **Diseño y Generación de Reportes**.

La generación de un reporte resulta de la combinación del **diseño del reporte** y los **datos de la entidad** que se mostraran en el reporte (datos del reporte), esta información se guarda en archivos temporales que luego son eliminados. Los puntos que se tocarán en esta sección son dos archivos que participan en la generación de un reporte y descripción de la información que se puede configurar en los editores

3.1 CONFIGURACIÓN DEL EDITOR DEL MODELO DE NEGOCIO

Estando en la ventana principal del *Editor del Modelo de Negocio* se podrá seleccionar el ítem Opciones dentro del menú Configuración, aquí se podrán configurar las ubicaciones de algunos archivos utilizados por VRS y las opciones de log de errores disponibles. Al seleccionar esta opción se abrirá la ventana de configuración la cual tiene las fichas: Fuentes de Información; Plantillas de Reportes; Log de Errores; Generación de Reportes.

Para modificar cualquier opción basta ingresar el dato y seleccionar el botón Aceptar.

Se debe tener en cuenta que el botón Aceptar guarda los cambios realizados en todas las fichas, no solo en la ficha actual, y cierra la ventana.

3.1.1 Configuración del Repositorio de Proxies

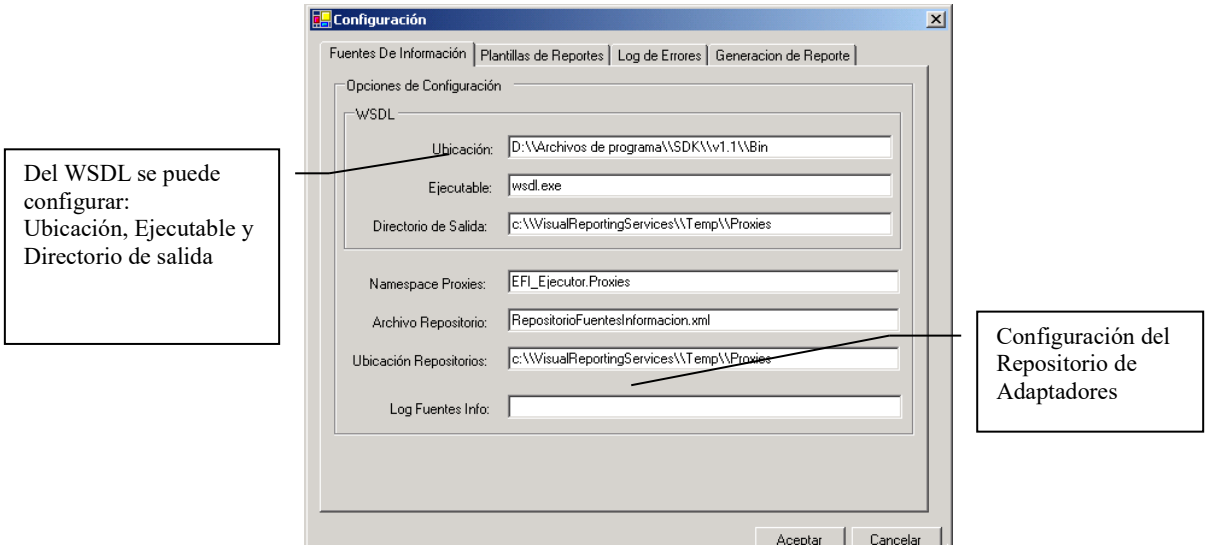
Los datos que se pueden modificar en esta ficha conforman las distintas opciones de configuración del Repositorio de Proxies. Este componente se encarga del almacenamiento de los proxies para las fuentes de información. Es conveniente que la modificación de estos datos quede a cargo de un usuario experto y sea realizada por un usuario final.

Importante: Se recomienda no cambiar estos datos una vez que se hayan creado fuentes de información en los esquemas, dado que al cambiar estos datos se deberán volver a definir estas fuentes en los esquemas para que el acceso a ellas se haga en forma correcta.

Los parámetros configurables son los siguientes:

- Ubicación del WSDL, que indica donde se encuentra el ejecutable del aplicativo WSDL
- Ejecutable del WSDL, que indica el nombre del ejecutable del aplicativo WSDL.
- Directorio de salida, indica donde se ubican los archivos de salida del WSDL.
- Namespace de los proxies, indica en que namespace se generan los proxies,
- Indica la ubicación del Repositorio de Adaptadores para las FI.
- Archivo del repositorio en el que se almacenan los datos del repositorio.
- Ubicación del repositorio, ubicación del archivo del repositorio.
- Log Fuentes Info, archivo de logueo del generador de proxies.

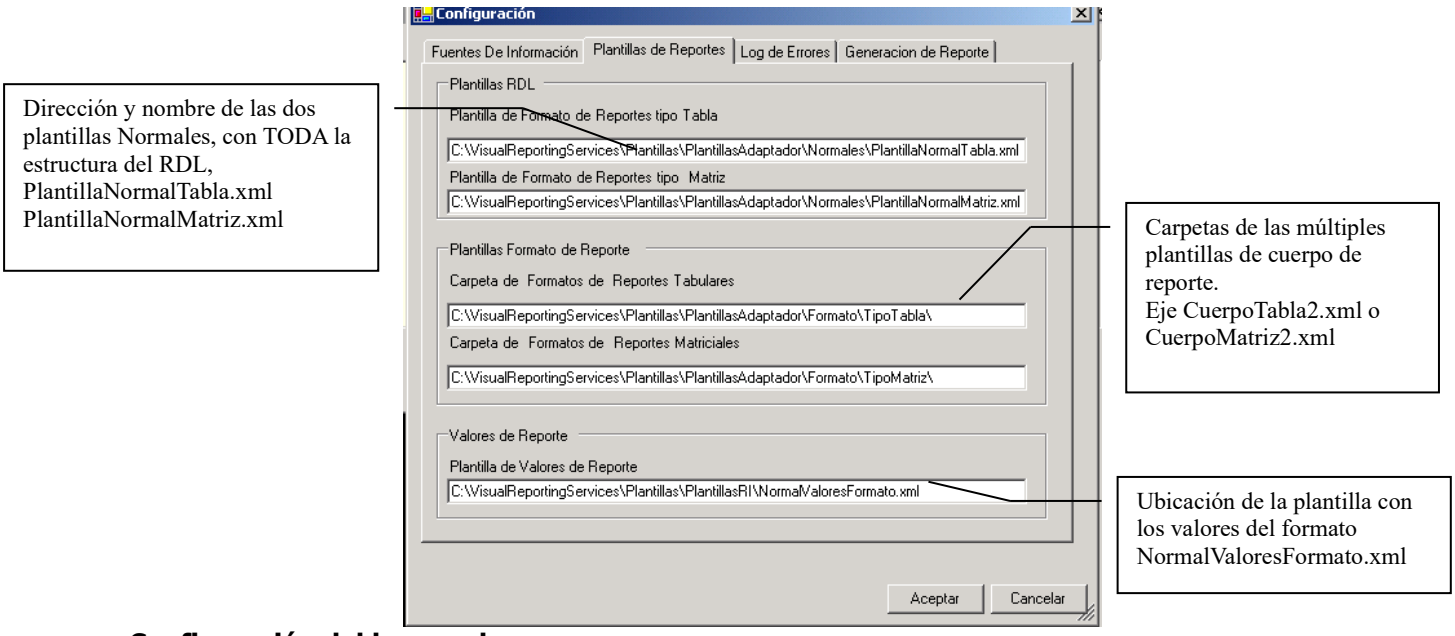
En la siguiente imagen se muestra la pantalla con la información por defecto



3.1.2 Configuración de Diseño del Reporte

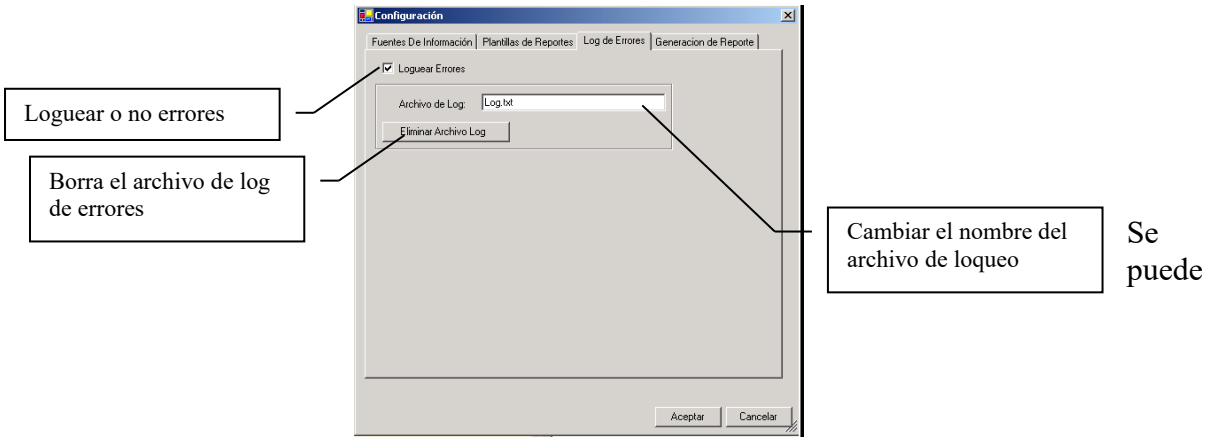
Cada editor podrá tener diferentes plantillas o las mismas pero ubicadas en diferentes carpetas. Para indicar la ubicación de las plantillas se utiliza la ficha de Plantillas de Reporte, para llegar a ella en el editor se debe seleccionar Configuración / Opciones / Plantillas de Reportes, y aquí se podrá ingresar la ubicación de las plantillas. También desde esta ficha se debe indicar donde están instaladas esas carpetas sino se realizo la instalación por defecto.

En la siguiente ventana se muestra la ficha con los valores por defecto:



Configuración del logueo de errores

Mediante el menú de configuración (Configurar/ Opciones) en el Editor se puede configurar el logueo de errores al ejecutar un conjunto de comandos, es decir que los errores podrán ser guardados para un posterior análisis.



Se puede configurar:

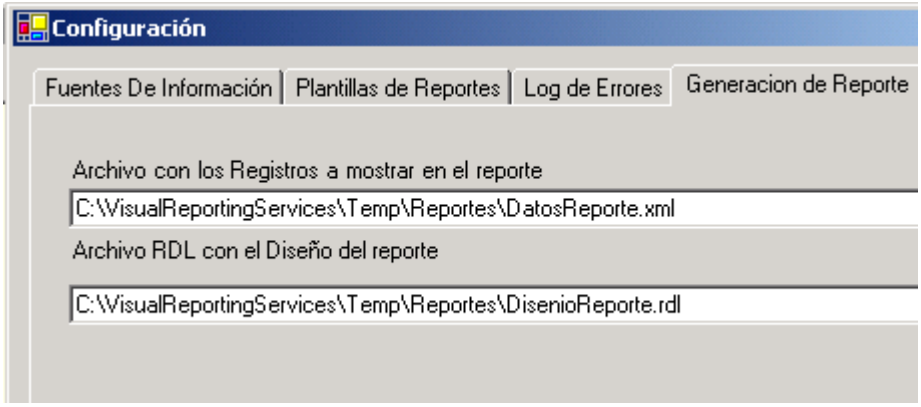
- Loguear o no errores,
- Indicar el nombre de archivo de log
- Eliminar el archivo de log de errores.

3.1.4 Configuración de la Generación del Reporte

Esta ventana permite configurar la carpeta y nombre donde se mantienen temporalmente los archivos necesarios para la generación del reporte, por defecto el nombre del primer

archivo (archivo de datos del reporte) es DatosReportes.xml y el del segundo (archivo con el diseño del reporte) es DisenioReporte.xml.

La siguiente imagen muestra la ventana con los nombres y ubicación por defecto de estos archivos.



4 INSTALACIÓN Y CONFIGURACIÓN DE EJEMPLOS

En esta sección se indica donde encontrar los archivo de ejemplos y como configurar las fuentes de información que son utilizadas en los ejemplos disponibles para el sistema. Los archivos de ejemplo son copiados, cuando se instalan los editores de VRS, en el mismo lugar que el producto dentro de la sub carpeta EjemplosDeScript.
Se tienen los siguientes archivos de ejemplo para la el Editor de Modelo de Negocio:

- Ej1-titularesMagma.txt
- Ej2-Cines.txt
- Ej3-MovimientosDeUnCliente.txt
- Ej4-pruebasRSMatrizYAgrupar.txt

Dentro de estos ejemplos se utilizan las siguientes fuentes de información, las cuales tiene que haber sido instaladas previamente para el buen funcionamiento de los script:

- WSMagma
- WebServiceCines
- CapaDatos
- AccesoADatos
- RemotingCines

Estas fuentes de información no son instaladas con el producto pero se encuentran en el mismo CD de VRS bajo la carpeta WebServicesEjemplo.

Utilizan bases de datos de MS Access y SQL Server. Como base de datos de SQL Server se tiene la base Magma_Demo y Cines, que deben estar disponibles en MS SQL Server local.

El **WSMagma** accede a la base de datos de Magma_Demo, para utilizar este WS se debe ubicar en la url `http://localhost/WSMagma` y presenta los siguientes métodos:

```
public DataSet ObtenerTitulares()
public DataSet ObtenerMovimientos()
public DataSet ObtenerSaldoHastaFecha(string fecha, long
cod_tit)
public DataSet
ObtenerMovimientoTitularDespuesDEFecha(string fecha, long
cod_tit)
```

Para utilizar esta FI se deberá compartir para la web (Web Sharing) la carpeta en la que se encuentre el WS con el nombre WSMagma y dar los accesos correspondientes a estas

carpetas. También se deberá tener instalada la base de datos Magma_Demo, para la cual se provee un back up.

El WS **WebServiceCines** ubicado en la url http://localhost/WebServiceCines el cual permite acceder a la base de datos Sabor.MDB, los métodos que ofrece son:

```
public DataSet ObtenerClientes()
public DataSet ObtenerClientesPorNombre(string pNombre)
public DataSet ObtenerClientesConFiltros(string pNombre,
string pIngresoDespuesDe, string pIngresoAntesDe)
public DataSet ObtenerPeliculas(long pId, string pNombre,
bool pProximoEstreno, DateTime pFechaEstreno, decimal
pDuracionMinima)
public DataSet ObtenerPeliculasGenero(string pGenero)
public DataSet ObtenerSalas()
public DataSet ObtenerComplejos()
public DataSet ObtenerProgramaciones()
public DataSet ObtenerPeliculaIdGenero()
```

El servicio **CapaAccesoADatos** permite acceder también a la base de datos Sabor.MDB pero por medio de una DLL, para ello se tiene que ubicar el archivo CapaDatos.dll en el directorio `D:\ProyectoDeGradoFI\WebServicesEjemplo\CapaDatos\bin\Debug`

Las funcionalidades ofrecidas por este servicio son:

```
public static DataSet ObtenerClientes()
public static DataSet ObtenerClientesPorNombre(string
pNombre)
public static DataSet ObtenerClientesPorNombre(string
pNombre, string pIngresoDespuesDe, string pIngresoAntesDe)
public static DataSet ObtenerPeliculas()
public static DataSet ObtenerPeliculas(long pId, string
pNombre, bool pProximoEstreno, DateTime pFechaEstreno, decimal
pDuracionMinima)
public static DataSet ObtenerPeliculasGenero(string
pGenero)
public static DataSet ObtenerSalas()
public static DataSet ObtenerProgramaciones()
public static DataSet ObtenerPeliculaIdGenero()
```

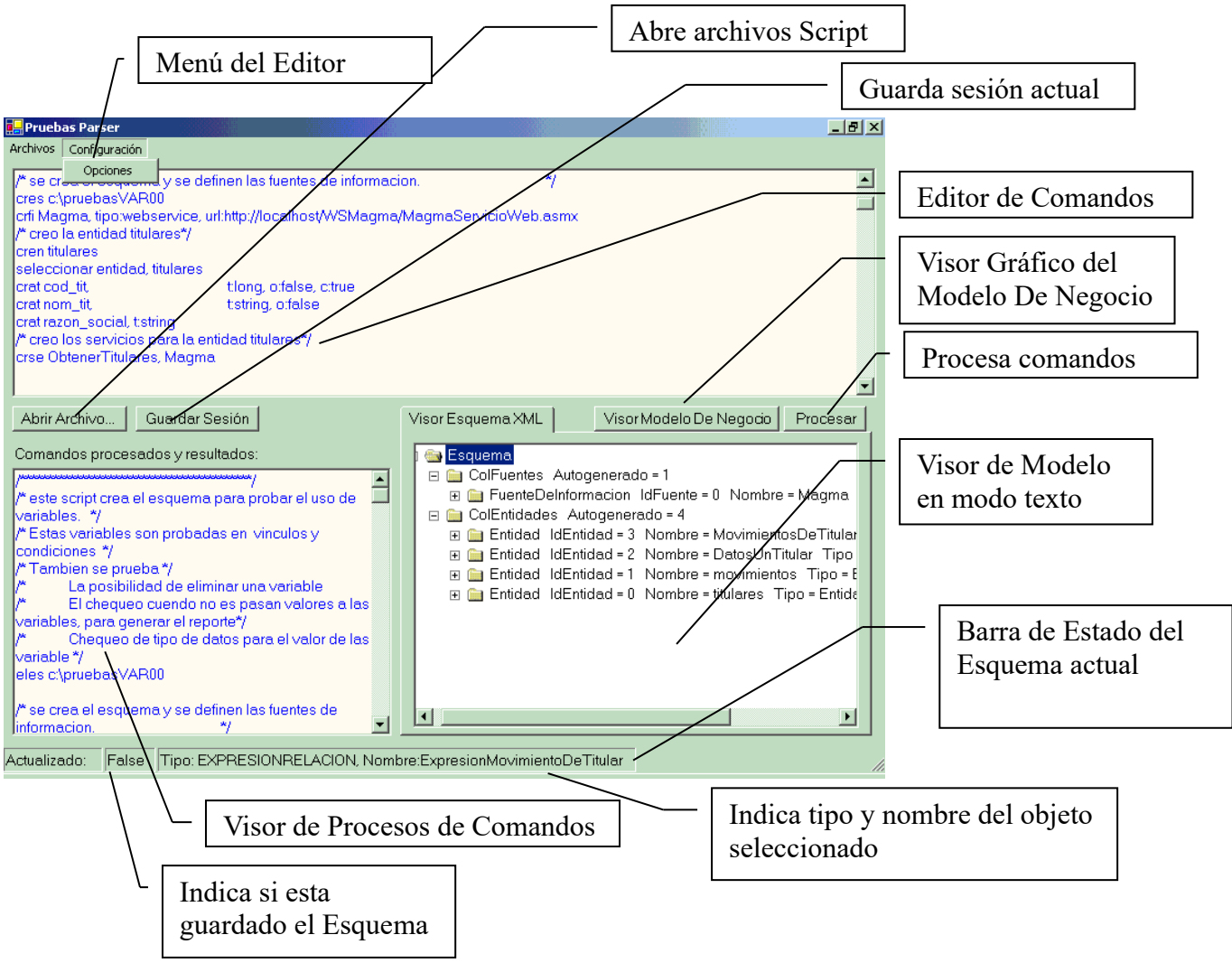
También se encuentran disponibles las fuentes de información AccesoASQLServer y RemotingCines, estas proveen servicios con las mismas firmas que los provistos por CapaDatos, diferenciándose de esta por el origen de los datos, los datos y la tecnología mediante la cual acceden a ellos.

Los directorios en los que sean ubicadas estas fuentes deberán ser tenidos en cuenta cuando se definan las fuentes de información en los scripts de ejemplos. La ubicación por defecto utilizada en estos scripts es D:\ProyectoDeGradoFI\WebServicesEjemplo\.

Se recomienda que la instalación de las fuentes de información sea llevada a cabo por un usuario experto, por lo que se pueden encontrar más detalles de cómo configurarlas en el “Manual de Programador”.

5 UTILIZAR EL EDITOR DE MODELO DE NEGOCIOS

El UF interactúa con el sistema usando uno de los dos editores de Visual Reporting Services, este editor es denominado Editor de Modelo de Negocios y consta de: tres áreas de trabajo, un menú, cuatro botones y una barra de estado.



Las áreas de trabajo son las siguientes:

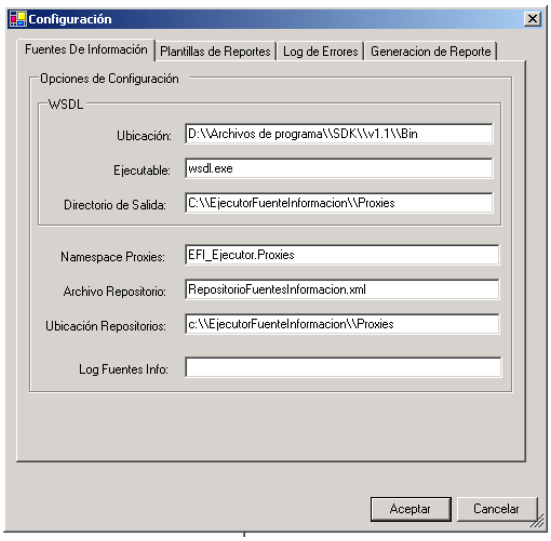
- **Editor de Comandos** esta área permite ingresar los comandos a ejecutar, para editar el modelo de negocios o generar reportes, estos comandos están descriptos en el documento **5 - Lenguaje de Edición de Metadata.doc**
- **Visor de Procesador de Comandos** en esta área se muestra el resultado de la ejecución de los comandos, en caso que se ingrese un comando el cual no pueda ejecutar, en esta área se mostrará un mensaje de error, si es correcto entonces se muestra el comando ingresado.
- **Visor de Esquema** muestra en un árbol con texto, los datos completos del esquema que se esta editando actualmente. Se puede ver el archivo **Análisis de la metadata y su Persistencia.doc** para tener una descripción completa de la información acá presentada.

En el menú se presentan dos opciones que son Archivo y Configuración, en donde:

- **Archivo** se divide en dos opciones:

- **Guardar Sesión Como**, la cual es equivalente al botón **Guardar Sesión** y la tarea que realiza es guardar el resultado de la sesión actual. Cada sesión comienza cuando se abre por última vez el editor y finaliza cuando se cierra, la información que guarda es todo lo mostrado en el área Visor de Procesador de Comandos.
- **Abrir** es equivalente al botón **Abrir Archivo**, en este caso permite abrir un archivo con comandos para ser ejecutados en el editor, todo el contenido del archivo abierto será colocado en el área Editor de Comandos.

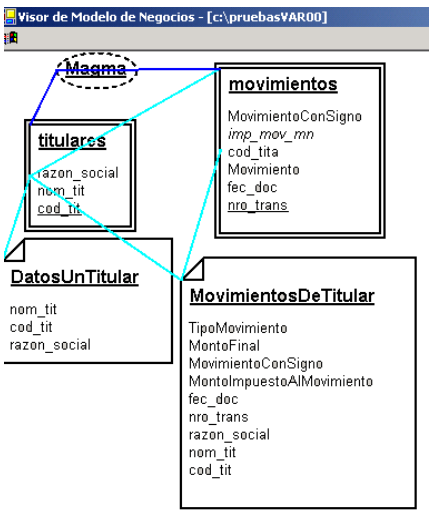
- **Configuración** solo tiene un sub menú que es **Opciones** el cual abre una nueva ventana que permite configurar las variables de entorno de trabajo. Esta ventana esta dividida en las fichas **Fuente de Información**, **Plantillas de Reportes**, **Log de Errores** y **Generación de Reportes**.



Este formulario se describió más profundamente en este documento más arriba.

Los botones restantes son dos **Visor de Modelo de Negocio** y **Procesar**.

El botón **Visor Modelo de Negocios**, permite generar una representación gráfica del esquema que se esta editando actualmente, como la que se muestra en la figura de más abajo.



En esta representación gráfica se visualiza el esquema con:

- Las **Fuentes de Información** con óvalos de líneas punteadas y su nombre

- Las **Entidades** representadas con rectángulos con diferentes bordes según el tipo de entidades que sea. Dentro del recuadro de cada entidad se representa sus atributos, donde cada atributos tiene diferente formato según sus valor en las propiedades: clave, oculto horizontal o múltiples.
- Las **Relaciones** entre las entidades y las fuentes de información.

El botón **Procesar** permite ejecutar una lista de comandos ingresada en el área Editor de Comandos, cada comando de la lista termina con un enter, en caso que los comandos no sean correctos o no se puedan aplicar por no tener el objeto correctamente seleccionado, se interrumpe la ejecución y se muestra un error en el área Visor de Procesador de Comandos.

La **barra de estado**, que se encuentra en la parte inferior de la pantalla, indica si el esquema esta guardado o no, además indica el nombre y tipo del objeto seleccionado actualmente.

5.1 Forma de Trabajar

La forma de trabajar con el Editor de Modelo de Negocios se describe en los siguientes pasos:

a. Si es necesario se configuro el ambiente de trabajo usando la opción de menú Configuración/Opciones.
b. Abro un esquema creado anteriormente.
c. Ingresar uno o muchos renglones con comandos, en el Editor de Comandos, los cuales pueden ser ingresados a mano o provenir desde un archivo.
d. Asegurarse de tener un enter (nuevo renglón) al final del último comando.
e. Seleccionar el botón Procesar para ejecutar los comandos.
f. Verificar si los comandos se ejecutaron correctamente en el área del visor de Procesador de Comandos.
g. Se puede visualizar el estado actual usando el:
   Visor de Esquema que muestra en forma detallada pero en modo texto o
   Visor de Modelo de Negocio que muestra en forma gráfica pero menos detallada.
h. Se puede guardar el esquema generado para una posterior edición.
i. Se puede generar reportes con alguno de los conceptos representados.

5.2 Modificar una Representación de Modelos de Negocio

Luego de tener instalado y configurado el ambiente de trabajo como se indicó en puntos anteriores, se pueden comenzar a utilizar el Editor de Modelo de Negocios.

A grandes rasgos los pasos para modificar la representación de un modelo de negocios con el Editor de Modelo de Negocio son:

a. Abrir un esquema creado anteriormente.
b. Crear nuevas Entidades, que se correspondan con nuevos conceptos que habitualmente se usan en el negocio, y en cada Entidad.
    a. Agregarle los atributos.
    b. Crear las relaciones entre las entidades
    c. Crea una expresión de relación.
    d. Crear los vínculos con los cálculos de cada atributos para obtener los datos de la entidad

e. Si el concepto lo requiere agregar, alguna o todas, las operaciones de: Agrupación, Ordenación, Criterios o Variables.

c. En cualquier momento puede configurar el diseño del reporte asociado a una Entidad definida anteriormente, o dejar el diseño por defecto
d. En cualquier momento puede generar un reporte con una entidad ya creada que representa un concepto del negocio.
e. En cualquier momento se puede guardar el Esquema para una posterior edición.

A continuación detallamos cada uno de estos pasos y otras operaciones alternativas que se puedan realizar, indicando cuales son los comandos que se pueden aplicar en cada momento, no se indicaran los parámetros ni la utilidad de cada comando para ello ver el documento **5 - Lenguaje de Edición de Meta-Modelo.doc**

5.2.1 Abrir Modelo de Negocio (Esquema)

Cada esquema representa un modelo de negocio, por lo tanto la edición del modelo de negocio se hace por medio de los comandos que afectan al esquema, estos comandos son:

- renombrarEsquema o renES
- eliminarEsquema o elES
- abrirEsquema o abES
- guardarEsquema o guES
- cerrarEsquema o ceES

5.2.2 Edición de Conceptos del Negocio (Entidades)

Se puede editar nuevos conceptos luego de abierto el esquema, cada concepto va a estar representado por alguno de los tipos de entidad.

Los tipos de conceptos que se pueden representar con este editor son: EntidadesRelacionales o EntidadesReportes; dependiendo del calculo que se deba realizar para obtener los datos de cada atributo de la entidad.

Los comandos que se pueden aplicar una vez seleccionada una entidad son los comandos de: Atributos, Relaciones y dependiendo el tipo de entidad se podrán aplicar los comandos de Expresiones de Relación y/o los de Variables.

5.2.2.1 Entidades Relacionales

Una Entidad Relacional se relaciona con un concepto el cual para obtener sus datos depende de los datos de otro concepto definido previamente, esta relacionado entre conceptos se representa en una Expresión de Relación.

Las características de las Expresiones de Relación se indicarán más adelante.

5.2.2.2 Entidades Reportes

Un concepto pasa a representarse por una Entidad Relacional cuando se crea una variable en la Entidad Relacional.

Las características de las variables se indicarán más adelante.

5.2.3 Datos Relevantes de los Conceptos (Atributos)

Cada concepto contendrá un conjunto de datos relevantes los cuales son representados por los atributos, es por ello que una vez creada una entidad se debe crear los atributos de esa entidad.

Los comandos que se pueden aplicar sobre los atributos son:

- crearAtributo (crAt)
- modificarAtributo (modAt)
- eliminarAtributo (elAt)

5.2.4 Conexiones entre Conceptos (Relaciones)

La definición de Relaciones permite representar la conexión que existe entre dos conceptos los cuales tiene que haber sido definidos previamente.

Los comandos que se pueden aplicar sobre las Relaciones son:

- crearRelacion (crRe)
- modificarRelacion (crdRe)
- eliminarRelacion (elRe)

5.2.5 Operaciones entre los Conceptos (Expresiones)

Las diferentes operaciones que se pueden realizar entre dos o más conceptos del negocio se representan por medio de las Expresión de Relación, en estas expresiones es donde se encuentran las operaciones de agrupamiento y/ o de ordenación y/ o condiciones que deben de cumplir los datos de la entidad, además de estas operaciones también permite definir los cálculos vinculados con cada atributo de la entidad actual.

Los comandos que se pueden aplicar sobre las expresiones son:

- crearExpresionRelacion (crEx)*
- modificarExpresionRelacion (modEx)*
- eliminarExpresionRelacion (elEx)*

5.2.6 Representación de Cálculos de los Atributos (Vínculos)

Los vínculos permite representar el cálculo de los datos para cada atributo, estos datos se podrán obtener de varias formas diferentes, las cuales pueden ser: una constante, una variable, por medio de la relación con un atributo de otra entidad, o por aplicar una función o una operación a los puntos antes comentados.

La única operación posible sobre los vínculos es:

- vincular

5.2.7 Variables

Es común que un concepto requiera de información externa para determinar los datos de la entidad que lo representa, por ejemplo el identificador de un cliente si quiere obtener los movimientos de un solo cliente, para estos casos se puede saber que operaciones se realizará con esta información pero solo se conoce su valor en el momento de generar el reporte, es para la representación de este tipo de conceptos que se requiere definir una variable en la entidad que los representa.

Los comandos sobre las variables son:

- crearVariable (crVa)
- eliminarVariable (elVa)

5.2.8 Otras Operaciones

Otras operaciones comunes en los conceptos de negocio son las operaciones de ordenamiento de datos, o la agrupación y las condiciones.

Los comandos que se pueden aplicar sobre una expresión de relación para realizar estas operaciones son:

- condicion (co)
- agrupar (ag)
- ordenar (or)

5.3 EJEMPLO

El siguiente ejemplo muestra todos los pasos a realizar y comandos a ejecutar para la generación de un Esquema y un reporte, este ejemplo está en forma más extendida en el documento **Lenguaje de Edición de Metadata.doc**.

Los pasos sugeridos en puntos anteriores para la generación del modelo de negocio y en particular un esquema son:

1. Abrir un esquema nuevo.
2. Crear las Entidades, que se correspondan con los conceptos que habitualmente se usan en el negocio, y en cada entidad.
    a. Agregarle los atributos
    b. Crear las relaciones entre las entidades
    c. Crear una expresión de relación.
    d. Agregar los cálculos que permitan vincular cada atributos con los datos de la entidad
    e. Si el concepto lo requiere agregar operaciones de: Agrupación, Ordenación, Criterios o Variables.
3. Se puede configurar el diseño del reporte asociado a una entidad o dejar el diseño por defecto
4. Una vez creada una entidad que representa un concepto del negocio, se puede generar un reporte con los datos de la entidad.
5. Cuando esta definido un modelo de negocio se puede guardar para una posterior edición.

Supongamos que previamente tenemos el esquema TitularesMagma, el cual usa la FI `MagmaServicioWeb` definida con un Web Service, generado por el siguiente script,

```
cres c:\TitularesMagma
crfi Magma, tipo:webservice,
url:http://localhost/WSMagma/MagmaServicioWeb.asmx
/* creo la entidad titulares*/
cren titulares
seleccionar entidad, titulares
crat cod_tit,             t:long, o:false, c:true
crat nom_tit,             t:string, o:false
crat razon_social, t:string
/* creo los servicios para la entidad titulares*/
crse ObtenerTitulares, Magma

liberar
/* guarda el esquema en c:\TitularesMagma.xml
gues
```

Supongamos que el UF desea obtener los datos de un Titular en particular el cual será seleccionado por su `cod_tit`

Pasamos a detallar cada uno de estos pasos, con los respectivos comandos.

1. Abrir un esquema y guardarlo

```
abes c:\TitularesMagma
gues
```

2. Crear las Entidades, que se corresponde con los conceptos que habitualmente se usan en el negocio.

```
cren DatosUnTitular
```

3. Agregarle a la entidad los atributos, previa selección de la entidad a modificar.

```
seleccionar entidad, DatosUnTitular
crat cod_tit, t:long, o:false, c:false
crat nom_tit, t:string, o:false
crat razon_social, t:string
```

4. Crear las relaciones entre las entidades

```
crearRelacion RelacionDatosUnTitular, n:titulares, al:cod_tit,c:Na1
```

5. Crear una expresión de relación.

Para este ejemplo se crea un servicio llamado `ExpresionDatosUnTitular`

```
crearExpresionRelacion ExpresionDatosUnTitular
```

6. Agregar a la expresión seleccionada, los cálculos que permitan vincular cada atributos con los datos de la entidad

```
seleccionar EXPRESIONRELACION, ExpresionDatosUnTitular

vincular cod_tit,          RelacionDatosUnTitular.cod_tit
vincular nom_tit,       RelacionDatosUnTitular.nom_tit
vincular razon_social,     RelacionDatosUnTitular.razon_social
```

7. Si el concepto lo requiere agregar operaciones de Agrupación, Ordenación, Criterios o Variables.

```
crva @NroTitular, t:long
condicion individual,      RelacionDatosUnTitular.cod_tit = @NroTitular
```

8. Se puede configurar el diseño del reporte asociado a una entidad o dejar el diseño por defecto, también se puede configurar otra serie de características en cuanto al formato del reporte, como ser:

- Ancho de columnas

```
ModificarAnchoColumna cod_tit, 3
ModificarAnchoColumna nom_tit, 5
```

- Formato del reporte para cambia el formato del reporte (`CuerpoTabla2`), tamaño de la hoja y del los márgenes, indicar si se muestra un encabezado y un pie y el logo de la empresa (`c:\Logo.jpg`):

```
ModificarFormatoReporte CuerpoTabla2, 30, 20, 5, 6, false ,true , false , 'Mi
Titular', 1, 2, 3, 4,'c:\Logo.jpg'
```

9. Una vez creada una entidad que representa un concepto del negocio, y realizadas las configuraciones requeridas, se puede generar un reporte con los datos de la entidad.

En este caso como la entidad tiene una variable en el momento de generar el reporte se debe indicar el valor de la variable.

```
generarreporte DatosUnTitular, @NroTitular:10
```

6 DISEÑO Y GENERACIÓN DE REPORTES

En la generación de los reportes hay varios puntos que son de interés, en esta sección analizaremos los siguientes puntos:

- **Plantillas de Reportes**: muestra como cambiar la presentación de las diferentes áreas del reporte
- Metodología para **generar reportes**, donde se describa los pasos a seguir para la generación del mismo.
- **Tipos de reportes**: donde se describen los diferentes tipos de reportes y como pasar de uno a otro.
- Detalle de como son mantenidos tanto los reportes como sus diseños.

6.1 PLANTILLAS DE REPORTES

Una plantilla de reporte es un archivo, el cual sirve como modelo para la generación de un reporte o un área particular del reporte, la mayor parte de la información de las plantilla no cambia solo cambian algunos valores y/o alguna información menor, la cual debe mantener el formato de la plantilla; la plantilla aporta estructura y valores de inicialización a la generación del diseño del reporte. Por ejemplo en el caso de los atributos, la plantilla mantiene la siguiente información:

```
<TableCells>
	<TableCell>
		<ReportItems>
			<Textbox Name="Id">
				+<Style>
				<Value>=Fields!Id.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
</TableCells>
```

Al generarse el diseño del reporte vemos la información siguiente, donde se mantiene el mismo formato que en la plantilla pero se agregaron tanto nodos TableCell como atributos de la entidad existen, colocando el nombre de cada atributo como nombre de un cuadro de texto y de un campo. Se resalta en negrita los cambios realizados:

```
<TableCells>
	<TableCell>
		<ReportItems>
			<Textbox Name="Cedula">
				+<Style>
				<Value>=Fields!Cedula.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
		<TableCell>
		<ReportItems>
			<Textbox Name="Nombre">
				+<Style>
				<Value>=Fields!Nombre.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
		<TableCell>
		<ReportItems>
			<Textbox Name="Direccion">
				+<Style>
				<Value>=Fields!Direccion.Value</Value>
			</Textbox>
		</ReportItems>
	</TableCell>
</TableCells>
```

Previa a la generación de un reporte se debe crear las plantillas de formato de reporte; excepto que se desee usar las plantillas prediseñadas que vienen con el sistema.

En la generación del formato del reporte se utilizan tres tipos de plantillas diferentes, que son Plantillas Normales de RDL, Plantillas de Representación Intermedia y Plantillas de Formato de Cuerpo.

6.1.1 Plantillas Normales de RDL

Esta plantilla dará el formato de reporte por defecto, por lo tanto todas aquellas entidades que no modifiquen su formato tendrán el que aporta esta plantilla.

6.1.2 Plantillas de Representación Intermedia

Esta plantilla es usada para mantener una representación genérica e independiente de los reportes. Esta plantilla mantendrá información para la configuración de la hoja y el nombre del la Plantilla de Formato de Cuerpo de Reporte.

No es conveniente editar esta plantilla y en caso de hacerlo muy probablemente se tenga que editar el código de la aplicación.

6.1.3 Plantillas de Formato de Cuerpo

Son plantillas que determinan como serán mostrados los datos en cuanto a colores, tipos, de letras bordes, sombreados, etc., es decir que permiten cambiar la presentación estética del reporte.

Es muy común crear este tipo de plantillas, es por ello que se implemento un método de simplifique el hecho de generar estas plantillas, los pasos son:

a. Generar un archivo RDL con el formato que se quiera aplicar a los reportes, para ello ver el documento **8 - Manual de Programador.doc**

b. Ejecutamos el comando `CrearPlantillaCuerpoReporte` para genera una plantilla de cuerpo de reporte nueva, esta plantilla es guardada en el lugar indicado por el Editor en el menú Configuración/ Opciones/ Plantillas de Reportes/ Plantillas de Formatos de Reportes

c. Una vez creada la plantilla de formato de reporte, ejecutando el comando `ModificarFormatoReporte` se puede relacionar un formato de reporte a una entidad.

6.2 Generación De Reportes

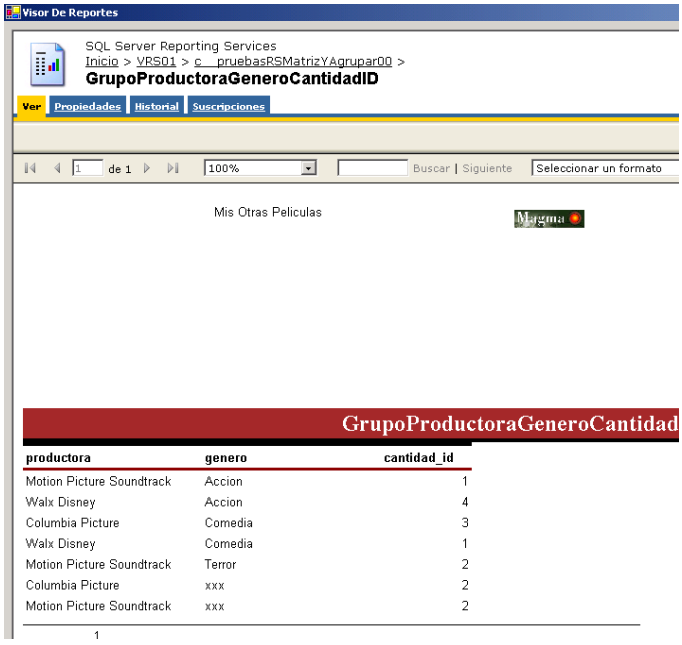
Una vez abierto un esquema el cual contiene un conjunto de entidades, podemos generar un reporte con el formato deseado, y configurar algunos datos a mostrar en ese reporte. Los pasos son:

a. Modificar el formato del reporte de una entidad con el comando ModificarFormatoReporte
b. Generar el reporte con el comando generarreporte
c. Guardar o imprimir el reporte.

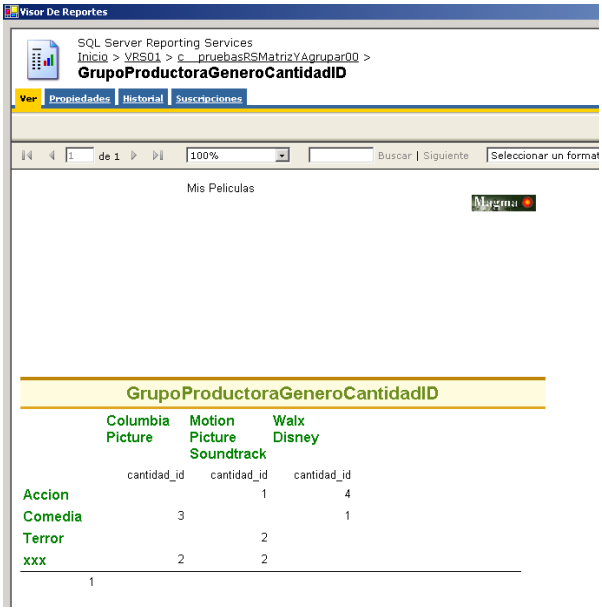
6.2.1 Tipo de Reportes

Existen dos tipos de reportes que son los reportes tipo matriz y los reportes tipo tabla, los cuales mostramos en los ejemplos siguientes:

Reporte tipo Tabla



Reporte tipo Matriz



La diferencia entre ellos es que los reportes tipo tabla tienen sus datos en una serie de columnas donde el título de cada columna es fijo, en cambio los reportes tipo matriz tiene columnas y filas con títulos que dependen de los datos de la entidad relacionada.

Los reportes más comunes son los de tipo tabla, pero si se desea generar reportes tipo matriz entonces la entidad a reportear debe cumplir tres condiciones que son excluyentes:

- Debe existir por lo menos un atributo que tenga valor true en la propiedad horizontal. Los datos asociados a estos atributos se mostrarán como títulos de columna.
- Se debe agregar una operación de agrupamiento con todos los atributos horizontales y otros atributos no horizontales, los datos asociados a los atributos no horizontales estarán como títulos de fila.
- Los atributos que no estén en la operación de agrupamiento, se colocaran sus datos asociados en el área de detalle.

En conclusión para ser un reporte tipo matriz deberá tener atributos horizontales y agrupamientos donde se incluyan los atributos horizontales, sino no se realizará un reporte tipo matriz.

6.2.2 Ubicación del Diseño de Reporte en RS

En esta versión de VRS que usa como servidor de reporte MS RS se guarda el diseño del reporte en el servidor de Reporting Services en la carpeta VRS01 y dentro de la carpeta que tiene igual nombre que el esquema.

Por seguridad no se guarda los datos relacionados con un reporte, debido a esto si en alguna etapa posterior se desea ver el reporte directamente desde el servidor nos indica que no se puede ejecutar el comando c:\...\DatosReporte.xml.