UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

UNIVERSIDAD POLITÉCNICA DE VALENCIA
ESCUELA TÉCNICA SUPERIOR DE INFORMÁTICA APLICADA

Un Marco de Trabajo para el Desarrollo de Indicadores y Cuadros de Mando en el Contexto de un ERP

Autor: Sergio Galietero Olivares

Director: Patricio Letelier Torres

2 de Junio de 2010

Agradecimientos

Durante el desarrollo de la presente memoria he tenido la ayuda de diversas personas a las cuales me gustaría agradecer su apoyo, ya que quizás sin ellos la elaboración de la misma me habría sido imposible.

En primer lugar, me gustaría agradecer el esfuerzo de mi director de proyecto Patricio Letelier para que este proyecto llegase a buen puerto. Pese a mi poca convicción en algunos momentos, me ha inculcado que la superación debe estar siempre presente en toda persona, y que dicha superación ha de forjarse desde el trabajo. Pienso que con este proyecto me he superado.

En segundo lugar, a mi familia, la cual seguro se siente orgullosa al ver que he conseguido acabar este trabajo. Ante todo a mis padres, que me han brindado la posibilidad de llegar hasta aquí y que son mi ejemplo a seguir en el día a día. Gracias por vuestro cariño y apoyo.

En tercer lugar, agradecer a los socios de ADD Informática por haberme permitido desarrollar este proyecto basado en las experiencias vividas en dicha empresa. Gracias, porque habéis hecho que crezca como profesional.

Y por último, no me gustaría acabar sin agradecer a todas aquellas personas que han tenido que ver algo con este proyecto: aquellos que me han aconsejado, que me han ayudado, que me han hecho pasar un buen rato mientras trabajaba en el, o que simplemente se han preocupado del estado del mismo. Gracias a todos.

Índice

Capítulo 1. Introducción

1.1.- Motivación

A día de hoy, el saber el funcionamiento tanto interno como externo de una empresa es de vital importancia, ya que el no saber las debilidades y las fortalezas de la misma, y las amenazas y oportunidades que ofrece el entorno pueden hacer que en un corto intervalo de tiempo dicha empresa pierda toda la relevancia que con trabajo y esfuerzo le ha costado adquirir.

La importancia de estos hechos no ha sido pasada por alto por las tecnologías de la información, desarrollando una rama denominada Business Intelligence. Dicha rama ofrece la posibilidad de inferir conocimiento sobre la empresa y el entorno con datos de los mismos, siendo una gran fuente de ayuda para la toma de decisiones.

Este conocimiento será comunicado a determinadas personas dentro de la empresa (directores, gerentes…), y a partir del mismo, comienza un proceso de toma de decisiones en el que dichas personas, apoyándose en el conocimiento extraído, llegarán a realizar la última fase de dicho proceso: la toma de acciones.

Para ello, en algunas ocasiones es obligado el transformar los datos que haya dentro de la empresa en otros que se adapten al conocimiento que se desea ofrecer, y una vez estructurados dichos datos de forma correcta, presentarlos de modo que la inferencia de conocimiento al usuario de esta información sea casi directa.

1.2.- Objetivos

Los objetivos de la presente memoria son:

1.- Estudiar y proponer un marco tecnológico sobre el cual se pueden desarrollar soluciones de Business Intelligence adaptadas a la empresa en la cual se desarrollará el proyecto final de carrera.

2.- Estudiar las distintas alternativas presentes en el marco tecnológico del proyecto a la hora de desarrollar un indicador de gestión, es decir, las herramientas y metodologías que se pueden llevar a cabo para realizar el paso de conversión de datos en información explotable. Las alternativas se explorarán e ilustrarán con el desarrollo de dos indicadores.

3.- Evaluar las alternativas exploradas analizando sus pros y sus contras.

1.3.- Organización de la Memoria

Para llevar a cabo este objetivo la memoria se ha estructurado en capítulos, siendo el primero de ellos este de introducción.

En el segundo se ofrecerá al lector una introducción a Business Intelligence, dando a ver su importancia en el actual entorno empresarial y analizándose todos los componentes que forman este concepto.

Durante el tercer capítulo se enunciará el marco tecnológico en el cual se ha desarrollado la solución de Business Intelligence: cuál ha sido la aplicación base para la obtención de datos y que herramientas han servido para llevar a cabo dicha solución.

En el cuarto capítulo se van a detallar los dos métodos de desarrollo posibles en el marco tecnológico dado a la hora extraer los datos para convertirlos en información. A este proceso se le denominará a partir de ahora y durante todo el presente documento implementar un indicador (cuya definición está presente al inicio del capítulo), y para ilustrar dicha implementación, se hará la misma de dos indicadores por los dos métodos existentes.

Llegados al quinto capítulo se podrá observar las distintas formas de explotar la información extraída durante el punto anterior, ofreciendo unas guías para el desarrollo de estas interfaces de explotación.

En el sexto capítulo se enunciarán las conclusiones a las que se ha llegado durante los capítulos cuatro y cinco de la memoria, ofreciendo pros y contras de cada uno de los métodos de desarrollo y de las formas de explotar la información.

Finalmente se adjuntará la bibliografía utilizada para la elaboración del presente, y un anexo adicional: en él se explica una herramienta de extrema utilidad para uno de los dos métodos de desarrollo.

Capítulo 2. Business Intelligence

En el presente capitulo se realizará una descripción del concepto de Business Intelligence y todos los componentes que se ven englobados dentro de él, así como cuáles son sus principales características y cuáles han sido las claves del éxito del mismo.

Este apartado de la memoria ha sido desarrollado a partir de las referencias [BUS1] [BUS2] [BUS3] [BUS4] [BUS5] de la bibliografía.

2.1.- Definición

Business Intelligence puede ser descrito como un concepto que integra por una parte el almacenamiento de grandes cantidades de datos, y por otra el procesamiento de los mismos, siendo el principal objetivo la transformación de estos en conocimiento y en decisiones en tiempo real a través del análisis y la exploración.

El concepto de Business Intelligence parte de la premisa de que, si se cuenta con un conjunto de datos lo suficientemente grande y se aplica un análisis a los mismos, se puede inferir conocimiento. Una frase muy popular en el mundo de Business Intelligence enuncia: “Business Intelligence es el proceso de convertir datos en conocimiento, y el conocimiento en acción, para la toma de decisiones”. Dicha afirmación se ve ilustrada perfectamente en la Figura 1.

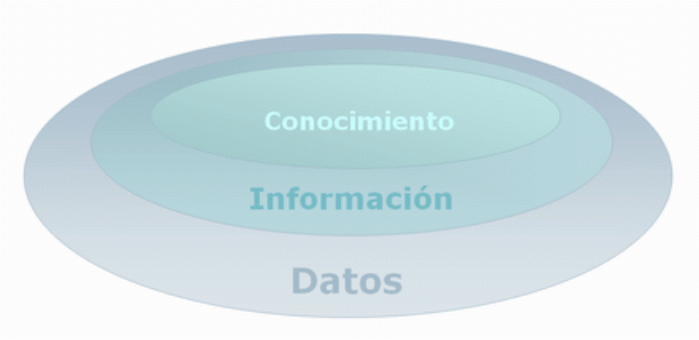


Figura 1.- Conversión de datos en conocimiento

Business Intelligence se centra principalmente en los procesos de recolección y utilización efectiva de la información, al objeto de mejorar la forma de operar de una organización, ofreciendo a sus usuarios el acceso a la información que es clave a la hora de llevar a cabo sus tareas habituales, con el fin de poder tomar decisiones oportunas basadas en datos certeros y correctos.

Al contar con información correcta y exacta en tiempo real, es posible identificar y corregir situaciones que, en un futuro, se pueden convertir en problemas para la organización, ofreciendo también la posibilidad de identificar nuevas oportunidades o readaptase ante la ocurrencia de sucesos inesperados.

Cuanto más relevante y útil sea la inteligencia que posea una organización sobre un negocio (clientes, proveedores, socios, operaciones), mayor será su ventaja competitiva sobre sus competidores, pudiéndose tomar unas mejores decisiones.

2.2.- Breve Historia de Business Intelligence (BI)

A continuación, se van a enumerar los que se podrían denominar hitos más importantes en la historia de Business Intelligence, abarcando hechos anteriores a la creación de este concepto, pero que son antecedentes que han ayudado a la definición del mismo:

- 1969: Creación del concepto de base de datos (Codd)
- 1970-1979: Desarrollo de las primeras bases de datos y las primeras aplicaciones empresariales (SAP, JD Edwards, Siebel, PeopleSoft). Estas ofrecieron la posibilidad de realizar "data entry" en los sistemas, aumentando la información disponible, pero no fueron capaces de ofrecer un acceso rápido y fácil a la misma.
- 1980-1988: Creación del concepto de Almacén de Datos (Ralph Kimball, Bill Inmon), y aparición de los primeros sistemas de reporting. A pesar de esto, la funcionalidad ofrecida era escasa, además de ser complicado para el usuario. En esta época ya existían sistemas de bases de datos relativamente potentes, pero no había aplicaciones para facilitar su explotación.
- 1989: Introducción del término Business Intelligence (Howard Dresner)
- 1990-1999: Business Intelligence 1.0. Proliferación de múltiples aplicaciones BI. Estos proveedores resultaban caros, pero facilitaron el acceso a la información. Hay que decir en su contra, que en algunos casos, agravaron el problema que pretendían resolver, ya que los sistemas de información carecían de integridad.
- 2000-2009: Business Intelligence 2.0. Se consolidan las aplicaciones de Business Intelligence en algunas plataformas (Oracle, SAP, IBM, Microsoft, etc). A parte de la información estructurada, se empieza a considerar otro tipo de información y documentos no estructurados.

2.3.- Principales Características que Ofrecen los Sistemas Business Intelligence

- Accesibilidad a la información. Los datos son la fuente principal de este concepto. Lo primero que deben garantizar las herramientas y técnicas de Business Intelligence es el acceso de los usuarios a los datos, independientemente de la procedencia de estos.
- Apoyo a la toma de decisiones. Lo que se busca con las herramientas de Business Intelligence es ir más allá en la presentación de la información, de modo que los usuarios de

dicha información tengan acceso a herramientas de análisis que les permitan seleccionar y modificar solo los datos que les interesen.

- Orientación al usuario final. Se busca que, independientemente de los conocimientos técnicos de los usuarios, las herramientas sean lo suficientemente sencillas como para ser utilizadas por estos.

2.4.- Factores Críticos del Éxito de Business Intelligence

Los factores críticos que marcan el éxito de Business Intelligence son los siguientes:

- Proveen acceso a datos adecuados. Esto se logra ya que los datos que se tienen están organizados.
- Incrementan la habilidad de los usuarios para entender los resultados. El hecho de saturar a las personas de datos que no logran entender crea más problemas de los que resuelve. Puede que diez años atrás el problema fuese la obtención de datos, pero a día de hoy el problema es el manejo de los mismos de una manera eficaz y eficiente.
- Incrementan el entendimiento de los negocios por parte de los usuarios. Es esencial saber lo que nos están diciendo los datos, para tomar decisiones consecuentemente. Esto es quizás lo más difícil de implementar en una herramienta de Business Intelligence
- Ayudan a comunicar los hallazgos y a tomar decisiones. Estas herramientas facilitan la comunicación dentro de la empresa y ayudan en el proceso de toma de decisiones.

2.5.- Principales Ventajas del Uso de Business Intelligence

El uso de herramientas Business Intelligence por parte de las organizaciones proporciona una serie de ventajas que se enumeran a continuación:

- Alta capacidad de respuesta a las necesidades de sus clientes.
- Reconocimiento de las necesidades de los clientes.
- Habilidad para actuar ante cambios de mercado.
- Optimización de las operaciones.
- Análisis de calidad como base de proyecciones futuras.
- La mejor utilización posible de los recursos.

2.6.- Componentes del Business Intelligence

Durante los siguientes puntos se van a estudiar los distintos componentes de una solución Business Intelligence. Para comenzar, se observará cual es el típico esquema de una solución de este tipo, pasando a continuación a ofrecer las definiciones y características de los distintos componentes que forman este esquema.

2.6.1.- Esquema de una solución Business Intelligence

Al objeto de seguir el desarrollo de este punto, se ha adjuntado la Figura 2, que esquematiza una solución de Business Intelligence, especificando todos y cada uno de sus componentes que serán explicados a continuación.

Una solución de Business Intelligence parte de los sistemas de origen de una organización (bases de datos, ERPs, ficheros de texto, etc.), sobre los que suele ser necesario aplicar una transformación estructural para optimizar su proceso analítico.

Para ello se realiza una fase de extracción, transformación y carga (ETL) de datos. Esta etapa suele apoyarse en un almacén intermedio que actúa como pasarela entre los sistemas fuente y los sistemas destino (generalmente un almacén de datos), y cuyo principal objetivo consiste en evitar la saturación de los servidores funcionales de la organización.

La información resultante, ya unificada, depurada y consolidada, se almacena en un almacén de datos corporativo, que puede servir como base para la construcción de distintos datamarts departamentales. Estos datamarts se caracterizan por poseer la estructura óptima para el análisis de los datos de esa área de la empresa, ya sea mediante bases de datos transaccionales (OnLine Transaction Processing, OLTP) o mediante bases de datos analíticas (OnLine Analytical Processing, OLAP).

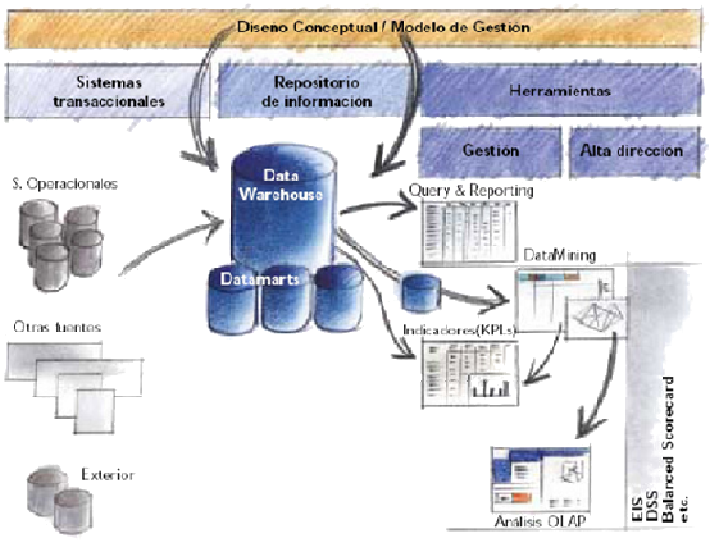


Figura 2.- Esquema de los distintos componentes de un sistema de Business Intelligence

Los datos albergados en el almacén de datos o en cada datamart se explotan utilizando herramientas comerciales de análisis, reporting, etc. En estas herramientas se basa también la construcción de productos BI más completos, como los sistemas de soporte a la decisión (Decission Support System, DSS), los sistemas de información ejecutiva (Executive Information System, EIS) y los cuadros de mando integrales (CMI) o Balanced Scorecard (BSC).

2.6.2.- El Proceso ETL

El término ETL procede de las siglas Extract – Transform – Load que significan Extraer, Transformar y Cargar, refiriéndose siempre a los datos de una empresa. Este término se puede definir como el proceso que organiza el flujo de los datos entre diferentes sistemas de una organización, aportando los métodos y herramientas necesarios para mover datos desde múltiples orígenes a un almacén de datos, reformateándolos, limpiándolos y cargándolos.

Extraer (Extract)

La primera tarea del proceso ETL consiste en extraer los datos almacenados en los distintos sistemas de origen. En un número elevado de casos, en los proyectos de almacenamiento de datos se fusionan datos que provienen de distintos sistemas de origen. Cada sistema puede usar una organización distinta de los datos, o formatos diferentes. Dichas fuentes pueden ser bases de datos, ficheros planos, etc. Sea cual sea su estructura, la tarea de extracción convierte todos estos datos a un formato preparado para comenzar con el proceso de transformación.

El requerimiento imprescindible a la hora de realizar la tarea de extracción es que la misma cause un impacto mínimo en los sistemas de origen. Si la cantidad de datos a extraer es muy elevada, el sistema se puede ralentizar, o incluso colapsarse, por lo que las grandes operaciones de extracción se suelen realizar en momentos donde el impacto sobre el sistema sea el mínimo posible.

Transformar (Transform)

En la tarea de transformación se aplican una seria de funciones sobre los datos extraídos al objeto de convertirlos en datos preparados para su carga. Algunas fuentes de datos tan solo requerirán mínimas transformaciones, mientras que otras necesitaran de un gran número de ellas. Entre las operaciones de transformación podemos encontrar las siguientes:

- Traducción y codificación de códigos.
- Obtención de valores calculados.
- Generación de nuevos campos.
- División de la información.
- Unión de datos de múltiples fuentes.

Cargar (Load)

Es la tarea mediante la cual los datos ya transformados en la fase anterior son cargados en el sistema de destino. Durante esta fase se interactúa directamente con las bases de datos de destino, por lo

que se aplicarán todas las restricciones y disparadores que se hayan definido en la misma, contribuyendo este hecho a que se garantice la calidad de los datos durante el proceso integro.

2.6.3.- Almacenes de Datos

Introducción a los Almacenes de Datos

Desde el comienzo de la utilización de los computadores, las organizaciones han venido usando los datos de sus sistemas operacionales para atender las necesidades de información. Para ello, se puede acceder directamente a la información contenida en las aplicaciones operacionales, o extraer los datos desde las bases de datos operacionales combinando la información de distintas formas no estructuradas, con el objetivo de complacer dicha demanda de información de la mejor manera posible.

Estos dos métodos enunciados anteriormente han ido evolucionando a través del tiempo, y a día de hoy, las organizaciones manejan normalmente una información inconsistente, sobre la cual se deben tomar una serie de decisiones importantes en el devenir de las mismas.

La gestión administrativa reconoce que la mejor forma de elevar la eficiencia de una organización consiste en hacer el mejor uso de la información ya existente dentro de la misma. A pesar de que esto se ha venido intentando desde hace muchos años, todavía no se ha conseguido el uso totalmente efectivo de esta información.

La razón principal por la que este logro no ha sido completamente efectivo ha sido la evolución de las computadoras, basada en la tecnología de la información y de sistemas. La mayor parte de las organizaciones hacen lo posible para conseguir una información lo más limpia, consistente y estructurada posible, estando todo esto supeditado a la arquitectura del software desarrollado.

Por ello, y a día de hoy, los almacenes de datos y todas las tecnologías relacionadas con los mismos son el centro de atención de las grandes organizaciones, ya que proveen un ambiente perfecto para que las organizaciones hagan un mejor uso de la información que está gestionada por una o varias aplicaciones operacionales.

Definición de Almacén de Datos

Un almacén de datos es una colección de datos en el cual se encuentra integrada información de una organización, y que se usa como soporte a la hora de llevar a cabo el proceso de toma de decisiones gerenciales. Esto último quiere decir que, a pesar de tener implementado un almacén de datos, se necesitará de otras herramientas para la explotación de la información, ya que el almacén de datos tan solo actúa como “contenedor” de información.

Principales Características de un Almacén de Datos

Las principales características de un almacén de datos son:

- Orientación a temas: la información se clasifica en base a los aspectos que son de interés para la organización. En el entorno de las aplicaciones operacionales, los datos están

orientados a las funciones que se desarrollan dentro de la organización (préstamos, fabricación, etc. mientras que en el entorno de los almacenes de datos, estos se organizan dependiendo de los sujetos que van a recibir la misma (director general, jefe de departamento, etc.).

- Integrado: integra la información disponible de varios sistemas operacionales. Así pues, habrá veces que haya que llevar a cabo un proceso de transformación ya que el mismo dato se representa de distinta manera en los distintos sistemas operacionales para que el resultado sea totalmente consistente.
- No volátil: es de suma importancia saber que la información de un almacén de datos no se actualiza ni se elimina, tan solo se va incrementando.
- De tiempo variante: una de las diferencias significativas entre una aplicación operacional y un almacén de datos es que en este último cualquier información de cualquier instante de tiempo puede ser requerida en un determinado momento. Es decir, el almacén de datos va a ser una especie de "deposito histórico" de información.

Si se contraponen las características de un almacén de datos y una base de datos operacional, se puede deducir que se está ante conceptos totalmente distintos a pesar de actuar los dos como "contenedores" de información. En la siguiente tabla se pueden apreciar las diferencias entre las dos:

Base de Datos Operacional	Almacén de Datos
Datos Operacionales	Datos Aptos para la Toma de Decisiones
Orientado a la Aplicación	Orientado al Sujeto
Información Actual	Información Actual + Histórica
Detallada	Detallada + Resumida
Optimizada para Inserción, Actualización, Eliminación y Consulta	Optimizado para Consulta
Muchas Transacciones y Ligeras	Pocas Transacciones y Pesadas

Estructura de un Almacén de Datos

Un almacén de datos siempre sigue el modelo multidimensional de datos, siendo este muy distinto al modelo entidad – relación que poseen las bases de datos.

El modelo de datos multidimensional se podría definir como un conjunto de medidas descritas dentro de unas dimensiones. A cada una de estas medidas asociada a unas determinadas dimensiones se le denominará hecho.

Este modelo es sencillo de comprender si se utiliza una estructura de cubo rubik expuesta en la Figura 3:

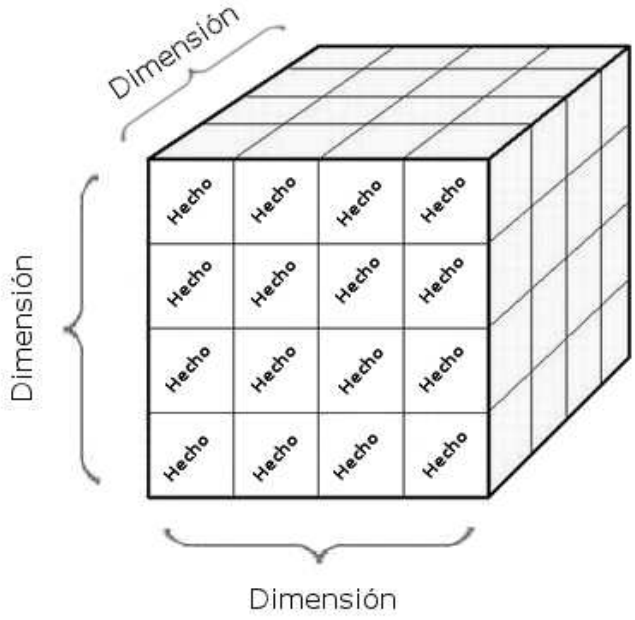


Figura 3.- Ejemplo General de Cubo Rubik

En el cubo se puede observar como los ejes del mismo son las dimensiones, las cuales están compuestas de elementos denominados miembros. Cada casilla de dentro del cubo rubik sería un hecho, el cual podría tener una medida asociada además de la información implícita que nos aporta el estar situada en una casilla (información sobre las dimensiones).

Para estructurar toda esta información en una base de datos relacional (ya que vamos a usar un sistema de explotación ROLAP), se utilizarán en esencia dos tipos de tablas:

- Tablas de Dimensiones: tablas en las que están depositados todos y cada uno de los valores que puede tomar un hecho para una dimensión. A cada uno de estos valores se les denominará miembro de la dimensión.
- Tabla de Hechos: tabla en la cual se guardaran los hechos, es decir, la combinación de miembros de una dimensión y la medida asociada a dicha combinación.

En el cubo rubik de ejemplo anterior, una dimensión sería el tipo de producto, y los miembros de la dimensión serían los distintos productos que fabrica o vende la organización. Por otro lado, un hecho sería el importe de las ventas (medida) que ha habido de un determinado producto en una región y fecha.

Dentro de las tablas de dimensión se tendría toda la información que podría ser de interés para los usuarios: en el ejemplo podría ser el tipo de producto, su forma de envase… colocando esta información dentro de la tabla de dimensión relacionada (en los ejemplos sería la tabla de dimensión producto).

A continuación, y mediante la Figura 4 se puede ver ejemplificado el caso real enunciado anteriormente:

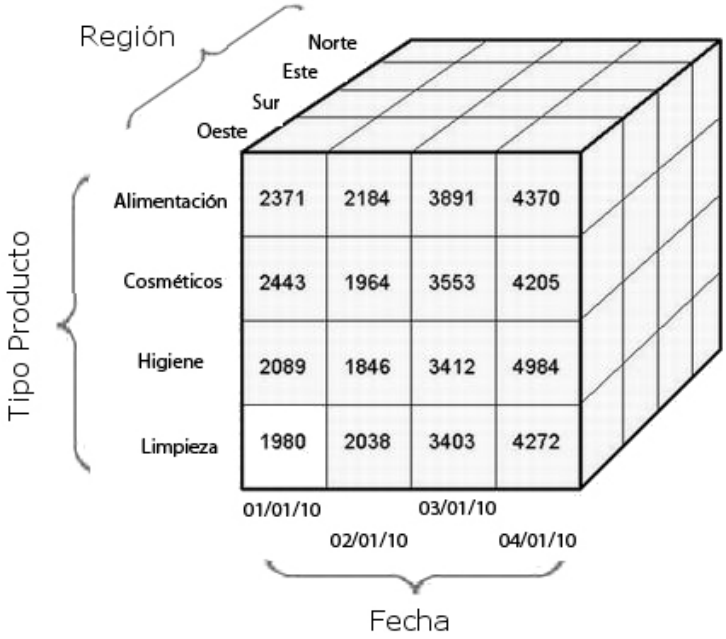


Figura 4: Ejemplo cubo Rubik adaptado a un caso real

Por otro lado, en las tablas de hechos tan solo se guardarían referencias a distintos elementos de las dimensiones, y en el caso que hicieran falta, medidas que aporten más información acerca del hecho ocurrido.

A este modo de estructurar la información se le denomina diagrama en estrella (las tablas de dimensión tan solo pueden estar relacionadas con las tablas de hechos), y se puede ver representado en la Figura 5.

Pero este no es el único modo de estructurar la información, ya que el modelo multidimensional también permite jerarquizar las dimensiones. Con esto lo que se logra es que determinados miembros de una dimensión puedan ser agrupados dentro de un miembro de otra dimensión superior.

Un buen punto de partida para la explicación de las jerarquías de dimensiones seria la dimensión producto de nuestro cubo rubik. En el ejemplo, se puede ver el importe de ventas de un producto un día en una región. Pero, ¿y si se quisiera ver la información agrupada por tipo de producto en vez de por cada producto? ¿Sería necesaria la elaboración de una nueva tabla de hechos?

La respuesta es no. Para ello, se crea una nueva tabla de dimensiones denominada tipo de producto, y se relaciona cada producto con un tipo de producto. De esta forma, se evita el problema de elaborar una nueva tabla de hechos con una información repetida, y con la posibilidad de que la herramienta de explotación de la información nos permita navegar y cambiar la información al antojo del usuario.

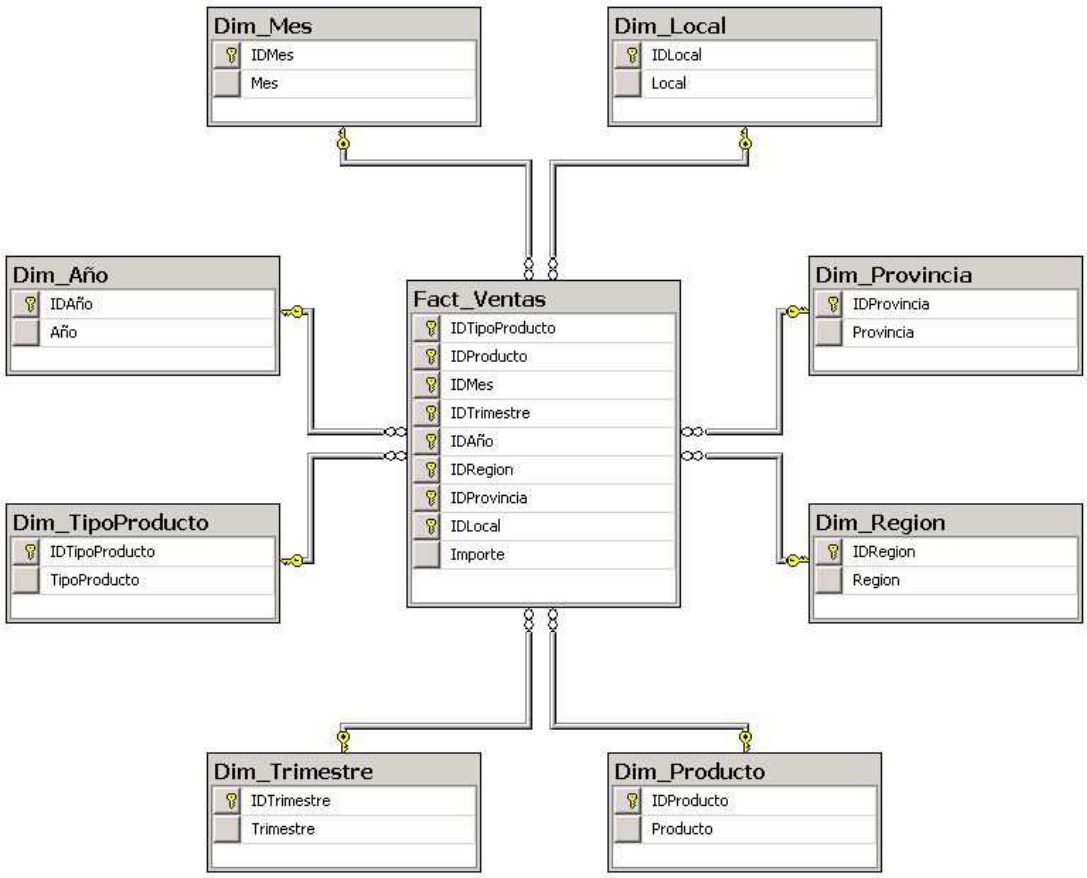


Figura 5.- Ejemplo de diagrama en estrella

De este modo, se pueden crear jerarquías cuya explotación puede ser interesante para el estudio de la información. Un ejemplo de estas seria la jerarquía geográfica, en la que por ejemplo, podemos tener varias tablas de dimensiones (local, población, provincia, comunidad, país, etc.) que se podrían relacionar entre sí (un local está dentro de una población, que a su vez pertenecerá a una provincia, dicha provincia estará en una comunidad, y la comunidad en un país) pudiendo inferir de este modo estadísticas muy variadas.

A este modo de estructurar la información se le denomina diagrama copo de nieve (las dimensiones pueden estar relacionadas con tablas de hechos, y también con otras dimensiones), y puede verse representado en la Figura 6.

A cuan específica es la información que se está mostrando en un momento en la tabla de hechos se le denomina granularidad. En el ejemplo comentado anteriormente, el granulo más fino correspondería al local (ya que no hay una forma de desagregar en un granulo más fino el local en términos geográficos), mientras que el granulo más grueso correspondería a país (porque la información ya no se puede agregar más con las dimensiones que se cuentan).

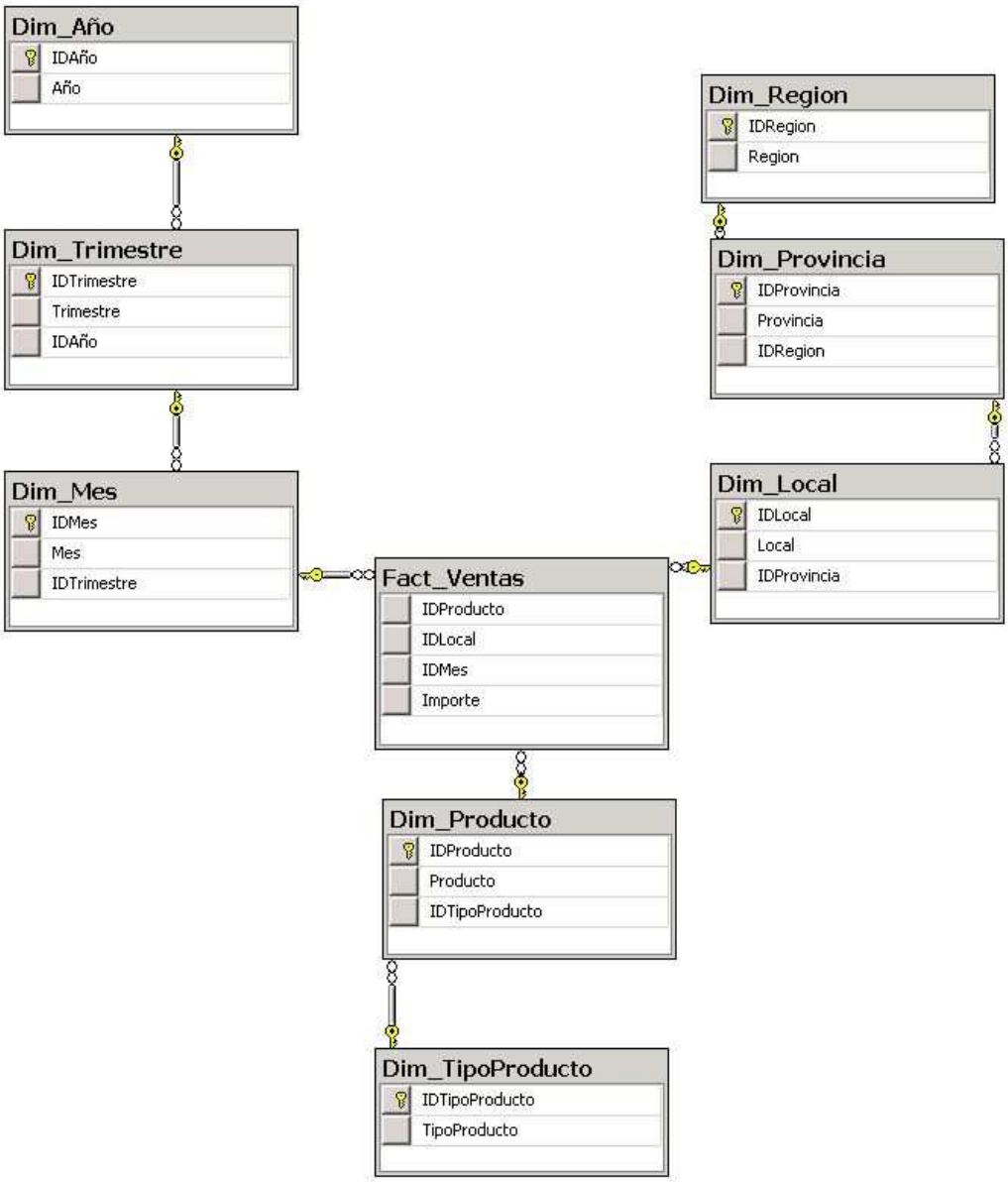


Figura 6.- Ejemplo de diagrama de copo de nieve

Mención aparte merece la dimensión tiempo, con la cual se podría estructurar una jerarquía interesante (como podría ser día, mes, año) al objeto de lograr unas estadísticas más trabajadas. Sabiendo esto, la mayoría de programas que permiten hacer datawarehousing, ofrecen herramientas que facilitan al usuario la construcción de la dimensión tiempo sin tener que construir más de una tabla de dimensiones.

Con el objetivo de no repetir tablas de dimensiones, y a la hora de tener una mejor visualización de nuestro almacén de datos, este último se compone datamarts. Los datamarts no son más que subconjuntos del almacén de datos, y son definidos al objeto de satisfacer las necesidades de los distintos miembros o conjuntos de miembros de una organización. Así, por ejemplo, algunos datamarts de un almacén de datos más elaborado serían los mostrados en la Figura 7.

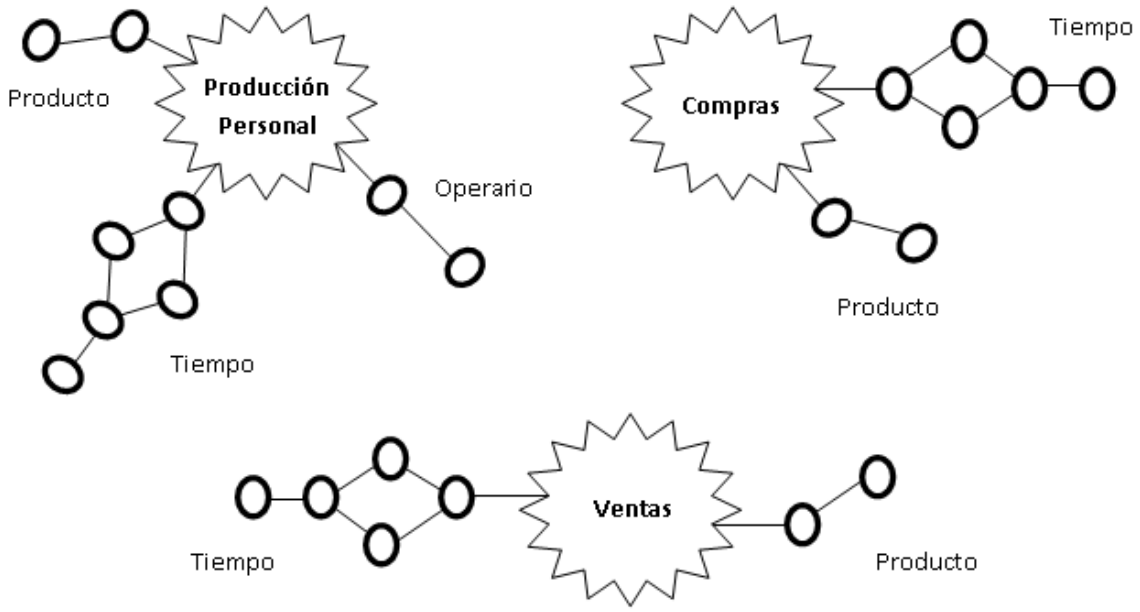


Figura 7.- Ejemplos de Datamarts

2.6.4.- Query & Reporting

Los sistemas de explotación de Query & Reporting permiten la elaboración de informes y listados, tanto de detalle como de información agregada, a partir de la información disponible en los almacenes de datos y en los datamarts.

Este sistema de explotación debe permitir una gran flexibilidad de adaptación dependiendo del usuario, su experiencia y su formación. Por ello, se recomienda el mantenimiento de los sistemas en al menos tres niveles de dificultad:

- Los usuarios poco expertos podrán solicitar la ejecución de informes o consultas predefinidas según unos parámetros predeterminados.
- Los usuarios con experiencia media podrán generar consultas flexibles mediante una aplicación de interfaz gráfica.
- Los usuarios altamente experimentados podrán escribir la consulta en el lenguaje de obtención de datos.

2.6.5.- Análisis OLAP

La solución comúnmente adoptada a la hora de explotar almacenes de datos se denomina OLAP (On-Line Analytical Processing), la cual tiene como objetivo agilizar la consulta a grandes cantidades de datos.

En contraposición a OLTP (On-Line Transaction Processing), OLAP ofrece unos mejores tiempos de respuesta a la hora de realizar consultas a una base de datos. Esto es debido a que la primera solución está enfocada a las transacciones (insertar, eliminar y actualizar), mientras que la segunda está enfocada a la consulta de datos.

Una consulta a un almacén de datos se puede resumir como la obtención de medidas sobre uno o varios hechos, estando estos parametrizados por atributos de las dimensiones y restringidas por condiciones de las mismas.

Así por ejemplo, se podría tener una consulta que enunciara lo siguiente:

- Importe total de las ventas durante el año 2010 de los productos del tipo limpieza, por trimestre y por producto.

Si se lee detenidamente la consulta se pueden enlazar los distintos conceptos que aparecen en la definición de consulta con la enunciada en el párrafo anterior.

- Hecho: Ventas
- Medida: Importe
- Restricciones: Tipo de producto Limpieza, Año 2010
- Parámetros de la consulta: Trimestre y Producto

Una vez realizada la consulta se obtendría el siguiente resultado:

		Trimestre			
		Trimestre I	Trimestre II	Trimestre III	Trimestre IV
Producto	Amoniaco	10 000	12 000	11 000	12 000
	Friegasuelos	15 000	16 000	18 000	20 000
	Limpiacristales	8 000	7 000	9 000	8 000

Además de la posibilidad de realizar consultas tan flexibles (que con menor o mayor complejidad se pueden realizar mediante selecciones, concatenaciones y agrupamientos tradicionales), la verdadera potencia de OLAP se halla en sus operadores a la hora de refinar o manipular consultas. Estos operadores son los siguientes:

- Agregación (Roll Back): permite eliminar un criterio de agrupación en el análisis, agregando los grupos actuales. Si se hiciese la agregación de producto, el resultado sería el mostrado a continuación:

		Trimestre			
		Trimestre I	Trimestre II	Trimestre III	Trimestre IV
Tipo Producto	Limpieza	33 000	35 000	38 000	40 000

- Disgregación (Drill Down): permite introducir un nuevo criterio de agrupación en el análisis, disgregando los grupos actuales. Si se hiciese disgregación sobre el trimestre, obtendríamos el siguiente informe:

		Mes											
		E	F	M	A	M	J	J	A	S	O	N	D
Producto	Amoniaco	2k	4k	4k	5k	3k	4k	4k	4k	3k	4k	5k	3k
	Friegasuelos	4k	5k	6k	5k	5k	6k	6k	7k	5k	4k	8k	8k
	Limpiacristales	2k	3k	3k	3k	2k	2k	2k	3k	4k	3k	2k	2k

- Proyectar (Slice & Dice): selecciona y proyecta datos en el informe. Un ejemplo de proyectar en el informe de partida sería el siguiente:

		Trimestre	
		Trimestre I	Trimestre III
Producto	Limpiacristales	8 000	9 000

- Pivotar (Pivot): reorienta las dimensiones del informe.

		Producto		
		Amoniaco	Friegasuelos	Limpiacristales
Trimestre	Trimestre I	10 000	15 000	8 000
	Trimestre II	12 000	16 000	7 000
	Trimestre III	11 000	18 000	9 000
	Trimestre IV	12 000	20 000	8 000

Con estas operaciones, se puede deducir que el número de consultas que se pueden realizar es muy elevada, pudiendo conseguir una consulta específica para cada situación de la organización que se quiera representar.

Tipos de Sistemas OLAP: ROLAP y MOLAP

A partir de la solución OLAP, se desarrollaron varios tipos de sistemas de explotación de almacenes de datos, de los cuales destacaremos los dos más conocidos: ROLAP y MOLAP.

El sistema ROLAP utiliza una arquitectura de tres niveles. La base de datos relacional maneja los requerimientos de almacenamiento de datos, y el motor ROLAP proporciona la funcionalidad analítica. El nivel de base de datos usa bases de datos relacionales para el manejo, acceso y obtención del dato. El nivel de aplicación es el motor que ejecuta las consultas multidimensionales de los usuarios.

El motor ROLAP se integra con niveles de presentación, a través de los cuáles los usuarios realizan los análisis OLAP. Después de que el modelo de datos para el almacén de datos se ha definido, los datos se cargan desde el sistema operacional.

Los usuarios finales ejecutan sus análisis multidimensionales, a través del motor ROLAP, que transforma dinámicamente sus consultas a consultas SQL. Se ejecutan estas consultas SQL en las bases de datos relacionales, y sus resultados se relacionan mediante tablas cruzadas y conjuntos multidimensionales para devolver los resultados a los usuarios.

Por otro lado, el sistema MOLAP utiliza una arquitectura de dos niveles: la bases de datos multidimensionales y el motor analítico. La base de datos multidimensional es la encargada del manejo, acceso y obtención del dato.

El nivel de aplicación es el responsable de la ejecución de los requerimientos OLAP. El nivel de presentación se integra con el de aplicación y proporciona un interfaz a través del cual los usuarios finales visualizan los análisis OLAP. Una arquitectura cliente/servidor permite a varios usuarios acceder a la misma base de datos multidimensional.

La información procedente de los sistemas operacionales, se carga en el sistema MOLAP, mediante una serie de rutinas por lotes. Una vez cargado el dato elemental en la base de datos multidimensional, se realizan una serie de cálculos por lotes, para calcular los datos agregados, a través de las dimensiones de negocio, rellenando la estructura de la base de datos multidimensional.

Tras rellenar esta estructura, se generan unos índices y algoritmos de tablas hash para mejorar los tiempos de accesos a las consultas. Una vez que el proceso de compilación se ha acabado, la base de datos multidimensional está lista para su uso. Los usuarios solicitan informes a través de la interfaz y la lógica de aplicación de esta última obtiene el dato.

El gran punto diferenciador entre los dos sistemas de explotación de datos es el volumen de los mismos a los cuales el sistema puede hacer frente: mientras que los sistemas MOLAP son adecuados para almacenes que tengan un tamaño de entre 3 y 5 GB, los sistemas ROLAP se comportan de una manera más eficaz ante almacenes de datos con un volúmenes superiores a los anteriormente enunciados.

2.6.6.-Minería de Datos

Definición

La minería de datos consiste en la extracción no trivial de información que se halla de manera implícita en los datos. Esta información es desconocida a priori, y puede resultar de utilidad para algún proceso. En otras palabras, la minería de datos prepara, sondea y explora los datos para sacar información oculta en ellos.

En el concepto de minería de datos se engloba un conjunto de técnicas dirigidas a extraer conocimiento procesable, implícito en bases de datos. Las bases de estas técnicas se encuentran en la inteligencia artificial y en el análisis estadístico. Mediante los modelos extraídos utilizando técnicas de minería de datos se aborda la solución a problemas de predicción, clasificación y segmentación.

La minería de datos no es más que una de las fases esenciales de un proceso mucho más amplio cuyo objetivo es el descubrimiento de conocimiento en bases de datos (KDD, Knowledge Discovery from Databases).

El proceso de extracción de conocimiento de bases de datos

El proceso de extracción de conocimiento es el proceso mediante el cual, partiendo de unos datos iniciales, se infiere conocimiento al objeto de apoyar a la toma de decisiones. Este es un proceso iterativo e interactivo: iterativo ya que la salida de alguna de las fases que lo componen puede hacer volver a pasos anteriores, siendo estas necesarias para extraer un conocimiento de alta calidad; interactivo ya que el usuario debe interactuar con los procesos, entradas y salidas que conforman el proceso (ayuda a la preparación de datos, validación de conocimiento extraído…).

En las siguientes líneas se explica cuales son las fases del proceso de extracción de conocimiento, ofreciendo una pequeña descripción de cada una:

1. Fase de integración y recopilación: en esta fase se determinan las fuentes de información que pueden ser útiles y donde conseguirlas. A continuación, se transforman todos los datos a un formato común, frecuentemente mediante un almacén de datos que consiga unificar de manera operativa toda la información recogida, detectando y resolviendo las inconsistencias. Este almacén de datos facilita enormemente la "navegación" y visualización previa de sus datos, para discernir qué aspectos interesan ser estudiados.
2. Fase de selección, limpieza y transformación: una vez recopilados los datos, se deben tomar una serie de decisiones como que hacer con los datos incorrectos y con los datos incompletos. Además, se proyectan los datos para considerar únicamente aquellas variables o atributos que van a ser relevantes, con el objetivo de hacer más fácil la tarea propia de la minería, así como para conseguir unos resultados mas útiles. Estas dos primeras fases podrían estar englobadas dentro del nombre "preparación de datos".
3. Minería de datos: es la fase más característica del proceso de extracción de conocimiento, y por ello, muchas veces se utiliza esta fase para nombrar todo el proceso. El objetivo de esta fase es producir nuevo conocimiento que pueda utilizar el usuario. Esto se realiza construyendo un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender mejor los datos o para explicar situaciones pasadas. Para ello, es necesario tomar una serie de decisiones antes de empezar el proceso:
    - Determinar qué tipo de tarea de minería es el más apropiado
    - Elegir el tipo de modelo.
    - Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando.
4. Evaluación e interpretación: durante esta fase se mide la calidad de los patrones descubiertos, siendo estos analizados por expertos, volviendo a las fases anteriores si es necesario para incurrir en una nueva iteración.
5. Difusión y uso: una vez construido validado el modelo, este puede usarse principalmente con dos finalidades: que un analista recomiende acciones basándose en el modelo, o bien aplicar el modelo a distintos conjuntos de datos.

En la Figura 8 se puede observar el proceso de extracción de conocimiento ordenado y con todos sus inputs y outputs.

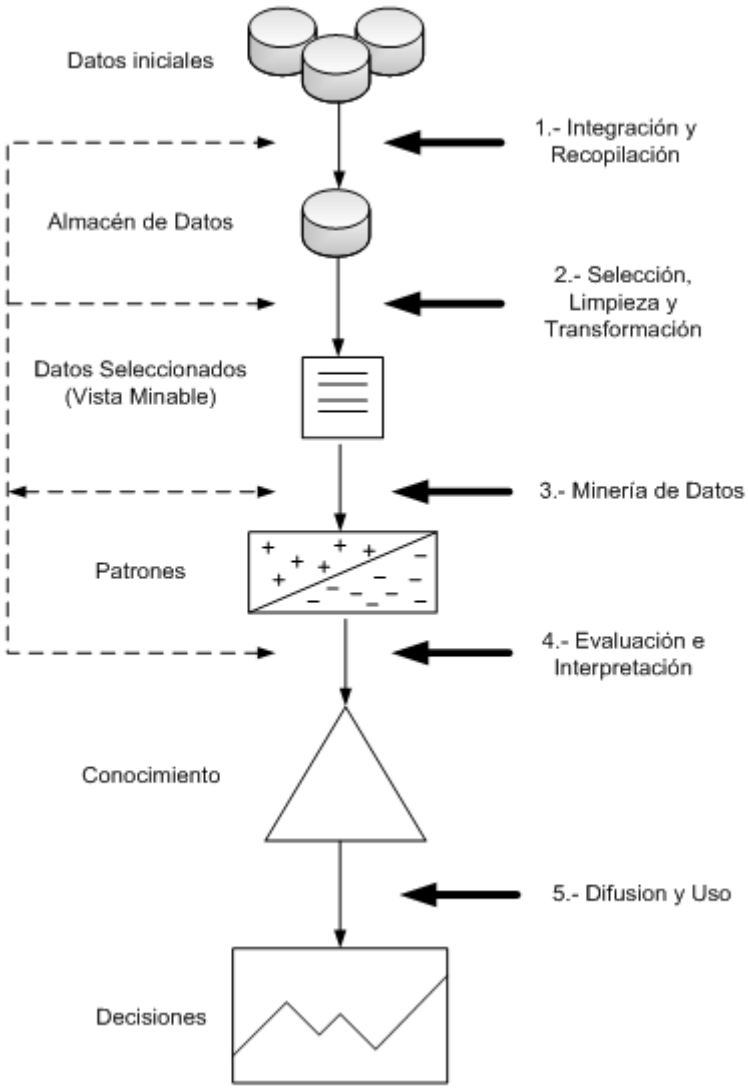


Figura 8.- Diagrama de las tareas de minería de datos

Tareas de la Minería de Datos

Dentro de la minería de datos hemos de distinguir distintos tipos de tareas, cada una de las cuales puede considerarse como un tipo de problema a ser resuelto por un algoritmo de minería de datos. Las tareas de minería de datos se pueden dividir en dos grupos:

- Tareas predictivas: tareas en las que hay que predecir uno o más valores para uno o más ejemplos. Al finalizar estas tareas, los ejemplos irán acompañados de una salida (clase, categoría o valor numérico) o un orden entre ellos.
- Tareas descriptivas: el objetivo de estas tareas no es predecir nuevos datos, sino describir los ya existentes.

A continuación, se comentan las principales tareas:

- Clasificación: tarea predictiva cuyo principal objetivo es predecir a qué grupo pertenecerá una instancia dentro de un conjunto de grupos previamente enunciado.
- Regresión: tarea predictiva que consiste en aprender una función real que asigna a cada instancia un valor real.
- Agrupamiento: tarea descriptiva por excelencia, consistente en obtener grupos "naturales" a partir de los datos.
- Correlaciones: tarea descriptiva que se usa para examinar el grado de similitud de los valores de dos variables numéricas.
- Reglas de asociación: tarea descriptiva en la que el objetivo es identificar relaciones no explícitas entre atributos categóricos.

Técnicas y Algoritmos de Minería de Datos

Según el objetivo del análisis, los algoritmos utilizados se pueden clasificar en dos categorías:

- Algoritmos supervisados (o predictivos). Son aquellos que predicen un dato o un conjunto de datos desconocidos a priori, a partir de otros conocidos.
- Algoritmos no supervisados (o de descubrimiento de conocimiento). Son aquellos que descubren patrones o tendencias dentro de los datos.

Algunas de las técnicas más conocidas de minería de datos son las siguientes:

- Redes neuronales. Son un paradigma de aprendizaje y proceso automático inspirado en el sistema nervioso de animales. Consiste en un conjunto de neuronas interconectadas dentro de una red que colabora al objeto de conseguir un estimulo de salida.
- Regresión lineal. Una de la más utilizadas, rápida y eficaz, pero insuficiente a la hora de relacionar más de dos variables.
- Árboles de decisión. Es un modelo de predicción que, dada una base de datos, construye diagramas de construcciones lógicas similares a los sistemas de predicción basados en reglas.
- Agrupamiento. Procedimiento de agrupación de una serie de vectores según criterios de distancias. Se trata de agrupar aquellos datos que tengan más características comunes, determinando a priori un número de grupos.

2.6.7.- Cuadros de Mando Integrales (CMI)

El cuadro de mando integral (CMI), también conocido como Balanced Scorecard (BSC) o dashboard, es una herramienta de control empresarial que permite establecer y monitorizar los objetivos de una empresa y sus diferentes áreas o unidades.

También puede ser considerado como una aplicación que ayuda a una organización a establecer sus objetivos estratégicos e iniciativas necesarias para cumplirlos, mostrando en cada instante cuan

cerca o lejos se está de cumplirlos. Para ello es necesaria la elaboración de un modelo de negocio que refleje las interrelaciones entre los diferentes componentes de la empresa (mediante un plan estratégico).

A diferencia de otras herramientas de explotación de información, los cuadros de mando están más enfocados al seguimiento de los indicadores (financieros o no financieros) de objetivos estratégicos de la organización que al análisis minucioso de la información. Por ello, es de vital importancia que los indicadores estén bien definidos y que estos tengan un alto grado de relación con el objetivo estratégico a cumplir.

La aportación que ha convertido al CMI en una de las herramientas más significativas de los últimos años es que se cimenta en un modelo de negocio. El éxito de su implantación radica en que el equipo de dirección se involucre y dedique tiempo al desarrollo de su propio modelo de negocio.

Metodología Kaplan & Norton

A pesar que hay un buen número de metodologías a la hora de elaborar cuadros de mando, la mayoría de éstos siguen la elaborada por Kaplan y Norton durante los años 90, materializada en el libro "The Balanced Scorecard".

Kaplan y Norton consideraron que los antiguos modelos de gestión empresarial basados únicamente en indicadores financieros estaban completamente obsoletos, ya que no se tenían en cuenta los factores no financieros y la influencia que podían tener sobre los objetivos estratégicos de la organización.

Para la definición de estos objetivos, se tuvo en cuenta que los mismos pueden organizarse en cuatros distintas áreas o perspectivas:

- Perspectiva Financiera: serán aquellos objetivos que incorpora la visión de los accionistas, midiendo la creación de valor para la organización. Un indicador ejemplo de los objetivos estratégicos incluidos en esta perspectiva sería el beneficio de la organización.
- Perspectiva de Cliente: son los objetivos que responden al posicionamiento de la organización dentro del mercado o el segmento de mercado donde esta se mueve. Un indicador ejemplo de los objetivos estratégicos incluidos en esta perspectiva sería la cuota de mercado.
- Perspectiva Interna: objetivos derivados de los procesos internos de la organización que son críticos para el posicionamiento de mercado. Un indicador ejemplo de los objetivos estratégicos incluidos en esta perspectiva sería la productividad de la planta de producción de una organización.
- Perspectiva de Aprendizaje y Crecimiento: teniendo en cuenta que los recursos materiales y las personas son las claves del éxito de cualquier organización, esta perspectiva incluye a todos aquellos objetivos que reflejan el éxito de la organización a la hora de la inversión en recursos materiales y el aprendizaje de los trabajadores. Un indicador ejemplo de los objetivos estratégicos incluidos en esta perspectiva sería el porcentaje de trabajadores que dominan una determinada tecnología.

Pese a que estas son las cuatro perspectivas más genéricas, no son obligatorias, y en determinados tipos de empresas surgirán nuevas perspectivas. En la Figura 9 se puede observar cómo interactúan las perspectivas entre sí.

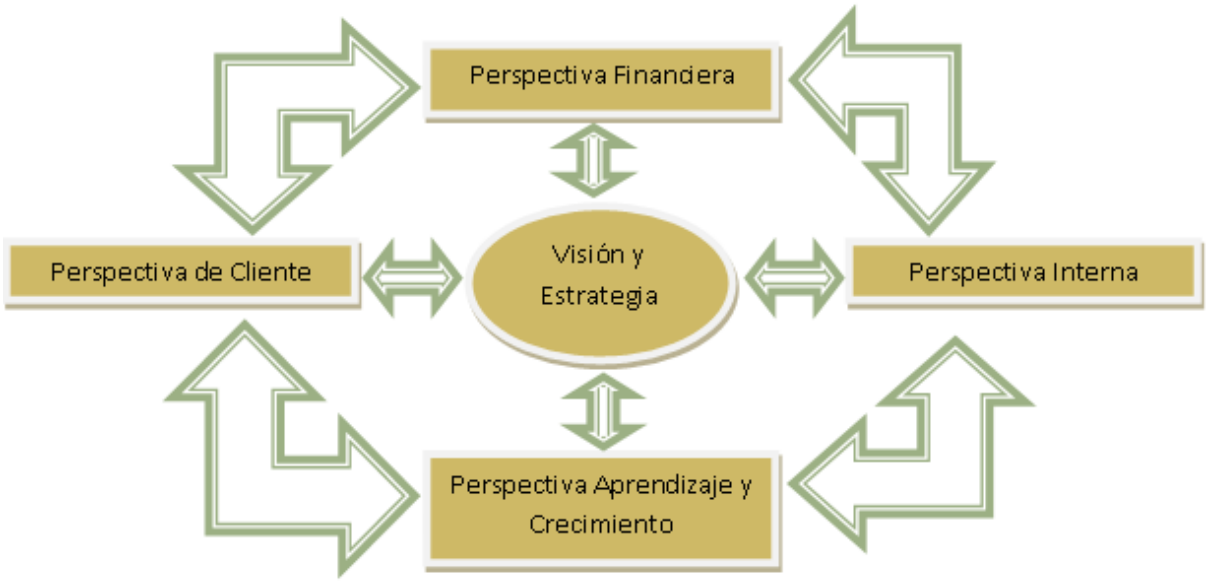


Figura 9.- Diagrama de interacción de las distintas perspectivas

Beneficios de la Implantación de un Cuadro de Mando Integral

- La traducción de un modelo de negocio en indicadores clave facilita el consenso en toda la empresa, no sólo en la dirección.
- Da una visión más amplia de cómo las acciones realizadas día a día afectan no solo al corto plazo, sino al largo plazo.
- Una vez que el cuadro de mando integral está en marcha, se pueden comunicar los planes a toda la organización, aunando esfuerzos y remando toda ella en la misma dirección.
- Permite detectar anomalías en el desarrollo del plan estratégico u operativo.

Riesgos de la Implantación de un Cuadro de Mando Integral

- Un modelo de negocio poco elaborado y sin la colaboración de todos los componentes de la organización es inútil.
- Los indicadores deben ser escogidos con cuidado para no distorsionar el cumplimiento de los objetivos.

Capítulo 3.- Marco Tecnológico

En este capítulo de la memoria se van a detallar todas las tecnologías que componen el marco tecnológico de la presente memoria.

En primer lugar, se describirán herramientas específicas desarrolladas en la PYME que ha adoptado este proyecto, denominadas Resiplus y Resiplus BE. La primera de ellas será presentada brevemente, ya que no es objetivo de la presente el conocer profundamente la aplicación, mientras que la segunda sí que se detallará más en profundidad ya que ofrece un soporte directo a soluciones Business Intelligence.

En segundo lugar, se detallarán las herramientas de Microsoft pertenecientes al marco tecnológico. Estas son el motor de bases de datos Microsoft SQL Server 2008 y el CMS orientado a la empresa Microsoft Office SharePoint 2007, ofreciéndose una breve descripción de ambas con las funcionalidades que ofrecen, y el porqué de la elección de las mismas.

3.1. - Resiplus

Resiplus es un ERP de gestión de residencias para la tercera edad o centros de día desarrollado por la empresa ADD Informática. Con este programa se puede controlar hasta el más mínimo detalle de estos tipos de centros, al objeto de tener almacenada y organizada toda la información de estos de una manera completamente estructurada.

La aplicación Resiplus comenzó a desarrollarse en 1997, y a día de hoy recibe una nueva actualización mensual mejorando la calidad del producto gracias al feedback ofrecido por sus usuarios.

Resiplus puede ser instalado de dos modos distintos: aplicación cliente o aplicación servidor. La aplicación servidor será la encargada de, además de ofrecer una interfaz de usuario para el acceso a los datos, almacenar la base de datos asociada. Mientras tanto, la aplicación cliente tan solo contendrá la interfaz grafica para el acceso a datos.

3.1.1.- Requerimientos de Resiplus

A continuación se enuncian los distintos requerimientos de la aplicación Resiplus:

- Requerimientos software:
  - Sistema operativo Windows 2000, XP, 2003 Server, 2008 Server y Vista.
  - Servidor de bases de datos Microsoft SQL Server.
  - .NET Framework 2.0
- Requerimientos hardware
  - Procesador Pentium III o superior

- 512 MB de memoria RAM en los clientes, 1GB en el servidor
- 160 MB de espacio libre en el disco duro para los clientes, 1GB para el servidor

3.1.2.-Areas Funcionales de Resiplus

Resiplus ofrece la administración, organización y almacenamiento de toda la información producida dentro de los procesos que se llevan a cabo diariamente en residencias de la tercera edad o centros de día.

Entrando un poco más en detalle, se van a explicar las distintas áreas funcionales en las cuales está dividido Resiplus:

- Residentes: una de las áreas más importantes y hacia la que va más centrada Resiplus es hacia la gestión de residentes. Por ello, esta es el área funcional más extensa de la aplicación, ya que se ofrecen diferentes funciones para un control y gestión, orientadas cada una a un tipo de profesional específico dentro de la residencia.
  - Administración: se ofrece la gestión de cualquier dato acerca de un residente, desde los datos personales propios, datos de información de contacto de familiares, etc.
  - Económico: se puede gestionar en todo momento la situación de pagos del residente, además de los criterios a partir de los cuales se valorarán dichos pagos.
  - Médico: permite a cualquier profesional conocer cualquier aspecto medico del residente que va a tratar: tratamientos, patologías, alergias, informes médicos.
  - Enfermería: el área de enfermería incumbe desde el control en los tratamientos de los residentes hasta la atención de enfermería que éste ha recibido.
  - Trabajador Social: aquí el trabajador social podrá manejar toda la información de este ámbito tal como las ayudas a los residentes, incapacidades, pensiones, etc.
  - Psicólogo: el profesional de este ámbito tendrá total control de su información, desde informes individualizados de los residentes hasta un registro de actividades programadas.
  - Animador Sociocultural: permite que se creen desde actividades individualizadas hasta el control de grupos de actividades mediante calendarios.
  - Fisioterapeuta: de un solo vistazo el fisioterapeuta puede obtener información sobre los residentes que puede ir desde su asistencia a actividades hasta el seguimiento de las mismas.
  - Dietista: Resiplus permite a estos profesionales crear dietas genéricas e individualizadas, además de un seguimiento de las mismas.
  - Terapeuta Ocupacional: listados e informes permiten al terapeuta ocupacional estar completamente enterado del estado de los residentes.
  - Supervisores: las actividades del día de este tipo de profesional también están contemplados en la aplicación, ofreciendo un control absoluto sobre los residentes.
  - Gerocultores: permite a este tipo de usuarios una fácil y directa introducción de la información que le atañe, como las caídas, cambios posturales, etc.
  - Atención Espiritual: permite que se lleve un informe personalizado de cada residente en cuanto a atención espiritual.

- Personal: la aplicación permite llevar a cabo un control exhaustivo sobre el personal de la residencia, desde su gestión hasta los plannings de trabajo, datos económicos de los mismos, curriculares, formación recibida, etc.
- Proveedores: en esta parte de la aplicación se permitirá llevar a cabo una valoración de calidad de los distintos proveedores que suministran sus artículos a las residencias o a los centros de día.
- Almacén: permite una gestión de pedidos, albaranes, devoluciones, facturas, etc. todo lo relacionado con la entrada y salida de artículos dentro de la residencia.
- Farmacia y medicamentos: la gestión de farmacia es una de las partes importantes de la aplicación, manteniendo siempre un control exhaustivo sobre la gestión de medicamentos (generación automática de pedidos, listados e dichos pedidos, etc.)
- Gestión económica: en esta área se permitirá la generación de la facturación de la residencia, teniendo en cuenta distintos factores de facturación como la localización del centro.

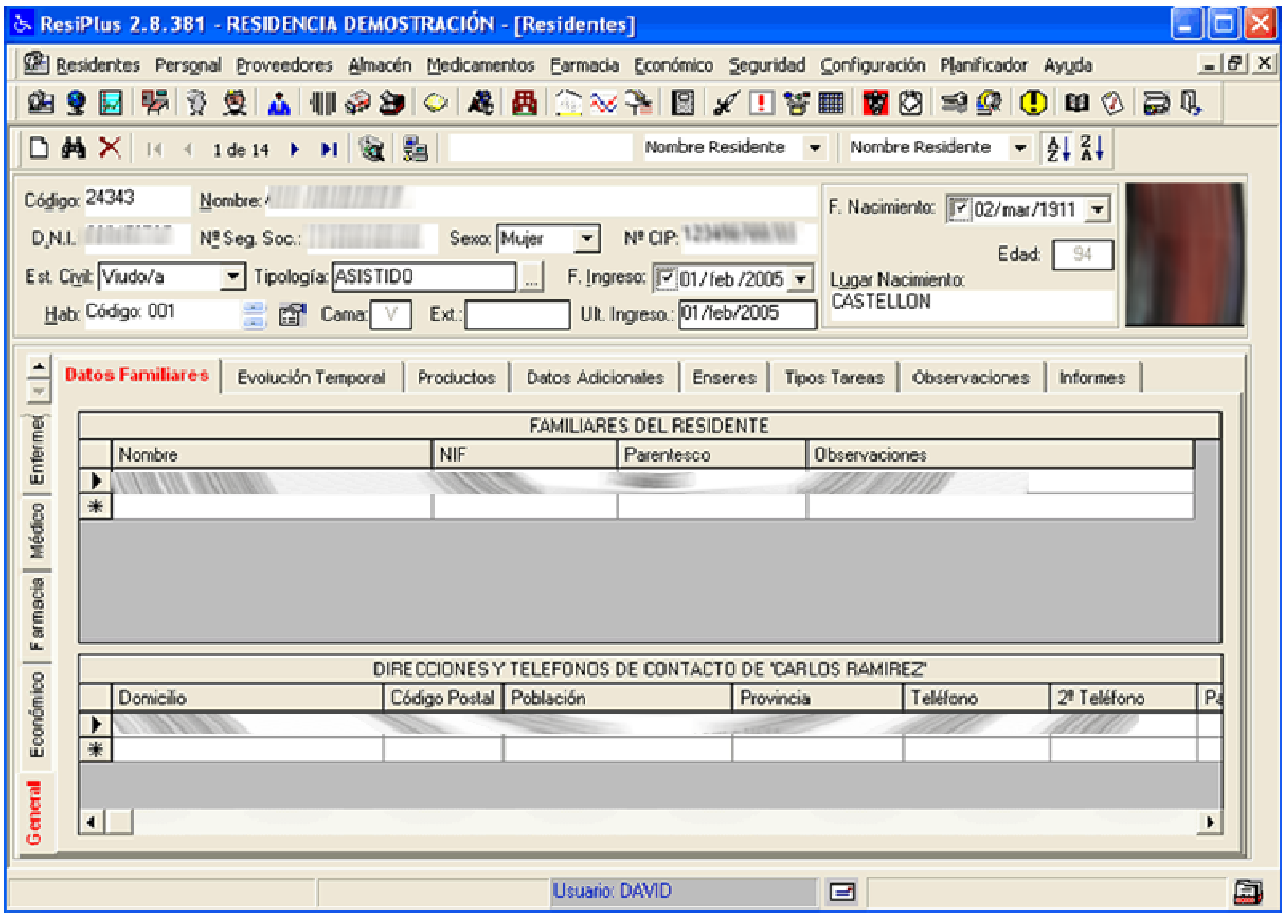


Figura 10.- Interfaz de Residentes dentro de Resiplus

3.1.3.- La Base de Datos de Resiplus

Para acabar, se va a describir cual es la estructura de bases de datos que crea Resiplus cuando este es instalado, ya que será de vital importancia conocer ciertos detalles a la hora de llevar a cabo un proyecto de Business Intelligence.

Cabe destacar cómo se van a estructurar las distintas bases de datos creadas, ya que al objeto de poder administrar el control de un grupo de residencias, la instalación de Resiplus puede estar desagregada en distintas instalaciones.

En la instalación principal o central (también llamada servidor), se creará la base de datos "DatosUsuarios" que contendrá referencias a la localización de cada una de las residencias que compone el grupo.

Por otro lado, en cada centro se crearan cuatro bases de datos denominadas "DatosResidencia_xxx_01", "DatosResidencia_xxx_02", "DatosResidencia_xxx_03" y "DatosResidencia_xxx_04", donde se almacenarán los datos correspondientes a una residencia (dicha desagregación es para que la base de datos no crezca desmedidamente).

3.2. - Resiplus BE – Business Intelligence Edition

Resiplus BE – Business Intelligence Edition es una aplicación Web que permite recopilar información o ejecutar código en tipo real en centros que estén configurados para ello, y mostrar dicha información bien en forma de graficas, en tablas de datos o simplemente guardarla en otro destino como puede ser un almacén de datos. Esta aplicación está desarrollada, al igual que Resiplus, por ADD Informática

Esto es de gran utilidad cuando, por ejemplo, se tienen distintas bases de datos, cuyas estructuras son las mismas y se quiere obtener algún informe o cualquier otro tipo de información estructurada sobre el conjunto de bases de datos.

Con esta utilidad se tiene la capacidad de obtener información de cada una de las bases de datos de Resiplus ejecutando assemblies o consultas SQL predeterminadas, y centralizarla de la forma deseada, ya bien sea una tabla de una base de datos, un informe o un conjunto de graficas.

Dicha información puede ser adquirida de forma manual lanzando exportaciones (concepto que se estudiará en siguientes puntos) o programándolas, pudiendo así realizar una adquisición de datos en un instante de tiempo predefinido por el usuario.

La interfaz de la pantalla principal de Resiplus BE puede ser observada en la Figura 11.

Figura 11.- Interfaz principal de Resiplus BE

3.6.1.- Requerimientos de Resiplus BE

Los requerimientos para que Resiplus BE pueda ejecutarse en una maquina son los siguientes:

- Requerimientos software:
  - Sistema operativo Windows 2000 Professional, Windows 2000 Server, Windows XP Professional o Windows 2003 Server, Windows Vista o Windows 7, todos ellos con Internet Information Server (IIS) funcionando.
  - Framework .NET versión 1.1 (o superior).
  - Base de datos de usuarios de Resiplus, además de Gestión de Usuarios.
  - Microsoft SQL Server 2005 Standard (o superior).
  - Internet Explorer 6 o superior.
- Requerimientos hardware:
  - Al estar contenida en una página web, esta utilidad no tiene requisitos hardware.

3.6.2.- Funcionalidades

A continuación, se van a detallar todas las funcionalidades que ofrece Resiplus BE. Las más importantes para el desarrollo del presente documento serán tratadas detalladamente, mientras que las demás funcionalidades tan solo serán ligeramente introducidas.

Registro de Acceso

El registro de acceso es un formulario en el cual el usuario cumplimentará un formulario de búsqueda, obteniendo como resultado aquellos accesos que se ajusten a los criterios rellenados en el formulario. En la Figura 12 se puede observar la interfaz del registro de acceso.

Los criterios a rellenar en el formulario son los siguientes:

1. Seleccione el o los ficheros accedidos: aquí se pueden seleccionar las distintas partes de Resiplus BE a las que se puede acceder, pudiendo seleccionar una o varias partes para ver los accesos a la misma.
2. Seleccione el o los usuarios: aquí podremos seleccionar el usuario o grupo de usuarios sobre el que queremos realizar la consulta. También se puede realizar una selección múltiple.
3. Seleccione el o los tipos de acceso: el tipo o tipos de acceso utilizados por el usuario o grupos de usuarios seleccionados.
4. Desde fecha: fecha inicial de la búsqueda dentro del registro de acceso.
5. Hasta fecha: fecha final de la búsqueda dentro del registro de acceso.

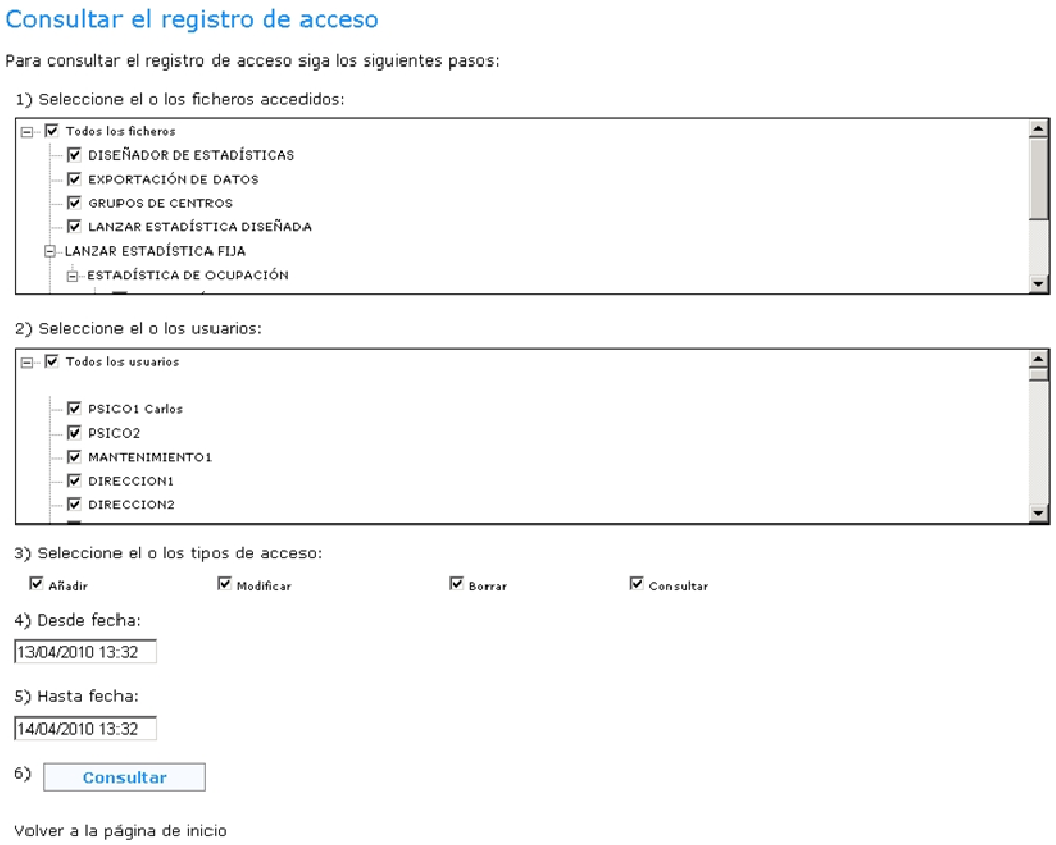


Figura 12.- Interfaz del registro de acceso de Resiplus BE

Grupos de Centros

En grupos de centros podemos realizar agrupaciones de centros dados de alta en la aplicación web. Esto es útil para realizar agrupaciones que faciliten la visualización de estadísticas, como por ejemplo agrupaciones por localización geográfica, o por tamaño de los distintos centros.

Esta funcionalidad nos ofrece tres opciones:

- Crear Grupo: mediante dos sencillos pasos se podrá crear un grupo de centros. En el primero de ellos, se escogerán los centros que pertenecerán al grupo, mientras que en el segundo se le pondrá un nombre al grupo y una descripción al mismo.
- Modificar Grupo: en esta opción se ofrece la posibilidad de modificar todos los parámetros que conforman un grupo, es decir, su nombre, su descripción, y el conjunto de centros que lo conforman.
- Borrar Grupo: aquí se podrá seleccionar un grupo previamente creado al objeto de eliminarlo.

Estos grupos creados se podrán utilizar posteriormente a la hora del lanzamiento de gráficas y exportaciones.

Diseñador de Gráficas

Esta funcionalidad sirve para realizar diseños de gráficas a partir de consultas SQL. La potencia de esta funcionalidad es muy limitada, por lo que su uso es poco frecuente dentro de los clientes.

Gestión de Assemblies

Los assemblies no son más que librerías de código en formato dll que se ejecutarán en el servidor donde esté instalado Resiplus BE o en cualquiera de los centros que estén dados de alta en el mismo.

Estos assemblies serán tratados con más detenimiento en secciones posteriores del siguiente documento, centrándose este punto en explicar la funcionalidad de gestión de assemblies. Concretamente, esta funcionalidad nos ofrece tres opciones:

- Añadir assembly: en esta opción se pondrá un nombre al assembly y una descripción del mismo, teniendo que seleccionar mediante un explorador de Windows la ruta donde está la librería desarrollada. Una vez añadido, este assembly estará presente en otras funcionalidades de Resiplus BE, como por ejemplo exportación de datos.
- Cambiar assembly: aquí se podrá seleccionar un assembly ya creado al objeto de cambiar su librería dll.
- Quitar assembly: en esta opción se pueden eliminar aquellos assemblies que no tengan ya ninguna utilidad dentro de Resiplus BE.

Los assemblies serán parte fundamental de un proceso ETL (extracción – transformación – carga) detallado en el documento, ya que dentro de ellos se realizarán todas aquellas consultas SQL y transformaciones necesarias al objeto de conseguir llevar a cabo dicho proceso. Para realizar dicho proceso será necesaria la creación y lanzamiento de exportaciones de datos.

La captura de una de las interfaces de gestión de assemblies se puede observar en la Figura 13.

Figura 13.- Gestión de assemblies de Resiplus BE

Exportación de Datos

La funcionalidad exportación de datos permite el lanzamiento de exportaciones predefinidas por los usuarios con el fin de crear estadísticas y graficas, ejecutando o no código previamente picado en assemblies.

Mediante esta funcionalidad se conseguirán los datos necesarios para, posteriormente, mediante el diseñador de graficas o el visualizador de indicadores, ver las distintas graficas o tablas para análisis de dicha información.

La funcionalidad exportación de datos está formada por las siguientes opciones:

- Crear exportación: mediante cinco pasos, se podrá crear una exportación de datos para su posterior lanzamiento. Los pasos son los siguientes:
    - Paso 1: Selección la entidad de la que quiere exportar datos. Aquí seleccionaremos aquella entidad u objeto del que se quieren exportar los datos. Entre sus opciones podemos observar residentes, personal, familiares, etc.
    - Paso 2: Seleccione los campos que quiere exportar. Una vez seleccionada la entidad predefinida, se seleccionarán los campos de ésta que quieren ser exportados.
    - Paso 3: Establezca los criterios de selección de los datos que se van a exportar siguiendo los siguientes pasos. En este paso se podrán añadir filtros para que, a la hora de exportar datos, sólo se obtengan aquellos que pasen los filtros definidos por el usuario. Se podrá seleccionar operadores como que contenga, que empiece, que termine, etc. al objeto de tener criterios de filtración lo suficientemente concretos.
    - Paso 4: Seleccione los assemblies que deben ejecutarse con la exportación. Este es el paso clave de las exportaciones de datos. Este paso es visible gracias a la Figura 14. Aquí se deben seleccionar los distintos assemblies que se deben ejecutar, donde y cuando. Esto es debido a que las líneas de código se pueden ejecutar tanto en el servidor donde está instalado Resiplus BE, como en el grupo de residencias que

estén dados de alta en la base de datos de Resiplus. Las estadísticas se pueden lanzar en dos sitios distintos:

- En el unificador: es como se conoce el servidor donde está instalado Resiplus BE.
- En cada centro: en todos y cada uno de los centros seleccionados en el posterior paso de lanzar exportación.

Además, también se puede seleccionar cuando se va a lanzar el assembly, para crear así el orden de precedencias adecuado para la exportación. Las posibilidades son:

- Antes de la estadística: código que se ejecutará en primera instancia, sin enviar los resultados de la ejecución a ningún lado. Opción solo disponible si el assembly se ejecuta en cada centro
- Antes de enviar la estadística: el código de esta opción se ejecutará en primer lugar, mandando su resultado a los centros o al unificador dependiendo del orden especificado.
- Después de enviar la estadística: este código se ejecutará justo después de que se hayan mandado los resultados de la ejecución. Opción solo disponible si el assembly se ejecuta en cada centro.
- Después de la estadística: el código de esta opción se ejecutará después de haber ejecutado todos los paquetes de código.

Figura 14.- Interfaz de creación de exportación de datos en Resiplus BE

- Paso 5: Rellenar los datos de la exportación. Aquí se rellenarán la descripción de la exportación, alguna observación por si fuese necesaria, y para quien va a estar disponible dicha exportación.

- Clonar exportación: en esta opción se ofrece la posibilidad de clonar una exportación previamente creada.

- Modificar exportación: aquí se ofrece la posibilidad de modificar una exportación previamente creada, cambiando sus características de lanzamiento.
- Borrar exportación: en esta opción se podrá eliminar una exportación creada con anterioridad.
- Lanzar exportación: esta opción permite el lanzamiento de exportaciones de datos previamente creadas para que ejecuten el código de sus assemblies en el unificador o en los centros que se indiquen. Los pasos a completar son los siguientes:
    - Paso 1: Seleccione la estadística de exportación de datos que quiere lanzar. Aquí se seleccionará una de las exportaciones previamente creadas
    - Paso 2: Seleccione los grupos de centros o centros sobre los que quiere lanzar la estadística. En este paso se seleccionan los centros en los cuales se quieren ejecutar aquellos assemblies cuyo lugar de ejecución es el centro.
    - Paso 3: Escriba alguna observación para la estadística. Aquí se introducirá una observación para tener el lanzamiento identificado (ya que se pueden lanzar distintas exportaciones con el mismo nombre).

- Ver exportación: en esta opción se pueden observar las distintas exportaciones lanzadas, el porcentaje completado de las mismas, su fecha y hora de inicio, observaciones, quien la lanzó, etc. Todo ello para saber si dicha exportación fue exitosa o surgió algún fallo durante la ejecución de la misma.

La ventana del asistente para el lanzamiento de una estadística de exportación de datos puede ser visionada en la Figura 15.

Figura 15.- Interfaz de lanzamiento de exportaciones en Resiplus BE

Indicadores

En esta sección de la utilidad se podrán explotar todos los indicadores de gestión (concepto explicado en la introducción al capítulo 4) que tenga el cliente implementados en su sistema. Así mismo, se podrán leer las ayudas de los mismos.

Además, dentro de esta sección también se incluye el mantenimiento de las tablas maestras, concepto que será desarrollado en el punto de 4.3, denominado “Consideraciones Generales”.

Programaciones

Esta funcionalidad da la posibilidad al usuario de programar el lanzamiento de exportaciones de datos de una manera bien simple.

Para ello, se puede definir su frecuencia de lanzamiento (cada cuanto se van a lanzar) entre un gran número de posibilidades: diaria, cada x días, solo un día a la semana, solo una vez al mes o al año a una hora determinada… Entre las opciones de programaciones se encuentran:

- Crear programación: crear una programación de una exportación de datos existente
- Clonar programación: crear una copia idéntica de una programación ya existente
- Modificar programación: modificar una programación previamente creada.
- Borrar programación: eliminar una programación que haya sido creada con anterioridad
- Comprobar el estado del servicio: las programaciones funcionan gracias a un servicio web que está siempre en funcionamiento a no ser que se indique lo contrario. Esta opción sirve para desvelar si el servicio está activo y funcionando correctamente.

3.3.- Microsoft SQL Server 2008

Este punto de la memoria ha sido desarrollado a partir de las referencias [TEC1] [TEC2] de la bibliografía.

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. La primera versión de este sistema data de 1989, siendo la versión 2008 la estudiada en la presente memoria.

3.3.1.-Requerimientos de Microsoft SQL Server 2008

Los requerimientos para que Microsoft SQL Server 2008 se pueda ejecutar son los siguientes:

- Requisitos Software:
    - Windows 2003 Server, Windows 2008 Server, Windows Vista o Windows 7
    - Framework .NET versión 3.5
    - Microsoft Windows Installer 4.5
    - Microsoft Internet Explorer 6 (o superior)
- Requisitos Hardware:
    - Procesador de 1 GHz (recomendado 2 GHz)
    - 512 MB de memoria RAM (recomendado 2 GB)
    - 3 GB de espacio libre en el disco duro.

3.3.2.-Tipos de Soluciones Ofrecidas

Entre las soluciones ofrecidas por Microsoft SQL Server 2008 destacan las detalladas a continuación:

OLTP

SQL Server 2008 es uno de los motores de bases de datos más prestigiosos disponibles en la actualidad, ofreciendo grandes posibilidades de escalabilidad, un alto rendimiento a todo tipo de exigencias por parte del usuario e implementando unas directivas de seguridad al objeto de mantener la integridad de los datos.

Business Intelligence

SQL Server 2008 ofrece una gran cantidad de utilidades al objeto de convertirse en una plataforma interesante a la hora de implementar soluciones de Business Intelligence. Algunas de estas utilidades son Analysis Services y Reporting Services (explicadas en puntos posteriores), que permiten al usuario la creación de informes y el análisis de los mismos.

Data Warehouse

SQL Server 2008 se postula como una gran plataforma de data warehouse, ya que permite la implementación de soluciones completas, permitiendo integrar datos en el almacén más rápidamente, y gestionar grandes volúmenes de datos, además de una gran gestión de usuarios a la hora de dividir la información en vistas independientes. Para ello, hace uso de utilidades como Integration Services (explicada en puntos posteriores), especializada en procesos de ETL.

Desarrollo de aplicaciones

SQL Server 2008 es el eje en torno a la que gira una plataforma completa de programación de datos ofrecida por Microsoft, que permite el acceso y la manipulación de datos críticos para el negocio desde cualquier dispositivo, plataforma u origen de datos.

3.3.2.- Componentes Ofrecidos

Entre todas las utilidades ofrecidas, en este punto se va a centrar la atención en aquellas que ofrezcan soporte a la hora de implementar una solución de Business Intelligence, como son Integration Services, Analysis Services y Reporting Services.

Integration Services

SQL Server Integration Services (también conocido como SSIS) es un componente de Microsoft SQL Server que puede ser usado para realizar una gran variedad de tareas de migración de datos y mantenimiento de bases de datos

Para ello, hace uso de paquetes de datos cuyo contenido esta especificado en forma de workflows, convirtiéndose en una herramienta rápida y flexible para la construcción de almacenes de datos y para la creación de aplicaciones de mantenimiento automatizadas de bases de datos y la actualización de cubos de datos multidimensionales.

Analysis Services

Analysis Services es un componente de SQL Server que ofrece dos funcionalidades: el procesamiento analítico en línea (OLAP) y la minería de datos para aplicaciones de Business Intelligence.

Dicho componente admite OLAP para el diseño, creación y administración de estructuras multidimensionales que contienen datos provenientes de distintos orígenes, como bases de datos relacionales.

En cuanto a la minería de datos, Analysis Services permite el diseño, creación y visualización de modelos de minería de datos construidos a partir de orígenes de datos, todo ello mediante el uso de una gran variedad de algoritmos, entre ellos los estudiados en el punto 2.6.6 de la memoria.

Reporting Services

Reporting Services es el componente de SQL Server que satisface las necesidades de información de los usuarios mediante la generación de informes. Este componente está preparado tanto para la generación como para el envió de una gran variedad de informes, tanto impresos como interactivos.

Para ello, este componente puede ser administrado mediante una interfaz web, ofreciendo además una gran selección de servicios web al objeto de soportar el desarrollo de aplicaciones de informes personalizadas.

3.3.3.- Versiones de Microsoft SQL Server 2008

A continuación, se va a exponer un cuadro en el cual se comparan versiones existentes de Microsoft SQL Server 2008, enunciando características y restricciones dependiendo de la versión.

En este recuadro tan solo se van a exponer las versiones más relevantes del producto, que son la versión Express, Standard y Enterprise, con no todas sus características, sino las más relevantes para el desarrollo de la memoria. En el caso de querer ampliar esta información, se puede hacer uso de la referencia [TEC2] de la bibliografía.

	Microsoft SQL Server 2008 Express	Microsoft SQL Server 2008 Standard	Microsoft SQL Server 2008 Enterprise
Escalabilidad			
Compresión y Particionamiento	No	No	Si
Tamaño Máximo de Bases Datos	4 Gb	Ilimitado	Ilimitado
Disponibilidad			
Soporte Multi-Instancias	16	16	50
Snapshots de Bases de Datos	No	No	Si
Compresión de Copias de Seguridad	No	No	Si
Seguridad			
SQL Server Audit (aplicación extra)	No	No	Si
Encriptación Transparente de Bases de Datos	No	No	Si
Administración			
SQL Server Management Studio	Si (Versión Recortada)	Si	Si
SQL Server Agent	No	Si	Si
Herramientas de Desarrollo			
Integración en Microsoft Visual Studio	Si	Si	Si
Business Intelligence Development Studio	No	Si	Si
Integration Services			
Funcionalidad Básica	Si	Si	Si
Desarrollo y Utilidades de Integration Services	No	Si	Si
Análisis Multidimensional			
Analysis Services	No	Si	Si
Características Avanzadas de Analysis Services	No	No	Si
Reporting			
Report Server	No	Si	Si
Report Designer	No	Si	Si
Report Manager	No	Si	Si

Después de ver como componentes indispensables para el desarrollo de una solución de Business Intelligence como serían Analysis Services o el desarrollo con Integration Services están incluidas a partir de la versión Standard del programa, se puede concluir que para llevar a cabo una solución de este tipo es indispensable la adquisición de dicha versión como mínimo, siendo muy recomendable la utilización de la versión Enterprise.

3.3.4.- Elección de Microsoft SQL Server2008 y Posibles Alternativas

Debido a restricciones del marco tecnológico (Resiplus necesita de SQL Server para funcionar), la elección de este motor de bases de datos como tecnología a utilizara no da lugar a ninguna posible alternativa.

A pesar de estar atados en cuanto a este aspecto, la elección de esta tecnología sigue siendo un acierto, ya que con la numerosa cantidad de componentes que ofrece, puede ayudar a llevar a cabo un proyecto de Business Intelligence de numerosas formas.

Por ello, y debido a que esta tecnología ha sido impuesta desde un principio, se explotarán todos los componentes que nos ofrece la herramienta, a destacar entre ellos Integration Services y Analysis Services.

3.4. - Microsoft Office SharePoint Server 2007

Este punto de la memoria ha sido desarrollado a partir de las referencias [TEC3] [TEC4] [TEC5] de la bibliografía.

Microsoft Office SharePoint Server 2007 es un conjunto integrado de funcionalidades de servidor que pueden ayudar a mejorar la eficacia de la empresa al proporcionar utilidades de administración de contenido y búsqueda empresarial globales, acelerando de este modo los procesos empresariales compartidos y facilitando el uso compartido de la información, con el objetivo de obtener una mejor visión empresarial.

Office SharePoint Server 2007 integra las intranets, extranets y aplicaciones Web de toda la empresa en una misma plataforma, en lugar de tener diferentes sistemas independientes los unos de los otros. Además, proporciona a los profesionales de las tecnologías de la información y a los programadores tanto la plataforma como las herramientas necesarias para la administración de servidores, extensibilidad de las aplicaciones e interoperabilidad.

Con todo ello, se puede concluir que Microsoft Office SharePoint Server 2007 es un CMS (Content Management System) completamente enfocado al mundo empresarial, ofreciendo una gran cantidad de servicios que ayudarán a la empresa en casi cualquier cometido que se proponga.

3.4.1.- Requerimientos de Microsoft Office SharePoint Server 2007

Los requisitos para poder correr Microsoft Office SharePoint Server 2007 en un equipo son los siguientes:

- Requisitos Software:
    - Windows 2003 Server, Windosws 2008 Server, Windows Vista o Windows 7
    - Framework .NET versión 3.0
    - Microsoft SQL Server Express Edition
- Requisitos Hardware:
    - Procesador de 2,5 GHz (recomendado doble nucleo de 3 GHz)
    - 1 GB de memoria RAM (recomendado 2 GB)
    - 3 GB de espacio libre en el disco duro (formateado como NTFS)

3.4.2.- Áreas Funcionales de Microsoft Office SharePoint Server 2007

Como se dijo en la introducción, Microsoft Office SharePoint Server se puede definir como un CMS orientado al mundo empresarial, ofreciendo una gran variedad de funcionalidades que permiten llevar a cabo los objetivos de la empresa.

Dicho conjunto de funcionalidades será dividido en distintas áreas funcionales, al objeto que el usuario pueda ver englobadas distintas funcionalidades en categorías parecidas.

Colaboración

Tecnologías que permiten a los equipos trabajar conjuntamente de forma eficaz, proporcionando mecanismos intuitivos, flexibles y seguros para compartir información a través de wikis y blogs, publicando documentos y colaborando en ellos, manteniendo listas de tareas, dirigiendo encuestas, desarrollando y manteniendo plantillas de sitios personalizadas para usos empresariales específicos e implementando flujos de trabajo.

Portales

Instalaciones que ofrecen las capacidades para personalizar la experiencia del usuario de un sitio Web de empresas, dirigir el contenido a varias audiencias en función de conjuntos de reglas, facilitar automáticamente la exploración intuitiva por el sitio Web al adaptar la exploración a los derechos del usuario, proporcionar la completa administración del contenido del sitio y las instalaciones estructurales.

Búsqueda Empresarial

Capacidad fundamental para buscar de forma rápida y sencilla el contenido relevante distribuido en un amplio abanico de sitios: bibliotecas de documentos, repositorios de datos de aplicaciones empresariales y otros orígenes, (incluidos el uso compartido de archivos, varios sitios Web, carpetas

públicas de Microsoft Exchange y bases de datos de Lotus Notes) y para encontrar el personal adecuado para responder a preguntas o involucrarse en proyectos.

Administración de Contenido

Servicios para la creación, publicación y administración de contenido, independientemente de si éste existe en documentos diferentes o está publicado en páginas Web. Los escenarios de administración de contenido incluyen administración de documentos, administración de registros y administración de contenido Web.

Formularios Comerciales e Integración

Capacidad para implementar de forma rápida y eficaz procesos empresariales basados en formularios, (desde diseño a publicación y a acceso a usuarios) mediante exploradores Web estándar o una aplicación de cliente avanzada como Microsoft Office InfoPath 2007. Además, incluye la capacidad de conectarse a sistemas estructurados, como bases de datos y aplicaciones de línea de negocio, y la posibilidad de obtener acceso a esa información de múltiples maneras.

Business Intelligence

Capacidad para proporcionar información fundamental para los objetivos empresariales a través de una amplia gama de mecanismos, desde hojas de cálculo basadas en el servidor que obtienen acceso a los datos profesionales en tiempo real y llevan a cabo sofisticados análisis a la presentación de indicadores clave de rendimiento (KPI) en sitios Web de empresas.

3.4.3.- Elección de Microsoft Office SharePoint Server 2007 y Posibles Alternativas

Si se quieren barajar las distintas alternativas de gestores de contenidos enfocados a la actividad empresarial dentro del mercado, se puede acudir a un estudio realizado por Gartner en el cual, de un vistazo, se pueden observar en un cuadrante los gestores de contenidos empresariales más importantes del mercado posicionados en base a unos atributos. Dicho cuadrante puede ser estudiado en la Figura 16.

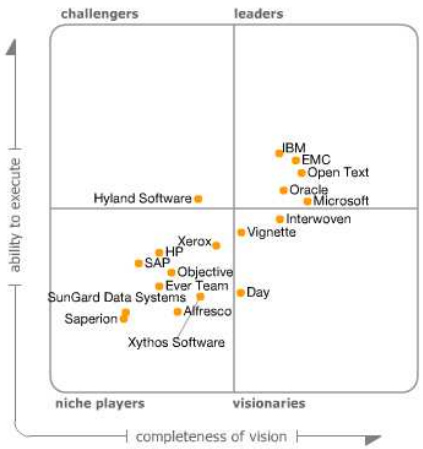


Figura 16.-Cuadrante mágico de Gartner de CMSs enfocados a la empresa

Al estar impuesta por Resiplus la utilización de Microsoft SQL Server 2008, se decidió la utilización de Microsoft Office SharePoint Server 2007 como gestor de contenidos debido a que, además de ser uno de los más potentes del mercado (líder y visionario dentro del cuadrante de Gartner), conforman en su unión un entorno tecnológico ideal a la hora de desarrollar soluciones de Business Intelligence.

Durante el presente documento, Microsoft Office SharePoint Server 2007 será utilizado como gestor de contenidos, más específicamente como el gestor de dashboards para la explotación de la información, estando dichos dashboards desarrollados en el quinto capítulo de la memoria.

Capítulo 4. El Desarrollo de Indicadores de Gestión

Antes de comenzar este capítulo de la memoria, cabe dejar claro el concepto de indicador de gestión, ya que será un concepto utilizado en numerosas ocasiones durante próximos capítulos.

Un indicador de gestión no es más que la definición de un hecho rodeado de una serie de circunstancias en su entorno. Así por ejemplo, se podría definir como indicador de gestión el número de caídas producidas en una residencia durante el mes de Diciembre de 2009.

En este punto de la memoria se aprenderá a desarrollar indicadores de gestión personalizables por el usuario de los mismos a partir del marco tecnológico previamente enunciado.

Para ello, y en primer lugar, se realizará una pequeña introducción al workflow de desarrollo de estos indicadores, realizando breves descripciones de cada una de las fases o tareas que lo componen. En segundo lugar, se enunciará la tecnología utilizada para el desarrollo de este punto de la memoria.

Una vez hecho esto, se presentarán una serie de consideraciones generales que cualquier desarrollador debe tener en cuenta a la hora de implementar un indicador de gestión en el marco tecnológico asociado. En cuarto y quinto lugar, se explicarán los dos métodos de implementación de indicadores de gestión posibles dentro del marco tecnológico enunciado en el capítulo 3.

Para acabar, e ilustrando los demás puntos, se van a desarrollar dos indicadores de gestión, cada uno de ellos por los dos métodos, y servirán para que el lector se haga una idea de cómo debe desarrollarse el proceso de diseño e implementación de un indicador.

4.1.- Workflow de Desarrollo de Indicadores

Como casi cualquier proceso que se lleva a cabo dentro de una empresa de desarrollo de software, dicho proceso puede ser esquematizado y formalizado en un flujo de trabajo o workflow.

Después de utilizar un workflow de desarrollo común a cualquier otro proceso de desarrollo de software, se llegó a la conclusión que a este había que realizarle unas cuantas modificaciones al objeto de que el proceso quedase bien formalizado.

Una vez hechas dichas modificaciones, el workflow resultante puede verse en la Figura 17.

A continuación, se va a hacer un breve paseo por este workflow, al objeto que el lector comprenda quien y que se va a realizar en cada una de las actividades que lo conforman.

TAREA 1.- INTRODUCIR INCIDENCIA (TODOS)

Esta tarea no consiste más que en la introducción del nombre del proyecto de desarrollo de indicadores a llevar a cabo en el planificador de tareas que gestiona los procesos dentro de la empresa. Cualquier usuario está capacitado para realizar dicha tarea.

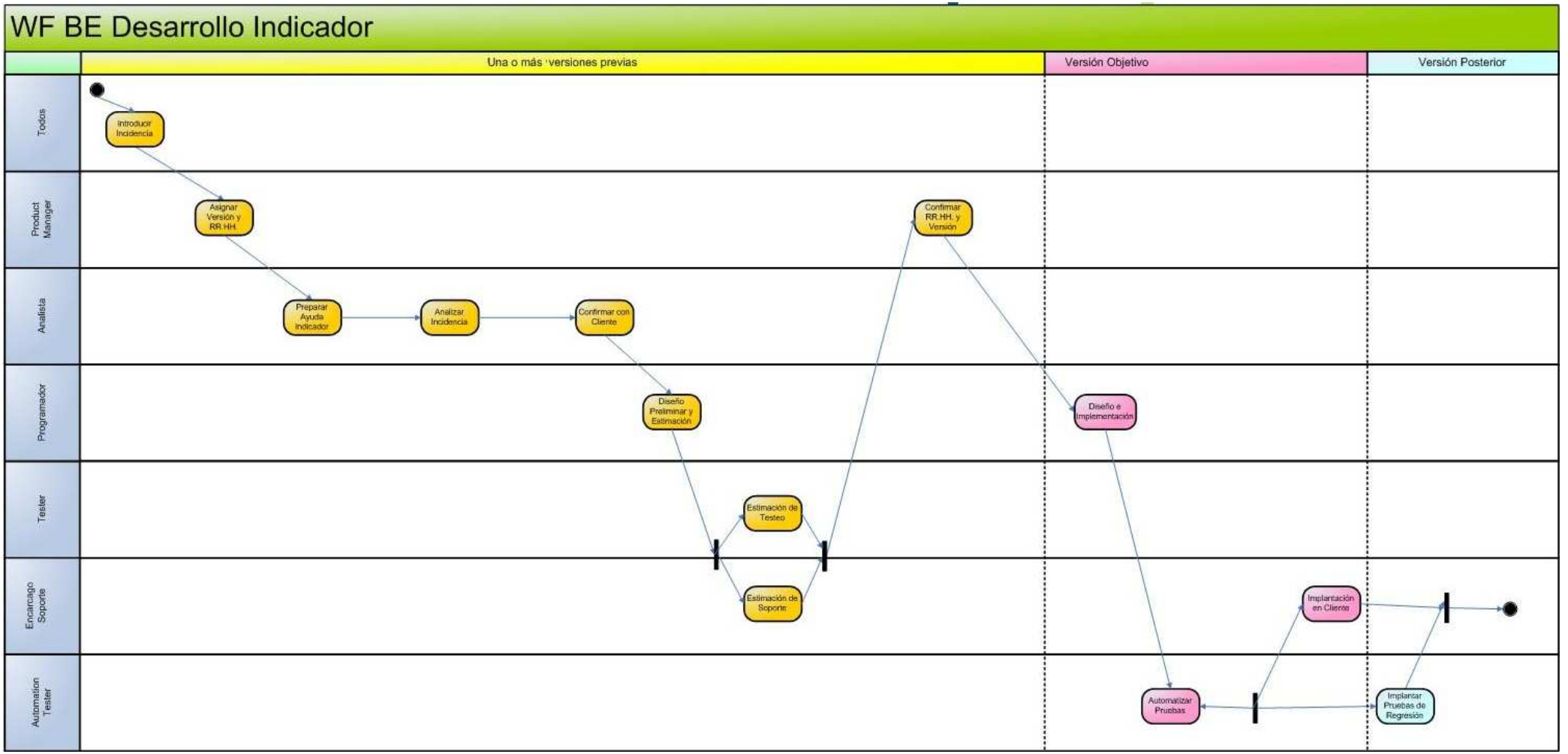


Figura 17.- Workflow de desarrollo de indicadores

Tarea 2.- Asignar Versión y Recursos Humanos (Product Manager)

El product manager del proyecto debe ser el encargado de asignar la versión a la que corresponde el desarrollo, así como concretar quienes serán los encargados de llevar a cabo las tareas del proceso asignándole para ello los distintos roles.

Tarea 3.- Preparar Ayuda del Indicador (Analista)

Previamente al análisis de la incidencia, el analista debe elaborar un pequeño documento indicando donde se pueden localizar en Resiplus las distintas ventanas al objeto de rellenar los datos que serán de relevancia dentro del indicador. Así por ejemplo, si se desea elaborar un indicador de caídas, en esta ayuda se indicará como introducir las caídas y asignarlas a residentes dentro de Resiplus. Dicho documento de ayuda será mandado al cliente que demanda el indicador de gestión.

Tarea 4.- Analizar Incidencia (Analista)

Una vez realizado el documento de ayuda, el analista deberá continuar realizando el análisis del indicador. Para ello, debe elaborar un documento en el cual se debe introducir la especificación del indicador, traducir dicha especificación a tablas de dimensiones y de hechos, introduciendo también parte del contenido del documento de ayuda al objeto que diseñadores y testers sepan donde introducir la información en Resiplus.

Esta es una de las tareas cruciales en el desarrollo de indicadores, y un mal análisis de los mismos puede desembocar en unos resultados completamente diferentes a lo que el cliente espera.

Tarea 5.- Confirmar con el Cliente (Analista)

Cuando ya esté elaborado el documento de análisis, éste debe ser validado con el cliente al objeto de evitar posteriores cambios en el mismo. En esta reunión el analista intentará describir el indicador en términos que el cliente pueda entender, siendo el contenido de la charla una explicación del documento de análisis pero en términos más informales.

Tarea 6.- Diseño Preliminar y Estimación (Programador)

En esta etapa, el programador inicialmente asignado al desarrollo del indicador debe realizar un diseño preliminar del mismo, elaborando para ello un documento de diseño en el cual se adjuntaran claves más específicas para el desarrollo del indicador, como pueden ser el diseño de las tablas de hechos y dimensiones, KPIs, medidas, etc. e incluso algunas pruebas que el programador crea que sean decisivas a la hora de implementar posteriores pruebas de regresión.

Dichos documentos tendrán un contenido muy similar al de los puntos “Definición del Indicador” y “Definición del Almacén de Datos” en los indicadores desarrollados como ejemplo en los puntos de la memoria 4.6 y 4.7, por lo que no se cree de transcendental importancia el adjuntar estos documentos dentro de la memoria.

Una vez elaborado el documento, el programador deberá realizar una estimación aproximada del tiempo que costará implementar el indicador.

Tarea 7.- Estimación de Testeo (Tester)

Con el documento de diseño ya elaborado, el tester encargado de implementar las pruebas de regresión del indicador deberá realizar una estimación de su coste temporal en base a lo contenido en los documentos de análisis y diseño.

Tarea 8.- Estimación de Soporte (Soporte)

El departamento de soporte será el encargado de implementar el indicador dentro del cliente, por lo que se deberá realizar una estimación del coste temporal que esto supondrá.

Tarea 9.- Confirmar Recursos Humano y Versión (Product Manager)

Al objeto de evitar la sobrecarga de agentes a la hora de implementar un gran conjunto de indicadores, durante esta fase el product manager confirmará o reestructurará la asignación de los agentes a cada una de las tareas del proceso, confirmando también la versión para la cual el indicador estará implementado.

Tarea 10.- Diseño e Implementación (Programador)

Esta es la tarea más costosa dentro de todo el workflow de desarrollo de indicadores. En ella, el programador deberá implementar el indicador, estando este proceso explicado detalladamente en el actual capítulo de la memoria.

Además, y conforme se implementa el indicador, el programador deberá ir anotando en el documento de diseño pruebas interesantes para que posteriormente el tester las automatice.

Tarea 11.- Automatizar Pruebas (Tester)

Una vez el indicador está implementado, el tester comenzará con la implementación y automatización de las pruebas que hay que realizar para comprobar que el indicador funciona de la manera idónea.

Tarea 12.- Implantación en Cliente (Soporte)

A la vez que se realiza la anterior tarea, el encargado de soporte deberá ir preparando al cliente para la implementación del indicador, dando luz verde a dicha implementación el tester cuando éste considere que el indicador funciona correctamente.

Tarea 13.- Implantar Pruebas de Regresión (Tester)

Una vez el indicador esta implementado en el cliente, y al objeto de evitar que por posibles cambios en la base de datos de Resiplus los indicadores dejen de funcionar correctamente, las pruebas automatizadas se pondrán en regresión, donde estarán en ejecución continuamente siendo la empresa misma la primera enterada en el caso de que cualquier cambio trastoque el funcionamiento del indicador.

4.2.- Tecnología Utilizada

A lo largo de este capítulo de la memoria, y además del marco tecnológico enunciado durante el capítulo 3 de la memoria, se hará uso para el desarrollo de indicadores el kit de desarrollo de software Microsoft Visual Studio 2008.

4.2.1.- Microsoft Visual Studio 2008

Microsoft Visual Studio es un entorno de desarrollo de software para sistemas operativos Windows. Soporta una gran cantidad de lenguajes de programación, como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.

La versión inicial (Microsoft Visual Studio v6.0) data de finales de julio de 1998, y la última versión que está siendo desarrollada en la actualidad es la 2010, a pesar de que en el presente documento se estudiará la versión 2008.

Dicha versión permite a los desarrolladores la creación de aplicaciones, sitios y aplicaciones web, servicios web en cualquier entorno que soporte la plataforma .NET, servicios Windows… una gran cantidad de proyectos que pueden intercomunicarse entre sí independientemente del medio en el que se hallen implementados.

La Interfaz del Entorno de Desarrollo de Visual Studio 2008

En este punto se va a detallar como es la estructura de la interfaz de cualquier proyecto de desarrollo de software, detallando cuales son las partes comunes a la gran mayoría de tipos de proyectos.

En la Figura 18 se puede observar como es la interfaz de Visual Studio 2008.

En la parte superior de la interfaz, se puede observar la barra de herramientas típica de cualquier aplicación de Windows, además de la común botonera en las cuales están los accesos directos a las funciones más utilizadas de la aplicación.

En la izquierda, se tiene la caja herramientas, desde la cual se pueden arrastrar y soltar distintos componentes dependiendo del tipo de proyecto que se esté implementando. Así por ejemplo, en el caso de un proyecto de programación visual como Visual C#, se encontraran componentes como formularios, etiquetas, cuadros de texto, etc. siendo esta interfaz completamente diferente a la encontrada, por ejemplo, en un proyecto de Integration Services.

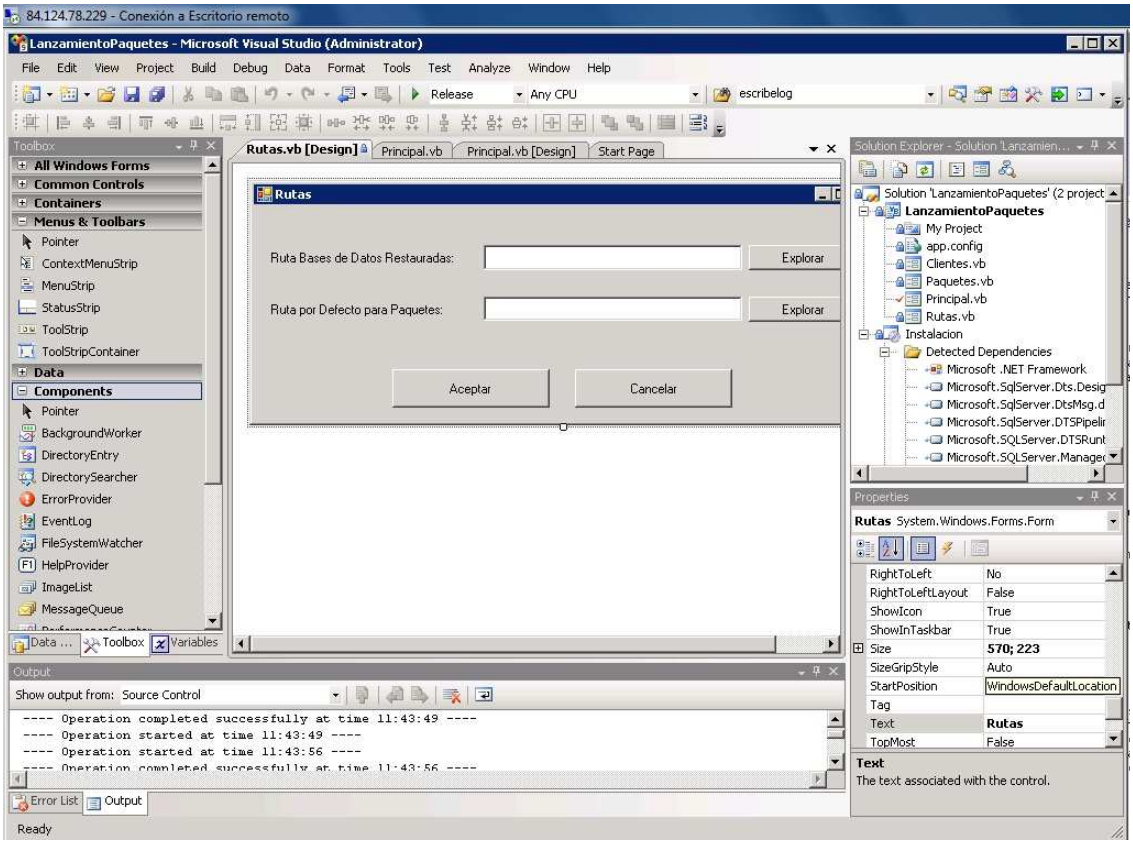


Figura 18.- Interfaz de desarrollo de Microsoft Visual Studio 2008

En la parte inferior se tiene la barra de salida y errores, común a todos los tipos de proyectos, y que informa al usuario en caso de que durante la compilación de la aplicación desarrollada haya habido un error o una advertencia.

En la derecha se puede observar el explorador de la solución, donde se reflejara la estructura de archivos y carpetas del proyecto que se esté desarrollando. Dicha estructura será diferente dependiendo del tipo de proyecto desarrollado.

Por último, y en la parte central, se tiene el escritorio de desarrollo, que es donde se irán abriendo los diferentes archivos de la solución, pudiendo arrastra y soltar componentes o funciones de la caja de herramientas dentro de los mismos.

Elección de Microsoft Visual Studio 2008 y Posibles Alternativas

La elección de Microsoft Visual Studio 2008 como entorno de desarrollo de software es indiscutible para realizar el trabajo presente en la memoria: es el único software que permite el desarrollo de proyectos de cubos multidimensionales de Analysis Services y paquetes de Integration Services (estudiados en puntos posteriores).

Por ello, el estudio de posibles alternativas queda relegado ya que estos dos tipos de proyectos son obligatorios a la hora de llevar a cabo la solución de Business Intelligence tal y como se pensó desde un inicio.

4.3.- Consideraciones Generales

Antes de comenzar con la implementación de indicadores de gestión, hay que tener en cuenta un par de factores existentes en el entorno en el cual se desean desarrollar dichos indicadores.

El primero de ellos, y quizás el más importante, es que los indicadores van a atacar a distintas bases de datos, cada una correspondiente a un instalación de Resiplus independiente. Esto parece un factor no demasiado importante, pero es vital ahondar un poco en él.

En cada una de estas bases de datos tendremos datos que serán independientes de una residencia a otra, como por ejemplo los residentes. Dichos residentes tendrán su identificador propio en su base de datos de Resiplus, cosa lógica por otro lado. Pero hay que tener en cuenta que realizando un proyecto de almacenes de datos, lo que se desea es tener todos esos datos en una misma tabla, independientemente del centro del que provengan, y tenerlos identificados en todo momento.

Por ejemplo, se da el caso de que el residente con identificador 1 de la residencia 001 es Vicente, mientras que el identificador 1 de la residencia 002 es Juan. En el momento en el que estos datos se introduzcan en la tabla de dimensión "Residentes" del almacén de datos, su identificador se perderá (ya que no pueden ser repetidos), y se dejará de saber de dónde proviene cada residente.

Por ello es crucial la creación de un identificador propio del almacén, que a su vez enlace con el registro correspondiente a su base de datos origen. A este identificador se le pasará a denominar código unificado, que estará formado en estos casos por el identificador de la residencia a la que pertenece el registro, un guion y el identificador propio del registro.

Así pues, en el ejemplo enunciado anteriormente, en el almacén se tendrían dos registros: Vicente con código unificado 001-1, y Juan con código unificado 002-1.

Este código unificado deberá estar presente tanto en el almacén de datos como en la base de datos origen, para poder así en todo momento realizar un enlace entre estas dos instancias.

En las ilustraciones 23 y 24, se puede ver la variación introducida y el efecto que este produce:

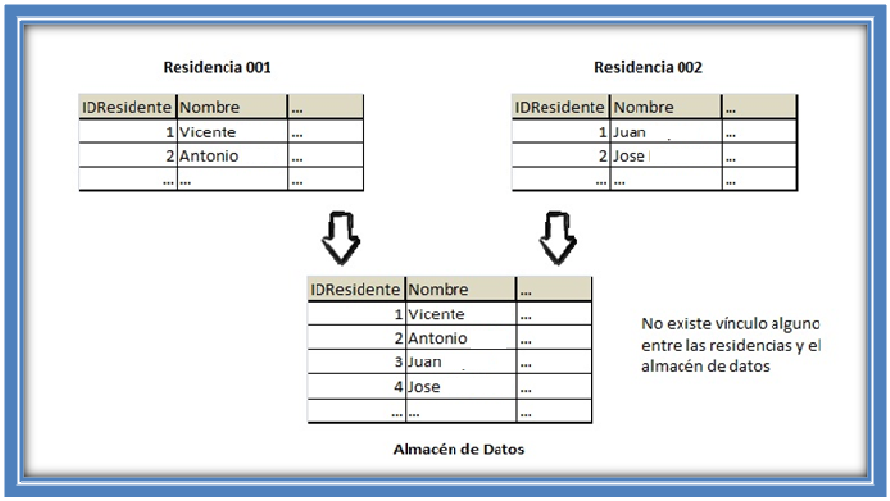


Figura 19.- Ejemplo de la no-relación entre almacén y residencias

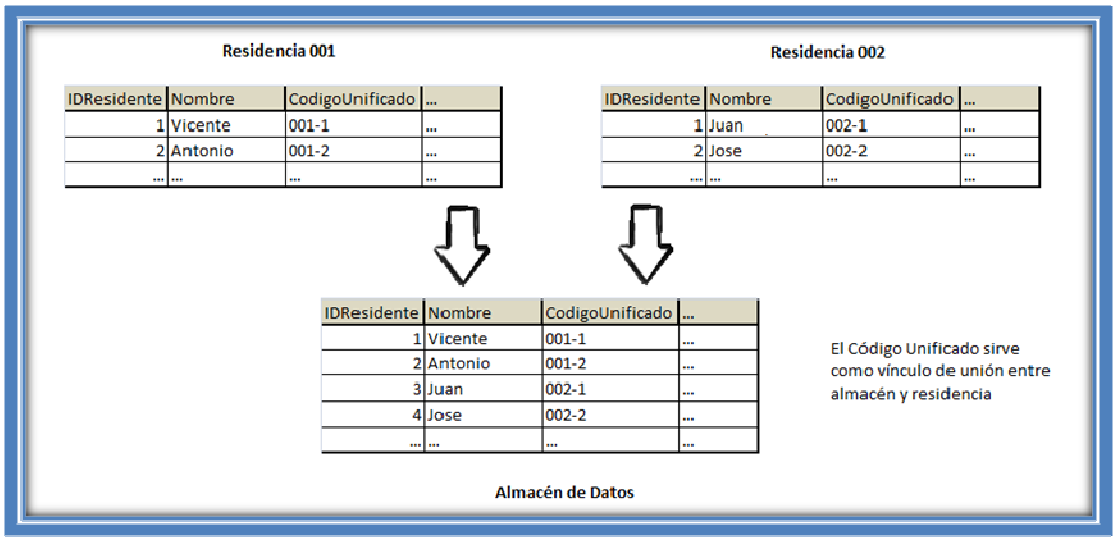


Figura 20.- Ejemplo de la relación entre almacén y residencias mediante el código unificado

El segundo factor a tener en cuenta a la hora de rellenar tablas importantes de la base de datos de Resiplus son los distintos criterios de introducción de datos en algunas tablas.

Por ejemplo, imagínese dos residencias del mismo cliente en las cuales se tienen un conjunto de caracteres de plaza distintos e introducidos por los usuarios: en la residencia 001 se cuenta con los caracteres de plaza público, semipúblico, semiprivado y privado; mientras tanto, en la residencia 002 se tienen los caracteres plaza público, cuasi-público, cuasi-privado y privado.

Si se tiene en cuenta que existe una correspondencia entre los caracteres de plaza (es decir, semipúblico y cuasi-público son los mismos caracteres de plaza, así como semiprivado y cuasi-privado), se tendría un grave problema ya que con una introducción directa de los registros se obtendrían seis caracteres de plaza distintos, a pesar de ser tan solo cuatro: público, privado y las uniones resultantes de semipúblico y cuasi-público y semiprivado y cuasi-privado.

Por ello, y al objeto de que en los informes de explotación la información reflejada sea lo más real posible, se introduce aquí también el concepto de código unificado, pero con una visión un tanto distinta a la vista anteriormente, y un nuevo concepto denominado tablas maestras, que serán las tablas correspondientes al almacén de datos a rellenar, que podrán ser accedidas desde Resiplus BE. Por ejemplo, para este caso, se debería rellenar la tabla maestra de Carácter de Plaza.

A diferencia del uso anterior del código unificado, en este caso hay que seguir unos pasos bien distintos, ya que si bien el proceso de relleno de códigos unificados anterior puede ser realizado de una forma automática, aquí deberán entrar en juego distintos agentes.

En primer lugar, se tiene en cuenta que cada residencia, de partida, tendrá sus propios caracteres de plaza previamente introducidos, con valores que podrían ser distintos en cada residencia. Por ello, el encargado de la empresa para el mantenimiento de los indicadores debe ocuparse de recopilar todo este conjunto de valores.

Una vez recopilados, requerirá de una reunión con los responsables correspondientes al objeto de unificar los criterios de todas las residencias. Una vez hecho esto, y mediante Resiplus BE, el encargado de mantenimiento de indicadores deberá rellenar la tabla maestra de dimensiones (tabla

del almacén de datos) correspondiente con la unificación resultante de la reunión con los responsables, comunicando a cada una de las residencias los códigos unificados de cada uno de los caracteres de plaza que hay introducidos en las bases de datos.

De este modo, se ha conseguido crear un enlace entre los registros de carácter de plaza de cada una de las residencias con el introducido en el almacén, al igual que anteriormente se consiguió con los residentes.

En la Figura 21 se observa como tres residencias con distintos caracteres de plaza logran unificarlo en tan solo uno:

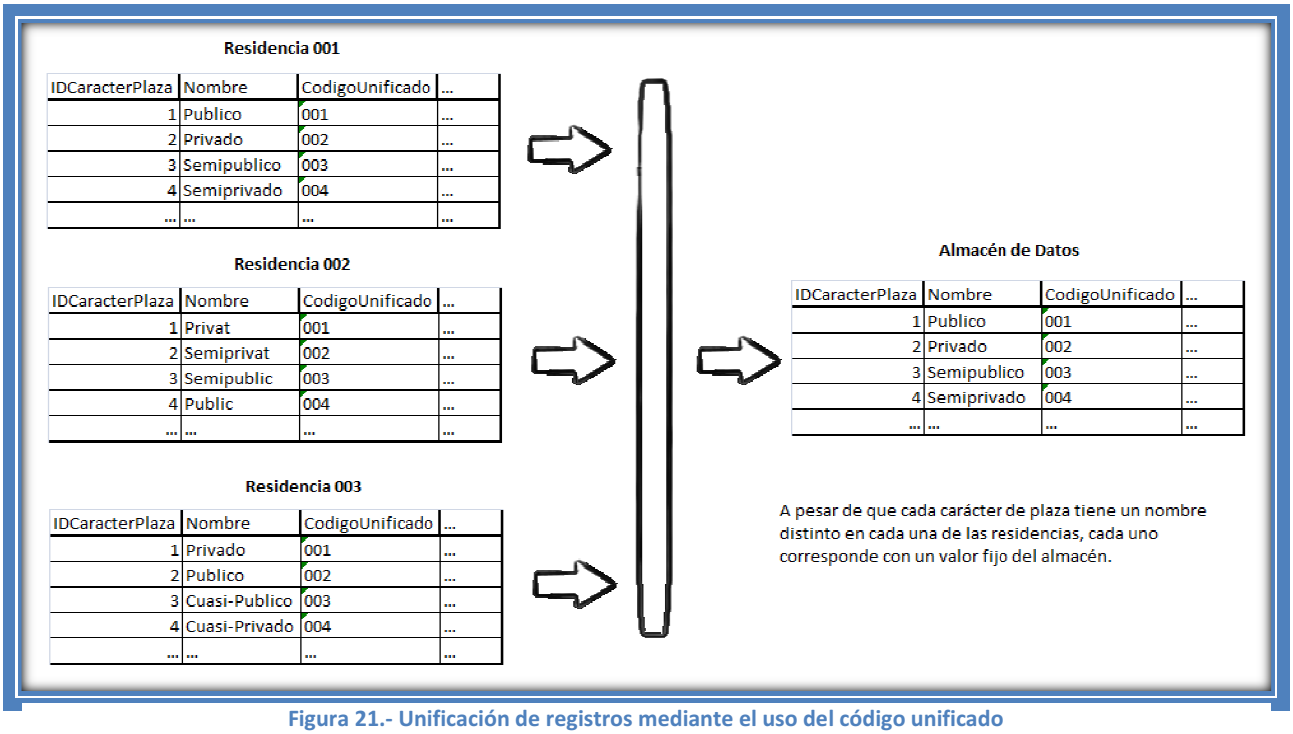


Figura 21.- Unificación de registros mediante el uso del código unificado

Cabe destacar que la aplicación de este método será posible siempre que el compromiso entre residencias y administración central (donde esté instalado Resiplus BE) sea fuerte, y haya una buena comunicación si, por ejemplo, se agrega un nuevo carácter plaza el cual deberá ser definido por la administración central adjudicándole un carácter de plaza propio o uno que ya estuviese anteriormente.

A pesar de lo pesado de la tarea, con ella se podrán conseguir los mejores resultados, ya que la variación en el nombre de registros en distintas residencias es algo muy habitual en el entorno estudiado.

4.4.- Desarrollo por el Método de Assemblies

El primer método de desarrollo que se va a esquematizar en la presente memoria es el método de assemblies. Se recuerda que un assembly es un fragmento de código compilado de modo que puede ser incluido dentro de Resiplus BE al objeto de que este código pueda ser ejecutado en distintas localizaciones mediante exportaciones, obteniendo un resultado que será almacenado en el servidor principal del cliente.

Este subapartado de la memoria comenzará comentando la estrategia elegida para el despliegue de indicadores, y a continuación se explicarán detenidamente las funciones principales de cada uno de los tipos de assemblies en los que se basa el proyecto. En primer lugar, se describirá el assembly de creación del almacén de datos, se seguirá explicando los assemblies de dimensión, en tercer lugar se determinarán las funciones de los assemblies de hechos, y para acabar se puntualizarán aquellas que sean importantes para el desarrollo del unificador – rellenador.

4.4.1.- Estrategia de Despliegue de Indicadores

Para realizar el proyecto de almacén de datos y explotación del mismo, se tuvo que decidir una estrategia de ETL uniforme a la hora de desarrollar la solución mediante assemblies. Después de debatir diferentes opciones, se decidió la opción más disgregada posible: cada uno de los indicadores y de las dimensiones que lo compongan tendrán un assembly propio, además del assembly propio de almacén, que serán lanzados cada uno en una exportación independiente.

Por ello, la siguiente tarea fue separar el proceso para que este fuese ordenado. A priori, este orden parece muy sencillo, pero es de vital importancia que este orden sea materializado, ya que si no pueden aparecer distintas restricciones de integridad que pueden llevar al hecho de que el despliegue de indicadores sea fallido.

En la Figura 22 se puede observar el orden de despliegue.

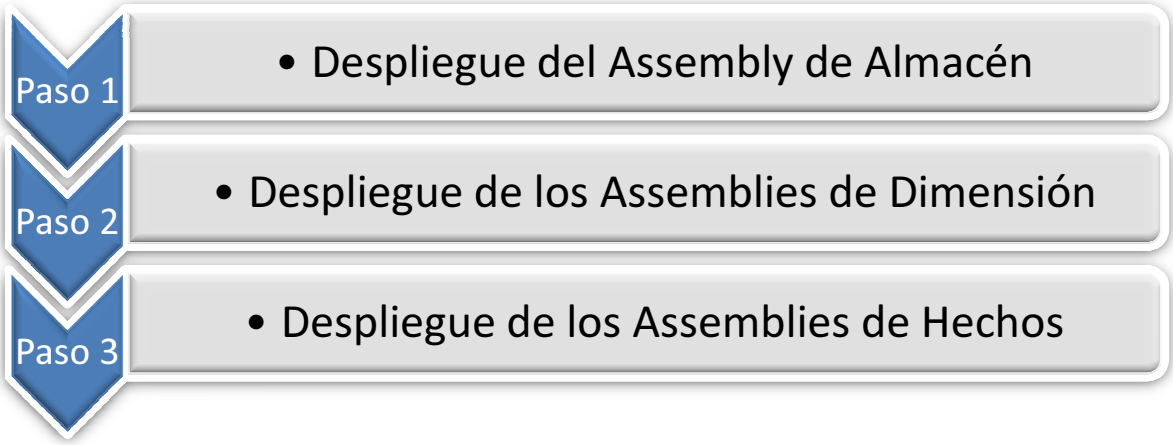


Figura 22: Diagrama de Despliegue de Indicadores

Este orden es el lógico a seguir al objeto de no caer en errores de restricciones de integridad de base de datos, referencias a valores nulos…

Si se estudia detenidamente, es imposible cargar los datos de las dimensiones (paso 2) y de los hechos (paso 3) sin tener antes las tablas necesarias para ser rellenadas (paso 1). Asimismo, es ya sabido que los datos de las tablas de hechos (paso 3) van a hacer referencia a los datos cargados en las tablas de dimensiones (paso 2).

Por ello, es de extrema importancia que el despliegue de los assemblies sea realizado en este orden, ya que si no se producirán errores como los ya comentados anteriormente, los cuales no permitirán a los usuarios la explotación de sus indicadores.

Cabe destacar que los assemblies de extracción de datos (dimensiones y hechos) necesitan de otro assembly que transforme y cargue los datos en el servidor destino. Este assembly se denominará unificador – rellenador, y deberá ser añadido en todas y cada una de las exportaciones de dimensiones o hechos que se realicen.

Una vez dicho esto, y teniendo en cuenta que los assemblies no son más que un código que se ejecuta en distintas ubicaciones, haciendo operaciones de extracción – transformación – carga, puede deducirse que estos assemblies van a ser muy similares entre sí, sobre todo aquellos que son del mismo tipo.

Por ello, los assemblies del mismo tipo usaran unas funciones predeterminadas que serán modificadas al objeto que estas hagan las operaciones que el programador desee. En los siguientes subpuntos de la memoria se explicarán uno a uno los tres tipos de assemblies que se van a desarrollar, detallando un poco mas esas funciones que el programador deberá modificar.

Además, en último lugar, se hablará de un assembly peculiar, el assembly rellenador de dimensiones genéricas, del cual se explicará su razón de ser y posteriormente su funcionamiento.

4.4.2.- Assembly Creador del Almacén de Datos

Este assembly realiza la tarea de crear el almacén de datos. Para ello, se ejecutan una serie de funciones que crean todas las tablas necesarias en el almacén de datos, todas las claves primarias y ajenas de las mismas, relaciones…

A continuación se comentarán las funciones contenidas dentro de la rutina principal del assembly, detallando las más interesantes:

createDatabase(usuario,password)

Esta función se conecta a SQL Server con el usuario y el password introducidos como argumentos. Una vez conectado, se crea la base de datos con nombre especificado en una variable global del assembly. Si dicha base de datos ya existe, esta no se creará.

changeDatabase(BBDD)

Una vez creada la base de datos, con esta función se cambia la conexión de modo que ya no esté conectada a SQL Server, sino a la base de datos creada en el paso anterior.

createTableNombreTabla(nombretabla)

En este paso se crearán todas y cada una de las tablas de dimensiones y hechos necesarios en el almacén de datos. Para ello, cada tabla necesitará de una función de este tipo, necesitando por ejemplo la tabla "Dim_Centro" del almacén la función createTableDimCentro("Dim_Centro").

A continuación se adjunta el código fuente de una función ejemplo, en este caso createTableFactCuras("Fact_Curas"), que creará la tabla de hechos de curas:

Código 1.- Creación de la tabla "Fact_Curas"

```
static void createTableFactCuras(string tabla)
{
    ArrayList atributos = new ArrayList();
    ArrayList indicesUnicos = new ArrayList();
    ArrayList indicesNoUnicos = new ArrayList();

    atributos.Add("IDTiempo int NOT NULL REFERENCES
    Dim_Tiempo(IDTiempo)");
    atributos.Add("IDResidente int NOT NULL REFERENCES
    Dim_Residente(IDResidente)");
    atributos.Add("IDCaracterPlaza int REFERENCES
    Dim_CaracterPlaza(IDCaracterPlaza)");
    atributos.Add("IDTipologia int REFERENCES
    Dim_Tipologia(IDTipologia)");
    atributos.Add("IDCentro int NOT NULL REFERENCES
    Dim_Centro(IDCentro)");
    atributos.Add("IDCurasOrigen int REFERENCES
    Dim_CurasOrigen(IDCurasOrigen)");
    atributos.Add("IDCurasLocalizacion int REFERENCES
    Dim_CurasLocalizacion(IDCurasLocalizacion)");
    atributos.Add("IDCurasEstadio int REFERENCES
    Dim_CurasEstadio(IDCurasEstadio)");
    atributos.Add("NumeroCuras int NOT NULL");

    createTable(tabla, atributos);

    #region indexesUniques

    createIndexUnique(tabla, indicesUnicos);

    #endregion

    #region indexesNonUniques

    indicesNoUnicos.Add("IDTiempo");
    indicesNoUnicos.Add("IDResidente");
    indicesNoUnicos.Add("IDCaracterPlaza");
    indicesNoUnicos.Add("IDTipologia");
    indicesNoUnicos.Add("IDCentro");
    indicesNoUnicos.Add("IDCurasOrigen");
    indicesNoUnicos.Add("IDCurasLocalizacion");
    indicesNoUnicos.Add("IDCurasEstadio");

    createIndexNonUnique(tabla, indicesNoUnicos);

    #endregion
```

```
}
```

createTable(nombretabla, atributos)

A esta función se le pasan el vector de atributos que se crearon en la función anterior, que deben estar contenidos en la tabla correspondiente al parámetro nombretabla.

createIndexUnique(nombretabla, indicesUnicos)

En esta función se ejecutarán los comandos necesarios para dar de alta como claves primarias los miembros del vector de indicesUnicos en la tabla correspondiente.

createIndexNonUnique(nombretabla, indicesNoUnicos)

Mediante esta función se darán de alta en la tabla nombretabla las claves ajenas introducidas en el vector indicesNoUnicos.

createUser()

Esta función crea un nuevo rol de usuario para la conexión a la base de datos creada en funciones anteriores.

4.4.3.- Assemblies de Dimensión

Este tipo de assembly se ocupa de adquirir aquellos registros necesarios de aquellas tablas del origen de datos que se consideren dimensiones, realizando las transformaciones necesarias para que el assembly unificador - rellenador tan solo tenga que cargar los datos en el destino oportuno.

Cabe destacar que es recomendable tener un assembly para cada dimensión que se desee cargar, ya que esto ofrece ventajas a la hora de corregir fallos, además de disminuir la complejidad del código del assembly unificador - rellenador.

Como ejemplo de este tipo de assembly se ha escogido el de la dimensión Residente, explicándose las funciones principales que se realizan dentro de la rutina principal del mismo:

crearCodigoUnificado()

Con esta función se creara la columna código unificado en la fuente de datos en el caso de que este no existiese.

actualizarCodigoUnificado()

En esta función, en primer lugar, se realiza una consulta a la fuente de datos para conseguir todas las claves primarias de los residentes, creando además otra columna que será el código unificado de cada uno. Una vez hecho esto, solo queda actualizar cada uno de los residentes en el origen de datos con su código unificado.

obtenerCamposNombreIndicador(resultado)

Mediante esta función se adquirirán los datos correspondientes a la dimensión mediante una consulta SQL que no se limitará a obtener tan solo el dato crucial del nombre del elemento, sino que también se adquirirán todos aquellos campos que quieran ser guardados en la dimensión, además del código unificado, elemento crucial para tener enlazadas las bases de datos origen y destino.

Por ejemplo, el assembly de la dimensión Residente usado en el presente proyecto adquiere además del código unificado, el nombre del residente.

normalizarCampos(resultado)

Esta función realiza un proceso de normalización de los datos adquiridos para tener un formato común en el almacén de datos destino. Por ejemplo, la función retirará en el campo nombre de residentes caracteres tales como la coma o el apostrofe, sustituyéndolos por un espacio. Esta función es útil para tener unos datos lo más homogéneos posibles.

Cabe destacar que las consultas y transacciones (para adquirir registros y actualizar códigos unificados) que se realizan dentro de este assembly son de complejidad muy baja.

4.4.4.- Assembly de Hechos

Este tipo de assembly realizara la tarea de adquirir aquellos registros necesarios de aquellas tablas del origen de datos que se consideren hechos, realizando las transformaciones necesarias para que el assembly unificador - rellenador tan solo tenga que cargar los datos en el destino oportuno.

Al igual que en las dimensiones, es recomendable tener un assembly por cada tabla de hechos que se desee rellenar, siendo las ventajas las mismas que las anunciadas anteriormente.

Para ejemplificar este tipo de assemblies se ha escogido el hecho de Caídas. Dicho assembly posee dentro de su rutina principal las siguientes funciones:

crearTablaResultado(resultado)

Con esta función se crea la tabla que en pasos posteriores se le pasara al unificador - rellenador para que este realice la tarea de carga. Como normal general, esta tabla tendrá como columnas los códigos unificados de cada una de las dimensiones a las que haga referencia el hecho y la medida del mismo en el caso de que esta fuese necesaria.

GetFactTableData(tabla)

Esta es la función clave de este assembly. Mediante la ejecución de una complicada consulta SQL se obtiene una tabla que tendrá la gran mayoría de las columnas a rellenar en la tabla de resultado. A continuación se muestra la consulta introducida en el assembly del hecho de Curas al objeto de ejemplificar la dificultad de la misma:

Código 2.- Consulta SQL del Hecho Curas

```
SELECT rcu.IDResidente AS IdResidente, res.CodigoUnificado AS ResidenteUnificado,
rcu.IDOrigen AS IDOrigen, cor.CodigoUnificado AS OrigenUnificado,
rcu.IDLocalizacion AS IDLocalizacion, clo.CodigoUnificado AS
LocalizacionUnificado, rcu.IDEstadio AS IDEstadio, ces.CodigoUnificado AS
EstadioUnificado, rcu.fechahora AS Tiempo
FROM ResiCuras AS rcu
INNER JOIN (SELECT IDResidente, CodigoUnificado FROM Residentes) AS res
ON rcu.idresidente=res.idresidente
LEFT JOIN (SELECT idorigen, CodigoUnificado FROM CurASOrigen) AS cor
ON rcu.idorigen=cor.idorigen
LEFT JOIN (SELECT idlocalizacion, CodigoUnificado FROM CurasLocalizacion) AS clo
ON rcu.idlocalizacion=clo.idlocalizacion
LEFT JOIN (SELECT idestadio, CodigoUnificado FROM CurasEstadio) AS ces
ON rcu.idestadio=ces.idestadio
```

Cabe destacar que esta función adquiere los códigos unificados de cada una de las dimensiones presentes en la tabla de hechos. Esto es porque este assembly se ejecuta en el centro, y no en el servidor central donde está el almacén, pudiendo haber identificadores repetidos entre las distintas bases de datos de las residencias. Por ello es de vital importancia el código unificado, ya que es la única manera de poder enlazar los registros del origen de datos (base de datos de la residencia) con el destino (almacén de datos).

Una vez hecho esto, se procederá a la creación de la tabla de resultados parcial mediante la función crearTablaResultado, y rellenándola mediante la asignación de los valores conseguidos con la consulta SQL, añadiendo un plus de información mediante las siguientes dos funciones:

GetCaracterPlaza(IDResidente, fecha, conexion)

Debido a que todos los hechos suelen tener como dimensión asociada el carácter plaza del residente, siendo este un atributo variable y difícil de obtener a partir de la fuente de datos, se ha creado una función específica para obtener el carácter plaza de un residente en un determinado día.

GetTipologia(IDResidente, fecha, conexion)

Al igual que el caso anterior, también se ha creado una función que fácilmente devuelva la tipología de un residente (atributo variable) para una fecha dada, evitando así escribir muchas veces complicadas consultas SQL dentro de la obtención de los datos de la tabla de hechos.

Con la tabla de resultados ya rellena, tan solo falta pasarla al unificador - rellenador para que éste haga su tarea.

4.4.5.- Assembly Unificador – Rellenador

El assembly unificador – rellenador es, sin lugar a dudas, el assembly con mayor complejidad computacional de todos los enunciados hasta ahora. El mismo se encarga de unificar todos los datasets de resultados obtenidos de ejecutar los distintos assemblies en los orígenes de datos (bases

de datos de cada una de las residencias), que éstos tengan una estructura homogénea y cargarlos en el destino (almacén de datos).

A continuación, se detallará un poco más el funcionamiento del mismo mediante sus principales funciones:

TodosResultadosRecibidos(lintIDEstadisticaLanzada)

Es una simple función que hace que el assembly espere a que todos los demás assemblies se ejecuten en sus orígenes de datos, envíen sus datasets de resultado y comenzar a realizar las tareas principales del assembly unificador – rellenador.

UnificarDatasetsResultados(lintIDEstadisticaLanzada)

Esta función extraerá todos los registros de los datasets enviados desde los distintos orígenes y los convierte en una sola tabla. Por ello es de vital importancia que la función anterior espere a la ejecución de los assemblies de dimensiones y hechos en cada uno de sus orígenes, ya que si no podrían perderse registros de algún origen.

A partir de aquí, en función de si el destino de los datos es una tabla de dimensiones o de hechos, se realizarán distintas funciones. En el caso de que la tabla destino sea una tabla de dimensiones se realiza la siguiente función:

RellenarDimension(resultadoUnificado)

Ya que los datos obtenidos de ejecutar un assembly de dimensión no requieren de una posterior transformación, esta función lo único que realiza es insertar los registros contenidos en la tabla resultadoUnificado en la correspondiente tabla de dimensiones.

Sin embargo, si la tabla destino es una tabla de hechos, las funciones a realizar serán las siguientes:

CrearTablaAuxiliar(resultadoUnificado)

Esta función crea una réplica de la tabla de hechos del almacén para hacer un transvase de información entre la tabla con la unión de los resultados (se recuerda que los registros tendrán los códigos unificados de todas las dimensiones y las medidas asociadas) a una tabla auxiliar (que ya no contendrá los códigos unificados, sino los identificadores propios del almacén).

convertUnificadoToId(resultadoUnificado, auxiliarResultado)

La tarea de esta función es realizar un volcado de todos los registros contenidos en la tabla resultadoUnificado en una nueva tabla llamada auxiliarResultado, pero convirtiendo los códigos unificados de todos y cada uno de los registros en identificadores propios del almacén. Esta es quizás la tarea más costosa de todo el assembly, ya que deberá realizar una consulta por cada campo de dimensión de cada uno de los registros, lo que supone una gran carga computacional.

Por ejemplo, si se contase con una tabla de resultados con 20 000 registros, y esta tuviese cinco dimensiones (una cifra para nada elevada), se tendrían que realizar la friolera de 100 000 consultas SQL.

RellenarHecho(auxiliarResultado)

Al igual que la función RellenarDimension, esta función lo único que realiza es introducir todos y cada uno de los registros de la tabla auxiliarResultado en la tabla de hechos correspondiente.

4.4.6.-Assembly Rellenador de Dimensiones Genéricas

Cabe destacar que, además de las dimensiones que hemos estudiado anteriormente, hay otras muchas cuyos registros no cambian (sólo tienen un espectro de valores posibles bastante reducido).

En tal caso, no hace falta la creación de un complicado assembly como los vistos anteriormente para el relleno de dicha dimensión: mediante una sencilla transacción de SQL se pueden introducir dichos valores en la tabla correspondiente sin tener que acudir a la base de datos origen al objeto de adquirir los valores (ya que estos son conocidos).

Por ello, se llegó a la conclusión de que crear un assembly extra para realizar dichas transacciones SQL seria una simplificación para el programador: el denominado assembly "RellenaDimensiones".

Este assembly es una sucesión de transacciones SQL para cada una de las dimensiones predefinidas (bastante numerosas si desplegamos una gran variedad de indicadores). Para ello, no hará más que crear una conexión al almacén de datos destino y ejecutar contra el mismo las distintas transacciones.

A modo de ejemplo, se adjunta la transacción realizada para rellenar la tabla "Dim_EstadoResidente", la cual tan solo tiene seis valores posibles:

Código 3.- Transacción SQL de Relleno de la Dimensión Estado del Residente

```
if  not exists (Select * From Dim_EstadoResidente)
begin
DBCC CHECKIDENT ('Dim_EstadoResidente' ,RESEED,1)
INSERT INTO Dim_EstadoResidente (EstadoResidente) VALUES ('Baja Voluntaria')
INSERT INTO Dim_EstadoResidente (EstadoResidente) VALUES ('Expulsado')
INSERT INTO Dim_EstadoResidente (EstadoResidente) VALUES ('Fallecido')
INSERT INTO Dim_EstadoResidente (EstadoResidente) VALUES ('Hospital')
INSERT INTO Dim_EstadoResidente (EstadoResidente) VALUES ('Residencia')
INSERT INTO Dim_EstadoResidente (EstadoResidente) VALUES ('Ausente')
End
```

De esta forma, el programador ya conocerá a priori los códigos para cada uno de estos casos, pudiendo aventurarse a introducir los valores directamente sin necesidad de que el unificador – rellenador deba realizar ningún trabajo.

4.5.- Desarrollo por el Método de Paquetes de Integration Services

El desarrollo por el método de paquetes de Integration Services es muy distinto al explicado anteriormente mediante assemblies. Junto a Resiplus BE, los assemblies tienen la capacidad de realizar la función de ETL, mediante consultas SQL y código desarrollado por el programador, pero con una potencia bastante limitada.

Sin embargo, los paquetes de Integration Services si ofrecen funciones específicas y optimizadas de ETL al objeto de que el desarrollo de estos sea mucho más sencillo para el programador. Todo esto, unido a la interfaz drag and drop de la utilidad, hacen que el desarrollo de un indicador sea mucho más amigable para el programador mediante esta modalidad.

Para ello, se comenzará explicando grosso modo cuáles son los componentes principales de un paquete de Integration Services, para a continuación pasar a explicar todas aquellas operaciones y tareas que se pueden realizar dentro del flujo de control o flujo de datos.

Una vez conocidas todas las utilidades que nos ofrece un paquete de Integration Services, se procederá a trazar una estrategia mediante la cual abordar el desarrollo de indicadores de gestión con este tipo de paquetes.

4.5.1.- Componentes de un Paquete de Integration Services

A diferencia de un assembly, un paquete de Integration Services es una aplicación específica para realizar diversas operaciones sobre bases de datos. Estas operaciones pueden ser realizar copias de seguridad, restaurar, trasladar datos de un sitio a otro, procesar cubos de Analysis Services… un amplio abanico de posibilidades que pueden ser realizadas con la mera ejecución de un paquete.

Estos paquetes deben ser desarrollados dentro de Visual Studio versión 2005 o superior (Visual Studio 2008 en el ejemplo), ofreciendo este una interfaz muy atractiva que permitirá al usuario crear un paquete con las tareas específicas que este quiera realizar. La Figura 23 ejemplifica la interfaz de un proyecto de Integration Services.

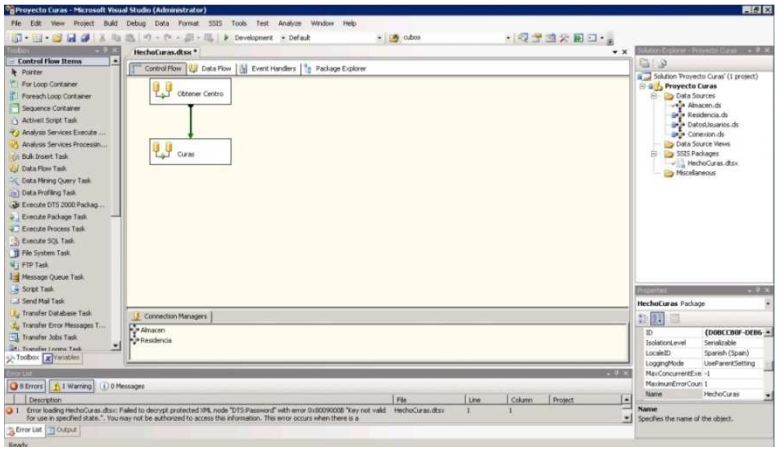


Figura 23.- Captura de un proyecto de Integration Services

En ella, se pueden observar partes típicas de cualquier entorno de desarrollo visual: la caja de herramientas, el explorador de la solución, la barra de salida o la barra de propiedades del objeto seleccionado, además de otras menos usuales que están redondeadas y señalizadas al objeto de estudiarlas, ya que estas son las partes especificas de las que goza un proyecto de Integration Services. Para ello, se inserta la Figura 24 con la zona de trabajo ampliada, indizando cada parte especifica del área de trabajo con una letra, cada una con su explicación posterior.

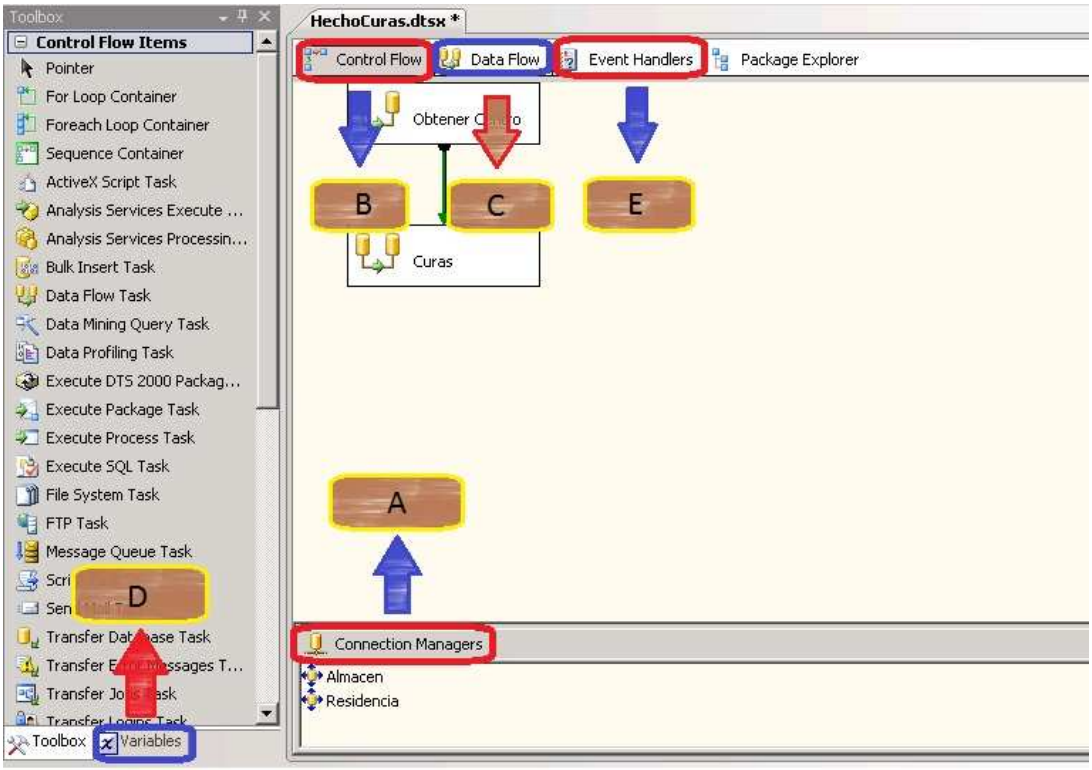


Figura 24.- El área de trabajo de un proyecto de Integration Services

a.- Conexiones

Imprescindibles a la hora de comunicarse con las bases de datos y realizar las operaciones oportunas sobre las mismas. Las conexiones que se pueden crear pueden ser de OLE DB, ADO.NET, a partir de un fichero plano, de Analysis Services, etc. una gran variedad existente para poder seleccionar el usuario aquella que satisfaga sus necesidades.

En los casos que se estudiarán en la memoria, las conexiones son de tipo OLE DB, ofreciéndose para la creación de las conexiones la interfaz de la Figura 25.

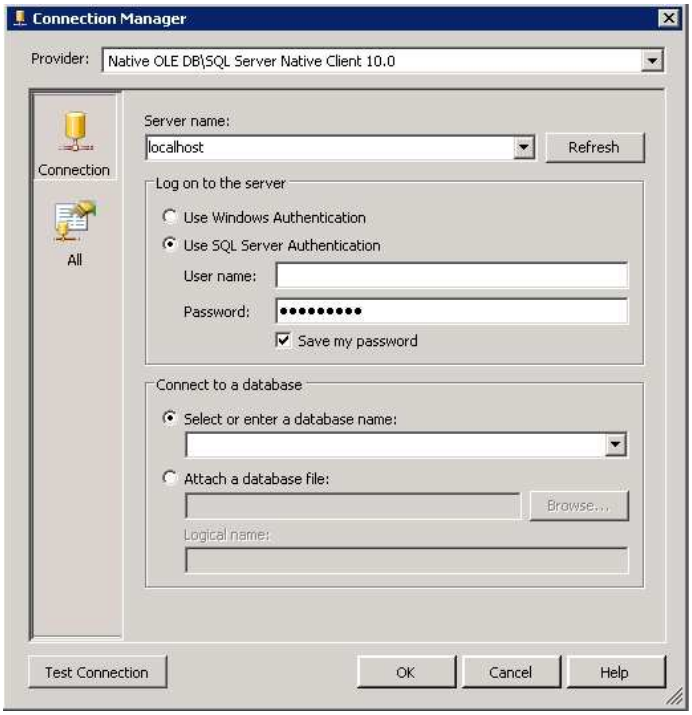


Figura 25.- Interfaz de conexión del tipo OLE DB

Pudiéndose seleccionar el proveedor de bases de datos, el servidor de bases de datos, el tipo de autenticación a utilizar (de Windows o de SQL, especificando usuario y contraseña), y la base de datos a la que se realiza la conexión. Si se desean especificar más detalles sobre la conexión, tan solo habrá que ir a la pestaña "All", donde se podrán ver detalles adicionales sobre la misma.

También existe un útil botón de testeo de la conexión, para comprobar que los datos introducidos son correctos y la conexión se realiza correctamente

b.- Flujo de Control

En el flujo de control se especifica cuál es el orden de realización de las distintas operaciones que se realizan dentro de un paquete de Integration Services. Dichas operaciones engloban a un gran conjunto de tareas: realización de copias de seguridad, restauración de las mismas, consultas de minería de datos, etc. y sobre todo, las tareas de flujo de datos, que merecen un tratamiento separado a este apartado, ya que se puede decir que son una subtarea dentro del flujo principal.

Cabe destacar que se puede crear un orden de ejecución entre dichas tareas, al objeto de que estas se ejecuten ordenadamente y evitar errores de precedencia.

En puntos posteriores, se explicaran aquellos elementos que pueden formar parte de un flujo de control, todos ellos visibles en caja de herramientas y "arrastrables" dentro del área de flujo de control. Cabe destacar que un paquete de Integration Services tan solo puede tener un flujo de control.

c.- Flujos de Datos

Un flujo de datos permite al usuario realizar operaciones de ETL entre una o varias bases de datos. Dentro del flujo de datos solo podrán realizarse operaciones de este tipo, entre las cuales están la extracción, la ordenación, la unión o la concatenación de datos.

Al igual que ocurría con los flujos de control, también se puede crear un orden de precedencia entre las distintas tareas de un flujo de datos.

En puntos posteriores se podrán observar las distintas operaciones de ETL que se pueden realizar. Cabe destacar que un mismo paquete de Integration Services puede poseer más de un flujo de datos.

d.- Variables

Las variables son una parte importante de un paquete de Integration Services, ya que permiten al programador el paso de valores entre distintos elementos de flujo que no están correlacionados.

Además, las variables también sirven para introducir parámetros externos y propios de la ejecución de un paquete de Integration Services, ya que el valor de las mismas puede ser especificado justo el momento antes de la ejecución del paquete.

e.- Eventos

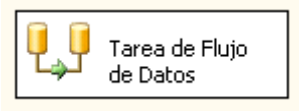
Un paquete de Integration Services puede tener acciones asociados a distintos eventos que se producen durante la ejecución del mismo, como por ejemplo el fallo de la ejecución del mismo, la ejecución con advertencias, etc.

Las acciones que se pueden realizar cuando se produzca dicho evento pueden ser dos: la ejecución de otro paquete de Integration Services o la ejecución de un programa externo.

4.5.2.- Operaciones y Tareas de Flujo de Control

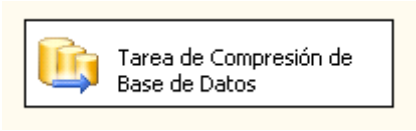
A pesar de existir una gran cantidad de tareas, aquí solo se adjuntarán las más útiles al objeto de ilustrar su funcionamiento al lector de la memoria, ya que las verdaderamente interesantes para el cometido de la realización de los siguientes puntos son las operaciones de las tareas de flujo de datos.

Tarea de Flujo de Datos



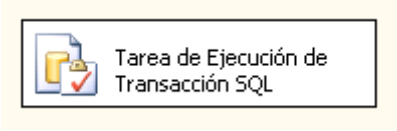
Como ya se comentó anteriormente, la tarea de flujo de datos permite al programador realizar operaciones de ETL sobre una o más base de datos. Las tareas de este tipo tendrán una interfaz específica para adjuntar dentro de la misma diversas operaciones de este tipo.

Tarea de Compresión de Base de Datos



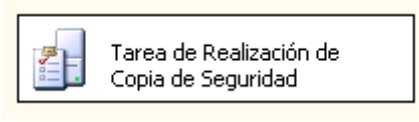
Esta tarea permitirá realizar a una base de datos la popular orden "SHRINK". Para ello, se debe especificar una base de datos mediante una conexión. Además se pueden especificar distintos parámetros, como solo realizar la operación si el archivo de la base de datos supera un determinado tamaño, o que hacer con el espacio liberado

Tarea de Ejecución de Transacción SQL



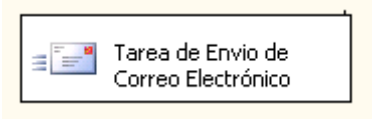
También se pueden realizar distintas transacciones contra una base de datos utilizando esta tarea, en la cual se deberá adjuntar la transacción SQL a realizar además de la conexión a la base de datos contra la que se va a ejecutar dicha transacción. Además, como parámetro adicional se puede especificar un "timeout" para la transacción.

Tarea de Realización de Copia de Seguridad



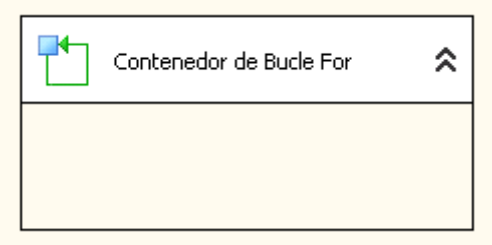
Con esta tarea se pueden realizar copias de seguridad de bases de datos, especificando para ello la conexión a la base de datos de la cual se desea realizar la copia de seguridad, y opciones tales como la forma de la copia (archivo u otra base de datos), expiración de la misma y otras muchas opciones interesantes.

Tarea de Envío de Correo Electrónico



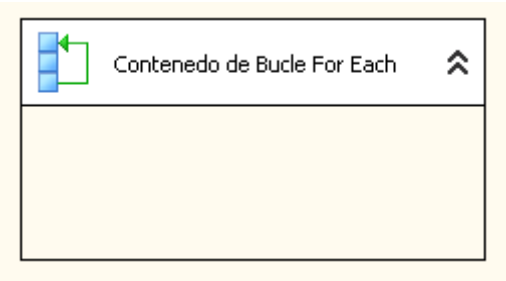
También existe una tarea mediante la cual mandar uno o más correos electrónicos a distintos usuarios, mediante la especificación de una conexión a un servidor de correo, adjuntando posteriormente el título y el contenido del mismo.

Contenedor de Bucle "For"



A pesar de no ser una tarea en sí, este contenedor nos ofrece la posibilidad de ejecutar un conjunto de tareas un determinado número de veces, al igual que haría una instrucción "For" de cualquier lenguaje de programación. Para ello, no hay más que arrastrar las tareas dentro de este contenedor.

Contenedor de Bucle "For Each"

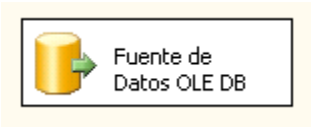


Permite la posibilidad de ejecutar un conjunto de tareas para un conjunto determinado de expresiones, que serán introducidos dentro de las opciones del contenedor. Para ello, y al igual que ocurría con el anterior contenedor, no hay más que arrastrar las tareas dentro de este contenedor.

4.5.3.- Operaciones y Tareas de Flujo de Datos

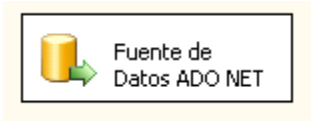
Mediante la creación de flujos de datos se podrán crear distintas operaciones de ETL. Aquí se mostrarán un subconjunto de las operaciones que se pueden realizar dentro de una tarea de flujo de datos.

Fuente de Datos OLE DB



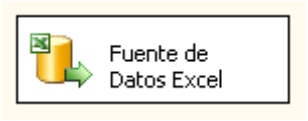
Esta operación es considerada de extracción de datos y mediante la especificación de una conexión de base de datos, se puede realizar una consulta SQL al objeto de extraer de una tabla los datos deseados. Por si el programador no es muy ducho en lenguaje SQL, también se ofrece una interfaz mediante la cual especificar los datos que se deben extraer de la tabla correspondiente.

Fuente de Datos ADO NET



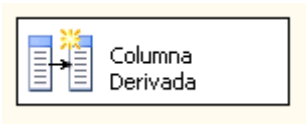
Con esta operación de extracción de datos se podrán adquirir un conjunto de datos a partir de una conexión de ADO NET.

Fuente de Datos Excel



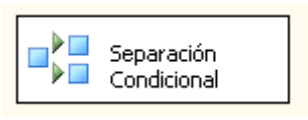
Esta operación de extracción permite la adquisición de datos a partir de un fichero Excel. Para ello habrá que especificar el fichero de Excel en cuestión, además del conjunto de filas y columnas a adquirir.

Columna Derivada



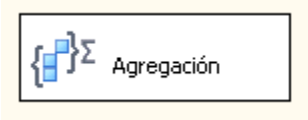
Mediante esta operación de transformación se podrá agregar una nueva columna a los datos adquiridos anteriormente, siendo el valor de esta un valor fijo, o un valor derivado a partir de un valor de cualquier otra columna.

Separación Condicional



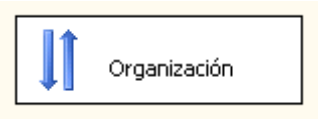
Con esta operación de transformación se podrá separar un conjunto de datos en dos o más nuevos conjuntos de datos a partir de una condición. Los conjuntos de datos resultantes podrán ser tratados a partir de este punto de manera independiente, pudiendo así hacer las transformaciones necesarias a cada uno de ellos.

Agregación



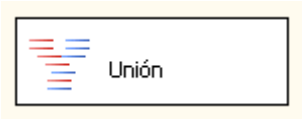
Esta operación de transformación permite agrupar en un mismo registro todos aquellos registros repetidos dentro de nuestro conjunto de datos. Para ello, tan solo hay que indicar las columnas por las cuales se desea realizar la agregación.

Organización



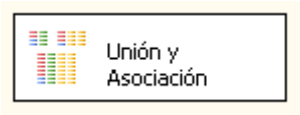
No es una operación de transformación pura, ya que tan solo permite realizar una organización del conjunto de datos a partir de una o varias columnas existentes en el mismo.

Unión



Esta operación de transformación permite la unión de dos o más conjuntos de datos en uno solo, sirviendo esto, por ejemplo, para volver a unir dos conjuntos de datos que anteriormente habían sido separados mediante una separación condicional.

Unión y Asociación

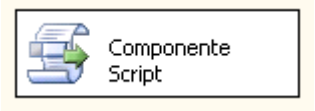


Esta quizás sea la operación de transformación más útil de todo el conjunto. Con ella, se permite unir dos conjuntos de datos en uno solo mediante la asociación de campos de los dos conjuntos. En realidad, lo que realiza esta operación no es más que lo que realizan las instrucciones “INNER JOIN”, “LEFT JOIN” y “RIGHT JOIN” de SQL, pero pudiendo realizar dicha unión de una manera mucho más visual e intuitiva.

Conversión de Datos

Esta útil operación de transformación permite convertir el tipo de datos de una columna a otro tipo de datos que convenga, habiendo un conjunto de casos en los que la transformación sea imposible (por ejemplo, podremos pasar de un entero a un entero largo, pero no pasar de una cadena de texto a un entero).

Componente Script



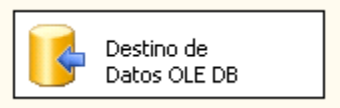
Mediante esta operación de transformación se pueden pasar todos los datos de un conjunto por código Basic o C#, de modo que se podrán hacer las modificaciones oportunas a dicho conjunto de

datos. Esta operación es de extrema utilidad, por ejemplo, cuando queremos realizar la derivación de un conjunto de registros. Además, brinda la posibilidad del uso de variables del paquete, las cuales deberán ser introducidas en las propiedades "ReadOnlyVariables" y "ReadWriteVariables", dependiendo si la variable a utilizar es de solo lectura o de lectura y escritura.

Para la programación de un componente script se debe hacer uso de unas funciones que ya vienen predefinidas. Dichas funciones son:

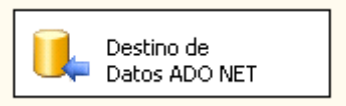
- *NombreInput_ProcessInputRow(filadelInput)*: realiza las operaciones que hay contenidas en esta función para todas y cada uno de los registros del input.
- *PreExecute()*: realiza las operaciones contenidas en la función antes de procesar fila alguna.
- *PostExecute()*: realiza las operaciones contenidas en esta función una vez ya se han procesado todos los registros del input.

Destino de Datos OLE DB



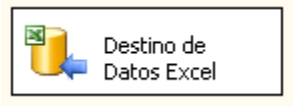
Con esta operación carga se podrá introducir un conjunto de datos dentro de una base de datos mediante una conexión OLE DB. Para ello, tan solo hay que indicar la correspondencia entre los datos existentes en el conjunto de datos y la estructura de la tabla en la cual se quieren cargar dichos datos.

Destino de Datos ADO NET



Otra operación de carga, que realizará la misma tarea que la anterior, pero esta vez utilizando para ello una conexión de tipo ADO NET.

Destino de Datos Excel



Esta última operación de carga permitirá realizar la exportación de los datos adquiridos y transformados mediante nuestro flujo, siendo el destino de estos un archivo de Excel, pudiendo ser éste un archivo ya existente o un nuevo archivo.

4.5.4.- Estrategia de Implementación con Integration Services

Al igual que se hizo con el método de implementación mediante assemblies, y una vez conocidas todas las operaciones que se pueden realizar con Integration Services, llega el momento de trazar una estrategia de implementación propia para este método.

Teniendo en cuenta que con Integration Services se puede crear más de un flujo de datos dentro de un paquete, se pueden hacer ligeras modificaciones con respecto a la estrategia utilizada en los assemblies.

En primer lugar se pensó que sería bastante útil el tener todas las operaciones dentro de un mismo paquete. Esta idea fue rechazada ya que, a pesar de ser posible debido a que se pueden añadir distintos flujos de datos cada uno con un orden específico, a la hora de acometer la revisión de posibles fallos durante la ejecución, sería mucho más complicado que tener una disgregación en flujos más pequeños.

Por ello, se recapacitó y se propuso una estrategia idéntica a la que se seguía con los assemblies pero adaptada a Integration Services: la creación de un paquete específico para la creación del almacén, un paquete por cada una de las dimensiones del proyecto y un paquete por cada uno de los hechos.

Este modelo tiene la ventaja de que, en el momento en el que se produce un fallo, éste está completamente localizable para el programador, que lo podrá revisar y reparar sin tener que perder tiempo en su localización.

Pero por otro lado, tiene la contrapartida de que el número de paquetes a controlar por parte del mismo es inmenso, y a la hora de su ejecución esto puede ser una tarea más bien engorrosa.

Por ello, se optó por una solución compromiso entre las dos soluciones anteriormente enunciadas. Teniendo en cuenta que las operaciones que se realizan para la ETL de una dimensión son poco pesadas, se optó por introducir todos los flujos de datos de todas las dimensiones del proyecto en un mismo paquete de Integration Services.

De este modo, la implementación debe seguir el siguiente patrón: un paquete con el objetivo de crear el almacén de datos, otro paquete para la carga de todas las dimensiones del almacén, y un paquete por cada una de las tablas de hechos a rellenar. Como bien habrá intuido el lector de la presente, el orden de ejecución de los paquetes será el mismo que el de ejecución de los assemblies anteriormente explicados, y adjuntado también en la Figura 26.

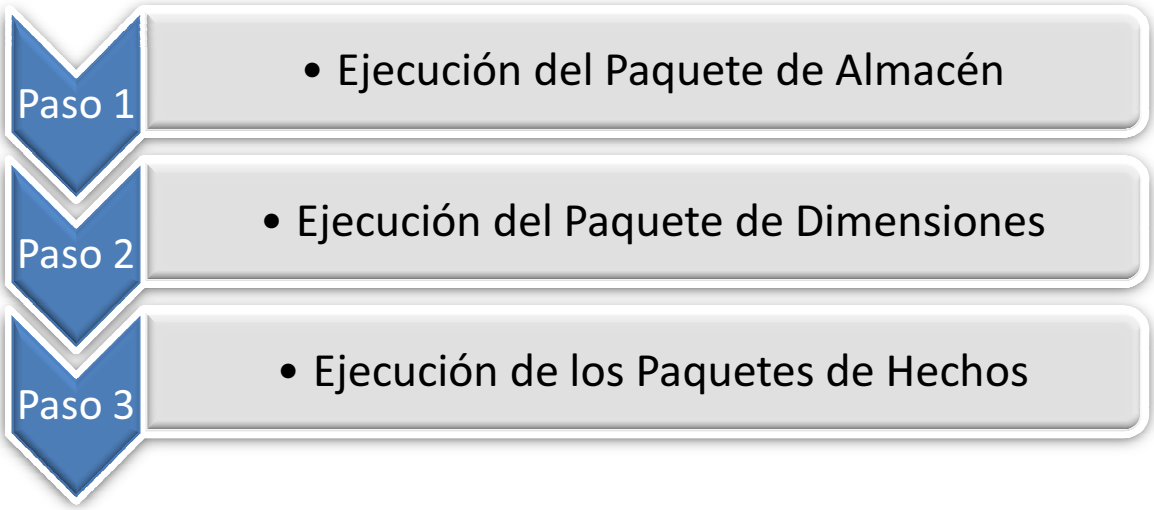


Figura 26.- Orden de ejecución de paquetes de Integration Services

Además, hay que tener en cuenta el relleno de aquellas dimensiones que tienen un espectro de valores que no puede variar. Para estos casos, se decidió crear dichos registros dentro de la creación del almacén, a efectos de repartir la carga computacional entre los paquetes de almacén y de dimensiones (ya que es imposible que la inserción de dichos registros en el momento de crear el hecho por posibles dependencias existentes).

Por los mismos motivos, se ha optado por introducir también dentro del paquete de creación del almacén la consulta que crea los registros correspondientes a la dimensión de tiempo, ya que puede ser considerada como una dimensión semi-genérica (ya que los datos son siempre los mismos y conocidos a priori, introduciendo en la tabla de dimensiones el espectro de fechas que interese).

A lo largo de la memoria no se ha dicho cual sería el modo de ejecución de estos paquetes de Integration Services. Dichos paquetes pueden ser fácilmente ejecutados realizando doble clic sobre ellos una vez están implementados, teniendo que realizar la farragosa tarea de cambiar las conexiones dependiendo de la residencia contra la cual va a ser ejecutado el paquetes. Esta tarea puede parecer poco costosa, pero si se da el supuesto de que hay doce residencias contra las que lanzar los paquetes, si que supone una enorme cantidad de esfuerzo por parte del usuario

Además, si se analiza la tarea que realizan estos paquetes, sería mucho más cómodo realizar una ejecución de forma automática, desentendiéndose el usuario de tareas tales como cambiar las conexiones, realizando ejecuciones con la frecuencia que decida el usuario.

Para este cometido se implementó una aplicación capaz de ejecutar los paquetes que el usuario creyese conveniente contra los centros que este deseara, evolucionando posteriormente a un servicio de Windows, con el cual el usuario debe despreocuparse de tareas como la modificación de conexiones a la base de datos. Dicha aplicación está completamente detallada en el anexo 2 de la memoria, donde se podrá ver su interfaz y funcionamiento.

Mejora en la Implementación: las Tablas Auxiliares

Una vez implementados los primeros indicadores, se observó que sería de utilidad el tener ciertos datos que son accedidos por muchos indicadores de una manera más accesible de la que los tiene la propia base de datos de Resiplus.

Como ejemplo, se tiene la tipología de la plaza de los residentes. En la tabla donde están introducidas, tan solo esta introducido el día en el que un residente toma una determinada tipología, no existiendo registros para aquellos días en los cuales no hay un cambio de tipología. Pero para su obtención, sería mucho más directo el saber para un determinado día cual es la tipología de un residente, sin tener que ir mirando todos los registros de cambio de tipología de la tabla para saber cuál es la tipología en dicho día.

Por ello, se crea un nuevo concepto denominado tablas auxiliares, que no serán más que tablas creadas en el almacén con conocimiento inferido de la base de datos origen. El motivo de la creación de estas tablas es el hecho de ofrecer una información consultada muy a menudo pero estructurada en la base de datos de Resiplus de tal forma que sea muy difícil su consulta.

Dentro de ellas se incluirá esa información difícil de consultar que, de este modo, quedará almacenada para que el programador tenga un fácil acceso a la misma. En su momento se especificará como es la creación y el relleno de estas tablas.

Dichas tablas auxiliares serán rellenadas en el paquete de dimensión ya que, a pesar de no ser una dimensión pura, se asemeja más a una dimensión que a un hecho.

4.6.- Desarrollo del Indicador de Visitas Familiares

En este apartado de la memoria se estudiará la implementación del indicador de Visitas Familiares, uno de los más demandados por aquellas empresas que poseen Resiplus BE ya que de esta manera pueden tener controlado el flujo de entradas y salidas de familiares de cada residencia.

En primer lugar, se enunciará una definición formal del mismo (describiéndolo y estudiando la estructura de la base de datos a la cual se va a atacar). Luego, se procederá a comentar e ilustrar la estructura de las tablas que vamos a crear dentro de nuestro almacén de datos. Posteriormente, se realizará la descripción de cómo implementarlos tanto en assemblies como en Integration Services.

4.6.1.- Definición del Indicador

A continuación se va a ilustrar en tan solo una frase lo que se desea que realice nuestro indicador mediante una sencilla definición:

“Número total de visitas familiares realizadas a los distintos residentes (adjuntando el carácter de la plaza y la tipología que tienen en el momento en el que ocurre la visita de un familiar), indicando al centro al cual pertenecen y organizadas por día, mes, trimestre o año”.

Una vez realizada la definición, se procede a desgranarla al objeto de observar todos los componentes que existen dentro de la misma:

- Hecho: visitas familiares.
- Medida: número total.
- Dimensiones: residente, tipología, carácter plaza, centro y tiempo.

4.6.2.- Definición de las Tablas del Almacén de Datos

En este punto, se va a ilustrar la estructura de las tablas del almacén de datos encargadas de contener toda la información correspondiente al indicador.

Como bien se apunto en párrafos anteriores, se necesitará de una tabla de hechos a la cual se le llamará “Fact_VisitasFamiliares”, y cinco tablas de dimensiones denominadas “Dim_Centro”, “Dim_Residente”, “Dim_Tiempo”, “Dim_CaracterPlaza” y “Dim_Tipologia”, conteniendo cada una la información que se muestra en la Figura 27.

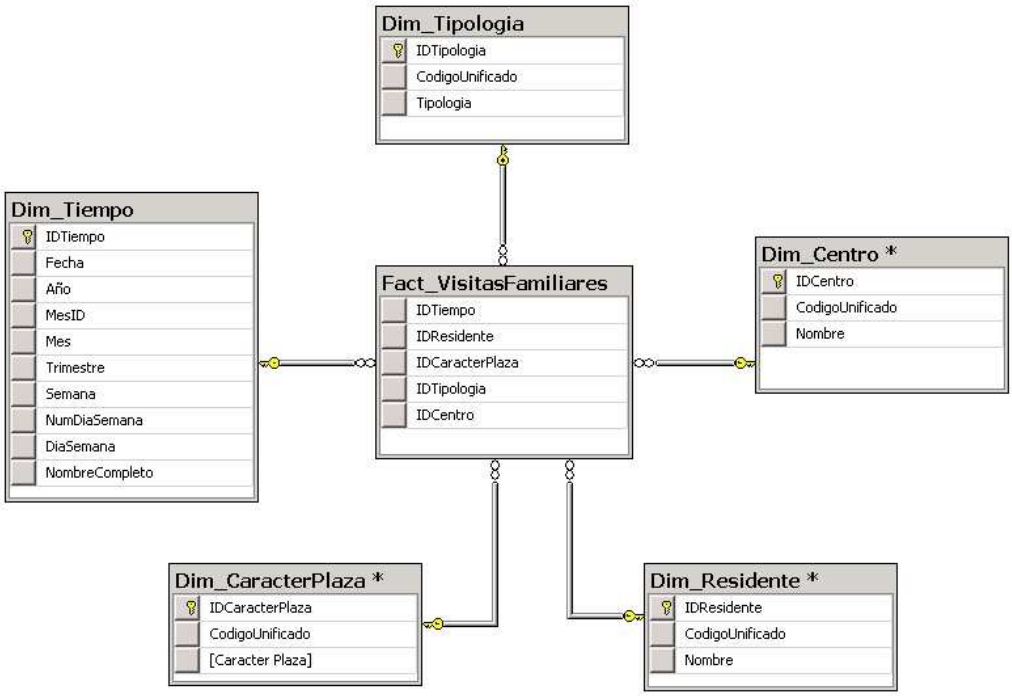


Figura 27.- Estructura de tablas del almacén de datos para el indicador de visitas de familiares

4.6.3.- Implementación del Indicador mediante Assemblies

Como ya se describió con anterioridad, la solución mediante assemblies se estructura en tres pasos: creación de tablas de dimensiones y hechos, relleno de tablas de dimensiones y relleno de tabla de hechos. Por ello, se cree conveniente estudiar separadamente cada una de estas tareas.

Modificación del Assembly Almacén para Creación de Tablas

Para comenzar, se deberá modificar el assembly almacén de modo que este cree las tablas correspondientes al indicador, siendo estas en nuestro caso las tablas de dimensión de centro, residente, carácter plaza, tipologia y tiempo y la tabla de hechos de visitas familiares.

A continuación, se mostrará el código de la función necesaria para la creación de la tabla de dimensión "Dim_Centro", y para la tabla de hechos "Fact_VisitasFamiliares", al objeto de tener ejemplificada la creación de una tabla de cada tipo, y ver como se crean las relaciones existentes entre dichas tablas en el almacén.

Código 4.- Creación de la tabla "Dim_Centro"

```
static void createTableDimCentro(string tabla)
{
    ArrayList atributos = new ArrayList();
    ArrayList indicesUnicos = new ArrayList();
    ArrayList indicesNoUnicos = new ArrayList();

    atributos.Add("IDCentro int IDENTITY(1,1) PRIMARY KEY");
    atributos.Add("CodigoUnificado nvarchar(19)");
    atributos.Add("Nombre varchar(40)");

    createTable(tabla, atributos);

    #region indexesUniques

    indicesUnicos.Add("CodigoUnificado");

    createIndexUnique(tabla, indicesUnicos);

    #endregion

    #region indexesNonUniques

    indicesNoUnicos.Add("Nombre");

    createIndexNonUnique(tabla, indicesNoUnicos);

    #endregion
}
```

Código 5.- Creación de la tabla "Fact_VisitasFamiliares"

```
static void createTableFactVisitasFamiliares(string tabla)
{
    ArrayList atributos = new ArrayList();
```

```
    ArrayList indicesUnicos = new ArrayList();
    ArrayList indicesNoUnicos = new ArrayList();

    atributos.Add("IDTiempo int NOT NULL REFERENCES
    Dim_Tiempo(IDTiempo)");
    atributos.Add("IDResidente int NOT NULL REFERENCES
    Dim_Residente(IDResidente)");
    atributos.Add("IDCaracterPlaza int REFERENCES
    Dim_CaracterPlaza(IDCaracterPlaza)");
    atributos.Add("IDTipologia int REFERENCES
    Dim_Tipologia(IDTipologia)");
    atributos.Add("IDCentro int NOT NULL REFERENCES
    Dim_Centro(IDCentro)");

    createTable(tabla, atributos);

    #region indexesUniques

    createIndexUnique(tabla, indicesUnicos);

    #endregion

    #region indexesNonUniques

    indicesNoUnicos.Add("IDTiempo");
    indicesNoUnicos.Add("IDResidente");
    indicesNoUnicos.Add("IDCaracterPlaza");
    indicesNoUnicos.Add("IDTipologia");
    indicesNoUnicos.Add("IDCentro");

    createIndexNonUnique(tabla, indicesNoUnicos);

    #endregion
}
```

Las tablas restantes (es decir, “Dim_Tiempo”, “Dim_Residente”, “Dim_CaracterPlaza” y “Dim_Tipologia”), serán creadas de forma homónima a la tabla “Dim_Centro”, cambiando tan solo el nombre de sus campos y el tipo de los mismos.

Creación de los Assemblies de Dimensiones

Si se estudia el esquema de tablas de Resiplus adjuntado anteriormente, se puede deducir que la topología de las diferentes dimensiones es bien distinta en cada caso.

Mientras que los datos de la dimensión residente se extraerán de una tabla que está en la misma base de datos que la tabla en la que se basará el hecho, el código unificado del centro se halla en una base de datos distinta a la del hecho o los datos propiamente dicho, y no encontramos referencia alguna a los datos de la dimensión tiempo.

Por otro lado, hay dos dimensiones las cuales cumplen los parámetros para no ser rellenadas con assemblies, sino mediante el relleno manual de tablas maestras. Dichas dimensiones serian “Dim_CaracterPlaza” y “Dim_Tipologia”, ya que dichos atributos pueden tener un nombre distinto en cada uno de los centros pero en el fondo vienen a ser los mismos valores.

A continuación se separará la creación de cada uno de los assemblies correspondientes a cada una de las dimensiones necesarias para la implementación del indicador.

Creación del Assembly de la Dimensión Residente

En este subapartado se describirá la implementación del assembly de la dimensión residente con motivo de ver todas las tareas que realiza.

En primer lugar, y como ya se explico en puntos anteriores, es de crucial importancia la creación del campo "CodigoUnificado" en la base de datos origen y el relleno del mismo, ya que este será el único enlace existente entre dicha base de datos y la base de datos destino (el único campo por el que luego podrán ser relacionados)

Una vez hecho esto, ya está todo preparado para traer los registros de la base de datos origen a la base de datos destino. Como los campos necesarios son los enunciados en los requisitos del indicador, la consulta a la base de datos origen quedaría de la siguiente forma:

Codigo 6.- Consulta SQL de la Dimensión Residente

```
SELECT Nombre, CodigoUnificado, Sexo FROM Residentes
```

Dichos registros serán introducidos en una tabla auxiliar, la cual será el output de este assembly y el input del assembly unificador - rellenador al objeto de que este rellene la tabla de dimensión "Dim_Residente".

Creación del Assembly de la Dimensión Centro

Como ya se ha comentado a lo largo de este punto, la creación del assembly de la dimensión centro dista un poco de la creación de un assembly de dimensión genérico como podría ser el de residente, ya que los códigos unificados de los centros no están almacenados en la misma base de datos donde están almacenados los registros origen de las tablas de dimensiones o hechos, sino en una externa denominada "DatosUsuarios".

Dicha base de datos posee una tabla denominada "BDs", dentro de la cual están registradas todas y cada una de las residencias pertenecientes al grupo creado con Resiplus, teniendo cada una un valor distinto para el campo "IDBD", el cual hará la función de código unificado en esta dimensión.

Por ello, la primera consulta a realizar sería la siguiente:

Código 7.- Primer Consulta SQL de la Dimensión Centro

```
SELECT IDBD AS CodigoUnificado FROM BDs WHERE RutaFicheroServidor LIKE '%" &
lstrDatabase & "'
```

Donde "lstrDatabase" es la ruta de la base de datos en el servidor de SQL, variable la cual esta almacenada dentro de Resiplus BE, y por tanto, puede ser pasada fácilmente al assembly. Cabe destacar que esta consulta tan solo devolverá un registro, ya que cada base de datos tendrá una ruta distinta y de esta forma se conseguirá el "IDBD" correspondiente a la base de datos que se está atacando en este momento.

La segunda consulta será la mostrada a continuación:

Código 8.- Segunda Consulta SQL de la Dimensión Centro

```
SELECT NombreResidencia AS Nombre, CONVERT(NVARCHAR,'') AS CodigoUnificado
FROM datosResidencia
```

Como en la tabla "DatosResidencia" de cada base de datos tan solo hay un registro, aquí es donde se obtiene el nombre de la residencia a la cual está atacando el assembly. Se puede también observar como el campo "CodigoUnificado" de dicha tabla es un valor vacio: esto es debido a que posteriormente, dicho campo cogerá el valor extraído en la consulta anteriormente adjuntada.

Con estas dos consultas ya se tiene tanto el nombre del centro o residencia como su código unificado, y tan solo quedara por insertar el registro compuesto por estos datos dentro de la tabla de dimensión correspondiente.

Creación del Assembly de la Dimensión Tiempo

La dimensión tiempo quizás sea la más particular de todas, ya que los datos de dicha dimensión no están almacenados en ninguna tabla de la base de datos. Pero si se piensa detenidamente, ¿por qué debieran ser almacenados?

Al contrario que las dimensiones residente o centro, los datos de la dimensión tiempo se van a conocer sin necesidad alguna de recurrir a las bases de datos, ya que el año está compuesto de 365 días (366 los años bisiestos) que se pueden generar sin la necesidad de recurrir a la base de datos.

Por ello, tan solo realizando la siguiente transacción SQL se obtendrán los registros necesarios dentro del almacén de datos:

Código 9.- Transacción SQL de Relleno de la Dimensión Tiempo

```
SET LANGUAGE spanish
SET DATEFIRST 1
DECLARE @fi DATETIME, @ff DATETIME
SET @fi = '" & fechaInicio & "'
SET @ff = '" & fechaFin & "'
WHILE @fi <= @ff
BEGIN
       INSERT INTO Dim_Tiempo (IDTiempo, Fecha, Año, MesID, Mes,
       Trimestre, Semana, NumDiASemana, DiASemana)
       SELECT YEAR(@fi)*10000+MONTH(@fi)*100+DAY(@fi) AS IDTiempo,
       @fi AS Fecha,
       YEAR(@fi) AS Año,
       MONTH(@fi) AS MesID,
       DATENAME(mm, @fi) AS Mes,
       DATEPART(q, @fi) AS Trimestre,
       DATENAME(ww, @fi) AS Semana,
       DATEPART(dw, @fi) AS NumDiASemana,
       DATENAME(dw, @fi) AS DiASemana
       SET @fi = @fi + 1
END
```

Creación del Assembly de Hecho de Visitas Familiares

Al igual que se ha hecho con los demás assemblies, en este subapartado se estudiará que modificaciones o añadidos hay que realizar en el assembly base de hechos al objeto de conseguir el indicador de visitas familiares.

En este caso, los cambios a realizar son tan solo dos: modificar la consulta de adquisición de datos y la tabla auxiliar donde se almacenarán los registros de dicha consulta, siendo esta tabla suministrada posteriormente al assembly unificador – rellenador.

En primer lugar, la consulta con la cual se conseguirán los registros será la siguiente:

Código 10.- Consulta SQL del Hecho Visitas Familiares

```
SELECT d.idresidente AS IdResidente, b.CodigoUnificado AS CodigoUnificado,
d.fechahoraentrada AS Tiempo
FROM VisitasFamiliares AS d
INNER JOIN (SELECT idresidente, CodigoUnificado FROM Residentes) AS b
ON d.idresidente=b.idresidente
```

Mientras que la tabla auxiliar a crear debe seguir la siguiente estructura:

Visitas Familiares (Tabla Auxiliar)

Nombre del Campo	Tipo de Campo
ResidenteUnificado	String
CaracterPlazaUnificado	String
TipologiaUnificado	String
CentroUnificado	String
TiempoUnificado	DateTime

Como se puede observar, con la consulta incluida tan solo se obtiene el residente, la fecha de la visita familiar, e implícitamente el centro (ya que el assembly se estará lanzando sobre una residencia).

Para la obtención del carácter plaza y la tipología del residente, y como ya se comento en puntos anteriores, tan solo hay que ejecutar dentro del assembly de hechos las funciones “GetCaracterPlaza()” y “GetTipologiaPlaza()” pasando como argumentos el residente y la fecha en la que se produjo la visita. Con todo esto, ya se podrán introducir todos los datos obtenidos de consulta y función dentro de la tabla auxiliar mediante el unificador – rellenador.

Resumen de Exportaciones a Crear en Resiplus BE

Una vez creados los assemblies, tan solo queda agregarlos a Resiplus BE y crear las exportaciones necesarias para su ejecución.

A continuación se enuncian las distintas exportaciones que se deben crear al objeto de rellenar el almacén de datos con la información del indicador de visitas de familiares:

Creador Almacén		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Almacen.dll	En el unificador	Antes de enviar la estadística

Dimensión Tiempo		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Dim_Tiempo.dll	En el unificador	Antes de enviar la estadística

Dimensión Residente		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Dim_Residente.dll	En cada centro	Antes de enviar la estadística
UnificadorRellenador.dll	En el unificador	Después de la estadística

Dimensión Centro		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Dim_Centro.dll	En cada centro	Antes de enviar la estadística
UnificadorRellenador.dll	En el unificador	Después de la estadística
Hecho Visitas Familiares		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Fact_VisitasFamiliares.dll	En cada centro	Antes de enviar la estadística
UnificadorRellenador.dll	En el unificador	Después de la estadística

4.6.4.- Implementación del Indicador mediante Integration Services

Una vez vista la implementación del indicador de visitas de familiares mediante el método de assemblies, se va a proseguir especificando cual sería la implementación vista desde el método de Integration Services.

Como se comentó en el punto 4.4 de la memoria, la distribución elegida a la hora de realizar las distintas tareas en paquetes para el relleno del almacén de datos será el siguiente:

- Un paquete para la creación del almacén.
- Un paquete para el relleno de todas las dimensiones.
- Un paquete para cada uno de los hechos que se quieran rellenar.

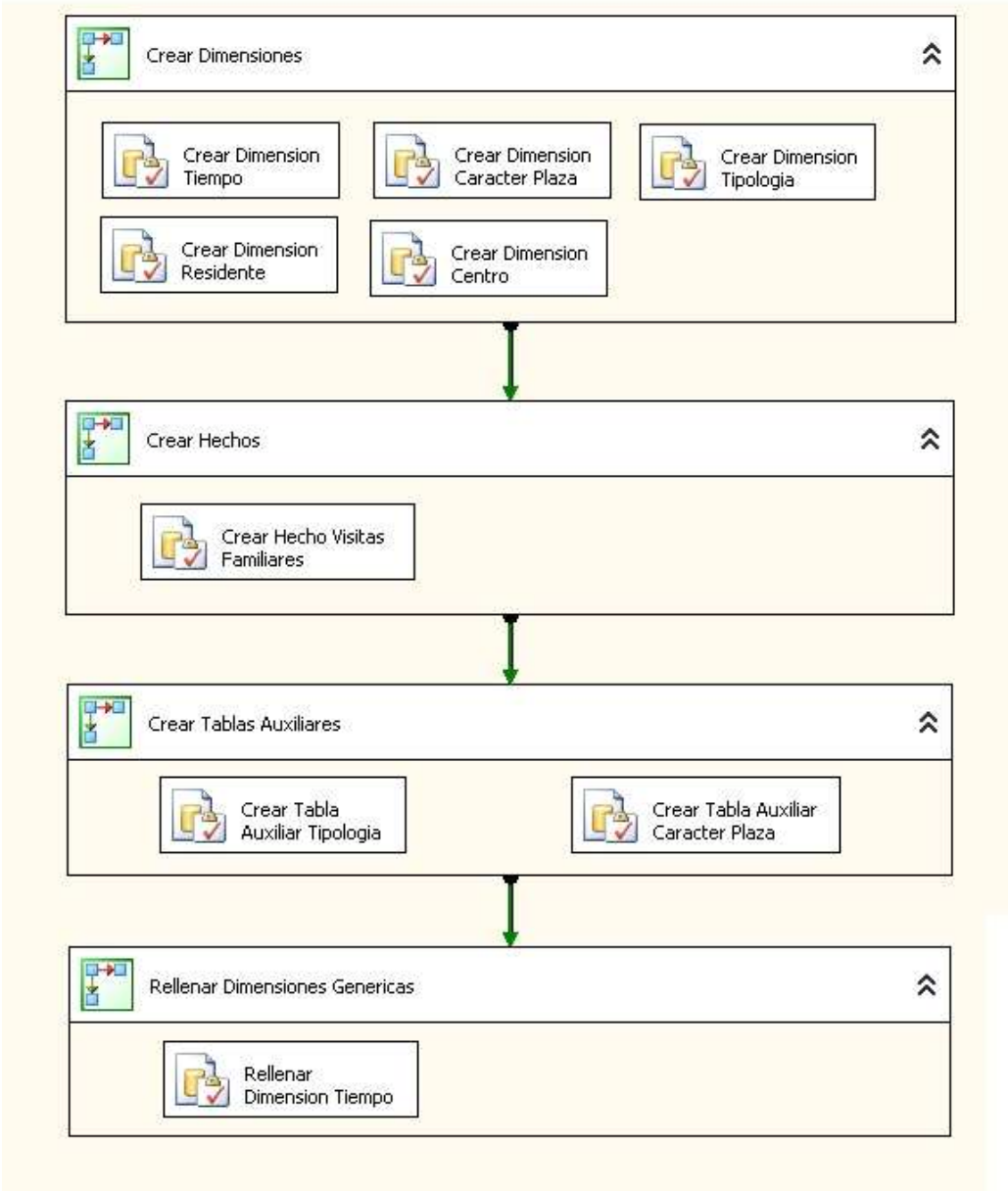
Paquete Creador del Almacén

El paquete de creación del almacén quizás sea el más sencillo de entender por parte del lector, ya que no son más que un conjunto de consultas SQL ejecutadas ordenadamente al objeto de crear las tablas necesarias para posteriormente ser rellenadas por los paquetes de hechos y de dimensión.

Por ello, y aprovechando las utilidades de contenedores que provee Integration Services, se va a estructurar este paquete de una forma bien visual: en primer lugar, y al igual que ocurría con los assemblies, se deberán crear las pertinentes tablas de dimensiones; en segundo lugar, se crearán las

tablas de hechos necesarias, además de hacer las adiciones que se consideraron en el apartado 5 de este capítulo. Para ello, se crearán cuatro contenedores en los cuales se añadirán las transacciones de creación de las tablas de dimensión, las de creación de tablas de hechos, las de creación de tablas auxiliares, y las transacciones de inserción de registros en las dimensiones genéricas, respectivamente.

El resultado visual se muestra a continuación:



A continuación, se va a mostrar la creación de una de las tablas mediante consultas SQL para cada tipo de tablas que se poseen. En primer lugar, se tiene la creación de una tabla de dimensiones, en este caso la dimensión residente.

Código 11.- Transaccion SQL de Creacion de Tabla de Dimension Residente

```
GO
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
if not exists (Select * From information_schema.tables where
table_name='Dim_Residente')
CREATE TABLE [dbo].[Dim_Residente](
 [IDResidente] [int] IDENTITY(1,1) NOT NULL,
 [CodigoUnificado] [nvarchar](120) NULL,
 [Nombre] [nvarchar](125) NULL
CONSTRAINT [PK__Dim_Residente__7C8480AE] PRIMARY KEY CLUSTERED
(
 [IDResidente] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY],
 CONSTRAINT [IX_Dim_Residente_CodigoUnificado] UNIQUE NONCLUSTERED
(
 [CodigoUnificado] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

En segundo lugar, se tiene la creación de una de las tablas de hechos, las cuales se relacionan con las tablas de dimensiones como se puede observar. A continuación se ejemplifica la creación de la tabla de visitas de familiares.

Codigo 12.- Transaccion SQL de Creacion de Tabla de Hechos de Visitas de Familiares

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
if not exists (Select * From information_schema.tables where
table_name='Fact_VisitasFamiliares')
CREATE TABLE [dbo].[Fact_VisitasFamiliares](
 [IDTiempo] [int] NOT NULL,
 [IDResidente] [int] NOT NULL,
 [IDCaracterPlaza] [int] NULL,
 [IDTipologia] [int] NULL,
 [IDCentro] [int] NOT NULL
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Fact_VisitasFamiliares]  WITH CHECK ADD FOREIGN
KEY([IDCaracterPlaza])
REFERENCES [dbo].[Dim_CaracterPlaza] ([IDCaracterPlaza])
GO
ALTER TABLE [dbo].[Fact_VisitasFamiliares]  WITH CHECK ADD FOREIGN KEY([IDCentro])
REFERENCES [dbo].[Dim_Centro] ([IDCentro])
GO
ALTER TABLE [dbo].[Fact_VisitasFamiliares]  WITH CHECK ADD FOREIGN
KEY([IDResidente])
REFERENCES [dbo].[Dim_Residente] ([IDResidente])
GO

ALTER TABLE [dbo].[Fact_VisitasFamiliares]  WITH CHECK ADD FOREIGN KEY([IDTiempo])
REFERENCES [dbo].[Dim_Tiempo] ([IDTiempo])
GO
ALTER TABLE [dbo].[Fact_VisitasFamiliares]  WITH CHECK ADD FOREIGN
KEY([IDTipologia])
REFERENCES [dbo].[Dim_Tipologia] ([IDTipologia])
GO
```

Por último, se crean las tablas auxiliares, creadas al objeto de tener cierta información mucho más accesible de lo que se tenía en las bases de datos de Resiplus. A continuación se adjunta la creación de la tabla auxiliar de carácter de plaza

Codigo 13.- Transaccion SQL de Creacion de Tabla Auxiliar de Carácter Plaza

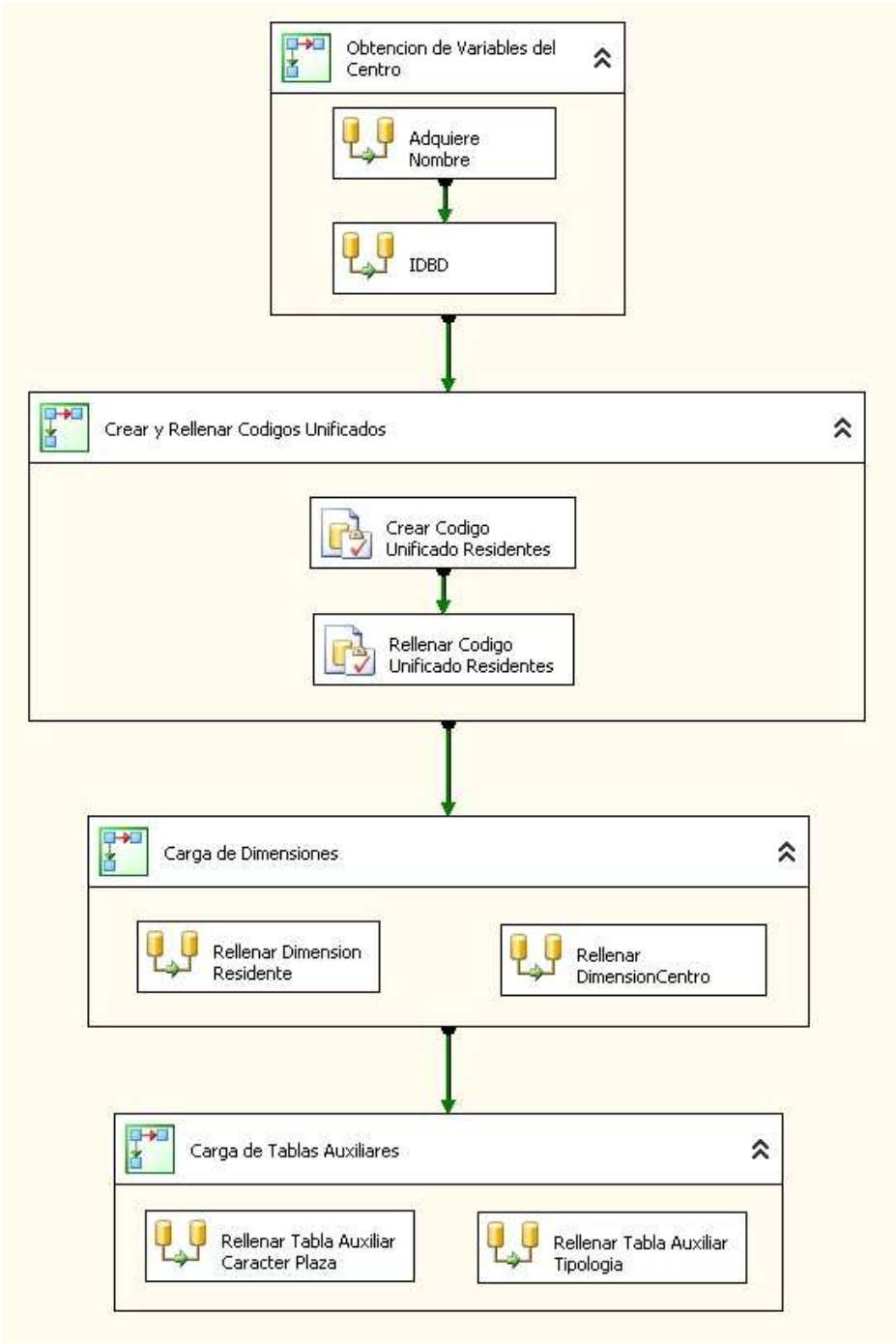
```
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
if not exists (Select * From information_schema.tables where
table_name='CaracterPlaza')
CREATE TABLE [dbo].[CaracterPlaza](
 [IDResidenteAlmacen] [int] NULL,
 [Fecha] [datetime] NULL,
 [IDCaracterPlazaAlmacen] [int] NULL,
) ON [PRIMARY]
```

Además, aquí también se rellenará la tabla correspondiente a la dimensión tiempo. Para ello, lo único necesario es crear una tarea de flujo de control de transacción SQL introduciendo dentro del mismo la transacción ya vista en el assembly de la dimensión tiempo. A parte de esta, no se tiene ninguna otra dimensión a rellenar cuyos registros no difieran a lo largo del tiempo.

Cabe destacar que las conexiones utilizadas en todo momento para la creación de estas tablas corresponden con una conexión al almacén de datos

Paquete de Dimensiones

Como ya se enunció en puntos anteriores, el relleno de las tablas de dimensiones se va a realizar mediante un solo paquete. Además, con la aparición de las tablas auxiliares, también se deberá reparar en su relleno, que también se realizará dentro de de este paquete. A continuación se adjunta el flujo de control correspondiente al paquete:



En primer lugar, se debe obtener el centro al cual está atacando el paquete. Para ello se tienen un par de flujos de datos que ayudan a realizar la tarea. Para ello, se hará uso de variables del paquete, las cuales permiten trasladar datos entre flujos de datos. Las variables que se utilizarán son las siguientes:

Nombre de la Variable	Tipo de la Variable
Nombre	String
CodigoUnificado	Int32

En segundo lugar, y para cada dimensión que necesite de él, se deberá crear el campo de código unificado en la base de datos origen. En el caso actual, se deberá añadir dicho campo en la tabla de

residentes, ya que para la tabla correspondiente al centro llamada “DatosResidencia” ya se tiene otro campo denominado código que realizará dichas funciones (se supone que los campos del código unificado para las tablas de la base de datos “CaracteresPlaza” y “Tipologia” ya están creados).

Una vez hecho esto, se rellenarán los códigos unificados de las tablas que lo requieran, debiendo en nuestro caso rellenar los correspondientes a los residentes mediante una transacción.

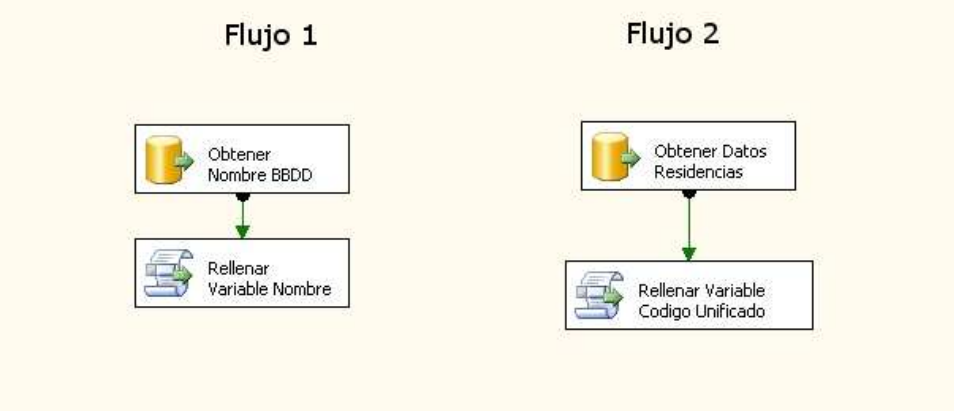
En tercer lugar, se realiza la extracción de los datos contenidos en las tablas de la base de datos origen, se realizan las modificaciones pertinentes, y se cargan en la base de datos destino. Para ello, se tiene un flujo de datos independiente para cada dimensión.

Por último, se realizará la ETL de las tablas auxiliares, para lo cual también se necesitará un flujo de datos independiente para cada tabla auxiliar que se desee rellenar.

Una vez explicado a grandes rasgos el flujo de control, se va a explicar más detenidamente los contenidos de este paquete para que el lector pueda entender lo que hace cada uno

Contenedor “Obtención de Variables del Centro”

En este contenedor se dispone de dos flujos de datos que se adjuntan a continuación:



En el flujo 1 se realizan dos pasos: en el primero, se obtiene el nombre de la base de datos a la cual está atacando el paquete de dimensión, rellenando en segundo lugar la variable “Nombre” para su uso posterior. Para ejemplificar como es el relleno de una variable, se adjunta el código fuente de este componente ya que será utilizado a menudo en pasos posteriores:

Codigo 14.- Pasar Valores a Variables dentro de un Componente Script

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    valor = Row.Nombre
End Sub
```

```
Public Overrides Sub PostExecute()
    Me.Variables.Nombre = valor
```

```
    End Sub
```

Como se observa, en primer lugar hay que guardar el valor que se quiere almacenar en la variable en otra variable exclusiva en el script. Como en el flujo de datos que se tiene no hay más que un registro, tan solo hay que pasar el valor (en caso de haber mas registros se deberán procesar al objeto de conseguir el valor que se quiera para la variable).

Una vez hecho esto, y siempre en la función "PostExecute", hay que pasar el valor de la variable del script a la variable del paquete, de igual forma que se observa en el código.

Cabe recordar que para poder acceder a la variable "Nombre" desde dentro del script, esta debe ser añadida en el casillero de "ReadWriteVariables" en las propiedades del componente.

Mientras, en el flujo 2 se obtienen el nombre y el IDBD (se recuerda que hace las funciones de código unificado de los centros) de de todas las residencias mediante un origen de datos, pasando dichos datos a un script que comparará todos los nombres de bases de datos con el adquirido en la variable del flujo 1. Cuando se halle un registro coincidente, se guardara el valor del campo IDBD de dicho registro en la variable "CodigoUnificado". De este modo, ya se ha obtenido el código unificado del centro.

Contenedor de Creación y Relleno de Códigos Unificados"

Como bien indica el nombre del contenedor, en el mismo se crearán y se rellenarán los códigos unificados de aquellas tablas de la base de datos origen que lo requieran. En este caso puntual, tan solo hace falta la creación y relleno del código unificado en la tabla de "Residentes", que se realizará mediante dos transacciones SQL. Estas dos transacciones ya han sido adjuntadas anteriormente en el método de creación del indicador mediante assemblies, por lo que no se adjuntarán de nuevo.

Contenedor de Carga de Dimensiones

En este contenedor se efectuará la ETL de las dimensiones que deben ser rellenadas automáticamente antes de la ejecución del paquete de hecho. En el caso actual, se trata de las dimensiones correspondientes a residente y a centro. Por ello, se pasa a estudiar detenidamente cada uno de estos dos flujos.

FLUJO DE DATOS DE RELLENO DE LA DIMENSIÓN RESIDENTE

Como se puede observar, este flujo tan solo consta de una fuente de datos y un destino de datos, sin ninguna transformación entre ellas.

Mientras que para realizar la extracción de la fuente de datos utilizamos la misma consulta SQL que se adjunto por el método de assemblies, la carga de datos se realiza mediante la asociación de campos que posee el componente de destino de datos. Para ejemplificar visualmente esa asociación, se adjunta la Figura 28.

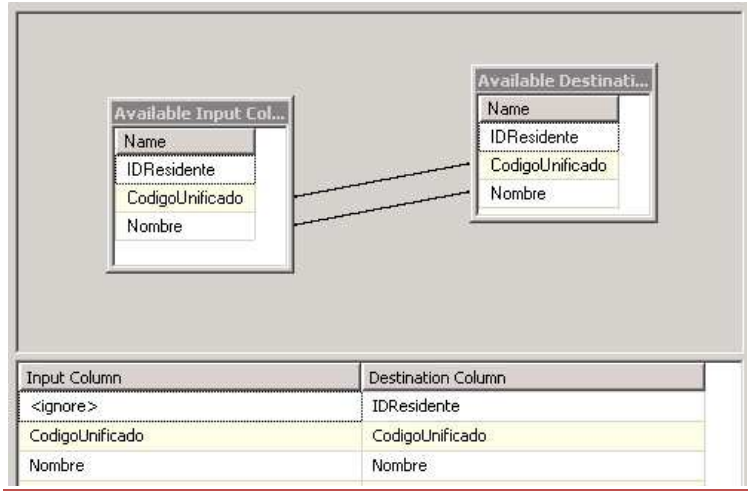


Figura 28.- Asociación de campos en el destino de datos

A la izquierda se ve el origen y a la derecha el destino. Como se observa, no hay correspondencia entre "IDResidente" de centro y almacén, ya que estos son independientes

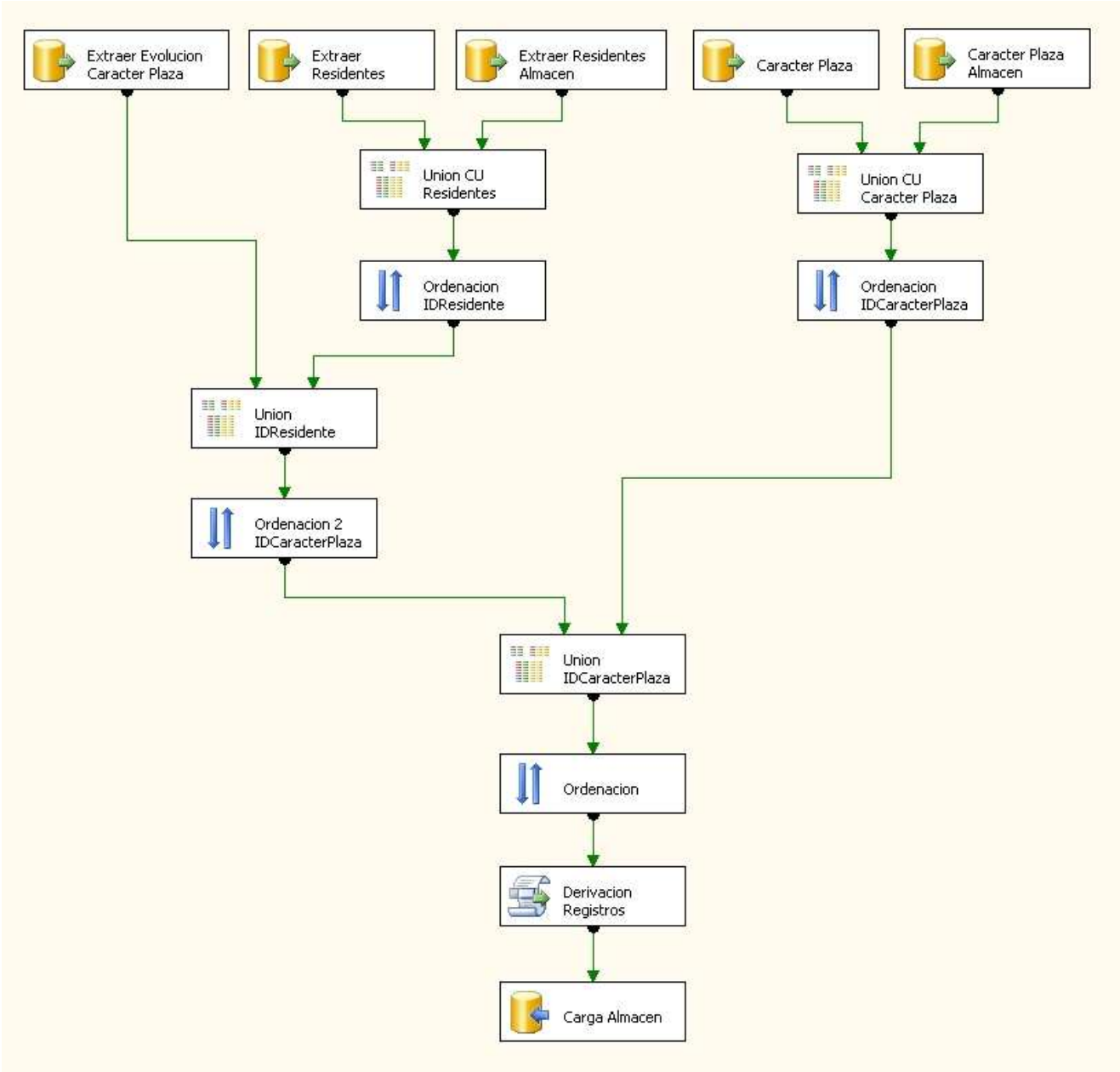
FLUJO DE DATOS DE RELLENO DE LA DIMENSIÓN CENTRO

Como se puede observar, este flujo de datos es similar al anterior, pero esta vez si añadiendo una transformación intermedia, que no hace más que agregar una columna derivada llamada "CodigoUnificado" con el valor de la variable de mismo nombre del paquete.

Contenedor de Carga de Tablas Auxiliares

En este contenedor se realizará el relleno de las tablas auxiliares, las cuales ayudaran posteriormente a adquirir datos que están guardados de una manera bastante incómoda en Resiplus. Se tienen dos flujos de datos: uno para el relleno de la tabla auxiliar de carácter plaza y otro para el relleno de la tabla auxiliar tipología. Para ejemplificar, se va a explicar el flujo de datos correspondiente al relleno de la tabla auxiliar carácter plaza, siendo equivalente al de tipología excepto por los nombres de las tablas a las cuales se debe atacar.

FLUJO DE DATOS DE RELLENO DE LA TABLA AUXILIAR DE CARÁCTER PLAZA



Se puede observar en este flujo de datos como, a partir de cinco componentes de extracción de datos, se cargan determinados datos en una tabla auxiliar.

En primer lugar, se tiene un componente que extrae todos los registros de la evolución de los caracteres de plaza de los residentes en el centro. Dichos registros estarán conformados por un identificador de residente, un identificador de carácter de plaza, y una fecha de cambio de carácter plaza.

Por otro lado, y si se agrupan de dos en dos los cuatro componentes siguientes de obtención de datos, los dos primeros se encargan de, a partir de las bases de datos del centro y el almacén, obtener pares con la correspondencia de identificador de residente del centro e identificador de residente del almacén. Asimismo, los dos últimos componentes se encargaran de obtener pares con la misma correspondencia anterior, pero ahora de los caracteres de plaza.

Esto se realizará mediante componentes de asociación y unión (INNER JOIN), los cuales nos permiten realizar transformaciones de conjuntos de datos de esta índole. Además, se realiza una ordenación adicional para cada conjunto, siendo el identificador del centro el criterio de ordenación.

Una vez extraídos todos los datos necesarios, se analiza cómo actúa el flujo en sí. Si se observa la parte más hacia la izquierda, la primera operación después de la extracción de los datos de la evolución del carácter de plaza es una asociación y unión (INNER JOIN) con los tuplas de identificadores de centro y almacén del residente obtenidas anteriormente.

De este modo, ya se ha conseguido asociar a los registros de evolución el identificador de residente del almacén, el cual debe ser almacenado dentro de las tablas auxiliares.

En segundo lugar, y previa ordenación por el campo de código unificado de carácter plaza, se realiza una asociación y unión (INNER JOIN) igual que la realizada anteriormente, pero esta vez con los tuplas de identificadores del centro y del almacén del carácter de la plaza, siendo este primero el campo de asociación.

Así, ya se tiene las evoluciones con sus correspondientes identificadores de residente y de carácter de plaza del almacén, por lo que ya se puede pasar de esta forma de representación de evolución (un registro por cada cambio de carácter de plaza) a una representación donde cada registro representará el carácter de plaza del residente en un determinado día.

Para ello, se hace uso del script a continuación adjuntado:

Código 15.- Script de Derivacion de Registros para Tablas de Evolucion

```
Public Overrides Sub PreExecute()
   MyBase.PreExecute()
   IDResidente = -999
   fechaant = "9/9/9999"
   fechabucle = "9/9/9999"
End Sub
```

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
   If Row.IDResidente <> IDResidente Then
      fechaant = Date.Today
      fechabucle = Row.DesdeFecha
      While fechabucle <= fechaant

         Me.Output0Buffer.AddRow()
         If Row.IDResidenteAlmacen_IsNull Then
         Else
            Me.Output0Buffer.IDResidenteAlmacen = Row.IDResidenteAlmacen
         End If
         Me.Output0Buffer.Fecha = fechabucle
         If Row.IDCaracterPlazaAlmacen_IsNull Then
         Else
            Me.Output0Buffer.IDCaracterPlazaAlmacen =
            Row.IDCaracterPlazaAlmacen
         End If
         fechabucle = fechabucle.AddDays(1)
      End While
      IDResidente = Row.IDResidente
      fechaant = Row.DesdeFecha.AddDays(-1)
   Else
      fechabucle = Row.DesdeFecha
      While fechabucle <= fechaant
```

```
         Me.Output0Buffer.AddRow()
         If Row.IDResidenteAlmacen_IsNull Then
         Else
            Me.Output0Buffer.IDResidenteAlmacen = Row.IDResidenteAlmacen
         End If
         Me.Output0Buffer.Fecha = fechabucle
         If Row.IDCaracterPlazaAlmacen_IsNull Then
         Else
            Me.Output0Buffer.IDCaracterPlazaAlmacen =
            Row.IDCaracterPlazaAlmacen
         End If
         fechabucle = fechabucle.AddDays(1)
      End While
      IDResidente = Row.IDResidente
   fechaant = Row.DesdeFecha.AddDays(-1)
   End If
End Sub
```

Como se observa, el código no es excesivamente visual, por lo que se va a intentar explicar lo que realmente realiza. Hay que tener en cuenta que se parte de una ordenación realizada por los campos de residente y fecha, siendo ambas ordenaciones descendentes

Para comenzar, inicializa variables que almacenan el identificador del residente procesado en último término, la fecha de la última evolución, y la fecha de iteración para derivar nuevos registros.

En el caso de que el residente anterior sea el mismo que el del registro que se está procesando actualmente, se deberán crear los registros correspondientes a esta evolución, es decir, un registro por cada día desde la fecha de evolución actual hasta la fecha de evolución del registro anterior menos un día, todos con el carácter de plaza correspondiente a la evolución actual.

Para cada registro, en el caso de que el residente anterior sea distinto al que se está procesando actualmente, se supone que esta es la ultima evolución de dicho residente, por lo que se deberán crear un registro por cada día desde la fecha de la evolución actual hasta la fecha de hoy, todos ellos con el carácter de plaza de la evolución actual.

Por ello, y para la detección de que el primer registro es el último cambio de evolución del último residente ordenado por identificador, se inician las variables locales a valores inexistentes dentro de las bases de datos, detectando de esta forma un cambio de residente para el primer registro de todo el conjunto.

A raíz de esta derivación ya se tienen los registros tal y como se desean, y tan solo quedará cargarlos en el almacén de datos mediante un componente de carga en la tabla auxiliar correspondiente. Cabe destacar que el flujo de datos de carga de la tabla auxiliar de tipología realiza exactamente las mismas operaciones, pero siempre teniendo en cuenta que las tablas donde antes eran referentes a carácter de plaza (evolución y demás) ahora serian referentes a la tipología.

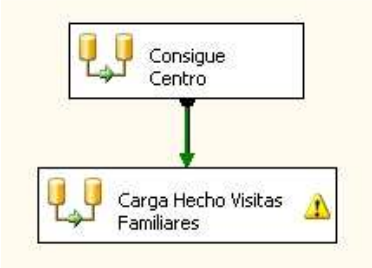
Con todo esto, ya se ha conseguido el relleno de dimensiones y tablas auxiliares, así que ya es el momento de ponerse en marcha para la creación del hecho mediante su correspondiente paquete.

Paquete del Hecho Visitas Familiares

Una vez rellenadas las tablas de dimensiones, hay que proseguir con el relleno de la tabla de hechos. Para ello, se necesitará de un paquete el cual deberá contener la siguiente variable:

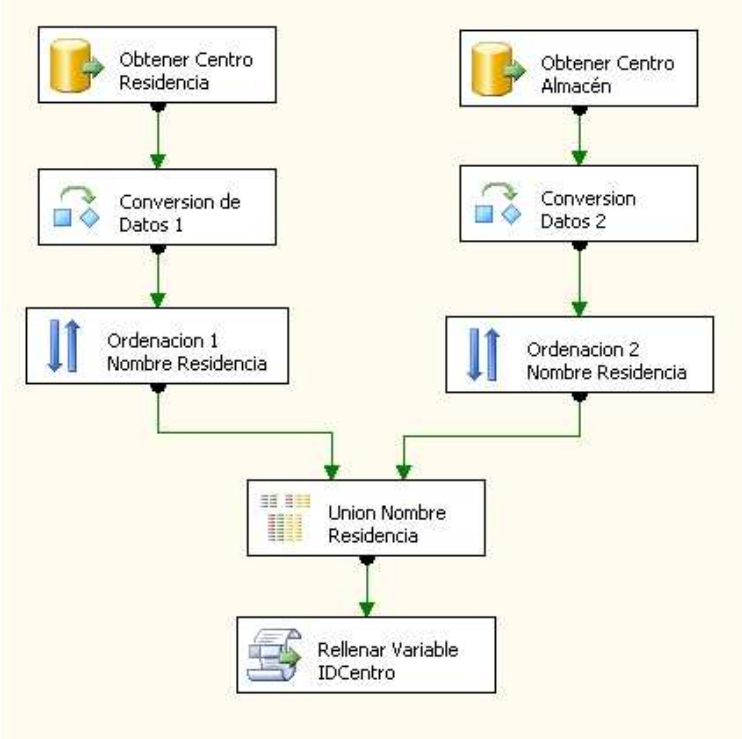
Nombre de la Variable	Tipo de la Variable
IDCentro	Int32

Para comenzar a analizar este nuevo paquete, obsérvese el flujo de control del hecho que aquí se está tratando:



Como se puede observar, se consta de tan solo dos tareas de flujos de datos: la primera corresponde con la adquisición del centro para el cual se está lanzando el paquete, y la segunda como la realiza la ETL propiamente dicha.

Flujo de Datos de Consecución de Centro



En esta tarea, y como se puede deducir a partir del nombre de los componentes utilizados, no se hace más que coger el registro de la residencia a la que se ataca de la tabla “DatosResidencia” y realizar una unión y asociación por la izquierda (LEFT JOIN) con los datos de las residencias disponibles en el almacén de datos. De esta forma, ya se habrá obtenido el identificador de centro correspondiente al almacén, que se guardara en la variable del paquete “IDCentro” mediante un componente script.

Flujo de Datos de Carga de Visitas Familiares

A partir de aquí se va a ver como se realiza la ETL del hecho en sí. Debido al vasto tamaño del flujo de datos encargado, este se va a ir explicando en pequeñas partes al objeto de que el lector pueda entender su funcionamiento ordenado, denominando a cada parte como tramo. En la Figura 29 se ofrece la posibilidad de ver todo el diagrama junto, con los tramos delimitados para seguir la explicación.

Se ha de tener en cuenta que las salidas de un tramo corresponderán con las entradas del tramo posterior, excepto del primer y último tramo, los cuales no tendrán entradas y salidas respectivamente.

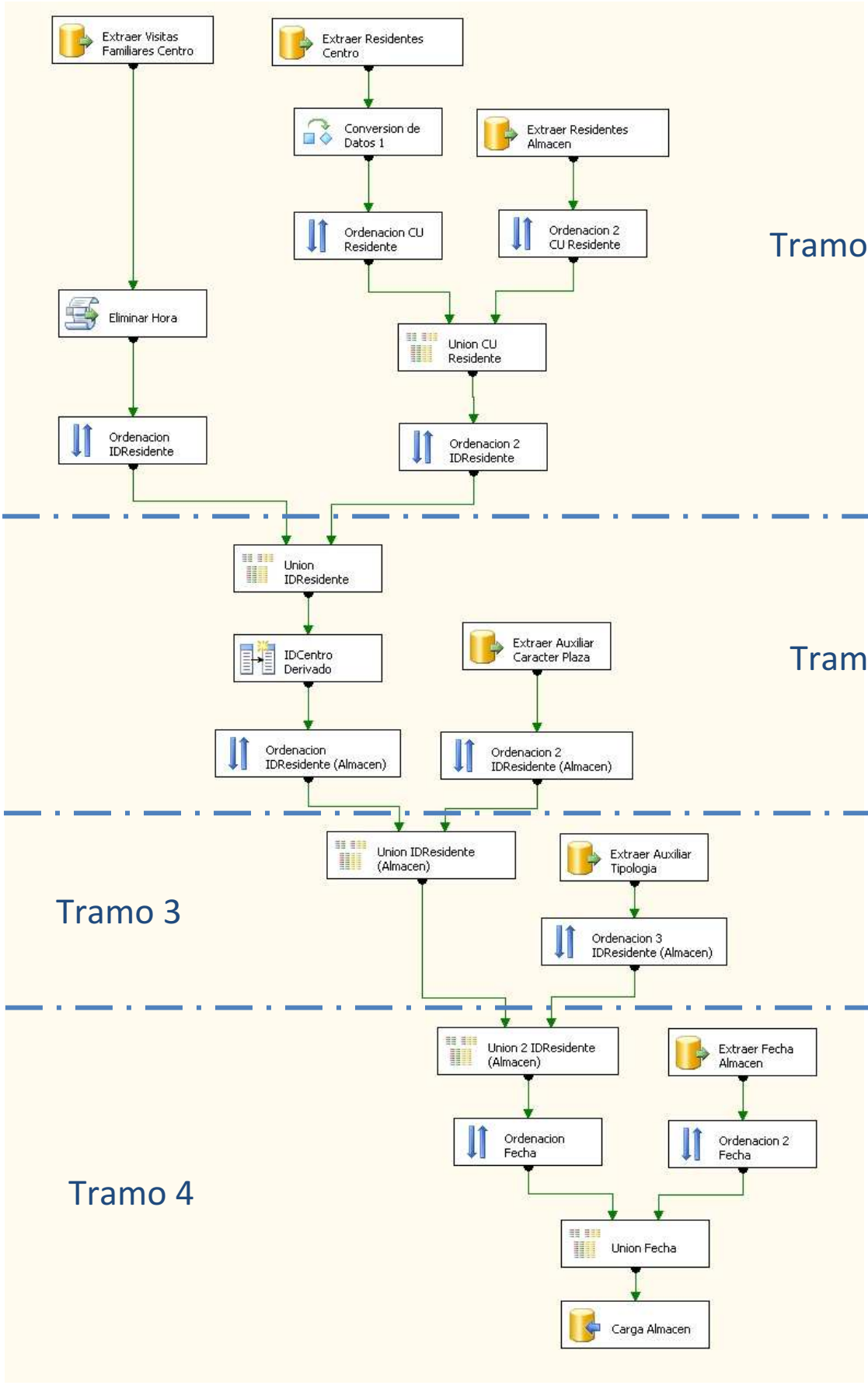
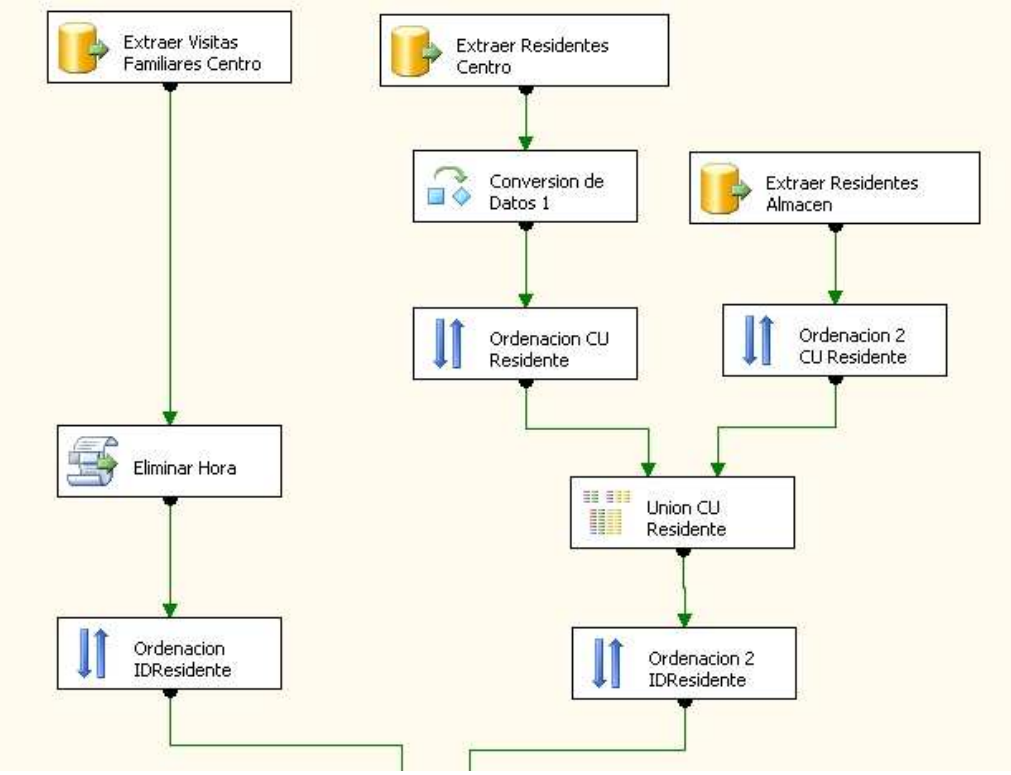


Figura 29.- Flujo de datos de "Carga Hechos Visitas Familiares"

TRAMO 1: OBTENCIÓN DE LAS VISITAS FAMILIARES Y OBTENCIÓN DE RESIDENTE



Como se puede observar, se diferencian dos ramales en los cuales se harán diferentes tareas de adquisición de datos.

En el ramal izquierdo, consulta mediante, se adquieren los datos correspondientes a las visitas familiares. Dicha tarea se efectuará mediante la selección de los campos necesarios (en este caso, "IDResidente" y "FechaHora" de la tabla "VisitasFamiliares").

Una vez hecho esto, y como en esta tabla de Resiplus se almacena tanto la fecha como la hora en el mismo campo, se procede a la eliminación de la hora mediante un script ya que no interesa tenerla, e incluso seria incomoda para realizar una unión y asociación posterior.

Dicho script se adjunta a continuación, ya que es muy habitual su uso y puede ser de utilidad analizar su código:

Codigo 16.- Script de Eliminación de Hora en Registros de Fecha

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)

    fechaactual = Row.fecha

    hora = Row.fecha.Hour * (-1)
    Min = Row.fecha.Minute * (-1)
    seg = Row.fecha.Second * (-1)

    fechaactual = DateAdd(DateInterval.Hour, hora, fechaactual)
    fechaactual = DateAdd(DateInterval.Minute, Min, fechaactual)
    fechaactual = DateAdd(DateInterval.Second, seg, fechaactual)

    Me.Output0Buffer.AddRow()
    Me.Output0Buffer.fecha = fechaactual
```

```
        Me.Output0Buffer.idresidente = Row.idresidente

    End Sub
```

El comportamiento de este script es muy sencillo: lo único que hace es, para cada registro, almacena en distintas variables los valores de la hora, minuto y segundo multiplicadas por menos uno, realizando posteriormente una adición con el valor del propio registro. De esta forma, se consigue eliminar la hora del registro, quedando solamente la fecha del mismo.

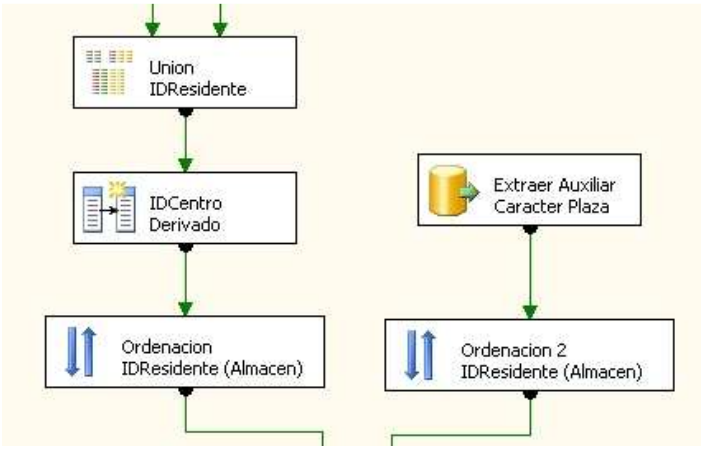
El último paso de este ramal es la ordenación de los registros por el identificador del residente, con miras de realizar una asociación y unión posterior.

Mientras tanto, en el ramal derecho se trata de adquirir los datos pertenecientes a los residentes. Ya que a la hora de cargar los datos en el almacén se necesitan tener los identificadores correspondientes al mismo.

Por ello, se debe conseguir una relación de registros cuyos dos datos sean el identificador de la residencia y el identificador del almacén, para una vez obtenidos, realizar la asociación y unión (INNER JOIN) con el hecho en sí. Para ello, y mediante dos componentes de obtención de datos, se obtienen los registros correspondientes a los identificadores del centro y del almacén, realizando posteriormente una asociación y unión por el código unificado.

De dicho modo, ya se tiene cada identificador del centro relacionado con su identificador del almacén, por lo que el siguiente paso es realizar una ordenación por el residente del almacén al objeto de realizar otra asociación y unión por dicho campo.

TRAMO 2: OBTENCIÓN DE CARÁCTER DE PLAZA



Como se puede observar, en este tramo también se dispone de dos ramales.

El primer paso del ramal izquierdo es la asociación y unión (INNER JOIN) por el campo identificador de residente del centro, habiendo obtenido de esta forma ya el identificador del residente de la residencia.

De esta asociación y unión se consiguen tuplas con el identificador de residente del almacén y la fecha en la que se realizó dicha visita. El identificador de residente del centro, llegados a este punto, es desechable, ya que no es requerido para ninguna asociación y unión posterior.

El siguiente paso sería crear un campo derivado en los registros para añadir el identificador de centro correspondiente el almacén, que se recuerda que introducido en la variable "IDCentro" del paquete. Por lo tanto, el relleno de dicho campo, tendrá el valor de la variable anteriormente enunciada.

El último paso sería la ordenación por el identificador de residente del almacén y por la fecha, cuyos valores servirán para realizar una posterior asociación y unión.

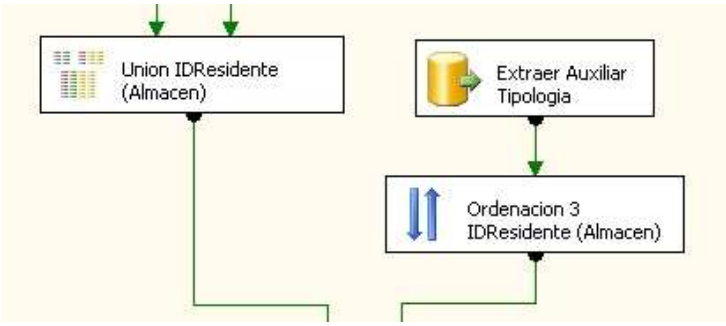
Mientras tanto, por el ramal derecho se realiza una consulta a la tabla auxiliar de los caracteres de plaza. Ahora bien se podrá ver la utilidad de este tipo de tablas: si no se hubiese realizado la tarea de carga de la tabla auxiliar, ahora se tendrían que realizar una serie de operaciones, sino iguales, muy parecidas a las realizadas en la ETL de dicha tabla auxiliar.

Por ello, lo realizado en esos pasos posteriores son tareas adelantadas, y teniendo en cuenta que esto sucedería en numerosas ocasiones (ya que casi todos los indicadores requieren de la dimensión carácter plaza), se puede deducir que la creación de estas tablas auxiliares es un acierto en optimización.

Hay que tener en cuenta que la tarea de extracción de datos que se observa en el diagrama consigue tuplas de valores con el identificador de residente del almacén, una fecha y el identificador de carácter plaza del almacén.

Como último paso del ramal derecho de este tramo, se realizará la pertinente ordenación por el identificador del residente del almacén y fecha, al objeto de realizar una asociación y unión con los registros obtenidos por el ramal izquierdo.

TRAMO 3: OBTENCIÓN DE TIPOLOGÍA



En este caso, el primer y único paso del ramal izquierdo seria la realización de la asociación y unión por la izquierda (LEFT JOIN) de los registros del ramal izquierdo y derecho del tramo anterior. Dicha asociación y unión debe ser por la izquierda ya que, a diferencia de las anteriores, puede ser que no haya algún código unificado del carácter de plaza introducido correctamente en el almacén o en el centro. Si en dicho caso se realizase una unión y asociación normal (INNER JOIN), el registro que no tuviese asociación con ningún registro de carácter de plaza no sería salida de dicho componente, perdiendo información, cosa que no se desea.

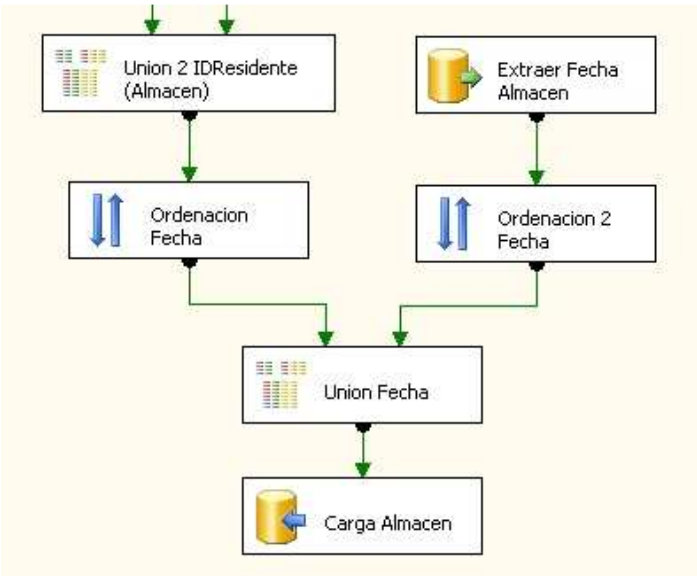
Además, el haber definido el campo de identificador carácter de plaza del almacén con la capacidad de soportar valores nulos nos va a dar una mayor versatilidad a la hora de realizar operaciones con los cubos de datos (ya se verá posteriormente como dichos valores pueden tomar un valor especificado por el usuario).

Como resultado de esta asociación y unión se obtienen tuplas con el identificador de residente del almacén, el identificador de centro del almacén, la fecha en la que se realizó dicha visita, y adicionalmente y conseguido en este último paso, el identificador del carácter plaza del almacén.

Al mismo tiempo, por el ramal derecho se procede a realizar los mismos pasos que los realizados anteriormente con el carácter de plaza, pero esta vez con la tipología. En primer lugar se extraerán las tuplas conformadas por identificador de residente del almacén, fecha e identificador de la tipología del almacén.

Al objeto de realizar una posterior asociación y unión con los registros del ramal izquierdo, se ordenan los datos del ramal derecho por identificador de residente del almacén y fecha. Nótese que esto no se realiza en el ramal izquierdo ya que dichos registros ya padecieron dicha ordenación durante el segundo tramo, por lo que sería innecesario realizar la misma ordenación de nuevo.

TRAMO 4: OBTENCIÓN DE TIEMPO Y CARGA



Como se puede observar, este último tramo también cuenta con dos ramales. Para comenzar, en el ramal izquierdo se realiza la asociación y unión por la izquierda (LEFT JOIN) de las salidas del tramo anterior.

Esta asociación y unión da como resultado tuplas con el identificador de residente del almacén, el identificador de centro del almacén, la fecha en la que se realizó dicha visita, además de, ahora ya sí, los identificadores del carácter plaza y de la tipología del almacén.

Por último, y como paso final de este ramal izquierdo, se realiza una ordenación por el campo fecha, el cual servirá para realizar la última asociación y unión del paquete.

Mientras tanto, por el ramal derecho se extraen del almacén las tuplas correspondientes al identificador de la fecha y la fecha en si, por lo que no queda más que realizar la pertinente ordenación por el campo fecha.

El siguiente paso será la asociación y unión (INNER JOIN) entre los registros de los dos ramales asociando mediante el campo fecha. En este caso sí que se está delante de una asociación y unión normal, ya que cualquier valor de fecha debe estar introducido en el almacén de datos, y si no es así, posiblemente estemos delante de un registro erróneo que no será asociado y, por lo tanto, no será parte de los datos finales a cargar.

Como se puede deducir, mediante todas estas asociaciones y uniones se han ido consiguiendo todos los datos necesarios para rellenar la tabla de hechos del almacén: el identificador del residente del almacén, el identificador de centro del almacén, los identificadores de carácter de plaza y tipología del almacén, y en este último paso, el identificador del tiempo (habiendo desechado por el camino aquellos campos que no hacían falta, como el identificador de residente del centro).

Por lo tanto, y mediante un componente de destino de datos, se procede a realizar la carga de los registros en el almacén. La correspondencia de los campos se muestra en la Figura 30.

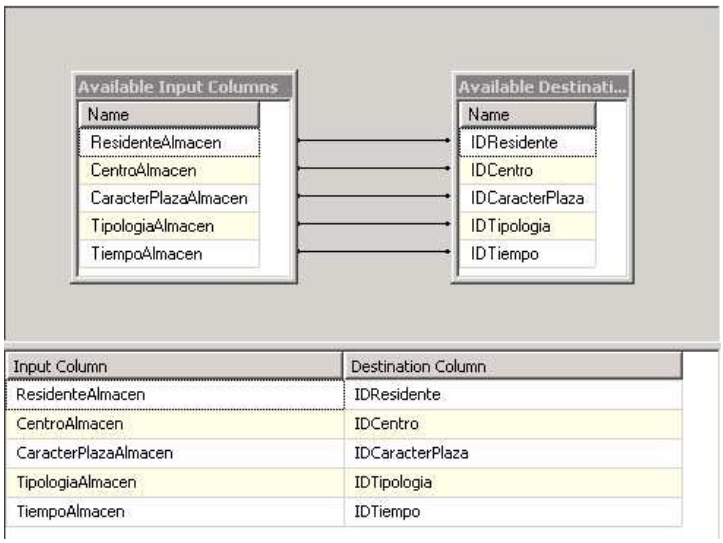


Figura 30.- Correspondencia en el destino de datos para el indicador de visitas de familiares

Con esto, ya se tiene el primer indicador de gestión desarrollado con Integration Services, que, como se ha podido observar, ofrece una estrategia mucho mas desagregada que en el desarrollo de assemblies.

4.7.- Desarrollo del Indicador de Curas de Úlceras

A continuación se va a estudiar como implementar el indicador de curas de úlceras, un indicador mucho más complicado que el anterior de visitas familiares, ya que en él hay muchas más dimensiones y muchos más requisitos que tener en cuenta.

Para ello, se comenzará adjuntando la definición del indicador (con una descripción del mismo, además de adjuntar las tablas de bases de datos a las cuales debe atacar). A continuación, se comentará la estructura de las tablas del almacén de datos a crear para el almacenamiento de la información. En último lugar, se implementará el indicador tanto con el modelo de assemblies como con el modelo de Integration Services.

4.7.1.- Definición del Indicador

En primer lugar se va a definir en tan solo una frase la especificación del indicador para a continuación desgranarla en sus distintos componentes:

"Número total de curas de úlceras realizadas a los distintos residentes (adjuntando el carácter de la plaza y la tipología que tienen en el momento en el que ocurre la cura), indicando al centro al cual pertenecen y organizadas por día, mes, trimestre o año. Además, se requerirá información de la cura y de la úlcera en si, como son la localización de la úlcera, el tipo, el estadio y la procedencia de la misma, además del tamaño en el momento de la cura y si dicha úlcera está curada."

Una vez realizada la definición, se procede a desgranarla al objeto de observar todos los componentes que existen dentro de una dimensión:

- Hecho: curas de úlceras.
- Medida: número total.
- Dimensiones: residente, tipología, carácter plaza, centro, tiempo, úlcera, tipo de úlcera, localización de úlcera, procedencia de la úlcera, estadio de la úlcera y úlcera curada.
- Atributos del hecho: tamaño de la úlcera.

Aquí se han agregado un par de dimensiones que, a priori pueden ser innecesarias, pero que si son añadidas, pueden ser muy útiles a la hora de desgranar la dimensión.

Por ejemplo, parece innecesaria una dimensión úlcera como tal, ya que esto no influye en el número de las curas, pero que si se añade, se podrán realizar agrupaciones de modo que se podrá saber en todo momento el número de curas realizadas a una misma úlcera.

Por otro lado, también llama la atención la dimensión úlcera curada, la cual parece no aportar información adicional, pero que añadiéndola, se podrá saber el número de úlceras que han sido curadas durante un periodo determinado de tiempo, lo cual si puede ser interesante para el usuario final de la información.

4.7.2.- Definición de las Tablas del Almacén de Datos

A continuación se va a desglosar cada una de las tablas necesarias que se deberán crear en la base de datos al objeto de guardar toda la información.

Como ya se apuntó en el desglose del indicador, se necesitará de una tabla de hechos a la cual se denominará "Fact_ÚlcerasCuras", y cinco tablas de dimensiones denominadas "Dim_Centro", "Dim_Residente", "Dim_Tiempo", "Dim_CaracterPlaza", "Dim_Tipologia", "Dim_Úlcera", "Dim_CurasLocalizacion", "Dim_CurasEstadio", "Dim_ÚlcerasProcedencias", "Dim_ÚlcerasTipos" y "Dim_ÚlceraCurada", formando el esquema relacional mostrado en la Figura 31.

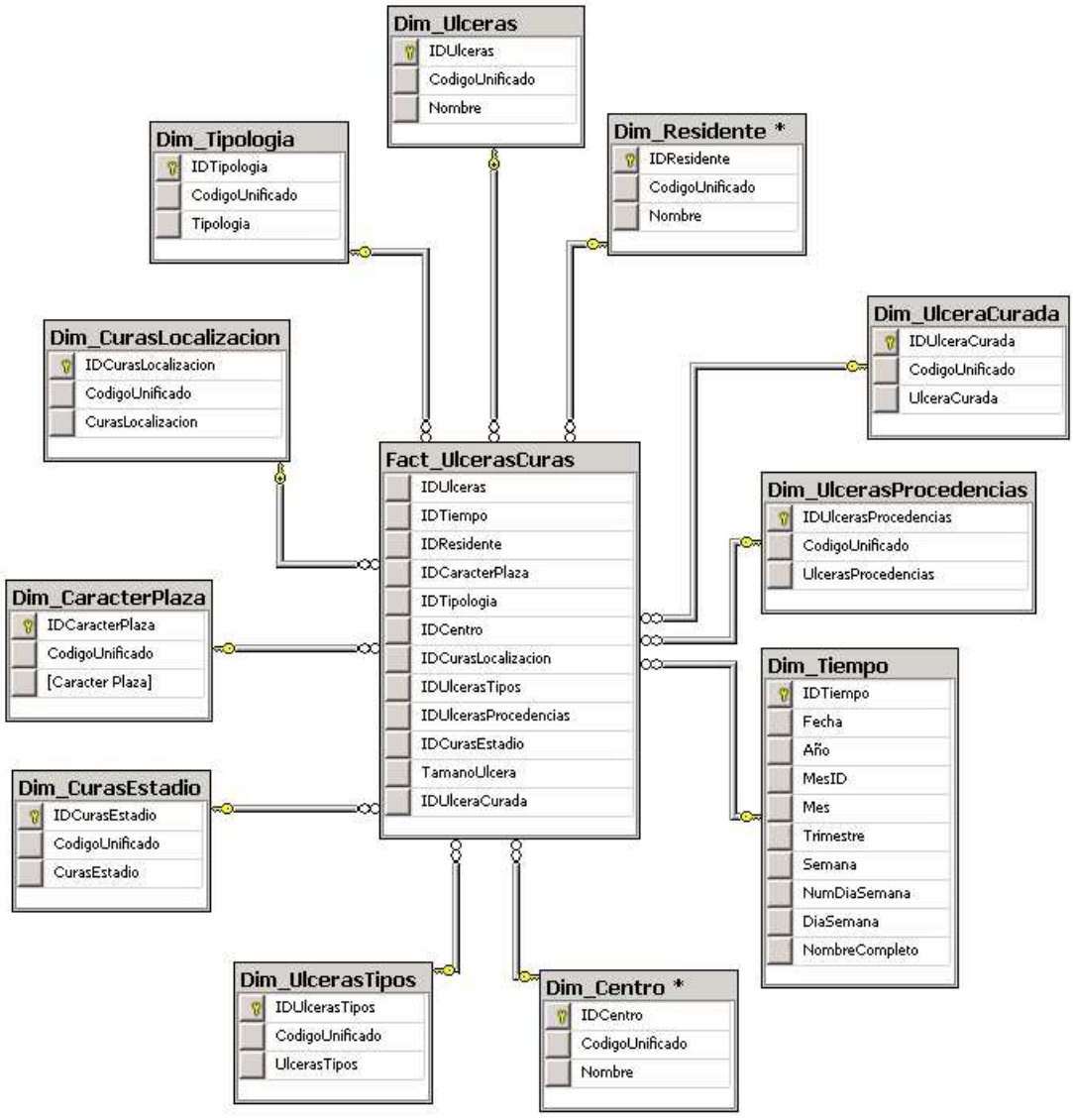


Figura 31.- Esquema relacional para el indicador de curas de úlceras en el almacén de datos

4.7.3.- Implementación del Indicador mediante Assemblies

En este apartado se va a estudiar la implementación del indicador de curas de úlceras mediante la metodología de assemblies, separando dicho proceso en tres pasos descritos en profundidad: la modificación del assembly almacén, la creación de los assemblies de dimensiones necesarios y la creación del assembly de hechos.

Modificación del Assembly Almacén para Creación de Tablas

En un principio se comenzará modificando el assembly del almacén al objeto de crear todas las tablas necesarias, ya ilustradas en el esquema relacional del indicador.

A continuación se adjunta el código necesario para la creación de la tabla de dimensión "Dim_ÚlcerasProcedencias", y para la tabla de hechos "Fact_ÚlcerasCuras", con motivo de ejemplificar la creación de una tabla de cada tipo, y ver la creación de relaciones de claves ajenas mediante este modelo:

Código 17- Creación de la Tabla "Dim_ÚlcerasProcedencias"

```
static void createTableDimÚlcerasProcedencias (string tabla)
{
    ArrayList atributos = new ArrayList();
    ArrayList indicesUnicos = new ArrayList();
    ArrayList indicesNoUnicos = new ArrayList();

    atributos.Add("IDÚlcerasProcedencias int IDENTITY(1,1) PRIMARY
    KEY");
    atributos.Add("CodigoUnificado nvarchar(19)");
    atributos.Add("ÚlcerasProcedencias nvarchar(50)");

    createTable(tabla, atributos);

    #region indexesUniques

    indicesUnicos.Add("CodigoUnificado");

    createIndexUnique(tabla, indicesUnicos);

    #endregion

    #region indexesNonUniques

    createIndexNonUnique(tabla, indicesNoUnicos);

    #endregion
}
```

Código 18.- Creación de la Tabla "Fact_ÚlcerasCuras"

```
static void createTableFactÚlcerasCuras(string tabla)
{
    ArrayList atributos = new ArrayList();
    ArrayList indicesUnicos = new ArrayList();
    ArrayList indicesNoUnicos = new ArrayList();
```

```
        atributos.Add("IDÚlceras int NOT NULL REFERENCES
        Dim_Úlceras(IDÚlceras)");
        atributos.Add("IDTiempo int NOT NULL REFERENCES
        Dim_Tiempo(IDTiempo)");
        atributos.Add("IDResidente int NOT NULL REFERENCES
        Dim_Residente(IDResidente)");
        atributos.Add("IDCaracterPlaza int REFERENCES
        Dim_CaracterPlaza(IDCaracterPlaza)");
        atributos.Add("IDTipologia int REFERENCES
        Dim_Tipologia(IDTipologia)");
        atributos.Add("IDCentro int NOT NULL REFERENCES
        Dim_Centro(IDCentro)");
        atributos.Add("IDCurasLocalizacion int REFERENCES
        Dim_CurasLocalizacion(IDCurasLocalizacion)");
        atributos.Add("IDÚlcerasTipos int REFERENCES
        Dim_ÚlcerasTipos(IDÚlcerasTipos)");
        atributos.Add("IDÚlcerasProcedencias int REFERENCES
        Dim_ÚlcerasProcedencias(IDÚlcerasProcedencias)");
        atributos.Add("IDCurasEstadio int REFERENCES
        Dim_CurasEstadio(IDCurasEstadio)");
        atributos.Add("IDÚlceraCurada int REFERENCES
        Dim_ÚlceraCurada(IDÚlceraCurada)");
        atributos.Add("TamanoÚlcera float NULL");

        createTable(tabla, atributos);

        #region indexesUniques

        createIndexUnique(tabla, indicesUnicos);

        #endregion

        #region indexesNonUniques

        indicesNoUnicos.Add("IDTiempo");
        indicesNoUnicos.Add("IDResidente");
        indicesNoUnicos.Add("IDCaracterPlaza");
        indicesNoUnicos.Add("IDTipologia");
        indicesNoUnicos.Add("IDCentro");
        indicesNoUnicos.Add("IDÚlcera");
        indicesNoUnicos.Add("IDCurasEstadio");
        indicesNoUnicos.Add("IDÚlceraCurada");

        createIndexNonUnique(tabla, indicesNoUnicos);

        #endregion
    }
```

Las tablas de dimensión restantes serán creadas de la misma forma que se ha hecho con la tabla “Dim_ÚlcerasProcedencias”, realizando los cambios oportunos de nombres de campos y tipos de los mismos.

Creación de los Assemblies de Dimensiones

Como se puede observar, hay dimensiones para las cuales ya se crearon assemblies de dimensiones en el desarrollo del indicador de visitas familiares. Dichos indicadores son totalmente reusables, por lo que no se deberán volver a implementar.

Por otro lado, y al igual que ocurría en el indicador de visitas familiares, hay dimensiones las cuales deben ser rellenadas mediante las tablas maestras. Dichas dimensiones serian “Dim_CurasLocalizacion”, “Dim_CurasEstadio”, “Dim_ÚlcerasProcedencias”, “Dim_ÚlcerasTipos”.

Las dos dimensiones nuevas restantes serian “Dim_Úlcera” y “Dim_ÚlceraCurada”, siendo bien distintas entre sí. Mientras la primera de ellas sigue una topología parecida a la dimensión residente vista en la implementación del indicador de visitas familiares, la segunda de ellas merece un tratamiento especial.

Los posibles valores para la dimensión “Dim_ÚlceraCurada” tan solo va a tener 2 valores posibles: ‘Si’ o ‘No’. Por ello, no hace falta recurrir a la base de datos origen, sino que podríamos rellenar dicha tabla con los valores por defecto. En este punto entra en juego el assembly “RellenaDimensiones”.

Creación del Assembly de la Dimensión Úlcera

A continuación se va a describir como implementar el assembly de la dimensión úlcera, viendo todas las tareas que realiza y comprobando su similitud con el anteriormente implementado assembly de la dimensión residente.

Como ya ocurría en este, la primera tarea será la creación y el relleno del campo “CodigoUnificado” en la base de datos de origen, creando de esta forma el enlace entre las úlceras de las base de datos origen y destino.

Con el campo ya creado y rellenado en la base de datos origen, no queda más que realizar la consulta a dicha base de datos para rellenar la tabla auxiliar creada dentro del assembly, siendo los campos requeridos los enunciados en la descripción del assembly.

Código 19.- Consulta SQL de la Dimensión Úlceras

```
SELECT IDÚlcera, CodigoUnificado FROM Úlceras
```

Como se puede observar, en la consulta no se hace referencia alguna al campo “Nombre”, ya que este no existe en la base de datos origen. Dicho campo no es más que un nombre predefinido formado en primer lugar por la palabra “Úlcera” seguida del código unificado de la misma.

Una vez se tienen todos los campos requeridos para rellenar la dimensión en la tabla auxiliar, no queda más que mandarla como output del assembly para que el assembly unificador – rellenador haga su tarea.

Modificación del Assembly “RellenaDimensiones”

Como se ha comentado en subapartados anteriores, la creación de la dimensión “Dim_ÚlceraCurada” merece un tratamiento especial, ya que esta cuenta con unos valores predefinidos conocidos a priori e iguales para todos los centros.

Como los posibles valores de esta dimensión son “Curada” o “No Curada”, se puede hacer uso del assembly “RellenaDimensiones”, encargado de introducir estos pequeños espectros de valores que nunca cambian en dimensiones.

Código 20.- Transacción SQL de Relleno de la Dimensión Úlcera Curada

```
INSERT INTO Dim_ÚlceraCurada (CodigoUnificado, ÚlceraCurada) VALUES (0,'No
Curada')
```

```
INSERT INTO Dim_ÚlceraCurada (CodigoUnificado, ÚlceraCurada) VALUES
(1,'Curada')
```

Con esto, el programador ya tiene conocimiento de que siempre una úlcera ya curada tendrá como código 1 y aquella que no esté curada tendrá como código cero, haciendo el proceso de creación del hecho algo más sencillo.

Creación del Assembly de Hecho de Curas de Úlceras

Al igual que ocurría con la creación del assembly de hecho de visitas de familiares, las modificaciones a realizar en la base del assembly de hechos son dos: modificar la consulta de con la cual se adquieren los registros de la base de datos origen y estructura de la tabla auxiliar donde se guardaran dichos registros, siendo esta tabla el output de este assembly y el input para el assembly unificador – rellenador.

A continuación, se adjunta la consulta para la obtención de registros:

Código 21.- Consulta SQL del Hecho de Curas de Úlceras

```
SELECT ucur.IDÚlcera as IDÚlceras, ulc.CodigoUnificado as ÚlceraUnificado,
ulc.Curado as ÚlceraCurada, res.IDResidente as IDResidente,
res.CodigoUnificado as ResidenteUnificado, ucur.FechaHora as Tiempo,
cest.CodigoUnificado as EstadioUnificado, ucur.Tamano as Tamano,
cloc.CodigoUnificado as LocalizacionUnificado, utip.CodigoUnificado as
TipoUnificado, upro.CodigoUnificado as ProcedenciaUnificado
FROM ÚlcerasCuras as ucur
INNER JOIN (SELECT IDÚlcera, IDResidente, CodigoUnificado, Curado,
IDLocalizacion, IDÚlceraTipo, IDÚlceraProcedencia FROM Úlceras) as ulc
on ucur.IDÚlcera=ulc.IDÚlcera
INNER JOIN (SELECT IDResidente, CodigoUnificado FROM Residentes) as res
on ulc.IDResidente=res.IDResidente
LEFT JOIN (SELECT IDEstadio, CodigoUnificado FROM CurasEstadio) as cest
on ucur.IDEstadio= cest.IDEstadio
LEFT JOIN (SELECT IDLocalizacion, CodigoUnificado FROM CurasLocalizacion) as
cloc
on ulc.IDLocalizacion=cloc.IDLocalizacion
LEFT JOIN (SELECT IDÚlceraTipo, CodigoUnificado FROM ÚlcerasTipos) as utip
on ulc.IDÚlceraTipo=utip.IDÚlceraTipo
LEFT JOIN (SELECT IDÚlceraProcedencia, CodigoUnificado FROM
ÚlcerasProcedencias) as upro
on ulc.IDÚlceraProcedencia=upro.IDÚlceraProcedencia
```

Lógicamente, al ser un hecho con muchas más dimensiones que el hecho de visitas familiares, la consulta es mucho más extensa y complicada que la vista en dicho indicador.

La tabla auxiliar creada para introducir los distintos valores sigue la siguiente estructura:

Curas Úlceras (Tabla Auxiliar)	
Nombre del Campo	**Tipo de Campo**
ResidenteUnificado	String
CaracterPlazaUnificado	String
TipologiaUnificado	String
CentroUnificado	String
ÚlcerasUnificado	String
CurasLocalizacionUnificado	String
ÚlcerasTiposUnificado	String
CurasEstadioUnificado	String
ÚlceraCuradaUnificado	String
TiempoUnificado	DateTime
TamanoÚlcera	String

Si se comparan la consulta de adquisición de datos con la tabla auxiliar, se comprueba que los datos correspondientes al carácter de la plaza del residente y a la tipología del mismo no son adquiridos mediante la primera, ya que como se ha comentado anteriormente existen funciones específicas para ello al objeto de simplificar la consulta SQL.

Además, este assembly poseerá una pequeña función de código denominada ComprobacionCuradas() que tendrá como argumento de entrada la tabla obtenida en la consulta SQL y proporcionará la misma tabla, pero esta vez con la diferencia que los códigos del campo "ÚlceraCuradaUnificado" ya vendrán puestos correctamente con relación a los introducidos anteriormente en la tabla "Dim_ÚlceraCurada"

Resumen de Exportaciones a Crear en Resiplus BE

A continuación se enuncian las exportaciones a crear al objeto de rellenar las tablas correspondientes al hecho de curas de úlceras.

Dimensión Úlceras		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Dim_Úlcera.dll	En cada centro	Antes de enviar la estadística
UnificadorRellenador.dll	En el unificador	Después de la estadística

Rellenar Dimensiones Genéricas		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
RellenaDimensiones.dll	En el unificador	Antes de enviar la estadística

Hecho Úlceras Curas		
Nombre del Assembly	**Donde Ejecutarlo**	**Cuando Ejecutarlo**
Fact_ÚlcerasCuras.dll	En cada centro	Antes de enviar la estadística
UnificadorRellenador.dll	En el unificador	Después de la estadística

Se da por supuesto que, con anterioridad, se ha lanzado el indicador previamente implementado correspondiente a las visitas de familiares. Si no, habría que crear también las exportaciones

correspondientes a la creación del almacén, dimensión tiempo, dimensión residente y dimensión centro.

4.7.4.- Implementación del Indicador mediante Integration Services

Ya integrado el indicador mediante el método de assemblies, se va a proceder a explicar cómo sería la implementación mediante Integration Services.

Para ello, no hay más que realizar las modificaciones pertinentes a los paquetes creados en la implementación del indicador de visitas de familiares por el método de Integration Services. En caso de partir de cero, sería necesaria la creación de los paquetes de almacén y de dimensión tal cual se creó en dicho punto.

Paquete Creador del Almacén

En primer lugar, se van a estudiar los cambios a introducir en el paquete de almacén al objeto de implementar el indicador de curas de úlceras.

Partiendo del conocimiento que se tiene de dicho indicador, se sabe que habrá que crear el siguiente grupo de tablas de dimensiones: “Dim_CurasLocalizacion”, “Dim_CurasEstadio”, “Dim_ÚlcerasProcedencias”, “Dim_ÚlcerasTipos”. Estas tablas se rellenaran mediante el método de tablas maestras, ya que tienen valores que pueden diferir dependiendo del que introdujo los datos en cada residencia, pero tienen un mismo significado.

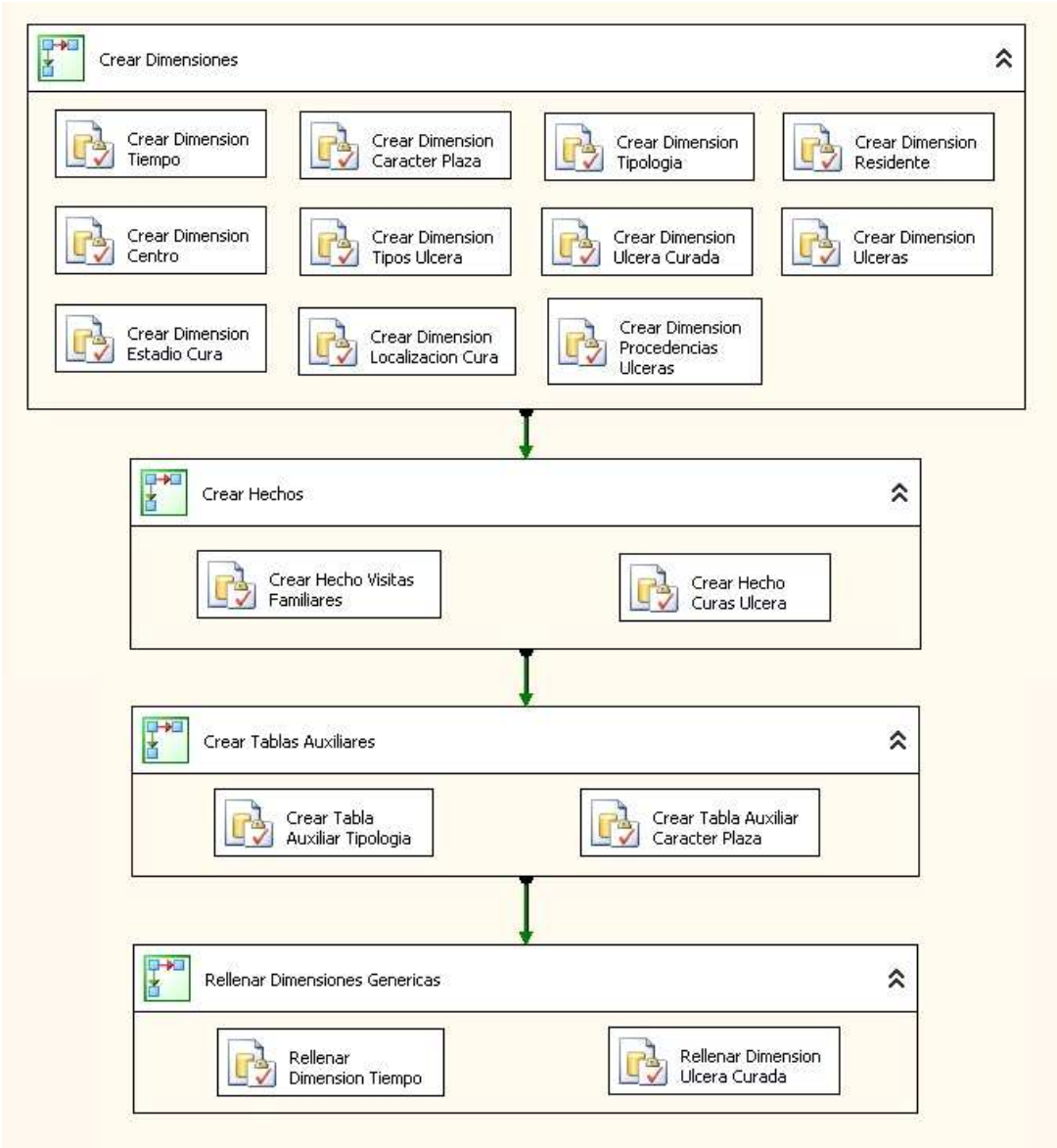
En segundo lugar, también se debe crear la tabla de dimensión “Dim_Úlcera”, la cual será una enumeración de las úlceras que hayan ocurrido en cada residencia y se rellenará mediante el paquete de dimensión.

Por otro lado, hay que destacar también la existencia de la tabla de dimensión “Dim_ÚlceraCurada”, que tiene unos valores fijos (“Curada” y “No Curada”). Por lo tanto, además de crear esta tabla en el paquete que se está estudiando, también hay que encargarse del relleno de la misma. Para ello se utilizará un componente de transacción SQL con el código ya adjuntado en el método de assemblies.

Además, no hay que olvidar la tabla de hechos propiamente dicha, que tendrá que ser creada también dentro de este paquete. Dicha tabla se denominará “Fact_ÚlcerasCuras”, y estará relacionada con todas las tablas de dimensiones que conforman el indicador.

Cabe destacar que para este indicador no hace falta la creación de ninguna tabla auxiliar adicional, ya que no hay nuevos datos que deseen ser accedidos de forma instantánea y que en la base de datos de Resiplus sean poco accesibles.

El resultado del flujo de control es el siguiente:



A continuación se va a adjuntar la creación de una de las tablas de dimensiones, la correspondiente a la tabla "Dim_ÚlcerasTipos". Esta consulta SQL será a introducida dentro de la tarea de transacción SQL al objeto de ser ejecutada:

Código 22.- Transacción SQL de la Creación de la Tabla de Dimensión Tipos de Úlceras

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
if not exists (Select * From information_schema.tables where
table_name='Dim_ÚlcerasTipos')
CREATE TABLE [dbo].[Dim_ÚlcerasTipos](
 [IDÚlcerasTipos] [int] IDENTITY(1,1) NOT NULL,
 [CodigoUnificado] [nvarchar](19) NULL,
 [ÚlcerasTipos] [nvarchar](50) NULL,
PRIMARY KEY CLUSTERED
(
 [IDÚlcerasTipos] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY],
 CONSTRAINT [IX_Dim_ÚlcerasTipos_CodigoUnificado] UNIQUE NONCLUSTERED
```

```
(
 [CodigoUnificado] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Para acabar, también se va a incluir el código de la creación de la tabla de hechos del indicador, llamada "Fact_ÚlcerasCuras". Como se observa, es de vital importancia la creación de todas las claves ajenas correspondientes cada una con una tabla de dimensiones creadas en el contenedor anterior de creación de dimensiones.

Código 23.- Transacción SQL de la Creación de la Tabla de Hechos de Curas de Úlceras

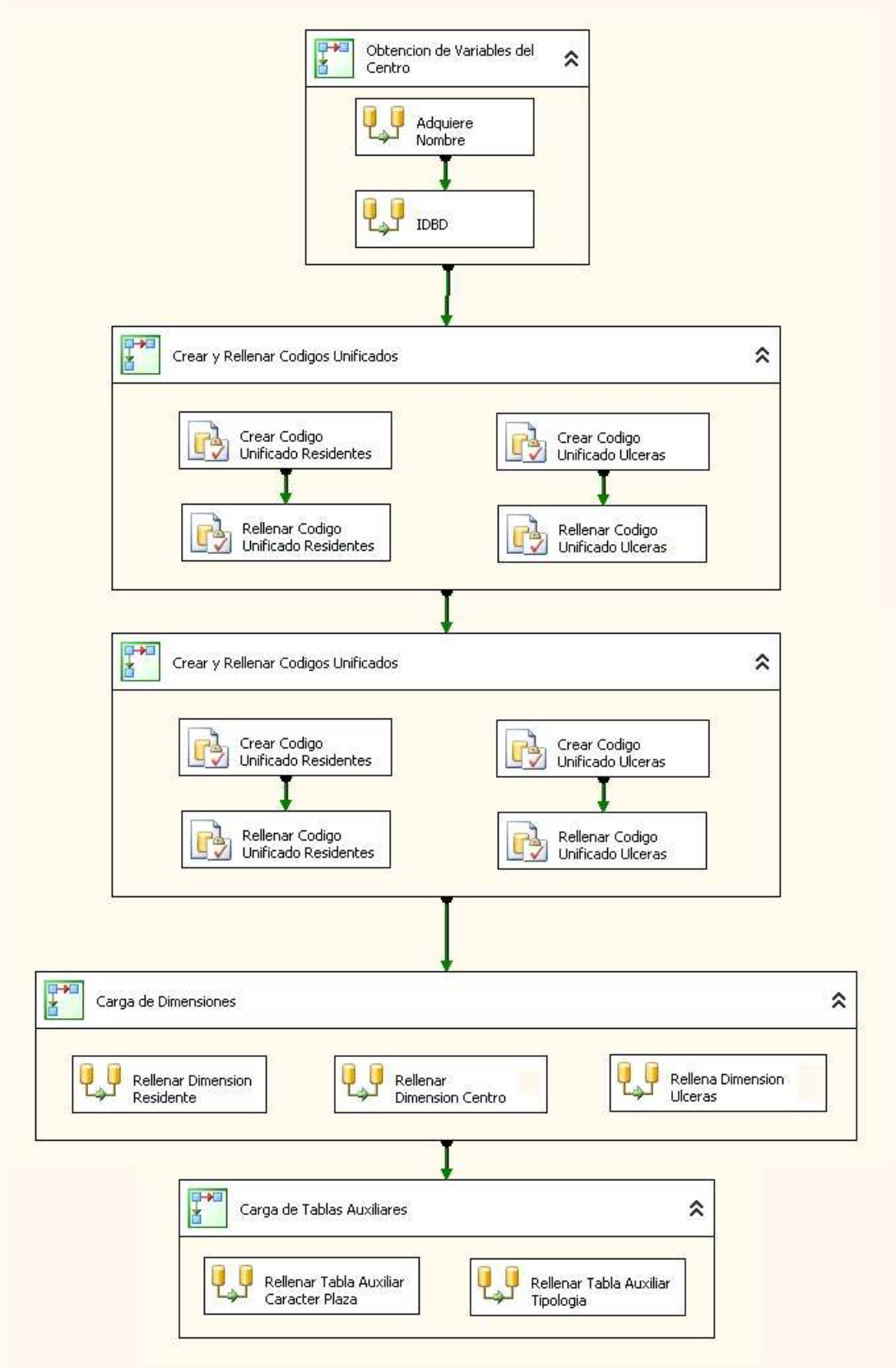
```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
if not exists (Select * From information_schema.tables where
table_name='Fact_ÚlcerasCUras')
CREATE TABLE [dbo].[Fact_ÚlcerasCuras](
 [IDÚlceras] [int] NOT NULL,
 [IDTiempo] [int] NOT NULL,
 [IDResidente] [int] NOT NULL,
 [IDCaracterPlaza] [int] NULL,
 [IDTipologia] [int] NULL,
 [IDCentro] [int] NOT NULL,
 [IDCurasLocalizacion] [int] NULL,
 [IDÚlcerasTipos] [int] NULL,
 [IDÚlcerasProcedencias] [int] NULL,
 [IDCurasEstadio] [int] NULL,
 [TamanoÚlcera] [float] NULL,
 [IDÚlceraCurada] [int] NULL
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN
KEY([IDCaracterPlaza])
REFERENCES [dbo].[Dim_CaracterPlaza] ([IDCaracterPlaza])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN KEY([IDCentro])
REFERENCES [dbo].[Dim_Centro] ([IDCentro])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN
KEY([IDCurasLocalizacion])
REFERENCES [dbo].[Dim_CurasLocalizacion] ([IDCurasLocalizacion])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN
KEY([IDCurasEstadio])
REFERENCES [dbo].[Dim_CurasEstadio] ([IDCurasEstadio])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN KEY([IDResidente])
REFERENCES [dbo].[Dim_Residente] ([IDResidente])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN KEY([IDTiempo])
REFERENCES [dbo].[Dim_Tiempo] ([IDTiempo])
GO
```

```
ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN KEY([IDTipologia])
REFERENCES [dbo].[Dim_Tipologia] ([IDTipologia])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN KEY([IDÚlceras])
REFERENCES [dbo].[Dim_Úlceras] ([IDÚlceras])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN
KEY([IDÚlcerasTipos])
REFERENCES [dbo].[Dim_ÚlcerasTipos] ([IDÚlcerasTipos])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN
KEY([IDÚlcerasProcedencias])
REFERENCES [dbo].[Dim_ÚlcerasProcedencias] ([IDÚlcerasProcedencias])
GO

ALTER TABLE [dbo].[Fact_ÚlcerasCuras]  WITH CHECK ADD FOREIGN
KEY([IDÚlceraCurada])
REFERENCES [dbo].[Dim_ÚlceraCurada] ([IDÚlceraCurada])
GO
```

Todas estas tareas de ejecución de transacción SQL deben utilizar la conexión correspondiente al almacén de datos.

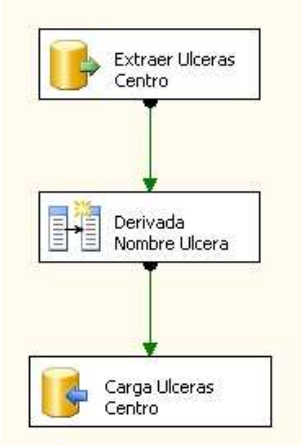
Paquete de Dimensiones

Si se observa lo enunciado durante la modificación del paquete de almacén, el lector puede caer en la cuenta de que la única dimensión adicional que debe rellenarse para la implementación de este indicador es la dimensión úlcera. Por ello, se añadirá al flujo de control una nueva tarea de flujo de datos, siendo el resultado el mostrado a continuación:



Como ya se hizo con las demás dimensiones rellenadas mediante este paquete, se va a estudiar el flujo de datos correspondiente al relleno de la dimensión úlcera, ya que las demás dimensiones adicionales se rellenan mediante el método de tablas maestras, o automáticamente.

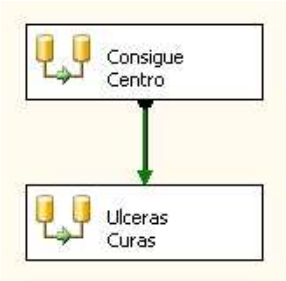
Flujo de Datos de Relleno de la Dimensión Úlcera



Como se puede observar, la carga de esta dimensión es bien similar a la carga de la dimensión de los residentes, excepto que hay un paso intermedio en el cual se crea el nombre de la úlcera. Dicho nombre estará compuesto por la palabra "Úlcera", un guión, y el código unificado de dicha úlcera. De este modo, ya solo queda la carga de los datos en el almacén.

Paquete del Hecho Curas de Úlceras

Una vez rellenas las dimensiones, se va a estudiar el paquete que rellena el hecho. Para ello, se cuenta con el siguiente flujo de control:



Como se puede observar, se cuenta con dos tareas de flujo de datos al igual que ocurría con el paquete del hecho de visitas familiares: el primero de ellos para la obtención del centro, y un segundo encargado del relleno de la tabla de hechos.

Ya que el flujo de datos correspondiente a la obtención del centro ya está explicado en puntos anteriores, se va a proseguir directamente con el flujo correspondiente al relleno de la tabla de hechos. Se recuerda que dicho flujo se encargará de realizar todas las operaciones de ETL necesarias para obtener el indicador propiamente dicho.

Como ya ocurría con el indicador de visitas de familiares, el flujo de datos correspondiente al relleno de la tabla de hechos es de un tamaño considerable, por lo que se va a intentar desagregar en tramos de modo que el lector pueda entenderlo sin ningún problema. El flujo entero se puede ver en la Figura 32.

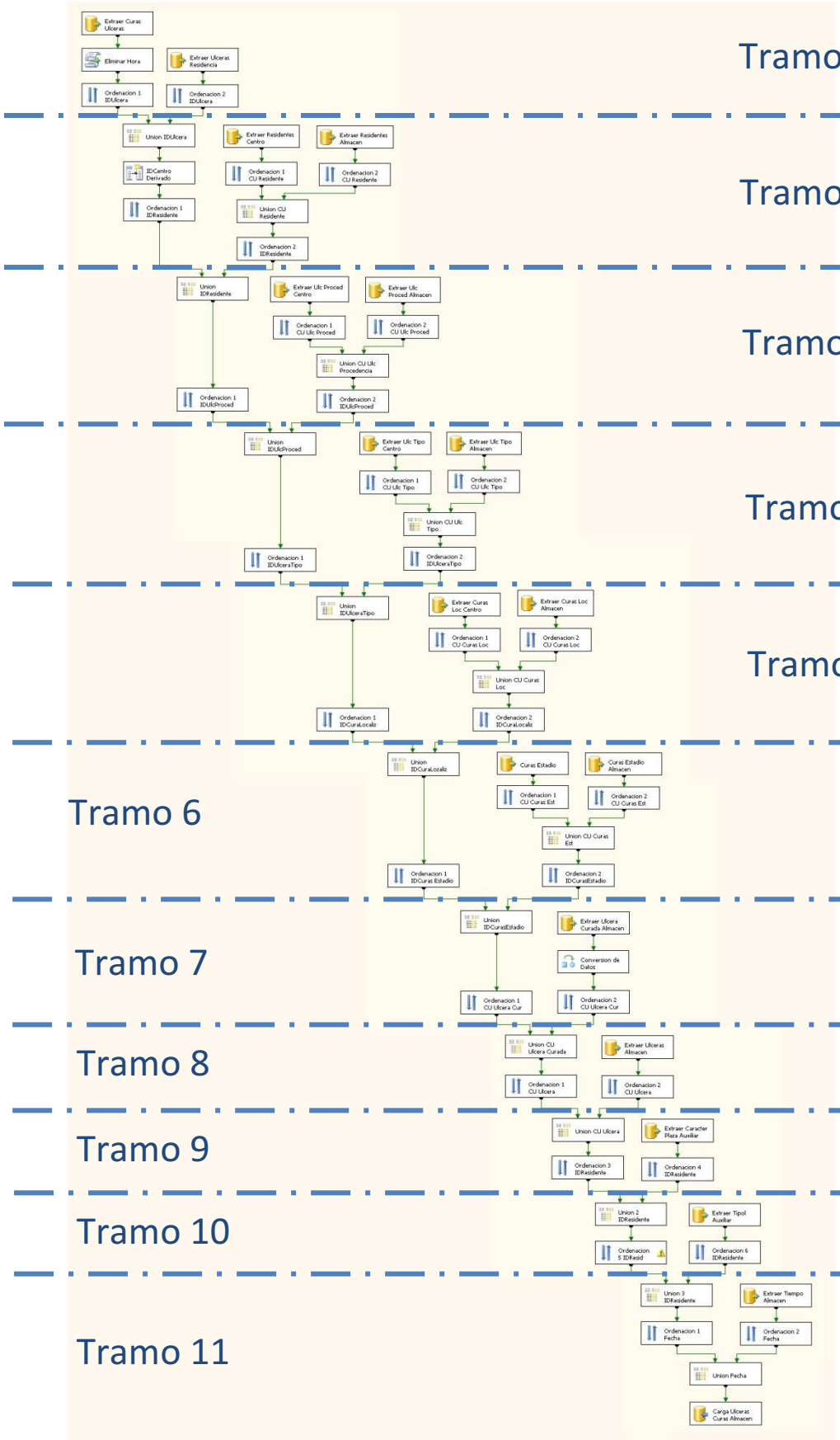


Figura 32.- Flujo de datos de Úlceras Curas

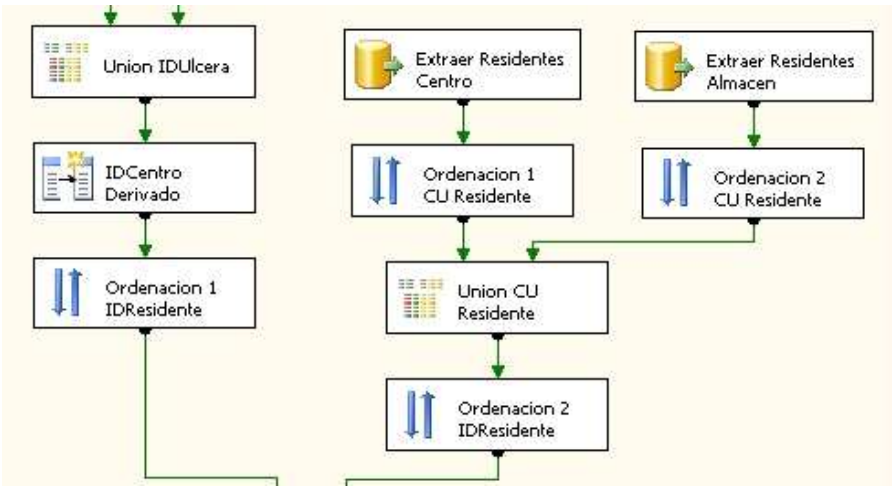
TRAMO 1

En un principio, se cuenta con dos ramales que comienzan con una extracción de datos. En el izquierdo, se extraen las curas de úlceras de la tabla correspondiente del centro. Dicha tabla contiene datos exclusivos de la cura de la úlcera, como pueden ser el estadio de la misma o su tamaño.

Después de esto, y al igual que ocurrió con las visitas de familiares, hay que eliminar la hora del campo fecha de la cura de la úlcera. Para ello, se utiliza el mismo componente script que se explicó en el desarrollo del anterior indicador de gestión. Para finalizar, se realiza una ordenación por el campo de la úlcera.

Por otro lado, en el ramal de la derecha se extraen todas las úlceras de la base de datos del centro, al objeto de conseguir su código unificado e identificadores que indican características de la úlcera que no cambian, como su procedencia o su tipo, además del código unificado de la misma. Una vez hecho esto, se realiza la ordenación por el campo identificador de la úlcera para una posterior asociación y unión.

TRAMO 2



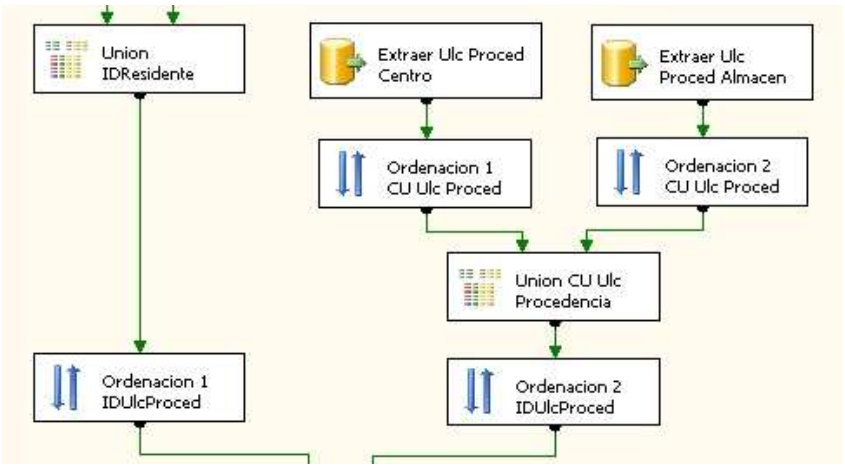
A pesar de no tener dos ramales claramente diferenciados, se va a intentar ilustrar el comportamiento en primer lugar de los tres componentes de más a la izquierda de la imagen, para a continuación explicar el resto.

En estos primeros tres componentes más alineados al margen izquierdo, la primera operación es la asociación y unión (INNER JOIN) por el campo identificador de úlcera del centro, utilizando para ello las salidas del primer tramo. De este modo, y con los campos extraídos en las fuentes de datos del tramo uno, ya tendremos tuplas formadas por la fecha de la cura, el identificador de residente en el centro, el identificador de la úlcera en el centro y su código unificado, los identificadores del tipo de úlcera, procedencia, localización y estadio en el centro, además del tamaño de la misma y un campo booleano que indica si la úlcera está curada o no (mediante 1 o 0 respectivamente).

A estos resultados se les añade un nuevo campo, que será el identificador del centro en el almacén, cuyo valor será el obtenido mediante una variable en el primer flujo de datos del flujo de control del paquete. Por último, solo falta ordenar los registros por el identificador del residente en el almacén.

Mientras tanto, los demás componentes tratan de conseguir tuplas que contengan el identificador del residente en el centro con su correspondiente identificador del almacén. Para ello, hace uso de dos componentes de extracción de datos (uno del centro y otro del almacén), realizando una ordenación por el código unificado, y posteriormente realizando una asociación y unión (INNER JOIN) por dicho campo. Una vez hecho esto, solo hay que ordenar los registros por el identificador de residente en el almacén para una posterior asociación y unión.

TRAMO 3

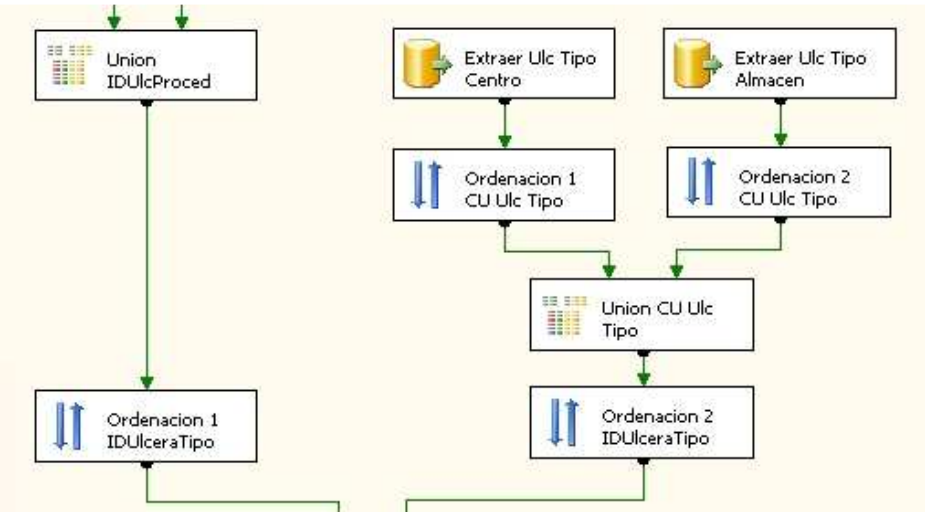


A partir de este momento, se van a tener tramos muy similares, con dos componentes completamente alineados al margen izquierdo, y un conjunto restante de componentes restante que se dedica a preparar datos para una nueva asociación y unión con los datos correspondientes a los componentes del margen izquierdo.

En este caso, los componentes del margen izquierdo se dedican, en primer lugar de realizar una asociación y unión (INNER JOIN) entre las salidas resultantes del tramo anterior, resultando tuplas con la fecha de la cura, el identificador de residente en el almacén (dato nuevo), el identificador del centro en el almacén, el identificador de la úlcera en el centro y su código unificado, los identificadores del tipo de úlcera, procedencia, localización y estadio en el centro, además del tamaño de la misma y el campo booleano de úlcera curada. Una vez realizada esta tarea, se realiza un ordenación por el campo identificador de la procedencia de la úlcera, para una posterior asociación y unión.

Por otro lado, el resto de componentes se encargan de la obtención de tuplas con el identificador de la procedencia de la úlcera en el centro y en el almacén, al objeto de conseguir un nuevo dato para el flujo de datos. Para ello, mediante dos componentes de extracción de datos, dos ordenaciones y una posterior asociación y unión por la izquierda (LEFT JOIN). Una vez hecho, tan solo queda ordenar por el identificador de la procedencia de la úlcera en el centro.

TRAMO 4

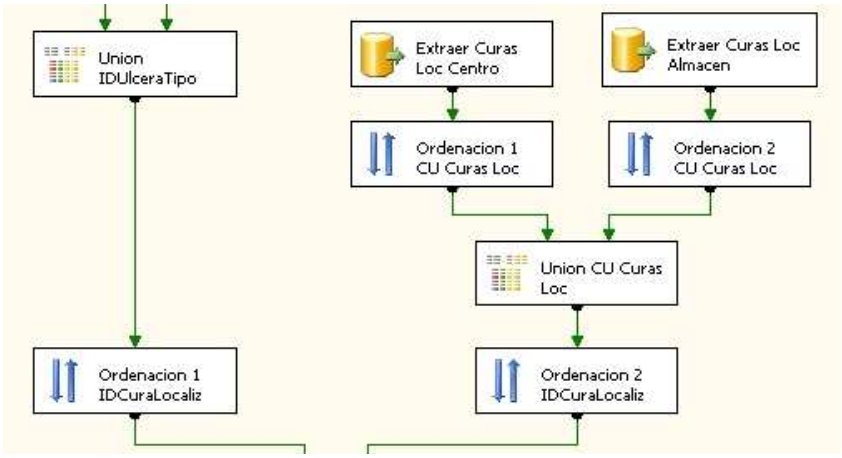


Al igual que ocurría en el caso anterior, en este tramo estudiaremos primero los componentes alineados al margen izquierdo y después el resto de componentes.

Para comenzar, en los componentes de la izquierda se realiza la asociación y unión por la izquierda (LEFT JOIN) por el campo identificador de procedencia de la úlcera, al objeto de conseguir su identificador en el almacén. El resultado de esta asociación serian tuplas con la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el centro y su código unificado, los identificadores del tipo de úlcera, localización y estadio en el centro, el identificador de la procedencia de la úlcera en el almacén (dato nuevo), además del tamaño de la misma y el campo booleano correspondiente a úlcera curada. Por último, se realiza una ordenación por el identificador de tipo de úlcera, que será el nuevo campo que se obtendrá en el próximo tramo.

Mientras tanto, el resto de los componentes se dedican a la obtención de registros con pares de valores con el identificador del tipo de la úlcera en el centro y en el almacén, al objeto de realizar una posterior asociación y unión. Para ello, obtiene los datos mediante dos componentes de extracción de las respectivas tablas de centro y almacén, ordenando por el código unificado y realizando una posterior asociación y unión por la izquierda (LEFT JOIN), ordenando los registros resultantes por el identificador del tipo de úlcera en el centro.

TRAMO 5

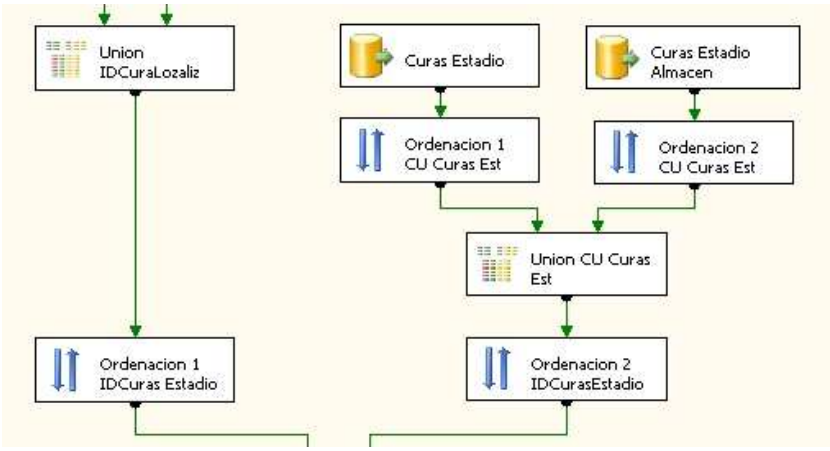


Al igual que en los dos anteriores tramos, se van a explicar primero los componentes del margen izquierdo para proseguir con el resto de componentes.

El primer componente del margen izquierdo, y mediante las dos salidas del tramo anterior, realiza una asociación y unión por la izquierda (LEFT JOIN) mediante el identificador del tipo de úlcera del centro. Con esto se consiguen tuplas consistentes en la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el centro y su código unificado, los identificador la localización y estadio en el centro, los identificadores de la procedencia y del tipo de la úlcera en el almacén (dato nuevo), además del tamaño de la misma y el campo booleano que informa si la úlcera está curada. Una vez realizado esto, se ordenan los registros resultantes por el identificador de la localización de la cura.

Por otro lado, los demás componentes se encargan de la consecución de pares con el identificador de la localización de cura del centro y del almacén. Esto se efectúa mediante dos componentes de obtención de datos (una consulta al centro y otra al almacén), la ordenación por el código unificado de la localización de la cura en ambos grupos de registros, para realizar una posterior asociación y unión por la izquierda (LEFT JOIN), obteniendo ya el resultado deseado, quedando solo realizar una ordenación por el identificador del centro para una posterior asociación y unión.

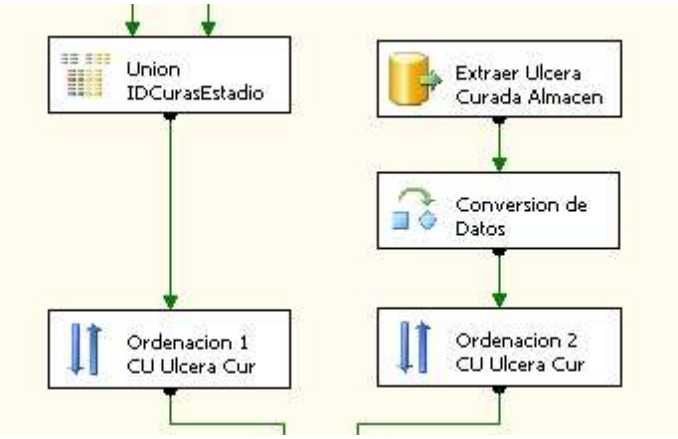
TRAMO 6



Como se lleva haciendo desde hace tres tramos, se va a empezar analizando los dos componentes más alineados a la izquierda. El primero de estos componentes pegados al margen izquierdo se dedica a hacer la unión y asociación por la izquierda (LEFT JOIN) con las salidas pertenecientes al tramo anterior. Con esto, se obtienen registros que contienen la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el centro y su código unificado, el identificador del estadio de la cura en el centro, los identificadores de la procedencia, del tipo de la úlcera y de la localización de la cura en el almacén (dato nuevo), además del tamaño de la misma y el campo booleano que informa si la úlcera está curada. Una vez realizado esto, se ordenan los registros resultantes por el identificador de la localización de la cura. Mientras, en el segundo componente se realiza la pertinente ordenación por el identificador del estadio de la cura al objeto de una posterior asociación y unión.

Mientras tanto, los demás componentes, al igual que en tramos anteriores, se dedican a conseguir tuplas con la correspondencia entre el identificador del estadio de la cura en el centro y en el almacén. Para ello, y como siempre, hace uso de dos componentes de extracción de datos a centro y almacén, ordenando los registros por el código unificado del estadio de la cura y realizando una asociación y unión por la izquierda (LEFT JOIN) de dichos registros. Consecuentemente, terminará realizando una ordenación por el identificador correspondiente al centro.

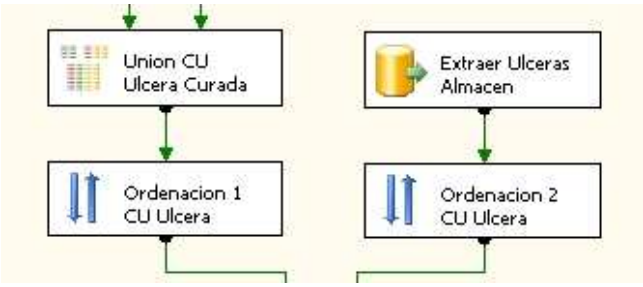
TRAMO 7



El primer paso de los componentes alineados al margen izquierdo es una asociación y unión por la izquierda (LEFT JOIN) de los registros de las salidas del tramo anterior. Con ello, se consiguen tuplas con la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el centro y su código unificado, los identificadores de la procedencia, del tipo de la úlcera, de la localización y del estadio de la cura en el almacén (dato nuevo), además del tamaño de la misma y el campo booleano que informa si la úlcera está curada. Como último paso, no hay más que ordenar los campos por el campo booleano de la úlcera curada (que coincide con los valores del código unificado del almacén).

Mientras tanto, se ve como la cantidad del resto de componentes se ha reducido: esto es debido a que ya no tenemos que realizar más consultas a tablas que han sido rellenadas mediante el método de tablas maestras (los cuatro casos anteriores). Aquí únicamente se ha de extraer los datos del almacén de la dimensión de úlcera curada, y se organizan por el campo código unificado, para una asociación y unión en el siguiente tramo.

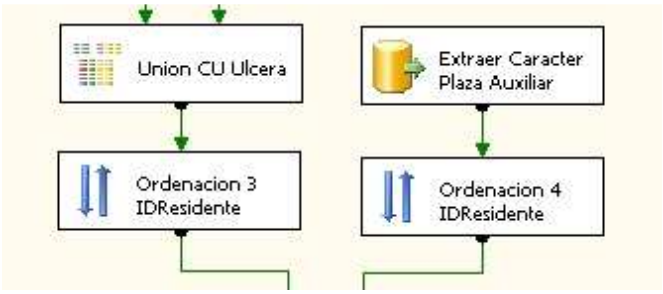
TRAMO 8



Para comenzar, se realiza la asociación y unión (INNER JOIN, ya que en este caso todos los valores deben asociarse con alguno ya que la úlcera no puede estar en mas estados que curada o no curada) consiguiendo un conjunto de registros que contienen la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el centro y su código unificado, los identificadores de la procedencia, del tipo de la úlcera, de la localización y del estadio de la cura en el almacén, además del tamaño de la misma y el campo booleano que informa si la úlcera está curada (pero esta vez ya listo para su introducción en el almacén). A estos registros se les aplica una ordenación por el código unificado de la úlcera (que ya fue obtenido en el primer tramo).

Mientras tanto, en el ramal derecho se extraen todos los datos de la tabla de úlceras del centro y realizando la misma ordenación que la anunciada anteriormente, ya que en el próximo paso se realizará una asociación y unión.

TRAMO 9



En el ramal izquierdo se comienza haciendo la unión y asociación por la izquierda (LEFT JOIN) con los registros de las salidas del tramo anterior, teniendo ya los datos correspondientes a la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el almacén (dato nuevo), los identificadores de la procedencia, del tipo de la úlcera, de la localización y del estadio de la cura en el almacén, además del tamaño de la misma y el campo booleano que informa si la úlcera está curada. El siguiente paso es ordenar dichos registros por el identificador de residente del almacén y la fecha, al objeto de conseguir en los siguientes pasos el carácter plaza y la tipología del residente en dicha fecha.

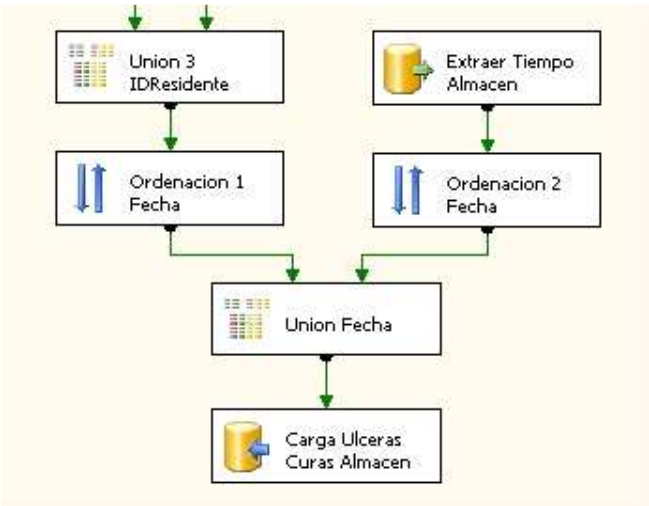
Para ello, el ramal derecho se encarga de realizar una consulta para extraer todos los registros de la tabla auxiliar de carácter plaza, realizando la misma ordenación que en el ramal izquierdo para ahora realizar la asociación y unión.

TRAMO 10

En primer lugar, en el ramal izquierdo se realiza la asociación y unión por la izquierda (LEFT JOIN) al objeto de conseguir el carácter plaza del residente en una fecha determinada. Con esto, las tuplas obtenidas ya contienen, además de este carácter plaza, la fecha de la cura, el identificador de residente en el almacén, el identificador del centro en el almacén, el identificador de la úlcera en el almacén, los identificadores de la procedencia, del tipo de la úlcera, de la localización y del estadio de la cura en el almacén, además del tamaño de la misma y el campo booleano que informa si la úlcera está curada. Como bien se muestra en la figura, se muestra en el componente de ordenación una advertencia, debido a que esta ordenación es innecesaria, ya que es la misma que la realizada en el tramo anterior y los registros ya están ordenados por el identificador del residente y la fecha.

Mientras, por el ramal derecho, y al igual que en el tramo anterior, se realiza la extracción de la tabla auxiliar de tipologías y se ordenan los registros por el identificador de residente y fecha para la posterior asociación y unión.

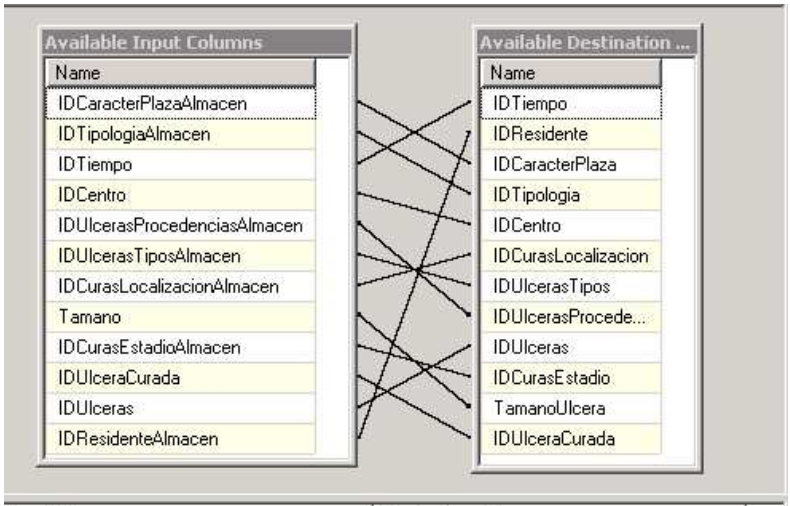
TRAMO 11



En esta penúltima unión y asociación por la izquierda (LEFT JOIN) ya se logra obtener todos los identificadores necesarios del almacén, excepto la fecha. Por ello se realiza la ordenación por dicho campo al objeto de hacer la unión y asociación (INNER JOIN) con los registros correspondientes a los extraídos en el ramal derecho.

En este ramal derecho, y al igual que se ha venido haciendo en pasos anteriores, se ha realizado una consulta a la tabla del almacén de la dimensión tiempo obteniendo todos los registros y ordenando por el campo fecha.

Con esta unión y asociación, ya se dispone de todos los registros formateados de modo que ya pueden ser introducidos en el almacén. Para ello, se ha de creado un componente de destino de datos con la correspondencia reflejada en la Figura 33.



Input Column	Destination Column
IDTiempo	IDTiempo
IDResidenteAlmacen	IDResidente
IDCaracterPlazaAlmacen	IDCaracterPlaza
IDTipologiaAlmacen	IDTipologia
IDCentro	IDCentro
IDCurasLocalizacionAlmacen	IDCurasLocalizacion
IDUlcerasTiposAlmacen	IDUlcerasTipos
IDUlcerasProcedenciasAlmacen	IDUlcerasProcedencias
IDUlceras	IDUlceras
IDCurasEstadioAlmacen	IDCurasEstadio
Tamano	TamanoUlcera
IDUlceraCurada	IDUlceraCurada

Figura 33.- Correspondencia del destino de datos para el indicador curas de úlceras

Como se ha podido observar con el desarrollo de este indicador, el hecho de crear un indicador nuevo mucho más grande que el de visitas de familiares afecta en distinta manera a los dos métodos: mientras que para el método de assemblies se aumenta la complejidad, en el método de desarrollo de Integration Services tan solo se aumenta el tamaño del paquete mediante la adición de mas componentes.

Capítulo 5. Explotación de la Información

Una vez realizado el proceso de ETL, se deben estudiar las posibilidades para explotar el almacén de datos que se ha construido anteriormente.

Para ello, se va a empezar explicando el funcionamiento de Analysis Services de Microsoft, el cual permitirá crear un cubo de datos para su posterior explotación mediante distintos métodos. Todo esto se ilustrará mediante la creación de un cubo.

En el presente documento se estudiarán tres métodos de explotación de cubos: mediante hojas de Microsoft Excel, dashboards de Microsoft SharePoint y dashboards de Microsoft PerformancePoint. Para cada uno de dichos métodos de explotación se implementará un ejemplo para que el usuario pueda ver los resultados finales de cada uno de ellos.

5.1.- Tecnologías Utilizadas

Además de Visual Studio 2008 para la creación de los cubos de Analysis Services, en el siguiente punto se van a utilizar dos nuevas tecnologías, Microsoft Excel y Microsoft PerformancePoint.

La información para el desarrollo de los siguiente dos apartados de la memoria ha sido extraida de las referencias de la bibliografía [TEC5] [TEC6] [TEC7] [TEC8].

5.1.1.- Microsoft Excel 2007

En todo paquete de ofimática no puede faltar una de las aplicaciones por excelencia que es la hoja de cálculo: una de las más conocidas y potentes es Microsoft Excel.

Su primera aparición dentro de este paquete fue en 1993, y tal fue la relevancia de esta aplicación que las interfaces de los otros dos componentes del paquete (Word y PowerPoint) fueron rediseñadas para ser acordes a la interfaz de la hoja de cálculo.

A día de hoy, la última versión de Microsoft Excel es la doceava, también conocida como Microsoft Excel 2007.

Funcionalidad Ofrecida

De sobra es conocida para cualquier usuario del mundo de la informática el sinfín de aplicaciones que se le puede dar a la hoja de cálculo Microsoft Excel: cálculos estadísticos, financieros, contables (tanto con vista de informe como con gráficos), creación de formularios…

Pero en este punto se va a hablar a que tipos de objetos puede conectarse Microsoft Excel, permitiendo la importación de datos de estos objetos en la hoja de cálculo:

- Base de datos de Microsoft Access: la información de una tabla, consulta o formulario de Access puede ser importada en la hoja de calculo
- Pagina web: carga la información contenida dentro de los ficheros XML de una página web.
- Texto plano: un texto que ha sido previamente formateado puede ser cargado dentro de la hoja de cálculo.
- Fichero XML: carga la información contenida en un fichero XML
- SQL Server: permite la conexión a una base de datos contenida dentro de este motor, introduciendo los datos de la misma dentro de la hoja de cálculo.
- Analysis Services: conecta la hoja de cálculo con cubos creados mediante Analysis Services.

Ya que en los siguientes puntos de la memoria se desarrollaran cubos de Analysis Services, es de vital importancia este último tipo de conexión, ya que permitirá al usuario la explotación de la información de un cubo de datos multidimensional dentro de una aplicación conocida por el usuario y de una manera amigable y sencilla.

Elección de Microsoft Excel 2007 y Posibles Alternativas

La elección de Microsoft Excel como herramienta de explotación de cubos de datos viene obligada por el contexto: hoy día, no hay ninguna otra aplicación comercial que pueda realizar conexiones directas a cubos creados para Analysis Services (aunque si se indaga por Internet, se pueden encontrar plugins para aplicaciones gratuitas que permite la creación de este tipo de conexiones).

Pero por otra parte, debido a lo extendido que se halla el paquete de ofimática Microsoft Office, se cree conveniente la elección de Microsoft Excel como herramienta de explotación, ya que es posible que los usuarios deban manejar estos informes, por lo que para ellos el encontrarse en un entorno que les es familiar e incluso han utilizado más de una vez siempre es una ventaja.

5.1.2.- Microsoft PerformancePoint Server 2007

Microsoft Office PerformancePoint Server 2007 es un conjunto de aplicaciones para la gestión del rendimiento, diseñadas con el objetivo de ayudar a las organizaciones a mejorar el rendimiento tanto operacional, como financiero, a través de todos los departamentos y todos los niveles de la organización.

Básicamente, con PerformancePoint Server se podrá monitorizar el progreso del rendimiento dentro de una empresa, analizar qué es lo que está ocurriendo y porqué y planificar desde la presupuestación, con el objetivo de generar informes de gestión.

Áreas Funcionales de Microsoft Performance Point Server 2007

Como se podrá adivinar después de leer los párrafos anteriores, las áreas funcionales de Microsoft Performance Point Server 2007 son dos, y a su vez hay distintas aplicaciones dentro de ellas para llevar a cabo el cometido de cada una.

- Planificación: permite crear los planes de acción que realizara la empresa.
    - Planning Server: Permite acceder a vistas de datos sobre los que están basados planes de acción previamente creados.
    - Planning Business Modeler: Interfaz para la creación de modelos de negocio, que son unidades básicas de almacenamiento de datos en Planning Server.
- Monitorización y Análisis: permite ver el estado de los distintos planes de acción creados con anterioridad.
    - Monitoring Server: Monitorizacion de los planes de acción y modelos de negocio creados con las aplicaciones de modificación.
    - Dashboard Designer: permite la creación de dashboards que, una vez implementados, serán exportados a Microsoft Office SharePoint Server. Dichos dashboards mejoran los que se pueden crear dentro de esta última aplicación. Esta es la única aplicación de todo el paquete que será usada dentro de la memoria.

Elección de Microsoft Office PerformancePoint 2007 y Posibles Alternativas

Con el objetivo de ofrecer a los usuarios una información mucho más visual que la ofrecida por los informes de Microsoft Excel o los dashboards de SharePoint, se observó que esta aplicación de Microsoft permitía la creación de otros dashboards mucho más atractivos e interactivos.

Además, si el gestor de contenidos de ámbito empresarial elegido es Microsoft Office Sharepoint Server 2007, no existen más alternativas a la hora del diseño de dashboards, por lo que la elección si se quieren ofrecer unos mejores dashboards a los clientes es obligatoria.

Cabe destacar que en la versión 2010 de Microsoft Office SharePoint Server vendrá integrada toda la funcionalidad incluida en Microsoft Office PerformancePoint Server, denotando esto la utilidad de esta última aplicación.

5.2.- Desarrollo de un Cubo de Analysis Services

Durante este apartado se va a ver cómo crear un cubo de datos de un indicador de gestión, para que posteriormente sea explotado por cada uno de los métodos de explotación anunciados anteriormente.

Dicho cubo será el correspondiente a un indicador que no es uno de los implementados como ejemplo, sino uno nuevo. Dicho indicador se llama caídas, y su definición es la siguiente:

"Número total de caídas ocurridas de los distintos residentes (adjuntando el carácter de la plaza y la tipología que tienen en el momento en el que ocurre la cura), indicando al centro al cual pertenecen y organizadas por día, mes, trimestre o año. Además, se requerirá información del lugar donde se ha producido la caída."

Este mismo indicador será el utilizado para ilustrar el desarrollo de este punto y del siguiente apartado 3, por lo que es recomendable la comprensión de la definición antes de proseguir con la lectura del documento (si se desea ver la estructura del almacén de datos resultante, está adjuntada en la figura 53).

5.2.1.- Creación del Proyecto

Para comenzar, se debe crear un proyecto de Visual Studio del tipo "Analysis Services Project". Una vez introducido el nombre y la carpeta contenedora del mismo, se muestra la interfaz mostrada en la Figura 34.

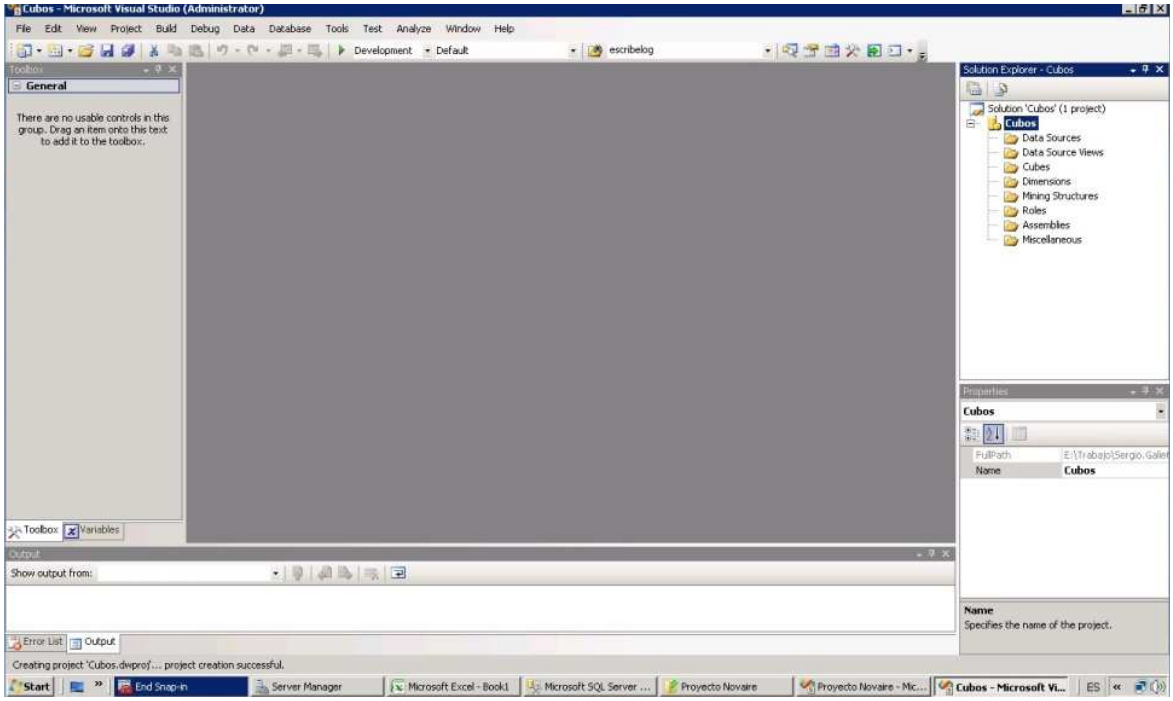


Figura 34.- Interfaz de un proyecto de Analysis Services

En la misma se pueden observar todos los componentes de la misma. En la parte izquierda de la pantalla se ve la típica caja de herramientas de todo proyecto de Visual Studio, pero que en este tipo de proyectos es innecesaria y puede ser cerrada si el usuario desea.

En la parte derecha se observa el también típico explorador de soluciones de Visual Studio, donde ya se ven los tipos de objetos que van a tener este tipo de proyectos (a destacar "Data Sources", "Data Source Views", "Cubes" y "Dimensions", que serán los utilizados en la presenta memoria).

En la parte central se postrarán una vez abiertos cada uno de estos tipos de componentes, ofreciendo pestañas dependientes del tipo de objeto que se esté modificando.

5.2.2.- Creación de Data Source

Una vez vistos todos los componentes de la interfaz, se va a comenzar a crear un cubo de datos para explotar la información introducida en el almacén de datos.

Para ello, el primer paso es la creación de un Data Source. Para ello, no hay más que ir a los componentes de este tipo en el explorador de la solución, y crear una nueva seleccionando la opción correspondiente al hacer clic con el botón derecho del ratón.

El resultado de realizar dicha acción es una ventana en la cual están las conexiones ODC creadas por el usuario, con la posibilidad de crear una nueva. Como en el caso del ejemplo se va a utilizar una nueva conexión, se hace clic sobre el botón "New", dando como resultado una nueva ventana en la cual hay que configurar una conexión de la fuente de datos, cuya interfaz es idéntica a la que se observó en los proyectos anteriormente implementados de Integration Services. Dicha ventana puede observarse en la Figura 35.

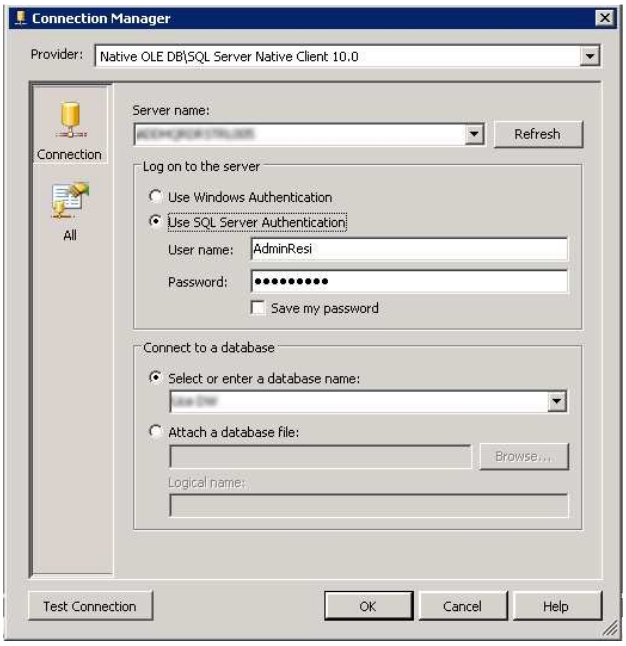


Figura 35.- Creación de conexión al almacén de datos

Se recuerda que en la misma hay que introducir el servidor de bases de datos donde está el almacén de datos, el nombre del mismo, y las credenciales de autenticación. Una vez esté correctamente configurada, se hace clic en "Ok", volviendo a la anterior pantalla de conexiones, estando seleccionada la creada por defecto, pudiendo proseguir haciendo clic sobre el botón "Next". Con ello aparece la ventana de la Figura 36.

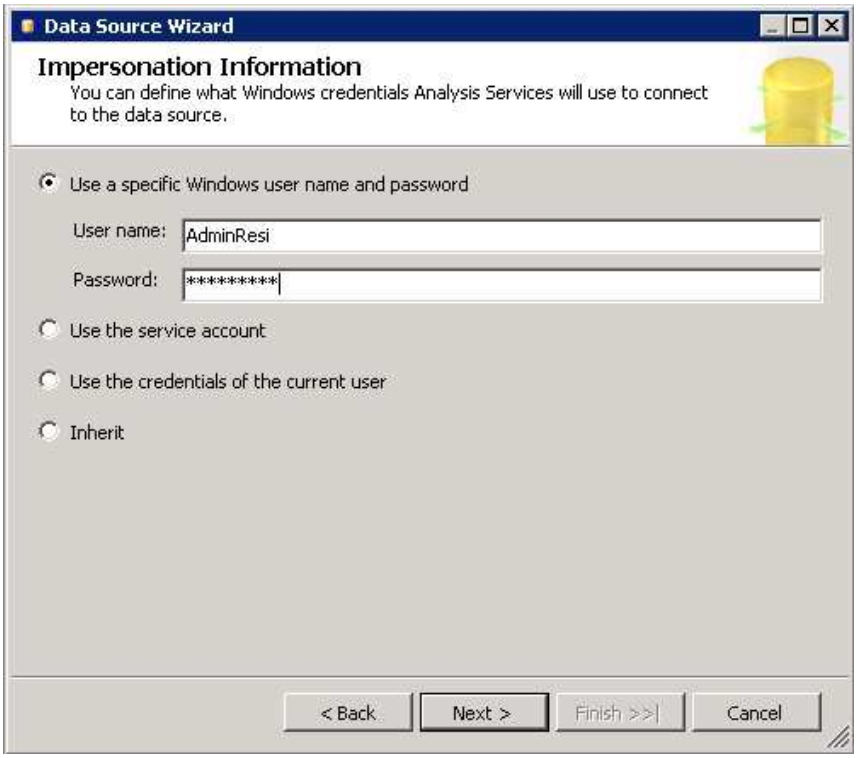


Figura 36.- Credenciales para la conexión a la base de datos

En ella se exige como va a conectarse Analysis Services a dicho almacén de datos, habiendo opciones para que el usuario rellene las credenciales, que herede dichas credenciales, que las herede, o que se utilice la cuenta de servicio. Es recomendable utilizar siempre la opción de utilizar la cuenta de servicio, y para proseguir, hacer clic sobre el botón "Next", dando como resultado la ventana representada en la Figura 37.

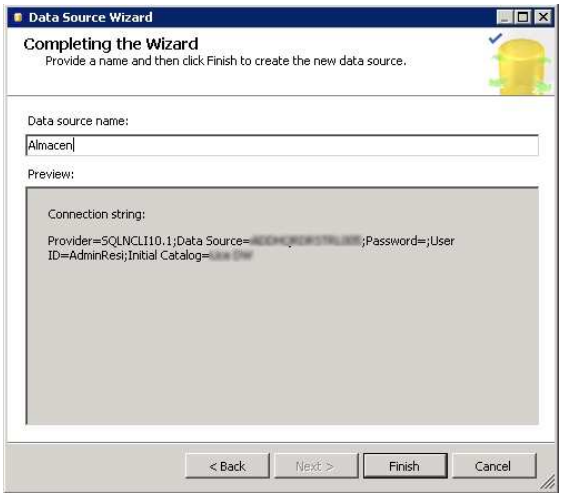


Figura 37.- Introducir el nombre de la conexión

En ella, se observa la cadena de conexión que se utilizará debiendo el usuario introducir un nombre para el Data Source. Con esto ya se tiene implementado dicho Data Source, por lo que se puede proseguir con la creación del Data Source View.

5.2.3.- Creación de Data Source View

Una vez creada la conexión a la fuente de datos, es obligada la creación de una o más vistas para saber a qué tablas va a poder acceder dicha conexión. A este nuevo componente se le denomina Data Source View. Para ello, no hay más que pinchar sobre el nombre de este tipo de objetos en el explorador de soluciones y haciendo clic sobre la opción “New”.

El primer paso consiste en seleccionar un Data Source ya creada o crear una nueva para utilizarla en la creación del Data Source View. En este ejemplo, como ya se ha creado dicha Data Source, no hace falta más que seleccionarlo de la lista, como se observa en la Figura 38. Para continuar con la creacion, no hay más que hacer clic en el botón “Next”

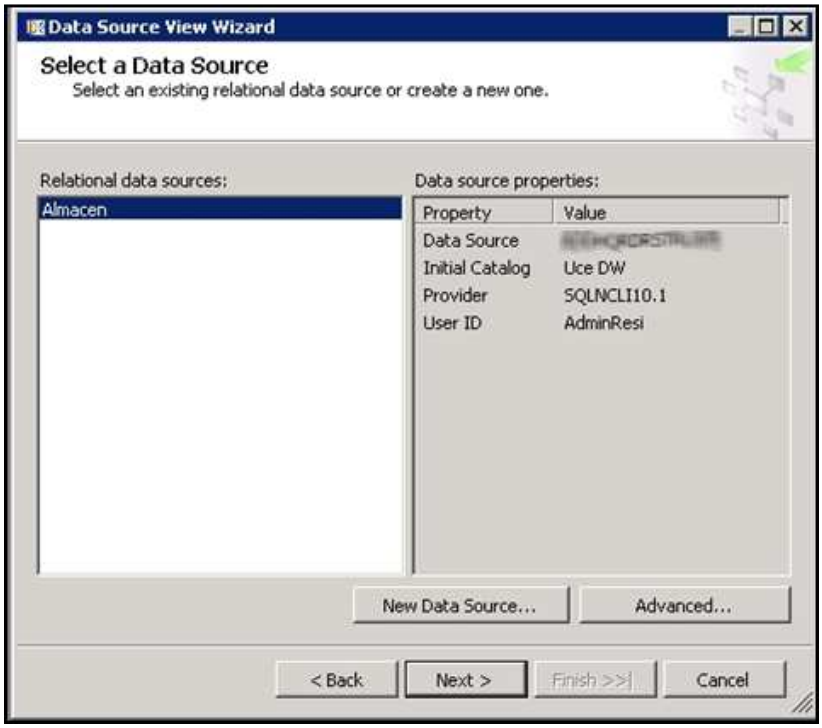


Figura 38.- Selección del Data Source para la creación del Data Source View

En el siguiente paso se ha de seleccionar aquellas tablas del almacén de datos que corresponden con el cubo que se desea crear. Para el caso de la creacion del cubo Caídas, se ha de seleccionar la tabla de hechos “Fact_ResiCaidas”, y las tablas de dimensiones “Dim_Residente”, “Dim_Tiempo”, “Dim_Centro”, “Dim_CaracterPlaza”, “Dim Tipologia” y “Dim_LugarCaidas”, como se muestra en la Figura 39.

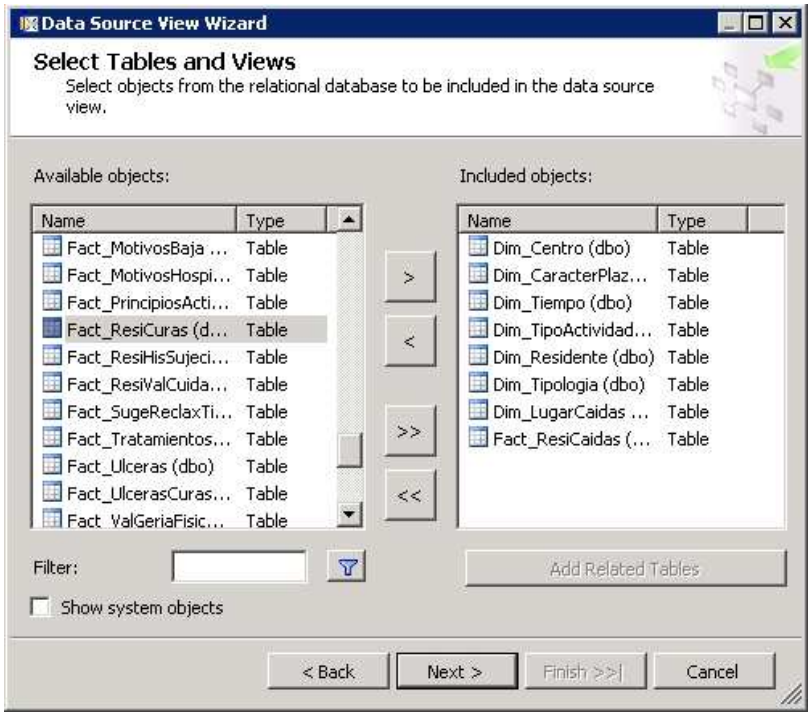


Figura 39.- Tablas a añadir en Data Source View

Una vez seleccionadas las tablas, se prosigue haciendo clic sobre el botón “Next”, ofreciendo ahora una última pantalla de introducción del nombre de Data Source View. En el caso ejemplo, se denominará “VistaAlmacen”

Con esto, ya estará el Data Source View creado, el cual se podría reutilizar para la creacion de nuevos cubos añadiendo las tablas correspondientes a los mismos (en este caso solo se seleccionaron aquellas que tenían que ver con el indicador Caídas)

5.2.4.- Creación del Cubo de Datos

Una vez creada la fuente de datos, y una vista de la misma, ya se puede hacer el uso de las dos al objeto de crear un cubo de datos. Para ello, hay que seleccionar en el explorador de la solución la carpeta “Cubes”, hacer clic con el botón derecho y seleccionar la opción “New”. La primera interfaz que surge después de estos pasos es la mostrada en la Figura 40.

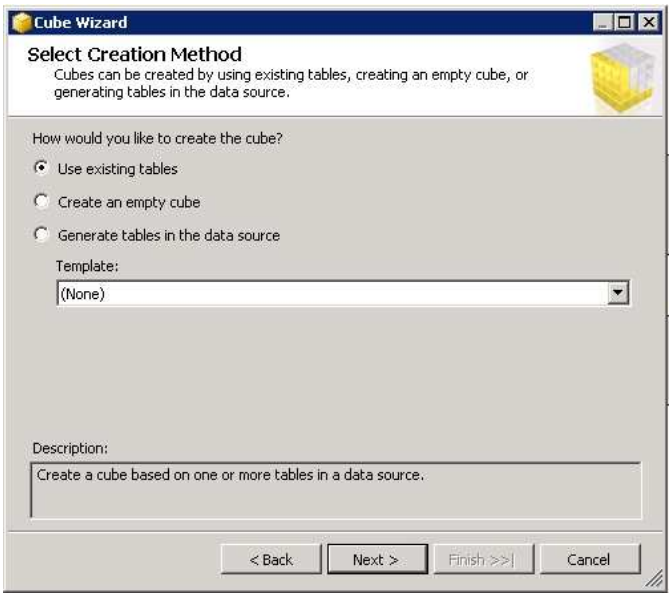


Figura 40.- Primera interfaz de creación de cubos

En ella se selecciona como se ha de crear el cubo: usando tablas existentes, creando un cubo vacio, o generando tablas en la fuente de datos. En el caso del ejemplo, se selecciona la opción “Use existing table” y se hace clic sobre “Next”, apareciendo la interfaz de la Figura 41.

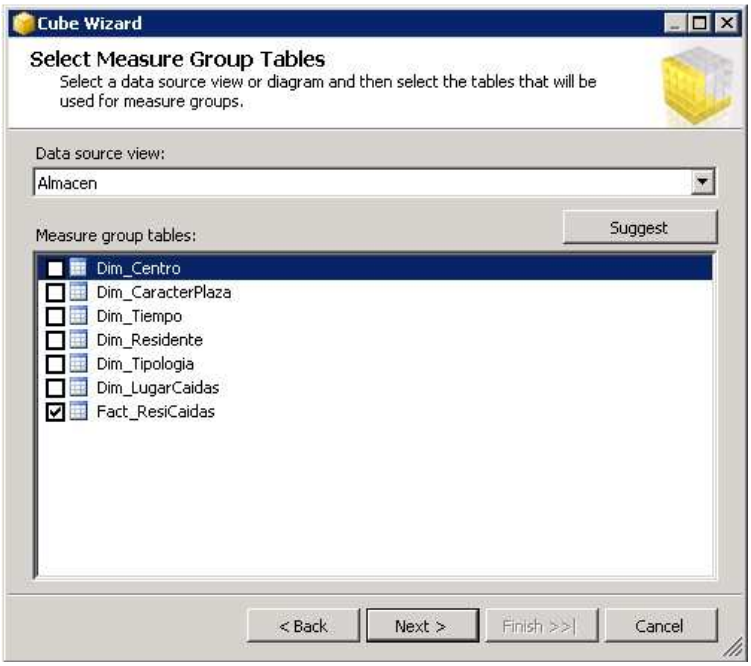


Figura 41.- Interfaz de selección de la tabla de hechos

En ella, hay que seleccionar el Data Source View previamente creado, apareciendo una vez hecho esto todas las tablas agregadas al mismo. De todas ellas, se ha de seleccionar aquella que corresponde a la tabla de hechos, en el caso del ejemplo “Fact_ResiCaidas”. Para continuar, hacer clic sobre “Next”, apareciendo la interfaz de la Figura 42.

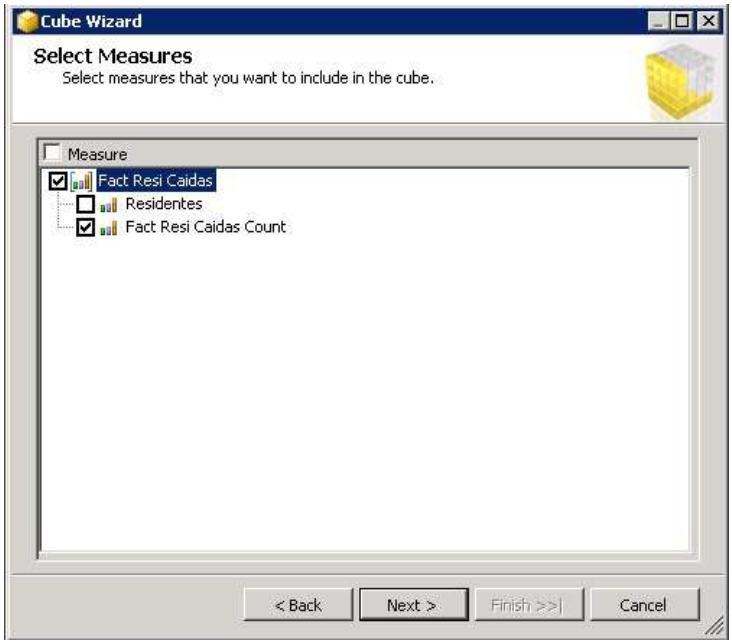


Figura 42.- Selección de las medidas del cubo

En ella hay que seleccionar las distintas medidas que se desean crear en el cubo. En el ejemplo, como tan solo queremos una cuenta de registros, se selecciona aquella acabada en "Count". En el caso que existiesen más candidatas a medidas en la tabla de hechos (atributos no relacionados con otras tablas) estos aparecerían en esta interfaz. Para continuar, hacer clic sobre "Next", apareciendo la interfaz de la Figura 43.

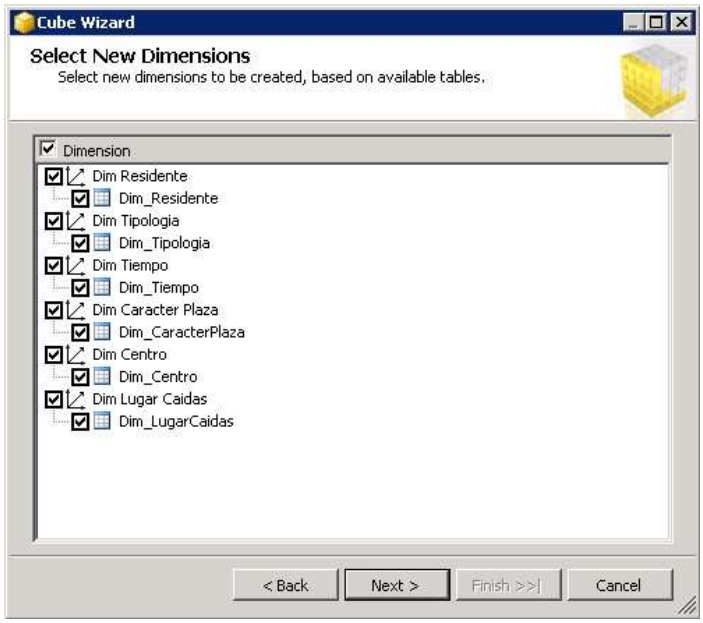


Figura 43.- Selección de las dimensiones del cubo

En este paso se crearán las dimensiones correspondientes al cubo, cogiendo por defecto el asistente todas aquellas tablas que están relacionadas con la tabla de hechos, debiendo el usuario seleccionar aquellas que realmente se correspondan a dimensiones. En el caso de ejemplo, todas ellas son válidas, por lo que se hace clic sobre "Next", apareciendo la última interfaz, mostrada en la Figura 44.

Figura 44.- Introducción del nombre del cubo

En ella no hay más que introducir el nombre del cubo a crear, que será añadido a la base de datos de cubos que se crea en Microsoft SQL Server y que tendrá el nombre del proyecto de Analysis Services creado.

Además de la creación del cubo de datos, se habrán creado todas las dimensiones relacionadas con dicho cubo, todas con el prefijo “Dim” delante, los cuales sería conveniente renombrar para que el resultado final sea mucho más uniforme. Al final, la forma debe ser la de la Figura 45.

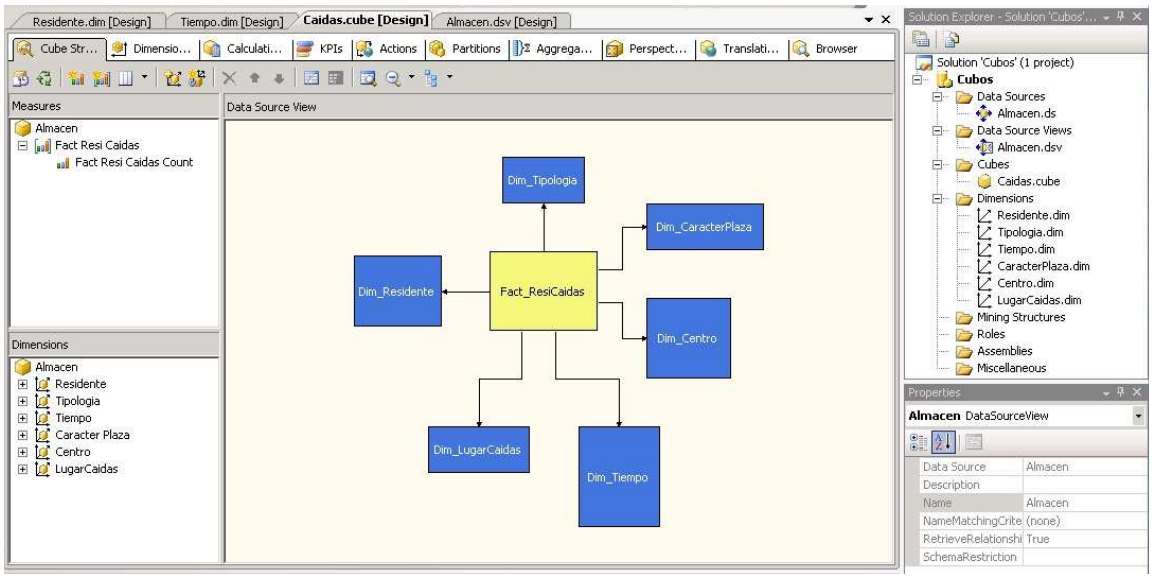


Figura 45.- Captura final de los pasos de creación del cubo

5.2.5.- Modificaciones al Cubo y Dimensiones Creados

Una vez realizados estos pasos hay que realizar unas cuantas modificaciones tanto a los cubos como a las dimensiones al objeto de que la posterior explotación sea correcta.

Modificación en la Dimensión Tiempo

Como ya se dijo en puntos anteriores, Analysis Services ofrece un tratamiento especial para determinadas dimensiones, y una de ellas es la dimensión tiempo. Para que la dimensión temporal del ejemplo sea reconocida como tal, hay que acudir a las propiedades del objeto, y en la propiedad "Type" seleccionar "Time", como se muestra en la Figura 46.

Figura 46.- Reconocimiento de la dimensión temporal

Una vez hecho esto, se deberán añadir a la estructura de la dimensión todos los campos de la tabla de dimensiones. Para ello, hay que acudir a la pestaña "Dimension Structure", y arrastrar desde "Data Source View" todos los campos al recuadro "Attributes", quedando como se muestra en la Figura 47.

Figura 47.- Atributos de la dimensión tiempo

Una vez hecho esto, hay que decirle al cubo de datos que elemento dentro de una dimensión temporal es cada atributo. Por ejemplo, si se selecciona el atributo "Año", hay que acudir a sus propiedades y en el atributo "Type" asignar la opción "Years". Esto se muestra en la Figura 48.

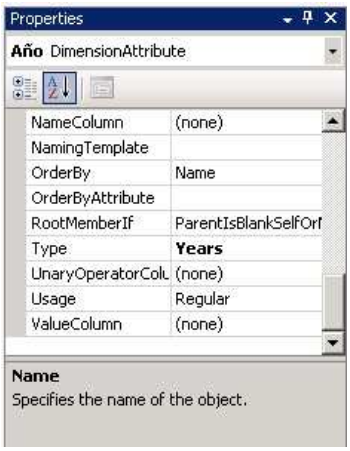


Figura 48.- Cambio del tipo de los campos

La correspondencia de los campos con tipos es la que se muestra en la siguiente tabla:

Campo	Tipo
Año	Years
Fecha	Days
Mes	Months
Trimestre	Trimesters

Una vez hecho esto, se van a ocultar aquellos campos que no vayan a ser visibles en la explotación final. El primero de ellos es "IDTiempo", el cual sí que es un índice, pero no debe ser visible, por ello hay que seleccionar en su atributo "AttributeHierarchyVisible" a "False", como se muestra en la Figura 49.

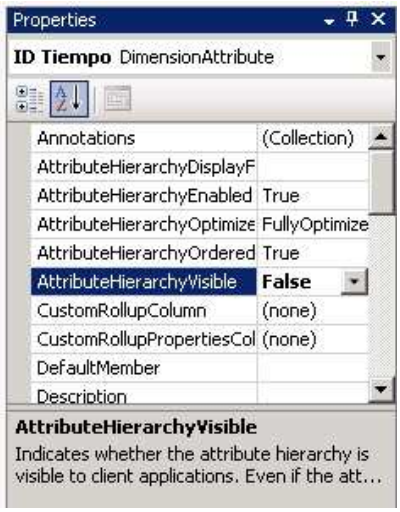


Figura 49.-Cambio de los atributos de IDTiempo

Además, se tienen otra serie de campos que no deben ser índices (al menos no de la dimensión) y tampoco deben ser visibles. Estos atributos son "Mes ID", "Num Dia Semana" y "Semana", cuyos atributos "AttributeHierarchyEnabled" y "AttributeHierarchyVisible" a "False", mientras que el atributo "AttributeHierarchyOptimized" debe tener el valor "Not Optimized". Esto se muestra en la Figura 50.

Figura 50.- Propiedades de distintos campos de la dimensión tiempo

Una vez modificados todos los atributos de la dimensión, no queda más que crear las relaciones existentes en la misma para crear la jerarquía temporal. Para ello, hay que acudir a la pestaña "Attribute Relationship", eliminando las relaciones creadas por defecto y añadiendo el conjunto que se muestra en la Figura 51.

Figura 51.- Relaciones entre atributos de la dimensión tiempo

Todas estas relaciones dan como resultado el grafico mostrado en la Figura 52.

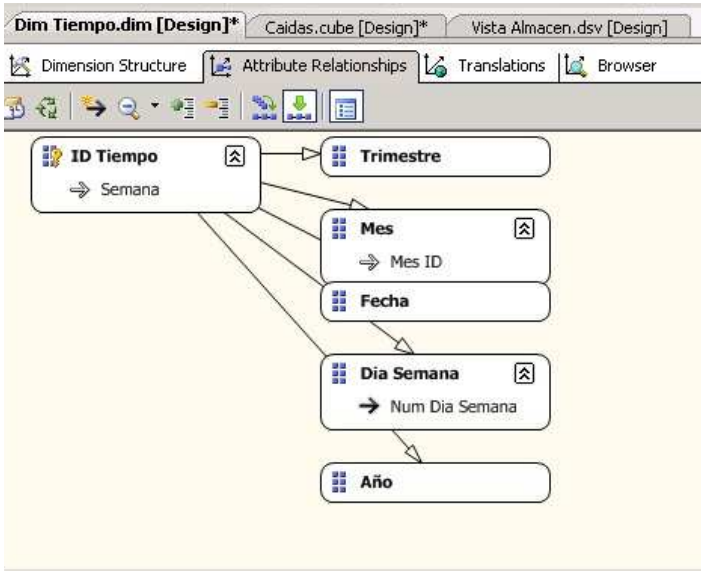


Figura 52.- Gráfico de las relaciones entre los atributos de la dimensión tiempo

Para que los campos “Mes” y “Dia Semana” sean ordenados respectivamente por los campos “IDMes” y “Num Dia Semana”, habrá que acudir a sus propiedades, cambiando el atributo “OrderBy” al valor “Attribute Key”, siendo el atributo “OrderByAttribute” el campo ordenador para cada uno de estos dos campos iniciales.

Como último paso, hay que crear la jerarquía personalizada. Para ello no hay más que volver a la pestaña “Dimension Structure”, y en el recuadro “Hierarchies” se crea una nueva jerarquía (botón derecho, “New”) y se arrastran los campos correspondientes a “Año”, “Trimestre”, “Mes” y “Fecha”, en dicho orden. Por último, se cambia el nombre de la misma a “Jerarquía Temporal”, siendo el resultado final el mostrado en la Figura 53.

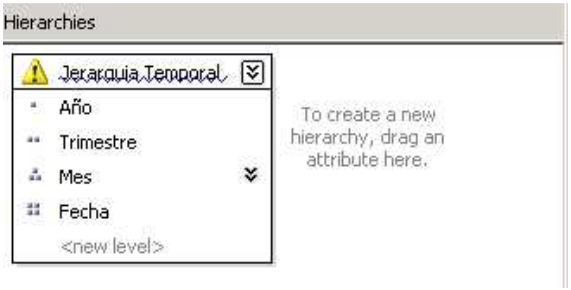


Figura 53.- Jerarquía temporal creada por el usuario

Con esto, la dimensión tiempo está totalmente finalizada y lista para explotar, pero aún quedan cambios por hacer en el resto de las dimensiones.

Modificaciones en las Demás Dimensiones

El resto de las dimensiones también necesitan una serie de modificaciones al objeto de poder ser explotadas con una calidad suficiente.

El primer paso para todas ellas seria, al igual que se hizo con la dimensión tiempo, arrastrar todos los campos de la tabla de dimensiones al recuadro “Attributes” en la pestaña “Dimension Structure”. Como ejemplo, se tiene la Figura 54, que muestra los atributos de la dimensión “LugarCaidas”.

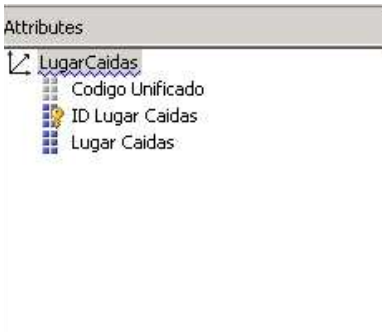


Figura 54.- Atributos de la dimensión “LugarCaidas”

Una vez hecho esto, hay que cambiar las propiedades de los atributos identificadores y códigos unificados para cada dimensión. En el caso de los primeros, y al igual que sucedió en la dimensión tiempo, hay que cambiar el valor de la propiedad “AttributeHierarchyVisible” a “False”.

Para los códigos unificados, hay que cambiar los mismos atributos de la dimensión tiempo que no eran ni índices ni debían estar visibles. Por ello, sus propiedades “AttributeHierarchyEnabled” y “AttributeHierarchyVisible” a “False”, mientras que el atributo “AttributeHierarchyOptimized” debe tener el valor “Not Optimized”.

Con todo esto, ya se tendrá listo para visualizar correctamente todos los datos del cubo de datos construido, pero falta una rectificación estética para algunas dimensiones.

Se recuerda que algunas columnas de la tabla de hechos podían tener valor “Null” (bien porque no se hayan definido los códigos unificados en los centros o bien porque LEFT JOINS hayan dado como fruto registros con valores nulos). Por ello, y para evitar que aparezca la palabra “Unknown” en los informes finales, se debe cambiar una propiedad de estas dimensiones.

En este caso, las dimensiones que sufren dichas modificaciones son las correspondientes a “Tipologia”, “CarácterPlaza” y “LugarCaidas”, acudiendo a sus propiedades para cambiar los atributos “UnknownMember” y “UnknownMemberName”, como se muestra en la Figura 55.

Figura 55.- Propiedad de miembros desconocidos para la dimensión “LugarCaidas”

Los valores a introducir, en este caso, serian los de “Visible” y “No Definido”, siendo este último el valor de aquellos miembros desconocidos.

Con todo esto, se daría por completada la modificación de las dimensiones, quedando tan solo la modificación del cubo propiamente dicho.

Modificación en el Cubo de Datos

Para acabar, se van a realizar una serie de modificaciones a las medidas del cubo de datos. Estas medidas se pueden ver en el recuadro “Measures” de la pestaña “Cube Structure” teniendo el cubo abierto.

Tan solo para mejorar el aspecto estético de las mismas, se va a cambiar el nombre del grupo de las medidas a “Caidas”, cambiando el nombre particular de la medida a “Número Caidas”, como se puede observar en la Figura 56.

Figura 56.- Modificación en la medida y en su grupo de medidas

Con esta pequeña modificación el cubo de datos creado ya está listo para su proceso y posterior explotación.

5.2.6.- Procesamiento del Cubo de Datos

Con el cubo y sus dimensiones ya correctamente formateadas, se va a proceder a procesarlo al objeto de inicializar una base de datos de cubos con el nombre del proyecto en el servidor de Analysis Services de Microsoft SQL Server.

Para ello, no hay más que hacer clic derecho en el proyecto y hacer clic sobre la opción emergente "Process", apareciendo la ventana de la Figura 57.

Figura 57.- Preparación para el procesamiento del cubo de datos

En ella se ofrece el objeto a procesar y algunas opciones que no son tan interesante como para ser estudiadas, por lo que no queda más que hacer clic sobre el botón “Run”, apareciendo una nueva ventana emergente que informa acerca del progreso del procesamiento. Si todo ha salido bien, aparecerá una ventana tal como la de la Figura 58.

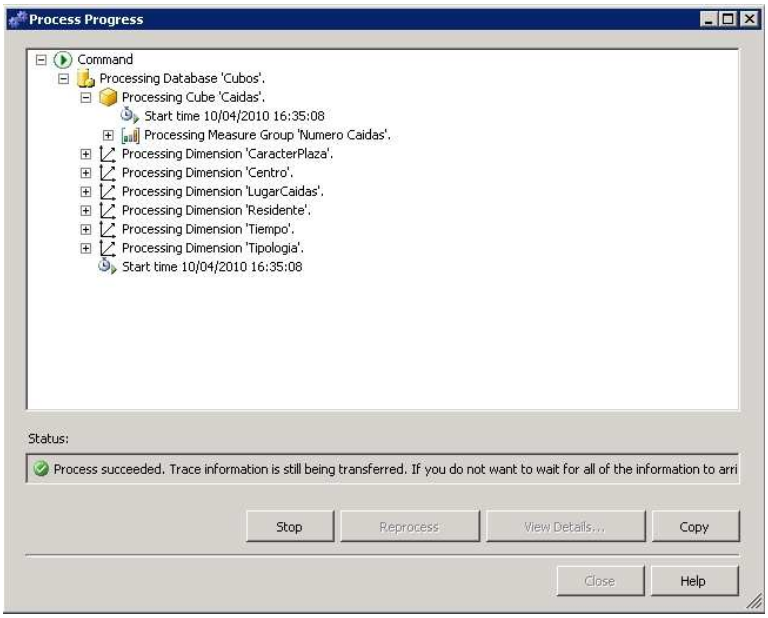


Figura 58.- Éxito en el procesamiento del cubo de datos

En ella se puede observar como todas las dimensiones y el cubo de datos han sido procesados correctamente, por lo que la elaboración del cubo ha sido un éxito y se puede comenzar a explotar inmediatamente.

5.3.- Explotación mediante Microsoft Excel

Una vez desarrollado el cubo de datos de Analysis Services se abre un abanico de posibilidades a la hora de explotar la información. Una de estas posibilidades, y quizás la más sencilla y útil se trata de la explotación de los cubos de datos mediante una hoja de Microsoft Excel.

En este punto se va mostrar cómo crear una hoja de Microsoft Excel conectada a un cubo de Analysis Services, y una vez cargada la información, que posibilidades nos ofrece dicha aplicación para explotarla idóneamente.

5.3.1.- Creación de la Conexión al Cubo de Analysis Services

Para comenzar a crear la conexión al cubo de Analysis Services se ha de partir de una hoja de Microsoft Excel vacía. Una vez creada, hay que acudir al menú "Data", hacer clic sobre el botón "From Other Sources", y hacer clic sobre la opción "From Analysis Services". Esto se refleja en la Figura 59.

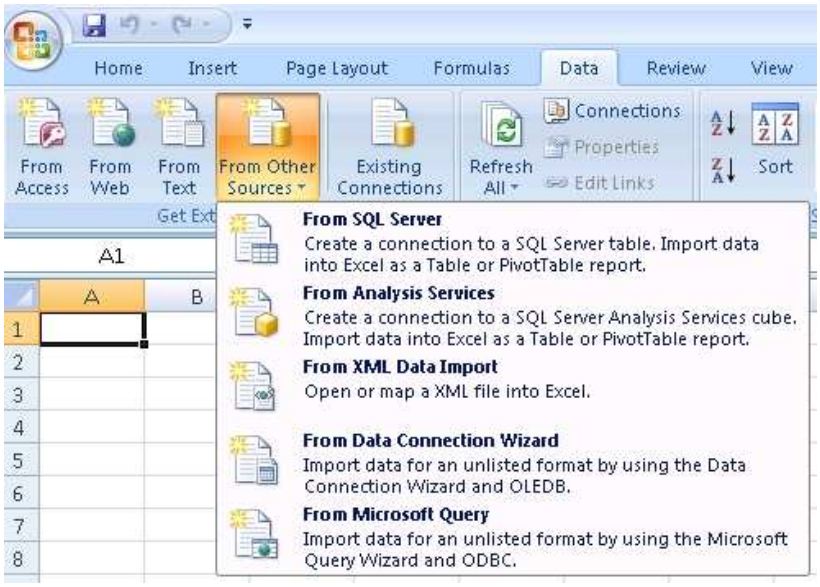


Figura 59.- Creación de la conexión a Analysis Services

A continuación, aparecerá una ventana como la de la Figura 60, en la cual se debe rellenar el nombre del servidor de bases de datos en el cual está contenido el cubo de Analysis Services, especificando las credenciales de conexión al mismo.

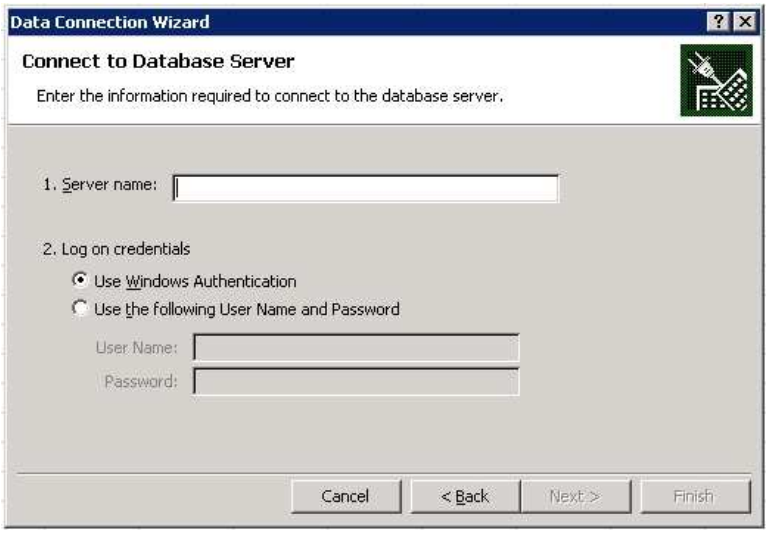


Figura 60.- Credenciales para la conexión a Analysis Services

En el paso siguiente aparecerá un desplegable en pantalla en el cual se observa todas y cada una de las bases de datos de cubos que hay disponibles en el servidor introducido con anterioridad. Cada base de datos de cubos tendrá sus cubos propios debiendo el usuario seleccionar uno de ellos para

proseguir. En este ejemplo, se selecciona el cubo denominado “Caidas”, como se muestra en la Figura 61, y se prosigue con el proceso.

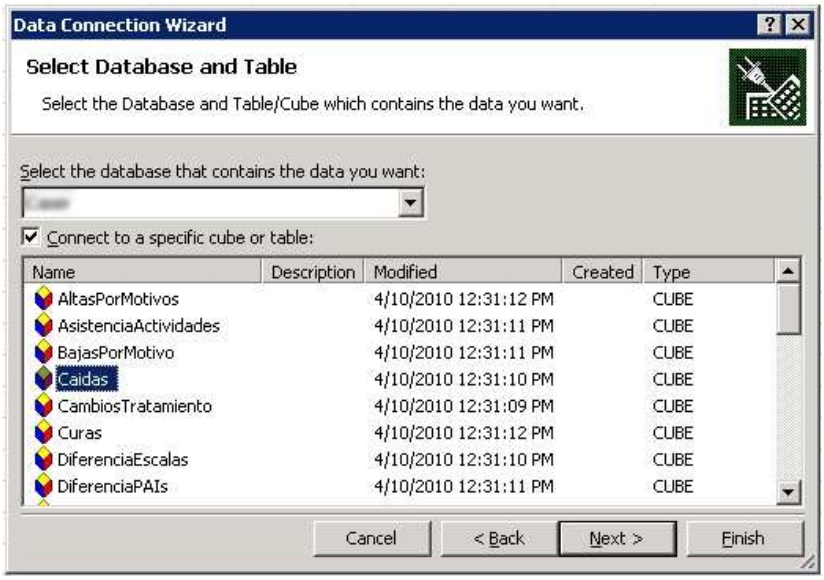


Figura 61.- Selección del cubo de datos al que realizar la conexión

Después de esto, se ha de guardar la conexión creada durante este proceso. Se puede dejar la ruta por defecto, o cambiarla mediante un explorador de rutas. Además, se puede añadir un nombre amistoso y una descripción a dicha conexión como se muestra en la Figura 62, al objeto de encontrarla con más facilidad a posteriori o reaprovecharla.

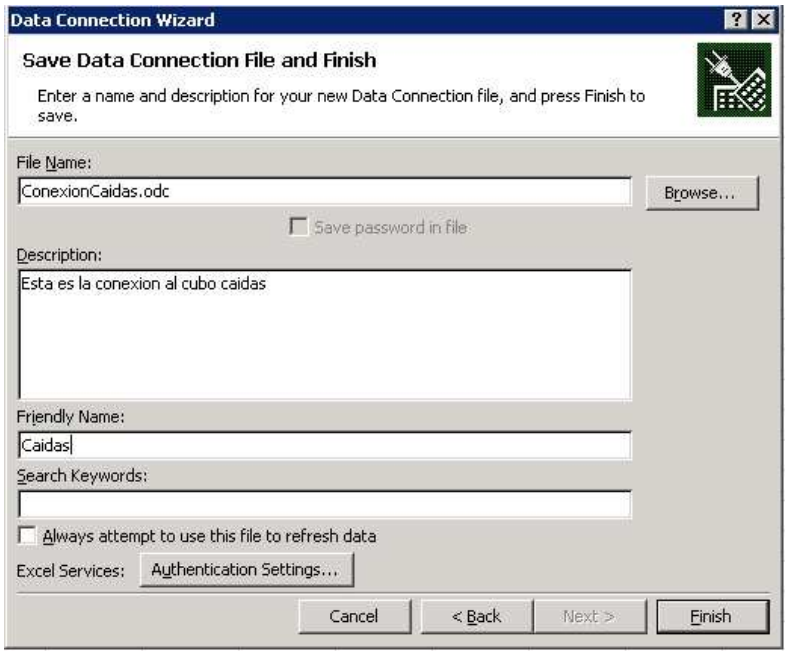


Figura 62.- Pantalla de propiedades de la conexión creada

Como último paso, hay que especificar en qué celda de la hoja de trabajo se van a importar los datos, además de especificar si se quieren en forma de “PivotTable”, “Pivot Chart / PivotTable Report”, e incluso crear la conexión sin adjuntar datos. En este ejemplo, se selecciona la opción “PivotTable Report”, como se muestra en la Figura 63.

Figura 63.- Opciones de importación de datos de Analysis Services

Con esto ya se habrá creado un Pivot Table, que se aprenderá a explotar en el siguiente punto

5.3.2.- Explotación de PivotTable

Una vez completados todos los pasos, se observa que la celda especificada ha sido rellenada con una figura predeterminada, así como el lado derecho de la aplicación, donde ha aparecido una especie de caja de herramientas, que servirá para realizar una explotación exhaustiva de la información, pudiéndose observar en la Figura 64.

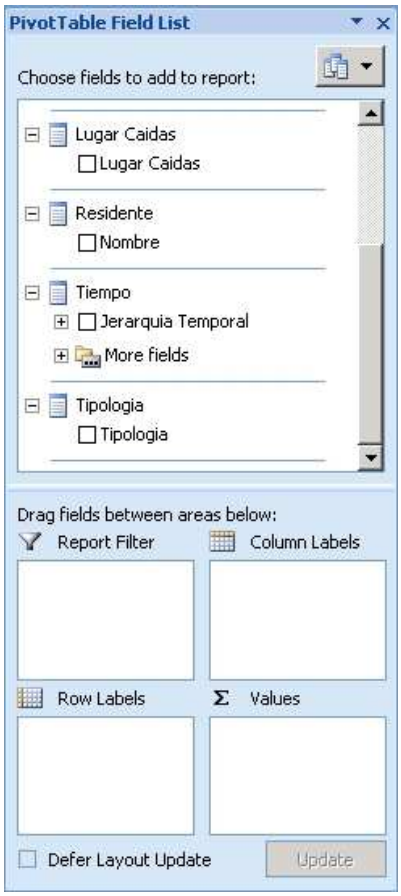


Figura 64.- Caja de herramientas del Pivot Table

Como se puede ver en la misma, ésta está dividida en cinco zonas. La primera de ellas nos indica que campos desean ser agregados al informe. En el caso de seleccionar alguna, se ve cómo cambian los recuadros situados más abajo, ya que este recuadro crea un informe por defecto con tan solo seleccionar los campos a ver, ofreciendo las demás zonas una mayor personalización.

Las cuatro zonas inferiores se denominan "Report Filter", que corresponden a los filtros añadidos al informe, "Column Labels", que son las dimensiones añadidas a las columnas, "Row Labels", dimensiones añadidas a las filas, y "Values", que son las medidas añadidas al informe.

Así por ejemplo, si se desea construir un informe en el cual se vean las caídas producidas por centro y tipología, podemos crear un informe en el cual los centros estén en las filas y las tipologías en las columnas, siendo los valores a mostrar los correspondientes a la medida de número de caídas. La configuración del informe sería la de la Figura 65.

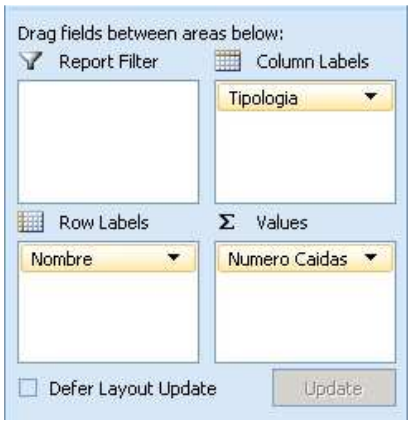


Figura 65.- Configuración informe 1 de Analysis Services

Y como resultado se obtendrá el informe representado en la siguiente Pivot Table:

Numero Caidas	**Column Labels**					
Row Labels	**Asistido**	**Semiasistido**	**Suprasistido**	**Valido**	**No Definido**	**Grand Total**
Residencia 1	2	219	21	405	2	649
Residencia 2		126		209	2	337
Residencia 3	1	5	1	2	29	38
Grand Total	**3**	**350**	**22**	**616**	**33**	**1024**

Cabe destacar que esta tabla es completamente manejable, así que si, por ejemplo, se cambian las filas por columnas (cambiando de manera rápida la configuración del informe), tendremos la misma Pivot Table pero pivotada, tal y como se muestra a continuación:

Numero Caidas	Column Labels			
Row Labels	**Residencia 1**	**Residencia 2**	**Residencia 3**	**Grand Total**
Asistido	2		1	3
Semiasistido	219	126	5	350
Suprasistido	21		1	22
Valido	405	209	2	616
No Definido	2	2	29	33
Grand Total	**649**	**337**	**38**	**1024**

Además, se puede introducir más de una dimensión dentro de las filas y de las columnas. Por ejemplo, si se desea un informe en el cual se desean ver las caídas producidas dependiendo del carácter plaza, la residencia en la que se ha producido, y el lugar de la caída, se puede construir mediante el configurador la forma en la que se muestra en la Figura 66.

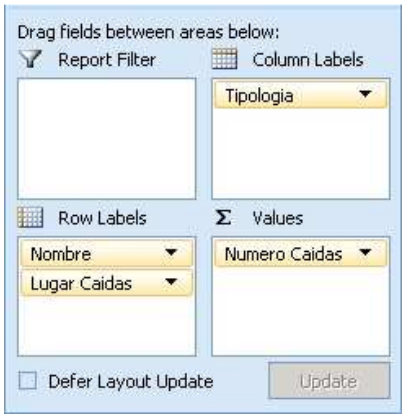


Figura 66.- Configuración informe 2 de Analysis Services

Dando como resultado el siguiente informe:

Numero Caidas	Column Labels					
Row Labels	**Asistido**	**Semiasistido**	**Suprasistido**	**Valido**	**No Definido**	**Grand Total**
- Residencia 1	**2**	**219**	**21**	**405**	**2**	**649**
Aseo		11	2	19		32
Aseo Habitacion		18	2	37		57
Comedor		8		34		42
Escalera				1		1
Exterior		2		1		3
Habitacion		36	4	72	1	113
Pasillo	2	123	11	191	1	328
Patio Exterior		1		1		2
Recepcion		15	1	43		59
Salon				1		1
No Definido		5	1	5		11
+ Residencia 2		**126**		**209**	**2**	**337**
+ Residencia 3	**1**	**5**	**1**	**2**	**29**	**38**
Grand Total	**3**	**350**	**22**	**616**	**33**	**1024**

Como se puede observar, al lado de cada una de las residencias hay un símbolo de desplegar y ocultar, y haciendo clic sobre él, ocultaremos y mostraremos la dimensión que esta anidada dentro (por la definición del informe, la dimensión Lugar Caídas esta anidada dentro de la dimensión tiempo)

5.3.3.- Uso de Filtros en PivotTable

Una vez visto como se pueden construir informes, es hora de agregar filtros a los mismos. De partida, se cuenta con el informe anterior en el cual se muestran las caídas producidas organizadas por la residencia y lugar caídas en las filas y por la tipología en las columnas.

Pero se da el caso que las caídas de más de tres años de antigüedad ya no son interesantes para el lector del informe, por lo que del mismo sobran un gran número de caídas. Al objeto de no añadir otra dimensión en filas o columnas, y mejorar la visibilidad del informe, se puede arrastrar el componente año de la dimensión tiempo al recuadro correspondiente a "Report Filter", obteniendo una configuración como la de la Figura 67.

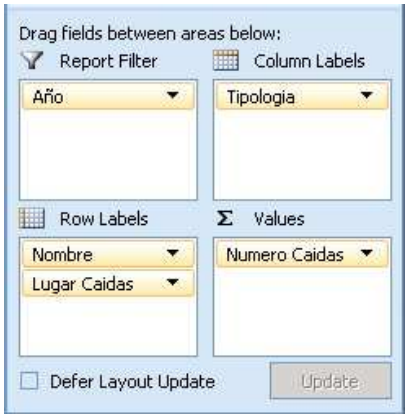


Figura 67.- Configuración informe 3 de Analysis Services

Se observa como unas celdas más arriba del Pivot Table se ha añadido un filtro, sobre el cual se puede hacer clic, surgiendo todos los años de la dimensión tiempo. Como en el ejemplo enunciado se desean las caídas de tres años, se necesita una selección múltiple, por lo que se marca la casillas "Multiple Selection" y aquellos años requerido, en este caso 2008, 2009 y 2010, como se muestra en la Figura 68.

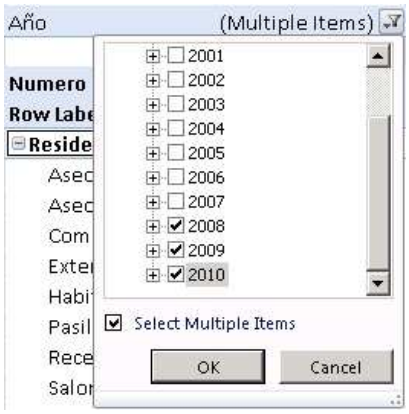


Figura 68.- Configuración del filtro para mostrar determinados años

Obteniendo como resultado el siguiente informe:

Numero Caidas	Column Labels					
Row Labels	Asistido	Semiasistido	Suprasistido	Valido	No Definido	Grand Total
- **Residencia 1**	**1**	**112**	**21**	**145**	**1**	**280**
Aseo		4	2	9		15
Aseo Habitacion		10	2	8		20
Comedor		4		13		17
Exterior		1				1
Habitacion		18	4	23	1	46
Pasillo	1	66	11	81		159
Recepcion		5	1	9		15
Salon				1		1
No Definido		4	1	1		6
+ Residencia 2		**33**		**80**		**113**
+ Residencia 3		**3**			**14**	**17**
Grand Total	**1**	**148**	**21**	**225**	**15**	**410**

Como se deduce, entre las opciones de configuración del informe y filtrado del mismo, se puede construir casi cualquier informe requerido, además de hacerlo de una forma bien sencilla.

5.3.4.- Inserción de Gráficos Pivot Chart

Como ya es conocido, Microsoft Excel permite realizar una gran cantidad de diagramas con los que ilustrar el contenido de sus celdas. Esta posibilidad también se ofrece a la hora de ilustrar gráficas de Pivot Tables, denominadas Pivot Charts, las cuales ofrecen una gran funcionalidad y un acabado esplendido.

Por ejemplo, se requiere hacer un informen en el cual se observar la evolución a lo largo de los años en cada una de las residencias. Para ello, se comienza construyendo el informe correspondiente, cuya configuración se ve reflejada en la Figura 69.

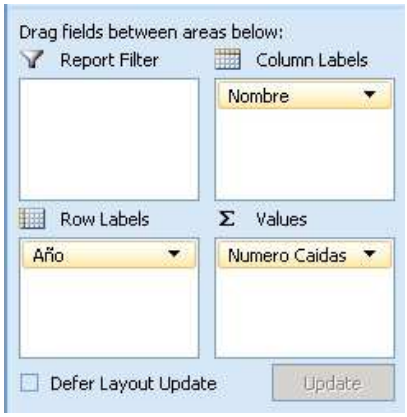


Figura 69.- Configuración informe 4 de Analysis Services

Una vez hecho esto, no hay más que ir a las opciones de la Pivot Table y hacer clic sobre el botón "Pivot Chart", tal y como se muestra en la Figura 70.

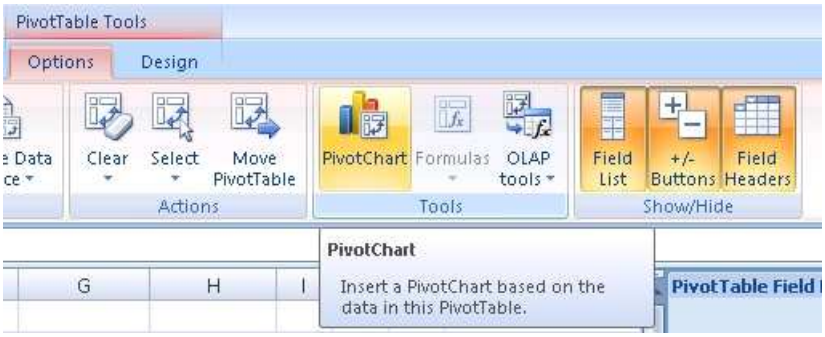


Figura 70.- Adición de Pivot Chart a la hoja de Microsoft Excel

Con ello, surge una ventana como la de la Figura 71, en la cual se puede seleccionar uno de los distintos tipos de gráficos de la librería de Microsoft Excel.

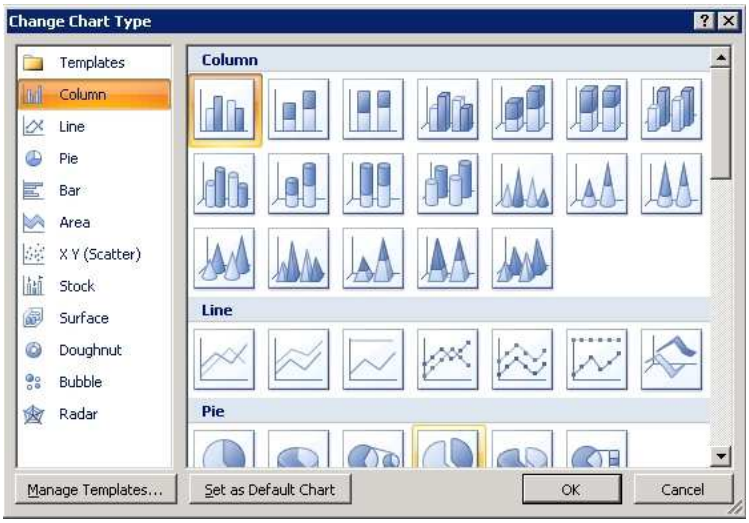


Figura 71.- Tipos de gráficos insertables en Microsoft Excel

En este caso, se seleccionará el diagrama de líneas con puntos, dando como resultado el grafico de la Figura 72.

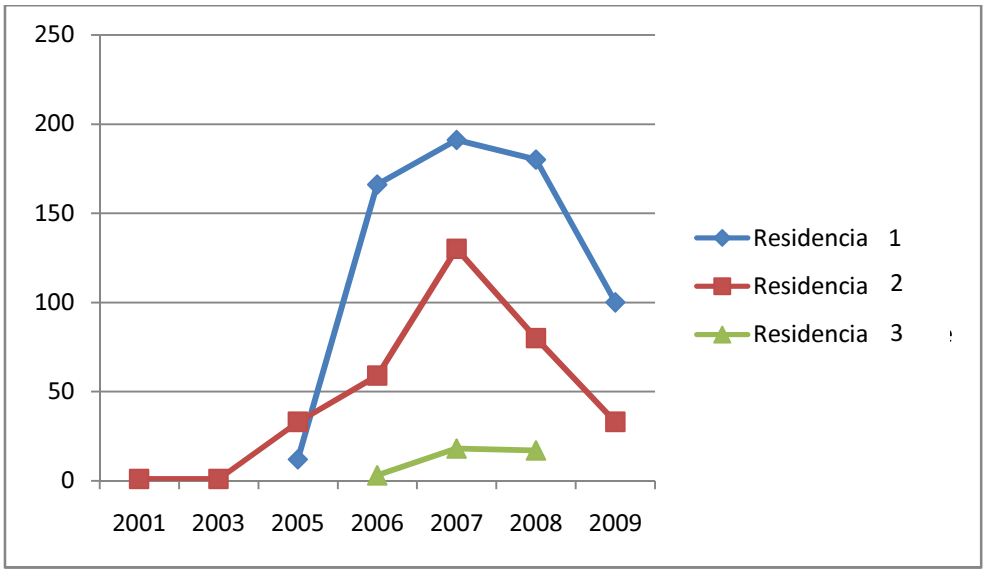


Figura 72.- Pivot Chart 1 de la conexión a Analysis Services

Si se observa, ahora se pueden cambiar las propiedades del Pivot Chart tal y como se haría con cualquier gráfico de Microsoft Excel, surgiendo solo el cambio de que, cuando se tenga el mismo seleccionado, se podrá modificar tal y como se hacía con la Pivot Table. Si se observa la Figura 73, se ve como los nombres de los recuadros contenedores han cambiado a otros propios de graficas.

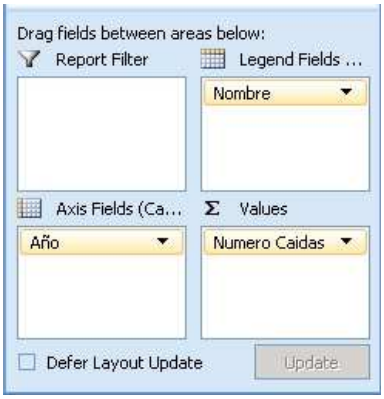


Figura 73.- Configuración Pivot Chart 1 de Analysis Services

Para visionar las posibilidades que ofrece esta fácil modificación, se parte del Pivot Chart anterior para construir uno diferente en el cual se quieren visionar las caídas en cada mes para cada residencia en un diagrama de barras, pero tan solo mostrando las caídas producidas en el año 2007

Para ello, no hay más que arrastrar y soltar los componentes para que queden de la forma mostrada en la Figura 74.

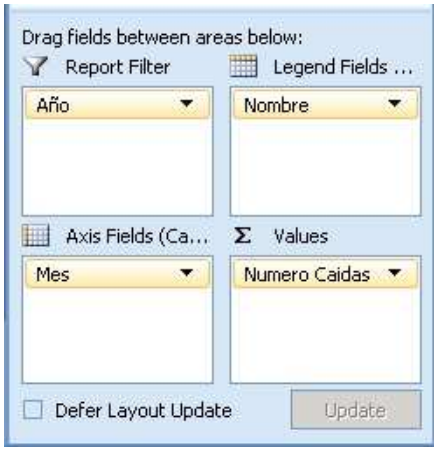


Figura 74.- Configuración Pivot Chart 2 de Analysis Services

Una vez hecho esto, solo hay que cambiar el tipo de gráfico. Para ello se acude a las opciones del Pivot Chart y se hace clic sobre la opción "Change Chart Type", seleccionando ahora uno por columnas.

Por último, no hay más que seleccionar tan solo el año 2007 en el filtro que se ha creado en una de las celdas de la hoja de cálculo, obteniendo un Pivot Chart tal como el mostrado en la Figura 75.

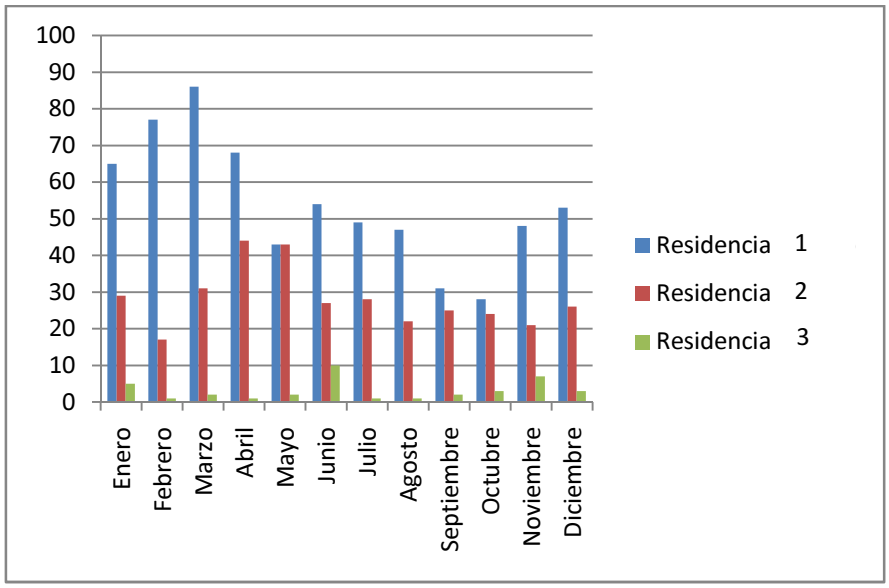


Figura 75.- Pivot Chart 2 de la conexión a Analysis Services

Como se deduce, las posibilidades de construcción son infinitas y tan solo dependen de cuan abierta sea la mente del usuario a la hora de construir los informes.

5.4.- Explotación Mediante Dashboards de SharePoint

Durante el desarrollo de este punto se va a explicar el desarrollo de un dashboard en Microsoft Office SharePoint Server 2007, basando su contenido en el indicador de caídas previamente descrito para el desarrollo del cubo de datos y la explotación mediante Microsoft Excel.

5.4.1.- Dashboards en SharePoint

Como ya se enunció en el capítulo 2 de la memoria, un dashboard o cuadro de mandos integral es una vista integrada / unificada que ilustra el estado y rendimiento en un ámbito determinado: residente, empleado, área, unidad, departamento, centro o grupo, etc.

Un dashboard en términos de interfaz de usuario, y en el ámbito de SharePoint, se implementa con scorecards y vistas de reports.

Un scorecard es una colección de KPIs (Key Performance Indicators). Un KPI es una medida que permite responder a una pregunta respecto al cumplimiento o no cumplimiento de un objetivo.

Cada KPI tiene metas (targets) y umbrales (thresholds). Los valores del KPI se comparan con las metas para evaluar el rendimiento en el contexto del KPI. Los umbrales permiten establecer categorías en el rendimiento, por ejemplo: malo, regular, bueno y muy bueno, y esto a su vez, permite decorar con colores o iconos el estado del KPI.

En la Figura 76 muestra un scorecard, el cual incluye los valores de las metas y la decoración correspondiente a los umbrales. Además, se han definido metas diferentes para cada año. Por otra parte, se ha incluido una decoración que resume la tendencia respecto al cumplimiento de la meta (supuestamente respecto a si mejora o no comparándola con el año anterior).

	FY 2003			FY 2004		
	Value	Goal and Status	Trend	Value	Goal and Status	Trend
Revenue	$33,683,805	$26,864,605	1	$52,714,103	$38,736,376	1
Internet Revenue	$5,762,134	$7,779,293	-1	$16,473,618	$6,338,348	1
Channel Revenue	$27,921,671	$33,143,000	1	$36,240,485	$43,240,000	1
Expense to Revenue Ratio	29.40%	25.00%	1	21.50%	25.00%	-1
Product Gross Profit Margin	10.53%	12.00%	-1	11.65%	12.00%	1
Growth in Customer Base	46.06%	30.00%	1	440.41%	30.00%	1

Figura 76.- Ejemplo de scorecard

Las vistas de reports son gráficas o tablas contenidas en un report. Un report es un informe, que en el contexto tecnológico del SharePoint se corresponde con un fichero Excel, el cual, además de

contar con la funcionalidad típica de Microsoft Excel, está conectado a un cubo y accede a su información mediante tablas y/o gráficas dinámicas.

En la Figura 77 muestra en la zona superior izquierda un scorecard y bajo ella una vista de report con representación tabular. En las zonas de la derecha se muestran una colección de gráficos correspondientes a otras vistas de reports.

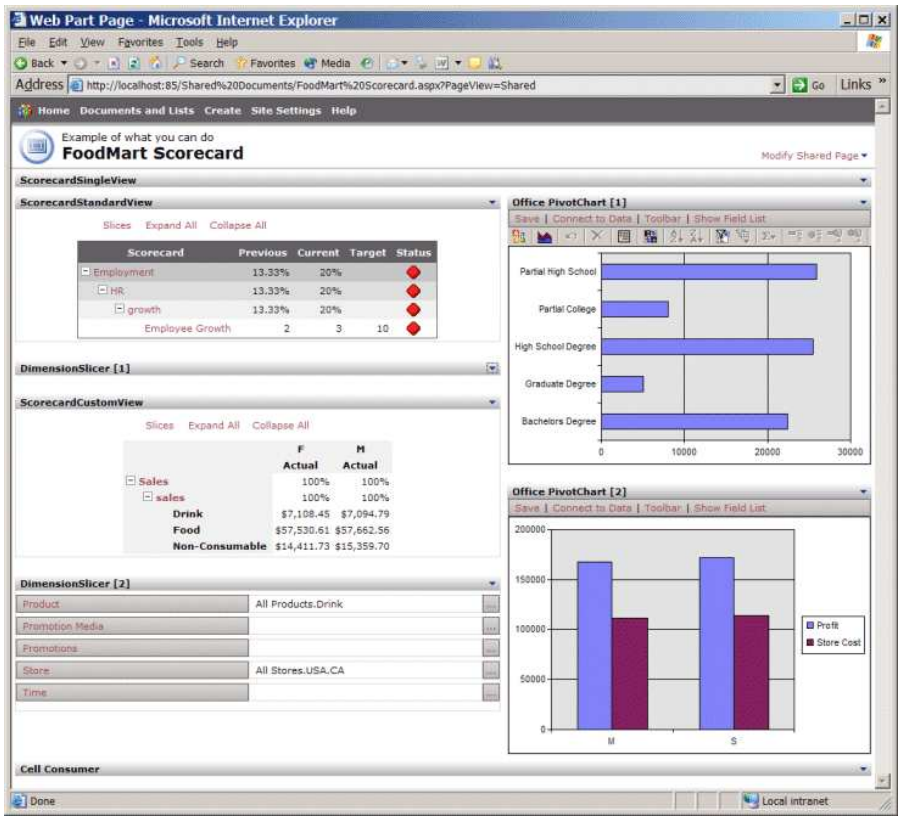


Figura 77.- Ejemplo de Dashboard

5.4.2.- Pasos previos a la creación del dashboard: crear vistas de reports

El primer paso en el desarrollo de un dashboard es estructurar adecuadamente los reports en vistas, que permitirán al usuario consultarlas y poder utilizarlas en la elaboración del dashboard.

Crear un report

Para crear un report tenemos dos posibilidades: crear el report directamente en el sitio de SharePoint o crear previamente el Excel y después guardarlo en la librería de reports del sitio de SharePoint.

A continuación, únicamente se describen los pasos a realizar para seguir el método de la primera de las dos opciones:

1. En la página principal, hacer clic sobre el enlace “Reports” (señalado en rojo en la Figura 78).

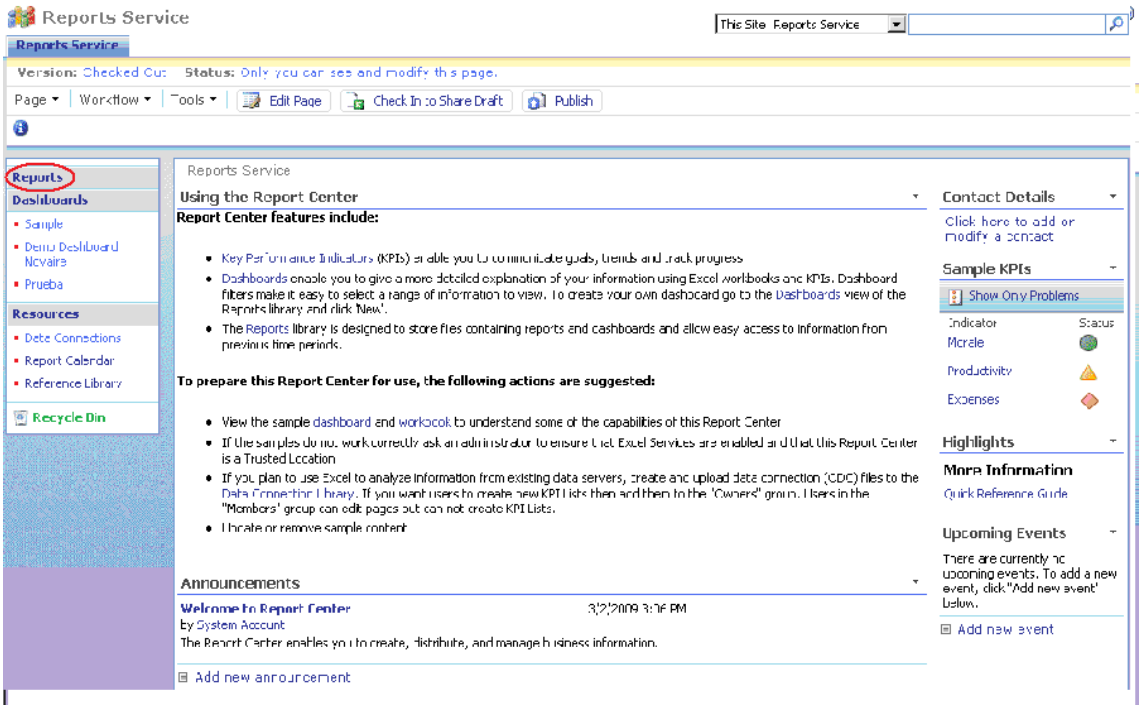


Figura 78.- Vista de Reports Service

2. Hacer clic sobre la lista desplegable junto al menú “New” y seleccionar “Report” (indicado en la Figura 79)

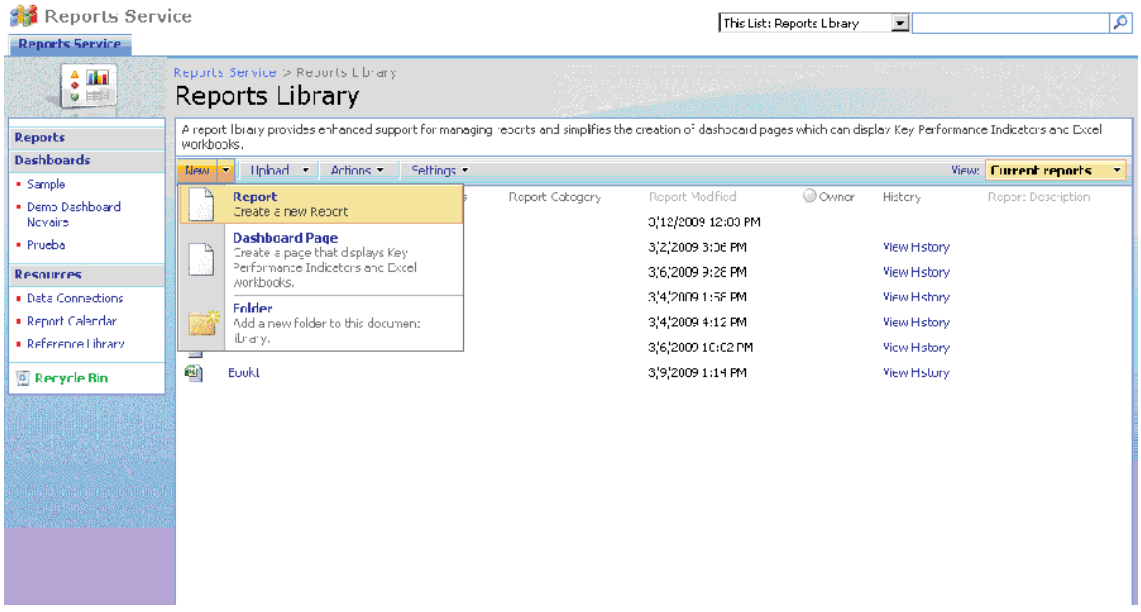


Figura 79.- Vista de Reports Library

3. Rellenar los campos principales del report como son el tipo, el nombre y una breve descripción del mismo y pulsar el botón “OK” para dar de alta el archivo, como se puede observar en la Figura 80.

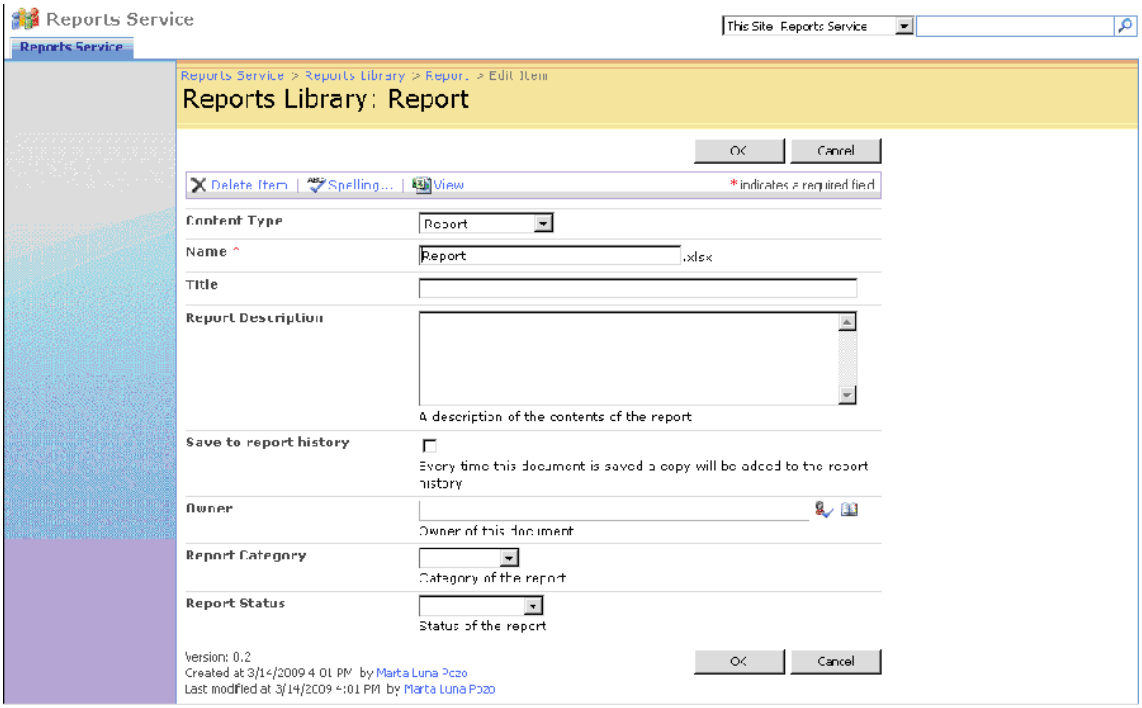


Figura 80.- Interfaz de creación del report

4. Una vez completada la acción, se podrá ver como el report que se acaba de crear ha sido añadido en el apartado Reports Library.
5. Para editar el report recién creado, se debe seleccionar de la lista desplegable que aparece junto al nombre del report la opción de "Edit in Microsoft Office Excel" (Figura 81) y posteriormente guardar los cambios que se realicen al mismo.

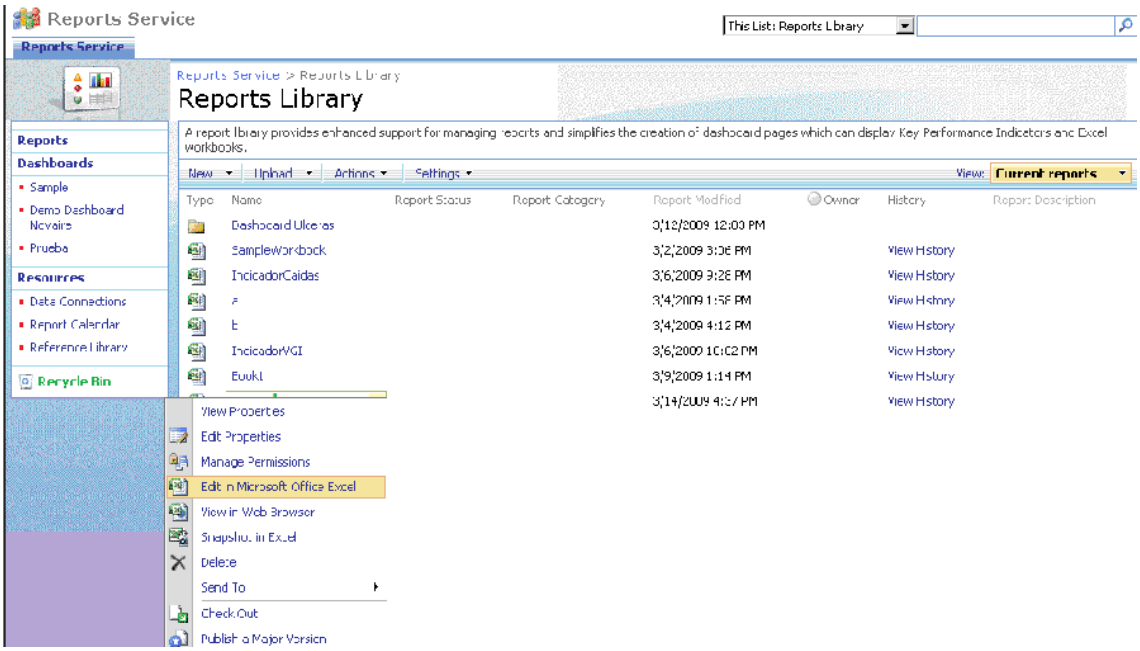


Figura 81.- Vista de Reports Library

Configurar la conexión del report

Es muy importante a la hora de crear un report, asociarle una conexión y después subirla al apartado de Data Connections de SharePoint junto al resto de conexiones siguiendo los pasos detallados a continuación (a destacar que los cinco primeros pasos son idénticos a los del punto 5.2.1):

1. Indicar a Microsoft Excel que se va a realizar una conexión a los cubos de Analysis Services, como se observa en la Figura 82.

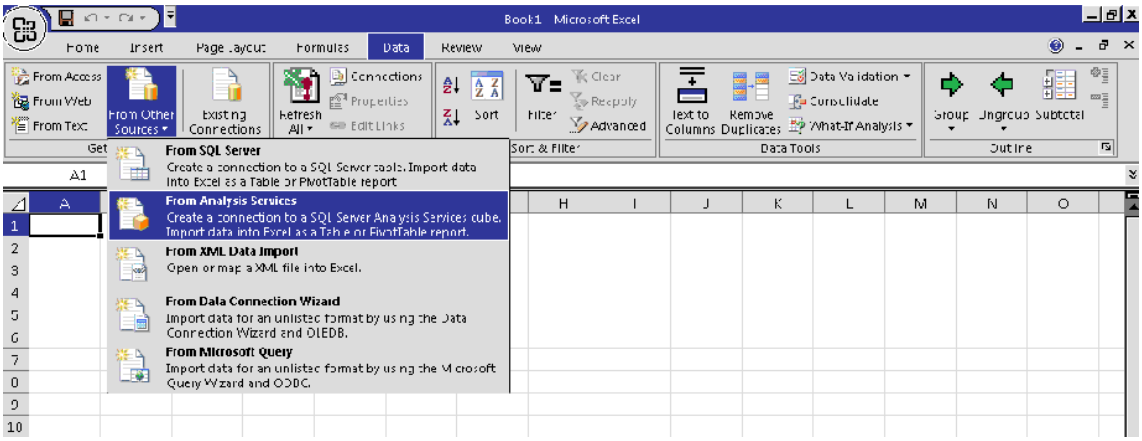


Figura 82.- Punto de partida en la creación de la conexión

2. Poner el nombre del servidor donde se hallan los cubos (Figura 83).

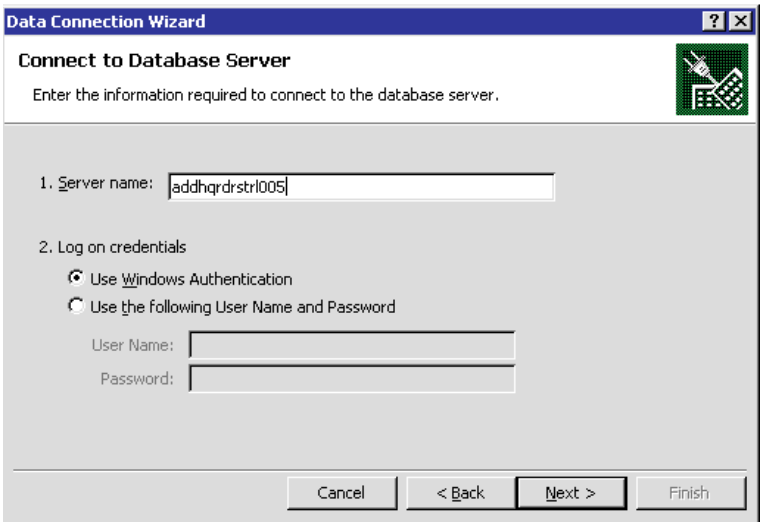


Figura 83.- Paso 1 de creación de la conexión

3. Seleccionar el cubo al cual se desea conectar (Figura 84).

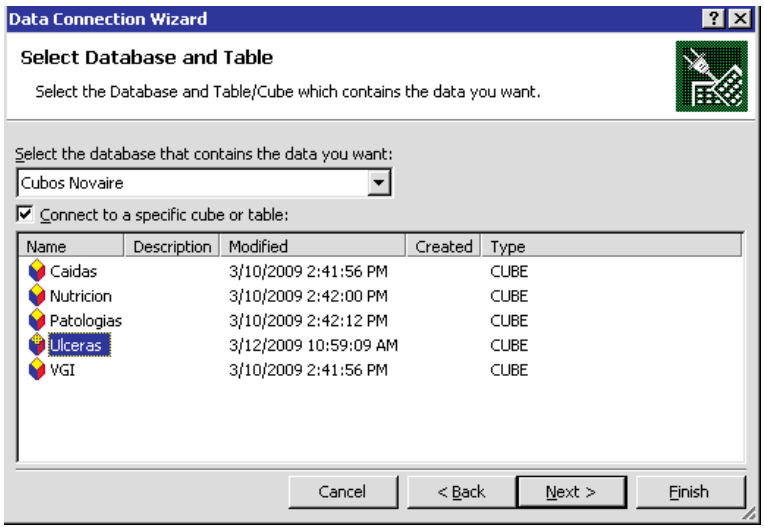


Figura 84.- Paso 2 de creación de la conexión

4. Darle nombre a la conexión que acabamos de crear y finalizar el proceso de creación de la nueva conexión (Figura 85).

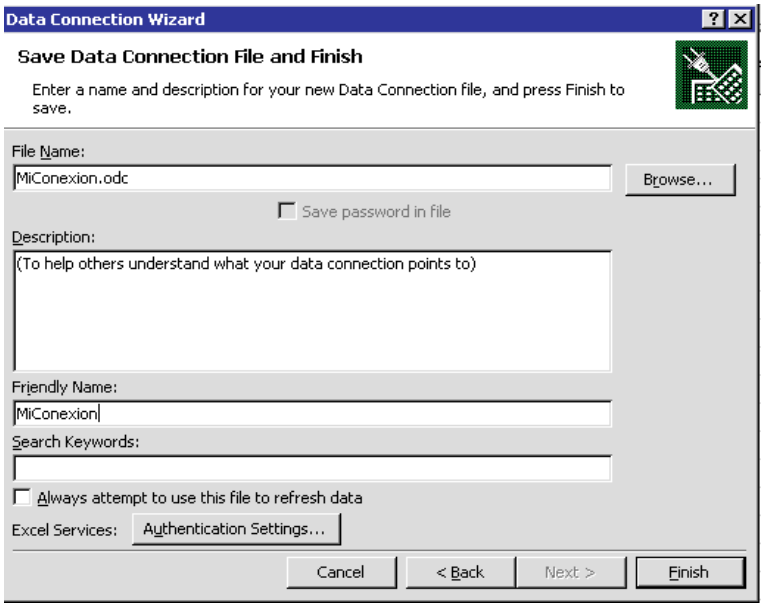


Figura 85.- Paso 3 de creación de la conexión

5. Una vez se tiene creada la conexión, para utilizarla en el Excel se debe hacer clic sobre el icono de conexiones existentes y seleccionar la conexión deseada en la lista que aparece, la cual recoge todas las conexiones definidas hasta el momento.
6. Ya en Microsoft SharePoint, hacer clic sobre el enlace de “Data Connections” que aparece en la columna situada a la izquierda y a continuación seleccionar Office Data Connection File de la lista desplegable junto al menú “New”, como se muestra en la Figura 86.

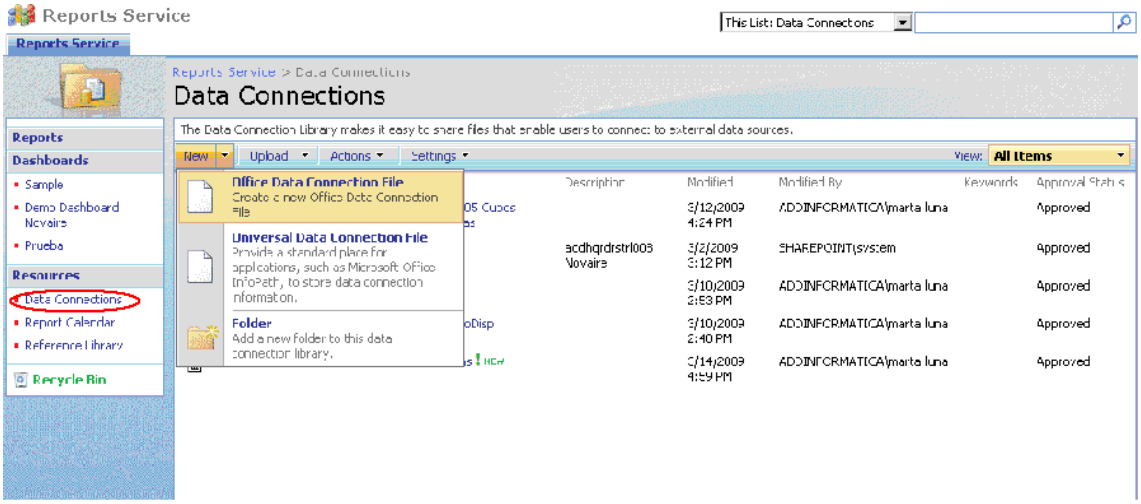


Figura 86.- Vista de Data Connections

7. Explorar el sistema de archivos hasta encontrar el archivo .odc asociado a la conexión que se desea subir a SharePoint, como se muestra en la Figura 87.

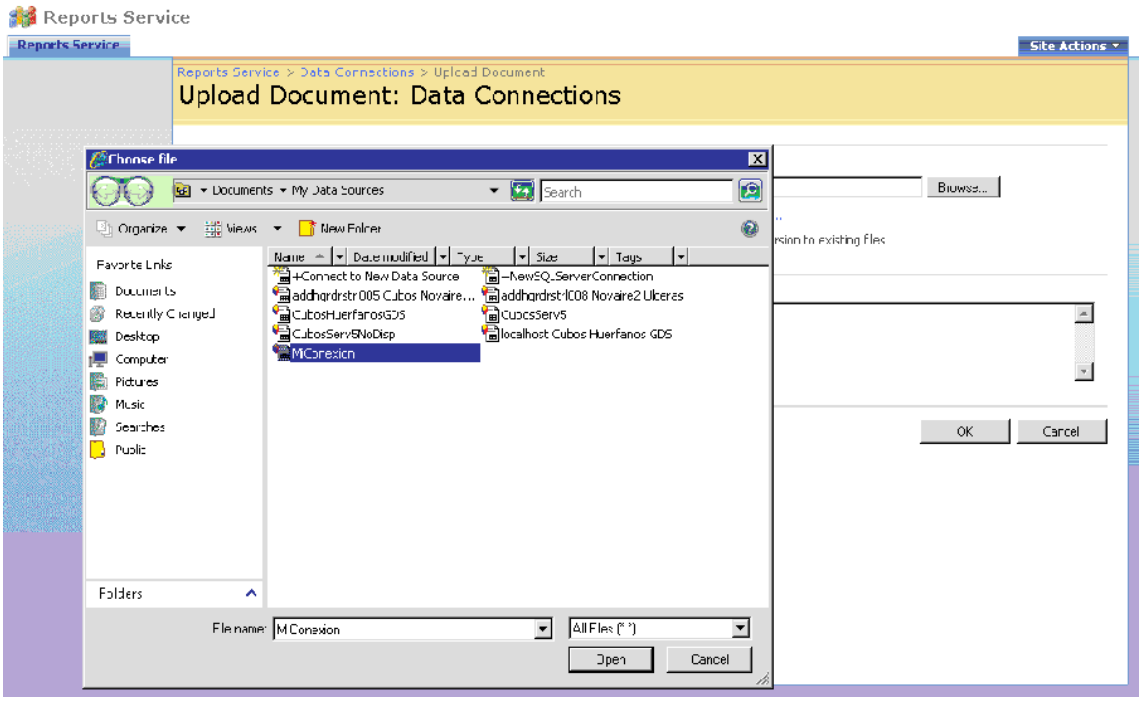


Figura 87.- Interfaz para subir la conexión

8. A continuación rellenar los campos asociados a la conexión tales como nombre, título y una breve descripción (Figura 88)

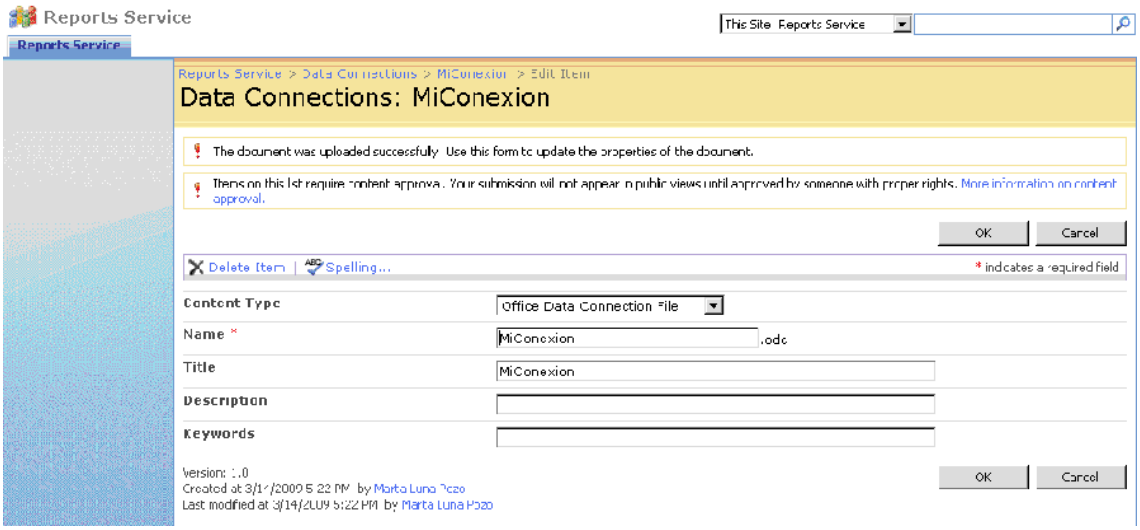


Figura 88.- Relleno de datos de Data Connection

9. Observar que el estado de aprobación de la nueva conexión aparece como pendiente. Para aprobarla, hacer clic sobre la opción "Approve/Reject" de la lista desplegable asociada al nombre de la conexión que acabamos de crear. La Figura 89 muestra el estado pendiente de la conexión antes de realizar su aprobación.

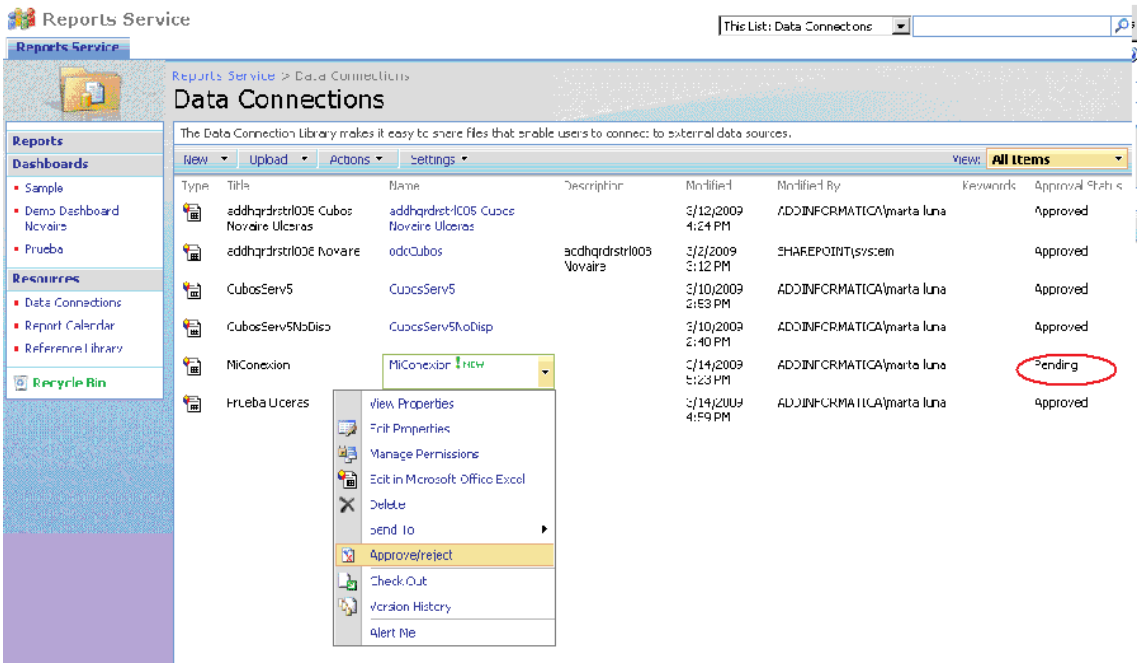


Figura 89.- Ejemplo de conexión pendiente de aprobación

Hasta el momento, se han creado los reports y subido a SharePoint todas las conexiones que se van a utilizar. A continuación, lo que se va a hacer es detallar cómo se va a estructurar cada uno de los reports en vistas.

Crear vistas en los reports

Cada report va a corresponderse con un libro Excel con varias hojas de cálculo, cada una de las cuales va a ofrecer una vista diferente del report.

La forma de estructurar los reports es la siguiente:

- En la primera hoja siempre ha de aparecer una tabla resumen del report que sea lo suficientemente significativa como para obtener toda la información recogida a grandes rasgos. Esto se puede observar en la Figura 90.

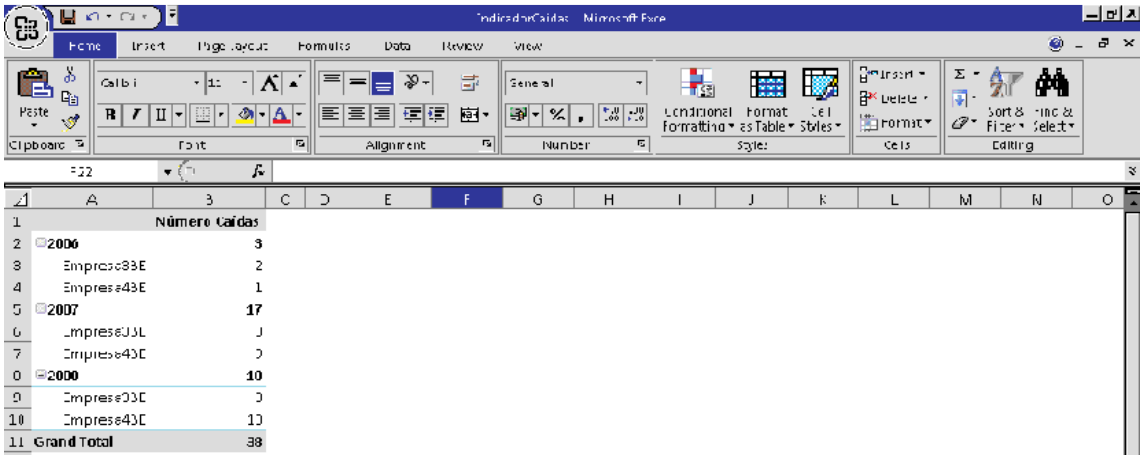


Figura 90.- Primera hoja de libro Excel para ser importado en SharePoint

- Cada una de las hojas sucesivas, ha de ser una vista parcial del report que recoja sólo ciertos aspectos del mismo. Dicha vista constará bien de una tabla resumen o de una tabla resumen con un gráfico asociado. Además, será posible incorporar filtros a dichas vistas para que los datos recogidos puedan cambiar de forma dinámica.

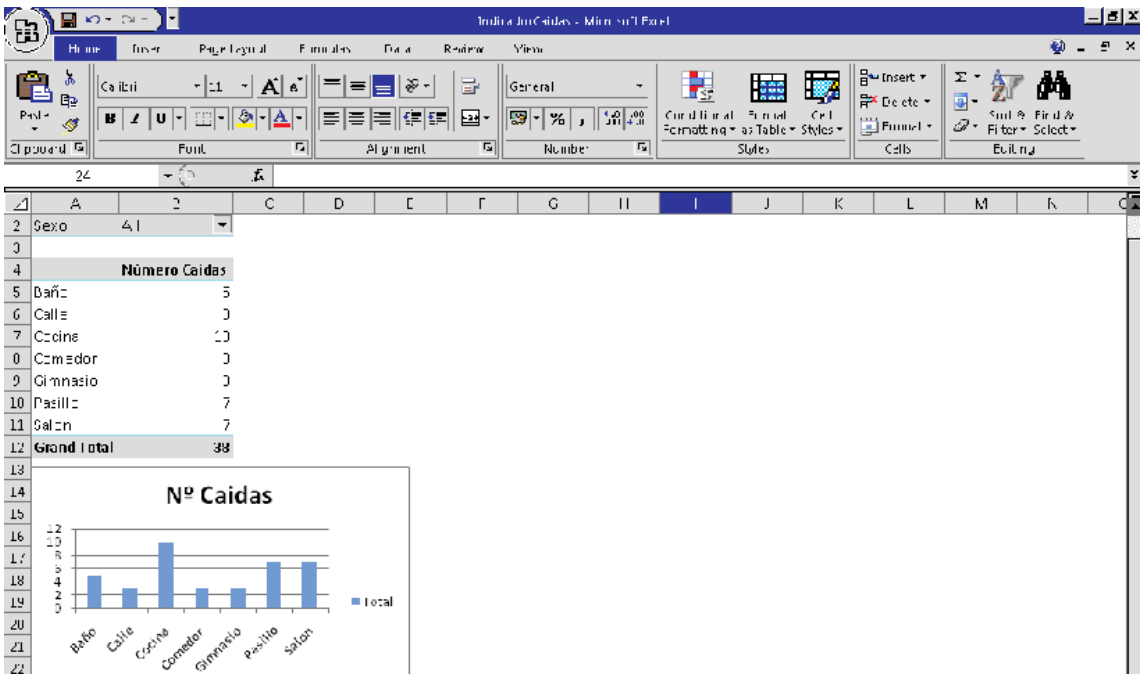


Figura 91.- Patrón de hoja estándar de Microsoft Excel para importar a SharePoint

A la hora de consultar los reports, el usuario no va a ver las hojas de Excel tal cual, sino que sólo va a ver los elementos que aparecen en ellas de manera individual junto con los elementos que se han definido en ellas, es decir, el programador va a seleccionar rangos de celdas en la hoja Excel y darles un nombre para que dicho rango sea considerado como un elemento más del report. Esto es útil para aquellas ocasiones en las que se desee ver a la vez una tabla resumen, su gráfico asociado y sus filtros en caso que los tuviera definidos. Por tanto, el usuario es capaz de ver cada una de las tablas o gráficos que aparecen en el report y todos los rangos de celdas que hayamos definido.

Por todo lo dicho anteriormente, se deduce que es de vital importancia dotar a cada uno de los elementos del report de un nombre adecuado para que el usuario sepa exactamente lo que representa cada uno. Por cada hoja de Excel del report, se debe dar un nombre a la hoja propiamente dicha, a la tabla resumen, al gráfico asociado a dicha tabla y al rango de celdas que contienen tanto la tabla resumen con sus filtros asociados como el gráfico. A continuación se van a detallar los pasos a realizar para dotar a cada elemento con su nombre correspondiente.

1. Poner nombre a la hoja del Excel (Figura 92)

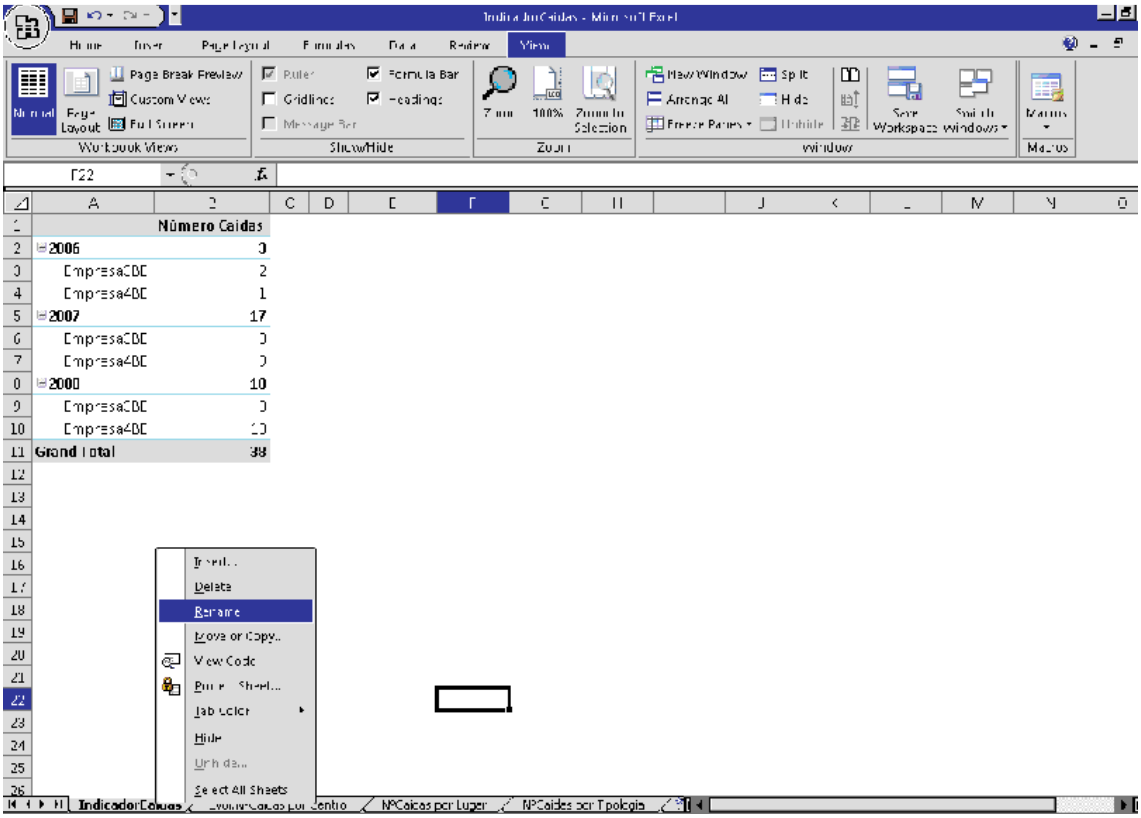


Figura 92.- Renombrado de una hoja Excel

2. Cambiarle el nombre a la tabla resumen (Figura 93).

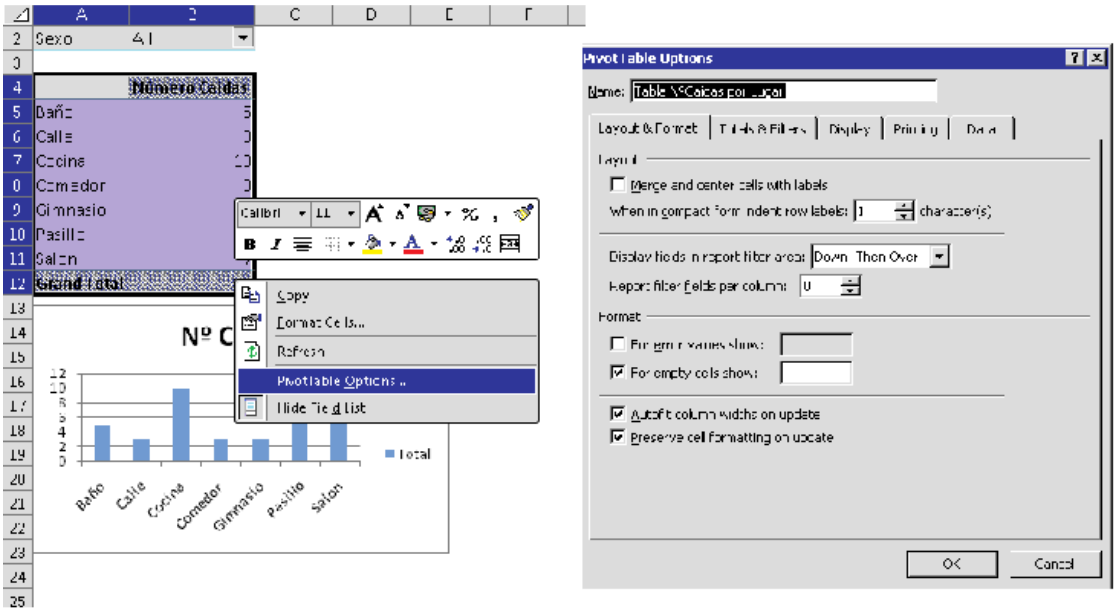


Figura 93.- Renombrado de la tabla resumen

3. Cambiarle el nombre al gráfico (Figura 94).

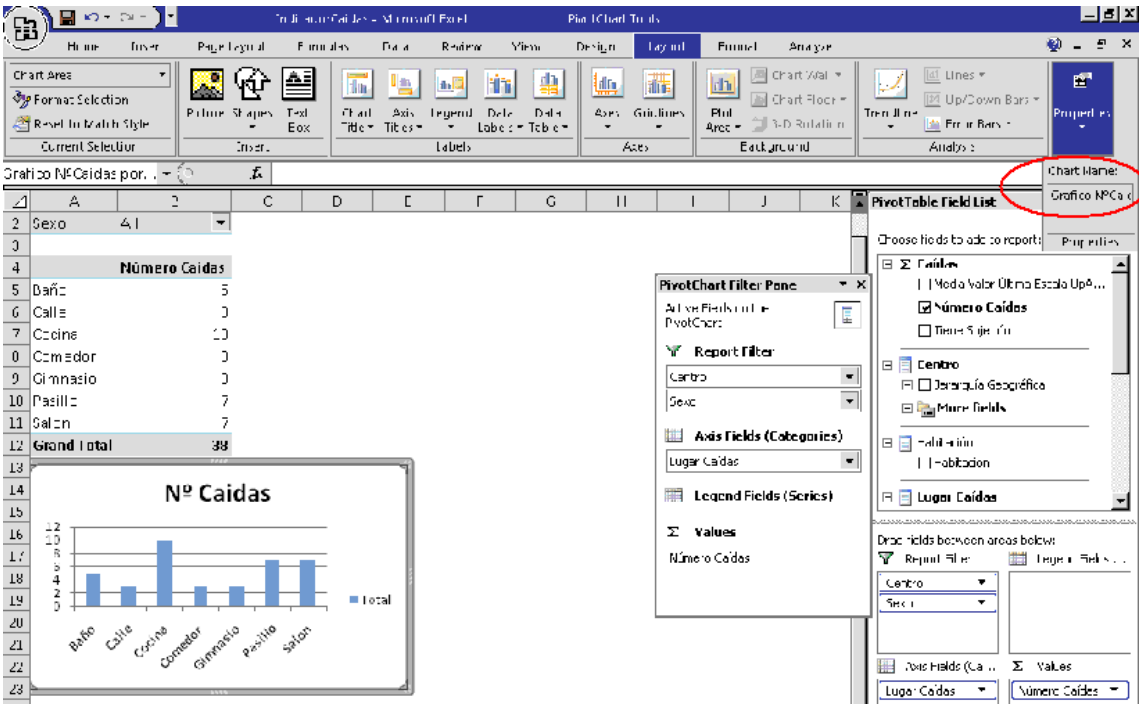


Figura 94.- Renombrado de un gráfico de Excel

4. Poner nombre al rango de celdas que cubre tanto la tabla resumen con filtros como el gráfico asociado (Figura 95).

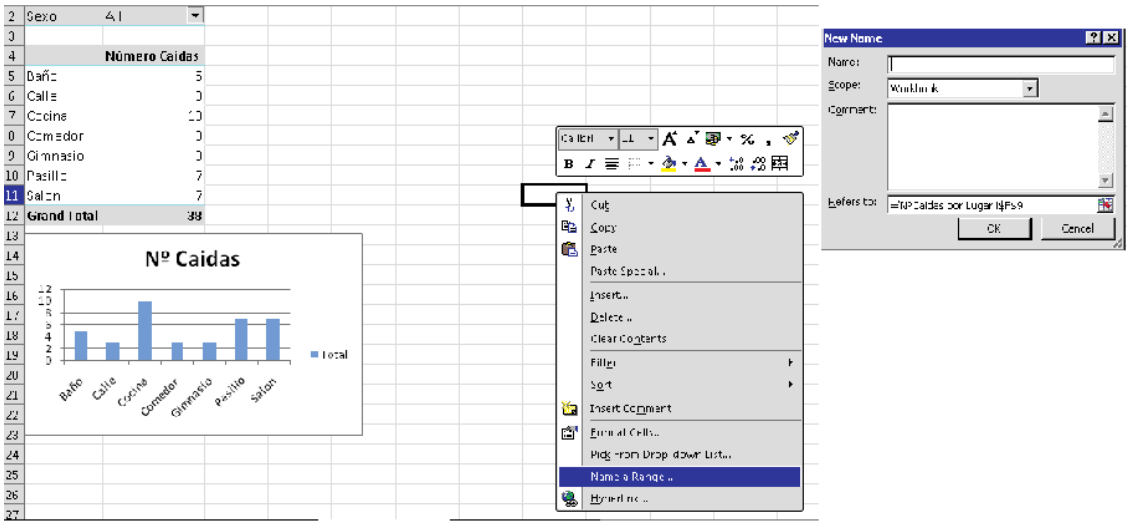


Figura 95.- Renombrado de celdas y filtros

En lo que respecta a los filtros, solamente decir que se puede poner como filtro cualquier dimensión con sólo seleccionarla y ponerla en el apartado de Report Filter que aparece marcada en la Figura 96. Los filtros son de especial interés porque los resultados de las tablas resumen y de las gráficas varían de forma dinámica en función de lo que seleccionemos en los filtros.

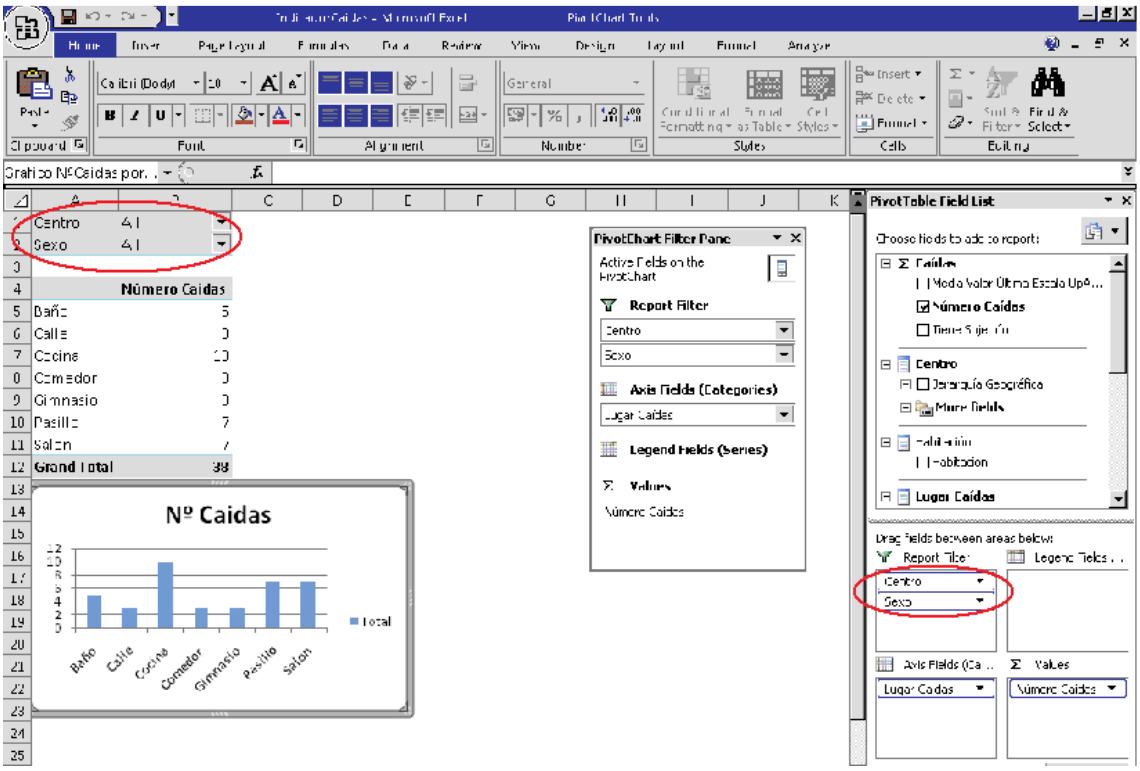


Figura 96.- Ejemplo de filtros dinámicos con Microsoft Excel

Una vez se ha dotado de nombre a todos los elementos del report, para poder ver luego la lista desplegable de elementos donde se selecciona el elemento que se desea ver, en el cuadro de

diálogo que aparece cuando vamos a guardar el report, se debe marcar la pestaña “Publish” y seleccionar “Excel Services” (Figura 97)

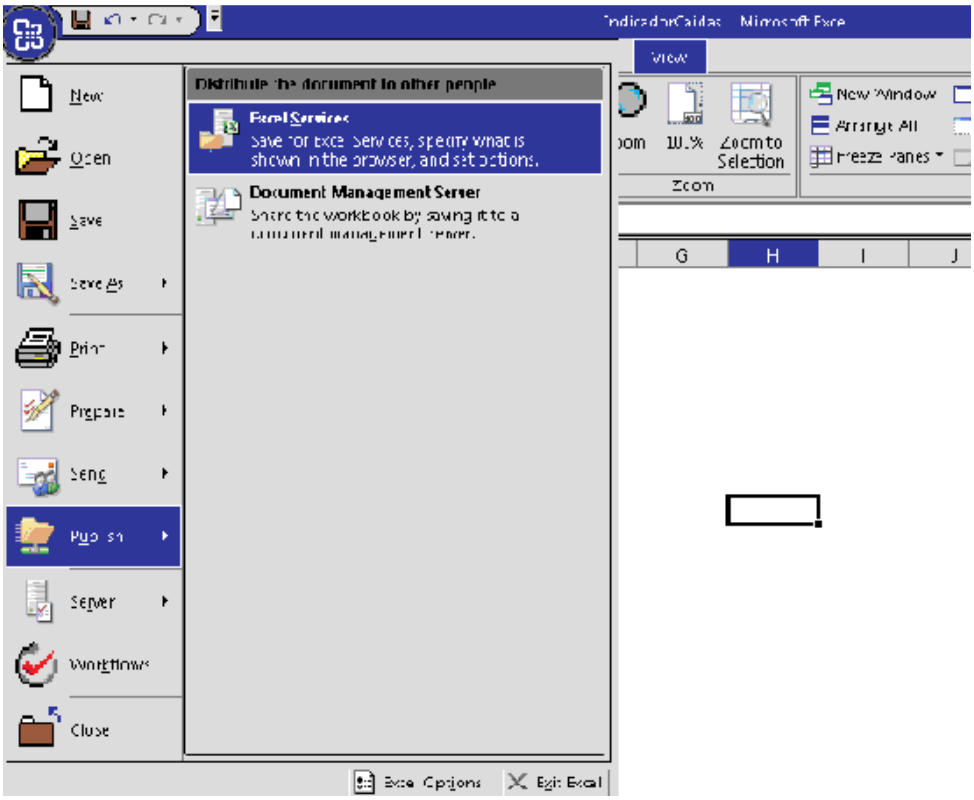


Figura 97.- Publicación de Excel Services

A continuación, se hace clic sobre el botón correspondiente a “Excel Services Options” y se indica que sólo se quiere ver en el navegador los elementos con nombre “Items in the Workbook”, marcando en la lista los que se desean ver, tal y como se ha hecho en la Figura 98. De esta forma se consigue que el usuario sólo pueda ver los elementos que se hayan indicado previamente.

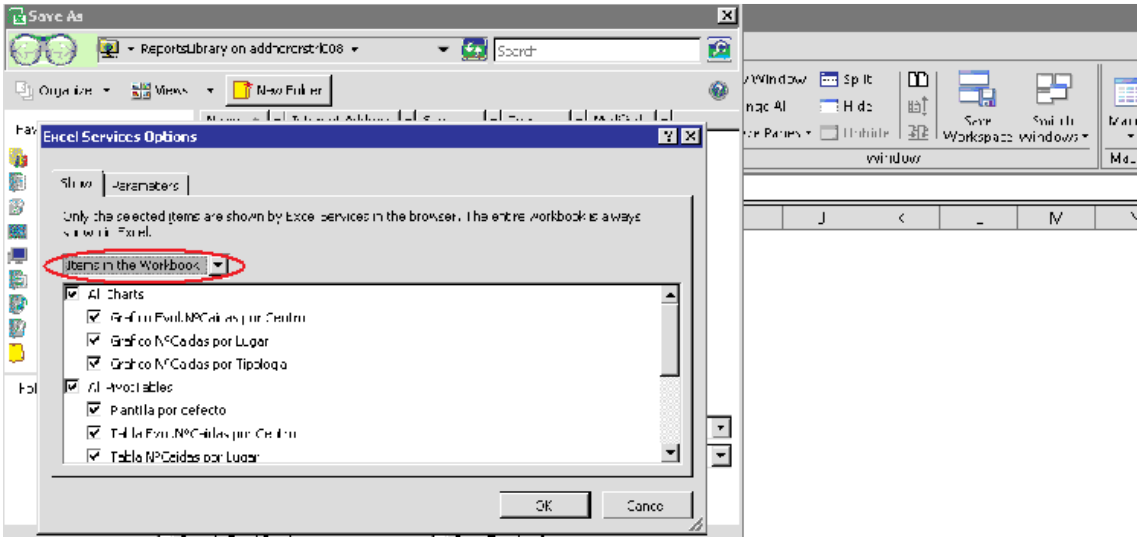


Figura 98.- Elementos a importados mediante Excel Services

Consultar las vistas previamente definidas

En este momento, lo que se tiene es el report correctamente estructurado en vistas para poder ser utilizadas en la elaboración del dashboard, como se muestra en la Figura 99.

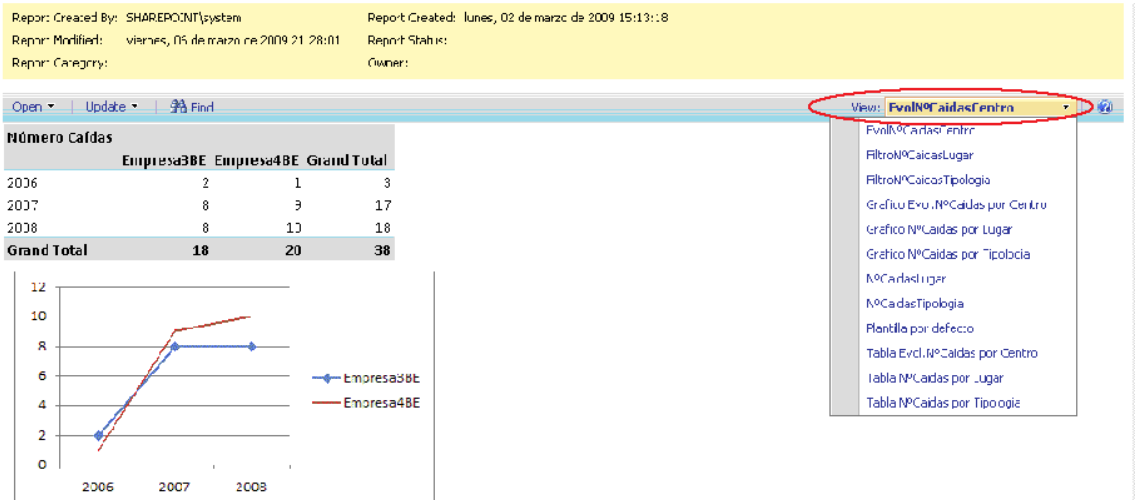


Figura 99.- Ejemplo de report correctamente estructurado

Para consultar el report, tan sólo se debe hacer clic sobre el enlace asociado al nombre del mismo o seleccionar de la lista desplegable asociada al mismo la opción "View in Web Browser". Al hacer clic sobre dicho enlace, se llega a una página donde se muestra uno de los elementos del report, que en este caso debería ser la tabla resumen de la primera hoja del report, es decir, aquella que resume toda la información importante del report que se está consultando. En dicha página se puede observar la pestaña "View", la cual tiene asociada una lista desplegable que contiene todos los elementos del report que se pueden consultar. Seleccionando cada uno de los diferentes elementos de dicha lista, se podrá navegar por el contenido de todo el report.

Crear un dashboard

Para crear un dashboard se debe ir a la pestaña "Dashboards" o dentro de la pestaña de "Reports", seleccionar la opción "Dashboards" del desplegable asociado a "View", como se observa en la Figura 100.

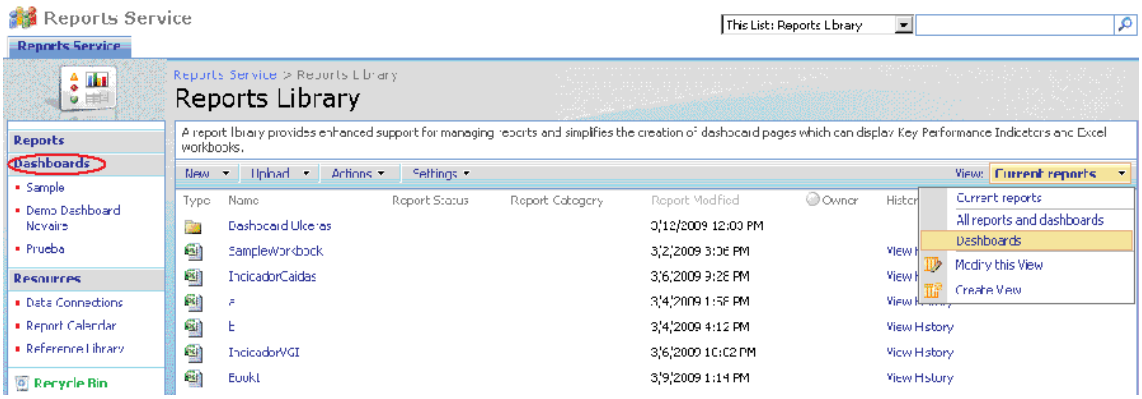


Figura 100.- Llegar al menú Dashboards desde Reports Library

Una vez allí, se debe hacer clic en New y seleccionar Dashboard page (Figura 101).

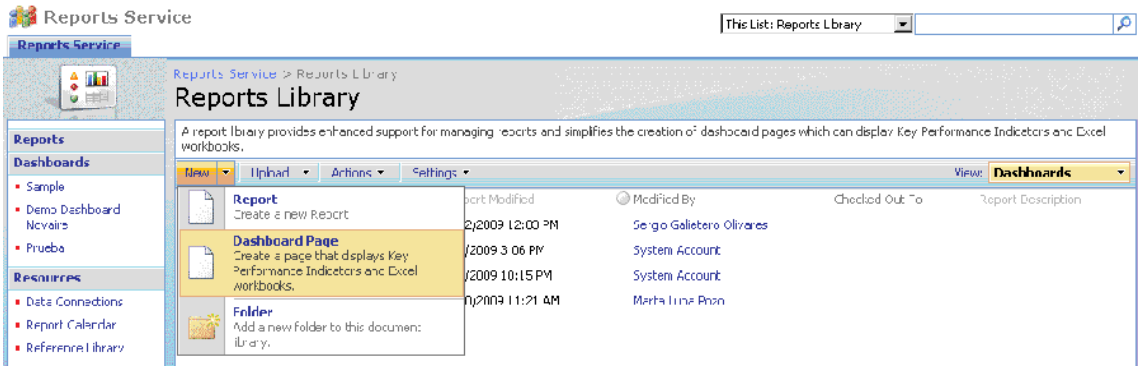


Figura 101.- Creación de la página de dashboard

A continuación se deben rellenar los campos relativos al nombre y título y dejar el resto de campos tal y como vienen por defecto, como se observa en la Figura 102.

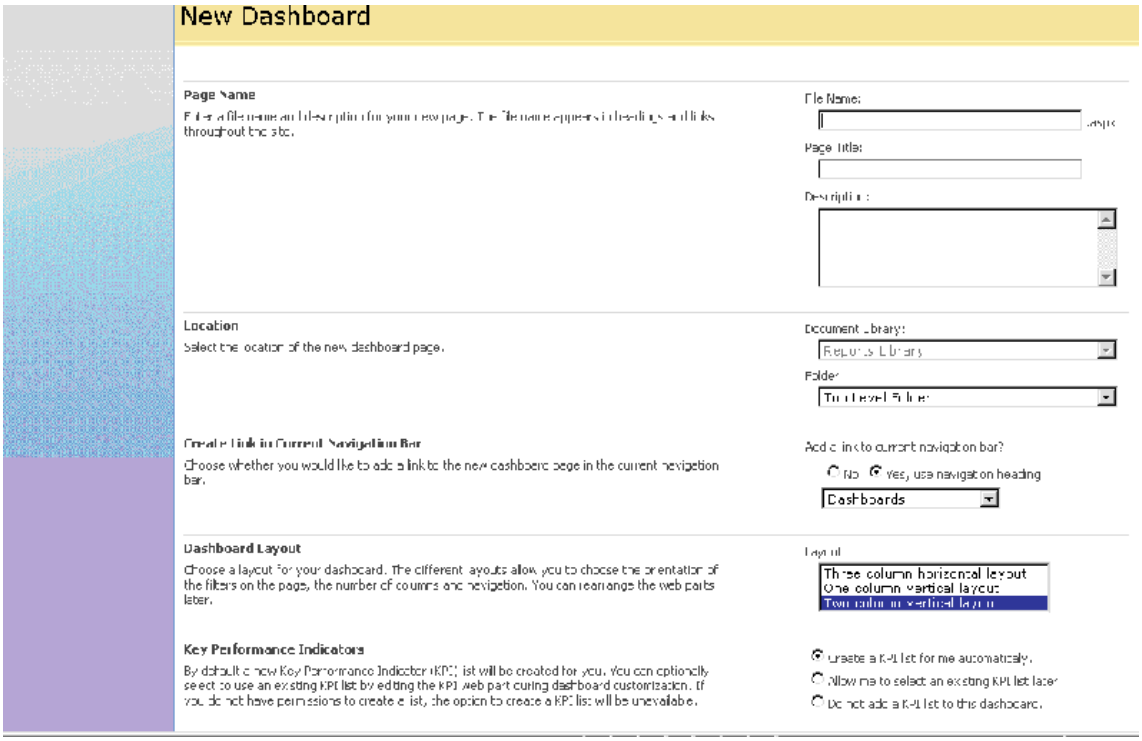


Figura 102.- Propiedades del dashboard a crear

Definir la estructura del dashboard mediante webparts

Todo dashboard está estructurado en zonas (*webparts*). Las webparts pueden ser de diferentes tipos:

- Lista de KPI's: Muestra un conjunto de indicadores visuales de los datos que permiten al usuario rápidamente obtener información sobre factores importantes.
- Excel Web Access: Muestra tablas o gráficas de un libro Excel.
- Filtros: Los hay de diferentes tipos y permiten mostrar los datos del dashboard según ciertos parámetros.

Definir webparts para mostrar el contenido de los reports

En los siguientes pasos, se van a añadir un conjunto de webparts de tipo Excel Web Access, tantas como reports se tengan, que permitirán mostrar el contenido de las vistas definidas en los reports.

Para editar el dashboard se debe hacer clic sobre el enlace asociado al dashboard y luego seleccionar "Open the tool pane" o bien hacer clic sobre "Edit Page" en la parte superior de la página (Figura 103).

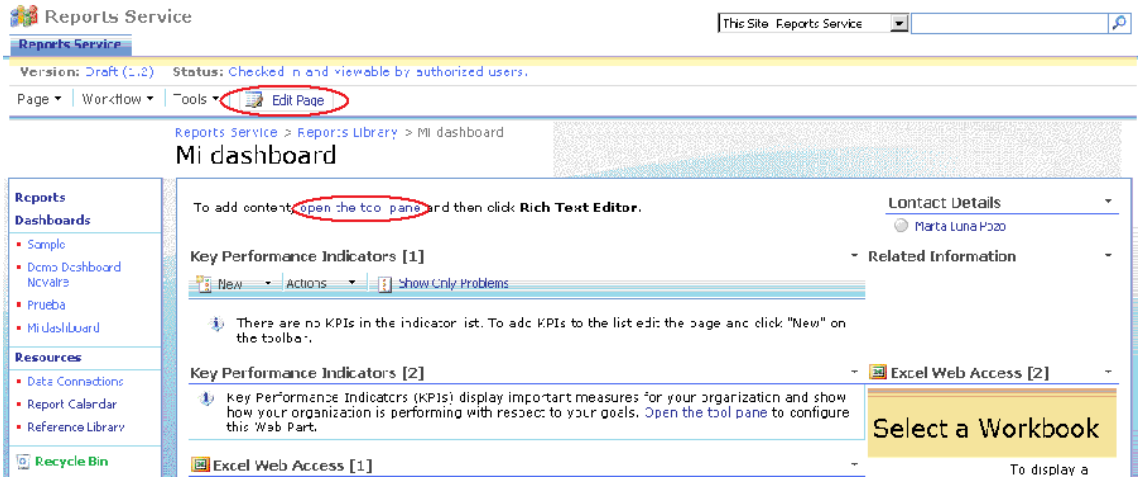


Figura 103.- Pasar a modo edición del dashboard

Cuando se esté en modo edición, hay que hacer clic sobre "Add a webpart", apareciendo un cuadro de diálogo donde se debe seleccionar el tipo de webpart que se desea añadir (Figura 104), en este caso "Excel Web Access".

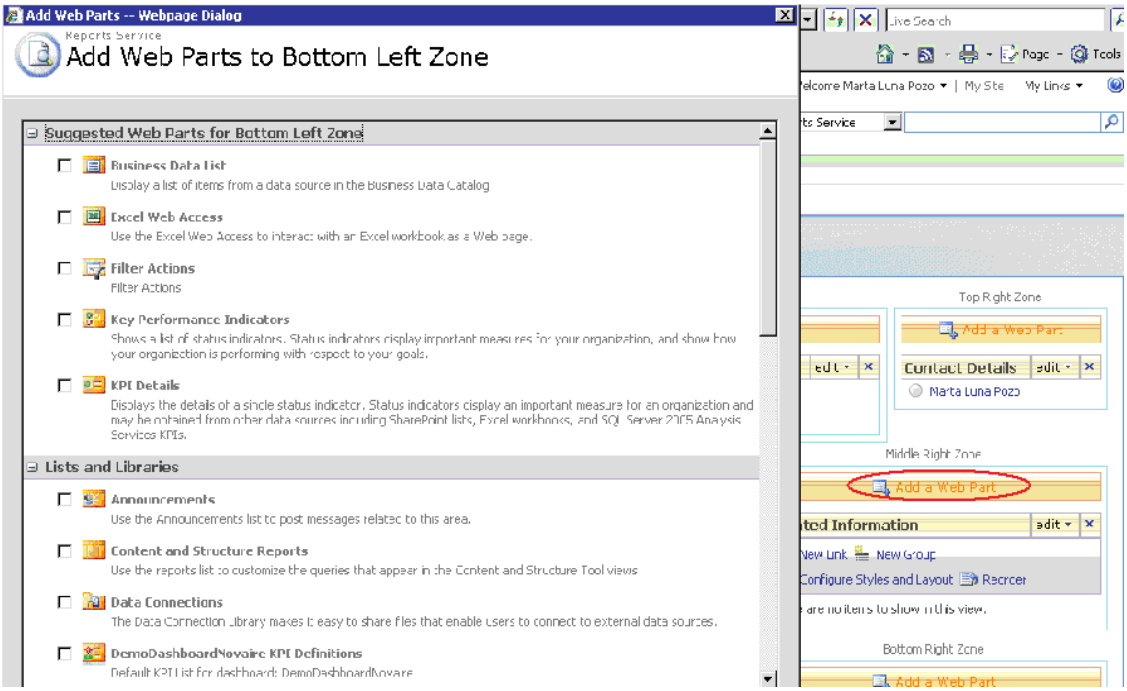


Figura 104.- Tipos de webparts agregables a un dashboard de SharePoint

Para modificar el contenido de la nueva webpart existen varias posibilidades. Si es la primera vez que se va a editar la webpart, se puede hacer clic sobre el enlace “Click here to open the tool pane” o seleccionar la opción “Modify Shared Web Part” de la lista desplegable que aparece en la esquina derecha superior de la webpart. Estos dos casos están marcados en la Figura 105.

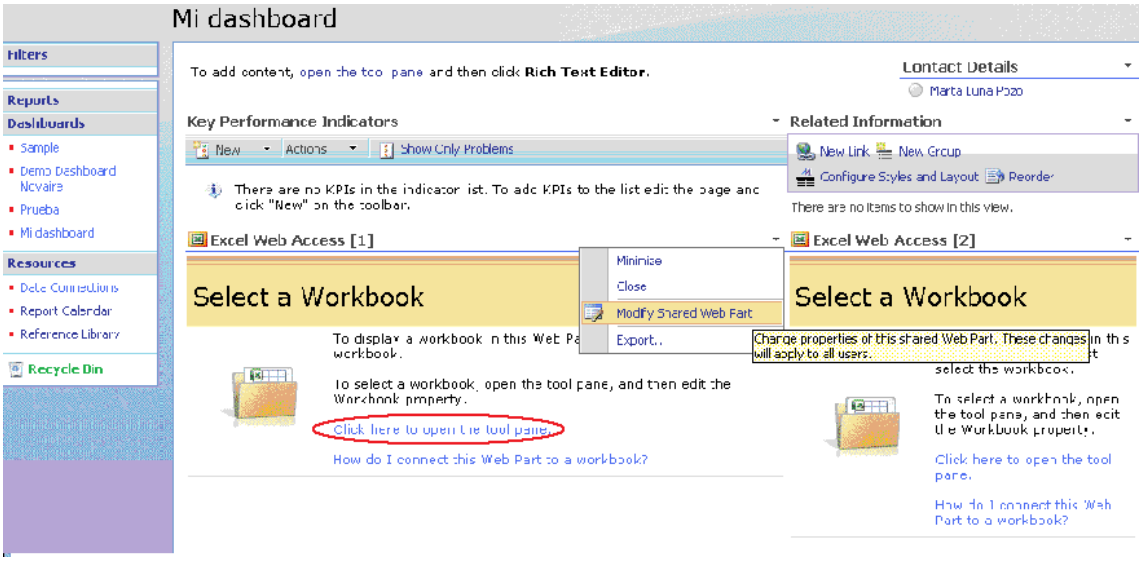


Figura 105.- Opciones de modificación de contenido para un dashboard

Al seleccionar cualquiera de las dos opciones, aparece un panel a la derecha en el que se debe rellenar toda una serie de campos relacionados con la webpart de Excel. Concretamente, en la Figura 106 están marcados los más relevantes.

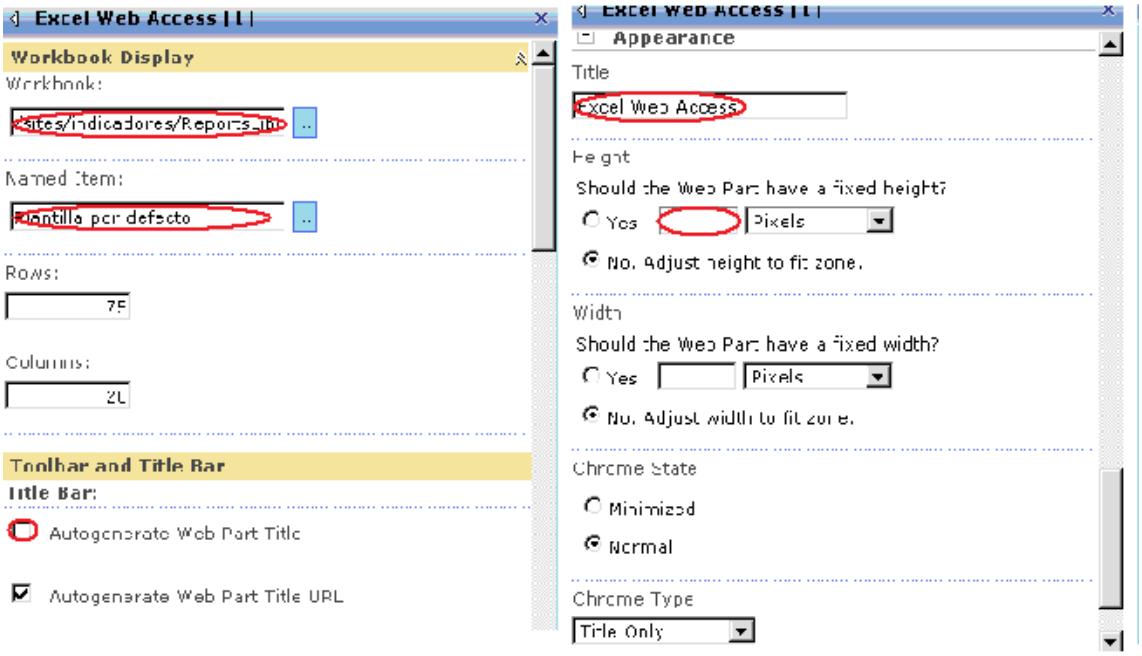


Figura 106.- Lista de campos relevantes para la webpart

El atributo Workbook es el report al cual se va a hacer referencia en la webpart y el Named Item es el elemento que se desea que salga por defecto en la webpart, aunque este último no es necesario ponerlo. Si se desea poner un título a la webpart, es importante desmarcar la opción de autogenerar

el título. A la hora de ajustar las dimensiones de la webpart, se deben cambiar los valores de "Height" y "Width" hasta encontrar unos valores que se ajusten a las dimensiones deseadas.

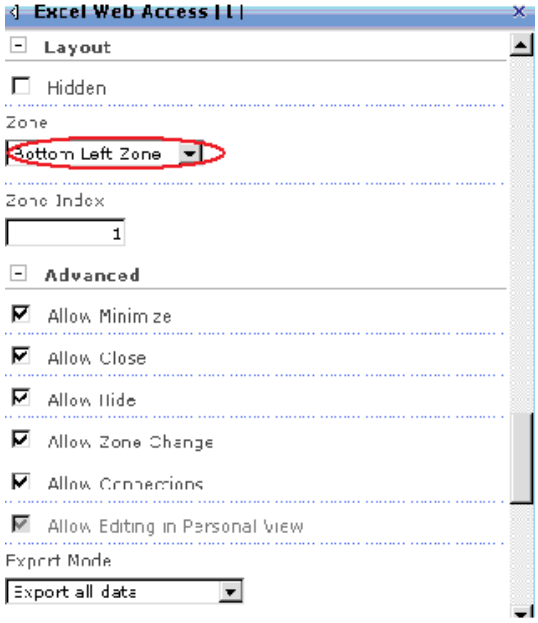


Figura 107.- Propiedad de zona de inserción dentro del dashboard

Para especificar la situación concreta de la webpart en el dashboard, se debe cambiar el atributo "Zone" e indicar si se desea que la webpart esté arriba, abajo, derecha o izquierda. En el caso que se haya escogido una distribución en 3 columnas del dashboard, también se tiene la posibilidad de situar las webparts en el centro. Esta opción se puede ver claramente en la Figura 107.

Definir una webpart para mostrar una lista de KPIs

Una vez se han definido todas las webparts que referencian archivos Excel, es hora de definir los KPI sobre los reports que se tienen definidos para la empresa utilizando la lista de KPIs que ha sido generada automáticamente al crear el dashboard. Para ello, hay que seguir una serie de pasos.

En primer lugar, para crear un KPI se debe ir a la webpart que contiene la lista de KPIs y hacer clic sobre la lista desplegable que aparece junto al menú New. Tal y como se observa, se puede crear un indicador KPI de varias formas, según si se utilizan KPIs ya definidos en otra plataforma, si se utilizan datos existentes en un libro Excel o si se inserta la información del KPI manualmente. En el caso estudiado, sólo se va a definir KPI basados en la información contenida en archivos Excel, como se muestra en la Figura 108.

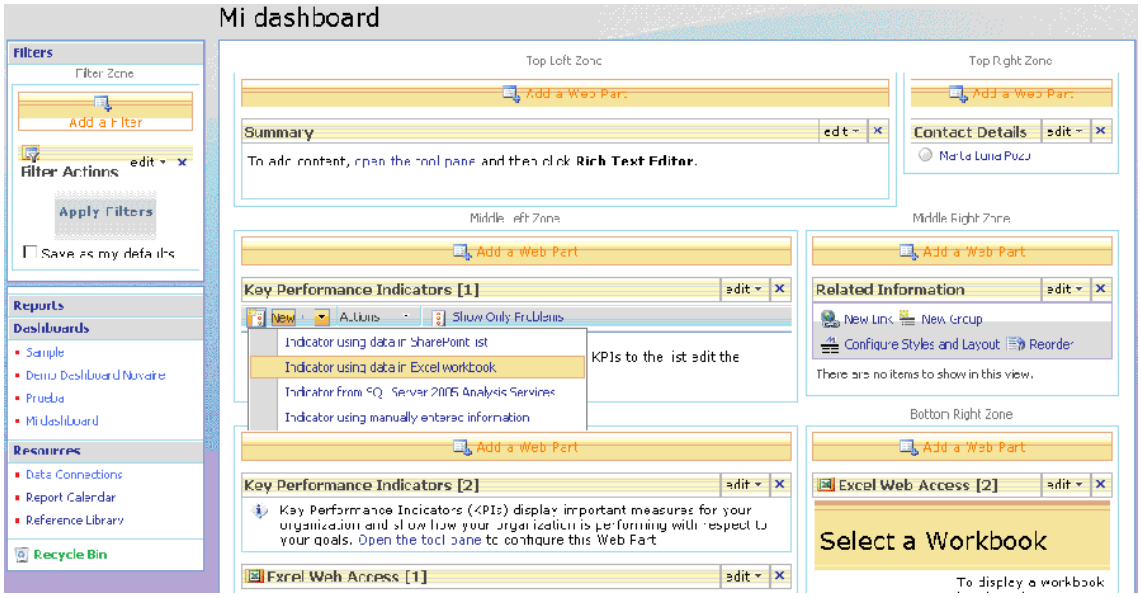


Figura 108.- Crear un KPI a partir de un libro Excel

Para ello, se selecciona la opción de crear el indicador usando los datos de un libro Excel y se rellenan los datos referentes al indicador KPI como son el nombre, una descripción de lo que indica, cómo interpretarlo, el archivo Excel al que referencia, etc. Estos datos se observan detalladamente en la Figura 109.

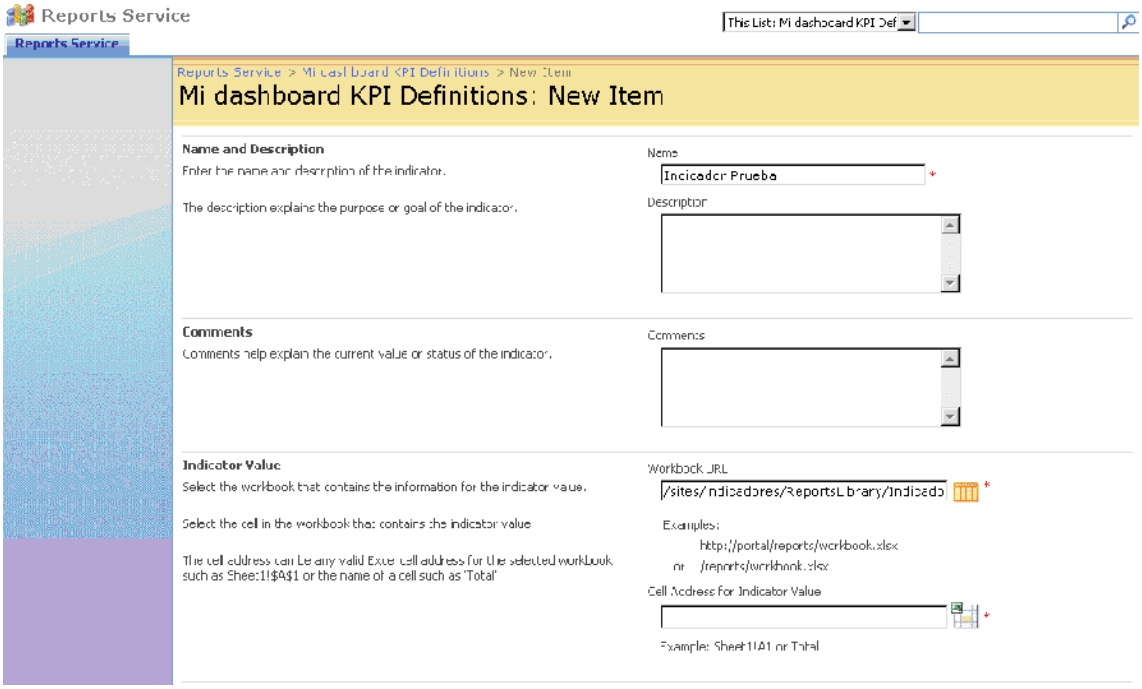


Figura 109.- Atributos del KPI a definir

En el campo que pide qué celda indica el valor del indicador, se debe abrir el archivo Excel asociado y seleccionar dicha celda. Además, es preciso indicar cuál era el objetivo de la empresa para ese indicador y el valor a partir del cual la empresa considera que ese objetivo se encuentra en parte satisfecho. Ambos valores se pueden introducir manualmente o bien usando las celdas del Excel en caso que estuvieran definidos (Figura 110)

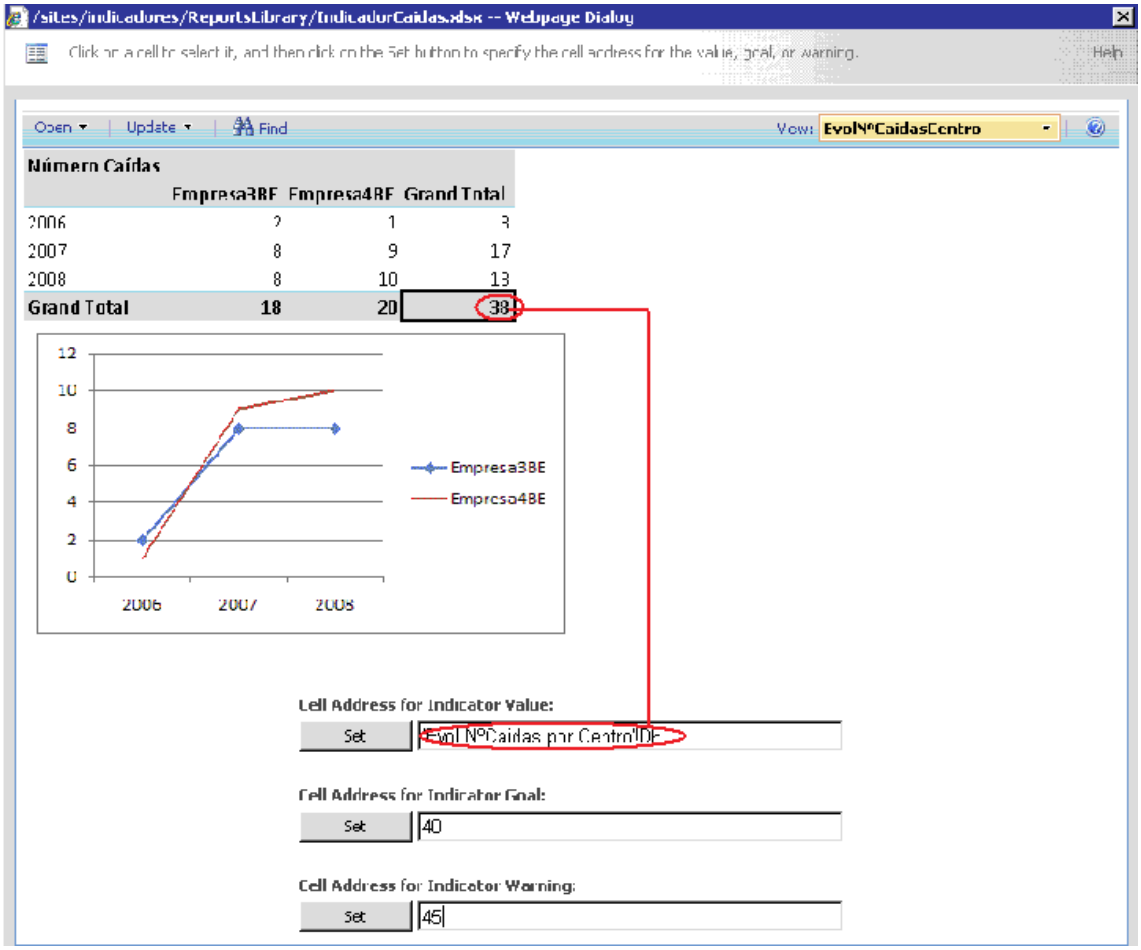


Figura 110.- Especificación de los distintos valores del KPI a partir de un libro Excel

Finalmente, se indica cómo se interpretan los valores del indicador, es decir, si éstos son mejores cuanto más grandes o no y el icono con el cual se representa cada uno de los estados del indicador en función de su grado de satisfacción, como se muestra en la Figura 111.

Figura 111.- Relleno de valores de para indicar el grado de satisfacción

A continuación, se definen de la misma forma el resto de indicadores KPI y cuando ya se hayan definido todos y no se tengan que añadir más webparts, se hace clic sobre la opción de guardar y parar la edición de la página situada en la parte superior de la misma.

Consultar el dashboard resultante y opciones de visualización de los KPIs

El dashboard resultante después de guardar todos los cambios realizados en las webparts es el que aparece en la Figura 112.

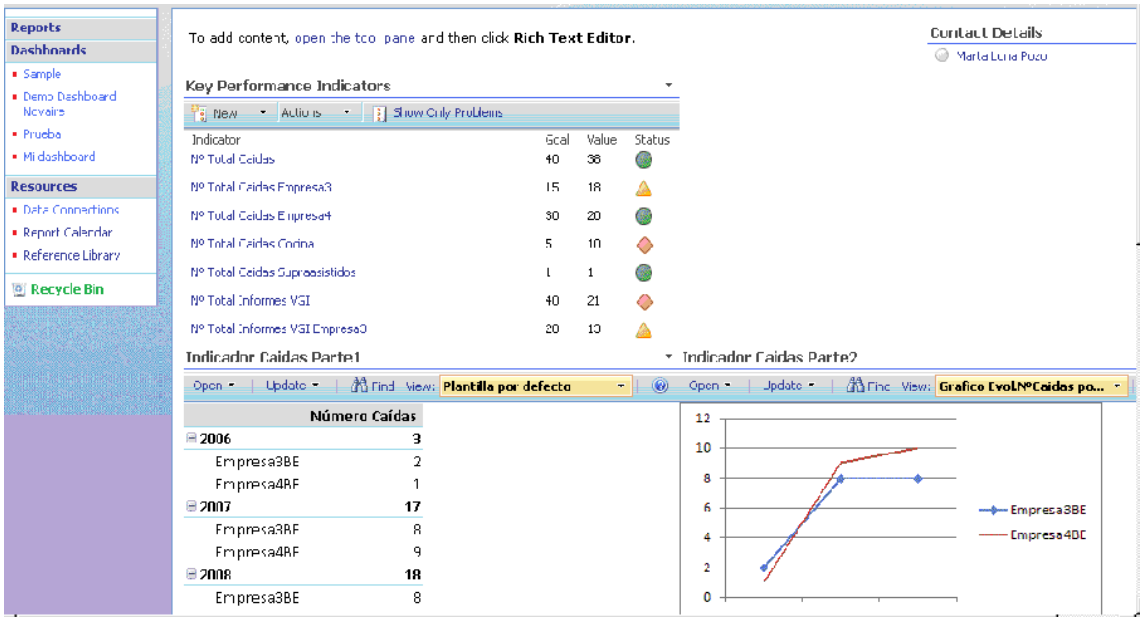


Figura 112.- Ejemplo de dashboard en SharePoint

En lo que respecta a los KPIs, existen diferentes iconos para representar el estado de cada uno de los indicadores. Para cambiar de un tipo de iconos a otro, se ha de seleccionar la opción “Icons” de la lista desplegable asociada a “Actions” y elegir el tipo de icono deseado, como se muestra en la Figura 113.

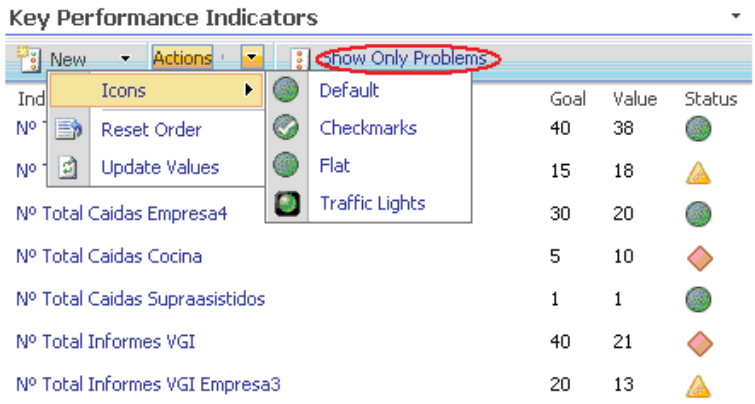


Figura 113.- Diferentes tipos de iconos a mostrar en los KPIs de SharePoint

Además, si lo que se desea es mostrar sólo aquellos KPIs cuyo grado de satisfacción no es el deseado, tan sólo hay que hacer clic sobre la opción de "Show Only Problems", cambiando esta opción a "Show All Indicators" en el momento que se hace clic, como se observa en la Figura 114.

Figura 114.- Opción para mostrar subconjuntos de KPIs

5.5.- Explotación Mediante Dashboards de Performance Point

Una vez estudiadas las funcionalidades para la creación de dashboards en SharePoint, se va a proceder a estudiar este mismo aspecto con una aplicación más específica como es Dashboard Designer de PerformancePoint.

Dicha utilidad nos permite un diseño de dashboards mucho más ágil e intuitivo que SharePoint debido a que es una aplicación específica para llevar a cabo ese cometido, por lo que Dashboard Designer abarca todas las posibilidades que ofrece SharePoint además de otras funcionalidades interesantes.

Durante el presente punto se utilizará como ejemplo un dashboard creado a partir de un indicador de gestión denominado úlceras, cuya definición es la siguiente:

"Número total úlceras aparecidas a los distintos residentes (adjuntando el carácter de la plaza y la tipología que tienen en el momento en el que ocurre la cura), indicando al centro al cual pertenecen y organizadas por día, mes, trimestre o año. Además, se requerirá información de la úlcera en si, como son la localización de la úlcera, el tipo y la procedencia de la misma."

Como se puede observar, este indicador es bien similar al de curas de úlceras, guardando esta vez tan solo la información de la aparición de la úlcera y aquellos datos de la misma que no cambian, es decir, la que permanece estática en todo momento (la localización de la úlcera no varía, así como su tipo o su procedencia).

En el desarrollo de este punto se van a explicar los diferentes componentes que se pueden crear en PerformancePoint ejemplificados con el indicador anteriormente enunciado.

5.5.1.- Introducción

Para llegar a entender el funcionamiento del Dashboard Designer de PerformancePoint , se tiene que partir de unas hipótesis que ayudaran al implementador en dicha labor.

- Los dashboards creados con Dashboard Designer de Performance Point actúan como webparts de SharePoint. Estos dashboards pueden ser previsualizados en cualquier navegador web, pero deben ser subidos a SharePoint al objeto de poder explotar toda su potencia y funcionalidad.
- Todos los elementos que se pueden crear dentro del Dashboard Designer de PerformancePoint están basados en conexiones. Esto contrasta con el modo de crear elementos de SharePoint, donde se necesita el soporte de una hoja Excel, en la cual se crean los elementos, para poder importarlos dentro de un dashboard.
- Se pueden dar permisos a usuarios para poder visualizar tanto el dashboard como sus partes. Dichos permisos se pueden encontrar en la carpeta “properties” de cada elemento, y pueden ser el principal problema de que un elemento o un dashboard no pueden ser visualizados.

5.5.2.- Elementos del Dashboard

Data Source

Como ya se ha comentado en las hipótesis de partida, Dashboard Designer de PerformancePoint no parte una hoja Excel para crear contenidos útiles del dashboard. Para ello, necesita de la creación de Data Sources.

Dichas Data Sources se pueden relacionar con cubos, tablas de bases de datos, e incluso hojas Excel, pero en el ejemplo que se estudia a lo largo de la guía se utilizará una conexión con un cubo (Analysis Services).

El Data Source se configura fácilmente mediante un asistente, contando con una característica especial para configurar la dimensión tiempo. Para ello, una vez creado el Data Source, se debe ir a la pestaña “Time”, como se muestra en la Figura 115.

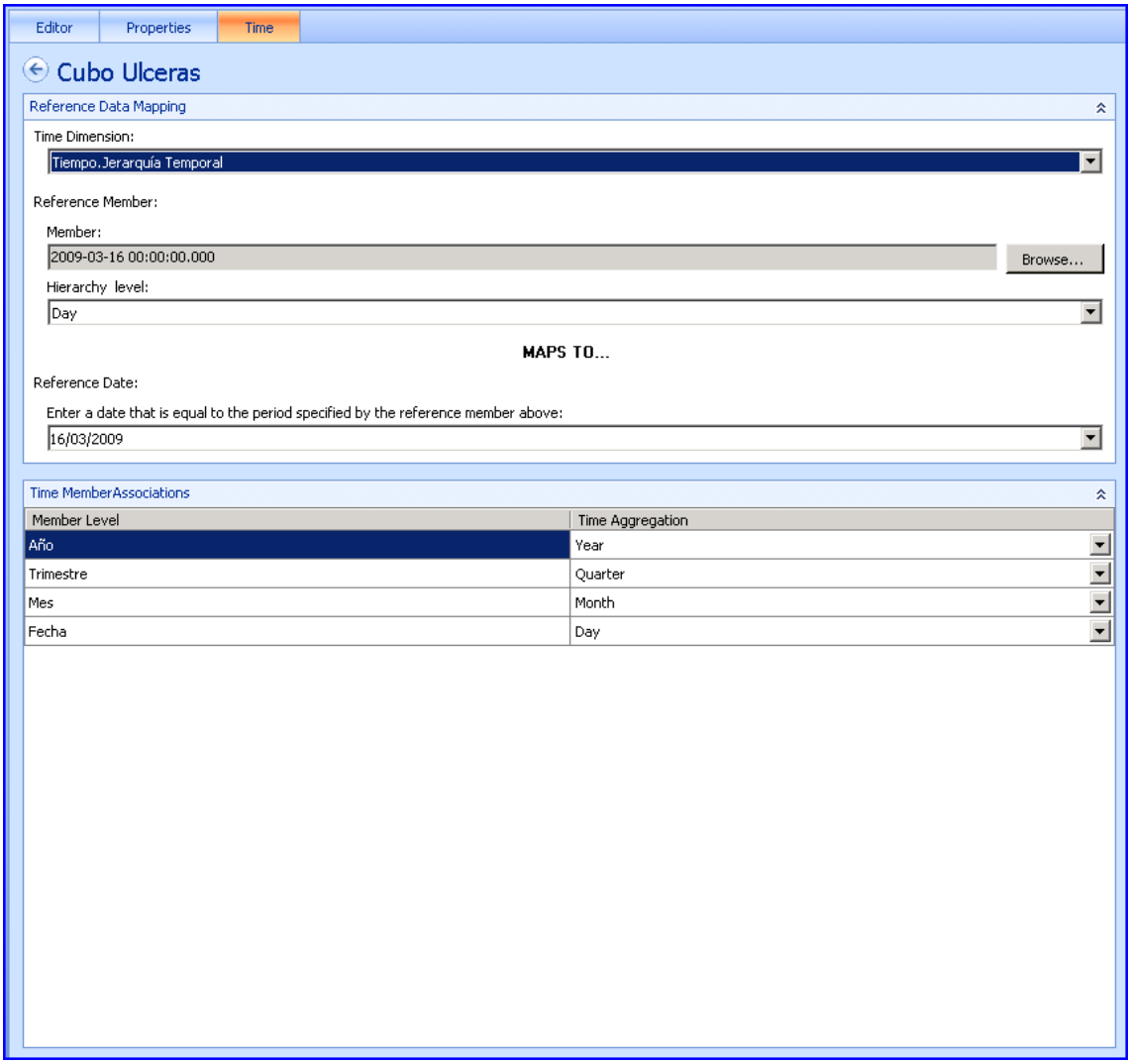


Figura 115.- Pestaña "Time" de una conexión a un cubo de Analysis Services

Como se puede observar, para el correcto funcionamiento de la inteligencia temporal que posee Dashboard Designer de PerformancePoint, se tiene que llevar a cabo un mapeo de dicha dimensión con elementos predefinidos.

En el desplegable "Time Dimension" se debe especificar la jerarquía temporal previamente creada en el cubo, seleccionando un miembro del nivel más bajo como referencia en "Member" (así como cual es el nivel temporal que ocupa en "Hierarchy Level"). Teniendo en cuenta cual es la referencia seleccionada anteriormente, se debe seleccionar dentro del calendario desplegable cual fue el elemento que seleccionamos.

Por último, queda mapear cada uno de los atributos de la jerarquía temporal con una agregación temporal. Siguiendo estos pasos se habrá conseguido que Dashboard Designer interprete la jerarquía temporal y pueda aplicar inteligencia temporal.

Reports

Con casi total seguridad, se puede afirmar que los reports son los elementos de un dashboard que más atención llaman a simple vista. En el Dashboard Designer de PerformancePoint se puede elegir entre un variado abanico de reports que ayudarán a saber el estado de la actividad empresarial. En la Figura 116 se pueden ver todos los tipos de reports con los que se cuenta, y a continuación se estudiarán los más representativos.

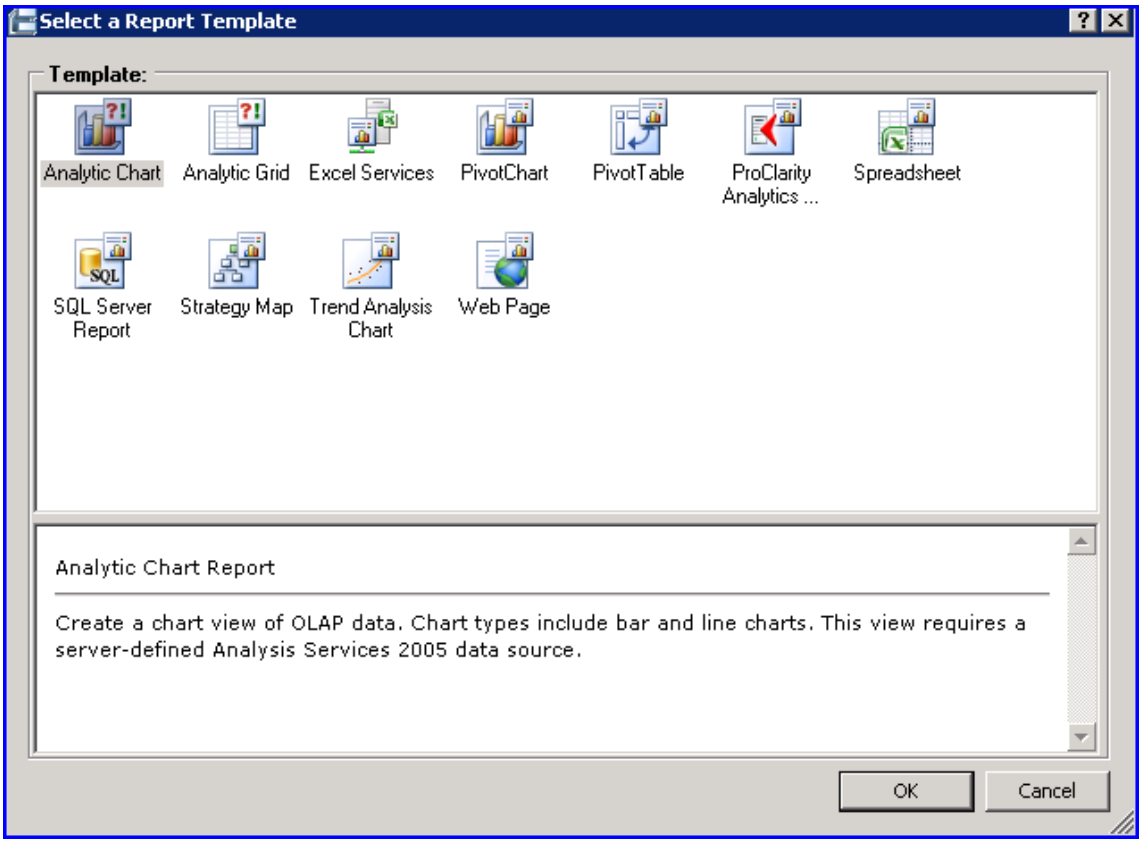


Figura 116.- Clases de reports disponibles en Dashboard Designer de Microsoft PerformancePoint

Analytic Chart

Este tipo de report se basa en un Data Source previamente creado. Tan solo introduciendo un nombre de report y su Data Source, tendremos ya un área de dibujado en el cual podremos ver plasmado el grafico.

Para mostrar contenido, tan solo debemos arrastrar a las posibles partes medidas y dimensiones, consiguiendo asi un gráfico. En la Figura 117, se ha arrastrado la dimensión “Úlceras Localización” a “Series” y “Número Úlceras” a “Bottom Axis”. El resultado se puede ver a continuación.

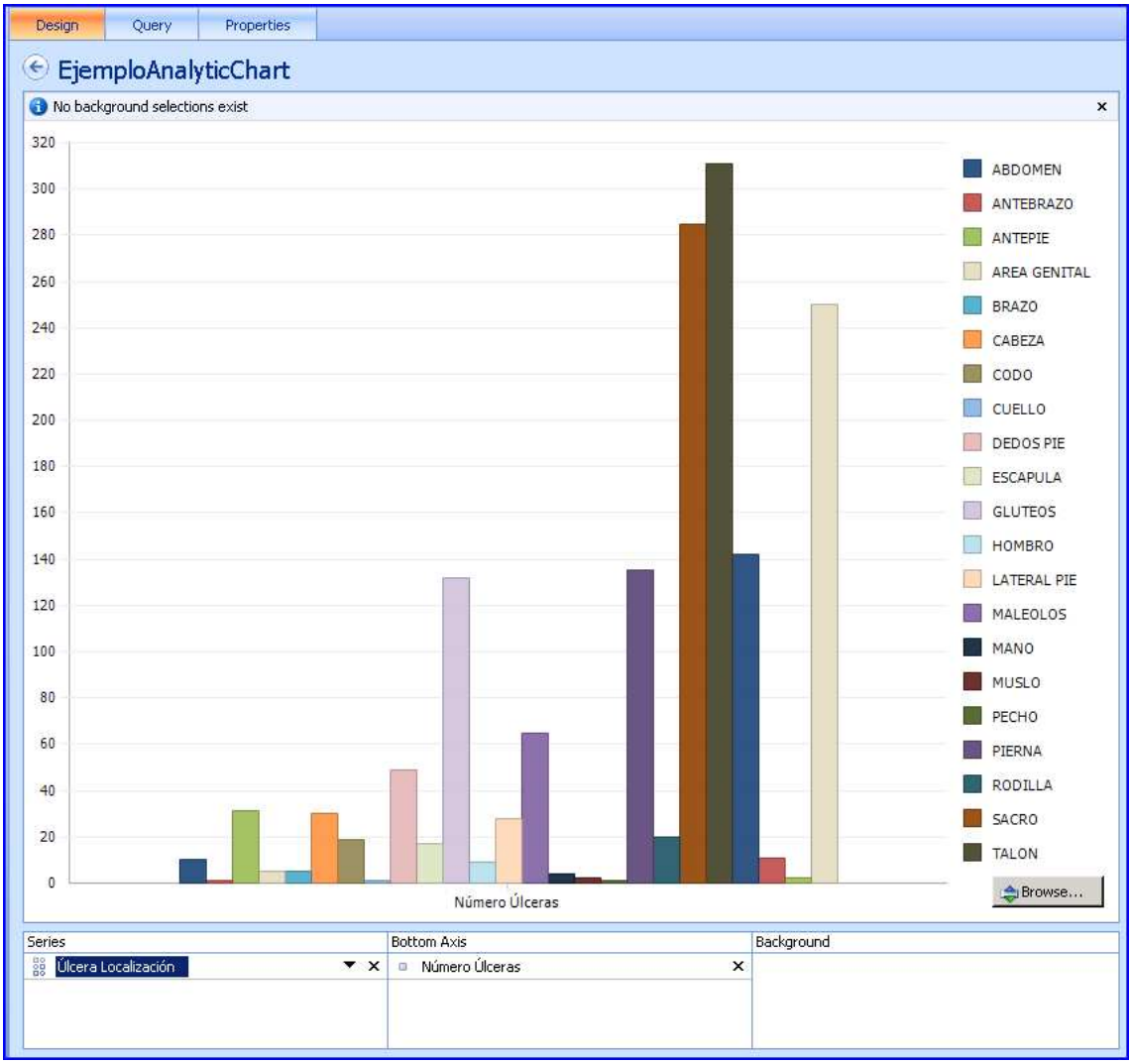


Figura 117.- Ejemplo de Analytic Chart

Cabe destacar que existe la posibilidad de navegar por el grafico con la opción Browse, pudiendo seleccionar entre distintos tipos de grafico. En la siguiente imagen se puede ver cómo, a partir de Úlcera Localización, se pueden desagregar los datos de aquella columna que seleccionemos en cualquiera de las dimensiones disponibles en el cubo. Por ejemplo, si se quieren desagregar los datos de una localización determinada en su distribución por centros, tan solo se debe hacer clic derecho sobre la localización y a continuación “Drill Down To” “Centro”, como se muestra en la Figura 118, mostrando en una nueva grafica los datos desagregados, como en la Figura 119.

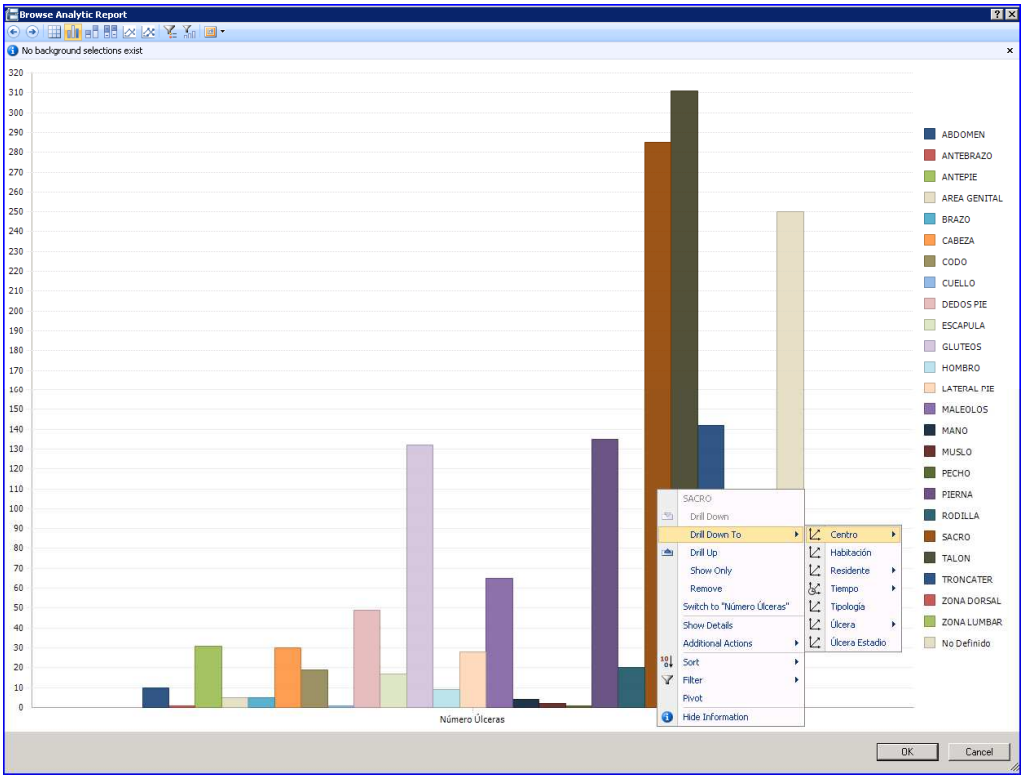


Figura 118.- Analytic Chart previo al "Drill Down To" de la dimensión centro

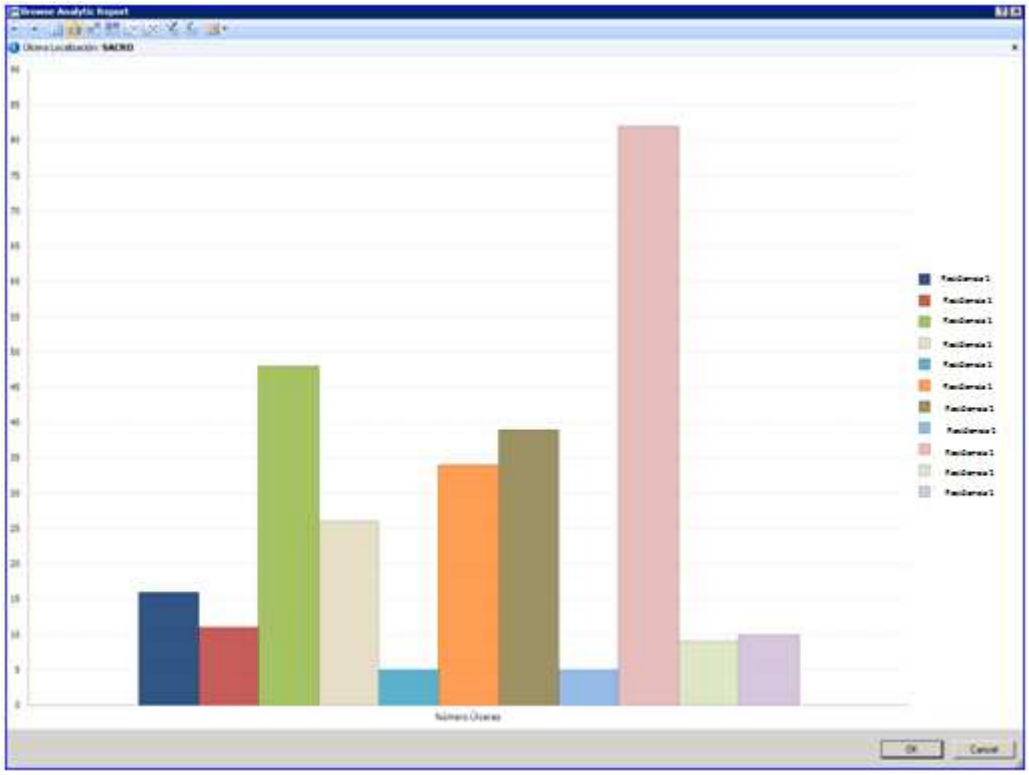


Figura 119.- Analytic Chart con las úlceras en la localización sacro divididas en centros

En la gráfica se puede observar la distribución de las distintas úlceras cuya localización es SACRO en los distintos centros.

Cabe destacar que se puede agregar más de una dimensión para obtener gráficos de distinta índole, y que ninguna de las características anteriormente vistas pueden ser ocultadas de cara al usuario final (para que dicho gráfico permanezca estático).

Pivot Chart

Si hubiese que establecer un símil, este tipo de report seria el sucedáneo a crear un report a partir de los datos de una hoja Excel. Es más, estos reports se basan en una conexión ajena a cualquier Data Source. Para crearlo, tan solo se debe introducir un nombre quedando una pantalla tal como esta.

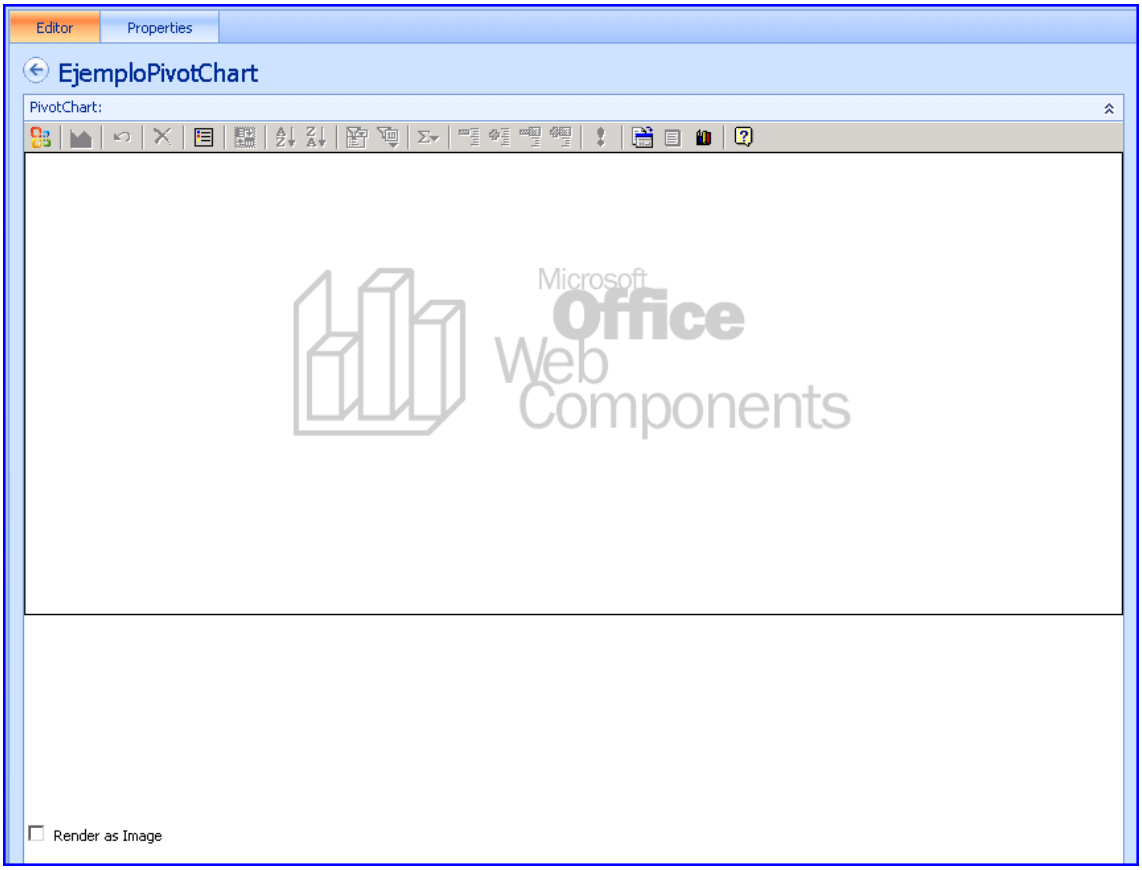


Figura 120.- Pivot Chart vacio

Una vez hecho esto, se debe especificar una cadena de conexión para que se recojan los datos. Esto se hace pulsando el botón “Chart Wizard”, y seleccionando la opción “Data from a database table or query”. En la pestaña “Data Details” se debe especificar la cadena de conexión así como el cubo o tabla al cual se va a atacar. También en “Type” se puede seleccionar el tipo de grafico a visualizar.

En cuanto se acepta, está a disposición del usuario una “Field List” (de la cual se pueden arrastrar medidas y dimensiones al grafico), y un cuadro de “Command and Options” con el cual configurar la accesibilidad que se le dará al usuario. Un ejemplo de “Pivot Chart” sería la Figura 121.

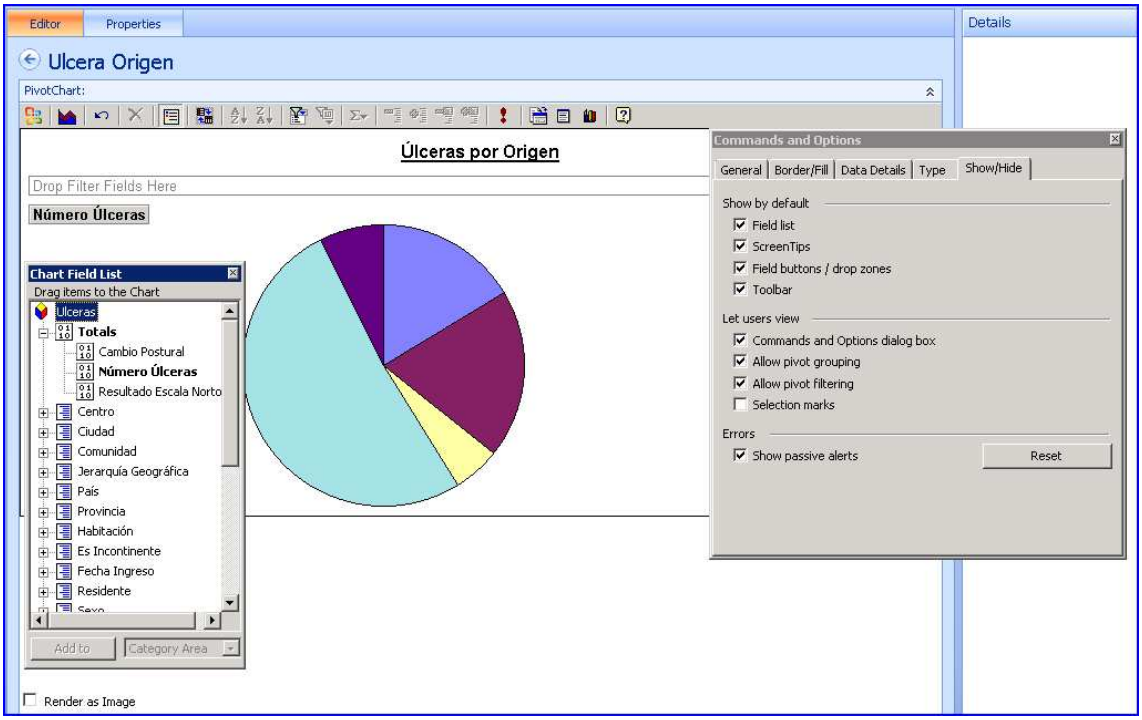


Figura 121.- Ejemplo de Pivot Chart

En la ventana “Command and Options” se puede habilitar / deshabilitar las opciones que se deseen que pueda ver el usuario. Entre dichas opciones, destacan “Field List” (lista de campos del cubo que se podrán arrastrar hasta las zonas de la grafica), “ScreenTips” (las pistas acerca de la gráfica, tales como el numero de úlceras en este ejemplo), “Field Buttons / Drop Zones” (zonas donde el usuario podrá arrastrar medidas desde el “Field List”), “Toolbar” (Barras de Herramientas donde el usuario podrá cambiar cosas tales como el tipo de grafico) y “Command and Options Dialog Box” (cabe destacar que si se deshabilita esta opción, luego estará inaccesible para siempre la configuración del grafico, por lo que hay que estar seguro de las opciones seleccionadas).

Pivot Table

Este tipo de informe ofrece la misma funcionalidad que la que puede tener un Excel a la hora de elaborar Pivot Table. Si se establecen correctamente las propiedades del mismo, se obtiene absolutamente la misma funcionalidad que esta ofrece (pero con la ventaja de no tener hoja Excel y poder ejecutarla en el propio entorno web).

Para crearla, se deben seguir los mismos pasos seguidos con el Pivot Chart (crear el objeto y establecer la conexión que va a utilizar el mismo), obteniendo como resultado la Figura 122.

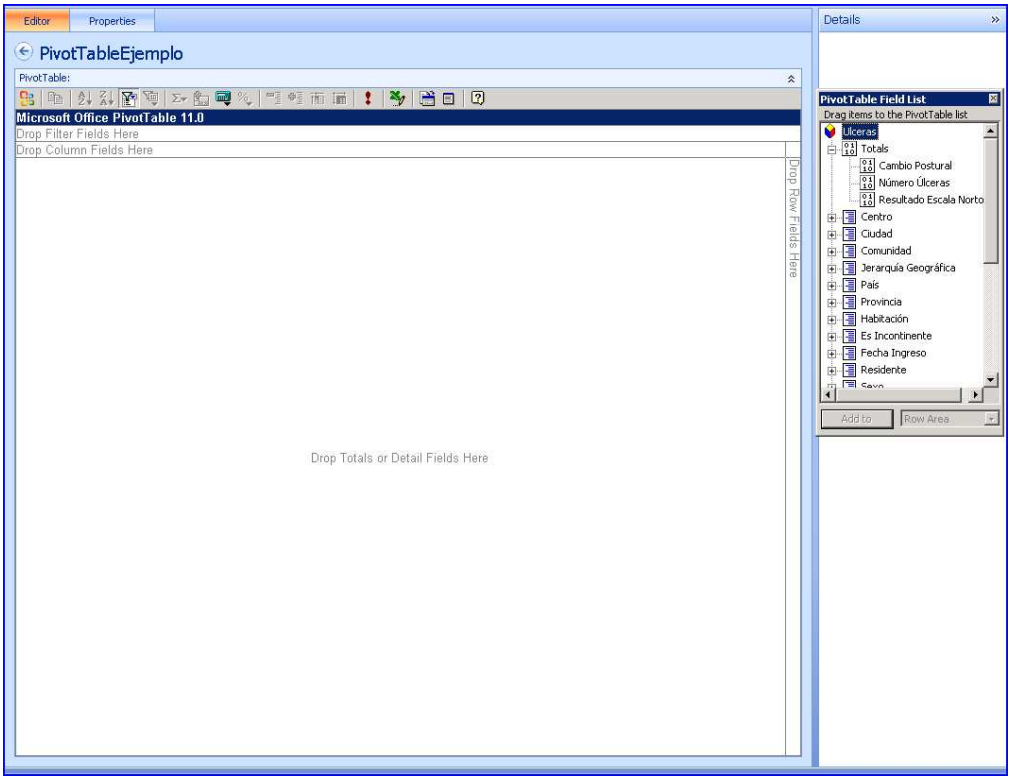


Figura 122.- Ejemplo de Pivot Table vacio

Como se muestra en la imagen, existen zonas donde colocar dimensiones (filas, columnas y filtros) y medidas del hecho (zona central). Configurando correctamente el informe y ajustando sus propiedades (las cuales son similares a las del objeto PivotChart), se puede obtener un informe como el mostrado en la Figura 123.

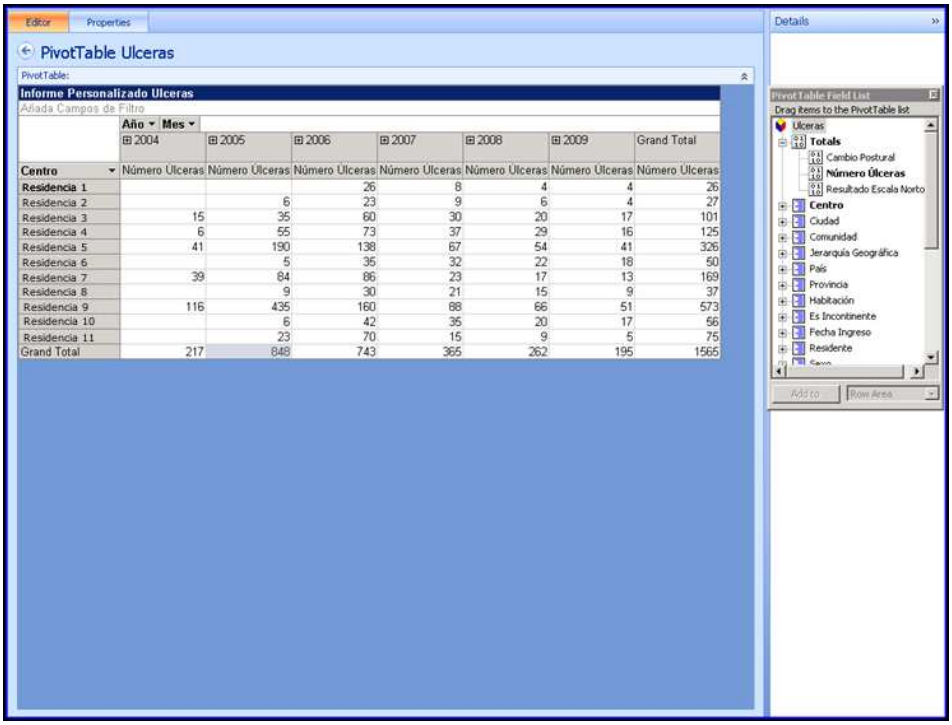


Figura 123.- Ejemplo de Pivot Table que muestra el número de úlceras por año y centro

El cual nos muestra las úlceras existentes cada año en cada residencia, siendo además posible desglosarlas por mes. Además, las opciones “Field List” y “Drop Zones” están activadas de modo que el usuario puede arrastrar nuevos campos a la lista, quitar los ya existentes, añadir filtros… Con el cometido de que elabore el informe a su gusto.

Indicators

Un indicador, en este contexto, es una serie de imágenes que, relacionadas a un determinado KPI, mostrando de un vistazo cual es el estado de la empresa en un ámbito.

Como se puede ver en la Figura 124, Dashboard Designer de PerformancePoint ofrece una gran cantidad de indicadores para elegir el que más se adapte a nuestras necesidades.

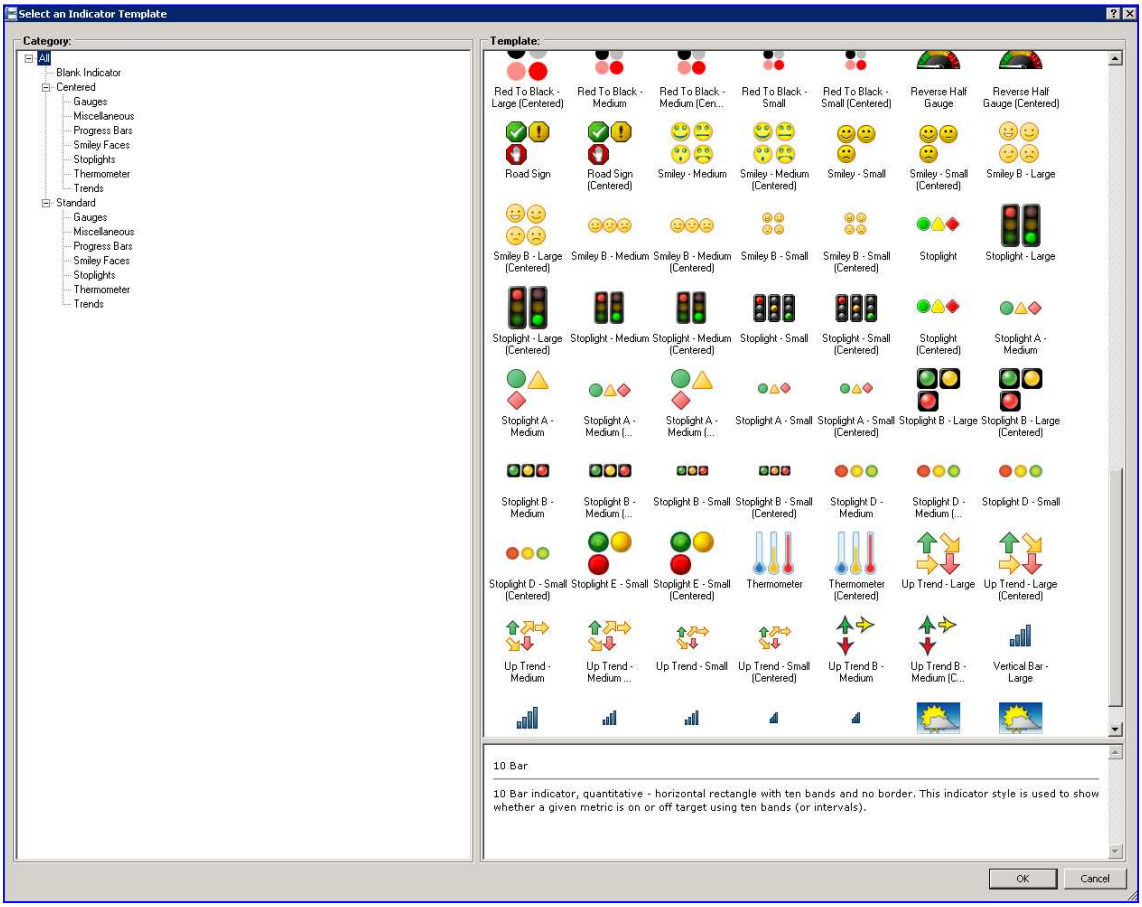


Figura 124.- Ejemplos de indicadores en Microsoft PerformancePoint

Su creación es extremadamente sencilla, por lo que se obvia en esta guía dichos pasos. Lo único a reseñar es que, una vez elegido el diseño, se cree el mismo indicador tanto de tipo “Centered”, como de tipo “Standard”, para conseguir un mejor acabado de los Scorecard (objetos que se estudiaran más adelante y que permiten visionar los KPIs con sus correspondientes indicadores).

KPIs

Un KPI (Key Performance Indicator) es una serie de valores-objetivo que se marcan para un determinado conjunto de medidas de hecho y dimensiones al objeto de conseguir saber si la actividad empresarial se está llevando a cabo correctamente. A su vez, a estos se les asignan indicadores para que así se sepan sin llegar a ver los valores si se están cumpliendo los objetivos o no. Dichos KPIs serán agregados a un Scorecard donde se podrán visionar los datos

Cuando se intenta crear un KPI nuevo, se pueden elegir dos tipos: “Blank” o “Objective”. A lo largo de este documento solo se estudiarán los “Blank KPI”. Si se selecciona dicho tipo, se debe introducir el nombre del mismo, y comenzar a configurarlo. La pantalla mostrada es la de la Figura 125.

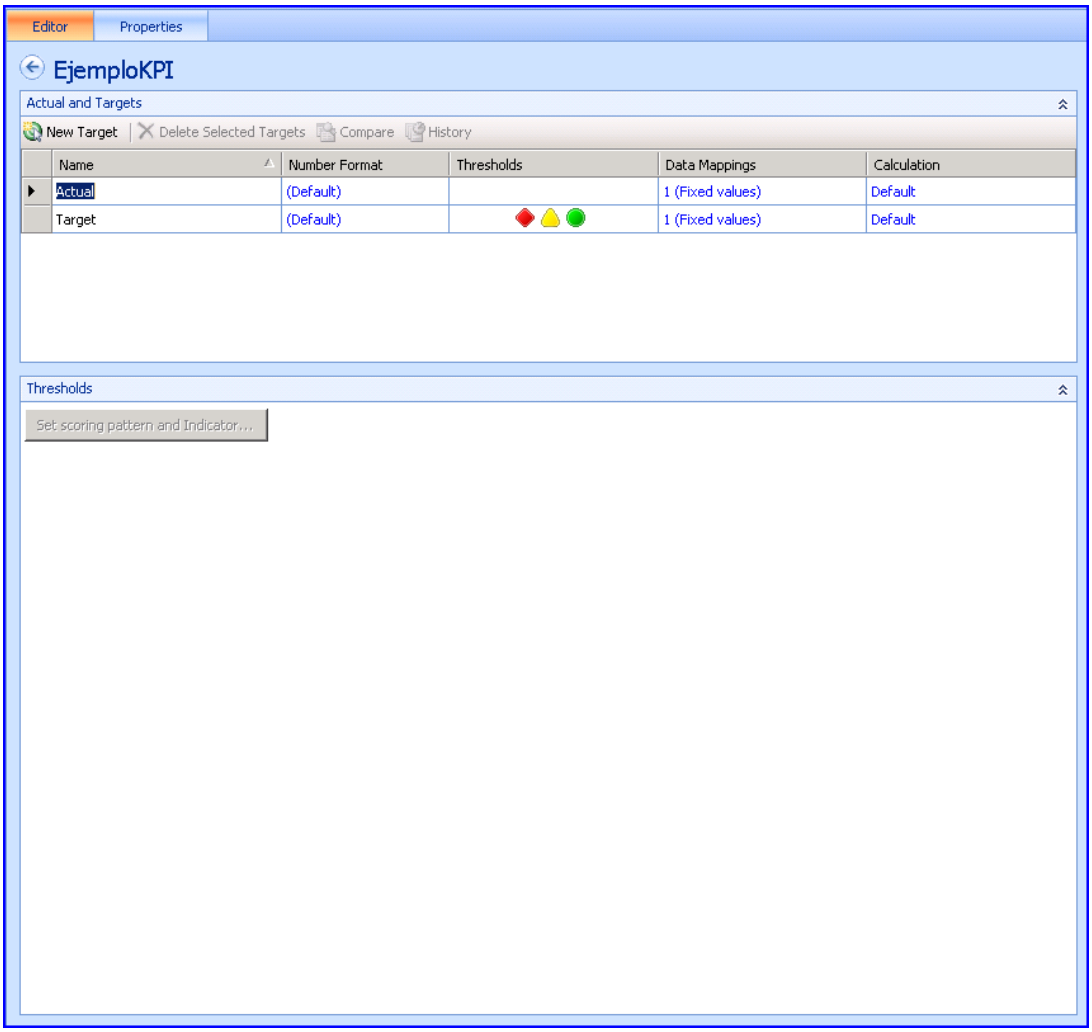


Figura 125.- Pantalla inicial de la creación de un KPI

Como se puede observar, tenemos dos filas: una llamada “Actual” y otra llamada “Objective”. En “Actual” se puede conocer el valor actual del dato mapeado, mientras que en “Objective” se podrá ver este mismo valor acompañado de un KPI que seleccione el usuario a su gusto

La primera propiedad a cambiar debe ser “Number Format”, eligiendo uno a gusto del usuario (normalmente con dos decimales tenemos el detalle necesario). En segundo lugar, se debe mapear

el dato (“Data Mappings”) a estudiar en el Scorecard. Este puede ser un valor fijo o por el contrario, un mapeo a una medida dentro de un cubo (normalmente el caso que más interesa).

Por otro lado, se debe definir el campo “Calculation”, dejando este varias opciones entre las cuales elegir: “Default”, “No Value”, “Source Data”, “Average of Children”… siendo esta última casi siempre la más recomendable (ya que muestra la media de los hijos).

Una vez configurado esto, se debe relacionar un Indicador al “Target” del KPI y los valores relacionados a cada imagen del indicador. Para ello, se selecciona el “Target” y a continuación se presiona el botón “Set Scoring Pattern and Indicator”.

En el primer paso se debe indicar si el indicador es cuanto más mejor, cuanto menos mejor o contra más cerca del objetivo mejor (“Increasing is Better”, “Decreasing is Better”, “Closer to Target is Better”) (Figura 126).

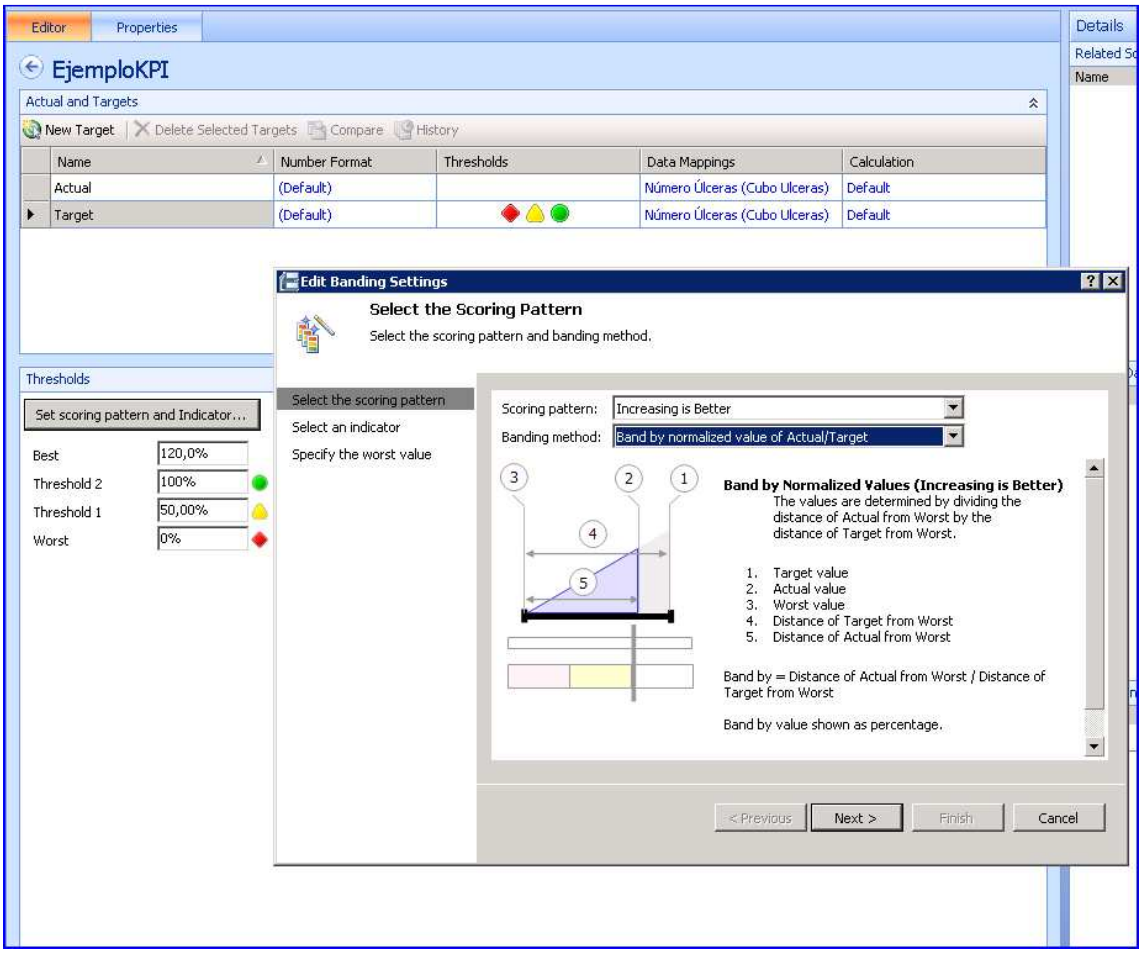


Figura 126.- Primer paso de la creación de un KPI

A continuación, se selecciona el “Banding Method”, cuyas posibilidades son “Band by Normalized Value” (donde se establece un valor base y unos porcentajes que superen/incrementen dicho valor) o “Band by Numeric Value” (estableciendo valores numéricos). La última opción es útil para los “Objective KPI”, por lo que no se estudiará en el presente documento.

En el caso que se está implementando, como se van a estudiar las úlceras anuales se han introducido unos valores acordes a estas, y visibles en la Figura 127.

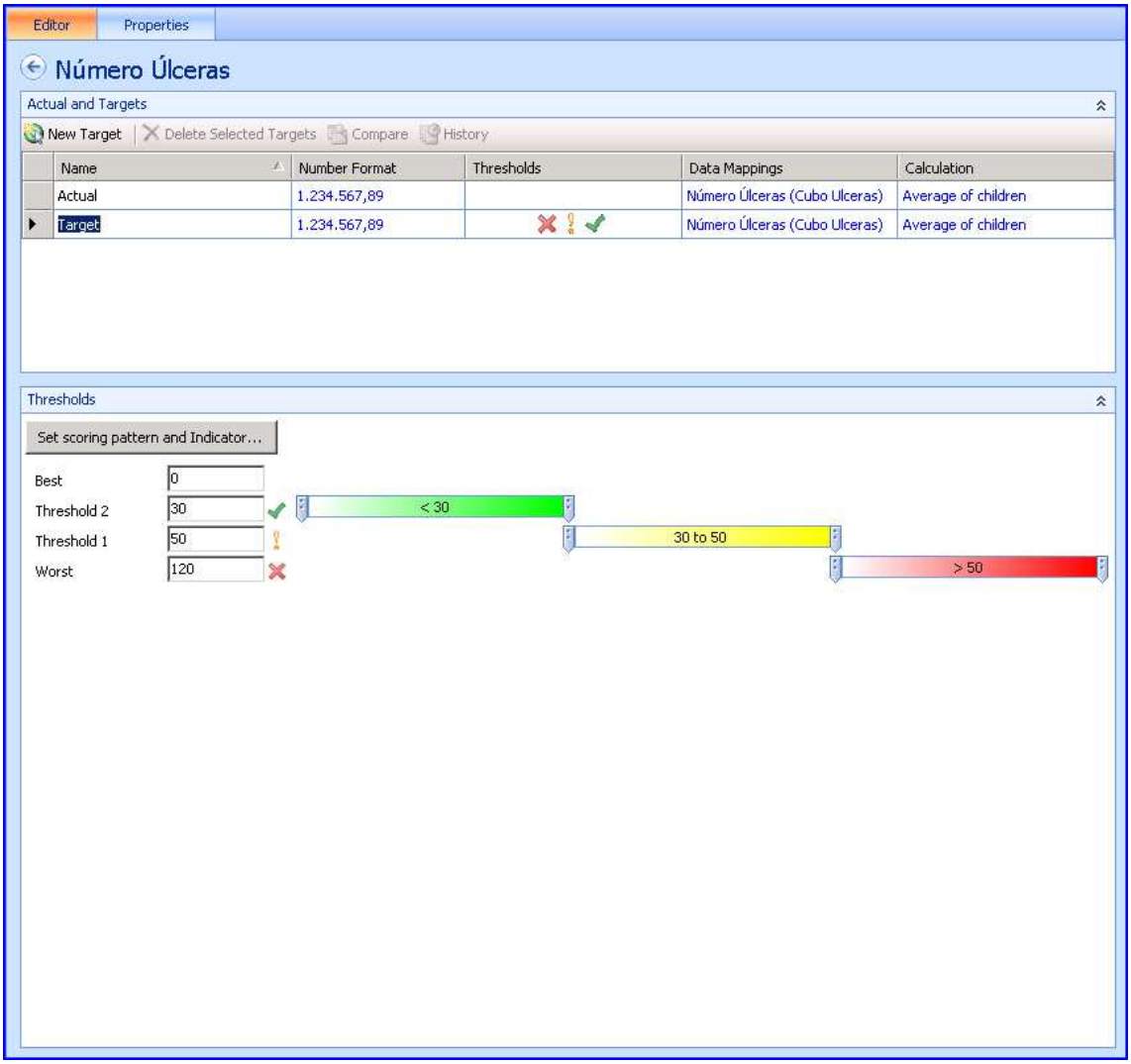


Figura 127.- Captura de los valores introducidos para el KPI

Scorecard

Un Scorecard es un informe tabular que muestra datos extremadamente importantes de la empresa, que indican el funcionamiento de la misma, acompañados de KPIs para saber si estos datos van cumpliendo los objetivos empresariales.

Lo más cómodo en el caso estudiado es crear un Scorecard basado en Analysis Services. En el primer paso se debe introducir un nombre de Scorecard, y a continuación el Data Source utilizado (cubo del cual se extraen los datos).

En el tercer paso, se tiene la opción de importar KPIs directamente desde un cubo, o crearlos en el propio Dashboard Designer. En el caso que se elija la segunda opción, podremos crear nuevos KPIs en el Dashboard Designer o coger alguno ya creado.

Los siguientes pasos es conveniente no seguirlos, de forma que así quedará un informe nuevo que el cliente podrá configurar a su gusto.

En la Figura 128 se muestra un Scorecard en el cual se muestran las Úlceras que ha habido a lo largo de la historia de toda la residencia.

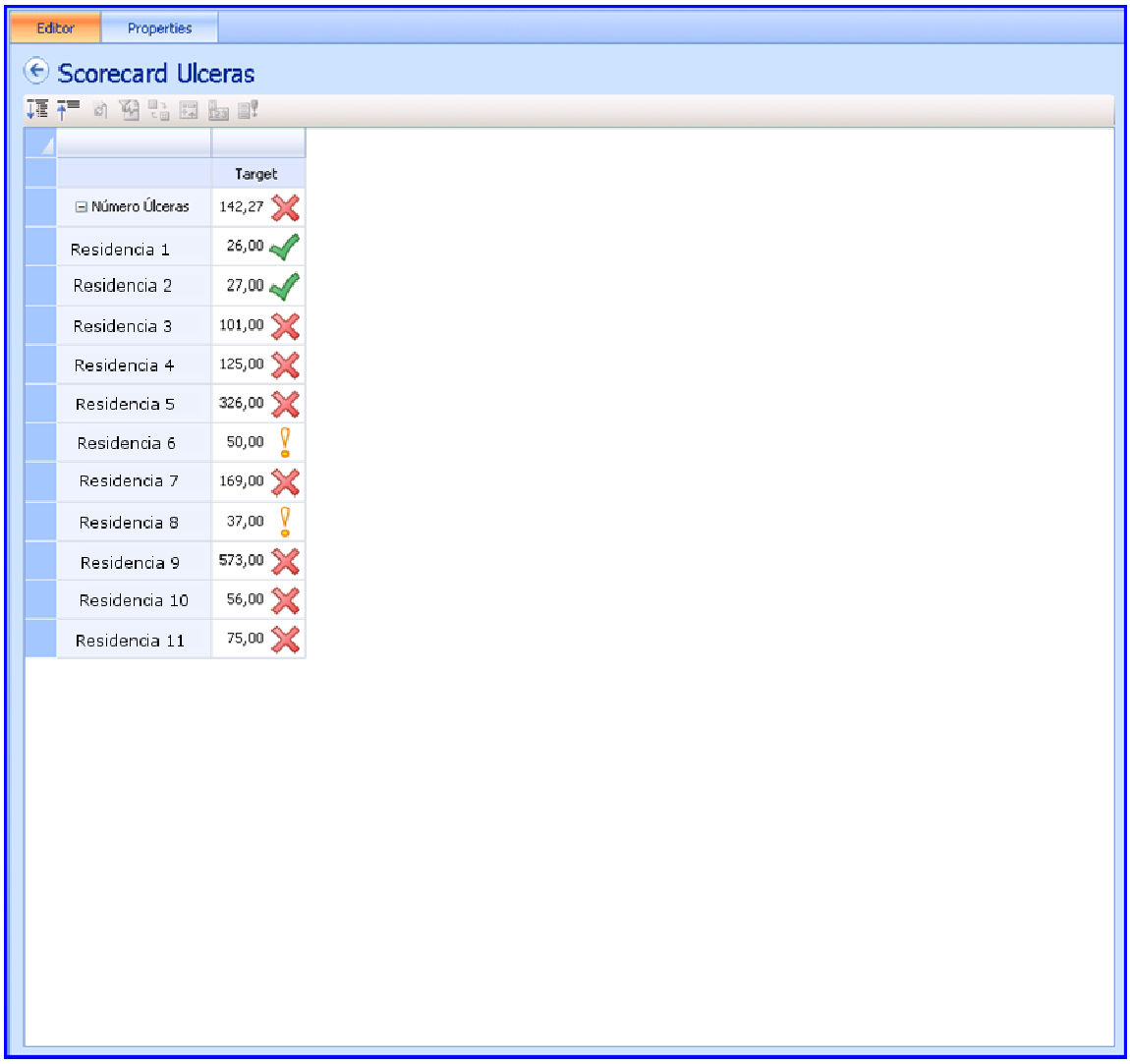


Figura 128.- Scorecard de úlceras totales en las distintas residencias

Dentro del Dashboard se añadirán filtros para que estos datos (totales de la historia de la residencia) coincidan con los del KPI (anuales por residencia).

5.5.3.- Creacion del Dashboard

Aspectos Generales

Un dashboard, en el contexto de PerformancePoint, se puede definir como un contenedor de todos los objetos que se han estudiado anteriormente. En el dashboard se pueden añadir dichos elementos e interactuar con ellos al objeto de conseguir toda la información necesaria.

Un mismo dashboard puede estar compuesto por distintas páginas, que pueden ser utilizadas, por ejemplo, para mostrar objetos de cada tipo. Por ejemplo, imagínese un dashboard de úlceras, del

cual se han hecho distintos informes de distintos tipos: informes médicos, de gestión… estos informes podrían ser agrupados por estos tipos en distintas páginas, y así facilitar la navegación por el dashboard.

De cada página se tiene un "dashboard content", como se observa en la Figura 129: aquí es donde se introducirán los distintos objetos para construir el dashboard. Al igual que en otras aplicaciones de Microsoft, el contenido se puede dividir en zonas, y en estas zonas agregar más de un objeto. Por ello, es harto importante establecer lo que va a ocupar cada objeto para que el dashboard tenga una apariencia óptima.

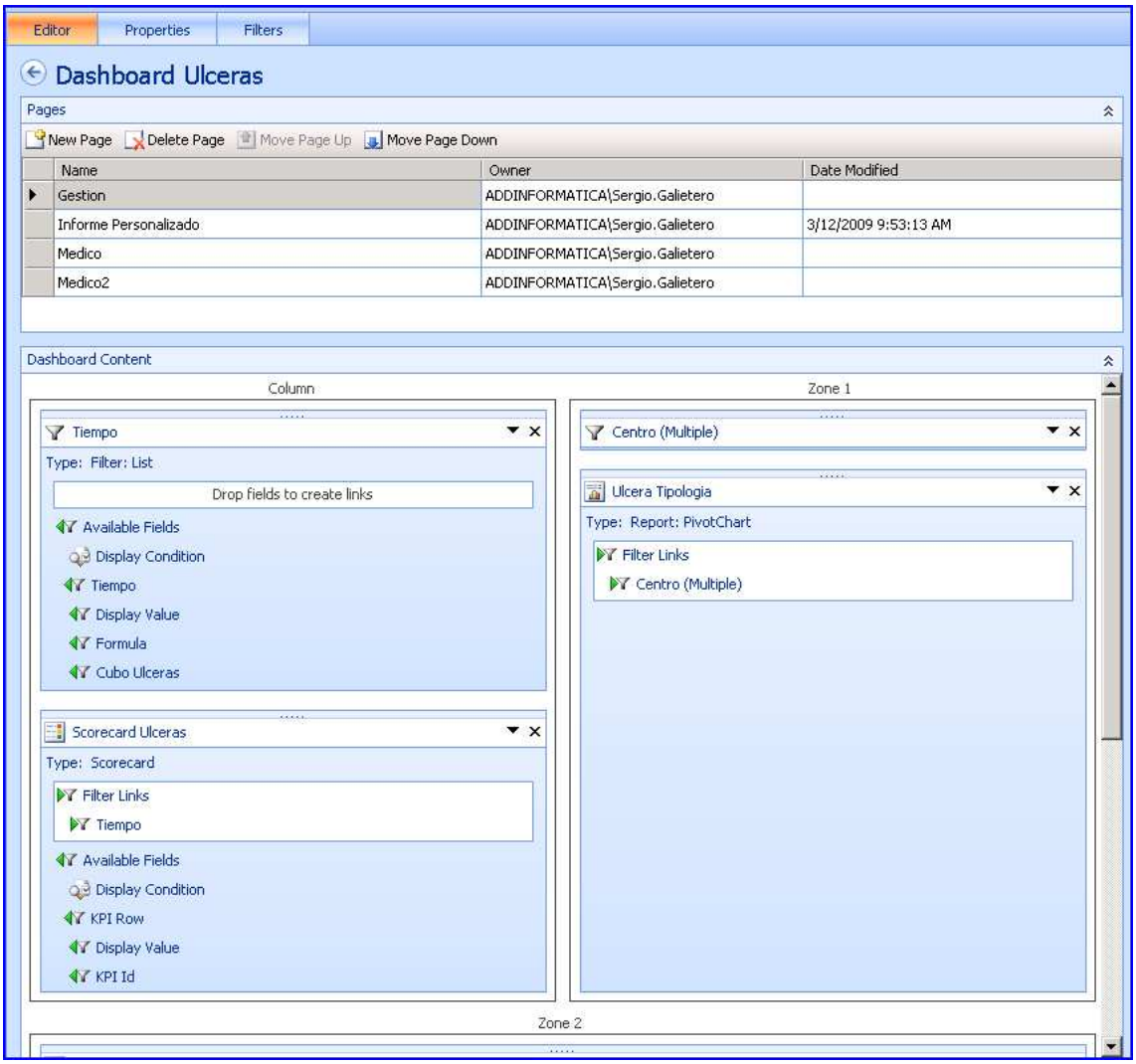


Figura 129.- Ejemplo de contenido de un dashboard

Para editar el tamaño de las zonas, tan solo se debe hacer clic derecho sobre la zona y establecer las propiedades deseadas. Por otro lado, si se quieren establecer las del objeto, se debe pulsar el triangulo que está al lado del aspa (el cual retira el objeto del dashboard).

En el ejemplo actual, se ha introducido una primera página con un par de filtros (que veremos a continuación), un scorecard y dos reports, obteniendo el resultado que se puede observar en la Figura 130 (Edit, Preview):

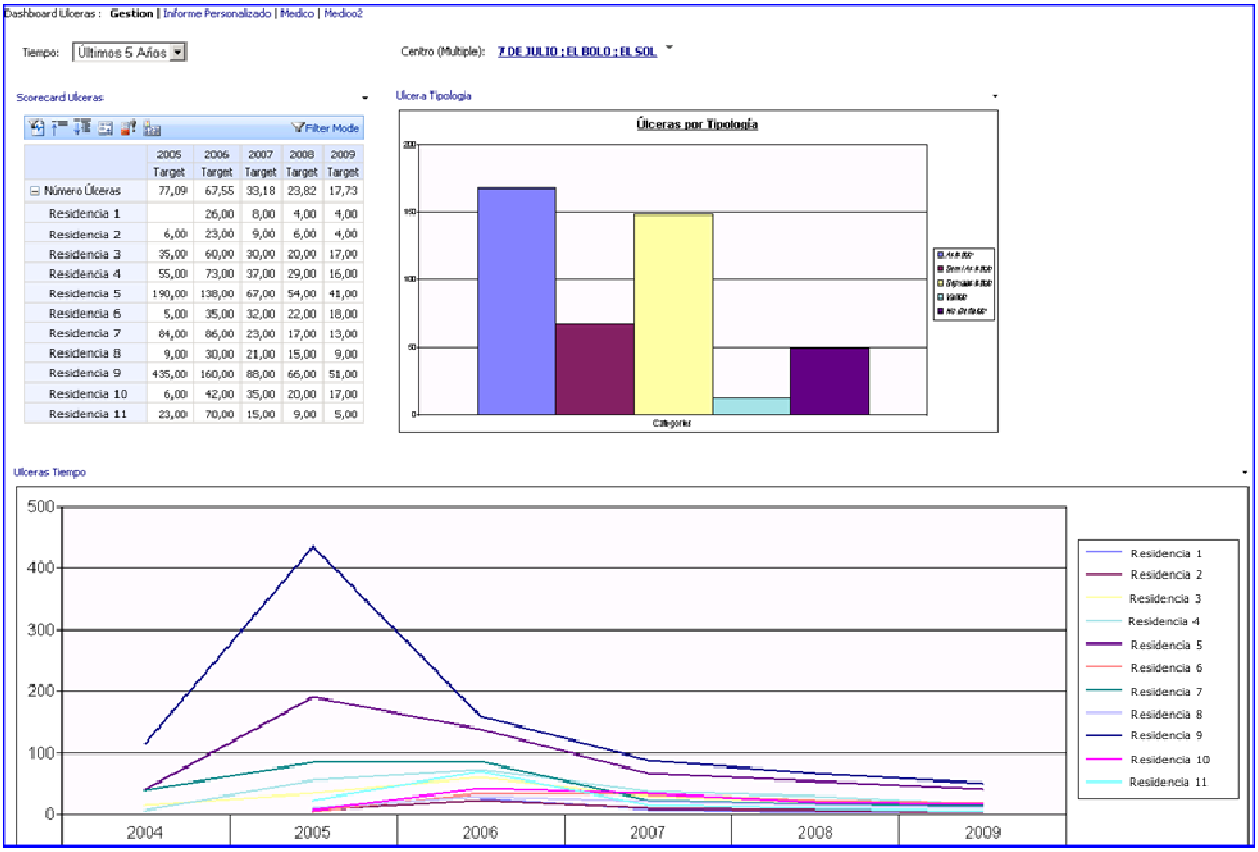


Figura 130.- Ejemplo de dashboard ya construido

Filtros

En la pestaña “Filters” del diseñador, se pueden definir filtros de distintos tipos como se muestra en la Figura 131.

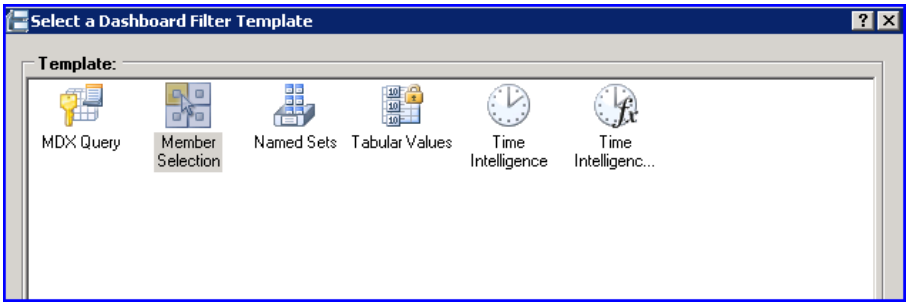


Figura 131.- Tipos de filtros en Microsoft PerformancePoint

Aquí se analizarán los tipos “Member Selection”, ”Time Intelligence” y “Time Intelligence Calendar”.

Member Selection

Este filtro se basa en mostrar datos dimensionados por un miembro o por un conjunto de estos. Para ello, se crea un nuevo filtro de este tipo, se introduce el nombre, su Data Source, y en el tercer paso se selecciona la dimensión en la cual están contenidos los miembros, así como los miembros que se desean mostrar en la lista.

En el siguiente paso se seleccionará el tipo de lista a visionar, entre los cuales hay tres tipos: “List”, “Tree” o “Multi-Selected Tree”, como se observa en la Figura 132.

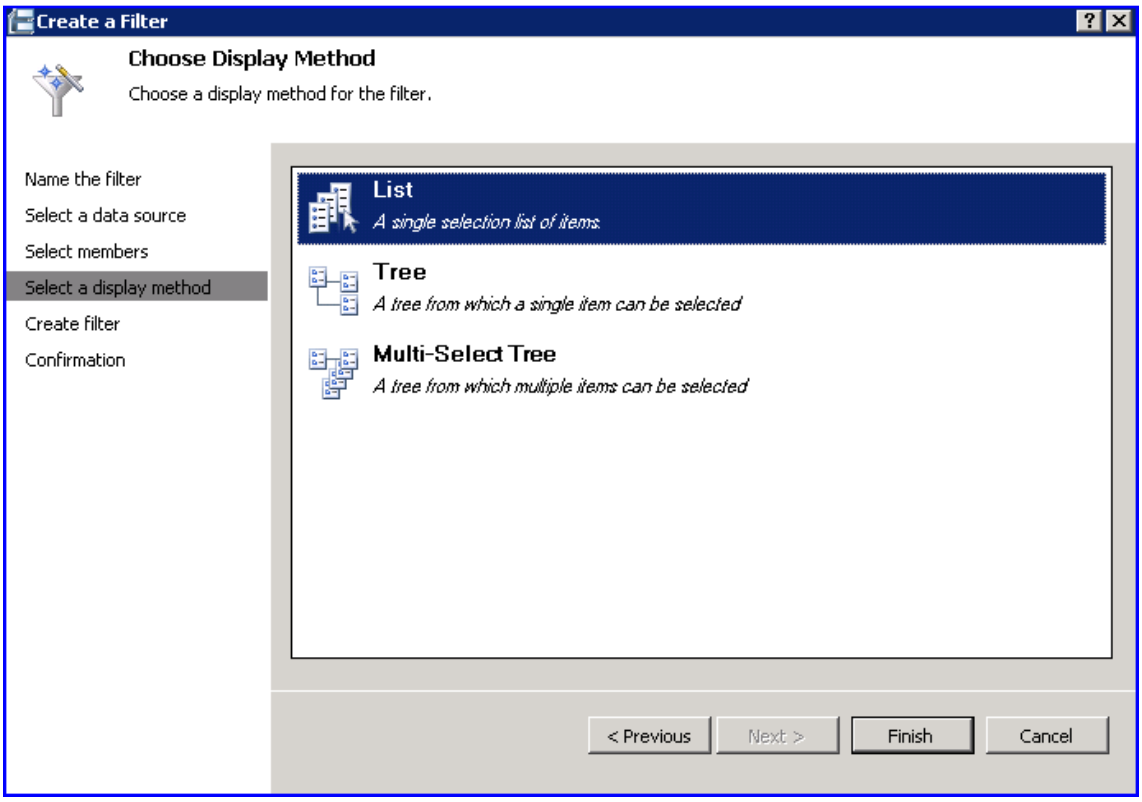


Figura 132.- Ejemplos de visualización de filtros en Microsoft PerformancePoint

El tipo “List” muestra un desplegable del cual solo podemos seleccionar un elemento. “Tree” ofrece la misma posibilidad que el anterior, aunque se visualiza como un árbol. Por último, y quizás más interesante, “Multi-Selected Tree” ofrece la posibilidad de seleccionar distintos miembros y así hacer agrupaciones al antojo del usuario.

Time Intelligence

Este filtro se basa en la inteligencia temporal definida anteriormente dentro de un Data Source. Los dos primeros pasos son iguales al anterior tipo de filtro, siendo la única variación el paso mostrado en la Figura 133.

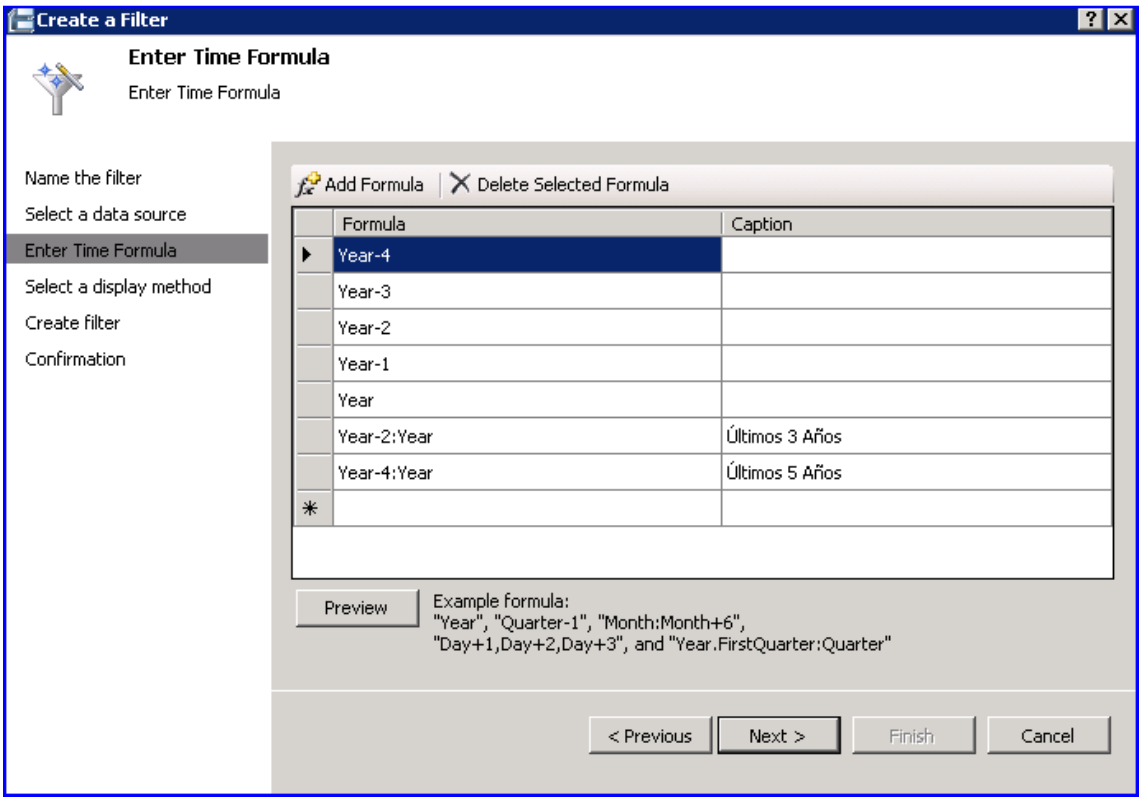


Figura 133.- Paso adicional para la inteligencia Temporal

Como se puede observar, hay una determinada sucesión de formulas con sus correspondientes capturas. Estas formulas son bien fáciles de definir, y conformarán los miembros disponibles dentro del filtro. Para más instrucciones sobre cómo diseñar estas formulas, lea la referencia [PPO1] de la bibliografía.

Time Intelligence Calendar

Este filtro se basa en la inteligencia temporal definida anteriormente dentro de un Data Source, pero a diferencia del anterior caso, se basa en un calendario y en una fórmula que se introducirá una vez enlazado a su objeto correspondiente.

Enlazar Filtros a Objetos

Para hacer esto, tan solo se deben arrastrar los filtros dentro del área de la página de Dashboard, y una vez hecho esto, arrastrar el filtro a los “Filter Links” de un determinado objeto. En la Figura 134 se ve un ejemplo de objetos ya enlazados a filtros.

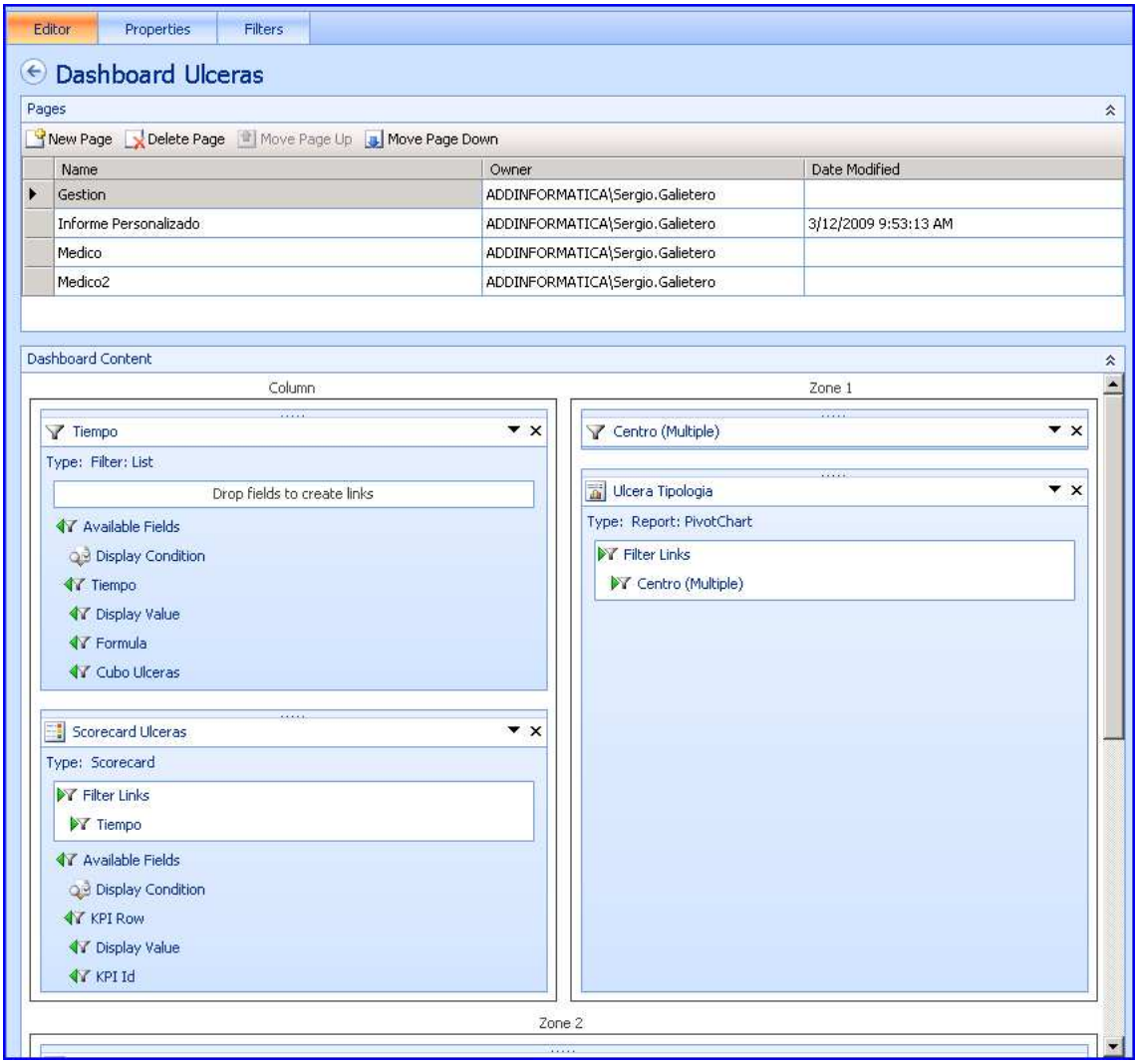


Figura 134.- Ejemplo de objetos ya enlazados a filtros

En dicha figura, el Scorecard “Úlceras” está enlazado al filtro tiempo (parte izquierda del Dashboard), mientras que el Report “Úlcera Tipologia” está enlazado al filtro Centro (Multiple) (parte derecha del Dashboard).

5.5.4.- Publicar el Dashboard en SharePoint

Para publicar un dashboard hecho con Dashboard Designer de PerformancePoint, tan solo hay que hacer clic en la opción correspondiente e indicar el sitio web donde se desea publicar, tal y como indican la Figura 135 y 144.

Figura 135.- Publicación del dashboard en Microsoft SharePoint

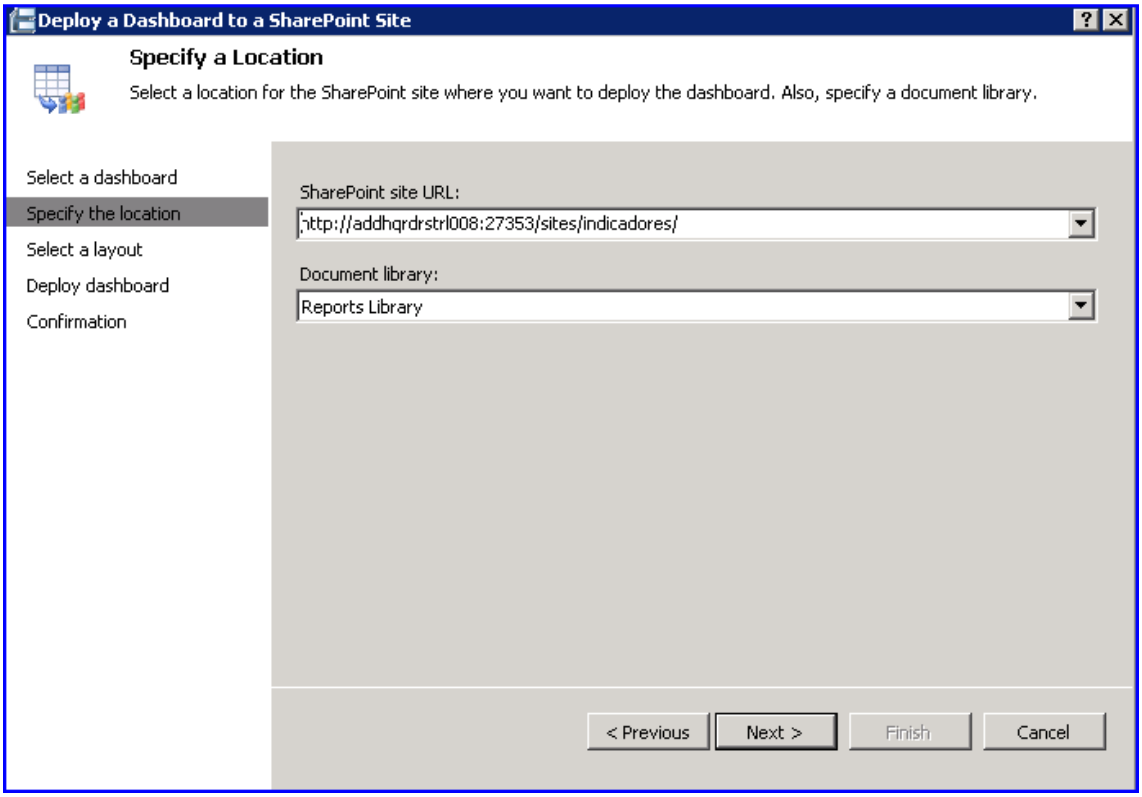


Figura 136.- Introducción del dashboard en un sitio especifico de Microsoft SharePoint

Capítulo 6. Conclusiones

A día de hoy, el acometer un proyecto de Business Intelligence es casi obligatorio para cualquier empresa que se quiera diferenciar con respecto a las demás empresas que realizan una misma actividad. Por ello, en la PYME que se desarrolló este proyecto, se acometió el proyecto de implementar una solución de este tipo partiendo del marco tecnológico enunciado en el capítulo 3 de la memoria. Con éste, la primera disyuntiva que se planteó fue la elección de una herramienta ETL lo suficientemente potente como para crear un conjunto de indicadores bastante grande y que el coste temporal de la ejecución de los mismos no fuese suficientemente elevado. Por ello, y partiendo de la base que Resiplus BE era una aplicación programada y testada por la empresa, se decidió comenzar con el desarrollo de assemblies, cosa que no prosperó debido a la aparición de indicadores de gestión con un volumen de datos desmesurado. Cuando éstos aparecieron, se comenzaron a buscar alternativas, y surgió la de Integration Services y sus paquetes, que aportaban las suficientes ventajas como para cambiar el modelo de implementación.

La primera ventaja que llamó la atención desde la perspectiva de la programación fue la sencillez de implementación de los indicadores con este método, ya que la interfaz visual en forma de workflows ayuda mucho al desarrollador, teniendo que desentenderse en casi todo momento de complejas instrucciones de programación, base de la implementación con assemblies.

Una vez implementado el primer paquete, surgieron dos ventajas significativas más: la primera de ellas fue la sencillez con la cual los fallos se pueden localizar con este método de implementación, ya que usan la consola misma de SQL ofreciendo un informe detallado de donde y cuando ha fallado la ejecución, mientras que los fallos obtenidos con los assemblies se reflejaban en la interfaz de Resiplus BE, pero sin ofrecer en cuanto apenas información del error surgido. Además, se compararon los costes temporales de la ejecución del mismo indicador desarrollado mediante el método de assemblies y mediante el método de Integration Services, siendo la ejecución de éste segundo mucho más rápida, y sin dejar duda sobre afirmaciones que se encontraban por la red: Integration Services está implementado para realizar consultas y transacciones SQL de una manera completamente optimizada, por lo que es difícil que otra herramienta no orientada completamente al uso de estas transacciones ofrezca unos resultados tan buenos como Integration Services.

Si se acude a las entrañas de ambas implementaciones, se puede observar que para el método de assemblies hay que para realizar operaciones realmente recurrentes en el proceso de creación de indicadores, llevándose a cabo muchas funciones realmente costosas, estando el programador muy atado de manos a la hora de ofrecer soluciones a problemas de este tipo. Sin embargo, con el método de Integration Services el desarrollador puede dar rienda suelta a su imaginación creando optimizaciones para la utilidad que desea implementar: un ejemplo de esto fue el desarrollo de tablas auxiliares, que mejoró mucho la eficiencia en la ejecución de los paquetes desarrollados. Además, a la hora de implantar los indicadores en el cliente era mucho más sencilla la adición de un paquete a una librería de paquetes para que un servicio Windows lo ejecutase sin hacer nada más que la adición de exportaciones y programaciones dentro de Resiplus BE.

Con todas estas ventajas que se han enumerado, en la empresa se pasó de un desarrollo de assemblies para Resiplus BE a un desarrollo de paquetes de Integration Services, los cuales demostraron que ofrecían mucha más funcionalidad, sencillez y rapidez que los primeros.

Una vez decidido el método por el cual realizar las operaciones de ETL, había que estudiar los diferentes métodos de explotación del almacén creado. Para ello, se decidió que parte fundamental del desarrollo sería la creación de cubos de datos, al objeto de, posteriormente, realizar una mejor explotación.

Para comenzar, se estudio la explotación mediante Microsoft Excel, la cual ofrecía una gran variedad de características que contentaron a la empresa: familiaridad con la aplicación, usabilidad, efectividad, etc. todas ellas hacían parecer que Microsoft Excel sería la gran candidata a modelo de explotación del almacén de datos.

Pero después de estudiar posibles alternativas, se llegó a la conclusión que es mucho mejor tener un resultado mucho más directo y visual que el ofrecido por Microsoft Excel, el cual nos permite crear muchas más variantes, pero que para una persona no acostumbrada al uso de informes como los construidos puede que el proceso de explotación se le haga cuesta arriba. Por ello, se comenzaron a estudiar alternativas para ofrecer a los clientes cuadros de mando o dashboards, descubriendo que se tenían dos posibilidades para acometer el desarrollo de estos objetos dado el marco tecnológico ofrecido por la empresa: Microsoft Office SharePoint Server y Microsoft Performance Point Server.

Del primero, destacar que ofrece la posibilidad de que los dashboards sean implementados en el mismo SharePoint, siendo una contrapartida el que los dashboards deben estar basados en archivos de Microsoft Excel, con todas las limitaciones que esto acarrea: limitaciones de gráficos, no interactividad de los mismos, etc. a pesar de que el hecho de estar disponibles para su explotación en un entorno web sea una ventaja significativa

Sin embargo, la segunda alternativa estaba basada directamente en conexiones a los cubos de datos, ofreciendo unas ventajas bastante significativas con respecto a la SharePoint: nuevos tipos de gráficos, construcción de dashboards de múltiples páginas, interactividad total con los informes desarrollados, posibilidad de ofrecer la misma funcionalidad que los archivos de Microsoft Excel, etc. y todo mediante una sencilla interfaz de desarrollo y sin tener que depender de archivos Excel previamente creados, como sucedía con SharePoint, y siendo explotables mediante la interfaz web de este CMS.

Y quizás este sea el contrapunto más grande de PerformancePoint: el hecho de que no sea independiente, sino que dependa de SharePoint es una gran desventaja, ya que SharePoint es una utilidad con un coste monetario elevado, y no muchos clientes de los indicadores de gestión están dispuestos a acometer en un coste tan elevado. Por ello, y a pesar de pensar que las soluciones visuales (SharePoint y PerformancePoint) ofrecen muchas más ventajas que Microsoft Excel, el método de explotación utilizado por la mayoría de empresas es este último, ofreciendo para ello al cliente una pequeña formación de cómo utilizar esta clase de archivos.

Con todo esto, se ha visto como desarrollar una solución Business Intelligence dentro de un marco tecnológico real, ofreciendo en el presente todas las alternativas estudiadas para su implementación y explotación.

Según lo descrito antes, los objetivos del proyecto final de carrera han sido cumplidos con éxito: se ha expuesto de una manera detallada el marco tecnológico donde se ha desarrollado el proyecto, se han estudiado las alternativas para el desarrollo de indicadores de gestión, ejemplificando cada una de ellas y ofreciendo conclusiones obtenidas a partir del desarrollo de los mismos, y por último se ha ofrecido una visión pormenorizada de distintas herramientas útiles para la explotación de la información conseguida a partir del desarrollo de indicadores de gestión, concluyendo los pros y los contras de cada una de estas herramientas.

Por último, es importante destacar el desafío que ha supuesto este proyecto final de carrera para el autor, ya que en un principio partía con muy poco conocimiento acerca de las tecnologías de Business Intelligence, pero paso a paso, y con la colaboración de mi director de proyecto y otros compañeros pienso que he alcanzado un buen grado de conocimiento de las tecnologías acompañado de una valiosa experiencia.

Capítulo 7.- Bibliografía

[BUS1] Información extraída de la página http://www.sinnexus.com/business_intelligence/

[BUS2] Información extraída de la página

http://www.ibermatica.com/publicaciones/BusinessIntelligence.pdf

[BUS3] Información extraída de la página http://www.businessintelligence.info/

[BUS4] Hernández, José; Ramírez, María José.; Ferri, César - "Introducción a la Minería de Datos" - Pearson Prentice Hall, 2004.

[BUS5] Kaplan, Robert S.; Norton, David P. - "The Balanced Scorecard" - Harvard Business School Press (Boston), 1992

[TEC1] Información extraída de la página http://msdn.microsoft.com/es-es/library/ms143506.aspx

[TEC2] Información extraída de la página http://msdn.microsoft.com/en-us/library/cc645993.aspx

[TEC3] Información extraída de la página

http://office.microsoft.com/es-es/sharepointserver/HA101656533082.aspx

[TEC4] Información extraída de la página

http://download.microsoft.com/download/4/a/9/4a90a2ab-7555-4adf-bf27-085c29b9f8db/MOSS_2007_Guia_Evaluacion_RTM.PDF

[TEC5] Tisseghem, Patrick – "SharePoint Server 2007" – Anaya Multimedia, 2007

[TEC6] Información extraída de la página http://es.wikipedia.org/wiki/Microsoft_Excel

[TEC7] Información extraída de la página

http://www.vb-mundo.com/Microsoft-Performance-Manager.asp

[TEC8] Andersen, E.; Aziza, B; Fitts, J; Hoberecht, S.; Kashani, T. – "Microsoft Office Performance Point Server 2007" – Wiley Publishing, 2008

[PPO1] Información extraída de la página

http://office.microsoft.com/en-us/help/HA102411381033.aspx

Anexo 1: Lanzamiento de Indicadores Desarrollados con Integration Services con una Utilidad de Escritorio o un Servicio Windows.

1.- Motivación

En un principio, en los primeros proyectos de desarrollo de indicadores los paquetes de Integration Services fueron contemplados como una alternativa a la hora de realizar las operaciones ETL de algunos clientes, dejándose la ejecución de los mismos como un tarea independiente que se realizaría mediante la creación de jobs de Microsoft SQL Server.

El problema de este modelo de ejecución surgió cuando se llego a un número tan elevado de paquetes a ejecutar que dicha creación de jobs daba fallos (ya que había jobs que se intentaban crear pero que todavía no habían sido finalizados), produciéndose un desbarajuste que impedía visualizar correctamente los datos cargados en los cubos. A dicho problema se le denominó problema de solapamiento, ya que intenta realizar una tarea que todavía no ha finalizado en su ciclo anterior.

Debido al problema de solapamiento se plantearon distintas alternativas. La primera de ellas fue realizar el lanzamiento de los paquetes con una frecuencia más elevada, opción inviable debido a que se tendrían datos demasiado obsoletos, además de la posibilidad de que, en un futuro, el problema se podría reproducir (en el momento que hubiese el doble de paquetes con la misma carga computacional, esto volvería a ocurrir).

La segunda alternativa consistía en crear una aplicación que, mediante código, ejecutase los paquetes de Integration Services cíclicamente al objeto de que no se produjese dicho solapamiento. Más adelante se decidió mudar dicha aplicación a un servicio Windows, el cual estaría en segundo plano ejecutándose constantemente para conseguir un resultado óptimo.

Como se puede deducir, la opción elegida fue esta última, y en el presente anexo se va a desarrollar su funcionamiento interno, su evolución y las funciones más importantes que hacen más fácil esta tarea de ejecución de paquetes de Integration Services.

2.- Funcionamiento Interno

En este apartado se va a enunciar cuales son los pasos que realiza la aplicación de una forma resumida, debiendo acudir al código de la aplicación para obtener una descripción más detallada.

En primera instancia, mediante un configurador o manualmente, se obtendrá el cliente (o almacén) para el cual se quieren lanzar los paquetes, así como el servidor donde están contenidas las bases de datos de dicho cliente.

Una vez hecho esto, y en el caso de seleccionados todos los paquetes y todas las residencias en la aplicación Windows (indistintamente en el servicio), se procede al lanzamiento de los paquetes, de la manera que se ilustró en el punto de la memoria, desgranando dicho esquema en el de la Figura 137, mucho más detallado.

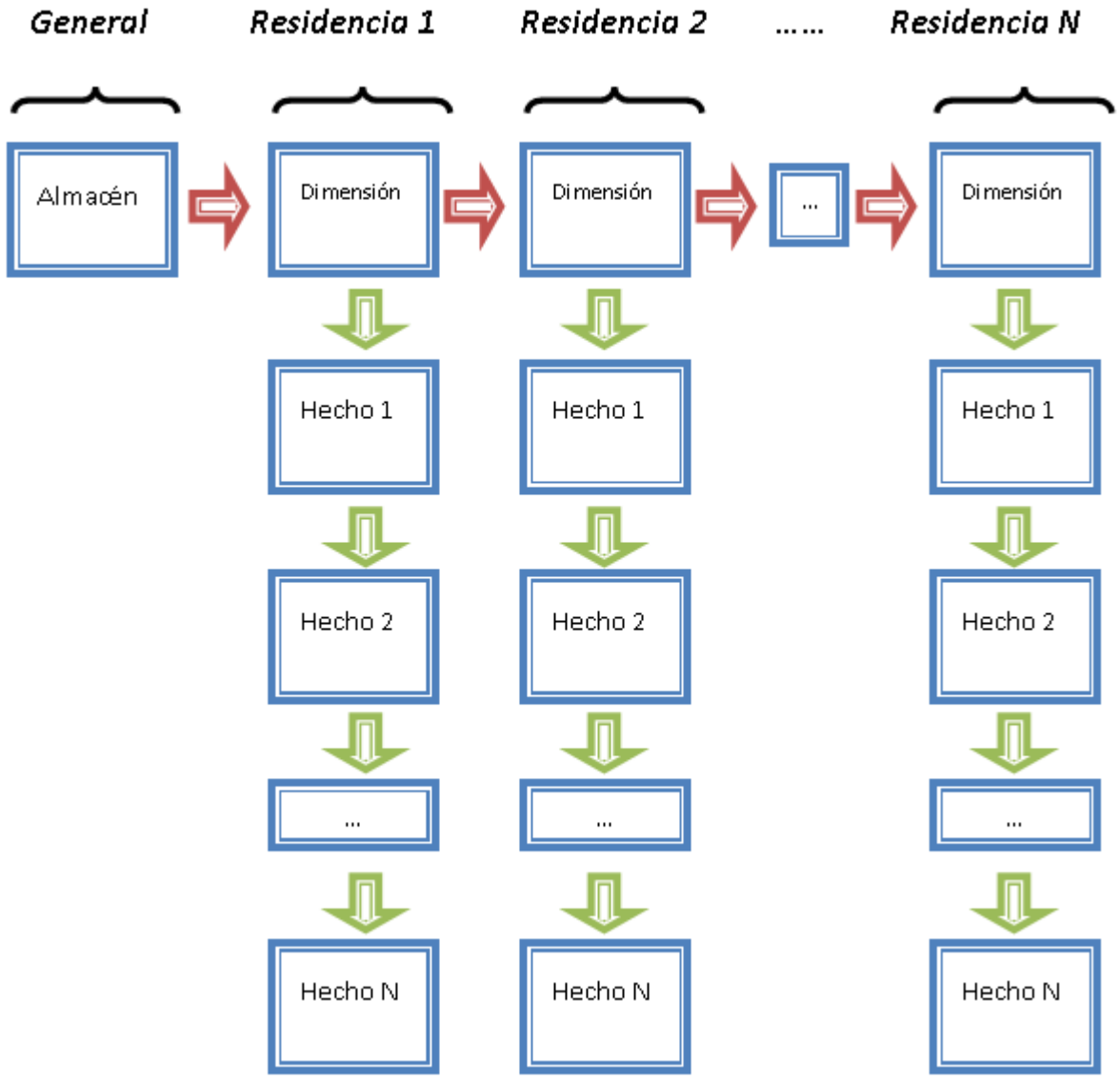


Figura 137.- Detalle de la ejecución de paquetes de Integration Services

En primer lugar se lanzará el paquete de almacén, el cual no necesita de datos de ninguna residencia para su ejecución, ya que tan solo crea la estructura del mismo.

En segundo lugar, lanzará el paquete de dimensión contra la primera residencia, y una vez realizado esto, todos y cada uno de los paquetes de hechos contra esta primera residencia.

Una vez lanzados todos los paquetes de hechos contra la residencia, se repite el mismo proceso (mismos paquetes de dimensión y hechos) contra la siguiente residencia que se halle localizada en el servidor.

De esta forma, se repetirá el proceso anterior contra todas y cada una de las residencias que haya en el servidor.

Una vez obtenidos todos los datos de todas las residencias, tan solo queda restaurar este nuevo almacén construido encima del almacén que ataca a los cubos. Para ello, se crea una copia de seguridad y se restaura encima de la base de datos correspondiente a la atacada por los cubos.

Con los datos ya correctos en el almacén adecuado, tan solo queda procesar los cubos para que se actualicen los mismos y los datos que vea el cliente en los dashboards o en las hojas de Excel sean los nuevos.

Una vez hecho esto, solo queda que esperar a que pase el tiempo para realizar una nueva ejecución, que empezará, al menos, un día después de la ejecución del paquete almacén (este tiempo podrá ser mas en caso de que la ejecución completa tarde demasiado tiempo, debido al gran número de paquetes o por cualquier otra circunstancia)

3.-Desarrollo

3.1.- Aplicación de Windows

En un principio, y como se ha comentado en la motivación, se planteo implementar una aplicación para el lanzamiento de los paquetes. Dicha aplicación fue completamente desarrollada, con la idea que, en un futuro, fuese de utilidad a la hora de realizar pruebas de nuevos indicadores.

Esta aplicación consta de la interfaz gráfica mostrada en la Figura 138.

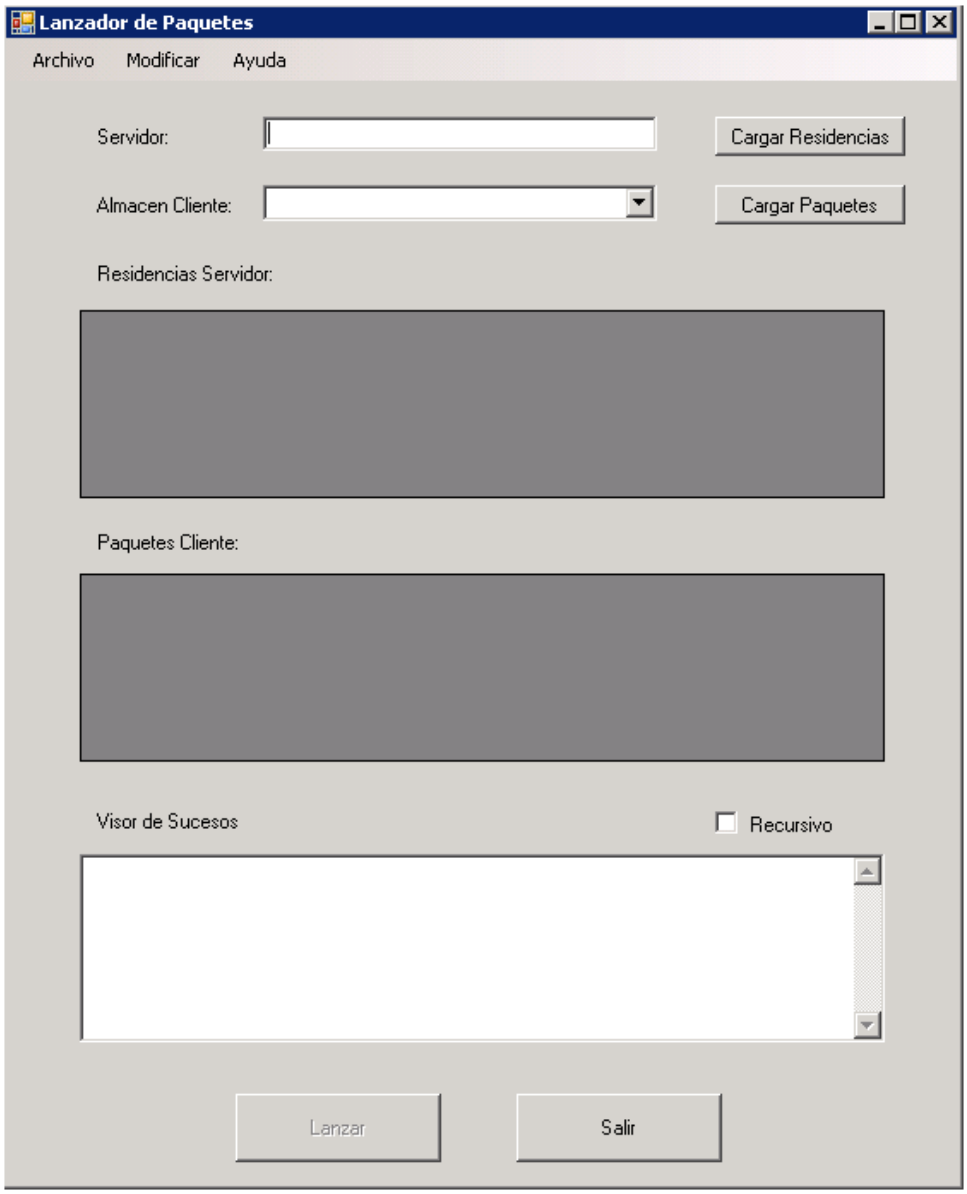


Figura 138.- Interfaz de la aplicación lanzadora de paquetes

En primer lugar, se debe introducir el servidor donde se encuentra la base de datos de DatosUsuarios de Resiplus, la cual especificará el nombre y la ruta de todas y cada una de las bases de datos del cliente.

Una vez hecho esto, se debe seleccionar el nombre del almacén del cliente al que se quiere atacar, con lo cual se cargaran todos los paquetes que tenga dicho cliente (dicha información está guardada en una base de datos que se definirá en siguientes pasos). El resultado obtenido será algo parecido al que se tiene en la Figura 139.

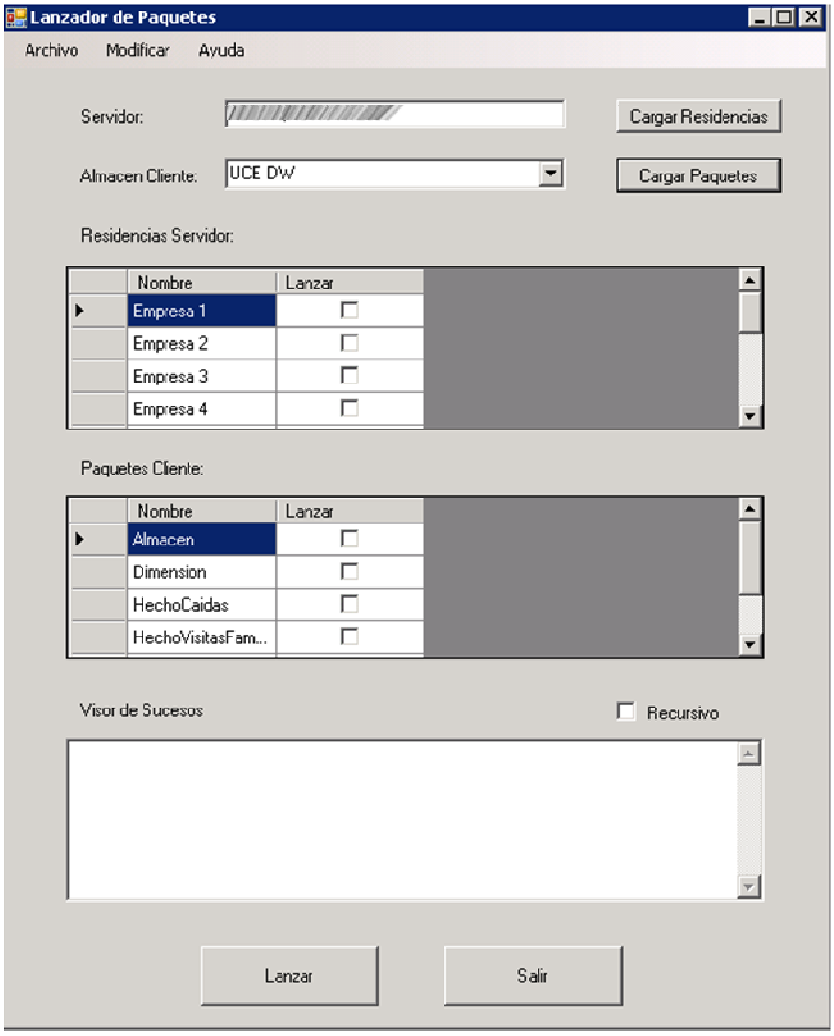


Figura 139.- Interfaz gráfica una vez cargados los centros y los paquetes

Con esto, se rellenarán los dos contenedores con las distintas residencias contenidas en el servidor y los paquetes que tiene cada cliente. Visto esto, se debe seleccionar para qué empresas se van a lanzar los paquetes seleccionados. En el caso que no se seleccione ningún paquete de almacén o dimensión aparecerá un mensaje de advertencia.

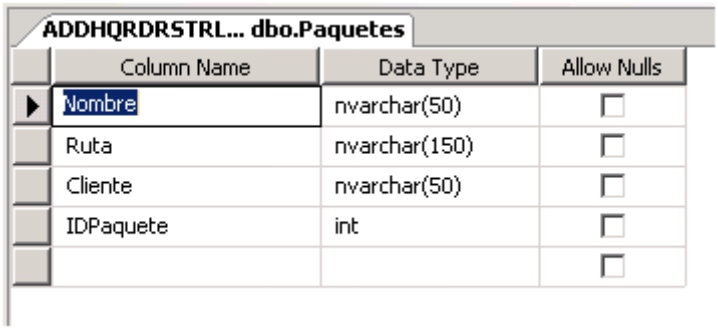
Una vez seleccionadas las residencias y los paquetes que se desean lanzar, se puede seleccionar la opción “recursivo” para que la ejecución se realice recursivamente.

Si se hace clic en lanzar, comenzará la ejecución, la cual irá escribiendo un log tanto en el visor de sucesos de la aplicación como en un archivo de texto plano, el cual registrará tanto los fallos como los aciertos del lanzamiento.

Cabe destacar que esta aplicación, además de ofrecer la misma funcionalidad que el servicio, puede resultar muy útil para realizar pruebas (ya que se puede aislar la residencia y el paquete que da el error) sin la necesidad de acudir a Resiplus BE, realizar la exportación…

3.2.- Base de Datos

Para automatizar el lanzamiento de los paquetes, se tiene que crear una base de datos de nombre "Clientes", con una única tabla llamada "Paquetes", que debe tener la estructura mostrada en la Figura 140.



ADDHQRDRSTRL... dbo.Paquetes

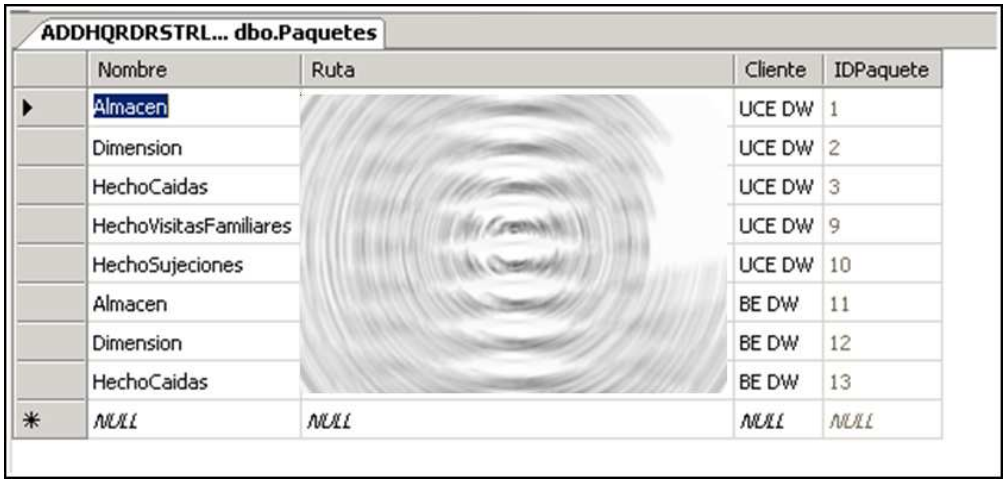
Column Name	Data Type	Allow Nulls
Nombre	nvarchar(50)	☐
Ruta	nvarchar(150)	☐
Cliente	nvarchar(50)	☐
IDPaquete	int	☐
		☐

Figura 140.- Estructura de la base de datos para la ejecución de los paquetes

El campo de "IDPaquete" debe ser el identificador, además de ser autonumérico. En cada uno de los campos se debe rellenar lo siguiente:

- Nombre: nombre del paquete; si es de almacén se llamará "Almacen", si es de dimensión se llamará "Dimension", y si es de hechos se llamara Hecho seguida del nombre de hecho.
- Ruta: aquí se pondrá la ruta donde está contenida el paquete (el nombre del fichero será el nombre puesto en la columna anterior).
- Cliente: el cliente al que pertenece el paquete.

Un ejemplo de base de datos rellenada sería la mostrada en la Figura 141.



ADDHQRDRSTRL... dbo.Paquetes

Nombre	Ruta	Cliente	IDPaquete
Almacen		UCE DW	1
Dimension		UCE DW	2
HechoCaidas		UCE DW	3
HechoVisitasFamiliares		UCE DW	9
HechoSujeciones		UCE DW	10
Almacen		BE DW	11
Dimension		BE DW	12
HechoCaidas		BE DW	13
NULL	NULL	NULL	NULL

Figura 141.- Base de datos para la ejecución de los paquetes

La aplicación atacará a esta base de datos al objeto de adquirir los nombres de los paquetes de cada cliente además de la ruta en el cual está contenido.

3.3.- Servicio de Windows

El servicio Windows es la evolución de la aplicación de Windows creada anteriormente. El servicio realizará los mismos pasos que realiza dicha aplicación, teniendo en cuenta que se necesita un archivo de configuración llamado "ConfLanz.txt" y que estará contenido en el directorio donde está el ejecutable del servicio.

Este archivo tendrá los siguientes parámetros separados por la cadena "---":

- Parámetro 1: Almacén objetivo.
- Parámetro 2: Servidor.
- Parámetro 3: Ruta del fichero de texto plano de log.
- Parámetro 4: Ruta de los ficheros de bases de datos.
- Resto de parámetros: Cubos a procesar.

Al contrario de lo que sucedía en la aplicación Windows, este servicio no ofrece ningún tipo de configuración a la hora de ejecutar solo determinados paquetes para unas determinadas residencias.

Por defecto, el servicio ejecutará todos los paquetes que están introducidos dentro de la base de datos para todas las residencias que hay en el servidor actual.

4.- Guía del Código del Servicio

En este apartado se van a explicar cuáles son las funciones más importantes dentro del código creado para implementar la ejecución de paquetes de Integration Services.

4.1.- Funciones

comenzar()

En esta función se inicializan los siguientes parámetros leídos de un fichero de texto plano:

- "almacenobjetivo": almacén de datos al que va a atacar nuestra aplicación.
- "servidorobjetivo": servidor en el que está incluido el almacén especificado anteriormente.
- "rutalog": ruta de datos en la cual se crearán los distintos archivos de log.
- "rutaBBDD": ruta donde están contenidas los ficheros de la base de datos, tanto las copias que se realizan como los ficheros de las bases de datos atacadas.
- "procesar()": vector de cadenas de texto donde estarán contenidas las distintas bases de datos de Analysis Services que se van a procesar.

Una vez inicializados los parámetros, se procede a ejecutar las siguientes funciones:

- CargaResidendias()

- CargaPaquetes()
- OrdenarPaquetes()

Con la lista de paquetes y residencias objetivo ya elaborada, se procederá al lanzamiento iterativo de las siguientes funciones:

- LanzarPaquetes()
- MatarConexiones()
- sobreescribirAlmacen()
- CreaJobProcesarCubos(String ())
- espera()

CargaResidencias()

Esta función realiza una consulta SQL adquiriendo el nombre y la ruta de todas las bases de datos contenidas en “DatosUsuarios”.

CargaPaquetes()

Realiza una consulta SQL adquiriendo el nombre y la ruta de los paquetes a lanzar, en este caso, los especificados en la base de datos “Clientes” creada manualmente. Todo paquete que tenga como almacén objetivo el mismo que el adquirido en la función “comenzar()” será lanzado en pasos posteriores.

OrdenarPaquetes()

Como no es preciso que los registros de la tabla de paquetes estén ordenados conforme a su orden de lanzamiento, se ha de realizar una función de ordenación para lograr este orden.

LanzarPaquetes()

Dependiendo del tipo de paquete a lanzar, se llamará a una función o a otra, debido a que cada paquete tiene unas conexiones distintas. Las posibilidades son las siguientes:

- lanzaPaqueteAlmacen()
- lanzaPaqueteDimension()
- lanzaPaqueteHecho2Con()
- lanzaPaqueteHecho3Con()

lanzaPaqueteAlmacen()

Ejecutará el paquete de “Almacén” del cliente especificado. Las conexiones necesarias para realizar este lanzamiento son “Almacén” (que es construida a partir del “almacenobjetivo” y “servidorobjetivo”) y “Conexión” (una conexión que lo único que hace es conectarse al servidor, y que se crea a partir del parámetro “servidorobjetivo”)

lanzaPaqueteDimension()

Ejecutará el paquete de “Dimension” contra la residencia correspondiente. Las conexiones necesarias para realizar el lanzamiento son “Almacén”, “Residencia” (conexión a la residencia contra la que se está lanzando el paquete) y “DatosUsuarios” (conexión necesaria para rellenar la tabla de identificadores de las residencias).

lanzaPaqueteHecho2Con()

Ejecutará los paquetes de Hechos que tengan tan solo 2 conexiones: la conexión “Almacén” y la conexión “Residencia”.

lanzaPaqueteHecho3Con()

Ejecutará los paquetes de Hechos que dispongan de 3 conexiones: “Almacén”, “Residencia” y “Residencia2” (Segunda base de datos de la residencia).

MatarConexiones()

Esta función realiza una consulta SQL que matará las conexiones al almacén de destino de la restauración en el que se hará de la base de datos. Esto sucede porque es posible que la conexión entre el cubo de Analysis Services y el almacén de datos este activo y no deje realizar la restauración de los nuevos datos.

sobreescribirAlmacen()

Esta función realizará la copia de seguridad del nuevo almacén elaborado con la ejecución de los paquetes y la restaurará encima de los datos antiguos del almacén.

CreaJobProcesarCubos(String())

Esta función crea un Job de SQL en el cual se procesaran todos los cubos de Analysis Services especificados en el vector que se le pasa a la misma.

espera()

Esta función dejará en espera al servicio hasta el día siguiente para realizar un nuevo lanzamiento.