UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Proceso de desarrollo de indicadores de gestión. Un caso de estudio en el contexto de un ERP.

Autor: Juan Vicente Pérez Palanca

Director: Dr. Patricio Letelier Torres

Septiembre del 2010

Tesis presentada para cumplir con los requisitos finales para la obtención del título de Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información, de la Universidad Politécnica de Valencia, 2010.

Agradecimientos

Estos tres últimos años han sido para mi, los más importantes e intensos de mi trayectoría profesional. En este tiempo he tenido la enorme suerte de conocer y trabajar con personas que me han ayudado (de una forma u otra) en la consecución de la presente tesis de máster, y motivado a seguir adelante con mis objetivos. A todas ellas mil gracias, pero en especial quisiera hacer mención de agradecimiento a las siguientes personas.

En primer lugar quisiera mencionar a mi director de proyecto Patricio Letelier, al cual admiro como persona y como profesional. Me gustaría agradecerle la gran oportunidad que me ha dado, la confianza que ha depositado en mí y sobretodo el tiempo y el esfuerzo que me ha dedicado.

A mi compañera de trabajo, Marta, en estos malos momentos, le quiero citar la siguiente frase: *"Nunca una noche ha vencido al amanecer, y nunca un problema ha vencido a la esperanza"*.

También me gustaría agradecer a los socios de ADD Informática el haber confiado en mí para llevar a cabo el desarrollo de este trabajo.

A mis padres les agradezco que me hayan ayudado, apoyado y motivado en los momentos más duros. Sobretodo quiero acordarme de mi abuelo, gracias por haberme cuidado desde muy pequeño y haber sido como un hermano para mí.

Y en último lugar y más importante, quisiera especialmente darle las gracias a mi pareja, complice y amiga, Mabel. Muchas gracias por toda tu comprensión, por entenderme y animarme a realizar esta tesis, por estar ahí siempre, gracias.

Índice

Capítulo 1. Introducción

1.1. Motivación

Gestionar la información en las empresas es, hoy en día, un factor clave para poder sobrevivir en un mercado cambiante, dinámico y global. Aprender a competir con esta información es fundamental para la toma de decisiones, el crecimiento y la gestión de nuestra empresa. La disciplina denominada Inteligencia de Negocios (*Business Intelligence*) nos acerca a los sistemas de información que nos ayudan a la toma de decisiones en nuestra organización. Toda empresa dispone, no importa su tamaño, de sistemas de información más o menos sofisticados que son convenientes de analizar y optimizar mediante soluciones Business Intelligence.

Sin embargo, el desarrollo de una solución de Business Intelligence conlleva todas las dificultades de un proyecto de ingeniería; complejidad de la implementación, requisitos volátiles, desafíos tecnológicos, desarrollo colaborativo, dificultad para asegurar la calidad, etc. Además, el mercado cada vez exige plazos de entrega más reducidos y presupuestos más ajustados. Por todo esto, parece obvio que es necesario y fundamental el uso de metodologías y estándares que se centren en la mejora de proceso de este tipo de soluciones.

TUNE-UP es una metodología que encaja perfectamente en aquellas soluciones de Business Intelligence que ofrecen un valor de negocio al cliente mediante entregas cortas y continuas, y que por tanto, requieren un desarrollo iterativo e incremental.

El presente trabajo se ha realizado en el contexto de un convenio universidad-empresa durante 3 años. La PYME es una empresa de desarrollo de software que tiene un ERP perteneciente al sector socio-sanitario y cuenta con más de 700 clientes. El autor de este proyecto ha participado en el desarrollo y mejora de ResiPlus BI (*ResiPlus Business Intelligence*), suite de Inteligencia de Negocios ofertada por esta empresa.

1.2. Objetivo del proyecto

El objetivo de este proyecto es proponer un marco tecnológico para una solución de Business Intelligence, contextualizado en el servicio de Inteligencia de Negocios ResiPlus BI, que permita ilustrar cómo se aborda el desarrollo de indicadores de gestión.

Concretamente, los objetivos específicos son:

- Validar que el marco tecnológico propuesto para el desarrollo de indicadores es el adecuado.
- Validar la utilidad de la metodología TUNE-UP y sus workflows en el desarrollo de indicadores.
- Desarrollar diferentes tipos de indicadores con sus correspondientes informes para ver las diferencias entre ellos a la hora de la implementación.
- Desarrollar un dashboard que muestre información relativa a cada uno de los indicadores desarrollados previamente.
- Desarrollar pruebas automatizadas que permitan validar el correcto funcionamiento de los indicadores.

1.3. Organización de la memoria

A continuación se presenta la estructura del proyecto:

El capítulo 1 incluye una introducción a la motivación de nuestro trabajo y el objetivo que persigue este proyecto en el área del Business Intelligence.

En el capítulo 2 se ha realizado una introducción al concepto de Business Intelligence, viendo cada uno de los componentes que lo forman, áreas funcionales, beneficios y los actuales protagonistas.

En el capítulo 3 se analizan todos los aspectos de un Data Warehouse y se habla de las herramientas OLAP utilizadas para su explotación.

El capítulo 4 explica los aspectos básicos de la metodología TUNE-UP e ilustra un workflow específico para soluciones de Business Intelligence.

En el capítulo 5 se propone un marco tecnológico para una solución Business Intelligence, describiendo cada uno de los componentes que lo forman.

El capítulo 6 ilustra cómo desarrollar indicadores en el marco tecnológico propuesto en el capítulo anterior, contextualizado en el caso de estudio ResiPlus BI.

Finalmente, el capítulo 7 concluye el proyecto con un resumen del trabajo realizado.

Adicionalmente, hay una serie de anexos que incluyen ejemplos de documentos utilizados durante el desarrollo de indicadores y unos tutoriales que permiten introducirnos en el uso de las tecnologias propuestas.

Capítulo 2. Business Intelligence

"No es posible gestionar lo que no se puede medir", decía William Hewlet. En este sentido, y en los tiempos que nos ocupan, hay una necesidad aún mayor de conocer el alcance de los principales indicadores del negocio. Para una correcta gestión es necesario tener la posibilidad de analizar rápidamente los puntos críticos, establecer tendencias, así como poder determinar relaciones causa efecto ante el cambio en determinados parámetros externos o internos. Así, el establecimiento de los denominados sistemas de Business Intelligence y el uso del potencial actual de las herramientas TIC[1] permiten un tratamiento cada vez más rápido, complejo e inmediato de los datos, de la información y, en definitiva, del conocimiento.

Los antiguos sistemas de información para la Dirección, que convertían datos operacionales en indicadores de gestión (la mayor parte de las veces de naturaleza económico-financiera), se han visto absorbidos y superados por un nuevo concepto del tratamiento de la información para la toma decisiones: Business Intelligence.

En este capítulo se realiza una introducción al concepto de Business Intelligence, viendo cada uno de los componentes que lo forman, áreas funcionales, beneficios y los actuales protagonistas en el mundo del Business Intelligence.

El material presentado en este capítulo ha sido extraído de [5] y [12].

2.1. ¿Qué es Business Intelligence?

Business Intelligence (BI) es la habilidad para transformar los datos en información, y la información en conocimiento, de forma que se pueda optimizar el proceso de toma de decisiones en los negocios (Figura 2.1.).

Figura 2.1. Datos, Información y Conocimiento

Desde un punto de vista más pragmático, y asociándolo directamente con las tecnologías de la información, podemos definir Business Intelligence como el conjunto de metodologías, aplicaciones y tecnologías que permiten reunir, depurar y transformar datos de los sistemas transaccionales e información desestructurada (interna y externa a la compañía) en información estructurada, para su explotación directa (reporting, análisis OLAP[2],etc.) o para su análisis y conversión en conocimiento, dando así soporte a la toma de decisiones sobre el negocio.

[1] Las tecnologías de la información y la comunicación (TIC o NTIC para *Nuevas Tecnologías de la Información y de la Comunicación* o IT para «Information Technology») agrupan los elementos y las técnicas utilizadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, Internet y telecomunicaciones.

[2] OLAP es el acrónimo en inglés de procesamiento analítico en línea (*On-Line Analytical Processing*).

La inteligencia de negocio actúa como un factor estratégico para una empresa u organización, generando una potencial ventaja competitiva, que no es otra que proporcionar información privilegiada para responder a los problemas de negocio: entrada a nuevos mercados, promociones u ofertas de productos, eliminación de islas de información, control financiero, optimización de costes, planificación de la producción, análisis de perfiles de clientes, rentabilidad de un producto concreto, etc.

2.2. Componentes del Business Intelligence

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

2.2.1. Esquema de una solución de Business Intelligence

Una solución de Business Intelligence (Figura 2.2.) parte de los sistemas de origen de una organización (bases de datos, ERPs3, 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 de datos. Esta etapa suele apoyarse en un almacén intermedio, llamado ODS[4], que actúa como pasarela entre los sistemas fuente y los sistemas destino (generalmente un Data Warehouse), 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 Data Warehouse corporativo, que puede servir como base para la construcción de distintos Datamarts departamentales.

Un Datamart es una base de datos especializada, departamental, orientada a satisfacer las necesidades específicas de un grupo particular de usuarios. En otras palabras, un Datamart es un subconjunto del Data Warehouse corporativo con transformaciones específicas para el área a la que va dirigido. 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 (OLTP[5]) o mediante bases de datos analíticas (OLAP).

[3] Los sistemas de planificación de recursos empresariales, o ERP (por sus siglas en inglés, *Enterprise resource planning*) son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

[4] ODS es el acrónimo en inglés de almacén operacional de datos (*Operational Data Store*).

[5] OLTP es el acrónimo en inglés de procesamiento de transacciones en línea (*OnLine Transaction Processing*).

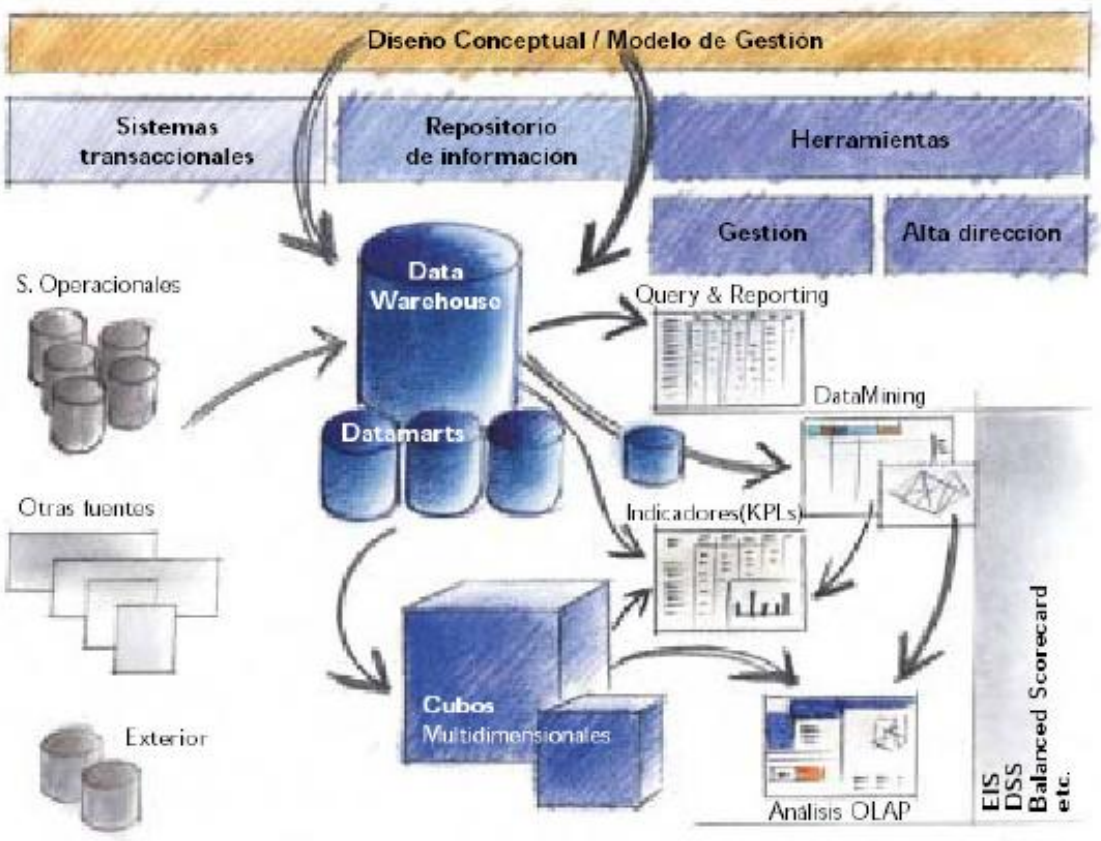


Figura 2.2. Esquema de una solución BI

Los datos albergados en el Data Warehouse 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 (DSS), los sistemas de información ejecutiva (EIS) y los cuadros de mando (CMI) o Balanced Scorecard (BSC).

2.2.2. Diseño conceptual

Para resolver el diseño de un modelo BI se deben contestar a tres preguntas básicas: cuál es la información requerida para gestionar y tomar decisiones; cuál debe ser el formato y composición de los datos a utilizar; de dónde proceden esos datos y cuál es la disponibilidad y periodicidad requerida. En otras palabras, el diseño conceptual tiene diferentes momentos en el desarrollo de una plataforma BI: en la fase de construcción del Data Warehouse y los Datamarts primarán los aspectos de estructuración de la información según potenciales criterios de explotación. En la fase de implantación de herramientas de soporte a la alta dirección se desarrolla el análisis de criterios directivos: misión, objetivos estratégicos, factores de seguimiento, indicadores clave de rendimiento o KPIs[6], modelos de gestión , en definitiva, información para el qué, cómo, cuándo, dónde y para qué de sus necesidades de información. Estos momentos no son, necesariamente, correlativos, sino que cada una de las etapas del diseño condiciona y es condicionada por el resto.

[6] KPI, del inglés *Key Performance Indicators*, o Indicadores Clave de Rendimiento, miden el nivel del desempeño de un proceso, enfocándose en el "cómo" e indicando qué tan buenos son los procesos, de forma que se pueda alcanzar el objetivo fijado.

2.2.3. Construcción y alimentación del Data Warehouse y/o de los Datamarts

Un Data Warehouse es una base de datos corporativa que replica los datos transaccionales una vez seleccionados, depurados y especialmente estructurados para actividades de query y reporting.

La vocación del Data Warehouse es aislar los sistemas operacionales de las necesidades de información para la gestión, de forma que cambios en aquéllos no afecten a éstas y viceversa (únicamente cambiarán los mecanismos de alimentación, no la estructura, contenidos, etc.).

No diseñar y estructurar convenientemente y desde un punto de vista corporativo el Data Warehouse y los Datamarts generará problemas que pueden condenar al fracaso cualquier esfuerzo posterior: información para la gestión obtenida directamente a los sistemas operacionales, florecimiento de Datamarts descoordinados en diferentes departamentos, etc.

En definitiva, según la estructuración y organización de cada compañía, pueden originarse situaciones no deseadas y caracterizadas generalmente por la ineficiencia y la falta de calidad en la información resultante.

2.2.4. Herramientas de explotación de la información

Es el área donde más avances se han producido en los últimos años. Sin embargo, la proliferación de soluciones mágicas y su aplicación coyuntural para solucionar aspectos puntuales ha llevado, en ocasiones, a una situación de desánimo en la organización respecto a los beneficios de una solución BI. Sin entrar a detallar las múltiples soluciones que ofrece el mercado, a continuación se identifican los modelos de funcionalidad o herramientas básicas (cada producto de mercado integra, combina, potencia, adapta y personaliza dichas funciones):

- **Query & Reporting**. Herramientas para la elaboración de informes y listados, tanto en detalle como sobre información agregada, a partir de la información de los Data Warehouses y Datamarts. Desarrollo a medida y/o herramientas para una explotación libre.
- **Cuadro de mando analítico** (EIS tradicionales). Elaboración, a partir de Datamarts, de informes resumen e indicadores clave para la gestión (KPI) que permitan a los gestores de la empresa analizar los resultados de la misma de forma rápida y eficaz. En la práctica es una herramienta de query orientada a la obtención y presentación de indicadores para la dirección (frente a la obtención de informes y listados).
- **Cuadro de mando integral o estratégico** (*Balanced Scorecard*). Este modelo parte de que la estrategia de la empresa es el punto de referencia para todo proceso de gestión interno. Con él, los diferentes niveles de dirección y gestión de la organización disponen de una visión de la estrategia de la empresa traducida en un conjunto de objetivos, iniciativas de actuación e indicadores de evolución. Los objetivos estratégicos se asocian mediante relaciones causa-efecto y se organizan en cuatro áreas o perspectivas: financiera, cliente, procesos y formación, y desarrollo. El cuadro de mando integral es una herramienta que permite alinear los objetivos de las diferentes áreas o unidades con la estrategia de la empresa y seguir su evolución.

- **OLAP** (*on-line analytical processing*). Herramientas que manejan cuestiones complejas de bases de datos relacionales, proporcionando un acceso multidimensional a los datos, capacidades intensivas de cálculo y técnicas de indexación especializadas. Permiten a los usuarios trocear sus datos, planteando *queries* sobre diferentes atributos o ejes. Utilizan un servidor intermedio para almacenar los datos multidimensionales precalculados, de forma que la explotación sea rápida.
- **Minería de Datos** (*Data Mining*). Son auténticas herramientas de extracción de conocimiento útil, a partir de la información contenida en las bases de datos de cualquier empresa. El objetivo que se persigue es descubrir patrones ocultos, tendencias y correlaciones, y presentar esta información de forma sencilla y accesible a los usuarios finales para solucionar, prever y simular problemas del negocio. El Data Mining incorpora la utilización de tecnologías basadas en redes neuronales, árboles de decisión, reglas de inducción, análisis de series temporales y visualización de datos.

2.3. Selección de herramientas de Business Intelligence

La selección de una u otra herramienta estará en función de múltiples aspectos a considerar:

- **Qué información se necesita**. Es importante no complicarse, sobre todo al principio, con indicadores y modelos complejos: indicadores selectivos, sencillos, admitidos por todos los usuarios, etc. son una buena fórmula en las primeras etapas del BI.
- **Para qué se quiere la información**. Bajo el concepto general "*soporte a la toma de decisiones*" se esconden múltiples necesidades particulares: contrastar que todo va bien, analizar diferentes aspectos de la evolución de la empresa, presentar información de forma más intuitiva, comparar información en diferentes periodos de tiempo, comparar resultados con previsiones, identificar comportamientos y evoluciones excepcionales, confirmar o descubrir tendencias e interrelaciones, necesidad de realizar análisis predictivos, etc., son todas ellas necesidades parciales dentro del concepto general.
- **A quién va dirigida**. Organización en general, gestión, dirección, dirección estratégica, etc.
- **Aspectos meramente técnicos** (tiempos de respuesta, integración, seguridad, etc.) y **funcionales** (navegación, entorno gráfico, etc.).

2.4. Áreas funcionales y beneficios del Business Intelligence

Originariamente, los sistemas de información a la dirección aportaban básicamente información económico-financiera. Con la extensión de las herramientas de Business Intelligence, este concepto abarca ahora todas las áreas funcionales de la empresa: recursos humanos, logística, calidad, comercial, marketing, etc.

En la actualidad, estas visiones funcionales han sido superadas por el concepto de CPM (Corporate Performance Management), que aporta información integral de la empresa en todas sus áreas y a través de todos sus ciclos de gestión: planificación, operación y análisis de resultados.

Entre los obstáculos tradicionales a la implantación BI se encuentra la dificultad para calcular su ROI (Return On Investment). La mayor parte de los beneficios producidos son intangibles, derivados de la mejora de la gestión de la compañía.

En términos económicos, se evidencia una reducción de costes por incremento de la eficiencia de la infraestructura TIC y un incremento de la productividad de los empleados, directamente derivado de la disponibilidad de información; estas magnitudes son difícilmente cuantificables, aunque diferentes fuentes las sitúan en torno al 5% y 10-15%, respectivamente.

Cualitativamente, los beneficios se derivan, obviamente, del incremento de la eficiencia en el proceso de toma de decisiones: mayor información, de mejor calidad, más fiable, compartida por toda la organización, menores tiempos de respuesta en su obtención, mejora de la comunicación en la empresa y creación de un lenguaje homogéneo.

En el debe de las implantaciones BI hay que destacar la dificultad de integración con el resto de sistemas de la compañía y, sobre todo, la dificultad para conjugar las expectativas de los usuarios con las soluciones implementadas, por lo que los aspectos de definición conceptual y selección de plataforma, junto a la gestión del cambio en la implantación de los proyectos, adquieren una importancia relevante. La experiencia dice que los factores puramente organizativos originan más de la mitad de los fracasos de proyectos BI.

Por todo ello, para el éxito de una estrategia BI podemos identificar los siguientes factores críticos:

- Importancia del diseño.
- Importancia de seleccionar y disponer de una plataforma tecnológica y de herramientas adecuadas.
- Alineación de los objetivos del departamento de sistemas de información y los usuarios.
- Importancia de consensuar con los usuarios.
- Importancia de contar con apoyo e impulso desde la dirección general.
- Importancia de contar con personal cualificado, tanto en las fases de diseño como de implantación.

2.5. Protagonistas del Business Intelligence

A continuación se muestra un informe publicado por Gartner Inc.[7][2] sobre las mejores plataformas de Business Intelligence (Figura 2.3.). El estudio está organizado en tres categorías de funcionalidad: integración, suministro de información y análisis.

[7] El *Cuadrante Mágico de Gartner* es una representación gráfica del mercado durante un periodo específico de tiempo. Muestra el análisis desarrollado por Gartner Inc., sobre cómo se comportan ciertos actores del mercado en función de diferentes criterios de evaluación. El informe de Gartner no apoya a ninguno de los productos o proveedores mencionados ni tampoco aconseja el uso o la selección de los proveedores colocados en el cuadrante de líderes.

Figura 2.3. Plataformas de Business Intelligence

Los líderes son definidos como aquellos proveedores que pueden ofrecer grandes implementaciones empresariales y soportar una amplia estrategia de BI. Según el gráfico, estos líderes son **SAP**, **Oracle**, **Microsoft**, **IBM**, **SAS**, **Information Builders** y **Microstrategy**. Además, se observa que no aparecen compañías habituales de dicho informe como Hyperion, Cognos o Business Objects, ya que todas ellas fueron adquiridas en los últimos años por parte de las compañias líderes. Concretamente, Hyperion fue adquirida por Oracle en 2007, Cognos fue adquirida por IBM en 2008 y finalmente Business Objects fue adquirida por SAP en 2008.

El resto de competidores son soluciones de nicho, que ofrecen soluciones de visualización avanzada (típicamente para cuadros de mando y análisis OLAP). Estos "pequeños" proveedores tienen hueco debido a la complejidad, limitaciones y costes que tienen los grandes proveedores en este segmento del mercado BI.

Capítulo 3. Data Warehouse

En el esquema de una solución de Business Intelligence, un Data Warehouse viene determinado por su situación central como fuente de información para las herramientas de análisis.

En el actual capítulo se explicará con más detalle los conceptos de Data Warehousing y Data Warehouse, cómo se diseña éste último y en qué consiste su proceso de extracción, transformación y carga llamado ETL[8].

El material presentado en este capítulo ha sido extraído de las siguientes fuentes:[1], [3], [9] y [13].

3.1. Data Warehousing y Data Warehouse

Para llevar a cabo BI, es necesario gestionar datos guardados en diversos formatos, fuentes y tipos, para luego depurarlos e integrarlos, además de almacenarlos en un solo destino o base de datos que permita su posterior análisis y exploración. Por esto, es imperativo y de vital importancia contar con un proceso que satisfaga todas estas necesidades. Este proceso se denomina Data Warehousing.

Data Warehousing es el proceso encargado de extraer, transformar, consolidar, integrar y centralizar los datos que la empresa genera en todos los ámbitos de su actividad diaria de negocios (compras, ventas, producción, etc) y/o información externa relacionada. Permitiendo de esta manera el acceso y exploración de la información requerida, a través de una amplia gama de posibilidades de análisis multivariables, con el objetivo final de dar soporte al proceso de toma de desiciones estratégico y táctico.

Data Warehousing posibilita la extracción de datos de sistemas operacionales y fuentes externas, permite la integración y homogeneización de los datos de toda la empresa, provee información que ha sido transformada y sumarizada, para que ayude en el proceso de toma de decisiones estratégicas y tácticas.

Data Warehousing, convertirá entonces los datos operacionales de la empresa en una herramienta competitiva, debido a que pondrá a disposición de los usuarios indicados la información pertinente, correcta e integrada, en el momento que se necesita.

Pero para que Data Warehousing pueda cumplir con sus objetivos, es necesario que la información que se extrae, transforma y consolida, sea almacenada de manera centralizada en una base de datos con estructura multidimensional denominada Data Warehouse.

Un Data Warehouse (DW)[9] es un repositorio central o colección de datos, en el cual se encuentra integrada la información de la organización que se usa como soporte para el proceso de toma de decisiones gerenciales.

El concepto de DW comenzó a surgir cuando las organizaciones tuvieron la necesidad de usar los datos que cargaban a través de sus sistemas operacionales para planeamiento y toma de decisiones. Para cumplir estos objetivos se necesitan efectuar consultas que sumarizan los datos, y que si se hacen sobre los sistemas operacionales reducen mucho el rendimiento de las transacciones que se están haciendo al mismo tiempo. Fue entonces

[8] ETL es el acrónimo en inglés de extraer, transformar y cargar (*Extract, Transform and Load*).

[9] A partir de este momento utilizaremos la sigla DW o el término almacén de datos para hacer referencia al Data Warehouse.

cuando se decidió separar los datos usados para reportes y toma de decisiones de los sistemas operacionales y diseñar y construir DWs para almacenar estos datos.

Las principales características que posee un DW se detallan a continuación:

- **Orientado al tema**: orientado a la información relevante de la organización. En un DW la información se clasifica en base a los aspectos de interés para la empresa, es decir, se diseña para consultar eficientemente información relativa a las actividades básicas de la organización, como ventas, compras y producción, y no para soportar los procesos que se realizan en ella, como gestión de pedidos, facturación, etc.
- **Integrado**: integra datos recogidos de diferentes sistemas operacionales de la organización y/o fuentes externas. Esta integración se hace estableciendo una consistencia en las convenciones para nombrar los datos, en la definición de las claves, y en las medidas uniformes de los datos.
- **Variable en el tiempo**: los datos son relativos a un periodo de tiempo y deben ser incrementados periódicamente. La información almacenada representa fotografías correspondientes a ciertos periodos de tiempo.
- **No volátil**: la información no se modifica después de que se inserta, sólo se incrementa. El periodo cubierto por un DW varía de 2 a 10 años.

3.2. Ventajas e inconvenientes de un Data Warehouse

La utilización de un DW por parte de una organización conlleva una serie de ventajas. Algunas de las ventajas más sobresalientes son las siguientes:

- Transforma datos orientados a las aplicaciones en información orientada a la toma de decisiones.
- Integra y consolida diferentes fuentes de datos (internas y/o externas) y departamentos empresariales, que anteriormente formaban islas, en una única plataforma sólida y centralizada.
- Provee la capacidad de analizar y explotar las diferentes áreas de trabajo y de realizar un análisis inmediato de las mismas.
- Permite reaccionar rápidamente a los cambios del mercado.
- Aumenta la competitividad en el mercado.
- Elimina la producción y el procesamiento de datos que no son utilizados ni necesarios, producto de aplicaciones mal diseñadas o ya no utilizadas.
- Mejora la entrega de información, es decir, información completa, correcta, consistente, oportuna y accesible. Información que los usuarios necesitan, en el momento adecuado y en el formato apropiado.
- Logra un impacto positivo sobre los procesos de toma de decisiones. Cuando los usuarios tienen acceso a una mejor calidad de información, la empresa puede lograr por sí misma: aprovechar el enorme valor potencial de sus recursos de información y transformarlo en valor verdadero; eliminar los retardos de los procesos que resultan de información incorrecta, inconsistente y/o inexistente; integrar y optimizar procesos a través del uso compartido e integrado de las fuentes de información; permitir al usuario adquirir mayor confianza acerca de sus

propias decisiones y de las del resto, y lograr así, un mayor entendimiento de los impactos ocasionados.

- Aumento de la eficiencia de los encargados de tomar decisiones.
- Los usuarios pueden acceder directamente a la información en línea, lo que contribuye a su capacidad para operar con mayor efectividad en las tareas rutinarias o no. Además, pueden tener a su disposición una gran cantidad de valiosa información multidimensional, presentada coherentemente como fuente única, confiable y disponible en sus estaciones de trabajo. Así mismo, los usuarios tienen la facilidad de contar con herramientas que les son familiares para manipular y evaluar la información obtenida en el DW, tales como: hojas de cálculo, procesadores de texto, software de análisis de datos, software de análisis estadístico, reportes, tableros, etc.
- Permite la toma de decisiones estratégicas y tácticas.

No obstante, el uso de los almacenes de datos no está exento de desventajas, siendo las más comunes las siguientes:

- Requiere una gran inversión, debido a que su correcta construcción no es tarea sencilla y consume muchos recursos, además, su misma implementación implica desde la adquisición de herramientas de consulta y análisis, hasta la capacitación de los usuarios.
- Existe resistencia al cambio por parte de los usuarios, ya que la forma de consultar la información para elaborar informes o tomar decisiones varía.
- Los beneficios del almacén de datos son apreciados en el mediano y largo plazo. Este punto deriva del anterior, y básicamente se refiere a que no todos los usuarios confiarán en el DW en una primera instancia, pero sí lo harán una vez que comprueben su efectividad y ventajas. Además, su correcta utilización surge de la propia experiencia.
- Si se incluyen datos propios y confidenciales de clientes, proveedores, etc, el depósito de datos atentará contra la privacidad de los mismos, ya que cualquier usuario podrá tener acceso a ellos.
- Infravaloración de los recursos necesarios para la captura, carga y almacenamiento de los datos.
- Infravaloración del esfuerzo necesario para su diseño y creación.
- Incremento continuo de los requerimientos del usuario.
- Subestimación de las capacidades que puede brindar la correcta utilización del DW y de las herramientas de BI en general.

3.3. Diferencias entre Data Warehouse y base de datos operacional

La mayoría de los sistemas operacionales están dirigidos a la carga de datos, con cientos o miles de transacciones diarias y repetitivas, que además requieren un tiempo de respuesta muy corto por parte de los usuarios. Ejemplos de este tipo de operaciones lo son las reservas de vuelos, depósitos bancarios, y reservaciones de hotel.

Las bases de datos operacionales deben estar diseñadas con el objetivo de hacer esta tarea lo más eficiente posible.

Si se contraponen las características de una base de datos operacional y un DW, 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 algunas de las diferencias existentes entre las dos:

	OLTP	Data Warehouse
Objetivo	Soportar actividades transaccionales diarias.	Consultar y analizar información estratégica y táctica.
Tipo de datos	Operacionales.	Para la toma de decisiones.
Modelo de datos	Normalizado.	Desnormalizado.
Consulta	SQL.	SQL más extensiones.
Datos consultados	Actuales.	Actuales e históricos.
Horizonte de tiempo	60 - 90 días.	5 - 10 años.
Tipos de consultas	Repetitivas, predefinidas	No previsibles, dinámicas
Nivel de almacenamiento	Nivel de detalle.	Nivel de detalle y diferentes niveles de sumarización.
Acciones disponibles	Alta, baja, modificación y consulta.	Carga y consulta.
Número de transacciones	Elevado	Medio o bajo
Tamaño	Pequeño - Mediano.	Grande.
Tiempo de respuesta	Pequeño (segundos - minutos).	Variable (minutos - horas).
Orientación	Orientado a las aplicaciones.	Orientado al negocio.
Sello de tiempo	La clave puede o no tener un elemento de tiempo.	La clave tiene un elemento de tiempo.
Estructura	Generalmente estable.	Generalmente varía de acuerdo a su propia evolución y utilización.

Tabla 1 Comparativa entre Base de Datos Operacional y DWs

3.4. Arquitectura de un Data Warehouse

En este punto y teniendo en cuenta que ya se han detallado las características generales del Data Warehousing, se definirán y describirán todos los componentes que intervienen en su arquitectura o ambiente utilizando el gráfico de la Figura 3.1.

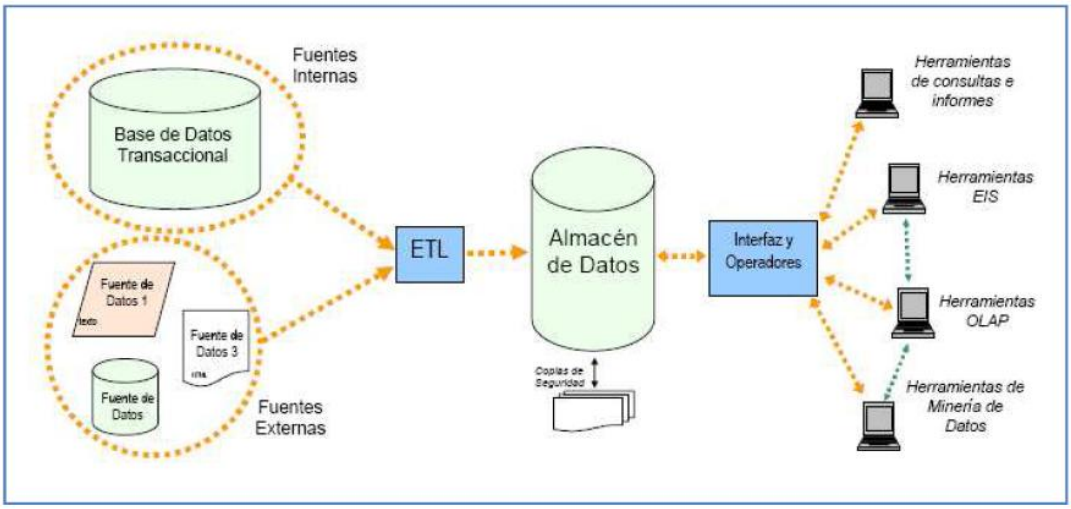


Figura 3.1. Arquitectura Data Warehouse

Tal y como se puede apreciar, el ambiente está formado por diversos elementos que interactúan entre sí y que cumplen una función específica dentro del sistema. Concretamente, en una arquitectura Data Warehouse encontramos los siguientes componentes:

- Datos operacionales procedentes de las bases de datos transaccionales y de otras fuentes de datos que genera la empresa en su accionar diario.
- Sistema ETL que realiza las funciones de extracción de las fuentes de datos (transaccionales o externas), transformación (limpieza, consolidación, etc.) y la carga del almacén, realizando las siguientes acciones:
  - Extracción de los datos.
  - Filtrado de los datos: limpieza, consolidación, etc. Un ejemplo de limpieza de datos sería eliminar los datos con valores nulos o incorrectos.
  - Carga inicial del almacén: ordenación, agregaciones, etc.
  - Refresco del almacén: operación periódica que propaga los cambios de las fuentes externas al almacén de datos.
- Repositorio propio de datos que contiene información relevante y metadatos[10].
- Interfaces y Gestores de Consulta que permiten acceder a los datos y sobre los que se conectan herramientas más sofisticadas (OLAP, minería de datos).
- Sistemas de Integridad y Seguridad que se encargan de un mantenimiento global, copias de seguridad, etc.

3.4.1. Sistema ETL de un Data Warehouse

La migración de los datos desde las fuentes operacionales al DW requiere la necesidad de procesos para extraer, transformar y cargar los datos, actividad que se conoce como ETL.

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

[10] Los metadatos son datos que describen o dan información de otros datos, que en este caso, existen en la arquitectura del Data Warehousing. Brindan información de localización, estructura y significado de los datos, básicamente mapean los mismos.

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 aplicaran 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.

3.4.2. Datamarts

Las corporaciones de hoy se esfuerzan por conducir sus negocios hacia una base internacional. Vemos compañías que surgieron en Estados Unidos y se expandieron a Europa, Asia y África. La expansión del negocio crea la necesidad de acceder a datos corporativos que están ubicados en diferentes puntos geográficos. Por ejemplo, un ejecutivo de ventas de una compañía con origen en Brasil que está situado en Chile puede necesitar acceso a la base de datos de la empresa para identificar los clientes potenciales que residen en Chile.

Este problema se soluciona creando versiones más pequeñas del DW, los Datamarts. Estas versiones se crean usando algún criterio particular, como por ejemplo el lugar geográfico. En el ejemplo anterior los datos de los clientes que residen en Chile se deben almacenar en el Datamart de la sucursal en ese país.

La existencia de los Datamarts crea nuevas formas de pensar cuando se diseñan los repositorios corporativos de datos. Algunas corporaciones reemplazan completamente el concepto de tener un DW central, por varios Datamarts más pequeños que se alimenten directamente de los sistemas operacionales (Figura 3.2.).

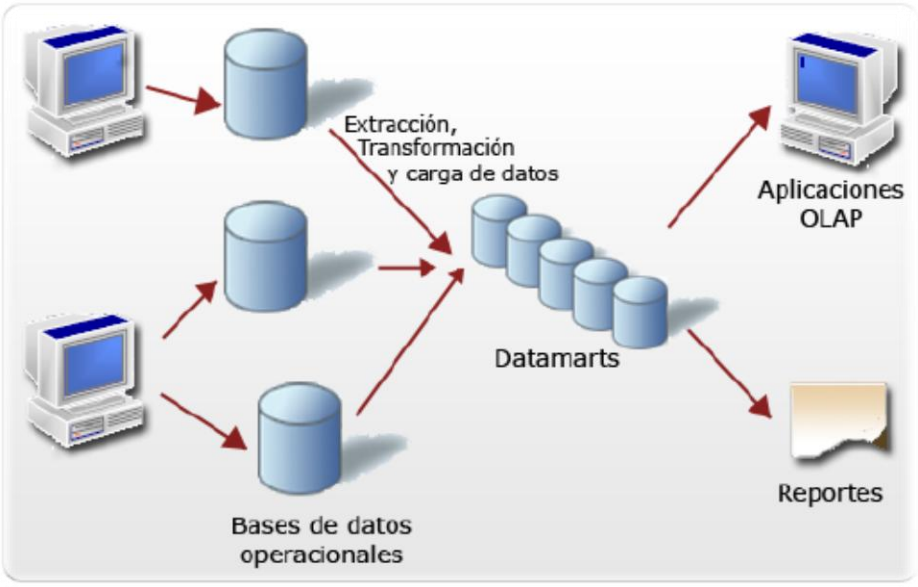


Figura 3.2. Arquitectura Datamarts

Otras compañías usan Datamarts para complementar sus DWs (Figura 3.3.). Mueven datos desde el DW hacia varios Datamarts con el fin de permitir un análisis más eficiente. La separación de los datos se determina según criterios como departamentos, áreas geográficas, periodos de tiempo, etc.

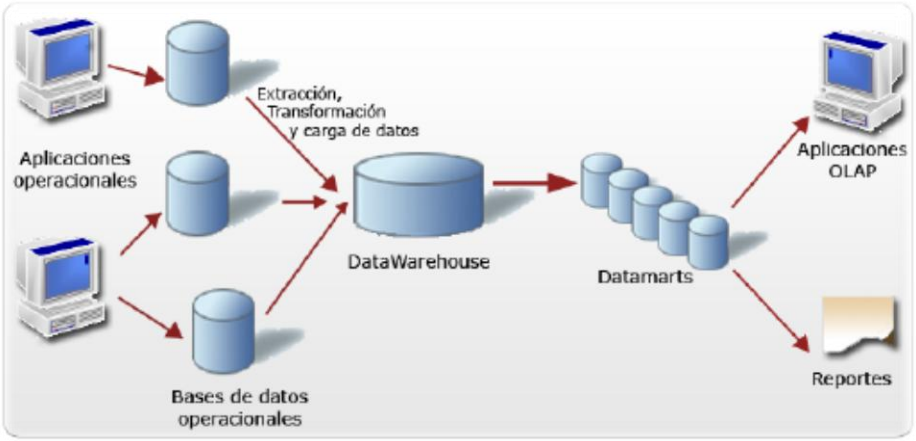


Figura 3.3. Arquitectura Datamarts como complemento al Data Warehouse

Finalmente, algunas organizaciones usan sus Datamarts como el primer paso de almacenamiento de datos operacionales. Luego los datos de todos los Datamarts se replican en un DW corporativo central (Figura 3.4.).

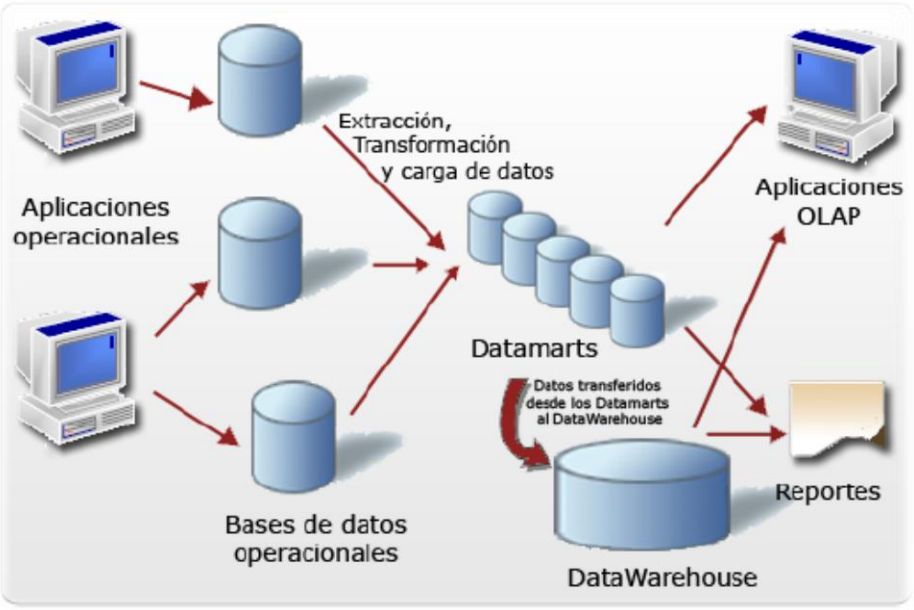


Figura 3.4. Arquitectura Datamarts como almacenamiento intermedio al DW

3.5. Diseño de un Data Warehouse

Debido a las diferencias en el propósito y objetivos de las bases de datos operacionales con las bases orientadas a análisis se originaron técnicas de diseño diferentes para estas últimas.

Al igual que el resto de proyectos de desarrollo software, la construcción de un DW sigue una serie de fases:

1. **Recogida y análisis de requisitos:** Se trata de discernir las fuentes necesarias del sistema de información de la organización (OLTP) y analizar los requisitos de usuario para detectar las consultas de análisis necesarias, nivel de agregación,etc. En esta etapa se deben tener claro cuáles son los diferentes focos sobre los que se va a centrar el almacén de datos.
2. **Diseño conceptual:** Se modela el sistema haciendo uso de modelos tales como el Entidad-Relación.
3. **Diseño lógico:** A partir de uno de los esquemas multidimensionales existentes, se realiza el modelado multidimensional de la base de datos.
4. **Diseño físico:** Se define el esquema a seguir en la organización física del DW y las herramientas OLAP (ROLAP, MOLAP u HOLAP[11]) y se diseña el ETL.
5. **Implementación:** Se realiza la carga del almacén (ETL) y la preparación de las vistas de usuario (herramientas OLAP).

Dado que ya conocemos las principales características de este tipo de base de datos, a continuación veremos cuáles son los requerimientos en cuanto al diseño lógico y físico de las mismas.

[11] ROLAP, MOLAP y HOLAP hacen referencia a las distintas formas existentes de almacenar datos dándoles la estructura adecuada para el análisis en línea. Estos sistemas serán analizados en un punto posterior.

3.5.1. Diseño lógico

Las premisas básicas en el diseño lógico son:

- La mayoría de los analistas de negocios van a querer ver datos totalizados. Estos datos en lo posible deben precalcularse y almacenarse de antemano para que esta recuperación sea rápida y eficiente. Es importante además discutir el nivel de granularidad y de detalle esperado por los analistas cuando hacen operaciones de Drill Down.
- El diseño debe estar dirigido por el acceso y por el uso, es decir, teniendo en cuenta qué tipo de reportes o resúmenes son los más frecuentes, y cuales los más urgentes.
- Un diseño normalizado no es bueno porque no resulta demasiado intuitivo para una persona de negocios, y podría volverse demasiado complejo.
- Todos los datos que se incluyan ya deben existir en las fuentes de datos operacionales, o ser derivables a partir de ellos.

Las dos técnicas de diseño más populares y utilizadas son el esquema en estrella y el esquema en copo de nieve. Estas técnicas siguen un esquema multidimensional de datos, el cual han adoptado las herramientas de explotación. De esta forma se ofrece al usuario una visión multidimensional de los datos que son objeto de análisis.

Esquema Estrella (*Star*)

Este esquema está formado por un elemento central que consiste en una tabla llamada **Tabla de Hechos**, que está conectada a varias **Tablas de Dimensiones**.

Las tablas de hechos contienen los valores precalculados que surgen de totalizar valores operacionales atómicos según las distintas dimensiones, tales como clientes, productos o periodos de tiempo.

Representan un evento crítico y cuantificable en el negocio, como ventas o costos. Su clave está compuesta por las claves primarias de las tablas de dimensión relacionadas (las foreign keys). Pueden existir varias tablas de hechos con información redundante, porque podrían contener distintos niveles de agregación de los mismos datos. Por ejemplo podría existir una tabla de hechos para las Ventas por Sucursal, Región y Fecha, otra para Ventas por Productos, Sucursal y Fecha, y otra tabla que almacene las Ventas por Cliente, Región y Fecha. En general las tablas de hechos tienen muchas filas y relativamente pocas columnas.

Las tablas de dimensión representan las diferentes perspectivas desde donde se ven y analizan los hechos de la tabla de hechos. A diferencia de las anteriores, su clave primaria está formada por un solo atributo, y su característica principal es que están desnormalizadas. Esto significa que si la dimensión incluye una jerarquía, las columnas que la definen se almacenan en la misma tabla dando lugar a valores redundantes, lo cual es aceptable en este esquema.

En general suelen tener muchas columnas pero pocas filas. Siempre que sea posible, es conveniente compartir las tablas de dimensión entre distintas tablas de hechos.

Una de las dimensiones más comunes es la que representa el tiempo, con atributos que describen periodos para años, cuatrimestres, periodos fiscales, y periodos contables. Otras dimensiones comunes son las de clientes, productos, representantes de ventas, regiones y sucursales.

En la Figura 3.5. vemos un ejemplo de esquema Estrella, donde la tabla de hechos es la tabla Ventas, y el resto son las tablas de dimensiones.

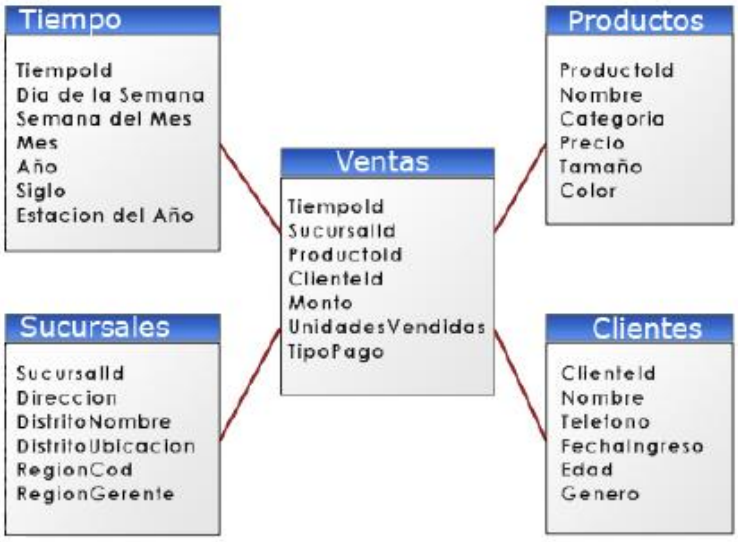


Figura 3.5. Esquema Estrella

El esquema estrella es el más usado porque maneja bien el rendimiento de consultas y reportes que incluyen años de datos históricos, y por su simplicidad en comparación con una base de datos normalizada.

Esquema Copo de Nieve (*Snowflake*)

Es una variante al esquema estrella en el cual las tablas de dimensión están normalizadas, es decir, pueden incluir claves que apuntan a otras tablas de dimensión.

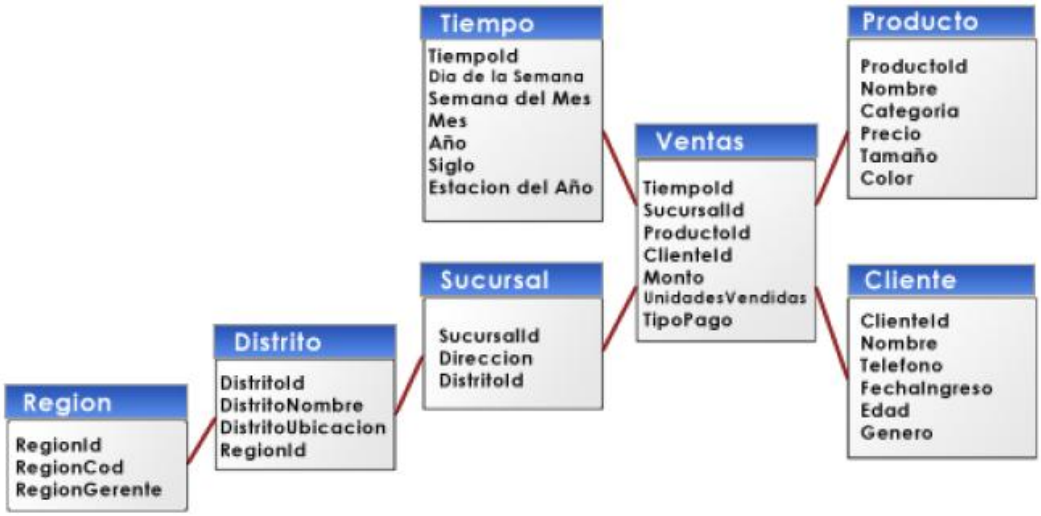


Figura 3.6. Esquema Copo de Nieve

En la 3.6. vemos un esquema similar al anterior, donde la tabla de dimensión Sucursal se expande en las tablas Distrito y Región. Ahora la tabla Sucursal contiene una columna clave DistritoId que apunta a la tabla Distrito, y esta a su vez tiene una columna RegionId que apunta a la tabla de dimensión Región.

Las ventajas de esta normalización son la reducción del tamaño y redundancia en las tablas de dimensión, y un aumento de flexibilidad en la definición de dimensiones.

Sin embargo, el incremento en la cantidad de tablas hace que se necesiten más operaciones de JOINs para responder a las consultas, lo que empeora el rendimiento. Otra desventaja es el mantenimiento adicional que requieren las tablas adicionales.

Una vez seleccionado el esquema multidimensional, para construir el almacén de datos seguiremos alguna de las metodologías de diseño de almacenes de datos existentes. Una de las metodologías más conocidas es la de Kimball [6][7], la cual está basada en el modelo relacional. Según esta metodología, los pasos a seguir en el diseño de un almacén de datos son los siguientes:

1. **Elegir un proceso de la organización para modelar.**

   Un proceso es una actividad de la organización soportada por un OLTP del cual se puede extraer información con el propósito de construir el almacén de datos. Es decir, es cada uno de los focos que hemos identificado en la etapa de análisis.

   Ejemplos de estos procesos podrían ser las ventas de productos, los pedidos de clientes o las compras a los proveedores.

2. **Decidir el gránulo o nivel de detalle de representación.**

   El gránulo es el nivel de detalle al que se desea almacenar información sobre la actividad a modelar (información diaria, semanal, mensual, etc.). Permite definir el nivel atómico de datos en el almacén de datos, determina el significado de las tuplas de la tabla de hechos y determina las dimensiones básicas del esquema.

   En un almacén de datos se almacena información a un nivel de detalle fino, no porque se vaya a interrogar el almacén a ese nivel sino porque ello permite clasificar y analizar la información desde muchos puntos de vista.

   Sin embargo es importante hacer un esfuerzo para no almacenar más detalle del necesario, o el consumo de espacio se disparará y pueden surgir problemas de rendimiento cuando se acceda a los datos.

3. **Identificar las dimensiones que caracterizan el proceso.**

   Se identifican las dimensiones que caracterizan la actividad al nivel de detalle que se ha elegido. De cada dimensión se debe decidir los atributos relevantes para el análisis de la actividad.

   Si entre los atributos de una dimensión existieran jerarquías naturales, éstas deben ser identificadas.

   En el caso de la dimensión tiempo, existe una jerarquía entre los atributos día-mes-año.

4. **Decidir la información a almacenar sobre el proceso.**

   Se decide la información que se desea almacenar en cada tupla de la tabla de hechos y que será objeto de análisis.

   Dicha información podría ser el importe total de las ventas de un producto en el día o el número total de clientes distintos que han comprado el producto en el día.

3.5.2. Diseño físico

Debido a que los DWs trabajan tanto con datos detallados como con datos resumidos, y frecuentemente almacenan algunos datos en forma redundante, el tamaño de estas bases

de datos puede ser enorme. Las bases de datos que se acercan o exceden el terabyte son llamadas VLDBs (*Very Large Databases*). Diseñar una base de datos de este tipo es un gran desafío y la tarea de mantenerlas es costosa.

Entre las decisiones de implementación que se deben tomar se incluyen el tamaño del espacio libre, el tamaño del buffer, el tamaño del bloque, y si se usa o no una técnica de compactación de la base de datos. Todas estas cuestiones afectarán el rendimiento del DW.

Los motores de base de datos están basados en reglas internas intrincadas, que deben entenderse y seguirse. Una situación común es que se deje el diseño de la base de datos a los programadores, quienes quizá no estén del todo familiarizados con el funcionamiento interno del motor, y como consecuencia creen un diseño pobre que no aproveche al máximo las características que brinda el producto.

3.6. Explotación de un Data Warehouse: Herramientas OLAP

Las herramientas OLAP se usan para convertir los datos corporativos, almacenados en la base de datos orientada al análisis (DW), en conocimiento útil para la toma de decisiones.

Mientras que el DW almacena la información a secas, es decir, tal y como ha sido obtenida de la base de datos operacional, los sistemas OLAP hacen agregaciones y sumarizaciones de estos datos, y los organizan en cubos o almacenamientos especiales para permitir una rápida recuperación ante una consulta.

3.6.1. ¿Qué es OLAP?

OLAP se define como el análisis multidimensional e interactivo de la información de negocios a escala empresarial. El análisis multidimensional consiste en combinar distintas áreas de la organización, y así ubicar ciertos tipos de información que revelen el comportamiento del negocio.

¿Por qué se dice que el análisis es interactivo?

Los usuarios de la herramienta OLAP se mueven suavemente desde una perspectiva del negocio a otra, por ejemplo, pueden estar observando las ventas anuales por sucursal y pasar a ver las sucursales con más ganancias en los últimos tres meses, y además con la posibilidad de elegir entre diferentes niveles de detalle, como ventas por día, por semana o por cuatrimestre. Es esta exploración interactiva lo que distingue a OLAP de las herramientas simples de consulta y reportes.

¿Por qué es útil la multidimensionalidad?

Es lo que permite a los analistas de negocios examinar sus indicadores clave o medidas, como ventas, costos, y ganancias, desde distintas perspectivas, como periodos de tiempo, productos y regiones. Estas perspectivas constituyen las dimensiones desde las que se explora la información.

¿Por qué a escala empresarial?

OLAP es robusto y escalable, al punto de permitir satisfacer completamente las necesidades de análisis de información de la organización. Se trabaja con fuentes de datos corporativas, que contienen datos de toda la empresa, y se comparte y cruza la información existente en todas las áreas de la misma.

Para proveer estas características, toda herramienta OLAP tiene tres principales componentes:

- Un modelo multidimensional de la información para el análisis interactivo: **cubo multidimensional**.
- Un motor OLAP que procesa las consultas multidimensionales sobre los datos: **arquitectura ROLAP, MOLAP o HOLAP**.
- Un mecanismo de almacenamiento para guardar los datos que se van a analizar: **Data Warehouse**.

Los usuarios de OLAP se centran en los conceptos de negocios, trabajando intuitivamente con ellos, sin necesidad de conocer cuestiones técnicas tales como el formato físico de los datos, instrucciones de lenguajes como SQL, nombres de tablas o columnas en la base de datos, o la arquitectura OLAP subyacente.

3.6.2. Diferencias entre procesamiento OLAP y OLTP

La compra, venta, producción, y distribución son ejemplos comunes de actividades de negocios del día a día. Estas actividades constituyen el llamado procesamiento operacional u OLTP (*Online Transactional Processing*), y las aplicaciones que soportan este procesamiento se diseñan con orientación a la carga de datos.

El planeamiento de recursos, planeamiento financiero, alianzas estratégicas, e iniciativas de mercado son ejemplos de actividades que generan y usan información basada en análisis y orientada a la toma de decisiones. Este tipo de actividades son soportadas por las aplicaciones de tipo OLAP.

Si se contraponen las características de OLAP y OLTP podemos apreciar las siguientes diferencias:

OLAP	OLTP
Define el comportamiento de un sistema de análisis de datos	Define el comportamiento habitual de un entorno operacional de gestión
Sólo Consulta	Altas/Bajas/Modificaciones/Consultas
Consultas pesadas y no predecibles	Consultas rápidas y escuetas
Gran volumen de información histórica	Poco volumen de información
Operaciones lentas	Transacciones rápidas
Bajo nivel de concurrencia. Usado a nivel gerencial y analistas de negocio	Gran nivel de concurrencia

Tabla 2 Comparativa entre OLAP y OLTP

Todas estas diferencias se reflejan en las características de las bases de datos subyacentes a ambos tipos de aplicaciones, y hacen que no sea posible la convivencia en una única base de datos de los entornos OLAP y OLTP.

3.6.3. Cubos, Dimensiones, Medidas y Operaciones aplicables

Cubo multidimensional

Un cubo es un subconjunto de datos de un almacén, que provee un mecanismo para buscar información con rapidez y en tiempos de respuesta uniforme independientemente de la cantidad de datos que lo formen o la complejidad del procedimiento de búsqueda.

Estos cubos multidimensionales o cubos OLAP se generan a partir de los esquemas en estrella o copo de nieve diseñados en el DW. Las tablas relacionadas en éste (tablas de hechos y de dimensiones) proporcionan la estructura de datos al cubo (Figura 3.7.).

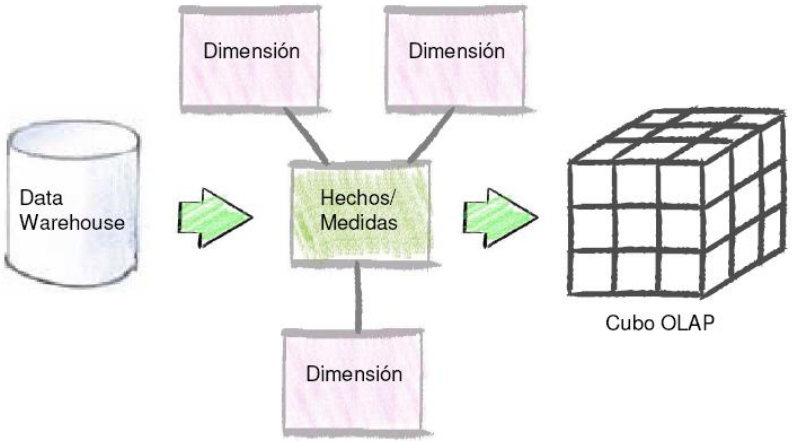


Figura 3.7. Estructura cubo multidimensional

En un cubo, la información se representa por medio de matrices multidimensionales o cuadros de múltiples entradas, que nos permite realizar distintas combinaciones de sus elementos para visualizar los resultados desde distintas perspectivas y variando los niveles de detalle. Esta estructura es independiente del sistema transaccional de la organización, y facilita y agiliza la consulta de información histórica ofreciendo la posibilidad de navegar y analizar los datos.

Aquí vemos como ejemplo un cubo multidimensional (Figura 3.8.) que contiene información de ventas discriminadas por periodos de tiempo, productos y zonas geográficas de las sucursales.

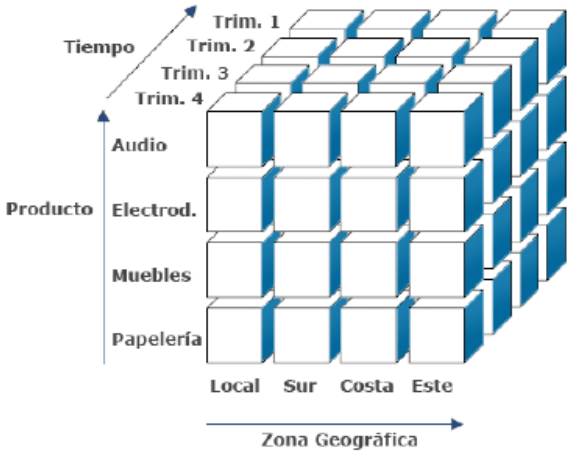


Figura 3.8. Ejemplo cubo multidimensional

Los ejes del cubo son las Dimensiones, y los valores que se presentan en la matriz, son las Medidas. Si comparamos estos elementos con los componentes del esquema multidimensional del DW, observamos que las dimensiones se corresponden con las diferentes tablas de dimensiones definidas en dicho esquema mientras que las medidas se corresponden con los tipos de valores precalculados contenidos en la tabla de hechos del esquema.

Una instancia del modelo está determinada por un conjunto de datos para cada eje del cubo y un conjunto de datos para la matriz.

Dimensiones

Son objetos del negocio con los cuales se puede analizar la tendencia y el comportamiento del mismo. Las definiciones de las dimensiones se basan en políticas de la compañía o del mercado, e indican la manera en que la organización interpreta o clasifica su información para segmentar el análisis en sectores, facilitando la observación de los datos.

Para determinar las dimensiones requeridas para analizar los datos podemos preguntarnos:

- ¿Cuándo? La respuesta a esta pregunta permite establecer la dimensión del tiempo y visualizar de manera comparativa el desempeño del negocio.
- ¿Dónde? Esta pregunta nos ubica en un área física o imaginaria donde se están llevando a cabo los movimientos que se desean analizar. Estos lugares se pueden ser zonas geográficas, divisiones internas de la organización, sucursales, etc.
- ¿Qué? Es el objeto del negocio, o el objeto de interés para determinada área de la compañía. Para estos casos se tienen los productos y/o servicios, la materia prima como elemento de interés para la división de abastecimientos, los vehículos para la sección de transportes, las maquinarias para el área de producción, etc.
- ¿Quién? En esta dimensión se plantea una estructura de los elementos que inciden directamente sobre el objeto de interés. En estos casos se hace referencia al área comercial o de ventas, o a los empleados de la organización si se está llevando a cabo un análisis a nivel del talento humano, etc.
- ¿Cuál? Habla de hacia dónde se enfoca el esfuerzo de la organización o una determinada área del negocio, para hacer llegar los productos o servicios. Una dimensión que surge de esta pregunta es la de clientes.

Medidas

Son características cualitativas o cuantitativas de los objetos que se desean analizar en las empresas. Las medidas cuantitativas están dadas por valores o cifras porcentuales.

Por ejemplo, las ventas en dólares, cantidad de unidades en stock, cantidad de unidades de producto vendidas, horas trabajadas, el promedio de piezas producidas, el porcentaje de aceptación de un producto, el consumo de combustible de un vehículo, etc.

Jerarquías de Dimensiones y Niveles

Generalmente las dimensiones se estructuran en jerarquías, y en cada jerarquía existen uno o más niveles, los llamados Niveles de Agregación o simplemente Niveles. Toda dimensión tiene por lo menos una jerarquía con un único nivel.

En la Figura 3.9. vemos un ejemplo de una dimensión de vendedores, que consiste de una única jerarquía, y tres niveles de agregación para agruparlos por ciudades y por regiones.

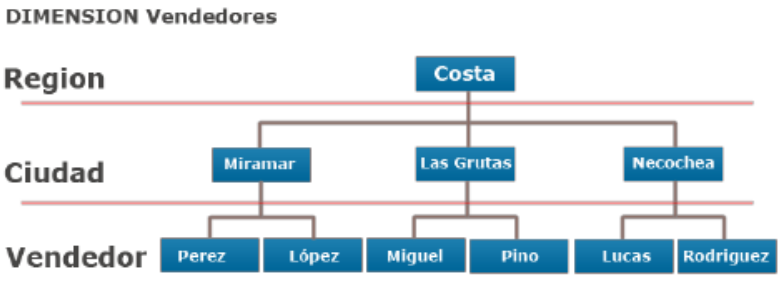


Figura 3.9. Ejemplo de una jerarquía y tres niveles de agregación

En la Figura 3.10. podemos ver una dimensión que cuenta con dos jerarquías:

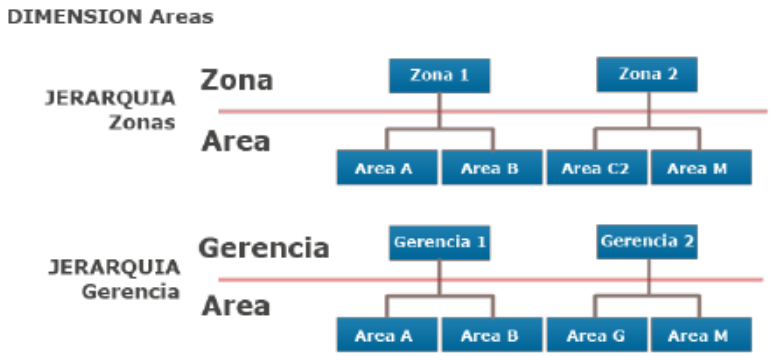


Figura 3.10. Ejemplo de dos jerarquías

En este caso, los niveles Zonas y Gerencia no están relacionados entre sí, a pesar de que ambos están relacionados con la dimensión Areas.

Operaciones multidimensionales

Una consulta a una herramienta OLAP consiste generalmente en la obtención de medidas sobre los hechos, parametrizadas por atributos de las dimensiones y restringidas por condiciones impuestas sobre las dimensiones.

Así por ejemplo, podríamos tener una consulta que enunciará lo siguiente: "*Importe total de las ventas durante el año 2010 de los productos del tipo limpieza, por trimestre y por producto*". Si leemos detenidamente la consulta podemos 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 obtendríamos el siguiente resultado:

Producto	**Trimestre**			
	Trimestre I	Trimestre II	Trimestre III	Trimestre IV
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), lo realmente interesante de las herramientas OLAP son sus operadores de refinamiento o manipulación de consultas. Estos operadores facilitan la agregación (consolidación) y la disgregación (división) de los datos:

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

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

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

Producto	Mes											
	E	F	M	A	M	J	J	A	S	O	N	D
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

Si las operaciones de agregación y disgregación se hacen sobre atributos de una dimensión sobre los que se ha definido una jerarquía, estas operaciones reciben el nombre de *Drill-Down* y *Roll-Up* respectivamente. Sin embargo, si dichas operaciones se hacen sobre atributos de dimensiones independientes, las operaciones reciben el nombre de *Drill-Across* y *Roll-Across*.

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

Producto	Trimestre	
	Trimestre I	Trimestre III
Limpiacristales	8 000	9 000

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

Trimestre	Producto		
	Amoniaco	Friegasuelos	Limpiacristales
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

3.6.4. ROLAP, MOLAP y HOLAP

Los cubos, las dimensiones y las jerarquías son la esencia de la navegación multidimensional del OLAP. Al describir y representar la información en esta forma, los usuarios pueden navegar intuitivamente en un conjunto complejo de datos. Sin embargo, el solo describir el modelo de datos en una forma más intuitiva, hace muy poco para ayudar a entregar la información al usuario más rápidamente.

Un principio clave del OLAP es que los usuarios deberían obtener tiempos de respuesta consistentes para cada vista de datos que requieran. Dado que la información se colecta en el nivel de detalle solamente, el resumen de la información es usualmente calculado por adelantado. Estos valores precalculados son la base de las ganancias de desempeño del OLAP.

En los primeros días de la tecnología OLAP, la mayoría de las compañías asumía que la única solución para una aplicación OLAP era un modelo de almacenamiento no relacional. Después, otras compañías descubrieron que a través del uso de estructuras de base de datos (esquemas de estrella y de copo de nieve), índices y el almacenamiento de agregados, se podrían utilizar sistemas de administración de bases de datos relacionales (RDBMS) para el OLAP.

Estos vendedores llamaron a esta tecnología OLAP relacional (ROLAP). Las primeras compañías adoptaron entonces el término OLAP multidimensional (MOLAP), estos conceptos, MOLAP y ROLAP, se explican con más detalle en los siguientes párrafos. Las implementaciones MOLAP normalmente se desempeñan mejor que la tecnología ROLAP, pero tienen problemas de escalabilidad. Por otro lado, las implementaciones ROLAP son más escalables y son frecuentemente atractivas a los clientes debido a que aprovechan las inversiones en tecnologías de bases de datos relacionales preexistentes.

Sistemas MOLAP

La arquitectura MOLAP usa unas bases de datos multidimensionales para proporcionar el análisis, su principal premisa es que el OLAP está mejor implantado almacenando los datos multidimensionalmente. Por el contrario, la arquitectura ROLAP cree que las capacidades OLAP están perfectamente implantadas sobre bases de datos relacionales Un sistema MOLAP usa una base de datos propietaria multidimensional, en la que la información se almacena multidimensionalmente, para ser visualizada en varias dimensiones de análisis.

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 (MDDB), se realizan una serie de cálculos por lotes, para calcular los datos agregados, a través de las dimensiones de negocio, rellenando la estructura MDDB.

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 MDDB está lista para su uso. Los usuarios solicitan informes a través de la interfaz, y la lógica de aplicación de la MDDB obtiene el dato.

La arquitectura MOLAP requiere unos cálculos intensivos de compilación. Lee de datos precompilados, y tiene capacidades limitadas de crear agregaciones dinámicamente o de hallar ratios que no se hayan precalculados y almacenados previamente.

Algunos fabricantes son: Oracle's Hyperion Essbase, Microsoft Analysis Services, TM1, SAS OLAP, Cognos PowerCubes.

Sistemas ROLAP

La arquitectura ROLAP, accede a los datos almacenados en un Data Warehouse para proporcionar los análisis OLAP. La premisa de los sistemas ROLAP es que las capacidades OLAP se soportan mejor contra las bases de datos relacionales.

El sistema ROLAP utiliza una arquitectura de tres niveles (Figura 3.11.). 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 Data Warehouse se ha definido, los datos se cargan desde el sistema operacional. Se ejecutan rutinas de bases de datos para agregar el dato, si así es requerido por el modelo de datos. Se crean entonces los índices para optimizar los tiempos de acceso a las consultas.

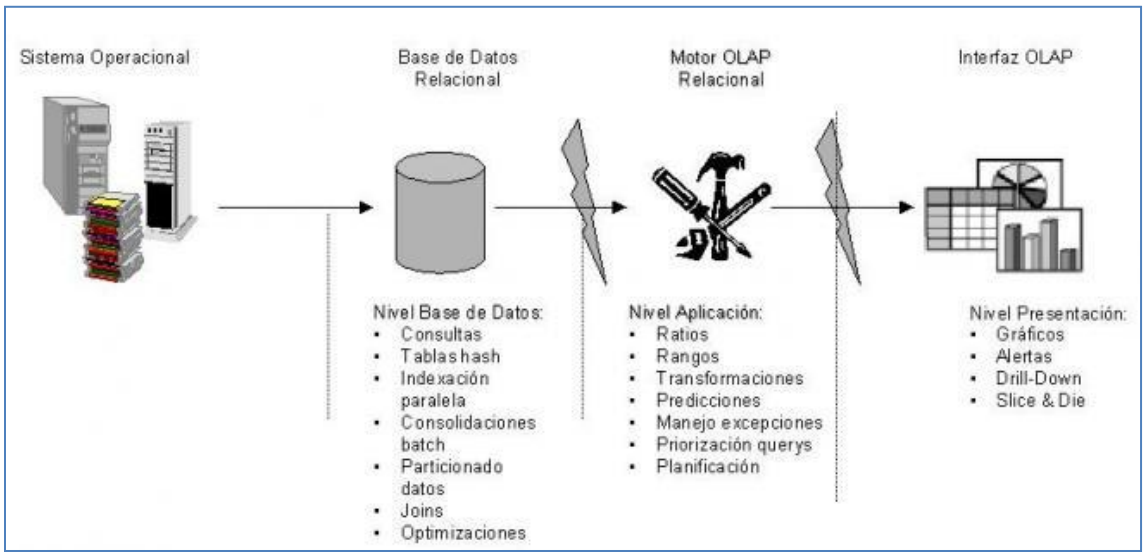


Figura 3.11. Sistema ROLAP

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.

La arquitectura ROLAP es capaz de usar datos precalculados si estos están disponibles, o de generar dinámicamente los resultados desde los datos elementales si es preciso. Esta arquitectura accede directamente a los datos del Data Warehouse, y soporta técnicas de optimización de accesos para acelerar las consultas. Estas optimizaciones son, entre otras, particionado de los datos a nivel de aplicación, soporte a la desnormalización y joins múltiples.

Algunos fabricantes son: Oracle's BI EE, SAP Netweaver BI, MicroStrategy, Cognos 8, BusinessObjects Web Intelligence.

Sistemas HOLAP

Un desarrollo un poco más reciente ha sido la solución OLAP híbrida (HOLAP), la cual combina las arquitecturas ROLAP y MOLAP para brindar una solución con las mejores características de ambas: desempeño superior y gran escalabilidad. Un tipo de HOLAP mantiene los registros de detalle (los volúmenes más grandes) en la base de datos relacional, mientras que mantiene las agregaciones en un almacén MOLAP separado.

Algunos fabricantes son: Microsoft Analysis Services, SAS OLAP, Oracle's Hyperion Essbase.

Cada alternativa tiene sus ventajas y desventajas. Algunas de las ventajas más importantes de cada enfoque son:

MOLAP

- Consultas rápidas debido a la optimización del rendimiento de almacenamiento, la indexación multidimensional y la memoria caché.
- Ocupar menor tamaño en disco en comparación con los datos almacenados en base de datos relacional debido a técnicas de compresión.
- Automatización del procesamiento de los datos agregados de mayor nivel.
- Muy compacto para conjuntos de datos de pocas dimensiones.
- El modelo de almacenamiento en vectores/matrices proporciona una indexación natural.
- Eficaz extracción de los datos lograda gracias a la pre-estructuración de los datos agregados.

ROLAP

- Soportan análisis OLAP contra grandes volúmenes de datos.
- Los sistemas pueden crecer hasta un gran número de dimensiones.
- Los tiempos de carga son mucho menores.
- Los datos se almacenan en una base de datos relacional que puede ser accedida por cualquier herramienta de generación de informes SQL.
- Es posible modelar datos con éxito que de otro modo no se ajustarían a un modelo dimensional estricto.

Resumiendo, el sistema ROLAP es una arquitectura flexible y general, que crece para dar soporte a amplios requerimientos OLAP. MOLAP es una solución particular, adecuada para soluciones departamentales con unos volúmenes de información y número de dimensiones más modestos.

Capítulo 4. Proceso de desarrollo de indicadores

En este capítulo se presenta una metodología que apoya el proceso de desarrollo de los indicadores objeto de este proyecto.

El material presentado en este capítulo ha sido extraído de la siguiente fuente [8].

4.1. ¿Qué es TUNE-UP?

TUNE-UP es una metodología que incorpora aspectos ágiles y tradicionales con un sentido marcadamente pragmático. TUNE-UP se caracteriza fundamentalmente por combinar los siguientes elementos:

- **Modelo iterativo e incremental** para el desarrollo y mantenimiento del software. El trabajo se divide en unidades de trabajo que son asignadas a versiones del producto. Se realizan ciclos cortos de desarrollo, entre 3 y 6 semanas, dependiendo del producto.
- **Workflows flexibles** para la coordinación del trabajo asociado a cada unidad de trabajo. Los productos, según sus características, tienen disponibles un conjunto de workflows los cuales se asignan a cada una de las unidades de trabajo. Cada unidad de trabajo sigue el flujo de actividades del workflow para completarla. Bajo ciertas condiciones se permite saltar hacia adelante o hacia atrás en el workflow, así como cambios de agentes asignados e incluso cambio de workflow. Por ejemplo, las típicas situaciones de re-trabajo en desarrollo de software ocasionadas por detección de defectos se abordan con saltos atrás no explícitos en el workflow.
- **Proceso de desarrollo dirigido por las pruebas de aceptación (Test-Driven)**. La definición de una unidad de trabajo es básicamente la especificación de sus pruebas de aceptación acordadas con el cliente. A partir de allí, todo el proceso gira en torno a ellas, se estima el esfuerzo de implementar, diseñar y aplicar dichas pruebas, se diseñan e implementan y luego se aplican sobre el producto para garantizar el éxito de la implementación.
- **Planificación y seguimiento continuo centrados en la gestión del tiempo**. En todo momento debe estar actualizado el estado de las versiones, de las unidades de trabajo, y del trabajo asignado a los agentes. El jefe del proyecto puede actuar oportunamente con dicha información, tomando decisiones tales como: redistribuir carga de trabajo entre agentes, cambiar los plazos de la versión, mover unidades de trabajo entre versiones, etc.
- **Control de tiempos**. Los agentes registran el tiempo que dedican a la realización de las actividades, el cual se compara con los tiempos estimados en cada una de ellas, detectando oportunamente desviaciones significativas. Esto permite a los agentes gestionar más efectivamente su tiempo, mejorar sus estimaciones y ofrecer al jefe del proyecto información actualizada del estado de la versión.

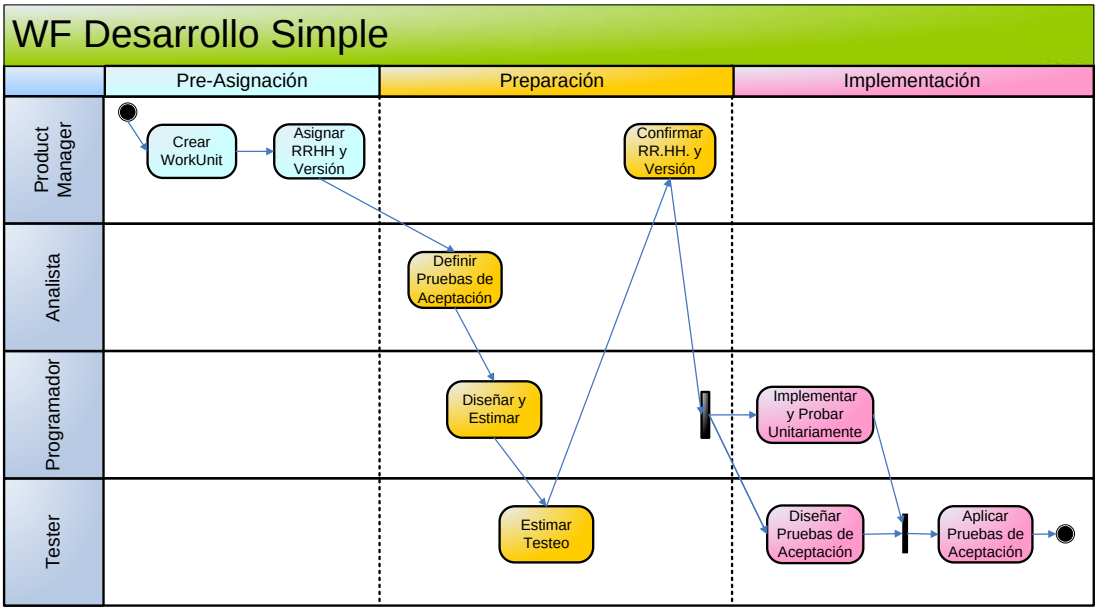


Figura 4.1. WF de desarrollo simple para unidades de trabajo

La Figura 4.1. ilustra un workflow simple para el desarrollo de una unidad de trabajo. Un workflow, en general, incluye actividades asociadas a tres fases del proceso:

- **Pre-Asignación:** actividades realizadas hasta la asignación de los RRHH y versión. La unidad de trabajo ha sido identificada pero aún no tiene una prioridad como para asignarle los RRHH y versión.
- **Preparación:** se pueden realizar cuando la unidad de trabajo ha alcanzado cierta prioridad y se le han asignado los RRHH y una versión. Se comienza a trabajar en la preparación de la unidad de trabajo, y debería concluirse antes del inicio de la versión objetivo en la cual se implementará. Incluye el análisis, las revisiones y estimaciones para su implementación.
- **Implementación:** se realizan durante la versión objetivo. Incluye la implementación, aplicación de pruebas e implantación.

Las actividades de cada workflow pueden variar significativamente dependiendo de factores tales como la cantidad y especialización de agentes participantes, validaciones o negociaciones predeterminadas con el cliente, etc.

Cada unidad de trabajo en una iteración del proyecto podría tener su propio workflow. Sin embargo, en la práctica basta con disponer de un reducido conjunto de workflows que permitan cubrir los tipos de unidades de trabajo que se presentan en el proyecto.

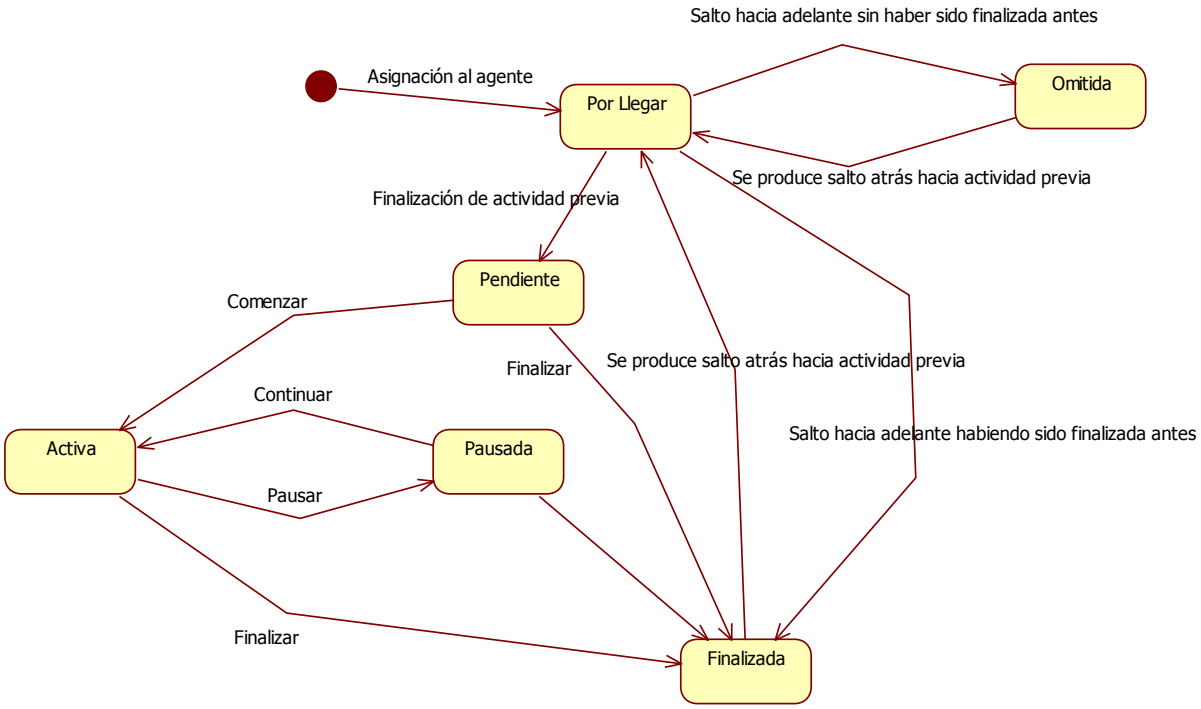


Figura 4.2. Posibles estados de una unidad de trabajo en una actividad

La Figura muestra los posibles estados (Por Llegar, Pendiente, Activa, Pausada, Finalizada y Omitida) en los que se puede encontrar una actividad asociada a una unidad de trabajo.

4.2. TUNE-UP Process Tool

TUNE-UP Process Tool es una herramienta de apoyo para la aplicación efectiva de la metodología TUNE-UP. La herramienta está formada por tres módulos principales: Planificador Personal, Gestor de Unidades de Trabajo, y Planificador de Versiones. A continuación se describe cada uno de estos módulos.

4.2.1. Planificador Personal

El Planificador Personal (PP) presenta el trabajo que tiene un agente. Cuando un agente inicia su jornada laboral, accede al PP (Figura 4.3.) para ver el trabajo que tiene asignado actualmente, y seleccionar qué va a realizar de acuerdo a sus prioridades. El PP ofrece una variedad de facilidades de filtrado y ordenamiento de información, además de datos de tiempos, para que el agente pueda determinar su elección de acuerdo a sus prioridades. La tabla de la izquierda de la figura resume las contabilizaciones de las unidades de trabajo según la actividad y estado en el que se encuentran, y la tabla de la derecha, muestra información de dichas unidades de trabajo incluyendo: producto, versión, descripción de la unidad de trabajo, tiempos calculados, estado de la actividad actual dentro del workflow, etc.

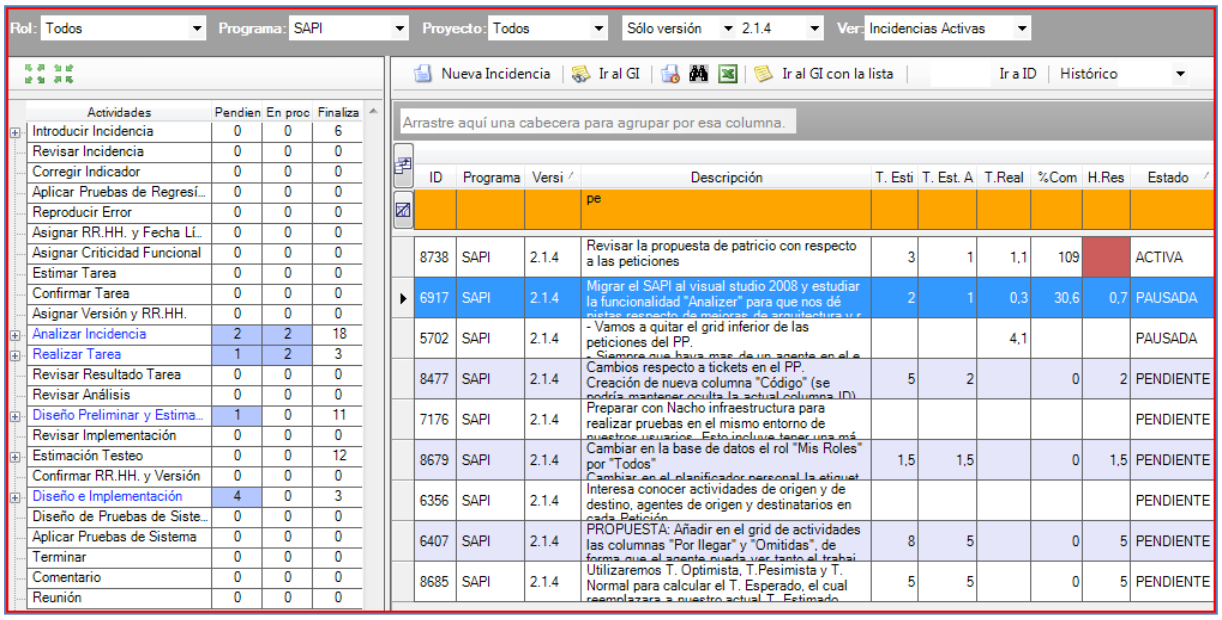


Figura 4.3. Fragmento de interfaz del Planificador Personal

4.2.2. Gestor de Unidades de Trabajo

Cuando el agente decide la unidad de trabajo que va a realizar, accede con ella al Gestor de Unidades de Trabajo (GUT) (Figura 4.4.) como apoyo esencial para realizar su tarea. En la parte superior se observan los datos generales de la unidad de trabajo, y en la parte inferior, un conjunto de pestañas con la funcionalidad que ofrece el GUT. Entre la funcionalidad disponible tenemos la posibilidad de activar/pausar/finalizar la actividad, mandar peticiones, añadir/consultar documentación asociada a la unidad de trabajo, ver los tiempos registrados en dicha unidad, etc.

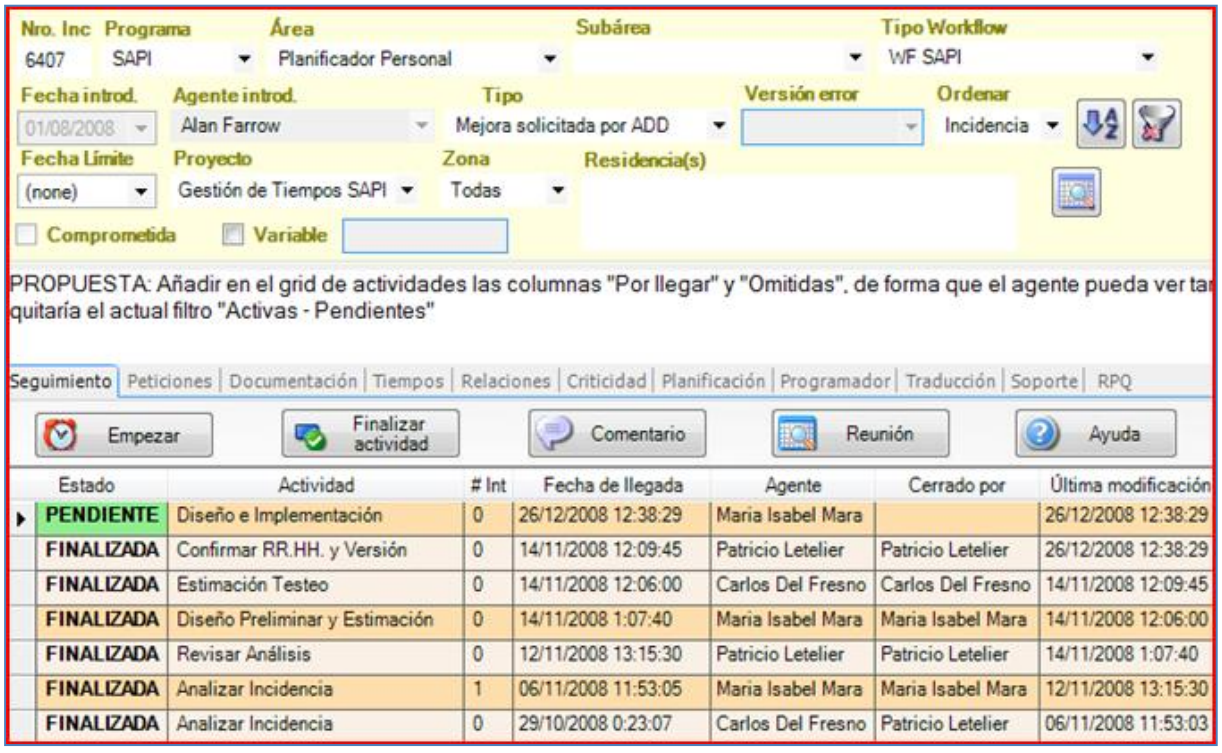
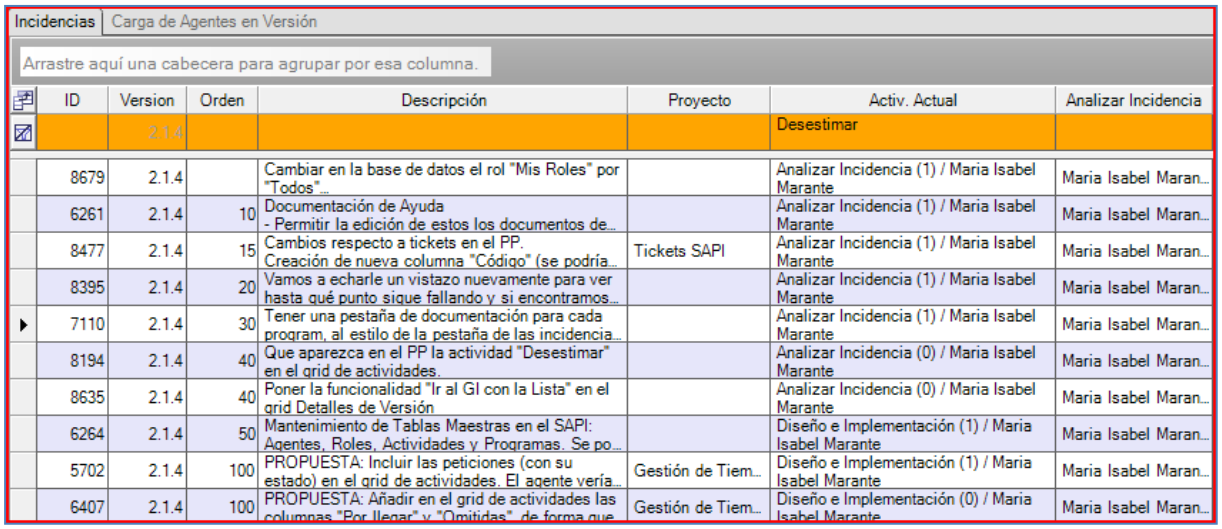


Figura 4.4. Gestor de Unidades de Trabajo

4.2.3. Planificador de Versiones

El Planificador de Versiones (PV) (Figura 4.5.) permite gestionar los productos, sus versiones, los workflows disponibles para cada producto, los agentes por defecto asignados a las actividades de los workflows y realizar el seguimiento continuo del estado actual de las versiones.



ID	Version	Orden	Descripción	Proyecto	Activ. Actual	Analizar Incidencia
	2.1.4				Desestimar	
8679	2.1.4		Cambiar en la base de datos el rol "Mis Roles" por "Todos"...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
6261	2.1.4	10	Documentación de Ayuda - Permitir la edición de estos los documentos de...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
8477	2.1.4	15	Cambios respecto a tickets en el PP. Creación de nueva columna "Código" (se podría...	Tickets SAPI	Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
8395	2.1.4	20	Vamos a echarle un vistazo nuevamente para ver hasta qué punto sigue fallando y si encontramos...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
7110	2.1.4	30	Tener una pestaña de documentación para cada program, al estilo de la pestaña de las incidencia...		Analizar Incidencia (1) / Maria Isabel Marante	Maria Isabel Maran...
8194	2.1.4	40	Que aparezca en el PP la actividad "Desestimar" en el grid de actividades.		Analizar Incidencia (0) / Maria Isabel Marante	Maria Isabel Maran...
8635	2.1.4	40	Poner la funcionalidad "Ir al GI con la Lista" en el grid Detalles de Versión		Analizar Incidencia (0) / Maria Isabel Marante	Maria Isabel Maran...
6264	2.1.4	50	Mantenimiento de Tablas Maestras en el SAPI: Agentes, Roles, Actividades y Programas. Se po...		Diseño e Implementación (1) / Maria Isabel Marante	Maria Isabel Maran...
5702	2.1.4	100	PROPUESTA: Incluir las peticiones (con su estado) en el grid de actividades. El agente vería...	Gestión de Tiem...	Diseño e Implementación (1) / Maria Isabel Marante	Maria Isabel Maran...
6407	2.1.4	100	PROPUESTA: Añadir en el grid de actividades las columnas "Por llegar" y "Omitidas", de forma que	Gestión de Tiem...	Diseño e Implementación (0) / Maria Isabel Marante	Maria Isabel Maran...

Figura 4.5. Planificador de Versiones

4.3. Workflow de desarrollo de indicadores

TUNE-UP dispone de un conjunto de workflows para cada tipo de producto. En el caso de un producto de Business Intelligence, se ha definido un workflow específico para el desarrollo de indicadores (Figura 4.6.).

Tal y como se observa, este WF se compone de numerosas actividades asociadas a diferentes fases del procesos de desarrollo que se ejecutan de forma secuencial o paralela. En la primera fase (parte izquierda) se incluyen las actividades relacionadas con análisis, revisiones y estimaciones; la segunda (parte central delimitada por las líneas discontinuas) abarca las actividades de implementación, testeo e implantación en el cliente del producto final y la fase final (parte derecha) incluye la implantación de pruebas de regresión[12].

[12] Se denominan **Pruebas de regresión** a aquellas pruebas que intentan descubrir defectos producidos por cambios recientemente realizados en partes de la aplicación. Por definición, una prueba aplicada después de la primera vez que resultó exitosa, es una prueba de regresión.

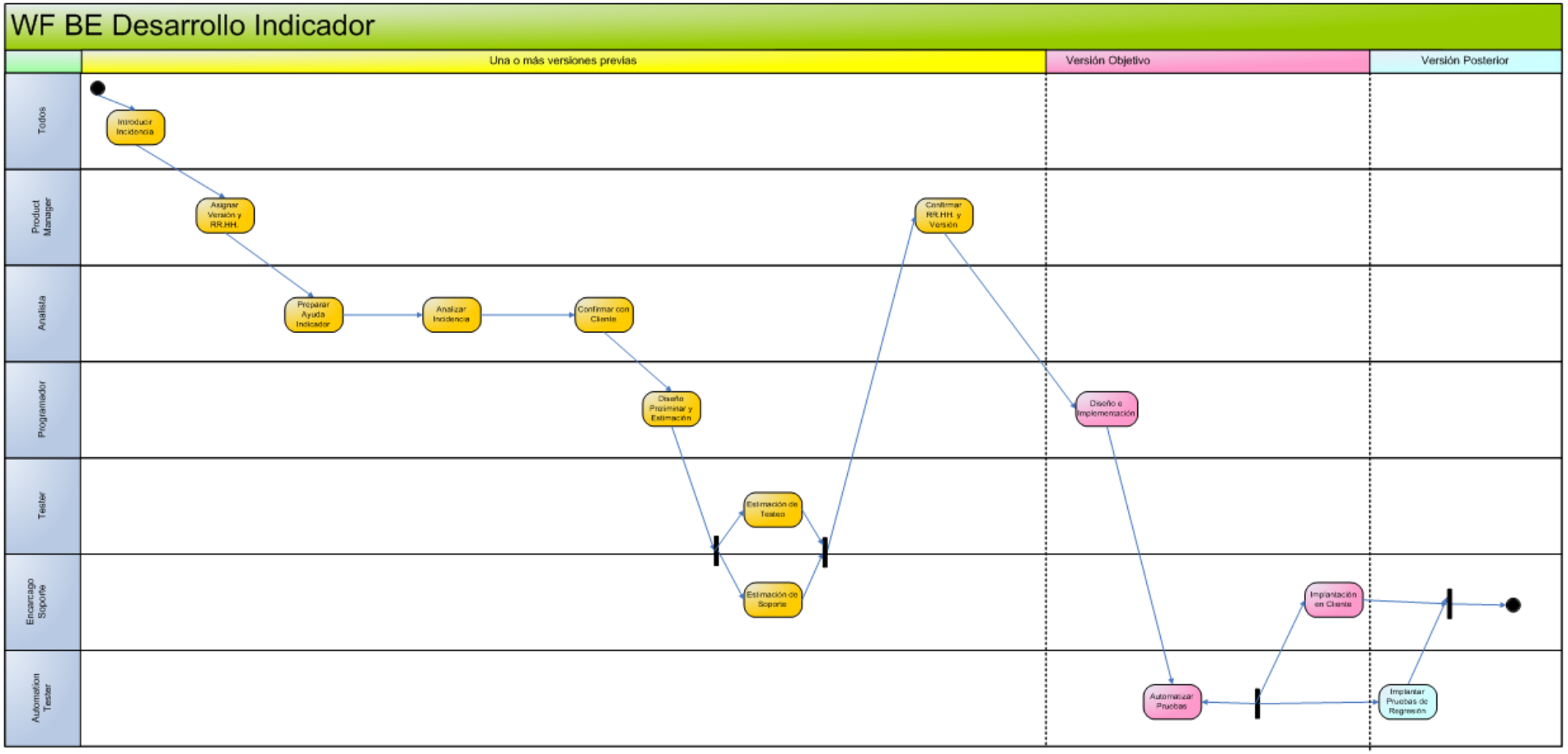


Figura 4.6. WF Desarrollo Indicador

Concretamente, las actividades que componen dicho WF son las siguientes:

- **Introducir Indicador:** Es la actividad de inicio del WF y en ella se rellenan los datos básicos relativos al indicador en el que se va a trabajar (Figura 4.7.). En nuestro caso, se pone una breve descripción del indicador a desarrollar.

  Esta actividad viene precedida por una serie de negociaciones entre el cliente y la empresa encargada del desarrollo del indicador. Una vez finalizado el proceso de negociación, el cliente envía a la empresa desarrolladora una plantilla del indicador deseado (Ver Anexo A) en la que indica la información que desea conocer y es entonces cuando se inicia este WF.

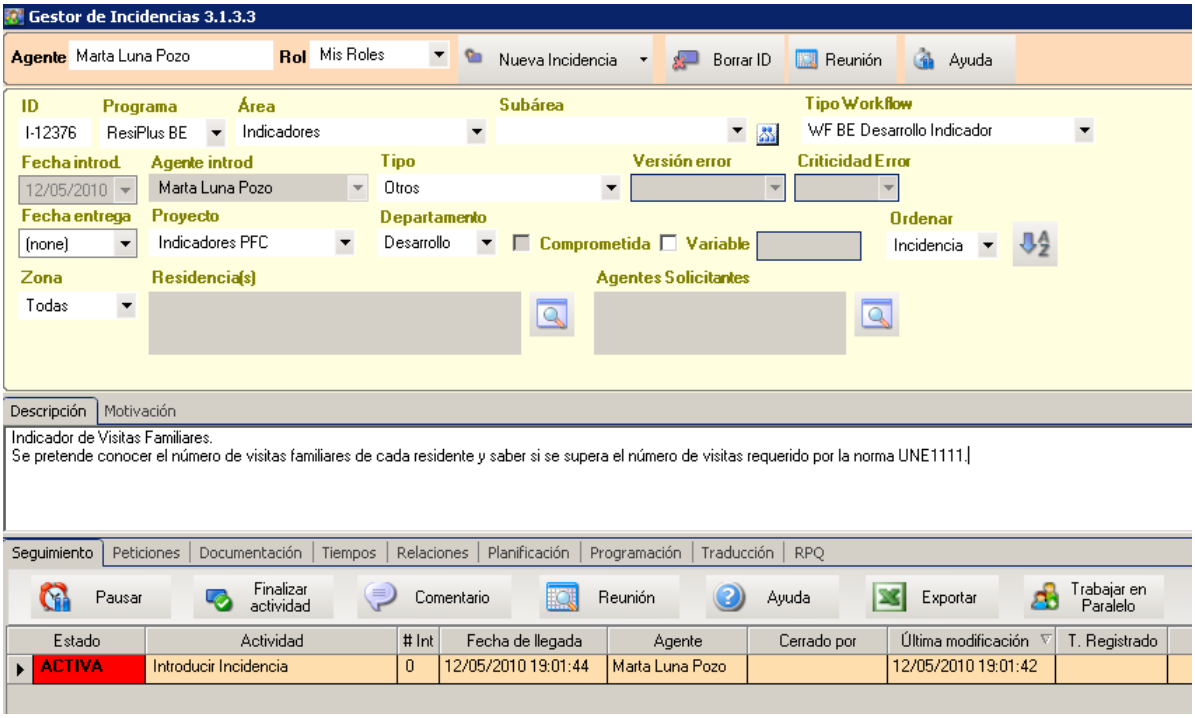


Figura 4.7. Gestor de Unidades de Trabajo Actividad Introducir Indicador

- **Asignar Versión y RR.HH.:** Se asigna el desarrollo del indicador a una versión del producto dependiendo de la dificultad del mismo y del estado de la versión en cuanto al trabajo pendiente de realizar, se valora el esfuerzo de implementación del indicador y se asigna recursos humanos.

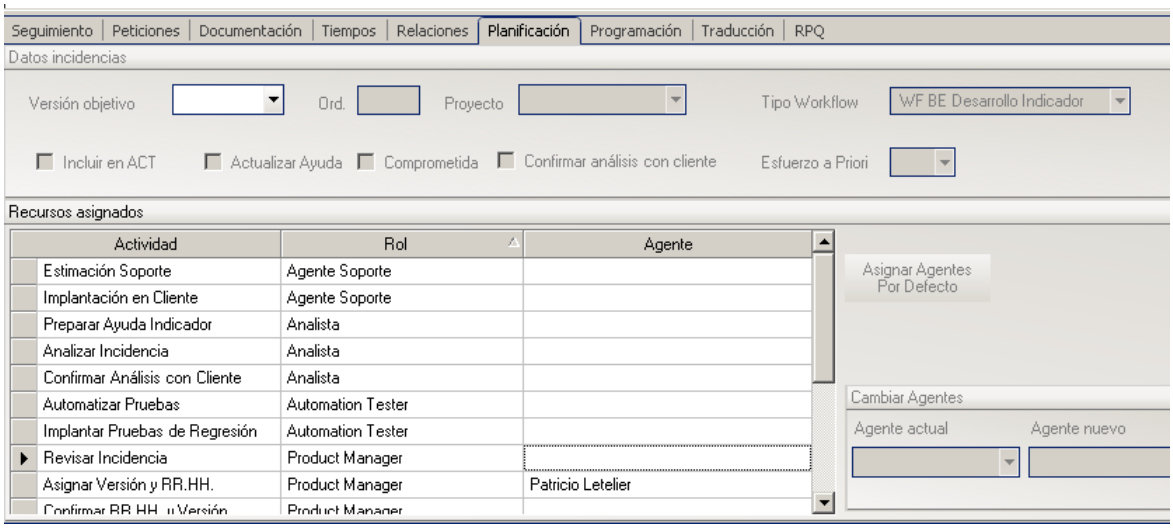


Figura 4.8. Gestor de Unidades de Trabajo Pestaña Planificación

- **Preparar Ayuda Indicador:** Se revisa el documento enviado por el cliente y si hay alguna cuestión que no esté clara, nos ponemos en contacto con el cliente para aclararla. A continuación, se elabora una pequeña demo de lo que será el producto final con datos de prueba y se prepara un documento de ayuda (Ver Anexo B) en el que se indica dónde hay que introducir la información en el programa para que el indicador muestre los datos correctamente.

  En lo que respecta a la construcción de la demo, se han de seguir los siguientes pasos:

  - Crear una base de datos de prueba con una estructura similar a la que tendrá la base de datos definitiva.
  - Introducir algunos datos en la base de datos creada anteriomente.
  - Crear el cubo de datos pertinente.
  - Construir el informe OLAP en Excel del indicador y definir un par de vistas por defecto.

  El documento de ayuda es de vital importancia, ya que en muchas ocasiones es posible introducir la información necesaria para calcular un indicador de varias formas, todas ellas válidas. Sin embargo, en el proceso de desarrollo del indicador, el programador se decanta por una de las formas para obtener los datos porque tratarlas todas sería muy complejo y por eso la necesidad de especificar en el documento de ayuda la forma correcta de introducir los datos. Dado esto, para que los datos que muestre el indicador sean correctos y se ajusten a la realidad, los usuarios deberán introducir la información tal y como se indica en dicho documento. De no ser así, los datos mostrados por el indicador no serán correctos y pueden llevarnos a errores en la toma de decisiones.

- **Analizar Incidencia:** Se elabora el documento de análisis de la incidencia (Ver Anexo C) en base a la plantilla del indicador enviada por el cliente y al resto de documentos (si los hay) en los que se especifiquen los requisitos del cliente con respecto al indicador. Dicho documento incluye la definición del hecho con sus dimensiones, medidas y KPIs, y una relación de las pruebas de aceptación que debe superar el indicador.

- **Confirmar con Cliente:** Se presenta al cliente la demo elaborada en actividades anteriores para confirmar que se ajusta al indicador solicitado por él mismo y se le explica detenidamente el documento de ayuda para despejar todas las dudas con respecto a la forma correcta de introducir la información en el programa. Si el cliente da el visto bueno a la demo, se sigue con el proceso normal de desarrollo del indicador. En otro caso, si la demo no se ajusta a lo que esperaba el cliente, se toma nota de las modificaciones que desea realizar el cliente, se rehace la demo y se modifica el documento de análisis para que se ajusten a los cambios solicitados. Una vez realizados los cambios, se vuelve a presentar la demo al cliente.

  Este proceso se repetirá hasta que el cliente dé por buena la demo para evitar hacer todo el proceso de desarrollo, con el coste de recursos y tiempo que eso conlleva, y que luego el resultado no sea el esperado por el cliente. De esta forma tenemos la certeza de que cuando el indicador esté finalizado, el resultado va a ser el deseado por el cliente.

- **Diseño Preliminar y Estimación:** Se realiza un estudio del documento de análisis cuestionando las pruebas de aceptación y su definición. A continuación se elabora

el documento de diseño (Ver Anexo D) y se realiza una estimación del tiempo necesario para la implementación del indicador. El documento de diseño también incluye información de cuál es el hecho con sus respectivas dimensiones con la diferencia de que en éste se indica de qué tablas de la base de datos operacional sacamos la información para cada una de las dimensiones y cómo se van a computar las medidas del indicador.

- **Estimación de Testeo:** Se hace una estimación conjunta, por parte del tester, del tiempo necesario para diseñar las pruebas y el tiempo necesario para aplicar dichas pruebas al indicador.
- **Estimación de Soporte:** Se hace una estimación, por parte del personal de soporte, del tiempo necesario para probar los indicadores cuando se encuentren implantados en el cliente.
- **Confirmar RR.HH. y Versión:** Se confirma la versión del producto en la que se va a desarrollar el indicador y los recursos humanos asociados a dicho desarrollo. En caso de que la incidencia se volviera urgente o si por el contrario su prioridad bajara, es en este momento cuando la moveríamos a otra versión y realizaríamos un ajuste de los recursos humanos asociados a la misma.
- **Diseño e Implementación:** Se crea el proceso ETL encargado de crear el almacén de datos, extraer los datos de la base de datos operacional, transformarlos y cargarlos en el almacén de datos. Tras esto, se crea el cubo correspondiente, las plantillas de Excel para poder visualizar los resultados del indicador y los informes estáticos solicitados por el cliente. Una vez finalizadas todas estas tareas, se suben las plantillas de Excel y los informes al portal de pruebas y se crea el dashboard, lo que supone el fin de la implementación.
- **Automatizar Pruebas:** Se crean los scripts encargados de interactuar con el programa y con las plantillas de Excel y se introducen datos que pueden llevar a errores para comprobar que el indicador funciona correctamente. A continuación, se lanzan las pruebas de forma automática y se compara si el resultado obtenido es el esperado utilizando las imágenes de Excel generadas como resultado de la prueba. Si la prueba es superada con éxito, se pasa a la implantación pero si no se ha superado, el tester devuelve el indicador al programador, indicando dónde se ha producido el fallo, para que éste resuelva el problema. Este proceso se repetirá hasta que el indicador pase todas las pruebas.
- **Implantación en Cliente:** Se suben los paquetes implementados al servidor del cliente y se crean programaciones de lanzamiento para que se lancen diariamente. Seguidamente, se suben las plantillas de Excel, los informes y el dashboard al portal del cliente y se configuran los permisos de acceso a estos elementos según las indicaciones del cliente.

  En lo que respecta al lanzamiento de paquetes, disponemos de un servicio que se encarga de esta tarea. Por tanto, si el cliente tiene contratado este servicio, en lugar de crear las programaciones, se configura el servicio para que ejecute los paquetes, indicándole la ubicación de los mismos, el nombre de las bases de datos, etc.
- **Implantar Pruebas de Regresión:** Consiste en cargar en el proyecto los scripts de automatización de las pruebas para que puedan ser ejecutadas en cualquier momento. Concretamente estas pruebas se realizan cada vez que sale una nueva versión del programa del que se extrae la información para los indicadores o cuando se realiza una modificación del indicador para comprobar que, pese a los

cambios introducidos en el programa base o en el indicador, el indicador sigue funcionando correctamente. Concretamente, se lanzan todas las pruebas y si hay alguna que no es superada, el tester informa del error al programador y éste lo soluciona modificando la implementación del indicador para que se ajuste a los cambios introducidos en el programa.

De esta forma se garantiza el correcto funcionamiento de los indicadores en todo momento, a pesar de que se realicen actualizaciones del programa.

Una vez finalizada la actividad de **"Implantación en Cliente"**, el proceso de desarrollo del indicador se da por concluido. No obstante, en muchas ocasiones, tras un período de tiempo de uso, el cliente solicita modificaciones del indicador. Dichas modificaciones no están contempladas en el WF de desarrollo visto anteriormente y por lo tanto, es necesario disponer de un nuevo WF (Figura 4.9.).

Las modificaciones habituales solicitadas por el cliente que suelen llevarse a cabo consisten en añadir o modificar las dimensiones o medidas del indicador, cambiar la forma de calcular cierta medida, añadir medidas al cubo, modificar los KPIs, etc. Sin embargo, en otras ocasiones, las modificaciones se realizan porque el programa del que se extraen los datos sufre cambios en la base de datos (eliminación de campos utilizados por el proceso ETL) o bien cambia la forma de introducir algún dato y por tanto, el proceso ETL debe atacar a otras tablas para que el indicador funcione correctamente.

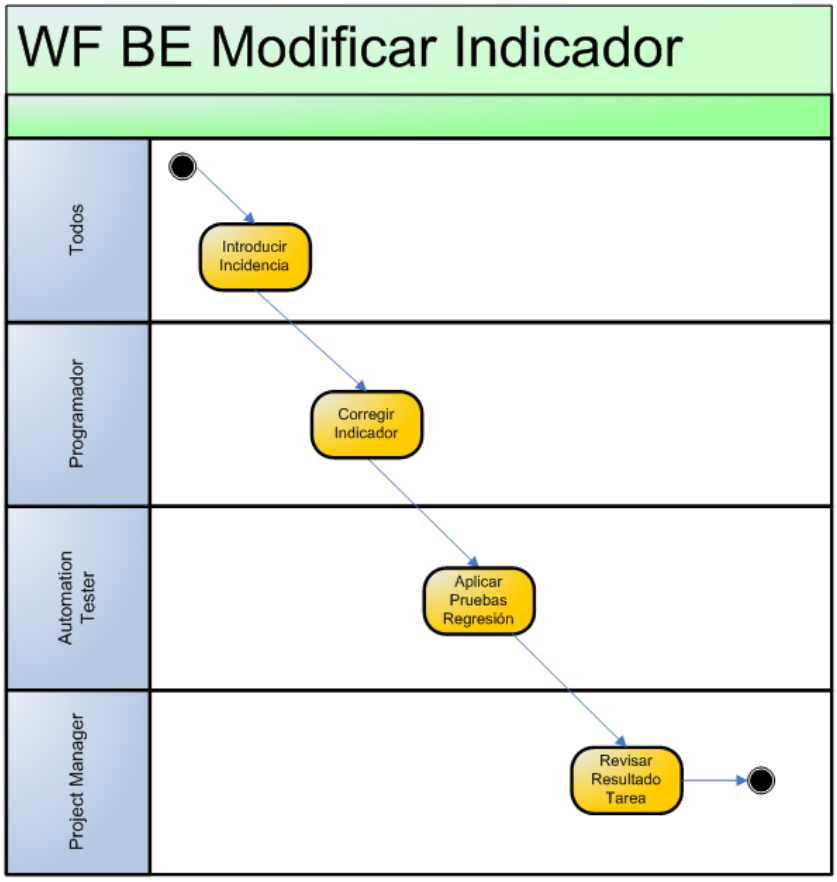


Figura 4.9. WF Modificar Indicador

Dicho WF se compone de 4 actividades:

- **Introducir Incidencia:** Es la actividad de inicio del WF y en ella se indica el indicador a modificar y los cambios a realizar.

- **Corregir Indicador:** Se modifican los documentos generados durante el desarrollo del indicador para que reflejen los cambios que se van a introducir en el indicador y se modifica el proceso ETL, los cubos y los informes en la medida que sea necesario.
- **Aplicar Pruebas de Regresión:** Se ejecutan las pruebas anteriores para ver que el indicador las pasa con éxito y en caso de que sea necesario añadir nuevas pruebas para verificar el correcto funcionamiento del indicador tras los cambios realizados, se crean nuevos scripts y se lanzan nuevamente las pruebas. Si las pruebas son superadas con éxito, se pasa a revisar la tarea pero si no se han superado, el tester devuelve el indicador al programador indicando dónde está el fallo para que éste lo resuelva. Este proceso se repetirá hasta que el indicador pase todas las pruebas.
- **Revisar Resultado Tarea:** Se comprueba que los cambios han sido realizados de forma satisfactoria y que se han pasado con éxito todas las pruebas.

Tras finalizar la última actividad, el proceso de modificación del indicador se da por concluido.

Capítulo 5. Marco tecnológico de una solución BI

En este capítulo se presenta el marco tecnológico en el que ha sido desarrollada la solución Business Intelligence objeto de este proyecto y se analiza cada una de las tecnologías utilizadas. Para no hacerlo muy extenso, en los apartados correspondientes a cada una de las tecnologías se hace referencia a un anexo que contiene un breve tutorial para entender mejor su funcionamiento. Dado que ofrecen multitud de funcionalidades, en los tutoriales sólo se va a describir la funcionalidad usada a la hora de desarrollar los indicadores de este proyecto.

El material presentado en este capítulo ha sido extraído de [10], [11] y [14].

5.1. Componentes del marco tecnológico

El marco tecnológico propuesto está formado por los siguientes componentes: **TUNE-UP Process Tool**, como herramienta para la planificación y seguimiento del proceso de desarrollo, **Microsoft Business Intelligence (BI)**, como suite de desarrollo de Inteligencia de Negocios, e **IBM Rational Functional Tester (RFT)**, como herramienta para el desarrollo de pruebas automatizadas y de regresión.

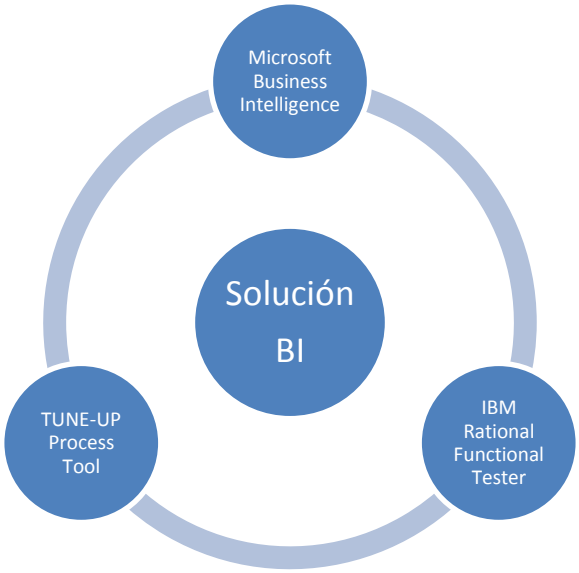


Figura 5.1. Componentes del marco tecnológico propuesto

Dado que TUNE-UP Process Tool fue descrita brevemente en el capítulo anterior, sólo se describirán los componentes de Microsoft BI e IBM RFT.

5.2. Microsoft Business Intelligence

Microsoft BI ofrece una suite completa de productos integrados, proporcionando acceso continuo a aplicaciones de amplia difusión e informes, dando cobertura a todos los aspectos del proceso de toma de decisiones. Basado en la plataforma de Inteligencia de Negocio de **Microsoft SQL Server** y con una difusión entre el público a través de **Microsoft Office**, este enfoque completo e integrado permite crear y poner en marcha aplicaciones de inteligencia de negocio.

Figura 5.2. Solución de Business Intelligence de Microsoft

5.2.1. Microsoft SQL Server

Microsoft utiliza como plataforma de Inteligencia de Negocios la herramienta conocida popularmente como SQL Server. Este software ofrece un entorno de desarrollo integrado, **Business Intelligence Development Studio**[13], para desarrollar e implementar proyectos de **Analysis Services (SSAS)**, **Integration Services (SSIS)** y **Reporting Services (SSRS)**. Cada tipo de proyecto proporciona plantillas para crear los objetos necesarios para las soluciones de Business Intelligence y ofrece varios diseñadores, herramientas y asistentes para trabajar con los objetos.

Todos estos proyectos son administrados mediante **SQL Server Management Studio**, un entorno integrado que permite el acceso a los componentes de SQL Server, así como a su configuración y administración. Dicho entorno cuenta con un motor de planificaciones y alertas llamado **SQL Server Agent(SSA).**

A continuación se describe brevemente cada una de las tecnologías de SQL Server utilizadas.

SQL Server Integration Services

SSIS es una plataforma para la creación de soluciones empresariales de transformación e integración de datos a partir de tareas y transformaciones fáciles de realizar para los no desarrolladores, empleando un modelo de objetos potente que soporta la creación de tareas y transformaciones de datos personalizadas.

Más información en el apéndice (Ver Anexo E.), donde se describe cómo construir paquetes ETL con SSIS.

SQL Server Analysis Services

SSAS es un servicio que ofrece funciones de procesamiento analítico en línea (OLAP) y minería de datos para aplicaciones de Business Intelligence. SSAS permite gestionar y

[13] Microsoft SQL Server consolida la administración de servidores y la creación de objetos comerciales en dos entornos integrados: SQL Server Management Studio y Business Intelligence Development Studio.

diseñar estructuras multidimensionales que contienen datos agregados desde otros orígenes de datos, como bases de datos relacionales.

Más información en el apéndice (Ver Anexo F), donde se describe cómo construir cubos OLAP con SSAS.

SQL Server Agent

SSA es un motor de planificaciones y alertas que permite crear trabajos flexibles con múltiples pasos y dependencias entre los distintos pasos de trabajo y configurar alertas que realicen acciones automatizadas o bien envíen mensajes a registros de errores.

Más información en el apéndice (Ver Anexo G), donde se describe cómo construir trabajos con el Agente SQL Server.

5.2.2. Microsoft Office

Dos de las soluciones de Microsoft Office más utilizadas como herramientas de difusión entre el publico de la Inteligencia de Negocios son: Microsft Office Excel y Microsoft Office SharePoint Server.

Microsoft Office Excel

Microsoft Office Excel ofrece un entorno para los usuarios de negocios, incorporando capacidades OLAP que permiten al usuario conectarse a cubos de Analysis Services y explotar su información mediante tablas y/o gráficas dinámicas.

Adicionalmente, previa la instalación de un add-in, también es posible realizar Minería de Datos.

Más información en el apéndice (Ver Anexo H), donde se describe cómo construir informes analíticos en Microsoft Office Excel.

Microsoft Office SharePoint Server

Microsoft Office SharePoint Server (MOSS) es una plataforma web que ayuda a las organizaciones a ofrecer capacidades de inteligencia empresarial a todos los empleados para compartir, controlar y reutilizar información empresarial con el fin de tomar mejores decisiones en la empresa. Las características de BI que ofrece MOSS son las siguientes:

- Acceso web y mediante programación a las hojas de cálculo de Microsoft Office Excel a través de Excel Services. Los Servicios de Excel extienden las capacidades de Microsoft Office Excel y permiten compartir ampliamente hojas de cálculo, con una mejor administración y seguridad y los modelos reutilizables de hojas de cálculo, utilizando un servicio de cálculo escalable basado en el servidor y una interfaz de usuario interactiva basada en Web.
- Desarrollo de cuadros de mandos (*dashboards*) basados en web que pueden incorporar indicadores clave de rendimiento enriquecidos y enlazados a datos, elementos web y libros que pueden alojarse dentro de la nueva plantilla del sitio del Centro de informes.

  Más información en el apéndice (Ver Anexo I), donde se describe cómo construir dashboards en Microsoft Office SharePoint Server.

5.3. IBM Rational Functional Tester

IBM RFT es una herramienta avanzada de testing funcional y de regresión automatizada, creada para los testers y los desarrolladores de interfaces de usuario que necesitan de un mayor control para probar sus aplicaciones Java, Microsoft Visual Studio .NET y Web.

RFT graba las interacciones del usuario con las aplicaciones, creando secuencias de lineas de codigo (*test scripts)* que al ser ejecutados reproducen dichas acciones. Durante la grabación, el usuario puede insertar puntos de verificación que extraen datos o propiedades específicas de la aplicación bajo prueba. Durante la reproducción, dichos puntos de verificación se usan para comparar la información grabada con la información que tiene la aplicación bajo prueba, garantizando de esta forma su consistencia.

Al desarrollar las actividades de grabación de pruebas, los testers tienen la opción de agregar código personalizado al test script. Esto permite realizar una cantidad ilimitada de tareas, como la manipulación de datos y configuraciones de entorno que a menudo son necesarias para asegurar que las maquinas están preparadas para correr el test. Cuando se está ejecutando el test, RFT genera un registro que contiene los resultados de las comparaciones de los puntos de verificación.

Las características más importantes de RFT son las siguientes:

- Es una herramienta de testing automatizada que sirve para hacer pruebas funcionales y de regresión de aplicaciones Java, Microsoft Visual Studio .NET y basadas en web.
- Ofrece a los testers avanzados una selección de idiomas de script y un editor de solidez, Java en Eclipse o Microsoft Visual Basic .NET en Visual Studio .NET, para la verificación del montaje y la personalización.
- Proporciona a los testers con poca experiencia funciones automatizadas para actividades como, por ejemplo, la generación de pruebas y el testing dirigido por datos.
- Incluye la tecnología ScriptAssure y funciones de coincidencia de patrón para mejorar la capacidad de recuperación del script de verificación dado los frecuentes cambios que se producen en la interfaz del usuario de aplicaciones.
- Incorpora soporte para el control de la versión para permitir un desarrollo paralelo de los scripts de verificación y el uso simultáneo por parte de equipos distribuidos por el mundo.
- Permite la realización de pruebas de aplicaciones creadas con VS.NET Winforms, J2SE/J2EE, HTML/DHTML, XML, JavaScript y applets de Java e incluye soporte exclusivo para la biblioteca SWT de Java asociada con el shell de Eclipse.
- Soporta el testing de aplicaciones 3270 (zSeries) y 5250 (iSeries) que utilizan las aplicaciones basadas en IBM Rational Functional Tester Extension for Terminal.

Más información en el apéndice (Ver Anexo J), donde se describe cómo desarrollar pruebas automatizadas en IBM Rational Functional Tester.

Capítulo 6. Caso de estudio

El siguiente capítulo incluye una descripción del caso de estudio ResiPlus BI, enmarcado en ofrecer servicios de Inteligencia de Negocios al sector socio-sanitario. El objetivo es ilustrar cómo se desarrollan e implentan los indicadores de negocio en el marco tecnológico propuesto en el capítulo anterior pero contextualizado en ResiPlus BI. Para ello se describen un conjunto de consideraciones previas en el entorno de Resiplus BI y se ejemplifica de forma práctica el desarrollo de varios indicadores realizando su análisis, diseño, implementación y pruebas automatizadas.

6.1. ¿Qué és ResiPlus BI?

ResiPlus es un programa informático que permite a las residencias, centros de día y de discapacitados gestionar sus centros.

ResiPlus ofrece una amplia gama de servicios a través de su plataforma de servicios ResiPlus PS (*ResiPlus Platform Services*). Entre los servicios contratables por los centros, podemos encontrar el servicio ResiPlus BI (ResiPlus *Business Intelligence*) encargado de ofrecer al cliente un servicio integral de Inteligencia de Negocio. Los Servicios BI entran en la esencia del negocio del cliente. Así, estos servicios están orientados al personal involucrado en la toma de decisiones de mayor nivel. El cliente podrá contar con informes OLAP, cuadros de mandos, análisis y predicciones de datos, etc. todo esto basado en tecnología de Almacenes de Datos y Minería de Datos.

6.2. Funcionamiento de ResiPlus BI

Cuando los centros contratan el servicio BI y a medida que van solicitando reports o indicadores[14], de forma transparente se desarrolla incrementalmente un Data Warehouse que les permite contar con información preparada para el análisis. De esta manera se logra una independencia total del sistema transaccional de ResiPlus, dando lugar a un nuevo almacén con una estructura mucha más simple y cuya información puede ser explotada rápidamente por los centros.

Mediante el servicio BI los centros acceden a los reports que van contratando y les permite analizar la información dentro de una amplia gama de perspectivas. Los centros también disponen de la opción de contratar dashboards para tener una vista integrada/unificada de los reports con el fin de ilustrar el estado y rendimiento en un ámbito determinado (residente, empleado, área, unidad, departamento, centro o grupo).

6.3. Consideraciones previas al desarrollo de indicadores

Antes de comenzar con la implementación de un indicador, hay que tener en cuenta una serie de factores existentes cuando el desarrollo se realiza en un entorno multiempresa (clientes con más de un centro o residencia).

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 una instalación de ResiPlus

[14] Utilizamos los términos report o indicador indistintamente, ya que el cliente a veces habla de indicadores de negocio y otras de reports como si fueran la misma cosa. Esto es así porque para obtener un report, previamente se necesita la definición de un indicador.

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 pero dado que estamos 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 porque los identificadores no pueden repetirse y ya no podremos 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 guión 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 siguientes figuras, podemos ver la variación introducida y el efecto que este produce:

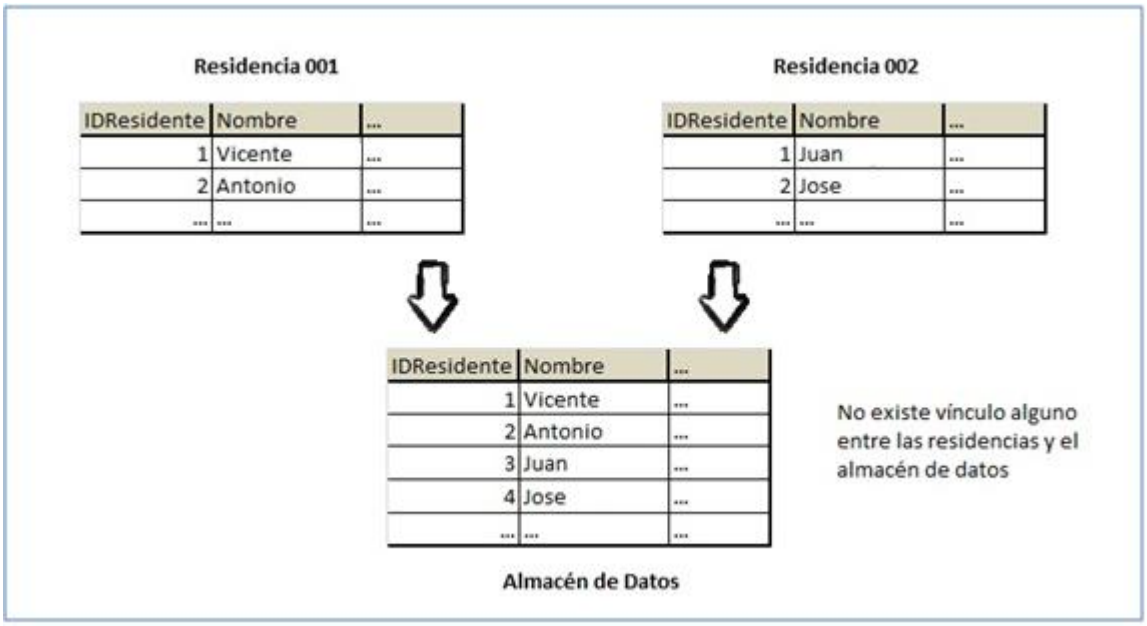


Figura 6.1. Ejemplo de la tabla Residentes sin código unificado

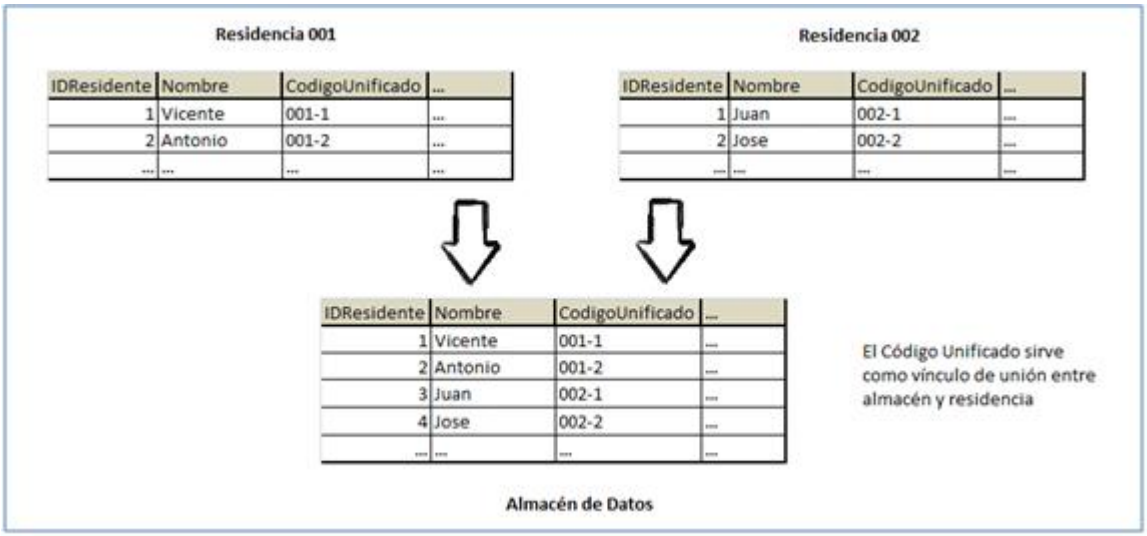


Figura 6.2. Ejemplo de la tabla Residentes con código unificado

El segundo factor a tener en cuenta es a la hora de rellenar tablas importantes de la base de datos de ResiPlus, queriendo a posteriori crear un indicador con los datos de dichas tablas.

Por ejemplo, imagínese dos residencias para un mismo cliente en las cuales se tienen un conjunto de caracteres de plaza distintos: 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 sólo cuatro: público, privado y las uniones resultantes de semipúblico y cuasi-publico y semiprivado y cuasi-privado.

Dada esta circunstancia, para que la información manejada por los indicadores 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.

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 cliente para el mantenimiento de los indicadores debe ocuparse de recopilar todo este conjunto de valores.

Una vez recopilados estos valores, requerirá de una reunión con los responsables correspondientes al objeto de unificar los criterios de todas las residencias del mismo grupo. Una vez hecho esto, y mediante ResiPlus BI, el encargado de mantenimiento de los indicadores deberá rellenar la tabla maestra de la dimensión 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 el siguiente ejemplo se observa como tres residencias con distintos caracteres de plaza logran unificarlo en tan sólo uno:

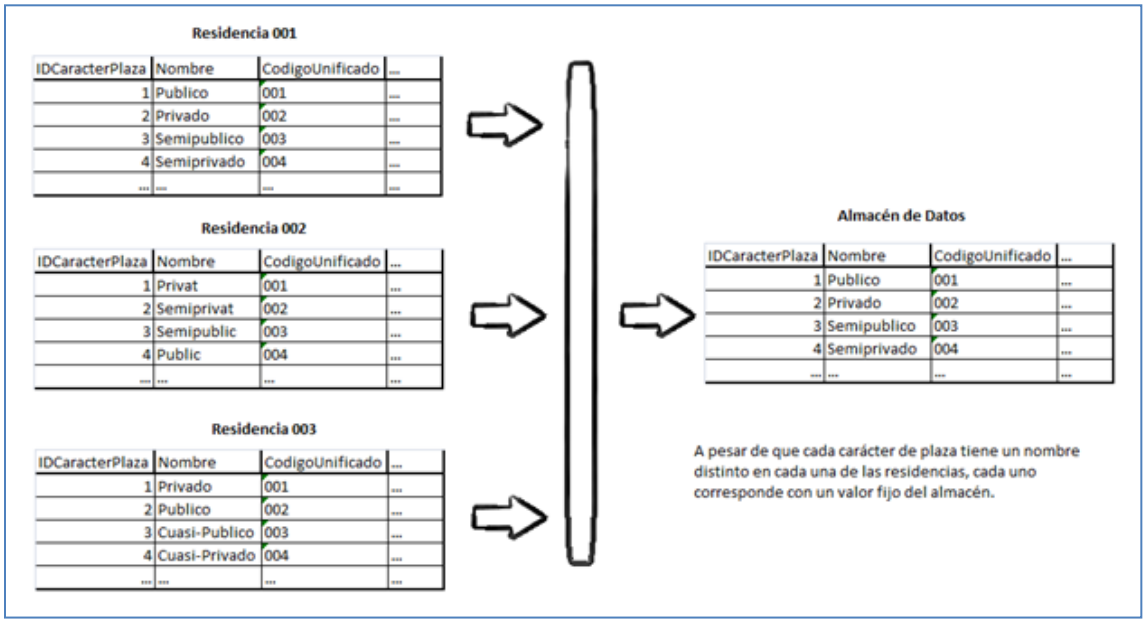


Figura 6.3. Ejemplo de la tabla Caracter Plaza con código unificado

Cabe destacar que la aplicación de este método será posible siempre que el compromiso entre residencias y administración central (donde esté instalado ResiPlus BI) sea fuerte, y haya una buena comunicación.

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 habitual en el entorno estudiado.

6.4. Introducción, análisis y diseño preliminar de indicadores

A continuación se describen una serie de indicadores de negocio relacionados con el programa Resiplus. Concretamente, se van a describir tres (Úlceras Curas, Diferencia Seguimientos Profesionales y Estancia Unidades) con el objetivo de observar las diferencias en el proceso de implementación dependiendo del tipo de indicador.

Generalmente, los posibles tipos de indicadores demandados son:

- **Indicadores que calculan totales en función de varias dimensiones.**

  En este grupo incluimos el indicador de Ulceras Curas y el de Visitas Familiares (utilizado en los tutoriales). La mayoría de los indicadores solicitados por los clientes son de este tipo.

- **Indicadores que calculan la diferencia de días transcurridos entre dos eventos** tales como seguimientos, escalas de valoración, etc.

  En este grupo incluimos el indicador de Diferencia Seguimientos Profesionales.

- **Indicadores que muestran una evolución.**

  En este grupo incluimos el indicador de Estancia Unidades. Estos indicadores son muy demandados, ya que con ellos el cliente está informado de lo que pasa diariamente en los centros (residentes en el hospital, úlceras activas, etc).

La forma de describir estos tres indicadores será adjuntando cada uno de los documentos originados en las actividades de introducción, preparación de ayuda, análisis y diseño, vistas en el proceso de desarrollo de indicadores del capítulo 4. Debido a que la mayoría se cumplimentan de manera análoga, únicamente se incluyen la totalidad de documentos de Úlceras Curas (indicador automatizado en el punto dedicado a las pruebas). Para el resto, se adjuntarán los documentos de diseño.

Plantilla de Indicadores de Úlceras Curas

INFORMACIÓN DEL INDICADOR

Hecho:	**Úlceras Curas**
Descripción:	**Este indicador muestra el número de curas de úlceras que se han realizado en una residencia.**
Finalidad:	**Se pretende conocer las personas a las que se les realizan un mayor número de curas para poder darles una mayor atención personal.**
Medidas:	**Sumatorio del número de curas y tamaño medio de las úlceras.**
Granularidad:	**Diaria**
KPI's:	**El tamaño medio de las úlceras para la Residencia 1 se espera que sea menor o igual a 3 cm. Los valores comprendidos entre 3 y 4 no son del todo malos y los valores a partir de 4 son malos (los mostraremos en color rojo).** **El tamaño medio de las úlceras para la Residencia 2 se espera que sea menor o igual a 1 cm. Los valores comprendidos entre 1 y 2 no son del todo malos y los valores a partir de 2 son malos (los mostraremos en color rojo).**
Valor por defecto**:**	**Para la Residencia 1 es 4.** **Para la Residencia 2 es 2.**

INSTRUCCIONES

1. **Hecho**. Nombre de la variable que se desea analizar. Ejemplos: Visitas familiares.

2. **Descripción**. Escuetamente describe el hecho. Ejemplo: Este indicador muestra el número de visitas por residente y por día.

3. **Finalidad**. Uso que se pretende dar al indicador. Ejemplo: Se pretende conocer las personas con menos visitas para poder darles una mayor atención personal.

4. **Medidas**. Operaciones a realizar con el hecho. Ejemplo: sumatorio de las visitas. Otros posibles valores podrían ser: Media, mínimo, máximo y moda.

5. **Granularidad Temporal**. Unidad mínima de detalle de la cual se puede obtener información. Normalmente se utilizara una granularidad diaria, aunque se pueden usar otros valores, como por ejemplo: semanal, mensual, trimestral y anual. Para nuestro Ejemplo utilizamos la diaria, lo que significa que obtendremos las visitas agrupadas por día.

6. **KPI's**. Campos a resaltar según su valor y la agrupación de estos. Ejemplo: Mostraremos en rojo aquellos campos que muestren menos de 4 visitas por mes y por residente.

7. **Dimensión.** Campos por los cuales queremos categorizar el hecho. Ejemplo: residentes, tipología, carácter plaza y tiempo. (No importa el orden de las dimensiones. Agregue cuantas desee)

1ª DIMENSIÓN

Descripción: Residente

Se mostrarán los datos del residente que padece la úlcera.

Atributos:

Nombre y apellidos, sexo y fecha de ingreso.

2ª DIMENSIÓN

Descripción: Centro

Se mostrará el centro al que pertenece cada residente.

Atributos:

Nombre del centro, ciudad, provincia, comunidad autónoma y país.

3ª DIMENSIÓN

Descripción: Tipología

Se mostrará la tipología de los residentes.

Atributos:

Tipologia.

4ª DIMENSIÓN

Descripción: Carácter Plaza

Se mostrará el carácter plaza de los residentes.

Atributos:

Carácter Plaza.

5ª DIMENSIÓN

Descripción: Tiempo

Se mostrará toda la información relativa a la fecha de la cura de la úlcera.

Atributos:

Fecha, día semana, semana, mes, trimestre, año y nombre completo de la fecha.

6ª DIMENSIÓN

Descripción: Úlcera

Se mostrará toda la información relativa a las úlceras.

Atributos:

Descripción, tipo, procedencia (donde le apareció al residente) y tamaño de la úlcera.

7ª DIMENSIÓN

Descripción: Cura

Se mostrará toda la información relativa a las curas.

Atributos:

Úlcera está curada o no, localización y estadio/gravedad de la cura.

Documento de Ayuda de Úlceras Curas

Indicador de Úlceras Curas

1. Descripción

Este indicador muestra las curas de las úlceras que se han realizado y las agrupa por:

Residente, **tipología**, **carácter plaza, centro, procedencia, localización, tipo, estadio, curada** y **tiempo**.

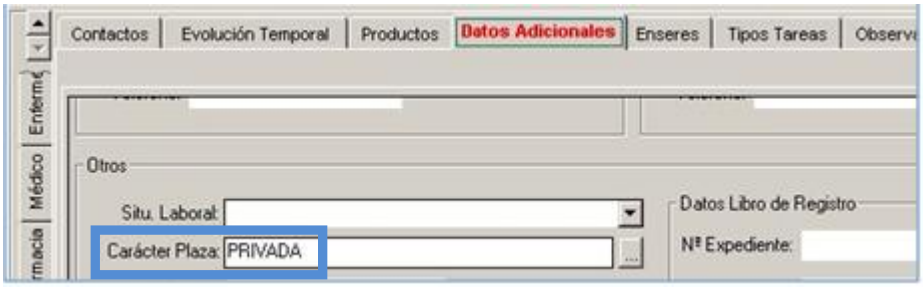
La **Granularidad** de este indicador es diaria. Esto significa que se sumara el número de curas que ha recibido un residente por día, lo cual conlleva que no se puede obtener información sobre el número de curas por una unidad de tiempo menor al día.

2. Dimensión

Para el correcto funcionamiento de este indicador debemos cumplimentar correctamente los siguientes campos:

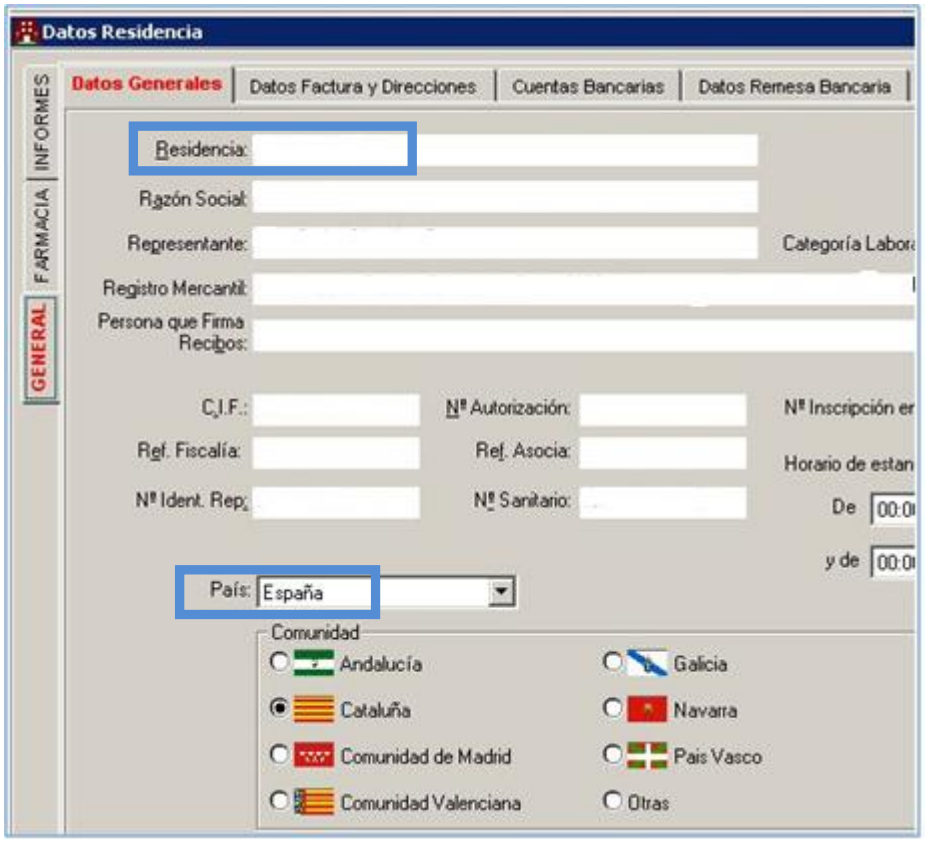
Tipología

Carácter Plaza

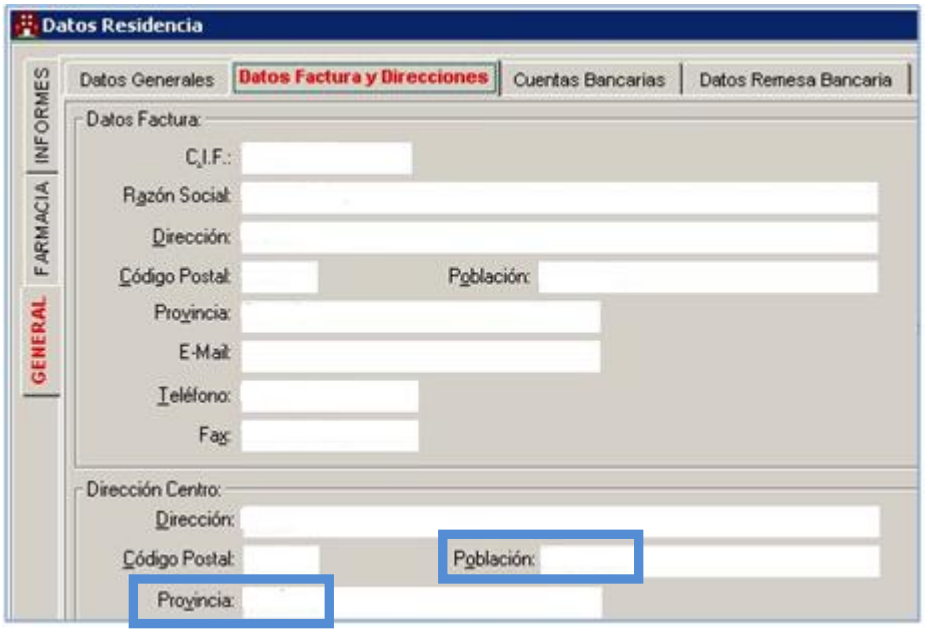


Los siguientes campos a rellenar se encuentran en la pestaña de Datos Residencia.

Residencia y País

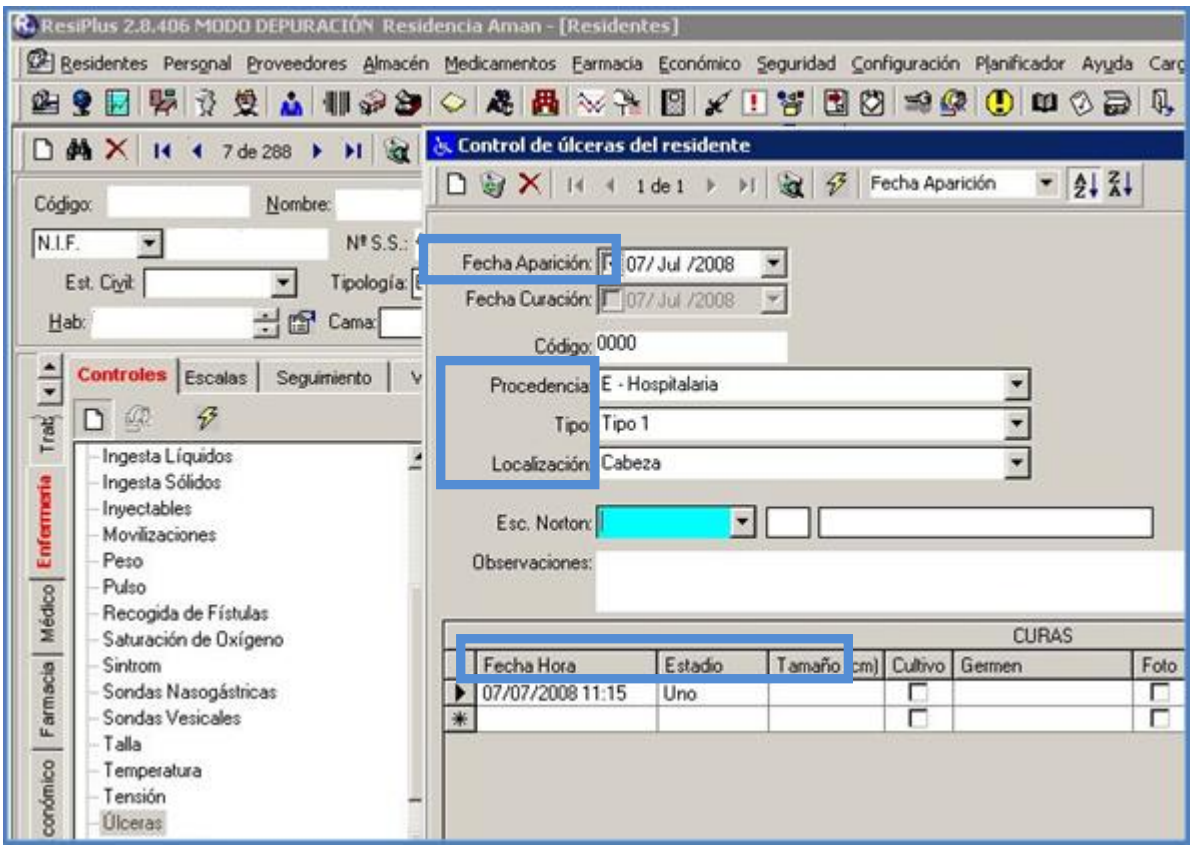


Población y provincia.



3. Hecho

Tenemos que introducir toda la información relativa a las úlceras y sus correspondientes curas: origen de la úlcera, tipo, localización, estadio, si está curada o no, tamaño y fecha de cada una de las curas.

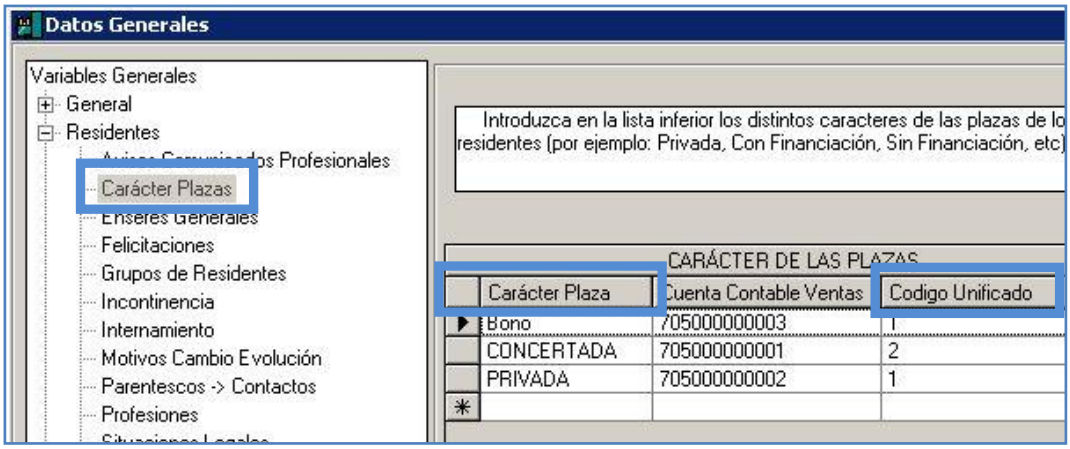


4. Código unificado

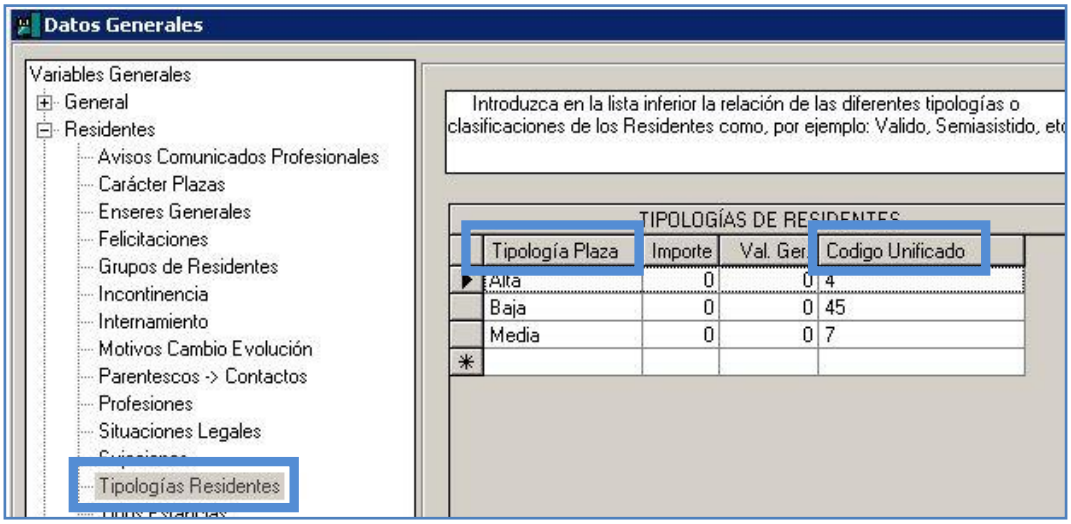
Además de los campos que muestran las capturas, debemos de rellenar el código unificado para cada una de las tablas maestras de ResiPlus y que este código coincida con el código unificado de las tablas maestras del Almacén de datos.

Para acceder a estas tablas maestras deberemos ir a Configuración→Datos Generales.

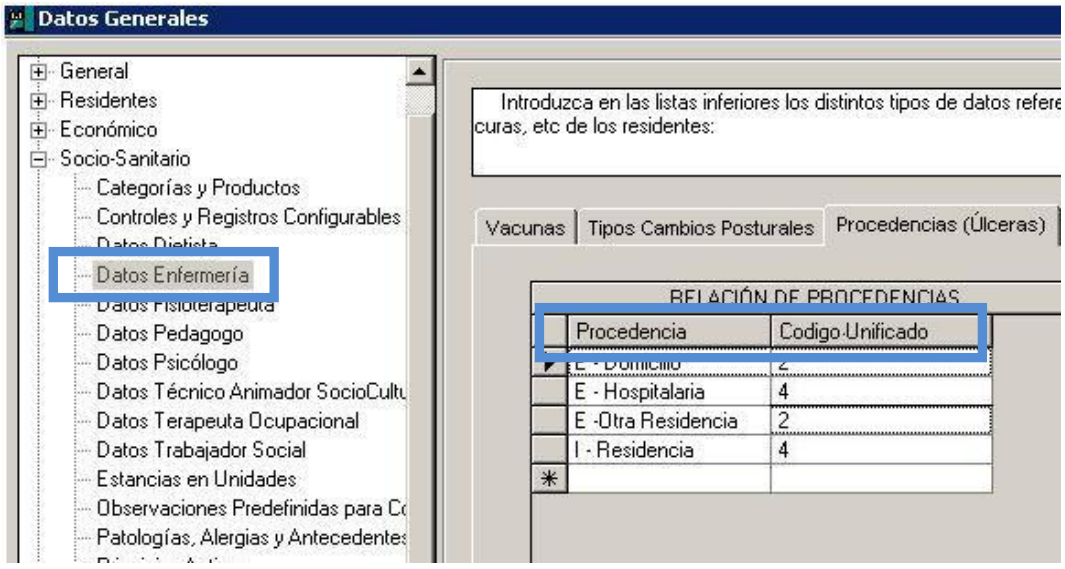
Una vez abierta la ventana Datos Generales, para cumplimentar el código unificado de Carácter Plaza, deberemos ir a Residentes/Carácter Plazas.



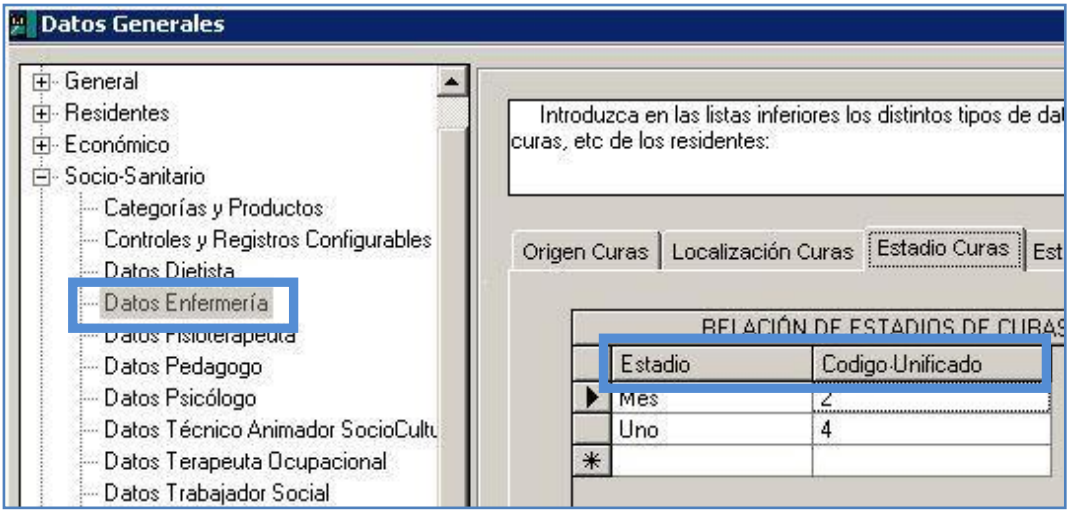
Lo mismo debemos hacer para la tipología, pero en este caso seleccionando Residentes/Tipologías Residentes.



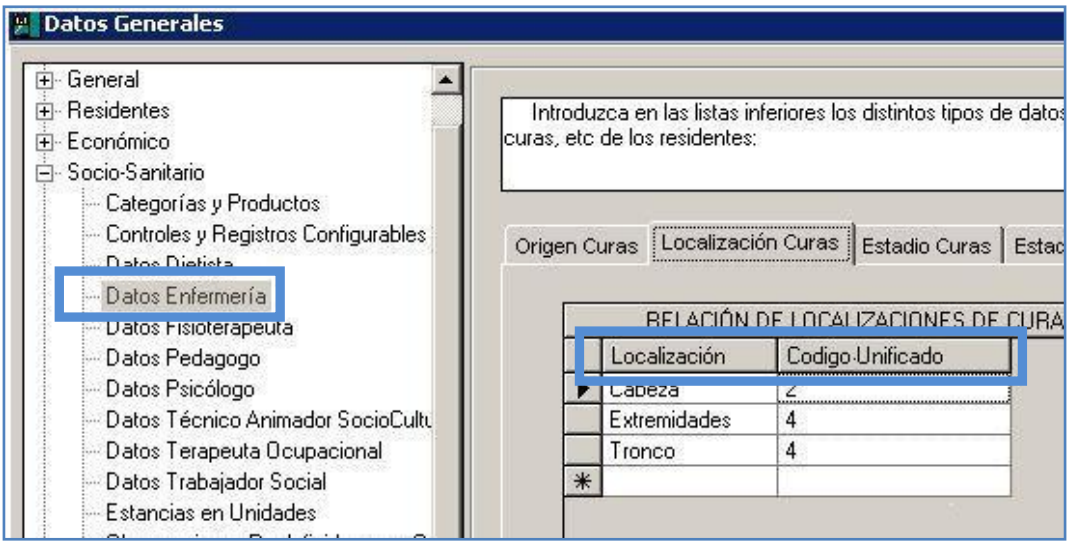
En el caso del origen de las úlceras, debemos seleccionar Socio-Sanitario/Datos Enfermería.



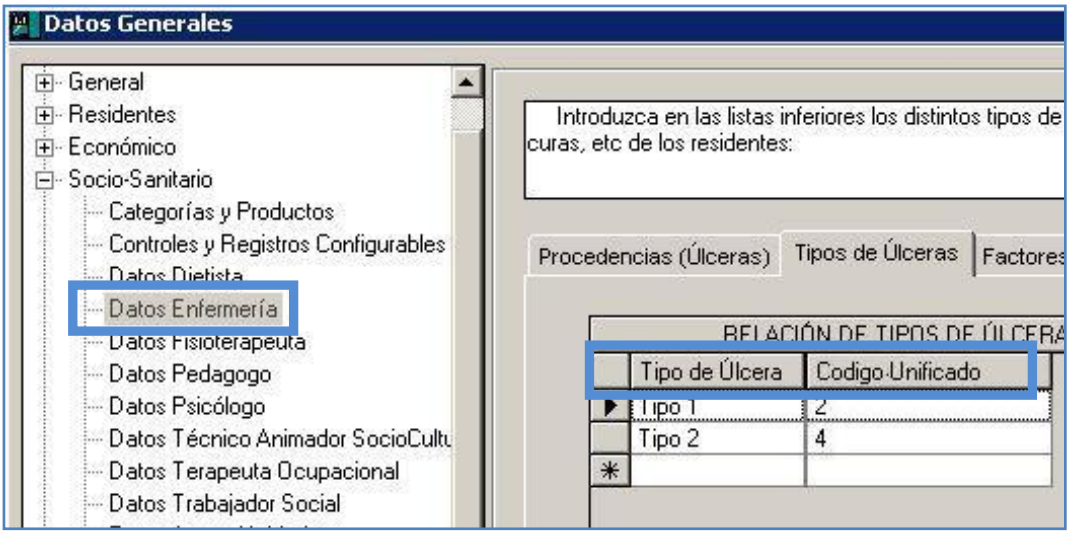
Para cumplimentar los códigos unificados de los estadios de las curas debemos seleccionar la pestaña Estadio Curas.



A continuación, vamos a la pestaña de Localización Curas para introducir los códigos unificados pertinentes.



Para finalizar nos desplazamos a la pestaña Tipos de Úlceras e introducimos los códigos unificados de los mismos.



5. Mantenimiento tablas maestras

Para que esos valores unificados puedan aparecer en el indicador tenemos que completar las tablas maestras del almacén de datos, tanto de tipologías como de carácter plaza. Esto se realiza desde el ResiPlusBI, como detallamos a continuación:

Accedemos al menú principal del BI y seleccionamos mantenimiento tablas maestras.

A continuación seleccionamos la tabla maestra a modificar. Debemos hacer que los códigos unificados de la base de datos de ResiPlus coincidan con el código unificado de nuestro almacén de datos. Los datos donde el código unificado no coincida aparecerán en el indicador como no definidos, mientras que aquellos en los que coincida aparecerán con la leyenda que indiquemos en la maestra de nuestro almacén.

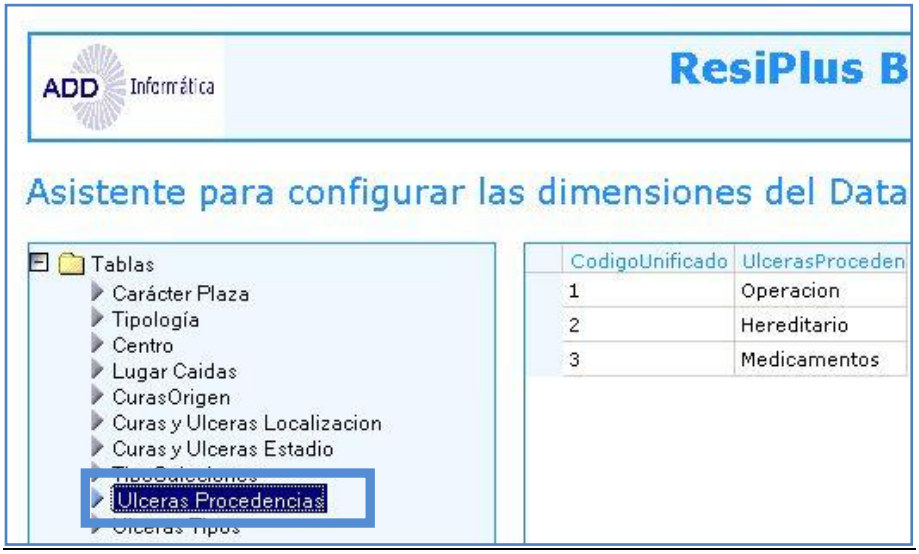
Maestra de Carácter Plaza

Maestra de Tipologías

Maestra de Curas Estadio

Maestra de Curas Localización

Maestra de Úlceras Procedencias



Maestra de Úlceras Tipos

Documento de Análisis de Úlceras Curas

	Departamento: Desarrollo	**Tipo:** Documento Análisis	ID5263
Programa/Proyecto: Indicadores tesis		**Analista:** Juanvi	**Fecha creación:** 01/05/2010

1. DESCRIPCIÓN

Creación del indicador de Úlceras Curas para el proyecto de indicadores de la tesis.

2. MOTIVACIÓN

El cliente pretende conocer a qué tipo de residentes se les realizan un mayor número de curas para poder darles una mayor atención personal.

3. DEFINICIÓN

3.1. Hechos

Úlceras Curas

Descripción: Sumatorio de las curas de úlceras realizadas a los residentes y el tamaño medio de las úlceras agrupadas por todas las dimensiones que se indican en la sección 3.3.

Granularidad Temporal: Diaria.

Frecuencia de Recolección de datos: 1 día

Frecuencia de refresco del cubo: 1 día

3.2. KPI

Descripción: Nos indicará gráficamente si el tamaño medio de las úlceras de los residentes es mayor que un valor dado.

Valor por defecto: Para la Residencia 1 es 4 y para la Residencia 2 es 2.

3.3. Medidas

*/*En este apartado es donde indicamos el resto de medidas que aparecen en la tabla de hechos sin contar lo que medimos en el hecho propiamente (Nº visitas que reciben los residentes cada día)*/*

Tamaño medio: Indica el tamaño medio que tienen la úlceras a las que se le realizan las curas.

3.4. Dimensiones

Dimensiones	Histórico
Centro	No (*)
Tiempo	No(*)
Residente	No(*)
Tipología	Sí
Carácter Plaza	Sí
Úlcera	No(*)
Tipo Úlcera	No(*)
Procedencia Úlcera	No(*)
Úlcera Curada	No(*)
Localización Cura	No(*)
Estadio Cura	No(*)

(*) Siempre se mostrarán los valores actuales, es decir, los valores que posean esas dimensiones en la base de datos de ResiPlus.

4. PRUEBAS DE ACEPTACIÓN

5263.1 Comprobar que si un residente recibe más de una cura en la misma úlcera en un día, aparece un registro por cada cura realizada.

5263.2 Comprobar que si hay alguna dimensión sin valor, siempre que no sea centro, residente ni tiempo, no se produce ningún error.

5263.3 Comprobar que si alguna dimensión no posee código unificado o no aparece en la tabla maestra del almacén, aparecerá como no definida.

Documento de Diseño de Úlceras Curas

	Departamento: Desarrollo	**Tipo:** Documento Diseño	ID5263
Programa/Proyecto: Indicadores tesis		**Analista:** Juanvi	**Fecha creación:** 01/05/2010

1. DESCRIPCIÓN

Creación del indicador de Úlceras Curas para el proyecto de indicadores de la tesis.

2. DISEÑO

Este indicador se carga por medio de un paquete ETL.

Dimensiones del indicador:

- **Tiempo** (Utilizaremos el campo FechaHora de la tabla UlcerasCuras)

  Atributos: IDTiempo, Fecha, Año, MesID, Mes, Trimestre, Semana, NumDiaSemana, DiaSemana, NombreCompleto.

- **Centro** (Utilizaremos la tabla DatosResidencia)

  Atributos: IDCentro, CodigoUnificado, Nombre, Ciudad, Provincia, Comunidad, Pais.

- **Carácter Plaza** (Utilizaremos el campo IDResidente de la tabla UlcerasCuras para posteriormente atacar a la tabla ResiHisCaracterPlaza con ese id y la fecha que indica el campo FechaHora y obtener el carácter plaza del residente en esa fecha)

  Atributos: IDCaracterPlaza, CodigoUnificado, CaracterPlaza.

- **Tipología** (Utilizaremos el campo IDResidente de la tabla UlcerasCuras para posteriormente atacar a la tabla ResiHisTipologiasPlaza con ese id y la fecha que indica el campo FechaHora y obtener la tipología del residente en esa fecha)

  Atributos: IDTipologia, CodigoUnificado, Tipologia.

- **Residente** (Utilizaremos el campo IDResidente de la tabla UlcerasCuras para luego obtener el id del almacén correspondiente para ese residente)

  Atributos: IDResidente, CodigoUnificado, Nombre, Sexo, FechaIngreso.

- **Úlcera** (Utilizaremos el campo IDUlcera de la tabla UlcerasCuras para enlazar con la tabla Ulceras)

  Atributos: IDUlceras, CodigoUnificado, Nombre.

- **Tipo Úlcera** (Utilizaremos el campo IDUlceraTipon de la tabla Ulceras para enlazar con la tabla UlcerasTipos)

  Atributos: IDUlcerasTipos, CodigoUnificado, UlcerasTipos.

- **Procedencia Úlcera** (Utilizaremos el campo IDUlceraProcedencia de la tabla Ulceras para enlazar con la tabla UlcerasProcedencias)

  Atributos: IDUlcerasProcedencias, CodigoUnificado, UlcerasProcedencias.

- **Úlcera Curada** (Utilizaremos el campo Curado de la tabla Ulceras)

  Atributos: IDUlceraCurada, UlceraCurada.

  En este caso no precisamos de tener un código unificado porque todas las residencias tienen los mismos valores para ese campo y por tanto podemos rellenar la dimensión con los valores utilizados por el programa.

- **Localización Cura** (Utilizaremos el campo IDLocalizacion de la tabla Ulceras para enlazar con la tabla CurasLocalizacion)

  Atributos: IDCurasLocalizacion, CodigoUnificado, CurasLocalizacion.

- **Estadio Cura** (Utilizaremos el campo IDEstadio de la tabla UlcerasCuras para enlazar con la tabla CurasEstadio)

  Atributos: IDCurasEstadio, CodigoUnificado, CurasEstadio.

Hechos del indicador:

- Úlceras Curas: Cada registro de la tabla de hechos indica la cura de una úlcera realizada a un residente (tabla UlcerasCuras). Por tanto el hecho estará formado por el conjunto de dimensiones citadas anteriormente: IDUlceras, IDTiempo, IDResidente, IDCaracterPlaza, IDTipologia, IDCentro, IDCurasLocalizacion, IDUlcerasTipos, IDUlcerasProcedencias, IDCurasEstadio, IDUlceraCurada y el campo TamañoUlcera. Este campo será considerado como medida del hecho porque será utilizado para obtener posteriormente el tamaño medio de una úlcera (campo Tamaño de la tabla UlceraCuras).

La **granularidad** o el nivel de detalle mínimo que podrá ser consultado será diario.

Miembros Calculados: Este indicador posee los siguientes miembros calculados:

- Tamaño Total: Es la suma de todos los tamaños de las úlceras.
- Número Curas: Es el número de registros de la tabla de hechos y se corresponde con el número total de curas realizadas.
- Tamaño medio: Es el tamaño medio de las úlceras que padecen nuestros residentes. Se define como un cociente Tamaño/Número Curas.

KPIs: Este indicador posee un KPI que se muestra en rojo para todos aquellos registros cuyo tamaño de úlcera sea mayor a 4 y 2 cm para la Residencia 1 y 2 respectivamente.

Cubo: Crearemos un cubo en modo Molap que se procesará cada 24 horas, inmediatamente después de la carga de los datos en nuestro almacén de datos. Poseerá las dimensiones enunciadas en el apartado anterior.

3. NOTAS DE DISEÑO

En lo que respecta a la técnica de diseño utilizada, usaremos el esquema en copo de nieve.

Documento de Diseño de Seguimientos Profesionales

	Departamento: Desarrollo	Tipo: Documento Diseño	ID5264
Programa/Proyecto: Indicadores tesis		Analista: Juanvi	Fecha creación: 01/05/2010

1. DESCRIPCIÓN

Creación del indicador Diferencia Seguimientos Profesionales para el proyecto de indicadores de la tesis.

2. DISEÑO

Este indicador se carga por medio de un paquete ETL.

Dimensiones del indicador:

- **Tiempo** (Utilizaremos el campo FechaHora de cada una de las tablas de seguimientos. Dichas tablas son las siguientes: ResiPsiSeguimientos, ResiFisSeguimientos, ResiDieSeguimientos, ControlesSeguimientos, ResiEnfSeguimientos, ResiTsoSeguimientos, ResiAscSeguimientos, ResiTocSeguimientos, ResiSupSeguimientos, ResiDirSeguimientos, ResiAuxSeguimientos, ResiAesSeguimientos, ResiPedSeguimientos y ResiPodSeguimientos)

  Atributos: IDTiempo, Fecha, Año, MesID, Mes, Trimestre, Semana, NumDiaSemana, DiaSemana, NombreCompleto.

- **Centro** (Utilizaremos la tabla DatosResidencia)

  Atributos: IDCentro, CodigoUnificado, Nombre, Ciudad, Provincia, Comunidad, Pais.

- **Carácter Plaza** (Utilizaremos el campo IDResidente de las tablas de seguimientos para posteriormente atacar a la tabla ResiHisCaracterPlaza con ese id y la fecha que indica el campo FechaHora y obtener el carácter plaza del residente en esa fecha)

  Atributos: IDCaracterPlaza, CodigoUnificado, CaracterPlaza.

- **Tipología** (Utilizaremos el campo IDResidente de las tablas de seguimientos para posteriormente atacar a la tabla ResiHisTipologiasPlaza con ese id y la fecha que indica el campo FechaHora y obtener la tipología del residente en esa fecha)

  Atributos: IDTipologia, CodigoUnificado, Tipologia.

- **Residente** (Utilizaremos el campo IDResidente de las tablas de seguimientos para luego obtener el id del almacén correspondiente para ese residente)

  Atributos: IDResidente, CodigoUnificado, Nombre, Sexo, FechaIngreso.

- **Tipo Profesional** (Asignaremos un IDTipoProfesional a los datos según la tabla de seguimientos utilizada)

  Atributos: IDTipoProfesional, TipoProfesional.

Hechos del indicador:

- Diferencia Seguimientos Profesionales: Cada registro de la tabla de hechos indica el número de días transcurridos entre la elaboración de dos seguimientos consecutivos de un mismo profesional (médico, psicólogo, etc) a un residente. Por tanto el hecho estará formado por el conjunto de dimensiones citadas anteriormente: IDTiempo, IDResidente, IDCaracterPlaza, IDTipologia, IDCentro, IDTipoProfesional y TiempoTranscurrido.

La **granularidad** o el nivel de detalle mínimo que podrá ser consultado será diario.

Miembros Calculados: Este indicador posee los siguientes miembros calculados:

- Número Días: Es la suma de todas las diferencias de días
- Tiempo Transcurrido Count: Es el número de registros de la tabla de hechos.
- Mínimo: Es el tiempo mínimo transcurrido entre dos seguimientos.
- Máximo: Es el tiempo máximo transcurrido entre dos seguimientos.
- Media: Es el número de días medio que transcurre entre dos seguimientos profesionales. Se define como un cociente Número Días/Tiempo Transcurrrido Count.

KPIs: Este indicador posee un KPI que se muestra en rojo para todos aquellos registros cuyo número de días transcurridos entre dos seguimientos sea mayor a 75 y 40 días para la Residencia 1 y 2 respectivamente.

Cubo: Crearemos un cubo en modo Molap que se procesará cada 24 horas, inmediatamente después de la carga de los datos en nuestro almacén de datos. Poseerá las dimensiones enunciadas en el apartado anterior.

3. NOTAS DE DISEÑO

En lo que respecta a la técnica de diseño utilizada, usaremos el esquema en copo de nieve porque alguna de las dimensiones incluye claves que apuntan a otras dimensiones.

Documento de Diseño de Estancia Unidades

	Departamento: Desarrollo	**Tipo:** Documento Diseño	ID5265
Programa/Proyecto: Indicadores tesis		**Analista:** Juanvi	**Fecha creación:** 01/05/2010

1. DESCRIPCIÓN

Creación del indicador de Estancia Unidades para el proyecto de indicadores de la tesis.

2. DISEÑO

Este indicador se carga por medio de un paquete ETL.

Dimensiones del indicador:

- **Tiempo** (Utilizaremos el campo FechaIngreso de la tabla ResiUniEstancias)

  Atributos: IDTiempo, Fecha, Año, MesID, Mes, Trimestre, Semana, NumDiaSemana, DiaSemana, NombreCompleto.

- **Centro** (Utilizaremos la tabla DatosResidencia)

  Atributos: IDCentro, CodigoUnificado, Nombre, Ciudad, Provincia, Comunidad, Pais.

- **Carácter Plaza** (Utilizaremos el campo IDResidente de la tabla ResiUniEstancias para posteriormente atacar a la tabla ResiHisCaracterPlaza con ese id y la fecha que indica el campo FechaHora y obtener el carácter plaza del residente en esa fecha)

  Atributos: IDCaracterPlaza, CodigoUnificado, CaracterPlaza.

- **Tipología** (Utilizaremos el campo IDResidente de la tabla ResiUniEstancias para posteriormente atacar a la tabla ResiHisTipologiasPlaza con ese id y la fecha que indica el campo FechaHora y obtener la tipología del residente en esa fecha)

  Atributos: IDTipologia, CodigoUnificado, Tipologia.

- **Residente** (Utilizaremos el campo IDResidente de la tabla ResiUniEstancias para luego obtener el id del almacén correspondiente para ese residente)

  Atributos: IDResidente, CodigoUnificado, Nombre, Sexo, FechaIngreso.

- **Unidad** (Utilizaremos el campo IDUnidad de la tabla ResiUniEstancias para enlazar con la tabla Unidades)

  Atributos: IDUnidad, CodigoUnificado, Unidad, IDTipoUnidad.

- **Tipo Unidad** (Utilizaremos el campo Tipo de la tabla Unidades)

  Atributos: IDTipoUnidad, TipoUnidad.

En este caso no precisamos de tener un código unificado porque todas las residencias tienen los mismos valores para ese campo y por tanto podemos rellenar la dimensión con los valores utilizados por el programa.

Esta dimensión se considera una subdimensión, ya que no va a aparecer directamente referenciada desde la tabla de hechos. De hecho, será referenciada desde la dimensión Unidad a través del campo IDTipoUnidad.

Hechos del indicador:

- Estancia Unidades: Cada registro de la tabla de hechos indica el número de días que los residentes pasan en las diferentes unidades de la residencia o de un hospital (enfermería, psiquiatría, etc.). Por tanto el hecho estará formado por el conjunto de dimensiones citadas anteriormente: IDTiempo, IDResidente, IDCaracterPlaza, IDTipologia, IDCentro, IDUnidad, TiempoEstancia, TotalEstancia y MinEstancia. En el campo Tiempo Estancia pondremos el número de días que lleva el residente en la unidad en cierta fecha, en Total Estancia, el número total de días que pasa el residente durante una estancia en una unidad y en Min Estancia, el período mínimo que pasa un residente en una unidad.

La **granularidad** o el nivel de detalle mínimo que podrá ser consultado será diario.

Miembros Calculados: Este indicador posee los siguientes miembros calculados:

- Total Estancia: Es la suma de todas las estancias en unidades.
- NumRegistros: Son aquellos valores no nulos del campo TotalEstancia.
- Mínimo Tiempo: Es el tiempo mínimo que los residentes han estado en las unidades.
- Máximo Tiempo: Es el tiempo máximo que los residentes han estado en las unidades.
- Media Estancia: Es el número de días medio que los residentes han estado en las unidades. Se define como un cociente TotalEstancia/NumRegistros.

KPIs: Este indicador posee un KPI que se muestra en rojo para todos aquellos registros cuyo tiempo de estancia sea mayor a 200 y 730 días para la Residencia 1 y 2 respectivamente.

Cubo: Crearemos un cubo en modo Molap que se procesará cada 24 horas, inmediatamente después de la carga de los datos en nuestro almacén de datos. Poseerá las dimensiones enunciadas en el apartado anterior.

3. NOTAS DE DISEÑO

En lo que respecta a la técnica de diseño utilizada, usaremos el esquema en copo de nieve porque alguna de las dimensiones incluye claves que apuntan a otras dimensiones.

6.5. Implementación de indicadores

A partir de la documentación generada en el punto anterior, vamos a implementar los tres indicadores de negocio, creando los respectivos paquetes, cubos e informes necesarios para la visualización de cada uno de ellos. Una vez finalizados, utilizaremos los tres indicadores junto con el indicador de Visitas Familiares, utilizado en los tutoriales, para elaborar un posible dashboard para nuestro cliente y crearemos un trabajo que se encargue de actualizar los datos de los indicadores diariamente.

Dado que el proceso de implementación de un indicador consta de diversas tareas, en los apartados siguientes trataremos cada una de ellas.

6.5.1. Creación paquetes ETL

A continuación vamos a crear los paquetes ETL necesarios para el desarrollo de los indicadores. Concretamente, estos paquetes han de crear la estructura del almacén de datos, rellenar los códigos unificados de las dimensiones que lo requieran, rellenar las dimensiones estáticas que no cogen información de tablas maestras y finalmente rellenar las tablas de hechos de los indicadores.

Además, en ocasiones será necesario disponer de tablas auxiliares para obtener información que es consultada frecuentemente (tipología de un residente en cierta fecha) y los paquetes ETL también han de encargarse de crear y rellenar estas tablas.

Dado que el número de tareas a realizar es elevado, vamos a crear varios paquetes ETL. El primero de ellos se encargará de construir el almacén de datos y rellenar las dimensiones estáticas; el segundo rellenará las dimensiones que utilizan código unificado pero su información no se introduce mediante tablas maestras (residentes, úlceras) y luego tendremos uno por cada indicador.

Puesto que en el tutorial de Integration Services se detalla paso a paso cómo crear un paquete ETL, en este punto no entraremos en detalle en cómo realizar las acciones vistas en ese tutorial (crear un proyecto, crear una conexión, etc).

Antes de comenzar la implementación de los paquetes ETL comentados anteriormente necesitamos crear un proyecto SSIS. Como ya creamos uno para el indicador de Visitas Familiares, utilizaremos ese mismo proyecto.

Añadimos 5 nuevos paquetes SSIS al proyecto y los renombramos para que tengan un nombre significativo. Los nombres de dichos paquetes serán Almacen, Dimension, HechoUlcerasCuras, HechoDiferenciaSeg y HechoEstanciaUnidades. Los pasos a seguir son:

1. Seleccionamos la carpeta **Paquetes SSIS** en el **Explorador de soluciones**, hacemos clic con el botón derecho y seleccionamos **Nuevo Paquete SSIS**. Automáticamente se creará un nuevo paquete con un nombre por defecto.
2. Seleccionamos el paquete recién creado y le cambiamos el nombre en la ventana **Propiedades**.

El proyecto en el que estamos trabajando consta de dos conexiones, una a la base de datos operacional de la que se extraen los datos para el indicador y otra al almacén de datos. No obstante, necesitamos añadir dos nuevas conexiones: una que se conecte a la base de datos **master** de nuestra máquina para poder crear las tablas del almacén de datos y otra que se conecte a la base de datos **DatosUsuarios** para poder obtener el nombre del centro para el que lanzamos los paquetes.

Después de crear los paquetes necesarios y las conexiones que utilizará nuestro proyecto, el Explorador de soluciones (Figura 6.4.) mostrará lo siguiente:

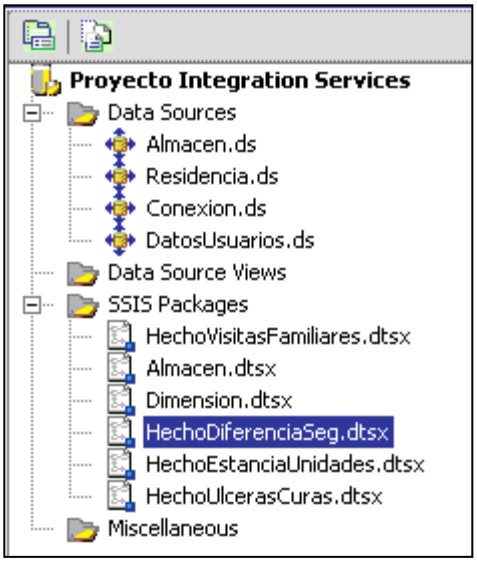


Figura 6.4. Explorador de soluciones

Hay que tener en cuenta que si queremos lanzar los paquetes para más de una residencia, tendremos que cambiar la configuración de la conexión Residencia para que apunte a la base de datos del nuevo centro para el que queremos lanzar los paquetes antes de lanzar dichos paquetes.

Paquete Almacen

El paquete Almacen es el encargado de construir el almacén de datos y rellenar las dimensiones estáticas. Dado que es el encargado de crear las tablas del almacén de datos, es el paquete que primero se ejecuta. Se ejecuta al principio del proceso y no se vuelve a ejecutar hasta que no se necesita introducir cambios en el paquete para crear nuevas tablas en el almacén o se desea borrar el contenido del almacén de datos.

En definitiva, las tareas que debe realizar este paquete son:

- Eliminar las tablas auxiliares.
- Eliminar las tablas de hechos y de las dimensiones cuyo código unificado es rellenado por un paquete ETL (residentes, úlceras).
- Crear las tablas de dimensiones.
- Crear las tablas de hechos.
- Crear las tablas auxiliares.
- Rellenar dimensiones estáticas.
- Reiniciar las semillas de aquellas tablas cuyo contenido es borrado para que los id empiecen en 1.

Lo primero que tenemos que hacer ir al panel de **Administradores de conexión**, hacer clic en **Nueva conexión desde origen de datos** y añadir los siguientes orígenes de datos: Almacen y Conexión.

Cuando tengamos las conexiones configuradas, pasamos a construir el flujo de control de nuestro paquete, el cual va a estar compuesto de las tareas indicadas previamente.

Para construir el flujo de control utilizaremos contenedores de secuencias, ya que agrupan tareas en una unidad lógica de trabajo. Dentro de estos contenedores de secuencias utilizaremos tareas que ejecutan SQL para introducir los scripts de la base de datos.

Dicho esto, arrastramos 6 contenedores de secuencias desde el cuadro de herramientas a la superficie de diseño de nuestro flujo de control y los conectamos unos con otros arrastrando las flechas verdes.

A continuación se detallan las acciones a realizar para cada uno de estos contenedores:

Eliminar Tablas Auxiliares

Esta tarea es la encargada de borrar las tablas auxiliares del almacén de datos. Se compone de tres tareas, tantas como tablas auxiliares necesitamos para los indicadores, cada una de las cuales contiene el código SQL necesario para borrar una tabla auxiliar (Figura 6.5.).

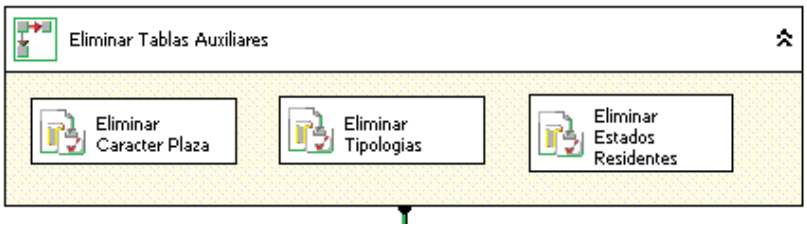


Figura 6.5. Contenedor de secuencia Eliminar Tablas Auxiliares

Dado que los indicadores necesitan información sobre la tipología, carácter plaza o estado de un residente para una fecha cualquiera y esta información no se almacena tal cual en el programa, necesitamos tablas auxiliares para almacenar esta información. El hecho de que sean tres tablas y no una solamente es porque preferimos tener la información separada.

Para rellenar el contenedor de secuencia hay que hacer lo siguiente:

1. Arrastramos una tarea que ejecuta SQL desde el cuadro de herramientas al interior del contenedor de secuencias.
2. Hacemos doble clic sobre la tarea para editar la tarea (Figura 6.6.) y rellenamos el campo **Nombre**, seleccionamos la conexión que utiliza, pulsamos el botón que aparece en el campo del comando SQL para introducir el código SQL que debe ejecutar y una vez introducido pulsamos **Aceptar**.

   El código SQL que debe ejecutar cada una de ellas debe comprobar que la tabla en cuestión existe y si existe, proceder a su borrado. A continuación se muestra el código de la tarea Eliminar Carácter Plaza a modo de ejemplo:

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[CaracterPlaza]') AND type in (N'U'))
DROP TABLE [dbo].[CaracterPlaza]
```

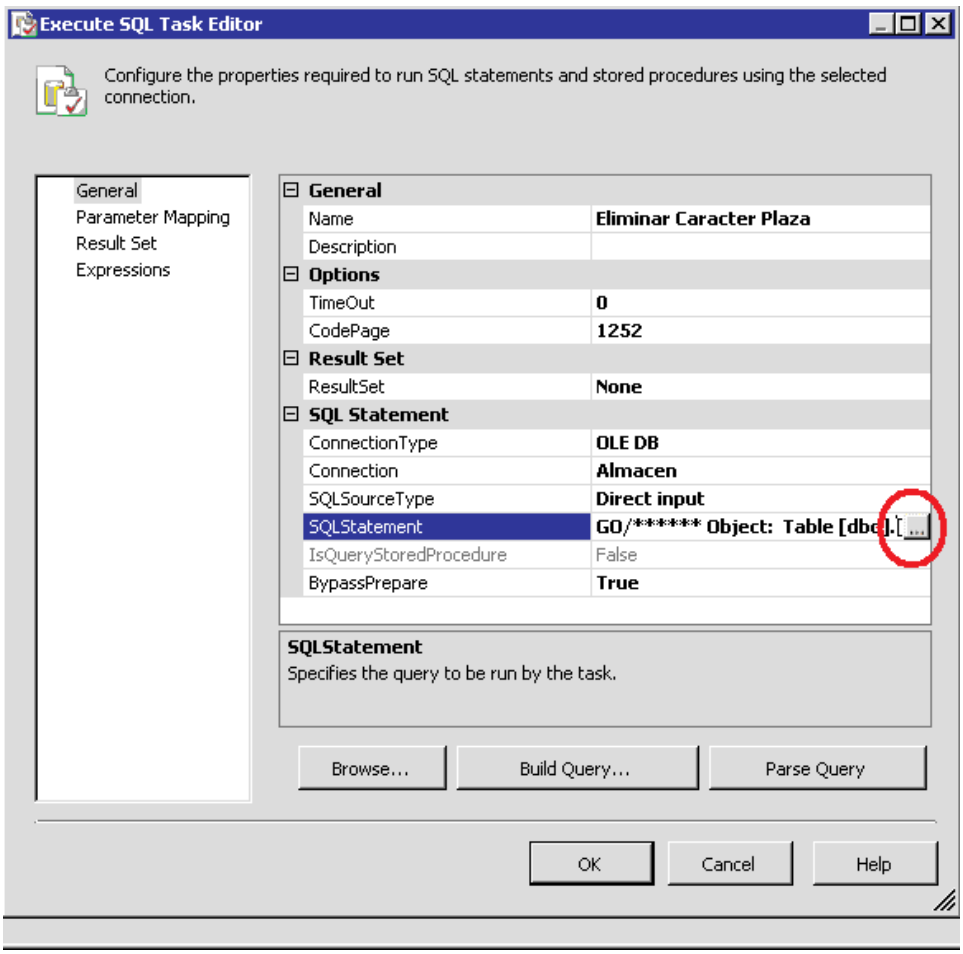


Figura 6.6.Editor tarea SQL

3. Repetimos los pasos 1-2 con las otras dos tareas.
4. Renombramos el nombre del contenedor de secuencias haciendo clic sobre el nombre o haciendo clic con el botón derecho, seleccionando la ventana Propiedades y cambiando el nombre.

Eliminar Hechos y Dimensiones

Esta tarea (Figura 6.7.) es la encargada de borrar las tablas de hechos y las tablas de las dimensiones cuyo código unificado es rellenado automáticamente con un paquete ETL, ya que dicho código unificado es susceptible de cambios en su construcción. Estas dimensiones son úlceras, residente y centro.

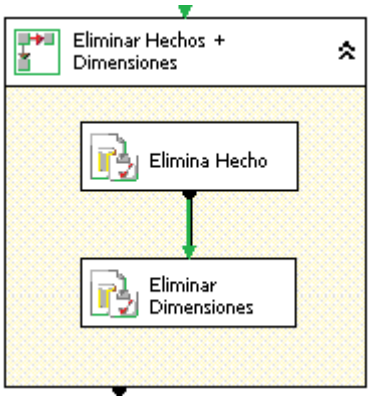


Figura 6.7. Contenedor de secuencia Eliminar Hechos y Dimensiones

Para rellenar el contenedor de secuencia hacemos lo mismo que antes con la diferencia de que ahora necesitamos establecer una relación de precedencia entre ambas tareas. De no hacerlo se produciría un error porque las dimensiones no pueden ser borradas antes que las tablas de hechos, ya que éstas últimas referencian a las de dimensiones.

El código SQL a introducir en las tareas es similar al anterior y lo único que hay que cambiar es el nombre de la tablas que se borran.

Crear Dimensiones

Esta tarea es la encargada de crear todas las tablas de dimensiones del almacén de datos. Se compone de tantas tareas como tablas de dimensiones necesitamos crear (Figura 6.8.).

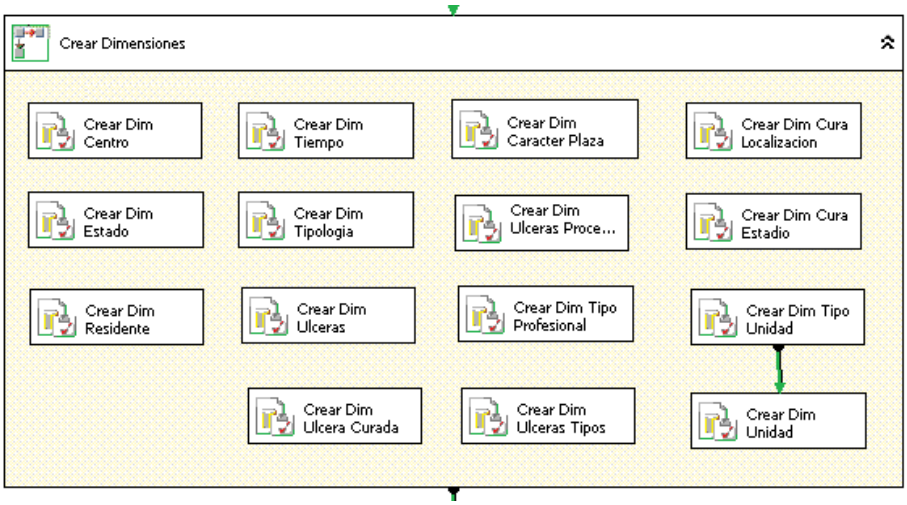


Figura 6.8. Contenedor de secuencia Crear Dimensiones

Para rellenar el contenedor de secuencia hacemos lo mismo que antes. Sim embargo, en este caso el código SQL a introducir en las tareas difiere, ya que ahora lo que hacemos es crear las tablas con sus atributos. Además es posible que haya relaciones de precedencia entre las dimensiones como la que se observa entre Dim Tipo Unidad y Unidad y eso también hay que indicarlo en dicho código. En el caso de estas dos dimensiones, Dim Tipo Unidad ha de ser creada previamente porque se trata de una subdimensión de Dim Unidad.

El código SQL de la dimensión Unidad sería el siguiente. En él aparecen en otro color las referencias a las otras tablas:

```
if not exists (Select * From information_schema.tables where table_name='Dim_Unidad')
  CREATE TABLE [dbo].[Dim_Unidad](
   [IDUnidad] [int] IDENTITY(1,1) NOT NULL,
   [CodigoUnificado] [nvarchar](19) NULL,
   [Unidad] [nvarchar](50) NULL,
   [IDTipoUnidad] [int] NOT NULL
  PRIMARY KEY CLUSTERED
  (
```

```
[IDUnidad] 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_Unidad_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
ALTER TABLE [dbo].[Dim_Unidad] WITH CHECK ADD FOREIGN
KEY([IDTipoUnidad])
REFERENCES [dbo].[Dim_TipoUnidad] ([IDTipoUnidad])
GO
```

El código del resto de tareas es similar pero no incluye la referencia a otra tabla.

Crear Hechos

Esta tarea es la encargada de crear todas las tablas de hechos del almacén de datos. Se compone de tantas tareas como tablas de hechos necesitamos crear (Figura 6.9.).

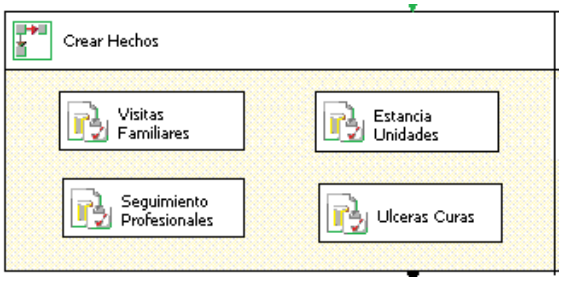


Figura 6.9. Contenedor de secuencia Crear Hechos

Para rellenar el contenedor de secuencia, arrastramos las tareas de SQL al interior del contenedor y editamos sus propiedades. En este caso, el código SQL de dichas tareas debe crear las tablas de hechos indicando las referencias de estas tablas a las tablas de dimensiones.

El código SQL de la tarea que crea el hecho Seguimientos Profesionales es el siguiente:

```
if not exists (Select * From information_schema.tables where
table_name='Fact_UlcerasCUras')
CREATE TABLE [dbo].[Fact_UlcerasCuras](
```

```
    [IDUlceras] [int] NOT NULL,
    [IDTiempo] [int] NOT NULL,
    [IDResidente] [int] NOT NULL,
    [IDCaracterPlaza] [int] NULL,
    [IDTipologia] [int] NULL,
    [IDCentro] [int] NOT NULL,
    [IDCurasLocalizacion] [int] NULL,
    [IDUlcerasTipos] [int] NULL,
    [IDUlcerasProcedencias] [int] NULL,
    [IDCurasEstadio] [int] NULL,
    [TamanoUlcera] [float] NULL,
    [IDUlceraCurada] [int] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Fact_UlcerasCuras] WITH CHECK ADD FOREIGN
KEY([IDCaracterPlaza])
REFERENCES [dbo].[Dim_CaracterPlaza] ([IDCaracterPlaza])
GO

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

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

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

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

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

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

ALTER TABLE [dbo].[Fact_UlcerasCuras] WITH CHECK ADD FOREIGN KEY([IDUlceras])
REFERENCES [dbo].[Dim_Ulceras] ([IDUlceras])
GO

ALTER TABLE [dbo].[Fact_UlcerasCuras] WITH CHECK ADD FOREIGN KEY([IDUlcerasTipos])
REFERENCES [dbo].[Dim_UlcerasTipos] ([IDUlcerasTipos])
GO

ALTER TABLE [dbo].[Fact_UlcerasCuras] WITH CHECK ADD FOREIGN KEY([IDUlcerasProcedencias])
REFERENCES [dbo].[Dim_UlcerasProcedencias] ([IDUlcerasProcedencias])
GO

ALTER TABLE [dbo].[Fact_UlcerasCuras] WITH CHECK ADD FOREIGN KEY([IDUlceraCurada])
REFERENCES [dbo].[Dim_UlceraCurada] ([IDUlceraCurada])
GO
```

El código del resto de tareas es similar pero cambian las referencias a las tablas de dimensiones, ya que no todos los hechos tienen las mismas dimensiones.

Crear Tablas Auxiliares

Esta tarea es la encargada de crear todas las tablas auxiliares. Se compone de tres tareas porque de momento sólo se necesitan tres tablas auxiliares (Figura 6.10).

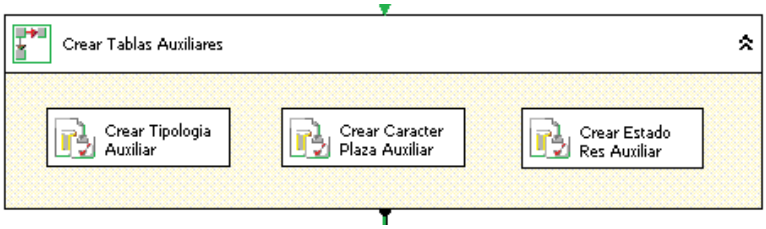


Figura 6.10. Contenedor de secuencia Crear Tablas Auxiliares

Para rellenar el contenedor de secuencia, arrastramos las tareas de SQL al interior del contenedor y editamos sus propiedades. En este caso, el código SQL de dichas tareas es como el siguiente:

```
if not exists (Select * From information_schema.tables where table_name='Tipologias')
CREATE TABLE [dbo].[Tipologias](
    [IDResidenteAlmacen] [int] NULL,
    [Fecha] [datetime] NULL,
    [IDTipologiaAlmacen] [int] NULL,
) ON [PRIMARY]
```

Las tablas auxiliares sólo van a estar compuestas de tres columnas: una para el residente, otra para la fecha y la última para almacenar la tipología, carácter plaza o estado. Esta última columna es la que varía de una tabla auxiliar a la otra.

Rellenar dimensiones

Esta tarea (Figura 6.11.) es la encargada de rellenar las dimensiones estáticas, es decir, las dimensiones que no se rellenan por medio de las tablas maestras que proporciona la interfaz del programa Resiplus BI. Estas dimensiones se caracterizan porque no tienen código unificado, ya que los valores son comunes para todas las residencias. Debido a esto, las dimensiones se rellenan con los valores utilizados por el programa Resiplus para poder enlazar con las tablas directamente, sin necesidad de comparar ningún tipo de código.

En este grupo también se incluyen las dimensiones que se crean con objeto de diferenciar datos. Un ejemplo de este tipo de dimensiones es la Dim Tipo Profesional, la cual no se enlaza con ninguna otra tabla, sino que se utiliza para identificar la tabla de la que proceden ciertos datos tal y como veremos en el paquete de Seguimientos Profesionales. Dicha tabla se rellena con los diferentes tipos de profesionales que realizan seguimientos.

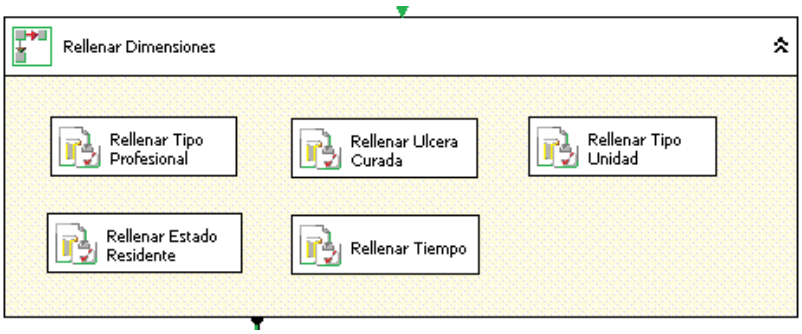


Figura 6.11. Contenedor de secuencia Rellenar Dimensiones

Para rellenar el contenedor de secuencia, arrastramos las tareas de SQL al interior del contenedor y editamos sus propiedades. En este caso, el código SQL de dichas tareas es como el siguiente:

```
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
```

Aparecerán tantas instrucciones de INSERT como valores queramos añadir a la dimensión. Según esto los posibles estados de un residente son: de baja voluntaria, expulsado, fallecido, en el hospital, en la residencia o ausente. Además, se ha añadido una instrucción para que los id de estos estados empiecen en 1 siempre.

Una vez hemos terminado de rellenar los contenedores de secuencia, para acabar el desarrollo de este paquete lo único que falta es arrastrar una tarea que ejecute SQL a la superficie de diseño del flujo de control. Dicha tarea será la encargada de reiniciar las semillas de aquellas tablas de dimensiones cuyo contenido fue eliminado al principio para que los id empiecen en 1. El código de esta tarea sería el siguiente:

```
DBCC CHECKIDENT ('Dim_Residente' ,RESEED,1)
DBCC CHECKIDENT ('Dim_Centro' ,RESEED,1)
DBCC CHECKIDENT ('Dim_Ulceras' ,RESEED,1)
```

Paquete Dimension

El paquete Dimension es el encargado de rellenar las dimensiones que utilizan código unificado pero su información no se introduce mediante tablas maestras (centro, residentes y úlceras) y de cargar los datos en las tablas auxiliares creadas previamente. Se ejecuta tras el paquete Almacen y tantas veces como residencias queremos utilizar para los resultados de nuestros indicadores, es decir, si queremos lanzar los indicadores para dos centros, debemos lanzar este paquete para cada uno de los centros, previo cambio de la configuración de la conexión Residencia.

En definitiva, las tareas que debe realizar este paquete son:

- Obtener ciertas variables que nos ayuden a identificar el nombre del centro para el que vamos a lanzar el paquete.
- Crear y rellenar los códigos unificados en las tablas de la base de datos operacional.
- Cargar las dimensiones que no se rellenan a través de las tablas maestras.
- Cargar las tablas auxiliares.

Antes de empezar con el desarrollo del paquete, debemos acceder a la interfaz del Resiplus BI y rellenar las tablas maestras que necesitamos para nuestros indicadores. Las tablas a rellenar son las que corresponden a las siguientes dimensiones: Carácter Plaza, Tipología, Curas Localización, Curas Estadio, Úlceras Procedencias, Úlceras Tipos y Unidad.

Para todas estas tablas, el campo Código Unificado ya ha sido creado y rellenado en su tabla homónima en la base de datos operacional y por eso solamente tenemos que rellenar las tablas del almacén de datos.

La Figura 6.12. muestra la interfaz del Resiplus BI para rellenar las tablas maestras.

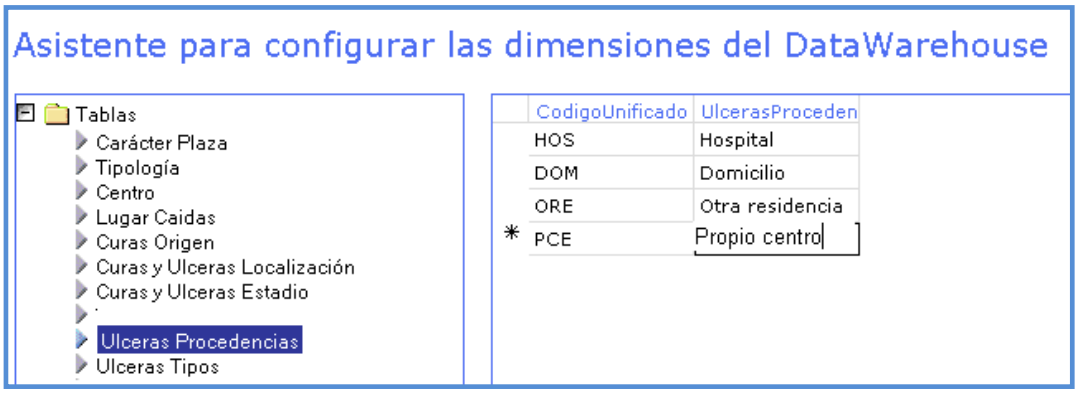


Figura 6.12. Interfaz Resiplus BI para rellenar la tabla maestra Ulceras Procedencias

A continuación, vamos al panel de **Administradores de conexión**, hacer clic en **Nueva conexión desde origen de datos** y añadir los siguientes orígenes de datos: Almacen, Residencia y DatosUsuarios.

Cuando tengamos las conexiones configuradas, necesitamos crear una serie de variables (Figura 6.13.) que vamos a utilizar posteriormente en el flujo de control. Estas variables se utilizan principalmente en las tareas encargadas de obtener el nombre del centro para el que queremos lanzar el paquete.

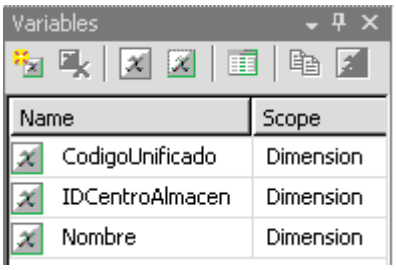


Figura 6.13. Variables definidas en el paquete

Ahora construiremos el flujo de control de nuestro paquete, el cual va a estar compuesto de las tareas indicadas previamente.

Para construir el flujo de control utilizaremos contenedores de secuencias, ya que agrupan tareas en una unidad lógica de trabajo. Dentro de estos contenedores de secuencias utilizaremos tareas que ejecutan SQL y tareas de flujo de datos.

Dicho esto, arrastramos 4 contenedores de secuencias desde el cuadro de herramientas a la superficie de diseño de nuestro flujo de control y los conectamos unos con otros arrastrando las flechas verdes.

Las acciones a realizar para cada uno de estos contenedores:

Obtención de variables

Esta tarea es la encargada de dar valor a las variables definidas anteriormente. Consta de dos tareas de flujos de datos entre las que existe una relación de precedencia (Figura 6.14.).

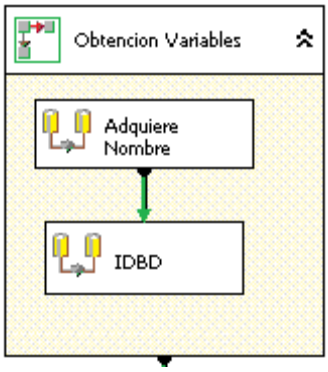


Figura 6.14. Contenedor de secuencia Obtención Variables

Para rellenar el contenedor de secuencia hay que hacer lo siguiente:

1. Arrastramos una tarea de flujo de datos desde el cuadro de herramientas al interior del contenedor de secuencias.
2. Hacemos clic sobre la tarea y seleccionamos la pestaña **Flujo de datos**.
3. Arrastramos un **Origen OLE DB** y un **Componente de script** a la superficie de diseño del flujo de datos y los conectamos con la flecha verde (Figura 6.15.). Utilizamos estas transformaciones porque son las que necesitamos en este caso pero éstas serán distintas según lo que se vaya a hacer.

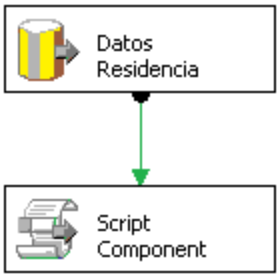


Figura 6.15. Flujo de datos asociado a la tarea Adquiere Nombre

   El funcionamiento de este flujo de datos es el siguiente: accede a la tabla DatosResidencia utilizando la conexión Residencia para adquirir el nombre de la residencia y realizar cierto formateo del mismo y el nombre resultante lo guarda en la variable Nombre por medio de un sencillo script.

4. Repetimos los pasos 1-3 con la otra tarea y construimos un flujo de datos similar que acceda a la tabla BDs utilizando la conexión DatosUsuarios y rellene la variable **CodigoUnificado**.
5. Renombramos el nombre del contenedor de secuencias y de las tareas de flujo de datos accediendo a la ventana de **Propiedades**.

En este caso concreto, no se detalla el contenido de los scripts ni de las consultas de los orígenes de datos porque son difíciles de comprender a simple vista y sólo un experto en el programa las entendería.

Crear y Rellenar Códigos Unificados

Esta tarea (Figura 6.16.) es la encargada de crear los campos de código unificado que no existen en la base de datos operacional y rellenar dichos campos. Consta de cuatro tareas

que ejecutan código SQL entre las que se existen relaciones de precedencia, ya que primero se crea el campo y luego se rellena.

Esta acción es necesaria para las dimensiones que no se rellenan a través de tablas maestras. Anteriormente indicamos que estas dimensiones eran centro, residente y úlcera pero para centro no se hace porque no existe ninguna tabla con toda la lista de centros.

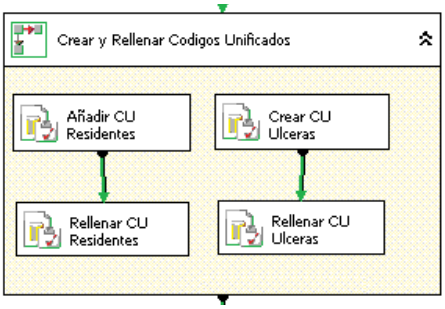


Figura 6.16. Contenedor de secuencia Crear y Rellenar Codigos Unificados

Para rellenar el contenedor de secuencia, arrastramos las tareas de SQL al interior del contenedor y editamos sus propiedades. Las tareas que crean el campo para el código unificado tienen un código similar a éste:

```
if not exists (select * from information_schema.columns where
Table_name='Residentes' and column_name='CodigoUnificado')
alter table Residentes
add CodigoUnificado nvarchar(50)
```

En cambio, para las tareas que rellenan el código unificado su código sería similar a éste:

```
declare @bd nvarchar(50)
select @bd=(select distinct replace(replace(table_catalog,'DatosResidencia','-
'),'_','') from information_schema.tables)
UPDATE Residentes
SET CodigoUnificado=convert(nvarchar,IDResidente)+@bd
```

Con este código, los códigos unificados de los residentes serán de la forma IDResidente-NombreBD.

Cargar Dimensiones

Esta tarea (Figura 6.17.) es la encargada de cargar datos en las dimensiones que no se rellenan a través de tablas maestras. Consta de tres tareas de flujo de datos, cuyo trabajo consiste en sacar información de la base de datos operacional y almacenarla en el almacén de datos.

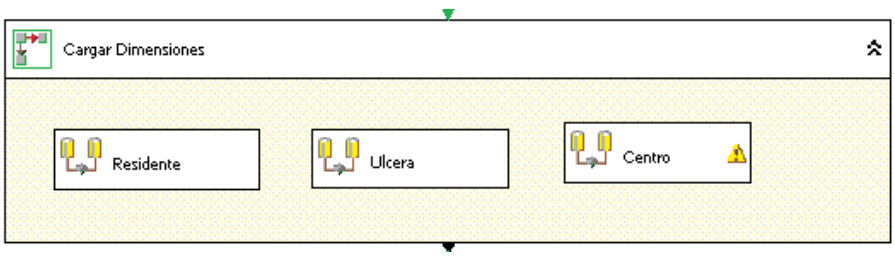


Figura 6.17. Contenedor de secuencia Cargar Dimensiones

Para rellenar el contenedor de secuencia, arrastramos las tareas de flujo de datos al interior del contenedor y creamos el flujo de datos de cada una de ellas tal y como se indicó anteriormente.

A continuación se detalla únicamente cómo crear el flujo de datos de la tarea Residente (Figura 6.18.), ya que el resto funcionan exactamente igual pero cambiando las tablas de origen y destino.

El flujo a construir es el siguiente:

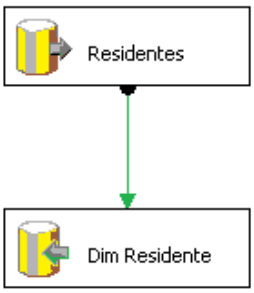


Figura 6.18. Flujo de datos asociado a la tarea Residente

Para construir este flujo arrastramos un **Origen OLE DB** y un **Destino OLE DB** a la superficie de diseño del flujo de datos.

Como lo que queremos es pasar todos los datos de los residentes de la tabla de la base de datos operacional a la del almacén de datos, hacemos doble clic sobre el **Origen OLE DB** y lo configuramos tal y como se indica más abajo (Figura 6.19.). Para ello, seleccionamos como conexión Residencia porque es la que se conecta a la base de datos operacional, introducimos el comando SQL y pulsamos **Aceptar**.

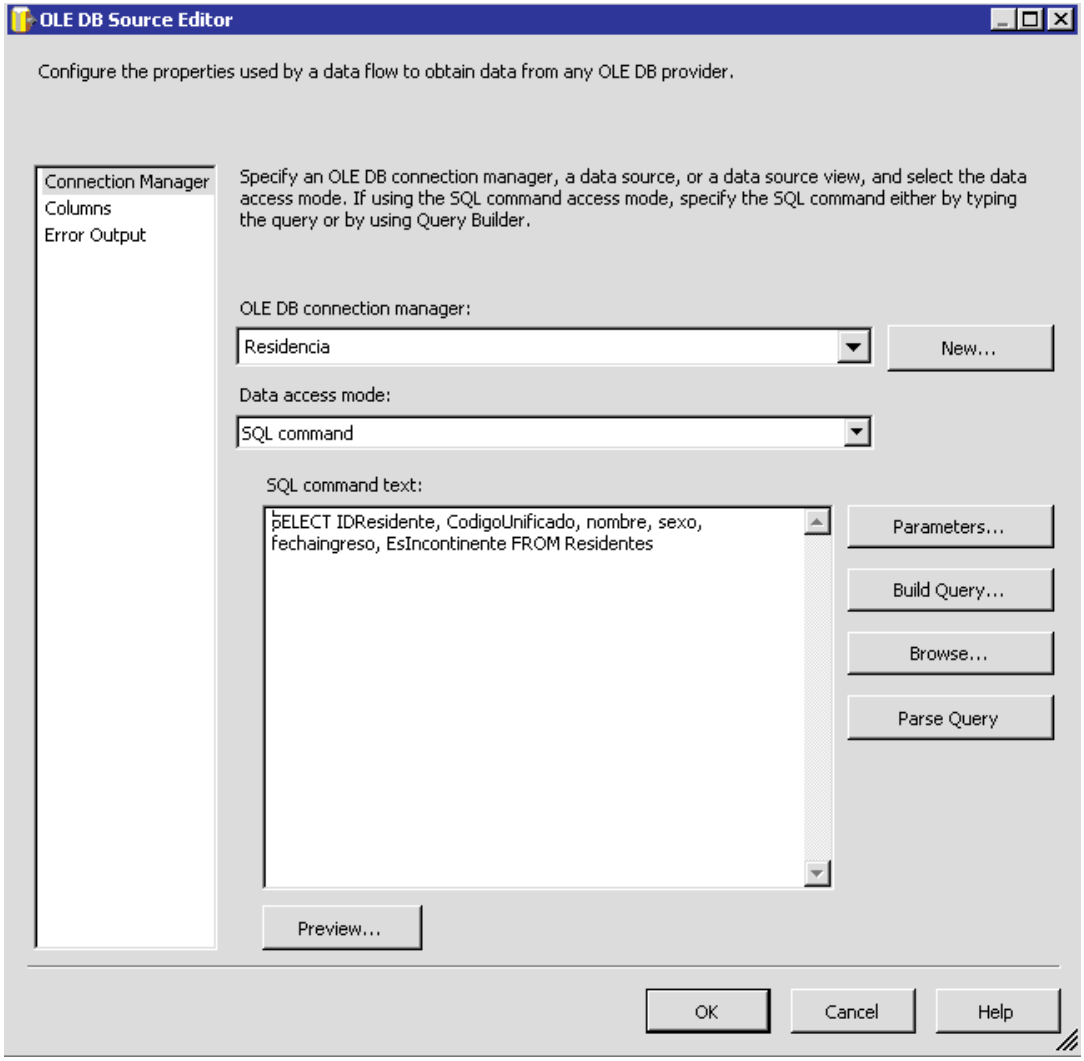


Figura 6.19. Ventana configuración Origen OLE DB Residente

Ahora hacemos doble clic sobre el **Destino OLE DB**, seleccionamos como **Modo de acceso a datos** la opción de **Carga rápida de tabla o vista** y seleccionamos la tabla que vamos a rellenar, que en este caso es Dim_Residente. Luego hacemos clic en **Asignaciones** y hacemos corresponder las columnas de entrada con las columnas de destino correspondientes.

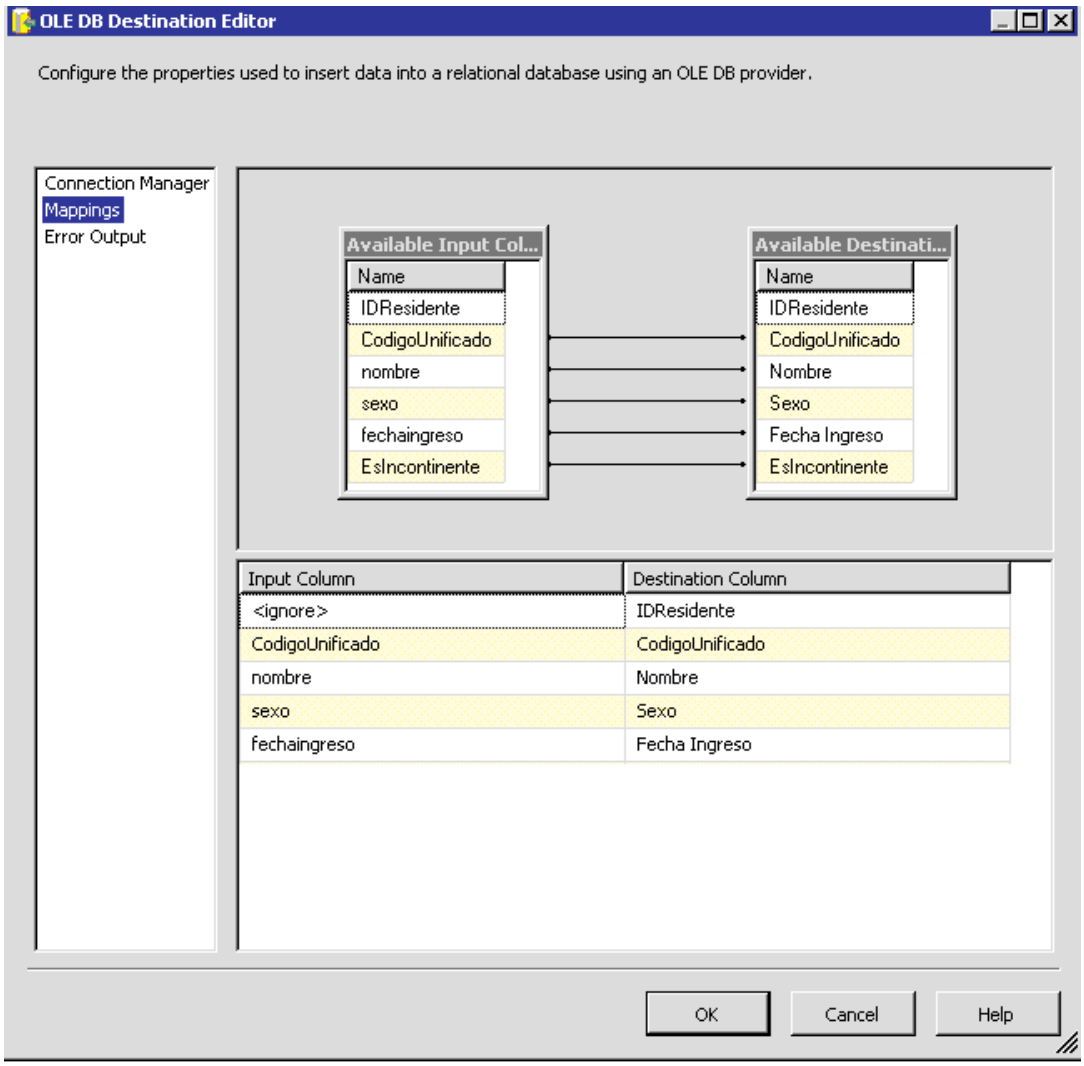


Figura 6.20. Ventana Asignaciones del Destino OLE DB Dim Residente

Cargar Tablas Auxiliares

Esta tarea (Figura 6.21.) es la encargada de cargar datos en las tablas auxiliares. Consta de tres tareas de flujo de datos, cuyo trabajo consiste en sacar información de las tablas de la base de datos operacional, procesarla para que se ajuste a las necesidades de la tabla y almacenarla en las tablas correspondientes del almacén de datos.

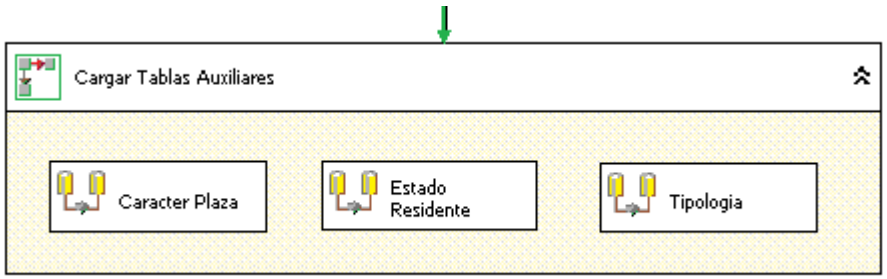


Figura 6.21. Contenedor de secuencia Cargar Tablas Auxiliares

Para rellenar el contenedor de secuencia, arrastramos las tareas de flujo de datos al interior del contenedor y creamos el flujo de datos de cada una de ellas tal y como se indicó anteriormente.

A continuación se detalla únicamente cómo crear el flujo de datos de la tarea Tipología, ya que el resto funcionan de forma parecida pero cambiando las tablas de origen y destino y el código de los scripts.

Dado que el flujo a construir es bastante grande en comparación con los vistos hasta el momento, para explicarlo vamos a dividir dicho flujo en varias partes.

La primera de estas partes es la que muestra la Figura 6.22.:

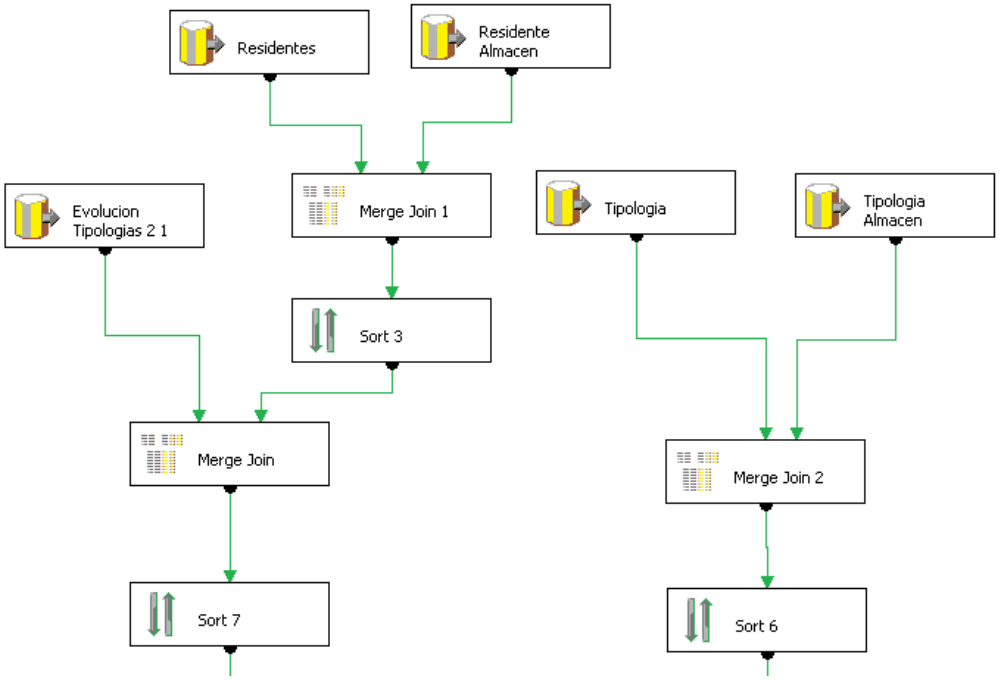


Figura 6.22. Parte 1 del flujo de datos asociado a la tarea Tipologia

En la tabla auxiliar necesitamos almacenar información sobre la tipología que tiene un residente en una cierta fecha. Como es una tabla del almacén, los IDs almacenados han de ser los correspondientes a las dimensiones del almacén y no los que aparecen en la base de datos operacional. Por este motivo, la información obtenida de la base de datos operacional ha de ser enlazada con la información del almacén de datos.

Lo primero que necesitamos es arrastrar a la superficie de diseño del flujo de datos un **Origen OLE DB** que utilizaremos para acceder a la tabla donde se guarda el histórico de tipologías de residentes. Esta tabla guarda un registro por cada cambio de tipología de un residente.

Hacemos doble clic en el origen (Figura 6.23.), seleccionamos Residencia porque nos conectamos a la base de datos operacional e introducimos la consulta directamente o a través de **Generar consulta**.

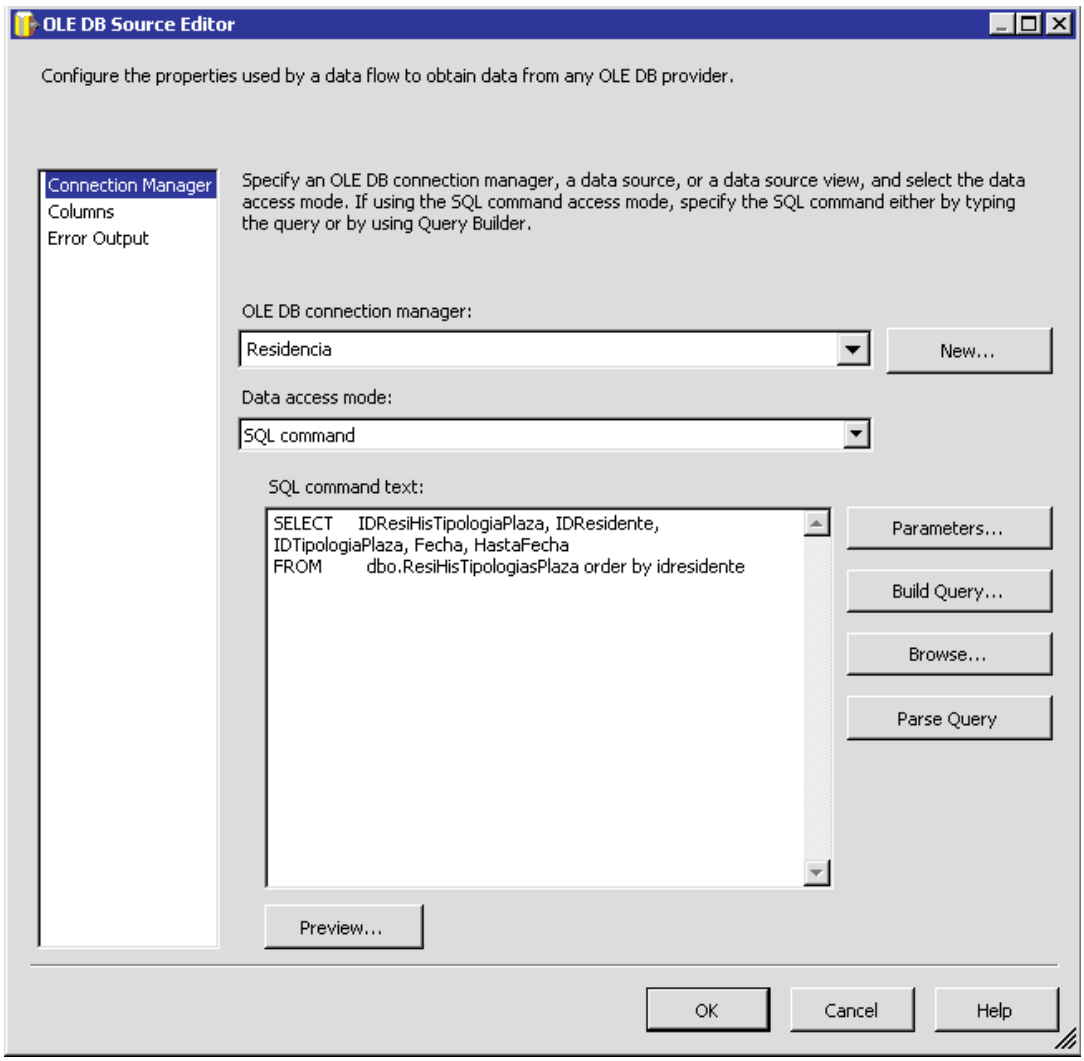


Figura 6.23. Ventana configuración OLE DB Evolucion Tipologias

El siguiente paso es obtener el IDResidente del almacén de datos que se corresponde con el IDResidente de los datos obtenidos. Como lo que vamos a hacer es unir los datos obtenidos con los datos del almacén usando el IDResidente, necesitamos ordenar los datos por dicho campo. Dicha ordenación se puede realizar utilizando una transformación **Ordenar** o bien modificando la configuración del origen.

Para realizar la ordenación de la segunda forma, debemos hacer clic sobre el origen y seleccionar **Mostrar Editor Avanzado**. Vamos a la última pestaña (Figura 6.24.), ponemos a cierto el campo **IsSorted**, seleccionamos la columna por la que deseamos ordenar el conjunto de datos (IDResidente) y ponemos la propiedad SortKeyPosition a 1.

De esta forma nos ahorramos poner una transformación y el paquete es algo más eficiente.

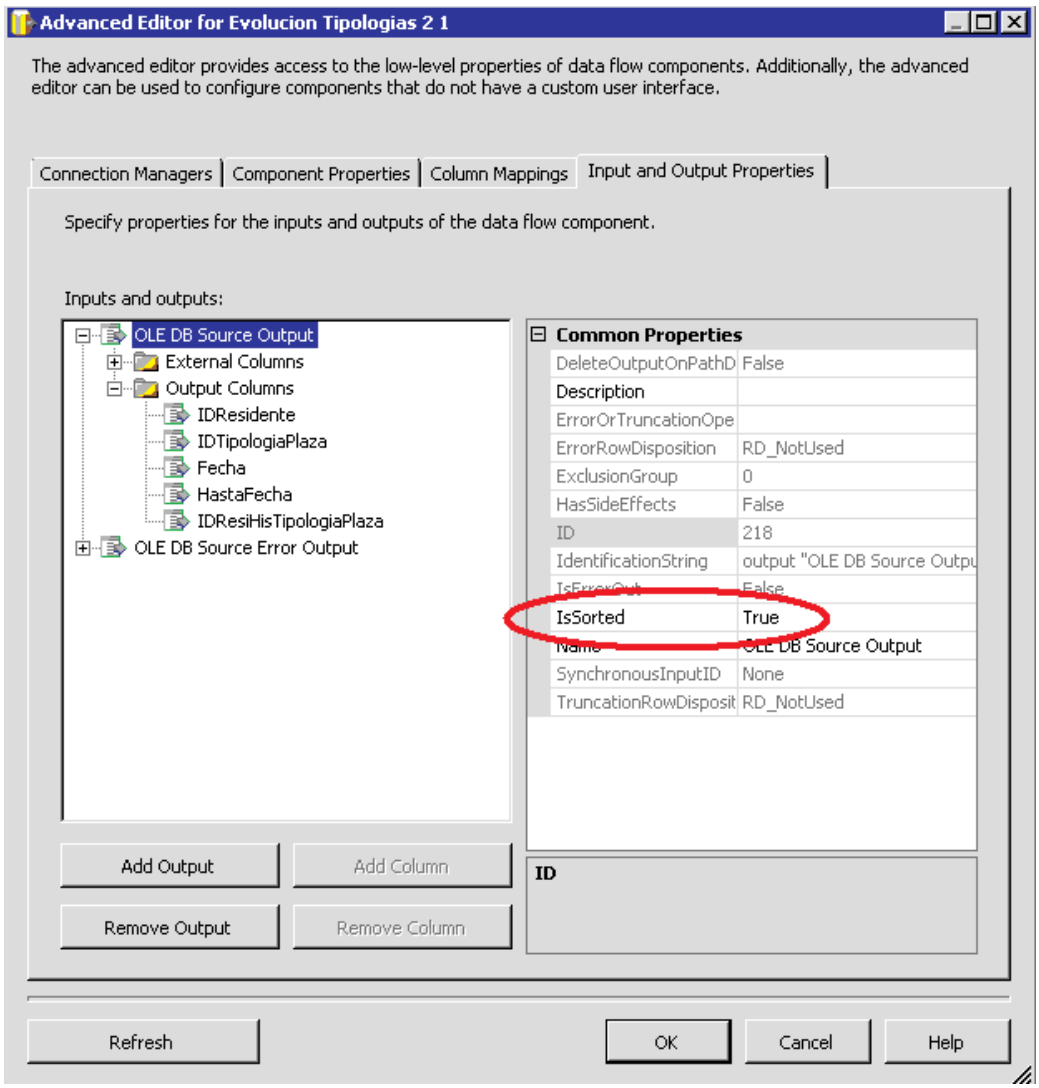


Figura 6.24. Editor Avanzado OLE DB Evolucion Tipologias

Cuando tenemos los datos ordenados, lo siguiente es obtener la correspondencia de IDs de la residencia y el almacén para los residentes. Para ello añadimos dos nuevos orígenes: uno que saque la información de los residentes de la residencia y otro que saque esta misma información pero del almacén de datos.

Como lo que nos interesa es la información de los IDs, la consulta sólo recupera los campos ID y CodigoUnificado (Figura 6.25.), siendo éste último el importante para enlazar con el almacén de datos por ser común en ambas bases de datos. La configuración del origen que accede a la base de datos operacional queda de la siguiente forma:

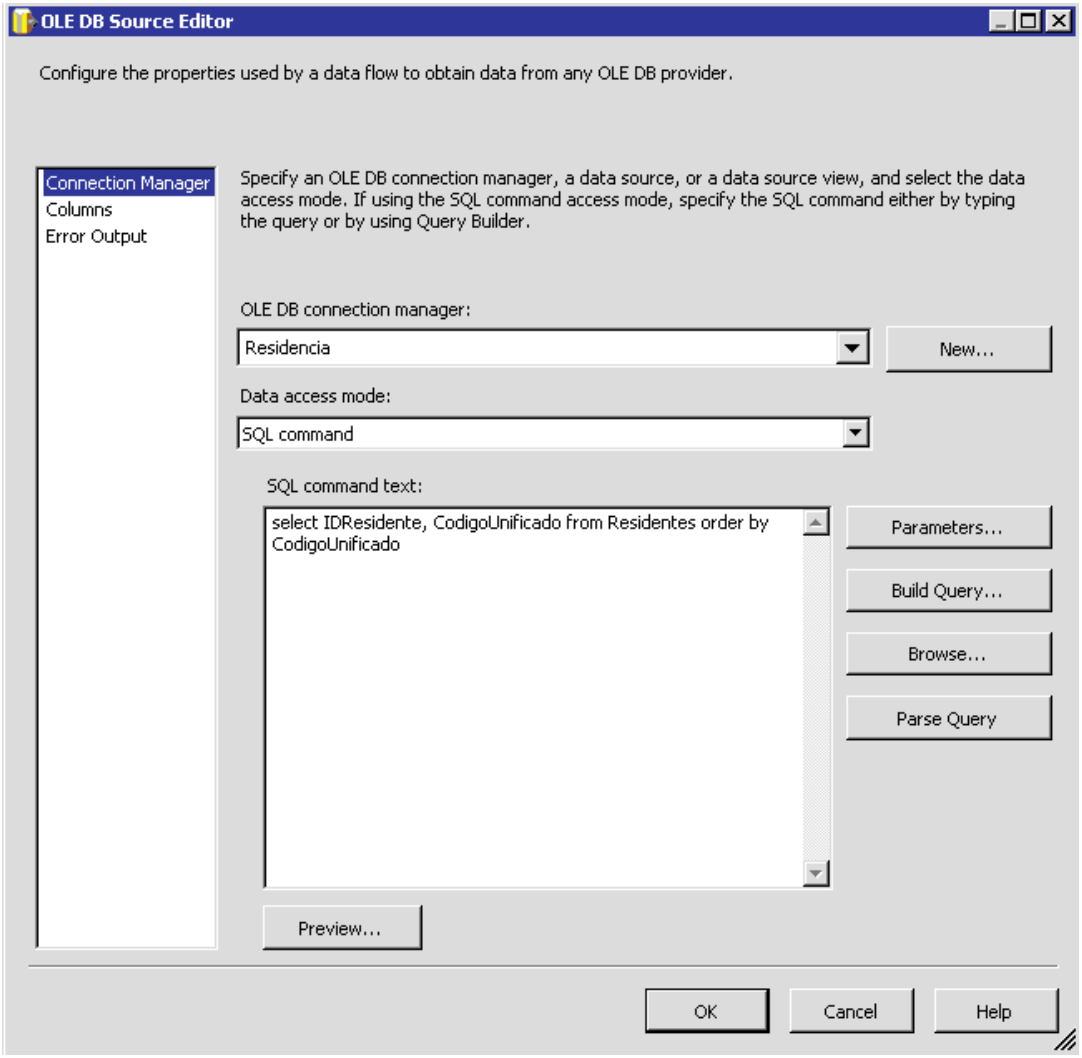


Figura 6.25. Ventana configuración OLE DB Residentes

En el caso del orígen de datos que accede al almacén, la configuración es similar. Las únicas diferencias son la conexión, en este caso será Almacen, y el nombre de la tabla de la consulta, que será Dim_Residente.

Al igual que hicimos anteriormente, ordenamos ambos conjuntos de datos por el campo CodigoUnificado para poder hacer la unión. Posteriormente, añadimos una transformación de **Combinación de mezcla** (Merge Join) y la configuramos tal y como indica la Figura 6.26.

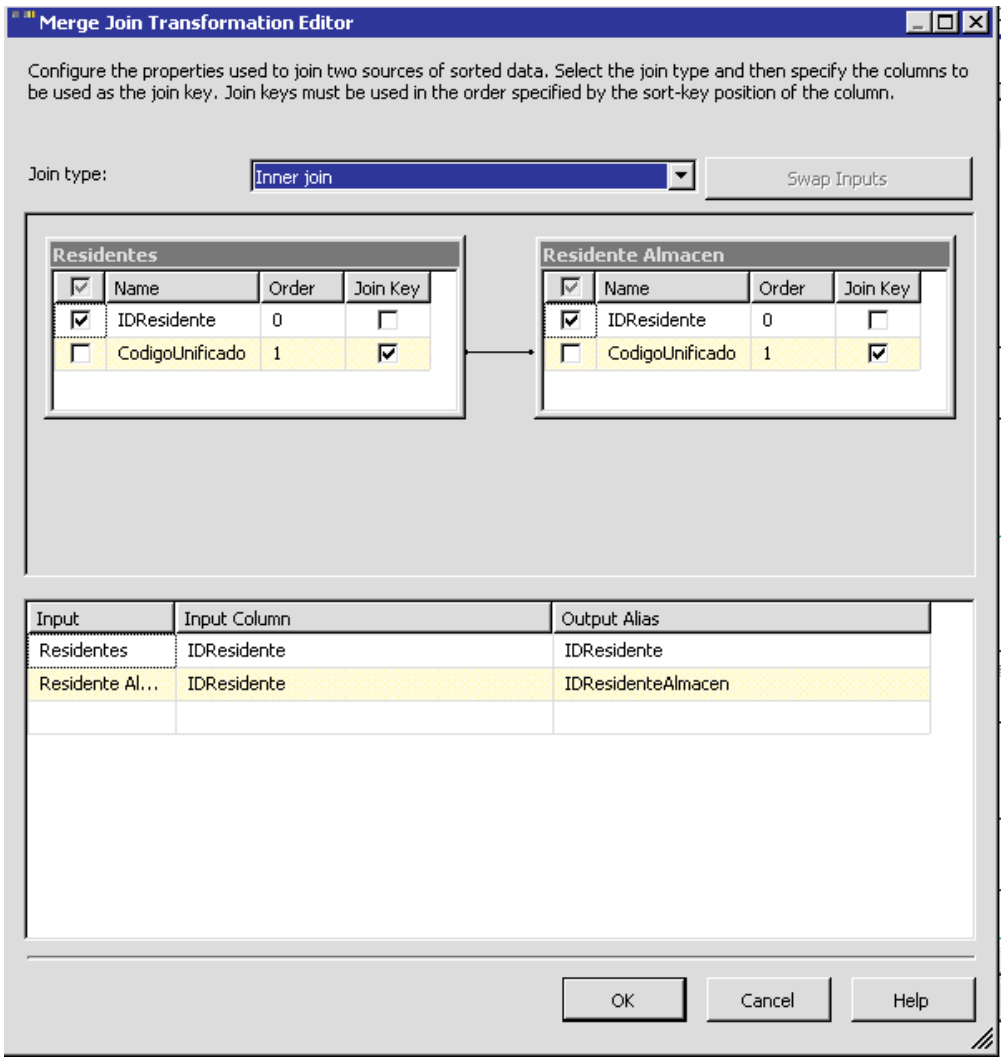


Figura 6.26. Configuración Merge Join Residentes

Nos quedamos con los residentes que aparezcan en ambas tablas (inner join) porque no queremos tener IDs nulos. Es muy importante renombrar las salidas, ya que ambos campos se llaman igual y no tenemos forma de diferenciarlos.

Ahora ya tenemos la correspondencia existente entre ambos y lo que tenemos que hacer es utilizar otra combinación de mezcla para unir estos datos con los datos de las tipologías. A partir de este momento, ya no es necesario pasar a través del flujo de datos el IDResidente de la conexión Residencia.

El siguiente paso es obtener el IDTipologia del almacén de datos que se corresponde con el IDTipologia de los datos obtenidos de la residencia.Para llevarlo a cabo, añadimos nuevas transformaciones y repetimos las acciones realizadas con los residentes.

La primera parte del flujo de datos acaba con dos conjuntos ordenados de datos. Por una parte tenemos la información de las tipologías procedente de la base de datos operacional enlazada con el IDResidente del almacén correspondiente y por la otra tenemos la correspondencia entre los IDs de las tipologías.

La segunda parte del flujo de datos es la que muestra la Figura 6.27.

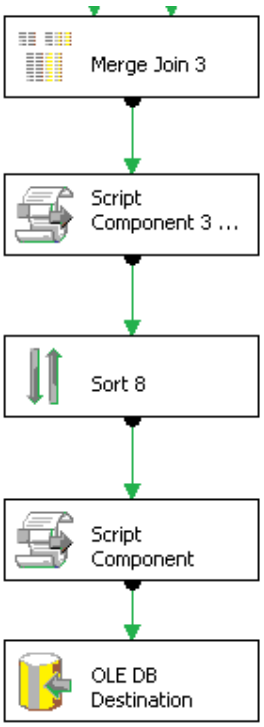


Figura 6.27. Parte 2 del flujo de datos asociado a la tarea Tipologia

Lo primero que se hace es añadir una transformación de **Combinación de mezcla** (Merge Join) para unir ambos conjuntos de datos utilizando el campo IDTipologíaPlaza.

Los datos de los que disponemos nos indican el momento en que los residentes cambian de tipología. Sin embargo, nosotros lo que queremos saber es la tipología de los residentes en cualquier momento y por tanto, necesitamos hacer un script que introduzca registros entre los cambios de tipologías y desde el último cambio de tipología hasta la fecha actual.

De esta forma tendremos un registro para cada día para cada residente con la tipología que tiene el residente en esa fecha.

Dicho esto, añadimos un Componente de script y configuramos sus salidas para que sean las que se indican en la Figura 6.28.

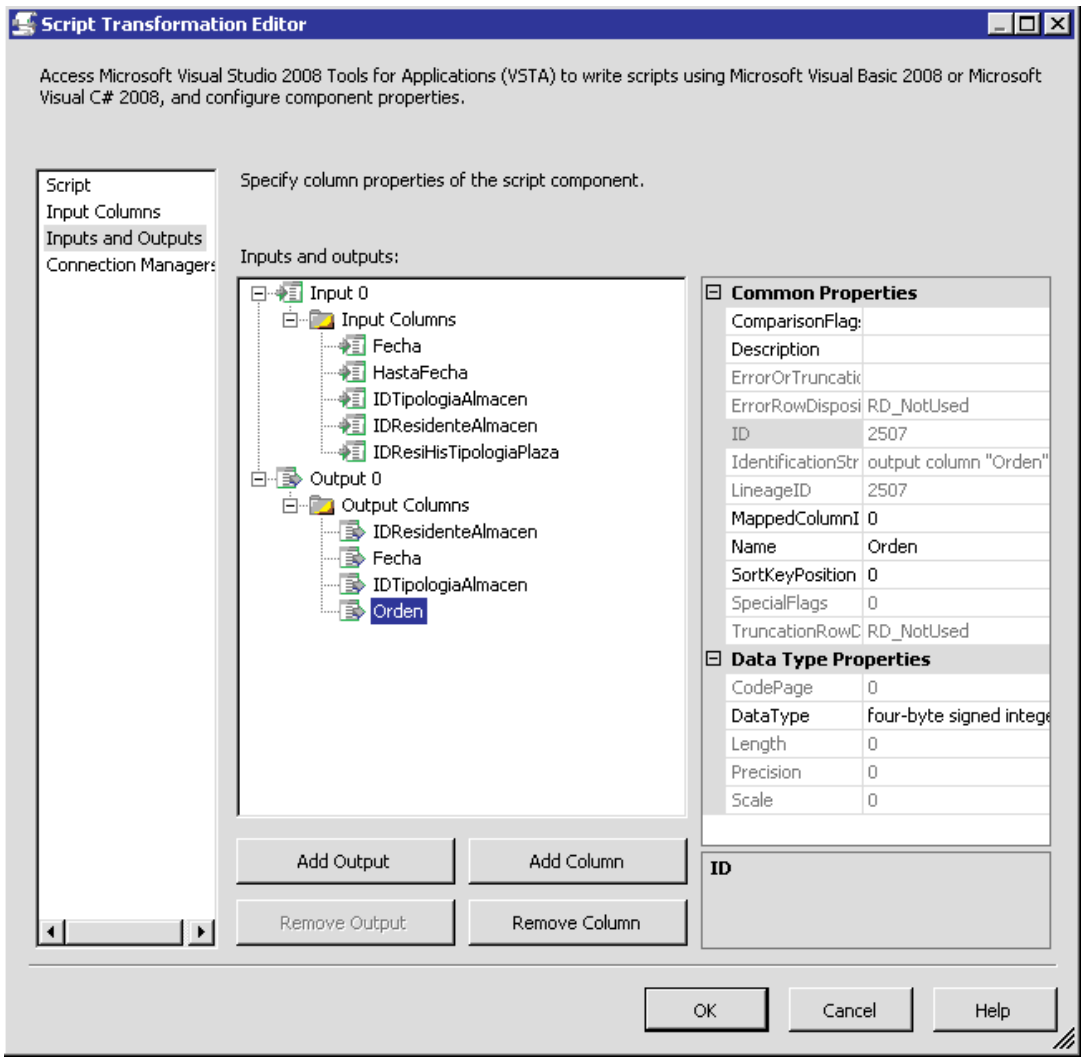


Figura 6.28. Salidas del componente de script

El campo Orden lo utilizaremos cuando se dé el caso de que haya un residente que tenga dos tipologías para un mismo día. En este caso, nos quedaremos con el registro cuyo campo Orden sea mayor, ya que implica que fue introducido posteriormente y por tanto, es la tipología más actual.

El contenido del script para realizar el procesamiento requerido es el siguiente. Se indica sólo el método de procesamiento de las filas de entrada porque es el único que tiene código.

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    fechaactual = Row.Fecha
    If Row.HastaFecha_IsNull Then
       fechafin = Date.Today
    Else
       fechafin = Row.HastaFecha
    End If
    While fechaactual <= fechafin
```

```
        Me.Output0Buffer.AddRow()
        Me.Output0Buffer.IDResidenteAlmacen = Row.IDResidenteAlmacen
        Me.Output0Buffer.Fecha = fechaactual
        Me.Output0Buffer.Orden = Row.IDResiHisTipologiaPlaza
        If Row.IDTipologiaAlmacen_IsNull Then
        Else
           Me.Output0Buffer.IDTipologiaAlmacen=Row.IDTipologiaAlmacen
        End If
        fechaactual = fechaactual.AddDays(1)
    End While
End Sub
```

Posteriormente necesitamos otro script para tratar el problema de las dos tipologías comentado anteriormente. Este script precisa que el conjunto de datos esté ordenado tal y como se indica en la Figura 6.29.

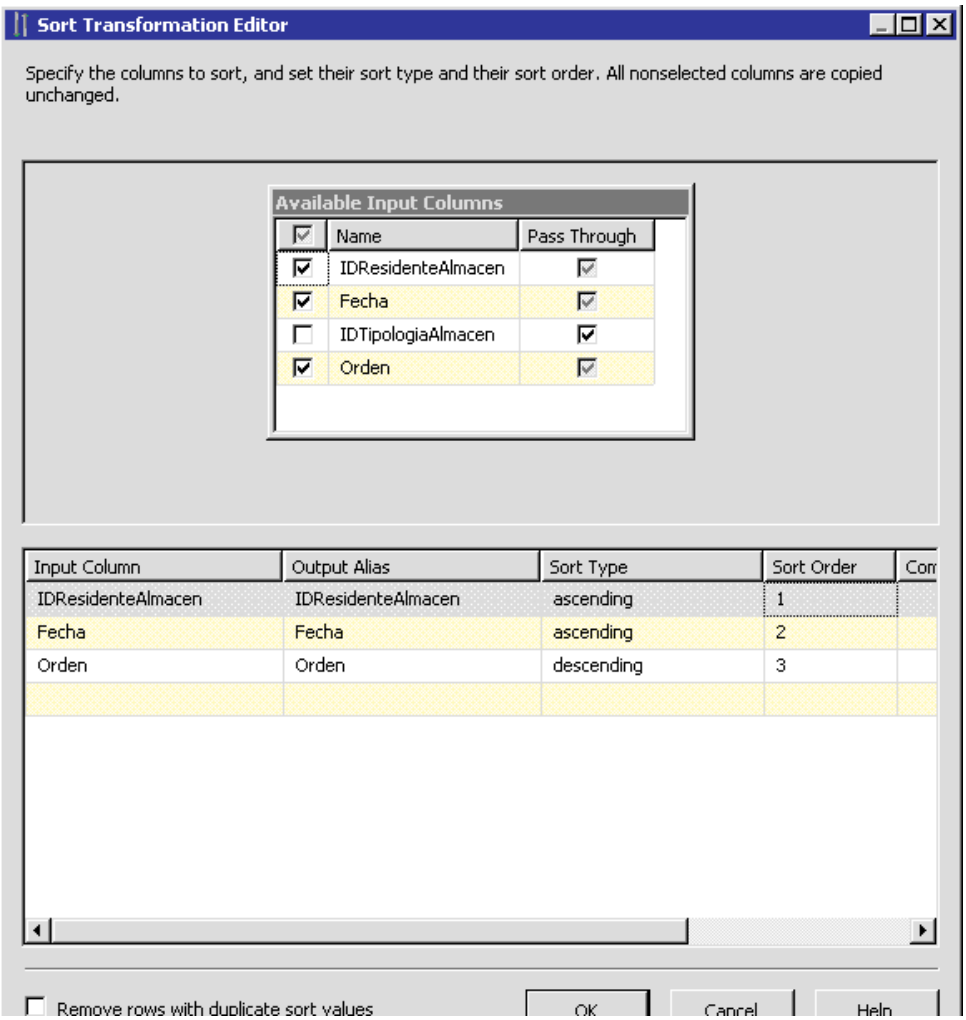


Figura 6.29. Columnas ordenación de la transformación Sort 8

Al estar ordenado por la columna Orden descendentemente, lo que queremos es quedarnos con el registro de mayor valor para dicha columna y eliminar el de menor valor para una misma fecha.

El código de este script sería:

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    If Row.IDResidenteAlmacen = ResidenteAnterior And Row.Fecha =
FechaAnterior Then
        Row.NextRow()
    Else
        Me.Output0Buffer.AddRow()
        Me.Output0Buffer.Fecha = Row.Fecha
        Me.Output0Buffer.IDResidenteAlmacen = Row.IDResidenteAlmacen
        If Row.IDTipologiaAlmacen_IsNull Then
        Else
            Me.Output0Buffer.IDTipologiaAlmacen=Row.IDTipologiaAlmacen
        End If
        ResidenteAnterior = Row.IDResidenteAlmacen
        FechaAnterior = Row.Fecha
    End If
End Sub
```

Para finalizar el flujo de datos, añadimos un destino con la misma configuración que siempre, seleccionamos la tabla que vamos a rellenar (Tipologias), hacemos clic en **Asignaciones** y hacemos corresponder las columnas de entrada con las columnas de destino correspondientes.

Una vez creados los dos flujos de datos que nos faltan, el desarrollo de este paquete habrá finalizado.

Paquete HechoUlcerasCuras

El paquete HechoUlcerasCuras es el encargado de obtener los datos relacionados con las curas de las úlceras de la base de datos operacional y rellenar la tabla de hechos del indicador de Ulceras Curas. Se ejecuta tras el paquete Dimension y tantas veces como residencias queremos utilizar para los resultados del indicador, es decir, si queremos lanzar el indicador para dos centros, debemos lanzar este paquete para cada uno de los centros, previo cambio de la configuración de la conexión Residencia.

Este indicador estaba incluido en el grupo de los indicadores que calculaban totales en función de varias dimensiones y por ello, la dificultad del mismo no radica en el procesamiento a realizar con los datos, sino en el elevado número de dimensiones con las que tenemos que enlazar los datos.

Lo primero que tenemos que hacer ir al panel de **Administradores de conexión**, hacer clic en **Nueva conexión desde origen de datos** y añadir los siguientes orígenes de datos: Almacen y Residencia.

Cuando tengamos las conexiones configuradas, necesitamos crear una variable llamada IDCentro para poder utilizarla en las diferentes tareas de flujo de datos que vamos a añadir al flujo de control del paquete. En ella guardaremos el valor del IDCentro almacenado en el almacén de datos correspondiente con el IDCentro para el que estamos lanzando el paquete.

Para construir el flujo de control utilizaremos dos tareas de flujo de datos, colocadas tal y como indica la Figura 6.30.

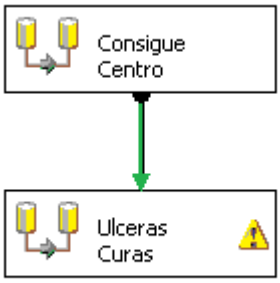


Figura 6.30. Flujo control del indicador UlcerasCuras

La primera de ellas se encargará de dotar de valor a la variable creada previamente y la segunda será la encargada de cargar los datos del indicador en la tabla de hechos.

A continuación se detalla el flujo de datos (Figura 6.31.) de la primera de las tareas.

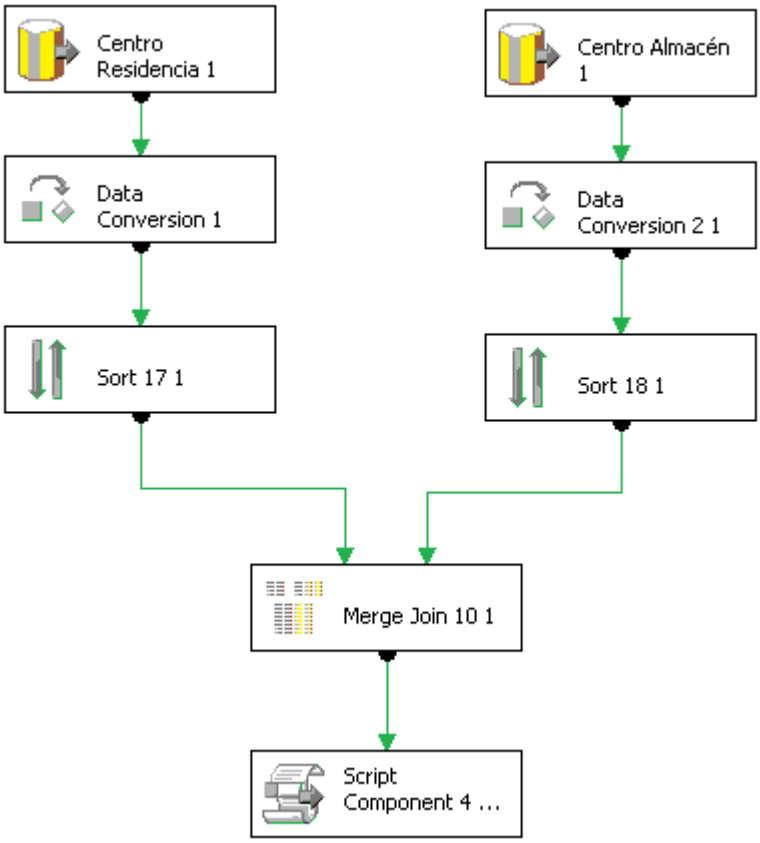


Figura 6.31. Flujo de datos asociado a la tarea Consigue Centro

Lo primero que necesitamos es arrastrar a la superficie de diseño del flujo de datos un par de orígenes de datos que accedan a la tabla DatosResidencia y Dim_Centro respectivamente para obtener los datos del centro. Concretamente, la configuración del origen de datos que accede a la base de datos operacional es la que aparece en la Figura 6.32.

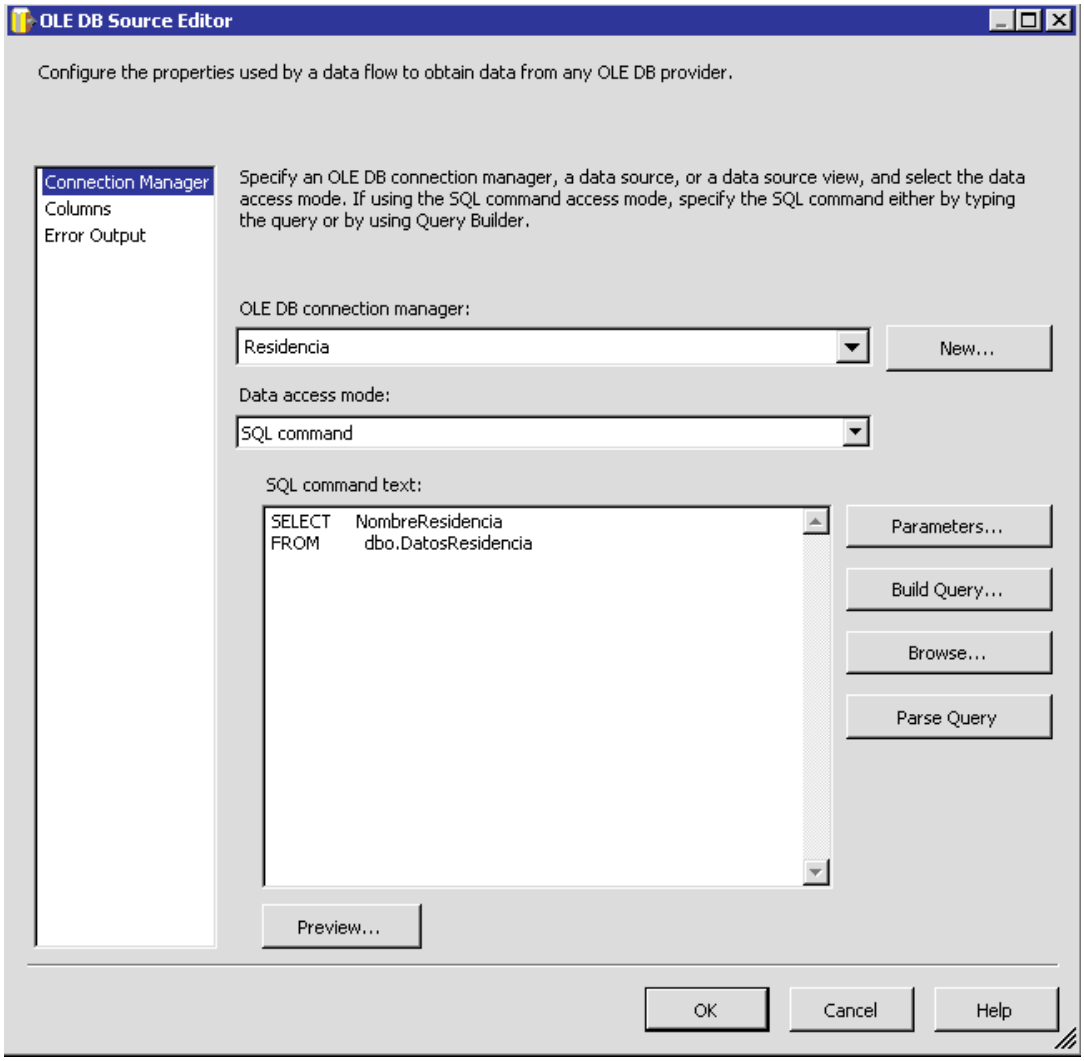


Figura 6.32. Configuración Origen OLE DB Centro Residencia

Después de configurar los orígenes de datos, lo siguiente es arrastrar un par de transformaciones de **Conversión de datos** para cambiar el tipo de datos de los campos NombreResidencia y Nombre al tipo Unicode String [DT_WSTR]. Esto es necesario porque esos campos necesitan tener el mismo tipo de datos para poder utilizarlos en la transformación de combinación de mezcla encargada de obtener el IDCentro del almacén de datos.

Tras la conversión de datos, ordenamos ambos conjuntos de datos por el campo resultante de la conversión y añadimos la transformación de combinación de mezcla. En dicha transformación, el campo por el que se unen ambos conjuntos de datos es el resultante de la conversión de datos.

Una vez tenemos el valor del IDCentro del almacén de datos, añadimos un Componente de script para almacenar dicho valor en la variable IDCentro.

El script de dicho componente es el siguiente:

```
Public Class ScriptMain
    Inherits UserComponent
    Dim valor As Integer
    Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
```

```
        Me.Output0Buffer.AddRow()
        valor = Row.IDCentro
        Me.Output0Buffer.IDCentro = Row.IDCentro
    End Sub
    Public Overrides Sub PostExecute()
        Me.Variables.IDCentro = valor //aquí rellenamos la variable
    End Sub
End Class
```

Ahora construiremos el flujo de datos de la segunda tarea. Dado que el flujo a construir es bastante grande, iremos construyendolo por partes pequeñas.

El flujo de datos comienza tal y como indica la Figura 6.33.

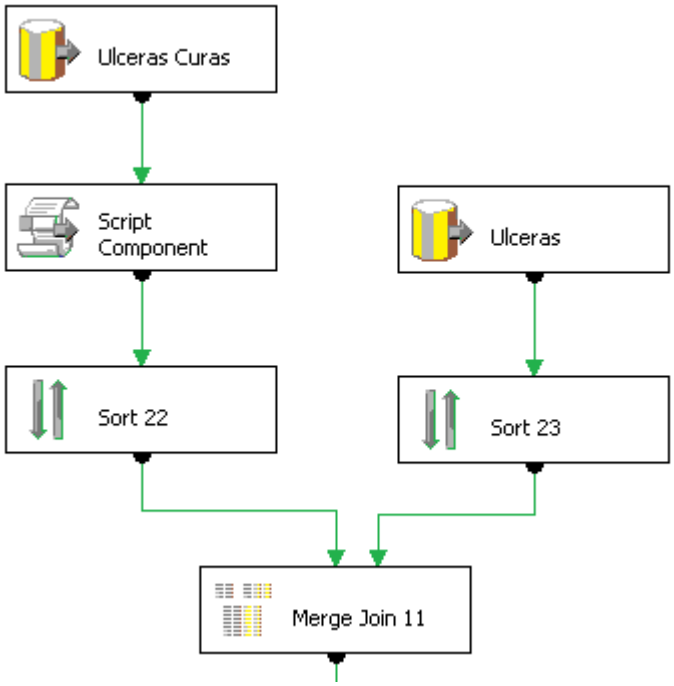


Figura 6.33. Parte 1 del flujo de datos asociado a la tarea Ulceras Curas

Cabe recordar que la tabla de hechos del indicador se compone de los siguientes campos: IDUlceras, IDTiempo, IDResidente, IDCaracterPlaza, IDTipologia, IDCentro, IDCurasLocalizacion, IDUlcerasTipos, IDUlcerasProcedencias, IDCurasEstadio, IDUlceraCurada y TamañoUlcera.

Como toda esta información no se encuentra en una única tabla, debemos añadir tantos orígenes de datos como tablas en las que se encuentra repartida esa información. En nuestro caso, esa información se encuentra en las tablas UlcerasCuras y Ulceras de la base de datos operacional.

Las consultas a introducir en dichos orígenes son las siguientes:

- UlcerasCuras

```
SELECT    IDUlcera, FechaHora, IDEstadio, Tamano, IDResidente
```

```
FROM        dbo.UlcerasCuras
```

- Ulceras

```
SELECT      IDUlcera, IDUlceraProcedencia, IDUlceraTipo, IDLocalizacion,
CodigoUnificado, Curado
FROM        dbo.Ulceras
```

El resto de información la obtendremos enlazando con las tablas auxiliares.

En la tabla de UlcerasCuras las curas llevan asociada una fecha y una hora. La hora carece de interés y además si no la eliminamos, no podremos enlazar los datos con la dimensión Tiempo. Por tanto, la siguiente transformación que se requiere es un componente de script que elimine la hora de los registros y deje únicamente la fecha. Además, dicho script ha de incorporar tratamiento de nulos para ser del todo correcto.

El código que se necesita para eliminar las horas de un registro es el que aparece a continuación:

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    fechaactual = Row.FechaHora
    hora = Row.FechaHora.Hour * (-1)
    Min = Row.FechaHora.Minute * (-1)
    seg = Row.FechaHora.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
    If Row.IDEstadio_IsNull Then
    Else
       Me.Output0Buffer.IDEstadio = Row.IDEstadio
    End If
    Me.Output0Buffer.IDResidente = Row.IDResidente
    Me.Output0Buffer.IDUlcera = Row.IDUlcera
    If Row.Tamano_IsNull Then
    Else
       Me.Output0Buffer.Tamano = Row.Tamano
```

```
        End If
    End Sub
```

Lo que hacemos en el script es sumarle a la fecha que tenemos, la misma hora pero con valor negativo para que la hora se quede a cero.

El siguiente paso es ordenar ambos conjuntos de datos por el campo por el que los vamos a unir (IDUlcera) y añadir una transformación de combinación de mezcla para unir ambos conjuntos de datos.

La Figura 6.34. muestra la siguiente parte del flujo de datos.

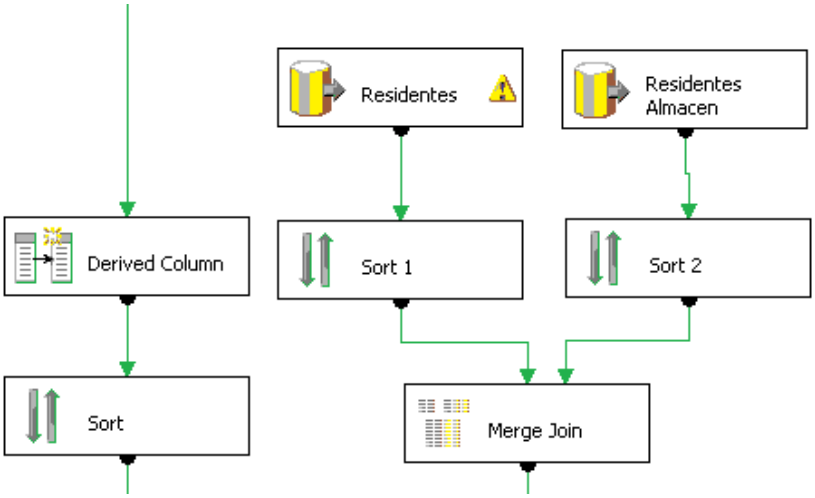


Figura 6.34. Parte 2 del flujo de datos asociado a la tarea Ulceras Curas

Añadimos una transformación de columna derivada (Figura 6.35.) para añadir al conjunto de datos el IDCentro del almacén usando la variable con el mismo nombre.

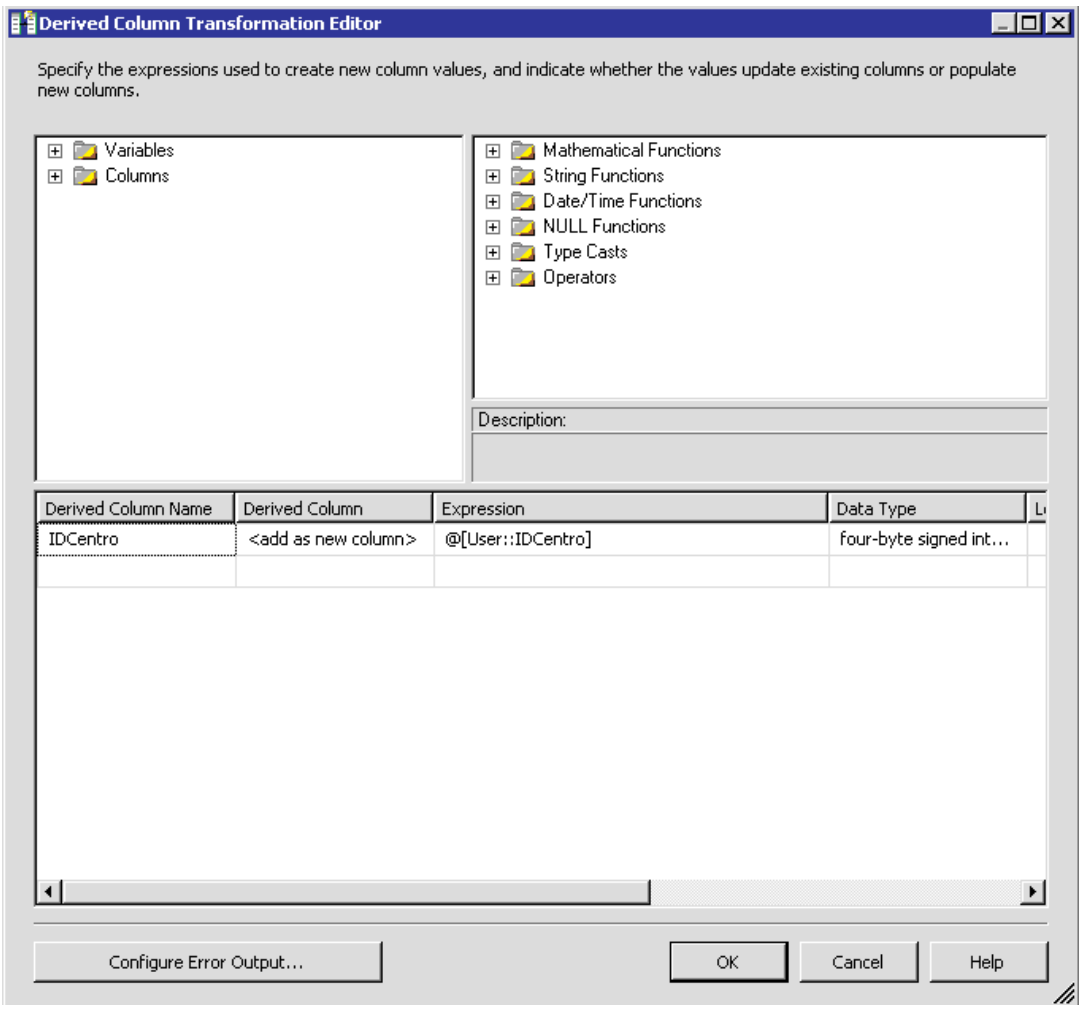


Figura 6.35. Configuración transformación Columna Derivada

En este momento tenemos toda la información necesaria y lo que falta es obtener los IDs del almacén que se corresponden con los de la base de datos operacional, los cuales se obtienen todos de la misma manera.

En primer lugar obtenemos el IDResidente del almacén de datos que se corresponde con el IDResidente de los datos obtenidos tal y como se indicó anteriormente, es decir, obtenemos el par IDResidente-IDResidenteAlmacen y lo enlazamos con la datos que tenemos para quedarnos solamente con el IDResidenteAlmacen.

La siguiente parte del flujo de datos es la que muestra la Figura 6.36.

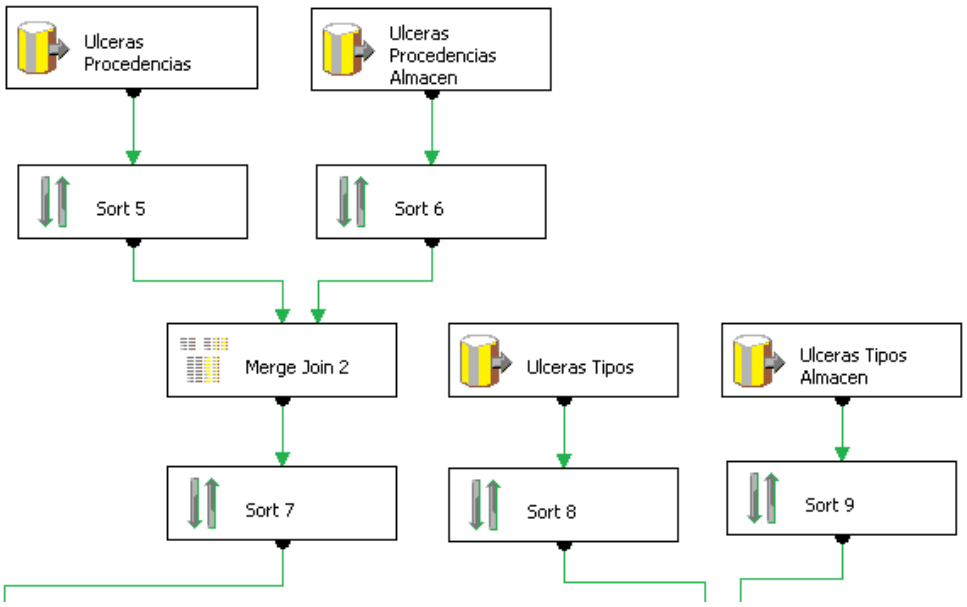


Figura 6.36. Parte 3 del flujo de datos asociado a la tarea Ulceras Curas

En esta parte lo que se trata es de obtener el IDUlcerasProcedencias y el IDUlcerasTipos del almacén de datos que se corresponden con sus homónimos de los datos que llevamos hasta el momento.

La Figura 6.37. muestra la parte que le sigue a la anterior.

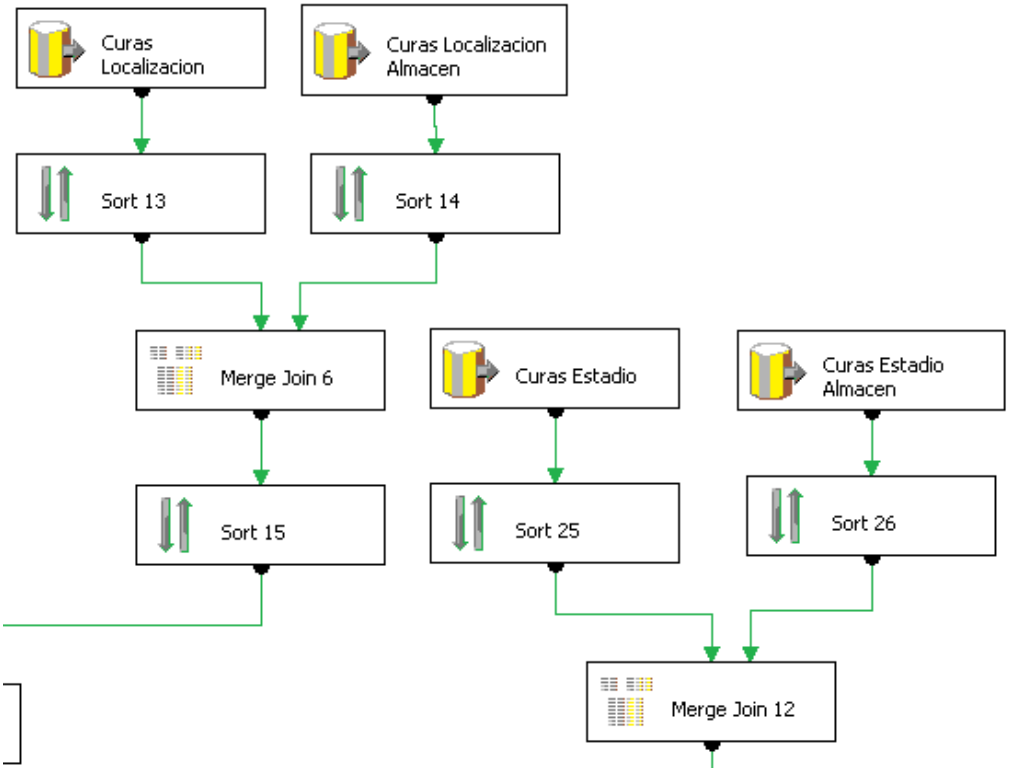


Figura 6.37. Parte 4 del flujo de datos asociado a la tarea Ulceras Curas

Aquí lo que queremos es obtener el IDCurasLocalizacion y el IDCurasEstadio del almacén de datos que se corresponden con el IDLocalizacion y el IDEstadio de los datos obtenidos.

A continuación nos encontramos con la parte que indica la Figura 6.38.

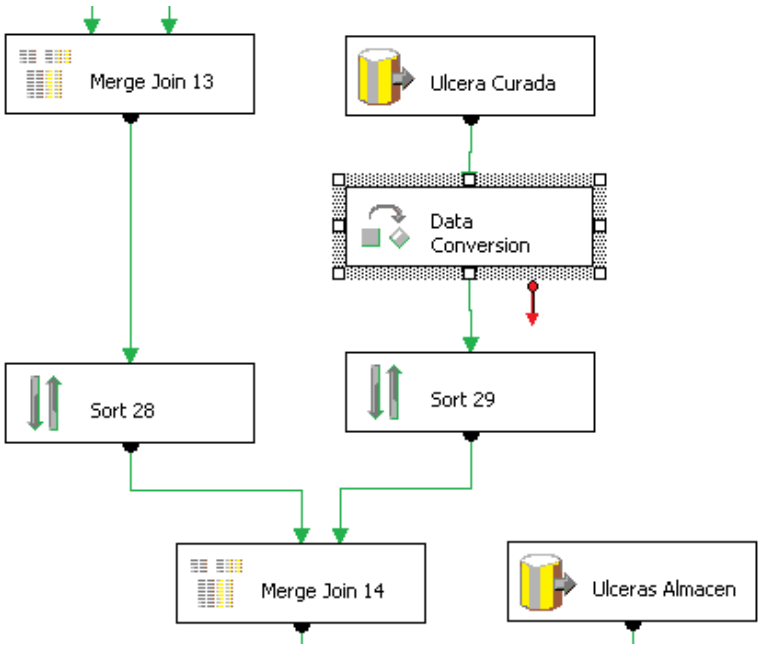


Figura 6.38. Parte 5 del flujo de datos asociado a la tarea Ulceras Curas

En lo que respecta a la dimensión Ulcera Curada, los campos a comparar son UlceraCurada (Dim_UlceraCurada) y Curado (flujo de datos izquierdo). Como el campo Curado es de tipo booleano y el otro es de tipo entero, debemos hacer una conversión del tipo de UlceraCurada para pasarlo a booleano y poder hacer la unión de datos. El resultado de dicha unión será el IDUlceraCurada del almacén de datos.

La siguiente parte es la mostrada en la Figura 6.39.

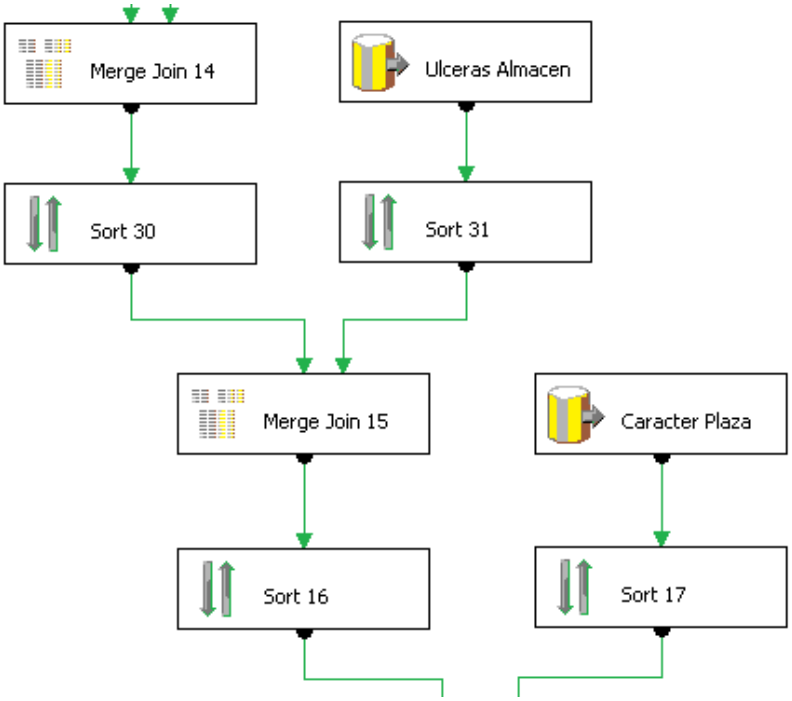


Figura 6.39. Parte 6 del flujo de datos asociado a la tarea Ulceras Curas

Aquí lo que queremos es obtener el IDUlceras del almacén de datos que se corresponde con el IDUlceras de los datos obtenidos utilizando el campo CodigoUnificado. Además, también se obtiene el IDCaracterPlaza que tienen asociados los residentes en la fecha de

las curas enlazando con la tabla auxiliar CaracterPlaza. En este caso, para enlazar con la tabla auxiliar utilizamos los campos IDResidente y Fecha. De utilizar sólo uno de ellos, no se podría obtener el IDCaracterPlaza.

La penúltima parte del flujo de datos se muestra en la Figura 6.40.

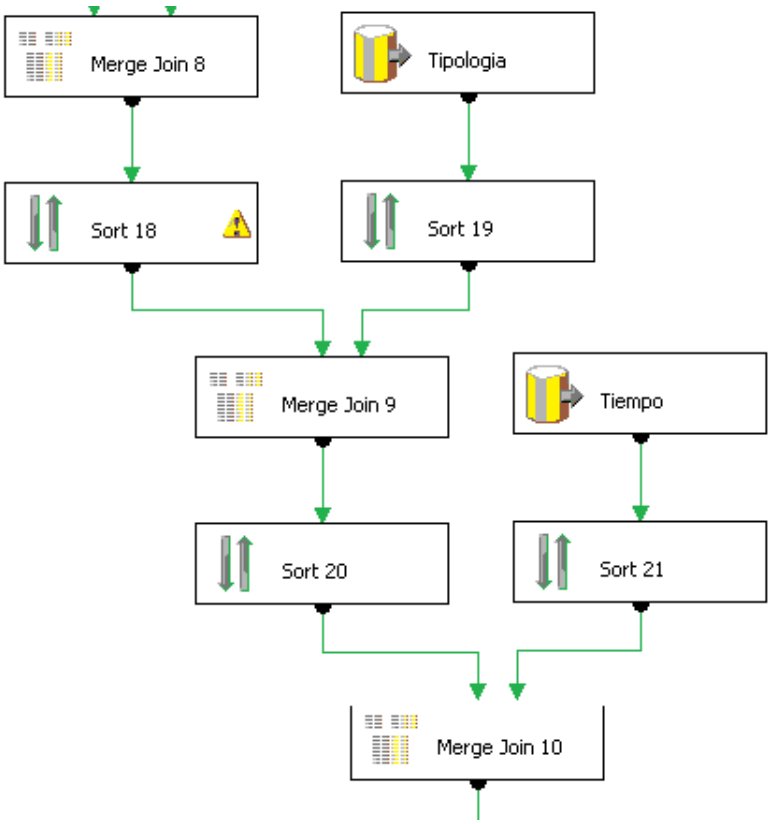


Figura 6.40. Parte 7 del flujo de datos asociado a la tarea Ulceras Curas

En esta parte se obtiene el IDTipología y el IDTiempo del almacén de datos que se corresponden con los IDs de los datos. En el caso de la tipología, utilizamos los campos IDResidente y Fecha para enlazar con la tabla auxiliar de tipologías. Sin embargo, para enlazar con la dimensión Tiempo sólo necesitamos el campo Fecha.

La última parte del flujo de datos es la que aparece en la Figura 6.41.

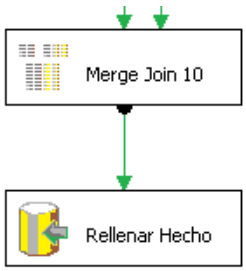


Figura 6.41. Parte 8 del flujo de datos asociado a la tarea Ulceras Curas

Una vez hemos enlazado con la dimensión Tiempo lo único que falta es añadir un destino de datos. Configuramos dicho destino como siempre, seleccionamos la tabla que vamos a rellenar (Fact_UlcerasCuras) y hacemos clic en **Asignaciones** (Figura 6.42.)**.**

Las asignaciones deben ser las siguientes:

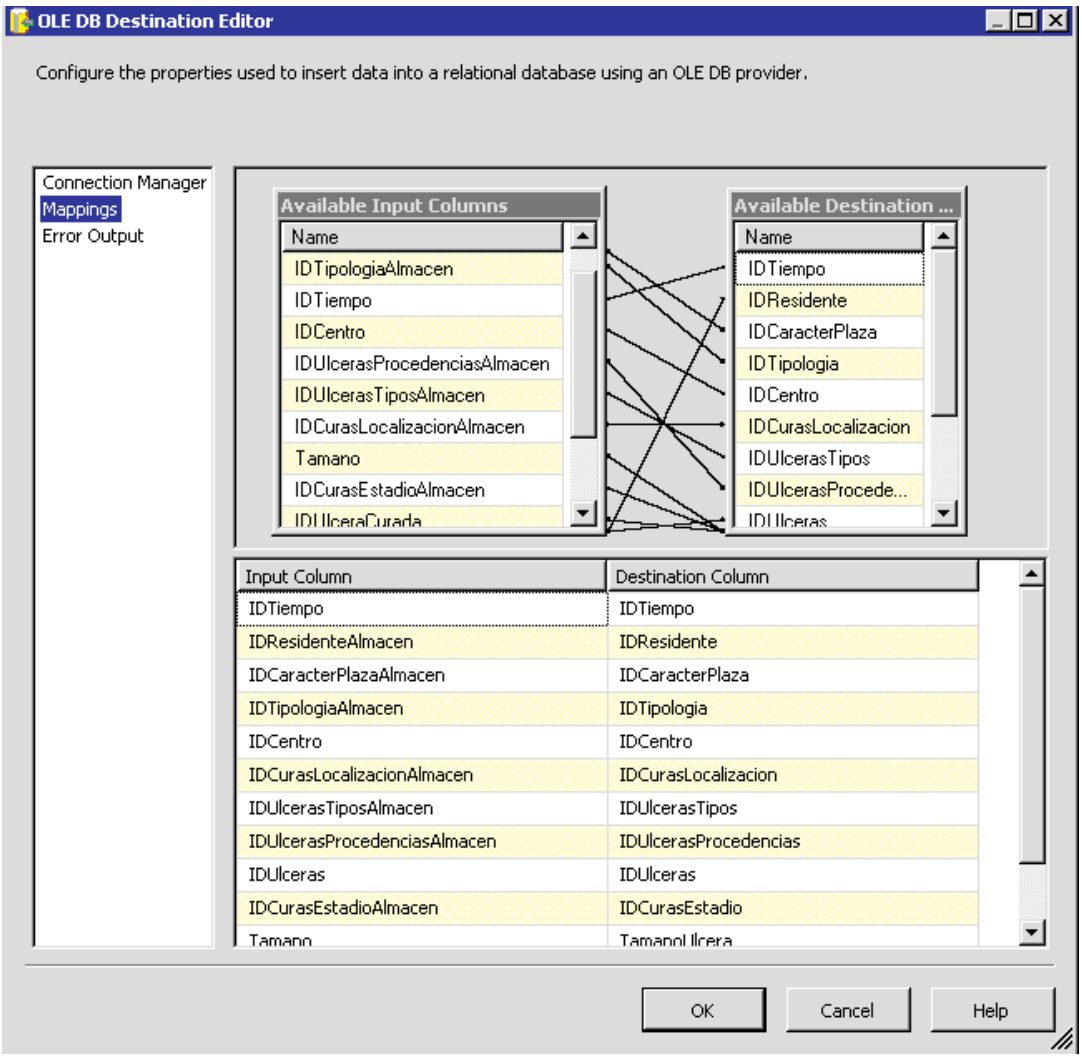


Figura 6.42. Ventana Asignaciones del Destino OLE DB Rellenar Hecho

Tras esto, el desarrollo de este paquete habrá finalizado.

Paquete HechoDiferenciaSeg

El paquete HechoDiferenciaSeg es el encargado de obtener los datos relacionados con los seguimientos profesionales de la base de datos operacional y rellenar la tabla de hechos del indicador de Seguimientos Profesionales. Se ejecuta tras el paquete Dimension y tantas veces como residencias queremos utilizar para los resultados del indicador, es decir, si queremos lanzar el indicador para dos centros, debemos lanzar este paquete para cada uno de los centros, previo cambio de la configuración de la conexión Residencia.

Este indicador estaba incluido en el grupo de los indicadores que calculaban la diferencia de días transcurridos entre dos eventos y por ello, la dificultad del mismo radica en la forma de calcular esa diferencia de días.

Lo primero que tenemos que hacer ir al panel de **Administradores de conexión**, hacer clic en **Nueva conexión desde origen de datos** y añadir los siguientes orígenes de datos: Almacen y Residencia.

Cuando tengamos las conexiones configuradas, necesitamos crear la variable IDCentro al igual que hicimos en el indicador anterior.

El flujo de control de este paquete será similar al del paquete anterior, ya que tenemos que realizar las mismas tareas: dotar de valor a la variable IDCentro y cargar los datos del indicador en la tabla de hechos.

Dado que el flujo de datos asociado a la primera tarea fue analizado en detalle en el paquete anterior, a continuación trataremos el flujo de datos asociado a la segunda tarea. Dicho flujo será parcialmente distinto al del otro indicador porque debemos realizar otros procesamientos por el tipo de indicador que es.

Cabe recordar que la tabla de hechos del indicador se compone de los siguientes campos: IDTiempo, IDResidente, IDCaracterPlaza, IDTipologia, IDCentro, IDTipoProfesional y TiempoTranscurrido.

Puesto que nos encontramos con que tiene dimensiones comunes con el otro indicador, sólo vamos a detallar las partes del flujo de datos que sean propias de dicho indicador.

El flujo de datos comienza tal y como se indica en la Figura 6.43.

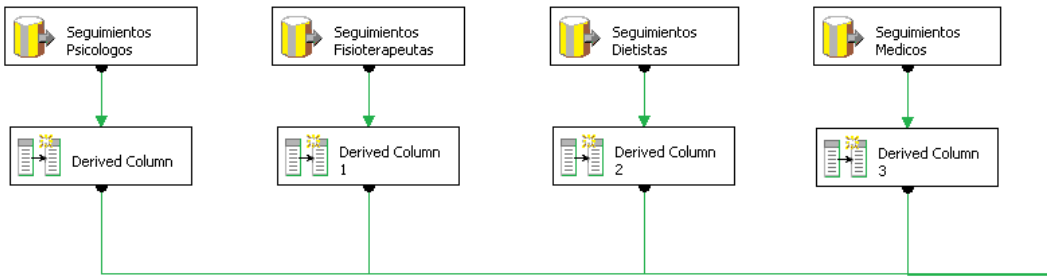


Figura 6.43. Parte 1 del flujo de datos del indicador Seguimientos Profesionales

El indicador calcula la diferencia de días transcurridos entre dos seguimientos de un mismo profesional para un mismo residente. Como calcula dicha diferencia de días para los seguimientos de cualquier profesional, necesitamos añadir tantos orígenes de datos como tablas diferentes de seguimientos profesionales existen.

Cada uno de estos orígenes es similar al que aparece en la Figura 6.44.: la conexión es la que accede a la base de datos operacional y la consulta sólo recupera el IDResidente y la FechaHora del seguimiento.

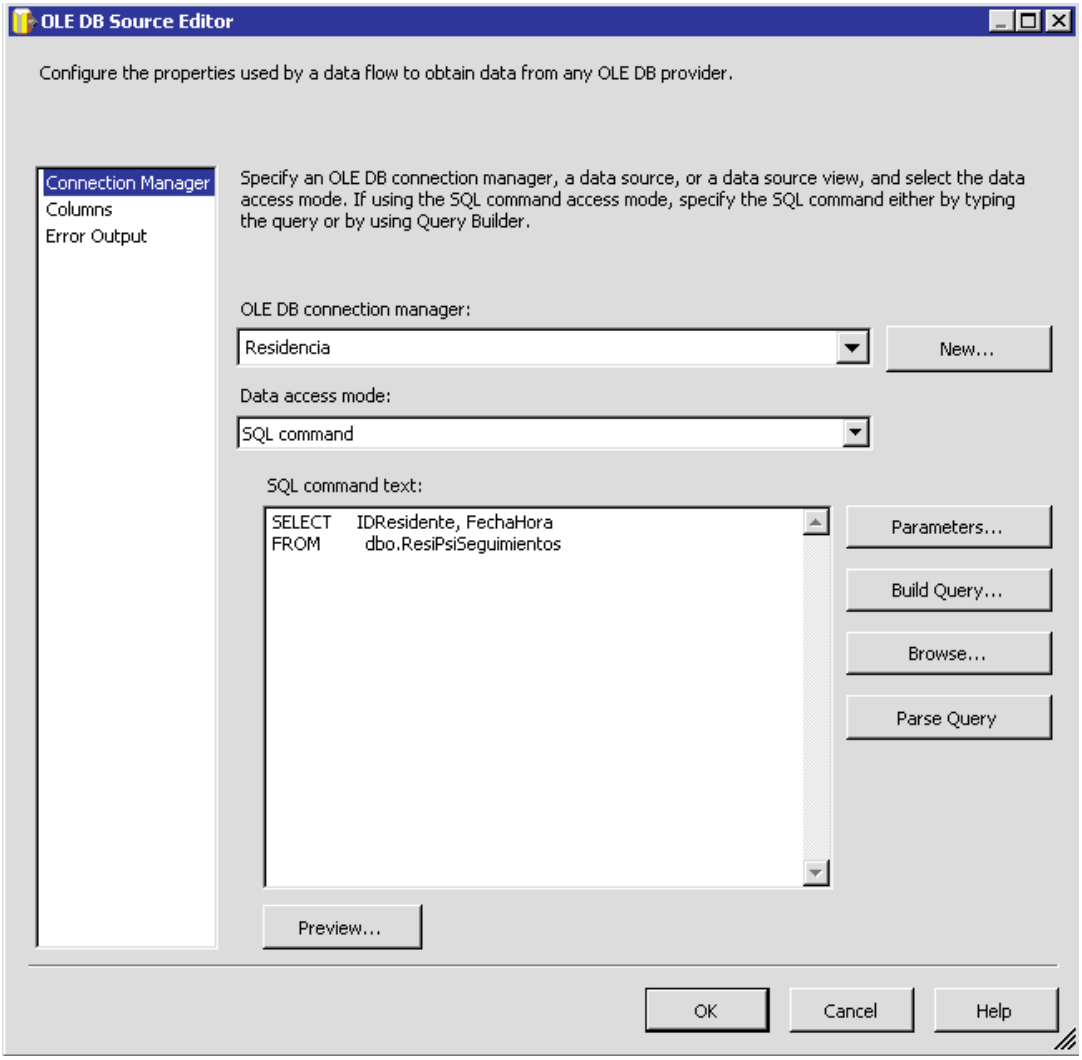


Figura 6.44. Configuración del Origen OLE DB Seguimientos Psicólogos

Tras configurar el origen de datos, añadimos una transformación de columna derivada (Figura 6.45.) para indicar el tipo de profesional asociado al seguimiento. El valor asignado a dicha columna se corresponde con el IDTipoProfesional del almacén de datos asociado a ese profesional.

Los posibles valores a introducir en dicha columna son los valores de la dimensión TipoProfesional, que son los siguientes:

1-Psicólogo	6-Trabajador social	11-Auxiliar
2-Fisioterapeuta	7-Animador social	12-Atención espiritual
3-Dietista	8-Terapia ocupacional	13-Pedagogo
4-Médico	9-Supervisor	14-Podólogo
5-Enfermera	10-Director	

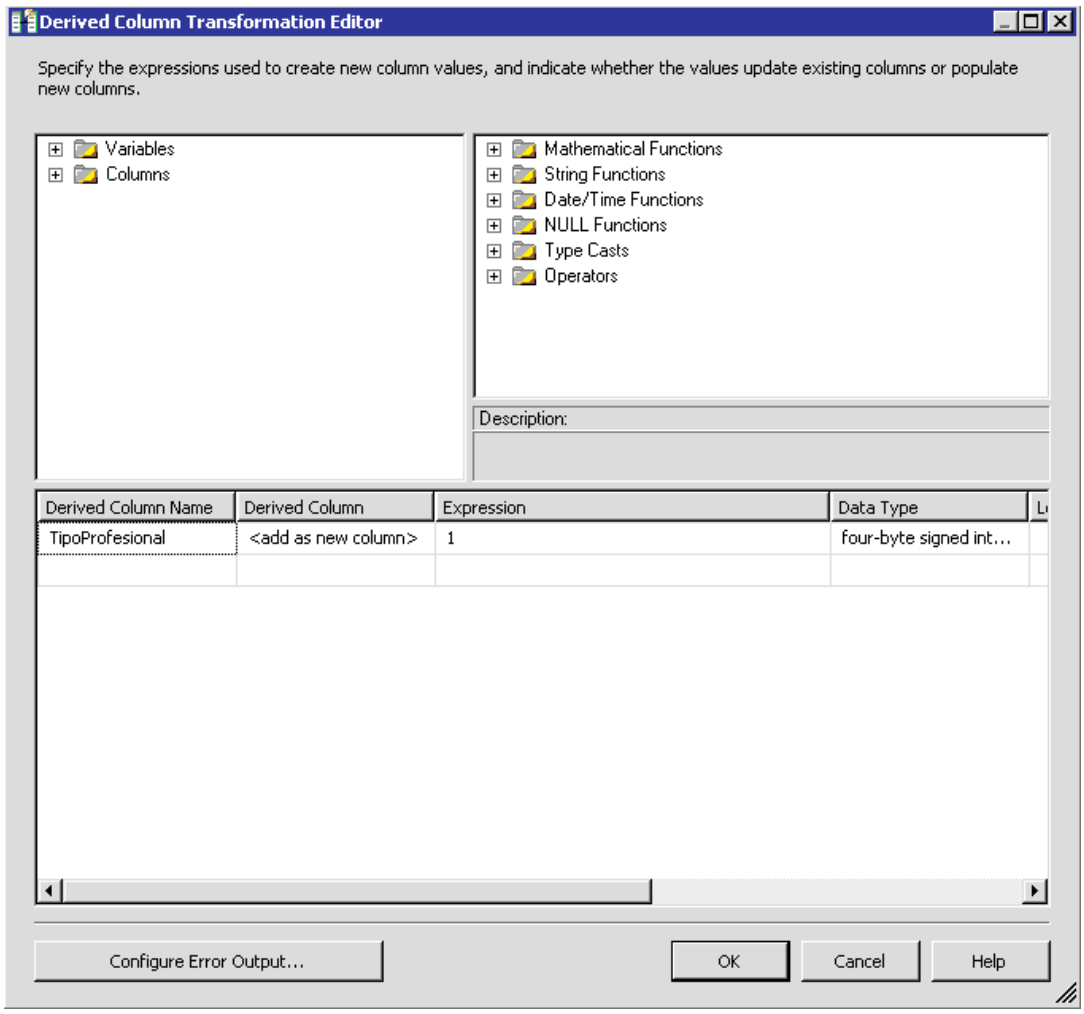


Figura 6.45. Configuración transformación Derived Column

El valor añadido es un 1 porque la tabla de la que hemos obtenido los datos es ResiPsiSeguimientos.

Hacemos lo mismo para el resto de tablas de seguimientos que aparecen en la imagen.

La segunda parte del flujo de datos es la que aparece en la Figura 6.46.

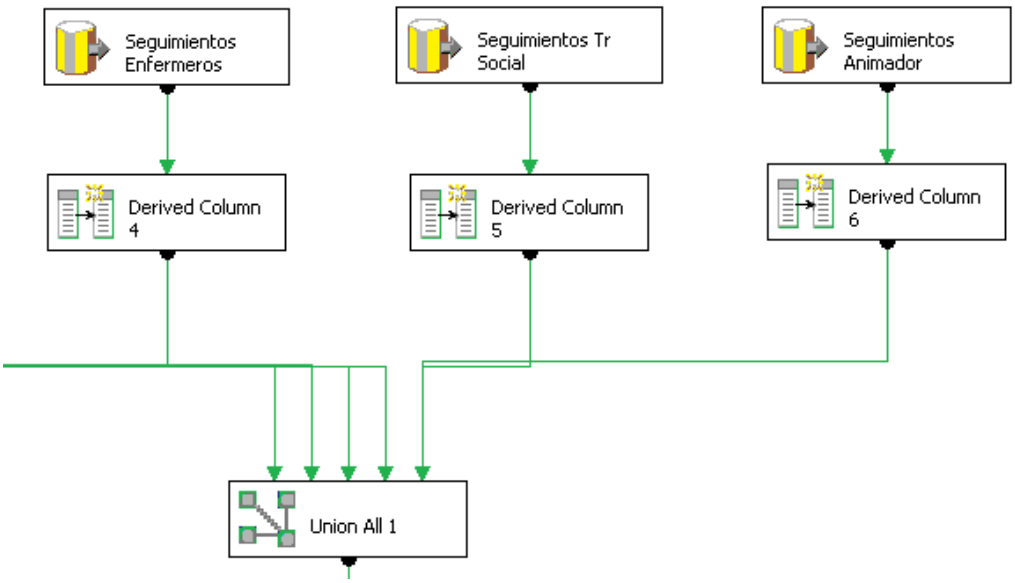


Figura 6.46. Parte 2 del flujo de datos del indicador Seguimientos Profesionales

Esta parte es similar a la anterior pero incluye una transformación Unión de todo para combinar todos los conjuntos de datos que tenemos hasta el momento. Como los campos se llaman de la misma forma en todas las tablas, las asociaciones entre columnas aparecen automáticamente tal y como se indica en la Figura 6.47.

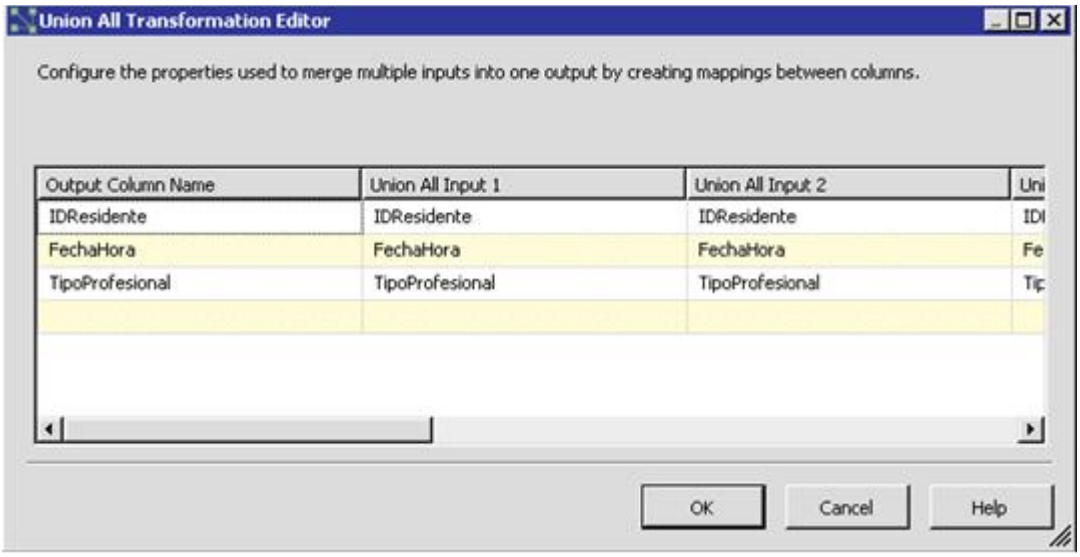


Figura 6.47. Configuración transformación Union All 1

Tras esto, repetimos el proceso para el resto de tablas de seguimientos que faltan por añadir al flujo de datos.

La Figura 6.48. muestra la siguiente parte del flujo de datos.

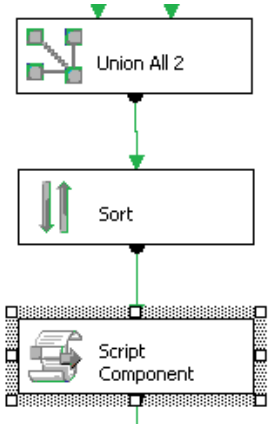


Figura 6.48. Parte 3 del flujo de datos del indicador Seguimientos Profesionales

La transformación **Unión de todo** la utilizamos para combinar todos los datos de las tablas de seguimientos que tenemos hasta el momento. La forma de combinarlos es la que se indicó anteriormente.

Como lo que queremos es calcular la diferencia de días transcurridos entre dos seguimientos de un mismo profesional para un mismo residente, debemos ordenar los datos por el TipoProfesional, IDResidente y FechaHora respectivamente.

Una vez los tengamos ordenados, añadimos un **Componente de script** cuyas entradas serán todos los campos que tenemos hasta ahora y cuyas salidas serán esos mismos campos y un campo adicional en el que guardaremos el tiempo transcurrido entre dos seguimientos.

El código del script sin incluir la declaración de las variables que utilizamos es el siguiente:

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    If contador <> 0 Then
        If Row.IDResidente = residente Then
            If Row.TipoProfesional = tipoProfesional Then
                If Row.FechaHora_IsNull Then
                Else
                    fecha2 = Row.FechaHora
                    Me.Output0Buffer.AddRow()
                    Me.Output0Buffer.IDResidente = Row.IDResidente
                    Me.Output0Buffer.Fecha = fecha2
                    Me.Output0Buffer.TipoProfesional=Row.TipoProfesional
                Me.Output0Buffer.TiempoTranscurrido = DateDiff(DateInterval.Day,
            fecha1, fecha2)
                    fecha1 = Row.FechaHora
                    residente = Row.IDResidente
                    tipoProfesional = Row.TipoProfesional
                End If
            Else
                If Row.FechaHora_IsNull Then
                Else
                    fecha1 = Row.FechaHora
                    residente = Row.IDResidente
                    tipoProfesional = Row.TipoProfesional
                End If
            End If
        Else
            If Row.FechaHora_IsNull Then
            Else
                fecha1 = Row.FechaHora
                residente = Row.IDResidente
                tipoProfesional = Row.TipoProfesional
            End If
```

```
            End If
        Else
            If Row.FechaHora_IsNull Then
            Else
                fecha1 = Row.FechaHora
                residente = Row.IDResidente
                tipoProfesional = Row.TipoProfesional
                contador += 1
            End If
        End If
    End Sub
```

El script funciona guardándose la información del primer seguimiento de un residente de un determinado tipo de profesional para calcular la diferencia de días con el siguiente seguimiento de ese profesional. Si tenemos ese segundo seguimiento, el script añade una fila y si no, pasamos al siguiente registro y comprobamos lo mismo. El hecho de no añadir filas siempre provoca que el número de filas resultantes tras el script sea inferior al número de filas entrantes.

La parte del flujo de datos que viene ahora es la que se indica en la Figura 6.49.

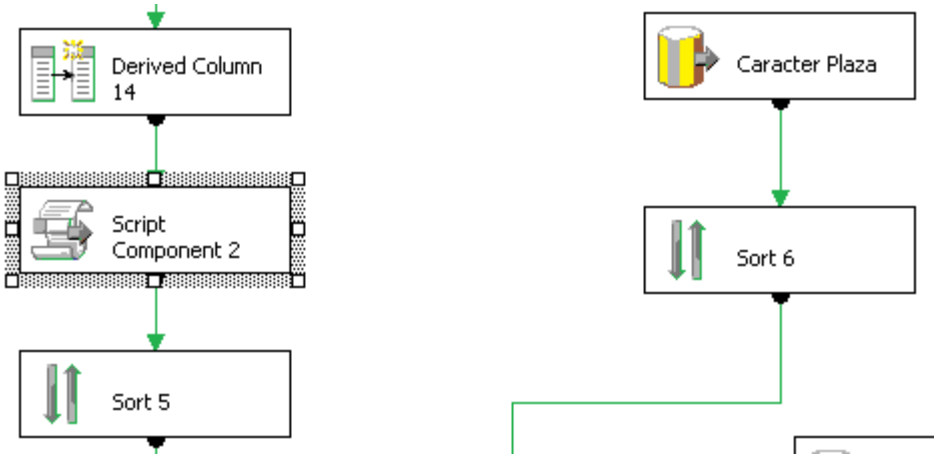


Figura 6.49. Parte 4 del flujo de datos del indicador Seguimientos Profesionales

Añadimos una transformación de columna derivada para añadir al conjunto de datos el IDCentro del almacén usando la variable con el mismo nombre. Además, añadimos otro componente de script para quitarle la hora a los seguimientos y poder enlazar posteriormente los datos con la dimensión Tiempo. El código utilizado para llevar a cabo dicho cambio es el mismo que se indicó en el indicador anterior.

Llegados a este momento, lo siguiente es enlazar con las tablas de dimensiones y las tablas auxiliares para obtener los siguientes IDs del almacén: IDResidente, IDTiempo, IDCaracterPlaza y IDTipologia. La forma de enlazar es la misma que se indicó en el paquete anterior.

El flujo de datos finaliza con la adición del destino de datos. La tabla destino es Fact_DiferenciaSeg y las asignaciones son las que aparecen en la Figura 6.50.

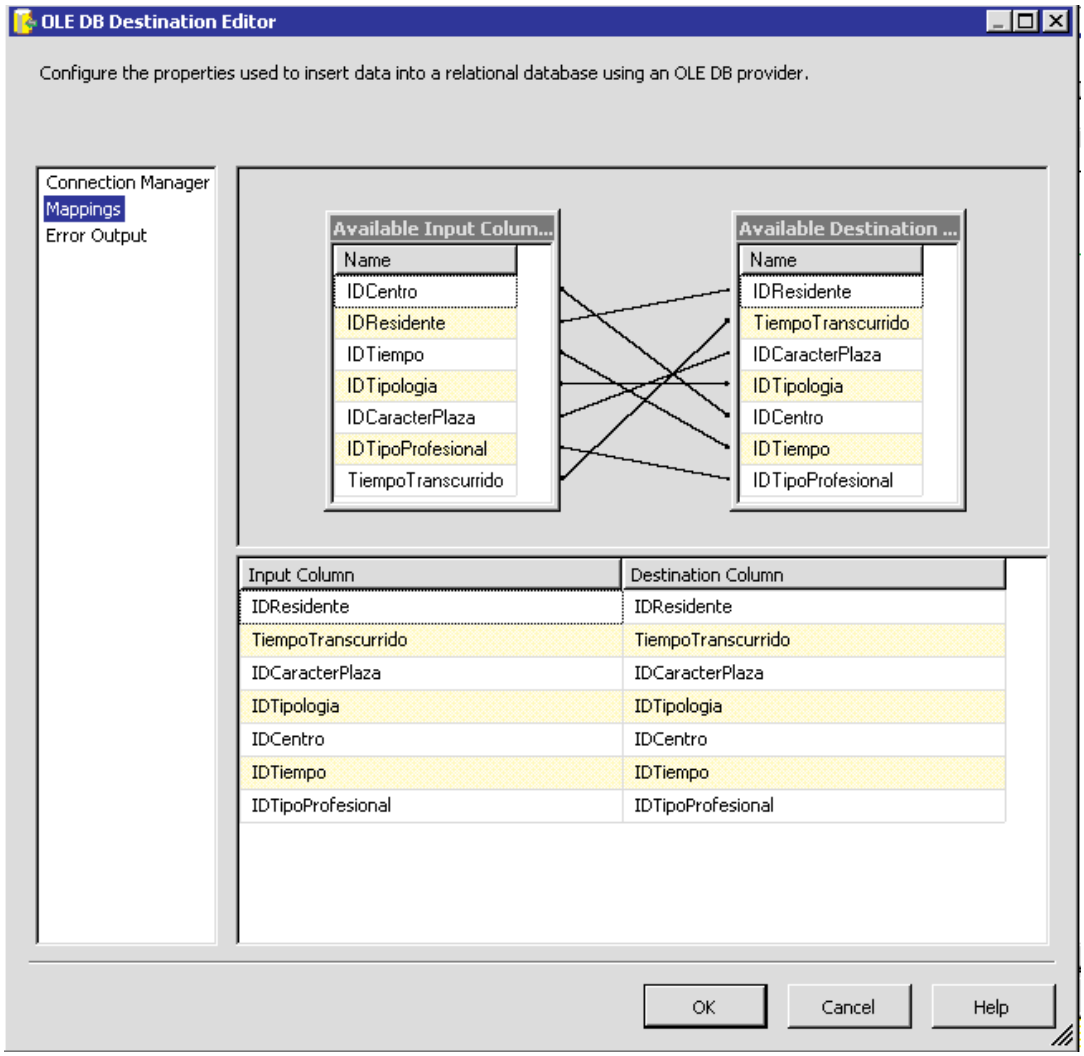


Figura 6.50. Ventana Asignaciones Destino OLE DB Seguimientos Profesionales

Tras esto, el desarrollo de este paquete habrá finalizado.

Paquete HechoEstanciaUnidades

El paquete HechoDiferenciaSeg es el encargado de obtener los datos relacionados con las estancias de los residentes en las diferentes unidades de la base de datos operacional y rellenar la tabla de hechos del indicador de EstanciaUnidades. Se ejecuta tras el paquete Dimension y tantas veces como residencias queremos utilizar para los resultados del indicador, es decir, si queremos lanzar el indicador para dos centros, debemos lanzar este paquete para cada uno de los centros, previo cambio de la configuración de la conexión Residencia.

Este indicador estaba incluido en el grupo de los indicadores que mostraban una evolución y por ello, la dificultad del mismo radica en la forma de derivar los registros para cada día.

Lo primero que tenemos que hacer ir al panel de **Administradores de conexión**, hacer clic en **Nueva conexión desde origen de datos** y añadir los siguientes orígenes de datos: Almacen y Residencia.

Cuando tengamos las conexiones configuradas, necesitamos crear la variable IDCentro al igual que hicimos en el indicador anterior.

El flujo de control de este paquete será similar al del paquete anterior, ya que tenemos que realizar las mismas tareas: dotar de valor a la variable IDCentro y cargar los datos del indicador en la tabla de hechos.

Dado que el flujo de datos asociado a la primera tarea fue analizado en detalle en el paquete anterior, a continuación trataremos el flujo de datos asociado a la segunda tarea. Dicho flujo será parcialmente distinto al del otro indicador porque debemos realizar otros procesamientos por el tipo de indicador que es.

Cabe recordar que la tabla de hechos del indicador se compone de los siguientes campos: IDTiempo, IDResidente, IDCaracterPlaza, IDTipologia, IDCentro, IDUnidad, TiempoEstancia, TotalEstancia y MinEstancia.

Puesto que nos encontramos con que tiene dimensiones comunes con el otro indicador, sólo vamos a detallar las partes del flujo de datos que sean propias de dicho indicador.

El flujo de datos comienza tal y como indica la Figura 6.51.

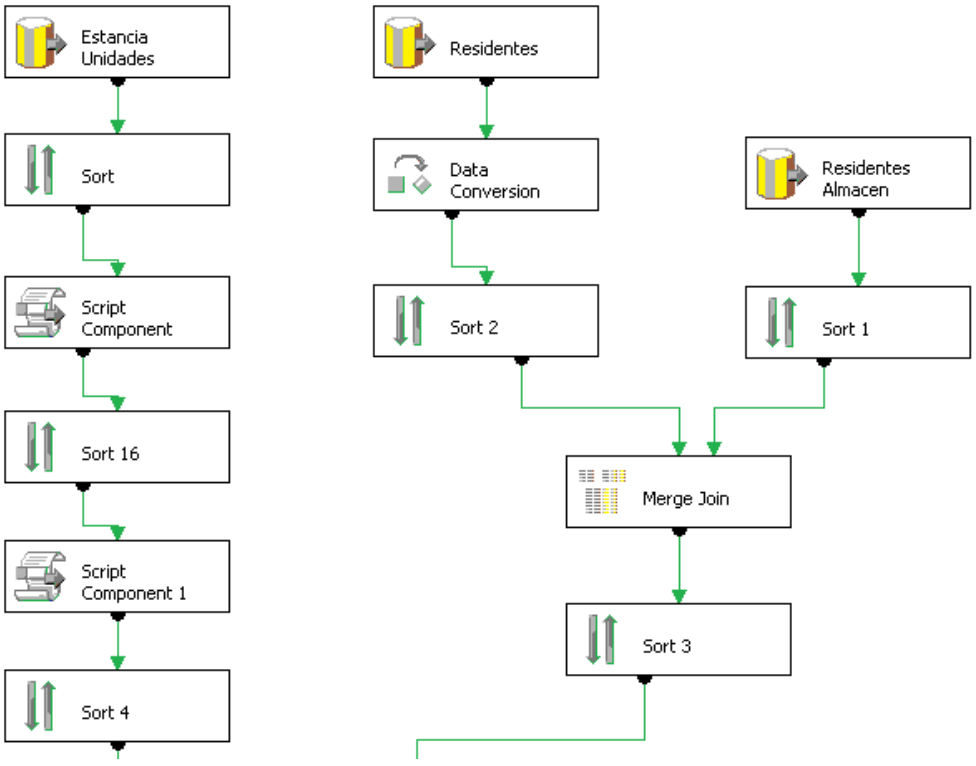


Figura 6.51. Parte 1 del flujo de datos asociado al indicador Estancia Unidades

Como toda la información que necesitamos se encuentra en la tabla de ResiUniEstancias, lo primero que hacemos es añadir un origen de datos que acceda a dicha tabla en la base de datos operacional. La consulta a introducir en dicho origen es la que se muestra en la Figura 6.52.

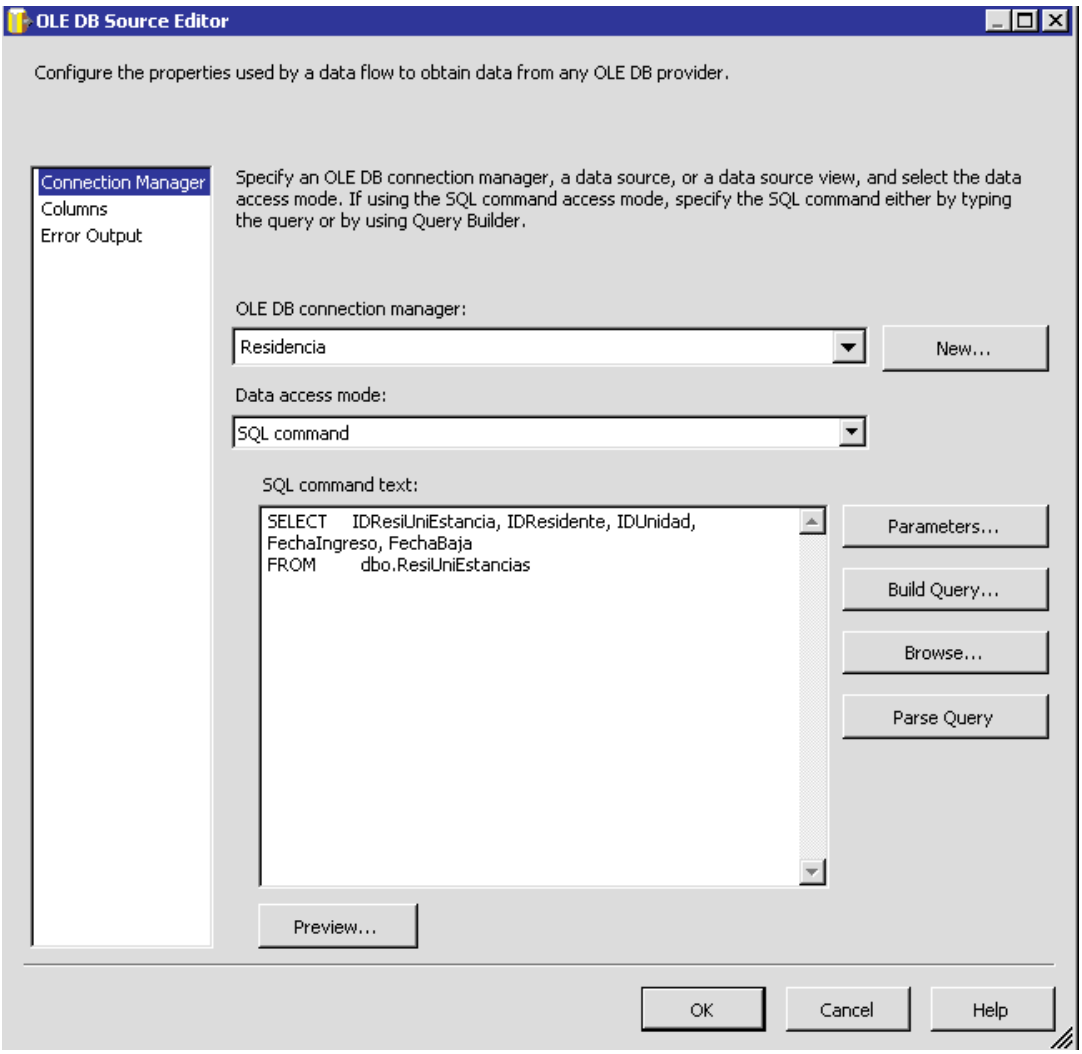


Figura 6.52. Configuración Origen OLE DB Estancia Unidades

A continuación ordenamos los datos ascendentemente según el IDResidente y la FechaIngreso en la unidad.

Lo siguiente que necesitamos es un script para derivar registros entre dos fechas. Dicho script asignará como fecha de baja de la unidad la fecha actual si ésta aparece a nulo o es superior a la actual y añadirá una fila por cada día incluido entre la fecha de ingreso y la fecha de baja de la unidad. Además, utilizará la columna de salida TiempoEstancia para indicar los días que lleva el residente en la unidad en una determinada fecha y la columna TotalEstancia para almacenar el tiempo total que ha permanecido el residente en la unidad.

El código de dicho script sería el siguiente:

```
fechadefecto = "31129999"
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
   fechaactual = Row.FechaIngreso
   If Row.FechaBaja_IsNull Then
        fechafinal = Now.Date
   Else
```

```
            If Format(Row.FechaBaja, "ddMMyyyy") = fechadefecto Then
               fechafinal = Now.Date
            Else
               fechafinal = Row.FechaBaja
               If fechafinal = fechaactual Then
               Else
                  fechafinal = fechafinal.AddDays(-1)
               End If
            End If
         End If
         While fechaactual <= fechafinal
            Me.Output0Buffer.AddRow()
            Me.Output0Buffer.IDResidente = Row.IDResidente
            If Row.IDUnidad_IsNull Then
            Else
               Me.Output0Buffer.IDUnidad = Row.IDUnidad
            End If
            Me.Output0Buffer.Fecha = fechaactual
            Me.Output0Buffer.TiempoEstancia = DateDiff(DateInterval.Day,
   Row.FechaIngreso, fechaactual) + 1
            If fechaactual = fechafinal Then
               Me.Output0Buffer.TotalEstancia = DateDiff(DateInterval.Day,
   Row.FechaIngreso, fechaactual) + 1
            End If
            fechaactual = fechaactual.AddDays(1)
         End While
      End Sub
```

A la salida del script ordenamos nuevamente los datos, pero esta vez ascendentemente según el IDResidente y descendentemente según la Fecha del registro.

Posteriomente añadimos un nuevo componente de script que se va a encargar de rellenar el campo MinEstancia que utilizaremos para calcular el tiempo mínimo de estancia de los residentes en las unidades.

Para rellenar dicho campo utilizamos el campo TotalEstancia tal y como se detalla en el código del script:

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    Me.Output0Buffer.AddRow()
    Me.Output0Buffer.Fecha = Row.Fecha
    Me.Output0Buffer.IDResidente = Row.IDResidente
    If Row.IDUnidad_IsNull Then
    Else
        Me.Output0Buffer.IDUnidad = Row.IDUnidad
    End If
    Me.Output0Buffer.TiempoEstancia = Row.TiempoEstancia
    If Row.TotalEstancia_IsNull Then
    Else
        Me.Output0Buffer.TotalEstancia = Row.TotalEstancia
        MinEstancia = Row.TotalEstancia
    End If
    Me.Output0Buffer.MinEstancia = MinEstancia
End Sub
```

Otra parte del flujo del datos es la que muestra la Figura 6.53.

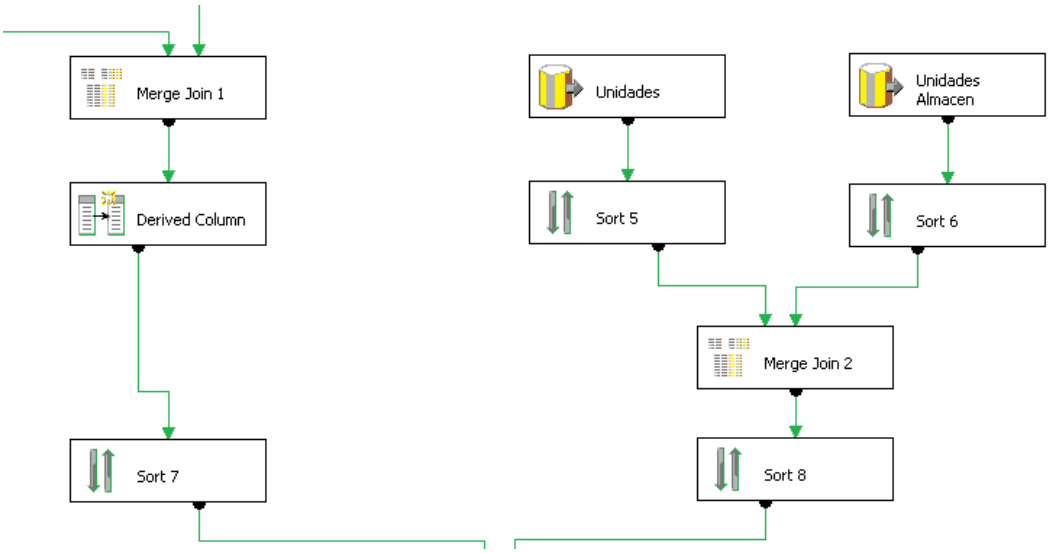


Figura 6.53. Parte 2 del flujo de datos asociado al indicador Estancia Unidades

En la parte izquierda, añadimos una transformación de columna derivada para añadir al conjunto de datos el IDCentro del almacén usando la variable con el mismo nombre. En cambio, en la parte derecha lo que hacemos es obtener el IDUnidad del almacén de datos que se corresponde con el IDUnidad de la base de datos operacional.

Llegados a este momento, lo siguiente es enlazar con las tablas de dimensiones y las tablas auxiliares para obtener los siguientes IDs del almacén: IDResidente, IDTiempo,

IDCaracterPlaza y IDTipologia. La forma de enlazar es la misma que se indicó en el paquete anterior.

El flujo de datos finaliza con la adición del destino de datos. La tabla destino es Fact_EstanciaUnidades y las asignaciones son las que aparecen en la Figura 6.54.

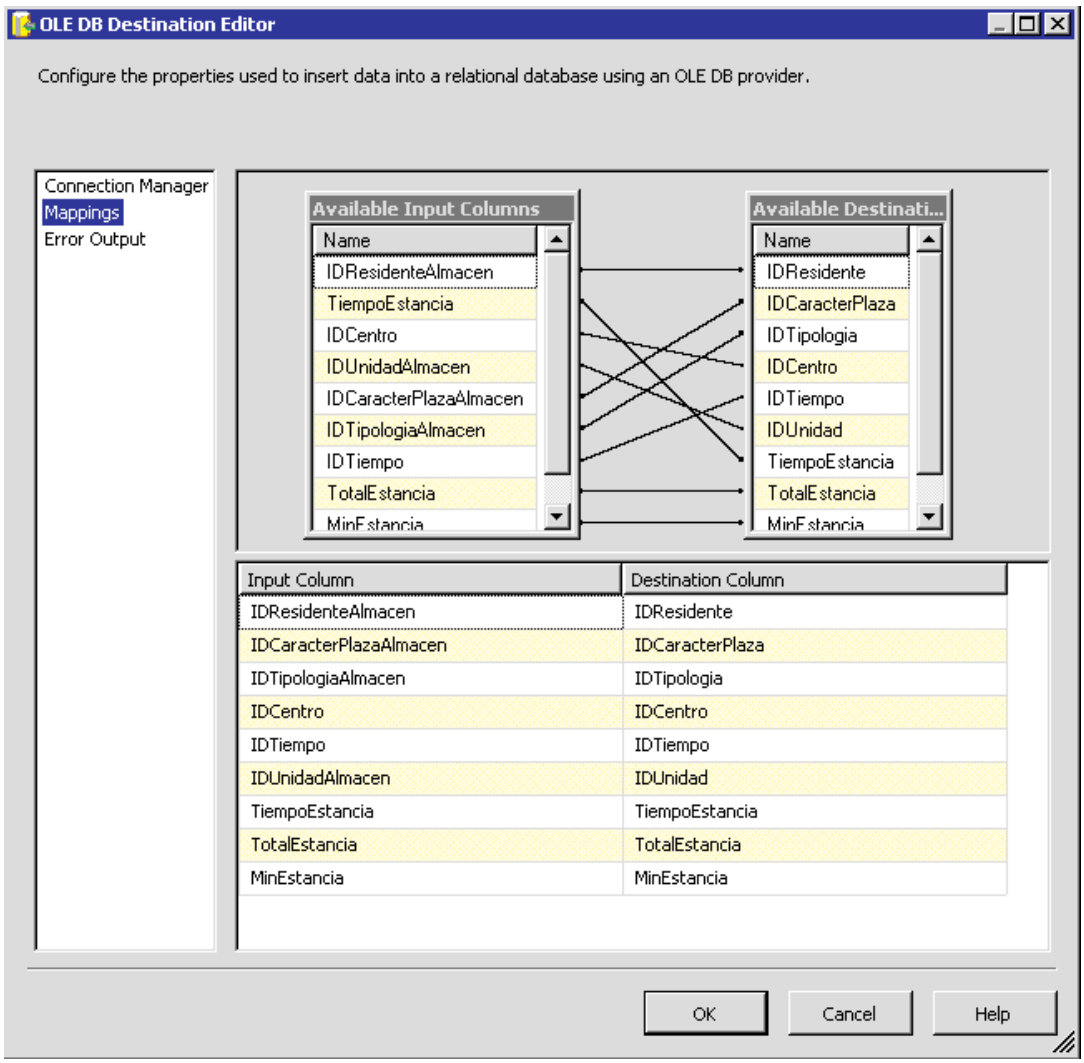


Figura 6.54. Ventana Asignaciones Destino OLE DB Estancia Unidades

Tras esto, el desarrollo de este paquete habrá finalizado.

Una vez finalizado el desarrollo de todos los paquetes ETL necesarios para la carga de nuestros indicadores, lo único que falta es ejecutar cada uno individualmente para comprobar que todo funciona correctamente. Recordamos que para ejecutar un paquete debemos seleccionar dicho paquete en el Explorador de soluciones, hacer clic con el botón derecho y seleccionar Ejecutar paquete.

6.5.2. Creación cubos OLAP

Ahora vamos a crear los cubos OLAP de cada uno de nuestros indicadores, incluyendo las medidas solicitadas para cada uno de ellos.

Puesto que en el tutorial de Analysis Services se detalla paso a paso cómo crear un cubo, en este punto no entraremos en detalle en cómo realizar las acciones vistas en ese tutorial (crear un proyecto, crear una vista de datos, etc).

Antes de comenzar con la implementación de los cubos comentados anteriormente necesitamos crear un proyecto SSAS. Como ya creamos uno para el indicador de Visitas Familiares, utilizaremos ese mismo proyecto.

Dicho proyecto tiene un origen definido que se conecta al almacén de datos y una vista de dicho origen de datos que incluye todas las tablas definidas para el indicador de Visitas Familiares. Dado que las tablas de los nuevos indicadores no aparecen en dicha vista automáticamente, lo primero que tenemos que hacer es añadir dichas tablas a la vista. Para añadir tablas a la vista de datos debemos:

1. Seleccionamos la vista del origen de datos en el Explorador de soluciones, hacemos clic en el botón derecho y seleccionamos **Abrir**.
2. Nos situamos sobre la vista, pinchamos con el botón derecho sobre ella y seleccionamos **Añadir/Quitar tablas**.
3. Seleccionamos todas las tablas que deseamos añadir a la vista en la lista de objetos disponibles (Figura 6.55.), pulsamos **>** y **Aceptar**.

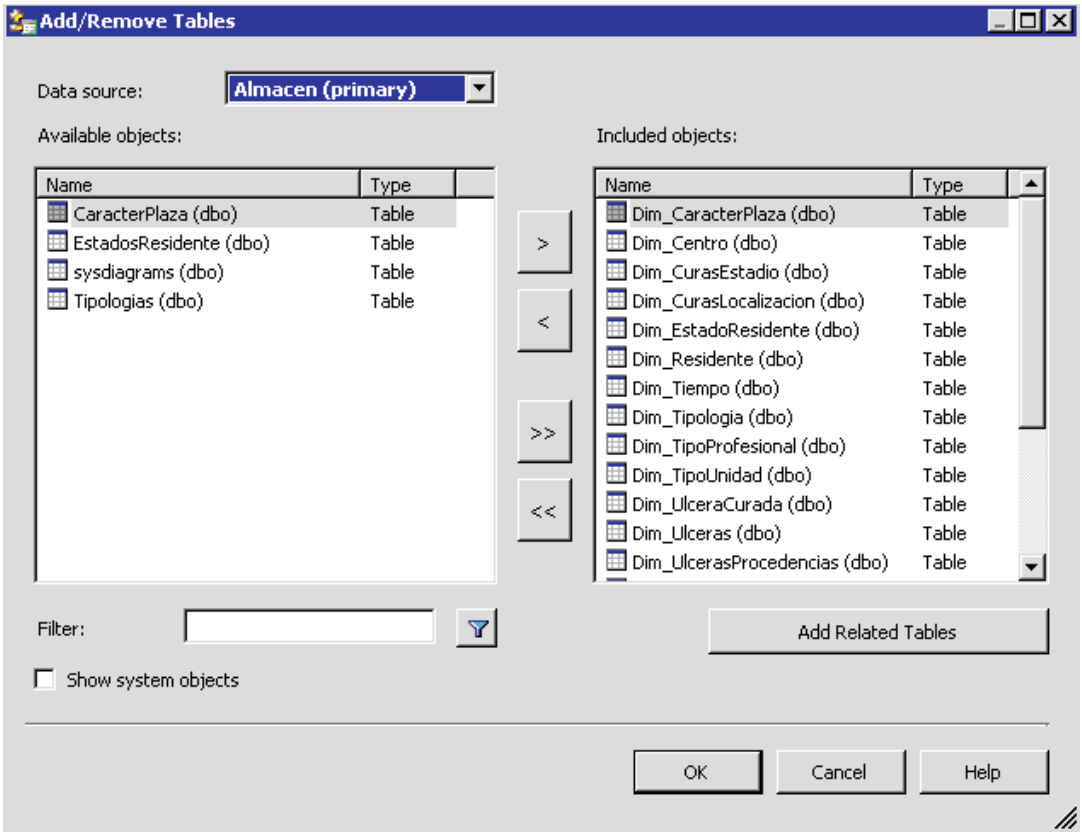


Figura 6.55. Ventana Añadir/Quitar tablas

Opcionalmente, podemos definir un diagrama o vista parcial de la anterior para cada uno de los indicadores. Para crear uno de estos diagramas debemos ir a la sección de los diagramas, hacer clic con el botón derecho y seleccionar **Nuevo Diagrama** (Figura 6.56.).

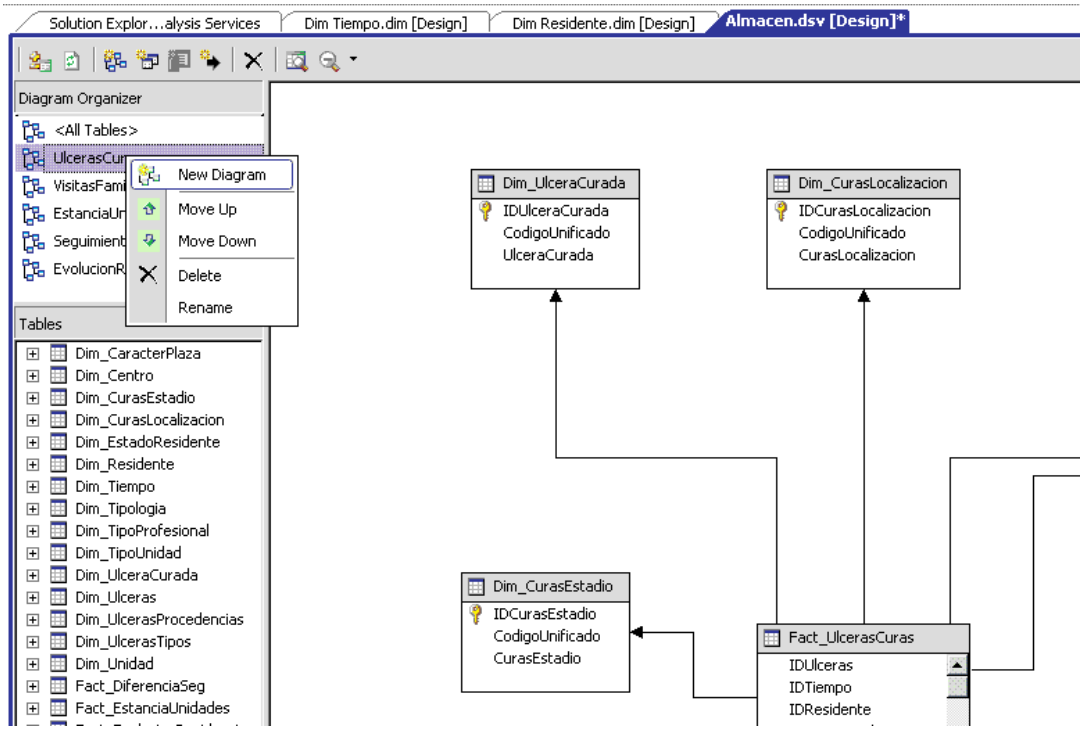


Figura 6.56. Crear nuevo diagrama

Cuando tengamos el nuevo diagrama, hacemos clic con el botón derecho sobre el diagrama, seleccionamos la opción **Mostrar Tablas** y seleccionamos las tablas que queremos que aparezcan en el diagrama. Por tanto, si queremos construir el diagrama del indicador de Ulceras Curas, debemos seleccionar todas las tablas con las que se relaciona la tabla de hechos y la tabla de hechos propiamente.

El siguiente paso es definir los cubos. En nuestro caso la arquitectura escogida es la MOLAP porque es la que mejor se comporta con datos de pocas dimensiones como los de nuestros indicadores.

Para definir los cubos realizamos las siguientes acciones:

1. Vamos al Explorador de soluciones, hacemos clic con el botón secundario en **Cubos** y luego seleccionamos **Nuevo cubo**.
2. Indicamos como método de generación **Usar tablas existentes** y seleccionamos una de las tablas de hechos de los nuevos indicadores (Fact_UlcerasCuras).
3. Aceptamos todas las medidas y dimensiones que aparecen por defecto, le damos nombre al cubo (UlcerasCuras) y pulsamos **Finalizar**.
4. Repetimos los pasos 1-3 para crear los cubos SeguimientosProfesionales y EstanciaUnidades.

La Figura 6.57. muestra uno de los cubos que acabamos de crear con todas sus relaciones:

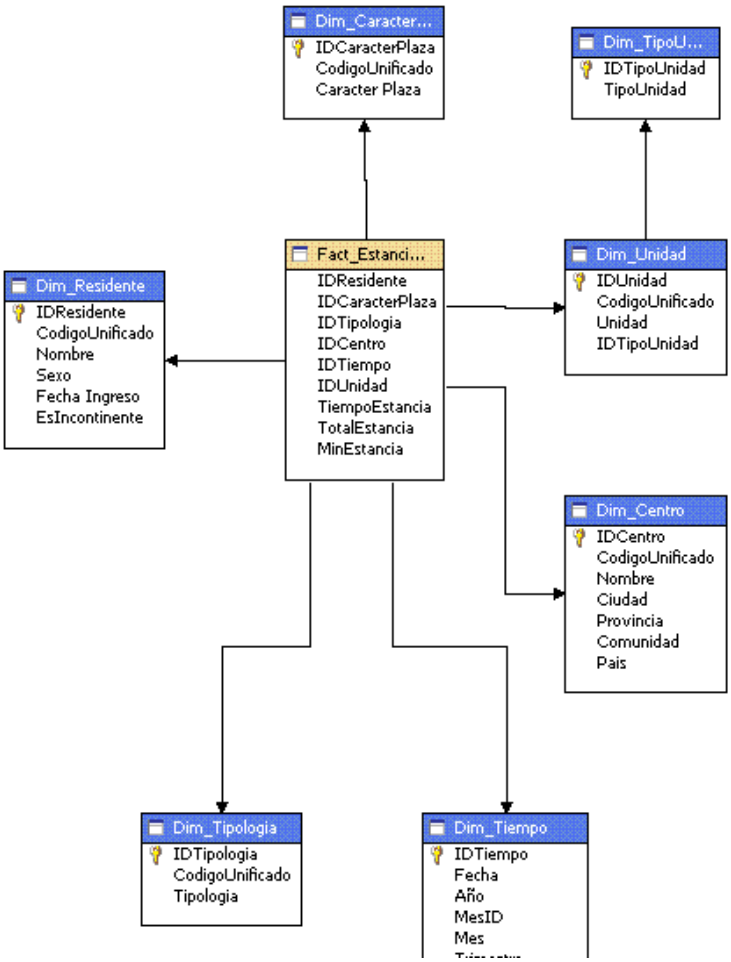


Figura 6.57. Cubo EstanciaUnidades

Al finalizar el proceso de creación de los cubos, se observa que éstos aparecen incluidos en la carpetas Cubos del Explorador de soluciones y las dimensiones aparecen en la carpeta Dimensiones.

A continuación modificaremos las dimensiones que acaban de ser añadidas haciendo doble clic en ellas (Figura 6.58.). Concretamente, tenemos que hacer lo siguiente:

1. Arrastraremos los atributos que queremos que se muestren posteriormente desde la vista del origen de datos a la parte de los atributos.

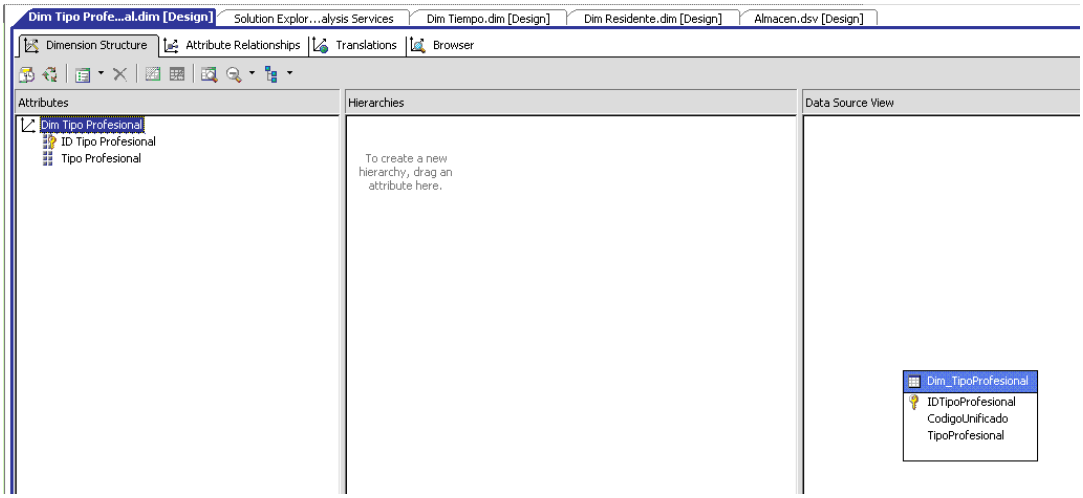


Figura 6.58. Ventana asociada a la Dimension Tipo Profesional

2. En la ventana de atributos, seleccionamos el nombre de la dimensión, hacemos clic con el botón secundario y seleccione **Propiedades**. Nos aseguramos de que la propiedad **UnknownMembe**r está 'Visible' y ponemos el valor 'No Definido' en la propiedad **UnknownMemberName**.
3. Modificamos las propiedades de los atributos de la dimensión que aparecen marcados como clave de la dimensión haciendo clic con el botón derecho sobre el atributo y modificando a 'False' el campo **AttributeHierarchyVisible** de la ventana **Propiedades** para que no se vean dicho atributos, ya que se trata de identificadores. Para el resto de atributos dejamos las propiedades tal y como aparecen por defecto.
4. Repetimos los pasos 1-3 para cada una de las nuevas dimensiones.

El cubo de EstanciaUnidades posee una particularidad, ya que una de sus dimensiones (Dim_Unidad) hace referencia a una subdimensión (Dim_TipoUnidad). Esta particularidad ha de ser tratada indicando en el cubo que, además de las dimensiones referenciadas, existe otra dimensión que es referenciada a través de otra. Para indicar esto, los pasos a seguir son los siguientes:

1. Hacemos doble clic sobre el cubo de EstanciaUnidades y seleccionamos la ficha **Uso de dimensiones**.
2. Hacemos clic con el botón secundario, seleccionamos **Añadir** Dimension y seleccionamos la dimensión Dim_TipoUnidad.
3. Hacemos clic sobre el botón que aparece junto a la dimensión que acabamos de añadir y rellenamos la pantalla que aparece tal y como indica la Figura 6.59.

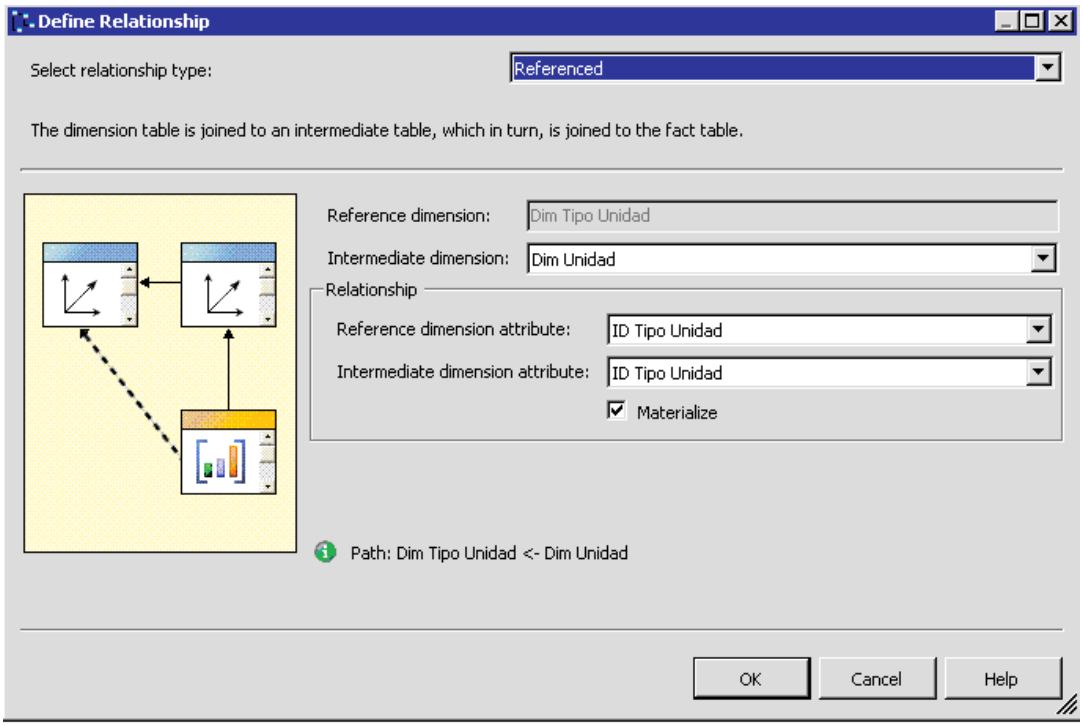


Figura 6.59. Definición de la relación entre Dim Unidad y Dim Tipo Unidad

Según esta información, la dimensión TipoUnidad es una dimensión referenciada por la dimensión Unidad a través del campo IDTipoUnidad.

Ahora es el momento de configurar las medidas de cada uno de los cubos y eliminar aquéllas que aparecen por defecto y que no nos interesan. A continuación se indican las medidas que vamos a definir para cada uno de los cubos:

Cubo UlcerasCuras

Queremos saber el número total de curas realizadas y el tamaño medio de las úlceras.

Las medidas y campos calculados a definir son los siguientes:

- Tamaño: Es la suma de todos los tamaños de las úlceras (Campo TamañoUlcera).
- Número Curas: Es el número de registros de la tabla de hechos y se corresponde con el número total de curas realizadas. Se define como Count of rows de Fact_UlcerasCuras.
- Tamaño medio: Es el tamaño medio de las úlceras que padecen nuestros residentes. Se define como un cociente Tamaño/Número Curas (Figura 6.60).

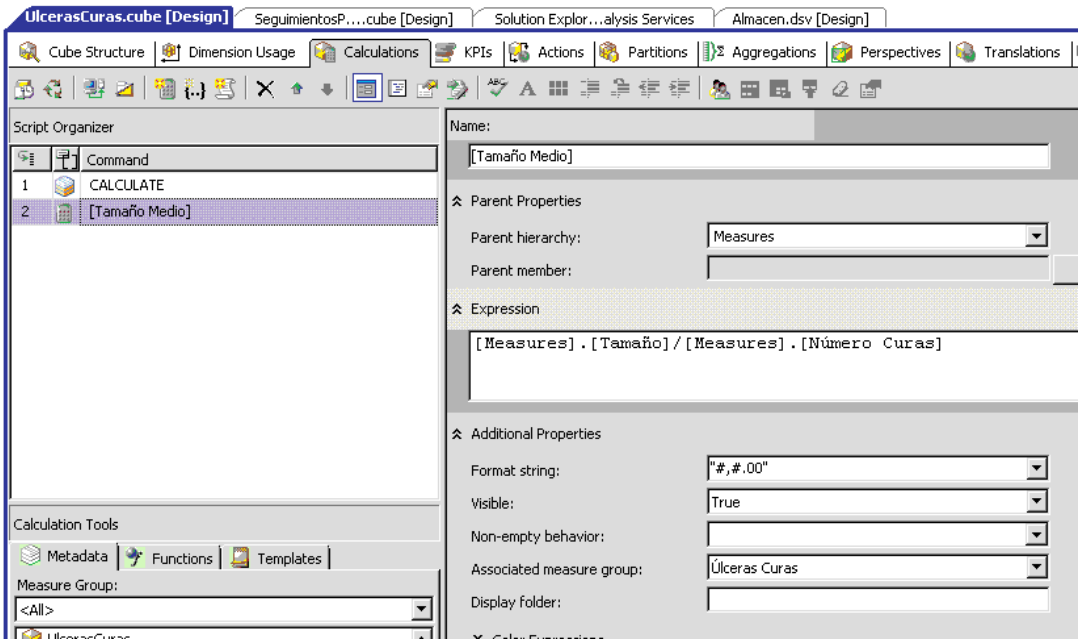


Figura 6.60. Definición de la medida Tamaño medio

Cubo SeguimientosProfesionales

Queremos saber el número total de días transcurridos entre seguimientos profesionales a los residentes, el tiempo mínimo transcurrido entre 2 seguimientos de un mismo profesional, el tiempo máximo transcurrido entre 2 seguimientos de un mismo profesional y el tiempo medio transcurrido entre seguimientos.

Las medidas y campos calculados a definir son los siguientes (Figura 6.61.):

- Número Días: Es la suma de todas las diferencias de días (Campo TiempoTranscurrido).
- Tiempo Transcurrido Count: Es el número de registros de la tabla de hechos. Se define como Count of rows de Fact_DiferenciaSeg.
- Mínimo: Es el tiempo mínimo transcurrido entre dos seguimientos. Se define como Minimun de TiempoTranscurrido.
- Máximo: Es el tiempo máximo transcurrido entre dos seguimientos. Se define como Maximum de TiempoTranscurrido.

- Media: Es el número de días medio que transcurre entre dos seguimientos profesionales. Se define como un cociente Número Días/Tiempo Transcurrrido Count.

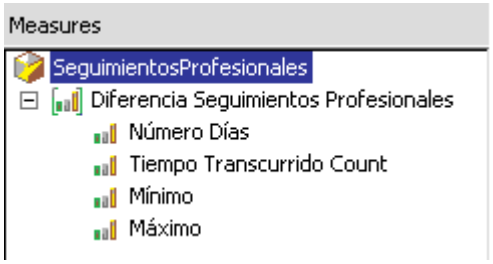


Figura 6.61. Medidas cubo Seguimientos Profesionales

Cubo EstanciaUnidades

Queremos saber el número total de días de estancia en unidades, el tiempo mínimo que un residente pasa en una unidad, el tiempo máximo que un residente pasa en una unidad y el tiempo medio de estancia de un residente en una unidad.

Las medidas y campos calculados a definir son los siguientes (Figura 6.62.):

- Total Estancia: Es la suma de todos las estancias en unidades (Campo TotalEstancia).
- NumRegistros: Se define como Count of non-empty values del campo TotalEstancia, ya que sólo queremos considerar los registros cuyo valor no sea nulo.
- Mínimo Tiempo: Es el tiempo mínimo que los residentes han estado en las unidades. Se define como Minimun de MinEstancia.
- Máximo Tiempo: Es el tiempo máximo que los residentes han estado en las unidades. Se define como Maximum del campo TotalEstancia.
- Media Estancia: Es el número de días medio que los residentes han estado en las unidades. Se define como un cociente TotalEstancia/NumRegistros.

Figura 6.62. Medidas cubo Estancia Unidades

Cuando acabemos de configurar las medidas, el siguiente paso a realizar es procesar todo el proyecto para ver que no hay ningún error. Para llevar a cabo esta acción seleccionamos el nombre del proyecto, hacemos clic con el botón secundario y seleccionamos **Procesar**.

La Figura 6.63. muestra que el proyecto ha sido procesado con éxito y por lo tanto, no hay ningún error.

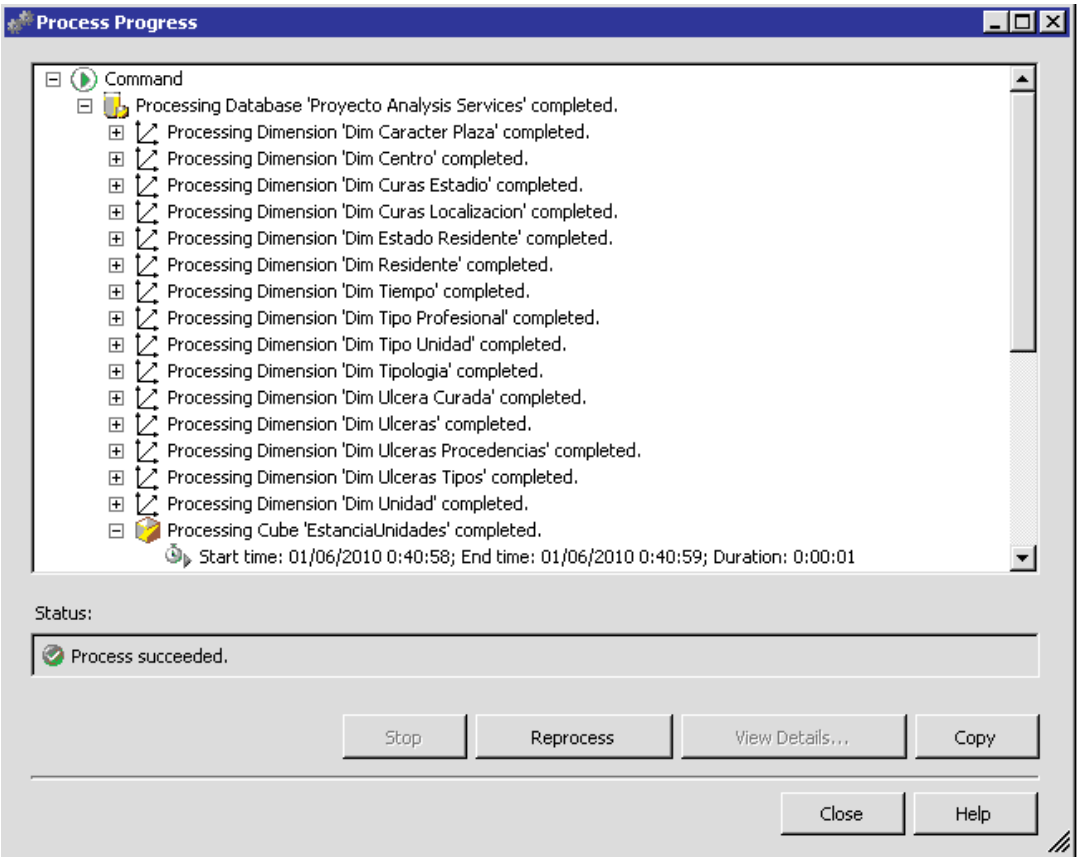


Figura 6.63. Ventana de progreso del procesamiento de los cubos

6.5.3. Creación trabajo de lanzamiento de paquetes

En este punto crearemos el trabajo encargado de actualizar los datos de los indicadores diariamente. Dicho trabajo se encargará de lanzar todos los paquetes ETL y procesar los cubos desarrollados en este capítulo.

Puesto que en el tutorial del Agente SQL Server se detalla paso a paso cómo crear un trabajo, en este punto no entraremos en detalle en cómo realizar las acciones vistas en ese tutorial (crear un trabajo, crear una programación, etc).

Como queremos obtener los datos de los indicadores para varias residencias, debemos lanzar los paquetes Dimension y de cada uno de los hechos tantas veces como residencias. En nuestro caso queremos lanzar los datos para 3 residencias y por este motivo cada uno de los paquetes se lanzará 3 veces a excepción de Almacen. Además, se necesita lanzar también el paquete Almacen para borrar el contenido del almacén de datos y rellenarlo con los datos más actualizados cada día.

El trabajo que necesitamos crear tiene que lanzar los paquetes y procesar los cubos en el siguiente orden:

1. Lanzamos el paquete ETL Almacen.
2. Lanzamos el paquete ETL Dimension para una de las residencias.
3. Lanzamos los paquetes ETL de cada uno de los indicadores para esa residencia.
4. Lanzamos los paquetes Dimension y de los indicadores para el resto de residencias.

5. Procesamos los cubos.

Suponiendo que ya hemos accedido al SQL Server Management Studio, los pasos a seguir para crear este trabajo son los siguientes:

1. Hacemos clic con el botón derecho en el nodo **Trabajos**, seleccionamos **Nuevo Trabajo** y rellenamos propiedades del mismo tales como el nombre y la descripción (Figura 6.64.).

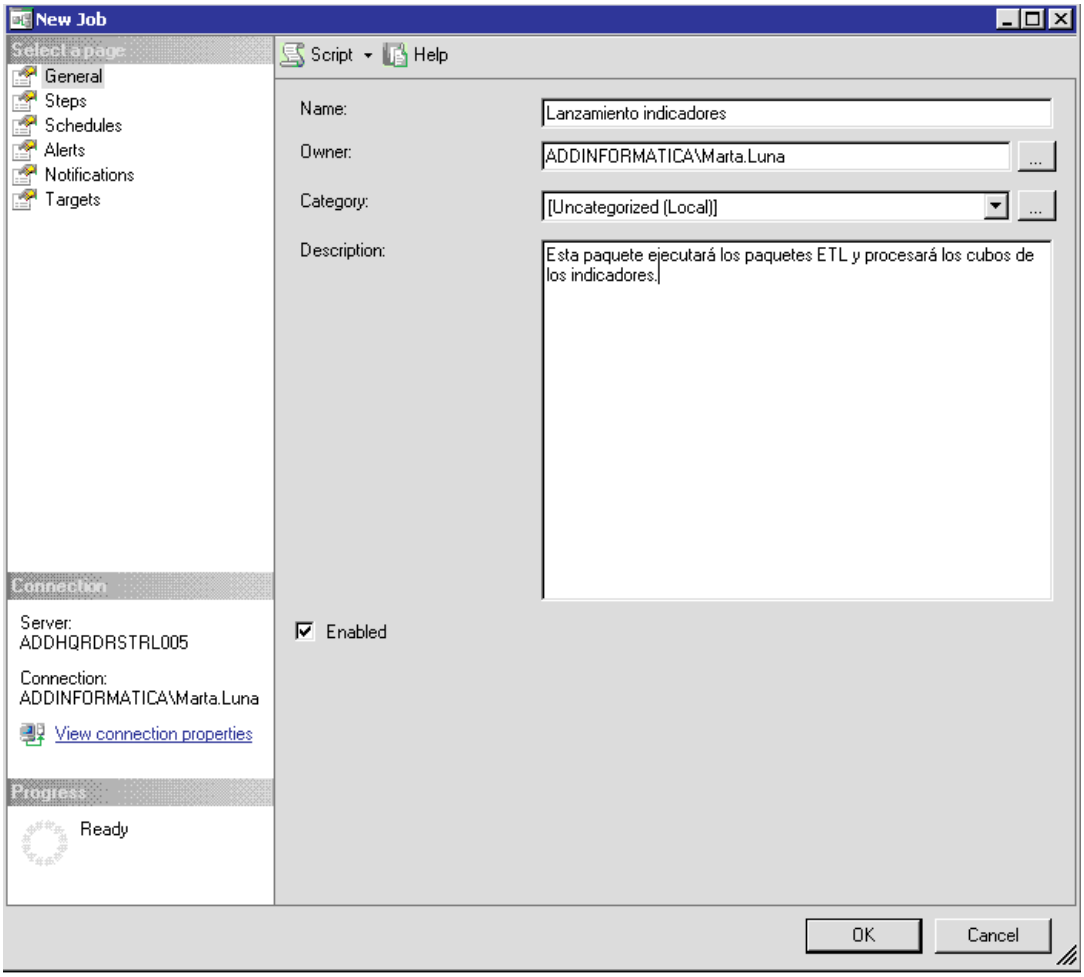


Figura 6.64. Ventana de configuración del trabajo

2. Seleccionamos la página **Pasos** y hacemos clic en **Nuevo**.
3. Especificamos el nombre para el paso (Almacen), el tipo de paso como **Paquetes de Integration Services**, deje la opción **Ejecutar como** que aparece, ponga como **Origen del paquete** el sistema de archivos (File system) e indique la ruta donde se encuentra el paquete que queremos ejecutar (Figura 6.65.).

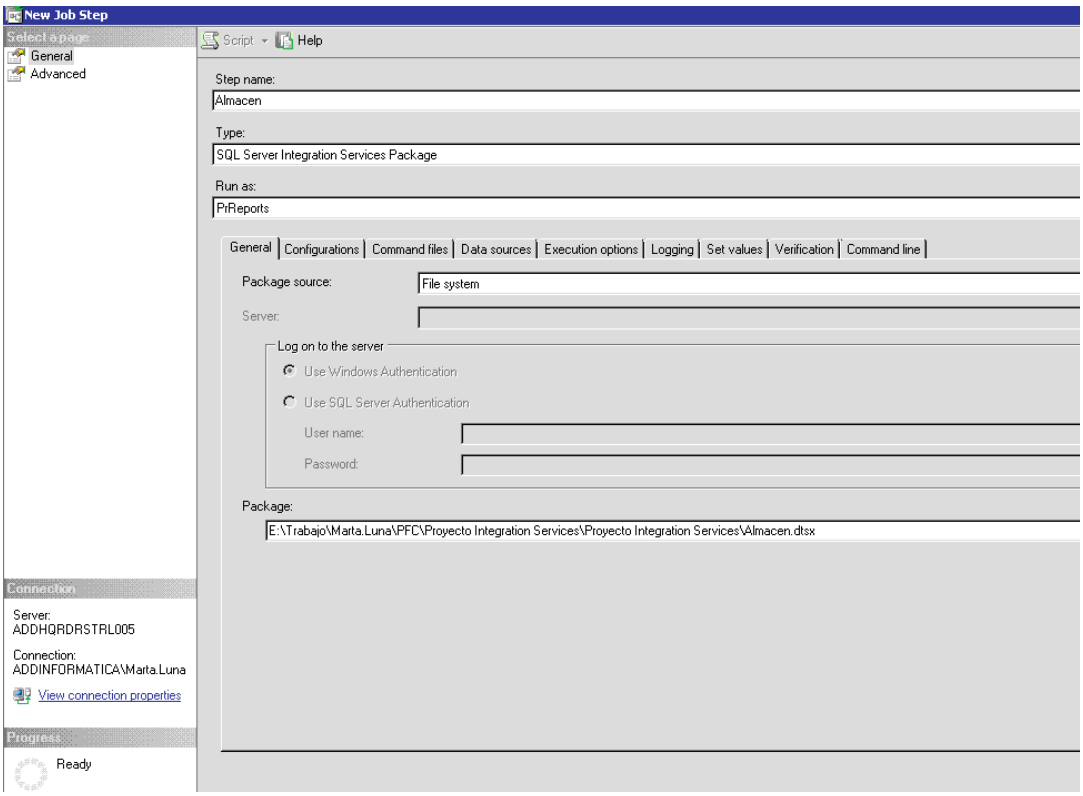


Figura 6.65. Ventana de configuración del paso Almacen

4. Hacemos clic en la página **Avanzado** y especifique **Ir al siguiente paso** como **Acción en caso de éxito**, 1 como **Número de reintentos** para que vuelva a intentar ejecutar el paquete y **Salir del trabajo e informar del error** como **Acción en caso de error**.

5. Repetimos los pasos 2-4 para cada uno de los paquetes de SSIS. No obstante, para estos pasos debemos ir a la pestaña **Data Sources** (Figura 6.66.) de las propiedades del paso y cambiar la conexión del origen de datos Residencia para que apunte a la base de datos de una nueva residencia.

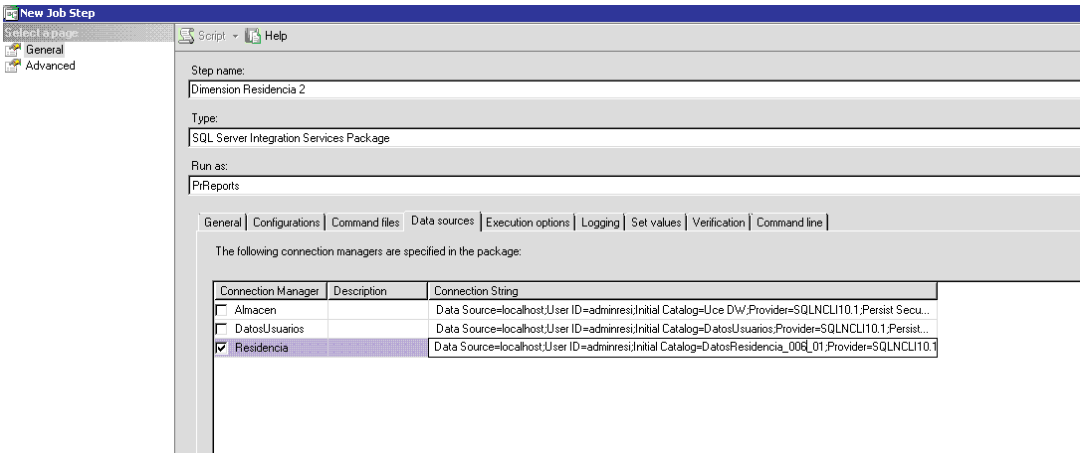


Figura 6.66. Pestaña Data Sources del paso Dimension Residencia 2

6. Creamos un nuevo paso para procesar los cubos de los indicadores. Ponemos como tipo de paso **Comando de Analysis Services**, dejamos la opción **Ejecutar como** que aparece, ponemos como **Servidor** la máquina local e indicamos en **Comando** la siguiente instrucción XML:

```
<Process xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
 <Object>
  <DatabaseID>Proyecto Analysis Services</DatabaseID>
 </Object>
 <Type>ProcessFull</Type>
 <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
</Process>
```

Esta instrucción procesará todos los elementos de nuestro proyecto de Analysis Services.

7. Por ser el último paso, hacemos clic en la página **Avanzado** y especificamos **Salir del trabajo e informar del éxito** como **Acción en caso de éxito**, 1 como **Número de reintentos** para que vuelva a intentar ejecutar el paquete y **Salir del trabajo e informar del error** como **Acción en caso de error**.
8. Haga clic en **Aceptar** para guardar el nuevo trabajo.

El trabajo resultante es el que muestra la Figura 6.67.

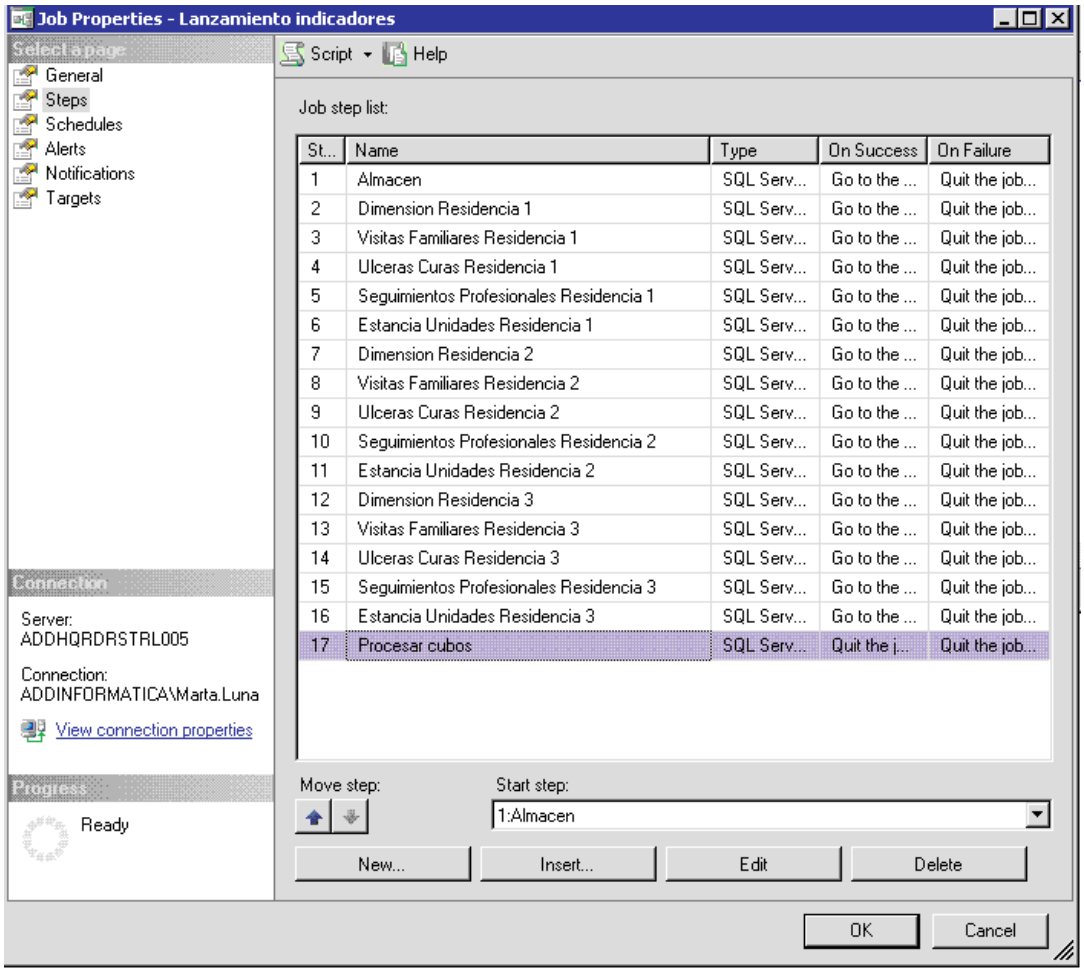


Figura 6.67. Lista de pasos del trabajo Lanzamiento Indicadores

Una vez tenemos creado el trabajo, lo que falta es crear la programación para dicho trabajo de forma que el trabajo se ejecute diariamente.

Para crear la programación del trabajo debemos seguir los siguientes pasos:

1. Seleccionamos la página **Programaciones** del trabajo y hacemos clic en **Nueva**.
2. Rellenamos los datos de la programación tal y como indica la Figura 6.68. y hacemos clic en **Aceptar**. Según esta configuración, el trabajo se va a lanzar todos los días a las 00:00 h.

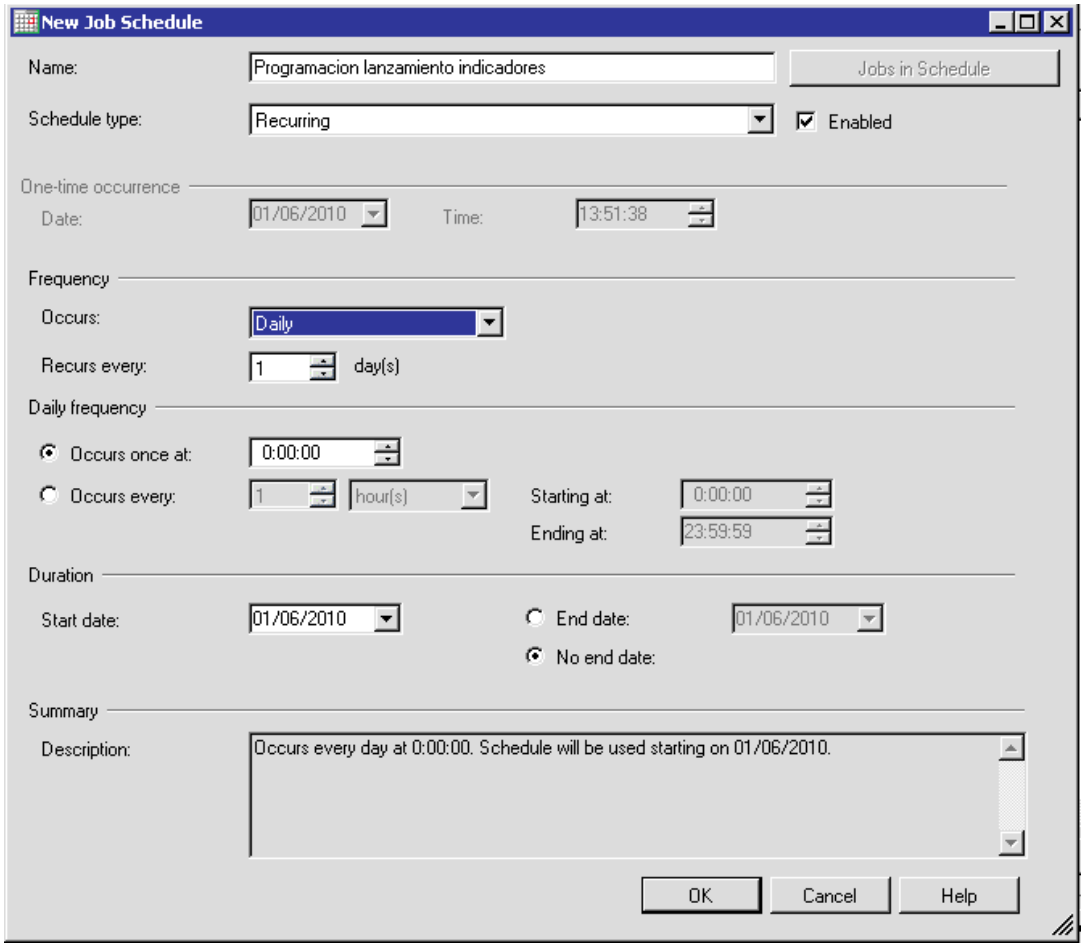


Figura 6.68. Configuración de la programación

Al finalizar la creación del trabajo y su respectiva programación, sólo falta ejecutar el trabajo para ver que funciona correctamente y poder dar el proceso de creación del trabajo por finalizado. Para ello hacemos clic con el botón derecho en el trabajo recién creado, seleccionamos la opción **Iniciar trabajo en el paso**, especificamos el paso **1** y hacemos clic en **Inicio.**

Observamos que el trabajo no contiene errores y se ejecuta correctamente, ya que en la ventana de progreso observaremos que pone éxito (Figura 6.69.).

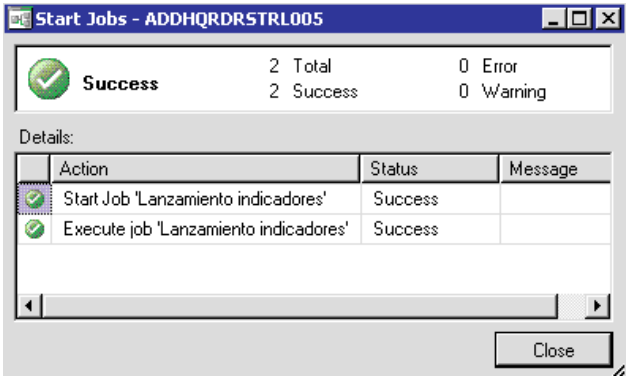


Figura 6.69. Ventana de progreso del trabajo ejecutado

6.5.4. Creación informes

En este punto crearemos los informes de los indicadores creados en puntos anteriores.

Puesto que en el tutorial de creación de informes con Excel se detalla paso a paso cómo crear un informe, en este punto no entraremos en detalle en cómo realizar las acciones vistas en ese tutorial (crear una conexión, crear un gráfico, etc).

Como ya tenemos el informe que creamos para el indicador de Visitas Familiares, necesitamos crear solamente tres informes.

Según el indicador al que asociemos el informe, la conexión al cubo será diferente. En el caso del indicador de Ulceras Curas, debemos configurar la conexión para que se conecte al cubo con ese mismo nombre. Los pasos a seguir son los siguientes:

1. Vamos a la ficha **Datos** del Excel, en el grupo **Obtener datos externos** hacemos clic en **De otras fuentes** y luego en **Desde Analysis Services** y se mostrará el **Asistente para la conexión de datos**.
2. Indicamos el nombre del servidor OLAP donde se encuentra nuestro cubo, seleccionamos **Usar autenticación de Windows** como **Credenciales de conexión** y a continuación seleccionamos la base de datos que contiene la información que desea (Proyecto Analysis Services) y cubo al que nos queremos conectar (UlcerasCuras) tal y como se indica en la Figura 6.70.

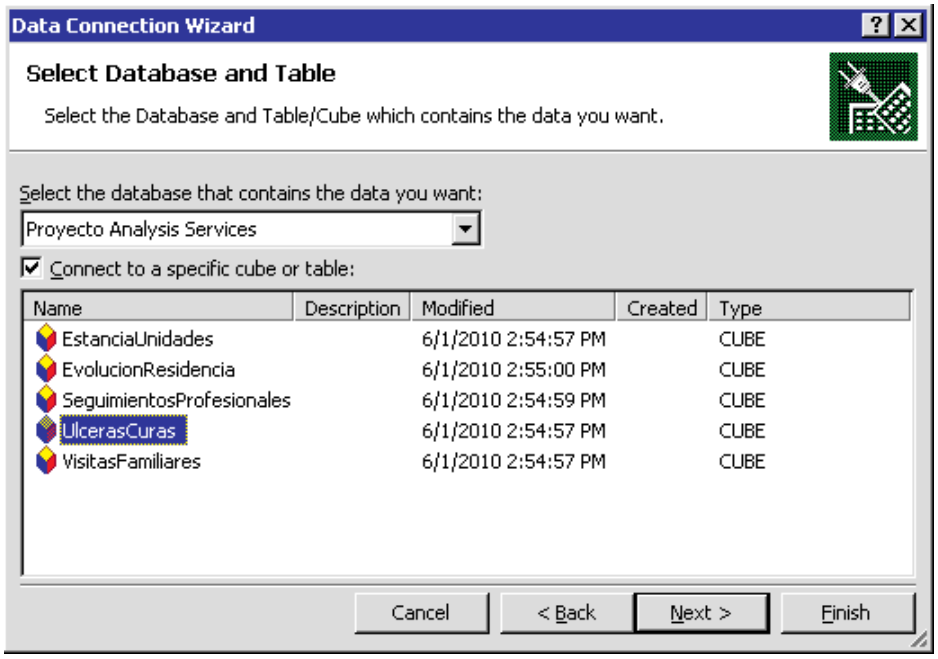


Figura 6.70. Configuración de la conexión del archivo Excel

3. Indicamos la ubicación en la que deseamos guardar el archivo de conexión de datos y pulsamos **Finalizar** para que empice la importación de datos.

Para crear las conexiones del resto de informes, seguimos los mismos pasos pero variando el cubo al que nos conectamos.

Una vez tenemos configurada la conexión de nuestro informe, el siguiente paso es crear las diferentes vistas (gráficas o tablas) que va a contener nuestro informe, las cuales serán utilizadas posteriormente en el dashboard.

El número de vistas a definir en el informe es variable y dependerá del usuario que lo va a utilizar. Es muy importante definir vistas que incluyan los datos para calcular los KPIs, ya que de no hacerlo no tendremos la posibilidad de utilizar los valores del libro de Excel en la definición de los mismos.

A continuación se definen un par de vistas muy sencillas para el indicador Ulceras Curas.

Con la primera de ellas obtenemos el número de curas realizadas en cada centro según el año (Figura 6.71.):

	Número Curas
2004	**2358**
Residencia 2	301
Residencia 3	2057
2005	**15233**
Residencia 2	6260
Residencia 3	8973
2006	**6109**
Residencia 2	4008
Residencia 3	2101
2007	**5**
Residencia 3	5
2008	**14**
Residencia 1	11
Residencia 2	2
Residencia 3	1
2009	**175**
Residencia 2	81
Residencia 3	94
2010	**62**
Residencia 2	4
Residencia 3	58
Grand Total	**23956**

Figura 6.71. Vista del número de curas totales por año y centro

Otra posibilidad de obtener información similar a ésta es colocar un filtro (Figura 6.72.) que nos permita seleccionar el año o años en los que queremos consultar el número de curas.

Año	All
	Número Curas
Residencia 1	11
Residencia 2	10656
Residencia 3	13289
Grand Total	**23956**

Figura 6.72. Vista del número de curas por centro con un filtro por año

El gráfico asociado a dicha tabla sería el que se muestra en la Figura 6.73.

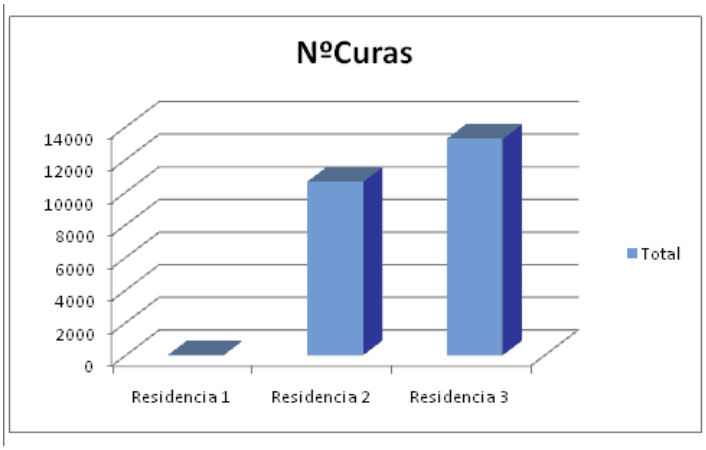


Figura 6.73. Gráfico del número de curas por centros

Otra de las posibles vistas a definir de este indicador es el número de curas según el estadio de las úlceras (Figura 6.74.).

	Número Curas
ALTA	104
I	208
II	388
III	162
IV	38
No Definido	23056
Grand Total	**23956**

Figura 6.74. Vista y gráfico del número de curas según el estadio de las úlceras

Tras definir las vistas del indicador, lo siguiente es dotar de nombre a cada una de estas vistas de la forma que se indica en el tutorial. Es importante dotar de nombre también al rango de celdas que contiene la tabla y el gráfico para luego utilizarlo como una vista en el dashboard.

La forma de guardar el informe para luego poder usar los elementos a los que les hemos dado nombre en el dashboard es utilizando la pestaña **Publish** del cuadro de diálogo, seleccionando **Excel Services** y en el cuadro de **Excel Services Options** (Figura 6.75.) seleccionando los elementos del libro de Excel que queremos que sean visibles de la lista **Items in the Workbook**.

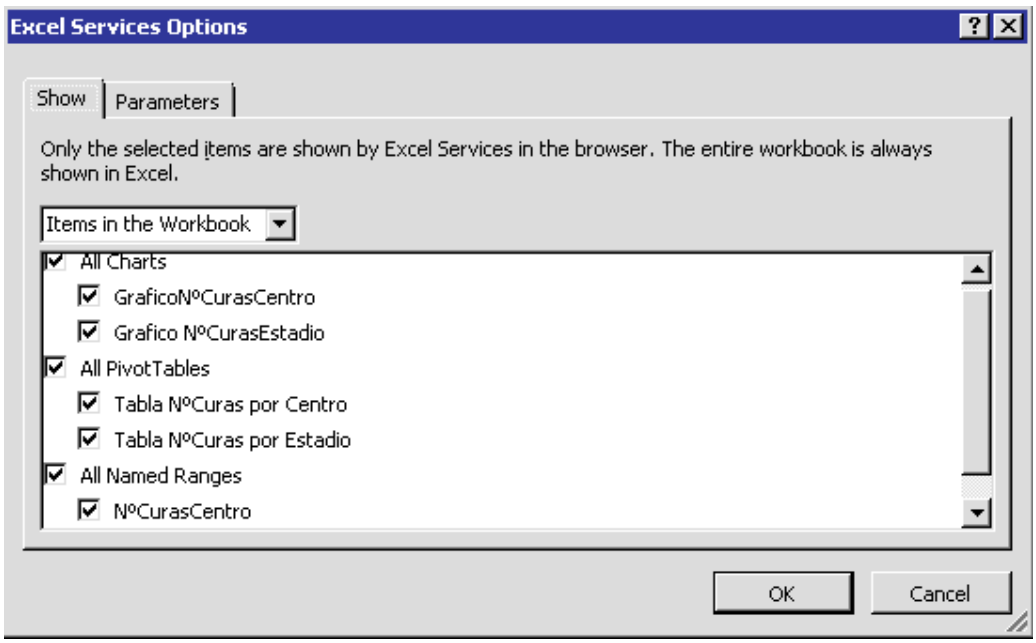


Figura 6.75. Elementos del libro Excel seleccionados para ser visibles

Finalizado el informe del primero de los indicadores, debemos hacer lo mismo para los indicadores de Seguimientos Profesionales y Estancia Unidades.

En el caso del indicador de Seguimientos Profesionales, una posible vista sería ver el tiempo medio transcurrido entre seguimientos según el centro y el año (Figura 6.76.).

Figura 6.76. Vista y gráfico del tiempo medio entre seguimientos por centro

Por otra parte, para el indicador de Estancia Unidades se puede definir una vista que muestre el tiempo de estancia medio en las unidades por residente (Figura 6.77.).

	Media Estancia
Residente 0001	796,00
Residente 0012	37,00
Residente 0013	48,00
Residente 0015	34,67
Residente 0020	835,00
Residente 0021	1,50
Residente 0022	5,00
Grand Total	**158,50**

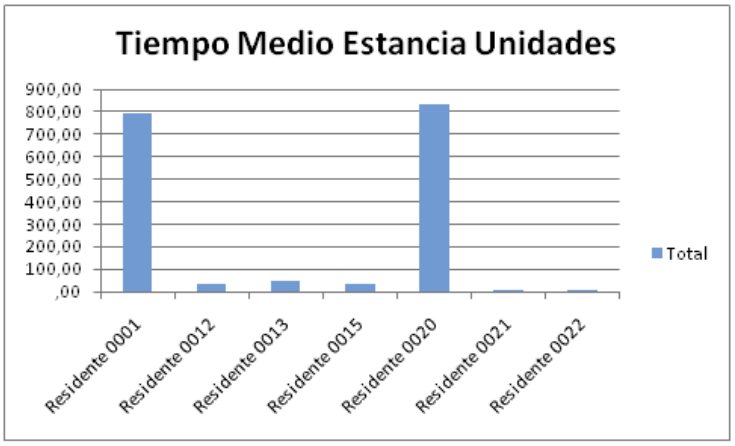


Figura 6.77. Vista y gráfico del tiempo medio de estancia en unidades por residente

Cuando hayamos acabado de crear las vistas de cada indicador y darle nombre a dichas vistas, el proceso de creación de los reports habrá finalizado.

6.5.5. Creación dashboard

Para finalizar el desarrollo e implementación de los indicadores, lo único que falta es subir los reports y sus conexiones al portal de Sharepoint y crear el dashboard.

En primer lugar subiremos los reports de la forma indicada en el tutorial de desarrollo de dashboards con Sharepoint. Tras subir los reports, podemos ver en la Figura 6.78. que han sido añadidos en el apartado de **Reports Library.**

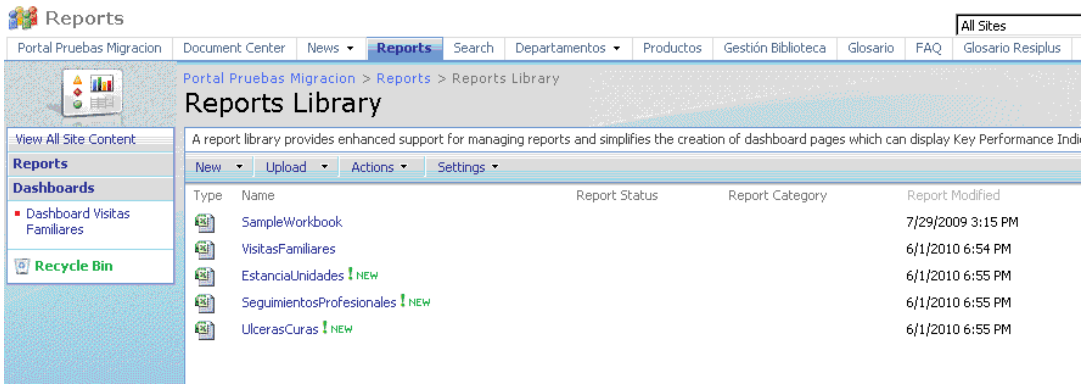


Figura 6.78. Lista de reports disponibles

Hacemos lo mismo con las conexiones en la página **Data Connections** (Figura 6.79.)pero en este caso, además tenemos que aprobar las conexiones que acabamos de subir.

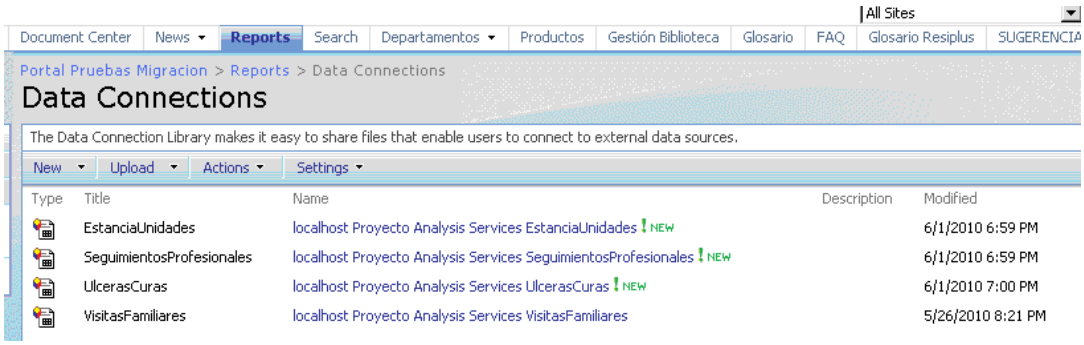


Figura 6.79. Lista de conexiones disponibles

Si consultamos los reports que acabamos de subir pinchando en sus enlaces asociados, podemos observar que sólo se muestran las vistas que hemos definido en cada uno de los informes. En caso que deseáramos consultar otra información, tenemos la opción de abrir el archivo de Excel desde la página de consulta para jugar con las tablas dinámicas hasta obtener la información deseada.

Para el report del indicador de Ulceras Curas, los elementos definidos en el report que aparecen en la pestaña View son los que aparecen en la Figura 6.80.

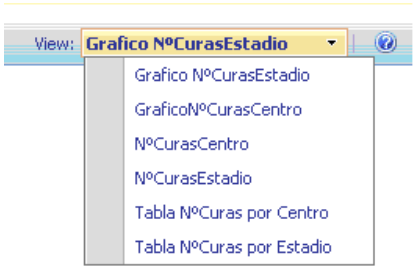


Figura 6.80. Elementos visibles para el indicador Ulceras Curas

A continuación vamos a crear el dashborard para nuestros cuatro indicadores. Para ello vamos a la página de **Dashboards**, hacemos clic en **New**, rellenamos el nombre de dicho dashboard y dejamos el resto de campos como aparecen por defecto.

Como lo que tenemos es cuatro indicadores, añadiremos a nuestro dashboard cuatro webparts de tipo **Excel Web Access** para que muestren las vistas definidas en el report. Después de añadirlas debemos editarlas para darles un título, indicar el report al que se conectan y modificar sus dimensiones para que se ajusten a las dimensiones del contenido que queremos mostrar. Todo esto lo haremos en la tabla que aparece cuando pinchamos el enlace **Click here to open the tool pane** o seleccionamos la opción **Modify Shared Web Part** de la lista desplegable que aparece en la esquina derecha superior de la webpart.

Después de añadir las webparts, el dashboard quedaría tal y como se muestra en la Figura 6.81.

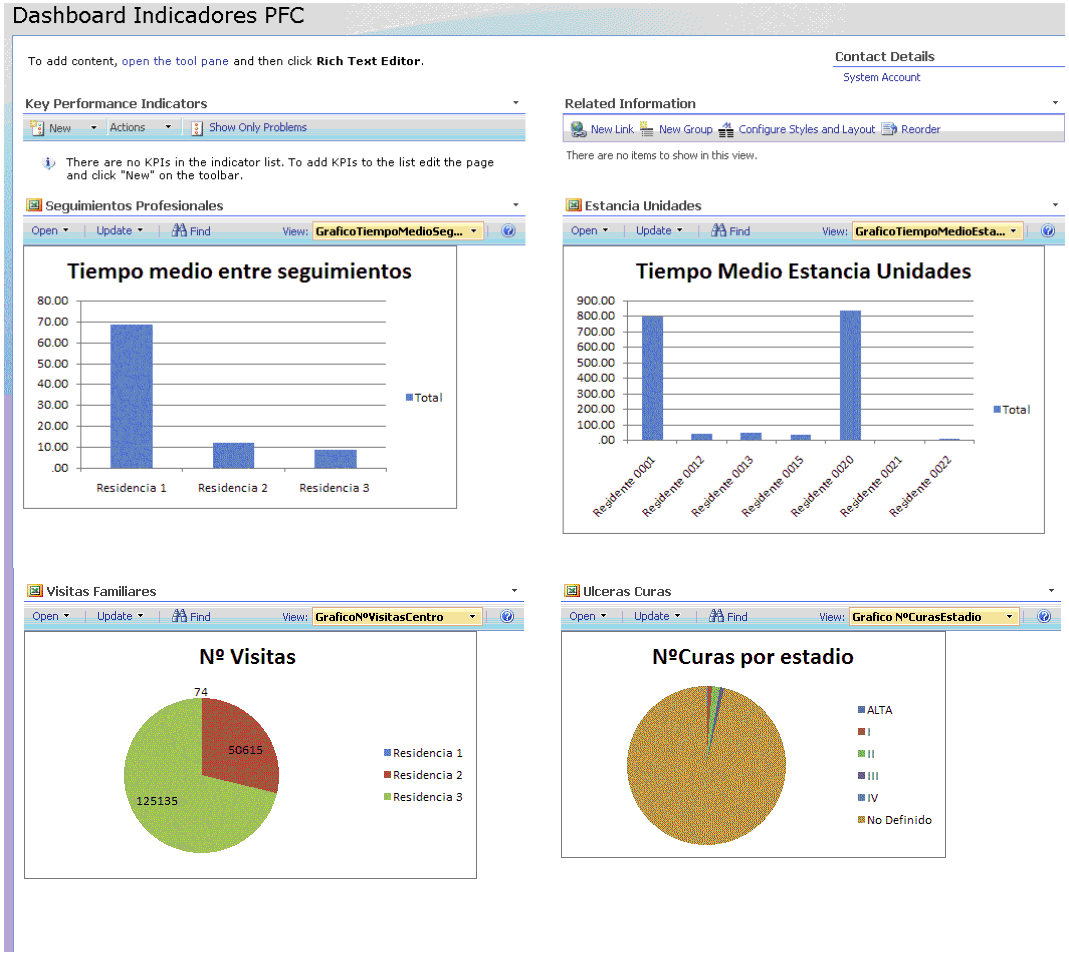


Figura 6.81. Dashboard compuesto de 4 indicadores

El siguiente paso es añadir los KPIs al dashboard usando la webpart que ha sido añadida automáticamente al crear el dashboard.

Los KPIs que debemos definir en el dashboard son los siguientes:

- El tamaño medio de las úlceras para la Residencia 1 se espera que sea menor o igual a 3. Los valores comprendidos entre 3 y 4 no son del todo malos y los valores a partir de 4 son malos.
- El tamaño medio de las úlceras para la Residencia 2 se espera que sea menor o igual a 1. Los valores comprendidos entre 1 y 2 no son del todo malos y los valores a partir de 2 son malos.
- El número medio de días transcurridos entre dos seguimientos para la Residencia 1 se espera que sea menor o igual a 50. Los valores comprendidos entre 50 y 75 no son del todo malos y los valores a partir de 75 son malos.
- El número medio de días transcurridos entre dos seguimientos para la Residencia 2 se espera que se menor o igual a 30. Los valores comprendidos entre 30 y 40 no son del todo malos y los valores a partir de 40 son malos.
- El tiempo medio de estancia de un residente en una unidad para la Residencia 1 se espera que sea menor o igual a 100. Los valores comprendidos entre 100 y 200 no son del todo malos y los valores a partir de 200 son malos.

- El tiempo el tiempo máximo que un residente pasa en una unidad para la Residencia 1 se espera que sea menor o igual a 365. Los valores comprendidos entre 365 y 730 no son del todo malos y los valores a partir de 730 son malos.
- El número medio de visitas por residente para la Residencia 1 se espera que sea mayor o igual que 1. Los valores comprendidos entre 0.8 y 1 no son del todo malos y los valores inferiores a 0.8 son malos.
- El número medio de visitas por residente para la Residencia 2 se espera que sea mayor o igual que 2. Los valores comprendidos entre 1 y 2 no son del todo malos y los valores inferiores a 1 son malos.

Para añadir los KPIs debemos utilizar la lista desplegable que aparece en la webpart. En nuestro caso para la definición de los KPIs utilizaremos los datos existentes en un libro de Excel.

La Figura 6.82. muestra la definición del KPI del tamaño medio de una úlcera para la residencia 2.

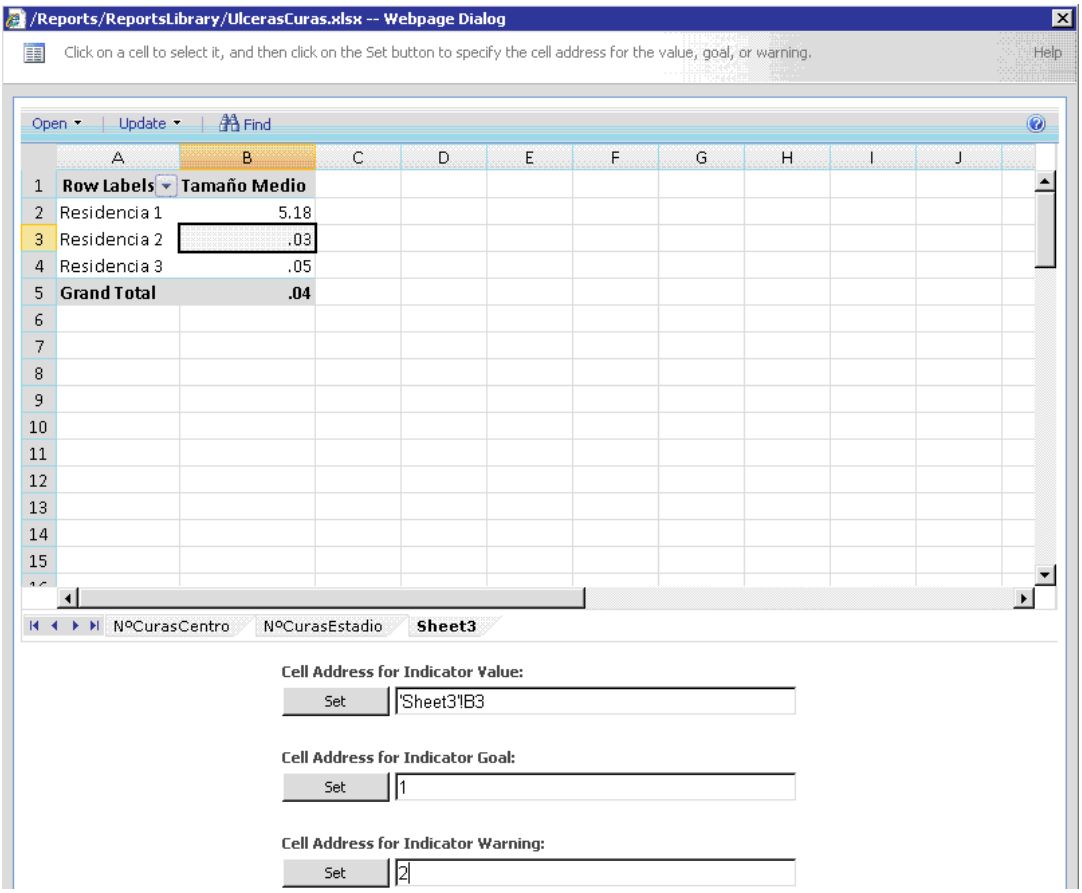


Figura 6.82. Definición KPI del tamaño medio de una úlcera para la Residencia 2

Después de añadir todos los KPIs con sus correspondientes valores, la webpart quedaría como se muestra en la Figura 6.83.

Figura 6.83. Lista de KPIs definidos para los indicadores

Según esto, los KPIs definidos no se alcanzan en la mayoría de los casos, ya que de 8 que hemos definido solamente 3 están en verde, lo cual indica que hemos alcanzado ese objetivo. Los KPIs de color amarillo están próximos a alcanzar el objetivo mientras que los de color rojo, están bastante lejos de alcanzar dicho objetivo.

Dicho esto y tras finalizar la creación del dashboard, podemos dar el proceso de desarrollo e implementación de los indicadores por finalizado.

6.6. Pruebas automatizadas de indicadores

Una vez implementados los tres indicadores, la siguiente actividad en el proceso de desarrollo es la fase de automatización. En este punto se van a automatizar cada una de las pruebas de aceptación redactadas en el documento de análisis del indicador de Úlceras Curas, para garantizar y validar el correcto funcionamiento del mismo.

Antes de comenzar con la implementación de las pruebas, es necesario tener claros algunos conceptos propios de RFT como: *script*, *datapool*, *object map*, etc… así como conocer los pasos de creación, grabación y ejecución de un script vistos en el tutorial de Rational Functional Tester.

Dada la gran cantidad de scripts de pruebas que pueden llegar a implementarse en el proceso de desarrollo de indicadores, es necesario definir un mecanismo que permita la reutilización de las mismos. A continuación vamos a comentar varias ideas y estrategias para abordar la implementación de pruebas automatizas reutilizables con RFT.

Estrategia de implementación de scripts reutilizables

Uno de los objetivos en la automatización de pruebas es la implementación de scripts reutilizables, los cuales mediante su combinación permitan realizar pruebas. Este objetivo es muy importante porque evita la duplicación de código y mejora el mantenimiento de los scripts y de las pruebas. La reutilización de los scripts es primordial, ya que si hay cambios en la interfaz, hay que modificar los scripts que contengan la interacción con la parte de la interfaz que cambia, por tanto, cuantos menos scripts se tengan de esa interfaz, menos scripts habrá que modificar.

A continuación se va a razonar como ha ido evolucionando la estrategia para la implementación de scripts y se comentará las razones del porqué de su rechazo o de su aceptación.

Una primera idea de implementación de scripts sería que cada prueba tuviera un script asociado con una agrupación de datos privada. El script contendría la interacción con las interfaces por las que pasa la prueba. Solo serviría para ejecutar esa prueba y mediante la agrupación de datos probar distintas combinaciones. Sin embargo esta forma de implementar scripts no es una buena solución ya que no favorece la reutilización, y como se ha comentado antes, si hay cambios en la interfaz habría que cambiar todas las pruebas que pasan por ella.

Figura 6.84. Esquema de la primera aproximación de la estructura de implementación de scripts

Por tanto, si se pretende reutilizar scripts hay que "partirlos". Cada vez que se implemente una prueba se tiene que hacer un script que incluya una composición de trozos de scripts. Es decir, que en lugar de tener un script que contenga toda la interacción con la aplicación, se tendrán scripts que interaccionen con partes de la interfaz. Con lo cual, cuando se quiera implementar una prueba solo habrá que hacer una combinación de esos scripts y el script resultante será el que ejecute la prueba. Este script se le llamará script Manejador.

Los scripts que invocará el Manejador son scripts utilitarios encargados por ejemplo de abrir el menu o cerrar la ventana y scripts de introducción de datos o también llamados Interactuadores. Cada script Interactuador tiene una agrupación de datos privada y se le invoca pasándole como argumento el número de registro que debe ejecutar. Los scripts Interactuadores suelen tener agrupaciones de datos con muchos registros. Esto puede ser un inconveniente, ya que la agrupación de datos será enorme porque contendrá todas las posibles combinaciones de datos de las pruebas.

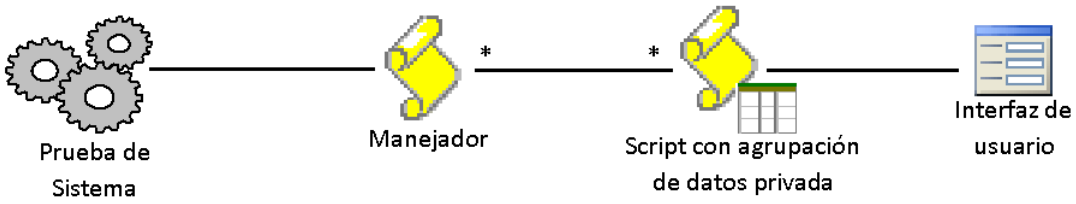


Figura 6.85. Esquema de la segunda aproximación de la estructura de implementación de scripts

Para evitar que las agrupaciones de datos sean tan grandes, también se pueden fragmentar y en lugar de ser privadas pueden ser compartidas. Con lo cual, el Manejador puede invocar al script indicando la agrupación de datos que debe utilizar en tiempo de ejecución.

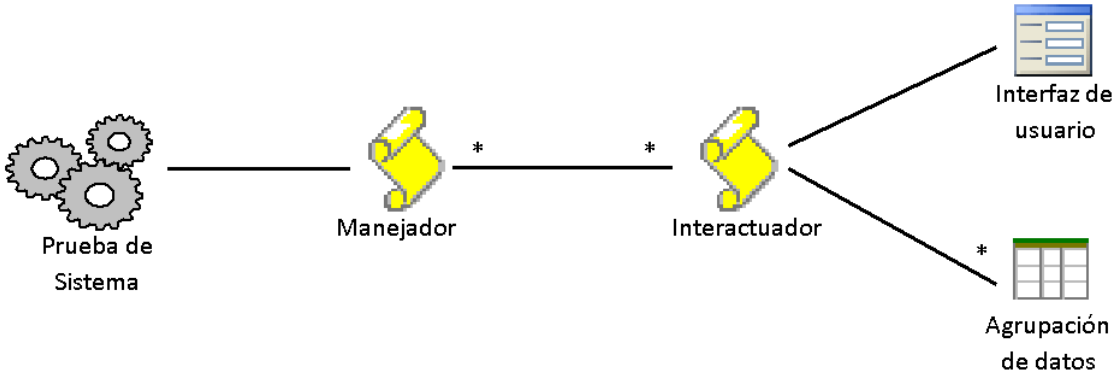


Figura 6.86. Esquema de la tercera aproximación de la estructura de implementación de scripts

En nuestro caso, la estrategia usada en la mayoria de implementaciones de scripts es la segunda aproximación, debido a que el número de registros de la agrupación de datos no es muy elevado. Sin embargo, en situaciones en las cuales se requiera una gran combinación de registros, seria recomendable utilizar el esquema de la tercera aproximación.

Personalización de scripts

Se han creado tres clases personalizadas, una para los scripts Manejadores, otra para los scripts Interactuadores y otra que proporciona métodos comunes a las dos clases anteriores.

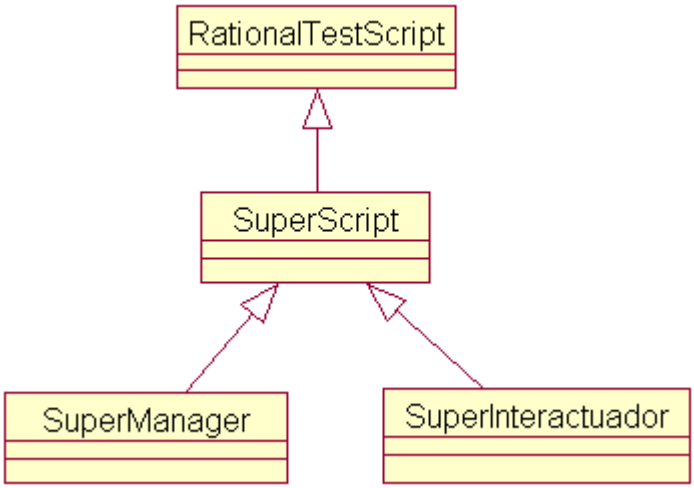


Figura 6.87. Jerarquía de herencia de los scripts personalizados

A continuación se describe de forma más detallada cada uno de los scripts personalizados:

- **SuperScript**: clase que hereda directamente de la clase RationalTestScript (proporciona la funcionalidad y sirve de raíz para la jerarquía de herencia de todos los scripts). Tiene una serie de propiedades y métodos que pueden utilizar los script. Si el script no es un script Manager o no utiliza una agrupación de datos para interactuar con una interfaz, puede heredar de esta clase.

  Entre los métodos que tiene definidos que hacen referencia a aplicaciones externas, aplicación que no está en la herramienta de pruebas, podemos destacar tres:
  - ConfigurarImprimir: configura una impresora virtual pasándole como parámetro la ruta y nombre de la imagen a generar.
  - ImprimirExcel: imprime el resultado de un fichero excel como imagen mediante una impresora virtual, guardando la imagen con el nombre y ruta anteriormente establecido.
  - CompararImagenes: compara dos imágenes, una de ellas previamente capturada y validada por los testers. Como resultado, si la comparación falla, creará una imagen mostrando en color rojo en que se diferencian, de forma que el tester pueda saber rápidamente que es lo que falla

- **SuperManager:** clase que hereda de SuperScript. Esta es la clase que deben heredar todos los scripts Managers. Tiene una serie de características utilizadas especialmente en los scripts que ejecuten las pruebas como por ejemplo: poder cambiar la fecha de sistema., saber en cuanto tiempo se ha ejecutado la prueba o en caso de fallar la prueba, matar todos los procesos invocados en esa prueba, así la ejecución de la siguiente prueba se podrá llevar a cabo.

  Entre los métodos que tiene definidos que hacen referencia a aplicaciones externas destacamos los siguientes:
  - RestaurarBD: restaura una copia de seguridad de la base de datos de ResiPlus y del almacén de datos de los indicadores.
  - ProcesarCubos: procesa los cubos de los indicadores para que obtengan los nuevos valores del almacén de datos.

  Este script suele ir aumentando de tamaño, debido a que actúa como contenedor de funciones y se va actualizando tras cada nueva grabación. Ejemplos: abrirResiplus, loguearUsuario, etc. Actualmente se está estudiando una aproximación orientada a objetos donde ir situando cada una de las funciones pertinentes.

- **SuperInteractuador:** clase que hereda de SuperScript. Esta es la clase que deben heredar los script que interactúan con datos con la interfaz, es decir que deben tener una agrupación de datos para ejecutarse. Este tipo de script siempre necesita la ruta de la agrupación de datos para poder usarla y el número de la instancia que debe ejecutar. Los script que son de interfaces en los que haya que introducir muy

pocos datos, no es necesario que se utilice una agrupación de datos, sino que basta con pasárselos mediante los argumentos y por lo tanto no tienen que heredar de esta clase.

Una vez conocida la estrategia de implementación y los scripts personalizados que van a utilizarse, a continuación vamos a ver paso a paso el proceso de automatización del indicador de Úlceras Curas.

Paso 1: Organización del proyecto

Antes de comenzar la grabación, hay que diseñar la organización del proyecto. El proyecto se estructura en las siguientes carpetas:

- La carpeta BDAS contiene las copias de seguridad de las bases de datos de Resiplus y del almacén de datos.

- La carpeta Clases incluye las clases que se van a utilizar, como SuperScript, SuperManager y SuperInteractuador.

- La carpeta Scripts, que a su vez se divide en las siguientes carpetas:
    - La carpeta RQ son los requisitos. Cada requisito está relacionado con una parte de la interfaz gráfica de la aplicación. Por tanto en está carpeta almacenamos los scripts de tipo SuperInteractuador y SuperScript.
    - La carpeta ST representa las pruebas de aceptación, que a su vez se corresponden con las pruebas de sistema. Contendrá un listado de carpetas con el nombre de cada indicador automatizado. En la carpeta de cada indicador, se almacena un fichero .odc que representa una conexión al cubo, y que es utilizado posteriormente para obtener una imagen del fichero excel que represente un punto de verificación. Debido a que en cada indicador se realizan dos tipos de pruebas, una histórica, que realiza una carga de todos los datos, y otra diaria, en la que insertamos nuevos datos, cada indicador estará formado por dos carpetas: diario e histórico. En estas carpetas se guardan los scripts de tipo SuperManager. Cada carpeta diaria e histórica contiene una carpeta VP, en la cual se almacenan los puntos de verificación.
    - En la carpeta Utilidades, están los script utilitarios, que son el script que maneja el programa comparador de imágenes, el que maneja la configuración de la impresora virtual, y el que contiene la interacción con el programa que restaura la base de datos de la aplicación.

La organización en carpetas que representa el proyecto quedaría de la siguiente manera:

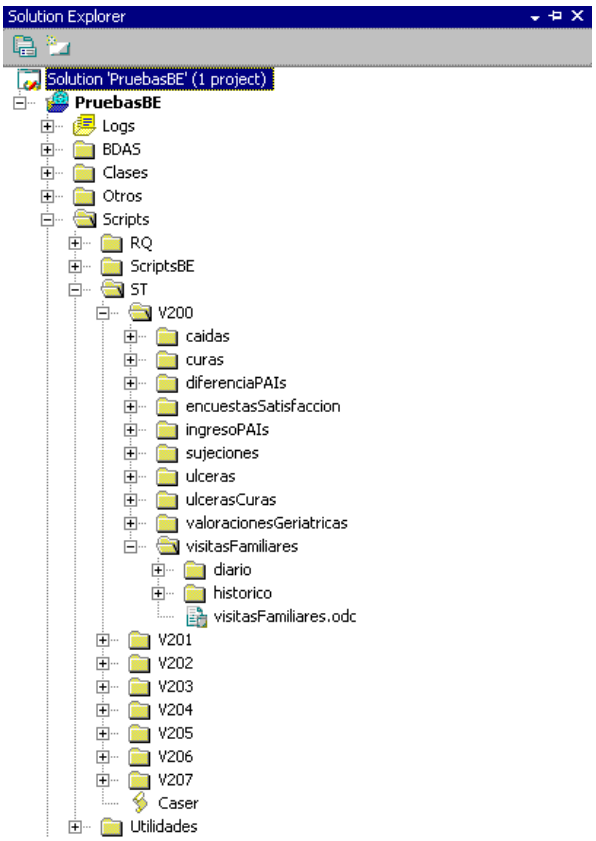


Figura 6.88. Organización proyecto RFT

Paso 2: Identificar nuevas interfaces a grabar

En el documento de ayuda de Úlceras Curas, explicado en el punto 6.4, se detallan las interfaces relacionadas con el indicador, por lo que se deben grabar los scripts necesarios para el acceso e introducción de datos en cada una de estas interfaces.

En la mayoria de casos, muchas de las interfaces se corresponden con dimensiones que ya están grabadas, tales como Centro, Residente, Tipología y Carácter Plaza. Esto es debido a que son dimensiones genéricas utilizadas en todos los indicadores. En el caso de que todas las interfaces estuvieran grabadas, los pasos de grabación de scripts, que veremos a continuación, no se realizarían, y únicamente crearíamos los scripts Manejadores.

En esta ocasión, obviamos los pasos de grabación de los scripts de dimensiones debido a que son los mismos pasos que veremos a continuación en la grabación del indicador en sí.

Paso 3: Grabar script de acceso al menu del indicador

Según la estructura utilizada en nuestro proyecto, debemos entrar a la carpeta RQ y crear una carpeta nueva llamada ulcera. Dentro de esta carpeta debemos crear un nuevo script. Para ello debemos pulsar con el botón derecho del ratón sobre la carpeta que acabamos de crear y nos aparecerá un menú contextual. Pulsamos sobre la opción **Add Script Using Recorder** para crear un nuevo script de grabación.

A continuación nos aparecerá un cuadro de diálogo para escribir el nombre del script. En nuestro caso, debe llamarse menuUlcera para saber que este script es el que usaremos para acceder a la ventana de Úlceras.

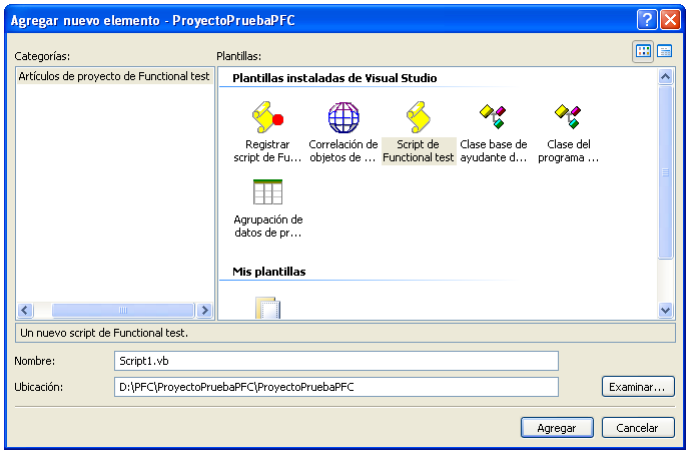


Figura 6.89. Pantalla para agregar un nuevo elemento

Después de renombrar el script, pulsaremos sobre el botón **Agregar** para continuar el proceso de creación. En ese momento nos aparecerá una ventana de configuración del script, en la que debemos seleccionar si el mapa de objetos será privado o público, la clase base del script, si el datapool es privado o no y otras características.

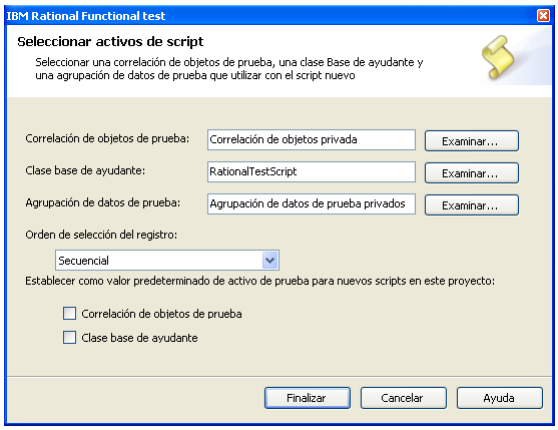


Figura 6.90. Pantalla de configuración del script

En nuestro caso, solo vamos a cambiar la clase base del script y las otras opciones las dejaremos por defecto. Para cambiar la clase debemos pulsar sobre el botón **Examinar** de la fila donde se encuentra la opción **Clase base de ayudante**. Como este es un script en el que no se van a crear datapools y solo es de interactuación, la clase base que utilizaremos es del tipo SuperScript, por lo que seleccionaremos esta y pulsaremos el botón **Aceptar** y posteriormente el botón **Finalizar**.

Una vez hecho esto, nos aparecerá el asistente de grabación, el cual ya estará en disposición de grabar. En esta situación deberíamos tener ya la aplicación a grabar abierta por lo que si no es así debemos pausar la grabación, abrirla y dejar la aplicación preparada para grabar.

En ese momento debemos pulsar sobre la opción Enfermería de la barra de menús lateral, posteriormente en la pestaña Controles y por último seleccionamos la opción Úlceras del árbol Controles. Otra posibilidad, que es la usada en nuestra prueba, es el uso del teclado para moverse por los nodos del árbol.

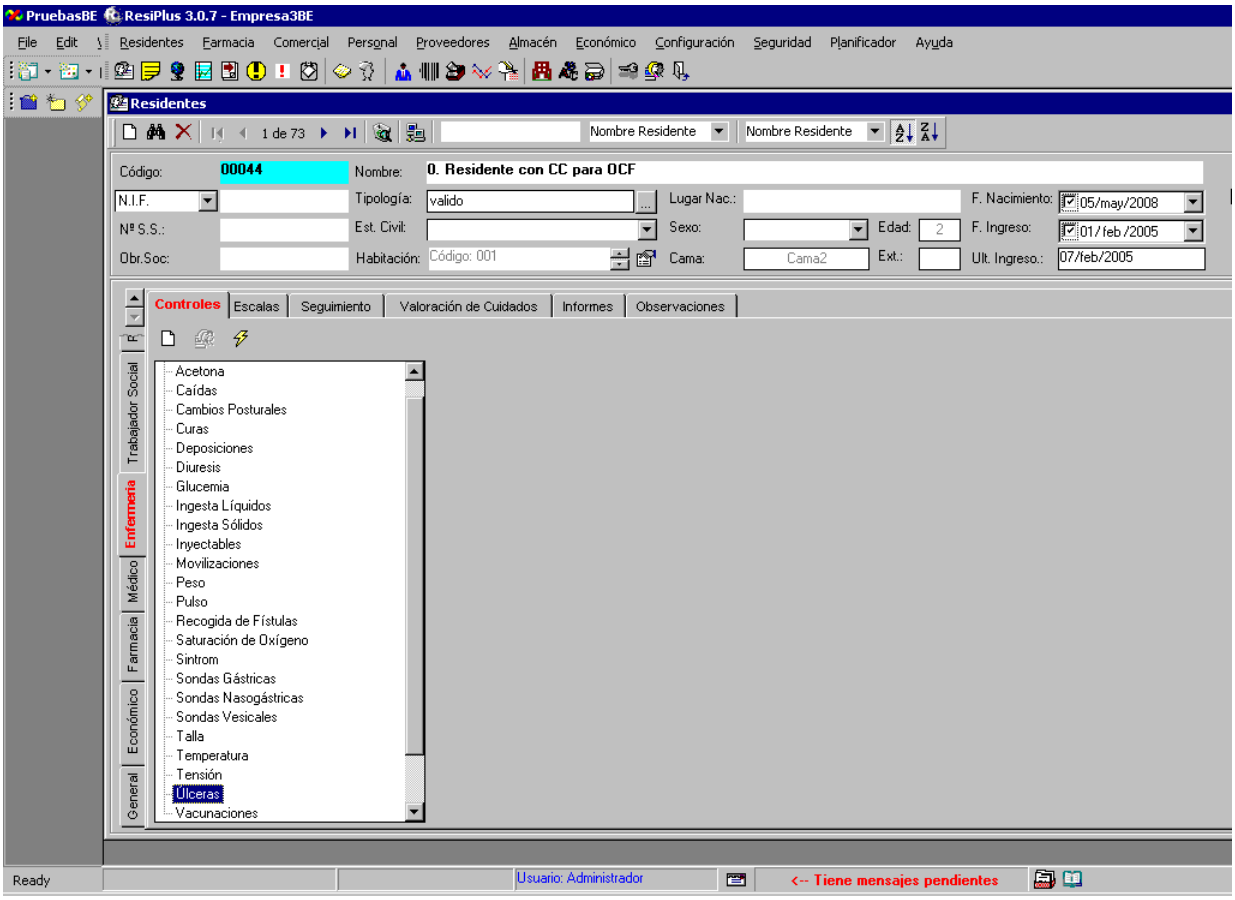


Figura 6.91. Menu Úlceras

Una vez hayamos terminado, debemos pulsar sobre el botón de **Stop** del asistente de grabación para finalizar la grabación. De esta forma tenemos ya el script creado y a punto para ser utilizado.

Paso 4: Grabar script de inserción de datos en la interfaz del indicador

Para crear el script que interactuará con la interfaz de Úlceras e insertará los datos, seguiremos parte del proceso anterior. Primero pulsaremos con el botón derecho del ratón la carpeta de ulceras y en el menú contextual pulsaremos sobre la opción **Add Script Using Recorder**.

Renombraremos el script llamandolo anyadirUlcera, ya que en este script se insertarán datos en la interfaz, y pulsaremos sobre el botón **Agregar**. En ese momento nos aparecerá una ventana de configuración del script, en la que debemos cambiar la clase base, pero en este caso, el tipo de clase será SuperInteractuador.

Una vez hayamos seguido todos los pasos anteriores, como antes, nos aparecerá el asistente de grabación por lo que ya debemos tener la interfaz de Úlceras visible en Resiplus.

Para que el script haga lo que debe hacer, que es insertar la fecha de la úlcera, procedencia, tipo, localización, fecha de la cura, estadio, tamaño y si está curada o no, deberemos hacer nosotros lo mismo, es decir, insertar una úlcera tal y como lo deberá hacer después el script de forma autónoma.

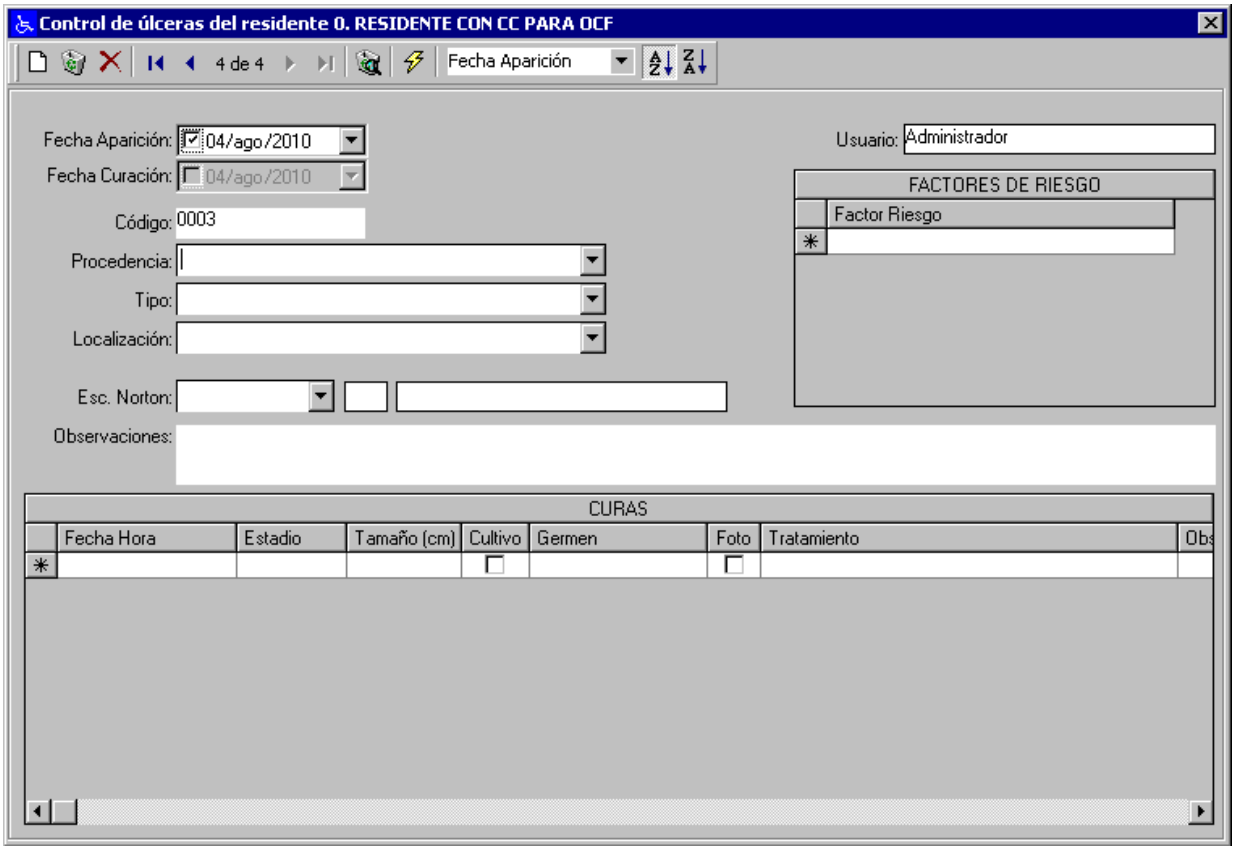


Figura 6.92. Interfaz de introducción de úlceras y curas

Una vez hayamos terminado, deberemos pulsar sobre el botón de **Stop** del asistente. Hecho esto ya tendremos el script grabado, pero nos hará falta crear el datapool y modificar el script con los metodos necesarios para la inserción de los datos del datapool en la interfaz.

Paso 5: Crear datapool con los datos necesarios para la prueba

Para crear un datapool privado, debemos pulsar con el botón derecho del ratón la carpeta de ulceras y en el menú contextual pulsaremos sobre la opción **Add Test Datapool**. Posteriormente nos aparecerá la ventana con la que nombraremos al datapool. Como este datapool va a ser privado, da igual el nombre a asignar, por lo que podemos pulsar sobre el botón **Agregar**.

Una vez tenemos el datapool creado, debemos añadir las columnas y filas necesarias que usaremos en las pruebas.

Private Test Datapool

	FechaUlcera	Procedencia	Tipo	Localizacion	FechaCura	Estadio	Tamanyo	Curada
0	23/07/2007	Domicilio	Grado I	cabeza				False
1	15/07/2008	Otros	Grado V	Corazon				False
2	15/07/2008							False
3	15/07/2008	Hospital						False
4	15/07/2008		Grado III					False
5	15/07/2008			pie decha				False
6	03/03/2008	Domicilio	Grado I	cabeza	03/03/2008 00:00	uno	23	False
7	15/07/2008	Hospital	Grado II	tronco	15/07/2008 00:00	Cinco	6	True
8	15/07/2008				15/07/2008 00:00			False
9	15/07/2008				15/07/2008 00:00		2,34	False
10	15/07/2008				15/07/2008 00:00	dos	-999	True
11	15/07/2008				15/07/2008 00:00	dos		False

Figura 6.93. Datapool de Úlceras Curas

Por último, tenemos que modificar el script para que acepte los valores del datapool. Las siguientes lineas de código muestran el resultado del script tras la modificación.

```
#Region " Script Header "
' Functional Test Script
' author juan.perez
Imports Microsoft.VisualBasic
Imports Rational.Test.Ft
Imports Rational.Test.Ft.Object.Interfaces
Imports Rational.Test.Ft.Object.Interfaces.SAP
Imports Rational.Test.Ft.Object.Interfaces.Siebel
Imports Rational.Test.Ft.Script
Imports Rational.Test.Ft.Value
Imports Rational.Test.Ft.Vp
#End Region
Namespace Scripts.RQ.V200.ulcera

    Public Class anyadirUlcera
        Inherits anyadirUlceraHelper

        Public Function TestMain(ByVal args() As Object) As Object

            ' Window: ResiPlus.exe: ResiPlus 2.8.407 - Empresa3BE -
[Residentes]
            SSCommandWndClassWindow2().Click(AtPoint(12,12))

            ' Window: ResiPlus.exe: ResiPlus 2.8.407 - Empresa3BE -
[Residentes]
            'SSCommandWndClassWindow().Click(AtPoint(11,12))

            ' Window: ResiPlus.exe: Control de úlceras del residente 0.
RESIDENTE CON CC PARA OCF
            ControlDeÚlcerasDelResidente0R().InputKeys("^{F1}")
            _17Sep2008Window().Click(AtPoint(22,8))

ControlDeÚlcerasDelResidente0R().InputChars(DpString("FechaUlcera"))

            ' Window: ResiPlus.exe: Control de úlceras del residente 0.
RESIDENTE CON CC PARA OCF
            ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
            ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")

ControlDeÚlcerasDelResidente0R().InputChars(DpString("Procedencia"))
            ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
```

```
            ControlDeÚlcerasDelResidente0R().InputChars(DpString("Tipo"))
            ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")

ControlDeÚlcerasDelResidente0R().InputChars(DpString("Localizacion"))

            If Not DpString("FechaCura") Is Nothing AndAlso Not
DpString("FechaCura").Equals("") Then

                CURASWindow().Click(AtPoint(31, 44))

ControlDeÚlcerasDelResidente0R().InputKeys("{ExtHome}+{ExtEnd}")

ControlDeÚlcerasDelResidente0R().InputChars(DpString("FechaCura"))
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")

ControlDeÚlcerasDelResidente0R().InputChars(DpString("Estadio"))
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")

ControlDeÚlcerasDelResidente0R().InputChars(DpString("Tamanyo"))
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")
                ControlDeÚlcerasDelResidente0R().InputKeys("{TAB}")

                'mover a curadas

                If DpBoolean("Curada") Then

                    Dim fechaCuracion As Date =
Date.Parse(DpString("FechaCura"))

                    ControlDeÚlcerasDelResidente0R(ANY,
MAY_EXIT).InputKeys("^{F3}")
                    SíButton().Click(AtPoint(36, 10))

                    SolicitudDeFechaWindow().InputKeys("{ExtRight}")

SolicitudDeFechaWindow().InputChars(fechaCuracion.Day.ToString())
                    SolicitudDeFechaWindow().InputKeys("{ExtRight}")

SolicitudDeFechaWindow().InputChars(fechaCuracion.Month.ToString())
                    SolicitudDeFechaWindow().InputKeys("{ExtRight}")

SolicitudDeFechaWindow().InputChars(fechaCuracion.Year.ToString())

                    AceptarButton().Click(AtPoint(47, 10))

                End If

            End If

            ControlDeÚlcerasDelResidente0R(ANY,
MAY_EXIT).Click(CLOSE_BUTTON)

            Return Nothing
        End Function
    End Class
End Namespace
```

Figura 6.94. Script anyadirUlcera

Tras la creación de todos los SuperScripts y SuperInteractuadores el proyecto quedaría de la siguiente forma:

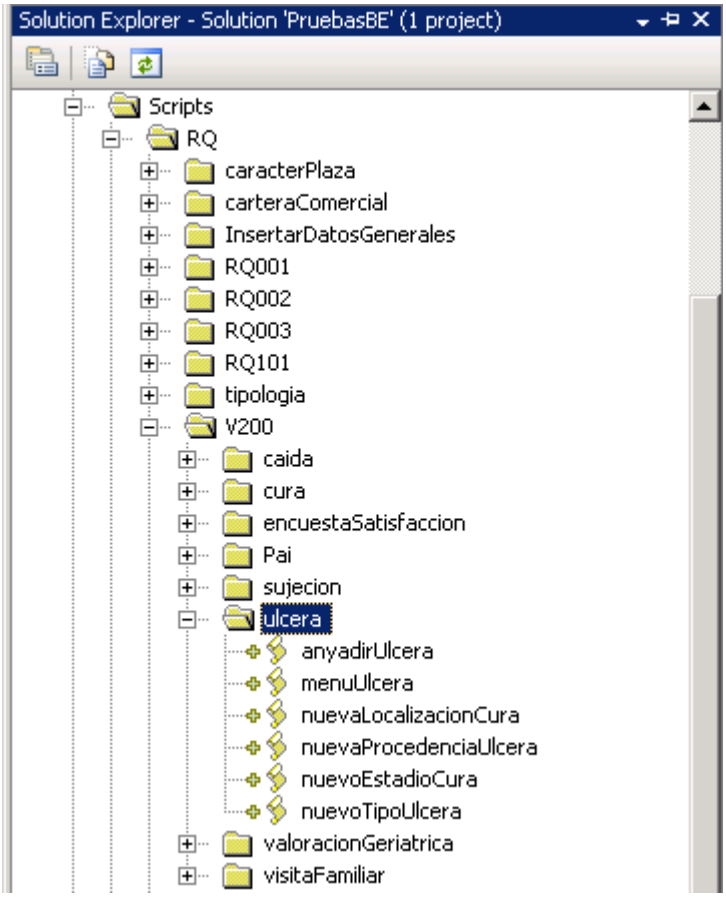


Figura 6.95. Resultado carpeta RQ

Paso 6: Crear los métodos que usaremos en las pruebas en la clase SuperManager

En la clase SuperManager crearemos los métodos que se llaman desde los scripts de las pruebas. Estos métodos invocarán los scripts que interactuan con la aplicación, tanto los SuperScripts como los SuperInteractuadores.

En el caso de nuestro indicador, necesitamos un método que abra la ventana de úlceras de la aplicación y que inserte una línea del datapool en la ventana. El resto de métodos que a continuación mostramos son referentes a las dimensiones del indicador.

```
#Region "Ulceras"

        Public Sub anyadirUlcera(ByVal posicionUlcera As Integer, ByVal
menuUlcera As Boolean, ByVal volverMenuResidente As Boolean)
            'If menuUlcera Then
CallScript("Scripts.RQ.V200.ulcera.menuUlcera")
            If menuUlcera Then
CallScript("Scripts.RQ.V205.antecedentes.menuControles", New Object()
{"ulceras"})
            CallInteractuador("Scripts.RQ.V200.ulcera.anyadirUlcera",
posicionUlcera)
            If volverMenuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente", New Object() {"volver"})
```

```
        End Sub

        Public Sub nuevaLocalizacionUlceraCura(ByVal posicionUlcera As
Integer, ByVal menuUlcera As Boolean, ByVal menuResidente As Boolean, ByVal
volverMenuResidente As Boolean)
            If menuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente")
            'If menuUlcera Then
CallScript("Scripts.RQ.V200.ulcera.menuUlcera")
            If menuUlcera Then
CallScript("Scripts.RQ.V205.antecedentes.menuControles", New Object()
{"ulceras"})

CallInteractuador("Scripts.RQ.V200.ulcera.nuevaLocalizacionCura",
posicionUlcera)
            If volverMenuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente", New Object() {"volver"})
        End Sub

        Public Sub nuevaProcedenciaUlcera(ByVal posicionUlcera As Integer,
ByVal menuUlcera As Boolean, ByVal menuResidente As Boolean, ByVal
volverMenuResidente As Boolean)
            If menuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente")
            'If menuUlcera Then
CallScript("Scripts.RQ.V200.ulcera.menuUlcera")
            If menuUlcera Then
CallScript("Scripts.RQ.V205.antecedentes.menuControles", New Object()
{"ulceras"})

CallInteractuador("Scripts.RQ.V200.ulcera.nuevaProcedenciaUlcera",
posicionUlcera)
            If volverMenuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente", New Object() {"volver"})
        End Sub

        Public Sub nuevoEstadioUlceraCura(ByVal posicionUlcera As Integer,
ByVal menuUlcera As Boolean, ByVal menuResidente As Boolean, ByVal
volverMenuResidente As Boolean)
            If menuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente")
            'If menuUlcera Then
CallScript("Scripts.RQ.V200.ulcera.menuUlcera")
            If menuUlcera Then
CallScript("Scripts.RQ.V205.antecedentes.menuControles", New Object()
{"ulceras"})
            CallInteractuador("Scripts.RQ.V200.ulcera.nuevoEstadioCura",
posicionUlcera)
            If volverMenuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente", New Object() {"volver"})
        End Sub

        Public Sub nuevoTipoUlcera(ByVal posicionUlcera As Integer, ByVal
menuUlcera As Boolean, ByVal menuResidente As Boolean, ByVal
volverMenuResidente As Boolean)
            If menuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente")
            'If menuUlcera Then
CallScript("Scripts.RQ.V200.ulcera.menuUlcera")
            If menuUlcera Then
CallScript("Scripts.RQ.V205.antecedentes.menuControles", New Object()
{"ulceras"})
```

```
            CallInteractuador("Scripts.RQ.V200.ulcera.nuevoTipoUlcera",
posicionUlcera)
            If volverMenuResidente Then
CallScript("Scripts.RQ.RQ101.menuResidente", New Object() {"volver"})
        End Sub

        Public Sub nuevaLocalizacionUlceraCura(ByVal posicionUlcera As
Integer)

CallScript("Scripts.RQ.InsertarDatosGenerales.InsertarDatosGenerales", New
Object() {3, 3, "Scripts.RQ.V200.ulcera.nuevaLocalizacionCura",
posicionUlcera})
        End Sub
        Public Sub nuevaProcedenciaUlcera(ByVal posicionUlcera As Integer)

CallScript("Scripts.RQ.InsertarDatosGenerales.InsertarDatosGenerales", New
Object() {3, 3, "Scripts.RQ.V200.ulcera.nuevaProcedenciaUlcera",
posicionUlcera})
        End Sub
        Public Sub nuevoEstadioUlceraCura(ByVal posicionUlcera As Integer)

CallScript("Scripts.RQ.InsertarDatosGenerales.InsertarDatosGenerales", New
Object() {3, 3, "Scripts.RQ.V200.ulcera.nuevoEstadioCura", posicionUlcera})
        End Sub
        Public Sub nuevoTipoUlcera(ByVal posicionUlcera As Integer)

CallScript("Scripts.RQ.InsertarDatosGenerales.InsertarDatosGenerales", New
Object() {3, 3, "Scripts.RQ.V200.ulcera.nuevoTipoUlcera", posicionUlcera})
        End Sub


#End Region
```

Figura 6.96. Métodos de Úlceras Curas

Paso 7: Crear los scripts de pruebas

Según la estructura utilizada en nuestro proyecto, debemos entrar a la carpeta ST y crear una carpeta nueva llamada ulcerasCuras. Como explicamos anteriormente, a cada indicador se le realizan dos tipos de prueba de carga, una histórica y otra diaria. Por tanto, crearemos dos subcarpetas más: historico y diario

En este caso, para no extendernos demasiado, únicamente crearemos el script de prueba diario. Para esto, pulsamos con el botón derecho del ratón la carpeta diario y en el menú contextual pulsamos sobre la opción **Add Empty Script**. Posteriormente nos aparecerá la ventana con la que nombraremos al script de prueba. En este caso lo llamaremos diario y pulsaremos sobre el botón **Agregar**.

A continuación aparece la ventana de configuración del script, en la que solo vamos a cambiar su clase base. Para ello, debemos pulsar sobre el botón **Examinar** de la fila donde se encuentra la opción **Clase base de ayudante** y seleccionar el tipo SuperManager. De esta forma, el script podrá invocar a los métodos creados en el paso anterior. Luego pulsaremos el botón **Aceptar** y posteriormente el botón **Finalizar**.

Llegados a este punto, solo falta invocar los metodos necesarios que permitan aplicar las pruebas de aceptación definidas en el documento de análisis, punto 6.4.

```
#Region " Script Header "
' Functional Test Script
' author juan.perez
Imports Microsoft.VisualBasic
Imports Rational.Test.Ft
Imports Rational.Test.Ft.Object.Interfaces
Imports Rational.Test.Ft.Object.Interfaces.SAP
Imports Rational.Test.Ft.Object.Interfaces.Siebel
Imports Rational.Test.Ft.Script
Imports Rational.Test.Ft.Value
Imports Rational.Test.Ft.Vp
#End Region

Imports System.Collections

Namespace Scripts.ST.V200.ulcerasCuras.diario

    Public Class diario
        Inherits diarioHelper

        Public Function TestMain(ByVal args() As Object) As Object

            restaurarBD("ulcerasCuras", True, "Uce DW", 0)

            abrirResiplus(New Date(2008, 7, 15))
            logearUsuario(0)
            seleccionarEmpresa(2) 'Residencia 3

            nuevoEstadioUlceraCura(0, True, True, True) 'uno =
            nuevoEstadioUlceraCura(2, True, True, True) 'Cinco = 5

            nuevoResidente(1, False) 'Juan Vicente Pez Palanca
            'añadir cura sin codigo unificado en Resilus y fecha pasada
            anyadirUlcera(6, True, True) 'uno (No Definidos), FechaUlcera y
Cura= 03/03/2008 (2 curas historico)

            buscarResidente("Residente 0002", True)
            'introducir curas con los nuevos valores
            anyadirUlcera(7, True, True) 'estadio=Cinco CURADA

            nuevoResidente(5, False) 'Victor Morente Garc
            'introducir curas con valores vacios
            anyadirUlcera(8, True, False) 'estadio y tamanyo=   -
            anyadirUlcera(9, False, False) 'estadio=- y tamanyo= 2.34
            anyadirUlcera(10, False, False) 'estadio=dos y tamanyo= -999
CURADA
            anyadirUlcera(11, False, False) 'estadio=dos y tamanyo= -

            cerrarResiplus()

            '********** modificar Uce DW *********
            Dim consulta(0) As String

            'los atributos nuevos creados anteriormente
            consulta(0) = "Insert into Dim_CurasEstadio (CodigoUnificado,
CurasEstadio) values ('5','Cinco')"

            For Each aux As String In consulta
                CallScript("Scripts.Utilidades.EjecutarSQL", New Object()
{aux, "Uce DW", Nothing, True})
            Next
```

```
            abrirResiplusBE()
            lanzaExportacion("CASER Indicadores")
            cerrarResiplusBE()

            '********** procesar los cubos **********
            procesarCubos()

            configurarImprimir("ulcerasCuras")
            imprimirExcel("ulcerasCuras")
            CompararImagenes("ulcerasCuras")

        Return Nothing
        End Function
    End Class
End Namespace
```

Figura 6.97. Métodos de Úlceras Curas

Paso 8: Crear puntos de verificación

En primer lugar debemos almacenar el fichero .odc, que representa una conexión al cubo del indicador, en la carpeta ulcerasCuras. A continuación, creamos una subcarpeta llamada VP (Verification Point) dentro de la carpeta diario del indicador. En esta carpeta se almacenarán tanto los ficheros excel, correctamente configurados para visualizar la información del indicador, como las imágenes generadas a partir de los mismos.

Más información de cómo configurar un archivo excel para que se conecte al cubo de un indicador en el apéndice, Anexo H.

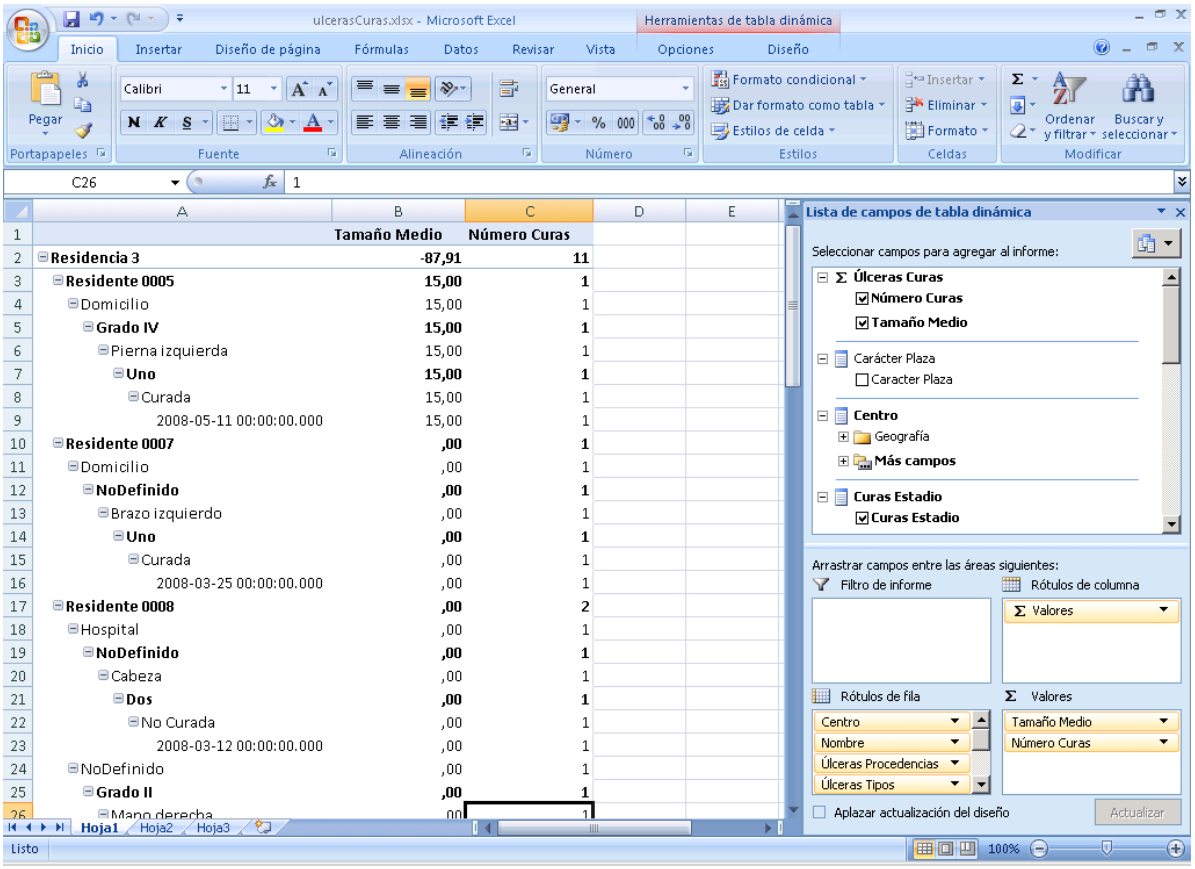


Figura 6.98. Excel Úlceras Curas

Una vez configurado el archivo excel, hay que ejecutar el script de prueba comentando el método CompararImagenes para poder generar las imagenes resultantes del mismo. Cuando se haya comprobado que las imágenes son correctas y asegurado que el script se reproduce correctamente, hay que renombrar las imágenes añadiendo al nombre "_VP", ya que esta es la notación seguida para poder distinguir y comparar las imágenes. Dichas imágenes se guardan dentro de la carpeta VP.

Finalmente, la carpeta ST del proyecto quedaría de la siguiente forma:

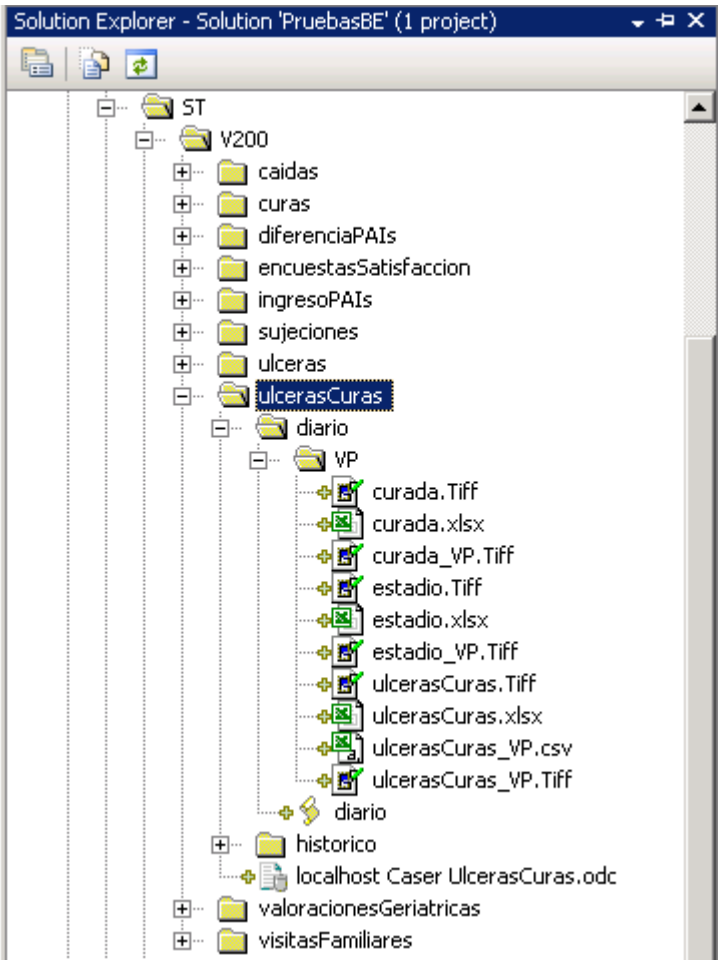


Figura 6.99. Resultado carpeta ST

Paso 9: Ejecutar la prueba

Cuando se haya acabado de implementar la automatización del indicador y verificar que funciona correctamente, las pruebas ya podrán ser ejecutadas posteriormente en regresión.

Capítulo 7.Conclusiones

Actualmente las herramientas de Business Intelligence suponen un apoyo fundamental en la toma de decisiones de los directivos de las empresas. Estas herramientas permiten la construcción de soluciones a medida y ofrecen de forma sencilla una potente explotación de la información generando una ventaja competitiva para la toma de decisiones.

La solución de Business Intelligence desarrollada en este proyecto es un ejemplo de solución hecha a medida. En la actualidad, dicha solución es utilizada por varios clientes con múltiples residencias. Debido a los buenos resultados, son varios los clientes que están interesados en soluciones similares.

En cuanto a los objetivos de este proyecto, podemos afirmar que se han cumplido con éxito:

- Se ha constatado que el marco tecnológico propuesto para el desarrollo de los indicadores era adecuado, ya que se han podido desarrollar los indicadores, reports y dashboards de forma fácil y los resultados obtenidos han sido relativamente buenos.
- Se ha validado la utilidad de la metodología TUNE-UP y su correspondiente workflow para el desarrollo de indicadores, puesto que nos ha guiado en todo el proceso de desarrollo.
- Se han desarrollado varios tipos de indicadores indicando las diferencias en la implementación existentes entre cada uno de ellos y se ha comprobado durante la creación de los informes asociados a cada uno de ellos, la gran cantidad de formas de explotar la información de dichos indicadores gracias a las herramientas OLAP.
- Se ha desarrollado un dashboard en el que se presenta la información de los indicadores desarrollados y los KPIs definidos para dichos indicadores, cerciorándonos así de la utilidad de ver la información de forma gráfica para entender mejor los datos.
- Se han automatizado las pruebas de aceptación de uno de los indicadores desarrollados, describiendo en detalle la estrategia de implementación utilizada con el objetivo de crear scripts de pruebas reutilizables.

Respecto a la experiencia adquirida en el desarrollo de indicadores son muchas las cosas a comentar pero a continuación especifico las más relevantes:

- En los inicios del desarrollo de indicadores nos centrábamos en la potencia de las tablas dinámicas pero con el paso del tiempo, la tendencia ha cambiado y ahora nos centramos en desarrollar dashboards y reports estáticos porque se ha comprobado que es lo que prefieren utilizar los usuarios.
- El número de indicadores desarrollados hasta el momento está en torno a los 60.
- El tipo de indicadores que predomina entre los desarrollados son aquellos que calculan totales en función de varias dimensiones. A continuación, con un número relativamente más bajo, nos encontramos con los indicadores que muestran evoluciones y finalmente, en muy pocos casos, indicadores que calculan diferencias de dias transcurridos entre dos eventos.
- El tiempo de desarrollo de los indicadores, incluyendo todas las actividades del WF, oscila entre las 10 horas y las 60 horas aproximadamente dependiendo de la dificultad para obtener los datos del indicador, el número de dimensiones, etc. Concretamente, el tiempo medio de desarrollo suele ser de unas 30 horas.
- En varias ocasiones ha sido necesario modificar un indicador que había dejado de funcionar debido a cambios realizados en el ERP.

- Los códigos unificados son bastante problemáticos porque en muchas ocasiones el cliente no los rellena y debido a esto los resultados obtenidos por el indicador carecen de interés. Además, cuando se trata de centros con muchas residencias, se ha comprobado que utilizan diferentes códigos unificados para hacer referencia a lo mismo, con lo que la unificación de datos no es posible.
- Existen indicadores implantados que no se utilizan bien porque el usuario encuentra demasiada compleja la interpretación de los datos o porque la persona encargada de trabajar con ellos todavía no se ha puesto a ello.
- El tiempo de carga de los datos de cada indicador varía considerablemente de un indicador a otro (podemos tener indicadores que se carguen en cuestión de segundos y otros que tarden 20 minutos), ya que algunos manejan millones de registros. Dicho esto, es de suponer que si tenemos que lanzar 30 indicadores para un cliente que cuenta con 10 centros, los tiempos de carga de los datos de todos estos indicadores para esos centros serán bastante elevados.
- A pesar de que los almacenes de datos están pensados para guardar información histórica de un período que varía de 2 a 10 años, nuestros clientes prefieren que la información almacenada comprenda desde los últimos años hasta la fecha actual. Esto es así porque para ellos resulta de gran utilidad poder consultar la información más reciente usando los reports definidos para los indicadores. Por este motivo, se realiza un refresco diario de los datos de los indicadores.
- El desarrollo de los indicadores lo lleva a cabo un equipo de 3 personas que ejercen los roles de analista, desarrollador y tester.

Personalmente este proyecto refleja el trabajo realizado durante los últimos 3 años y supone la maduración del proyecto final de carrera que realice en la escuela, Técnicas avanzadas de explotación de datos y su utilización en un caso de estudio real, en el cual me introducía en los conceptos de almacenes de datos, cubos OLAP, minería de datos, etc… haciendo uso de la tecnología proporcionada por Microsoft.

En cuanto a líneas de trabajo futuras, actualmente se está extendiendo el proyecto en el ámbito del desarrollo de reports estáticos incluidos en la norma UNE 158101, utilizando la tecnología Microsoft Reporting Services. Además se están incorporando mecanismos de seguridad en los indicadores para restringir el acceso a los reports de dichos indicadores e incluso restringir la visualización de parte de los datos. De esta forma se consigue que los usuarios sólo accedan a los reports cuyo permiso les ha sido concedido y que únicamente visualicen los datos que son de su interés. Junto a esto, se está investigando la manera de dar un servicio centralizado de soluciones BI a los clientes desde la empresa del ERP pero existe mucha problemática relacionada con la transmisión segura de datos entre la empresa cliente y la del ERP.

Referencias

[1] Bernabeu, R. Dario. *Data Warehousing y metodología Hefesto*. Julio 2008. Obtenido de http://www.dataprix.com/es/data-warehousing-hefesto

[2] Business Intelligence fácil. *Cuadrante mágico de Gartner para las plataformas BI*. Febrero 2010. Obtenido de http://www.businessintelligence.info/mercado/cuadrante-magico-2010.html

[3] Hernández Orallo, José. *Parte II: Almacenes de datos*. 2003. Obtenido de http://users.dsic.upv.es/~jorallo/cursoDWDM/dwdm-II.pdf

[4] Hotek, Mike. *Paso a Paso Microsoft SQL Server 2008.* Anaya Multimedia. 2009.

[5] Ibermatica. *Business Intelligence*. Obtenido de http://www.ibermatica.com/publicaciones/BusinessIntelligence.pdf

[6] Kimball, Ralph. *A Dimensional Modeling Manifesto*. 1997. Obtenido de http://www.ralphkimball.com/html/articles_search/articles1997/9708d15.html

[7] Kimball, Ralph; Ross, Margy. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* (Second Edition ed.).2002.

[8] Marante Estellés, María Isabel. *Planificación y seguimiento en proyectos de desarrollo y mantenimiento de software dirigido por la gestión de tiempos*. Diciembre 2009.

[9] Martínez Orol, Alfredo. *OLAP y el diseño de cubos*. Marzo 2007. Obtenido de http://www.gestiopolis.com/canales8/ger/olap-online-analytic-processing.htm

[10] Microsoft Business Intelligence. *Microsoft Business Intelligence en la economía actual*. Febrero 2009. Obtenido de http://www.microsoft.com/spain/bi/default.mspx

[11] Microsoft TechNet. *Microsoft SQL Server*. Obtenido de http://technet.microsoft.com/es-es/library/bb545450%28en-us%29.aspx

[12] Sinnexus. *¿Qué es Business Intelligence?.* http://www.sinnexus.com/business_intelligence/index.aspx

[13] Zvenger, Patricia.*Introducción al Soporte de Decisiones*. Diciembre 2005.

[14] Sánchez Barba, Hector. *Pruebas software automatizadas para garantizar la calidad de una aplicación industrial*. Junio 2008.

Anexos

En estos anexos se adjuntan algunos ejemplos de documentos utilizados durante el proceso de desarrollo de indicadores de negocio y se incluyen pequeños tutoriales de las herramientas utilizadas en dicho proceso.

Parte del material presentado en este capítulo ha sido extraído de [4] y [14].

Anexo A: Plantilla Indicador

Plantilla Indicadores

INFORMACIÓN DEL INDICADOR

Hecho: **Visitas familiares**

Descripción: **Este indicador muestra el número de visitas por residente y por día.**

Finalidad: **Se pretende conocer las personas con menos visitas para poder darles una mayor atención personal.**

Medidas: **Sumatorio y media del número de visitas familiares.**

Granularidad: **Diaria**

KPI's: **Mostraremos en rojo aquellos campos que muestren menos de 4 visitas por mes y residente.**

Valor por defecto**:**

INSTRUCCIONES

8. **Hecho**. Nombre de la variable que se desea analizar. Ejemplos: Visitas familiares.

9. **Descripción**. Escuetamente describe el hecho. Ejemplo: Este indicador muestra el número de visitas por residente y por día.

10. **Finalidad**. Uso que se pretende dar al indicador. Ejemplo: Se pretende conocer las personas con menos visitas para poder darles una mayor atención personal.

11. **Medidas**. Operaciones a realizar con el hecho. Ejemplo: sumatorio de las visitas. Otros posibles valores podrían ser: Media, mínimo, máximo y moda.

12. **Granularidad Temporal**. Unidad mínima de detalle de la cual se puede obtener información. Normalmente se utilizara una granularidad diaria, aunque se pueden usar otros valores, como por ejemplo: semanal, mensual, trimestral y anual. Para nuestro Ejemplo utilizamos la diaria, lo que significa que obtendremos las visitas agrupadas por día.

13. **KPI's**. Campos a resaltar según su valor y la agrupación de estos. Ejemplo: Mostraremos en rojo aquellos campos que muestren menos de 4 visitas por mes y por residente.

14. Dimensión. Campos por los cuales queremos categorizar el hecho. Ejemplo: residentes, tipología, carácter plaza y tiempo. (No importa el orden de las dimensiones. Agregue cuantas desee)

1ª DIMENSIÓN

Descripción: Residente

Se mostrarán los datos del residente visitado.

Atributos:

Nombre y apellidos, sexo y fecha de ingreso.

2ª DIMENSIÓN

Descripción: Centro

Se mostrará el centro al que pertenece cada residente.

Atributos:

Nombre del centro, ciudad, provincia, comunidad autónoma y país.

3ª DIMENSIÓN

Descripción: Tipología

Se mostrará la tipología de los residentes.

Atributos:

Tipologia

4ª DIMENSIÓN

Descripción: Carácter Plaza

Se mostrará el carácter plaza de los residentes.

Atributos:

Carácter Plaza

5ª DIMENSIÓN

Descripción: Tiempo

Se mostrará toda la información relativa a la fecha de la visita.

Atributos:

Fecha, día semana, semana, mes, trimestre, año y nombre completo de la fecha.

Anexo B: Documento de Ayuda del Indicador

Indicador de Visitas Familiares

1. Descripción

Este indicador muestra las visitas recibidas por residente y las agrupa por:

Residente, **tipología**, **carácter plaza, centro** y **tiempo**.

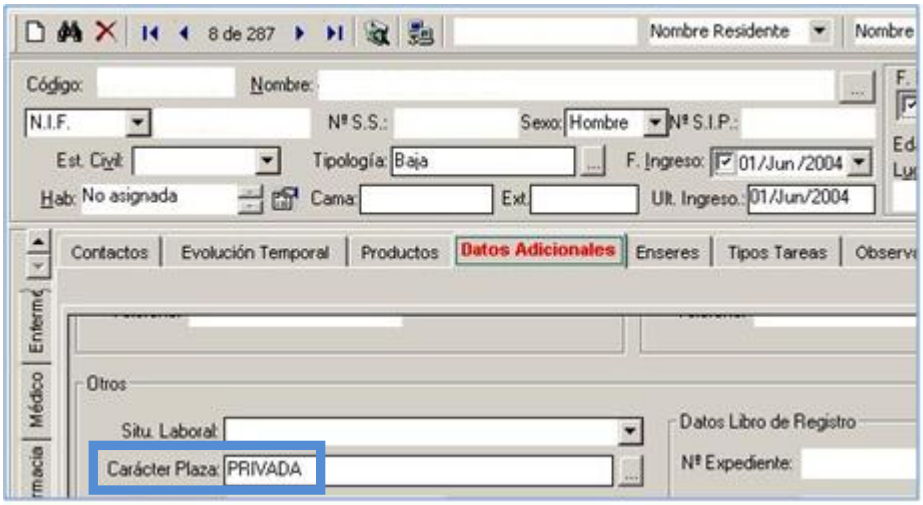
La **Granularidad** de este indicador es diaria. Esto significa que se sumara el número de visitas que ha recibido un residente por día, lo cual conlleva que no se puede obtener información sobre el número de visitas por una unidad de tiempo menor al día.

2. Dimensión

Para el correcto funcionamiento de este indicador debemos cumplimentar correctamente los siguientes campos:

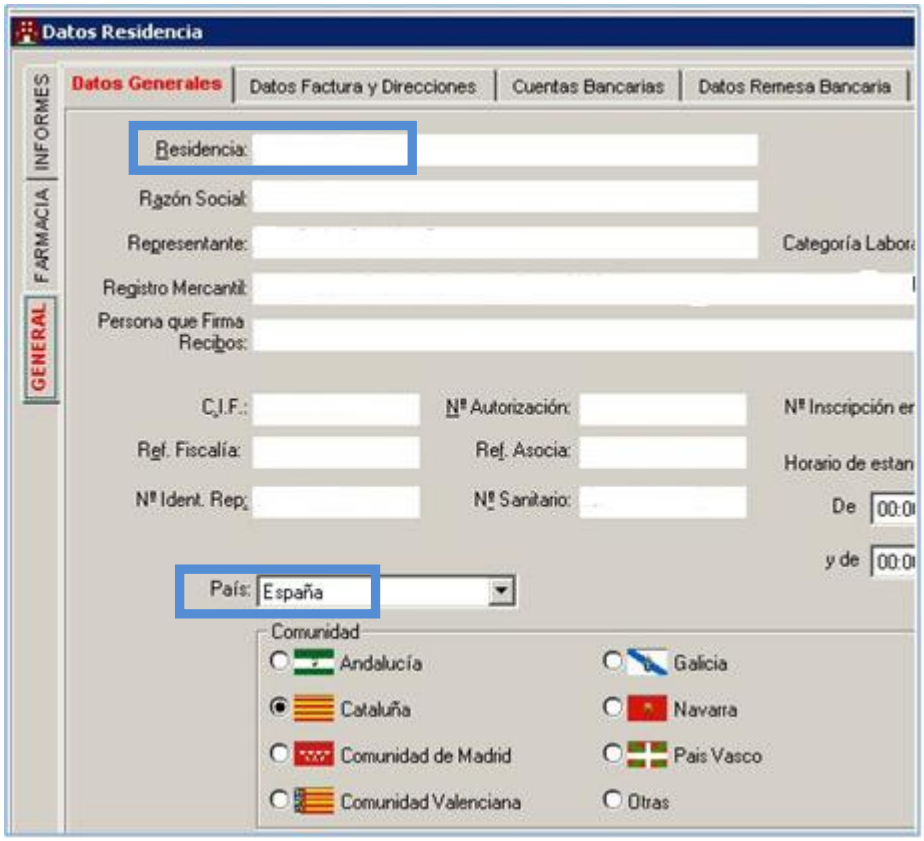
Tipología

Carácter Plaza

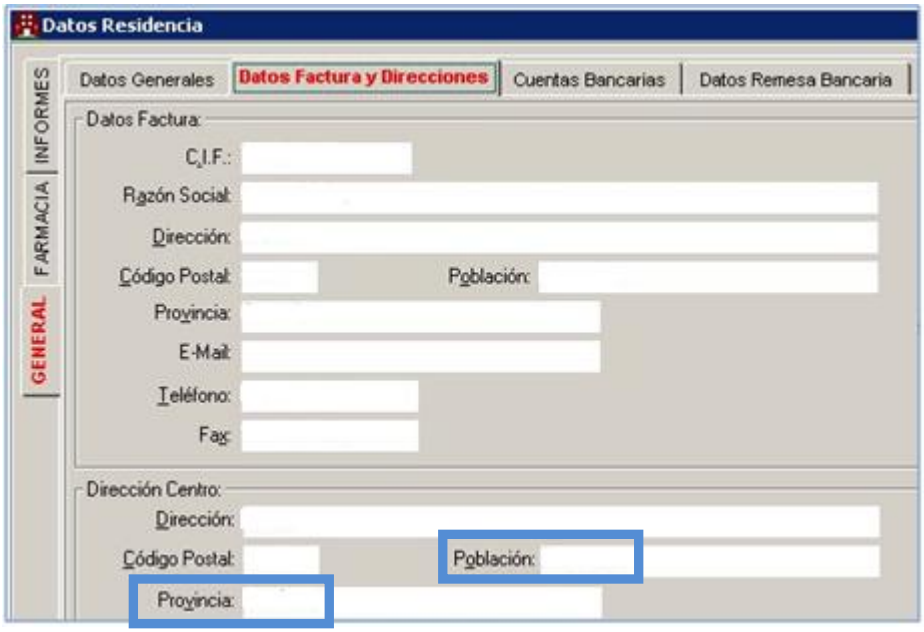


Los siguientes campos a rellenar se encuentran en la pestaña de Datos Residencia.

Residencia y País

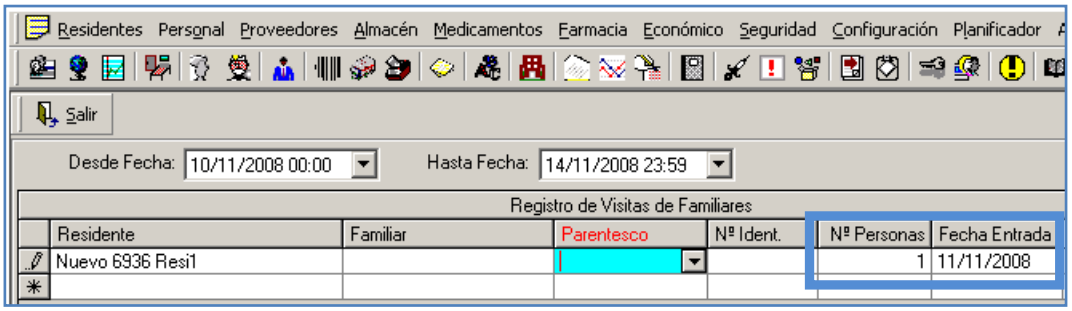


Población y provincia.



3. Hecho

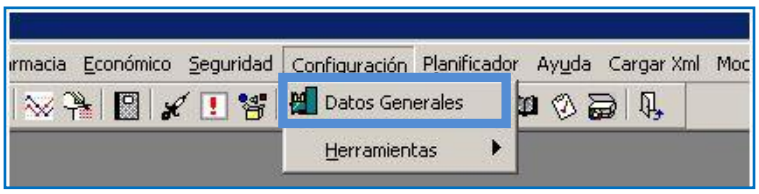
Tenemos que introducir toda la información relativa a cada una de las visitas: residente visitado, familiar que hace la visita, el parentesco de dicho familiar, número de personas que realizan la visita y fecha de entrada.



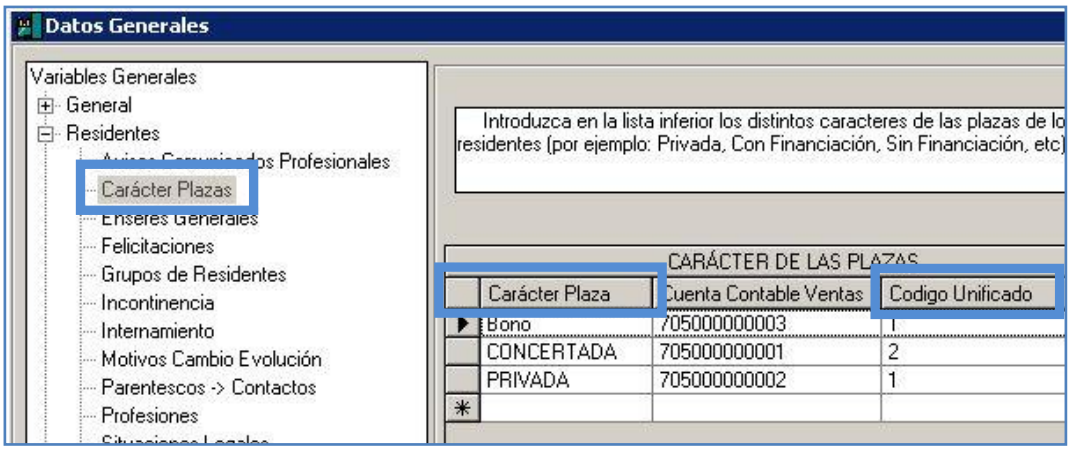
4. Código unificado

Además de los campos que muestran las capturas, debemos de rellenar el código unificado para cada una de las tablas maestras de ResiPlus y que este código coincida con el código unificado de las tablas maestras del Almacén de datos.

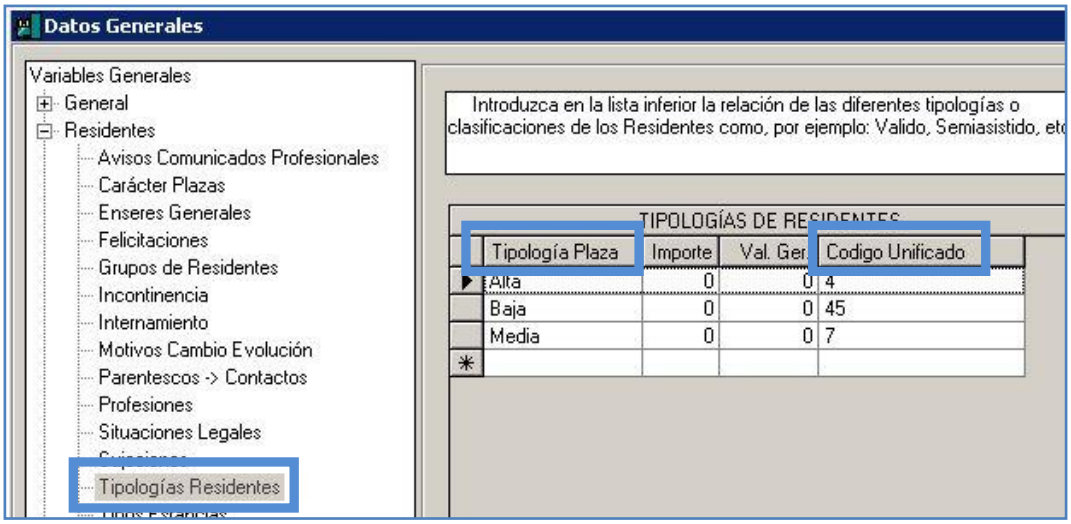
Para acceder a estas tablas maestras deberemos ir a Configuración→Datos Generales.



Una vez abierta la ventana Datos Generales, para cumplimentar el código unificado de Carácter Plaza, deberemos ir a Residentes/Carácter Plazas.



Lo mismo debemos hacer para la tipología, pero en este caso seleccionando Residentes/Tipologías Residentes.



5. Mantenimiento tablas maestras

Para que esos valores unificados puedan aparecer en el indicador tenemos que completar las tablas maestras del almacén de datos, tanto de tipologías como de carácter plaza. Esto se realiza desde el ResiPlusBI, como detallamos a continuación:

Accedemos al menú principal del BI y seleccionamos mantenimiento tablas maestras.

A continuación seleccionamos la tabla maestra a modificar. Debemos hacer que los códigos unificados de la base de datos de ResiPlus coincidan con el código unificado de nuestro almacén de datos. Los datos donde el código unificado no coincida aparecerán en el indicador como no definidos, mientras que aquellos en los que coincida aparecerán con la leyenda que indiquemos en la maestra de nuestro almacén.

Maestra de Carácter Plaza

Maestra de Tipologías

Anexo C: Documento de Análisis

	Departamento: Desarrollo	**Tipo:** Documento Análisis	ID5263
Programa/Proyecto: Indicadores PFC		**Analista:** Marta	**Fecha creación:** 01/05/2010

1. DESCRIPCIÓN

Creación del Indicador de Visitas Familiares para el proyecto de Indicadores PFC.

2. MOTIVACIÓN

El cliente pretende conocer qué tipo de residentes recibe menos visitas para darle más atención personal.

3. DEFINICIÓN

3.1. Hechos

Visitas Familiares

Descripción: Sumatorio de las visitas recibidas por los residentes agrupadas por todas las dimensiones que se indican en la sección 3.3.

Granularidad Temporal: Diaria.

Frecuencia de Recolección de datos: 1 día

Frecuencia de refresco del cubo: 1 día

3.2. KPI

Descripción: Nos indicará gráficamente si el número de visitas familiares es mayor que un valor dado.

Valor por defecto: 4

3.3. Medidas

*/*En este apartado es donde indicamos el resto de medidas que aparecen en la tabla de hechos sin contar lo que medimos en el hecho propiamente (Nº visitas que reciben los residentes cada día)*/*

3.4. Dimensiones

Dimensiones	Histórico
Centro	No (*)
Tiempo	No(*)
Residente	No(*)
Tipología	Sí
Carácter Plaza	Sí

(*) Siempre se mostrarán los valores actuales, es decir, los valores que posean esas dimensiones en la base de datos de ResiPlus.

4. PRUEBAS DE ACEPTACIÓN

5263.1 Comprobar que si un residente recibe más de una visita en un día, todas las visitas aparecen contabilizadas en un único registro.

5263.2 Comprobar que si hay alguna dimensión sin valor, siempre que no sea centro, residente ni tiempo, no se produce ningún error.

5263.3 Comprobar que si alguna dimensión no posee código unificado o no aparece en la tabla maestra del almacén, aparecerá como no definida.

Anexo D: Documento de Diseño

	Departamento: Desarrollo	**Tipo:** Documento Diseño	ID5263
Programa/Proyecto: Indicadores PFC		**Analista:** Marta	**Fecha creación:** 01/05/2010

1. DESCRIPCIÓN

Creación del Indicador de Visitas Familiares para el proyecto de Indicadores PFC.

2. DISEÑO

Este indicador se carga por medio del ETL.

Dimensiones del indicador:

- **Tiempo** (Utilizaremos el campo FechaHoraEntrada de la tabla VisitasFamiliares)
- **Centro** (Utilizaremos la tabla DatosResidencia)
- **Residente** (Utilizaremos el campo IDResidente de la tabla VisitasFamiliares para luego obtener el id del almacén correspondiente para ese residente)
- **Carácter Plaza** (Utilizaremos el campo IDResidente de la tabla VisitasFamiliares para posteriormente atacar a la tabla ResiHisCaracterPlaza con ese id y la fecha que indica el campo FechaHoraEntrada y obtener el carácter plaza del residente en esa fecha)
- **Tipología** (Utilizaremos el campo IDResidente de la tabla VisitasFamiliares para posteriormente atacar a la tabla ResiHisTipologiasPlaza con ese id y la fecha que indica el campo FechaHoraEntrada y obtener la tipología del residente en esa fecha)

Hechos del indicador:

- VisitasFamiliares: Sumatorio de las visitas familiares agrupado por residente y día. Se añadirá el campo NumVisitas a la tabla de hechos. Dicho campo se cumplimentará con 1 para cada registro de la tabla residentes, siempre que no se repita el IDResidente ni la fecha de la visita. En caso de que estos campos aparezcan repetidos, se incrementará el campo NumVisitas en 1 por cada registro repetido. Este dato se obtendrá en la tabla visitas familiares.

La **granularidad** será diaria, por lo que si un residente tiene más de una visita el mismo día, mostraremos todas en un mismo registro.

Miembros Calculados y KPI: Este indicador posee un KPI que se muestra en rojo para todos aquellos registros cuyo número de visitas por mes y residente es inferior a 4.

CUBO

Crearemos un cubo en modo Molap que se procesará cada 24 horas, inmediatamente después de la carga de los datos en nuestro almacén de datos. Poseerá las dimensiones enunciadas en el apartado anterior.

3. NOTAS DE DISEÑO

Anexo E: Desarrollo de paquetes ETL en Microsoft Integration Services

SQL Server Integration Services (SSIS) se ha diseñado para proporcionar los componentes de infraestructura que permiten el traslado de datos de un sistema a otro, al tiempo que se manipulan los datos para que adopten el formato deseado a medida que se produce dicho traslado.

En este tutorial, conoceremos cuáles son los componentes disponibles dentro de SSIS, así como a construir paquetes SSIS flexibles para su entorno. Dado que los documentos de ejemplo mostrados anteriormente hacían referencia al indicador de visitas familiares, ese es el indicador que vamos a utilizar como ejemplo de este tutorial.

A continuación se indican los pasos necesarios para poder desarrollar el paquete ETL del indicador en cuestión, suponiendo que el almacén de datos ha sido creado y cargado.

Paso 1: Crear un proyecto de Integration Services

En primer lugar, crearemos un nuevo proyecto de SSIS y nos familiarizaremos con los componentes básicos del entorno.

Para crear un proyecto de SSIS, seguimos estos pasos:

1. Hacemos clic en **Inicio**, seleccionamos **Todos los programas**, seleccionamos **Microsoft SQL Server** y, después, hacemos clic en **SQL Server Business Intelligence Development Studio**. De esta forma se abrirá el entorno de desarrollo de Microsoft Visual Studio.
2. Cerramos la ficha **Página de Inicio**. En el menú **Archivo** de Visual Studio, seleccionamos **Nuevo** y hacemos clic en **Proyecto**.
3. En el cuadro de diálogo **Nuevo proyecto**, seleccionamos **Proyectos de Business Intelligence** en el panel **Tipos de proyecto**, y seleccionamos **Proyecto de Integration Services** en el panel **Plantillas**.
4. Cambiamos el nombre del proyecto, el cúal cambiará también el nombre de la solución, y hacemos clic en **Aceptar**.

La Figura E.1. muestra la pantalla resultante tras la creación del proyecto.

En la parte inferior de la pantalla, verá tres pestañas: **Lista de errores**, **Lista de tareas** y **Resultados**. La Lista de errores muestra todos los errores y advertencias relativas a su paquete durante el diseño o depuración. La Lista de tareas es una zona donde podrá hacer un seguimiento de una lista de tareas que necesita lanzar mientras trabaja dentro de la solución. Las ventanas de Resultados mostrarán la información del estado y progreso junto con cualquier mensaje que se produzca durante la ejecución de un paquete.

En la parte derecha de la pantalla verá las ventanas **Explorador de soluciones** y **Propiedades**. El Explorador de soluciones se utiliza para agrupar todos los objetos en un proyecto y le permite agregar **Fuentes de datos**, **Vistas de fuentes de datos** y **Paquetes** al proyecto. En la ventana Propiedades puede ver y editar las propiedades del objeto seleccionado.

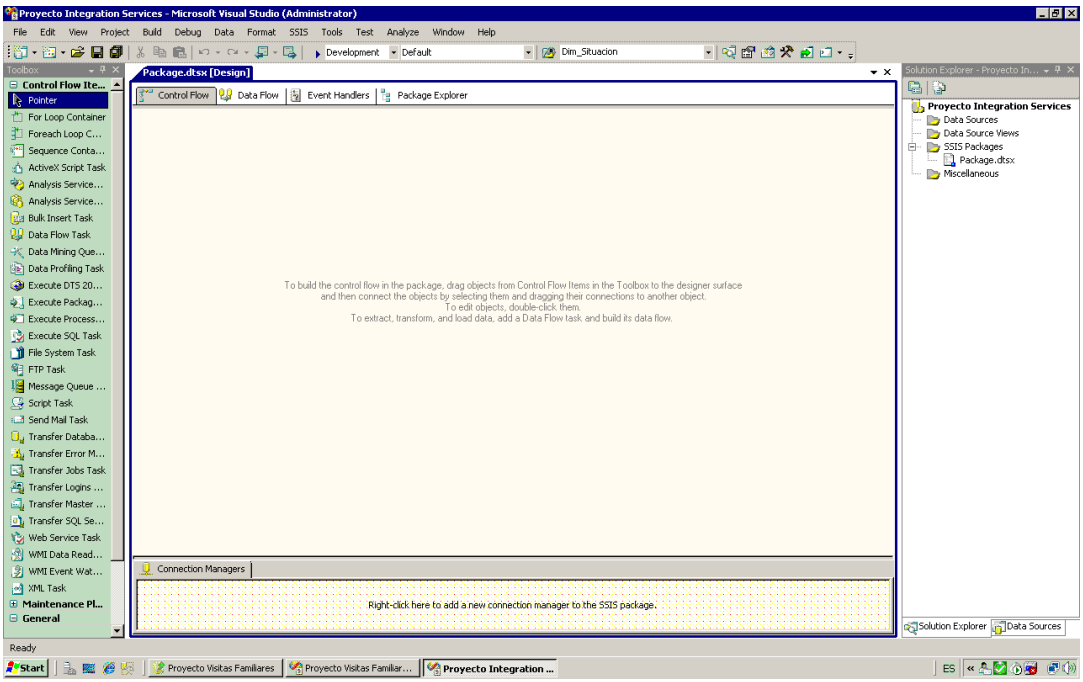


Figura E.1. Pantalla resultante tras la creación de un proyecto

Paso 2: Renombrar el paquete SSIS

Para cambiar el nombre que tiene por defecto el paquete cuando creamos un nuevo proyecto debemos:

1. Seleccionar el paquete recién creado en el **Explorador de soluciones**.
2. Cambiar el nombre (Campo Nombre) en la ventana de **Propiedades** (Figura E.2.) y hacer clic en Sí cuando aparezca un cuadro de diálogo preguntando si desea renombrar el objeto del paquete.

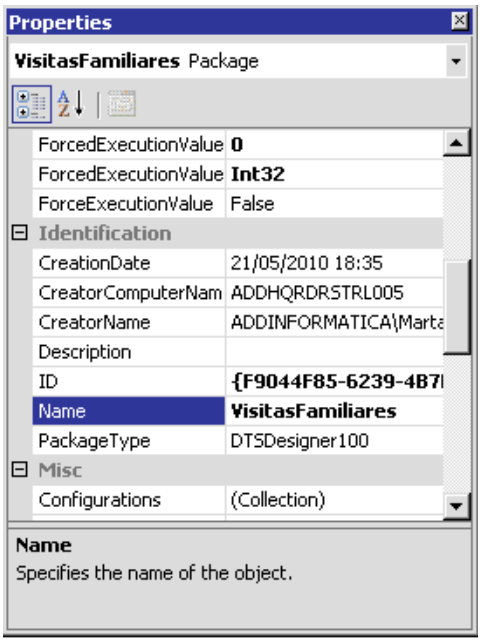


Figura E.2. Ventana Propiedades del paquete SSIS

Las dos zonas donde pasaremos la mayor parte de nuestro tiempo son el **Cuadro de herramientas** a la izquierda y la superficie de diseño en el panel central. El Cuadro de

herramientas contiene todas las tareas y transformaciones que pueden utilizarse a la hora de construir un paquete SSIS. La superficie de diseño se utiliza para construir el flujo y movimiento de datos del paquete.

La superficie de diseño se divide en cuatro pestañas: **Flujo de control**, **Flujo de datos**, **Controladores de eventos** y **Exploradores de paquetes**. La pestaña del Explorador de paquetes permite explorar un paquete de forma jerárquica con el fin de ver todos los componentes que se han configurado. La pestaña Flujo de control es el lugar donde se diseña el proceso de negocios que ejecutará el paquete. En la pestaña Flujo de datos diseñaremos la forma en que los datos se moverán y se manipularán. La pestaña Controladores de eventos nos permite especificar acciones en respuesta a un evento de paquetes.

SSIS incluye 40 tareas diferentes que pueden agregarse a un paquete. Las tareas se dividen en dos categorías principales: tareas de Flujo de control y tareas del Plan de mantenimiento.

En nuestro caso, solamente utilizamos tareas de Flujo de control, siendo las más utilizadas las siguientes:

- Contenedor de secuencias: Se utiliza para agrupar tareas en una unidad lógica de trabajo.
- Tarea Flujo de datos: Es una tarea esencial en casi todos los paquetes SSIS que lleva asociada un panel de Flujo de datos donde se podrá configurar de qué manera se mueven los datos y cualquier manipulación exigida a medida que se van procesando los datos.
- Tarea Ejecutar SQL: Permite ejecutar scripts de T-SQL.

En cuanto a las transformaciones más utilizadas dentro de las tareas de flujos de datos, tenemos las siguientes:

- Agregado: Permite realizar cómputos de agregados mientras los datos se transmiten a través del paquete.
- Ordenar: Ordena el conjunto de datos.
- División condicional: Se utiliza para dividir un origen de datos en diversas salidas basándose en una condición.
- Conversión de datos: Convierte tipos de datos.
- Columna derivada: Se utiliza siempre que necesitamos manipular datos, tales como la sustitución de cadenas vacías con un valor NULL, la eliminación de caracteres inválidos o el cálculo de nuevos valores.
- Mezclar: Combina dos corrientes de datos ordenados en una salida única.
- Unión de todo: Igual que la anterior pero puede combinar más de dos entradas de datos desordenadas.
- Combinación de mezcla: Combina conjuntos de datos ordenados con una funcionalidad similar a la ejecución de una declaración SELECT con INNER y OUTER JOIN.
- Multidifusión: Permite distribuir una entrada en múltiples salidas.

- Componente de script: Permite incrustar script Visual Basic .NET o C# en un flujo de datos.

Paso 3: Creación y configuración de las conexiones

Una vez comentadas las tareas y transformaciones más utilizadas, el siguiente paso en la creación del paquete ETL es la creación y configuración de las conexiones. Primeramente añadimos los orígenes de datos y luego agregamos las conexiones al panel de **Administradores de conexión** utilizando los orígenes de datos.

Para llevar a cabo esta tarea tenemos que:

1. Seleccionar la carpeta **Orígenes de datos** en el **Explorador de soluciones**, hacer clic en el botón derecho y seleccionar **Nuevo origen de datos**. Se abrirá el **Asistente para orígenes de datos** y hacemos clic en **Siguiente**.
2. En la página **Seleccione cómo definir la conexión**, hacemos clic en **Nueva**. Aparecerá el cuadro de diálogo **Administrador de conexión** (Figura E.3.). En este cuadro de diálogo se definen las propiedades de conexión para el origen de datos. Un administrador de conexión es una representación lógica de la conexión que se utilizará en tiempo de ejecución.
3. En la lista **Proveedor**, comprobamos que la opción **OLE DB nativo\SQL Native Client** está seleccionada.
4. En el cuadro de texto **Nombre de servidor**, escribimos el nombre del servidor donde se aloja la base de datos con la que queremos conectar. En este caso la base de datos se encuentra en el equipo local; no obstante, las bases de datos de origen generalmente se encuentran alojadas en uno o más equipos remotos.
5. Comprobamos que la opción **Utilizar autenticación de Windows** está seleccionada. En la lista **Seleccione o escriba un nombre de base de datos**, seleccionamos el nombre de la base de datos.

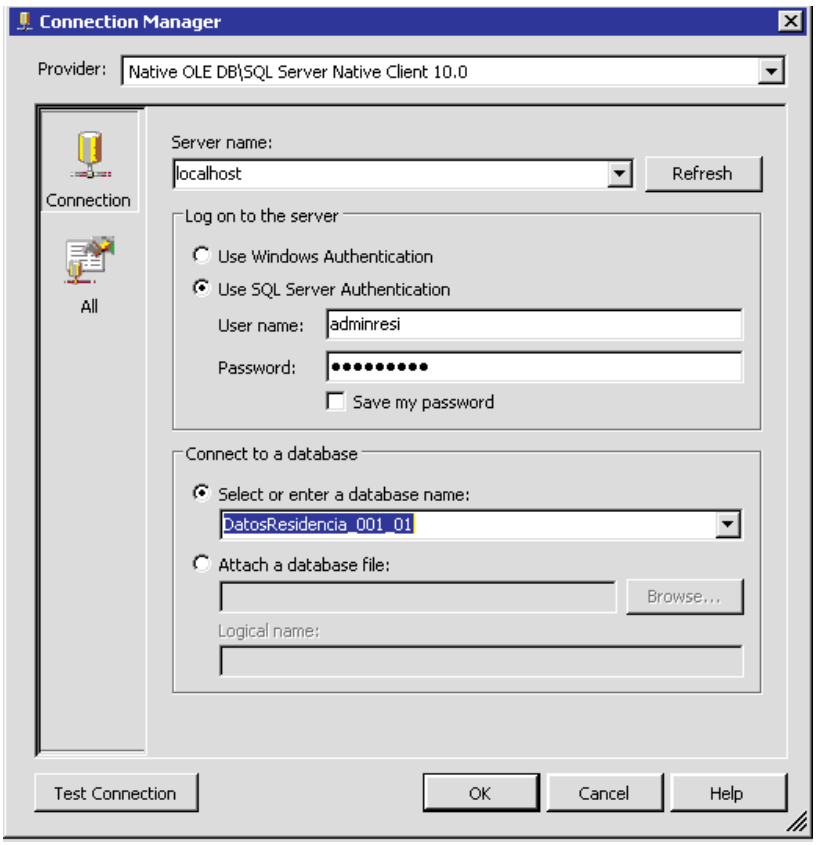


Figura E.3. Cuadro de diálogo Administrador de conexión

6. Hacemos clic en **Aceptar** y, a continuación en **Siguiente**.

Hasta el momento hemos creado un orígen de datos que se corresponde con la base de datos operacional de la que se extraen los datos para el indicador. Dado que nuestro indicador requiere una conexión a la base de datos operacional y otra conexión al almacén de datos, ya que es ahí donde se guarda la información a explotar por los indicadores, debemos repetir los pasos 1-6 para crear un orígen de datos asociado al almacén de datos.

7. Hacemos clic en el panel **Administradores de conexión**, seleccionamos **Nueva conexión desde origen de datos** y seleccionamos los orígenes de datos creados anteriormente.

A continuación, hay que especificar el proceso de negocio que queremos que ejecute el paquete por medio de tareas de flujo de datos agregadas al flujo de control. Sin embargo, en muchas ocasiones resulta necesario definir variables para poder utilizarlas en diferentes tareas de flujo de datos y por eso el siguiente paso es la definición de variables.

Paso 4: Definición de variables

Para agregar una variable a un paquete de SSIS realizaremos las siguientes acciones:

1. Ir al menú **Vista**, seleccionar **Otras Ventanas** y hacer clic en **Variables**.
2. En la ventana **Variables** (Figura E.4.), haga clic en el icono **Agregar variable** y se agregará la nueva variable a la lista. Dependiendo de qué parte del paquete tengamos seleccionada cuando agreguemos la variable, el ámbito de la misma será

diferente, es decir, si tenemos seleccionada una tarea de flujo de datos, el ámbito de la variable será esa tarea mientras que si no tenemos seleccionada ninguna tarea, el ámbito será todo el paquete.

3. Ponemos nombre a la variable e indicamos el tipo de datos y el valor por defecto de la misma.

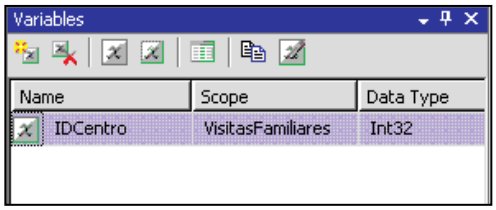


Figura E.4. Ventana Variables

En caso de que la información introducida con respecto a la variable no sea correcta, podemos eliminar la variable haciendo clic en el icono **Eliminar variable**.

Paso 5: Construcción del flujo de control

El flujo de control puede estar compuesto de varias tareas de flujo de control o de mantenimiento. Sin embargo, para nuestro indicador, sólo vamos a necesitar dos tareas de flujo de control (Figura E.5.): una que se encargue de rellenar la variable definida en el paso anterior y otra que extraiga los datos de la base de datos operacional y los almacene en la tabla de hechos del almacén de datos.

Para construir el flujo de control, tenemos que:

1. Arrastrar dos **Tareas de flujo de datos** desde el **Cuadro de herramientas** a la superficie de diseño de flujo de control.
2. Seleccionar la primera **Tarea de flujo de datos** y arrastrar la flecha verde hasta la segunda tarea para crear una restricción de precedencia.
3. Cambiar el nombre de las tareas para ponerle nombres más significativos. Para ello, haga clic con el botón derecho en la tarea correspondiente y seleccione **Cambiar Nombre** o bien seleccione la tarea y modifique el nombre en el panel de **Propiedades**.
4. Dado que aún no estamos listos para la ejecución de las tareas, hacemos clic con el botón derecho en las tareas y seleccionamos la opción **Deshabilitar**.

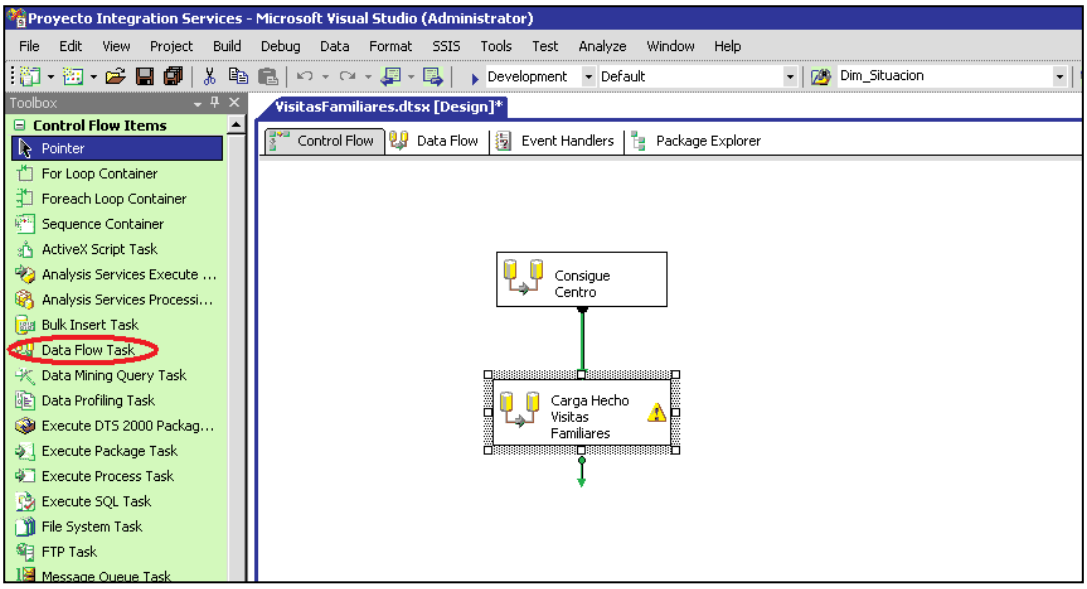


Figura E.5. Flujo de control del indicador Visitas Familiares

Dado que ya tenemos el flujo de control construido, ahora hay que configurar el flujo de datos para cada una de las tareas agregadas al flujo de control.

Paso 6: Construcción del flujo de datos

En este paso se va a proceder a la construcción del flujo de datos. Como hay que construir el flujo de datos de dos tareas, primero se construirá el flujo de datos correspondiente a la primera tarea y luego el de la segunda tarea.

El objetivo de la primera tarea es dotar de un valor a la variable *IDCentro* y para ello necesita obtener el valor almacenado en el almacén para el centro del que estamos obteniendo los datos. Concretamente, los pasos a seguir para construir este flujo de datos son:

1. Seleccionar la pestaña de **Flujo de datos** y arrastrar dos **Orígenes OLE DB,** dos **Conversiones de datos,** dos **Ordenar,** una **Combinación de mezcla** y un **Componente de script** a la superficie de diseño de flujo de datos.
2. Hacer doble clic en uno de los orígenes y elegir la conexión Residencia como administrador de conexiones. Repetir la acción en el otro orígen y elegir la conexión Almacén.
3. Seleccionar en el desplegable de **Modo de acceso a datos**, la opción **Comando SQL**. Otra posibilidad sería pinchar la opción **Tabla o vista** y seleccionar las columnas pero se recomienda la primera alternativa.
4. Introducir la consulta SQL o bien pinchar el botón de **Generar consulta**. Al pulsar este botón, hacemos clic en el botón derecho, seleccionamos **Añadir tabla** y marcamos las columnas que queremos que salgan en la consulta. Repetimos el proceso hasta que tengamos todas las tablas y columnas añadidas necesarias para la consulta y pulsamos **Aceptar**.
5. Haga clic en el botón **Vista previa** para asegurarse de que ha elegido el administrador de conexiones correcto y haga clic en **Cerrar**.
6. Arrastre la flecha verde de **Orígen OLE DB** hasta la transformación de **Conversión de datos** en ambos casos.

7. Haga doble clic en la transformación de **Conversión de datos** (Figura E.6.), seleccione la columna cuyo tipo desea cambiar, indique el nombre y el tipo de la nueva columna y pulse **Aceptar**.

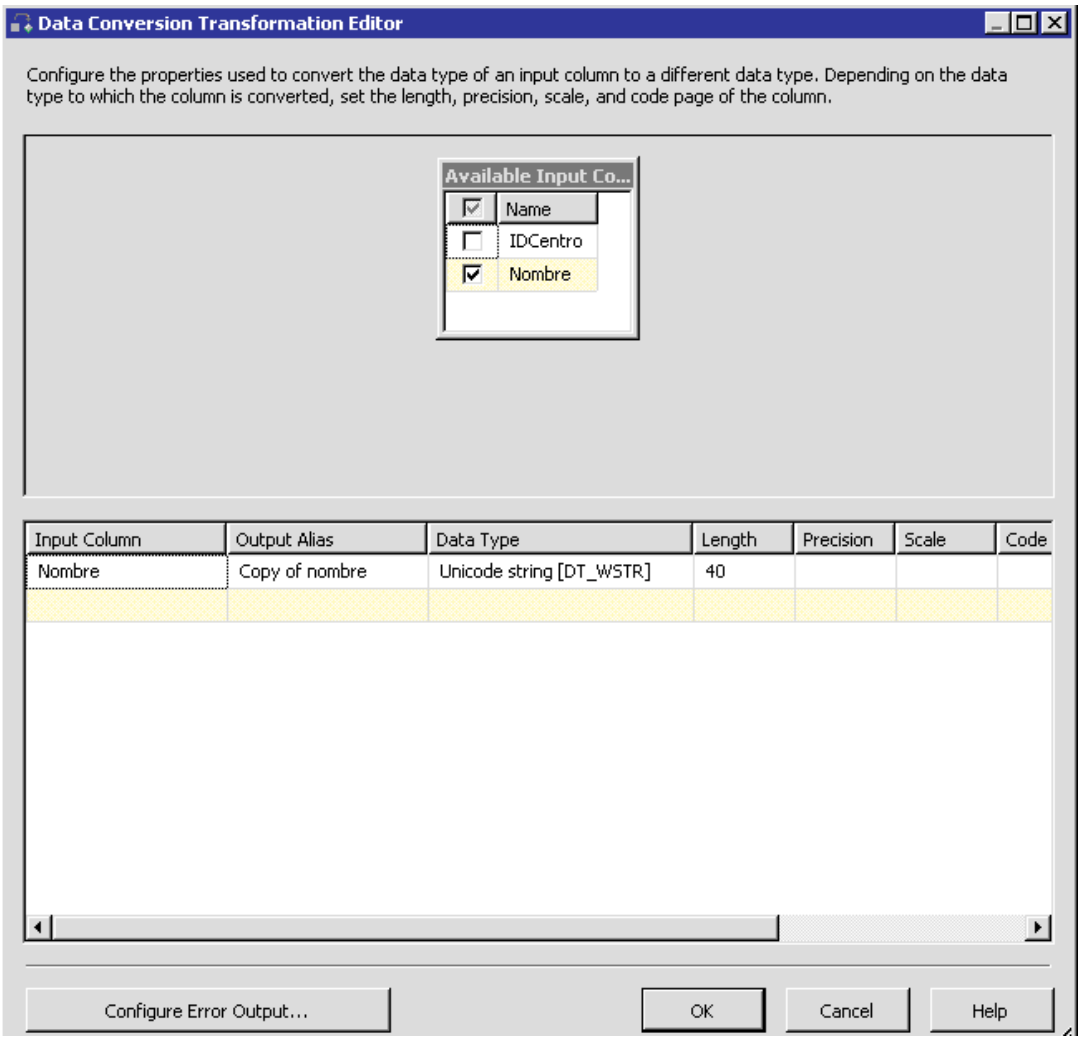


Figura E.6. Transformación Conversión de datos

8. Arrastre la flecha verde desde la transformación de **Conversión de datos** hasta la de **Ordenar**.
9. Haga doble clic en la transformación de **Ordenar**,seleccione la columna por la que desea ordenar los datos y haga clic en **Aceptar**.
10. Arrastre la flecha verde desde la transformación de **Ordenar** hasta la de **Combinación de mezcla**.
11. Haga doble clic en la transformación de **Combinación de mezcla** (Figura E.7.)**,** seleccione el tipo de unión (Inner join, Left outer join o Full outer join), seleccione las columnas que desea de cada conjunto de datos y haga clic en **Aceptar**. Se observa que ambos conjuntos de datos aparecen enlazados por la columna por la que los hemos ordenado en el paso previo.

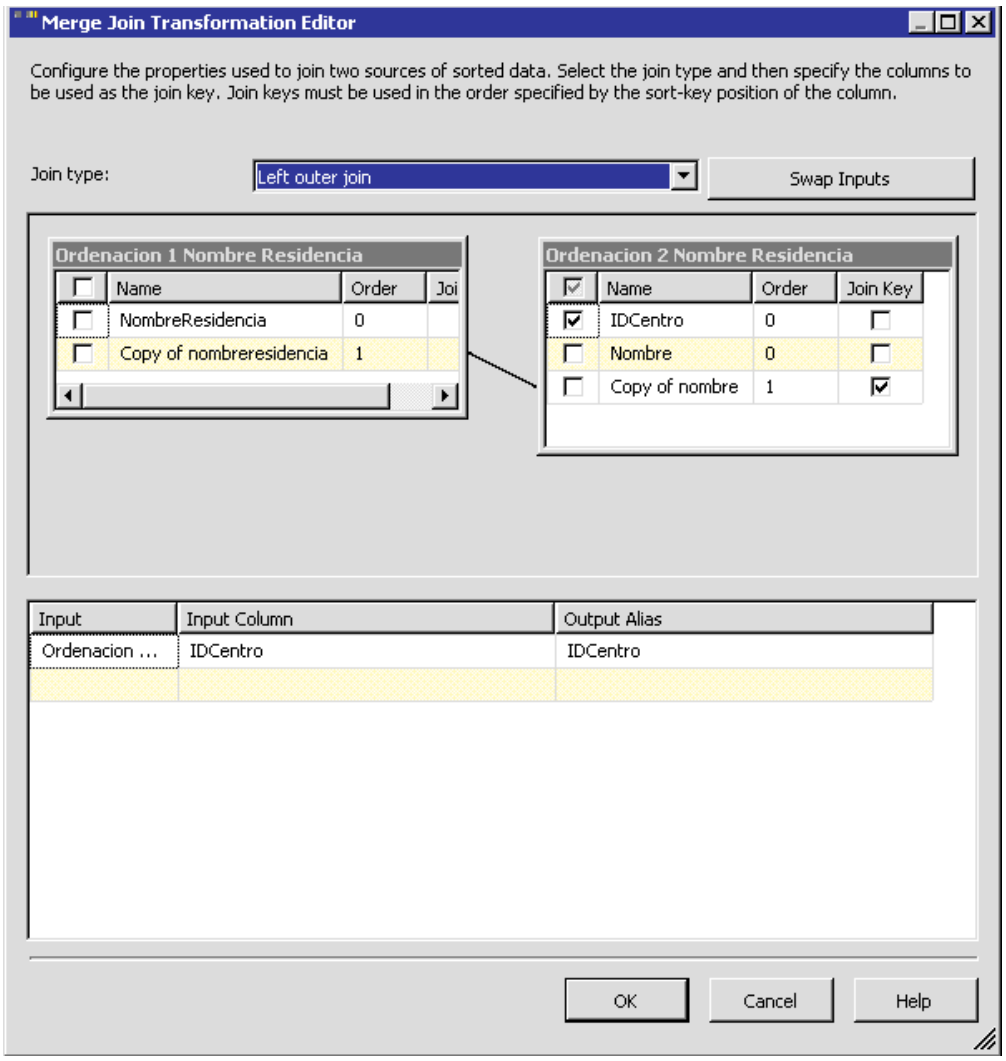


Figura E.7. Transformación Combinación de mezcla

12. Arrastre la flecha verde desde la transformación de **Combinación de mezcla** hasta la transformación de **Componente de script**.

13. Haga doble clic en la transformación de **Componente de script**. Primeramente, indique el lenguaje del script y las variables que se van a leer o escribir. Luego indique las columnas de entrada, las entradas y salidas del script utilizando los botones **Añadir Salida** y **Añadir Columna**. Es muy importante que para las salidas ponga el campo **SynchronousInputID** a ninguno (Figura E.8.). Finalmente, pulse el botón **Editar script** para poner el código del mismo, guárdelo y pulse **Aceptar**.

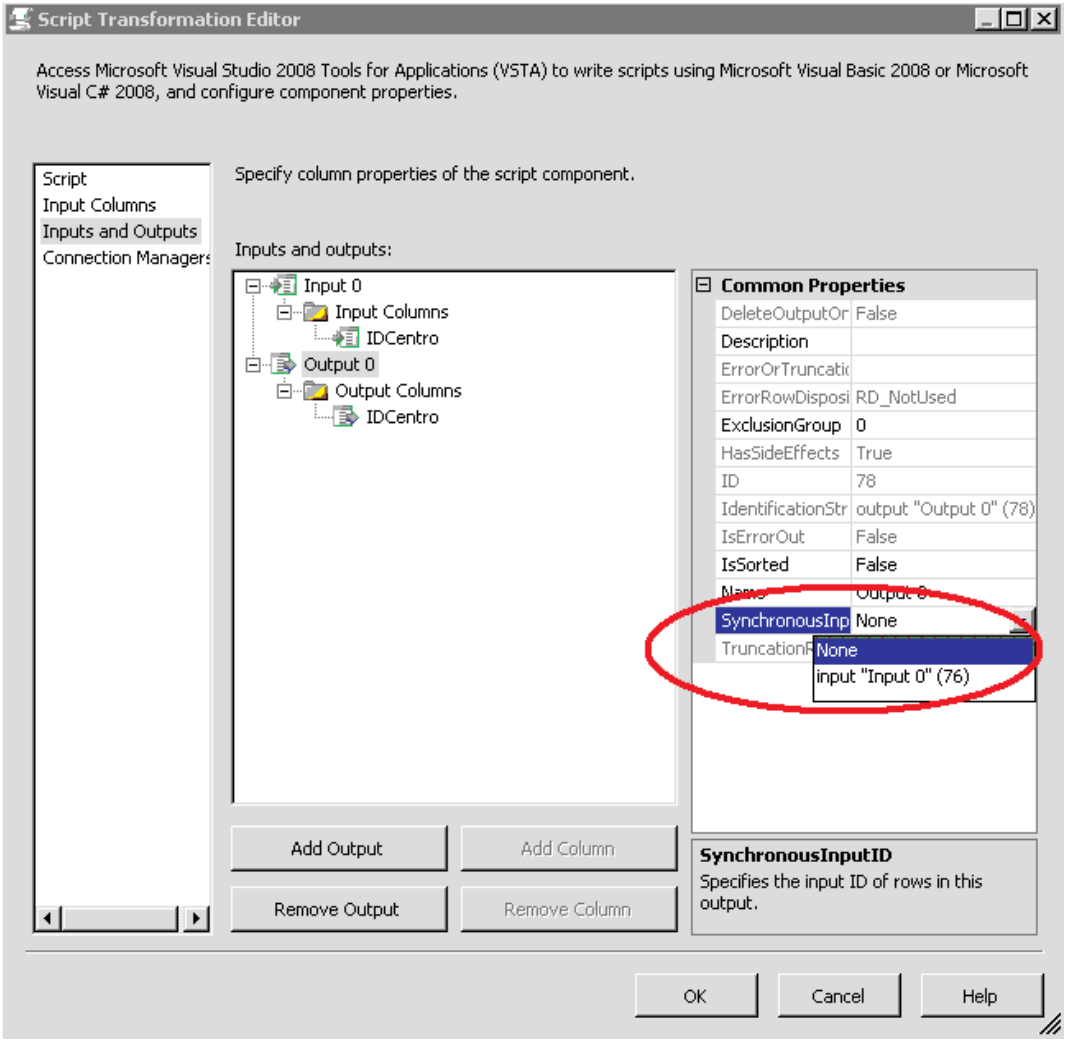


Figura E.8. Configuración de entradas y salidas del componente de script

14. Dé un nombre reconocible a los objetos del flujo de datos (Figura E.9.) tal y como se indicó anteriormente y guarde el paquete.

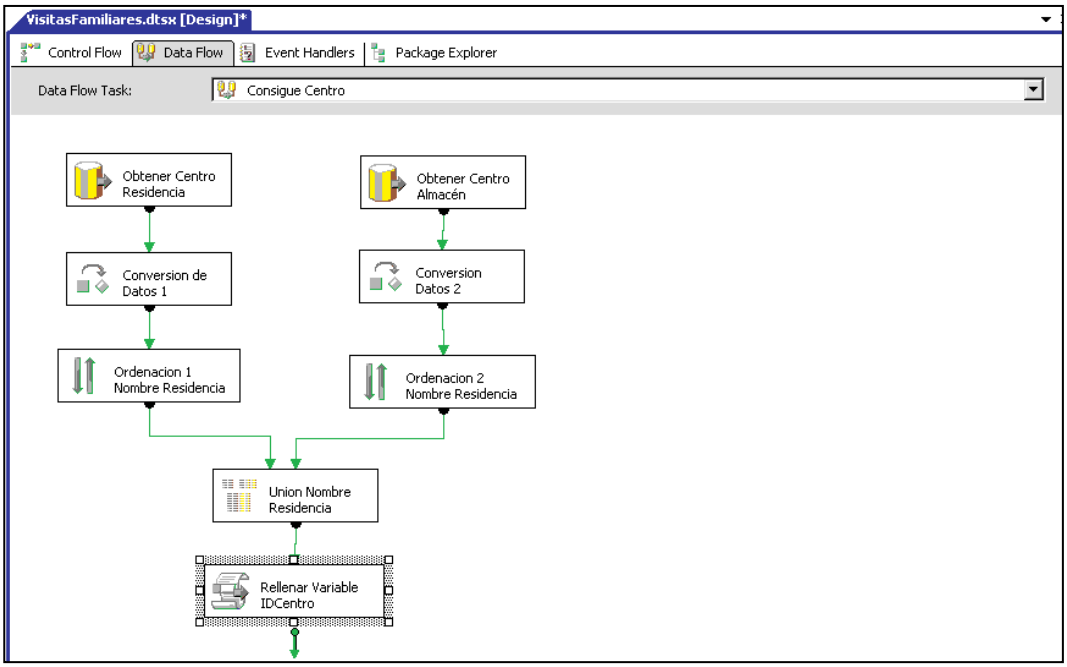


Figura E.9. Flujo de datos asociado a la tarea Consigue Centro

Finalizado el flujo de datos de la primera tarea, tenemos que construir el flujo de datos de la segunda tarea. Al igual que para la primera tarea, tenemos que añadir un **Orígen OLE DB** por cada tabla de la base de datos operacional o del almacén de datos que consultemos, transformaciones de **Ordenar**, transformaciones de **Conversión de datos**, transformaciones de **Combinación de mezcla** y arrastrar las flechas verdes de una transformación a otra para enlazarlas.

Puesto que ya hemos comentado en la tarea anterior cómo funcionan la mayoría de las transformaciones, a continuación sólo vamos a comentar de la segunda tarea aquéllas transformaciones no vistas hasta el momento.

- **Agregado**:

Hacemos doble clic en la transformación (Figura E.10.), seleccionamos las columnas sobre las que vamos a hacer alguna operación e indicamos la operación a realizar sobre cada una de ellas (Group by, Count, Count distinct, Sum, Average, Maximum, Minimum). Pulsamos **Aceptar** para guardar los cambios.

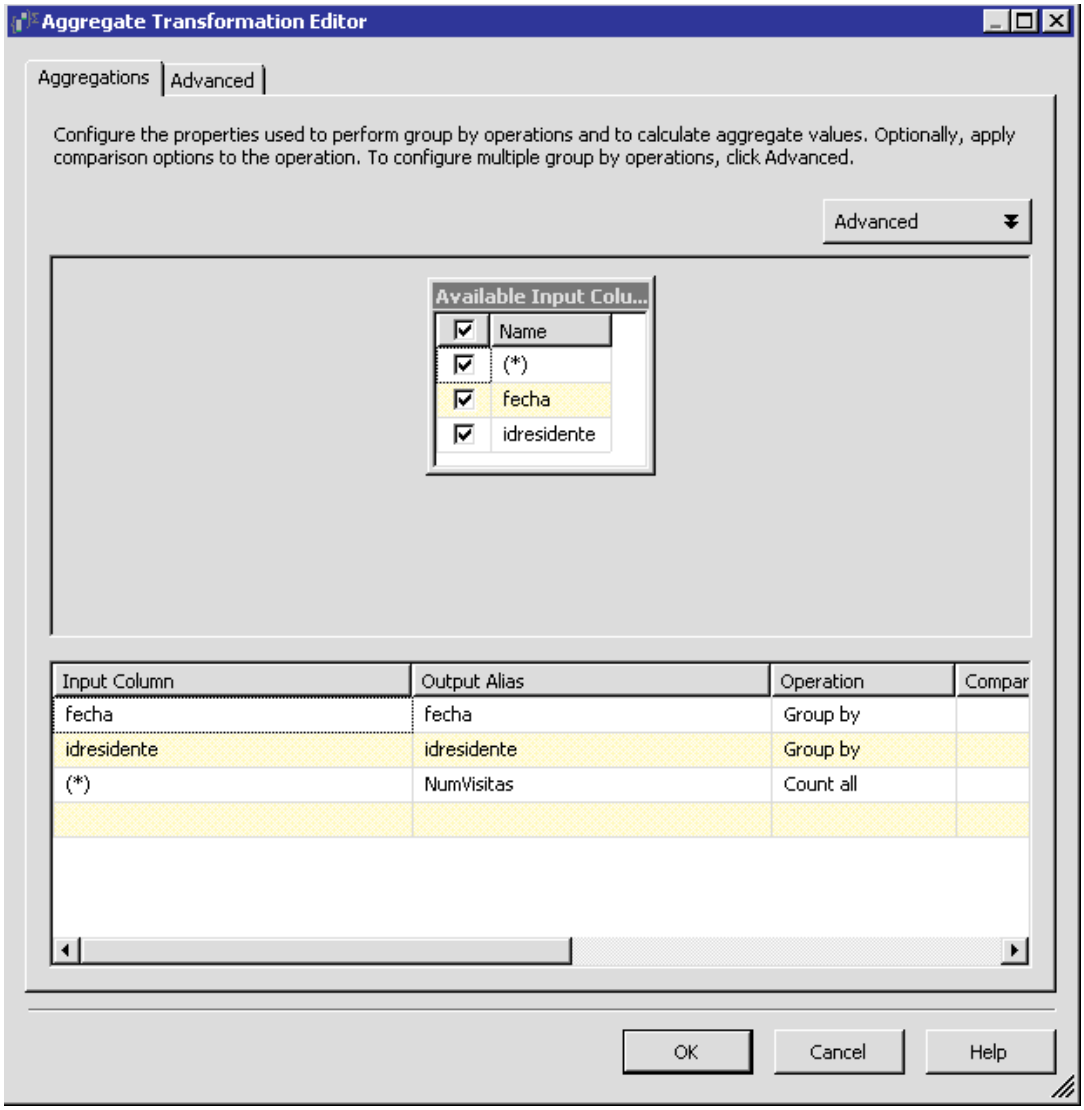


Figura E.10. Transformación Agregado

- **Columna derivada:**

Hacemos doble clic en la transformación (Figura E.11.), indicamos el nombre de la columna derivada y pulsamos **Aceptar**. En el campo **Expresión** ponemos cómo

rellenamos la columna, en nuestro caso es con la variable definida anteriormente y rellenada en el anterior flujo de datos pero puede rellenarse aplicando una función matemática u otro operador.

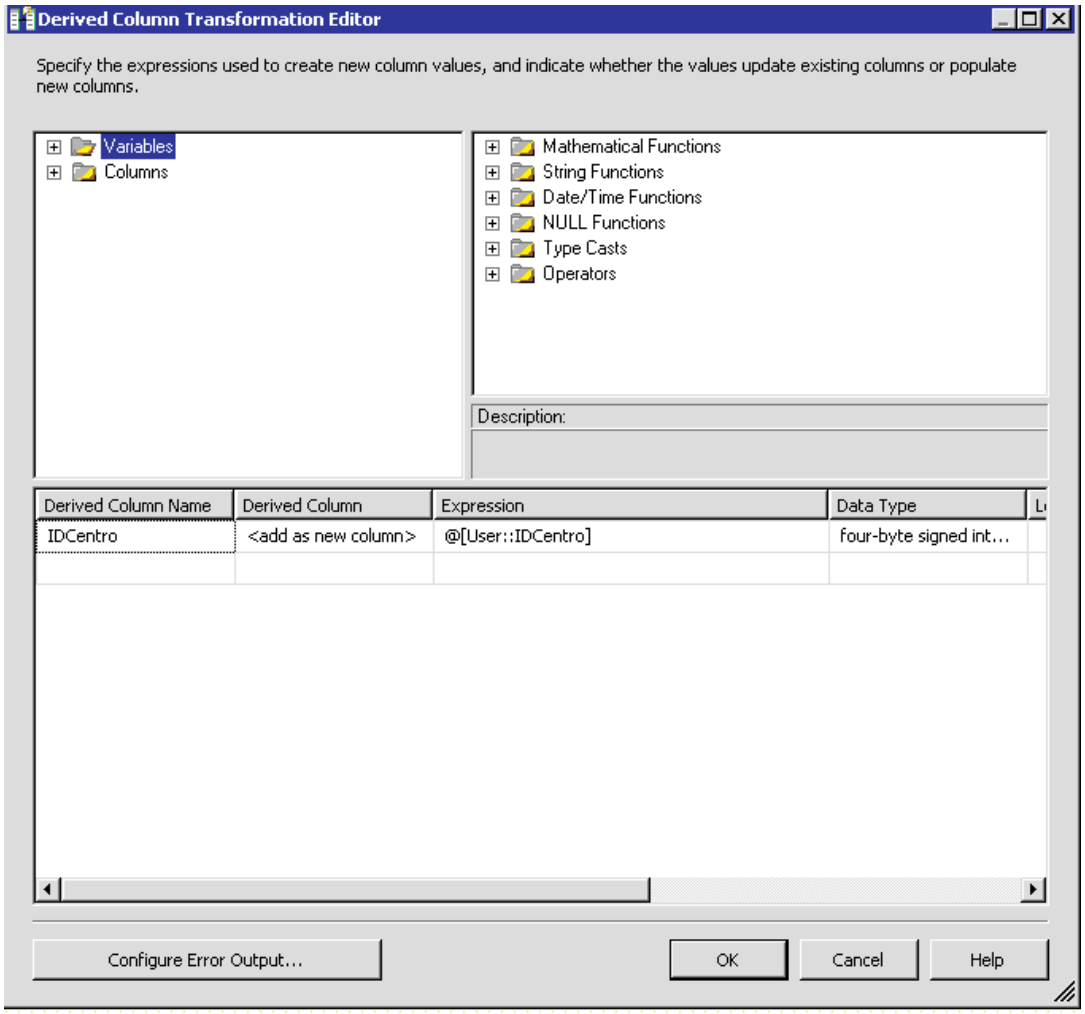


Figura E.11. Transformación Columna Derivada

- **Destino OLE DB:**

Hacemos doble clic en la transformación (Figura E.12), seleccionamos como **Modo de acceso a datos** la opción de **Carga rápida de tabla o vista** y seleccionamos la tabla que vamos a rellenar.

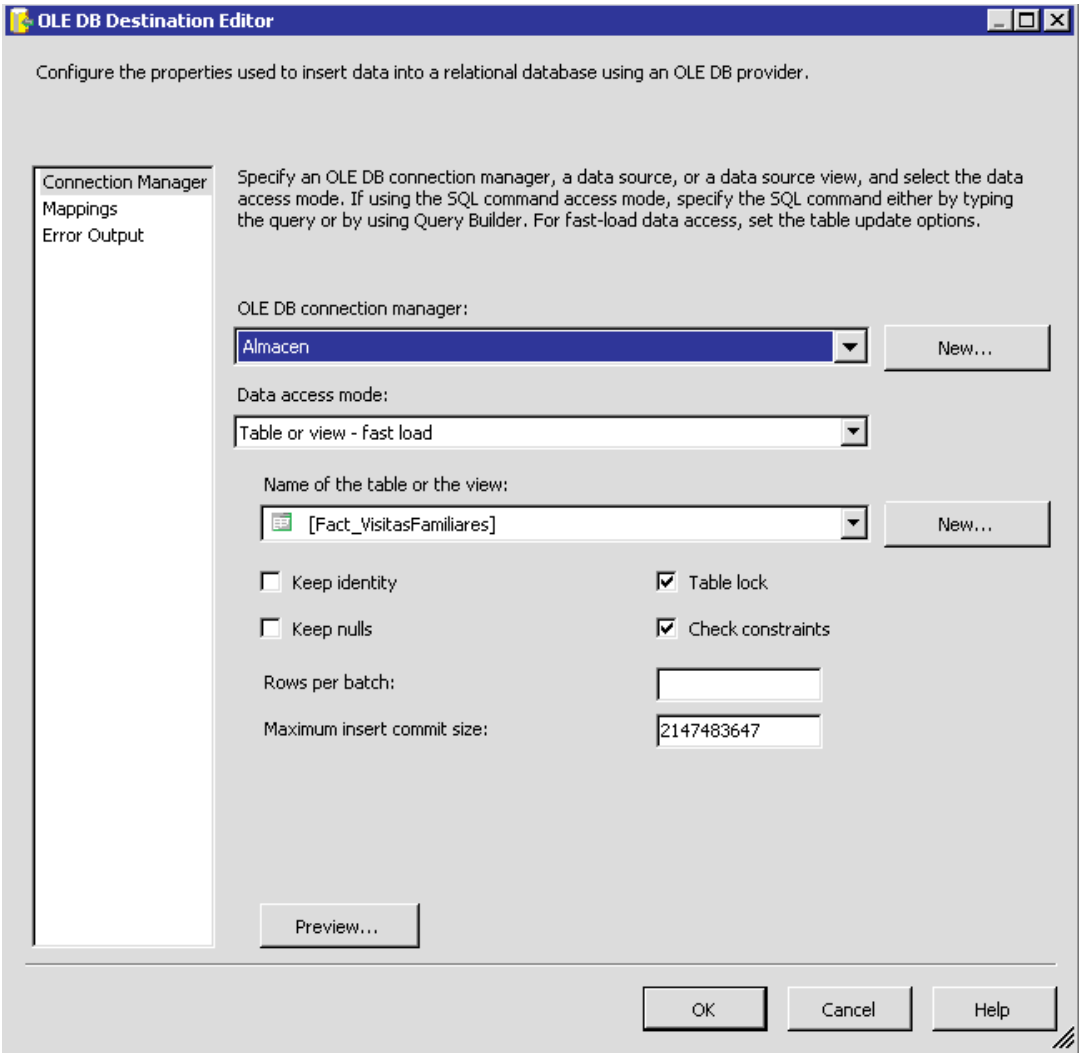


Figura E.12. Configuración Destino OLE DB

Hacemos clic en **Asignaciones** (Figura E.13.), asignamos las columnas de entrada apropiadas a las columnas de destino correspondientes y hacemos clic en **Aceptar**.

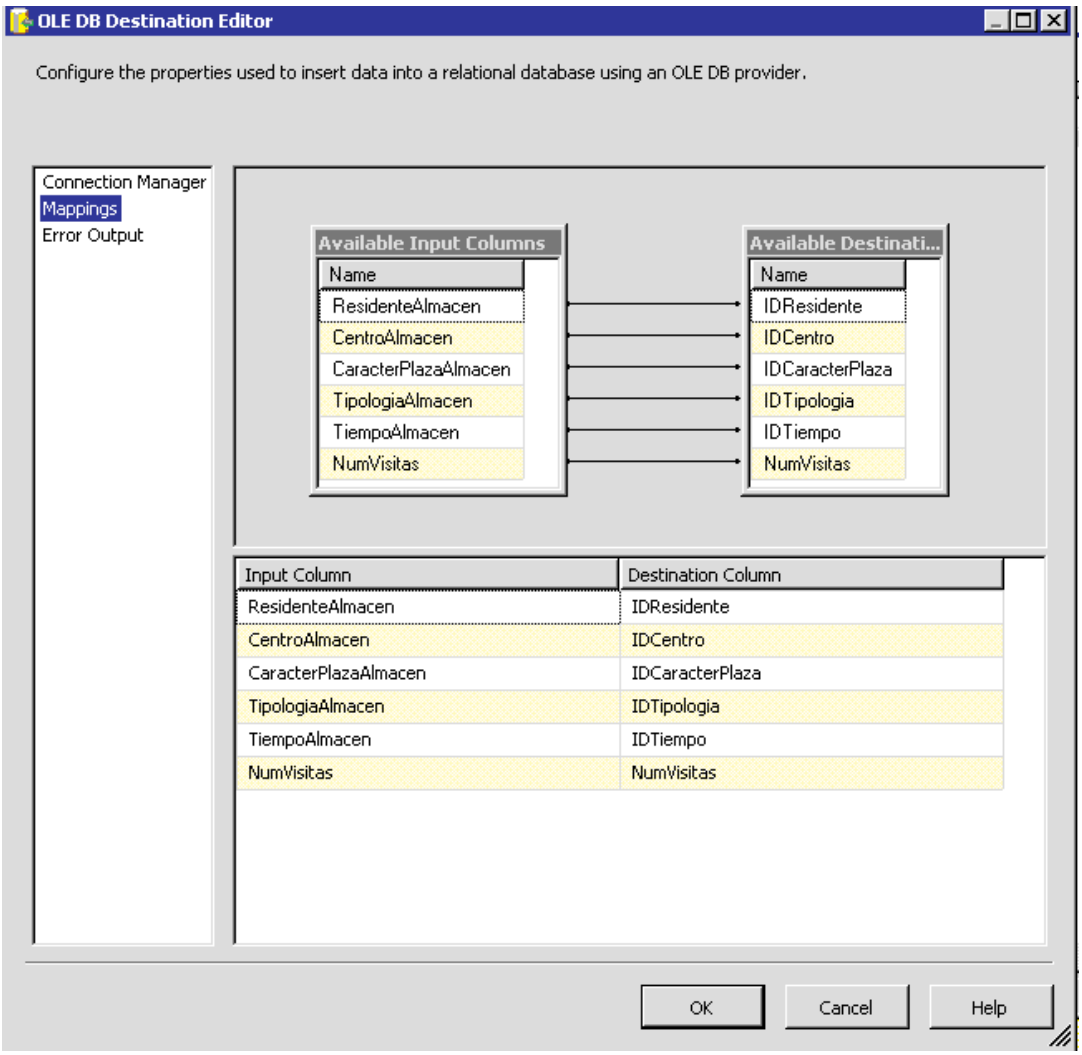


Figura E.13. Pestaña Asignaciones del Destino OLE DB

Una vez finalizados los dos flujos de datos, el siguiente paso es ejecutar el paquete SSIS y comprobar que todo funciona correctamente.

Paso 7: Ejecución del paquete

Para comprobar que el paquete recién construido funciona correctamente debemos ejecutarlo. Existe la posibilidad de ejecutar cada tarea del flujo de control por separado utilizando la opción **Ejecutar tarea** cuando haces clic con el botón secundario sobre la tarea o ejecutar el paquete entero. Los pasos a seguir son los siguientes:

1. Habilitar las dos tareas del flujo de control haciendo clic con el botón derecho en la tarea y seleccionando **Habilitar**.
2. Seleccionamos el paquete en el Explorador de soluciones, hacemos clic con el botón derecho y seleccionamos **Ejecutar paquete**.

Si ambas tareas aparecen de color verde significa que el paquete se ha ejecutado correctamente y se han almacenado los datos en el almacén. En caso de que alguna tarea tenga color rojo, significa que el flujo de datos de esa tarea contiene algún error y por lo tanto hay que revisarlo. Las tareas aparecerán de color amarillo mientras se estén ejecutando y no hayan finalizado.

En caso de que aparezcan errores, tenemos la posibilidad de colocar visores de datos sobre los flujos de datos que van de una transformación a otra para comprobar que los datos que pasan de una transformación a otra son los deseados. Para colocar un visor de datos tenemos que:

1. Colocarnos sobre la flecha verde que va de una transformación a otra, hacer clic con el botón derecho y seleccionar **Visores de datos**.
2. En el cuadro de diálogo **Editor de rutas de flujo de datos** (Figura E.14.), haga clic en **Visores de datos** y, a continuación, en **Agregar**.

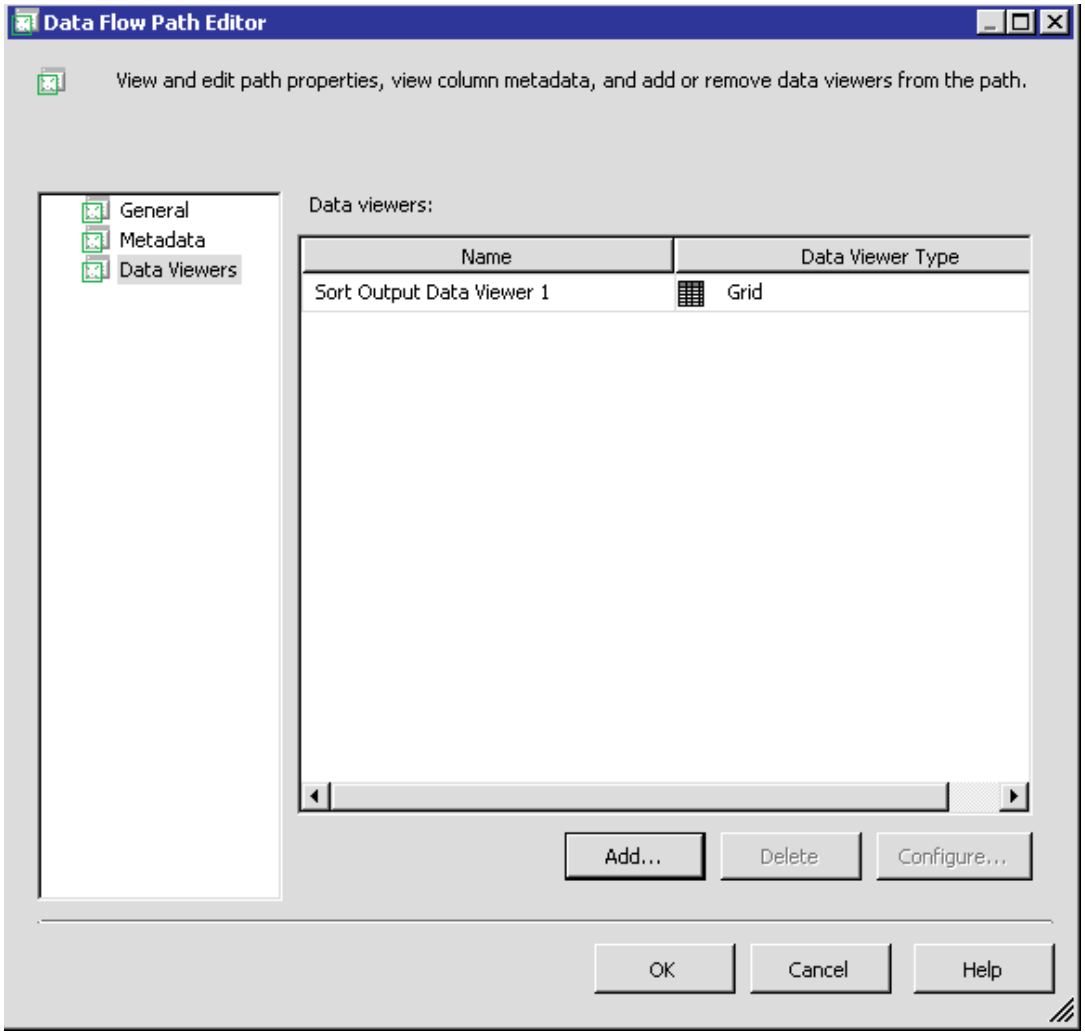


Figura E.13. Cuadro de diálogo Editor de rutas de flujo de datos

3. En el cuadro de diálogo **Configurar visor de datos**, haga clic en **Cuadrícula** y seleccione las columnas que desee mostrar en el visor de datos. De forma predeterminada, todas las columnas disponibles aparecen seleccionadas.
4. Haga clic en **Aceptar** dos veces, la primera para confirmar la configuración del visor de datos y la segunda para confirmar la adición del visor de datos.

Tras ejecutar el paquete y comprobar que todo funciona correctamente, el proceso de creación del paquete ETL en SSIS está finalizado.

Anexo F: Desarrollo de cubos OLAP en Microsoft Analysis Services

En este tutorial, aprenderemos a definir orígenes de datos, vistas de orígenes de datos, dimensiones y cubos en un proyecto de Analysis Services, facilitando al usuario la explotación de un Data Warehouse.

Para obtener más información sobre como modificar medidas, dimensiones, jerarquías, atributos, grupos de medida del proyecto, así como definir cálculos, indicadores de clave de rendimiento (KPI), acciones, perspectivas, traducciones y funciones de seguridad debemos consultar el tutorial de Microsoft.

Dado que los documentos de ejemplo mostrados anteriormente hacían referencia al indicador de visitas familiares, ese es el indicador que vamos a utilizar como ejemplo de este tutorial.

A continuación se indican los pasos necesarios para poder desarrollar el cubo OLAP del indicador en cuestión, suponiendo que el almacén de datos ha sido creado y cargado y el paquete ETL ha sido ejecutado.

Paso 1: Crear un proyecto de Analysis Services

En primer lugar, crearemos un nuevo proyecto de SSAS y nos familiarizaremos con los componentes básicos del entorno.

Para crear un nuevo proyecto de Analysis Services:

1. Hacemos clic en **Inicio**, seleccionamos **Todos los programas**, seleccionamos **Microsoft SQL Server** y, después, hacemos clic en **SQL Server Business Intelligence Development Studio**. De esta forma se abrirá el entorno de desarrollo de Microsoft Visual Studio.
2. Cerramos la ficha **Página de Inicio**. En el menú **Archivo** de Visual Studio, seleccionamos **Nuevo** y hacemos clic en **Proyecto**.
3. En el cuadro de diálogo **Nuevo proyecto**, seleccionamos **Proyectos de Business Intelligence** en el panel **Tipos de proyecto**, y seleccionamos **Proyecto de Analysis Services** en el panel **Plantillas**.
4. Cambiamos el nombre del proyecto, el cúal cambiará también el nombre de la solución, y hacemos clic en **Aceptar**.

En la parte derecha de la pantalla verá las ventanas **Explorador de soluciones** y **Propiedades**. El Explorador de soluciones se utiliza para agrupar todos los objetos en un proyecto y le permite agregar **Orígenes de datos**, **Vistas de origen de datos**, **Cubos**, **Dimensiones**, **Estructuras de minería de datos**, **Funciones**, **Ensamblados** y **Varios** al proyecto. En la ventana Propiedades puede ver y editar las propiedades del objeto seleccionado.

Paso 2: Definir un origen de datos

Tras crear el proyecto, generalmente se empieza a trabajar con el mismo definiendo uno o más orígenes de datos. Al definir un origen de datos, se especifica la información de la cadena de conexión que se utilizará para establecer la conexión con el Data Warehouse.

En el segundo paso definiremos un origen de datos realizando los siguientes pasos:

1. En el **Explorador de soluciones**, hacemos clic con el botón secundario en **Orígenes de datos** y a, continuación, hacemos clic en **Nuevo origen de datos**. Se abrirá el **Asistente para orígenes de datos** y hacemos clic en **Siguiente**.
2. En la página **Seleccione cómo definir la conexión**, hacemos clic en **Nueva**. Aparecerá el cuadro de diálogo **Administrador de conexión**. En este cuadro de diálogo se definen las propiedades de conexión para el origen de datos. Un administrador de conexión es una representación lógica de la conexión que se utilizará en tiempo de ejecución.
3. En la lista **Proveedor**, comprobamos que la opción **OLE DB nativo\SQL Native Client** está seleccionada.
4. En el cuadro de texto **Nombre de servidor**, escribimos el nombre del servidor donde se aloja nuestro Data Warehouse. En este caso la base de datos se encuentra en el equipo local; no obstante, las bases de datos de origen generalmente se encuentran alojadas en uno o más equipos remotos.
5. Comprobamos que la opción **Utilizar autenticación de Windows** está seleccionada. En la lista **Seleccione o escriba un nombre de base de datos**, seleccionamos el nombre de nuestro Data Warehouse.
6. Hacemos clic en **Aceptar** y, a continuación en **Siguiente**. Aparecerá la página **Información de suplantación**, en la cual deben definirse las credenciales de seguridad que Analysis Services debe utilizar para conectarse al origen de datos.
7. Seleccionamos **Utilizar cuenta de servicio** y hacemos clic en **Siguiente**. Esta cuenta tiene los permisos necesarios para obtener acceso al Data Warehouse.
8. En la página **Finalización del asistente**, hacemos clic en **Finalizar** para crear el nuevo origen de datos.

Paso 3: Definir una vista del origen de datos

Una vez definido el orígen de datos que utilizaremos en el proyecto, el paso siguiente consiste en definir una vista del origen de datos para el proyecto.

Una vista del origen de datos es una sola vista unificada de metadatos de tablas y vistas especificadas que el origen de datos define en el proyecto. Almacenar metadatos en la vista permite trabajar con ellos durante el proceso de desarrollo sin ninguna conexión abierta con ningún origen de datos subyacente.

Para definir una vista del origen de datos nueva realizaremos las siguientes acciones:

1. En el **Explorador de soluciones**, hacemos clic con el botón secundario en **Vistas de origen de datos** y, a continuación, hacemos clic en **Nueva vista de origen de datos**. Se abrirá el **Asistente para vistas de origen de datos** y hacemos clic en **Siguiente**.
2. En la pantalla **Seleccionar un origen de datos**, comprobamos que en **Orígenes de datos relacionales** aparece seleccionado nuestro origen de datos, y hacemos clic en **Siguiente**.
3. En la pantalla **Seleccionar tablas y vistas** (Figura F.1.), seleccionaremos las tablas de hechos y dimensiones de nuestro indicador de la lista de objetos disponibles en el origen de datos seleccionado. Podemos filtrar esta lista para facilitar la selección.

Para agregar las tablas a la lista de **Objetos Incluidos** tenemos que seleccionar las tablas que queramos incluir, mantener pulsada la tecla CTRL y hacer clic en **>** .

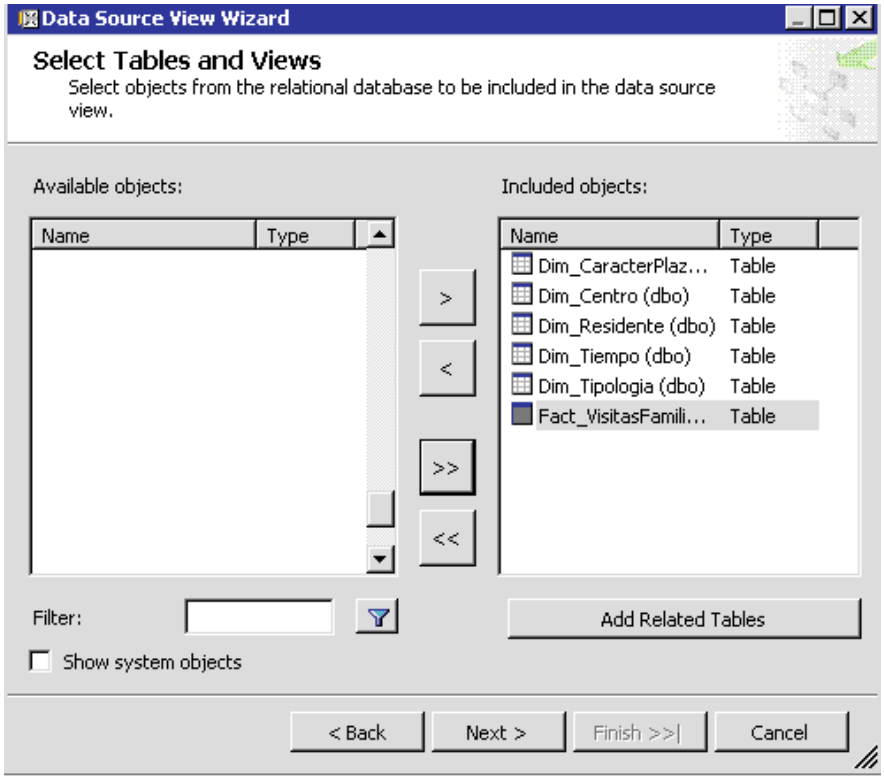


Figura F.1. Pantalla Seleccionar tablas y vistas

4. Hacemos clic en **Siguiente** y, a continuación hacemos en **Finalizar** para definir la vista de origen de datos.

El contenido de la vista se muestra en el Diseñador de vistas del origen de datos de Business Intelligence Development Studio. Este diseñador contiene los siguientes elementos:

- El panel **Diagrama**, en el que las tablas y sus relaciones se representan gráficamente.
- El panel **Tablas**, en el que las tablas y los elementos de esquema se muestran en una vista de árbol.
- El panel **Organizador de diagramas**, en el que podemos crear subdiagramas de modo que podamos ver los subconjuntos de la vista de origen de datos.
- Una barra de herramientas específica del Diseñador de vistas de origen de datos.

En la Figura F.2. se muestra la vista del origen de datos en el Diseñador de vistas de origen de datos.

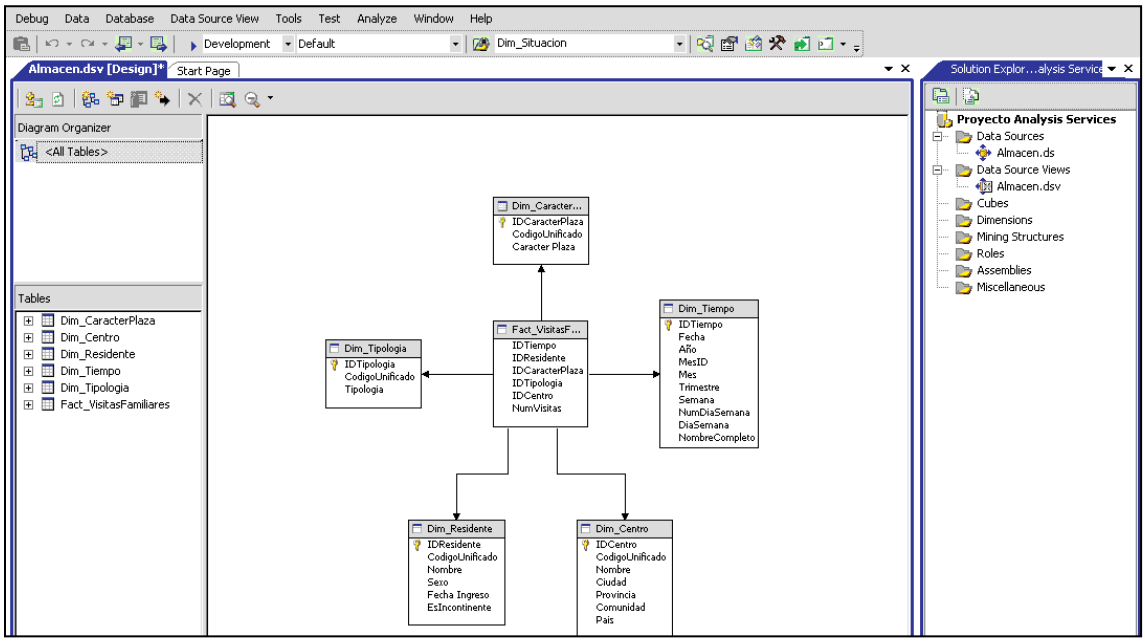


Figura F.2. Diseñador de vistas de origen de datos

Paso 4: Modificar los nombres de tabla predeterminados

En ocasiones, los nombres que tienen las tablas de la vista del origen de datos no resultan lo suficientemente claros y resulta conveniente cambiarlos. Para cambiar el nombre que tiene cada tabla, debemos hacer lo siguiente:

1. En el panel **Diagrama** del **Diseñador de vistas de origen de datos**, hacemos clic con el botón secundario en una de las tablas y luego en **Propiedades**. Aparecerá la ventana oculta **Propiedades**, en la que se muestran las propiedades del objeto seleccionado.
2. Cambiamos la propiedad **FriendlyName** por el nombre deseado. Realizamos esta misma acción para cada una de las tablas que se desee cambiar el nombre.

Paso 5: Definir un cubo

Definida la vista del origen de datos, estaremos preparados para definir el cubo de nuestro indicador. Para ello, utilizaremos el Asistente para cubos, el cual nos ayuda a definir las medidas y las dimensiones de un cubo. En el asistente, podemos definir un cubo basado en un origen de datos o podemos definir un cubo sin ningún origen de datos existente. Si definimos un cubo sin origen de datos existente, utilizaremos el asistente para generar el esquema para el origen de datos subyacente.

Para definir un cubo y sus propiedades seguiremos los siguientes pasos:

1. En el **Explorador de soluciones**, hacemos clic con el botón secundario en **Cubos** y, a continuación hacemos clic en **Nuevo cubo**. Aparecerá la página **Asistente para cubos** y hacemos clic en **Siguiente**.
2. En la página **Seleccionar método de generación**, seleccionamos **Usar tablas existentes** y hacemos clic en **Siguiente**.
3. Seleccionamos la tabla de hechos (Fact_VisitasFamiliares) para **Tablas de grupo de medida** y hacemos clic en **Siguiente**.

4. Aceptar la opción por defecto de las páginas **Seleccionar medidas** y **Seleccionar nuevas dimensiones** y hacer clic en **Siguiente**. En el caso de las medidas, el asistente selecciona como medida cada columna de tipo de datos numéricos de las tablas identificadas como tablas de hechos.
5. Indique el nombre del cubo recién creado y haga clic en **Finalizar**.

Al finalizar el proceso de creación del cubo, se observa que éste aparece incluido en la carpetas Cubos del Explorador de soluciones y las dimensiones aparecen en la carpeta Dimensiones. Adicionalmente, en el centro del entorno de desarrollo, el Diseñador de cubos muestra el cubo. Es importante mencionar que por defecto todos los cubos creados con el asistente son de tipo Molap pero puede modificarse accediendo a la ventana de propiedades del cubo.

En la Figura F.3. se muestran las tablas de dimensiones y de hechos en el diseñador. Observamos que la cabecera de la tabla de hechos es amarilla y las de las tablas de dimensiones son azules.

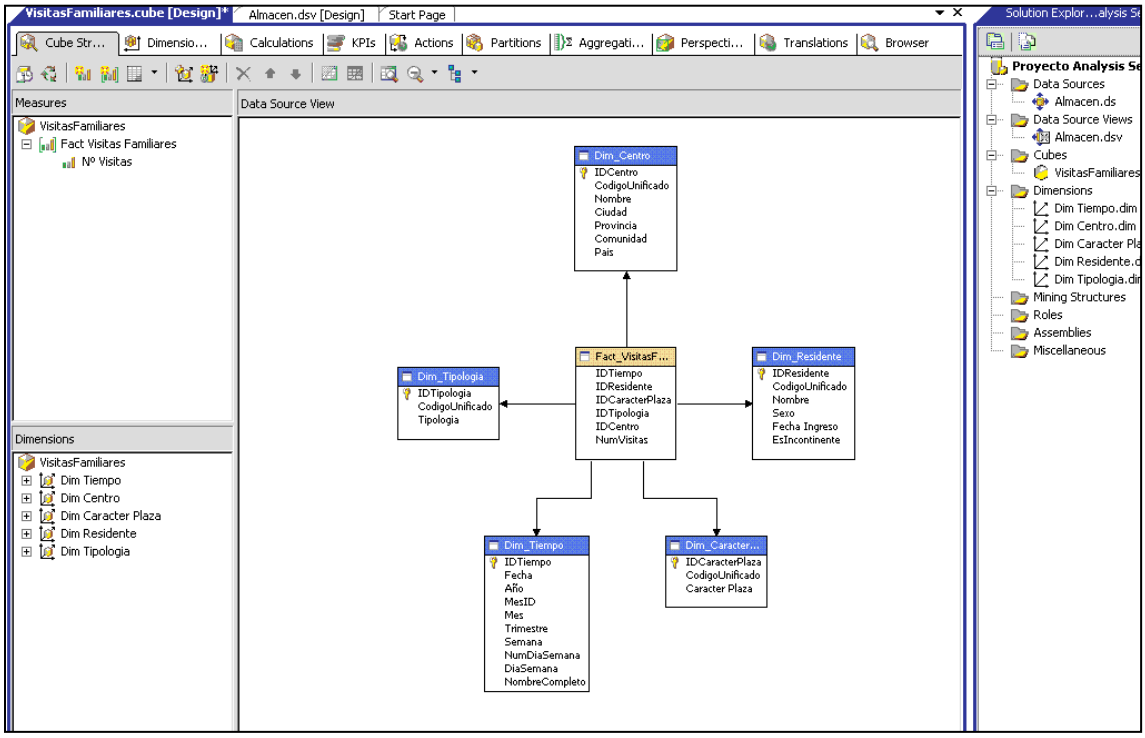


Figura F.3.Tablas de hechos y dimensiones del indicador Visitas Familiares

Tras utilizar el Asistente para cubos para definir un cubo, podemos revisar los resultados en el Diseñador de cubos. A continuación, se describen cada una de las fichas que forman parte del Diseñador de cubos, las cuales permiten gestionar la estructura de un cubo.

El Diseñador de cubos muestra las distintas vistas de un cubo a través de las siguientes fichas:

- **Estructura de cubo**: permite modificar la arquitectura de un cubo.
- **Uso de dimensiones**: utilizada para definir las relaciones entre las dimensiones y los grupos de medida, así como la granularidad de cada dimensión dentro de cada grupo de medida. Si utilizamos varias tablas de hechos, es posible que debamos identificar si las medidas se aplican a una o más dimensiones. Cada celda representa una posible relación entre el grupo de medida y la dimensión de la intersección.

- **Cálculos**: se utiliza para examinar los cálculos definidos para el cubo, definir nuevos cálculos para todo el cubo o un subcubo, reordenar los cálculos existentes y depurar los cálculos paso a paso mediante puntos de interrupción. Los cálculos permiten definir nuevos miembros y medidas en función de valores existentes, como un cálculo de beneficios, y definir conjuntos con nombre.
- **KPI**: permiten crear, editar y modificar los indicadores de clave de rendimiento (KPI) de un cubo. Los KPI permiten al diseñador determinar con rapidez la información que resulta de utilidad de un valor, como si el valor definido excede un objetivo o no llega al objetivo, o si la tendencia del valor definido mejor o empeora.
- **Acciones**: utilizada para crear o modificar las acciones de obtención de detalles, informes y otras acciones para el cubo seleccionado. Las acciones proporcionan a los clientes información sensible al contexto, comandos e informes a los que los usuarios finales pueden obtener acceso.
- **Particiones**: utilizamos esta ficha para crear y administrar las particiones de un cubo. Las particiones permiten almacenar secciones de un cubo en distintas ubicaciones con distintas propiedades, como las definiciones de agregación.
- **Perspectivas**: permite crear y administrar las perspectivas de un cubo. Una perspectiva es un subconjunto definido de un cubo y se utiliza para reducir la complejidad percibida de un cubo para el usuario de empresa.
- **Traducciones**: se utiliza para crear y administrar nombres traducidos de los objetos de cubo, como nombres de mes y de producto.
- **Examinador**: permite ver los datos del cubo.

A continuación modificaremos las dimensiones y medidas creadas por el asistente para la creación de cubos con el fin de proporcionar al usuario una interfaz de exploración de más fácil manejo. Agregaremos además un cálculo al cubo que nos proporcionará la media de visitas familiares realizadas a los residentes.

Paso 6: Modificar las dimensiones y creación de jerarquías

Para diseñar dimensiones, siga estos pasos:

1. En el **Explorador de soluciones**, haga doble clic en cualquiera de las dimensiones y se abrirá la ventana de diseño de la dimensión.
2. Arrastre desde la vista del origen de datos a la parte de los atributos todos aquellos atributos que desea que se muestren posteriormente.
3. En la ventana de atributos, seleccione el nombre de la dimensión, haga clic con el botón secundario y seleccione **Propiedades**. Asegúrese de que la propiedad **UnknownMembe**r está 'Visible' y ponga el valor 'No Definido' en la propiedad **UnknownMemberName**. Con esta última acción conseguimos que, si hay valores nulos en esa dimensión, todos se muestren como 'No Definido'.
4. Modificar las propiedades de los atributos de la dimensión que aparecen marcados como clave de la dimensión haciendo clic con el botón derecho sobre el atributo y modificando a 'False' el campo **AttributeHierarchyVisible** de la ventana **Propiedades** para que no se vean dicho atributos, ya que se trata de identificadores. Para el resto de atributos dejamos las propiedades tal y como aparecen por defecto.

En el caso de algunas dimensiones es de gran utilidad definir jerarquías que permitan a los usuarios navegar entre los datos de una manera más intuitiva. Concretamente, en el indicador de ejemplo, es conveniente definir una jerarquía para la dimensión Centro y otra para la dimensión Tiempo. Dado que ambas jerarquías son diferentes, a continuación se detallan los pasos para crear ambas.

- **Jerarquía Dimensión Centro:**

  1. Arrastre los atributos Pais, Comunidad, Provincia, Ciudad y Nombre en ese orden tal y como se aprecia en la Figura F.4.

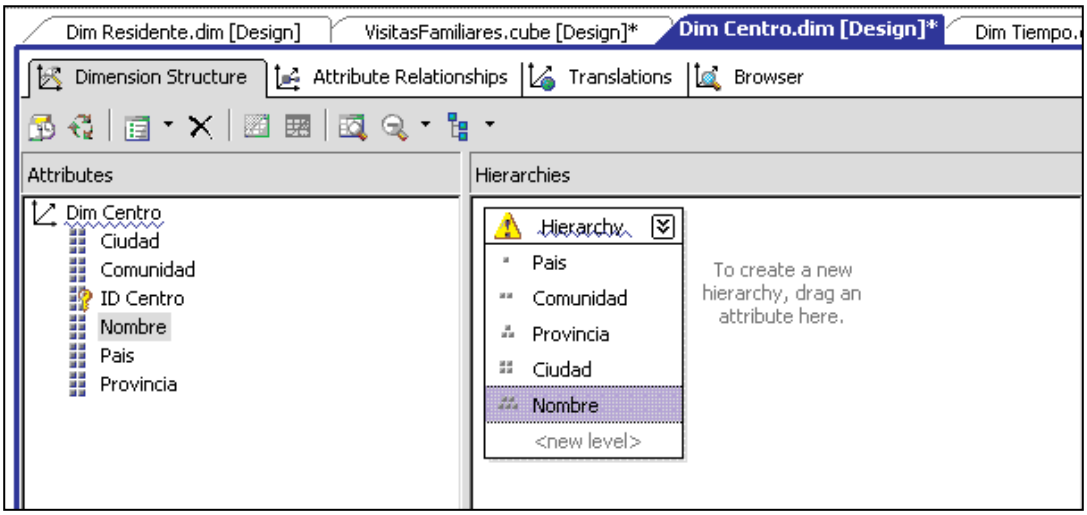


Figura F.4. Jerarquía Dimensión Centro

- **Jerarquía Dimensión Tiempo:**

  1. Arrastre los atributos Año, Trimestre, Mes, Semana, Dia Semana y Fecha en ese orden.

  2. Pasamos a la pestaña de **Relaciones de atributo,** pinchamos en la superficie de diseño con el botón derecho y seleccionamos **Nueva relación de atributo** (Figura F.5.)**.** Primeramente creamos una relación poniendo como atributo origen Mes y Mes ID como atributo relacionado y luego repetimos el proceso con los atributos Dia Semana y Num Dia Semana.

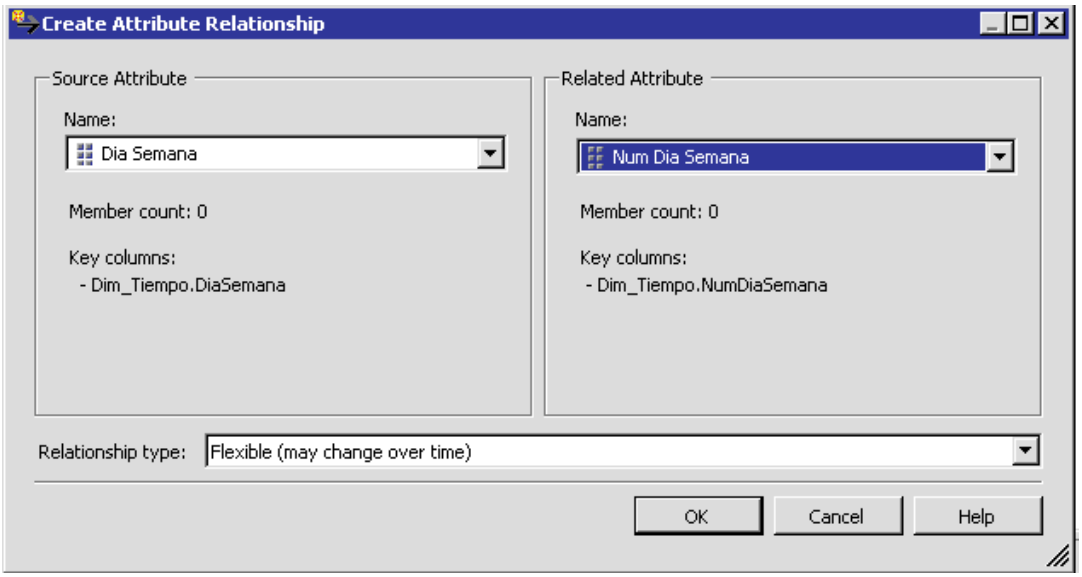


Figura F.5. Cuadro de diálogo para crear una nueva relación entre atributos

3. Seleccionamos el atributo Mes y pulsamos el botón derecho para acceder a las propiedades de dicho atributo. En el campo **OrderBy** seleccionamos la opción 'AttributeKey' y en el campo **OrderByAttribut**e seleccionamos 'Mes ID'. Repetimos el proceso con el atributo Dia Semana con la diferencia de que en el campo **OrderByAttribute** ponemos 'NumDia Semana'.

Una vez modificadas las propiedades de las dimensiones y creadas las jerarquías, se procede al cierre del diseñador de la dimensión.

Paso 7: Adición y modificación de medidas

Es posible que cuando creamos un cubo, las medidas que pone por defecto el asistente no se ajusten a lo que queremos o necesitemos crear nuevas medidas para realizar cálculos más específicos. Para llevar a cabo estas acciones, debemos seguir estos pasos:

1. En el **Explorador de soluciones**, hacemos doble clic sobre el cubo al que queremos modificar las medidas.
2. Vamos al apartado de **Medida**s en la parte izquierda, seleccionamos la medida a modificar, hacemos clic con el botón secundario y pinchamos en **Editar Medida**. Si lo que queremos es eliminar la medida, hacemos lo mismo y seleccionamos **Eliminar**.

Como nosotros lo que queremos es calcular la media de visitas por residente, necesitamos una medida que calcule el total de registros y un cálculo que calcule la media. Por tanto los siguientes pasos son crear la medida y luego el cálculo.

3. En el apartado de medidas, hacemos clic en el botón derecho y seleccionamos **Nueva medida**. Puesto que esta medida sólo la queremos para hacer el cálculo de la media, accedemos a sus propiedades y ponemos el campo **Visible** a 'False'.
4. Seleccionamos la pestaña **Cálculos** y luego en el botón **Nuevo miembro calculado.**
5. Creamos un cálculo que divida el número de visitas entre el número de registros y lo agregamos al grupo de medidas del hecho (Figura F.6.).

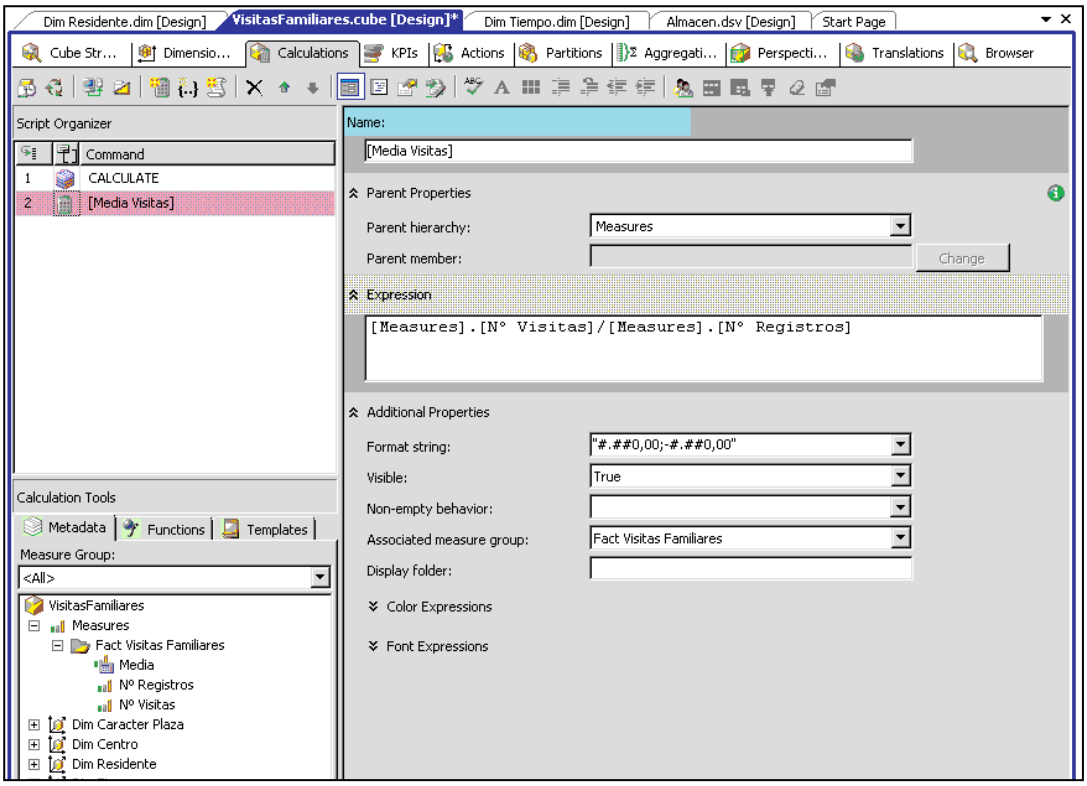


Figura F.6. Definición medida Media Visitas

Paso 8: Procesar y examinar el cubo

Para ver los datos del cubo, el último paso a realizar es procesar el cubo de la siguiente forma:

1. Haga clic con el botón derecho en cubo creado dentro del **Explorador de soluciones** y seleccione **Procesar** o bien haga lo mismo sobre el nombre del proyecto.
2. Haga clic en **Sí** para generar e implementar el proyecto.
3. Haga clic en **Ejecutar** para procesar el cubo (Figura F.7.).

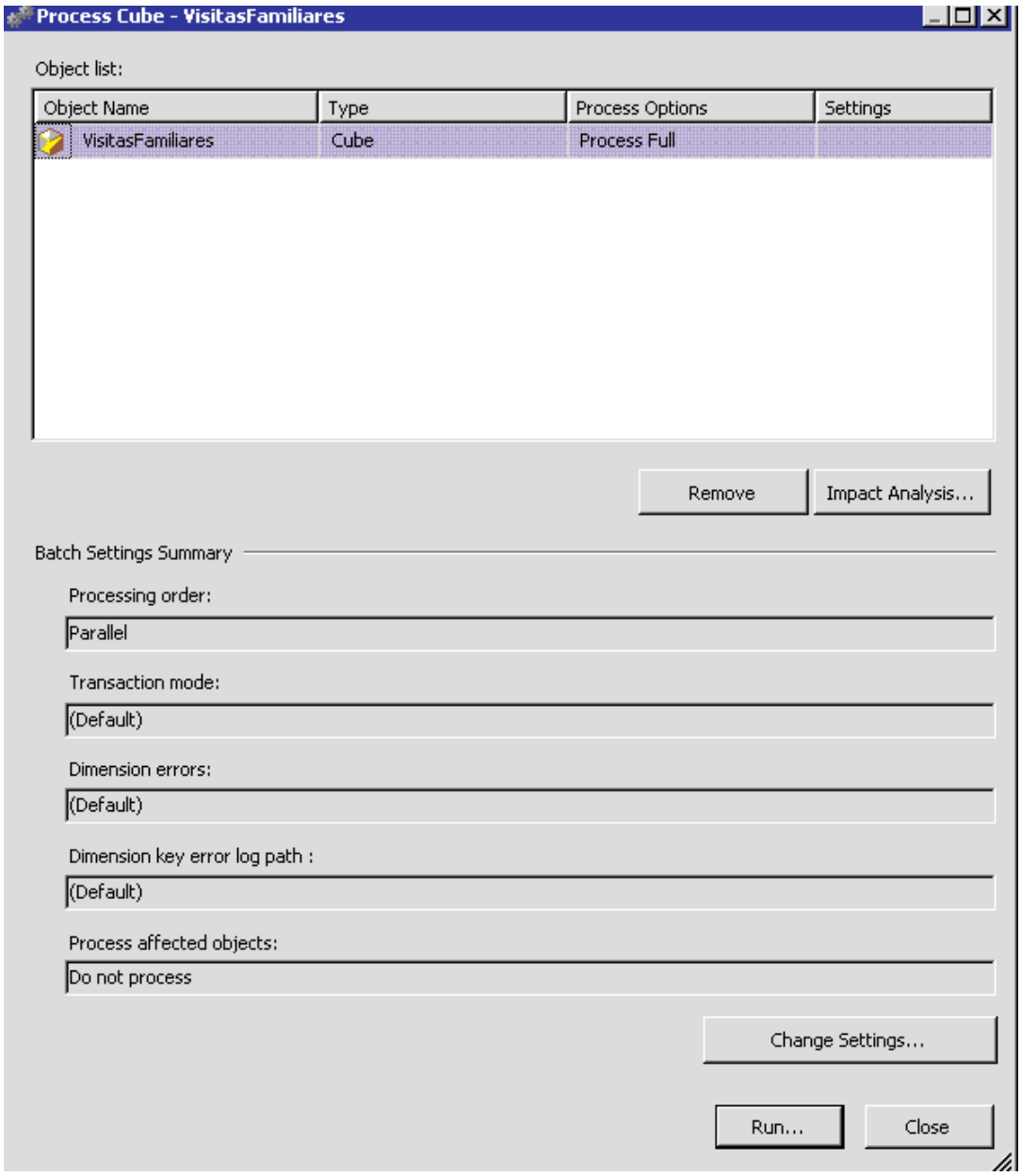


Figura F.7. Ventana de proceso del cubo Visitas Familiares

4. Pulse **Cerrar** cuando el proceso haya concluido.
5. Haga clic en la pestaña **Examinar** para explorar los contenidos del cubo.

Examinar un cubo implementado ayuda a comprender las modificaciones que deben llevarse a cabo para mejorar la funcionalidad del mismo. Por ejemplo, es posible que debamos definir criterios de ordenación de los miembros de la dimensión, eliminar atributos de dimensión innecesarios, definir jerarquías de usuario nuevas, modificar jerarquías de usuario existentes o configurar propiedades de medidas. Una vez que hemos implementado el cubo, los datos de éste pueden verse en la ficha Examinador del Diseñador de cubos y los datos de dimensión pueden verse en la ficha Examinador del Diseñador de dimensiones.

Las acciones a realizar para examinar un cubo implementado son:

6. En el **Diseñador de cubos** hacemos clic en la ficha **Examinador** (Figura F.8.), y luego hacemos clic en **Volver a conectar** en la barra de herramientas del diseñador.

En el panel izquierdo del diseñador se muestran los metadatos del cubo. A la derecha del mismo, podemos ver dos paneles más: el superior es el panel **Filtro** y el inferior es el panel **Datos**. Observamos que las opciones **Perspectiva e Idioma** están disponibles en la barra de herramientas.

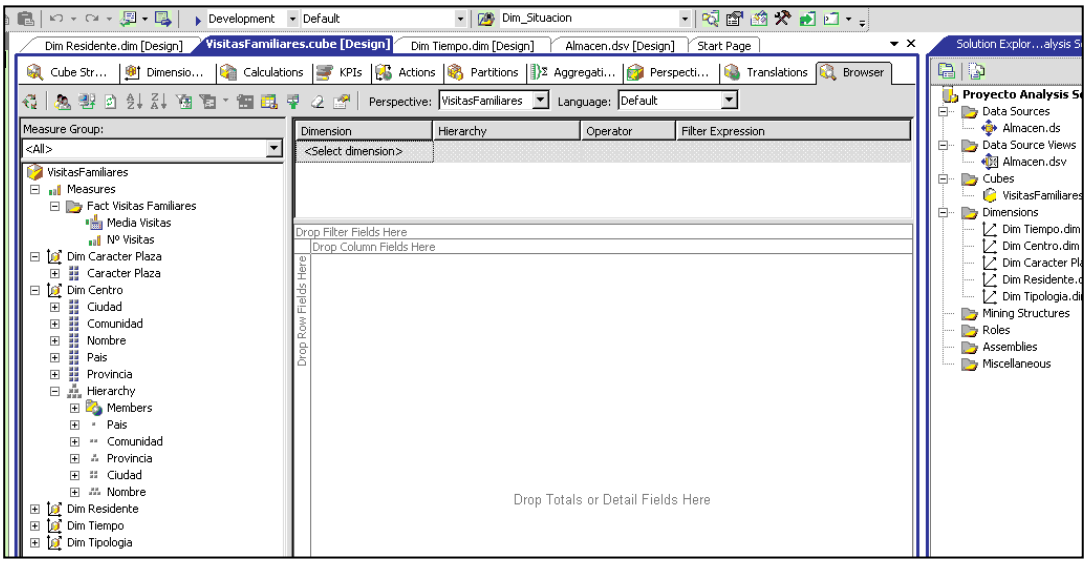


Figura F.8. Ficha Examinador del Diseñador de cubos

7. En el panel **Metadatos** expandimos **Measures** y arrastramos las medidas al área **Coloque Campos de totales o campos detallados aquí** del panel **Datos**.

8. En el panel **Metadatos** expandimos aquellas dimensiones por las cuales queremos dimensionar el cubo. Arrastramos las jerarquías de atributos indistintamente a las áreas **Coloque campos de fila aquí**, **Coloque campos de columna aquí** y/o **Coloque campos de filtro aquí** del panel **Datos**.

En la Figura F.9. se muestra un posible dimensionamiento.

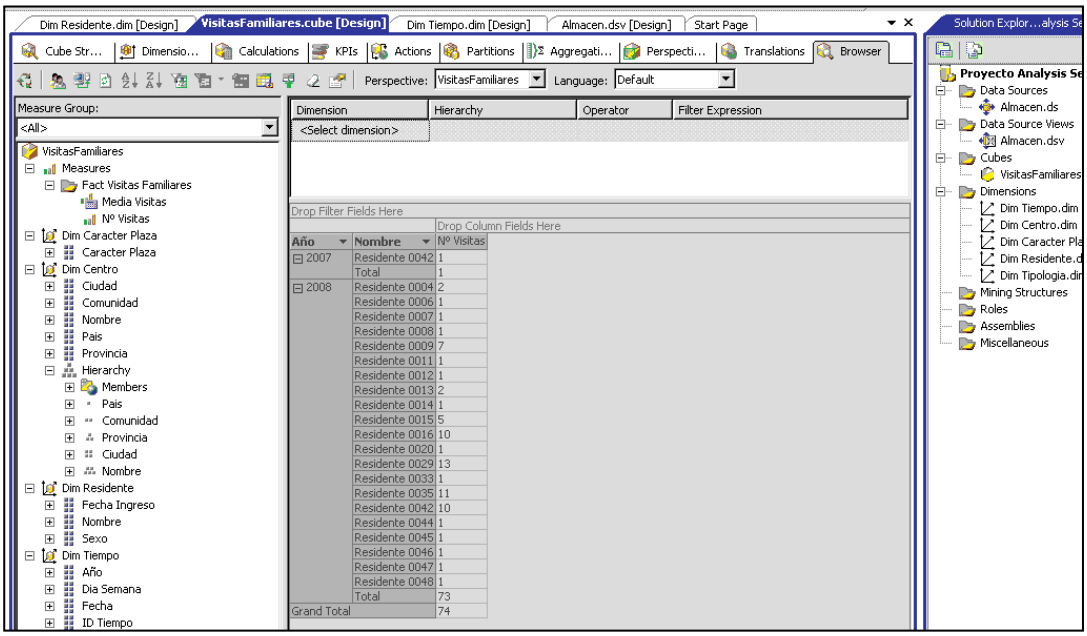


Figura F.9. Ejemplo de dimensionamiento del cubo Visitas Familiares

Después de hacer varias pruebas y comprobar que todo funciona correctamente, el proceso de creación del cubo OLAP en SSAS está finalizado.

Anexo G: Creación de trabajos con el Agente SQL Server

El Agente SQL Server es el motor de programación que incluye SQL Server para ejecutar tareas sobre una programación definida. Se utiliza para crear trabajos programados, planes de mantenimiento y configurar el envío de alertas basadas en condiciones definidas por el usuario.

En este tutorial conoceremos cómo funcionan los trabajos y aprenderemos a crear trabajos programados que lancen los paquetes ETL y procesen los cubos creados previamente. Los planes de mantenimiento y la configuración de alertas no se tratan por quedar fuera del alcance de este proyecto.

Los trabajos ofrecen el contenedor de ejecución que nos permite agrupar uno o más pasos en un proceso que sea necesario ejecutar. Si bien muchos de los trabajos que se cree tendrán una única tarea, SQL Server permite crear trabajos compuestos por multitud de tareas para las que es posible configurar el acceso y acciones de conmutación por error. Cada tarea o unidad de trabajo que deba realizarse está contenida en un paso de trabajo.

Los pasos de trabajo son los elementos de ejecución dentro de un trabajo. Pueden ejecutarse los siguientes tipos de pasos de trabajo:

- Transact-SQL.
- Tareas de replicación.
- Tareas del sistema operativo o archivos ejecutables.
- Tareas de Analysis Services.
- Paquetes de Integration Services.
- Scripts ActiveX.

Al igual que ocurre con cualquier código ejecutable, cada paso de trabajo se ejecuta bajo un determinado contexto de seguridad.

Para cada paso de trabajo puede configurar opciones de flujo de control, además de opciones de registro y notificaciones.

Las opciones de flujo de control le permiten especificar una acción de éxito o fracaso, como se muestra a continuación:

- Abandonar el trabajo informando de su éxito.
- Abandonar el trabajo informando del fracaso.
- Ir al siguiente paso.
- Ir a un número de paso específico.

Además de controlar las opciones de flujo, puede especificar el registro del paso. El registro puede dirigirse a un archivo que se sobrescribirá cada vez que el paso se ejecute, o puede anexarlo a un archivo en ejecución. Puede además registrar el resultado de un trabajo en una tabla, si bien por lo general esto no resulta recomendable debido a la carga de trabajo extra del registro en tabla frente al registro en un archivo de texto.

Ahora que ya conocemos un poco más sobre los trabajos del Agente SQL Server, estamos preparados para crear un trabajo. A continuación se indican los pasos necesarios para poder crear un trabajo que ejecute un paquete ETL desarrollado con SSIS y procese un cubo desarrollado con SSAS. Concretamente, el paquete ETL y el cubo que vamos a utilizar

para la creación del trabajo son los que hemos desarrollado para el indicador de visitas familiares.

Paso 1: Creación de un trabajo

A diferencia de SSIS, SSAS y SSRS que formaban parte de Business Intelligence Development Studio, el Agente SQL Server forma parte de SQL Server Management Studio (SSMS). Dado esto, para crear un trabajo lo primero que debemos hacer es iniciar dicha herramienta. Concretamente, los pasos a seguir son:

1. Hacemos clic en **Inicio**, seleccionamos **Todos los programas**, seleccionamos **Microsoft SQL Server** y después, hacemos clic en **SQL Server Management Studio**.
2. Cuando aparezca el cuadro de diálogo **Conectar con el servidor**, acepte las opciones predeterminadas y haga clic en **Conectar**. La Figura G.1. muestra la pantalla a la que accedemos.

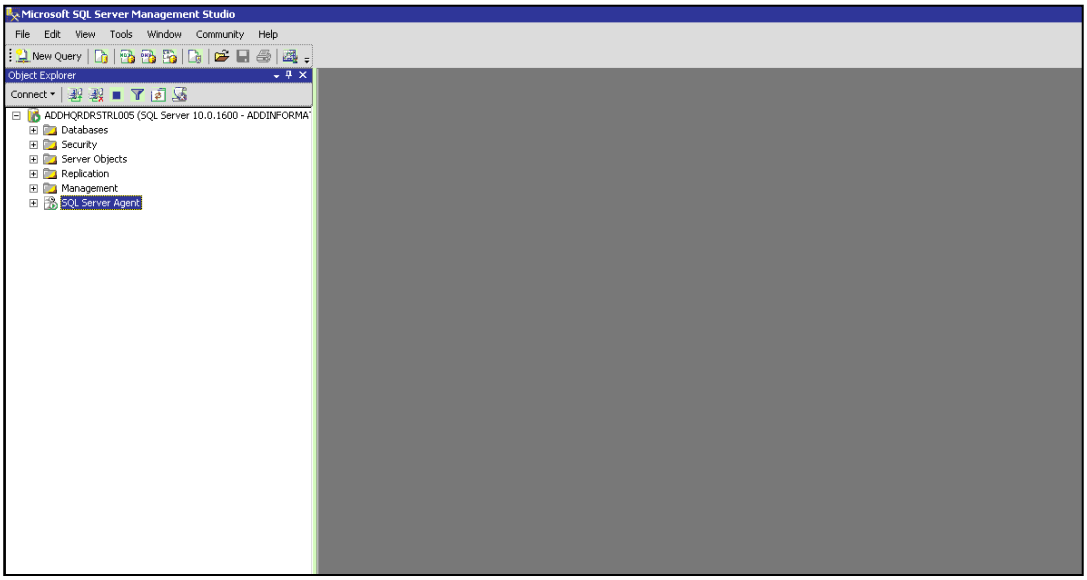


Figura G.1. Pantalla principal SQL Server Management Studio

3. Haga clic con el botón derecho en el nodo **Trabajos** y seleccione **Nuevo Trabajo**.
4. Dé un nombre al trabajo, deje el propietario que aparezca por defecto o ponga la cuenta sa, no seleccione ninguna categoría y agregue una descripción tal y como se indica en la Figura G.2.

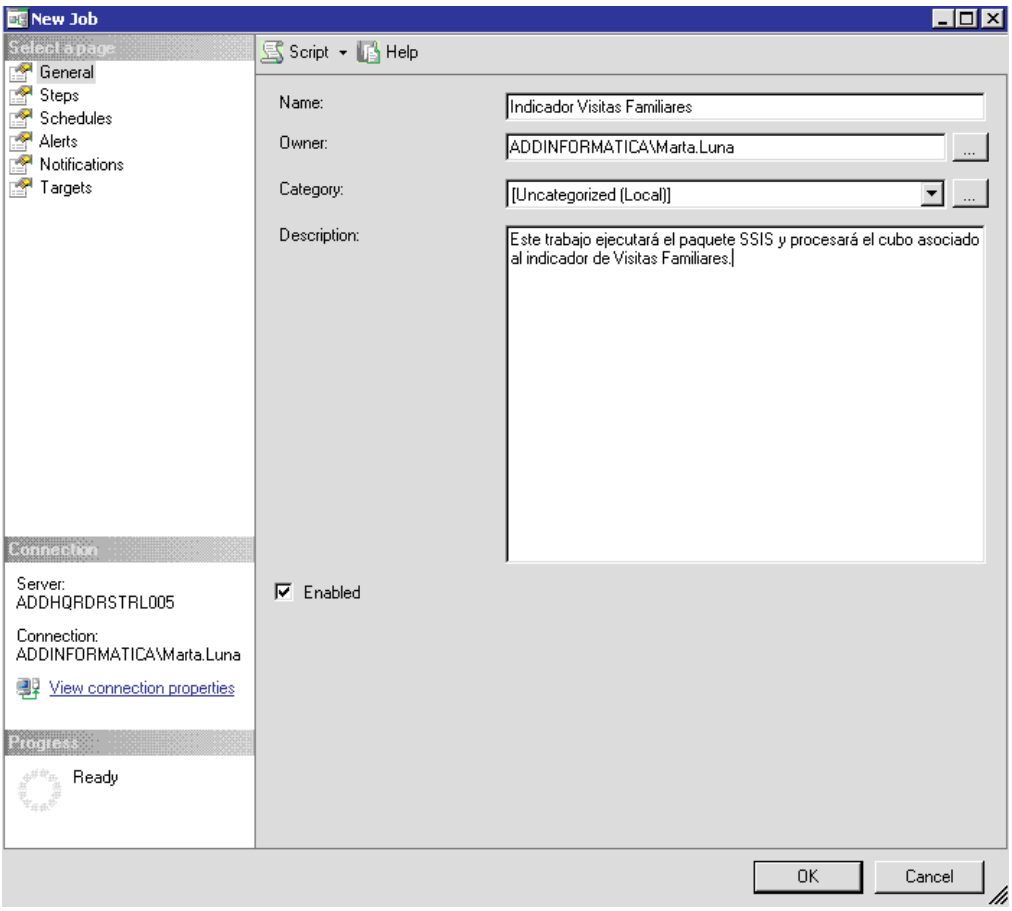


Figura G.2. Ventana configuración del trabajo Indicador Visitas Familiares

5. Seleccione la página **Pasos** y haga clic en **Nuevo** para abrir el cuadro de diálogo **Nuevo paso de trabajo**. En este caso, vamos a crear el paso que ejecute el paquete SSIS.

6. Especifique un nombre para el paso, el tipo de paso como **Paquetes de Integration Services**, deje la opción **Ejecutar como** que aparece, ponga como **Origen del paquete** el sistema de archivos (File system) e indique la ruta donde se encuentra el paquete que queremos ejecutar (Figura G.3.).

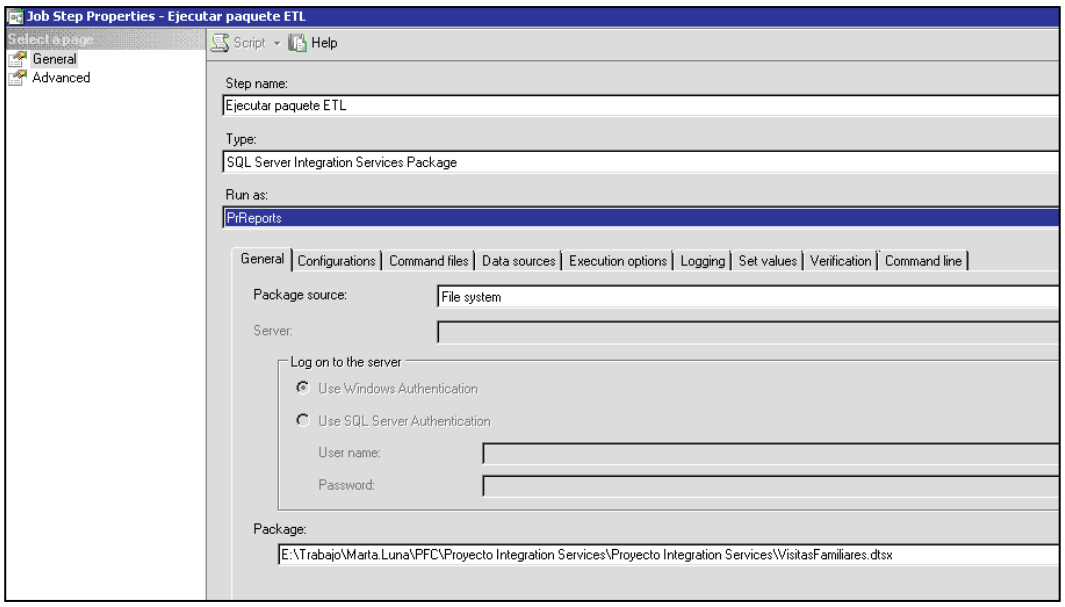


Figura G.3. Ventana de configuración del paso Ejecutar Paquete ETL

7. Haga clic en la página **Avanzado** y especifique **Ir al siguiente paso** como **Acción en caso de éxito**, 1 como **Número de reintentos** para que vuelva a intentar ejecutar el paquete y **Salir del trabajo e informar del error** como **Acción en caso de error**.
8. Haga clic en **Aceptar** para guardar el paso.
9. Haga clic en **Nuevo** para crear el siguiente paso de trabajo en cual vamos a procesar el cubo.
10. Especifique un nombre para el paso, el tipo de paso como **Comando de Analysis Services**, deje la opción **Ejecutar como** que aparece, ponga como **Servidor** la máquina local e indique en **Comando** la instrucción XML a ejecutar que aparece a continuación:

```
<Process
xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <Object>
    <DatabaseID>Proyecto Analysis Services</DatabaseID>
  </Object>
  <Type>ProcessFull</Type>
  <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
</Process>
```

11. Haga clic en la página **Avanzado** (Figura G.4.) y especifique **Salir del trabajo e informar del éxito** como **Acción en caso de éxito**, 1 como **Número de reintentos** para que vuelva a intentar ejecutar el paquete y **Salir del trabajo e informar del error** como **Acción en caso de error**.

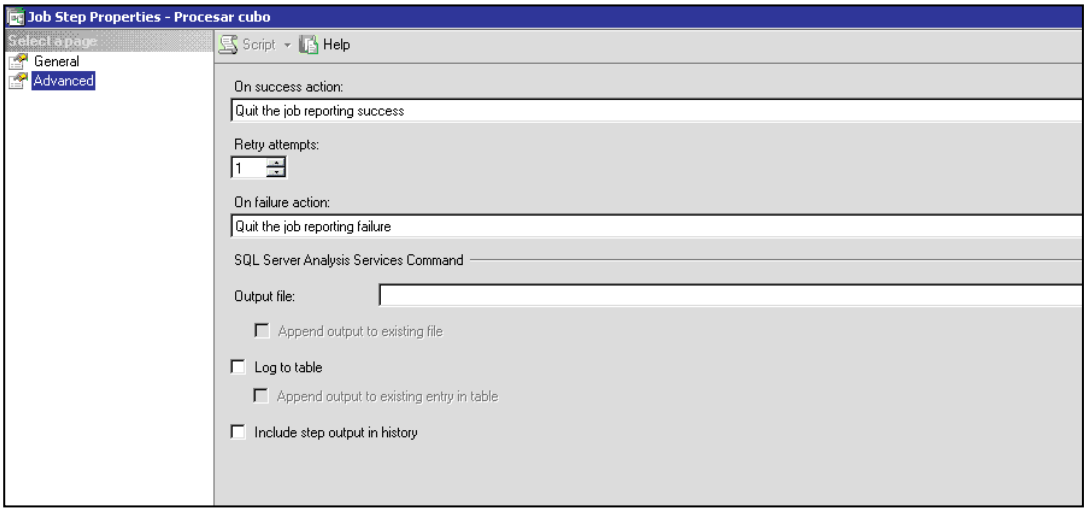


Figura G.4. Ventana Propiedades Avanzadas del paso Procesar cubo

12. Haga clic en **Aceptar** para guardar el paso.
13. Utilice las flechas que aparecen en la página de **Pasos** si desea cambiar el orden de los pasos.

Una vez que haya agregado uno o más pasos a un trabajo, podemos especificar una programación. Las programaciones son objetos independientes y como tal, podemos asignar la misma programación a tantos trabajos como sea necesario.

Una programación puede crearse bien a través del cuadro de diálogo de administración de programaciones, o bien durante la creación de un trabajo. A continuación se enumeran algunas de las propiedades que pueden configurarse para una programación:

- Tipo de frecuencia (diaria, semanal, mensual).
- Recurrencia a nivel diario, semanal o mensual.
- Recurrencia dentro de un día sobre un minuto o cada hora.
- Horas de inicio y fin.
- Fecha de inicio y fin para la validez de la programación.

Dicho esto, el siguiente paso es crear una programación que ejecute diariamente el trabajo que acabamos de crear.

Paso 2: Creación de una programación

Para crear la programación de un trabajo debemos seguir los siguientes pasos:

3. Seleccionamos la página Programaciones de nuestro trabajo y hacemos clic en **Nueva** para definir una nueva programación diaria o bien hacemos clic con el botón derecho en el nodo **Trabajo,** seleccionamos **Manejar programaciones** y hacemos clic en Nueva.
4. Rellenamos los datos relativos a la programación tal y como indica la Figura G.5. para que el trabajo se ejecute todos los días a medianoche.

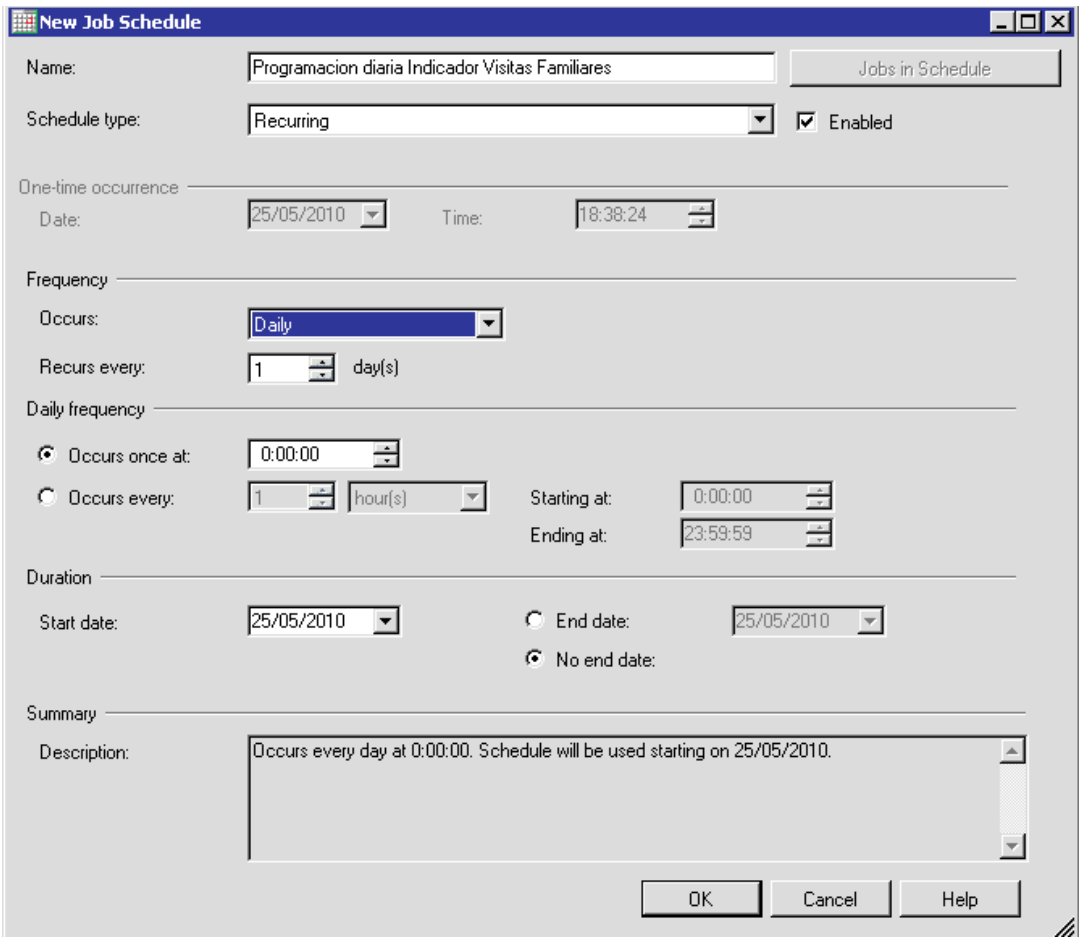


Figura G.5. Configuración de la programación

5. Haga clic en **Aceptar**.

Al finalizar la creación del trabajo y su respectiva programación, sólo falta ejecutar el trabajo para ver que funciona correctamente y poder dar el proceso de creación del trabajo por finalizado.

Paso 3: Ejecución de un trabajo

Para ejecutar un trabajo, hacemos clic con el botón derecho en el trabajo que deseemos ejecutar, seleccionamos la opción **Iniciar trabajo en el paso**, especificamos el paso **1** y hacemos clic en **Inicio.**

Si el trabajo se ejecuta correctamente, en la ventana de progreso (Figura G.6.) observaremos que pone éxito. En caso contrario, podemos ver en qué paso se ha producido el error y una descripción del mismo, haciendo clic con el botón derecho sobre el trabajo y seleccionando **Ver historial.**

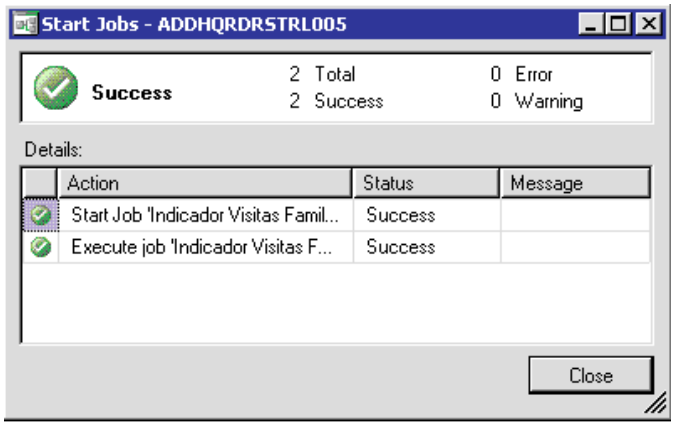


Figura G.6. Ventana de progreso del trabajo ejecutado

Si tenemos varios trabajos ejecutándose podemos supervisar la actividad actual de esos trabajos utilizando el **Monitor de actividad de trabajo**.

El Monitor de actividad de trabajo es accesible al expandir el Agente SQL Server en el Explorador de objetos de Management Studio y seleccionar **Ver actividad de trabajo**. Dicho monitor permite realizar las siguientes tareas:

- Iniciar y detener trabajos.
- Ver las propiedades de un trabajo.
- Ver el historial de un determinado trabajo.

La Figura G.7. muestra el Monitor de actividad de trabajo.

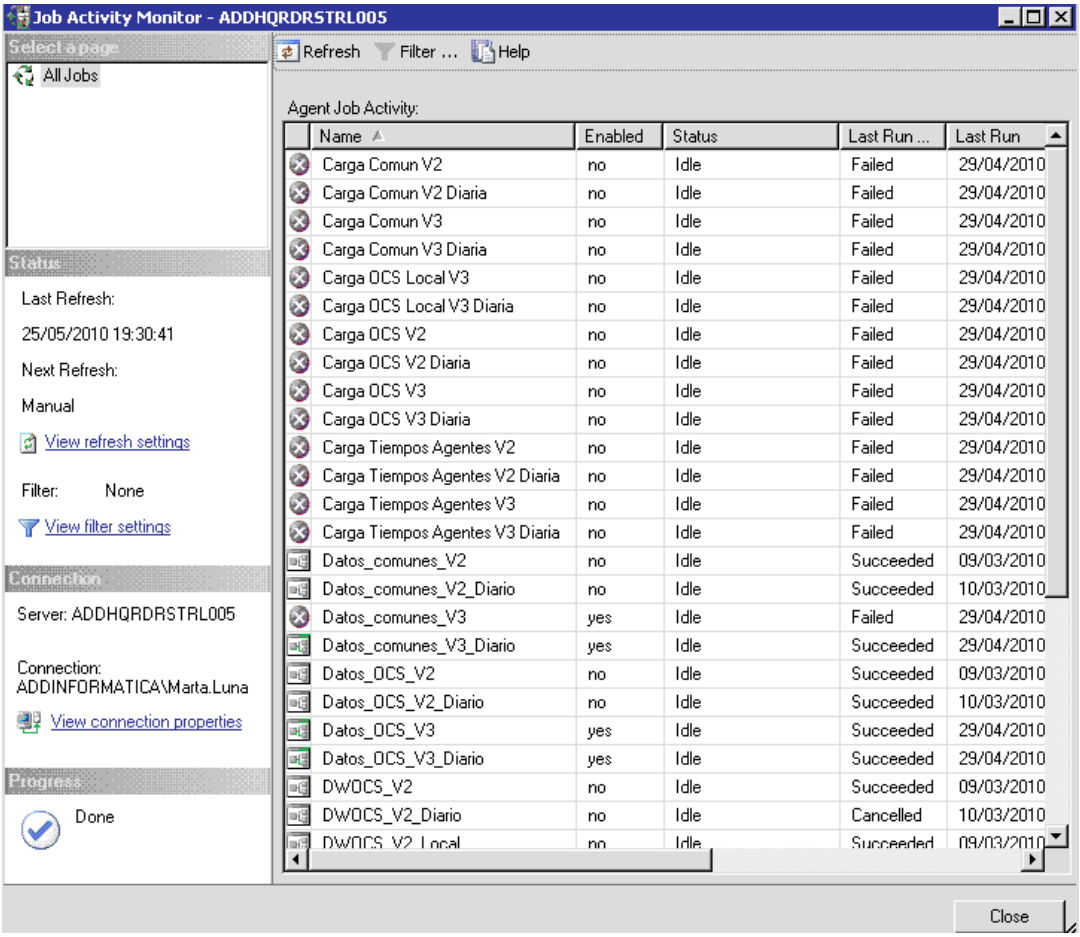


Figura G.7. Monitor de actividad del trabajo

Después de ejecutar el trabajo y comprobar que todo funciona correctamente, el proceso de creación del trabajo con el Agente SQL Server está finalizado.

Anexo H: Explotación de indicadores y creación de informes usando Microsoft Office Excel

Microsoft Office Excel es una de las herramientas de análisis y generación de informes más usadas en el mundo. Entre sus principales ventajas se encuentran:

- Fácil manejo
- Visibilidad en tiempo real de las tendencias de negocio.
- Mayor rapidez y calidad en la toma de decisiones.
- Ciclos más breves de análisis y elaboración de informes.
- Mayor flexibilidad de creación de informes gracias a los diseños de informes complejos, actualizables y muy personalizables.
- Acceso a información relevante contenida en múltiples orígenes de datos.

En nuestro caso es la herramienta utilizada para realizar la explotación de los datos de los indicadores porque incorpora capacidades OLAP que permiten al usuario conectarse a cubos de Analysis Services y explotar la información mediante tablas y/o gráficas dinámicas.

En este tutorial aprenderemos a recuperar los datos desde un cubo y a manejar las tablas dinámicas para poder explotar toda la información que nos proporciona el cubo. Concretamente, utilizaremos el cubo desarrollado anteriormente para el indicador de visitas familiares.

A continuación se indican los pasos necesarios para conectar un libro de Excel con un cubo de Analysis Services y utilizar una tabla dinámica para crear un informe de un indicador.

Paso 1: Configurar conexión con el cubo

Los pasos a seguir para conectar un libro de Excel con un cubo de Analysis Services son los siguientes:

4. Hacemos clic en **Inicio**, seleccionamos **Todos los programas**, seleccionamos **Microsoft Office** y después, hacemos clic en **Microsoft Office Excel 2007**.
5. Vamos a la ficha **Datos**, en el grupo **Obtener datos externos** hacemos clic en **De otras fuentes** y luego en **Desde Analysis Services** y se mostrará el **Asistente para la conexión de datos**.
6. En la pantalla de **Conectar con el servidor de la base de datos**, escribimos el nombre del servidor OLAP donde se encuentra nuestro cubo en el cuadro de texto **Nombre del servidor** y seleccionamos **Usar autenticación de Windows** como **Credenciales de conexión** para usar el nombre de usuario y la contraseña actuales de Windows. Después de rellenar estos datos, hacemos clic en **Siguiente**.
7. En **Seleccione la base de datos que contiene la información que desea** (Figura H.1.), seleccionamos la base de datos, nos aseguramos de que la opción **Conectar con una tabla o a un cubo específico** está activada, seleccionamos el cubo de Visitas Familiares y hacemos clic en **Siguiente**.

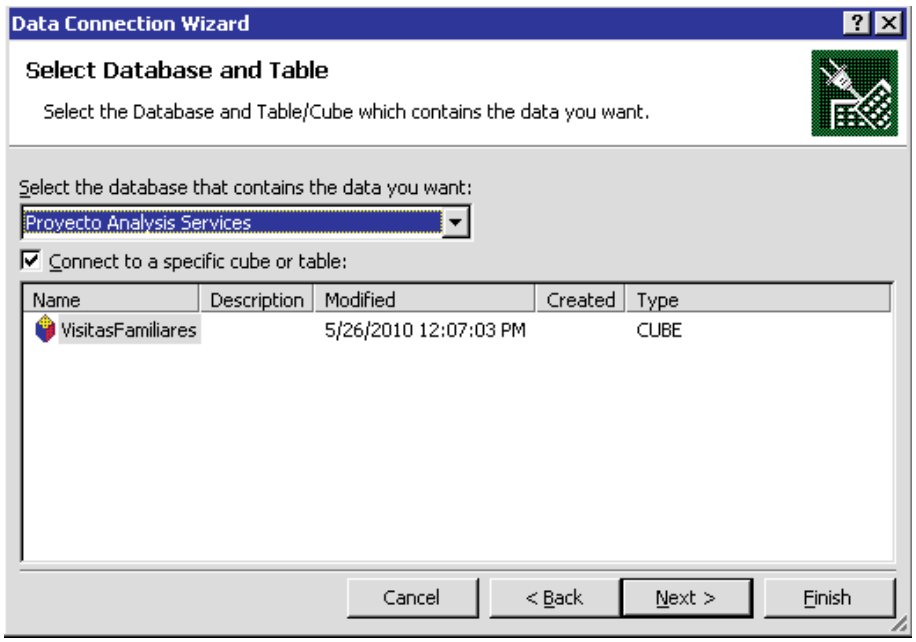


Figura H.1. Configuración de la conexión con el cubo Visitas Familiares

8. En la pantalla de **Guardar archivo de conexión de datos y finalizar**, cambiamos el nombre del archivo o dejamos el que aparece por defecto, cambiamos la ubicación predeterminada del archivo pulsando **Examinar** y opcionalmente escribimos una descripción del archivo, un nombre descriptivo y unas palabras de búsqueda comunes. Es importante hacer clic en **Intentar utilizar siempre este archivo para actualizar los datos** para asegurarnos de que siempre se utiliza el archivo de conexión cuando se actualiza la tabla dinámica. De esta forma, se asegurará de que todos los libros que utilicen ese archivo de conexión utilicen siempre las actualizaciones realizadas en dicho archivo.
9. Haga clic en **Finalizar** para cerrar el Asistente para la conexión de datos y se mostrará el cuadro de diálogo de **Importar datos**.
10. En **Seleccione cómo desea ver estos datos en el libro**, hacemos clic en **Informe de tabla dinámica** y en la parte de **¿Dónde situar los datos?,** seleccionamos **Hoja de cálculo existente** e indicamos la primera celda del rango de celdas donde deseamos situar el informe de tabla dinámica. Finalmente, hacemos clic en **Aceptar**.

Una vez configurada la conexión al cubo, estamos preparados para jugar con todas las dimensiones de nuestro indicador e ir obteniendo información desde diferentes perspectivas. Para ello, seleccionamos las dimensiones en la **Lista de campos de la tabla dinámica** que aparece en la parte derecha y las distribuimos en filas y columnas utilizando las áreas que aparecen en la parte inferior de dicha columna.

La Figura H.2. muestra una posible instancia de la tabla dinámica y el estado de la **Lista de campos de la tabla dinámica**.Concretamente, queremos saber el número de visitas que ha tenido cada residente por año.

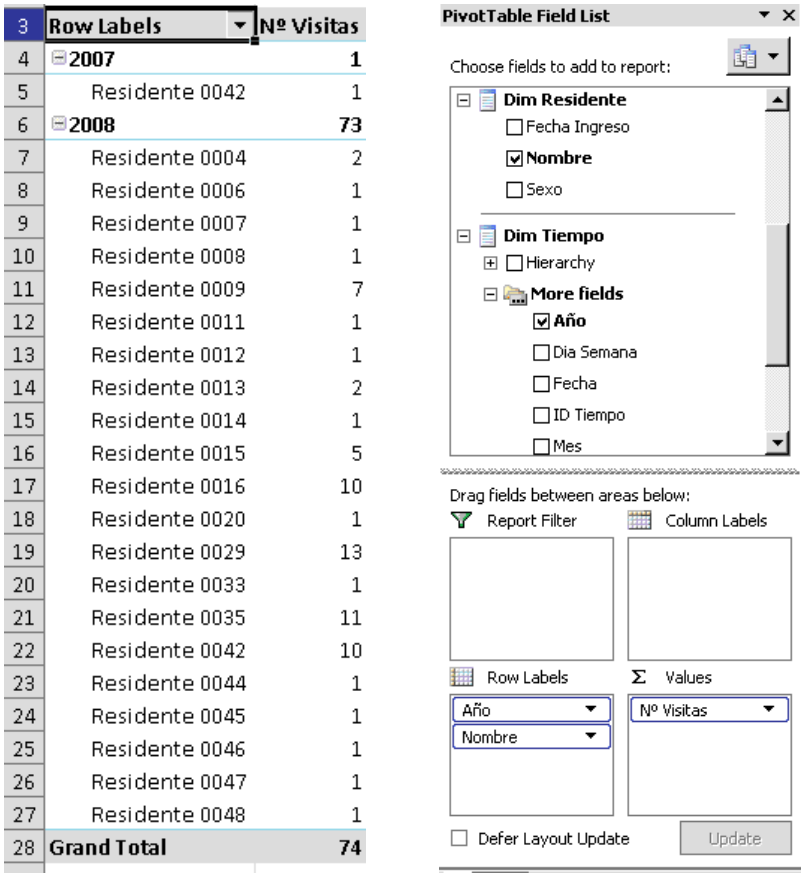


Figura H.2. Vista del número total de visitas por año y residente junto con la Lista de campos de la tabla dinámica

En caso de que perdamos de vista la **Lista de campos de la tabla dinámica**, simplemente tenemos que hacer clic en la tabla dinámica para volver a visualizarla o bien ir a la pestaña de **Opciones** y en el grupo de **Mostrar/Ocultar** seleccionar **Lista de campos**. En en ese grupo también donde indicamos si deseamos o no ver las cabeceras de los campos.

Tras jugar un poco con la tabla dinámica, el siguiente paso es crear algún gráfico que sirva de apoyo a la información proporcionada por la tabla dinámica.

Paso 2: Creación de un gráfico

Dependiendo del tipo de gráfico que deseemos utilizar, se requerirá que los datos tengan una disposición u otra. Cuando tengamos la disposición de datos adecuada para el gráfico, debemos realizar las siguientes acciones:

1. Seleccionamos las celdas que contienen los datos que deseamos utilizar en el gráfico.
2. En la ficha **Insertar**, en el grupo **Gráficos**, hacemos clic en el tipo de gráfico que queremos y a continuación, hacemos clic en el subtipo de gráfico que queremos (Figura H.3.). Si lo que queremos es ver todos los tipos de gráficos disponibles, al hacer clic en un tipo de gráfico, debemos hacer clic en **Todos los tipos de gráfico** y se nos mostrará el cuadro de diálogo **Insertar gráfico**.

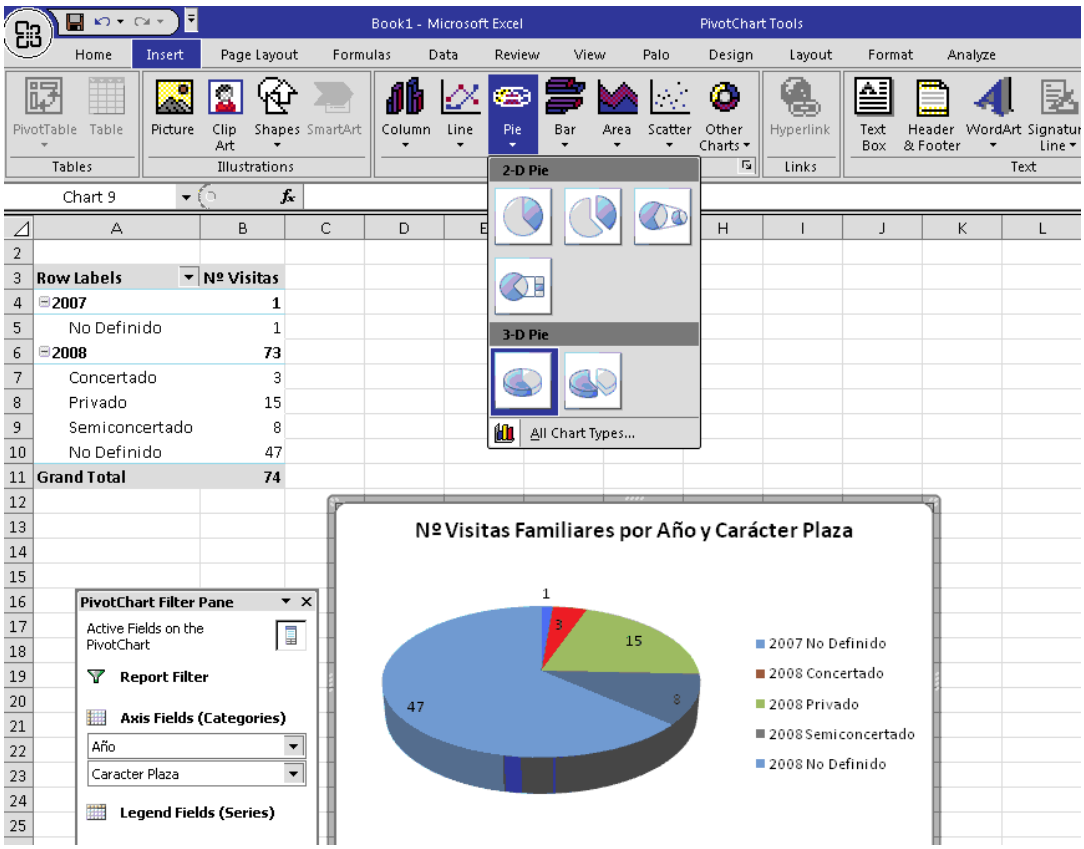


Figura H.3. Ejemplo de inserción de un gráfico

3. Movemos el gráfico hasta la posición deseada haciendo clic sobre el mismo y arrastrando y editamos las propiedades del gráfico tales como el título, la leyenda o el formato del gráfico haciendo clic con el botón derecho y seleccionando la opción correspondiente sobre cada uno de los elementos. También podemos editar dichas propiedades utilizando las pestañas **Diseño** y **Presentación**.

Opcionalmente, podemos añadir filtros a la tabla dinámica para trabajar con subconjuntos de datos y luego ver la información del gráfico en función de dichos filtros.

Paso 3: Añadir un filtro

Los datos pueden ser filtrados de diferentes maneras. Una de ellas es seleccionar para cada campo que aparece en la **Lista de campos de la tabla dinámica** los valores que deseamos mostrar utilizando la flecha que aparece junto al campo y la otra es arrastrar los campos de dicha lista al área **Filtro de informe** para que nos aparezca el filtro en la hoja de Excel. En nuestro caso vamos a utilizar la segunda para poder filtrar la información de los gráficos de manera dinámica. Los pasos a seguir para añadir un filtro son:

1. Arrastre los campos por los que desea filtrar la información al área **Filtro de informe** y encima de la tabla dinámica aparecerá el filtro que acabamos de seleccionar.
2. Seleccionar los valores que deseamos filtrar haciendo uso de la flecha desplegable que aparece en el filtro (Figura H.4.) y haga clic en **Aceptar**.

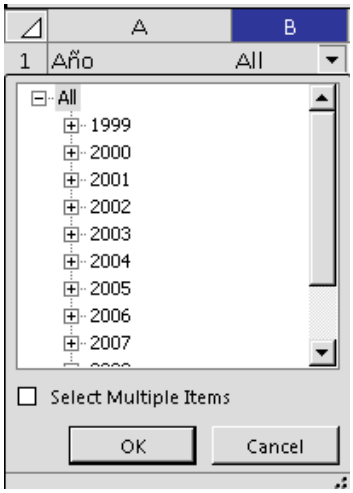


Figura H.4. Lista de valores disponibles por los que podemos filtrar

Hasta el momento sabemos cómo conectar un libro de Excel con un cubo, utilizar las tablas dinámicas, crear gráficos y filtros. Por tanto, para elaborar un informe completo de nuestro indicador, lo único que falta es analizar la información desde diferentes perspectivas y guardar cada una de estas perspectivas en una página diferente del libro de Excel. Cada perspectiva incluirá una instancia de la tabla dinámica y una serie de gráficos y filtros.

Concretamente, la forma de estructurar el informe que proponemos es la siguiente:

- En la primera hoja siempre ha de aparecer una tabla resumen del indicador que sea lo suficientemente significativa como para obtener toda la información recogida a grandes rasgos.
- Cada una de las hojas sucesivas, ha de ser una vista parcial del indicador que recoja sólo ciertos aspectos del mismo. Dicha vista constará bien de una tabla resumen o de una tabla resumen con un gráfico asociado.

Dado que cada hoja va a necesitar una conexión al cubo, para asegurarnos de utilizar siempre la misma debemos ir a la ficha de **Datos**, hacer clic en **Conexiones existentes** en el grupo **Obtener datos** externos y seleccionar la conexión utilizada anteriormente.

Una vez analizado nuestro indicador desde todas las perspectivas de interés, debemos guardar el informe y se dará el proceso de creación del informe por finalizado.

Anexo I: Desarrollo de dashboards en Microsoft Office SharePoint Server

Microsoft Office SharePoint Server (MOSS) es la plataforma web elegida para albergar los informes resultantes de nuestro proceso de desarrollo de indicadores porque, a través de sus portales, permite el acceso a hojas de cálculo de Excel y la creación de cuadros de mando basados en dichas hojas de cálculo.

En este tutorial, conoceremos lo que es un cuadro de mandos y sus componentes y aprenderemos a construir un cuadro de mandos basado en informes de Excel con MOSS.

Un cuadro de mandos (*dashboard*) es una vista integrada/unificada que ilustra el estado y rendimiento en un ámbito determinado: residente, empleado, área, unidad, departamento, centro o grupo, etc. En términos de interfaz de usuario se implementa con scorecards y vistas de reports.

Un scorecard es una colección de KPIs, entendiendo por KPI 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 I.1. se 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 I.1. Ejemplo de scorecard

Entendiendo por reports cada uno de los archivos de Excel que construimos durante el proceso de desarrollo de indicadores, las vistas de reports son gráficas o tablas contenidas en un report.

A continuación se indican los pasos necesarios para poder crear un dashboard en un portal de Sharepoint usando un informe creado en Excel. Suponemos que el portal en cuestión ha sido previamente creado.

Paso 1: Subir un report a la librería de reports del portal

En lo que respecta a los reports, éstos pueden ser creados y posteriormente subidos al portal de Sharepoint o creados directamente desde dicho portal. En nuestro caso, los

reports son creados al finalizar el proceso de desarrollo de los indicadores y por tanto los pasos a seguir para subir el report son los siguientes:

1. Abrimos un navegador de internet e introducimos la dirección URL del portal de Sharepoint.
2. Seleccionamos la pestaña **Reports** y pinchamos el enlace **Reports** que aparece en la barra de navegación lateral.
3. Pinchamos la lista desplegable junto al menú **Upload** y seleccionamos **Upload Document** (Figura I.2.).

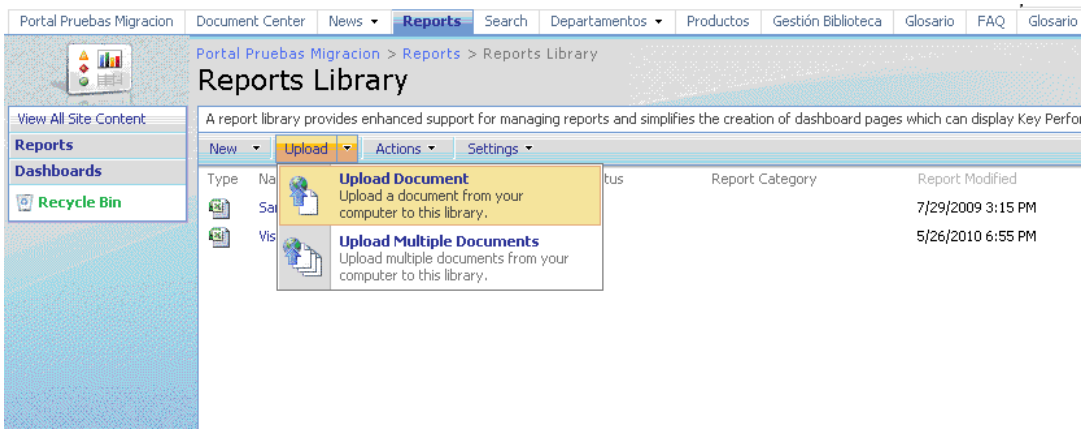


Figura I.2. Carga de un report

4. Pulsamos el botón **Browse** para buscar el archivo que deseamos cargar y cuando lo encontremos hacemos clic en **OK**.
5. Revisamos la información del report, añadimos una breve descripción del mismo si lo deseamos y pulsamos el botón **OK** para subir el archivo.

Una vez completada la acción, veremos que el report que acabamos de subir ha sido añadido en el apartado **Reports Library**.

Paso 2: Subir la conexión del report al portal

Para subir una conexión asociada a un report de Excel debemos:

1. Pinchar 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** (Figura I.3.).

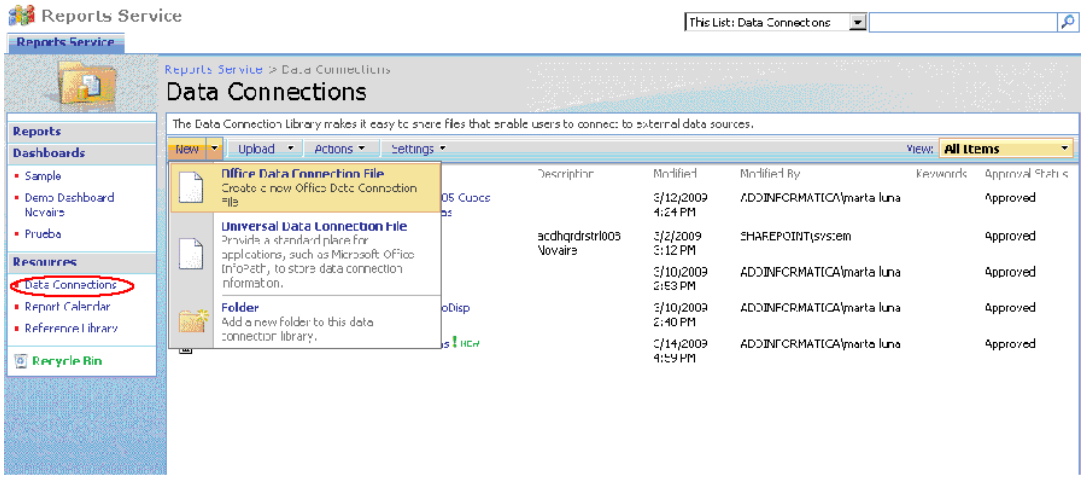


Figura I.3. Creación de una conexión

2. Explorar el sistema de archivos hasta encontrar el archivo .odc asociado a la conexión que queremos subir a SharePoint.
3. Rellenamos los campos asociados a la conexión tales como nombre, título y una breve descripción y pulsamos **OK**.
4. Aprobamos la nueva conexión pulsando la opción **Approve/Reject** de la lista desplegable asociada al nombre de la conexión que acabamos de crear (Figura I.4.), ya que cuando creamos la conexión su estado de aprobación aparece como pendiente.

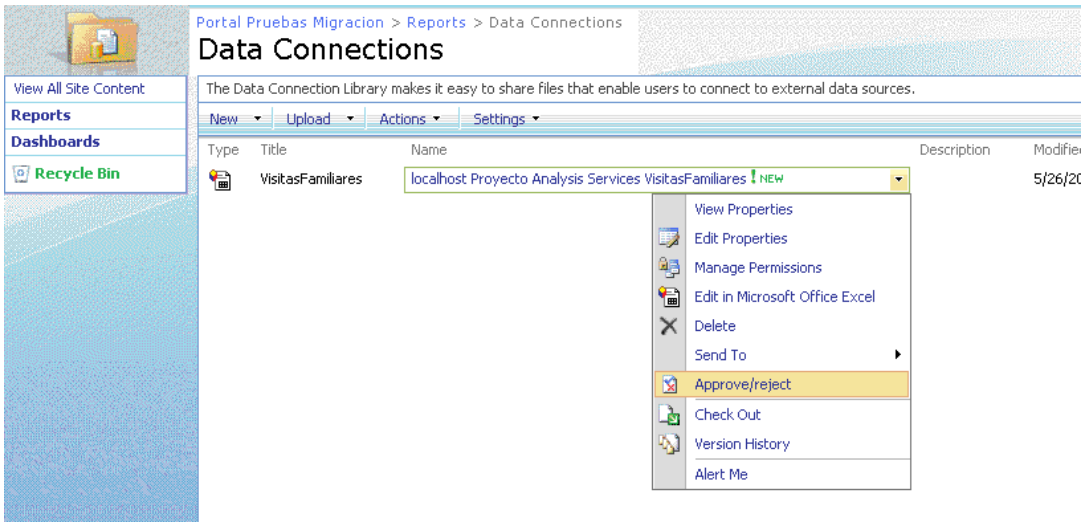


Figura I.4. Aprobación conexión

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 nosotros hemos definido en ellas, es decir, nosotros podemos 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 deseemos 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 elemento. Por cada hoja de Excel del report, debemos darle 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.

Dicho esto, el siguiente paso a realizar es editar el report para dotar de nombre a todos los elementos que hayamos definido en él, es decir, dotar de nombre a todas las vistas y demás elementos contenidos en el report.

Paso 3: Poner nombre a las vistas y demás elementos de un report

Los pasos a realizar para dotar de nombre a todos estos elementos son los siguientes:

1. Seleccionar de la lista desplegable que aparece junto al nombre del report que hemos subido la opción de **Edit in Microsoft Office Exce**l.

2. Ponemos nombre a cada hoja del Excel haciendo clic con el botón derecho sobre el nombre de la hoja y seleccionando **Rename**.
3. Cambiamos el nombre de las tablas dinámicas haciendo clic con el botón derecho sobre la tabla, seleccionando **PivotTable Options**, editando el campo **Name** y pulsando **OK**.
4. Seleccionamos el gráfico, vamos a la ficha **Layout** y modificamos el **Chart Name**.
5. Seleccionamos el rango de celdas que cubre tanto la tabla resumen con filtros como el gráfico asociado, hacemos clic con el botón derecho, seleccionamos **Name a Range**, ponemos el nombre correspondiente y hacemos clic en **OK**.

Una vez hemos dotado de nombre a todos los elementos del report, para poder ver luego la lista desplegable de elementos donde seleccionamos el elemento que queremos ver, en el cuadro de diálogo que aparece cuando vamos a guardar el report, debemos marcar la pestaña **Publish** y seleccionar **Excel Services** (Figura I.5.).

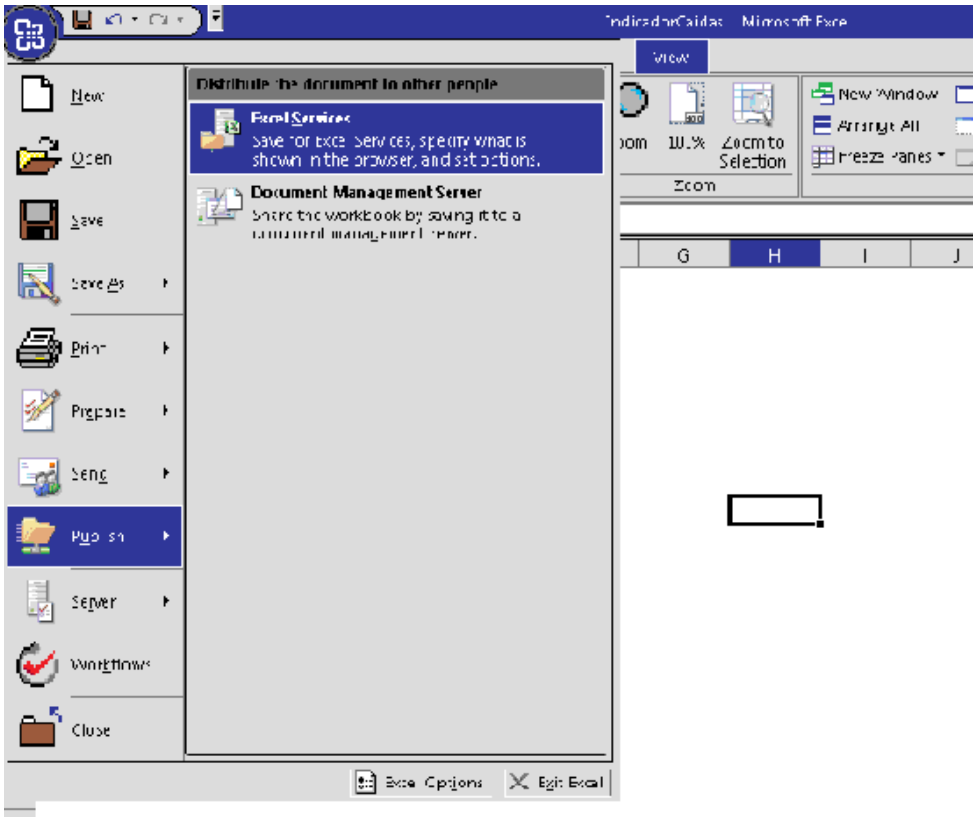


Figura I.5. Cuadro de diálogo Publish con la opción Excel Services marcada

A continuación, pulsamos el botón correspondiente a **Excel Services Options** e indicamos que sólo queremos ver en el navegador los elementos con nombre **Items in the Workbook** y marcamos en la lista los que queremos ver. De esta forma conseguimos que el usuario sólo pueda ver los elementos que nosotros hemos indicado previamente.

Paso 4: Consultar las vistas previamente definidas

En este momento, lo que tenemos es el report correctamente estructurado en vistas para poder ser utilizadas en la elaboración del dashboard.

Para consultar el report, tan sólo debemos pinchar el enlace asociado al nombre del mismo o seleccionar de la lista desplegable asociada al mismo la opción **View in Web**

Browser. Al pinchar dicho enlace, vamos a una página (Figura I.6.) donde se nos 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 estamos consultando. En dicha página, tenemos la pestaña **View**, la cual tiene asociada una lista desplegable que contiene todos los elementos del report que podemos consultar. Seleccionando cada uno de los diferentes elementos de dicha lista, conseguimos movernos por el contenido de todo el report.

Figura I.6. Vista de un elemento del report

Cuando tenemos el report correctamente estructurado en vistas, lo siguiente es crear un dashboard que use dichas vistas y definir un scorecard.

Paso 5: Crear un dashboard

A la hora de crear un dashboard debemos:

1. Pinchar el enlace **Dashboards** que aparece en la barra de navegación lateral o si estamos en la librería de informes, seleccionar la opción **Dashboards** del desplegable asociado a **View**.
2. Hacemos clic en **New** y seleccionamos **Dashboard page**.
3. Rellenamos los campos relativos al nombre y título y dejamos el resto de campos tal y como vienen por defecto y hacemos clic en **OK** (Figura I.7.). De esta forma, el dashboard que creemos estará compuesto por dos columnas verticales y tendrá una lista de KPIs automáticamente.

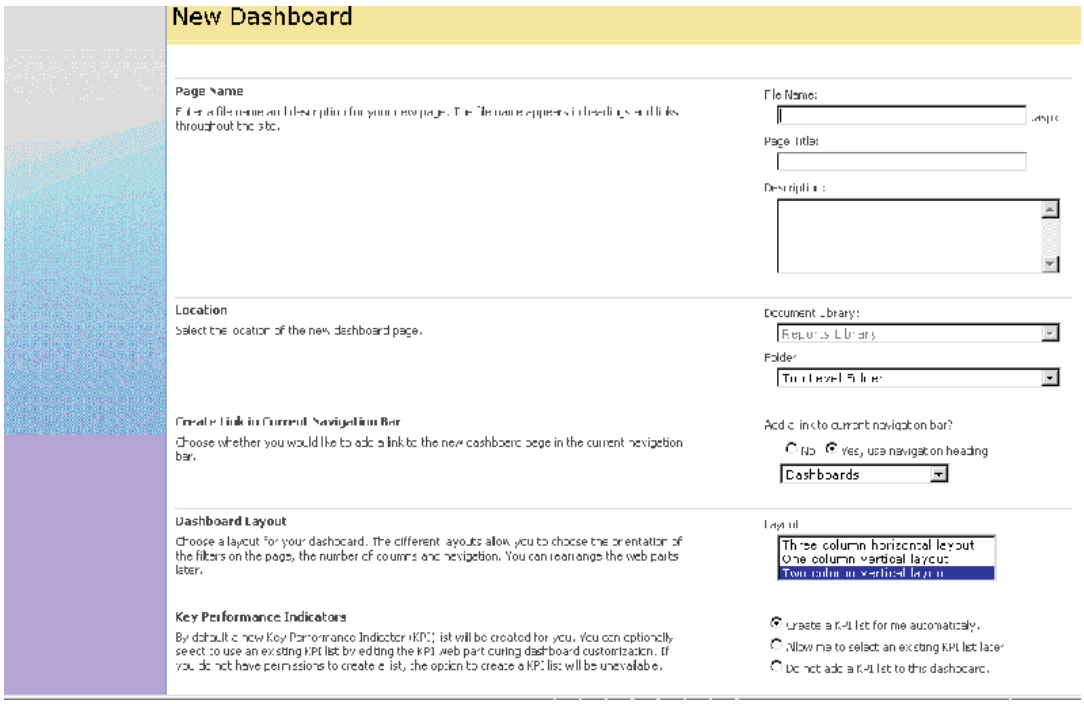


Figura I.7. Ventana creación de un dashboard

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:** Muestran tablas o gráficas de un libro Excel.
- **Filtros:** Los hay de diferentes tipos y permiten mostrar los datos del dashboard según ciertos parámetros.

Paso 6: Definir webparts para mostrar el contenido del report o reports

Ahora vamos a añadir un conjunto de webparts de tipo Excel Web Access que nos permitirán mostrar el contenido de las vistas definidas en los reports. Para ello debemos:

1. Pinchar el enlace asociado al dashboard y luego seleccionamos **Open the tool pane** o bien pinchamos **Edit Page** en la parte superior de la página tal y como se indica en la Figura I.8.

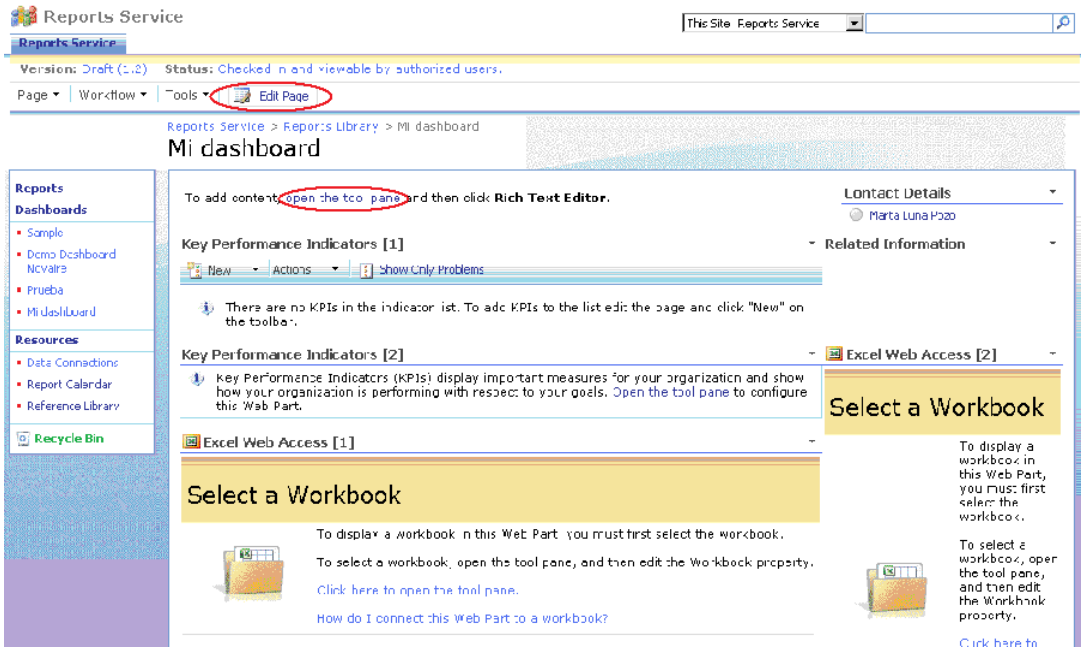


Figura I.8. Formas de editar un dashboard

2. Cuando nos encontremos en modo edición, pinchamos **Add a webpart** y aparece un cuadro de diálogo (Figura I.9.) donde debemos seleccionar el tipo de webpart que queremos añadir, en este caso **Excel Web Access**.

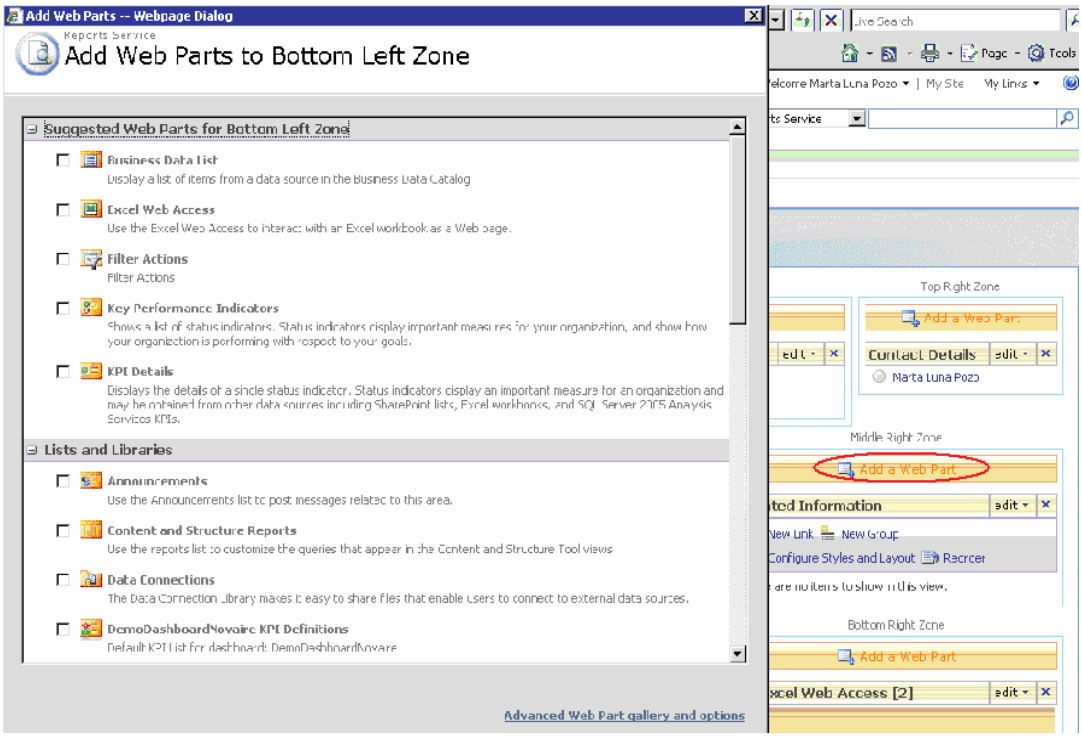


Figura I.9. Cuadro de diálogo de Add a webpart

3. Para modificar el contenido de la nueva webpart existen varias posibilidades (Figura I.10.). Si es la primera vez que vamos a editar la webpart, podemos pinchar 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. En cambio, si la webpart ya ha sido editada, siempre utilizaremos la segunda opción.

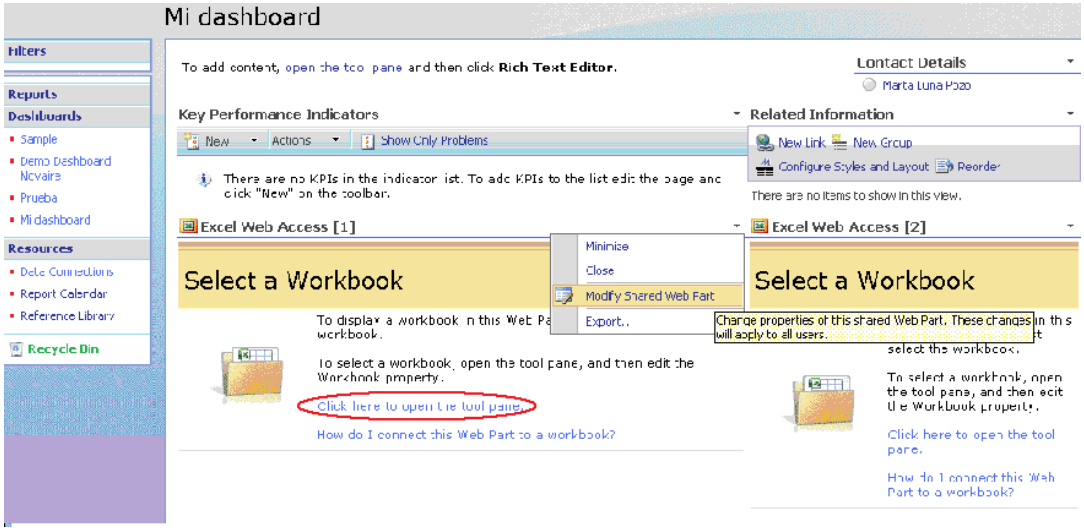


Figura I.10. Formas de modificar una webpart

4. Al seleccionar cualquiera de las dos opciones, aparece un panel a la derecha en el que debemos rellenar toda una serie de campos relacionados con la webpart de Excel. Concretamente, en la Figura I.11. que aparece a continuación están marcados los más relevantes.

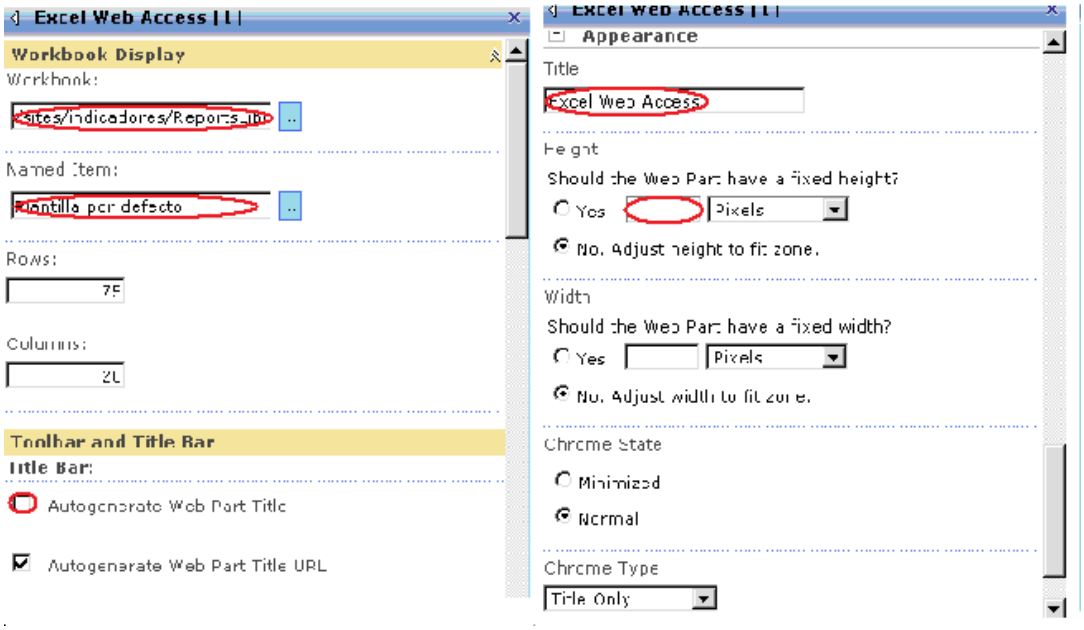


Figura I.11. Campos a rellenar en la configuración de una webpart

5. El atributo **Workbook** es el report al que vamos a hacer referencia en la webpart y el **Named Item** es el elemento que queremos que salga por defecto en la webpart, aunque este último no es necesario ponerlo. Si queremos ponerle un título nosotros a la webpart, es importante desmarcar la opción de autogenerar el título. A la hora de ajustar las dimensiones de la webpart, debemos cambiar los valores de **Height** y **Width** hasta encontrar unos valores que se ajusten a las dimensiones deseadas.

6. Para especificar la situación concreta de la webpart en el dashboard, debemos cambiar el atributo **Zone** e indicar si queremos que la webpart esté arriba, abajo, derecha o izquierda. En el caso que hayamos escogido una distribución en 3 columnas del dashboard, también tendremos la posibilidad de situar las webparts en el centro.

Paso 7: Definir una webpart para mostrar una lista de KPIs

Una vez hayamos definido todas las webparts que referencian archivos Excel, es hora de definir los KPI sobre los reports que tenemos definidos para la empresa utilizando la lista de KPIs que ha sido generada automáticamente al crear el dashboard. Para ello seguimos los siguientes pasos:

1. Para crear un KPI debemos ir a la webpart que contiene la lista de KPIs y pinchar la lista desplegable que aparece junto al menú **New** (Figura I.12.). Tal y como se observa, podemos crear un indicador KPI de varias formas, según si utilizamos KPIs ya definidos en otra plataforma, si utilizamos los datos existentes en un libro Excel o si insertamos la información del KPI manualmente. En nuestro caso, de momento sólo vamos a definir KPI basados en la información contenida en archivos Excel.

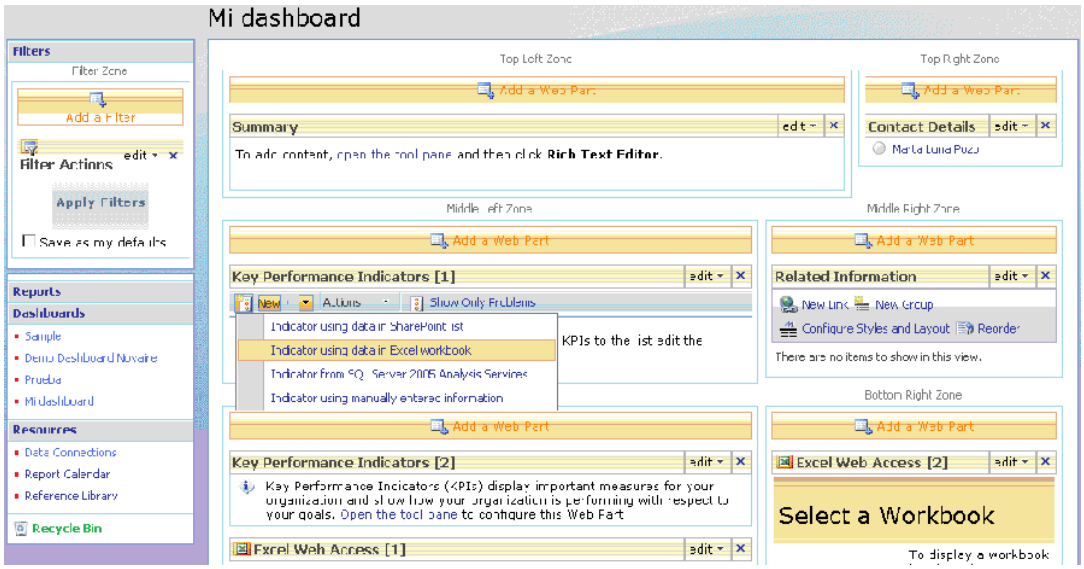


Figura I.12. Creación de un KPI usando un libro Excel

2. Seleccionamos la opción **Indicator using data in Excel workbook** y rellenamos los datos referentes al indicador KPI como son el nombre, una descripción de lo que indica, cómo interpretarlo y el archivo Excel al que referencia (Figura I.13.).

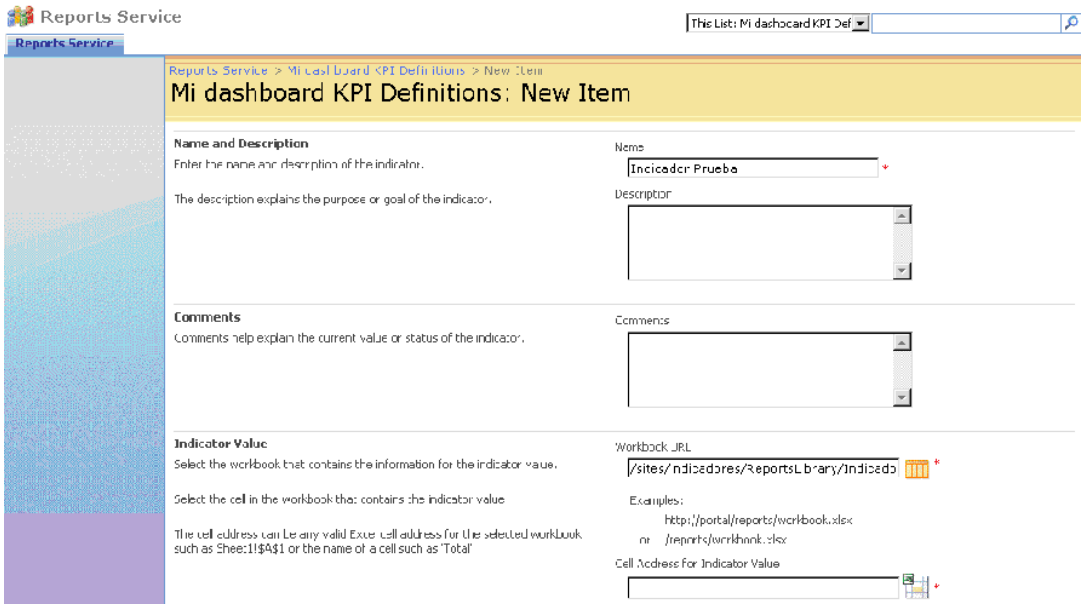


Figura I.13. Propiedades del KPI

3. En el campo que nos pide qué celda indica el valor del indicador, debemos abrir el archivo Excel asociado y seleccionar dicha celda. Además, debemos 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 los podemos introducir manualmente (Figura I.14.) o bien usando las celdas del Excel en caso que estuvieran definidos.

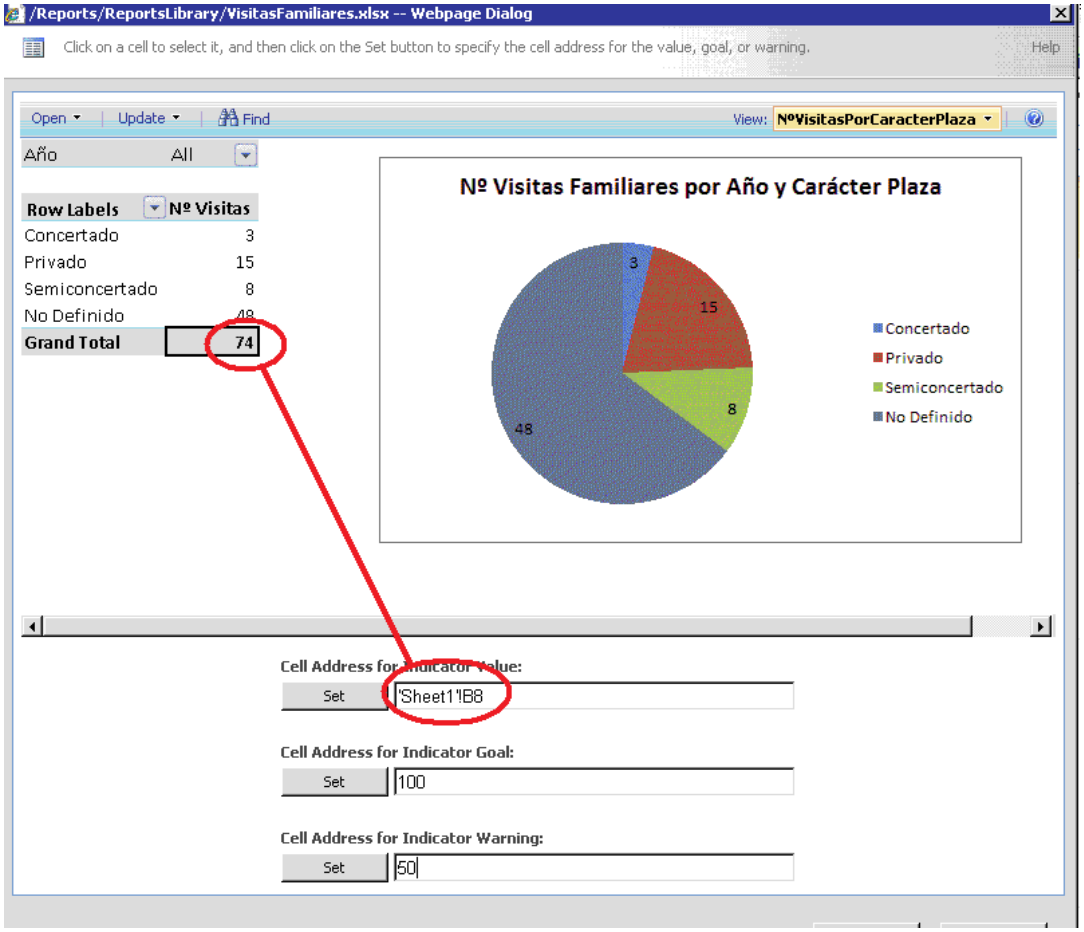


Figura I.14. Ejemplo de umbrales de KPI introducidos manualmente

4. Finalmente, indicamos cómo se interpretan los valores del indicador (Figura I.15.), es decir, si éstos son mejores cuanto más grandes o no y el icono con qué representamos cada uno de los estados del indicador en función de su grado de satisfacción.

Figura I.15. Interpretación de los valores del KPI

5. Definimos de la misma forma el resto de indicadores KPI y cuando ya los hayamos definido todos y no tengamos que añadir más webparts, seleccionamos la opción de guardar y parar la edición de la página situada en la parte superior de la página.

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, debemos seleccionar la

opción **Icons** de la lista desplegable asociada a **Actions** y elegir el tipo de icono deseado tal y como indica la Figura I.16.

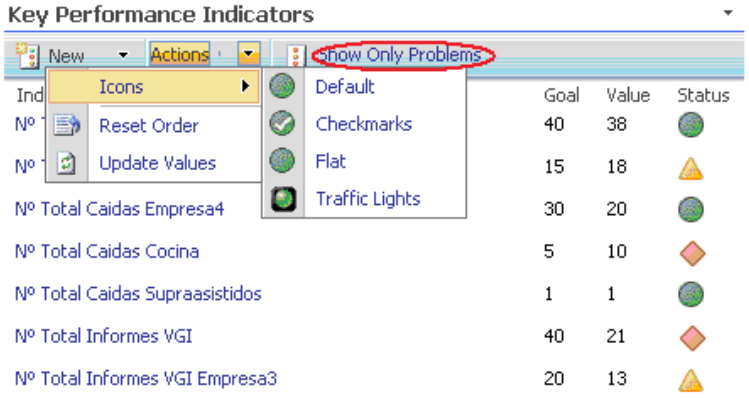


Figura I.16. Selección del tipo de iconos de los KPIs

Además, si lo queremos es mostrar sólo aquellos KPIs cuyo grado de satisfacción no es el deseado (Figura I.17.), tan sólo debemos pinchar la opción de **Show Only Problems**.

Figura I.17. KPIs no satisfactorios

Cuando hayamos acabado de definir todos los indicadores KPI, el proceso de creación del dashboard habrá finalizado.

Anexo J: Desarrollo de pruebas automatizadas en IBM Rational Functional Tester

En este tutorial se van a explicar detalladamente las funciones principales de Rational Functional Tester, introduciendo al usuario en el desarrollo de pruebas automatizadas.

Dado que los documentos de ejemplo mostrados anteriormente hacían referencia al indicador de visitas familiares, ese es el indicador que vamos a utilizar como ejemplo de este tutorial.

A continuación se indican los pasos necesarios para poder automatizar la interfaz de inserción de datos del indicador de visitas familiares.

Paso 1: Crear un proyecto de Rational Functional Tester

En primer lugar, crearemos un nuevo proyecto de RFT y nos familiarizaremos con los componentes básicos del entorno.

Para crear un nuevo proyecto hay que seguir los siguientes pasos:

1. Hacemos clic en **Inicio**, seleccionamos **Todos los programas**, seleccionamos **Microsoft Visual Studio** y, después, hacemos clic en **Microsoft Visual Studio**. De esta forma se abrirá el entorno de desarrollo de Microsoft Visual Studio.
2. Cerramos la ficha **Página de Inicio**. En el menú **Archivo** de Visual Studio, seleccionamos **Nuevo** y hacemos clic en **Proyecto**.
3. En el cuadro de diálogo **Nuevo proyecto**, seleccionamos **Proyectos de Functional Test** en el panel **Tipos de proyecto**, y seleccionamos **Proyecto de Functional Test** en el panel **Plantillas**.
4. Cambiamos el nombre del proyecto, el cúal cambiará también el nombre de la solución, y hacemos clic en **Aceptar**.

En la barra de herramientas, situada en la parte superior, se pueden ver un conjunto de opciones disponibles en el proyecto. La siguiente tabla muestra un listado de las más importantes:

	**Crear un script vacío**	Configura y crea un script en el proyecto sin código.
	**Crear una agrupación de datos de prueba**	Crea una agrupación de datos de prueba en el proyecto.
	**Crear una correlación de objetos de prueba**	Crea una correlacion de objetos de prueba en el proyecto.
	**Grabar un script**	Crea un script y muestra la ventana de grabación.
	**Insertar registro en script activo**	Muestra el cuadro de diálogo de grabación. El código que se genere se ubicará en el script en el que estaba el cursor antes de haber pulsado el botón.
	**Ejecutar script**	Ejecuta el script que se muestra en el editor de código.

	**Depurar script**	Ejecuta el script en modo depuración.
	**Configurar aplicaciones para prueba**	Muestra un cuadro de diálogo para añadir y configurar las aplicaciones de prueba.
	**Abrir inspector de objetos de prueba**	Muestra el cuadro de diálogo del inspector de objetos de prueba, en el que muestra información de los objetos de prueba.
	**Insertar punto de verificación en script activo**	Muestra un cuadro de diálogo para seleccionar un control de la aplicación e insertar un punto de verificación en el script actual.

A continuación vamos a definir brevemente cada uno de los artefactos utilizados por RFT:

Script

Un script es el fichero que contiene los datos y las instrucciones de la interacción con la interfaz, es el que interpreta RFT para su reproducción. Además de ese código, contiene otros ficheros que son privados de cada script, tales como los puntos de verificación, agrupación de datos, correlación de objetos y otra clase, llamada clase ayudante, que el usuario puede personalizar para añadir nuevos metodos o redefinir los existentes.

El método principal de un script es el TestMain(). Al ser reproducido o invocado siempre se ejecutará. Puede recibir un array de objetos como argumentos y devolver un objeto. A continuación se muestra el esqueleto de un script.

```
'Un script siempre hereda de la clase ayudente,
'en este caso ScriptHelper
Public Class Script Inherits ScriptHelper
   'Metodo principal
   Public Function TestMain(ByVal args() As Object) As Object

      'Codigo de la prueba

      Return Nothing
   End Function
End Class
```

Figura J.1. Esqueleto de un script

Desde un script se puede invocar otros scripts. La invocación se hace mediante el método CallScript(), que recibe como parámetro el nombre del script al que se quiere llamar. Como parámetros opcionales, se puede pasar los argumentos y el número de iteraciones de la agrupación de datos que se debe ejecutar. A continuación, se muestran unos ejemplos.

```
'Invoca al script Test1
CallScript("Test1")

'Invoca al script Prueba1 que está en el espacio de
'nombres Pruebas.InsertarUsuarios
CallScript("Pruebas.InsertarUsuarios.Prueba1")
```

```
'Invoca al script Test2 pasandole como parametros la variable args
Dim args(3) As Object
args(0) = "Te paso un doce y la fecha de hoy"
args(1) = 12
args(2) = Date.Now
CallScript("Test2", args)

'Invoca al script Test3 pasandole como parametros args. Se ejecutará
'cuatro veces con distintas registros de la agrupacion de datos
CallScript("Tes3", args, 3)
```

Figura J.2. Ejemplo de invocación de scripts

Desde un script también se puede iniciar programas o páginas web que previamente estén configurados en el proyecto. El siguiente codigo muestra como hacer posible esto.

```
'Inicia la aplicacion ClassicsJava
StartApp("ClassicsJavaA")
'Abre el la pagina www.google.es con el navegador
StartBrowser("www.google.es")
'El script se para durante 5 segundos
Sleep(5)
```

Figura J.3. Ejemplo de iniciar programa y página web

Agrupación de datos de prueba (datapool)

Un datapool es un conjunto de datos de prueba organizado en forma de tabla, es decir, es una colección de datos que suministra valores a las variables en un script durante su reproducción. Sirve para suministrar datos realistas a los test y hacer hincapié en el uso de una aplicación con una cantidad realista de datos.

Cuando se crea un datapool, se puede seleccionar los objetos de la aplicación a testear para hacer la estructura, es decir, para introducir las variables. En el datapool se pueden modificar y añadir datos. Con un solo script, se puede probar el mismo test con diferentes datos.

En la siguiente imagen se puede ver un ejemplo de un datapool. Las columnas, excepto la primera, son las variables que forman la estructura, y las filas son los registros o instancias que dan valor a las variables en una determinada iteración. La primera columna indica el número de iteración.

	IVA	Cuota_Total	DNI_Residente	IncluirFacturaci...	Incluirgenerar...
0	7	1000	01010101	True	True
1	7	1100	02020202	True	True
2	7	1200	03030303	True	True
3	7	1300	04040404	True	True
4	7	1400	05050505	True	True

Figura J.4. Ejemplo de datapool

A continuación se muestra un ejemplo de manipulación de datos de un datapool.

```
'Obtener valores de la agrupacion de datos

'Convierte del valor de la variable DNI_Residente en String
Dim dni As String = DpString("DNI_Residente")
'Convierte del valor de la variable IncluirFacturacion en Boolean
Dim incluirfacturacion As Boolean = DpBoolean("IncluirFacturacion")
'Convierte del valor de la variable que ocupa la posicion 1 en Integer
```

```
Dim cuotatotal As Integer = DpInt(1)


'Iterar sobre la agrupacion de datos

'Vuelve al registro número cero
DpReset()
'Indica si ya quedan más registros en la agrupacion de datos
If DpDone() Then
    'Pasa al siguiente registro
    DpNext()
End If
```

Figura J.5. Ejemplo de manipulación de un datapool

Correlación de objetos de prueba (object map)

Un object map es una lista que contiene los objetos de prueba o controles de la aplicación a probar. En la ventana para poder ver la correlación de objetos, se muestra la jerarquía padre-hijo en forma de árbol de los controles que contiene. Normalmente, las hojas del árbol serán aquellos controles utilizados en el script.

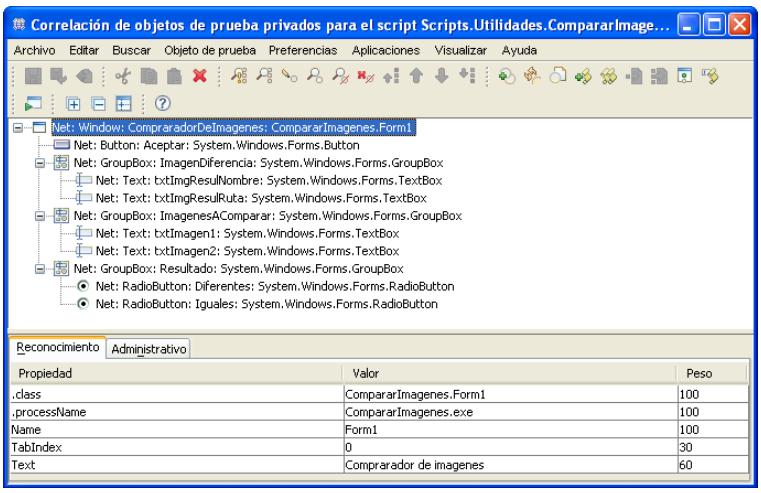


Figura J.6. Contenido de un object map

El principal objetivo de un object map es proporcionar un camino rápido para añadir objetos de prueba a un script, ya que estos contienen las propiedades de reconocimiento del control de la aplicación, y poderlos utilizar en el código del script. Cuando se graba un script, automáticamente se insertan los objetos de prueba en los que se hace alguna acción.

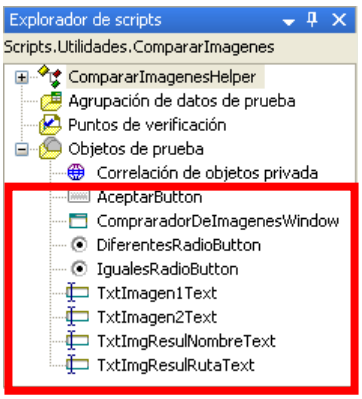


Figura J.7. Objetos de prueba en uso en un script

El reconocimiento de un control se basa en comparar una serie de propiedades como el nombre de la clase, el nombre del proceso, el texto, etc. guardadas en en object map, con el valor de dichas propiedades obtenidas en la aplicación en el momento de la ejecución del script.

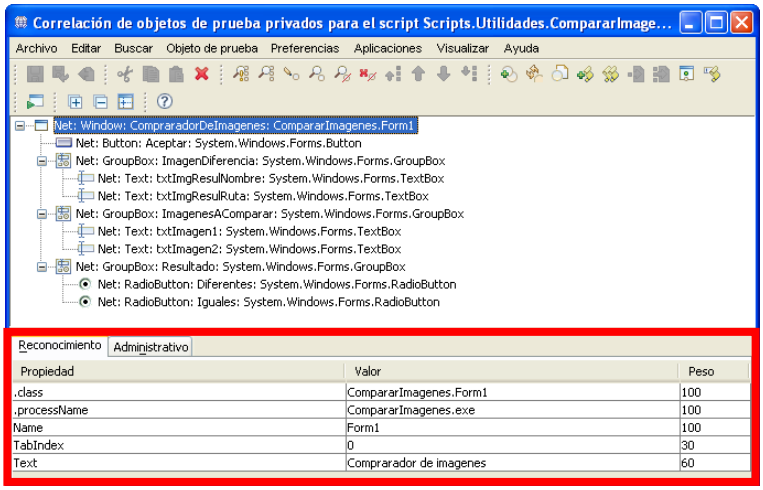


Figura J.8. Propiedades de reconocimiento de un objeto de prueba

Cada propiedad tiene un peso distinto, pudiendo ser un valor del cero al cien. Cuando se ejecuta el script, a la hora del reconocimiento, las propiedades con mayor peso serán más importantes que las que tengan menor peso. El peso de cada propiedad se puede editar. Si el reconocimiento de un objeto es débil, al ejecutar el script, tardará más en hacer la acción que tenga codificada y en el registro saldrá como resultado el mensaje WARN, indicando que control lo ha causado.

A continuación se muestra un ejemplo de uso de un objeto de prueba, tanto para obtener y establecer propiedades en tiempo de ejecución como para invocar eventos.

```
'Objeto de prueba que representa un textbox
Dim MiTextBox As
    Rational.Test.Ft.Object.Interfaces.TextGuiSubitemTestObject = ...

'Hace doble clic en el textbox
MiTextBox.DoubleClick()
'Hace clic en el punto (10,5) del textbox
MiTextBox.Click(AtPoint(10, 5))
'Hace una captura de imagen del control
Dim captura As System.Drawing.Bitmap = MiTextBox.GetScreenSnapshot()
'Obtiene la propiedad Width
Dim anchuraTextbox As Integer =
    CInt(MiTextBox.GetProperty("Width").ToString())
'Ejecuta el metodo ResetText
MiTextBox.Invoke("ResetText")
'Escribe 'Hola' en el textBox
MiTextBox.SetText("Hola")
'Indica si esta Habilitado
Dim estaHabilitado As Boolean = MiTextBox.IsEnabled()
```

Figura J.9. Propiedades de reconocimiento de un objeto de prueba

Puntos de verificación (verification point)

Los puntos de verificación que ofrece RFT tienen una limitación y es que solo sirven para comprobar propiedades de los controles. Así que, si se quiere verificar que ocurren ciertas condiciones que son externas a los controles, hay que usar los puntos de verificación manuales. Los puntos de verificación manuales son puntos de verificación que son programados por el tester utilizando métodos que escriben directamente en el registro.

Para implementar un punto de verificación manual, hay que utilizar el método LogTestResult() para escribir en el registro el resultado de la verificación. El primer parámetro es la cabecera que se mostrará en el registro, el segundo es un booleano para indicar si se ha pasado o no la condición, y un tercero para escribir información adicional. A continuación se muestra un ejemplo de implementación de un punto de verificación manual.

```
Dim a As Integer = ...
Dim b As Integer = ...

'Este punto de verificacion comprueba que el valor de a y de b son iguales
If a = b Then 'Si son iguales
    'entonces ha pasado el punto de verificación
    LogTestResult("a y b son iguales", True)
Else
    'Son distintos, no ha pasado el punto de verificación y escribimos como
    'informacion adicional los valores que toman las variables.
    LogTestResult("a y b son diferentes", False, "Valores: a = " & a & " b ="
& b)
End If
```

Figura J.10. Ejemplos de puntos de verificación manuales

Registro (log)

Después de la reproducción de un script, se puede ver el resultado en el registro. Los resultados incluyen cualquier evento, como puntos de verificación fallados, excepciones de scripts, reconocimientos de objetos débiles y cualquier otra información de la reproducción. Por defecto, el registro se mostrará en formato HTML.

Hay tres tipos de resultados en el registro:

- **PASS:** Ocurre cuando ha finalizado un script con éxito o cuando se ha pasado un punto de verificación.
- **FAIL:** Ocurre cuando ha ocurrido una excepción o error, cuando un punto de verificación no se ha pasado o cuando un script acaba y ha ocurrido un fallo durante su ejecución.
- **WARN:** Ocurre cuando el reconocimiento de un objeto es débil o cuando un script acaba y ha ocurrido un mensaje de aviso durante la ejecución del script.

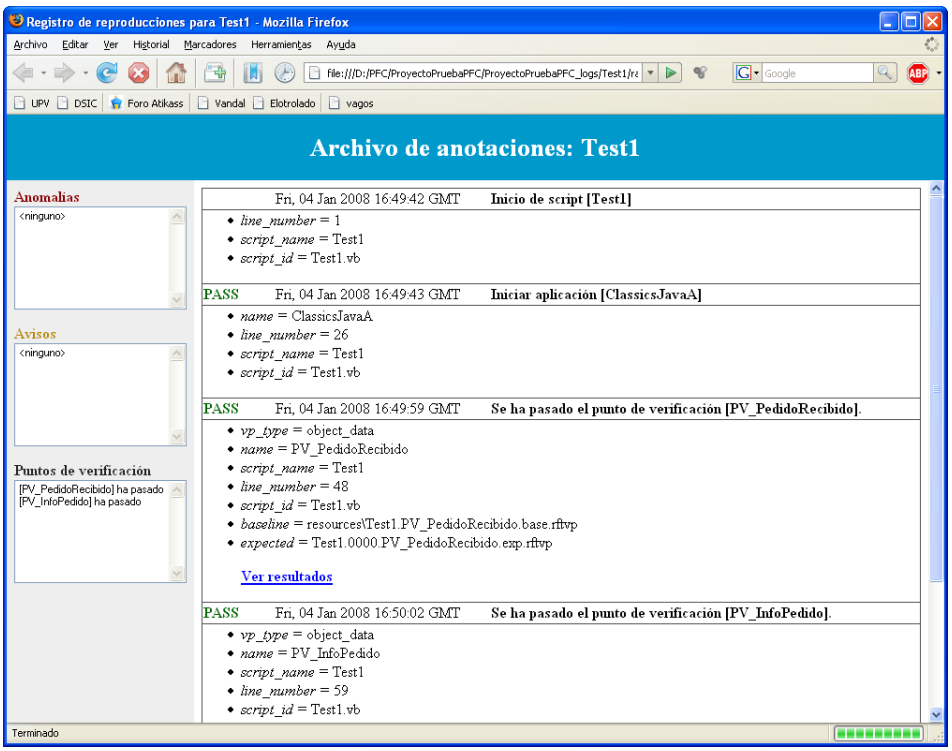


Figura J.11. Registro de RFT en formato HTML

Paso 2: Configuración de aplicaciones para prueba

Para tener disponibles aquellas aplicaciones en las cuales se quieren hacer pruebas, hay que introducirlas en la lista de aplicaciones y configurarlas. De esta forma, pueden ser ejecutadas desde los scripts. Para ello, hay que seguir los siguientes pasos:

1. Ir a **Configurar → Configurar aplicaciones para prueba** o pulsar en la barra de herramientas.
2. Hacer clic en **Añadir…**.

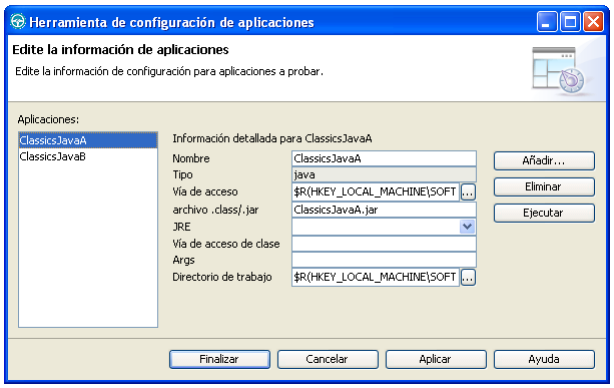


Figura J.12. Pantalla de configuración de aplicaciones (a)

3. Si el programa es un ejecutable, escoger la opción **Ejecutable o archivo de proceso por lotes**.

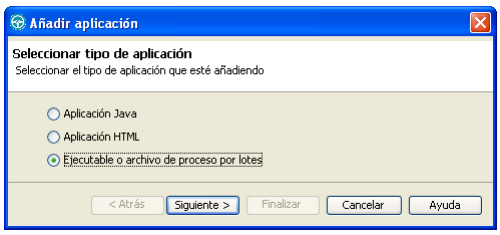


Figura J.13. Eligiendo el tipo de aplicación

4. Escoger la ruta de la aplicación haciendo clic en **Examinar**…. Hacer clic en **Finalizar.**

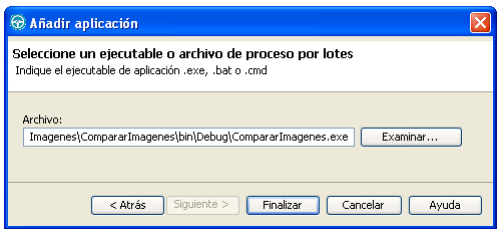


Figura J.14. Indicando la ruta

5. La aplicación aparecerá en la lista de aplicaciones. Si se quiere ejecutar con argumentos, estos se deben escribir en **Args**. Hacer clic en **Aplicar** si se quiere conservar la configuración de la aplicación para otros proyectos. Hacer clic en **Finalizar**.

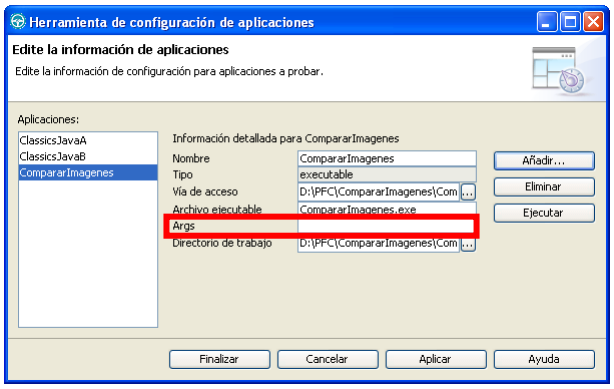


Figura J.15. Pantalla de configuración de aplicaciones (b)

Paso 3: Creación de scripts

Hay dos formas de crear scripts. Una de ellas es creando el script vacío, es decir el método principal del script no tiene código. Y la otra forma es al grabar, el código que se genere en la grabación es el que tendrá el script. A continuación se explicará cómo crearlos.

1. Hacer clic en la carpeta donde se quiere guardar el script en el explorador de soluciones. Para crear un script vacío ir a **Proyecto → Añadir script vacio…** o hacer clic en de la barra de herramientas. Si se quiere crear al grabar, ir a **Proyecto → Añadir script utilizando el grabador…** o pulsar de la barra de herramientas.
2. Aparecerá el siguiente cuadro de dialogo para escribir el nombre del script. Hacer clic en **Agregar**.

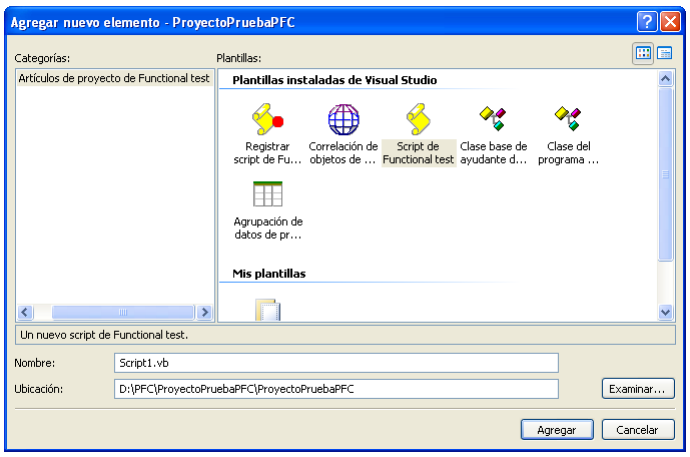


Figura J.16. Pantalla para agregar un nuevo elemento

3. La siguiente ventana permite la configuración de los scripts, en la cual se puede:
   - Elegir que object map y que datapool usar, siendo estos privados o compartidos.
   - Escoger la clase base ayudante, es decir, la clase a partir de la cual hereda el script. Se pueden hacer clases base ayudantes propias para añadir métodos o sobrescribir los existentes.
   - Elegir el orden de selección del registro, el cual indica cómo se recorrerán los registros del datapool. El orden puede ser secuencial o aleatorio.

   Para guardar la configuración, hacer clic en **Finalizar**. Si el script se ha creado al grabar, a continuación aparecerá la pantalla de grabación. El script aparecerá en el explorador de soluciones del proyecto.

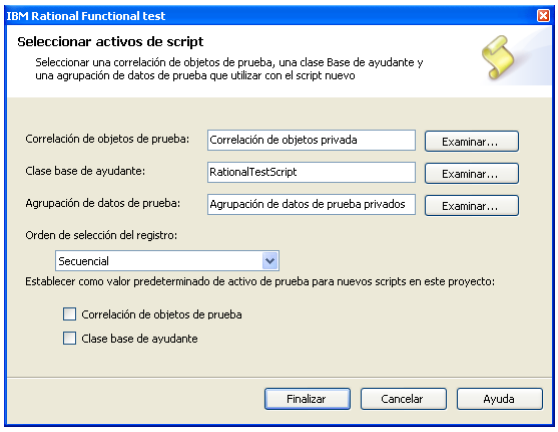


Figura J.17. Pantalla de configuración del script

Paso 4: Grabación de scripts.

La grabación de scripts se puede comenzar mediante el botón de la barra de herramientas o yendo a **Script → Añadir script utilizando el grabador...**, en cuyo caso creará un script y grabara en él. Si se quiere grabar partiendo de un script existente, hay que tener el script abierto en el editor de código, situar el cursor donde se quiera insertar el nuevo código e ir a **Script → Insertar registro** o pulsar el botón de la barra de herramientas.

A continuación se muestra la pantalla de grabación donde se registrarán todas las acciones que se realicen sobre la aplicación de prueba, excepto las acciones que se hagan sobre la propia pantalla. La zona que está dentro del rectángulo rojo es el monitor y muestra el código y los errores que se van generando en la grabación.

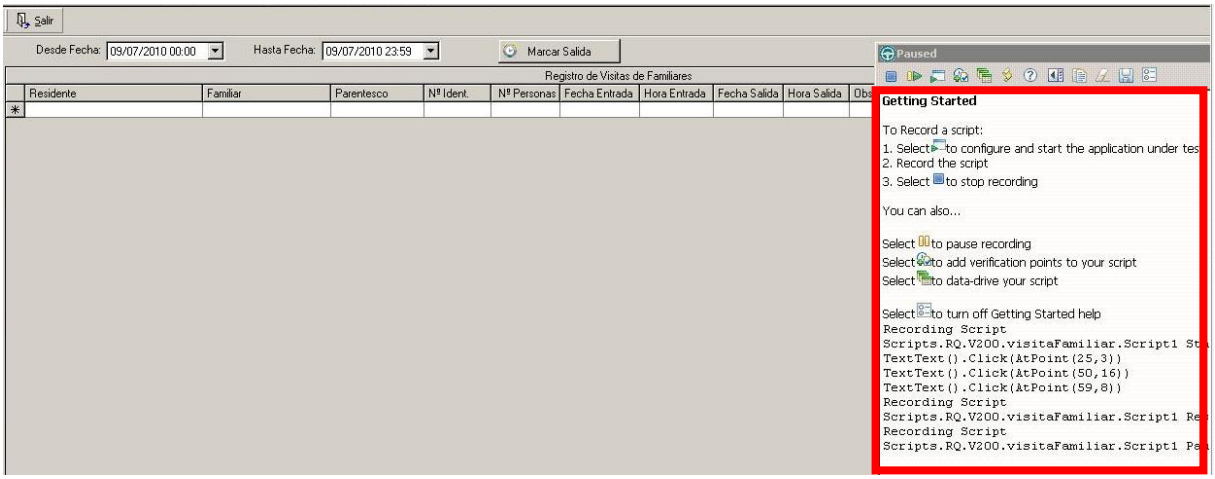


Figura J.18. Pantalla de grabación

La tabla siguiente muestra las opciones más imporantes de la pantalla de grabación.

	**Detener registro**	Cierra la ventana de grabación y para la grabación.
	**Pausar registro**	Pausa la grabación, las acciones que se hagan en la interfaz de la aplicación no se grabarán.
	**Reanudar registro**	Reanuda la grabación.
	**Iniciar aplicación**	Inicia la aplicación seleccionada.
	**Insertar punto de verificación y mandato de acción**	Inserta un punto de verificación.
	**Insertar mandatos controlados por datos**	Inserta variables en la agrupación de datos.
	**Insertar mandatos de soporte de script**	Inserta funciones como llamadas a otros scripts, suspensión del script, insertar comentarios en la entrada de registro, etc.

Paso 5: Crear datapool

Para crear un datapool hay que seguir los siguientes pasos:

1. Pulsar el botón  de la barra de herramientas. o ir a **Script → Crear una agrupación de datos de prueba.**
2. Posteriormente aparecerá un cuadro de dialogo para escribir el nombre del datapool. Hacer clic en **Agregar**.

3. Una vez creado el datapool, debemos añadir las columnas necesarias:
    - DesdeFecha: Es la fecha desde la que queremos visualizar las visitas realizadas en la residencia. En el caso de que esté vacia la columna, Resiplus inserta la fecha actual del sistema.
    - PosiciónVisita: Es la fila del grid de visitas donde se insertará la visita nueva.
    - PosicionResidente: Es la posición del residente que vamos a insertar en el desplegable de la columna Residente del grid de visitas, posteriormente esta columna se queda en desuso por la utilización de la columna Residente.
    - FechaEntrada: Es donde indicamos la fecha en la que se realiza la visita. En el caso de que esté vacia la columna, Resiplus inserta la fecha actual del sistema.
    - Residente: Es el nombre del residente al que se le hace la visita familiar.

4. Por último, añadir las filas que usaremos en las pruebas.


Private Test Datapool

	DesdeFecha	PosicionVisita	PosicionResidente	FechaEntrada	Residente
0		0	76		Sergio Galietero
1		2	31		Pablo Motos
2		2	31		
3		0	58		María Isabel Ma...
4		0	56		
5		1	56		
6		1	59		
7		0	77		
8		0	59		
9		2	59		Victor Morente
10		1	69		10 Joaquin
11		3	31		4 Max
12		0	99		
13		0	67		
14		1	67		
15	22/07/2008	1	67	22/07/2008	Pablo Motos
16	22/07/2008	3	67	22/07/2008	Sergio Galietero
17		8	53		María Isabel Ma...
18		5			Residente 0010
19		7			Residente 0013

Figura J.19. Resultado final del datapool

Paso 6: Modificar scripts.

Una vez creado el script y su correspondiente datapool, el script debe ser modificado:

- Renombrando los objetos de prueba, de forma que puedan ser identificados más facilmente y su reconocimiento sea más fuerte.
- Aceptando datos del datapool.

Paso 7: Ejecutar scripts.

Para ejecutar scripts:

1. Pulsar el botón  de la barra de herramientas o ir a **Script → Ejecutar**. Se ejecutará el script que está actualmente en el editor de código. En caso de quererlo ejecutar en modo depurador ir a **Script → Depurar** o pulsar el botón  .

2. En la pantalla de configuración de reproducción se puede indicar los argumentos que recibe el script para su ejecución y también el número de iteraciones que debe hacer sobre la agrupación de datos.

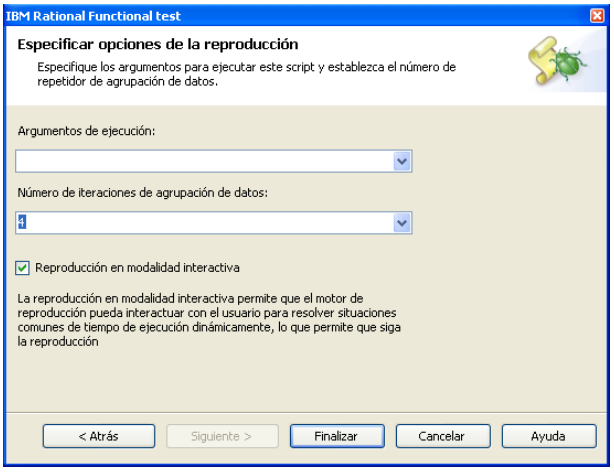


Figura J.20. Pantalla de configuración de opciones de la reproducción

3. Al comenzar la reproducción aparecerá la ventana de reproducción. Desde esta ventana se puede pausar la reproducción haciendo clic en o pulsando la tecla **F12** o parar haciendo clic en o pulsando la tecla **F11**.

   Una vez acabada la reproducción, en el navegador se abrirá el registro que se haya producido indicando el resultado de la prueba. Si se pulsa parar, la prueba fallará.

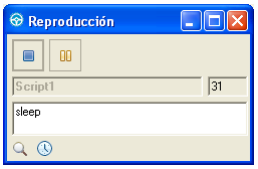


Figura J.21 Pantalla de reproducción