

SolidQ

Microsoft Business Intelligence: vea el cubo medio lleno

Salvador Ramos



ADVERTENCIA LEGAL

Todos los derechos de esta obra están reservados a SolidQ™ Press

El editor prohíbe cualquier tipo de fijación, reproducción, transformación o distribución de esta obra, ya sea mediante venta, alquiler o cualquier otra forma de cesión de uso o comunicación pública de la misma, total o parcialmente, por cualquier sistema o en cualquier soporte, ya sea por fotocopia, medio mecánico o electrónico, incluido el tratamiento informático de la misma, en cualquier lugar del mundo.

La vulneración de cualesquiera de estos derechos podrá ser considerada como una actividad penal tipificada en los artículos 270 y siguientes del Código Penal.

La protección de esta obra se extiende al mundo entero, de acuerdo con las leyes y convenios internacionales.

© SolidQ™ Press, 2011

Microsoft Business Intelligence: vea el cubo medio lleno

Serie Inteligencia de Negocios

Autor: Salvador Ramos

Editado por SolidQ™ Press

Apartado de correos 202

03340 Albufera, Alicante, España

<http://www.solidq.com>

Precio: **Gratuito**

ISBN: **978-84-936417-8-8**

Índice

1. Introducción	7
Business Intelligence	9
Conceptos importantes sobre BI	10
OLTP (Online Transactional Processing)	10
Data Warehouse	11
Data Marts	12
Esquema en estrella (<i>star schema</i>) y en copo de nieve (<i>snowflake schema</i>)	12
ETL (Extract, Transform and Load)	14
Extracción de los datos	16
Transformación de los datos	16
Carga en el Data Warehouse	16
OLAP (Online Analytical Processing)	16
MDX (MultiDimensional eXpressions)	18
KPI (Key Performance Indicator)	18
Componentes de una solución de BI	19
2. Microsoft Business Intelligence	21
Visión y estrategia	21
Cumpliendo con las necesidades de los usuarios	21
Componentes de la plataforma Microsoft BI	21
Herramientas cliente de SQL Server	24
BIDS (Business Intelligence Development Studio)	24
SSMS (SQL Server Management Studio)	25
B.O.L. (Books OnLine, libros en pantalla)	26
Otras herramientas cliente	26
Demos incluidas en el curso	27
3. Modelado dimensional: preparando los datos para su análisis	28
Tablas de Dimensiones (<i>Dimension Tables</i>)	28
SCD (Slowly Changing Dimensions)	29
Tablas de Hechos (<i>Fact Tables</i>)	32
Demos incluidas en el curso	33
4. Integration Services	34

Introducción	34
Demos incluidas en el curso	38
ControlFlow	38
Demos incluidas en el curso	42
Laboratorio incluido en el curso	43
DataFlow	44
Demos incluidas en el curso	48
Laboratorio incluido en el curso	49
Buenas prácticas	50
Recomendaciones para la carga de un <i>staging area</i>	50
Casuísticas habituales	53
Ordenaciones	53
Evitando Joins en el origen	53
Gestión del flujo de datos	54
Transformaciones	55
Actualización de datos	55
SCD (Slowly Changing Dimensions).....	56
Escritura de código VB.NET o C#.....	57
Uso de <i>buffers</i> de datos	57
Conclusiones	57
Demos incluidas en el curso. Carga de tablas de Dimensiones y de tablas Hechos	58
Laboratorio incluido en el curso. Carga de tablas de Dimensiones y de tablas de Hechos.....	59
Logging. Registro de actividad de los paquetes.....	60
Habilitar el Registro (<i>Logging</i>)	60
Selección del tipo de proveedor	61
Personalizando la información de Registro	62
Interactuando con el Registro mediante Scripts	64
Conclusiones	67
Demos incluidas en el curso	67
Configuración, despliegue y administración de paquetes.....	68
Paquete y casuística de ejemplo.....	69

Creación de la configuración del paquete	72
Despliegue del paquete en el servidor	78
Ejecución del paquete en el servidor.....	81
Programar la ejecución de paquetes	83
Conclusiones	87
Demos incluidas en el curso	88
Laboratorio incluido en el curso	88
5. Analysis Services	89
Introducción	90
El cubo. Conceptos básicos	91
Demo incluida en el curso online.....	95
Objetos de una base de datos de Analysis Services	95
Data Sources (DS).....	96
Data Source Views (DSV)	96
Demo incluida en el curso online.....	98
Cubos.....	98
Dimensiones.....	99
Demo incluida en el curso online.....	101
Construyendo (<i>build</i>) y Desplegando (<i>deploy</i>) el modelo	102
Demo incluida en el curso online.....	102
Procesamiento, Modos de Almacenamiento y Agregaciones	103
Procesamiento (<i>Processing</i>).....	103
Modos de Almacenamiento (<i>Storage Modes</i>).....	103
Agregaciones (Aggregations)	104
Personalizando el modelo de Analysis Services	107
Dimensiones.....	107
Atributos	109
Jerarquías y Niveles.....	110
Relaciones entre Atributos	113
Cubos.....	115
Medidas y Grupos de Medidas	115
Propiedades de los Grupos de Medidas	116
Propiedades de las Medidas	116

Otros componentes del cubo	117
Relaciones entre Dimensiones y Grupos de Medidas (Dimension Usage).....	118
Cálculos (Calculations) y KPIs	119
Aún hay más.....	121
Conclusiones sobre Analysis Services	124
Laboratorio incluido en el curso	125
6. Figuras	126

1. Introducción

Este libro sirve como material complementario a los cursos *Business Intelligence End To End Workshop* y *Microsoft BI Bootcamp*, diseñados e impartidos por **SolidQ**, a los que puede asistir tanto de forma presencial como Online.

Se centra en agregar un valor añadido a dichos cursos, junto con el resto de componentes que los forman:

- **Presentaciones PowerPoint**, que sirven de material de apoyo al instructor en las clases presenciales u online.
- **Demostraciones** que permiten al profesor mostrar al alumno de forma práctica y detallada los conocimientos teóricos adquiridos.
- **Laboratorios** con guías paso a paso y basados en buenas prácticas, para que el alumno ponga en práctica los conocimientos adquiridos en cada módulo, y pueda consultar al profesor cualquier duda surgida en su experiencia con el producto.
- **Batería de preguntas** que deberá responder el alumno, ideadas para ayudar a confirmar los conocimientos adquiridos, y servir como base para repasar con el profesor las dudas que hayan surgido al responderlas.
- **Foros** para plasmar las dudas por escrito en el momento en que surjan, para que sean respondidas por el profesor y comentadas por el resto de compañeros.
- **Consultas por email** para enviar dudas privadas que el alumno considere que no debe compartir con el resto de los compañeros. Por ejemplo, cuando afectan a casos prácticos con detalles de su negocio.
- **Tutorías** personalizadas, tanto para resolver dudas surgidas al estudiar el material, como para asesorar al alumno sobre casos reales que necesite resolver en su día a día.

Este libro también se puede leer de forma independiente, sin necesidad de atender al curso para el cuál fue diseñado, y ayudará al lector a entender las herramientas que ofrece Microsoft para el desarrollo de soluciones de Business Intelligence. El lector encontrará a lo largo de él múltiples referencias al material adicional del curso que, como hemos comentado anteriormente, ayudará a dar un enorme valor añadido a este contenido, aunque no será imprescindible su revisión para poder continuar con su lectura, ni dificultará en absoluto la comprensión de lo explicado.

El libro comienza exponiendo una base conceptual sobre Business Intelligence, para a continuación mostrar las herramientas que nos ofrece la plataforma de Microsoft, y finalmente adentrarnos con mayor profundidad en dos de los tres componentes que forman parte de SQL Server y que son el núcleo para la creación de una solución de Business Intelligence basada en esta plataforma: *Integration Services* como herramienta ETL, *Analysis Services* como herramienta OLAP y de minería de datos.

Dejaremos para un segundo ejemplar todo lo relacionado con la visualización de los datos. Lo que incluye a *Reporting Services* como gestor empresarial de informes, *Excel* como herramienta de visualización por Excelencia, y la distribución de contenido y colaboración basada en *SharePoint*, y principalmente en dos de sus servicios, *Excel Services* y *Performance Point Services*.

Business Intelligence

El término “Inteligencia de Negocio” (*Business Intelligence*), conocido habitualmente como “BI”, es muy utilizado hoy en día; lo utilizan tanto expertos, como las propias compañías que desarrollan productos de esta área. Hay una gran cantidad de definiciones, y tampoco es el objetivo de este capítulo hacer una definición exhaustiva del término, sino que va orientado principalmente a transmitir este concepto. Por ello comenzaremos con una de las definiciones que, desde mi punto de vista, mejor lo expresa y sintetiza:

“Es el conjunto de estrategias y tecnologías que nos van a ayudar a convertir los datos en información de calidad, y dicha información en conocimiento que nos permita una toma de decisiones más acertadas y nos ayude así a mejorar nuestra competitividad”

Si el lector lo prefiere, puede buscar, en internet y en otros libros, más información sobre este término, y sobre aplicaciones y casos prácticos que le ayuden a mejorar su entendimiento.

Un sistema de BI, es aquel en el que tenemos centralizados los datos de la empresa, procedentes de diversas aplicaciones, bases de datos y archivos (esto incluye archivos de texto, libros de Excel, o cualquier otro archivo que contenga datos relevantes para el negocio). Estos datos no están tal y como se obtienen de los diversos orígenes, sino que han sufrido una serie de procesos de transformación y limpieza, tienen una mayor calidad; están elaborados y dispuestos para responder a las preguntas de negocio que les van a realizar los usuarios de una forma eficaz, rápida y certera, evitando que haya más de una versión de la verdad.

Cada día es más importante para las empresas el hecho de disponer de una información de calidad, y les ayuda enormemente a obtener ventajas competitivas e identificar riesgos. Desde hace tiempo, ya no es suficiente con tener una inmensa cantidad de datos almacenados en nuestros sistemas, que crecen además de forma exponencial. Esos datos de nada nos valen si no somos capaces de comprender su significado, de elaborarlos y transformarlos en información de calidad, que sea capaz de responder a las preguntas de los usuarios de negocio como las siguientes:

- ¿Cuál es el estado de salud de mi empresa?
- ¿Cuál es el nivel de satisfacción de mis clientes? ¿y el de mis empleados? ¿Ha aumentado el grado de satisfacción con respecto al año anterior?
- ¿Cuál es la línea de productos más rentable? ¿Es la misma que el año anterior?
- ¿Cuál es el segmento de clientes al que deberíamos dirigir un nuevo producto?
- ¿Qué departamentos son los más productivos?

Y un largo etcétera, una gran cantidad de preguntas que nos podemos ir haciendo, y a las que debe responder nuestro sistema de BI.

Otro tema importante a tener en cuenta, son los usuarios de negocio que van a utilizar el nuevo sistema de BI, los cuales van a interrogarlo continuamente mediante una serie de aplicaciones, en las cuales deben formarse. Es muy importante decantarse por aplicaciones en las que podamos obtener una buena curva de aprendizaje, y hacer que este cambio sea lo menos traumático posible.

- Consideramos que las aplicaciones que integran un sistema de BI deben cubrir las siguientes actividades:
- Soporte a la toma de decisiones
- Consultas e informes
- Online Analytical Processing (OLAP)
- Análisis estadístico
- Predicciones y pronósticos
- Minería de datos

Conceptos importantes sobre BI

A continuación, y antes de seguir introduciéndonos en este apasionante mundo de la Inteligencia de Negocio, vamos a introducir una serie de conceptos básicos que debemos conocer, y que se irán utilizando y se irá profundizando en ellos a medida que vaya progresando nuestro conocimiento en este área. La finalidad de este punto es que el lector empiece a familiarizarse con dichos conceptos, y que cuando más adelante se vayan utilizando entienda mejor lo que se irá exponiendo.

OLTP (Online Transactional Processing)

Los sistemas OLTP están diseñados para gestionar un gran número de peticiones concurrentes sobre sus bases de datos, y que los usuarios puedan insertar, modificar, borrar y consultar dichos datos. Están enfocados a que cada operación (*transacción*) trabaje con pequeñas cantidades de filas, y a que ofrezcan una respuesta rápida. Habitualmente utilizan sistemas de bases de datos relacionales para gestionar los datos, y suelen estar altamente normalizados. En ellos es muy importante la integridad de los datos, y deben cumplir las propiedades ACID (Atomicity, Consistency, Isolation, Durability):

- **Atomicidad:** una operación, o se realiza por completo o no se realiza, nunca debe quedar a medias.
- **Consistencia:** sólo se ejecutan las operaciones que cumplen las reglas de integridad de la base de datos.
- **Aislamiento** (*Isolation*): una operación no puede afectar a otras, dos transacciones sobre los mismos datos son independientes y no generan errores entre sí.

- **Durabilidad:** una vez realizada una operación, ésta es persistente y no se puede deshacer.

Si el lector no tiene claro el concepto de **Transacción** le recomiendo que busque información adicional sobre este término que resulta imprescindible para la comprensión de este punto, y para el desarrollo de procesos ETL con origen o destino en sistemas OLTP.

Como ejemplo de este tipo de sistemas, podemos citar un ERP, un CRM, aplicaciones departamentales, aplicaciones de ventas, de comercio electrónico, y un largo etcétera. En general cualquier aplicación con la que el usuario interactúa para introducir datos operacionales al sistema.

Data Warehouse

Un Data Warehouse es una base de datos corporativa en la que se integra información depurada de las diversas fuentes que hay en la organización. Dicha información debe ser homogénea y fiable, se almacena de forma que permita su análisis desde muy diversas perspectivas, y que a su vez dé unos tiempos de respuesta óptimos. Para ello la información se encuentra altamente desnormalizada y modelada de una forma bastante diferente a los sistemas transaccionales, principalmente se utilizan los modelos en estrella (*star schema*) y en copo de nieve (*snowflake schema*) que estudiaremos más adelante.

Un Data Warehouse es mucho más que lo que hemos comentado hasta el momento. Según Bill Inmon se caracteriza por ser:

- **Orientado a temas:** los datos están organizados por temas para facilitar el entendimiento por parte de los usuarios, de forma que todos los datos relativos a un mismo elemento de la vida real queden unidos entre sí. Por ejemplo, todos los datos de un cliente pueden estar consolidados en una misma tabla, todos los datos de los productos en otra, y así sucesivamente.
- **Integrado:** los datos se deben integrar en una estructura consistente, debiendo eliminarse las inconsistencias existentes entre los diversos sistemas operacionales. La información se estructura en diversos niveles de detalle para adecuarse a las necesidades de consulta de los usuarios. Algunas de las inconsistencias más comunes que nos solemos encontrar son: en nomenclatura, en unidades de medida, en formatos de fechas, múltiples tablas con información similar (por ejemplo, hay varias aplicaciones con tablas de clientes).
- **Histórico (variante en el tiempo):** los datos, que pueden ir variando a lo largo del tiempo, deben quedar reflejados de forma que al ser consultados reflejen estos cambios y no se altere la realidad que había en el momento en que se almacenaron, evitando así la problemática que ocurre en los sistemas

operacionales, que reflejan solamente el estado de la actividad de negocio presente. Un Data Warehouse debe almacenar los diferentes valores que toma una variable a lo largo del tiempo. Por ejemplo si un cliente ha vivido en tres ciudades diferentes, debe almacenar el periodo que vivió en cada una de ellas y asociar los hechos (ventas, devoluciones, incidencias, etc.) que se produjeron en cada momento a la ciudad en la que vivía cuando se produjeron, y no asociar todos los hechos históricos a la ciudad en la que vive actualmente.

- **No volátil:** la información de un Data Warehouse, una vez introducida, debe ser de sólo lectura, nunca se modifica ni se elimina, y ha de ser permanente y mantenerse para futuras consultas.

Adicionalmente estos almacenes contienen metadatos (datos sobre los datos), que aportan un valor adicional, permitiendo tener información sobre su procedencia (sobre todo cuando tenemos múltiples fuentes), la periodicidad con la que han sido introducidos, la fiabilidad que nos ofrecen, etc. Todo ello nos aporta una ayuda adicional, tanto al usuario final como a los técnicos responsables de su mantenimiento, ayudando a estos últimos en aspectos como su auditoría y administración.

Data Marts

La diferencia de un Data Mart con respecto a un Data Warehouse es solamente en cuanto al alcance. Mientras que un Data Warehouse es un sistema centralizado con datos globales de la empresa y de todos sus procesos operacionales, un Data Mart es un subconjunto temático de datos, orientado a un proceso o un área de negocio específica. Debe tener una estructura óptima desde todas las perspectivas que afecten a los procesos de dicha área. Es más, según *Ralph Kimball*, cada Data Mart debe estar orientado a un proceso determinado dentro de la organización, por ejemplo, a pedidos de clientes, a compras, a inventario de almacén, a envío de materiales, etc. Para Ralph Kimball el conjunto de Data Marts forma el Data Warehouse.

Esquema en estrella (*star schema*) y en copo de nieve (*snowflake schema*)

A la hora modelar el Data Mart o Data Warehouse, hay que decidir cuál es el esquema más apropiado para obtener los resultados que queremos conseguir. Habitualmente, y salvo excepciones, se suele modelar la base de datos utilizando el **esquema en estrella (*star schema*)**, en el que hay una única tabla central, la tabla de hechos, que contiene todas las medidas y una tabla adicional por cada una de las perspectivas desde las que queremos analizar dicha información, es decir por cada una de las dimensiones (ver imagen):

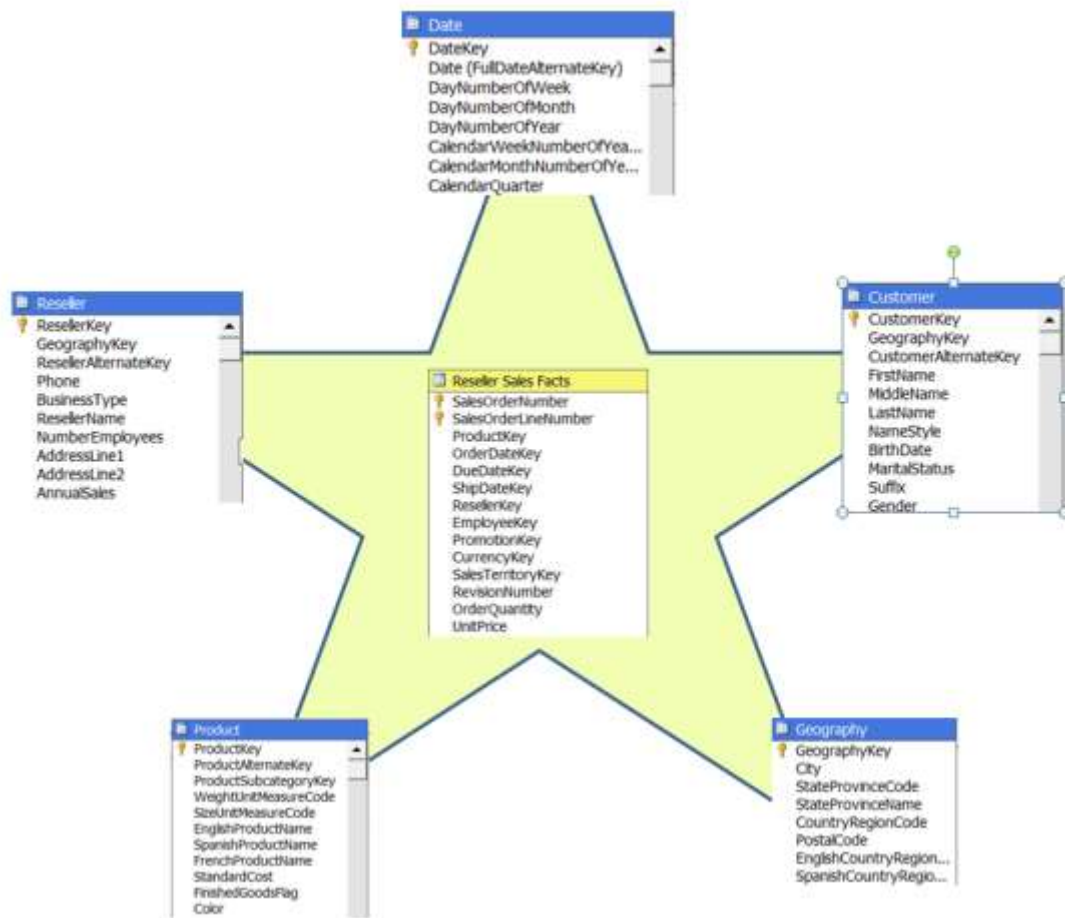


Figura 1-1 Esquema en estrella (star schema)

La otra alternativa de modelado es la utilización del **esquema en copo de nieve (snowflake schema)**. Esta es una estructura más compleja que el esquema en estrella. La diferencia es que algunas de las dimensiones no están relacionadas directamente con la tabla de hechos, sino que se relacionan con ella a través de otras dimensiones. En este caso también tenemos una tabla de hechos, situada en el centro, que contiene todas las medidas y una o varias tablas adicionales, con un mayor nivel de normalización (ver imagen):

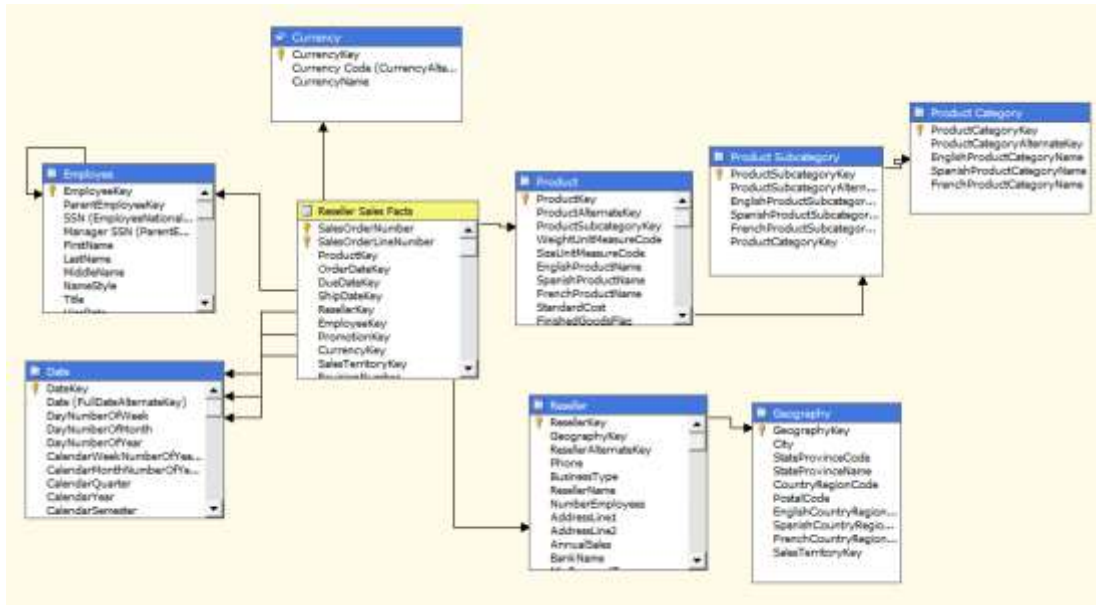


Figura 1-2 Esquema de copo de nieve (snowflake schema)

El modelo en estrella, aunque ocupa más espacio en disco (dato cada vez menos significativo), es más simple de entender por el usuario y ofrece un mejor rendimiento a la hora de ser consultado.

ETL (Extract, Transform and Load)

Un Data Warehouse, o un Data Mart, se cargan periódicamente, y en él se unifica información procedente de múltiples fuentes, creando una base de datos que cumple una serie de características descritas anteriormente. Esto implica que deben existir una serie de procesos que leen los datos de las diferentes fuentes, los transforman y adaptan al modelo que hayamos definido, los depuran y limpian, y los introducen en esta base de datos de destino. Esto es lo que se conoce como **procesos ETL**, procesos de Extracción, Transformación y Carga (*Load*).

Es muy importante diseñar un buen proceso ETL, en él se deben reconciliar todos los datos de las diferentes fuentes, realizar los cálculos necesarios, mejorar la calidad de los datos, y por supuesto, adaptarlos al nuevo modelo físico y almacenarlos en él.

En muchas ocasiones la información no pasa directamente de las fuentes al Data Mart o Data Warehouse, sino que lo hace a través de unas bases de datos intermedias, que son necesarias en muchas ocasiones dada la complejidad y disparidad de las fuentes. Habitualmente, los datos, antes de entrar en el Data Mart o Data Warehouse, se almacenan en un *área de staging* y/o un *ODS (Operational Data Store)*.

Un **área de staging** es un área temporal que se encuentra en el flujo de datos entre las fuentes y el Data Mart o Data Warehouse con el fin de facilitar la extracción de

datos, de realizar tareas de limpieza (*data cleansing*), de mejorar la calidad de los datos, e incluso de ser utilizada como caché de los datos operacionales o acceder a un nivel de detalle de los datos y de los cambios no almacenados en el Data Mart o Data Warehouse.

Un almacén operacional de datos (**ODS** *Operational Data Store*) es un área que va a dar soporte a los sistemas transaccionales, desde los que se alimenta con una periodicidad muy baja, y sirve como base de datos de consulta, a la que se conectan herramientas de *reporting* con el fin de que el sistema transaccional tenga una menor carga de trabajo. Se encuentra normalizado, y no es algo específico de un sistema de BI, puede existir o no, y es independiente de que exista o no un Data Mart o un Data Warehouse. Pero si existe, puede ser más apropiado obtener de él la información para alimentar el Data Mart o Data Warehouse que hacerlo desde el sistema transaccional, para así también quitar estas lecturas al sistema transaccional.

Las transformaciones suelen tener un cierto grado de complejidad, dado que los datos necesitan agregarse, analizarse, calcularse, procesarse estadísticamente, limpiarse, aumentar su calidad, etc.

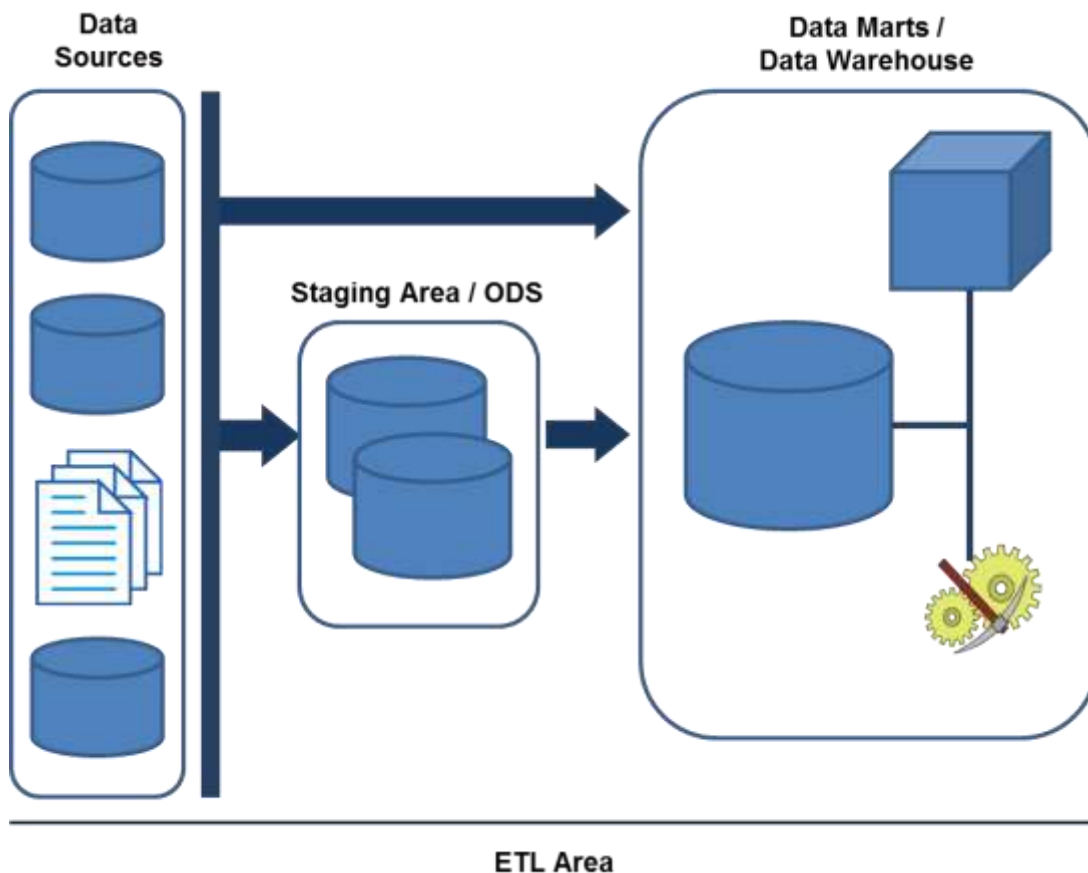


Figura 1-3 ETL

Como hemos comentado anteriormente, e incluso por las siglas, las operaciones ETL se pueden dividir en tres etapas:

Extracción de los datos

Uno de los aspectos fundamentales que debemos considerar a la hora de diseñar es si optamos por una actualización completa, que es mucho más simple, o si optamos por una actualización incremental, que es lo más conveniente. Si optamos por la primera alternativa el proceso consiste en borrar los datos y volver a cargarlos, pero si optamos por la segunda, habrá establecer una serie de controles y técnicas, entre las que destacamos las siguientes:

- Borrado sólo de parte de los datos, por ejemplo desde una fecha, y carga de los nuevos datos.
- Comparar los datos de origen y de destino, actualizando sólo los cambios.
- Uso de *triggers* u otras técnicas de replicación.
- Captura de datos mediante una aplicación diseñada específicamente para ello.
- Existencia de columnas *timestamp* que permitan seleccionar las filas modificadas desde la última extracción y aplicar dichos cambios en destino.

Lo habitual es que en nuestros procesos combinemos varias de estas técnicas, según la casuística que tengamos y diseño del proceso que hayamos realizado en cada caso.

Transformación de los datos

En los procesos de transformación, es preciso asegurarnos de que los datos sean válidos, de su integridad y de su utilidad, lo que suele incluir realizar cálculos y generar nuevos valores. Los datos deben ser depurados para eliminar inconsistencias, discrepancias y duplicidades. Estas transformaciones suelen conllevar cambios con respecto a la estructura de origen para adaptarla al destino, cambios en el contenido de los valores de origen y creación de nuevos valores en las filas de destino.

Carga en el Data Warehouse

Como último paso debemos realizar el proceso de incorporar los datos al Data Warehouse y/o a los diferentes Data Marts, y a los cubos OLAP. Todo ello según la presentación, formatos definidos y periodicidad de actualización propuesta.

OLAP (Online Analytical Processing)

El Procesamiento analítico en línea (**OLAP**) tiene como objetivo agilizar la consulta de grandes volúmenes de información. Para ello utiliza estructuras

multidimensionales, conocidas como cubos OLAP, que contienen datos pre calculados y agregados. Estos sistemas tienen una velocidad de respuesta muy superior a los sistemas OLTP.

Un cubo OLAP es un vector multidimensional, de N dimensiones, aunque por su nombre puede hacernos creer inicialmente que sólo tiene tres dimensiones. En él la información se almacena en cada una de estas dimensiones, de forma ordenada y jerarquizada, lo que nos ayuda a realizar un rápido análisis de su contenido. Una base de datos multidimensional puede contener varios de estos cubos OLAP.

Los usuarios piensan de forma multidimensional, queriendo analizar la información desde diferentes perspectivas (dimensiones), haciéndose preguntas como las siguientes:

- ¿Cuáles son las ventas actuales, comparadas con las del mismo periodo del año anterior? Quiero esta información desglosada por zona, por cliente, por vendedor, y por familia de producto.
- ¿Cuál es nuestra rentabilidad por cliente? ¿y por producto?
- ¿Cuáles son los pedidos pendientes por cliente, por tiempo y por producto?

Y por, y por ... Cada uno de estos “y por ...” sería una dimensión, mientras que las unidades, cantidades, importes, beneficios de ... ventas, compras, pedidos... serían las medidas.

Por tanto, un cubo OLAP está estructurado en **dimensiones**, que son las diferentes perspectivas desde las que queremos analizar la información, y en **medidas**, que son los diferentes hechos con valores concretos que solicita el usuario.

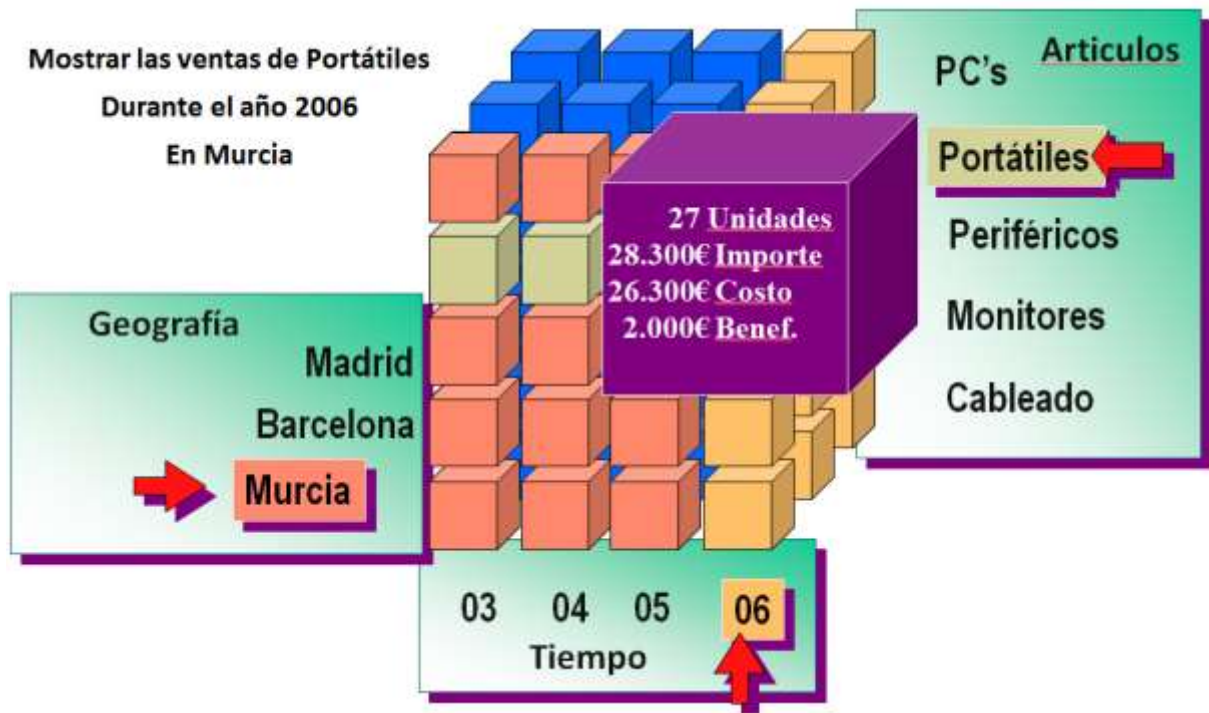


Figura 1-4 Cubos OLAP

Tanto cuando nos refiramos a bases de datos multidimensionales, como a OLAP, o al término cubos, o cubos OLAP, nos estamos refiriendo al mismo concepto explicado anteriormente.

MDX (MultiDimensional eXpressions)

El MDX es un lenguaje de consulta creado específicamente para hacer consultas sobre cubos OLAP, ofreciendo una sintaxis especializada para manipularlos. Al igual que utilizamos el lenguaje SQL para acceder a bases de datos relacionales, utilizaremos el MDX para acceder a bases de datos multidimensionales.

KPI (Key Performance Indicator)

Los indicadores clave de rendimiento, o KPI por sus siglas en inglés, nos permiten ir un paso más allá de ver un simple valor, pudiendo con ellos contextualizarlo. Veamos un ejemplo, si muestro unas ventas de 7 millones realizadas en el último trimestre es simplemente un dato, pero ¿realmente es un dato bueno, malo, regular ... con respecto a qué?. En cambio si decimos que nuestras ventas son de 7 millones y van regular con respecto a un objetivo de 8 millones que nos habíamos planteado para este trimestre, y que además tienen una tendencia a la baja con respecto al mismo trimestre del año anterior, estamos poniendo dicho valor dentro de un contexto.

- Fuentes de datos: serán nuestros diversos sistemas OLTP, ficheros Excel, ficheros planos, etc.
- Un Data Warehouse y/o diversos Data Marts: son las bases de datos ya modeladas de forma multidimensional que se alimentan periódicamente mediante procesos ETL.
- Sistemas OLAP y de minería de datos: que nos aportan una nueva fuente de información que amplía la potencia de nuestros sistemas de BI.
- Herramientas analíticas y de presentación: que permiten al usuario de negocio poder acceder a la información, compartirla y analizarla.

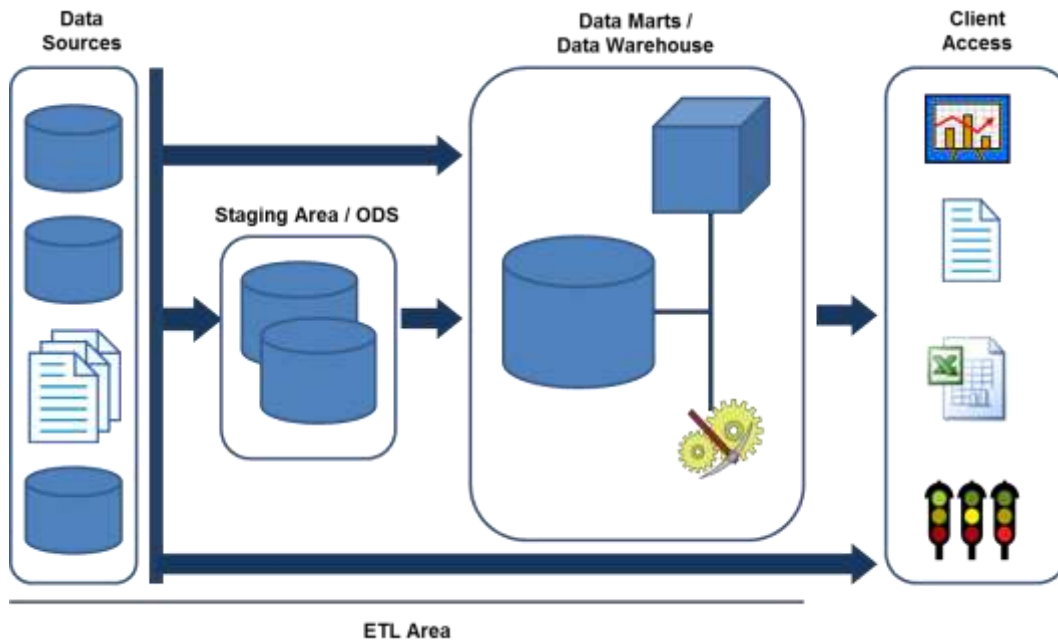


Figura 1-6 Solución de BI

2. Microsoft Business Intelligence

Una vez que nos hemos introducido en el mundo del BI a nivel general, vamos a centrarnos a partir de este momento en la plataforma que nos ofrece Microsoft para el desarrollo de soluciones de BI.

Visión y estrategia

Seguro que muchas veces has escuchado el término “democratización de BI”, utilizado por muchos fabricantes y gurús del Business Intelligence. Incluso algunos de ellos, incluido Microsoft, basan su apuesta en este mercado alrededor de este término, que probablemente sea la raíz del vocablo Gestión del Rendimiento, o al menos eso postula Howard Dresner, a quien se le atribuye la “creación” del término Business Intelligence. Pero, ¿Qué es realmente la Democratización de BI en el mundo Microsoft? Básicamente *se trata de acercar la potencia y funcionalidad de las soluciones de Inteligencia de Negocio al usuario final para así poder llegar a todos los usuarios de la organización*. Obviamente ahí Microsoft lleva ventaja, puesto que dispone de la suite ofimática más implantada del mercado, Microsoft Office.

Tal y como comentábamos al comienzo, la apuesta de Microsoft en este entorno es muy clara. Ofrecer una buena plataforma a nivel de servidor, basada en SQL Server y SharePoint, y aprovechar toda su base instalada de herramientas de productividad personal para incorporarles capacidades de Inteligencia de Negocio, haciendo que la curva de aprendizaje de los usuarios en el momento de acercarse a este mundo se reduzca drásticamente. Y hacer que todo esto tenga un coste muy competitivo, que facilite su extensión y utilización en toda la organización.

Cumpliendo con las necesidades de los usuarios

En cualquier solución de BI hay una serie de necesidades que tienen los usuarios que deben ser cubiertas. Por un lado necesitan una fuente de información única, consolidada y fiable, que es el Data Warehouse. Pero éste sin una serie de herramientas que permitan al usuario explotar dicha información, de nada sirve. Es por tanto una necesidad el que el usuario disponga de una serie de herramientas para realizar informes, analizar y compartir la información. Y en base a ellas podrá llegar a construir una solución de gestión del rendimiento (*Corporate Performance Management*), que le permita hacer un seguimiento medible del negocio, basándose en metodologías, métricas, procesos y sistemas necesarios para monitorizar.

Componentes de la plataforma Microsoft BI

Veamos ahora con mayor nivel de detalle qué herramientas nos ofrece Microsoft a la hora de construir una solución de BI, y qué componentes tiene cada una de ellas.

Comencemos por las herramientas que forman el núcleo y la base de la plataforma de BI, y que vienen todas ellas como parte de **SQL Server**:

- **Database Engine**: es el servicio principal para almacenar, procesar y proteger datos. El Database Engine (Motor de base de datos) proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa. Lo utilizaremos para crear y mantener las bases de datos relacionales.
- **Integration Services (SSIS)**: es una plataforma para la creación de soluciones empresariales de transformaciones de datos e integración de datos. Integration Services sirve para resolver complejos problemas empresariales mediante la copia o descarga de archivos, el envío de mensajes de correo electrónico como respuesta a eventos, la actualización de almacenamientos de datos, la limpieza y minería de datos, y la administración de objetos y datos de SQL Server
- **Analysis Services (SSAS)**: ofrece funciones de procesamiento analítico en línea (OLAP) y minería de datos para aplicaciones de Business Intelligence. Analysis Services admite OLAP y permite diseñar, crear y administrar estructuras multidimensionales que contienen datos agregados desde otros orígenes de datos, como bases de datos relacionales. En el caso de las aplicaciones de minería de datos, Analysis Services permite diseñar, crear y visualizar modelos de minería de datos que se construyen a partir de otros orígenes de datos mediante el uso de una gran variedad de algoritmos de minería de datos estándar del sector
- **Reporting Services (SSRS)**: es una plataforma de creación de informes basada en servidor que ofrece una completa funcionalidad de creación de informes para una gran variedad de orígenes de datos. Reporting Services contiene un completo conjunto de herramientas para crear, administrar y entregar informes, así como interfaces de programación de aplicaciones con las que los desarrolladores podrán integrar o extender el procesamiento de los datos y los informes en aplicaciones personalizadas. Las herramientas de Reporting Services trabajan en el entorno de Microsoft Visual Studio y están totalmente integradas con las herramientas y los componentes de SQL Server.

A continuación, y como herramienta cliente por excelencia, que nos va a permitir realizar una gran variedad de consultas al sistema, tenemos **Microsoft Excel**. Pero ésta es un arma de doble filo, ya que es una herramienta de la que se va a hacer un uso intenso, y debemos evitar que la información generada con ella quede almacenada de forma aislada por cada usuario. Debemos conseguir que esa información que va generando cada uno de forma independiente sea compartida con el resto de la organización, para ello, a nivel de servidor nos ofrece el producto

llamado **SharePoint Server**, que es la plataforma de colaboración empresarial con la que podemos administrar los contenidos a través de la conocida interfaz de Office. Éste permite a todos los usuarios participar en la administración de contenidos de una forma regulada y conforme a las normativas establecidas en la organización.

A SharePoint se le han incorporado dos servicios que van a ser ampliamente utilizados cuando construimos una solución de BI, estos son:

- **Excel Services:** simplifican la forma de usar, compartir, proteger y administrar libros de Excel como informes interactivos de manera coherente para toda la empresa. Como es un componente de SharePoint, puede aprovechar muchas de sus características, como controlar, proteger y administrar acceso a las hojas de cálculo, el rendimiento del servidor y la posibilidad de agregar nuevos usuarios.
- **Performance Point Services:** permite a los usuarios navegar por la información disponible de una forma fácil, sintética, concreta, visual e intuitiva para ellos. Nos permite crear paneles interactivos que muestran los indicadores clave de rendimiento (KPI) y visualizaciones de datos en forma de cuadros de mandos. También permite la realización de informes analíticos, y aplicar filtros sobre dichos informes y paneles interactivos.

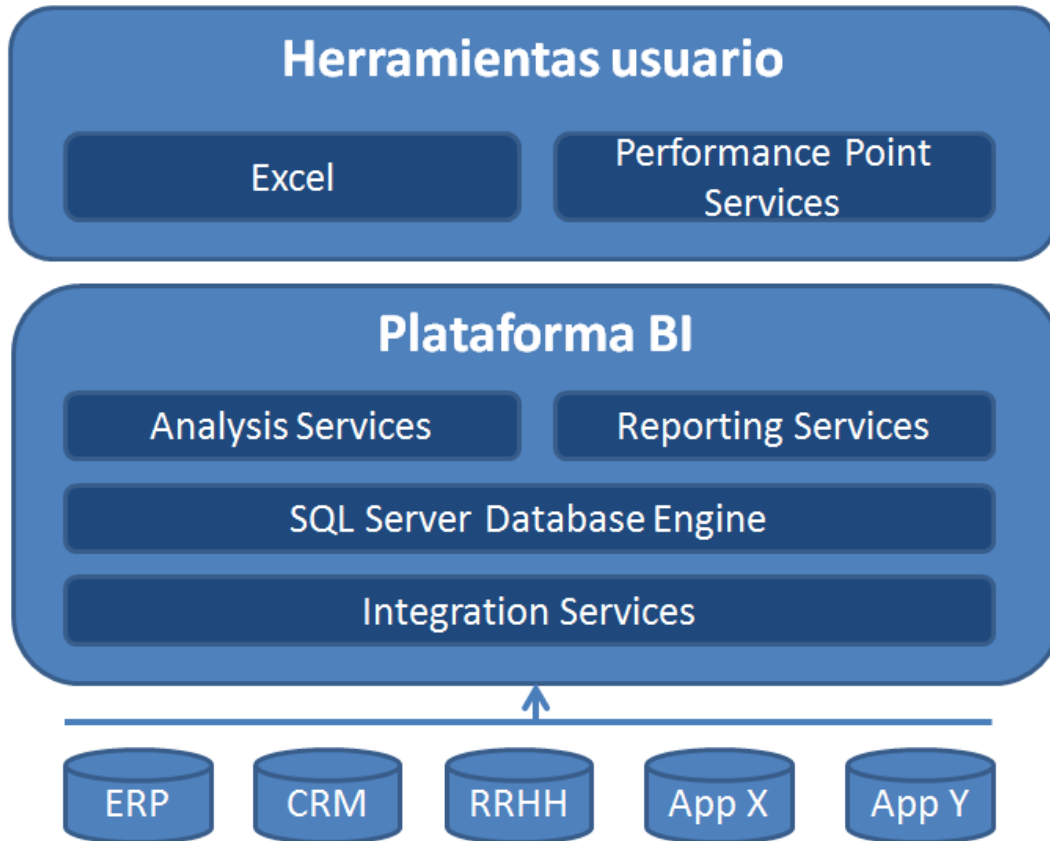


Figura 2-1 Componentes Microsoft BI

Herramientas cliente de SQL Server

Veamos a continuación las herramientas que vienen incluidas con SQL Server, que van a permitir tanto a desarrolladores como a administradores gestionar el producto e implementar soluciones basadas en él.

BIDS (Business Intelligence Development Studio)

BIDS es Visual Studio con una serie de tipos de proyectos adicionales específicos para Business Intelligence. Es el entorno de desarrollo para soluciones y proyectos de Integration Services, Analysis Services y Reporting Services, disponiendo de plantillas de proyecto específicas para la creación de los objetos requeridos en una solución de BI, y ofrece una serie de diseñadores, herramientas y asistentes para trabajar con dichos objetos.

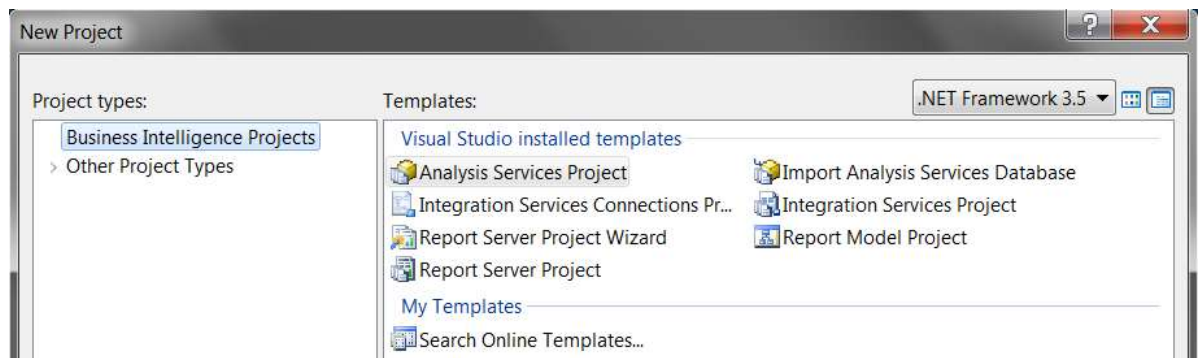


Figura 2-2 Tipos de proyectos Business Intelligence en BIDS

Si no tiene instalado Visual Studio, se instalará en su equipo incluyendo solamente el grupo de proyectos de Business Intelligence, y si ya tiene instalado Visual Studio, lo mantendrá, agregando dicho grupo de proyectos a los existentes.

SSMS (SQL Server Management Studio)

SSMS es una herramienta integrada para acceder, configurar, administrar y desarrollar sobre SQL Server. Lo que incluye, además de poder trabajar con el motor relacional, hacerlo con Integration Services, Analysis Services, Reporting Services y SQL Server Compact. Nos permite conectarnos a cualquier instancia de cualquier servidor con el que tengamos conectividad.

Con él podemos hacer tareas tan diversas como ejecutar un script Transact SQL, crear y gestionar objetos de forma visual, ejecutar scripts XMLA o consultas MDX sobre un servidor de Analysis Services, crear un plan de mantenimiento de una base de datos, crear trabajos y programar su ejecución, importar o exportar a un servidor un paquete de Integration Services, y un largo etcétera.

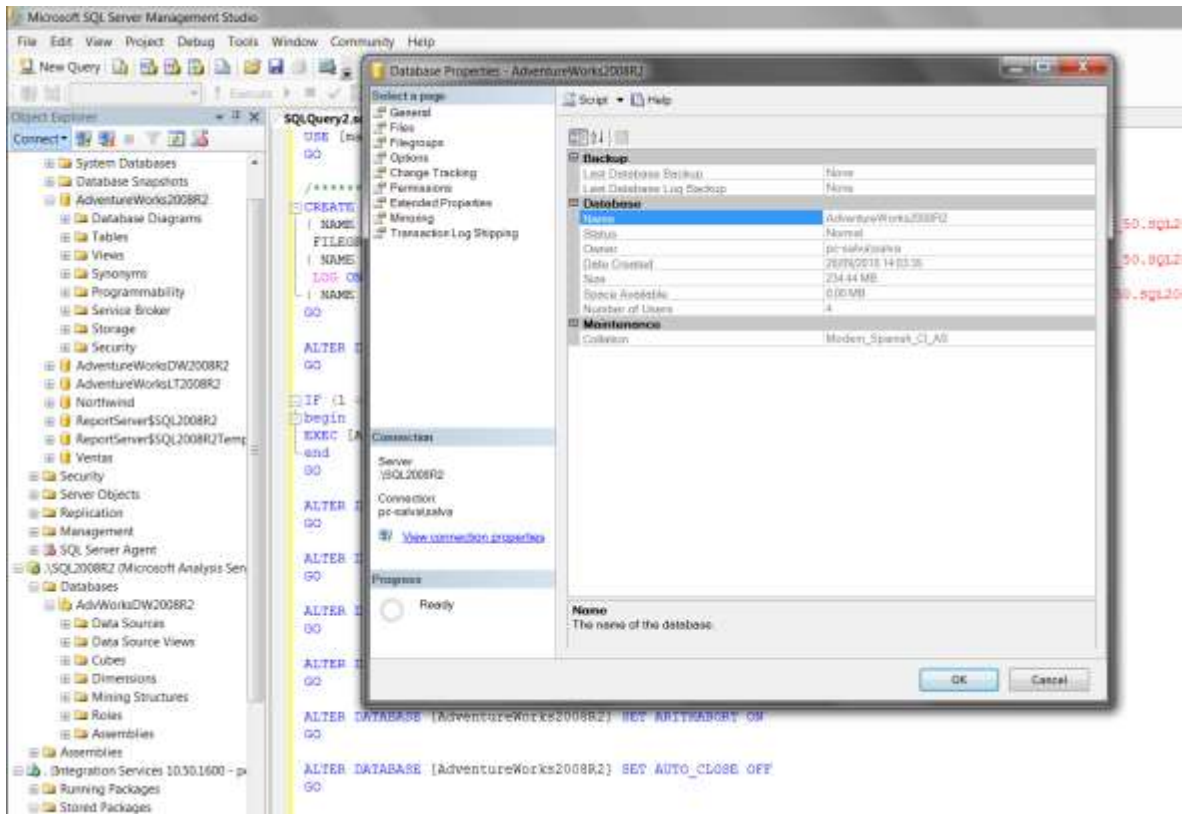


Figura 2-3 SQL Server Management Studio

B.O.L. (Books OnLine, libros en pantalla)

Los Books Online son la Ayuda de SQL Server. Es una documentación que siempre debemos tener instalada y a mano, ya que tiene una gran calidad, cubre prácticamente todos los aspectos del producto, y es muy clara a la hora de mostrar cualquier concepto, función, componente o cualquier otro detalle del producto.

Otras herramientas cliente

Hay otra serie de herramientas cliente, que vienen incluidas en el producto, y que no vamos a describir aquí, simplemente las vamos a citar para que el lector conozca su existencia y pueda buscar información de ellas, tanto en la propia ayuda del producto como en internet.

- SQL Server Configuration Manager
- Reporting Services Configuration Tool
- SQL Server Profiler
- Analysis Services Deployment Utility
- SQL Server Error and Usage Reporting
- Data Profile Viewer
- Dtexec utility
- Sqlcmd

- Etcétera

Y por último citar el sitio www.codeplex.com, donde podrá encontrar diversas aplicaciones desarrolladas por terceros, así como las bases de datos de ejemplo (AdventureWorks, Northwind, Contoso, etc.), y ejemplos de código.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al video '**Demo Intro 01A**' donde se puede ver la experiencia de usuario en una solución de Microsoft BI. Se muestran diversos informes de Reporting Services y su forma de navegación por la información, así como la realización de consultas ad-hoc sobre una base de datos de Analysis Services utilizando las *PivotTables* de Excel para ello. Esta demo le ayudará a comprender el objetivo final de este tipo de proyectos, que sin una buena experiencia de usuario que cubra sus expectativas dejan de tener valor.

3. Modelado dimensional: preparando los datos para su análisis

El modelo dimensional es una adaptación del modelo relacional, con el fin de optimizarlo para dar una rápida respuesta a las consultas realizadas por los usuarios. Aunque a nivel físico, una vez implementado en un sistema gestor de bases de datos relacionales, lo que allí encontramos son tablas y relaciones entre ellas, a nivel conceptual debemos conocer que existen dos tipos de tablas: tablas de dimensiones y tablas de hechos.

Un modelo dimensional se diseña siguiendo los esquemas en estrella (*star schema*) o copo de nieve (*snowflake*) vistos anteriormente.

Tablas de Dimensiones (*Dimension Tables*)

Son las que almacenan la información de las dimensiones. Una dimensión contiene una serie de atributos o características, por las cuales podemos agrupar, rebanar o filtrar la información. A veces estos atributos están organizados en jerarquías que permiten analizar los datos de forma agrupada, dicha agrupación se realiza mediante relaciones uno a muchos (1:N). Por ejemplo, en una dimensión Fecha es fácil que encontremos una jerarquía formada por los atributos Año, Mes y Día, otra por Año, Semana y Día; en una dimensión Producto podemos encontrarnos una jerarquía formada por los atributos Categoría, Subcategoría y Producto. Como se ha podido comprobar en los ejemplos, se puede dar el caso de que exista más de una jerarquía para una misma dimensión.

Las tablas de dimensiones, por lo general son muy anchas (contienen muchos atributos, y éstos pueden tener bastantes caracteres cada uno) y cortas (suelen tener pocas filas). Por ejemplo en una dimensión Producto, podemos encontrarnos con que tiene varias decenas de atributos, y que éstos están desnormalizados. No es extraño encontrarnos aquí valores ya en texto, no claves a otras tablas, por ejemplo categoría ('alimentación', 'textil', etc.), subcategoría ('congelados', 'frescos', 'bebidas', etc.), colores ('rojo', 'verde', 'amarillo', 'azul', etc.), tamaño ('pequeño', 'mediano', 'grande'), etc.

ProductoSK	ProductoBK	Categoría	Subcategoría	Producto	Peso	Tamaño	Color	Coste	FechaInicio	FechaFin
350	BK-M82B-44	Bicicleta	Bicicleta de montaña	Montaña: 100, negra, 44	21,13	44	Black	1898,0944	2005-07-01	2006-06-30
806	BK-M82B-44	Bicicleta	Bicicleta de paseo	Montaña: 100, negra, 44	21,13	44	Black	1898,0944	2007-06-30	NULL
352	BK-M68S-38	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23,35	38	Silver	1117,8559	2006-07-01	2007-06-30
581	BK-M68S-38	Bicicleta	Bicicleta de paseo	Montaña: 200, plateada, 38	23,35	38	Silver	1117,8559	2007-06-30	NULL
353	BK-R79Y-42	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23,35	38	Silver	1265,6195	2007-06-30	NULL

Figura 3-1 Tabla Dimensión Producto (*star schema*)

En una tabla de dimensiones, habitualmente no es posible utilizar la clave de negocio (*business key*) como clave principal (*primary key*), e incluso en el caso de que sea posible no es aconsejable. Una clave de negocio como clave principal no es aconsejable en muchos casos por temas de rendimiento, ya que desde este punto de

vista es recomendable utilizar números enteros de pocos bytes (en el caso concreto de SQL Server es muy recomendable usar el tipo de datos INT que ocupa 4 bytes). Si en el sistema transaccional tenemos, por ejemplo, una clave principal que es un char(10) siempre será más óptimo utilizar un tipo de datos numérico de menos bytes como clave principal en nuestra tabla de dimensiones. Adicionalmente hay casos en los que ya no es tan sólo un tema de rendimiento, sino que simplemente no es viable utilizar como clave principal de la tabla de dimensiones la clave principal de la tabla del transaccional. Por ejemplo, si tenemos varios orígenes de datos que queremos consolidar en una misma tabla de dimensiones, y cada uno utiliza un tipo de datos o longitud diferente, o simplemente para un mismo elemento en cada tabla de origen tiene un valor diferente. Por otro lado cuando queremos almacenar el historial de cambios, cumpliendo las características básicas que debe cumplir un Data Warehouse según *Inmon*, necesitamos tener, en muchos casos, varias filas con las diferentes versiones de los atributos y el periodo durante el que han estado vigentes, lo que implica que en la tabla de dimensiones habrá duplicidades en la clave de negocio, lo que impide que ésta pueda ser la clave principal.

Por tanto, lo habitual es que optemos por tener una clave principal diferente, esta clave es lo que se conoce con el nombre de clave subrogada.

Una **Clave subrogada** (*subrogate key*) es un identificador único que es asignado a cada fila de la tabla de dimensiones, en definitiva, será su clave principal. Esta clave no tiene ningún sentido a nivel de negocio, pero la necesitamos para identificar de forma única cada una de las filas. Son siempre de tipo numérico, y habitualmente también son auto incrementales. En el caso de SQL Server recomendamos que sean de tipo INT con la propiedad *identity* activada (es una recomendación genérica, a la que siempre habrá excepciones).

Una **Clave de negocio** (*business key*) es una clave que actúa como *primary key* en nuestro origen de datos, y es con la que el usuario está familiarizado, pero no puede ser clave principal en nuestra tabla de dimensiones porque se podrían producir duplicidades, como veremos más adelante al explicar el concepto de *Slowly Changing Dimensions*.

SCD (Slowly Changing Dimensions)

Como hemos comentado anteriormente, la información de los sistemas transaccionales puede ser modificada, aunque éstos sólo guardan la última versión. Por el contrario en un Data Warehouse, debemos reflejar ese historial de cambios para mostrar la verdad que había en el momento en que se produjeron los hechos.

Veamos un ejemplo. Si en nuestro sistema transaccional asociamos cada venta al comercial que la realiza, y éste a su vez depende de un director de zona. En la tabla de ventas queda reflejado el comercial que realiza la venta, y en la tabla del

empleado se almacena el director de zona del que depende, ya que tenemos los datos normalizados. ¿Qué ocurre si un comercial, por cualquier motivo, bien personal o bien laboral, le cambian la zona asignada?, ¿Y si además la nueva zona depende de otro director de zona?, ¿Y qué ocurre si sacamos un informe de ventas de ese nuevo director de zona? Pues que se le han trasladado a él todas las ventas que ha hecho este comercial durante toda su vida laboral en la empresa. Esto no es real, e imaginamos que su antiguo jefe de zona no estará en absoluto de acuerdo con estos informes de ventas, además de que no son ciertos. Cuando diseñamos un Data Warehouse debemos evitar esta problemática que tenemos en muchos sistemas transaccionales, donde sólo tenemos la versión actual de los datos. Para ello hay una serie de técnicas que nos permiten ir detectando los cambios que ocurren en el transaccional y dejándolos reflejados. Volviendo con el ejemplo anterior, en la tabla de dimensiones se deberían tener dos filas (o versiones) del empleado, una en la que se indica cuál es su jefe de zona antiguo, y durante qué periodo ha sido su jefe de zona, y otra que indica cuál es su jefe actual y desde cuándo. Adicionalmente, cada una de las ventas debe estar apuntando a la versión correcta del comercial, es decir, las ventas deben apuntar a la versión del comercial correspondiente al momento en que se produjeron, quedando así reflejado el jefe de zona y la zona que realmente tenía asignados en el momento de cada venta.

Por el contrario hay otros casos en los que no necesito reflejar el historial, por ejemplo, si corrijo el nombre de dicho comercial porque lo tenía mal escrito, no quiero tener dos versiones de él, una con el nombre mal escrito y otra con él bien escrito, sino que quiero que se sobrescriba y siempre aparezca la versión actual que es donde está escrito correctamente.

Vistos estos ejemplos pasemos a describir los diferentes tipos de SCD más habituales:

- **SCD tipo 1, Sobre escritura:** la nueva información sobrescribe a la antigua, no se guardan históricos y sólo se tiene la versión actual. Dicha sobre escritura se produce cuando se detecta algún error en los valores para corregirlo y mejorar la calidad del dato.
- **SCD tipo 2, historial de cambios:** refleja toda la información histórica. Por cada cambio que se produzca, se crea una nueva fila en la tabla de dimensiones con la fecha de inicio y una nueva clave subrogada, y se marca la fecha de fin de la versión anterior. Cada hecho que entra, debe comprobar a qué versión de la fila en la tabla de dimensiones se debe asociar (qué clave subrogada debe almacenar) en función de la fecha en la que se produzca.

Como hemos visto en los ejemplos, es habitual, que incluso en una misma tabla haya atributos de *tipo 1* y de *tipo 2*, y deberemos dar el tratamiento adecuando a cada caso en nuestros procesos ETL.

El término *Slowly Changing Dimension*, SCD por sus siglas en inglés, suele aparecer traducido como “dimensiones lentamente cambiantes”, aunque en la ayuda de SQL Server (*Books Online*) aparece como “dimensiones de variación lenta”, téngalo en cuenta el lector si busca información en español sobre este término.

Recomiendo al lector que consulte la bibliografía de *Ralph Kimball*, especialmente el libro *“The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (second edition)”* donde se exponen ampliamente estos conceptos y otros relacionados con el modelado dimensional, así como una amplia variedad de casos prácticos

Por último quiero concluir indicando los elementos que van a formar una tabla de dimensiones:

- **Clave subrogada:** Es la clave principal de la tabla de dimensiones. Nos permite identificar de forma única cada fila, suele ser un entero auto incremental. Es totalmente transparente al usuario de negocio, no la usará en ningún momento, ni tan siquiera tendrá conocimiento de su existencia.
- **Clave de negocio:** Es la clave con la que trabaja habitualmente el usuario, pero no puede ser la clave principal porque se pueden producir duplicidades.
- **Atributos de la dimensión:** serán cada una de las características que necesitemos almacenar. Lo habitual es que haya varias decenas de ellos, incluso que en algunos casos superen el centenar.
- **Fecha de Inicio y Fecha de Fin:** Servirán para conocer el periodo de vigencia de cada una de las versiones de los atributos

Básicamente, una tabla de un sistema OLTP tiene una clave de negocio que suele ser además clave principal, y una serie de atributos. Cuando tenemos una tabla de dimensiones, la clave de negocio deja de ser la clave principal, aparece una nueva clave principal, que es la clave subrogada, y se agregan dos columnas (FechaInicio y FechaFin) para gestionar los periodos de vigencia de cada versión. Además es habitual que tenga un mayor número de atributos, que son la recopilación de los existentes en las diferentes fuentes de datos, y algunos adicionales que se calculan en el ETL.

Volvamos a mostrar la figura de la tabla de la dimensión Producto donde se aprecia todo lo visto para tres productos. Vemos que los productos BK-M83B-44 y BK-M68S-38, han tenido dos versiones, dado que se cambiaron de categoría, así como las fechas en las que ha estado vigente cada una de ellas, mientras que el producto BK-R79Y-42 sólo tiene una versión. También podemos saber cuál es la versión actual de cada uno de ellos, ésta es la que la columna FechaFin vale NULL.

ProductoSK	ProductoBK	Categoría	Subcategoría	Producto	Peso	Tamaño	Color	Coste	FechaInicio	FechaFin
350	BK-M82B-44	Bicicleta	Bicicleta de montaña	Montaña: 100, negra, 44	21,13	44	Black	1898,0944	2005-07-01	2006-06-30
806	BK-M82B-44	Bicicleta	Bicicleta de paseo	Montaña: 100, negra, 44	21,13	44	Black	1898,0944	2007-06-30	NULL
352	BK-M68S-38	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23,35	38	Silver	1117,8559	2006-07-01	2007-06-30
581	BK-M68S-38	Bicicleta	Bicicleta de paseo	Montaña: 200, plateada, 38	23,35	38	Silver	1117,8559	2007-06-30	NULL
353	BK-R79Y-42	Bicicleta	Bicicleta de montaña	Montaña: 200, plateada, 38	23,35	38	Silver	1285,6195	2007-06-30	NULL

Figura 3-2 Tabla Dimensión Producto

Tablas de Hechos (*Fact Tables*)

Son tablas que representan un proceso de negocio, por ejemplo, las ventas, las compras, los pagos, los apuntes contables, los clics sobre nuestro sitio web, etc. Están formadas por los siguientes elementos:

- **Clave principal:** identifica de forma única cada fila. Al igual que en los sistemas transaccionales toda tabla debe tener una clave principal, en una tabla de hechos puede tenerla o no, y esto tiene sus pros y sus contras, pero ambas posturas son defendibles.
- **Claves externas(*Foreign Keys*):** apuntan hacia las claves principales (claves subrogadas) de cada una de las dimensiones que tienen relación con dicha tabla de hechos.
- **Medidas(*Measures*):** representan columnas que contienen datos cuantificables, numéricos, que se pueden agregar. Por ejemplo, cantidad, importe, precio, margen, número de operaciones, etc.
- **Metadatos y linaje:** nos permite obtener información adicional sobre la fila, como por ejemplo, que día se incorporó al Data Warehouse, de qué origen proviene (si tenemos varias fuentes), etc. No es necesario para el usuario de negocio, pero es interesante analizar en cada tabla de hechos qué nos aporta y si merece pena introducir algunas columnas de este tipo.

Las tablas de hechos, habitualmente son muy estrechas, tienen pocas columnas, además éstas son en su mayoría numéricas y de una longitud corta, de muy pocos bytes. Aunque sí que suelen ser muy largas, tienen un gran número de filas.

ProductoFK	FechaVentaFK	FechaEnvíoFK	FechaCobroFK	TiendaSK	ComercioSK	Albarán	Cantidad	PrecioUnit.	Descue.	PrecioCosto	Impuestos	OrigenFin	FechaOper
218	20050701	20050713	20050708	876	285	SO43659	6	5,70	0	3,3963	2,736	3	20050702
223	20050701	20050713	20050708	876	285	SO43659	2	5,1865	0	5,7052	0,8288	3	20050702
220	20050701	20050713	20050708	876	285	SO43659	4	20,1865	0	12,0278	6,4597	3	20050702
326	20050701	20050713	20050708	117	285	SO43660	1	419,4389	0	413,1463	33,5567	3	20050702
319	20050701	20050713	20050708	117	285	SO43660	1	874,794	0	884,7083	69,9835	3	20050702
300	20050701	20050713	20050708	442	288	SO43661	1	809,78	0	699,0928	64,7808	3	20050702
296	20050701	20050713	20050708	442	288	SO43661	1	714,7043	0	617,0281	57,1763	3	20050702
304	20050701	20050713	20050708	442	288	SO43661	2	714,7043	0	617,0281	114,3527	3	20050702
223	20050701	20050713	20050708	442	288	SO43661	4	5,1865	0	5,7052	1,6597	3	20050702

Figura 3-3 Tabla de Hechos de Ventas

En la figura anterior se muestra la tabla de hechos de ventas, en la que podemos apreciar que tiene una serie de claves externas a las dimensiones: Producto, Fecha (hay varias fechas), Tienda y Comercial. Adicionalmente tiene el número de venta; luego una serie de medidas: Cantidad, Precio Unitario, Descuento, PrecioCosto e Impuestos. Y por último un par de columnas de metadatos: OrigenFila (identifica el sistema de origen desde el que se obtuvo dicha fila) y FechaOper (fecha en la que la fila entró en la tabla de hechos).

Es importante a la hora de diseñar una tabla de hechos, tener en cuenta el nivel de granularidad que va a tener, es decir, el nivel de detalle más atómico que vamos a encontrar de los datos. No es lo mismo tener una fila por cada venta, que una fila donde se indiquen las ventas del día para cada artículo y tienda. A mayor granularidad, mayor será el número de filas de nuestra tabla de hechos, y dado que el espacio en disco y rendimiento no se ven notablemente afectados en los sistemas actuales, debemos llegar siempre al máximo nivel de granularidad que resulte útil a los usuarios.

La **agregación** es el proceso por el cual se resumen los datos a partir de las filas de detalle de máxima granularidad. Hoy en día disponemos de sistemas OLAP que se encargan de agregar dichos datos y ofrecerlos al usuario con una gran rapidez y eficacia.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al video '**Demo Intro 02A**' donde se muestra un modelo dimensional basado en la base de datos de ejemplo AdventureWorksDW. En él se muestran todos los detalles vistos en este capítulo a nivel teórico ya aplicados sobre una solución real.

4. Integration Services

Introducción

Integration Services es una herramienta que apareció con *SQL Server 2005*, dando un salto radical con respecto a las herramientas proporcionadas por versiones anteriores. Pasemos a incluir la propia definición que hace Microsoft de ella:

“Microsoft Integration Services es una plataforma para la creación de soluciones empresariales de transformaciones de datos e integración de datos. Integration Services sirve para resolver complejos problemas empresariales mediante la copia o descarga de archivos, el envío de mensajes de correo electrónico como respuesta a eventos, la actualización de almacenamientos de datos, la limpieza y minería de datos, y la administración de objetos y datos de SQL Server. Los paquetes pueden funcionar por separado o conjuntamente con otros paquetes para hacer frente a las complejas necesidades de la empresa. Integration Services puede extraer y transformar datos de muchos orígenes distintos, como archivos de datos XML, archivos planos y orígenes de datos relacionales, y, posteriormente, cargarlos en uno o varios destinos.

Integration Services contiene un variado conjunto de tareas y transformaciones integradas, herramientas para la creación de paquetes y el servicio Integration Services para ejecutar y administrar los paquetes. Las herramientas gráficas de Integration Services se pueden usar para crear soluciones sin escribir una sola línea de código. También se puede programar el amplio modelo de objetos de Integration Services para crear paquetes mediante programación y codificar tareas personalizadas y otros objetos de paquete.”

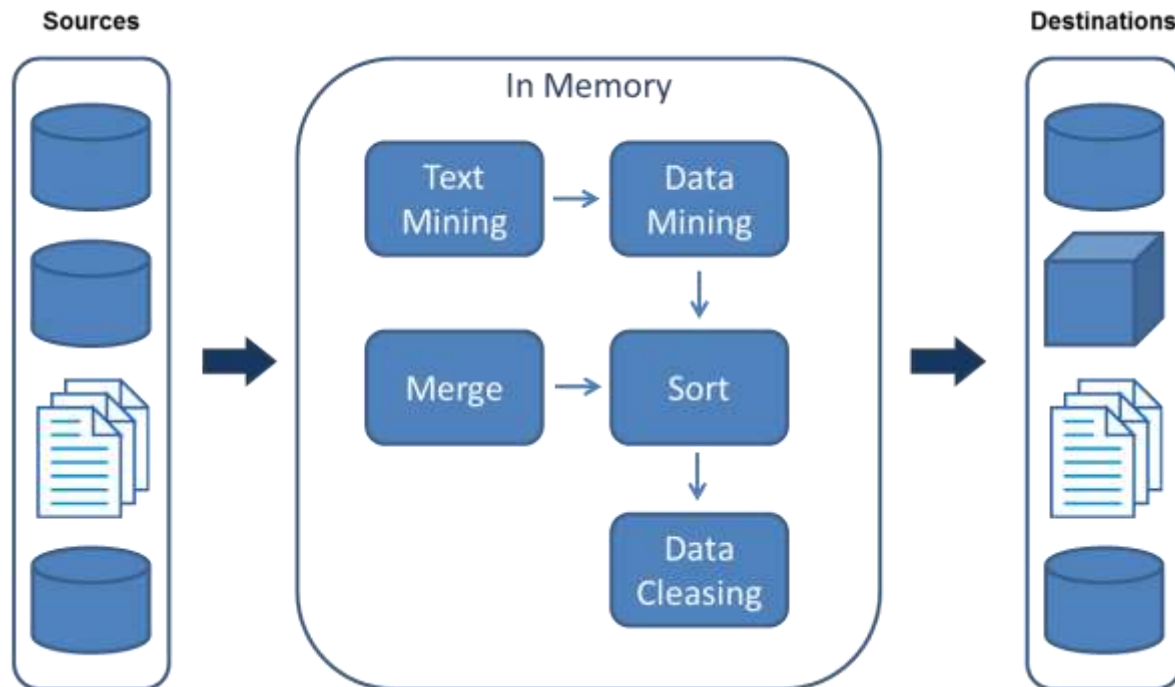


Figura 4-1 Integration Services

Como se indica en la definición anterior, no es sólo una herramienta para ETL, sino que también tiene una serie de tareas que serán utilizadas por los DBA's (*Data Base Administrators*) para la administración de los servidores, de las bases de datos y de sus objetos. Aunque aquí nos vamos a centrar en las capacidades de la herramienta para el desarrollo de procesos ETL, dejando para otra ocasión el uso de tareas orientadas a los administradores de bases de datos.

En la mayor parte de los proyectos de BI, tenemos un componente fundamental, que es nuestro Data Warehouse y/o los diversos Data Marts. En ellos dispondremos de los datos sobre los que se van a centrar las consultas de los usuarios. También va a servir como fuente de datos para la carga de los cubos de Analysis Services y para la explotación de la información desde herramientas de analíticas y de reporting.

En nuestros procesos de ETL, utilizaremos *Integration Services* para extraer los datos de orígenes de datos muy diversos, como bases de datos relacionales (*SQL Server*, *Oracle*, *MySQL*, u otra), archivos de texto, libros de *Excel*, y datos no tradicionales, como los archivos XML o los servicios web. Los transformaremos, limpiaremos y almacenaremos, posiblemente en un área intermedia (*Staging Area*), y finalmente en un *Data Mart* o *Data Warehouse*, siendo ambas, bases de datos relacionales, alimentadas con datos de los orígenes citados anteriormente. También realizaremos tareas de procesamiento periódico de los cubos de Analysis Services, los cuales tendrán como origen de datos dicho Data Mart o Data Warehouse.

Por último, destacar que este tipo de herramientas son muy potentes. Los desarrollos con ellas son muy rápidos, y nos permiten crear una gran cantidad de procesos en reducidos periodos de tiempo de desarrollo e implantación. Es muy sencillo crear un nuevo paquete de *Integration Services* que lea, transforme y mueva datos a un nuevo destino, luego de ese destino hacer nuevos procesos para los cuales sea su origen de datos, y así sucesivamente, sin un buen diseño previo del objetivo global a conseguir y de las dependencias que se van creando entre ellos. Por ello, considero que esta herramienta puede ser un arma de doble filo: bien usada nos puede ayudar tremendamente a generar dichos procesos, coordinar su ejecución, y manejar las cadenas de dependencia entre ellos. Pero utilizarla sin un previo diseño y análisis de todo lo anterior, puede crearnos un grave problema, incluso podemos llegar al punto de que el tocar simple proceso puede ser muy costoso, por las dependencias e implicaciones que tiene con otros procesos y que les haría dejar de funcionar correctamente. En este sentido, creo que podemos aplicar la famosa frase “una imagen vale más que mil palabras”:

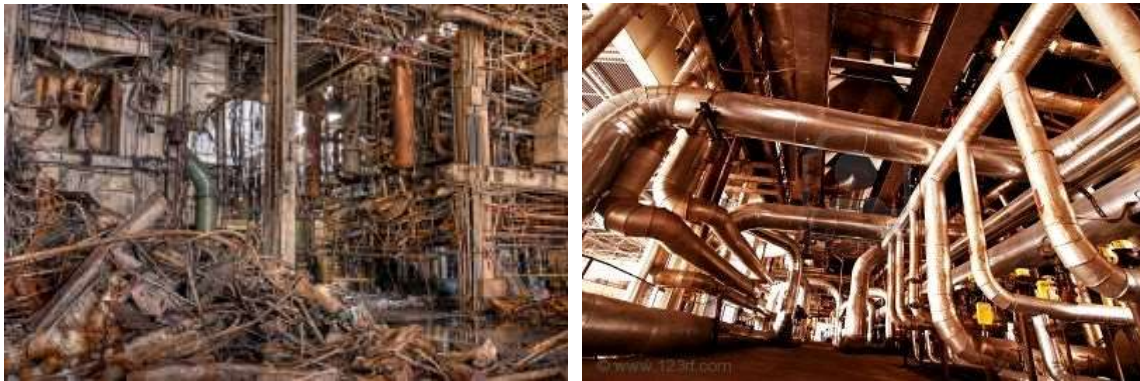


Figura 4-2 SSIS: Un arma de doble filo

Como ha podido comprobar, *Integration Services* es una herramienta muy completa y con muchas funcionalidades, e incluso con ciertos riesgos si no hacemos un uso apropiado de ella.

Integration Services es un servicio independiente que instalaremos y ejecutaremos en un servidor, y que será el encargado de almacenar y ejecutar los diversos procesos que hayamos definido. Estos procesos se almacenan en unos archivos XML que contienen toda la información de ese proceso, y que se llaman **paquetes**.

Vamos a citar una serie de características destacables del producto:

- Permite la integración con sistemas de bases de datos y con ficheros
- Obtiene un alto rendimiento al mantener los datos en memoria, además permite concurrencia y paralelismo.
- Permite gestionar alertas y notificaciones

- Tiene tareas avanzadas de “*data profiling*”, limpieza, y minería de texto y de datos.
- Desde el punto de vista del desarrollador, dispone de un entorno de desarrollo con el que se sentirá muy familiarizado, el *BIDS* (descrito anteriormente). Como en cualquier proyecto de *Visual Studio*, se pueden utilizar las herramientas de ciclo de vida.
- Desde el punto de vista de un DBA o un administrador de sistemas, existe un servicio, llamado SSIS, que es el que se encarga de la ejecución de los paquetes de *Integration Services*. También dispone de capacidades de configuración y despliegue en diversos entornos (desarrollo, pruebas, pre-producción, producción) sin necesidad de editar los paquetes con *BIDS* y hacer modificaciones adaptadas a cada uno de ellos.

Podemos concluir que es una potente herramienta que abarca todo el ciclo de vida de un proyecto de integración de datos, desde su fase inicial de desarrollo, pasando por la puesta en marcha en diferentes entornos, su ejecución programada y tareas de administración (log, auditoría,...)

A continuación vamos a comenzar a entender la herramienta desde el punto de vista del desarrollador, el cuál se tendrá que abordar el desarrollo de una solución basada en *Visual Studio*, y cuyo objetivo principal es la construcción, configuración y despliegue de los diferentes paquetes de *Integration Services*.

Un **paquete** está formado por una colección de conexiones, tareas de flujo de control (*ControlFlow*) y de flujo de datos (*DataFlow*), manejadores de eventos, variables y configuraciones. Estos paquetes se crean mediante *BIDS* y posteriormente se despliegan al servidor.

A continuación veremos los diferentes componentes que forman un paquete con más detalle.

Se recomienda instalar el producto en inglés y habituarse a la nomenclatura de los objetos en este idioma, dado que la mayor parte de la documentación y ejemplos está en este idioma. Otro inconveniente es que como los elementos en las barras de Herramientas aparecen por orden alfabético, cambian el orden en que se muestran en función del idioma utilizado, lo que hace más compleja la localización de los objetos en un idioma diferente. Por estos motivos todos los nombres de objetos en este documento aparecerán en inglés. Si el lector lo estima oportuno puede crearse su propia tabla con las traducciones que vea conveniente.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al video '**Demo SSIS 01A**' donde se puede ver el entorno de desarrollo para proyectos de *Integration Services*. Se muestra la creación de un proyecto y cómo se incluye dentro de una solución, para a continuación pasar a describir las diversas ventanas, pestañas y cajas de herramientas, así como los elementos que aparecen en cada una de ellas.

ControlFlow

Un paquete en sí es un contenedor que puede almacenar colecciones de los elementos citados anteriormente, en el que podemos crear diferentes tareas de flujo de control (*ControlFlow*) para proporcionar funcionalidad, agruparlos en contenedores, crear restricciones de precedencia y variables para intercambiar datos entre ellos.

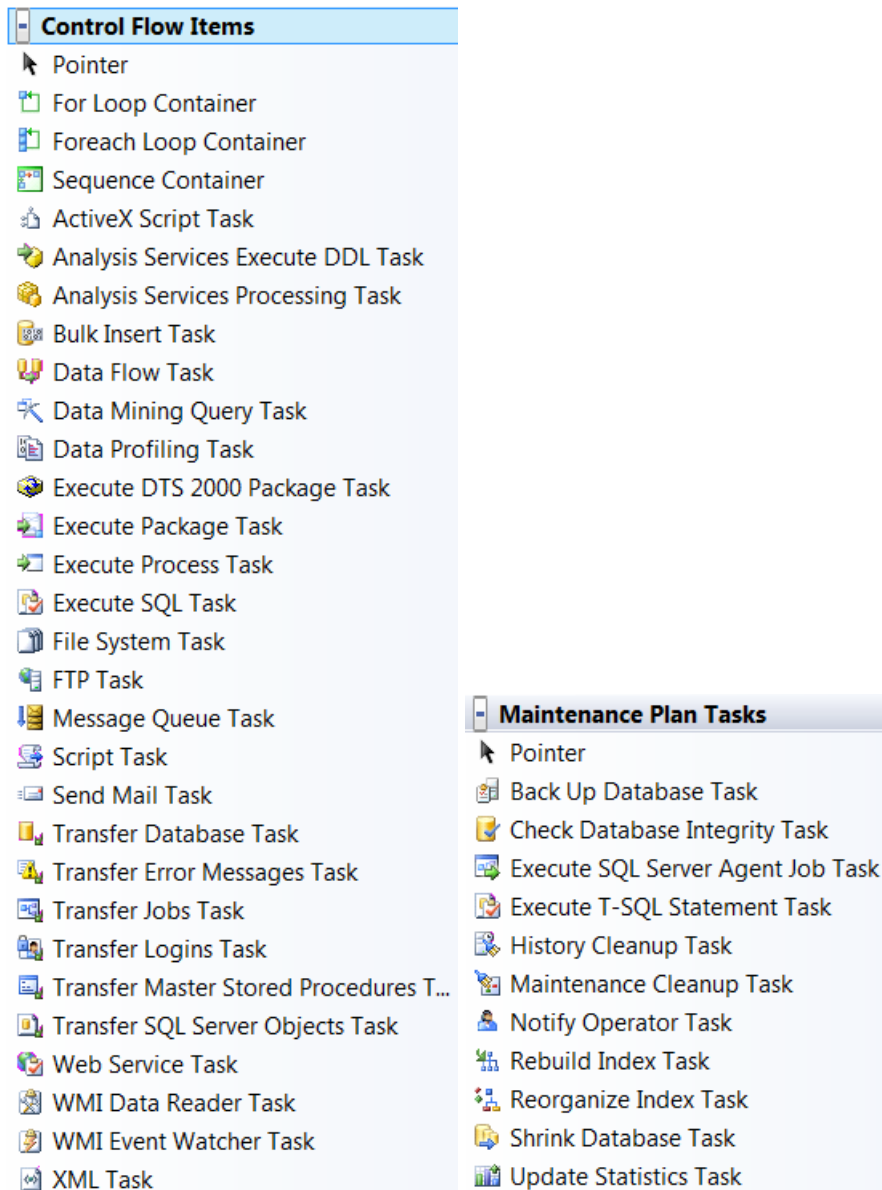


Figura 4-3 Relación completa de tareas del ControlFlow

Un contenedor es un objeto que a su vez es capaz de almacenar otros contenedores o tareas en su interior. Proporciona estructura a los paquetes y servicios a las tareas. Hay cuatro tipos de contenedores:

- *ForEach Loop container*: permite ejecutar los componentes que hay en su interior tantas veces como elementos tenga la colección a la que está asociado. Esta colección puede ser un conjunto de filas, un conjunto de archivos de una carpeta, etc.
- *For Loop container*: ejecuta los componentes un determinado número de veces. Es similar a la estructura *For Loop* de los lenguajes de programación.

- *Sequence container*: también agrupa tareas y contenedores, pero a diferencia de los dos anteriores, se ejecutará una sola vez. Su principal utilidad es proporcionar estructura y claridad al paquete.
- *Task container*: una tarea en sí es a su vez un contenedor que, a diferencia de los anteriores, no puede contener ningún elemento. Son las unidades de trabajo donde se proporciona la funcionalidad al paquete. Un paquete consta de una o más tareas.

Todos los elementos anteriores necesitan un nexo de unión entre ellos que nos permita ir definiendo el flujo a seguir en su ejecución. Este nexo es lo que se conoce como restricciones de precedencia (*precedence constraints*), que nos permitirán indicar si los contenedores o tareas se ejecutan en paralelo, si uno necesita de la finalización del otro para comenzar su ejecución, y además si necesita que se tras la ejecución del anterior se cumplan ciertas condiciones para su ejecución.

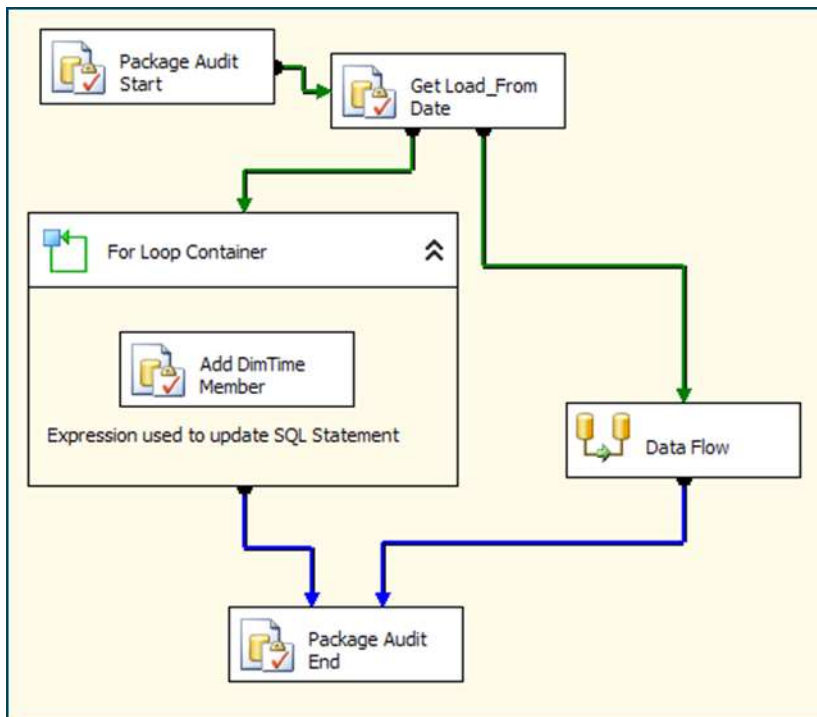


Figura 4-4 ControlFlow: contenedores, tareas y restricciones de precedencia

Vamos a ver una relación de las tareas más significativas que forman parte del *ControlFlow*, agrupadas según su funcionalidad:

Tareas de preparación de datos:

- *File System task*: realiza operaciones sobre archivos y carpetas del sistema de archivos. Puede crear, mover, copiar, renombrar o eliminar carpetas o archivos, así como cambiar atributos.
- *FTP task*: permite interactuar con un servidor FTP, pudiendo iniciar sesión, cargar o descargar archivos, y otras tareas de interacción mediante el protocolo FTP.
- *Web Service task*: permite ejecutar un método de un servicio web.
- *XML task*: permite trabajar con datos XML, recuperar documentos XML, aplicar operaciones a los documentos mediante XSLT y expresiones XPath, combinar varios documentos, o bien validar, comparar y guardar los documentos actualizados en archivos y variables.
- *Data Profiling task*: calcula diversos perfiles sobre los datos de una consulta SQL que le ayudan a familiarizarse con un origen de datos y a identificar en los datos problemas que deban corregirse. Tiene multitud de vistas sobre los datos, permitiéndonos entre otros, ver valores de una columna, redundancias, longitudes de los datos, etc.

Tareas de flujo de trabajo:

- *Execute Package task*: *Integration Services* permite que unos paquetes sean ejecutados desde otros paquetes como parte del *ControlFlow*. Para ello nos ofrece esta tarea.
- *Execute Process task*: permite ejecutar una aplicación o un archivo por lotes.
- *Send Mail task*: permite enviar mensajes de correo electrónico, por ejemplo, para indicar si las tareas han finalizado correctamente o no, o en respuesta a eventos provocados por el paquete en tiempo de ejecución.

Tareas de SQL:

- *Bulk Insert task*: es una forma muy eficaz de insertar grandes volúmenes de datos en SQL Server.
- *Execute SQL task*: permite ejecutar instrucciones SQL o procedimientos almacenados. Tenga en cuenta que habrá que utilizar el dialecto SQL del origen de datos al que esté conectado.

Tareas de scripting:

- *Script task*: permite extender las posibilidades utilizando VB.Net o C#, para realizar tareas que no están disponibles.

Tareas de Analysis Services:

- *Analysis Services execute DDL task*: permite crear, modificar o eliminar objetos de *Analysis Services*, mediante la generación de comandos *XMLA*.
- *Analysis Services Processing task*: procesa objetos de *Analysis Services*, como cubos, dimensiones y modelos de minería de datos.
- *Data Mining Query task*: ejecuta consultas de predicción basadas en modelos de minería de datos integrados en *Analysis Services*.

Tarea de flujo de datos:

- *Data Flow task*: encapsula el motor de flujo de datos que mueve datos entre orígenes y destinos, y permite al usuario transformar, limpiar y modificar datos a medida que se mueven.

Recomendamos que consulte los Books Online y revise con mayor detalle todas las tareas disponibles. Aquí hemos mostrado las que consideramos más habituales cuando realizamos procesos ETL, dejando de lado otra serie de tareas menos habituales u orientadas a DBA's.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda a los siguientes videos:

'Demo SSIS 02A' donde se muestra la funcionalidad de crear tareas en el *ControlFlow*, utilizando la funcionalidad de arrastrar y soltar desde el *Toolbox* los diferentes elementos.

'Demo SSIS 02B' que muestra la funcionalidad de *data profiling*, creando una tarea que obtenga la información de una tabla y la almacene en un fichero XML, que después será abierto con la utilidad *Data Profile Viewer* para examinar toda la información generada sobre los datos de dicha tabla.

'Demo SSIS 02C' para mostrar el uso del *For Loop Container* con un ejemplo didáctico que permite iterar N veces sobre una misma tarea o conjunto de éstas.

'Demo SSIS 02D' que muestra cómo establecer restricciones de precedencia (*precedence constraints*) entre las diferentes tareas y contenedores del *ControlFlow*.

'Demo SSIS 02E' que explica la utilización de variables en los paquetes, diversos usos que les podemos dar y ámbito de visibilidad de éstas.

Microsoft Business Intelligence: vea el cubo medio lleno

'Demo SSIS 02F' donde por último mostramos un ejemplo que utiliza las características del bucle *For Each Loop* para interactuar con los ficheros existentes en una carpeta determinada.

Laboratorio incluido en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al material del **'Lab SSIS 02A'**, realice paso a paso todo lo allí expuesto y responda a las preguntas que se incluyen. Este Lab le ayudará a asentar los conocimientos adquiridos sobre el *ControlFlow*, creando un paquete que utilice diversos contenedores, tareas y variables. Y por supuesto, si tiene cualquier duda sobre lo visto o cualquier situación que quiera resolver puede utilizar las sesiones de **tutorías** y los **foros** del curso para resolverlas.

DataFlow

Como hemos visto anteriormente *DataFlow Task* es la tarea más compleja y con más posibilidades de configuración, dada su amplitud, la vamos a estudiar a continuación con más detalle, y vamos a ver cada uno de los múltiples elementos que la forman.

Está compuesta de un conjunto de componentes conectados, que extraen datos, hacen transformaciones, enrutan o resumen dichos datos, y los graban en destino.



Figura 4-5 Relación completa de elementos del DataFlow

Antes de continuar, hay que tener en cuenta que tiene una gran similitud con el *ControlFlow*, pero no desde el punto de vista funcional que es muy diferente, sino desde el punto de vista visual, y el lector debe tener claro en cualquier momento si está posicionado en el *ControlFlow* o en el *DataFlow*, así como los diferentes elementos que le deben aparecer en cada caso. En el *ControlFlow* aparecerán las

diversas tareas y contenedores que hemos estudiado anteriormente. Por el contrario, en el *DataFlow* aparecerán una serie de orígenes de datos, de transformaciones y de destinos (ver figura anterior), que nos permitirán implementar las funcionalidades que necesitemos en los flujos de datos, consistiendo estos en leer de uno o varios orígenes, hacer las transformaciones y limpiezas necesarias, y grabar en uno o varios destinos.

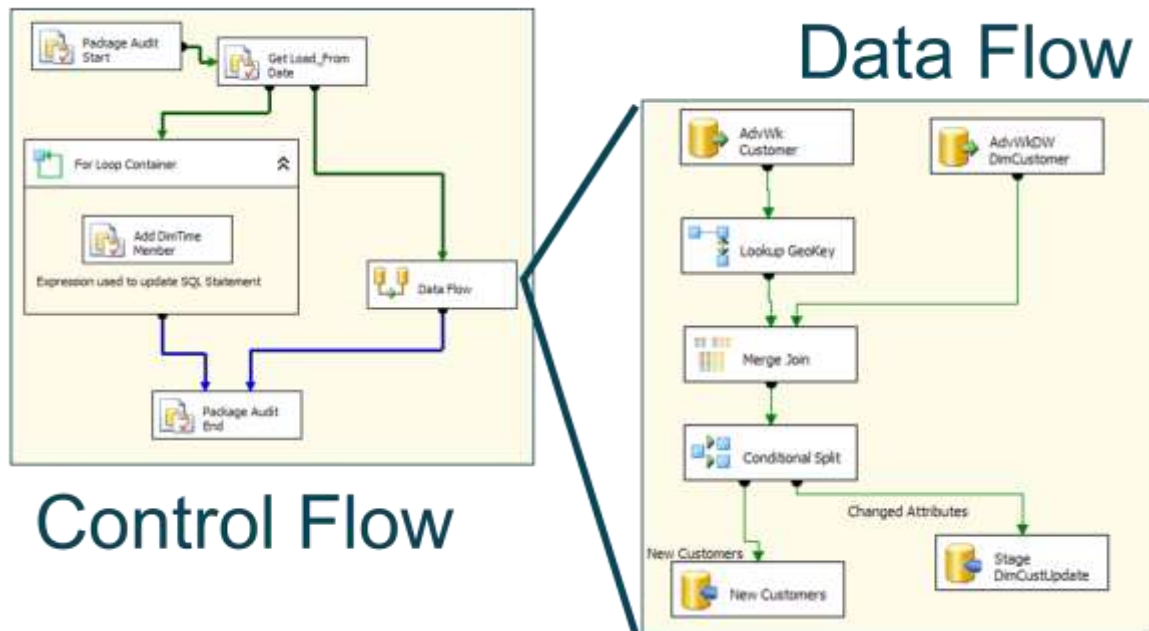


Figura 4-6 ControlFlow vs DataFlow

A continuación vamos a citar los elementos más utilizados disponibles en un *DataFlow*:

Orígenes de datos:

- *ADO.NET*: consume datos de un proveedor *.NET*.
- *Excel*: extrae datos de un archivo *Excel*.
- *Flat file*: extrae datos de un archivo plano.
- *OLE DB*: consume datos de un proveedor *Ole Db*. Suele ser el más utilizado.
- *Raw File*: son unos volcados de los buffers de *Integration Services* a disco. Lee datos en este formato.
- *Script Component*: utiliza un script con código *.Net* para extraer, transformar o cargar datos.

Transformaciones:

- *Aggregate*: aplica funciones de agregado (sum, min, max, avg, etc.) a valores de columnas y copia los resultados a la salida de la transformación. Para ello dispone de una cláusula *group by* que permite decidir los grupos que componen el agregado.
- *Audit*: permite agregar al flujo de datos columnas con información sobre el entorno de ejecución del paquete, como el usuario, el nombre del paquete, el equipo que lo ejecuta, la fecha y hora de ejecución, etc.
- *Caché Transform*: escribe los datos que vienen en el flujo de datos en un administrador de conexión de caché, permitiendo almacenar estos datos en un archivo *.raw*, que pueden ser utilizados por otros componentes como *Lookup*.
- *Character Map*: aplica funciones de cadenas sobre los datos de tipo carácter.
- *Conditional Split*: puede dirigir las filas a diferentes salidas en función de que cumplan una serie de condiciones. Es una estructura de decisión similar a la instrucción *CASE* de los lenguajes de programación, evalúa expresiones y dirige la fila de datos a la primera salida cuya expresión devuelva *True*.
- *Copy Column*: crea columnas nuevas que son una copia de las columnas indicadas.
- *Data Conversion*: convierte de un tipo de datos a otro creando una nueva columna con el resultado.
- *Data Mining Query*: realiza consultas de predicción en modelos de minería de datos, utilizando para ello el lenguaje *DMX (Data Mining eXpressions)*.
- *Derived Column*: crea nuevos valores de columna sobre los valores de entrada aplicando expresiones. El resultado puede crearse en una nueva columna o sustituir el valor en una de las columnas existentes. Tiene un pequeño lenguaje de expresiones que incluye ciertas funciones de transformación, tratamiento de caracteres, de fechas, etc. Por tanto puede sustituir el uso de transformaciones como *Data Conversion* o *Copy Column*, ya que en dicho lenguaje incluye funciones que pueden realizar las mismas tareas que esas transformaciones.
- *Export Column*: permiten leer datos del flujo de datos e insertar cada uno de ellos a un archivo. Por ejemplo, puedo extraer las imágenes de los artículos de la base de datos y guardarlas todas en una carpeta.
- *Fuzzy Grouping*: permite agrupar datos por similitud aplicando técnicas de lógica difusa, ayudando en las tareas de limpieza de datos.
- *Fuzzy Lookup*: permite buscar datos en una tabla de referencia por similitud aplicando técnicas de lógica difusa. Nos puede ayudar a buscar datos similares en una tabla maestra. Un ejemplo típico de uso es cuando tenemos una columna con un campo de tipo carácter que contiene un nombre de población y queremos hacer tareas de limpieza para encontrar en una tabla maestra el valor más aproximado.

- *Lookup*: busca valores en una tabla de referencia con una coincidencia exacta.
- *Merge*: permite combinar conjuntos de datos ordenados, obteniendo como resultado el conjunto de filas de las diferentes entradas.
- *Merge Join*: combina conjuntos de datos ordenados, obteniendo por cada uno de ellos una fila con el conjunto de columnas de ambas entradas. Permite realizar operaciones que en SQL hacemos con las cláusulas *INNER JOIN*, *LEFT/RIGHT OUTER JOIN* y *FULL OUTER JOIN*.
- *Multicast*: distribuye una copia del conjunto de datos que recibe por cada una de las salidas que creemos.
- *Ole Db Command*: permite ejecutar comandos SQL por cada una de las filas del flujo de datos.
- *Percentage Sampling*: con él podemos extraer un subconjunto de datos de ejemplo, que representará el porcentaje de filas indicado sobre el total de datos de entrada. Por ejemplo, podemos utilizarlo para obtener el 10% de las filas de una tabla.
- *Pivot y Unpivot*: permite pivotar datos entre filas y columnas. Por ejemplo, si tenemos las columnas país, año e importe, podemos hacer que mediante la transformación *Pivot* nos devuelva una matriz de dos dimensiones, con una fila por país, una columna por año, y en el cruce de éstas se mostrará el importe. La transformación *Unpivot* hace el proceso inverso.
- *Row Count*: cuenta las filas que pasan a través de ella y almacena el resultado en una variable.
- *Row Sampling*: igual a *Percentage Sampling*, pero en vez de devolver un porcentaje de filas sobre el total, devuelve el número de filas indicado sobre el total.
- *Script Component*: permite extraer, transformar o cargar datos mediante código personalizado escrito en VB.NET o C#. Es muy útil cuando tenemos que realizar, por ejemplo, un cálculo que no nos lo permite hacer ninguna de las transformaciones existentes.
- *Slowly Changing Dimension*: coordina la inserción y modificación en una tabla de dimensiones, aplicando los diversos tipos de cambios descritos anteriormente en el apartado dedicado a *Slowly Changing Dimensions (SCD)*, así como la gestión de miembros inferidos (*inferred members*), también vista anteriormente. Contiene un asistente que nos guía paso a paso en la implementación de esta casuística.
- *Sort*: permite ordenar los datos por las columnas que se le indiquen, y además tiene una propiedad que permite eliminar las filas duplicadas según el conjunto de columnas por el que realizamos la ordenación.
- *Term Extraction*: permite extraer términos de un texto. Funciona sólo con textos en inglés, ya que sólo tiene un diccionario lingüístico para este idioma.
- *Term Lookup*: busca términos en una tabla de referencia y cuenta los términos extraídos de dicho texto. Esta transformación resulta útil para crear

una lista personalizada de palabras basada en el texto de entrada, que incluye estadísticas de frecuencia de aparición de palabras.

- *Union All*: combina varios conjuntos de datos de entrada en uno sólo. Es la función opuesta a *Multicast*.

Destinos del flujo de datos:

- *ADO.NET*: carga datos en bases de datos compatibles con el proveedor .NET.
- *Data Mining Model Training*: entrena modelos de minería de datos.
- *DataReader*: expone los datos mediante la interfaz ADO.NET DataReader.
- *Dimension Processing*: carga y procesa una dimensión de Analysis Services.
- *Excel*: escribe los datos en *Excel*.
- *Flat File*: escribe los datos en un archivo plano.
- *OLE DB*: carga datos mediante un proveedor *Ole Db*.
- *Partition Processing*: carga y procesa particiones de Analysis Services.
- *Raw File*: son unos volcados de los buffers de *Integration Services* a disco. Escribe datos en este formato.

Hay una serie de orígenes de datos, transformaciones y destinos adicionales creados por terceros que amplían el abanico de elementos disponibles, y que una vez instalados aparecen en *BIDS* como un elemento más en la *toolbar* disponible para uso.

Recomendamos que consulte los Books Online y revise con mayor detalle todos los elementos disponibles, así como las descripciones de sus funcionalidades, características y ejemplos de uso.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda a los siguientes videos donde podrá ver la utilización de la mayor parte de los componentes descritos anteriormente:

‘**Demo SSIS 03A**’ donde se muestra el uso de conexiones ADO.NET tanto para el origen como para el destino de los datos.

‘**Demo SSIS 03B**’ donde se expone el uso de algunas transformaciones para hacer ciertos cambios sobre los datos de origen, y además se lleva la información a más de un destino.

‘**Demo SSIS 03C**’ para mostrar el uso de la transformación *Lookup* aplicada a unos de los casos más típicos de uso, como es la obtención de las claves subrogadas de una tabla de dimensiones a partir de las claves de negocio.

‘**Demo SSIS 03D**’ que muestra el uso de la transformaciones *Import/Export Column* para extraer o grabar archivos en columnas de una base de datos de SQL Server.

‘**Demo SSIS 03E**’ que explica el uso de un *script component* para realizar un cálculo que no está implementado dentro de la funcionalidad del resto de componentes de transformación, ni del metalenguaje del componente *derived column*, lo que nos obliga a desarrollar dicho cálculo con VB.NET o C#.

Laboratorio incluido en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al material del ‘**Lab SSIS 03A**’, realice paso a paso todo lo allí expuesto y responda a las preguntas que se incluyen. Este Lab le ayudará a asentar los conocimientos adquiridos sobre el *DataFlow*, creando un paquete que incluye en su *ControlFlow* una tarea de *DataFlow* que se encarga de leer datos de un origen, hacer una serie de transformaciones y grabarlos en un destino diferente. Y por supuesto, si tiene cualquier duda sobre lo visto o cualquier situación que quiera resolver puede utilizar las sesiones de **tutorías** y los **foros** del curso para resolverlas.

Buenas prácticas

A continuación nos vamos a centrar en las características de *Integration Services* como herramienta ETL, y más concretamente en el uso de sus flujos de datos. Vamos a mostrar una serie de buenas prácticas, consejos y recomendaciones a seguir a la hora de implementar procesos de *Extracción, Transformación, Limpieza y Carga* de datos en nuestro Data Mart o Data Warehouse, pasando éstos por un *área de Staging*, dado que ésta es una casuística muy típica. Tenga también en cuenta que aunque ciertos puntos van orientados al proceso que acabamos de comentar, otros muchos pueden ser aplicables a otros usos de la herramienta.

Recomendaciones para la carga de un *staging area*

Comencemos haciendo una serie de recomendaciones y marcando unas pautas que debemos tener en cuenta cada vez que abordemos una carga de datos, y para las cuales nos será de gran utilidad disponer de un área intermedia de almacenamiento (*staging area*):

- **Mínimo impacto sobre el origen:** para conseguir una máxima escalabilidad y para afectar lo mínimo posible al rendimiento de los servidores transaccionales que van a ser nuestros orígenes de datos, debemos consumir de ellos los mínimos recursos posibles, y centrar el consumo de dichos recursos en el servidor que se encarga de realizar los procesos ETL.
- **Trazabilidad del dato:** es muy importante saber dónde, cuándo y quién ha realizado cada cambio. Para ello deberemos, además de realizar los propios procesos de extracción y de obtención de diferencias entre el origen y el destino, de ir almacenando todos los cambios que nos encontremos, para, en un momento dado, poder seguir la pista de cualquier dato y de los cambios que ha sufrido.
- **Generación de *tablas Delta*:** siempre es una alternativa mucho más óptima tener una *tabla delta*, que tenga registradas las operaciones de inserción, actualización y borrado que se han ido produciendo en el origen, y aplicar esos cambios en el destino. Este proceso es mucho más eficiente, y por supuesto, consume muchos menos recursos en origen y destino, que hacer una lectura de dicho origen, compararlo con el destino, y obtener de ahí las diferencias para seguidamente aplicarlas al destino. Hay ciertos sistemas que ya nos permiten generar este tipo de tablas, pero si no es así en nuestro caso, siempre podemos generarlas en nuestros procesos ETL.
- **Limpieza de datos:** otro punto que no debemos olvidar en ningún momento, es que en este tipo de procesos, el objetivo no es solamente traer los datos al destino y hacer ciertas transformaciones. Es muy importante llevar a cabo una serie de tareas de limpieza de datos y detección de incoherencias. Si por ejemplo nos encontramos con un código de artículo del que nos llega una venta, pero aún no está en nuestro sistema. Si podemos hacer ciertas correcciones sobre datos que tenemos la certeza de que no son correctos, o

simplemente no entrarlos al destino y dejarlos en algún lugar para que alguien los revise.

Veamos muy brevemente, mediante una serie de imágenes, un ejemplo de carga de una tabla en el *área de Staging* cuyo origen es una base de datos relacional:

En este caso, disponemos de una columna en el origen de datos que nos indica la fecha de última modificación realizada en cada fila. Lo cual debemos aprovechar en el diseño de nuestro proceso, leyendo del origen sólo las filas que han cambiado desde la última ejecución correcta de dicho proceso. Esto implica que deberemos dividir nuestro proceso en dos bloques, el que se encarga de realizar las inserciones y modificaciones, y el que se encarga de los borrados, con el fin de evitar que considere filas eliminadas en nuestro destino aquellas que no hemos leído del origen, porque no han sido modificadas desde nuestro último proceso de carga.

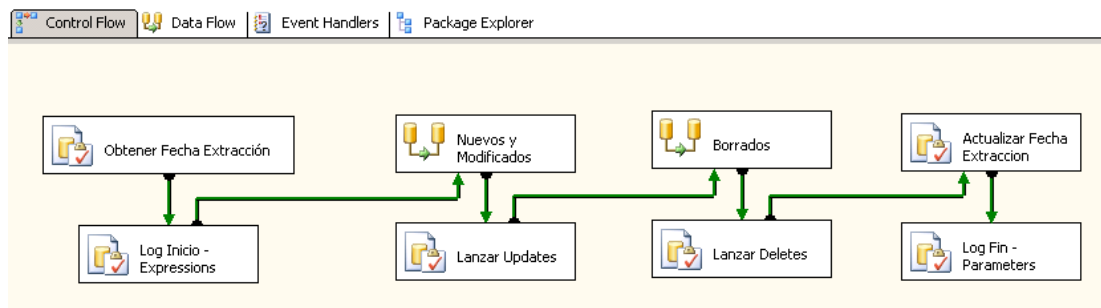


Imagen 4-1 Carga de una tabla en el área de Staging. Control Flow

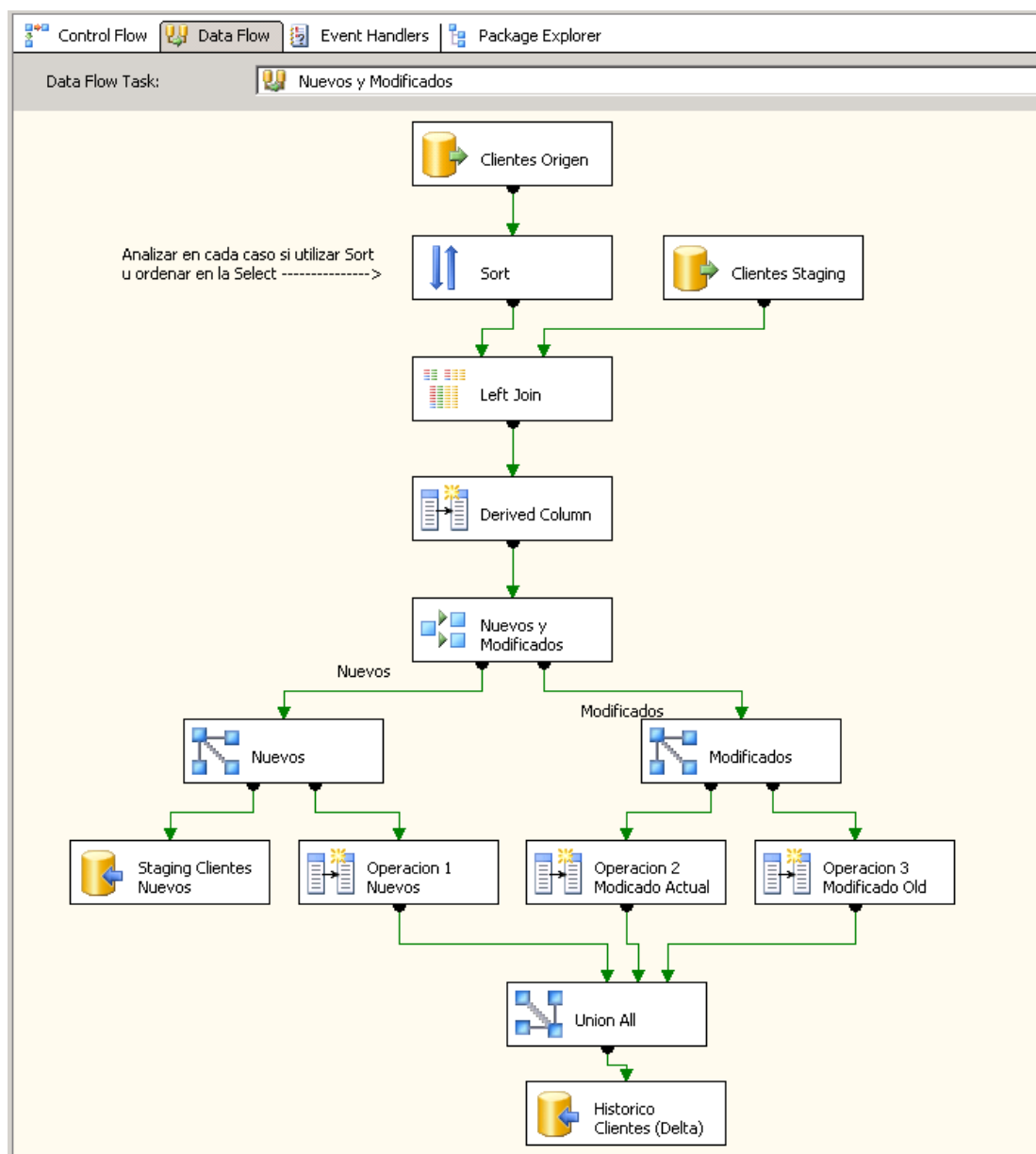


Imagen 4-2 Tratamiento de filas nuevas y modificadas – DataFlow

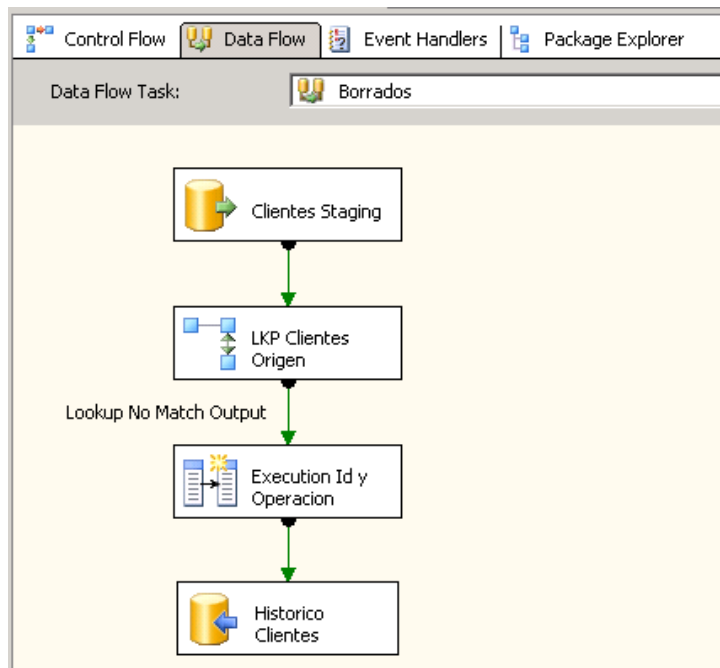


Imagen 4-3 Tratamiento de filas eliminadas – DataFlow

Casuísticas habituales

Hay una serie de casuísticas que son las que se suelen presentar con mayor frecuencia en la construcción de procesos ETL. A continuación vamos a describirlas brevemente y dar una serie de indicaciones sobre ellas, apoyándonos en sus características principales. Así como citar los pros y contras detectados en cada caso:

Ordenaciones

Sort

El componente *Sort* es muy útil para resolver ciertos casos, pero tiene el inconveniente de que hasta que no se han leído todas las filas no pueden ir fluyendo datos por los buffers hacia los siguientes componentes, ya que no podemos tener la certeza de que esa es la ordenación definitiva hasta que haya sido ordenada la última fila. Este componente puede aumentar el tiempo de ejecución de nuestros procesos, por lo que deberemos evaluar si tenemos otras alternativas, como obtener los datos ya ordenados del origen o del destino, y los pros y contras que esto puede suponer en cada caso concreto.

Evitando Joins en el origen

Siguiendo con las pautas marcadas anteriormente, una de las cosas que debemos evitar es consumir más recursos de los necesarios, tanto del origen como del

destino. Uno de los puntos en los que podemos conseguir estas mejoras es evitando la realización de *joins* complejas a la hora de obtener los datos. Siempre deberemos plantearnos hacer las *SELECTs* de las tablas por separado y hacer las *joins* en el servidor de SSIS, mediante los componentes que nos facilita para ello, que son:

Lookup

Permite acceder mediante unos datos de entrada a otra tabla, seleccionando la fila coincidente, y pudiendo devolver las columnas seleccionadas. Además permite configurar una caché que facilita el realizar estos procesos en memoria.

Merge Join

Permite combinar dos conjuntos de datos que previamente han sido ordenados. Los tipos de combinaciones permitidas son: *INNER*, *LEFT* o *FULL*.

Las principales diferencias entre ambos componentes que tendremos que tener en cuenta a la hora de utilizar uno u otro en nuestros diseños son:

- Las posibilidades de caché que nos ofrece el componente Lookup. En cada caso debemos estudiar si nos ofrecerá un mejor rendimiento o no el uso de esta caché.
- La necesidad de que los datos de origen vengan ordenados para el uso de Merge Join, y de que ambos utilicen el mismo *collation* en dicha ordenación (si se trata de datos alfanuméricos) para evitar obtener resultados inesperados. Mientras que Lookup no necesita que los datos le lleguen ordenados, aunque el que lo estén o no nos puede ayudar a la hora de optimizar el uso de la caché.

La decisión de uso de uno u otro componente no es sencilla. Incluso para resolver un mismo problema, dependiendo del volumen de datos que haya en cada una de las tablas que intervienen, del origen de cada uno de estos datos y de si está ordenado o no, hará que haya diferencias sustanciales de rendimiento según el componente que utilicemos.

Gestión del flujo de datos

Conditional Split

Nos permite que una fila vaya a una salida u otra en función del contenido de los datos. Es similar a una instrucción *CASE* de los lenguajes de programación.

Multicast y Union All

Multicast permite distribuir una entrada en una o más salidas, fluyendo por cada una de estas salidas una copia de los datos. Mientras que Union All permite justo lo contrario, es decir, unir varios de estos buffers de datos en uno sólo.

Transformaciones

En este caso nos vamos a centrar en el componente más completo y más ampliamente utilizado. Aunque hay otros como *Data Conversion*, *Character Map* o *Copy Column*, prácticamente todas sus funcionalidades están también incluidas en el componente Derived Column.

Derived Column

Permite crear nuevos valores para una columna, aplicando una serie de expresiones. Puede utilizar variables, operadores, funciones y columnas de entrada de la transformación. El resultado puede agregarse al flujo de datos como columna nueva o sustituir una de las columnas existentes. Integration Services tiene su propio conjunto de expresiones que pueden ser utilizadas en este componente, y que recomiendo al lector que estudie y conozca para aprovechar al máximo su uso.

Actualización de datos

Uno de los temas más importantes a tener en cuenta es el de la actualización de datos. Para ello básicamente tenemos dos componentes:

OLE DB Command

Ejecuta una instrucción SQL por cada fila del flujo de datos. Habitualmente se utiliza para actualización y borrado de filas. Es muy importante tener en cuenta que siempre debe incluirse una condición *WHERE* en dicha instrucción, para filtrar y hacer que los cambios realizados en la tabla afecten sólo a la fila en curso y no a todas las filas de la tabla.

SQL Task

Permite ejecutar desde un paquete cualquier instrucción SQL o procedimiento almacenado. Puede contener una o varias instrucciones. El código escrito tendrá que ir en el dialecto SQL correspondiente al gestor de bases de datos al que nos estamos conectando. Aunque es una tarea del *ControlFlow*, es conveniente citarla en este momento, dado su uso frecuente para la actualización de datos.

Hay casos, que bien por tratarse de un bajo volumen de datos, o bien por cualquier otro motivo en el que no nos preocupe el rendimiento, podemos optar por soluciones fila a fila. Aunque lo recomendable es hacer, siempre que sea posible, actualizaciones de conjuntos de datos, y no actualizaciones fila a fila. Para ello, la solución pasa por evitar el uso del componente *OLE DB Command* del *DataFlow*, y en su lugar, hacer inserciones masivas en una tabla temporal, para que una vez finalizada la ejecución del dataflow, en un siguiente paso se haga una actualización o borrado, mediante el componente del *ControlFlow* llamado *SQL Task*, haciendo join entre la tabla que queremos actualizar o borrar y esa tabla temporal que hemos creado en el *DataFlow*.

SCD (Slowly Changing Dimensions)

Anteriormente hemos citado este concepto. Ahora, ya centrados en *Integration Services*, nos queda por comentar que hay un componente en el *DataFlow* que, a modo de *wizard*, nos permite ir pasando por una serie de pantallas, configurando para cada una de las columnas qué tipo de cambios debe registrar (tipos 1 y 2), así como el tratamiento de miembros inferidos. Y una vez introducida esta información en el *wizard*, el componente “explota” generando una serie de cajitas adicionales con todos los componentes para hacer la gestión de cambios que hemos definido en él.

Básicamente le encuentro dos inconvenientes, el primero es, que si una vez diseñado el proceso, decido hacer cambios en alguno de los componentes que ha generado me deja hacerlo sin problemas. Pero si en un futuro necesito hacer cualquier modificación que implique volver a usar el wizard, éste eliminará todas las personalizaciones que habíamos realizado, volviendo a “explotar” tras finalizar la nueva ejecución del *wizard*, eliminando todo, y volviéndolo a generar sin tener en cuenta ninguna de las personalizaciones que habíamos incluido. Este inconveniente es importante saberlo, pero no tiene gran importancia, porque rara vez personalizamos lo que ha realizado el wizard.

El segundo, que sí que considero muy importante, es que es un componente con un muy mal rendimiento, y no es escalable en absoluto. Su principal problema es que todos los tratamientos los hace fila a fila, y en ningún caso orientado a conjuntos.

Mi recomendación es no utilizarlo, y proceder a diseñar nosotros mismos nuestros propios procesos SCD basados en el resto de componentes de la herramienta, haciendo un diseño que vaya orientado a conjuntos.

Otra alternativa es utilizar un componente de terceros. Hay uno del *Grupo Kimball* que funciona muy bien, y además, es gratuito. A estos componentes de terceros les veo en general dos inconvenientes: el primero es que si nos encontramos con un bug debemos esperar a que lo solucionen, y aun partiendo de que sea una empresa seria y de que se pondrá en ello, no tendremos una fecha fiable que poder

comunicar a nuestro cliente para la entrega del trabajo. Y, la segunda y principal, que nunca sabremos si en futuras versiones de *Integration Services* seguirá funcionando el componente o si sacarán nuevas actualizaciones para esas nuevas versiones del producto. Lo que nos deja con la incertidumbre de si en un futuro tendremos que rediseñar y rehacer todo el trabajo para poder migrar a una nueva versión de SQL Server.

Escritura de código VB.NET o C#

Hay dos elementos de Script, uno a nivel del *ControlFlow*, llamado **Script Task**, y otro a nivel del *DataFlow*, llamado **Script Component**. Ambos se utilizan para incluir código .Net (VB o C#) en los paquetes, y que además puede interactuar con las variables de dichos paquetes. Dando así una total flexibilidad a la hora de cubrir necesidades que no están incluidas en el resto de componentes y tareas del producto.

Uso de buffers de datos

Como ya conoce el lector, *Integration Services* a nivel de *DataFlow* trabaja con una serie de *buffers* por los que van fluyendo los datos y pasando por los diferentes componentes que hay incluidos en él. A continuación vamos a clasificar los componentes en tres tipos, en función de la utilización que hacen de estos *buffers*:

- **Streaming**: reutilizan el mismo buffer, la información pasa al siguiente componente a través del mismo buffer. Ejemplos: *Data Conversion, Derived Column, Lookup, Conditional Split*.
- **Bloqueo parcial**: copian los datos en un nuevo buffer, pero conforme se van procesando en los componentes van fluyendo dichos datos hacia el siguiente componente. Ejemplos: *Pivot, Un-Pivot, Merge, Merge Join, Union All*.
- **Bloqueo**: necesita todas las filas de entrada antes de que puedan continuar fluyendo los datos. También copian los datos a un nuevo buffer. Ejemplos: *Aggregate, Sort, Row sampling, Fuzzy Grouping*.

Es importante tener en cuenta estos detalles para hacer un buen diseño y conocer qué está ocurriendo a nivel de flujo de datos en cada uno de los componentes que utilizamos, y así facilitarnos nuestro objetivo de obtener el mejor rendimiento posible.

Conclusiones

En cualquier proyecto de BI, lo más habitual es que el principal foco de atención sea el *Data Warehouse*. Debemos optimizar su diseño todo lo posible, para que sea una fuente confiable de información, y responda al mayor número de preguntas de negocio posible. También debemos diseñar nuestros procesos ETL aplicando una serie de buenas prácticas que nos permitan que la información fluya desde el origen a él lo más rápido posible. Estas son las buenas prácticas que me gustaría destacar:

- Disponer de un *área de Staging*, que nos facilite la trazabilidad de los datos y la identificación de los cambios (qué, quién y cuándo han cambiado los datos).
- Intentar disponer de *tablas delta* (contienen sólo los cambios) siempre que sea posible.
- Hacer los procesos lo más escalables posibles, y con los menores consumos de recursos, tanto en el origen como en el destino, centrando la carga de trabajo en el servidor de *Integration Services*.
- Evitar actualizaciones fila a fila, e intentar que se hagan orientadas a conjuntos de filas.
- Gestionar adecuadamente el historial de cambios para reflejar la realidad del negocio, y no falsearla manteniendo sólo la última versión de los datos.
- Conocer a fondo cada uno de los componentes de *Integration Services*, para hacer un uso óptimo de la herramienta, y conseguir así mejorar todo lo posible el rendimiento.
- No olvidar en ningún caso la inclusión de procesos de limpieza y depuración de datos.

Demos incluidas en el curso. Carga de tablas de Dimensiones y de tablas Hechos

Si está siguiendo el curso, del cual este eBook es material complementario, acceda a los siguientes videos donde podrá ver una serie de procesos ETL ya orientados a plantear la resolución de forma didáctica de casuísticas habituales de carga de tablas de dimensiones y tablas de hechos:

‘**Demo SSIS 04A**’ donde se muestra el uso *Slowly Changing Dimension Transform* para cargar una tabla de dimensiones utilizando dicho componente. Anteriormente hemos comentado los inconvenientes de uso de este componente, pero hay casos en los que sí que puede ser utilizado y es recomendable que lo conozca con detalle.

‘**Demo SSIS 04B**’ muestra algunas casuísticas habituales, como la carga de una dimensión *parent-child*, carga de dimensiones de un modelamiento *Snowflake*, y carga de una dimensión *Fecha*.

‘**Demo SSIS 04C**’ muestra otros ejemplos de carga de dimensiones, en las que se tienen en cuenta la existencia de atributos con diferentes tipos de cambios, pero en vez de utilizar la *SCD Transform*, como se hizo en la ‘Demo SSIS 04A’, se hace un desarrollo personalizado utilizando otros componentes.

‘**Demo SSIS 05A**’ donde se realiza la carga de una tabla de hechos, representando una casuística muy habitual en estos casos, que es la importación de datos de un origen, haciendo los *Lookups* y validaciones pertinentes para obtener las claves

Microsoft Business Intelligence: vea el cubo medio lleno

subrogadas a cada una de las tablas de dimensiones y el almacenamiento en el destino (tabla de hechos de nuestro modelo dimensional).

‘**Demo SSIS 05B**’ gestiona diferentes granularidades de las tablas de hechos mediante el uso del *Conditional Split Transform*.

‘**Demo SSIS 05C**’ donde se muestra el uso de las tareas que tiene *Integration Services* para el procesamiento de dimensiones y cubos, para así cerrar el ciclo de un proceso típico de ETL de un modelo dimensional, que consiste en cargar las tablas de dimensiones, posteriormente cargar las tablas de hechos, y finalmente procesar los componentes de *Analysis Services* necesarios para que incluyan la misma información que el origen de datos relacional que acabamos de cargar.

Laboratorio incluido en el curso. Carga de tablas de Dimensiones y de tablas de Hechos

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al material de los siguientes laboratorios:

‘**Lab SSIS 04A**’ Creando paquetes para la carga de tablas de dimensiones.

‘**Lab SSIS 05A**’ Creando paquetes para la carga de tablas de hechos.

Realice paso a paso todo lo allí expuesto y responda a las preguntas que se incluyen. Este Lab le ayudará a asentar los conocimientos adquiridos sobre ETL de tablas de dimensiones y de tablas de hechos. Y por supuesto, si tiene cualquier duda sobre lo visto o cualquier situación que quiera resolver puede utilizar las sesiones de **tutorías** y los **foros** del curso para resolverlas.

Logging. Registro de actividad de los paquetes

Para cerrar el ciclo y tener una solución implementada aprovechando todas sus características, vamos a mostrar las posibilidades que nos ofrece *Integration Services* para registrar la actividad de nuestros paquetes cuando son ejecutados y poder hacer un seguimiento de todo lo que ha ido ocurriendo en dichas ejecuciones.

En más de una ocasión necesitaremos saber qué ha ocurrido durante la ejecución de un paquete. Normalmente si la ejecución va bien y nos cuadran los datos es posible que no hagamos mucho caso de esta opción de configuración, pero cuando las cosas no salen bien resulta muy útil poder revisar qué ha ocurrido, paso a paso y con el nivel de detalle deseado, en la ejecución de cada uno de los componentes del paquete. Por otro lado debido a un cambio en este área en la versión 2008 de *Integration Services* veremos donde se está ubicando esta información ahora cuando elegimos como destino un proveedor de SQL Server.

Habilitar el Registro (Logging)

En primer lugar para poder utilizar esta opción tendremos que habilitar el registro (*logging*) a nivel de paquete en nuestra solución, para poder llegar a esta opción podemos hacer clic con el botón derecho del ratón sobre cualquier zona libre del área de trabajo del flujo de control y elegir Registro (*Logging*).

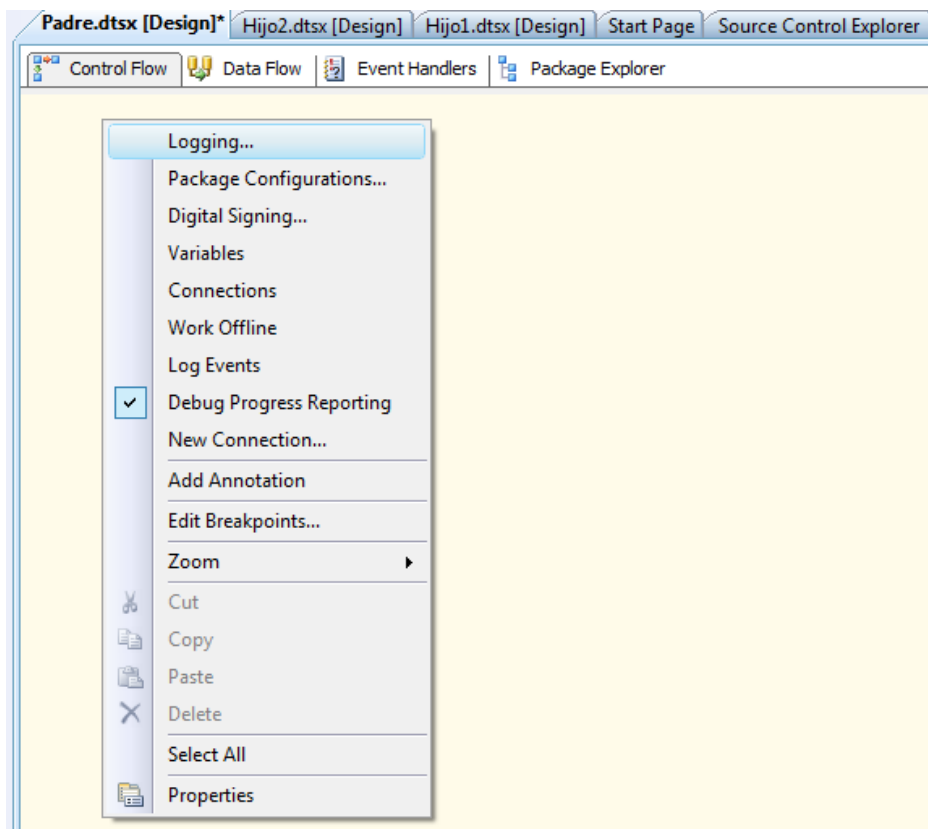


Figura 4-7 Habilitar Registro (Logging)

Selección del tipo de proveedor

Tras seleccionar la opción de registro vemos que podemos, en primer lugar, añadir un proveedor donde almacenar la salida que genera la ejecución del paquete, los proveedores válidos son:

- Proveedor para el Visor de eventos de Windows
- Proveedor para archivos de texto plano
- Proveedor para archivos XML
- Proveedor para *SQL Server*
- Proveedor para SQL Server Profiler

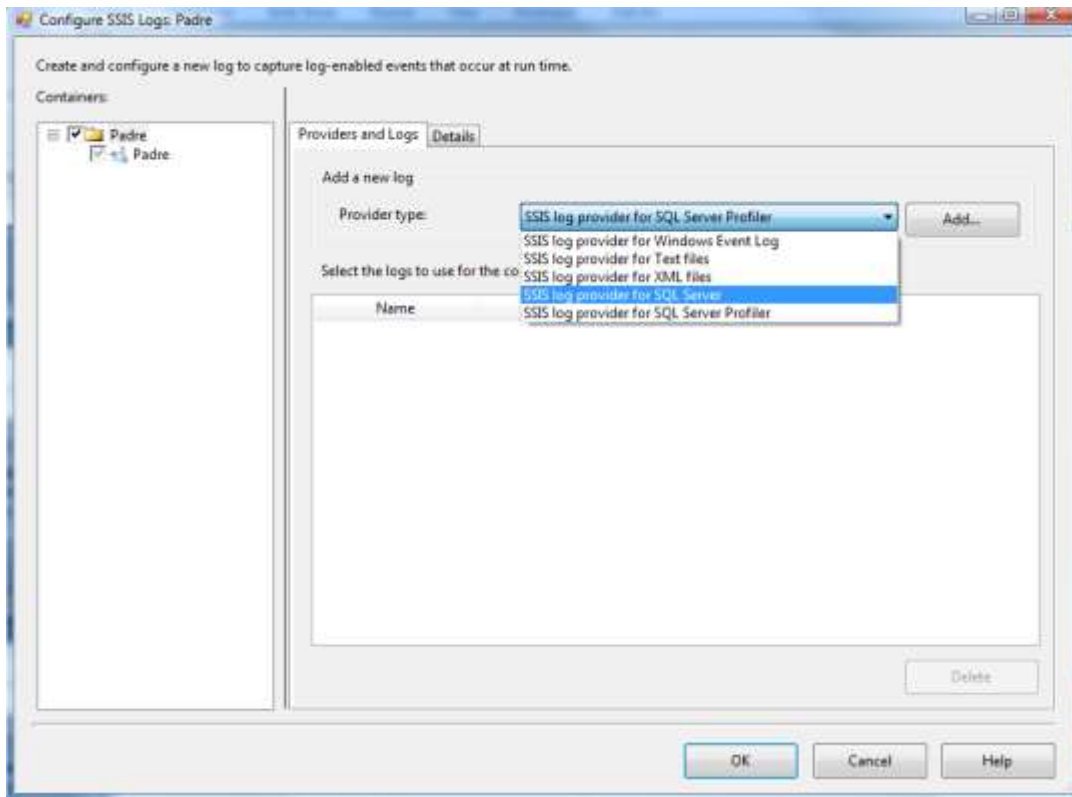


Figura 4-8 Selección de Tipo de Proveedor

Debe tenerse en cuenta que el hecho de registrar actividad añade cierto grado de penalización que puede variar dependiendo del nivel de detalle elegido y del proveedor.

En nuestro caso vamos a registrar la actividad de nuestro paquete en un proveedor de *SQL Server*. Por lo tanto seleccionamos del desplegable '*SQL Server*' y pulsamos sobre agregar. En la opción de conexión creamos una conexión nueva a una base de

datos de una instancia que tengamos designada con el propósito de almacenar esta información, para el ejemplo se proporciona una base de datos de *SQL Server 2008* de nombre 'Registro'.

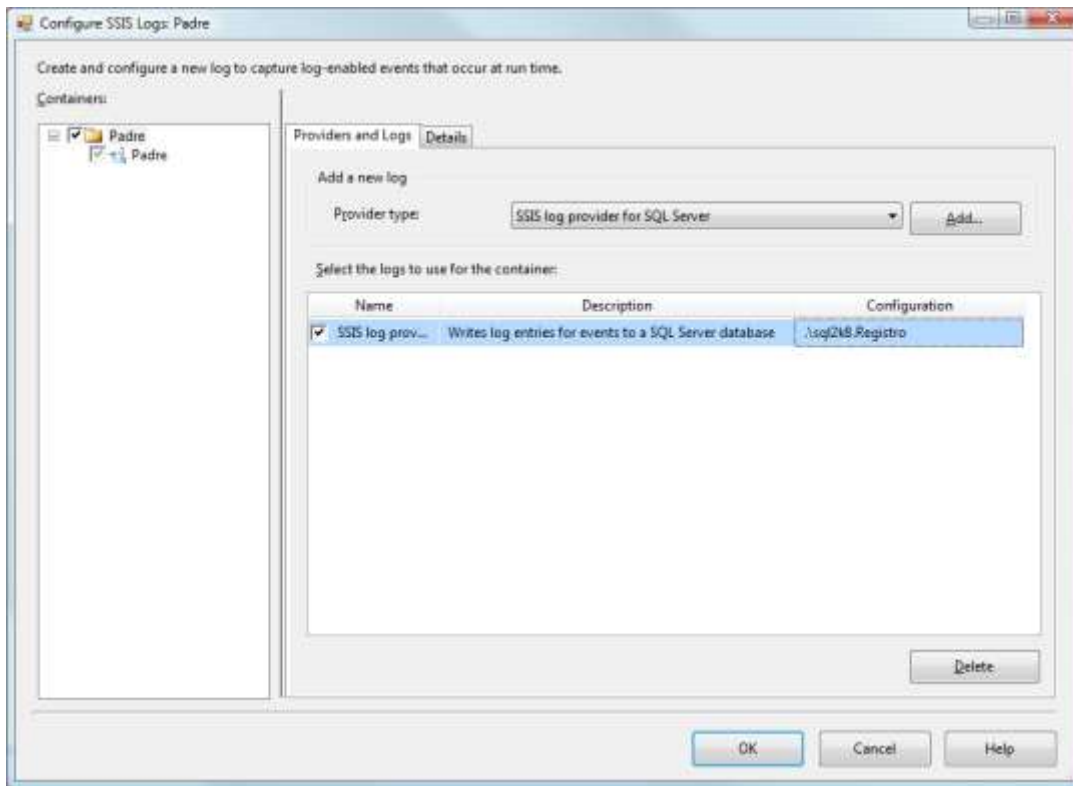


Figura 4-9 Selección de Conexión de Configuración

Personalizando la información de Registro

Otra característica importante es la posibilidad de personalizar la configuración de forma individual para cada elemento que incorporemos en nuestro flujo de control. Además podemos especificar qué eventos queremos que queden registrados, para esto nos situamos en la pestaña de detalles y elegimos el nivel de detalles deseado, vemos que podemos elegir entre opciones con varias severidades y fases de ejecución de cada tarea.

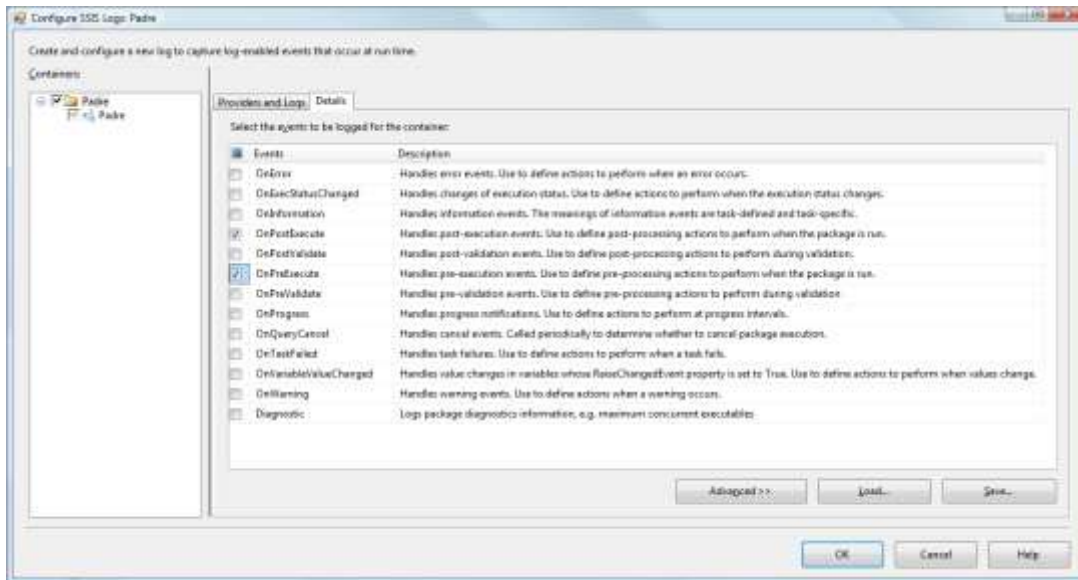


Figura 4-10 Selección de eventos

Las opciones avanzadas nos permiten personalizar aún más el nivel de detalle permitiéndonos incluir nombres de máquina, operador, identificador de ejecución Identificador de origen, entre otras cosas, por cada posible evento.

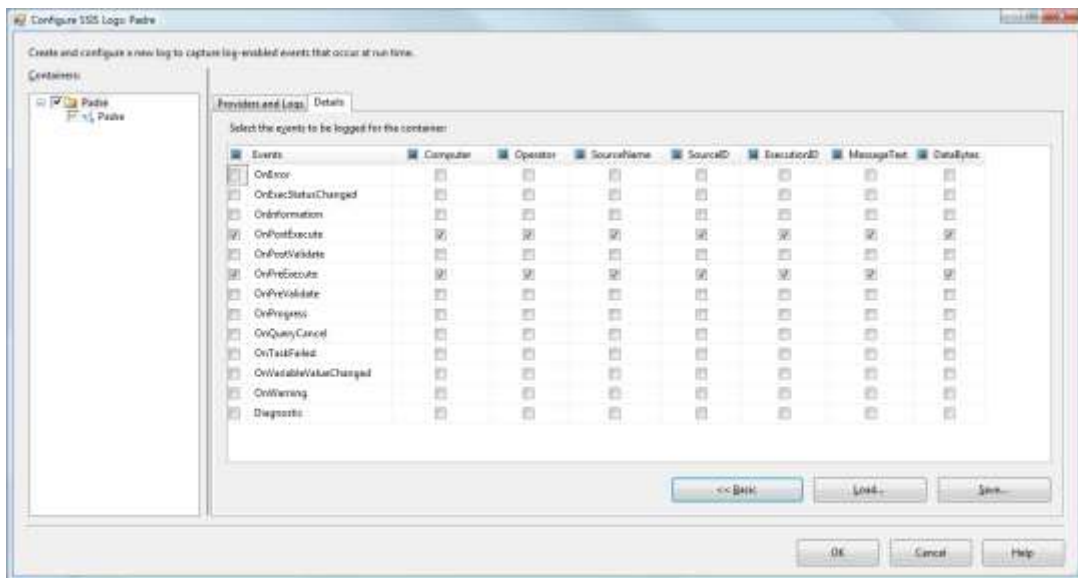


Figura 4-11 Personalización de columnas de cada evento

Los otros dos botones de que dispone la ventana, "Save" y "Load", nos sirven para almacenar esta configuración de registro y poder aplicarla posteriormente a otros paquetes sin necesidad de redefinir todo el proceso, simplemente cargaríamos el fichero de configuración que se genera al salvar la configuración.

Después de pulsar sobre 'Ok' la configuración de registro se encuentra almacenada en el paquete, pero la tabla no se creara hasta que el paquete no se ejecute al menos una primera vez. Y aquí una diferencia importante con respecto a SQL Server 2005, la tabla de registro cambia su ubicación y su nombre. Después de la ejecución del paquete, si todo ha funcionado bien, la encontraremos en la base de datos designada, en la carpeta de tablas de sistema, con el nombre '*dbo.sysssislog*' con el siguiente formato:

id	event	component	operation	source	success	error_code	error_message	start_time	end_time	duration	status	message
1	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
2	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
3	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
4	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
5	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
6	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
7	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
8	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
9	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
10	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
11	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
12	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
13	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
14	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
15	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
16	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
17	OnPkgInit	Pkg1	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	
18	OnPkgInit	Pkg2	Initialize	Package	True	0		2009-05-21 18:21:18.21	2009-05-21 18:21:18.21	0.00	Ok	Beginning of package execution: Based on the system configuration, B...

Figura 4-12 Estructura de la tabla '*dbo.sysssislog*'

Solo recordar que este comportamiento ha cambiado desde SSIS 2005 que antes utilizaba el nombre '*dbo.sysdts90log*' y la tabla no era considerada tabla de sistema.

Interactuando con el Registro mediante Scripts

Sobre registros, un apunte más antes de terminar, ya que además de las opciones que tenemos para registrar las opciones predefinidas podemos registrar información definida por el usuario mediante un *Script Component* allí donde los necesitemos. El proceso sería el siguiente , arrastramos a nuestra solución un *Script Component* que configuraremos en lenguaje *VB.NET* (también se puede hacer *C#*). En primer lugar, a nivel del *ControlFlow*, ubicamos la *tarea de Script* en el lugar donde nos interese saber qué valor tiene la variable. En este caso, como se trata de un ejemplo simple, está justo detrás de la tarea que invoca a otro paquete.

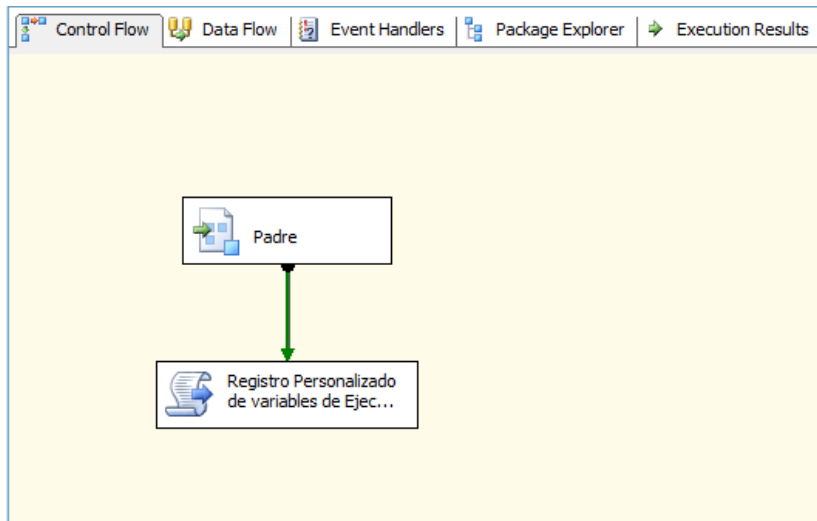


Figura 4-13 Ubicación del Script task

La tarea está configurada de la siguiente forma, en las variables de sólo lectura se incorpora la variable que nos interesa que quede registrada, en el ejemplo la variable se llama 'Variable'.

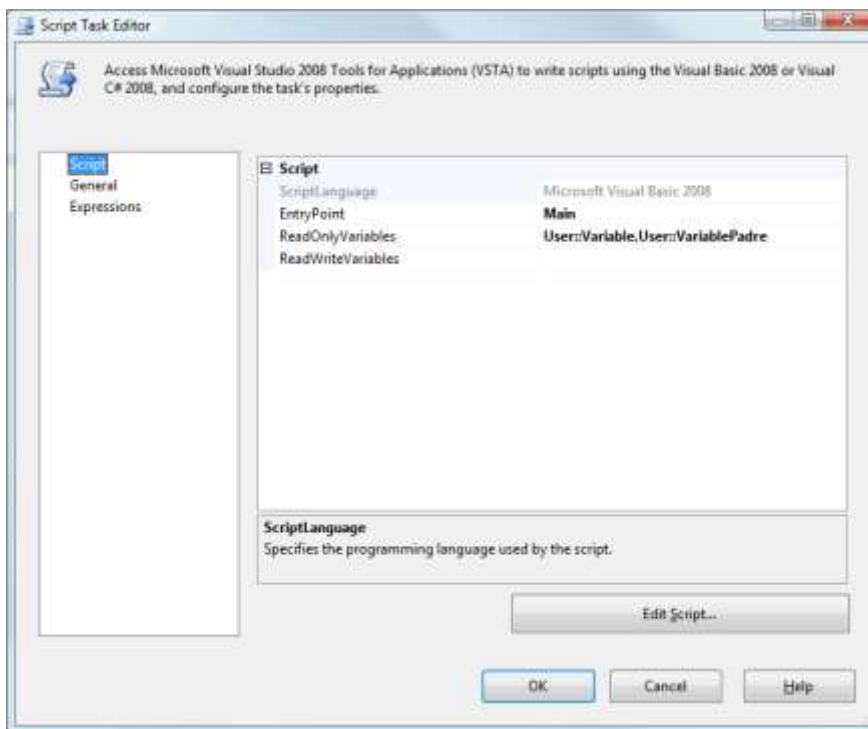


Figura 4-14 Variables de sólo lectura

En el script editaremos el código de la siguiente forma haciendo referencia a la variable y el mensaje será personalizado para poder identificarlo más tarde en la tabla.

```
Public Sub Main()  
.  
Dim databytes(0) As Byte  
  
Dts.Log("Mensaje personalizado:" &  
Dts.Variables("Variable").Value.ToString,1,databytes)  
.  
Dts.TaskResult = ScriptResults.Success  
End Sub  
End Class
```

También podemos visualizar varias variables mediante el mismo método concatenando:

```
Public Sub Main()  
.  
Dim databytes(0) As Byte  
Dts.Log("Mensaje personalizado : " & Dts.Variables("Variable").Value.ToString & " " &  
Dts.Variables("VariablePadre").Value.ToString, 1, databytes)  
.  
Dts.TaskResult = ScriptResults.Success  
End Sub  
End Class
```

Adicionalmente hay un paso más, volviendo a la configuración hay que habilitar el nuevo componente que aparecerá solo para los componentes de este tipo 'ScriptTaskLogEntry' para que muestre la personalización del Script.

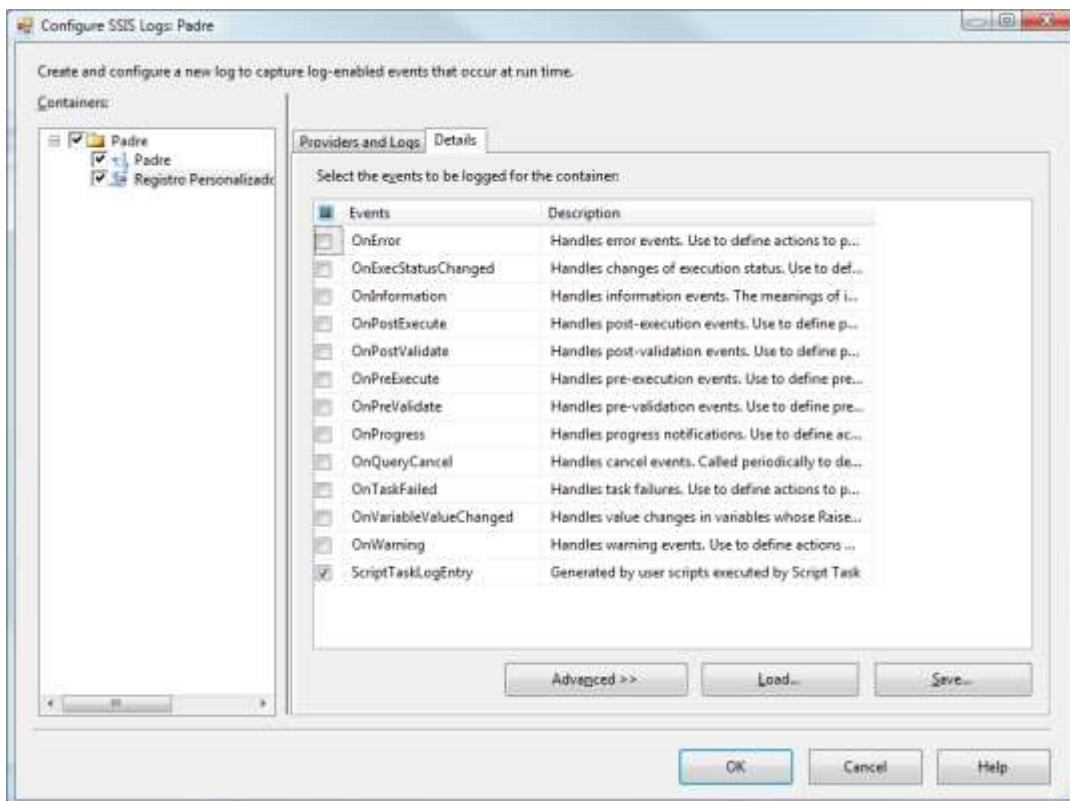
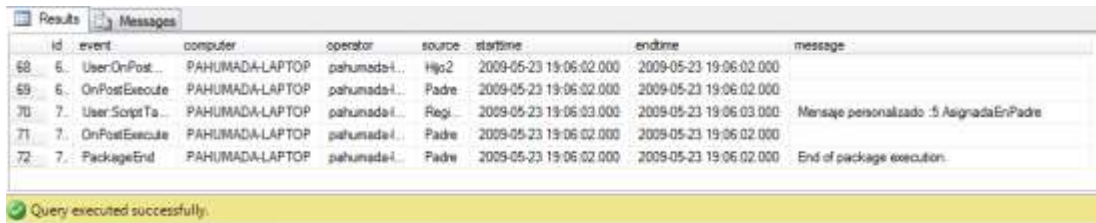


Figura 4-15 Habilitar el Registro personalizado

Una vez ejecutado el paquete con esta configuración la tabla de salida mostrará el mensaje personalizado en el punto que nos interese con los valores de las variables.



id	event	computer	operator	source	starttime	endtime	message
68	6. UserOnPost...	PAHUMADA-LAPTOP	pahumada-i...	Hijo2	2009-05-23 19:06:02.000	2009-05-23 19:06:02.000	
69	6. OnPostExecute	PAHUMADA-LAPTOP	pahumada-i...	Padre	2009-05-23 19:06:02.000	2009-05-23 19:06:02.000	
70	7. User Script Ta...	PAHUMADA-LAPTOP	pahumada-i...	Regi...	2009-05-23 19:06:03.000	2009-05-23 19:06:03.000	Mensaje personalizado :5 AsignadaEnPadre
71	7. OnPostExecute	PAHUMADA-LAPTOP	pahumada-i...	Padre	2009-05-23 19:06:02.000	2009-05-23 19:06:02.000	
72	7. PackageEnd	PAHUMADA-LAPTOP	pahumada-i...	Padre	2009-05-23 19:06:02.000	2009-05-23 19:06:02.000	End of package execution.

Figura 4-16 Salud de Registro de Configuración personalizable

Conclusiones

Hemos visto cómo poder registrar y personalizar información que genera *Integration Services* durante la ejecución de los paquetes. Se ha comprobado que, de una forma muy sencilla, podemos registrar mucha información que nos ayudará analizar cualquier problema relacionado con la ejecución de los paquetes, pero debe tener en cuenta que además de esta información, como se indicó en la sección de *Buenas prácticas*, podemos registrar información adicional sobre los datos que nos permitan auditarlos.

Mediante el Registro (*Logging*), podemos por ejemplo saber si una tarea finalizó correctamente, cuando se inició, cuando finalizó, quién la ejecutó, y otra información sobre los distintos eventos que vimos anteriormente, pero no nos permitirá tener información de cosas como: cuándo entró una fila, cuándo se modificó, cuando se borró, cuántas filas de una tabla de hechos no se encontraron en la tabla de dimensiones, y largo etcétera, ya que estos son datos que debemos generar nosotros en el propio diseño de nuestros paquetes y crear nuestra propia estructura de tablas para almacenarlos.

La utilización del Registro y de nuestra propia auditoría son elementos complementarios, y la unión de ambos nos da una gran potencia y flexibilidad a la hora de saber que ha ocurrido.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda a los siguientes videos donde podrá ver la utilización de la mayor parte de los componentes descritos anteriormente:

‘**Demo SSIS 06A**’ donde se muestra la activación del *Logging* y su configuración en un paquete de ejemplo, para que registre información de una serie de eventos. Así como la ejecución de dicho paquete y la muestra de los datos almacenados en la tabla ‘*dbo.sysssislog*’.

Configuración, despliegue y administración de paquetes

Una vez que hemos creado y depurado un paquete de *Integration Services*, debemos proceder a desplegarlo sobre el servidor, o posiblemente sobre varios servidores. Es habitual encontrarse con entornos de desarrollo, pruebas, preproducción y producción. En cada uno de ellos, los orígenes y destinos de nuestros datos pueden ser diferentes. La primera alternativa que tenemos es editar con BIDS (*Business Intelligence Development Studio*) cada uno de estos paquetes en el servidor correspondiente y actualizar los valores de las propiedades de cada uno de los componentes que accedan a datos, para que lo haga sobre el servidor que corresponda y con el usuario y password que tiene en dicho servidor. Esto puede ser una tarea pesada y repetitiva, y sobre todo poco flexible. Lo ideal es que este tipo de información sea independiente del paquete y se pueda cambiar sin necesidad de abrir dicho paquete y sin el uso de herramientas como Visual Studio.

Para ello, tenemos la configuración de paquetes *Integration Services*, que nos permite actualizar las propiedades que indiquemos previamente, durante su ejecución. Con esto obtendremos una serie de ventajas que harán que el mover los paquetes entre diferentes servidores, o incluso el que cambie un origen o un destino de los datos, lo podamos gestionar sin tener que editar el paquete con Visual Studio. Además hay que tener en cuenta que en muchas ocasiones quien gestiona dichos paquetes es un DBA o un operador que no tiene por qué estar familiarizado con herramientas de desarrollo, ni tiene por qué conocer el contenido del paquete, es más, en algunos casos ni debería tener acceso a ese contenido.

Con dicha configuración de paquetes, conseguimos que para cualquier cambio de los citados anteriormente, como por ejemplo, el paso de un entorno de desarrollo a otro de producción, sea mucho más simple, ya que sólo tenemos que acceder a la configuración y cambiar las cadenas de conexión. Por otro lado, podemos obtener una mayor flexibilidad, por ejemplo, podríamos cargar estos valores en variables y utilizarlos en un script task.

A la hora de almacenar esta configuración, tenemos diversas alternativas: archivos XML, variables de entorno, en el registro de Windows, en la configuración del paquete padre, o en una tabla de SQL Server. En este caso, la experiencia nos dice que lo más habitual es el almacenamiento en archivos XML y en tablas de SQL Server, en cualquiera de ellos podemos almacenar en un solo archivo o tabla la información de múltiples paquetes.

Los archivos XML son muy flexibles, pero en contraposición pueden ser fácilmente accesibles con cualquier editor, lo que puede suponer algún riesgo adicional para la información que contienen. Por supuesto que podemos tomar todas las medidas de seguridad que tenemos en el sistema de archivos. Otra alternativa es almacenar la información en una tabla de SQL Server, lo que supone un trabajo mínimo, aunque es algo menos flexible que los archivos XML, por ejemplo, si lo que necesitamos es

cambiar la ubicación de dicha tabla de configuración (pasarla a otra base de datos, o pasar la base de datos donde se encuentra a otro servidor) el paquete dejará de funcionar correctamente.

Una buena alternativa, que une las ventajas de ambas formas de almacenamiento, es crear un archivo XML en el cual no se almacenará información sobre los orígenes y destinos de datos que necesita el paquete, ni cualquier otra información de configuración que necesitemos almacenar. Simplemente se almacenará la cadena de conexión a la base de datos de configuración. Y ya en la tabla de SQL Server, se almacenará el resto de información de configuración del paquete, con todas las ventajas de seguridad que tenemos en el servidor, control de acceso a la base de datos, a la tabla y a las filas de ésta, así como la posibilidad de usar certificados, encriptación, etc.

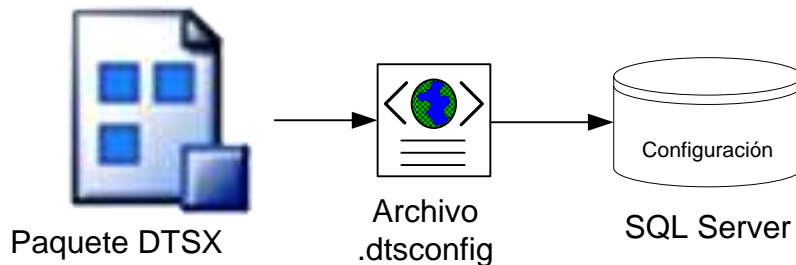


Figura 4-17 Acceso a datos de configuración

Paquete y casuística de ejemplo

Bien, pues una vez vista la teoría, pasemos a la práctica, vamos a hacer un paquete muy simple, que lea de la tabla *Employee* de la base de datos *AdventureWorks* y que grabe el contenido, sin más transformaciones, en una base de datos de destino, que llamaremos *Pruebas*, en una tabla llamada *DestinoEmpleados*. Ambas tablas tienen las mismas columnas, se encuentran en el mismo servidor, y en dicho servidor también se va a ejecutar el paquete. También tendremos una base de datos llamada *Configuración*, que será la que tenga una tabla llamada *'dbo.[SSIS Configurations]'* donde almacenaremos las configuraciones de los paquetes, entre otra información.



Figura 4-18 Entorno de Desarrollo

A continuación se incluyen unas imágenes sobre el paquete en cuestión:



Figura 4-19 Paquete "Importar Empleados"

La tarea SQL "Truncar DestinoEmpleados" ejecuta la instrucción
`TRUNCATE TABLE dbo.DestinoEmpleados`

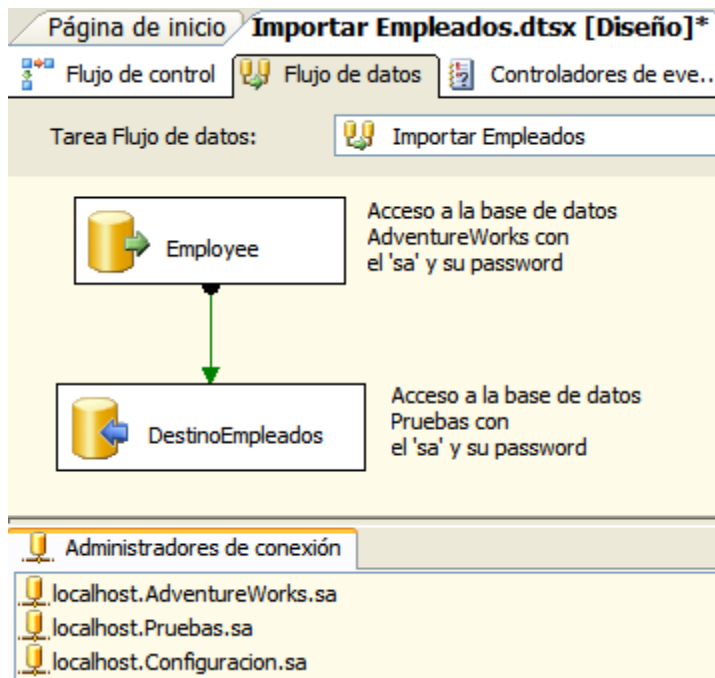


Figura 4-20 DataFlow del paquete "Importar Empleados"

El *OleDb Source* "Employee" ejecuta la siguiente instrucción:

```
SELECT * from HumanResources.Employee
```

*No es una buena práctica el uso de "SELECT * ...", es mejor indicar el nombre de las columnas explícitamente. Pero aquí lo hemos hecho por simplificar el ejemplo.*

Supongamos que en el entorno de producción tenemos un servidor dedicado a *Integration Services* en el que se ejecutará el paquete y que además tendrá una base de datos llamada *Configuracion* que almacenará la información de la configuración de los paquetes que se desplieguen en ese servidor, un servidor para el origen de datos al que accede nuestra aplicación OLTP que actualiza la base de datos AdventureWorks, y un tercer servidor donde se encuentra la base de datos Pruebas. Para que el paquete funcione en este nuevo entorno tendremos que hacer ciertos cambios en la configuración. Además se accederá a los datos con diferentes usuarios ya existentes en el nuevo entorno.

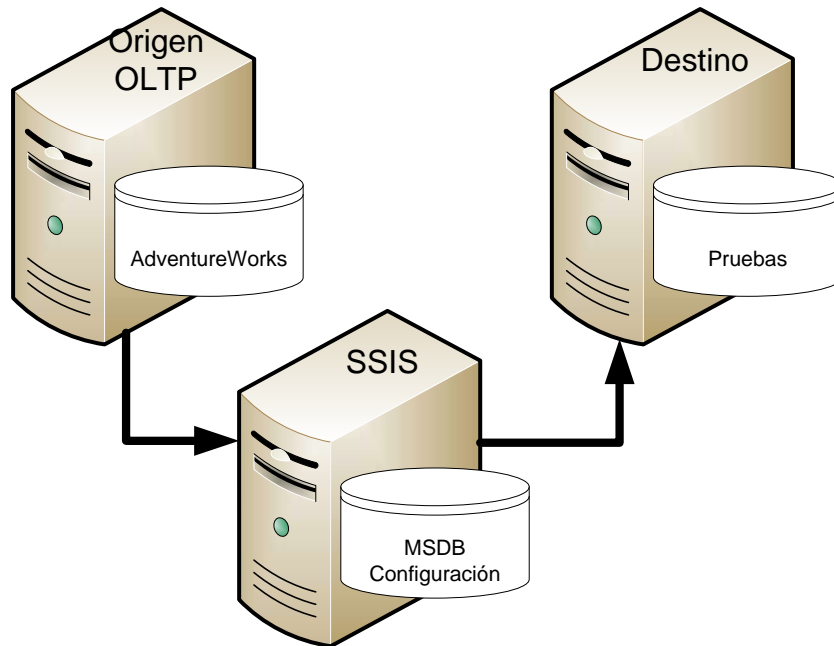


Figura 4-21 Entorno de Producción

Creación de la configuración del paquete

Vamos a comenzar a generar toda la información de configuración necesaria y a almacenarla, tanto en el archivo XML como en SQL Server, según lo visto anteriormente.

A continuación vamos a ir realizando paso a paso las tareas de configuración del paquete. Vamos al menú '*SSIS*' y elegimos la opción '*Configuraciones de paquetes...*'. En el formulario que aparece, habilitamos el CheckBox '*Habilitar configuraciones de paquetes*', y pulsamos el botón '*Agregar*', accediendo así al asistente de configuración de paquetes. En dicho asistente, pulsamos '*Siguiente*' en la primera pantalla de presentación. Nos encontramos en la pantalla de selección de tipo de configuración, aquí seleccionamos '*Archivo de configuración XML*' en el ComboBox, y damos el nombre '*Prueba01.dtsconfig*', almacenándolo en la misma ruta donde tenemos el paquete, tal y como se muestra en la siguiente imagen.

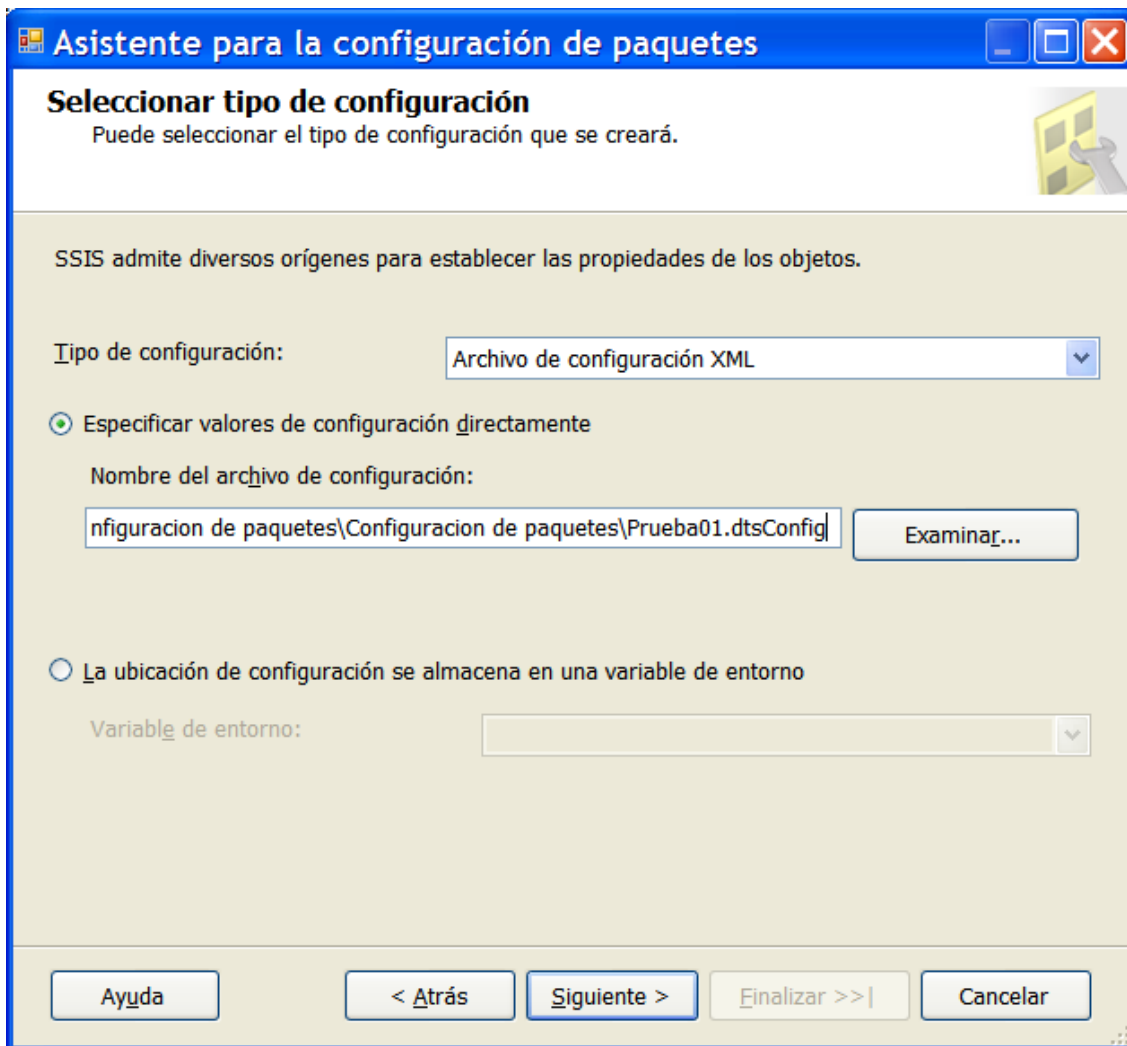


Figura 4-22 Seleccionar tipo de configuración, archivo XML

En la pantalla de selección de propiedades a exportar, elegimos 'ConnectionString' de la cadena de conexión a la base de datos 'Configuracion'.

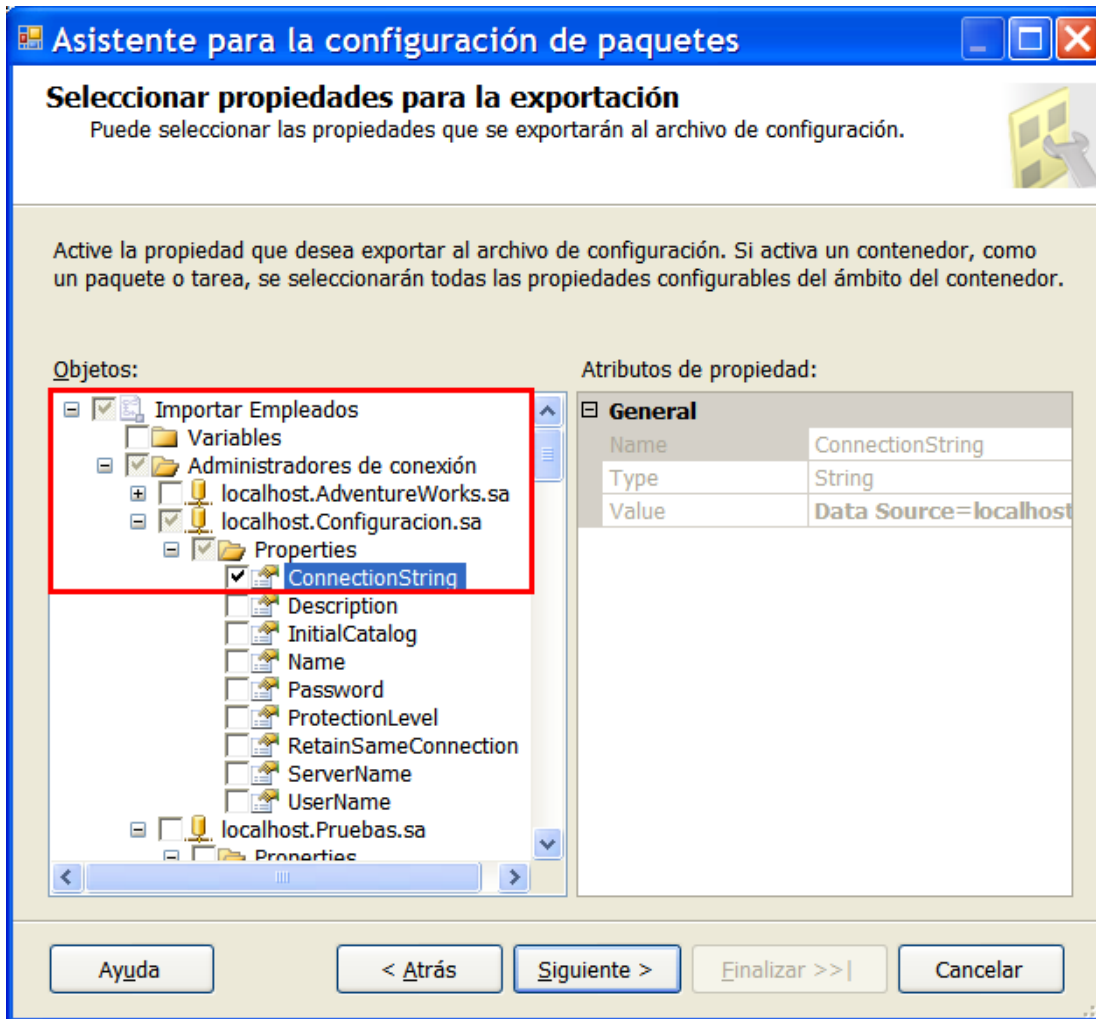


Figura 4-23 Selección de propiedades a almacenar en el archivo configuración, archivo XML

En la siguiente pantalla asignamos el nombre '*XML con conexión a Configuración*', y pulsamos '*Finalizar*'. Ya tenemos configurado el primer archivo XML, en el que, simplemente, vamos a almacenar la cadena de conexión a la base de datos '*Configuración*', que contendrá el resto de la información. Es conveniente, que dicha cadena de conexión utilice autenticación Windows, para no almacenar allí información que facilite el acceso a dicha base de datos. En nuestro caso, y sólo a modo de ejemplo, para el entorno de desarrollo también hemos utilizado el login 'sa' en esta cadena de conexión, pero eso no podremos hacerlo, bajo ningún concepto, en el entorno de explotación.

Hay que remarcar que este fichero XML puede ser usado por otros paquetes, evitando crear uno por cada paquete, y además conteniendo la misma información, ya que la base de datos de configuración se llama igual. Simplemente, daremos el mismo nombre a la hora de crearlo.

Vamos a repetir todos los pasos, ahora para grabar en la base de datos 'Configuracion' en la tabla 'dbo.[SSIS Configurations]', para ello, pulsaremos de nuevo el botón 'Agregar', accediendo de nuevo al asistente de configuración de paquetes. En la pantalla 'Seleccionar tipo de configuración', ahora elegiremos como tipo de configuración 'SQL Server', en conexión elegimos 'localhost.configuracion.sa', y en tabla de configuración pulsamos el botón 'Nueva', revisamos el código de creación de la tabla, y sin cambiar nada, pulsamos el botón 'Aceptar'. En filtro, asignamos el valor 'Conexiones01'.

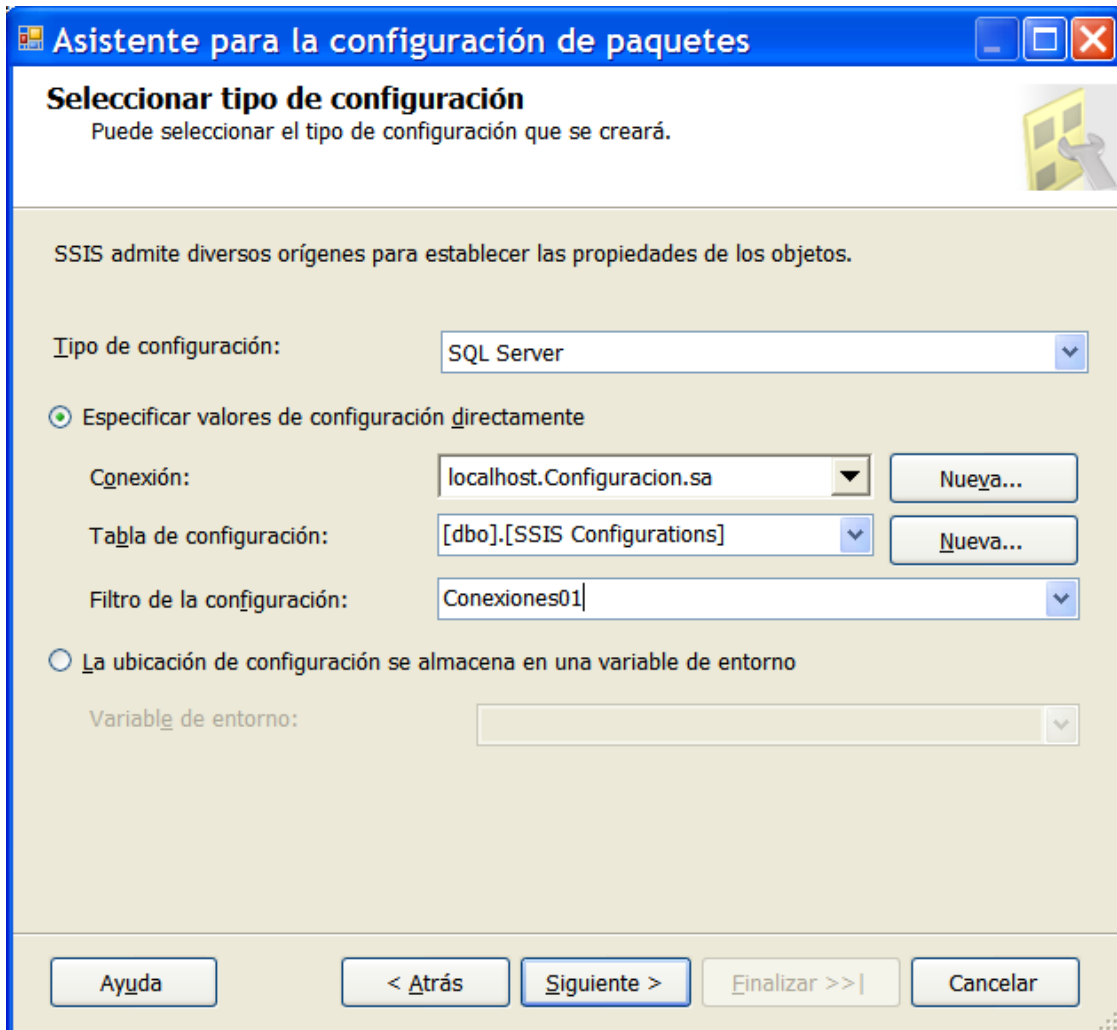


Figura 4-24 Seleccionar tipo de configuración, SQL Server

En este caso, lo que vamos a seleccionar, son las cadenas de conexión a los servidores de origen y destino de los datos.

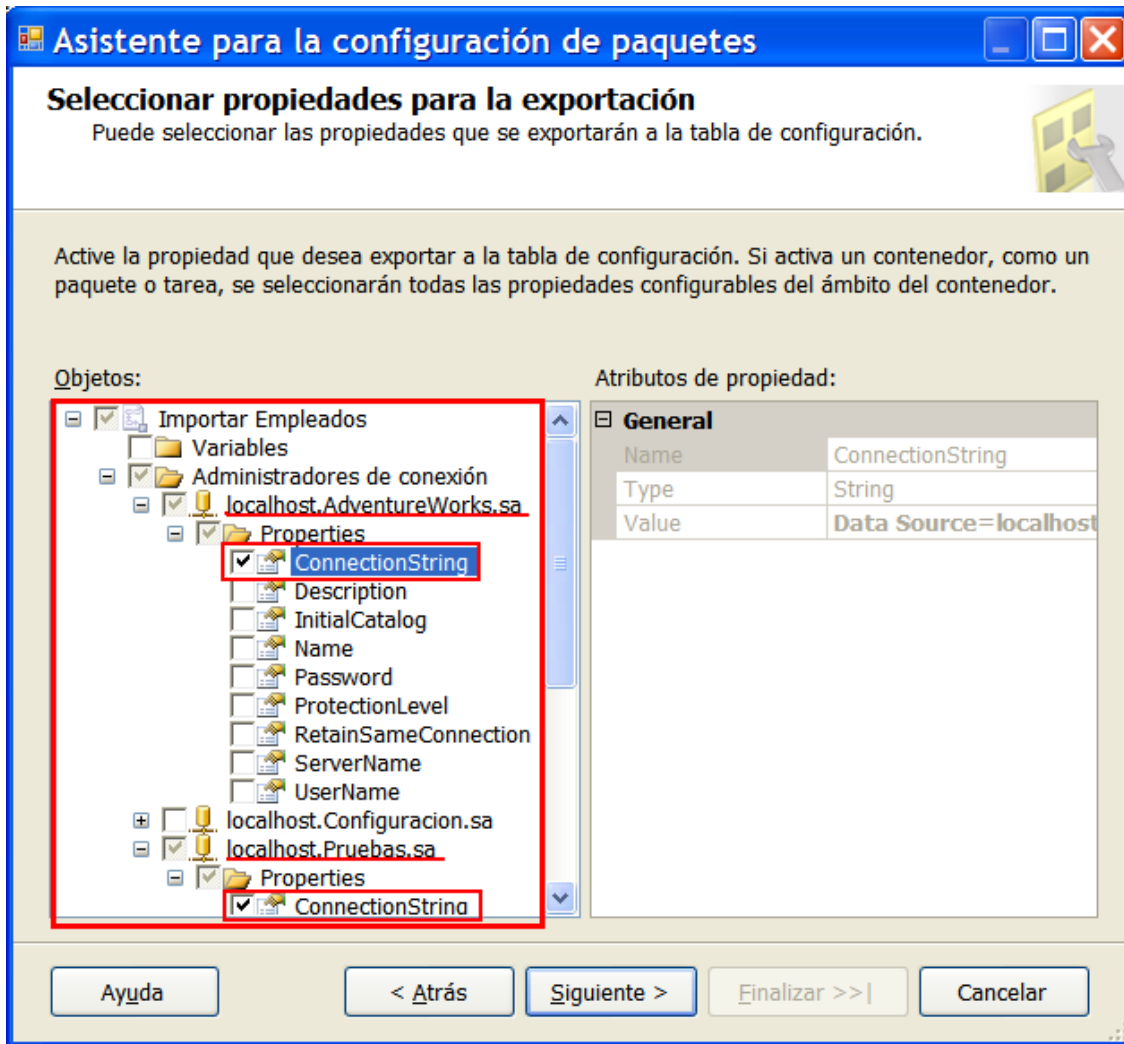


Figura 4-25 Selección de propiedades a almacenar en el archivo de configuración, SQL Server

Obteniendo como resultado final la creación de ambas configuraciones de paquetes, y el orden en que se utilizarán. En nuestro caso, en primer lugar, se leerá el archivo XML, y en segundo lugar, el resto de la configuración.

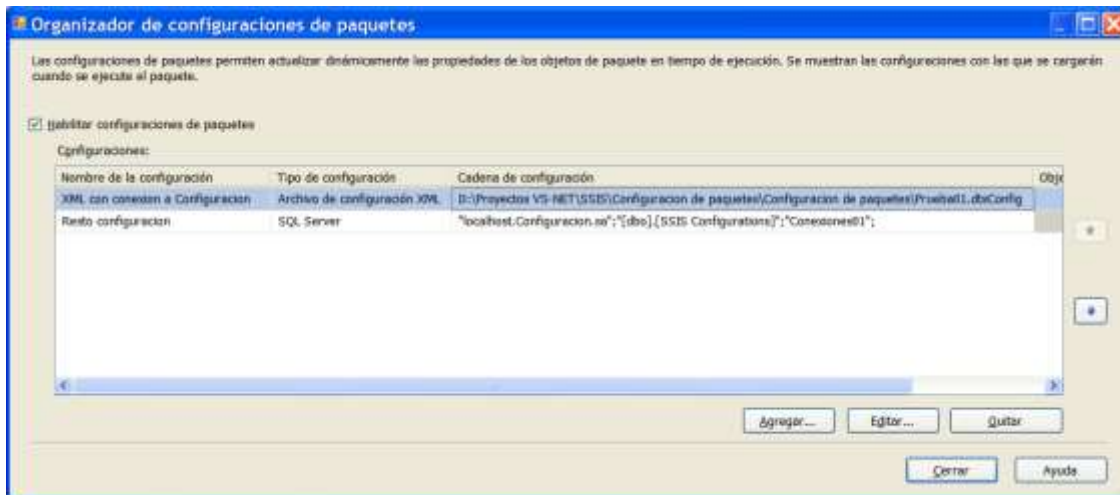


Figura 4-26 Resultado del organizador de configuraciones de paquetes



Figura 4-27 Información almacenada en la tabla "dbo.[SSIS Configurations]"

Al igual que con el archivo XML, podemos reutilizar esta información para diferentes paquetes, si van a utilizar conexiones comunes, y evitar duplicarla. Para ello, deberíamos crear una tercera configuración para conexiones específicas, aunque este punto lo dejamos en manos del lector, no lo vamos a desarrollar aquí por cuestiones de espacio.

Adicionalmente, podemos aprovechar esta base de datos, 'Configuraciones', para incluir en ella información de *logging* de ejecución de los paquetes, y así reutilizar la misma conexión.

Nota: nunca se almacena la contraseña de la cadena de conexión, ni en el archivo XML, ni en las filas de la tabla de configuración. Es por seguridad, por tanto el paquete no se va a poder ejecutar correctamente. Hay que editar el archivo o actualizar la fila de la tabla con el *ConnectionString* correcto, es decir, agregarle '...;password=xxxx;...'. Por supuesto, es recomendable, siempre que sea posible, utilizar autenticación integrada, con lo que evitamos este problema, y sobre todo mejoramos la seguridad.

Toda la metadata generada al aplicar configuración al paquete queda almacenada en dicho paquete. Si desea localizarla y estudiarla con más detalle, puede pulsar botón derecho sobre el paquete en *BIDS* y elegir la opción 'ver código' editando así

el código XML, donde puede buscar el término '*ConfigurationString*' y ver algo similar a este código:

```
...
<DTS:Configuration>
  <DTS:Property DTS:Name="ConfigurationType">1</DTS:Property>
  <DTS:Property
DTS:Name="ConfigurationString">C:\D\Borrar\Prueba01.dtsConfig</DTS:Property>
  <DTS:Property DTS:Name="ConfigurationVariable"></DTS:Property>
  <DTS:Property DTS:Name="ObjectName">XML con conexion a
Configuracion</DTS:Property>
  <DTS:Property DTS:Name="DTSID">{641217D5-CFD2-4A8E-A783-
3BC31FDA6645}</DTS:Property>
  <DTS:Property DTS:Name="Description"></DTS:Property>
  <DTS:Property
DTS:Name="CreationName"></DTS:Property></DTS:Configuration>
  <DTS:Configuration>
  <DTS:Property DTS:Name="ConfigurationType">7</DTS:Property>
  <DTS:Property
DTS:Name="ConfigurationString">"localhost.Configuracion.sa"; "[dbo].[SSIS
Configurations]"; "Conexiones01";</DTS:Property>
  <DTS:Property DTS:Name="ConfigurationVariable"></DTS:Property>
  <DTS:Property DTS:Name="ObjectName">Resto
configuracion</DTS:Property>
  <DTS:Property DTS:Name="DTSID">{00687A62-DB7D-47D8-9973-
EF5170B224ED}</DTS:Property>
  <DTS:Property DTS:Name="Description"></DTS:Property>
  <DTS:Property
DTS:Name="CreationName"></DTS:Property></DTS:Configuration>
  <DTS:Property
DTS:Name="LastModifiedProductVersion">10.50.1600.1</DTS:Property>
...
```

Despliegue del paquete en el servidor

Vamos seguir avanzando en temas de despliegue. En primer lugar vamos a utilizar la utilidad de implementación del paquete para agrupar los archivos necesarios en una misma carpeta. Para ello, necesitamos indicar en las propiedades del proyecto e indicarle que podremos actualizar la configuración del paquete una vez desplegado. Iremos al '*Explorador de soluciones*', haremos clic con el botón derecho sobre nuestro proyecto y elegiremos '*Propiedades*', accederemos a la '*utilidad de implementación*' y pondremos *AllowConfigurationChanges* a True, para permitir que en el posterior proceso de despliegue utilizando el *wizard* (ejecutando el archivo *.SSISDeploymentManifest* del que hablaremos más adelante) nos solicite la nueva ubicación de los archivos de configuración (*.dtsconfig*), consiguiendo así que en el proceso de despliegue se modifique el XML de los archivos *.dtsx* incluyendo la nueva ruta a los archivos *.dtsconfig* que le hemos indicado en el proceso de despliegue con dicho *wizard*.

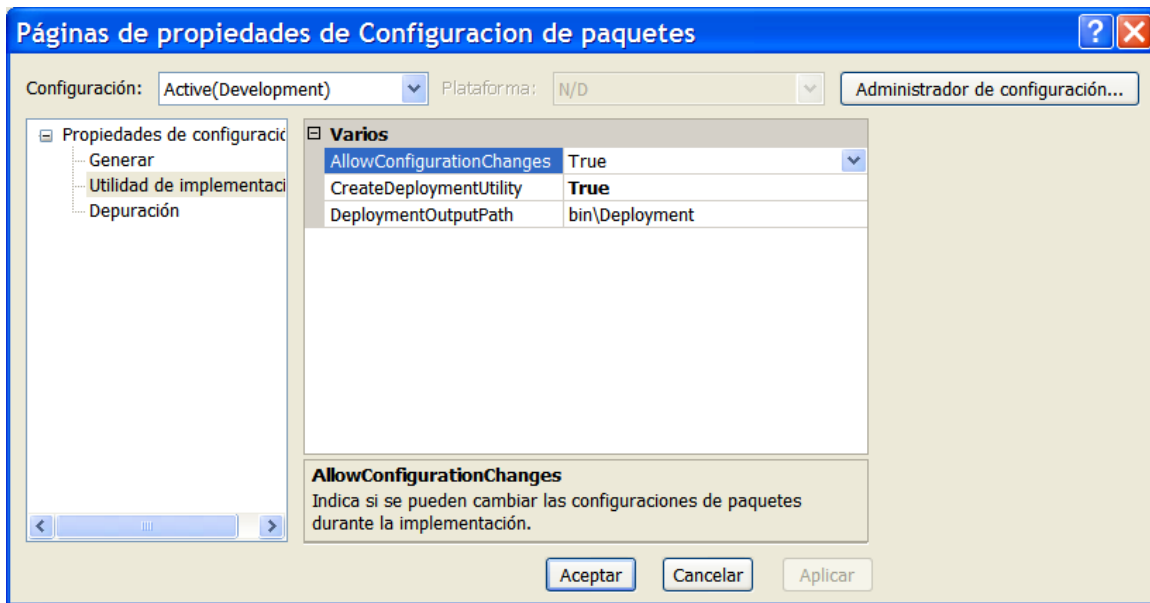


Figura 4-28 Propiedades de Configuración de paquetes

También podemos indicar que deseamos generar la utilidad de despliegue, poniendo a True la propiedad *CreateDeploymentUtility*, en esta misma ventana.

Al generar (*Deploy*) el proyecto se crea un archivo de manifiesto en la carpeta '*bin\deployment*' de dicho proyecto, allí se crea también una copia de los diferentes paquetes .dtsx del mismo y de los archivos XML de configuración. En nuestro caso en concreto, en dicha carpeta tendremos tres archivos:

- '*Configuracion de paquetes.SSISDeploymentManifest*', manifiesto.
- '*Importar Empleados.dtsx*', nuestro paquete dtsx
- '*Prueba01.dtsConfig*', archivo XML de configuración que creamos anteriormente.

Con todo ello ya podremos utilizar el *Asistente para la instalación de paquetes*, simplemente haciendo doble clic en el archivo '*.SSISDeploymentManifest*' y seguir los pasos que allí se indican. Por cuestiones de espacio y por la sencillez de uso, dejo como tarea pendiente al lector el ejecutar y probar esta utilidad. También podemos trasladar paquetes utilizando la utilidad de línea de comandos *DTUTIL*, para lo que no será necesario el archivo anterior (éste sólo es utilizado por la utilidad de instalación citada anteriormente, en cualquier otro caso vamos a prescindir de él). Pero en este caso lo vamos a hacer de forma manual, para conocer con detalle lo que finalmente acaban haciendo estas utilidades.

Copiaremos el archivo '*Prueba01.dtsConfig*' a la carpeta '*D:\SSISConfig*' en el servidor de destino, la cual tiene ya aplicada la seguridad a nivel del *File System* (NTFS) para evitar accesos no deseados. Copiaremos el archivo '*Importar*

Empleados.dtsx' a una carpeta temporal que hay en dicha máquina, llamada 'D:\Borrar', dicho archivo lo deberemos borrar una vez terminada la importación. Abriremos el *SQL Server Management Studio (SSMS)* para realizar la importación de dicho paquete en la base de datos *MSDB*.



Figura 4-29 Conexión al servidor de *Integration Services* desde *SSMS*

Para ello, una vez conectados al servicio de *Integration Services*, sobre la carpeta '*MSDB*', pulsamos botón derecho y hacemos clic en la opción '*Importar*'.

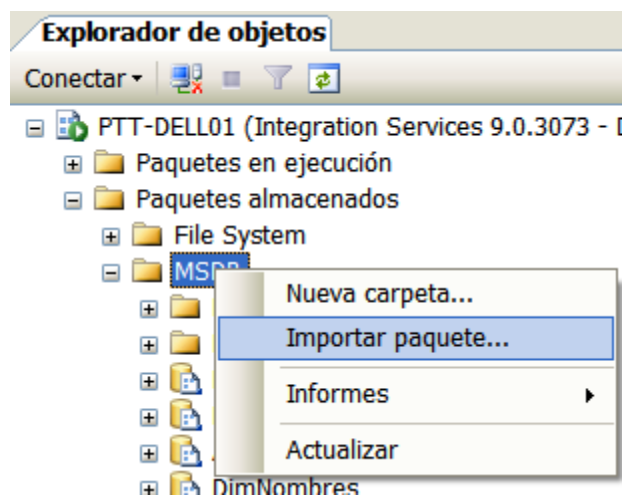


Figura 4-30 Opción Importar paquete... desde el explorador de objetos

Importar paquete

Ubicación del paquete: Sistema de archivos

Servidor:

Autenticación

Tipo de autenticación: Autenticación de Windows

Nombre de usuario:

Contraseña:

Ruta de acceso del D:\Borrar\Importar Empleados.dtsx

Importar paquete como

Nombre del paquete: Importar Empleados

Nivel de protección:

Aceptar Cancelar Ayuda

Figura 4-31 Datos del paquete a importar

Ejecución del paquete en el servidor

Bien, pues ya tenemos nuestro paquete desplegado en el servidor, y nuestro archivo de configuración XML también subido a la carpeta del servidor.

También deberemos tener disponible la base de datos '*Configuracion*' con la tabla '*dbo. SSIS Configurations*' en el servidor que hayamos decidido, en nuestro caso en el propio servidor de *Integration Services*.

Es el momento de ejecutar nuestro paquete, simplemente nos posicionamos sobre él con el cursor, pulsamos botón derecho y hacemos clic en '*Ejecutar paquete*', mostrándose la siguiente ventana, en la que indicamos su nombre y ubicación, y la ruta al archivo de configuración.

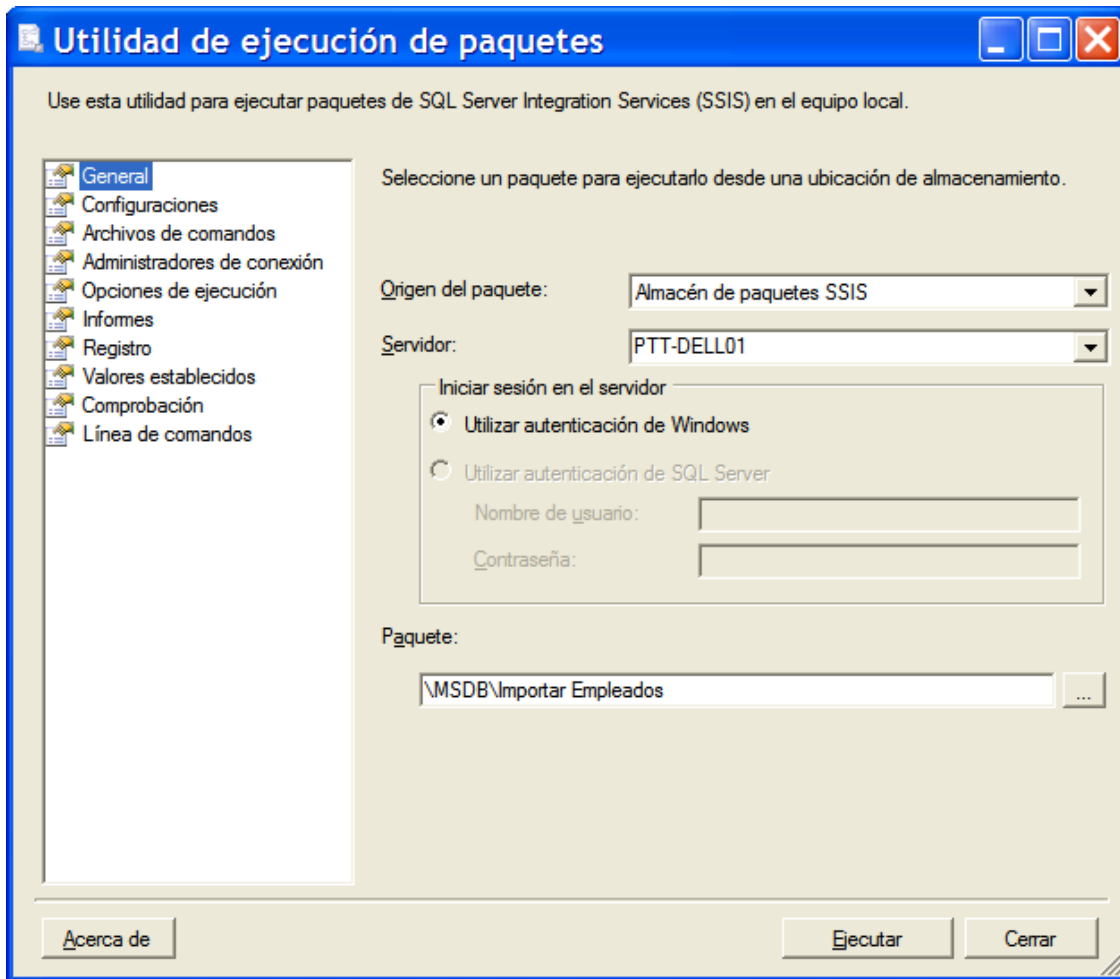


Figura 4-32 Utilidad de ejecución de paquetes, menú 'General'

Vamos al menú 'Configuraciones' (debajo de 'General') y agregamos el archivo de configuración XML

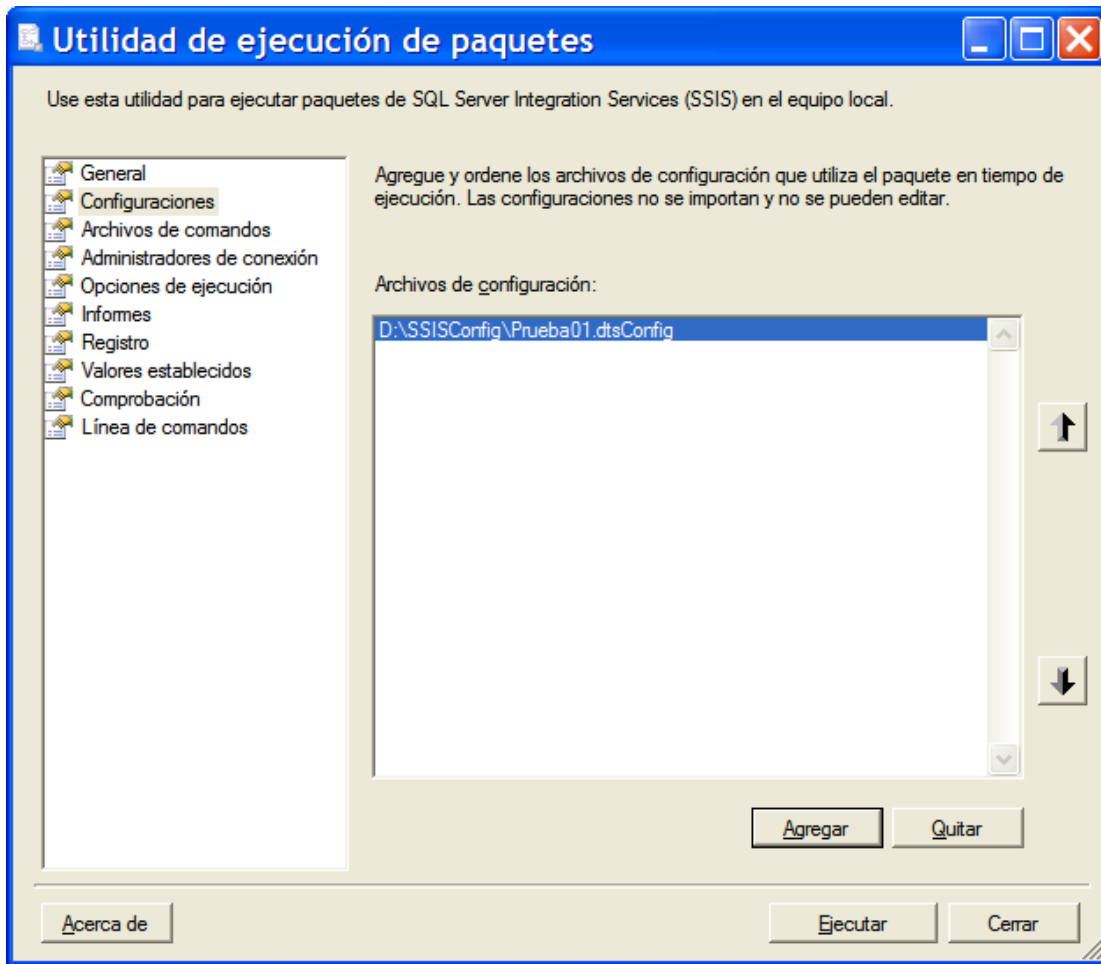


Figura 4-33 Utilidad de ejecución de paquetes, menú 'Configuraciones'

Finalmente, pulsamos el botón 'Ejecutar'.

Hay otras alternativas para la ejecución de paquetes, lo podemos hacer desde la línea de comandos utilizando DTEXEC, y pasando los parámetros apropiados, incluido el nombre del archivo de configuración.

Programar la ejecución de paquetes

Como hemos visto, podemos ejecutar cualquier paquete, tanto desde una utilidad gráfica como desde la línea de comandos. Pero es muy habitual que estas tareas no las hagamos de forma interactiva, sino que las queramos programar para que se ejecuten periódicamente, y en muchas ocasiones a horas en las que deberíamos estar en casa, o al menos fuera del trabajo.

Podemos utilizar el *Agente de SQL Server* para programar la ejecución de paquetes SSIS. Crearemos un nuevo trabajo (*Job*), y aplicaremos una programación para que se ejecute todas las noches a las 3:30AM.

Vamos a explicar con detalle cómo se realiza. Iremos al *SQL Server Agent*, y sobre la carpeta '*Trabajos*' ('*Jobs*'), pulsaremos botón derecho y haremos clic en la opción '*Nuevo trabajo...*'.

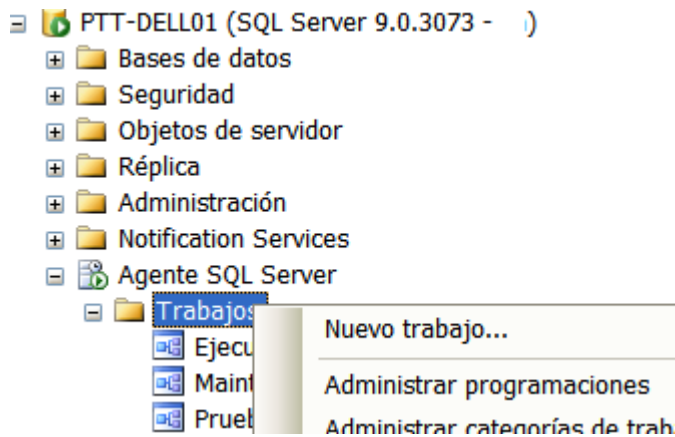


Figura 4-34 Nuevo Trabajo en el Agente de SQL Server

Nos aparece un formulario, en el menú '*General*', introduciremos el nombre del paquete (en nuestro ejemplo '*Paquete Importar Empleados*') y una descripción, para que quede mejor documentado.

En el menú '*Pasos*' ('*Steps*'), pulsaremos el botón '*Nuevo*', daremos un nombre ('*Paso Importar Empleados*'), indicaremos que el del tipo '*Paquete SQL Server Integration Services*', que se ejecutará con la cuenta del servicio *SQL Server Agent*. En la pestaña '*General*' indicaremos el nombre y la ubicación del paquete.

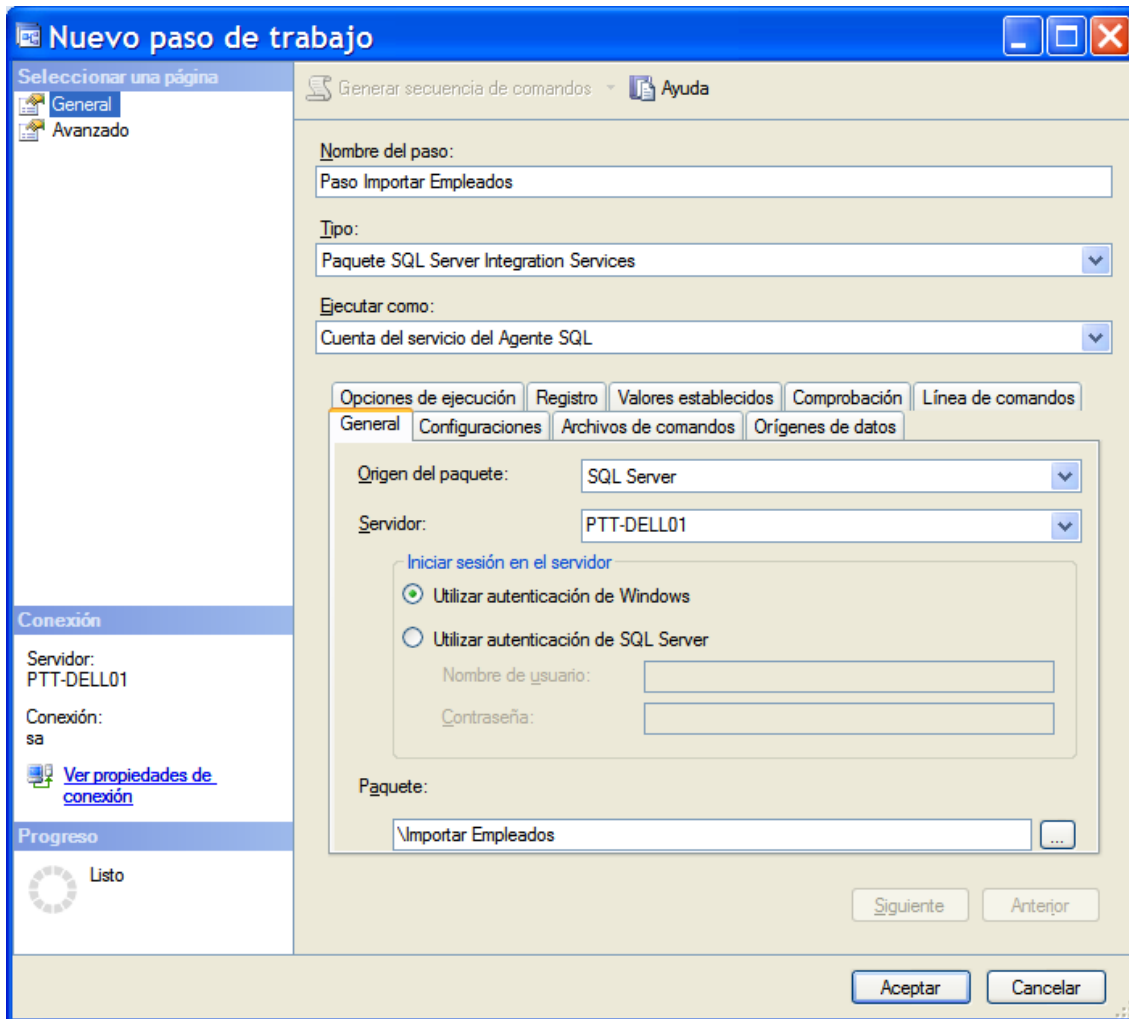


Figura 4-35 Configuración del primer paso (y único) de este Trabajo

En la pestaña 'Configuraciones' indicaremos la ruta de acceso a los archivos de configuración.

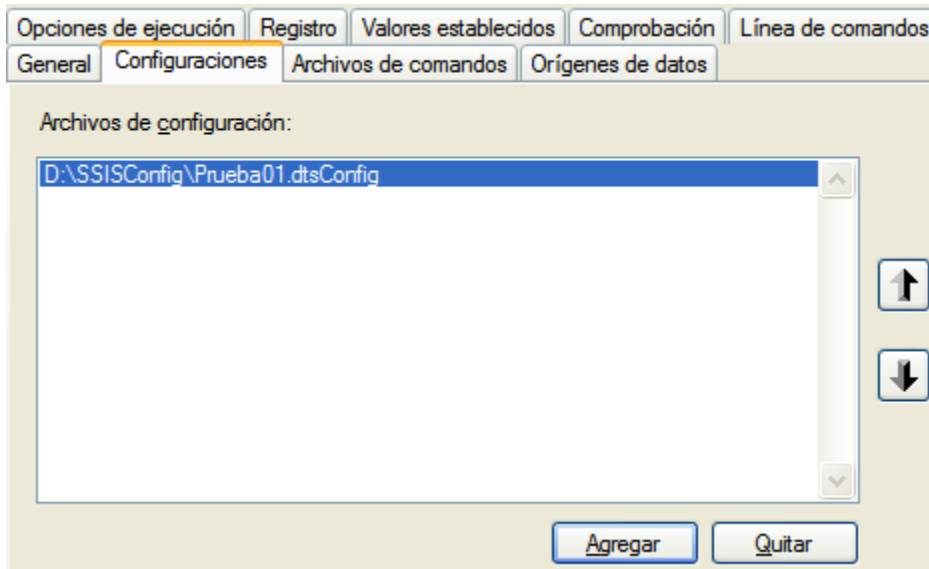


Figura 4-36 Pestaña 'Configuraciones'

Como podéis apreciar en la figura 9, hay otras muchas pestañas, pero por el momento, dejamos al lector la tarea de conocerlas e irse familiarizando más afondo con ellas.

Bien, ahora vamos a pasar a configurar la programación de la ejecución de dicho trabajo, iremos al menú '*Programaciones*' y pulsaremos el botón '*Nueva*' que aparece en la parte inferior. Allí tenemos un formulario muy completo, donde indicar con gran flexibilidad cuando se ejecutará dicho trabajo. En este caso lo vamos a programar, tal y como indicamos anteriormente, para que se ejecute todas las noches a las 3:30AM.

Figura 4-37 Programación del Trabajo

Ya tenemos nuestro trabajo en el servidor, preparado para ejecutarse automáticamente todas las noches.

Otra alternativa, a la hora de programar la ejecución del paquete, es hacer un archivo .bat que llame a la utilidad de línea de comandos *DTEXEC* y programar luego la ejecución del archivo de comandos (.bat). En circunstancias habituales, no recomiendo esta alternativa, sino la descrita anteriormente.

Conclusiones

Hemos creado un paquete de *Integration Services*. Para facilitar su despliegue en diferentes entornos (desarrollo, pruebas, preproducción, producción), hemos almacenado la información susceptible de cambiar de forma externa al paquete, para evitar que el cambio de entorno implique la modificación del paquete. Simplemente tendremos que modificar el archivo XML, o las filas correspondientes en la tabla '*dbo.SSIS Configurations*' de la base de datos '*Configuracion*', en función de las características del nuevo entorno de ejecución. Hemos explicado las utilidades

que tenemos para configurar y realizar el despliegue, y cómo hacerlo de forma manual. Finalmente, hemos explicado cómo ejecutar el paquete utilizando estas configuraciones externas, y como programarlo para que se ejecute periódicamente.

Demos incluidas en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda a los siguientes videos donde podrá ver la utilización de la mayor parte de los componentes descritos anteriormente:

‘**Demo SSIS 06C**’ donde se muestra la definición y uso de configuraciones.

‘**Demo SSIS 06D**’ donde se realiza todo el proceso de despliegue de un paquete.

‘**Demo SSIS 06E**’ muestra la interfaz gráfica de ejecución de paquetes llamada *DTExecUI*.

‘**Demo SSIS 06F**’ que realiza la programación de la ejecución de dicho paquete.

Laboratorio incluido en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al material del ‘**Lab SSIS 06A**’, realice paso a paso todo lo allí expuesto y responda a las preguntas que se incluyen. Este Lab le ayudará a asentar los conocimientos adquiridos sobre el despliegue de paquetes y la programación de su ejecución. Y por supuesto, si tiene cualquier duda sobre lo visto o cualquier situación que quiera resolver puede utilizar las sesiones de **tutorías** y los **foros** del curso para resolverlas.

5. Analysis Services

Microsoft, incluye Analysis Services como un componente de SQL Server, que nos va a permitir cubrir una serie de necesidades que tienen los usuarios de negocio a la hora de obtener información de nuestros sistemas. Hemos sintetizado dichas necesidades de la siguiente forma:

- Debemos permitir a los usuarios que realicen **consultas ad-hoc**. Si necesitan saber cuántos clientes tiene la empresa en la provincia de Murcia que han comprado un determinado artículo en el año 2008, que sean capaces de abrir Excel, conectarse a una fuente de datos, y con simples acciones de arrastrar y soltar en una PivotTable, obtener dicha información. Los usuarios deben poder obtener respuestas a las preguntas de negocio de les surjan de una forma ágil y dinámica como la que acabamos de describir. Todo esto con unos tiempos de respuesta prácticamente inmediatos, obteniendo el resultado en el momento que finalizamos la acción de arrastrar y soltar.
- También debemos permitir que los usuarios sean capaces de realizar sus propios informes, llegando al **autoservicio de informes** (*self service reporting*). Para ello, además de facilitarle herramientas de creación de informes orientadas al usuario (no al desarrollador), como Report Builder, tenemos que ofrecerle una **fuente de información de gran calidad** (limpia, amigable, elaborada y confiable) que le permita obtener información analítica para reflejarla en sus informes.
- Tanto las consultas ad-hoc como los informes y dashboards que hayamos definido deben tener un **tiempo de respuesta rápido**. Si utilizamos bases de datos relacionales, será muy complicado conseguir este objetivo, ya que para obtener ciertos valores, se necesitarán agregar valores (sumar, buscar el mínimo, el promedio, el máximo, etc.) a partir de un gran número de filas (millones en muchas ocasiones), y bien creamos una buena cantidad de tablas específicas que tengan estos datos pre calculados, o no será posible obtener un tiempo de respuesta válido para el usuario.
- Hay otro grupo de necesidades, como **segmentar o hacer predicciones**, en base a grandes volúmenes de datos. Al usuario de negocio le gustaría saber cuáles son las previsiones de ventas para el próximo trimestre, basándose en el historial de ventas de los últimos N años, así como obtener ciertas agrupaciones y segmentaciones en base a ese historial de ventas. No nos referimos a que el usuario quiera estudiar una agrupación en concreto, sino a que sea el propio sistema el que le proponga, en base a un estudio exhaustivo de los datos, esas agrupaciones. Todo esto se consigue mediante la **minería de datos** (*data mining*).

Introducción

Cuando nos introducimos en el mundo de las bases de datos multidimensionales y en los sistemas OLAP (*OnLine Analytical Processing*), debemos realizar un gran esfuerzo en cambiar la mentalidad, sobre todo, si estamos habituados a trabajar con bases de datos relacionales y sistemas OLTP (*OnLine Transactional Processing*). Una vez que tenemos estos conceptos básicos asimilados, debemos ponerlos en práctica ya sobre algún producto en concreto, y por tanto aprender también su nueva terminología. Como sabemos, cada producto tiene sus particularidades y términos propios.

OLAP (Online Analytical Processing)

Es una solución que nos permite agilizar la consulta a grandes cantidades de datos, utilizando estructuras multidimensionales (conocidas como cubos OLAP) que contienen datos resumidos y pre calculados.

En los sistemas relacionales la unidad para almacenamiento de la información es la *Tabla*; en los sistemas multidimensionales, la unidad de almacenamiento es el *Cubo*. Este término ha tenido tanta difusión en los medios, que incluso es muy habitual encontrar referencias de forma coloquial a estos sistemas con términos como “los cubos”, “cubos olap” incluso “cubos sql”. Y es habitual que se utilicen estos términos en los buscadores.

A partir de este momento, salvo en algunas excepciones como el apartado “El cubo. Conceptos básicos” y algunos incisos para aclarar algún término de forma conceptual, nos vamos a centrar, principalmente, en la terminología que utiliza *SQL Server* en su base de datos multidimensional, concretamente en su servicio llamado ***Analysis Services***.

En primer lugar, citaremos un concepto que revolucionó por completo *Analysis Services* a partir de la versión 2005, el de **UDM** (*Unified Dimensional Model*) unificando lo mejor del modelo relacional y del multidimensional, creando una capa intermedia que permite abstraerse de la fuente de datos y tener un único modelo que será utilizado como fuente de información. Los cubos de *Analysis Services* se basan en este concepto de UDM que acabamos de comentar.

A continuación conoceremos como es el almacenamiento de información en este sistema, para más adelante detallar cada uno de sus componentes principales, desde los orígenes de datos, y vistas sobre ellos que nos permitan hacer ciertas tareas de modelamiento, pasando por los *cubos*, con todas sus principales características, y por sus *dimensiones*, *jerarquías* y *medidas*. A continuación veremos cómo se interrelacionan todos estos elementos. Para cerrar el círculo, daremos un paso más allá y veremos la potencia que nos ofrece el *lenguaje MDX* y los potentes *cálculos* que podemos realizar con él, y añadiremos un poco más de sentido a los

datos con la utilización de los *KPI* (*Key Performance Indicator*), que nos permiten saber más allá de un *valor*, si éste va bien, mal o regular con respecto a un *objetivo* que nos hayamos marcado, y qué *tendencia* está siguiendo a lo largo del tiempo. Finalmente veremos de pasada otros componentes que tenemos disponibles como las *acciones*, las *traducciones*, las *perspectivas* y los *roles*.

El cubo. Conceptos básicos

Vamos a introducir al lector, de forma coloquial y con ejemplos, en algunos conceptos básicos y definiciones de *cubo*, *dimensiones*, *jerarquías* y *medidas*, para evitar que por desconocimiento de estos términos, no pueda seguir la lectura de este capítulo.

Cubos: Son estructuras de datos organizados y agregados, proporcionándonos un acceso muy rápido y estructurado a la información que contienen. Sus principales componentes son las dimensiones y las medidas.

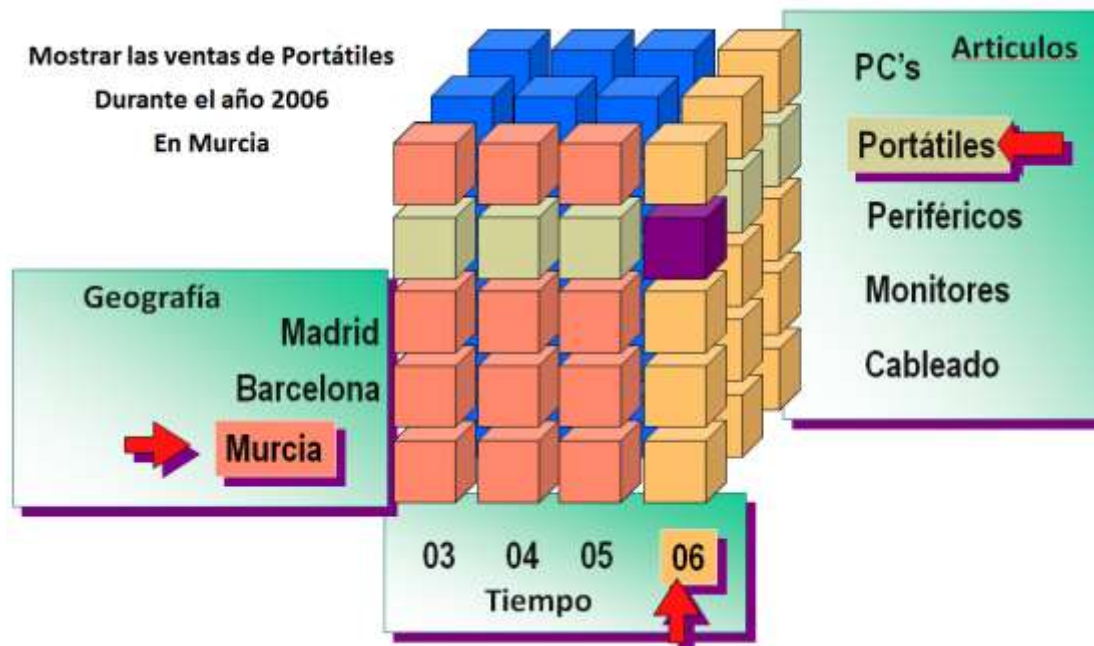


Figura 5-1 Cubos

Tal y como se ve en la figura anterior, pueden responder muy rápidamente a preguntas como ¿Cuáles han sido las ventas de portátiles en Murcia, en el año 2006?. Esa información está almacenada en una celda y nos la devolverá inmediatamente, de una forma mucho más eficaz que un sistema transaccional normalizado, que debería sumar miles o millones de filas para obtener ese valor.

Dimensiones: Son cada una de las perspectivas o ejes que componen un cubo. Aunque gráficamente sólo podemos representar un cubo en tres dimensiones, e incluso de ahí viene su nombre, realmente nos podemos encontrar cubos con muchas más dimensiones o perspectivas desde las que analizar la información. En el caso anterior, es fácil que haya otras dimensiones como Cliente, Empleado, Almacén, Tienda, etc. Y así poder responder a preguntas sobre estos términos también. Pongamos un ejemplo que sea una ampliación de la pregunta anterior: ¿Cuáles han sido las ventas de portátiles en Murcia, en el año 2006, en la tienda de Gran Vía, y realizadas por el empleado Pepe López ?

Atributos: Son cada una de las características de la dimensión. Por ejemplo, la dimensión Cliente, tendrá atributos como *Dirección, Email, Edad, Sexo, Estudios, etc.* Los cuales también estarán disponibles a la hora de analizar la información.

Jerarquías y niveles: Habitualmente tendemos a tener la información agrupada, y no siempre la queremos al mismo nivel de detalle. Para cubrir esta necesidad, dentro de una dimensión, disponemos de *jerarquías*, que nos permiten conseguir este objetivo agrupando la información a distintos niveles de detalle. Pongamos algunos ejemplos:

- En la dimensión Tiempo, es habitual encontrarnos con jerarquías con niveles: año, trimestre, mes, día.
- En la dimensión Geografía, con jerarquías con niveles: país, provincia, población.
- En la dimensión Artículo, con jerarquías con niveles: familia, subfamilia, grupo, subgrupo.

Miembros: Son cada uno de los elementos que tenemos disponibles en una dimensión. Haciendo una analogía con las *tablas*, podríamos decir que son el equivalente a las filas.

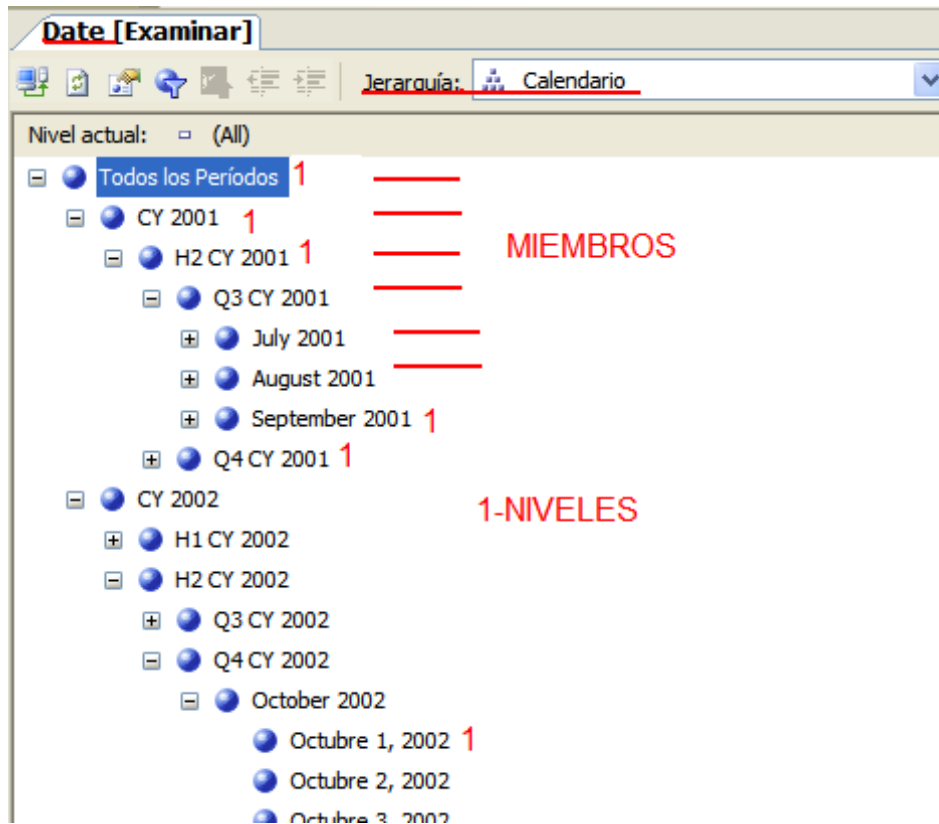


Figura 5-2 Dimensiones, Jerarquías, Niveles, Miembros

En la imagen anterior, mostramos los elementos de la dimensión Date (*Tiempo*). En ella tenemos una jerarquía llamada *Calendario*, compuesta por los niveles: *Todos*, *Año*, *Semestre*, *Trimestre*, *Mes* y *Día*. Los miembros que la forman son cada uno de los elementos mostrados a cualquier nivel, citaremos algunos: *Todos los periodos*, *CY 2001*, *CY 2002*, ..., *H1 CY 2001*, *H2 CY 2001*, ..., *Q1 CY 2001*, *Q2 CY 2001*, *Q3 CY 2001*, *Q4 CY 2001*, ..., *July 2001*, *August 2001*, ..., *October 1 – 2001* ... *October 31 – 2001*, etc.

Medidas: Son cada uno de los datos cuantificables que queremos analizar, un hecho que queremos medir, para cada intersección entre las diferentes dimensiones.

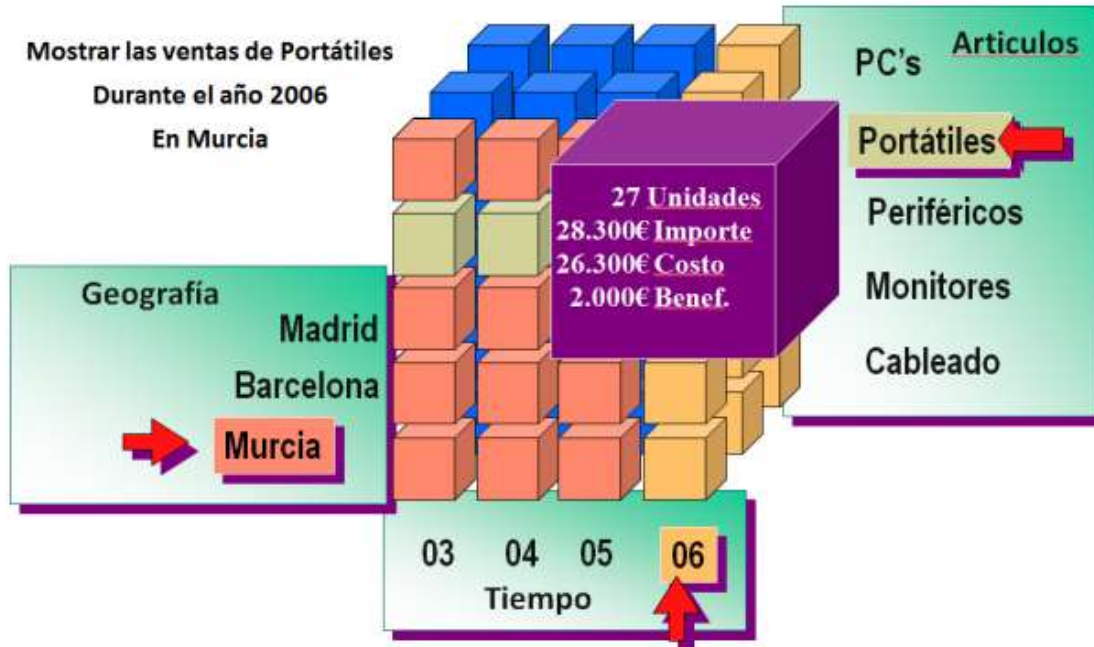
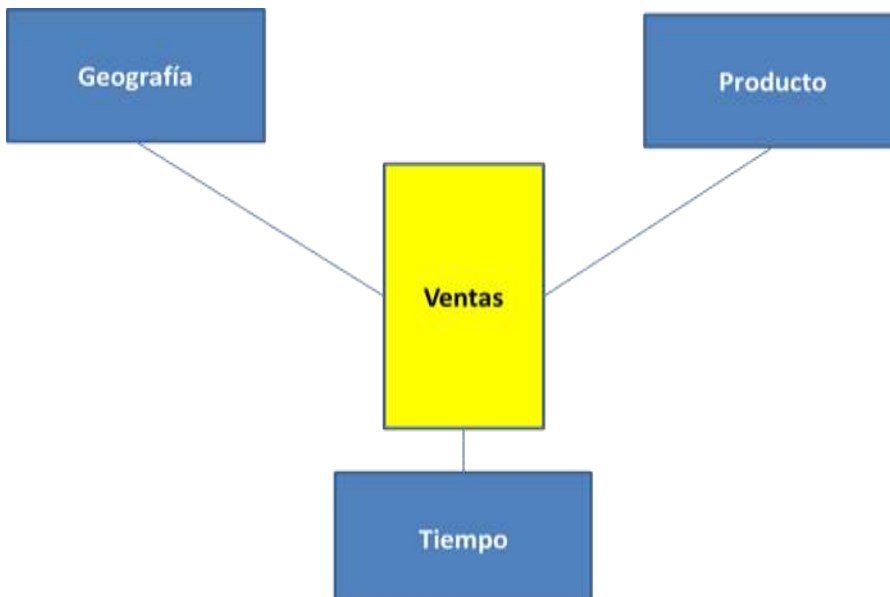


Figura 5-3 Medidas

Según la imagen anterior, tendríamos cuatro medidas: *Unidades*, *Importe*, *Costo* y *Beneficio*. Analizadas desde las perspectivas *Tiempo*, *Geografía* y *Producto*.

El cubo, está basado en un origen de datos modelado en estrella o copo de nieve, que a nivel relacional, tendrá un modelo físico, como el del ejemplo que se muestra en la siguiente figura.



Microsoft Business Intelligence: vea el cubo medio lleno

Figura 5-4 Modelo relacional en Estrella

Este modelo está compuesto por una tabla de hechos (*Ventas*) y tres tablas de dimensiones (*Geografía*, *Tiempo* y *Producto*).

Demo incluida en el curso online

Si estás siguiendo el curso, del cual este eBook es material complementario, accede al video de la '**Demo SSAS 01A**', en el que se muestra el acceso desde Excel a un cubo de Analysis Services, realizando diferentes consultas ad-hoc y mostrando datos de negocio con flexibilidad y de forma muy rápida.

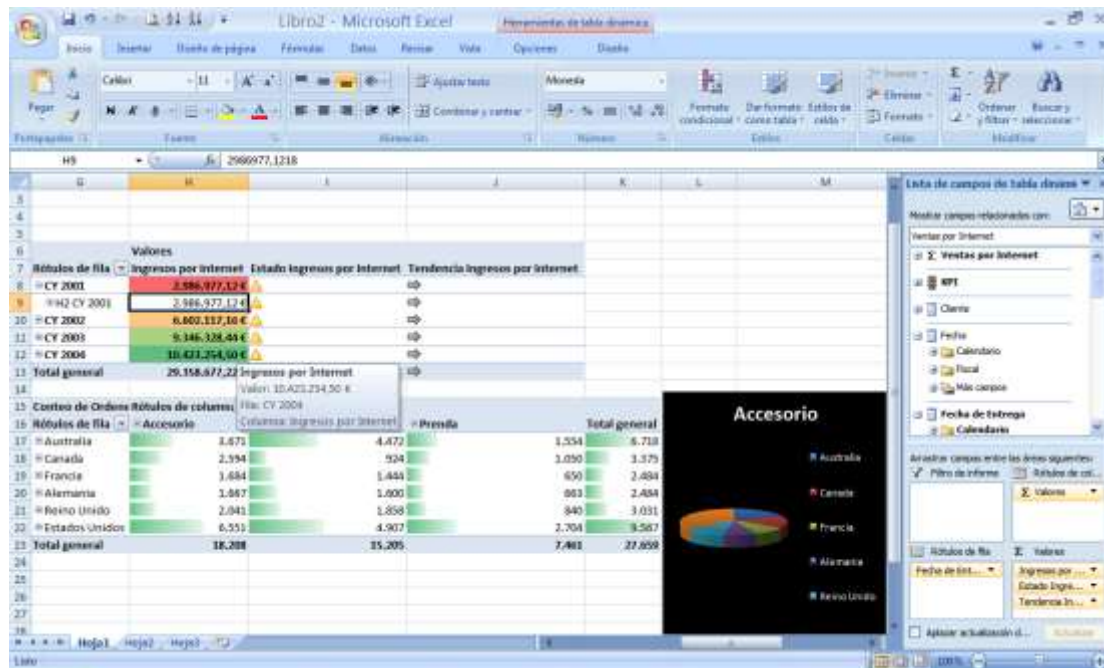


Figura 5-5 Consultas ad-hoc desde Excel, accediendo a Analysis Services

Objetos de una base de datos de Analysis Services

En Analysis Services existen una serie de objetos que podemos crear en una base de datos y que debemos conocer y familiarizarnos con ellos para poder llevar a cabo nuestros proyectos de creación de bases de datos multidimensionales, estos son:

- Orígenes de datos (*Data Sources*)
- Vistas de orígenes de datos (*Data Source Views*)
- Dimensiones (*Dimensions*)
- Cubos (*Cubes*)
- Estructuras de Minería (*Mining Structures*)
- Funciones (*Roles*)
- Ensamblados (*Assemblies*)

A continuación nos vamos a centrar en los cuatro primeros elementos. El resto quedan fuera del alcance de este libro. Los *Roles* nos permiten gestionar la Seguridad, los *Mining Structures* son estructuras de minería de datos, y los *Assemblies* permiten ampliar la funcionalidad del lenguaje MDX (*MultiDimensional eXpressions*) y de las expresiones DMX (*Data Mining eXtensions*).

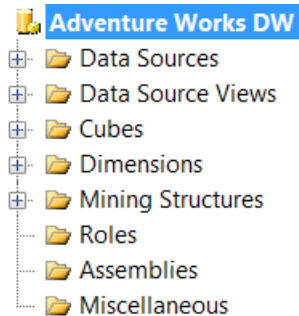


Figura 5-6 Objetos de una Base de Datos

Data Sources (DS)

Un *Data Source* es, básicamente, una conexión a una base de datos. Por el momento no vamos a ir más allá, y nos vamos a centrar en el uso del proveedor nativo Ole Db para SQL Server, que será el que utilizaremos habitualmente para conectarnos a nuestro Data Mart o Data Warehouse, que al fin y al cabo será una base de datos relacional en SQL Server.

Data Source Views (DSV)

En cualquier proyecto de este tipo debemos tener una o varias fuentes de datos (*Data Sources*), que nos sirvan para obtener la información que vamos a utilizar en los cubos y dimensiones posteriormente. *Analysis Services* nos permite la posibilidad de tener varios orígenes de datos y consolidarlos.

Una vez que nos conectamos de forma independiente a cada uno de los orígenes de datos, creando un *Data Source (DS)* para cada uno de ellos, como hemos indicado antes, podemos consolidar la información, construyendo un esquema lógico, basado en uno o varios *Data Sources*, a través del *Data Source View (DSV)*.

Un *DataSource View* es un esquema lógico en el que podemos incluir:

- **Cálculos con Nombre** (*Named Calculations*), son columnas calculadas en base a otras columnas, a las que asigno un nombre. Estos cálculos deben ser escritos en el dialecto SQL del origen de datos, es decir, si estoy accediendo a Oracle, tendré que usar el dialecto SQL de Oracle, si estoy accediendo a SQL Server tendré que utilizar Transact-SQL, y así sucesivamente según el origen. También lo podemos utilizar simplemente para renombrar columnas con nombres más amigables.

- **Consultas con Nombre** (*Named Queries*), son vistas sobre una o varias tablas. Nos pueden servir, por ejemplo, en el caso de que no tengamos un Data Warehouse, para hacer un modelado en estrella o en copo de nieve a partir de un origen normalizado. Nos ofrecen todas las posibilidades de la instrucción SELECT del dialecto SQL que estemos utilizando. Ejemplos de tareas que podemos hacer con ellas serían: desnormalizar, obtener un subconjunto de datos, poner alias para obtener nombres amigables, combinar datos de varias tablas, etc.
- **Relaciones a nivel lógico**, es una validación de integridad referencial a nivel lógico. Nos permiten crear Foreign Keys lógicas si éstas no existiesen en el origen. Aunque en los sistemas transaccionales no es habitual, además de ser una mala práctica, el que nos encontremos con la ausencia de Foreign Keys, en los Data Marts o Data Warehouses es más habitual, y no es considerado una mala práctica, sino que tiene sus pros y sus contras, y en unos casos se opta por la creación de FKs y en otros no. En caso de no disponer de Foreign Keys las podremos crear a nivel lógico.
- **Claves primarias lógicas**, aun cuando no hay una *primary key*, podemos crearla a nivel lógico. Con las Primary Keys ocurre lo mismo, en caso de no existir en el origen, podríamos crearlas a nivel lógico en el Data Source View. Esto ya es algo menos habitual, porque tener tablas sin Primary Key no es una buena práctica, ni en sistemas transaccionales, ni en Data Marts o Data Warehouses. Aun así, si nos encontrásemos un origen sin Primary Keys, siempre nos queda la posibilidad de crearlas a nivel lógico. Analysis Services requiere una Primary Key por cada tabla para construir los queries, bien sea física o lógica.

El *Data Source View*, realmente nos será de poca utilidad en el caso de tener un solo origen de datos y que además sea un Data Mart o Data Warehouse, ya modelado en estrella (o incluso si es un snowflake). Pero si no lo tuviésemos y quisiéramos consolidar información de diferentes fuentes e incluso modelar en base a vistas nuestra base de datos normalizada, para obtener un modelo en estrella, nos será de gran utilidad.

Imaginemos, por ejemplo, que nuestro cubo va a tener como origen de datos dos bases de datos diferentes. No habrá ningún problema en integrar en esta vista de origen de datos (DSV) que las dimensiones Geografía y Tiempo vengan de un Data Source, y las dimensiones Producto y Cliente vengan de otro Data Source. Como hemos comentado antes, podemos definir relaciones lógicas. Para establecer una relación, que será siempre del tipo 1:M, necesitamos tener Primary Key, y en el caso de no tenerlas, podemos crear Primary Keys lógicas. También podemos crear campos calculados; imaginaos que tenemos la cantidad, el descuento y el precio, pero no tenemos el importe, que es: $(\text{cantidad} - \text{descuento}) * \text{precio}$. En ese caso

podemos crear en el DSV una columna calculada con dicha fórmula. Así como renombrar columnas con nombres más amigables.

Con un Data Source View podemos también hacer justo lo contrario, es decir, crear una vista que nos simplifique el origen de datos. Imaginemos que nuestro Data Source es una base de datos con 500 tablas con un complejo modelo de datos, y que para el problema que estamos resolviendo actualmente, sólo necesitamos acceder a 15 tablas. Podemos utilizar el DSV para mostrar sólo las tablas que vamos a utilizar en nuestro proyecto.

Un cubo o una dimensión, siempre tendrá como origen un *Data Source View*, y éste tendrá, a su vez, como origen uno o varios *Data Sources*.

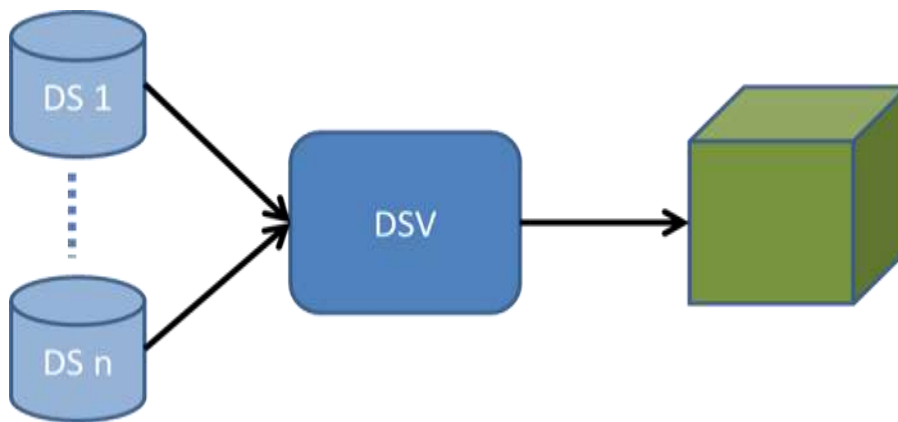


Figura 5-7 Data Sources, Data Source View y Cubo

Demo incluida en el curso online

Si estás siguiendo el curso, del cual este eBook es material complementario, accede al video de la '**Demo SSAS 01B**', en el que se muestra con detalle la creación de Data Sources y Data Source Views.

Cubos

Son, junto con los modelos de minería, los componentes principales de *Analysis Services*, y los que serán expuestos a los usuarios. El resto de objetos sirven de apoyo a la creación de los cubos. Tanto los *Data Sources* y los *Data Source Views* (vistos anteriormente), como las *Dimensiones* (que veremos a continuación), se crean con el fin de ser utilizadas posteriormente en la creación de los cubos.

Un **cubo**, en *Analysis Services*, es una estructura compuesta por diversas tablas de hechos y de dimensiones, obtenidas a partir de un *Data Source View*, relacionadas entre sí a través del uso de dimensiones (*dimension usage*), a las que puedo agregar una serie de cálculos avanzados en MDX (*calculations*), puedo también definirle indicadores clave de rendimiento (*KPIs*), definir acciones (*actions*) para dar dinamismo adicional a la información, diseñar el sistema de almacenamiento

Microsoft Business Intelligence: vea el cubo medio lleno

mediante particiones y agregaciones (*partitions and aggregations*), mostrar subconjuntos de datos mediante perspectivas (*perspectives*), e incluso mostrar los nombres de cada uno de los objetos en función del idioma de Windows seleccionado (*translations*).

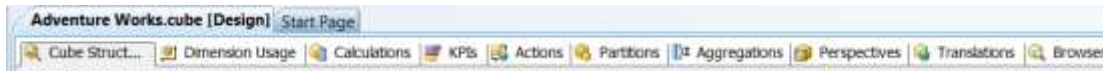


Figura 5-8 Componentes del Cubo

Estas estructuras están compuestas por Grupos de Medidas (*Measoure Group*), Medidas (*Measures*) y dimensiones del cubo (*cube dimensions*). Básicamente, un grupo de medidas, es un conjunto de medidas, que habitualmente provienen de una misma tabla de hechos. Por ahora, aunque más adelante veremos excepciones, consideremos un grupo de medidas como un conjunto de todas la medidas que provienen de una misma tabla de hechos.

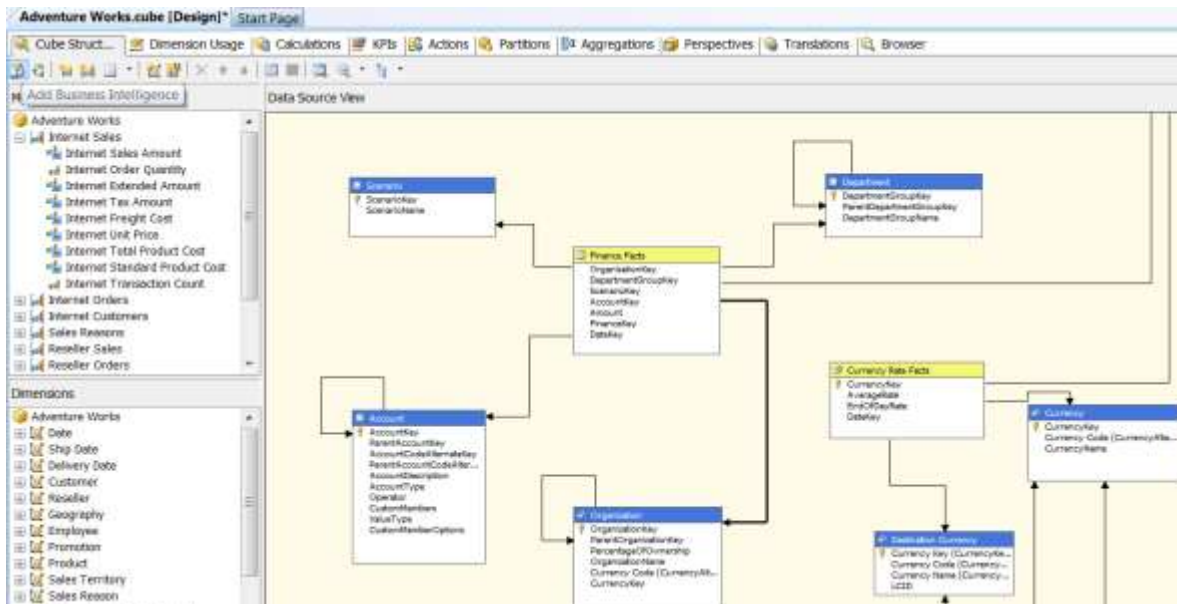


Figura 5-9 Estructura de un cubo: Medidas, Grupos de Medidas, Dimensiones de Cubo, Tablas de Hechos y de Dimensiones

Las Tablas de Hechos están representadas con color amarillo y las Tablas de Dimensiones con color azul.

Más adelante iremos estudiando con detalle cada uno de estos elementos.

Dimensiones

Son las diferentes perspectivas desde las que queremos analizar la información. Las dimensiones en sí, las que definimos a partir de la base de datos relacional, son lo que conocemos como dimensiones de base de datos. Son las que están basadas en tablas o vistas de un Data Source View. Agrupan atributos en tablas o vistas, a esos

atributos les damos un nombre entendible por el usuario, nombres amigables, y habitualmente, las clasificamos en jerarquías (jerarquías naturales), por ejemplo, la Geografía en Pais-Provincia-Población, el Tiempo en Año-Mes-Día o en Año-Semana-Día, el Producto en Categoría-Subcategoría-Producto, etc.

Veamos el ejemplo de la dimensión Tiempo, que procede de una sola tabla en nuestro origen:

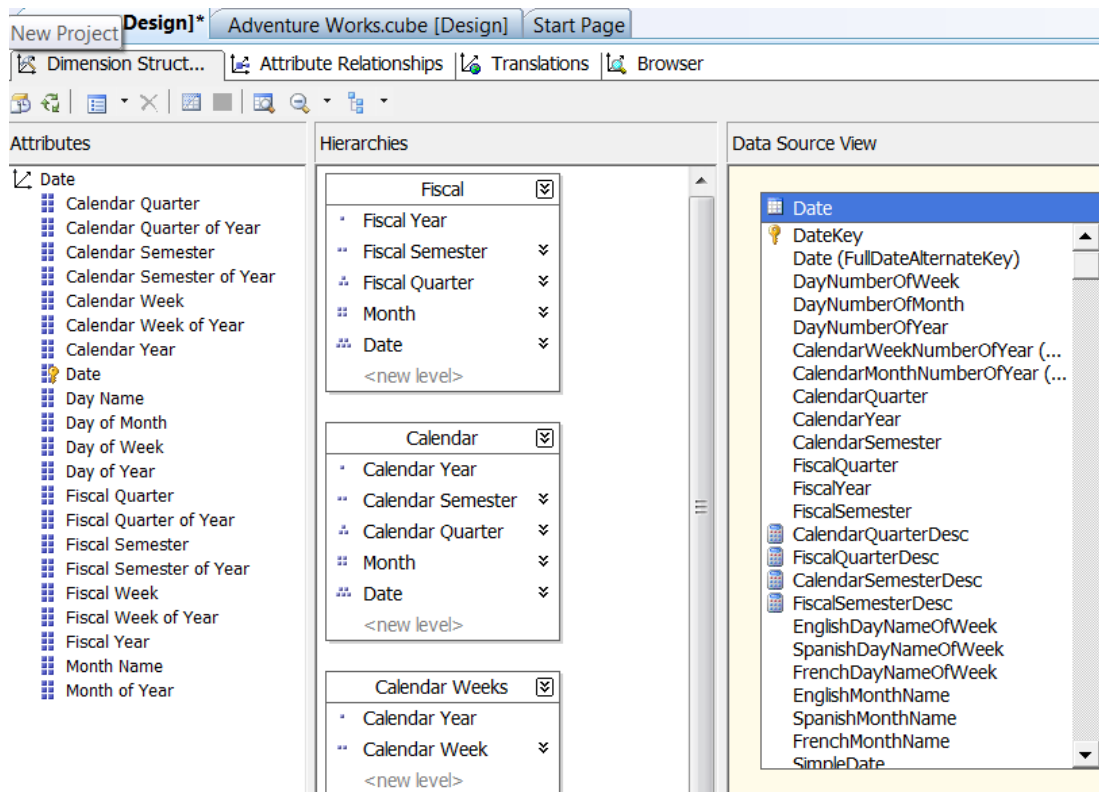


Figura 5-10 Dimensión Tiempo

Podremos también convertir varias tablas de dimensiones del origen relacional, modelado en snowflake en una sola dimensión.

Una dimensión, aunque nosotros la montemos como una sola dimensión en la base de datos de Analysis Services, puede que a nivel de nuestro origen relacional, si la tenemos definida como un snowflake (en la base de datos AdventureWorks tenemos un ejemplo con la dimensión Producto), tengamos tres tablas: Categoría, Subcategoría y Producto. A la hora de definir una dimensión Producto en Analysis Services, podemos usar estas tres tablas relacionadas, incluyendo atributos provenientes de las tablas Categoría y Subcategoría. No necesitamos tener obligatoriamente dimensión Categoría, dimensión Subcategoría y dimensión Producto. Para aclarar esto veamos la siguiente imagen:

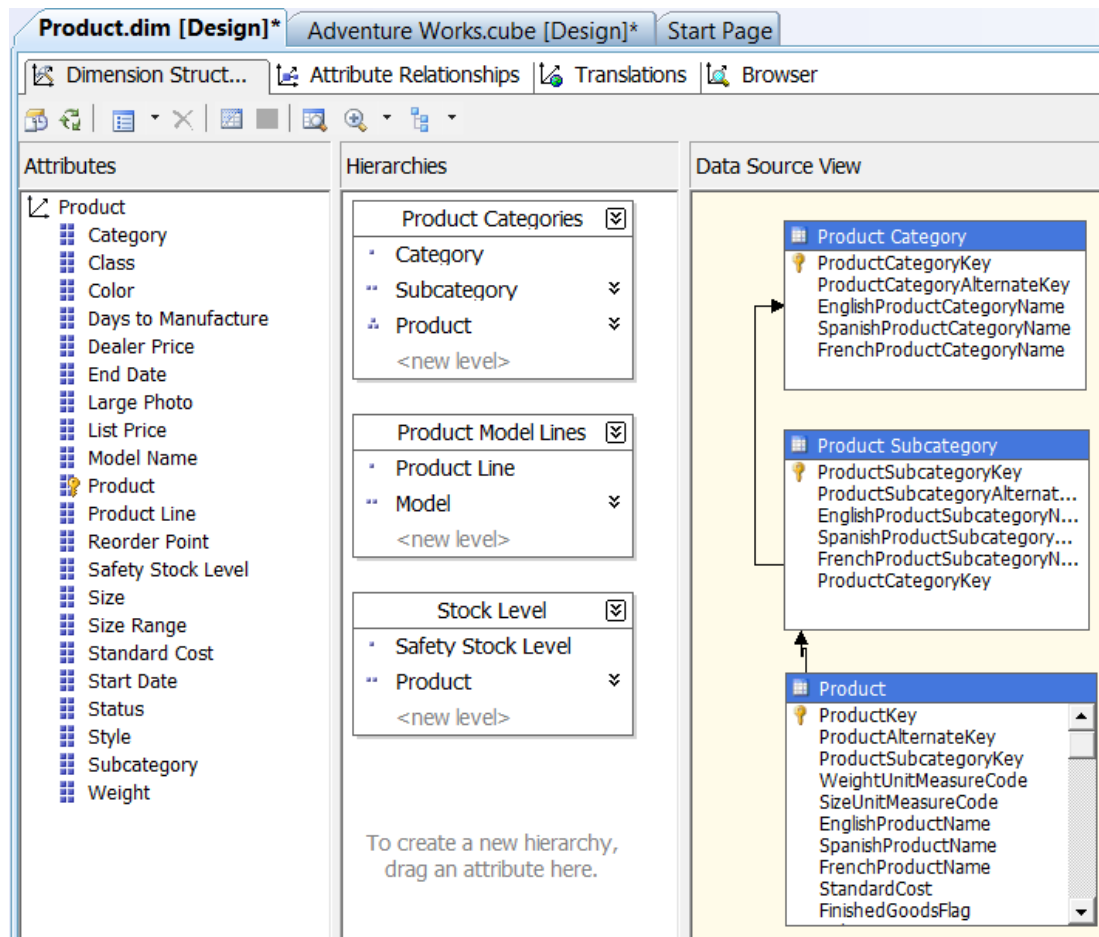


Figura 5-11 Dimension Producto

Todas las dimensiones tienen que tener un atributo clave (sería algo similar a la Primary Key del relacional) y pueden tener o no tener atributos no clave, aunque lo normal es que tengan atributos no clave, incluso que tengan decenas de ellos en algunos casos. No es habitual tener una dimensión con un solo atributo, ya que el análisis que podríamos hacer sería muy pobre.

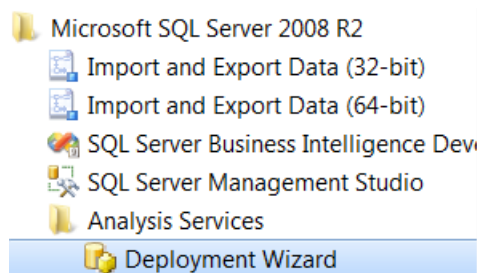
Demo incluida en el curso online

Si estás siguiendo el curso, del cual este eBook es material complementario, accede al video de la '**Demo SSAS 01C**', en el que se muestra con detalle cada uno de los componentes de un cubo, así como su creación mediante el asistente para la creación de cubos (*cube wizard*).

Construyendo (*build*) y Desplegando (*deploy*) el modelo

Una vez hemos finalizado la construcción de nuestros cubos, el siguiente paso sería desplegarlo sobre un servidor de Analysis Services, para ello tenemos varias alternativas:

- Hacer el despliegue desde Visual Studio: en nuestro proyecto, vamos al menú 'Build' y seleccionar la opción 'Deploy', realizándose el despliegue sobre el servidor indicado en las propiedades de dicho proyecto. Como en cualquier proyecto de Visual Studio, podremos tener configurados varios entornos (desarrollo, pruebas, pre-producción, producción, etc.) y por tanto, desplegar en cualquiera de ellos.
- Utilizar el *Deployment Wizard*: éste generará un archivo XML/A que posteriormente ejecutaremos en el servidor de destino. Para ejecutar dicha utilidad, ir a 'Start / MS SQL Server 2008 R2 / Analysis Services':



Demo incluida en el curso online

Si estás siguiendo el curso, del cual este eBook es material complementario, accede al video de la '**Demo SSAS 01D**', en el que se muestra con detalle una de las formas de despliegue, que es utilizando el propio *Visual Studio*, y la configuración de las propiedades del proyecto para indicarle los diferentes entornos a los que queremos desplegar y los datos de éstos (servidor, nombre de la base de datos, etc.)

Procesamiento, Modos de Almacenamiento y Agregaciones

A continuación veremos estos tres conceptos, con el fin de cerrar la introducción. Una vez que tenemos desplegada una base de datos de *Analysis Services* en un servidor, necesitamos procesar periódicamente para sincronizar los datos que se muestran con los de los orígenes relacionales. También debemos decidir los modos de almacenamiento de cada dimensión y de cada una de las particiones de los grupos de medidas. Y finalmente diseñar las agregaciones que quedarán almacenadas en la base de datos.

Procesamiento (*Processing*)

Un cubo contiene datos pre calculados, que permiten un acceso rápido por parte de los usuarios. Periódicamente debemos refrescar los datos del cubo para que incorporen la información que se ha ido generando en los orígenes de datos.

El procesamiento permite mantener actualizados los objetos de *Analysis Services*. Se encarga de llenar con datos los objetos a partir de orígenes de datos relacionales.

El trabajo de procesamiento está dentro de una transacción, que se puede confirmar o deshacer. Mientras se está procesando el cubo, se puede tener acceso a sus objetos para realizar consultas. Sólo dejarán de estar disponibles una vez procesados correctamente durante el tiempo en que se confirman los cambios, mientras que si se deshacen, no dejarán de estar disponibles.

Modos de Almacenamiento (*Storage Modes*)

Una vez que tenemos diseñado un cubo, independientemente de otros muchos factores, debemos decidir en qué modo se va a almacenar en nuestro sistema (SQL Server), y conocer las ventajas e inconvenientes de cada una de las formas de almacenamiento que disponemos, que son tres:

- **MOLAP** (*OLAP Multidimensional*): Es una estructura de almacenamiento multidimensional, muy optimizada para maximizar el rendimiento de las consultas. Los cambios que se producen en el origen de datos no serán visibles en el cubo hasta que se *procese* la información. Los datos y agregaciones se comprimen y escriben en disco.
- **ROLAP** (*OLAP Relacional*): Almacenamiento en una base de datos relacional. Permite a los usuarios ver los datos en tiempo real, aunque por el contrario, es más lento que MOLAP.
- **HOLAP** (*OLAP Híbrido*): Es una combinación de los modos MOLAP y ROLAP. Las agregaciones se comprimen y se escriben en disco.

Nota: *Procesar* consiste en leer la información del origen de datos, pre-calcularla, almacenarla en el cubo y dejarla lista para ser consultada por las herramientas cliente.

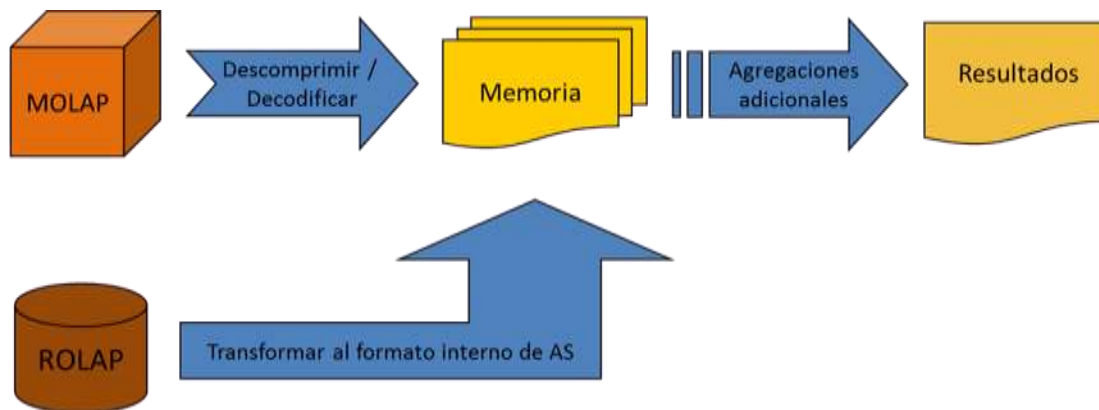


Figura 5-12 Recuperación de datos del servidor

Analysis Services permite ir seleccionando diversos sabores entre los extremos MOLAP y ROLAP en tiempo real, algunos de ellos basados en el uso de la caché automática (*proactive caching*). Tenemos una serie de configuraciones de almacenamientos estándar: ROLAP en tiempo real, HOLAP en tiempo real, MOLAP de baja latencia, MOLAP de latencia media, MOLAP automático, MOLAP programado y MOLAP. Además de poder realizar personalizaciones variando diversos parámetros sobre los modos citados. Puede ampliar información en la ayuda del producto.

Agregaciones (Aggregations)

Siempre que hablamos de este tipo de sistemas, aparece este término, incluso nos hemos visto en la necesidad de usarlo anteriormente al hablar del almacenamiento. Vamos a incluir aquí, tanto una definición como ejemplos de lo que son las agregaciones.

Las **agregaciones** son el núcleo del almacén analítico, con ellas obtenemos una enorme ganancia de rendimiento en las consultas, evitando leer gran número de celdas para obtener los resultados solicitados.

Son datos totalizados, ya pre-calculados y almacenados, para distintos niveles del cubo. Se guardan en celdas en las coordenadas especificadas por las dimensiones. No todas las agregaciones se calculan y almacenan, pero sí los metadatos de sus definiciones. El número de agregaciones tiene un crecimiento exponencial, por lo que el almacenamiento de todas ellas supondría también un crecimiento exponencial del tamaño del cubo y del tiempo de procesamiento.



Figura 5-13 Agregaciones (Aggregations)

Consideramos como datos base, los valores de cruce entre los niveles más bajos de cada dimensión, veamos un ejemplo:

- Dimensión Tiempo, jerarquía con niveles *Todos*, *Año*, *Mes*, *Día*.
- Dimensión Producto, jerarquía con niveles *Todos*, *Familia*, *SubFamilia* y *Producto*.
- Dimensión Geografía, jerarquía con niveles *Todos*, *País*, *Provincia* y *Población*.

En este caso el cruce de datos base sería a nivel de *Día*, *Producto* y *Población*. A partir de ahí, los niveles *Todos*, *Año* y *Mes* para el *Tiempo*; *Todos*, *Familia* y *SubFamilia* para el *Producto*; y *Todos*, *País* y *Provincia* para la *Geografía*, serán agregaciones que se pueden obtener a partir de los datos base. El *Año* lo obtendría sumando los 365 días o los 12 meses, según el diseño de agregaciones que tenga. El *Todos (de la dimensión Producto)*, se obtendría sumando todas las *Familias*, todas las *Subfamilias*, o todos los *Productos*, dependiendo igualmente de las agregaciones que tengamos almacenadas.

Como puede ver el lector es fundamental conocer el concepto de *Agregaciones*, y su diseño y almacenamiento. Debe tener también en cuenta que cuantas más agregaciones tengamos, más rápida será la ejecución de las consultas, y por el contrario más lento será el tiempo de procesamiento y mayor el tamaño necesario para el almacenamiento, dado que habrá un mayor número de celdas pre-calculadas

y almacenadas en el cubo. A fin de tener un balance entre el procesamiento de los cubos y el tiempo de respuesta, *Analysis Services* pre-calcula y guarda las agregaciones escogidas y genera las restantes al recibir la consulta.

No es el objetivo de este capítulo profundizar en el diseño de las agregaciones del cubo, sino simplemente que desde el principio, el lector tenga muy claro lo importante que es este concepto y las consecuencias que puede tener en el rendimiento de las consultas, en el tamaño de la base de datos y en los tiempos de procesamiento.

Personalizando el modelo de Analysis Services

Una vez introducidos los conceptos y elementos fundamentales, vamos a continuar profundizando en la creación de cubos y dimensiones, y sobre todo en la personalización de éstos y adaptación a las necesidades del negocio.

Dimensiones

Una vez vistos los datos a medir, pasemos a analizar las diferentes perspectivas desde las que los vamos a analizar. Vamos a conocer más a fondo esas dimensiones y sus características.

Las **Dimensiones** se basan en tablas o vistas del *Data Source View*. Se definen a nivel de base de datos, de forma independiente a los cubos. Y posteriormente, pueden instanciarse ninguna, una o varias veces en cada cubo. Por tanto pasaremos a explicar los conceptos de **dimensiones de bases de datos** y **dimensiones de cubos**.

Una **dimensión de base de datos** existe con independencia de que esté asociada o no a un cubo. Se basan en columnas de una o más tablas de dimensiones de nuestro *Data Source View*, que puede estar modelado en estrella (*star*) o en copo de nieve (*snowflake*).

Una **dimensión de cubo** es una instancia de una dimensión de base de datos en un cubo en particular, pudiendo tener varias instancias de ella, incluso en el mismo cubo. Por ejemplo, podemos tener una dimensión Tiempo a nivel de base de datos, y luego, en el cubo de Ventas, tener tres instancias: Fecha de pedido, Fecha de envío y Fecha de cobro. Esta múltiple instanciación es lo que se conoce como dimensión jugadora de roles (*role-playing dimension*). Cada instancia de una dimensión de base de datos puede tener propiedades distintas, por ejemplo, el nombre casi siempre lo cambiamos, tal y como hemos visto en el ejemplo anterior de la dimensión Tiempo, que juega los roles de Fecha de pedido, Fecha de envío y Fecha de cobro. Hay otras propiedades que también se pueden cambiar a nivel de dimensión de cubo, como su visibilidad (propiedad *visible*).

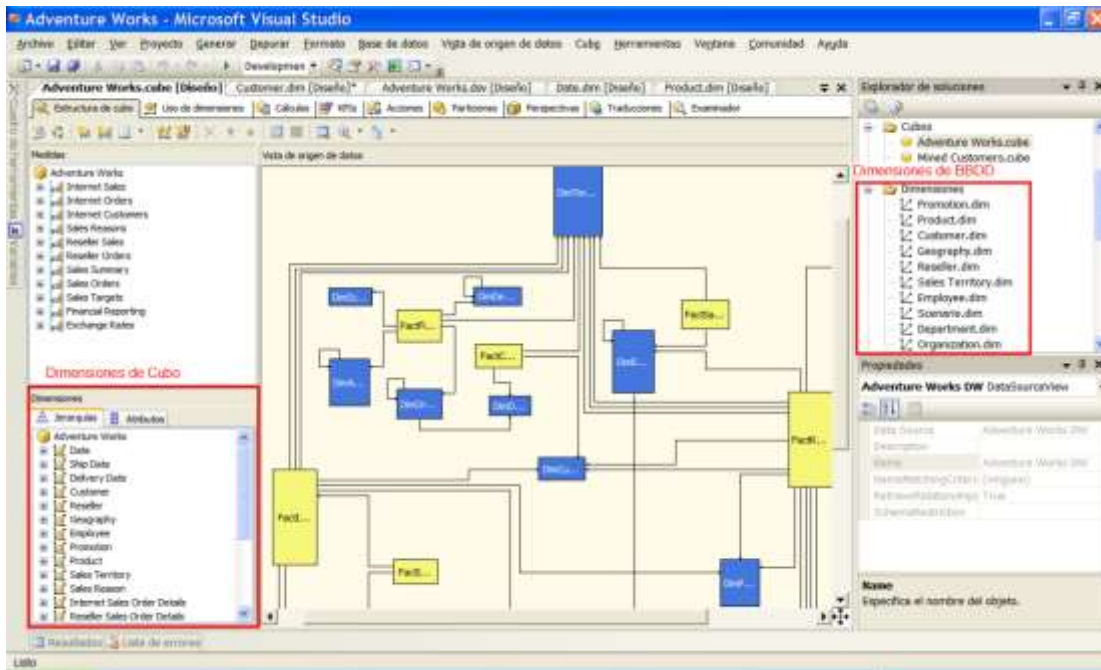


Figura 5-14 Dimensiones de Cubo y Dimensiones de Base de Datos

Algunas de las propiedades más significativas de las dimensiones son:

- **AttributeAllMemberName.** Podemos tener de manera opcional un nivel *All*, que está en la parte superior de la jerarquía y agrega todos los miembros del nivel inmediatamente inferior. Este es un miembro generado por el sistema, y su posible existencia dependerá de los valores que configuremos en la propiedad **IsAggregatable**, si se establece a *True* existirá un nivel *All*. Ese nivel podrá tener el nombre que indiquemos en la propiedad **AttributeAllMemberName**. Valores: el nombre que queramos que aparezca para ese nivel al navegar por la dimensión.
- **WriteEnabled.** Permite habilitar la dimensión para escritura. Valores: *True*, *False*.
- **Type.** Proporciona información a los servidores y a las aplicaciones cliente sobre el contenido de una dimensión. Se utiliza para *agregar inteligencia de dimensiones* a las aplicaciones. Valores: *Regular*, *Time*, *Geography*, *Organization*, *BillOfMaterials*, *Accounts*, *Customers*, *Products*, *Scenario*, *Quantitative*, *Currency*, *Rates*, *Channel*, *Promotion*. Estos valores se aplican a dimensiones y/o cubos.
- **StorageMode.** Indica el modo de almacenamiento de la dimensión. Valores: *ROLAP*, *MOLAP*.
- **ErrorConfiguration.** Indica la acción que realizará el servidor al encontrar diversos errores.
- **UnknownMember.** Permite especificar el tratamiento que recibirán los miembros desconocidos. Valores: *Visible*, *Hidden*, *None*.

- **UnknownMemberName.** Permite cambiar el valor predeterminado del miembro desconocido al que queramos. Por defecto el valor es *Unknown*.
- **ProactiveCaching.** Para la configuración del caché automático del cubo.

Atributos

Por cada una de las dimensiones nos encontramos con una colección de **Atributos** que se corresponden con una o más columnas de la tabla de dimensiones. Un atributo es cada una de las características de la dimensión por la cual podremos analizar la información. Por ejemplo, para la dimensión *Tiempo* tendremos atributos como *Año*, *Mes*, *Día*, *Trimestre*, *Trimestre fiscal*, *Año fiscal*, etc. Para la dimensión *Cliente* tendremos otros como *Nombre*, *Población*, *País*, *Estudios*, *Edad*, *Número de hijos*, *Sexo*, etc.

Entre todos ellos, debemos destacar que contaremos con un **atributo clave**, éste es un atributo con valores únicos, que identifica las columnas de la tabla de dimensiones con las claves externas de la tabla de hechos. Normalmente, representa a la columna que es clave principal (*primary key*) en la tabla de dimensiones.

Veamos las principales propiedades de los atributos:

- **KeyColumn.** Contiene la colección de columnas que constituyen la clave del atributo. Está compuesta por las columnas que hacen que el valor de cada miembro sea único. Por ejemplo, si tenemos en la tabla de Tiempo una columna para el año y otra para el mes, la colección *KeyColumn* estará formada por ambas columnas, ya que cada mes está repetido en cada uno de los años.
- **NameColumn.** Se utiliza para mostrar un valor descriptivo al usuario, cuando el valor el valor de la columna clave no es lo suficientemente descriptivo. Cuando se indica una columna en esta propiedad, será este valor el mostrado al usuario en lugar de *KeyColumn*.
- **ValueColumn.** Identifica la columna que proporciona el valor del atributo.
- **AttributeHierarchyDisplayFolder.** Identifica la carpeta en la que verán este elemento los usuarios finales desde las herramientas cliente.
- **DefaultMember.** Expresión MDX que define el miembro predeterminado del atributo.
- **OrderBy.** Especifica el método para ordenar los miembros del atributo.
Valores:
 - *Key*, los miembros se ordenan por la clave del atributo
 - *Name*, los miembros se ordenan por el nombre del atributo
 - *AttributeKey*, los miembros se ordenan por la clave del atributo especificado en la propiedad *OrderByAttribute*.

- **AttributeName**, los miembros se ordenan por el nombre del atributo especificado en la propiedad *OrderByAttribute*.
- **OrderByAttribute**. Si en *OrderBy* hemos decidido que se ordenen por otro atributo, es decir, hemos elegido el valor *AttributeKey* o *AttributeName*, en esta propiedad indicaremos el atributo por el que deseamos realizar la ordenación.
- **Type**. Especifica el tipo de información que contiene el atributo.
- **Usage**. Especifica el uso del atributo. Valores: *Regular*, *Key*, *Parent*.
- **IsAggregatable**. Indica si el atributo se puede agregar en una jerarquía.
- **DiscretizationMethod**. “Discretizar” consiste en convertir valores continuos en valores discretos. Por ejemplo, el salario es un valor continuo, puede haber cualquier valor, pero a nivel de análisis de información podemos convertirlo en continuo, haciendo tres grupos, el primero entre 0 y 10.000, el segundo entre 10.001 y 50.000 y el tercero para más de 50.000. Con la propiedad *DiscretizationMethod* elegimos el método a utilizar para este proceso. Valores: *None*, *Automatic*, *EqualAreas*, *Cluster*.
- **DiscretizationBucketCount**. Especifica el número de depósitos en los que se “discretizarán” los atributos. En el ejemplo anterior serían tres.
- **AttributeHierarchyEnabled**. Determina si se ha generado una jerarquía de atributos para ese atributo. Esto tiene implicaciones, ya que si no está habilitada no se puede utilizar ese atributo en una jerarquía definida por el usuario, ni se puede hacer referencia a la jerarquía de atributos desde instrucciones MDX.
- **AttributeHierarchyOptimizedState**. Determina el nivel de optimización aplicado a la jerarquía de atributos. Valores: *FullyOptimized*, *NotOptimized*. Por defecto está completamente optimizada, lo que implica que se generarán índices para optimizar las consultas. Es conveniente que no estén optimizadas aquellas jerarquías que no se utilicen para realizar consultas, y sólo estén para otros fines como la ordenación. Esto ahorrará tiempos de procesamiento.
- **AttributeHierarchyOrdered**. Indica si la jerarquía de atributos asociada está ordenada. Valores: *True*, *False*. Al igual que en el caso anterior, si no se va a utilizar para realizar consultas, mejor ponerla a *False*, y ahorrar tiempos de procesamiento innecesarios.
- **AttributeHierarchyVisible**. Indica si la jerarquía de atributos asociada está visible para las aplicaciones cliente. Valores: *True*, *False*. Se debe poner a *False* cuando no se utiliza para realizar consultas.

Jerarquías y Niveles

Es muy habitual que queramos organizar jerárquicamente los miembros de una dimensión, facilitando así rutas de navegación a través del cubo. Por ejemplo, para navegar a través de los miembros de la dimensión tiempo, es bastante habitual que

Microsoft Business Intelligence: vea el cubo medio lleno

lo hagamos en tres niveles: año, trimestre y mes, mostrándose los miembros tal y como se ve en la siguiente figura.

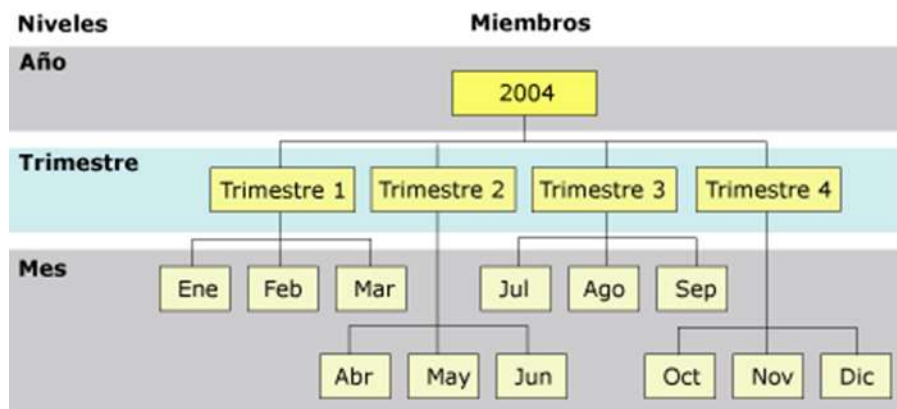


Figura 5-15 Jerarquía de la Dimensión Tiempo

En SQL Server disponemos para ello de las **jerarquías**, que nos permiten agrupar uno o más atributos organizados como niveles, a las que además podemos dar un nombre que más adelante será utilizado por el usuario para la navegación.

Hay dos tipos de jerarquías, las **jerarquías de atributos**, que son jerarquías de un solo nivel, que se crean automáticamente para cada atributo de la dimensión y que siempre tienen el mismo nombre que el atributo correspondiente; y las **jerarquías de varios niveles**, que se crean como jerarquías configuradas independientemente, y que son las que hemos visto en el ejemplo del tiempo y en las que nos vamos a centrar ahora.

Para crear estas jerarquías de varios niveles, simplemente tenemos que arrastrar los atributos desde el panel 'Atributos' al panel 'Jerarquías y niveles', y posteriormente hacer ciertos retoques en algunas propiedades que citaremos más adelante. Es posible definir varias jerarquías de varios niveles, incluso si en varias de ellas se utilizan los mismos atributos. Así para la dimensión Tiempo podremos definir una jerarquía con los niveles Año, Mes y Día, y otra con los niveles Año, Trimestre y Mes. O una jerarquía para la dimensión Producto con los niveles Talla, Color, Producto, y otra con Color, Talla, Producto.

Podemos clasificar las jerarquías en dos tipos:

- **Jerarquías naturales.** Cuando cada atributo incluido en la jerarquía es una propiedad de miembro del atributo situado inmediatamente por debajo. Por ejemplo: Familia, Subfamilia y Producto
- **Jerarquías de navegación.** No cumplen lo anterior, aunque se definen para que sean utilizadas para mejorar la navegación por el cubo. Por ejemplo: Color, Talla y Producto, y por otro lado, Talla, Color y Producto.

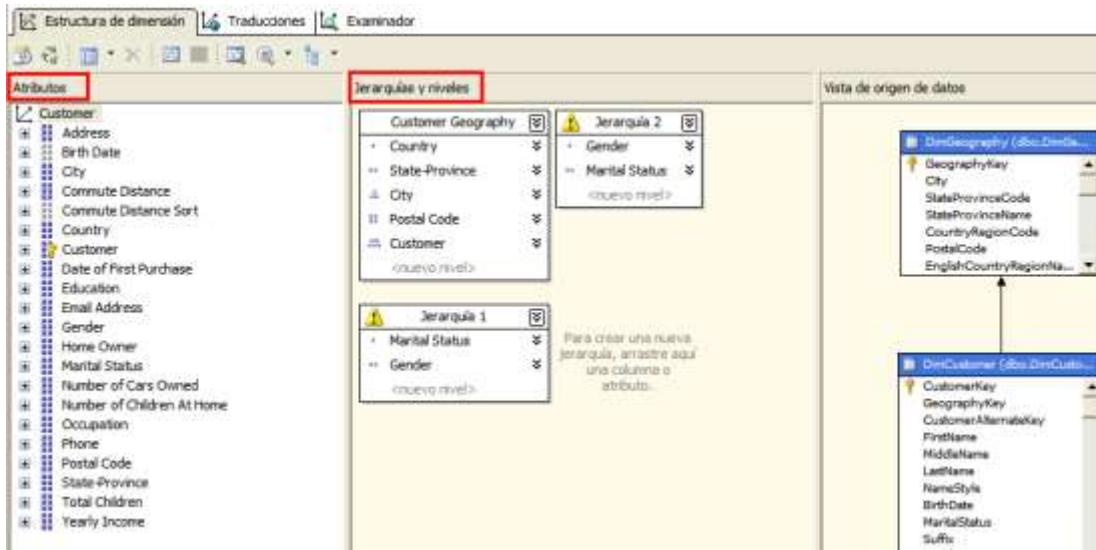


Figura 5-16 Atributos, Jerarquías y Niveles

Durante el procesamiento de los atributos, *Analysis Services* optimiza el almacenamiento y las agregaciones aprovechando las jerarquías naturales. Veremos más adelante cuando estudiemos “las relaciones entre atributos” cómo realizar un buen diseño de ellas para conseguir dicha optimización.

A continuación vamos a ver algunas propiedades de las *jerarquías de dimensiones de base de datos*:

- **AllMemberName.** Especifica el nombre del nivel *All* para dicha jerarquía.
- **DisplayFolder.** Especifica la carpeta en la que aparecerá la jerarquía en las herramientas cliente, facilitando así la clasificación y presentación al usuario.

Propiedades de los Niveles:

- **HideMemberIf.** Permite ocultar un nivel bajo determinadas condiciones. Permite crear jerarquías desiguales. Valores: *Never*, *OnlyChildWithNoName*, *OnlyChildWithParentName*, *NoName*, *ParentName*.
- **Name.** Nombre del nivel.

Y algunas propiedades de las *jerarquías de dimensiones de cubo*:

- **Enabled.** Indica si la jerarquía está habilitada en esa dimensión de cubo. Valores: *True*, *False*.
- **OptimizedState.** Indica si la jerarquía se optimiza en esta dimensión de cubo. Valores: *FullyOptimized*, *NonOptimized*.
- **Visible.** Indica si la jerarquía está visible en esta dimensión de cubo. Valores: *True*, *False*.

Relaciones entre Atributos

En *Analysis Services* todos los atributos de una dimensión están relacionados directa o indirectamente con el atributo clave. La mayor parte de los atributos suelen estar relacionados directamente con el atributo clave, sin embargo también nos encontramos con otra serie de atributos que se relacionan jerárquicamente con dicho atributo clave. Es el caso de las jerarquías naturales, cuyos atributos se relacionan unos con otros en función de los niveles que tenga, y ya el último nivel es el que se relaciona con el atributo clave. Veamos un ejemplo: si en la dimensión Cliente tenemos un atributo clave llamado Cliente y una jerarquía natural formada por los niveles País, Provincia, Población y Cliente, el país se relaciona con la provincia, la provincia con la población, y la población con el cliente.

Cuando definimos relaciones entre atributos debemos conocer y configurar correctamente las siguientes propiedades:

- **Cardinality.** Determina la “cardinalidad” de la relación. Valores:
 - *Many*, para relaciones de varios a uno.
 - *One*, para relaciones de uno a uno.
- **RelationshipType.** Indica si las relaciones de los miembros cambian a lo largo del tiempo. Valores:
 - *Rigid*, no cambian a lo largo del tiempo. Por ejemplo, la relación entre año y mes no cambia, un mes de un año, nunca pasará a ser de otro año. Cuando se realice una actualización incremental se mantendrán las agregaciones, produciendo un error si hubiese algún cambio.
 - *Flexible*, cambian a lo largo del tiempo. Por ejemplo, un cliente puede vivir en una provincia, y con el tiempo ir a vivir a otra provincia.
- **Visible.** Determina la visibilidad de la relación de los atributos.

La configuración de estas propiedades y la correcta definición de relaciones entre atributos, permitirá obtener un almacenamiento interno optimizado, lo que mejorará el tiempo de procesamiento, el tamaño del almacenamiento y el tiempo de respuesta de las consultas que reciba.

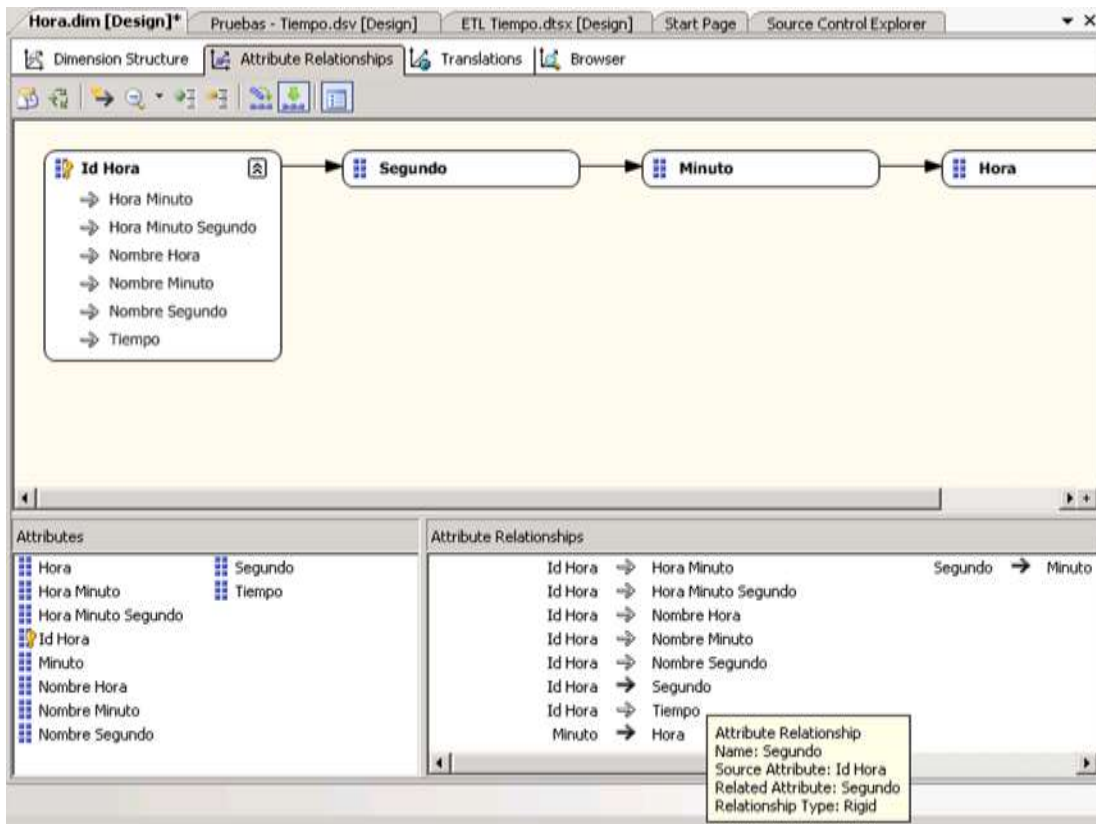


Figura 5-17 Dimensión Hora. Relaciones entre Atributos

Nota: La imagen anterior es de *Analysis Services 2008*. El nuevo diseñador de relaciones entre atributos (*attribute relationships*) es una de las mejoras en usabilidad con respecto a 2005. Puede comprobar que esta pestaña no está disponible en la imagen "Atributos, Jerarquías y Niveles" obtenida con la versión 2005.

Veamos un ejemplo, basado en la dimensión Hora, para entender lo comentado anteriormente. Supongamos que en la dimensión Hora tenemos, entre otros, los atributos Hora, Minuto, Segundo, y que IdHora es el atributo clave. Si definimos una relación de cada atributo con el atributo clave, Hora estará relacionado con IdHora, Minuto estará relacionado con IdHora, y Segundo también estará relacionado con IdHora. En ese caso, cuando se produce el diseño del almacenamiento, el motor entiende que para obtener las agregaciones a nivel *Todos (All)* debe sumar las celdas de todos los Segundos de las 24 Horas del día, por tanto sumar 86400 celdas; para obtener las agregaciones de cada Hora debe sumar todos los segundos, 3600 celdas; y así sucesivamente. En ningún momento es conocedor de que el nivel *Todos (All)*, que son las 24 Horas del día, es la suma de las horas, una Hora la suma de los minutos, y un Minuto la suma de los segundos. Por tanto el tiempo de respuesta será menor, tendrá que leer más celdas en el caso de que los datos solicitados por la

consulta no estén pre-calculados y almacenados. En cambio si definimos de forma optimizada las relaciones entre los atributos, indicaremos que el nivel *Todos (All)* está relacionado con la Hora, que la Hora está relacionado con los Minutos, y que los Minutos están relacionados con los Segundos. Además, dado que estas relaciones entre atributos de la dimensión Hora no cambiarán nunca, indicaremos que son relaciones *rígidas*. A partir de este momento el motor, cuando vaya a diseñar el almacenamiento tendrá información para conocer que para obtener la información de un Día le basta con sumar las 24 celdas de sus Horas, para obtener la información de una Hora sumará las 60 celdas de sus Minutos, y así sucesivamente. Obteniendo así un almacenamiento más optimizado, y las mejoras de rendimiento en las consultas comentadas anteriormente.

Cubos

Vamos a comenzar a estudiar más a fondo las medidas y grupos de medidas.

Medidas y Grupos de Medidas

Una **medida** es un hecho cuantificable que queremos analizar. Generalmente representa una columna de la tabla de hechos, habitualmente con datos numéricos, y por tanto cuantificables y agregables. También puede definirse una medida calculada utilizando una expresión MDX basada en otras medidas del cubo.

Un **grupo de medidas** es un conjunto de medidas, que procede generalmente de una misma tabla de hechos. Se utilizan para asociar dimensiones a medidas, como veremos más adelante.

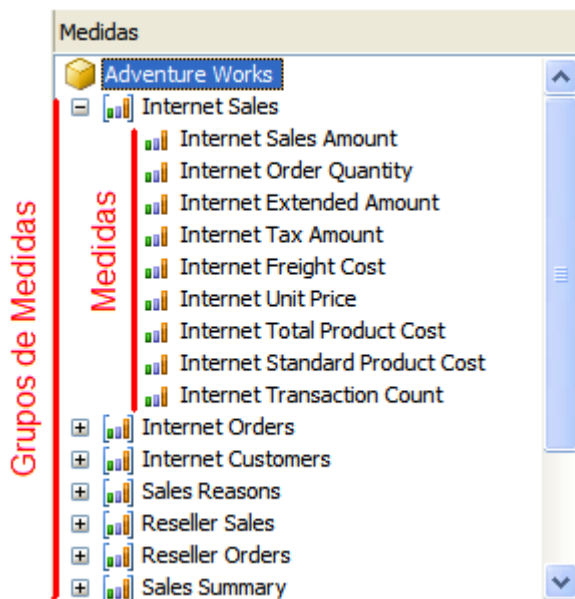


Figura 5-18 Medidas y Grupos de Medidas

Propiedades de los Grupos de Medidas

A continuación vamos a enumerar y definir algunas de las propiedades más significativas de los grupos de medidas:

- **ErrorConfiguration.** Permite configurar el comportamiento de los errores. Puede tener dos valores: *Predeterminada*, *Personalizada*. El valor por defecto es *Predeterminada*. Si lo cambiamos a *Personalizada*, aparecerán otra serie de propiedades para configurar qué acción realizar ante diferentes casos, como aparición de duplicados, de miembros desconocidos, aparición de errores, etc. Para ello tenemos disponibles las propiedades *KeyDuplicate*, *KeyErrorAction*, *KeyErrorLimit*, *KeyErrorLogFile*, *KeyErrorLimitAction*, *KeyNotFound*, *NullKeyConvertedToUnknown* y *NullKeyNotAllowed*.
- **StorageLocation.** Especifica la ubicación de almacenamiento para el grupo de medidas. Si no se especifica ninguna ubicación, se almacena en la misma que la base de datos en la que está este cubo.
- **ProactiveCaching.** Configuración de almacenamiento en caché automático para el grupo de medidas.
- **StorageMode.** Modo de almacenamiento predeterminado para el grupo de medidas. Por defecto tendrá el mismo modo de almacenamiento que el cubo.

Propiedades de las Medidas

Haremos lo mismo para las medidas, citando las propiedades más significativas y poniendo ejemplos de uso:

- **Name.** Especifica el nombre del objeto, en este caso de la medida.
- **Source.** Especifica los detalles del enlace (*binding*) de la medida a la columna del *Data Source View* que contiene dicha medida.
- **FormatString.** Permite elegir el formato en el que se presentará la información en el cliente. Puede utilizar algunos formatos predeterminados, como *Currency*, *C2*, *Standard*, ... y otros personalizados como *##*, *###0.00* ... Permite todas las expresiones de formato disponibles en MS Visual Basic.
- **MeasureExpression.** Define la fórmula utilizada para calcular el valor de la medida, basándose en otras medidas, que pueden ser del mismo o de otro grupo de medidas. Tiene las siguientes restricciones: sólo pueden intervenir dos medidas, y sólo pueden intervenir los operadores de multiplicación “*” y división “/”.
- **DisplayFolder.** Permite especificar la carpeta en la que aparecerá la medida en las herramientas cliente, facilitando así la clasificación y presentación al usuario.

- **Visible.** Determina si esta medida estará visible o no para el usuario. A veces nos encontramos con medidas que no queremos mostrar, pero sí utilizarlas, por ejemplo, en otros cálculos.
- **AggregateFunction.** Determina cómo se agregan las medidas. Dado que es un punto más amplio que los anteriores pasaremos a describirlo a continuación.

Analysis Services proporciona diversas **funciones de agregación**, que nos permiten agregar la información en las dimensiones. De manera predeterminada, la función de agregación es la suma, pero este comportamiento se puede modificar a través de la propiedad *AggregateFunction* vista anteriormente.

Según el grado de agregación, podemos clasificar estas funciones en:

- **Aditivas:** se pueden agregar, sin restricciones, en todas las dimensiones que están incluidas en el grupo de medidas. Funciones aditivas:
 - Sum
- **Semi-aditivas:** se pueden agregar sólo en algunas de las dimensiones que están incluidas en el grupo de medidas, pero no en todas. Veamos un ejemplo de lo que significa semi-aditividad: el stock disponible puede agregarse para la dimensión Geografía, ya que tiene sentido que sumemos el stock de todos nuestros almacenes a una fecha dada; pero no para la dimensión Tiempo, no tiene sentido que sumemos el stock de varios días. Funciones semi-aditivas:
 - Count
 - Min / Max
 - ByAccount
 - AverageOfChildren
 - FirstChild / LastChild / FirstNonEmpty / LastNonEmpty
- **No aditivas:** no se pueden agregar para ninguna dimensión incluida en el grupo de medidas. Funciones no aditivas:
 - DistinctCount
 - None

Otros componentes del cubo

Con lo visto hasta el momento, conocemos los conceptos básicos para la creación de un cubo simple, basado en una tabla de hechos y varias dimensiones. A partir de ahora vamos a seguir citando otra serie de componentes que tenemos disponibles en nuestros proyectos, pero que, dado el nivel y el objetivo de este artículo, serán tratados con menos profundidad. Por supuesto, no quiere decir que sean menos importantes. Con ellos podremos gestionar de forma eficaz las diversas tablas de hechos en cubo, y establecer las relaciones entre las dimensiones con las diferentes

tablas de hechos, cubriendo una casuística mayor. Iremos un paso más allá de tener una simple medida, procedente de una columna de nuestra tabla de hechos, utilizando el lenguaje MDX y otras funcionalidades que tenemos disponibles que se suelen utilizar con frecuencia.

Relaciones entre Dimensiones y Grupos de Medidas (Dimension Usage)

A partir de *Analysis Services 2005* se produce un gran cambio en el producto. Entre otras muchas mejoras, ya permite que un cubo pueda tener más de una *tabla de hechos*, y aparece un nuevo concepto que es el de las relaciones entre los *grupos de medidas* y las *dimensiones*.

Dimensiones	Internet Sales	Reseller Sales	Sales Summary	Sales Targets
Date	Date	Date	Date	Calendar Quarter
Date (Ship Date)	Date	Date	Date	
Date (Delivery Date)	Date	Date	Date	
Customer	Customer			
Reseller		Reseller		
Geography		Reseller		
Employee		Employee		Employee
Promotion	Promotion	Promotion	Promotion	
Product	Product	Product	Product	
Sales Territory	Sales Territory Region	Sales Territory Region	Sales Territory Region	Employee
Sales Reason	Sales Reasons			
Internet Sales Order Details	Internet Sales Order			
Reseller Sales Order Details		Reseller Sales Order		

Figura 5-19 Relaciones entre Dimensiones y Grupos de Medidas

Como hemos comentado anteriormente, no vamos a entrar a detalle, pero sí que vamos a explicar los tipos de relaciones que tenemos disponibles:

- **Normal (Regular).** Relaciona directamente una *tabla de hechos* con una *tabla de dimensiones*, a través de una o más columnas. Es el caso más habitual.
- **Referenciada (Referenced).** Una dimensión se relaciona con un grupo de medidas a través de otra dimensión. Se suele utilizar cuando modelamos en copo de nieve (*snowflake*).
- **Varios a Varios (Many-to-Many).** La tabla de dimensiones se combina con una tabla de hechos intermedia, y ésta a su vez se combina con una tabla de dimensiones intermedia con la que la tabla de hechos está combinada.

- **Hecho (Fact).** Se basan en la propia tabla de hechos, y el atributo clave de la dimensión será una columna de la tabla de hechos. Por ejemplo, si en la tabla de hechos tenemos el número de factura y queremos una dimensión Factura.

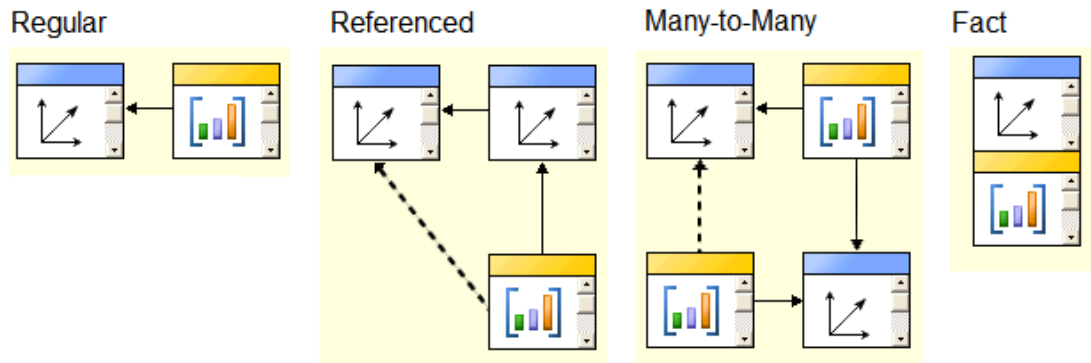


Figura 5-20 Tipos de Relaciones entre Dimensiones y Grupos de Medidas

Cálculos (Calculations) y KPIs

El MDX (*MultiDimensional Expressions*) es un lenguaje de consulta de bases de datos multidimensionales. A parte de permitir hacer consultas a la base de datos para obtener información almacenada en ella también se utiliza para la definición de *cálculos* (*miembros calculados, conjuntos con nombre y asignaciones de ámbito*) y *KPIs* en el servidor. Permitiendo añadir objetos que se calculan en tiempo de ejecución, y no están definidos en el cubo por ninguna otra vía, pudiendo referenciar parte del cubo, otros cubos o incluso información externa a los cubos.

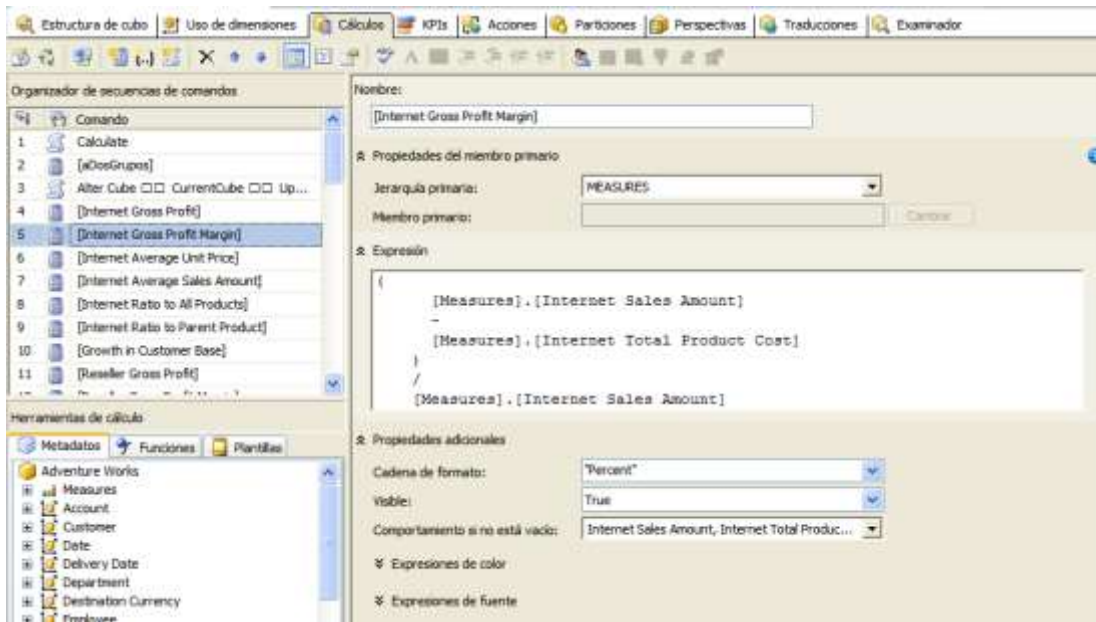


Figura 5-21 Cálculos (Calculations)

Hay ocasiones en las que no es suficiente con dar un valor, sino que queremos saber en qué contexto está ese valor, si es bueno ese resultado con respecto a un objetivo y qué tendencia sigue a lo largo del tiempo. Veámoslo con un ejemplo: tenemos las ventas de un comercial para el mes de Junio por 700.000€, pero este dato no nos es suficiente, ya que necesitamos saber si están bien, regular o mal, y para ello debemos indicarle en función de qué. En este caso imaginemos que hemos fijado un *objetivo* de ventas para ese comercial y ese mes de 900.000€, consideramos que unas ventas inferiores a 500.000€ es un mal resultado, que unas ventas entre 500.000€ y 750.000€ es un resultado regular, y que si se superan los 750.000€ es un buen resultado. En ese caso hemos obtenido un resultado regular y cercano a bueno, con respecto al objetivo fijado. Independientemente de este valor, las ventas de este comercial para el mes de junio, pueden haber subido, bajado, o haberse mantenido con respecto a un periodo anterior, por ejemplo el mes de mayo (o mes de junio del año anterior, depende de lo que queramos analizar), este dato nos indicará como ha sido la tendencia de ventas. Adicionalmente llevan asociados una serie de gráficos, como semáforos, cuenta-revoluciones, termómetros, emoticonos, flechas de tendencia y demás que nos permiten, además de representar el valor, asignarle una imagen. Un *KPI* está compuesto por: un *valor*, un *objetivo*, un *estado* y una *tendencia*.

Pues bien, utilizando el lenguaje MDX comentado anteriormente y la definición de *KPI* (Key Performance Indicator) que viene incluida en *Analysis Services* podremos crearlos en el servidor, al igual que los *cálculos*, para que sean calculados en tiempo de ejecución y devuelvan los resultados al cliente.

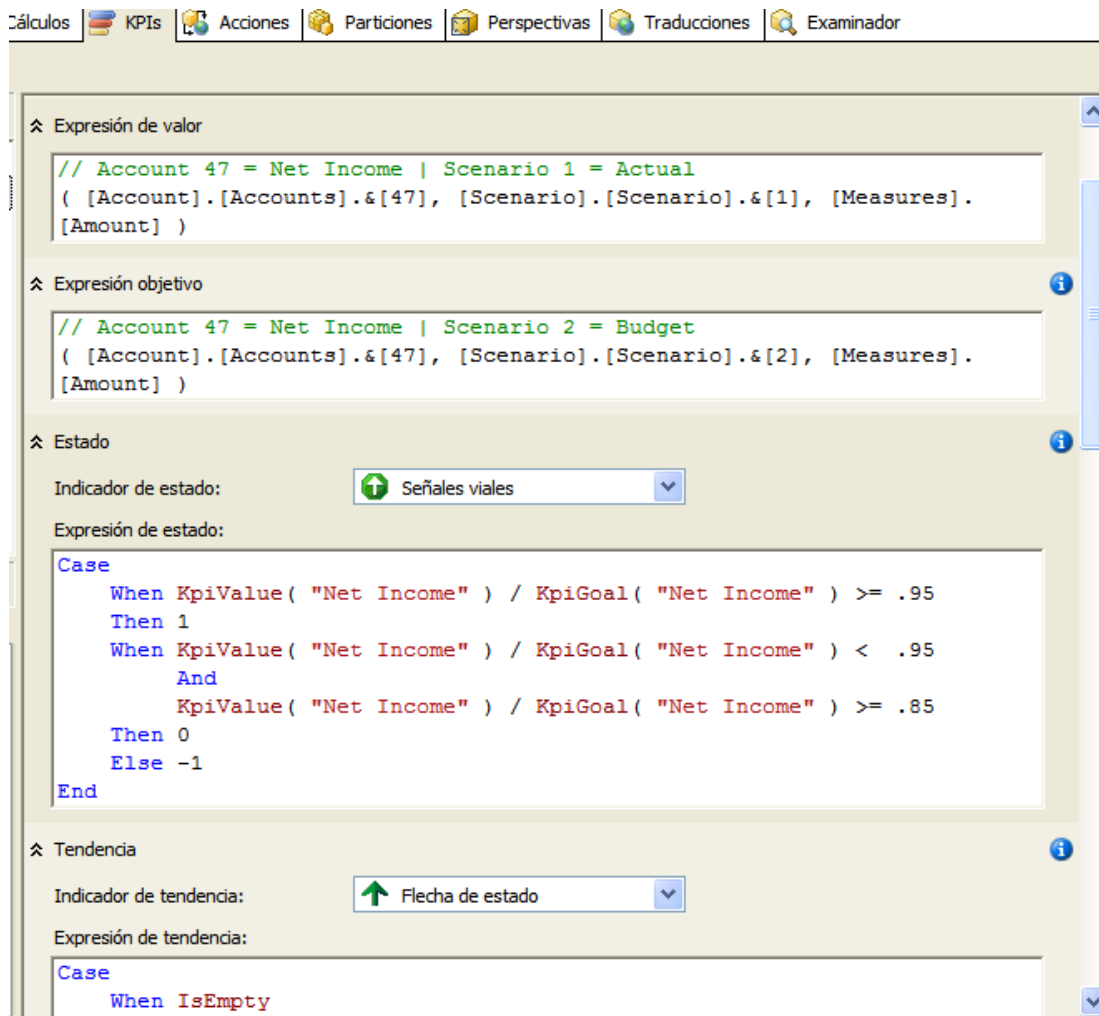


Figura 5-22 KPIs

Aún hay más

Finalmente, vamos a citar otros componentes que tenemos disponibles, para que el lector tenga conocimiento de su existencia, y si lo estima oportuno, profundice en su estudio:

Una **Acción** (*action*) es una instrucción MDX almacenada en el servidor, y que puede ser ejecutada por las aplicaciones cliente. Esto es muy flexible, y permite a la aplicación de BI ir más allá del cubo, ofreciendo la posibilidad de transformar las aplicaciones cliente en sofisticadas herramientas integradas en la operativa de la empresa. Tenemos los siguientes tipos de acciones:

- **Dirección URL.** Muestra una página en el explorador de internet.
- **Instrucción.** Ejecuta un comando Ole Db.
- **Conjunto de filas.** Devuelve un conjunto de filas a una aplicación cliente.

- **Propietario.** Realiza una operación mediante una interfaz distinta a las descritas aquí.
- **Conjunto de datos.** Devuelve un conjunto de datos a una aplicación cliente.
- **Obtención de detalles.** Devuelve una instrucción de obtención de detalles (*drill through*) a una aplicación cliente.
- **Informes.** Envía una solicitud con parámetros basada en una URL al servidor de *Reporting Services* y devuelve un informe a una aplicación cliente.

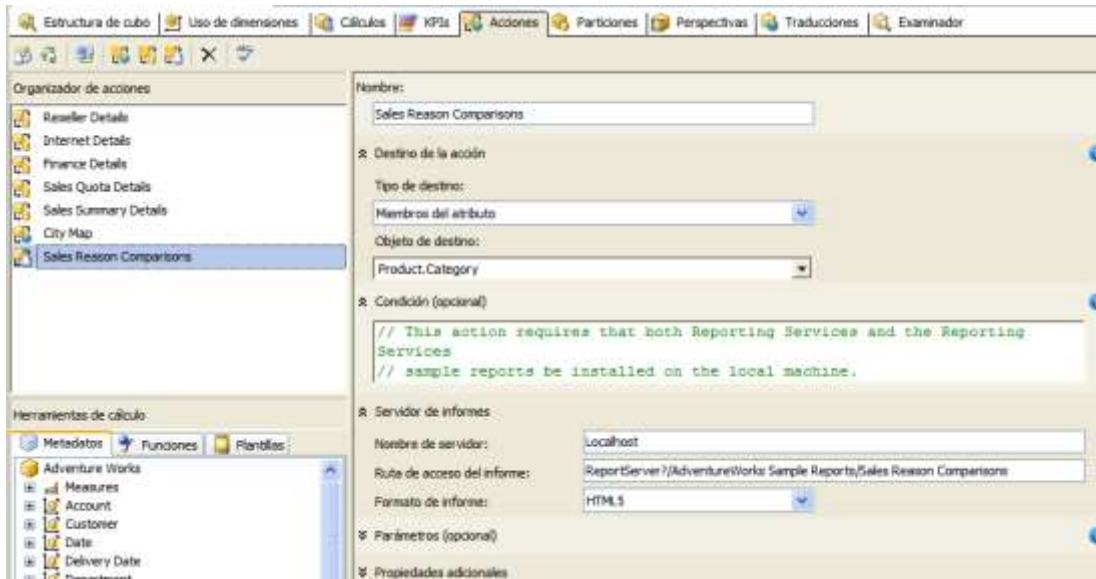


Figura 5-23 Acciones (Actions)

Una **Perspectiva** (*perspective*) es un subconjunto de un cubo creado para una aplicación o grupo de usuarios. Permite que se acceda a un subconjunto de los datos, facilitando así el análisis de la información, evitando confusiones, y eliminando toda aquella información que el usuario no deba conocer o utilizar. Tenga en cuenta el lector, que las perspectivas son simplemente un filtro que facilita la navegación por un subconjunto del cubo, pero en ningún caso una medida de seguridad que restrinja el acceso a dicha información.

Estructura de cubo Uso de dimensiones Cálculos KPIs Acciones Particiones Perspectivas			
Objetos de cubo	Tipo de objeto	Nombre de perspectiva	Nombre de perspectiva
Adventure Works	Name	Direct Sales	Channel Sales
Reseller Sales Amount	DefaultMeasure	Internet Sales Amount	Reseller Sales Amount
Grupos de medida			
Internet Sales	MeasureGroup	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Sales Amount	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Order Quantity	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Extended Amount	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Tax Amount	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Freight Cost	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Unit Price	Measure	<input type="checkbox"/>	<input type="checkbox"/>
Internet Total Product Cost	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Standard Product Cost	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Transaction Count	Measure	<input type="checkbox"/>	<input type="checkbox"/>
Internet Orders	MeasureGroup	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Order Count	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Internet Customers	MeasureGroup	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Customer Count	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sales Reasons	MeasureGroup	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sales Reason Count	Measure	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Reseller Sales	MeasureGroup	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Reseller Sales Amount	Measure	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Reseller Order Quantity	Measure	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 5-24 Perspectivas (Perspectives)

Como se aprecia en la figura anterior, podemos crear una perspectiva, asignarle un nombre, y marcar el *checkbox* de cada uno de los elementos que queramos tener disponibles en ella.

Una **Traducción** (*translation*) nos permite gestionar la compatibilidad multilingüe de *Analysis Services*. Contiene un identificador de idioma y enlaces a las propiedades de los objetos que se pueden mostrar en varios idiomas. Tenemos disponibles traducciones tanto a nivel del cubo, como de las dimensiones. Con las traducciones a nivel de dimensiones podremos incluso que ciertos valores aparezcan en un idioma según los valores almacenados en la base de datos, por ejemplo, si en la dimensión producto tenemos una columna con su nombre en inglés y otra con él en español, podemos hacer, mediante el uso de las traducciones se muestre al usuario dicho nombre en un idioma u otro.

Estructura de cubo Uso de dimensiones Cálculos KPIs Acciones Particiones Perspectivas Traducciones			
Idioma predeterminado	Tipo de obj...	Español (España)	Francés (Francia)
Adventure Works	Caption	Adventure Works	Adventure Works
Grupos de medida			
Internet Sales	Caption	Ventas por Internet	Ventes par Internet
Internet Sales Amount	Caption	Cantidad de Ventas por Internet	Montant des Ventes (Internet)
Internet Order Quantity	Caption	Cantidad de Ordenes por Internet	Nombre de Commandes (Internet)
Internet Extended Amo...	Caption	Cantidad Extendida por Internet	Montant Étendu (Internet)
Internet Tax Amount	Caption	Cantidad de Impuesto por Internet	Taxes (Internet)
Internet Freight Cost	Caption	Costo de Flete por Internet	Coût de Transport (Internet)
Internet Unit Price	Caption	Precio Unitario por Internet	Prix Unitaire (Internet)
Internet Total Product ...	Caption	Costo Total del Producto por Internet	Coût Total Produit (Internet)
Internet Standard Prod...	Caption	Costo Estándar del Producto por Internet	Coût Standard Produit (Internet)
Internet Transaction C...	Caption	Cuenta de Transacción por Internet	Nombre de Transactions (Internet)
Internet Orders	Caption	Ordenes por Internet	Commandes par Internet
Internet Order Count	Caption	Conteo de Ordenes por Internet	Nombre de Commandes (Internet)
Internet Customers	Caption	Cientes del Internet	Clients (Internet)
Customer Count	Caption	Cuenta de Clientes	Nombre de Clients
Sales Reasons	Caption	Razón de Ventas	Raisons de Ventes
Sales Reason Count	Caption	Conteo de Razones de Ventas	Nombre de Raisons de Vente
Reseller Sales	Caption	Ventas de Revendedor	Ventes par Revendeur
Reseller Sales Amount	Caption	Monto de Ventas del Revendedor	Montant des Ventes (Revendeur)
Reseller Order Quantity	Caption	Cantidad de Orden del Revendedor	Nombre de Commandes (Revendeur)

Figura 5-25 Traducciones (Translations)

Los **Roles** (*funciones*) nos permiten gestionar la seguridad del cubo. A través de ellos decidimos qué usuarios y grupos de usuarios pueden, o no pueden, acceder a los datos. Es posible definir la seguridad, incluso a nivel de miembro y de celda del cubo.

Conclusiones sobre Analysis Services

Hemos comentado los conceptos más importantes, aquellos que deben ser conocidos por cualquier persona que se introduzca en el mundo multidimensional a través de *SQL Server*. Muchos de ellos han sido comentados muy de pasada, más con el objetivo de que el lector conozca su existencia que de conocerlos a fondo. Esto ha sido debido al ámbito y tamaño de este capítulo, dado que el desarrollo en más profundidad de todo lo aquí citado da para escribir incluso varios libros.

Espero que le sirva como base a cualquier lector que desee introducirse en la creación de bases de datos multidimensionales con *Analysis Services*.

Laboratorio incluido en el curso

Si está siguiendo el curso, del cual este eBook es material complementario, acceda al material del '**Lab SSAS 01A**', realice paso a paso todo lo allí expuesto y responda a las preguntas que se incluyen. Este Lab le ayudará a asentar todos los conocimientos adquiridos sobre Analysis Services, y por supuesto, si tiene cualquier duda sobre lo visto o cualquier situación que quiera resolver puede utilizar las sesiones de **tutorías** y los **foros** del curso para resolverlas.

6. Figuras

Figura 1-1 Esquema en estrella (star schema)	13
Figura 1-2 Esquema de copo de nieve (snowflake schema)	14
Figura 1-3 ETL	16
Figura 1-4 Cubos OLAP	18
Figura 1-5 KPIs	19
Figura 1-6 Solución de BI	20
Figura 2-1 Componentes Microsoft BI	24
Figura 2-2 Tipos de proyectos Business Intelligence en BIDS	25
Figura 2-3 SQL Server Management Studio	26
Figura 3-1 Tabla Dimensión Producto (star schema)	28
Figura 3-2 Tabla Dimensión Producto	32
Figura 3-3 Tabla de Hechos de Ventas	33
Figura 4-1 Integration Services	35
Figura 4-2 SSIS: Un arma de doble filo	36
Figura 4-3 Relación completa de tareas del ControlFlow	39
Figura 4-4 ControlFlow: contenedores, tareas y restricciones de precedencia	40
Figura 4-5 Relación completa de elementos del DataFlow	44
Figura 4-6 ControlFlow vs DataFlow	45
Imagen 4-1 Carga de una tabla en el área de Staging. Control Flow	52
Imagen 4-2 Tratamiento de filas nuevas y modificadas – DataFlow	53
Imagen 4-3 Tratamiento de filas eliminadas – DataFlow	53
Figura 4-7 Habilitar Registro (Logging)	61
Figura 4-8 Selección de Tipo de Proveedor	61
Figura 4-9 Selección de Conexión de Configuración	62
Figura 4-10 Selección de eventos	63
Figura 4-11 Personalización de columnas de cada evento	63
Figura 4-12 Estructura de la tabla 'dbo.sysssislog'	64
Figura 4-13 Ubicación del Script task	65
Figura 4-14 Variables de sólo lectura	65
Figura 4-15 Habilitar el Registro personalizado	66
Figura 4-16 Salid de Registro de Configuración personalizable	67
Figura 4-17 Acceso a datos de configuración	69
Figura 4-18 Entorno de Desarrollo	70
Figura 4-19 Paquete "Importar Empleados"	70
Figura 4-20 DataFlow del paquete "Importar Empleados"	71
Figura 4-21 Entorno de Producción	72
Figura 4-22 Seleccionar tipo de configuración, archivo XML	73
Figura 4-23 Selección de propiedades a almacenar en el archivo configuración, archivo XML	74
Figura 4-24 Seleccionar tipo de configuración, SQL Server	75
Figura 4-25 Selección de propiedades a almacenar en el archivo de configuración, SQL Server	76

Figura 4-26 Resultado del organizador de configuraciones de paquetes	77
Figura 4-27 Información almacenada en la tabla "dbo.[SSIS Configurations]"	77
Figura 4-28 Propiedades de Configuración de paquetes.....	79
Figura 4-29 Conexión al servidor de Integration Services desde SSMS.....	80
Figura 4-30 Opción Importar paquete... desde el explorador de objetos.....	81
Figura 4-31 Datos del paquete a importar.....	81
Figura 4-32 Utilidad de ejecución de paquetes, menú 'General'	82
Figura 4-33 Utilidad de ejecución de paquetes, menú 'Configuraciones'	83
Figura 4-34 Nuevo Trabajo en el Agente de SQL Server	84
Figura 4-35 Configuración del primer paso (y único) de este Trabajo	85
Figura 4-36 Pestaña 'Configuraciones'	86
Figura 4-37 Programación del Trabajo.....	87
Figura 5-1 Cubos	91
Figura 5-2 Dimensiones, Jerarquías, Niveles, Miembros.....	93
Figura 5-3 Medidas	94
Figura 5-4 Modelo relacional en Estrella	95
Figura 5-6 Objetos de una Base de Datos	96
Figura 5-7 Data Sources, Data Source View y Cubo.....	98
Figura 5-8 Componentes del Cubo	99
Figura 5-10 Dimensión Tiempo	100
Figura 5-11 Dimension Producto	101
Figura 5-12 Recuperación de datos del servidor	104
Figura 5-13 Agregaciones (Aggregations)	105
Figura 5-14 Dimensiones de Cubo y Dimensiones de Base de Datos.....	108
Figura 5-15 Jerarquía de la Dimensión Tiempo	111
Figura 5-16 Atributos, Jerarquías y Niveles	112
Figura 5-17 Dimensión Hora. Relaciones entre Atributos	114
Figura 5-18 Medidas y Grupos de Medidas	116
Figura 5-19 Relaciones entre Dimensiones y Grupos de Medidas	118
Figura 5-20 Tipos de Relaciones entre Dimensiones y Grupos de Medidas.....	119
Figura 5-21 Cálculos (Calculations)	120
Figura 5-22 KPIs.....	121
Figura 5-23 Acciones (Actions).....	122
Figura 5-24 Perspectivas (Perspectives)	123
Figura 5-25 Traducciones (Translations).....	124



Acerca del Autor

Salvador Ramos

Mentor del área de Business Intelligence en **SolidQ**. En esta última etapa profesional ha participado en numerosos proyectos de BI para empresas de diversos sectores (asegurador, energía, comercio minorista, entidades públicas, automoción, turismo, etcétera) desempeñando los roles de consultor, formador y mentor. Tiene más de 20 años de experiencia en el sector de las tecnología de la información, habiendo desempeñado diversos cargos (director de TI, consultor, jefe de proyectos, analista, desarrollador y formador). Trabaja con SQL Server desde la versión 6.5. Certificaciones: MCP, MCTS y MCITP.

Colabora como docente en diversos cursos y masters en universidades españolas. Lo podéis seguir en su blog www.sqlserversi.com. Es presidente del grupo de usuarios Gusenet y co-fundador de PASS Spanish Group. Participa en eventos y talleres organizados por Microsoft o por grupos de usuarios, es speaker de INETA España y Latam. Escribe en diversas revistas del sector y es coautor de varios libros.

Es SQL Server MVP desde 2.003. MVP Bio:

<http://mvp.support.microsoft.com/profile/salvador>



Acerca de SolidQ

SolidQ es el estándar para servicios de administración de información al proporcionar servicios de administración de datos fiables, inteligencia de negocios, colaboración y soluciones de tecnología avanzada para plataformas de nube y en las instalaciones de Microsoft más confiable del mundo.

- SolidQ representa el más altamente reconocido equipo de BI y administración de datos con el mayor porcentaje de MVP de Microsoft (SQL Server) en el mundo.
- Como un asesor clave y proveedor de contenido a Microsoft, con más de treinta libros publicados y más apariciones de orador en conferencias técnicas, líderes del mundo, excepto Microsoft, SolidQ es el experto llamado por los expertos.
- Como confiamos en la capacidad de nuestro equipo y el valor que ofrecemos, incondicionalmente garantizamos su satisfacción con nuestros servicios.