Diseño de procesos ETL

Josep Curto Díaz

PID_00238514

UOC
Universitat Oberta de Catalunya

Índice

Introducción

Tras crear un modelo que representa los procesos de negocio relevantes para la organización y las diferentes perspectivas de análisis y haberlo implementado en el *data warehouse*, el siguiente paso es la carga de los datos. Para ello, se usa la integración de datos.

En un contexto empresarial, la integración puede darse en cuatro grandes áreas:

1) Integración de datos: proporciona una visión única de todos los datos de negocio se encuentren donde se encuentren. Este es el ámbito del presente documento y, en particular, en el contexto de la inteligencia de negocio.

2) Integración de aplicaciones: proporciona una visión unificada de todas las aplicaciones tanto internas como externas a la empresa. Esta integración se consigue mediante la coordinación de los flujos de eventos (transacciones, mensaje o datos) entre aplicaciones.

3) Integración de procesos de negocio: proporciona una visión unificada de todos los procesos de negocio. Su principal ventaja es que las consideraciones de diseño del análisis e implementación de los procesos de negocio son aislados del desarrollo de las aplicaciones.

4) Integración de la interacción de los usuarios: proporciona una interfaz segura y personalizada al usuario del negocio (datos, aplicaciones y procesos de negocio).

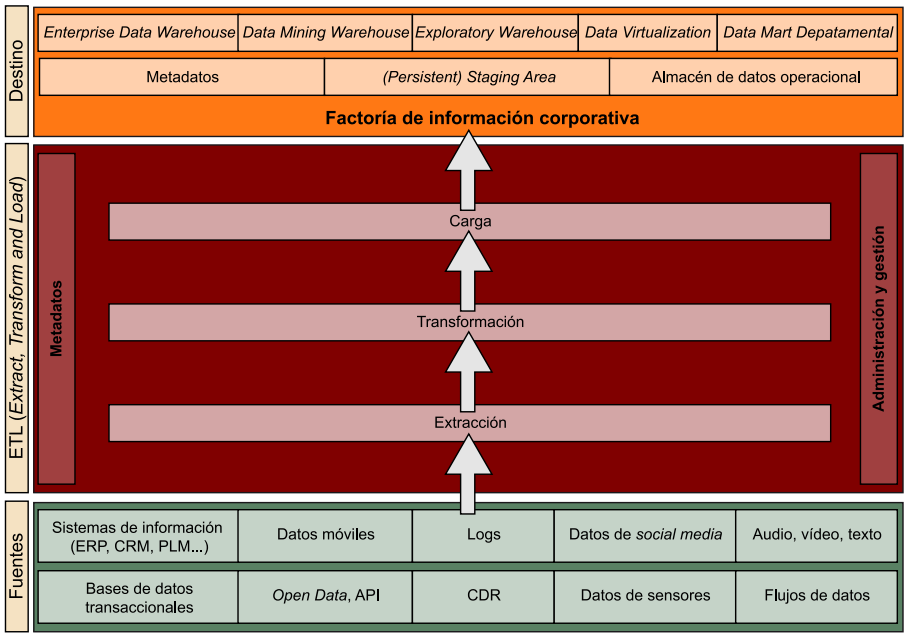
Este módulo se centrará en la integración de datos en general y en los procesos ETL (extracción, transformación y carga) en particular, que es una de las tecnologías de integración de datos que se usa en los proyectos de implantación de *business intelligence*. El objetivo es conocer las diferentes opciones de integración de datos en el ámbito de la inteligencia de negocio y, en particular, conocer el diseño de procesos ETL.

1. Integración de datos: ETL

En el contexto de la inteligencia de negocio, las herramientas ETL han sido la opción habitual para alimentar el *data warehouse*. La funcionalidad básica de estas herramientas está compuesta por:

- Gestión y administración de servicios.
- Extracción de datos.
- Transformación de datos.
- Carga de datos.
- Gestión de datos.

Figura 1. Funcionalidad ETL



Fuente: Josep Curto.

En los últimos años, estas herramientas han evolucionado e incluyen más funcionalidades propias de una herramienta de integración de datos. Podemos destacar:

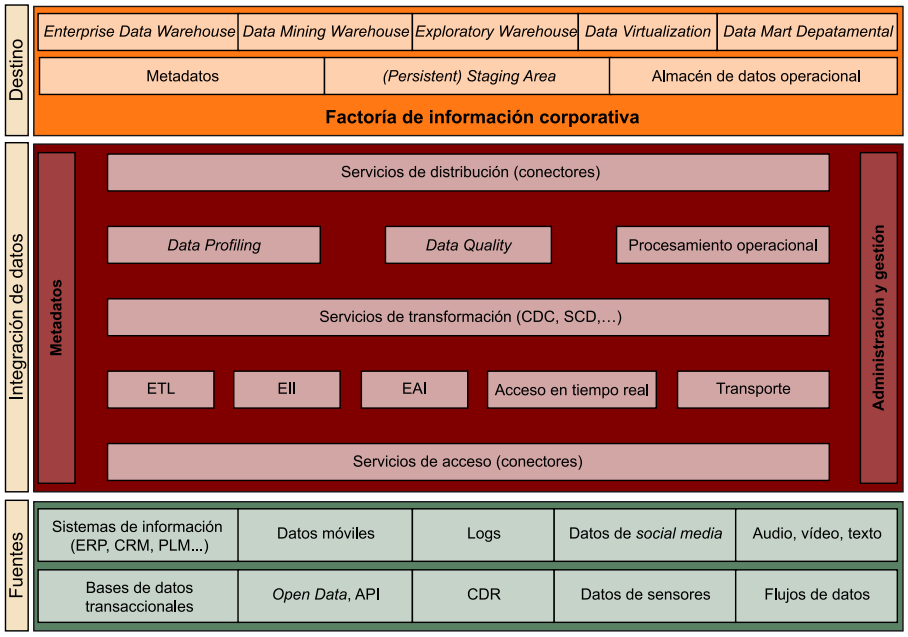
- Servicios de acceso/entrega de datos (vía adaptadores/conectores).

- Gestión de servicios.

- *Data profiling* o perfilado de datos, que permite analizar los datos para conocer su estructura, contenido, relaciones y reglas que puedan derivarse de ellos.

- *Data quality* o calidad de datos, que permite conocer, analizar y gestionar el nivel de calidad de un conjunto de datos.
- Procesos operacionales.
- Servicios de transformación: CDC, SCD, validación, agregación.
- Servicios de acceso a tiempo real.
- *Extract, transform and load* (ETL).
- *Enterprise information integration* (EII).
- *Enterprise application integration* (EAI).
- Capa de transporte de datos.
- Gestión de metadatos.

Figura 2. *Data integration*



Fuente: Josep Curto.

Esta evolución es consecuencia de diversos motivos. Por un lado, la evolución de las necesidades de negocio y, por otro, los diferentes tipos de datos vinculados a la organización:

a) Estructurados: contenidos en bases de datos relacionales.

b) Semiestructurados: en formatos legibles para máquinas, si bien no están completamente estructurados: HTML tabulado, Excel, CSV, etc., que pueden obtenerse mediante técnicas estándar de extracción de datos.

c) No estructurados: en formatos legibles para humanos, pero no para máquinas: Word, HTML no tabulado, PDF, etc., que pueden obtenerse mediante técnicas avanzadas como *text mining* u otras.

Las últimas versiones de herramientas de integración de datos incluyen mayores prestaciones de curación y calidad de datos, que se basan en técnicas de aprendizaje supervisado y no supervisado. Por ejemplo, en entornos internacionales donde el salario de un empleado puede denominarse de forma diferente (*wages* o *salary*) e incluso incluir prestaciones diferentes, la herramienta de integración de datos puede reconocer que son el mismo concepto de negocio.

El punto de partida adecuado es definir formalmente el concepto de integración de datos.

> Se entiende por **integración de datos** el conjunto de aplicaciones, productos, técnicas y tecnologías que permiten una visión única consistente de nuestros datos de negocio.

Respecto a la definición:

- Las aplicaciones son soluciones a medida que permiten la integración de datos en función del uso de productos de integración.

- Los productos comerciales desarrollados por terceros capacitan la integración mediante el uso de tecnologías de integración.

- Las tecnologías de integración son soluciones para realizar la integración de datos.

La integración de datos desempeña un papel cada vez más fundamental en las organizaciones, puesto que en la actualidad el dato puede residir en plataformas internas (sistemas operacionales y decisionales, *cloud* privado) y externas (internet de las cosas, *cloud* público, dispositivos móviles), y es necesario poder capturar y distribuir el dato a los sistemas de análisis necesarios.

1.1. Técnicas de integración de datos

En esta sección vamos a explicar en detalle las diferentes técnicas de integración de datos que existen, y que son las siguientes:

1) Propagación de datos: consiste en copiar datos de un lugar de origen a un entorno destino local o remoto. Los datos pueden extraerse del origen mediante programas que generen un fichero que debe ser transportado al destino, donde se utilizará como fichero de entrada para cargar en la base de datos de destino. Una aproximación más eficiente es descargar solo los datos que han cambiado en origen respecto a la última propagación realizada, generando un fichero de carga incremental que también será transportado al destino. Este tipo de procesos son habitualmente de tipo en línea y trabajan con una arquitectura de tipo *push*. Puede realizarse como:

Técnica *push*

Consiste en la actualización continua en línea del entorno destino mediante aplicaciones de integración de datos que capturan los cambios en origen y los transmiten a destino, donde son almacenados; los datos son automáticamente enviados al entorno remoto.

- Distribución.
- Intercambio bidireccional. Puede ser *master-slave* o *peer-to-peer*.

2) Consolidación de datos: consiste en capturar los cambios realizados en múltiples entornos origen y propagarlos a un único entorno destino, donde se almacena una copia de todos estos datos. Ejemplos de ello son un *data warehouse* o un ODS, alimentado por varios entornos de producción. Con esta técnica es difícil trabajar con tiempos de latencia bajos:

Latencia

En redes informáticas de datos, se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y trasmisión de paquetes dentro de la red.

a) Cuando no se requiere latencia baja, se suelen proveer los datos mediante procesos *batch* en intervalos prefijados (superior a varias horas). Se usan consultas SQL para conseguir los datos (lo que se denomina técnica *pull*).

Técnica *pull*

Es la otra cara de la moneda. La petición del proceso de consolidación de datos es la que genera la extracción de los datos del origen.

b) Cuando se requiere latencia baja, se utiliza la técnica *push*. En este caso, la aplicación de integración de datos debe identificar los cambios producidos en origen para transmitir solo esos cambios, y no todo el conjunto de datos del origen. Para ello, se suele emplear algún tipo de técnica de tipo CDC (*change data capture*).

3) Federación de datos: proporciona a las aplicaciones una visión lógica virtual común de una o más bases de datos. Esta técnica permite acceder a diferentes entornos de origen de datos, que pueden estar en los mismos o en diferentes gestores de datos y máquinas, y crear una visión de este conjunto de bases de datos como si fuese en la práctica una base de datos única e integrada. Funciona de la siguiente manera:

- Cuando una aplicación de negocio lanza una consulta SQL contra esta vista virtual, el motor de federación de datos descompone la consulta en consultas individuales para cada uno de los orígenes de datos físicos involucrados y la lanza contra cada uno de ellos.

- Cuando ha recibido todos los datos de respuesta a las consultas, integra los resultados parciales en un resultado único, realizando las agregaciones y/u ordenaciones necesarias para resolver la consulta original y devuelve los datos a la aplicación que lanzó la consulta original.

- Uno de los elementos clave del motor de federación es el catálogo de datos común. Este catálogo contiene información de la estructura de los datos, su localización y, en ocasiones, información sobre demografía de los datos (volumen de datos, cardinalidad de las claves, claves de distribución, etc.). Ello permite que se pueda optimizar la división de la consulta original al enviarla a los gestores de bases de datos de origen de datos y se elija el camino de acceso global a los datos más eficiente.

4) CDC (*change data capture*): se utilizan para capturar los cambios producidos por las aplicaciones operacionales en las bases de datos de origen, de tal manera que pueden ser almacenados y/o propagados a los entornos destino para que estos mantengan la consistencia con los entornos origen. A continuación trataremos las cuatro principales técnicas de *change data capture*:

a) CDC por aplicación: consiste en que la propia aplicación es la que genera la actualización de datos en origen, y se encarga de actualizar directamente los entornos destino, o almacenar localmente los cambios en una tabla de paso (*staging*) mediante una operación de INSERT dentro de la misma unidad lógica de trabajo.

b) CDC por *timestamp*: se puede emplear cuando los datos de origen incorporan un *timestamp* (por ejemplo, a nivel de fila si el origen es una tabla relacional) de la última actualización de esta. El CDC se limitará a escanear los datos de origen para extraer los datos que posean un *timestamp* posterior al de la última vez que se ejecutó el proceso de CDC: estos datos son los que han cambiado desde la última captura de datos y, por tanto, son los que deben actualizarse en los entornos destino.

c) CDC por *triggers*: Los *triggers* o disparadores son acciones que se ejecutan cuando se actualizan (por UPDATE, DELETE o INSERT) los datos de una determinada tabla sobre la que están definidos. Estos *triggers* pueden utilizar estos datos de la actualización en sentencias SQL para generar cambios SQL en otras tablas locales o remotas. Por lo tanto, una forma de capturar cambios es crear *triggers* sobre las tablas de origen, cuyas acciones modifiquen los datos de las tablas destino.

d) CDC por captura de *log*: consiste en examinar constantemente el fichero de *log* de la base de datos de origen en busca de cambios en las tablas que se deben monitorizar. Estos programas basan su eficiencia en la lectura de los buffers de memoria de escritura en el *log*, por lo que la captura de la información no afecta al rendimiento del gestor relacional al no requerir acceso al disco que contiene el fichero de *log*.

5) Técnicas híbridas: la técnica elegida en la práctica para la integración de datos dependerá de los requisitos de negocio para la integración, pero también en gran medida de los requisitos tecnológicos y de las probables restricciones presupuestarias. En la práctica se suelen emplear varias técnicas de integración, lo que se denomina una técnica híbrida.

1.2. Tecnologías de integración de datos

En esta sección vamos a explicar en detalle las diferentes tecnologías de integración de datos que están basadas en las técnicas que han sido presentadas. Estas son:

1) ETL: permite extraer datos del entorno origen, transformarlos según nuestras necesidades de negocio para integración de datos y cargar estos datos en los entornos destino. Los entornos origen y destino son usualmente bases de datos y/o ficheros, pero en ocasiones también pueden ser colas de mensajes de un determinado *middleware*, así como ficheros u otras fuentes estructuradas, semiestructuradas o no estructuradas. Está basada en técnicas de consolidación. Las herramientas de ETL en la práctica mueven o transportan datos entre entornos origen y destino, pero también documentan cómo estos datos son transformados (si lo son) entre el origen y el destino almacenando esta información en un catálogo propio de metadatos; intercambian estos metadatos con otras aplicaciones que puedan requerirlos y administran todas las ejecuciones y procesos de la ETL: planificación del transporte de datos, *log* de errores, *log* de cambios y estadísticas asociadas a los procesos de movimiento de datos. Este tipo de herramientas suelen tener una interfaz gráfica de usuario y permiten diseñar, administrar y controlar cada uno de los procesos del entorno ETL.

a) ETL de generación de código: constan de un entorno gráfico donde se diseñan y especifican los datos de origen, sus transformaciones y los entornos destino. El resultado generado es un programa de tercera generación (típicamente COBOL) que permite realizar las transformaciones de datos. Aunque estos programas simplifican el proceso ETL, incorporan pocas mejoras en cuanto al establecimiento y automatización de todos los flujos de procesos necesarios para realizar la ETL. Usualmente son los administradores de datos los encargados de distribuir y administrar el código compilado, planificar y ejecutar los procesos en lotes, y realizar el transporte de los datos.

b) ETL basados en motor: permite crear flujos de trabajo en tiempo de ejecución definidos mediante herramientas gráficas. El entorno gráfico permite hacer un *mapping* de los entornos de datos de origen y destino, las transformaciones de datos necesarios, el flujo de procesos y los procesos por lotes necesarios. Toda esta información referente a diseño y procesos del ETL es almacenada en el repositorio del catálogo de metadatos. Se compone de varios motores:

- **Motor de extracción**: utiliza adaptadores como ODBC, JDBC, JNDI, SQL nativo, adaptadores de ficheros planos u otros. Los datos pueden ser extraídos en modo *pull* planificado, típicamente soportando técnicas de consolidación en proceso por lotes, o mediante modo *push*, típicamente utilizando técnicas de propagación en procesos de tipo en línea. En ambos casos se pueden utilizar técnicas de *change data capture* (CDC) ya vistas.

- **Motor de transformación**: proporciona una librería de objetos que permite a los desarrolladores transformar los datos de origen para adaptarse a las estructuras de datos de destino, lo que permite, por ejemplo, la agregación de los datos en destino en tablas resumen.

- **Motor de carga**: utiliza adaptadores a los datos de destino, como el SQL nativo o cargadores masivos de datos para insertar o actualizar los datos en las bases de datos o ficheros de destino.

- **Servicios de administración y operación**: permiten la planificación, ejecución y monitorización de los procesos ETL, así como la visualización de eventos y la recepción y resolución de errores en los procesos.

c) ETL integrado en la base de datos: algunos fabricantes incluyen capacidades ETL dentro del motor de la base de datos (al igual que lo hacen con otro tipo de características, como soporte OLAP y minería de datos). En general, presentan menos funcionalidades y complejidad, y son una solución menos completa que los ETL comerciales basados en motor o de generación de código. Por ello, a los ETL integrados en base de datos se los clasifica en tres clases en relación con los ETL comerciales (basados en motor o de generación de código):

- **ETL cooperativos**: con ellos, los productos comerciales pueden usar funciones avanzadas del gestor de base de datos para mejorar los procesos de ETL. Ejemplos de ETL cooperativos son aquellos que pueden utilizar procedimientos almacenados y SQL complejo para realizar las transformaciones de los datos en origen de una manera más eficiente, o utilizar paralelismo de CPU en consultas para minimizar el tiempo de los procesos ETL.

- **ETL complementarios**: cuando los ETL de bases de datos ofrecen funcionalidades complementarias a los ETL comerciales. Por ejemplo, hay gestores de bases de datos que ofrecen soporte a MQT (*materializad query tables*) o vistas de agregación precalculadas, mantenidas y almacenadas por el gestor que pueden usarse para evitar transformaciones de datos realizadas por el ETL comercial. Además, otros gestores permiten la interacción directa mediante SQL con *middleware* de gestión de mensajes (por ejemplo, leyendo una cola de mensajes mediante una UDF o permitiendo la inserción de nuevos mensajes en colas mediante SQL) o con aplicaciones que se comunican mediante *web services*.

- **ETL competitivos**: algunos gestores ofrecen herramientas gráficas integradas que explotan sus capacidades ETL en lo que claramente es competencia con los ETL comerciales.

2) EII: el objetivo de la tecnología EII es permitir a las aplicaciones el acceso a datos dispersos (desde un *data mart* hasta fichero de texto o incluso *web services*) como si estuviesen todos residiendo en una base de datos común. Por lo tanto, se basa en la federación. El acceso a datos dispersos implica la descomposición de la consulta inicial (habitualmente en SQL) direccionada contra la vista virtual federada en subcomponentes, que serán procesados en cada uno de los entornos donde residen los datos. Se recogen los resultados individuales de cada uno de los subcomponentes de la consulta, se combinan adecuadamente y se devuelve el resultado a la aplicación que lanzó la consulta. Los productos de EII han evolucionado desde dos entornos origen diferenciados: las bases de datos relacionales y las bases de datos XML. Actualmente, la tendencia en productos EII es que soporten ambas interfaces a datos, SQL (ODBC y JDBC) y XML (XQuery y XPath). Los productos comerciales que implementan EII varían considerablemente en las funcionalidades que aportan, siendo el área más diferenciadora la optimización de las consultas distribuidas. Las características básicas de los productos que implementan soluciones de integración de datos EII son las siguientes:

- **Transparencia**: los datos parecen estar en un origen único.
- **Heterogeneidad**: integración de datos de diferentes fuentes (relacionales, XML, jerárquicos) y también no estructurados.
- **Extensibilidad**: posibilidad de federar cualquier fuente de datos.
- **Alta funcionalidad**: acceso en lectura y escritura a cualquier fuente soportada.
- **Autonomía**: acceso no disruptivo para los datos o las aplicaciones.
- **Rendimiento**: posibilidad de optimizar las consultas dependiendo del tipo y fuente de datos.

3) EDR: tiene el objetivo de detectar los cambios que suceden en las fuentes de origen. Está soportada por las técnicas de integración de datos de CDC (*change data capture*) y por la técnica de propagación de datos. Consta básicamente de los siguientes elementos:

a) Programa de captura: se encarga de recuperar los cambios producidos en la base de datos de origen. Esta captura puede ser realizada a través de una salida que lea constantemente el *log* de recuperación de la base de datos, a través de *triggers* o mediante una aplicación externa de usuario. El programa

de captura se apoya en una serie de tablas donde se almacena información de control del proceso de captura, como por ejemplo las tablas que son orígenes de replicación.

b) Sistema de transporte: los sistemas de transporte más comunes son a través de tablas de paso (*staging*) que dan lugar a la denominada replicación de tipo SQL, o a través de un *middleware* de gestión de colas, la denominada *queue-replication* o *q-replication*.

c) Programa de aplicación de cambios: es la pieza que o bien lee mediante SQL de las tablas de *staging* los cambios pendientes de aplicar en la *SQL-replication*, o lee del sistema de colas en la *q-replication*, y mediante la información de control almacenada en tablas realiza el mapeo entre datos de origen y destino, realiza las transformaciones necesarias a los datos y actualiza los datos de destino mediante SQL si se trata de destinos relacionales, o publica un registro XML para que pueda ser tratado por aplicaciones de propósito general.

d) Programa de administración: permite las definiciones necesarias de origen de datos y destinos, mapeos, transformaciones y establecer los intervalos de aplicación de cambios. Usualmente es una herramienta de tipo gráfico.

e) Utilidades: programas de utilidad que sirven para, por ejemplo, planificar una carga de datos inicial del destino a partir de los datos de origen.

1.3. Uso de la integración de datos

Los procesos de integración de datos se usan en múltiples tipologías de proyectos. Podemos destacar los siguientes:

1) Migración de datos: consiste en el movimiento de datos entre una fuente de origen y una de destino. Por ejemplo, para migrar de una versión de una base de datos a la siguiente.

2) Procesos de calidad de datos: consiste en auditar la calidad de una fuente de datos, detectar los problemas que existen y aplicar mejoras mediante técnicas de procesamiento. Por ejemplo, antes de una campaña de contacto a antiguos clientes es necesario revisar que los datos relevantes –como correo electrónico o teléfono– tienen el nivel de calidad adecuado para la iniciativa.

3) *Corporate performance management* (CPM): consiste en monitorizar y gestionar el rendimiento de una organización. Aquí los procesos de integración permiten extraer la información relevante para el control del rendimiento.

4) ***Master data management*** **(MDM)**: incluye los procesos, el gobierno, las reglas, los estándares y las herramientas que definen y gestionan los datos críticos de una organización para proporcionar un único punto de referencia. Por ejemplo, la integración de datos permite la consolidación de datos y su propagación.

5) ***Customer data integration*** **(CDI)**: consiste en MDM aplicado a los datos de cliente.

6) ***Product information management*** **(PIM)**: consiste en MDM aplicado a la información de producto.

7) ***Enterprise information management*** **(EIM)**: consiste en MDM aplicado a la gestión de la información empresarial.

8) ***Data warehousing***: en este proyecto, los procesos de integración de datos permiten extraer los datos, procesarlos en información y cargarlos en el *data warehouse*, *data marts*, etc.

9) ***Business intelligence*** **(BI)**: en este proyecto, los procesos de integración de datos se habrán usado en diferentes iniciativas, como carga de los datos en el *data warehouse*, la generación de informes operacionales, etc.

2. ETL en el contexto de Pentaho

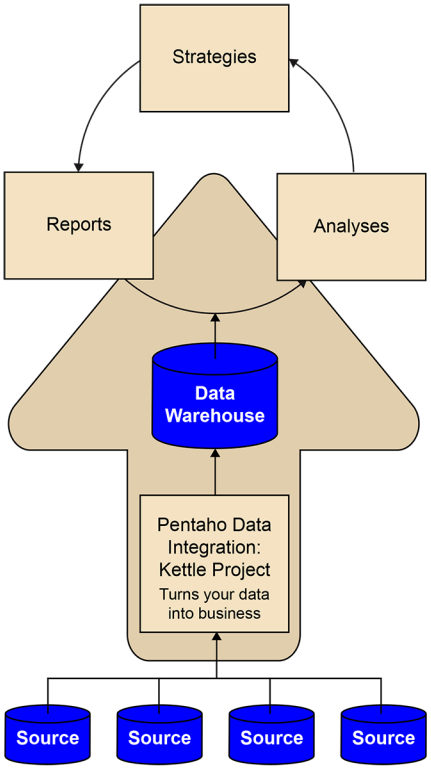
Pentaho Data Integration (PDI), anteriormente llamado Kettle, se inició en 2001 por Matt Casters. En el año 2006, Pentaho adquirió Kettle y lo renombró después de que este pasara a ser *open source*. De esta manera continuaba con la política de crear una *suite* completa de inteligencia de negocio *open source*. Matt Casters pasó a formar parte del equipo de Pentaho.

Logo de Kettle/PDI. Fuente: Pentaho.

PDI es una solución de integración de datos programada en java, orientada completamente al usuario y basada en un enfoque de metadatos. Los procesos ETL se encapsulan en metadatos que se ejecutan a través del motor ETL.

Esta herramienta permite cargar datos de múltiples fuentes de origen en un *data warehouse* para que posteriormente la información consolidada sea de utilidad a nivel operativo, táctico y estratégico.

Figura 3. PDI en el contexto tradicional del BI

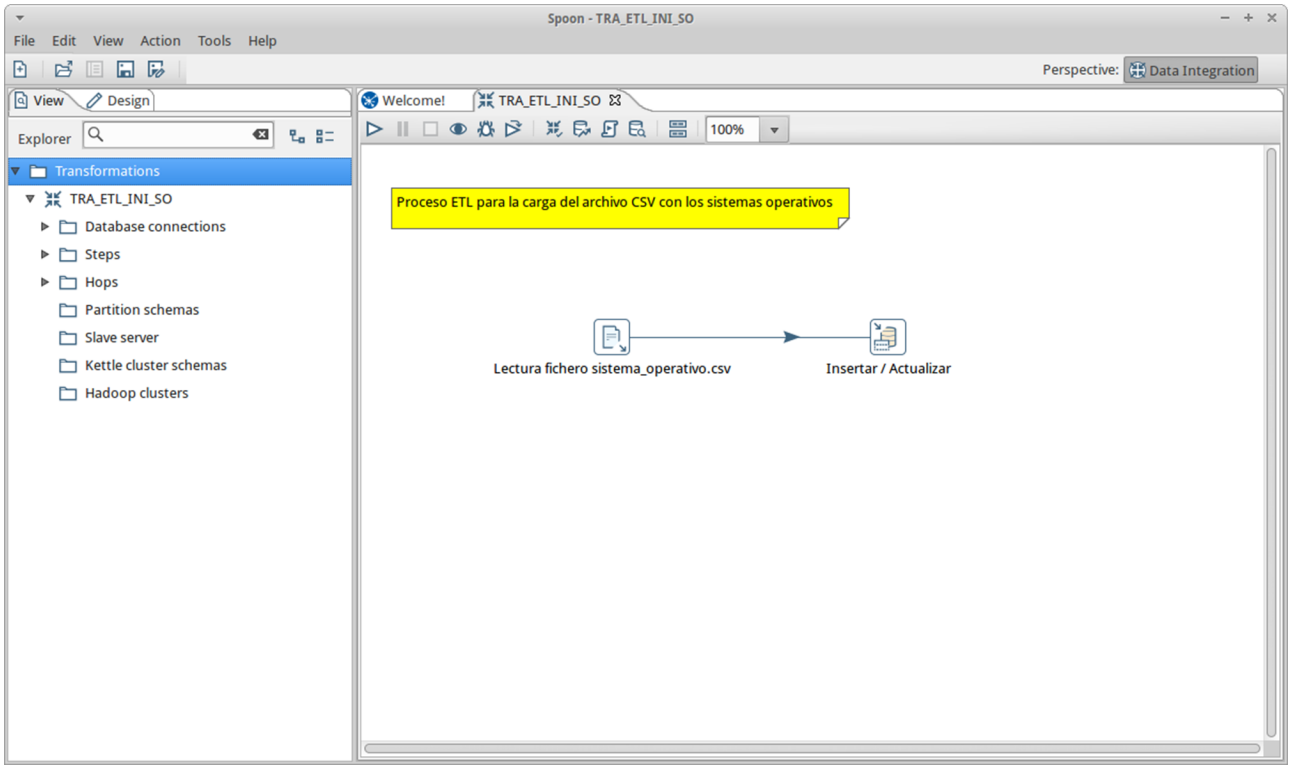


Fuente: Pentaho.

Las principales características de PDI son estas:

a) Entorno gráfico orientado al desarrollo rápido y ágil basado en dos áreas: la de trabajo y la de diseño/vista.

Figura 4. Entorno gráfico de PDI



Fuente: Josep Curto.

b) Multiplataforma (Microsoft Windows, Linux y Mac OS).

c) Incluye múltiple conectores a bases de datos, tanto propietarias como comerciales, relacionales y no relacionales, así como conectores a ficheros planos, Excel, XML u otros.

d) Arquitectura extensible mediante *plugins* y perspectivas:

- *Plugins*: permiten extender la funcionalidad de los procesos ETL. Por ejemplo, crear un conector específico a un servicio web. Estos *plugins* están disponibles a través del *marketplace* de PDI.

- Perspectivas: extiende la funcionalidad de PDI más allá del contexto de los procesos ETL. Por ejemplo, para la creación de esquemas OLAP.

e) Soporta uso de clúster, procesos ETL en paralelo y arquitecturas servidor maestro-esclavo.

f) Completamente integrado con la *suite* de Pentaho. Por ejemplo, un proceso ETL programado puede ser el origen de un informe.

g) Basado en el desarrollo de dos tipos de objetos:

- Transformaciones: permite definir las operaciones de transformación de datos.

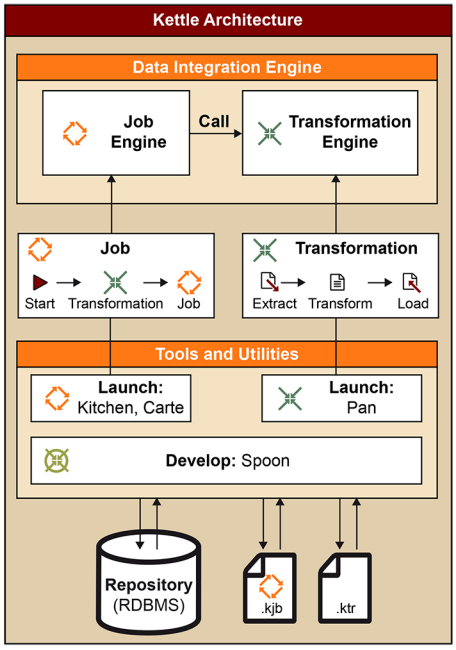
- Trabajos: permiten gestionar y administrar procesos ETL a alto nivel.

PDI está formado por cuatro componentes:

- **Spoon**: entorno gráfico para el desarrollo de transformaciones y trabajos.
- **Pan**: permite ejecutar transformaciones.
- **Kitchen**: permite ejecutar trabajos.
- **Carte**: es un servidor remoto que permite la ejecución de transformaciones y trabajos.

El siguiente diagrama presenta cómo trabajan de manera conjunta estos cuatro componentes y la relación entre transformaciones y trabajos.

Figura 5. Arquitectura de PDI



Fuente: Pentaho.

Cada proceso ETL (ya sea una transformación o un trabajo) se compone de uno o varios pasos. Los pasos disponibles para cada uno de ellos son diferentes y están clasificados por categorías. El entorno de desarrollo incluye un buscador de pasos que facilita encontrar el adecuado en cada momento. A continuación describimos las principales categorías y hacemos referencia a los pasos que contienen:

1) Pasos disponibles para trabajos:

- ***General* (generales)**: permite iniciar un trabajo, ejecutar transformaciones o trabajos entre otras operaciones.

- ***Mail* (correo)**: permite enviar correos, recuperar cuentas o validarlas.

- ***File management*** **(gestión de ficheros)**: permite realizar operaciones con ficheros, como crear, borrar, comparar o comprimir.

- ***Conditions*** **(condiciones)**: permiten realizar comprobaciones necesarias para procesos ETL, como la existencia de un fichero, una carpeta o una tabla.

- ***Scripting***: permiten crear scripts de JavaScript, SQL y *shell*.

- ***Big data***: permiten lanzar procesos de MapReduce, cargar datos en HDFS y enviar procesos ETL a plataformas para su ejecución, como Amazon EMR o Spark.

- ***Bulk loading*** **(carga en bruto)**: permiten realizar cargas *bulk* a MySQL, MSSQL, Acces y ficheros.

- **XML**: permiten validar XML y XSD.

- ***Utilities*** **(utilidades)**: permiten diferentes funciones para asegurar la ejecución de transformaciones, hacer *ping* a un *host* o borrar los datos de una base de datos.

- ***Repository*** **(repositorio)**: permiten realizar operaciones con el repositorio de transformaciones y trabajos.

- ***File transfer*** **(envío de ficheros)**: permiten enviar o tomar ficheros desde FTP y SFTP.

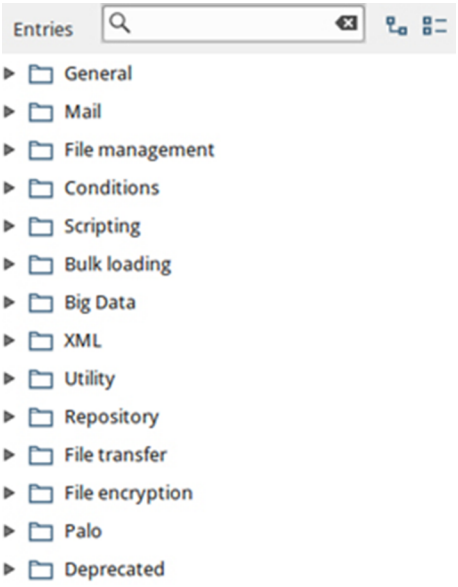
- ***File encription*** **(encriptación de ficheros)**: uso de PGP para el envío y la recepción de ficheros.

- **Palo**: creación/borrado de cubos en PALO.

- ***Deprecated*** **(obsoletos)**: incluye los pasos que desaparecerán en la siguiente versión del producto.

- **Historial**: solo aparece si se han usado previamente pasos y recopila los pasos frecuentemente usados por el desarrollador.

Figura 6. Pasos disponibles para trabajos



Fuente: Josep Curto.

2) Pasos disponibles para transformaciones:

- ***Input* (entrada)**: permite recuperar datos desde bases de datos (JDBC), Acces, CSV, Excel ficheros, LDAP, Mondrian, RSS u otros.

- ***Output* (salida)**: permite cargar datos en bases de datos u otros formatos de salida.

- ***Transform* (transformar)**: permite realizar operaciones con datos, como filtrar, ordenar, partir, añadir nuevos campos, mapear, etc.

- ***Utility* (utilidades)**: permite operar con filas o columnas y otras operaciones, como enviar un *email*, escribir a *log*.

- ***Flow* (flujo)**: permite realizar operaciones con el flujo de datos, como fusionar, detectar flujos vacíos, realizar operaciones diferentes en función de una condición, etc.

- ***Scripting***: permiten crear *scripts* de JavaScript, SQL, expresiones regulares, fórmulas y expresiones java.

- ***BA server* (servidor BA)**: permite conectarse al servidor de Pentaho y hacer operaciones (como recuperar variables).

- ***Lookup* (búsqueda de datos)**: permite añadir información al flujo de datos mediante la búsqueda en bases de datos y otras fuentes.

- ***Joins* (uniones)**: permite unir filas en función de diferentes criterios.

- ***Data warehouse* (almacén de datos)**: permite trabajar con dimensiones SCD.

- ***Validation* (validación)**: permite validar tarjetas de crédito, datos, direcciones de correo o XSD.

- ***Statistics* (estadística)**: permite realizar operaciones estadísticas sobre un flujo de datos.

- ***Big data***: permite cargar y extraer datos de Avro, Cassandra, Hadoop y MongoDB, entre otros.

- ***Agile***: permite el desarrollo de *data marts* ágiles.

- ***Criptography* (criptografía)**: permite el uso de PGP en transformaciones.

- **PALO**: permite hacer operaciones con PALO.

- ***Open ERP***: permite hacer operaciones con Open ERP.

- ***Job* (trabajo)**: permite realizar operaciones propias de un trabajo.

- ***Mapping* (mapeado)**: permite realizar el mapeo entre campos de entrada y salida.

- ***Inline* (embebido)**: permite realizar operaciones con *sockets*.

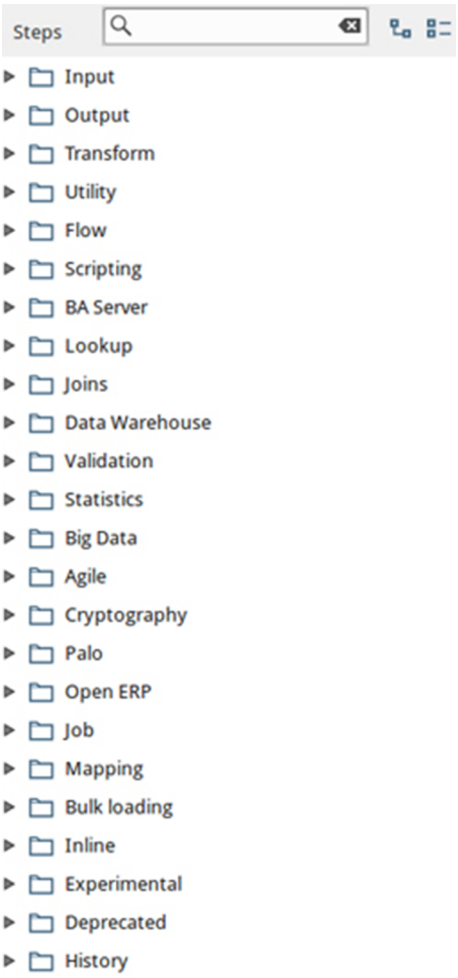
- ***Experimental***: incluye los pasos en fase de validación.

- ***Deprecated* (obsoleto)**: incluye los pasos que desaparecerán en la siguiente versión del producto.

- ***Bulk loading* (carga en bruto)**: permiten realizar cargas *bulk* a Infobright, LucidDB, MonetDB y Oracle.

- **Historial**: recopila los pasos frecuentemente usados por el desarrollador.

Figura 7. Pasos disponibles para transformaciones



Fuente: Josep Curto.

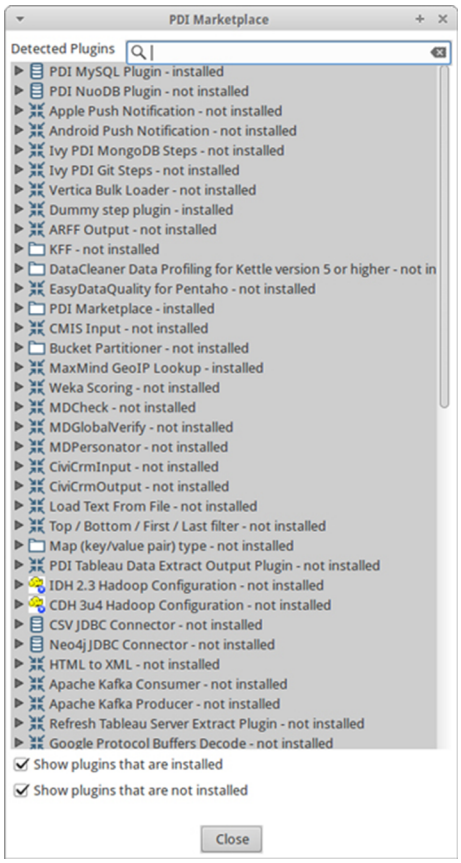
3) Pasos disponibles en el *marketplace*. El listado de pasos disponibles en el *marketplace* va creciendo poco a poco auspiciado por la comunidad. Entre los pasos disponibles destacan:

- Pasos que permiten trabajar con bases de datos relacionales y NoSQL, como MySQL, NuoDB, Vertica, MongoDB, Neo4j, etc.

- Pasos relacionados con tecnologías y plataformas de *big data*, como Cloudera, HortonWorks o MapReduce.

- Pasos que permiten ejecutar procesos de minería de datos de Weka o R.

Figura 8. PDI Marketplace

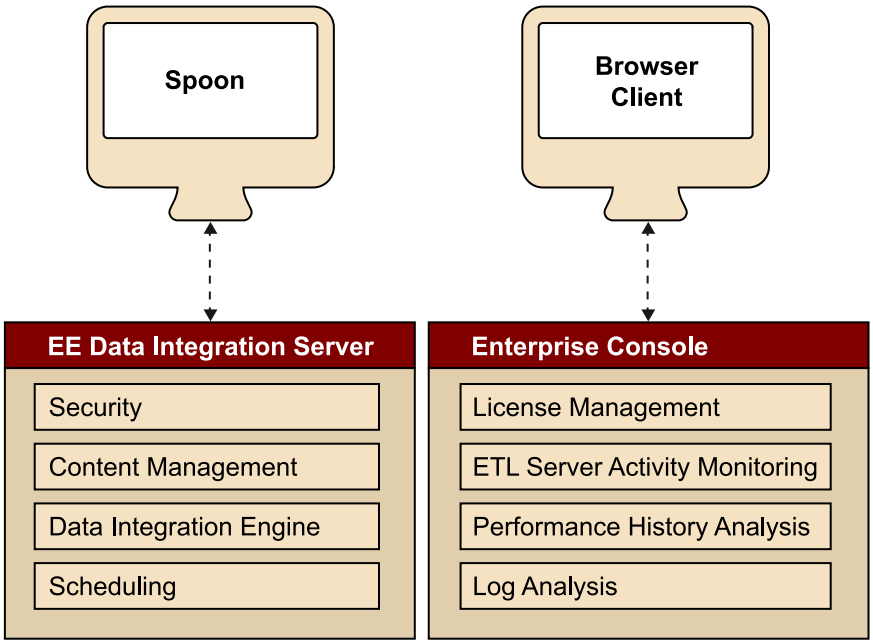


Fuente: Josep Curto.

Diferencias con la versión *enterprise*:

- Inclusión de una consola web para la administración, planificación y monitorización de procesos ETL.
- Gestión de licencias.
- Análisis de rendimiento de ETL.
- Análisis de *log*.
- Soporte profesional.
- Uso de perspectivas (para la creación de procesos ETL, de esquemas OLAP, etc.).

Figura 9. Diferencias con la versión empresarial



Fuente: Pentaho.

3. Caso práctico

3.1. Contexto

Con el objetivo de entender cómo se diseñan procesos ETL, en el caso práctico se partirá de una situación real simplificada.

Consideremos que se administra una única aplicación que está alojada en un servidor apache. El archivo *log* resultante está en *combined log format*, lo que significa que los campos que incluye en cada línea del archivo son:

- **IP**: desde la que se accede a un recurso de la aplicación.
- **RFC 1413**: identificador de la máquina en la red (uso interno). Este valor para aplicaciones web externas suele estar vacío.
- **Usuario remoto**: identificador del usuario. Sucede lo mismo que en caso anterior.
- **Fecha**: en formato [dd/MMM/yyyy:HH:mm:ss-XXXX].
- **Recurso**: aquello a lo que se accede.
- **Resultado**: respuesta por parte del servidor que cubre los diferentes escenarios posibles.
- **Tiempo**: segundos que se tarda en acceder al recurso.
- **Referente**: desde donde se accede al recurso.
- ***User-agent***: información del sistema operativo y del navegador que han sido usados para acceder al recurso.

En nuestro caso, se prescindirá de la información que proporcionan los campos: RFC 1413, usuario remoto y tiempo. Una forma de extender el modelo sería considerar está información y discutir cómo afecta a este.

También se considerará que, conociendo las necesidades informacionales, se ha preparado una colección de ficheros que contienen información que permitirá facilitar la carga de algunas de las dimensiones de nuestro *data warehouse*.

Resumiendo, la situación de partida está formada por cinco ficheros con los que se procederá a una carga inicial del *data warehouse*:

Carga incremental

La carga incremental del *data warehouse* queda fuera del ámbito de este documento, si bien se puede desarrollar a partir de la carga inicial.

- access.log: que contiene la información de acceso a nuestra aplicación web.
- navegador.csv: que contiene un listado de navegadores base.
- protocolo.csv: que contiene los protocolos de acceso estándar.
- resultado.csv: que contiene el resultado que puede proporcionar el servidor a un acceso.
- so.cv: que contiene un listado de sistemas operativos base.
- Allagents.xml: contiene un listado de robots que frecuentemente visitan páginas web para indexarlas; este fichero permitiría categorizar las visitas de los robots.

Estos último ficheros han sido creados partiendo de la situación real en la que normalmente en la carga inicial (la que se realizará en este caso) se precargan las dimensiones.

La estrategia que se seguirá en el proceso ETL será la siguiente:

1) Cargar las dimensiones navegador, protocolo, resultado y so a partir de los ficheros anteriores.

2) Complementar las dimensiones restantes a partir de la información presente en el fichero access.log y alimentar la tabla de hecho de visitas.

3) Crear un trabajo para lanzar todas las transformaciones de una manera única.

Se usará la siguiente notación:

- Para las transformaciones: TRA_ETL_INI_{Nombre de la dimensión o tabla de hecho que cargar}.
- Para los trabajos: JOB_CARGA_INI_{Nombre de la dimensión o tabla de hecho que cargar}.

3.2. Diseño con Pentaho Data Integration

Carpeta pentaho-solutions

Se puede acceder a esta carpeta desde el escritorio a través del enlace directo llamado pentaho y luego entrando en biserver-ce.

Este es el primer contacto con las herramientas de desarrollo de Pentaho. Antes de nada, es necesario estructurar el proyecto o solución de negocio. Las soluciones de negocio se guardaban tradicionalmente dentro de la carpeta pentaho-solutions. En las últimas versiones, los objetos de análisis se guardan a nivel de base de datos, si bien seguiremos la estructura anterior.

De esta manera hemos creado una carpeta llamada AEW (análisis de estadísticas web) y en ella varias carpetas que contienen los materiales de los diferentes capítulos:

- Datos.
- Modelo.
- ETL.
- OLAP.
- Reporting.
- Dashboard.

En el caso de PDI, se puede trabajar de dos maneras: mediante el repositorio de datos (que permite guardar las transformaciones en base de datos) o mediante ficheros. Trabajaremos de esta segunda manera para compartir más fácilmente los cambios.

Dentro de la carpeta ETL existen dos carpetas, llamadas datos y procesos ETL, cuyo nombre describe su funcionalidad.

Dentro de datos tenemos los seis ficheros presentados en el contexto.

Figura 10. Fuentes de origen

Fuente: Josep Curto.

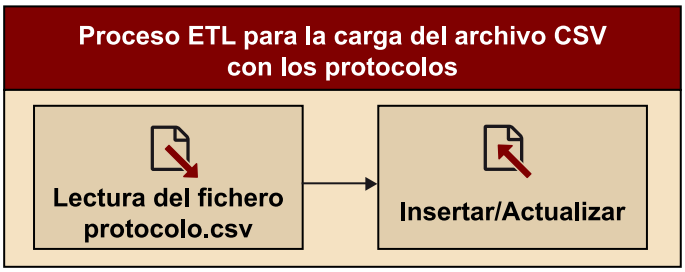
Siguiendo la estrategia que hemos definido anteriormente, el primer paso es la carga de los ficheros CSV. Son ficheros ya preparados en el formato de la base de datos, de manera que cargan directamente. Por tanto, entendiendo cómo se ha creado uno de los procesos ETL, es suficiente para el resto.

La transformación ETL llamada TRA_ETL_INI_PROTOCOLO tiene dos pasos:

- Lectura del fichero CSV.

- Insertar/actualizar de la base de datos a partir de la información extraída del fichero.

Figura 11. Pasos ETL para la carga de protocolo



Fuente: Josep Curto.

El fichero tiene la forma:

Figura 12. Información del fichero protocolos

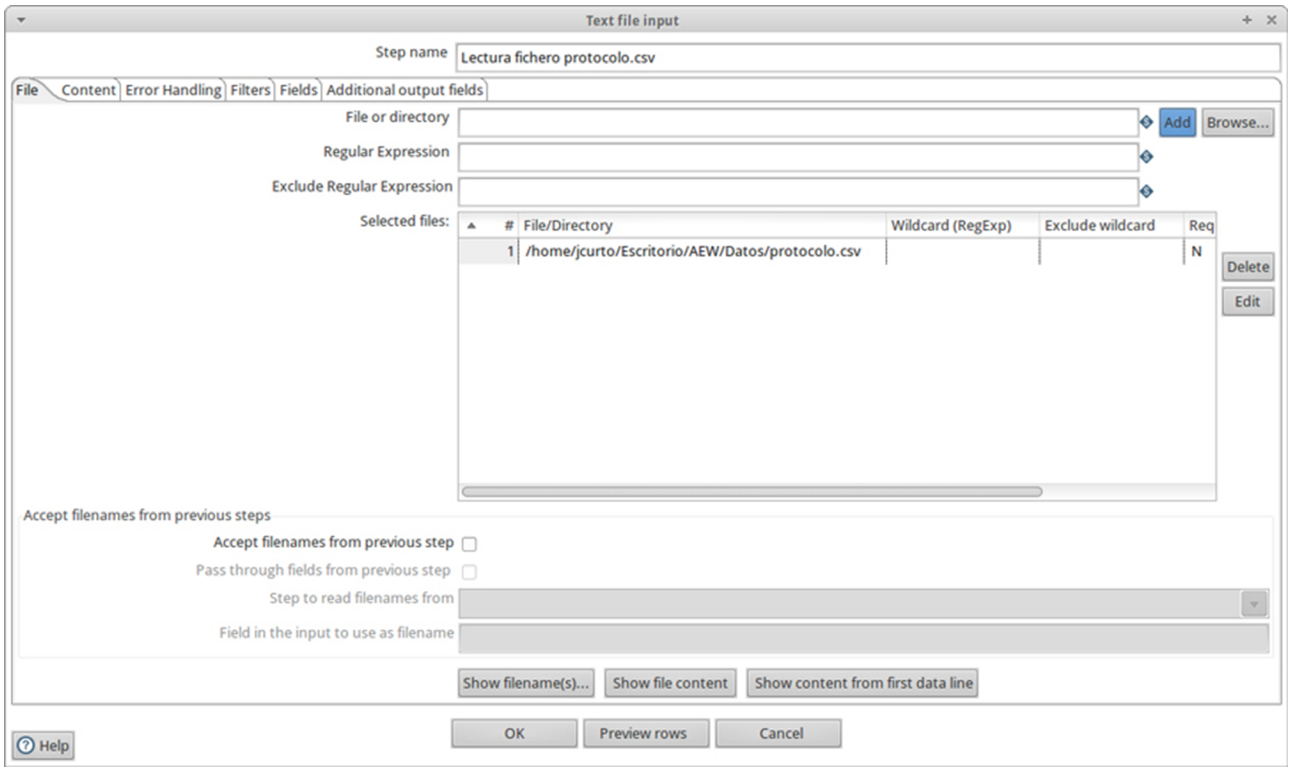
	A	B	C
1	0;Sin información		
2	1;HTTP		
3	2;FTP		
4	3;HTTPS		
5			
6			

Fuente: Josep Curto.

Para parametrizar el paso de lectura, se usa el paso *text file input* y entonces:

a) Se define el fichero que es la fuente de origen en la pestaña *file*.

Figura 13. Lectura del fichero CSV

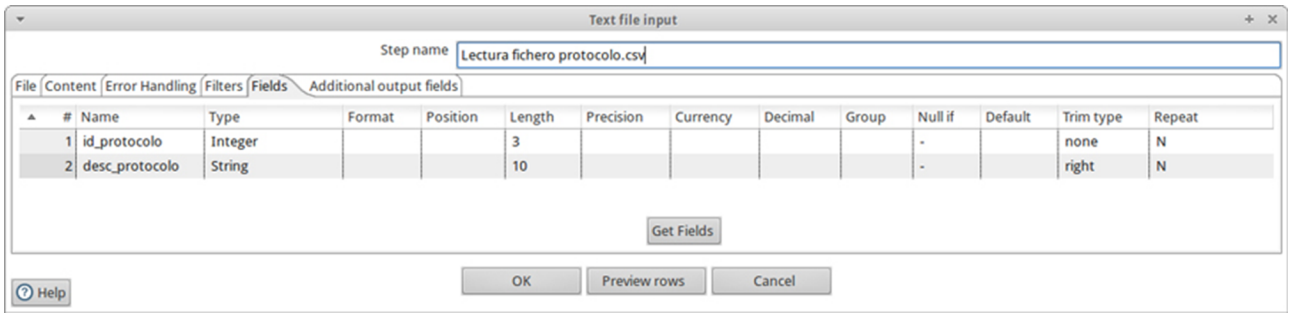


Fuente: Josep Curto.

b) Se define cuál es el separador y cómo está encapsulado el texto en la pestaña *content*.

c) Se definen los campos que cargar en la campaña *fields*. Para facilitar la tareas, se pulsa el botón *get fields*.

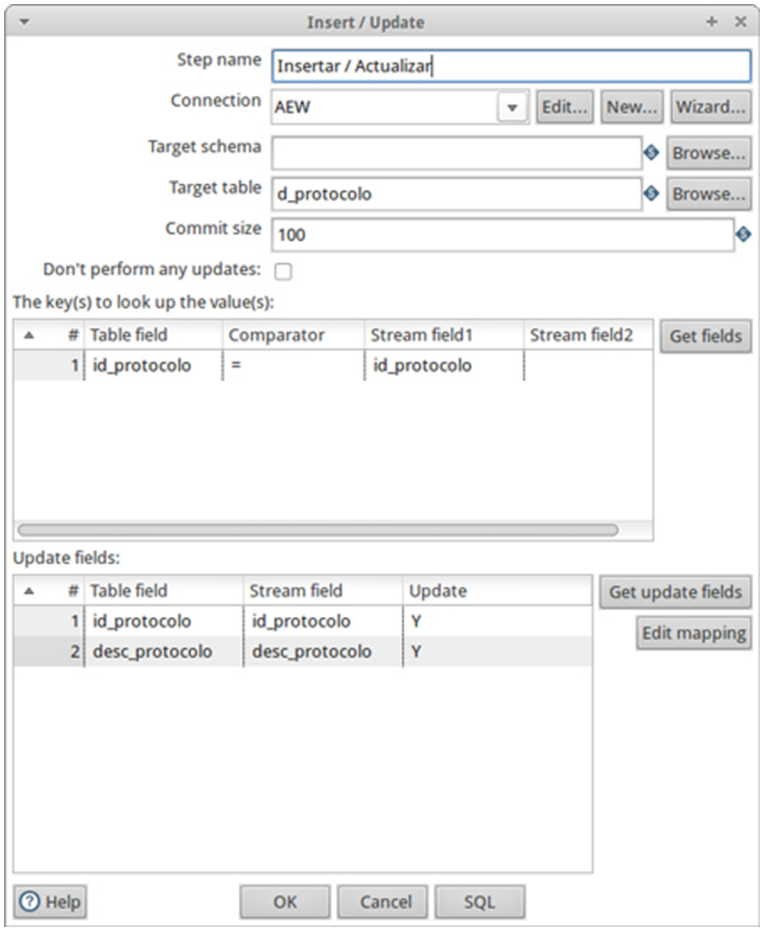
Figura 14. Campos del fichero



Fuente: Josep Curto.

d) Después se inserta la información mediante el paso *insert/update*, cuya parametrización es la siguiente:

Figura 15. Insertar/actualizar base de datos



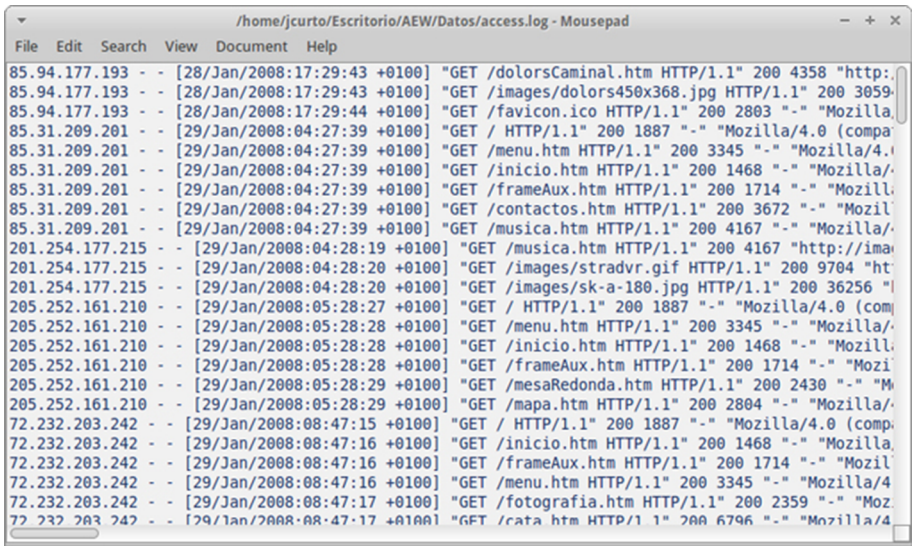
Fuente: Josep Curto.

De manera equivalente se realizan las otras dimensiones que se cargan a partir de los ficheros CSV: navegador, resultado y so.

Una vez cargadas estas cuatro dimensiones, a partir del fichero access.log se cargará la información de las dimensiones restantes primero y posteriormente la tabla de hecho.

La información del fichero access.log está en la forma de un *log* de un servidor de Apache.

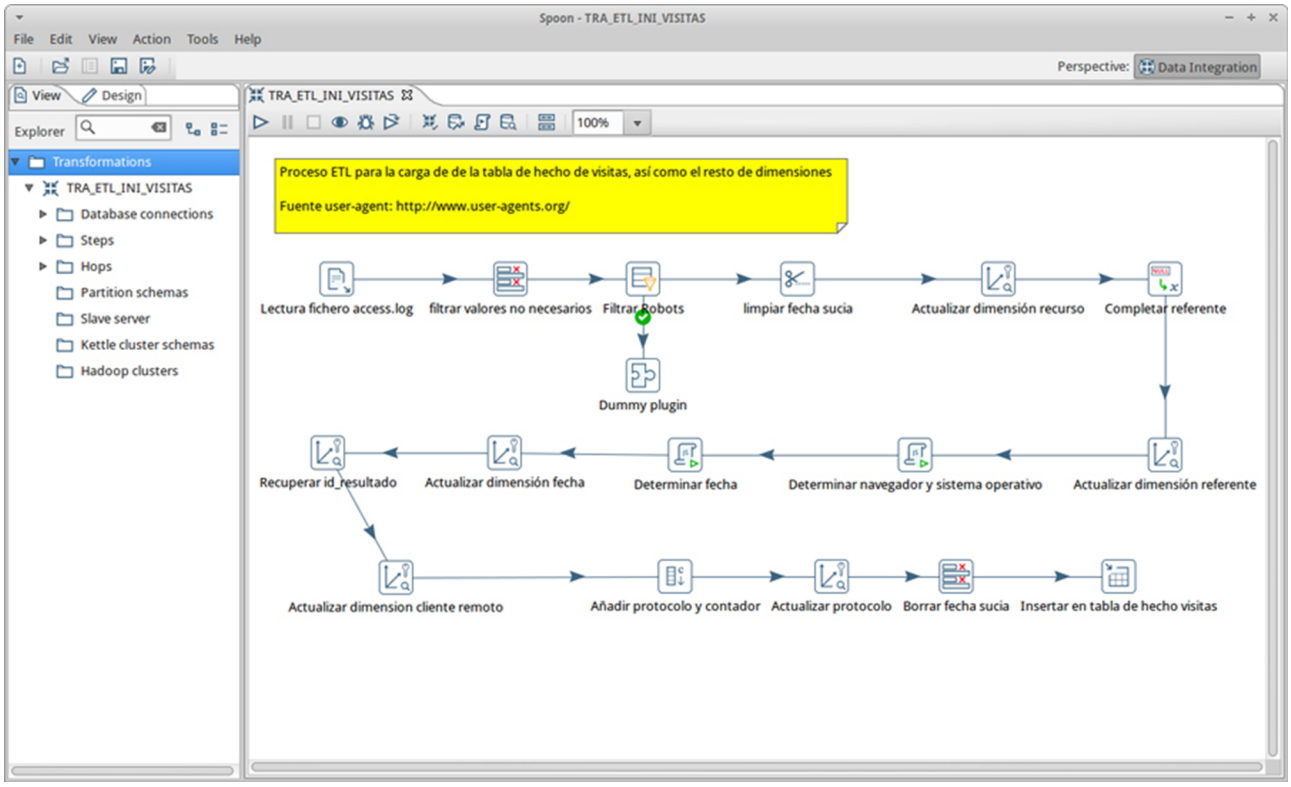
Figura 16. Contenido del fichero *log*



Fuente: Josep Curto.

La transformación creada que carga el resto de las dimensiones y la tabla de hecho y sus atributos es:

Figura 17. Pasos carga datos del fichero *log*

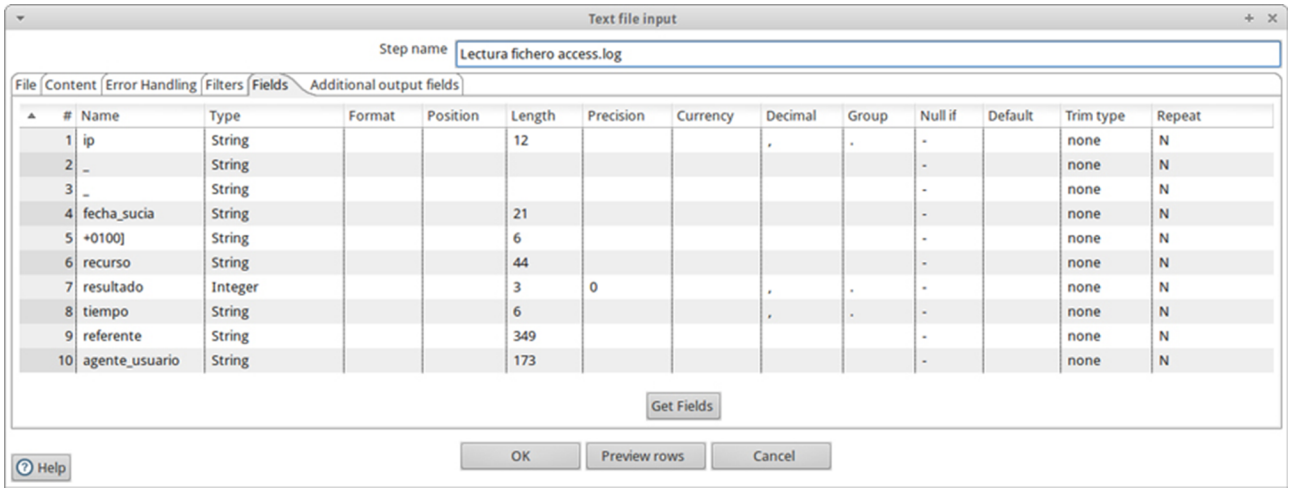


Fuente: Josep Curto.

Analicemos los diferentes pasos que componen esta transformación:

a) Lectura del fichero *log* con el que recuperamos los campos que forman parte del flujo de información usando el paso *text file input*.

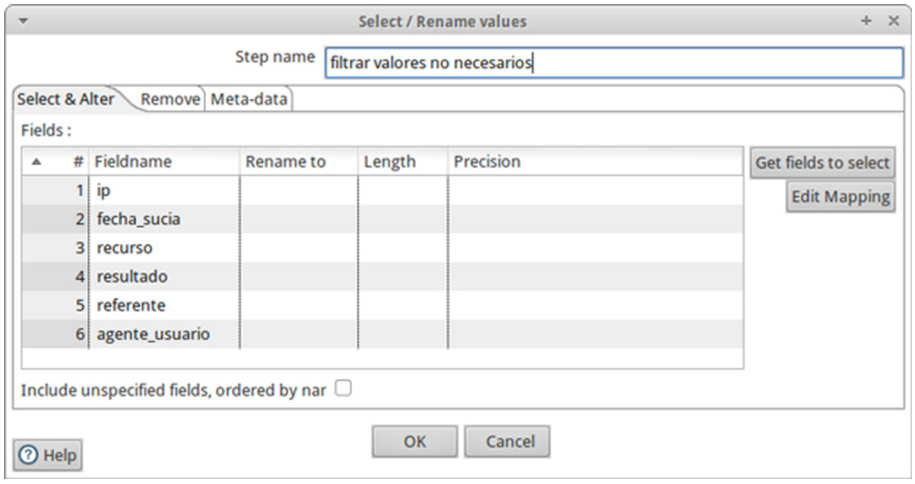
Figura 18. Campos del fichero *log*



Fuente: Josep Curto.

b) Dado que existe información que no será usada, se descarta mediante el paso *select / rename values* (los campos que no se seleccionan desaparecen del flujo).

Figura 19. Selección de campos relevantes



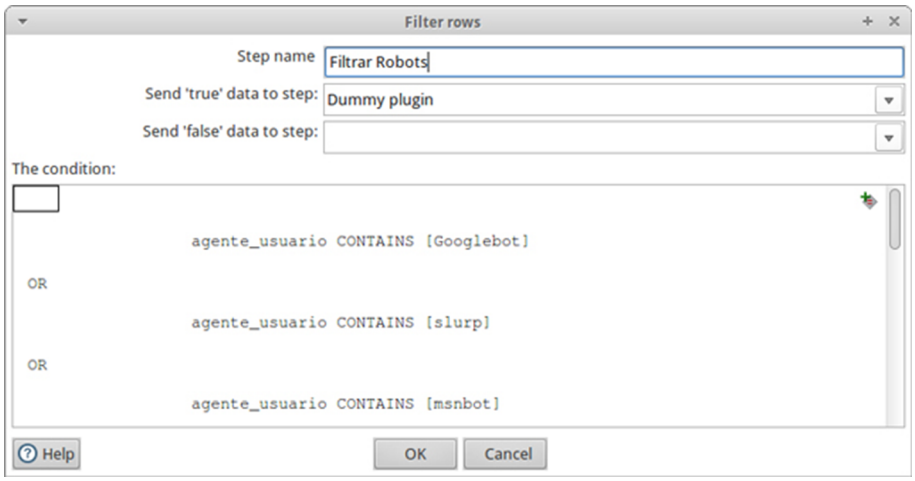
Fuente: Josep Curto.

c) Filtramos información de robots para quedarnos tan solo con visitas.

Nota

Este paso no está optimizado y no se basa en el fichero allagents.xml, lo que se deja al lector.

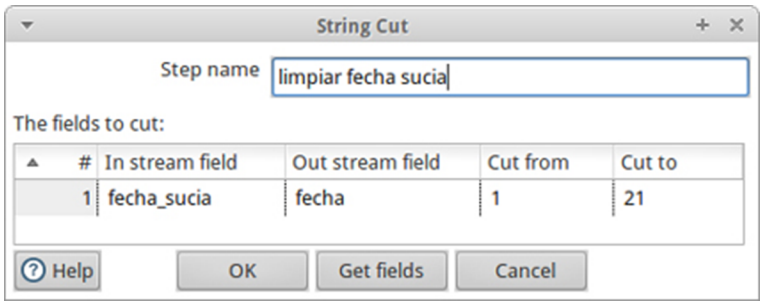
Figura 20. Filtrado de robots



Fuente: Josep Curto.

d) La fecha se ha extraído como una cadena y se borra el carácter sobrante mediante el paso *string cut*.

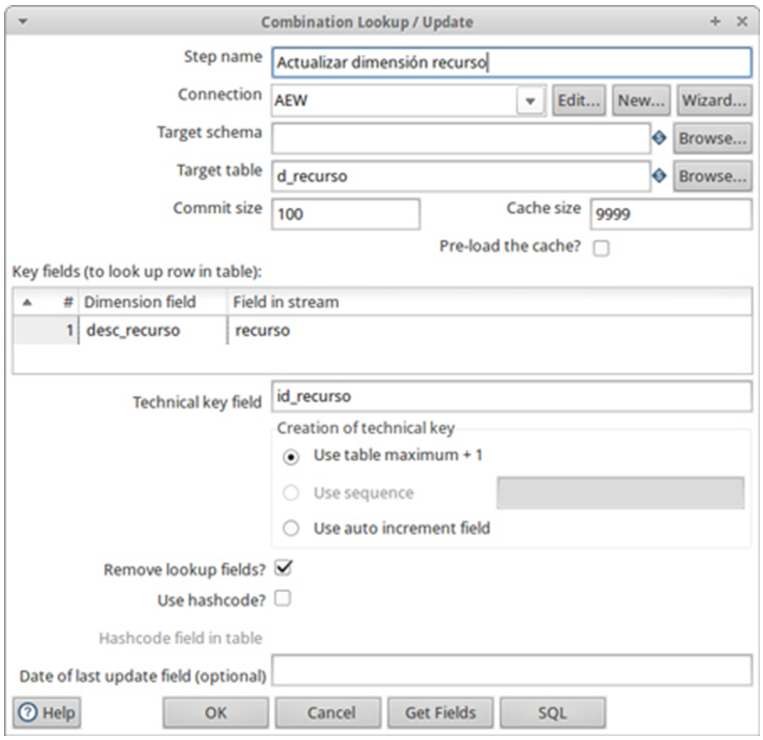
Figura 21. Cortar cadena



Fuente: Josep Curto.

e) Se actualiza la dimensión recurso a partir de la información entrante y se recupera el id_recurso mediante el paso *combination lookup / update*, que hace una búsqueda o actualiza.

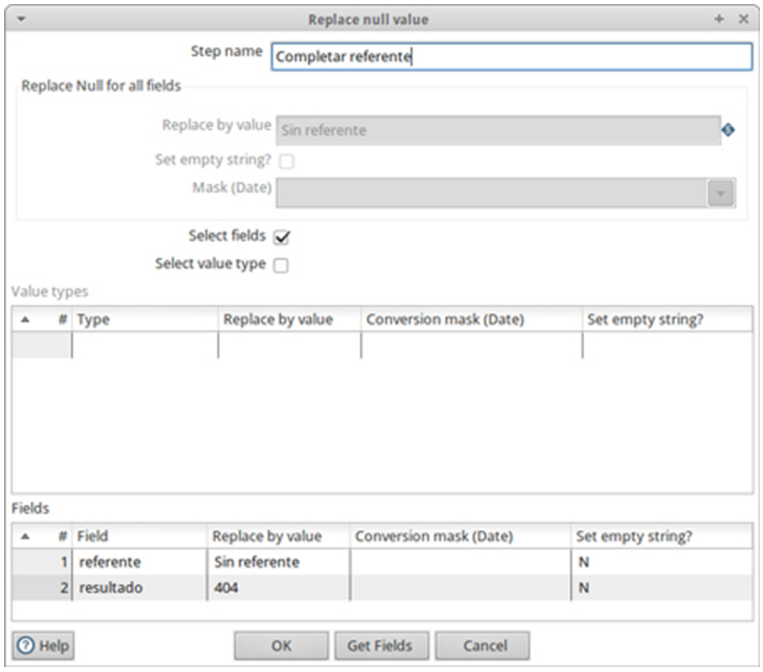
Figura 22. Actualizar dimensión recursos



Fuente: Josep Curto.

f) Completamos los nulos del campo referente mediante el paso *replace null value*.

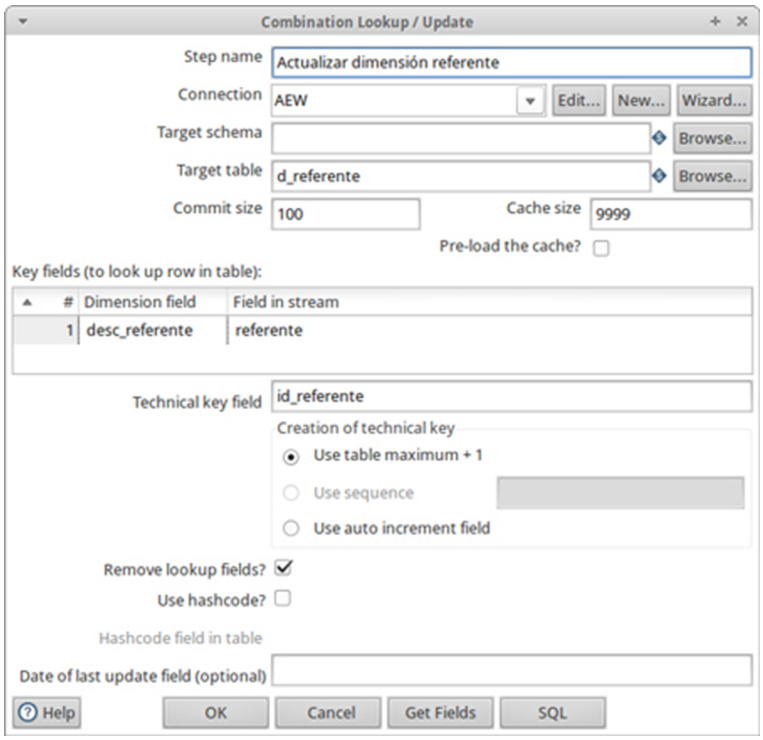
Figura 23. Tratamiento de los valores nulos



Fuente: Josep Curto.

g) Se actualiza la dimensión referente a partir de la información entrante y se recupera el id_referente mediante el paso *combination lookup / update*.

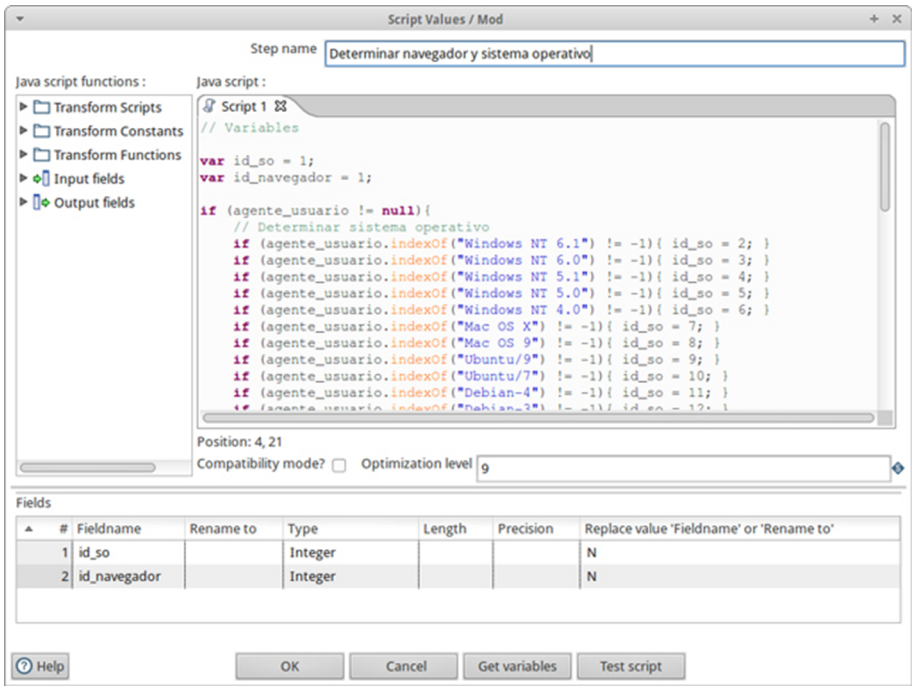
Figura 24. Actualizar dimensión referente



Fuente: Josep Curto.

h) A partir de la información en el campo agente_usuario, se determina el navegador y sistema operativo mediante el paso *script value modified javascript*.

Figura 25. Determinar navegador y sistema operativo mediante *script*



Fuente: Josep Curto.

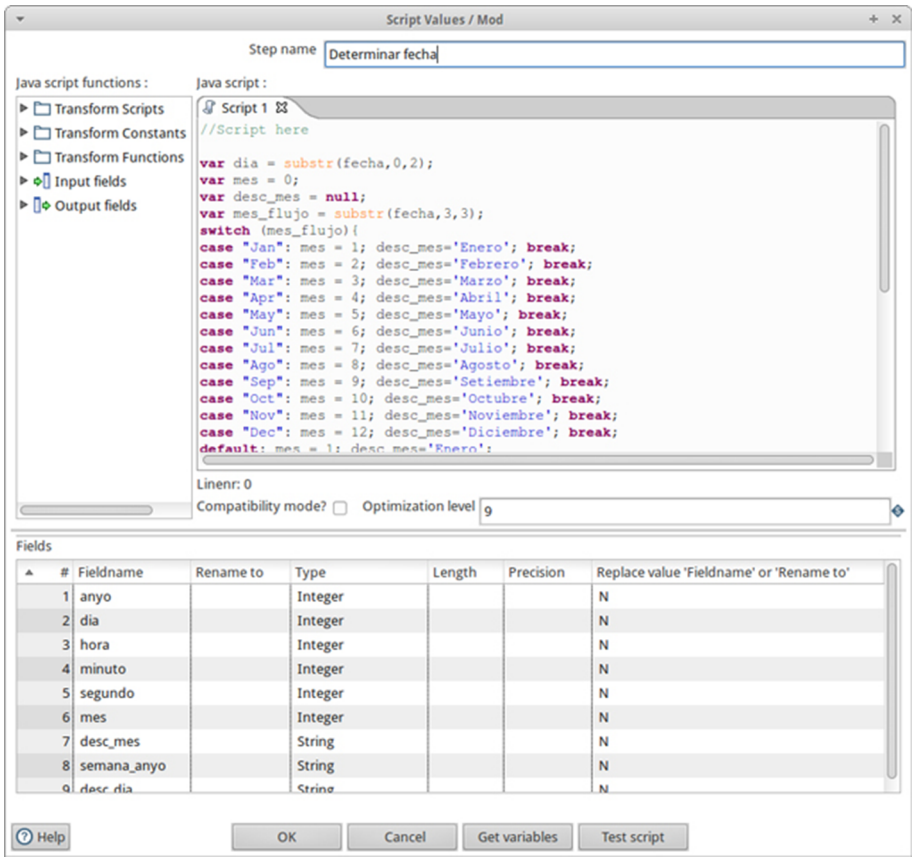
El código usado, y susceptible de mejora, es:

```
var id_so = 1;
var id_navegador = 1;

if (agente_usuario != null){
        // Determinar sistema operativo
        if (agente_usuario.indexOf("Windows NT 6.1") != -1){ id_so = 2; }
        if (agente_usuario.indexOf("Windows NT 6.0") != -1){ id_so = 3; }
        if (agente_usuario.indexOf("Windows NT 5.1") != -1){ id_so = 4; }
        if (agente_usuario.indexOf("Windows NT 5.0") != -1){ id_so = 5; }
        if (agente_usuario.indexOf("Windows NT 4.0") != -1){ id_so = 6; }
        if (agente_usuario.indexOf("Mac OS X") != -1){ id_so = 7; }
        if (agente_usuario.indexOf("Mac OS 9") != -1){ id_so = 8; }
        if (agente_usuario.indexOf("Ubuntu/9") != -1){ id_so = 9; }
        if (agente_usuario.indexOf("Ubuntu/7") != -1){ id_so = 10; }
        if (agente_usuario.indexOf("Debian-4") != -1){ id_so = 11; }
        if (agente_usuario.indexOf("Debian-3") != -1){ id_so = 12; }
        if (agente_usuario.indexOf("Fedora") != -1){ id_so = 13; }
        if (agente_usuario.indexOf("SUSE") != -1){ id_so = 14; }
        if (agente_usuario.indexOf("Solaris") != -1){ id_so = 15; }
        if (agente_usuario.indexOf("Mandriva") != -1){ id_so = 16; }
        if (agente_usuario.indexOf("Red Hat") != -1){ id_so = 17; }
        if (agente_usuario.indexOf("OpenSUSE") != -1){ id_so = 18; }
        // Determinar navegador
        if (agente_usuario.indexOf("MSIE 3") != -1){ id_navegador = 2; }
        if (agente_usuario.indexOf("MSIE 4") != -1){ id_navegador = 3; }
        if (agente_usuario.indexOf("MSIE 5") != -1){ id_navegador = 4; }
        if (agente_usuario.indexOf("MSIE 5.5") != -1){ id_navegador = 5; }
        if (agente_usuario.indexOf("MSIE 6") != -1){ id_navegador = 6; }
        if (agente_usuario.indexOf("MSIE 7") != -1){ id_navegador = 7; }
        if (agente_usuario.indexOf("MSIE 8") != -1){ id_navegador = 8; }
        if (agente_usuario.indexOf("Avant") != -1){ id_navegador = 9; }
        if (agente_usuario.indexOf("SeaMonkey/1") != -1){ id_navegador = 10; }
        if (agente_usuario.indexOf("Camino/1") != -1){ id_navegador = 11; }
        if (agente_usuario.indexOf("Firefox/1") != -1){ id_navegador = 12; }
        if (agente_usuario.indexOf("Firefox/2") != -1){ id_navegador = 13; }
        if (agente_usuario.indexOf("Firefox/3") != -1){ id_navegador = 14; }
        if (agente_usuario.indexOf("Firefox/4") != -1){ id_navegador = 15; }
        if (agente_usuario.indexOf("Flock/1") != -1){ id_navegador = 16; }
        if (agente_usuario.indexOf("Galeon/2") != -1){ id_navegador = 17; }
        if (agente_usuario.indexOf("Epiphany/2") != -1){ id_navegador = 18; }
        if (agente_usuario.indexOf("K-Meleon/1") != -1){ id_navegador = 19; }
        if (agente_usuario.indexOf("Opera/9") != -1){ id_navegador = 20; }
        if (agente_usuario.indexOf("Opera/10") != -1){ id_navegador = 21; }
        if (agente_usuario.indexOf("Konqueror/3") != -1){ id_navegador = 22; }
        if (agente_usuario.indexOf("Safari/3") != -1){ id_navegador = 23; }
        if (agente_usuario.indexOf("Safari/4") != -1){ id_navegador = 24; }
        if (agente_usuario.indexOf("Camino/2") != -1){ id_navegador = 25; }
        if (agente_usuario.indexOf("SeaMonkey/2") != -1){ id_navegador = 26; }
        if (agente_usuario.indexOf("SeaMonkey/3") != -1){ id_navegador = 27; }
        if (agente_usuario.indexOf("Konqueror/4") != -1){ id_navegador = 28; }
        if (agente_usuario.indexOf("Netscape/7") != -1){ id_navegador = 29; }
        if (agente_usuario.indexOf("Netscape/8") != -1){ id_navegador = 30; }
        if (agente_usuario.indexOf("Netscape/9") != -1){ id_navegador = 31; }
    }
```

i) Se determina la fecha mediante el paso *script value modified javascript*.

Figura 26. Determinar fecha mediante *script*



Fuente: Josep Curto.

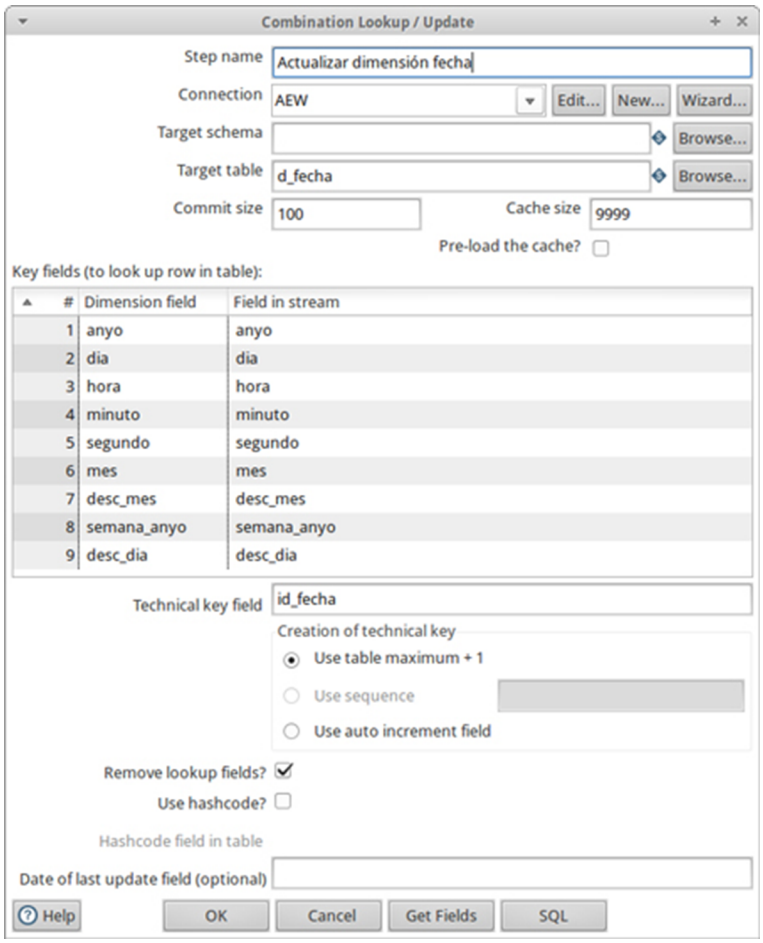
El código usado, y susceptible de mejora, es:

```
var dia = substr(fecha,0,2);
var mes = 0;
var desc_mes = null;
var mes_flujo = substr(fecha,3,3);
var (mes_flujo){
     case "Jan": mes = 1; desc_mes='Enero'; break;
     case "Feb": mes = 2; desc_mes='Febrero'; break;
     case "Mar": mes = 3; desc_mes='Marzo'; break;
     case "Apr": mes = 4; desc_mes='Abril'; break;
     case "May": mes = 5; desc_mes='Mayo'; break;
     case "Jun": mes = 6; desc_mes='Junio'; break;
     case "Jul": mes = 7; desc_mes='Julio'; break;
     case "Ago": mes = 8; desc_mes='Agosto'; break;
     case "Sep": mes = 9; desc_mes='Setiembre'; break;
     case "Oct": mes = 10; desc_mes='Octubre'; break;
     case "Nov": mes = 11; desc_mes='Noviembre'; break;
     case "Dec": mes = 12; desc_mes='Diciembre'; break;
     default: mes = 1; desc_mes='Enero';
}
var anyo = substr(fecha,7,4);
var hora = substr(fecha,12,2);
var minuto = substr(fecha,15,2);
var segundo = substr(fecha,18,2);

var semana_anyo = week(new Date(anyo,mes,dia));
var desc_dia = null;
switch ((new Date(anyo,mes,dia)).getDay()){
     case 0: desc_dia='Domingo'; break;
     case 1: desc_dia='Lunes'; break;
     case 2: desc_dia='Martes'; break;
     case 3: desc_dia='Miércoles'; break;
     case 4: desc_dia='Jueves'; break;
     case 5: desc_dia='Viernes'; break;
     case 6: desc_dia='Sábado'; break;
     default: desc_dia='Domingo';
}
```

j) Se actualiza la fecha y se recupera el id_fecha mediante el paso *combination lookup / update*.

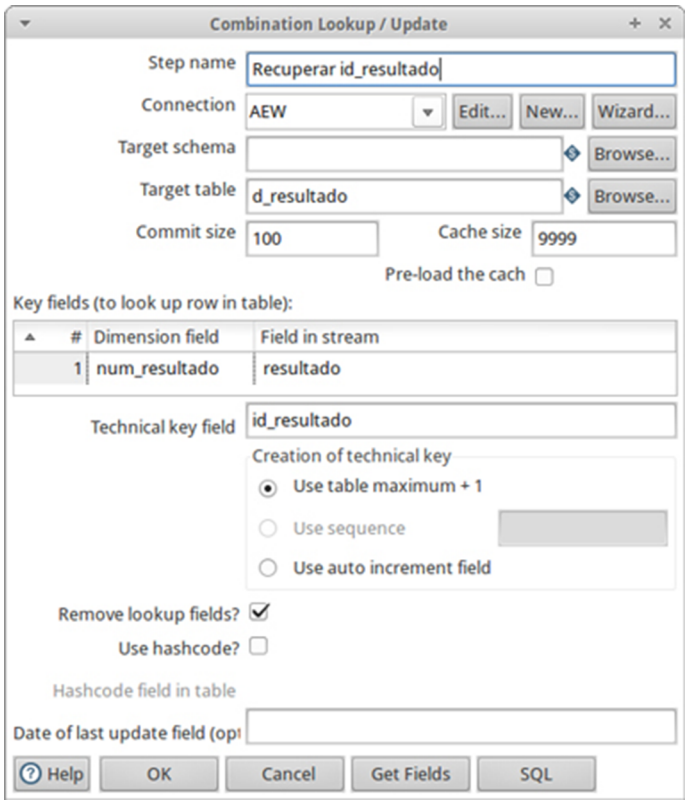
Figura 27. Actualizar dimensión fecha



Fuente: Josep Curto.

k) Se recupera id_resultado de la base de datos a partir de la información en el flujo mediante el paso *combination lookup / update*.

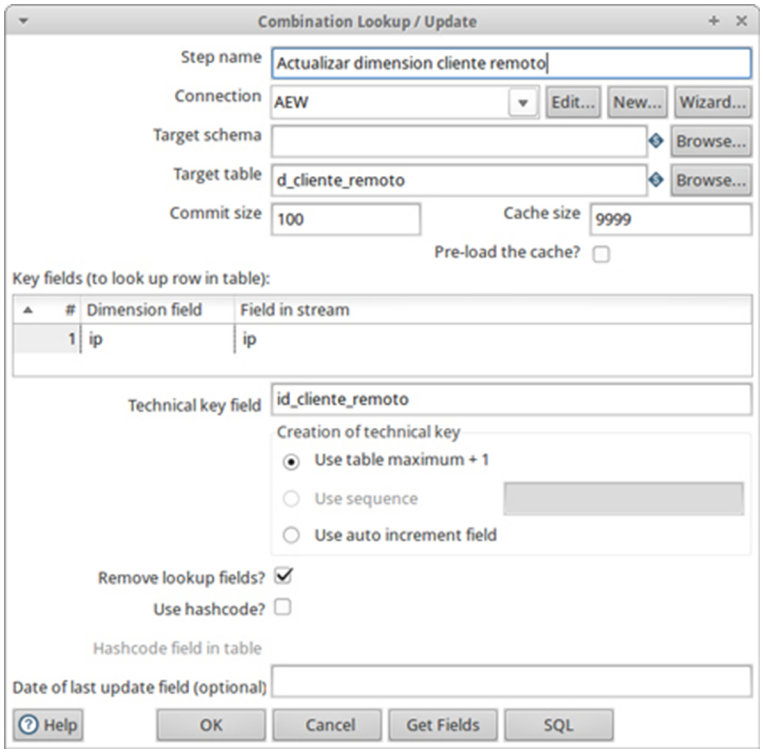
Figura 28. Recuperar id_resultado



Fuente: Josep Curto.

l) Se actualiza la dimensión cliente remoto y se recupera el id_cliente_remoto mediante el paso *combination lookup / update*.

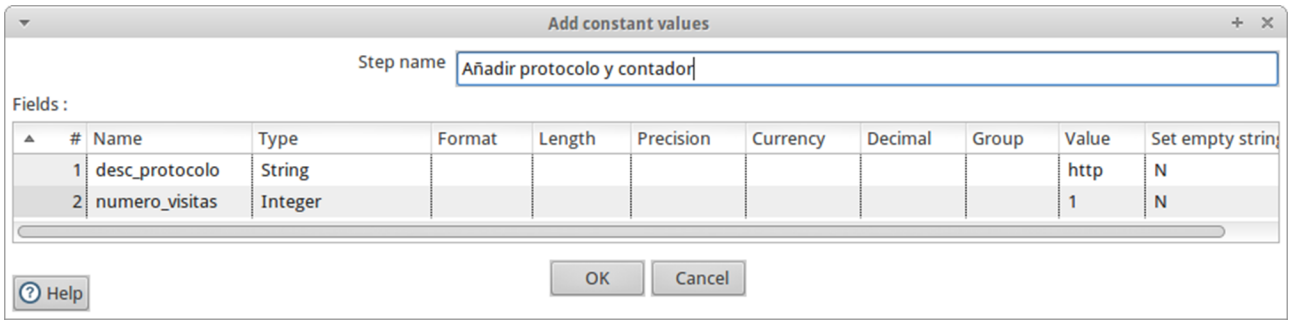
Figura 29. Actualizar dimensión cliente remoto



Fuente: Josep Curto.

m) Se añade el protocolo y el contador mediante el paso *add constant values*.

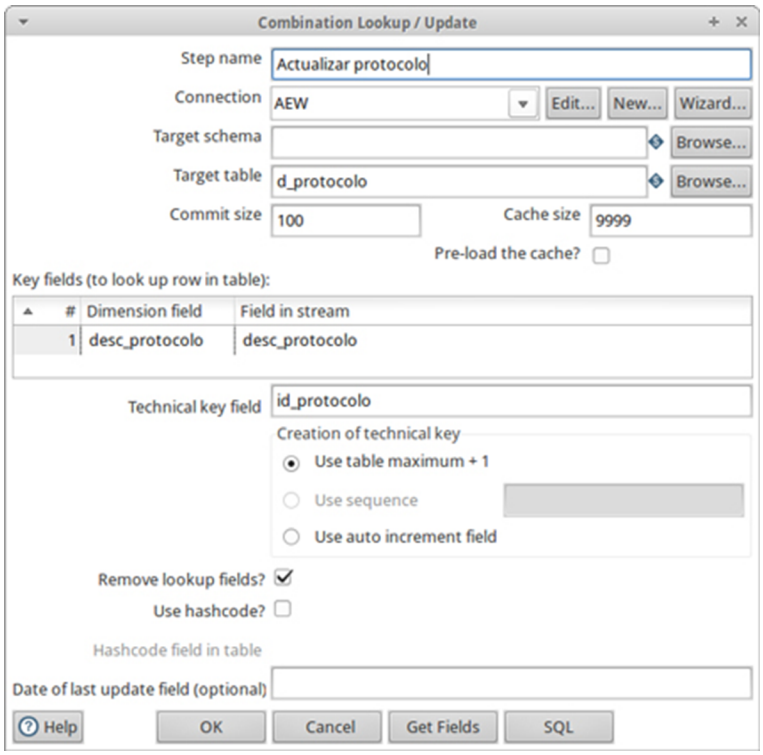
Figura 30. Añadir protocolo y visitas como constante



Fuente: Josep Curto.

n) Se actualiza la dimensión protocolo y se recupera el id_protocolo mediante el paso *combination lookup / update*.

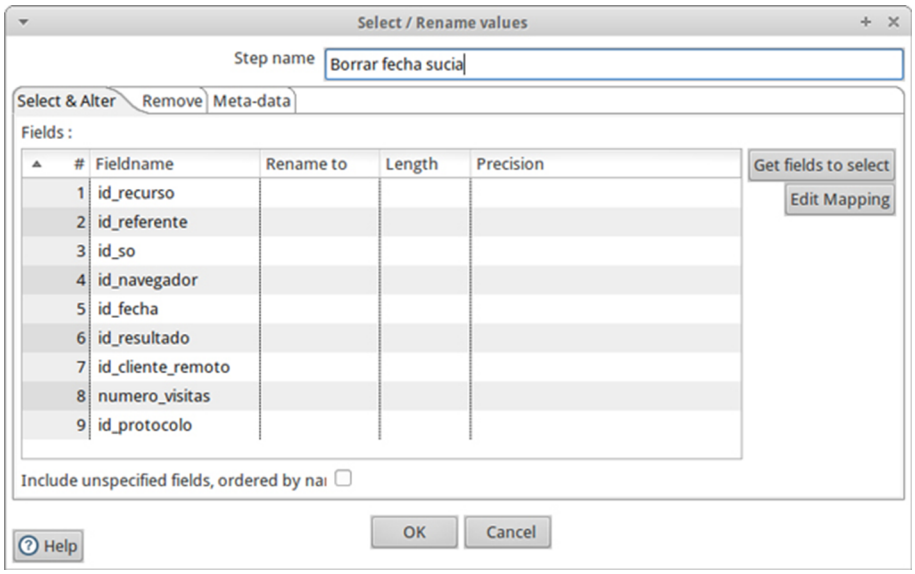
Figura 31. Actualizar dimensión protocolo



Fuente: Josep Curto.

ñ) Se quitan del flujo los campos que no son necesarios mediante el paso *select / rename values*.

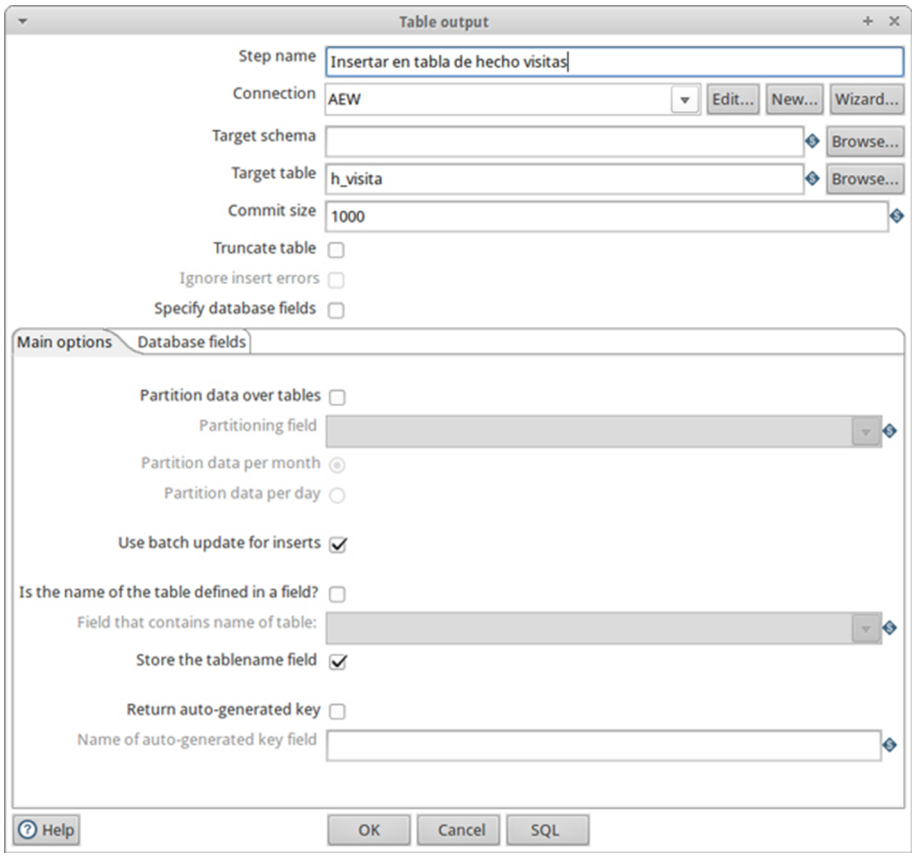
Figura 32. Filtrar valores relevantes



Fuente: Josep Curto.

o) Se inserta la información en la tabla de hecho visitas mediante el paso *table output*.

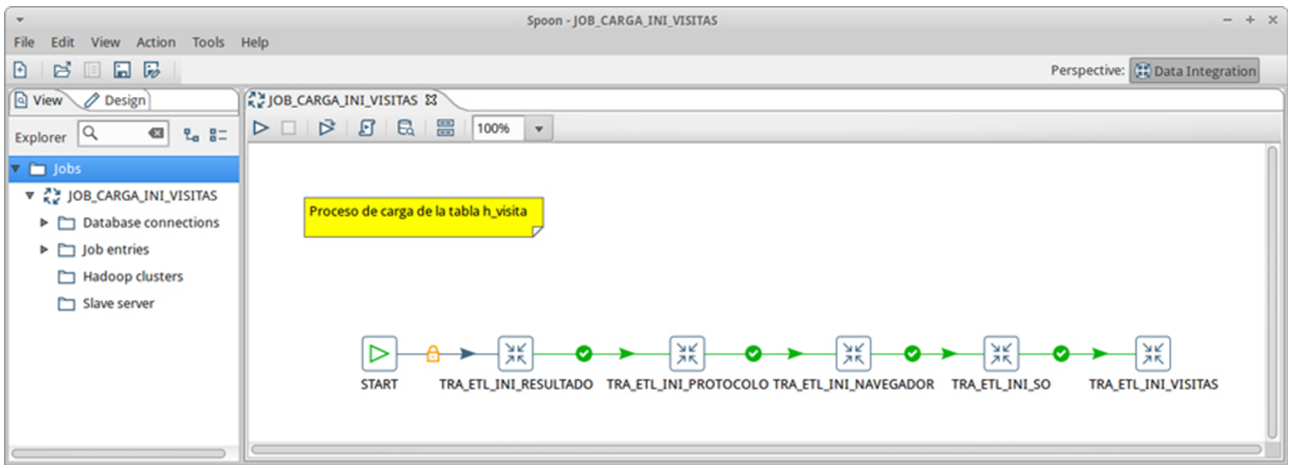
Figura 33. Insertar tabla de hecho



Fuente: Josep Curto.

p) Finalmente, se diseña un trabajo para ejecutar de modo secuencial todas las transformaciones.

Figura 34. Trabajo para ejecutar transformaciones



Fuente: Josep Curto.

Abreviaturas

BI *Business intelligence.*

CDC *Change data capture.*

CDI *Customer data integration.*

CPM *Corporate performance management.*

CPU *Central processing unit.*

CSV *Comma separated value.*

EAI *Enterprise application integration.*

EDR *Environmental data record.*

EII *Enterprise information integration.*

EIM *Enterprise information management.*

ETL *Extract, transform and load.*

HTML *Hypertext markup language.*

JDBC *Java database connectivity.*

JNDI *Java naming and directory interface.*

ODBC *Open database connectivity.*

ODS *Operational data store.*

OLAP *Online analytical processing.*

PDF *Portable document format.*

PDI *Pentaho data integration.*

PIM *Product information management.*

SCD *Slowly changing dimension.*

SI Sistemas de información.

SQL *Structured query language.*

XML *Extensible markup language.*

Bibliografía

Bouman, R.; Van Dongen, J. (2009). *Pentaho® Solutions: Business Intelligence and Data Warehousing with Pentaho® and MySQL*. Indianápolis: Wiley Publishing.

Inmon, W. H. (2005). *Building the Data Warehouse* (4.ª ed.). Hoboken: John Wiley & Sons.

Inmon, W. H.; Strauss, D.; Neushloss, G. (2008). *DW 2.0: The Architecture for the next generation of Data Warehousing*. Burlington: Morgan Kaufman Series.

Kimball, R. (2009). *Data Warehouse Toolkit Classics: The Data Warehouse Toolkit* (2.ª ed.); *The Data Warehouse Lifecycle Toolkit* (2.ª ed.); *The Data Warehouse ETL Toolkit*. Hoboken: John Wiley & Sons.

Knight, B. (2009). *Professional Microsoft SQL Server 2008 Integration Services*. Indianápolis: Wrox.

Anexo

Subsistemas ETL

Según la clasificación de Ralph Kimball, existen treinta y cuatro subsistemas ETL clasificados en cuatro grandes grupos:

- Extracción: consiste en extraer la información de la fuente de origen.
- Limpieza y conformación: consiste en acciones que permiten validar y aumentar la calidad de la información.
- Entrega: consiste en la preparación de la información para su posterior entrega.
- Gestión: aúna las tareas de administración de procesos ETL.

Vamos a detallar cada uno de los grupos.

1) Extracción

- ***Data profiling*** **(subsistema 1)**: consiste en la exploración de los datos para verificar su calidad y si cumple los estándares conforme los requerimientos.
- ***Change data capture*** **(subsistema 2)**: detecta los cambios para refinar los procesos ETL y mejorar su rendimiento.
- **Sistema de extracción (subsistema 3)**: permite la extracción de datos desde la fuente de origen a la fuente destino.

2) Limpieza y conformación

- ***Data cleaning*** **(subsistema 4)**: implementa los procesos de calidad de datos que permite detectar las incoherencias de calidad.
- **Rastreo de eventos de errores (subsistema 5)**: captura todos los errores que proporcionan información valiosa sobre la calidad de datos y permiten la mejora de estos.
- **Creación de dimensiones de auditoría (subsistema 6)**: permite crear metadatos asociados a cada tabla. Estos metadatos permiten validar la evolución de la calidad de los datos.

- **Deduplicación (subsistema 7)**: eliminar información redundante de tablas importantes como cliente o producto. Requiere cruzar múltiples tablas en múltiples sistemas de información para detectar el patrón que permite identificar cuándo una fila está duplicada.

- **Conformación (subsistema 8)**: permite identificar elementos equivalentes que permiten compartir información entre tablas relacionadas.

3) Entrega

- ***Slowly changing dimension* (SCD) (subsistema 9)**: implementa la lógica para crear atributos de variabilidad lenta a lo largo del tiempo.

- ***Surrogate key* (subsistema 10)**: permite crear claves subrogadas independientes para cada tabla.

- **Jerarquías (subsistema 11)**: permite hacer inserciones en estructuras jerárquicas de tablas.

- **Dimensiones especiales (subsistema 12)**: permite crear dimensiones especiales, como *junk*, mini o de etiquetas.

- **Tablas de hecho (subsistema 13)**: permite crear tablas de hecho.

- ***Pipeline* de claves subrogadas (subsistema 14)**: permite remplazar las claves operacionales por las claves subrogadas.

- **Constructor de tablas multivaluadas (subsistema 15)**: permite construir tablas puente para soportar las relaciones N:M.

- **Gestión para información tardía (subsistema 16)**: permite aplicar modificaciones a los procesos en caso de que los datos tarden en llegar.

- **Gerente de dimensión (subsistema 17)**: autoridad central que permite crear y publicar dimensiones conformadas.

- **Aprovisionador de tablas de hecho (subsistema 18)**: permite la gestión de las tablas de hecho.

- **Creador de agregadas (subsistemas 19)**: permite gestionar agregadas.

- **Creador de cubos OLAP (subsistema 20)**: permite alimentar de datos a esquemas OLAP desde esquema dimensionales relacionales.

- **Propagador de datos (subsistema 21)**: permite preparar información conformada para ser entregada para cualquier propósito especial.

4) Gestión

- **Programador de trabajos (subsistema 22)**: permite gestionar ETL de la categoría de trabajos.

- **Sistema de *backup* (subsistema 23)**: realiza copias de respaldo de los procesos ETL.

- **Reinicio y recuperación (subsistema 24)**: permite reiniciar un proceso ETL en el caso de error.

- **Control de versiones (subsistema 25)**: permite hacer control de versiones de un proyecto ETL y de los metadatos asociados.

- **Migración de versiones (subsistema 26)**: permite pasar proyectos en fase test a producción mediante versionado.

- **Monitorización de *workflow* (subsistema 27)**: dado que un proceso de ETL es un *workflow*, es necesario monitorizarlos para medir su rendimiento.

- **Ordenación (subsistema 28)**: permite calibrar los procesos ETL para mejorar su rendimiento.

- **Linealidad y dependencia (subsistema 29)**: identifica elementos dependientes. Permite identificar las transformaciones en las que participa o ha participado. Permite la trazabilidad del dato.

- **Escalado de problemas (subsistemas 30)**: suporta la gestión de incidencias.

- **Paralelismo/*clustering* (subsitema 31)**: permite el uso de procesos en paralelo, *grid computing* y *clustering* para mejorar el rendimiento y reducir tiempo del proceso.

- **Seguridad (subsistemas 32)**: gestiona el acceso a ETL y metadatos.

- ***Compliance manager* (subsistema 33)**: permite soportar la legislación vigente respecto a la custodia y responsabilidad de datos que debe aplicarse a la organización.

- **Repositorio de metadatos (subsistema 34)**: captura los metadatos de los procesos ETL, de los datos de negocio y de los aspectos técnicos.