Universidad Central "Marta Abreu" de Las Villas.

Facultad Matemática Física y Computación

Licenciatura en Ciencia de la Computación

TRABAJO DE DIPLOMA

Pasos del Pentaho Data Integration en un contexto *big data*

Autor: **Yuniel Morejón Hernández**

Tutor: **MSc. Lisandra Díaz De la Paz**

Lic. Juan Luis García Mendoza

"Año 57 de la Revolución"

Santa Clara

2015

DICTAMEN

Hago constar que el presente trabajo fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Licenciatura en Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del tutor

Firma del jefe del Laboratorio

El camino del éxito es angosto, difícil pero gratificante.

Dedicatoria

A mi mamá y mi papá:

Por la educación, el cariño y el apoyo recibido en todo momento, por querer y desearme lo mejor para mí y porque gracias a ellos soy, quien soy hoy en día.

A mi hermano:

Por estar junto a mí ayudándome y apoyándome prácticamente en todo momento de mi vida, por haberme permitido compartir con el parte de su experiencia en su momento de estudiante de la UCI, sus amigos que fueron amigos míos también.

A mis abuelos:

Por quererme como si fuera un hijo para ellos, preocupándose por mí en todo momento.

A mi hermana:

Por desearme tanto bien y porque la quiero mucho.

A mi sobrinito Arian de Jesus:

Porque lo quiero mucho y le deseo lo mejor.

A todos mis tíos:

Mi tío Osvaldo, mi tía Osoria, mi tía Milagrito, el Niño, el Nene, Oria que me quieren tanto.

A mis primos:

En especial mi primo Osvaldito que es como un hermano.

A toda mi familia:

Que me ha brindado apoyo y cariño.

Agradecimientos

- *A mi mamá, mi papá y mi hermano porque gracias a ellos es que estoy discutiendo este trabajo de diploma y por ser lo más grande que tengo en este mundo.*
- *A mis tutores MSc. Lisandra Díaz de la paz y mi amigo Lic. Juan luis García Mendoza por toda la ayuda brindada por el tiempo que me dedicaron en las revisiones y por numerosos consejos en la elaboración del trabajo de diploma.*
- *Dr. Amed Abel Leiva Mederos como principal consultante sobre el dominio de aplicación ABCD.*
- *A mis amigos que me acompañaron en los años de universidad y compartimos horas de estudio y los que me apoyaron en los momentos difíciles de la carrera.*
- *A todos aquellos que no he mencionado y de una forma u otra me dieron su ayuda.*

RESUMEN

El término *big data* hace referencia a una inmensa y compleja colección de datos (estructurados, no estructurados, semi-estructurados y mixtos) la cual, debido a su gran tamaño y características, imposibilita su tratamiento por medio de los tradicionales sistemas de bases de datos y aplicaciones de procesamiento de datos. En el presente trabajo de diploma se describen las tecnologías que se utilizan en cada fase *de big data* (recolección, almacenamiento, procesamiento, análisis y visualización) dentro de estas tecnologías se destaca el ecosistema Hadoop. Además se caracterizan los pasos del Pentaho Data Integration relacionados con *big data*. Finalmente se implementan transformaciones donde se ejemplifica el uso de *big data* en el Pentaho Data Integration y se pone de manifiesto su relación con los pasos de limpieza de datos en aras de resolver problemas de calidad de datos detectados en la suite Sistema Integrado de Automatización de Bibliotecas y Centros de Documentación.

Palabras claves: ABCD, almacenamiento, análisis, *big data*, Hadoop, PDI, procesamiento, recolección, visualización.

ABSTRACT

The term big data refers to a large and complex data sets (structured, non-structured, semi-structured and mixed) which, due to their large size and characteristics, precludes their treatment by traditional database systems and data processing applications. In this dissertation the technologies used in each phase of big data (collection, storage, processing, analysis and visualization) within the Hadoop ecosystem these technologies stands are described. Further characterized Pentaho Data Integration steps related to big data. Finally transformations where the use of big data is exemplified in the Pentaho Data Integration and shows their relationship to data cleansing steps in order to resolve data quality problems identified in the Integrated System of Automation of Libraries and Documentation Centers suite are implemented.

Keywords: ABCD, storage, analysis, big data, Hadoop, PDI, processing, collection, visualization.

Tabla de Contenido

Tabla de Contenidos.

Tabla de Contenido

Tabla de Contenido

Tabla de Figuras.

Tabla de Figuras

Lista de Tablas.

Lista de tablas

INTRODUCCIÓN.

Teniendo presente la diversidad de formatos, tipos y fuentes de datos que existen en la actualidad, se hace imprescindible contar con procesos de extracción, transformación, limpieza y carga (ETL) que permitan la manipulación de estos datos con la mayor calidad posible.

La herramienta Spoon, entorno de desarrollo integrado basado en Standard Widget Toolkit (SWT), pertenece al Pentaho Data Integration (PDI), permite realizar procesos ETL con su interfaz gráfica mediante la creación de transformaciones y trabajos (Casters et al., 2010).

En los últimos años la manera en que los usuarios interactúan con la tecnología ha cambiado de manera radical debido a la constante evolución de ésta. Revoluciones tecnológicas como la Web 2.0, blogs, foros de opinión, redes sociales, multimedia y dispositivos como los teléfonos inteligentes facilitan la conectividad y la generación de grandes cantidades de información que hasta hace poco eran impensables. Y no solo la sociedad de consumo ha avanzado tecnológicamente; campos como la ciencia, medicina o la economía también requieren cada vez más tratar con grandes cantidades de datos. Big data es el sector de las tecnologías de la información y las comunicaciones (TIC) que se preocupa de como almacenar y tratar grandes cantidades de información o conjuntos de datos de la manera más rápida y eficiente posible, adaptándose a todos los formatos estructurados semi-estructurados o no estructurados (Serrat Morros 2013).

Para el proceso de vastos volúmenes de datos existen tecnologías *big data* como Hadoop creado por Doug Cutting (Kestelyn, 2013), que lo nombró así por su elefante de juguete. Es un proyecto de alto nivel que está siendo usado por una comunidad global de contribuidores, mediante el lenguaje de programación java. Se desarrolló originalmente para apoyar la distribución del proyecto de motor de búsqueda Nutch (Holmes, 2012).

En trabajos de diplomas como "Mercado de datos para el departamento de Recursos Humanos de la UCLV", "Estudio del Pentaho Data Integration en los procesos de

integración de datos (ETL)" y "Automatización de los procesos de carga en el mercado de datos Recursos Humanos de la UCLV" se expone la utilidad de la herramienta Spoon para efectuar procesos ETL.

Para el tratamiento de *big data* Spoon incluye una serie de pasos tanto para desarrollar transformaciones como trabajos. Hasta ahora en la UCLV a nivel de grupo de investigación no se ha puesto en práctica algún proyecto diploma que involucre el uso y análisis de los pasos *big data* que soporta el PDI. Por consiguiente, en el presente proyecto de diploma se establece la relación que existe entre entradas y salidas *big data* que ofrece el PDI, transformaciones que conlleve el uso de pasos de limpieza de datos y trabajos que se pueden implementar. Para ello se toma como dominio de aplicación la base de datos no solamente sql (NoSQL) MARC perteneciente a la suite de Automatización de Bibliotecas y Centros de Documentación (ABCD) utilizada en todas las bibliotecas de la UCLV.

Objetivo general

Analizar la relación que existe entre la herramienta Pentaho Data Integration (PDI) con las tecnologías de *big data*, en aras de conocer su funcionamiento y determinar si para entradas *big data* los pasos de limpieza de datos implementados en el PDI pueden ser aplicados.

Objetivos específicos

1. Describir las tecnologías que se utilizan en cada fase de *big data*.
2. Caracterizar los pasos del PDI relacionados con *big data*.
3. Implementar transformaciones donde se ejemplifique el uso de *big data* en el PDI y se ponga de manifiesto su relación con los pasos de limpieza de datos en aras de resolver problemas de calidad de datos detectados en la suite ABCD.

Preguntas de investigación

1. ¿Qué características tienen las tecnologías que se utilizan en cada fase de *big data*?
2. ¿Cómo se configuran los pasos del PDI relacionados con *big data*?
3. ¿Qué pasos de limpieza de datos se pueden utilizar en la implementación de transformaciones y trabajos ante entradas y salidas *big data* para resolver problemas de calidad de datos de la suite ABCD?

Justificación de la investigación

Hoy en día la tendencia es a la aparición de datos cuya representación, almacenamiento, velocidad de arribo y procesamiento de los datos va más allá de las posibilidades que brindan los sistemas de bases de datos relacionales tradicionales. Con la llegada de los *big data* se ha abierto paso a las bases de datos NoSQL, las cuales permiten consultar grandes volúmenes de datos de manera rápida. Mientras mayor es el volumen de datos, mayor es la probabilidad de que estos presenten problemas de calidad y poder solucionar estos problemas constituye un desafío en la era de los *big data*. Por tanto, surge la necesidad de estudiar herramientas bien definidas y de amplio uso a nivel mundial como es el caso del PDI, de manera que se describa el funcionamiento de la misma ante pasos *big data*, haciendo énfasis en la implementación de transformaciones y trabajos donde se visualice el uso de pasos de limpieza de datos para dar solución a problemas de calidad de datos detectados en la base de datos NoSQL de la suite ABCD. Con lo cual el presente trabajo de diploma tiene implicaciones prácticas, valor teórico e impacto social porque el sistema de gestión bibliotecaria ABCD debe utilizarse en todos los Centros de Educación Superior (CES) del país.

El presente trabajo se estructura en tres capítulos:

En el capítulo I se presentan varios conceptos de *big data* desde el punto de vista de diferentes autores. Se exponen las principales características de los *big data*, las fases para su gestión y una arquitectura donde se aprecia su funcionamiento. Además, se exponen las diferentes tecnologías para el trabajo con *big data*. Finalmente se describen las principales características y ventajas de utilizar la herramienta PDI.

En el capítulo II se caracterizan las tecnologías *big data* que se encuentran en el PDI asociadas a los diferentes pasos para la implementación de transformaciones y trabajos, además se explica y visualiza mediante imágenes la configuración de estos pasos *big data*, finalmente se desarrolla una trasformación y un trabajo que hacen uso de algunos de los pasos *big data* que ofrece el PDI.

En el capítulo III se hace una breve caracterizaciónón de la suite ABCD (Sistema Integrado de Automatización de Bibliotecas y Centros de Documentación), se exponen los

componentes principales del formato MARC21 conciso, se mencionan del total de campos de dicho formato los que se seleccionaron para primeramente detectar las anomalías de la información almacenada, para posteriormente dado un esquema general de implementación de transformaciones que den solución a los errores encontrados, se implementan estas transformaciones mediante el PDI mostrando además el tiempo de ejecución brindado por este software al ejecutarlas.

CAPÍTULO I: ASPECTOS RELEVANTES SOBRE LAS TECNOLOGÍAS BIG DATA Y PENTAHO DATA INTEGRATION.

En el presente capítulo se mencionan varios conceptos de *big data* propuestos por diversos autores. Se exponen las principales características que poseen los *big data*, las fases para su manejo y una arquitectura basada en dichas fases donde se percibe su funcionamiento. Además, se presentan las disímiles tecnologías para el trabajo con *big data*. Finalmente se describen las principales características y ventajas de utilizar la herramienta Pentaho Data Integration.

1.1 Definiciones de *big data.*

Según (Sánchez González et al., 2014) el término *big data* hace referencia a una inmensa y compleja colección de datos (estructurados, no estructurados y semi-estructurados), la cual debido a su gran tamaño y características, imposibilita su tratamiento por medio de los tradicionales sistemas de bases de datos.

Big Data es un término aplicado a conjuntos de datos que superan la capacidad habitual de los *software* para capturarlos, gestionarlos y procesarlos en un tiempo razonable *(Rouse, 2011).*

Según (Laney, 2001), son activos de información caracterizados por su alto volumen, velocidad y variedad, que demandan soluciones innovadoras y eficientes de procesamiento para la mejora del conocimiento y toma de decisiones en las organizaciones.

1.2 Características de *big data.*

Big Data es descrita según (Laney, 2001) mediante el modelo de las tres V: *volumen, velocidad y variedad*.

- *Volumen:* Muchos factores han contribuido a que el volumen de los datos haya incrementado exponencialmente en las últimas décadas. Una de las causas de este crecimiento se debe al almacenamiento de datos de ubicación procedentes de dispositivos móviles (ubicación GPS, sensores inalámbricos), sistemas de

teledetección de cámaras, micrófonos, lectores de radiofrecuencia… Estos datos no estructurados crecen muy rápidamente.

- *Variedad:* Actualmente los datos se encuentran en diferentes formatos y provienen de una amplia gama de fuentes y dispositivos. Desde datos estructurados en sistemas de bases de datos (BD) tradicionales, semi-estructurados como documentos XML, JSON, hasta no estructurados tales como documentos de texto, redes sociales, emails, videos, imágenes, audios, datos numéricos científicos (áreas como la meteorología, la genética…) o sistemas de finanzas, entre otros.
- *Velocidad:* Determinada la cantidad y variedad de datos para procesar, la velocidad no es más que el tiempo de arribo y de procesamiento que se precisa para obtener la información correcta en un determinado momento. El tiempo de arribo pude ser en tiempo real, por lotes o por flujos de trabajo.

1.2.1 Clasificación de *big data.*

Según (Hashem et al., 2014), big data se clasifica en base a cinco aspectos: (i) origen de datos, (ii) contenido del formato, (iii) almacenamiento de datos, (iiii) organización o preparación de los datos, (iiiii) procesamiento de datos, ver Figura 1.

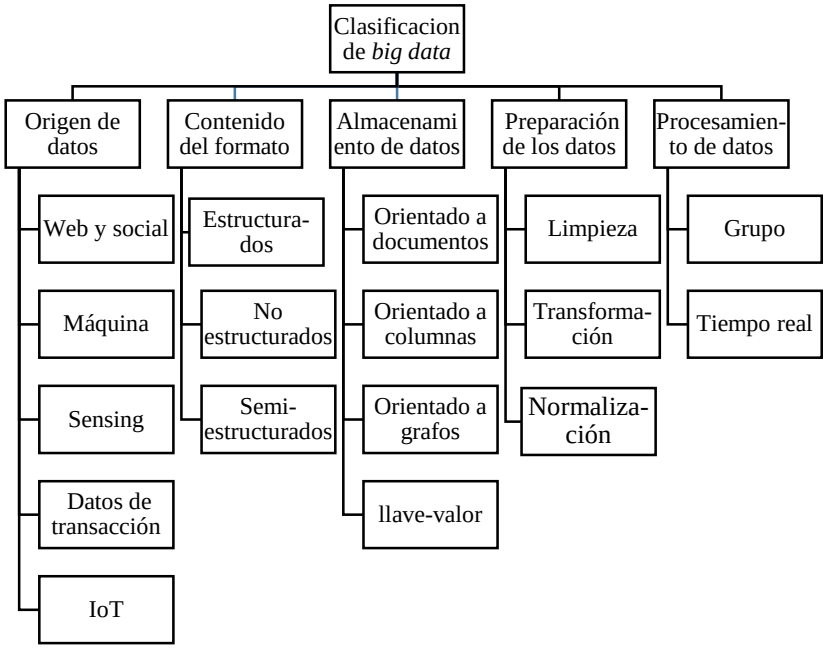


Figura 1- Clasificación de *big data*. Fuente [(Hashem et al., 2014)

1.2.1.1 Contenido del formato.

Datos Estructurados: Los datos estructurados constan de información ya procesada y filtrada, a menudo son tratados mediante el lenguaje estandarizado de consultas SQL, creado para la gestión y consulta de datos en sistemas gestores de bases de datos relacionales (RDBMS). Los datos estructurados son fáciles de importar, consultar, almacenar y analizar. Ejemplos de datos estructurados incluyen números, palabras, fechas y sobre todo bases de datos relacionales (Hashem et al., 2014).

Datos semi-estructurados: Es información procesada y con un formato definido pero no estructurado. De esta manera se puede tener la información definida pero con una estructura variable. Dos ejemplos son las bases de datos basadas en columnas y los ficheros con información en un lenguaje de etiquetas HTML o XML (Hashem et al., 2014).

No estructurados: Los datos no estructurados, tales como mensajes de texto, información de ubicación, videos, directorios de logs y datos de medios sociales son buenos ejemplos de datos no estructurados que no siguen un formato específico. (Hashem et al., 2014).

1.3 Fases de *big data.*

1. Recolección de Datos (Serrat Morros 2013)

   En esta etapa el sistema debe conectarse a sus fuentes de información y extraer la información. Las herramientas de recolección de datos pueden dividirse en dos grupos, dependiendo de cómo se conecten al origen de los datos:

   - *Batch o por lotes:* se conectan de manera periódica a la fuente de datos buscando nueva información. Generalmente se usan para conectarse a sistemas de ficheros o bases de datos, buscando cambios desde la última vez que se conectaron. Una herramienta para migrar datos periódicamente -una vez al día, por ejemplo- de una base de datos a otra es un ejemplo de recolección de datos por lotes.
   - *Streaming o por transmisión en tiempo real:* están conectados de manera continua a la fuente de datos, descargando información cada vez que ésta transmite. Se acostumbra a usar para monitorización de sistemas -para aumentar la seguridad y la detección de fallos, de conjuntos de sensores o para conectarse a redes sociales y descargar información en tiempo real.

   Actualmente las herramientas han evolucionado de manera que muchas de ellas ya pueden usarse de las dos maneras, tanto como para descargarse información en flujos como con procesos batch. En esta etapa, los datos pueden sufrir algún tipo de proceso o cambio si la aplicación así lo requiere, por ejemplo el filtrado de información no deseada o el formateo con el que se guarda finalmente en el sistema de almacenamiento.

2. Almacenamiento

   La capa de almacenamiento tiene, a grandes rasgos, dos elementos básicos: el sistema de ficheros y la base de datos. Hasta hace poco los sistemas de tratamiento de la información se centraban principalmente en las bases de datos pero, debido a que en los sistemas *big data* se busca la mayor variedad posible las bases de datos acostumbran a ser poco flexibles, los sistemas de ficheros han cobrado mayor importancia (Serrat Morros 2013).

3. Procesamiento y análisis

   Una vez se tienen los datos almacenados, el siguiente paso en un sistema *big data* es explotar la información para llegar a los resultados deseados. Las herramientas de análisis y procesamiento de información han evolucionado considerablemente, especialmente aquellas que trabajan sobre datos no estructurados. La necesidad de crear nuevas aplicaciones y que éstas ya estén adaptadas a los sistemas de almacenamiento más recientes, los sistemas distribuidos y las bases de datos (NoSQL) ha promovido la aparición de nuevos paradigmas (Serrat Morros 2013).

4. Visualización

   El apartado de visualización es el que menos ha cambiado respecto a las arquitecturas más tradicionales. Como se ha comentado en la fase de almacenamiento, los resultados a visualizar del procesamiento se acostumbran a consultar sobre bases de datos relacionales o SQL, ya que son las que ofrecen un menor tiempo de respuesta (Serrat Morros 2013).

1.3.1 Arquitectura *big data.*

La arquitectura de *big data* está compuesta generalmente por las cinco fases explicadas en el acápite 1.3 (recolección de datos, almacenamiento, procesamiento, análisis y visualización). Esta arquitectura no es nueva, sino que ya es algo generalizado en soluciones de inteligencia de negocios que existen hoy en día. Sin embargo, debido a las nuevas necesidades cada uno de estos pasos ha ido adaptándose y aportando nuevas tecnologías y oportunidades(Serrat Morros 2013).

En la Figura 2 se muestra el flujo que la información sigue en una arquitectura big data, con diversos orígenes de datos como bases de datos, documentos o datos recibidos en flujos que se reciben y almacenan a través de la capa de recolección de datos, con herramientas específicamente desarrolladas para tal función. Los datos recibidos pueden procesarse, analizarse y/o visualizarse tantas veces como haga falta y lo requiera el caso de uso en el cual se esté trabajando.

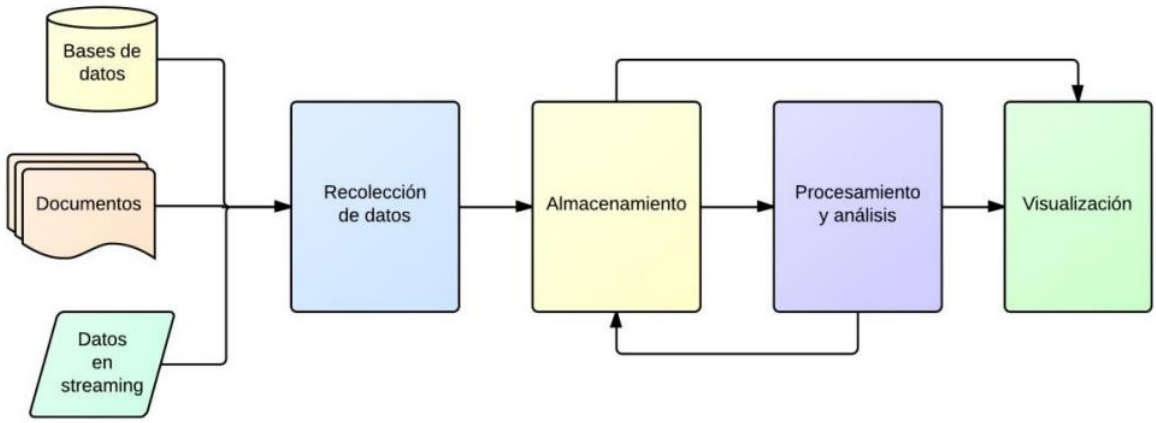


Figura 2- Arquitectura de *big data* por capas. Fuente[(Serrat Morros 2013)]

1.4 Tecnologías *big data.*

Los grandes volúmenes de datos, generados por miles de millones de personas y por miles de millones de dispositivos, han encontrado en las tecnologías *big data* herramientas capaces de procesarlos e indagar sobre ellos para extraer información útil y conocimiento no evidente.

Las dificultades más habituales que se encontraban ante grandes cantidades de datos se centraban en capturar, almacenar, buscar, analizar, compartir y visualizar la información. Las tecnologías *big data* se centran principalmente en proporcionar paliativos para estas dificultades, y para ello se apoyan en tres pilares:

- Sistemas de archivos distribuidos: Su objetivo principal es ofrecer alto rendimiento, escalabilidad y tolerancia a fallos para trabajar con infinidad de ficheros de manera simultánea. Algunos de estos sistemas son: GFS (Google File System), HDFS (Hadoop File System), Amazon S3, IBM GPFS (General Parallel File System)… etc.
- Bases de datos escalables: Su objetivo es almacenar y procesar grandes volúmenes de datos con fiabilidad y bajos tiempos de respuesta. Como se explicara posteriormente las bases de datos NoSQL basadas en clave-valor o en familia de

columnas han surgido especialmente para ello: Cassandra (Apache), Bigtable (Google), HBase (Hadoop)… etc.

- Software de tratamiento masivo: Su objetivo es conseguir repartir las necesidades computacionales para ejecutar un programa (realizar un cálculo… etc.) entre diversos servidores. En este tipo de tarea sobresale MapReduce en el acápite 1.4.2 se explica con más detalle.

1.4.1 Bases de datos NoSQL.

Las Bases de datos NoSQL existen como alternativa (o complemento) a los sistemas de administración de bases de datos relacionales basadas en tablas (RDBMS). A diferencia de los RDBMS, las bases de datos NoSQL no están basadas principalmente en tablas. Esa característica las torna menos eficientes que las RDBMS por la funcionalidad que depende de las relaciones entre elementos de datos, pero más agilizadas para manejar grandes cantidad de datos donde esas relaciones no tengan una importancia central. Mientras que los datos estructurados pueden almacenarse en bases de datos NoSQL, estos sistemas son especialmente aptos para manejar datos no estructurados y, en particular, para proporcionar alta escalabilidad y desempeño para recuperar e incorporar esos datos en grandes cantidades(Hadoop, 2013).

En la

Tabla *1* se muestran ejemplos de este tipo de base de datos NoSQL.

Tabla 1 –Bases de Datos NoSQL

Base de Dato NoSQL	Descripción	Referencia
Hbase	Base de datos NoSQL utilizado por el ecosistema Hadoop, basado en el modelo binario y se ejecuta sobre HDFS	(Bahrami and Singhal, 2015)

Cassandra	Base de datos NoSQL para el almacenamiento de datos distribuidos. Su objetivo principal es la escalabilidad lineal y la disponibilidad. Basada en un modelo de almacenamiento par clave-valor.	(Serrat Morros 2013)
MongoDB	Un almacén de datos de código abierto escrito en C++, proporciona un mecanismo para la consulta de documento. Soporta consultas dinámicas con uso automático de índices.	(Cattell, 2011)

1.4.2 Ecosistema Hadoop.

Hadoop aparece como una solución al problema planteado por el *big data*, una forma de tratar volúmenes masivos de datos de forma eficiente y rápida, capaz de almacenar toda clase de datos: estructurados, no estructurados, semi-estructurados, archivos de registro, imágenes, video, audio etc. (Sánchez González et al., 2014).

El proyecto Hadoop está compuesto básicamente sobre tres piezas: Hadoop Distributed File System (HDFS), Hadoop MapReduce y Hadoop Common.

Las principales características que presenta Hadoop son (Hadoop, 2013):

- *Tolerancia a fallos:* En clústeres de computadoras siempre existe la posibilidad de que alguno de los nodos esté caído. Por lo tanto es necesario que el sistema de ficheros sea capaz de recuperarse. Por ello uno de las características que utilizaHDFS con Hadoop es la replicación.
- *Acceso a datos en flujos.* Hadoop está pensado para consumir un alto volumen de datos. Por tanto HDFS debe proporcionar un gran ancho de banda para que Hadoop pueda procesar toda esa información. Hay que tener en cuenta que para Hadoop es más importante este ancho de banda que la latencia de acceso a los datos ya que se trabaja con procesos batch y no en tiempo real.
- *Modelo de coherencia simple:* En Hadoop los datos de entrada van a ser escritos una vez y van a ser leídos tantas veces como sea necesario. Por tanto, los datos almacenados en HDFS también siguen este modelo.
- *Mover la programación es más barato que mover datos:* En Hadoop se sigue la idea de que es más barato mover las aplicaciones o los algoritmos donde están los datos que mover estos mismos. Esta idea se cumple ya que se habla de colecciones de datos muy grandes por lo que sería muy costoso moverlos. HDFS provee de interfaces especializadas en este sentido.
- *Portabilidad entre hardware y software heterogéneo:* HDFS es portable entre distintas plataformas.

HDFS

De acuerdo con (Borthakur, 2007), el sistema de ficheros distribuido de Hadoop (HDFS) tiene una arquitectura maestro-esclavo, ver Figura 3. Un clúster HDFS consta de un solo *NameNode* y un servidor maestro, que administra el espacio de nombres del sistema de archivo y regula el acceso a los archivos por parte de los clientes. Además, hay un número de *DataNodes*, generalmente uno por cada nodo del clúster, que administra el almacenamiento de información conectado a los nodos sobre los que corren. HDFS expone un espacio de nombres del sistema de archivo y permite que los datos de usuario se almacenen en archivos.

Internamente, un archivo se divide en uno o más bloques y estos bloques se almacenan en un conjunto de *DataNodes*. El *NameNode* ejecuta operaciones de espacio de nombres del sistema de archivos como: abrir, cerrar, renombrar archivos y directorios. También determina la asignación de bloques para *DataNodes*. Los *DataNodes* son responsables de servir las peticiones para leer y escribir desde el archivo de clientes del sistema.

NameNode y DataNode son piezas de software diseñadas para ejecutarse en máquinas de productos básicos. Estas máquinas suelen correr un sistema operativo GNU/Linux. HDFS está construido utilizando el lenguaje Java y cualquier equipo que admita Java puede ejecutar el *NameNode* o el software *DataNode*.

La existencia de un único *NameNode* en un clúster simplifica en gran medida la arquitectura del sistema. El *NameNode* es el árbitro y el repositorio de todos los metadatos HDFS. El sistema está diseñado de tal manera que datos del usuario nunca fluyen a través de *NameNode.*

Según *(*Borthakur, *2013),* las principales características de HDFS son:

- *Sistema de ficheros amigable:* El esquema de HDFS está diseñado para que se parezca en la mayor medida posible a los sistemas de ficheros conocidos, especialmente al de Unix. Desde el espacio de nombres hasta los permisos de los ficheros y la seguridad.
- *Tolerancia a fallos de hardware:* Las instancias de HDFS es poder llegar a tener hasta miles de máquinas trabajando como servidores, almacenando en cada una de ellas una parte del sistema de ficheros. Con tal cantidad de componentes formando un sistema, un fallo de hardware que implique la caída o desconexión de uno o más de estos componentes es la regla, no la excepción.
- *Acceso en streaming:* En las aplicaciones para las que está pensado HDFS, el usuario necesita acceder a los datos con un rendimiento constante y elevado.
- *Grandes cantidades de datos*: No solo en cuanto a la capacidad total del sistema de ficheros sino también al volumen individual de los ficheros que lo componen. Un

tamaño normal y aconsejable para un fichero de HDFS puede ir de los gigabytes a los terabytes.

- *Modelo simple y coherente:* HDFS está pensado para aplicaciones que necesiten escribir el fichero una sola vez y que, una vez cerrado, no necesite cambios. De esta manera se puede conservar la coherencia de los datos y habilita su acceso rápido.
- *Portabilidad:* El sistema ha de ser portable a una gran cantidad de plataformas, tanto de hardware como de software.
- *Escalabilidad simple*: HDFS también permite la fácil expansión del sistema en caliente, pudiendo añadir nuevos nodos sin tener que pausar o parar los procesos que hay en ejecución en el clúster y sin tener que configurarlo; es decir, que el propio sistema se encarga de determinar que bloques de ficheros almacenará y que trabajos realizará.

Funcionamiento

El funcionamiento (ver Figura 3) consiste en que cada nodo en una instancia *Hadoop* típicamente tiene un único nodo de datos; un clúster de datos forma el clúster *HDFS*. La situación es típica porque cada nodo no requiere un nodo de datos para estar presente.

Cada nodo sirve bloques de datos sobre la red usando un protocolo de bloqueo específico para *HDFS*. El sistema de archivos usa la capa *TCP/IP* para la comunicación; los clientes usan *RPC* para comunicarse entre ellos. El HDFS almacena archivos grandes (el tamaño ideal de archivo es de 64 MB), a través de múltiples máquinas.

Consigue fiabilidad mediante la réplica de datos a través de múltiples *hosts*, y no requiere almacenamiento mediante un conjunto redundante de discos independientes (RAID) en ellos. Con el valor de replicación por defecto los datos se almacenan en tres nodos: dos en el mismo *rack*, y otro en un *rack* distinto.

Los nodos de datos pueden hablar entre ellos para reequilibrar datos, mover copias, y conservar alta la replicación de datos. *HDFS* no cumple totalmente con *POSIX* porque los requerimientos de un sistema de archivos *POSIX* difieren de los objetivos de una aplicación *Hadoop*, porque el objetivo no es tanto cumplir los estándares *POSIX* sino la

máxima eficacia y rendimiento de datos. *HDFS* fue diseñado para gestionar archivos muy grandes. *HDFS* no proporciona alta disponibilidad.

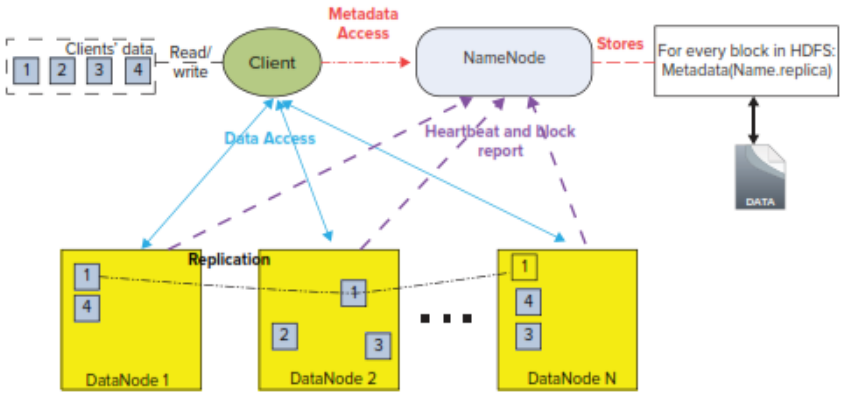


Figura 3- Arquitectura HDFS. Fuente [(Boris Lublinsky, 2013)]

MapReduce

MapReduce es un nuevo modelo de programación diseñado para dar soporte a la computación paralela sobre grandes conjuntos de datos repartidos entre varias computadoras. MapReduce está inspirado en los nombres de dos importantes funciones de la programación funcional: Map y *Reduce*. Las funciones Map y Reduce están definidas ambas con respecto a datos estructurados en tuplas del tipo (clave, valor) (Sánchez González et al., 2014).

La función Map toma uno de estos pares de datos con un tipo en un dominio de datos y devuelve una lista de pares en un dominio diferente: Map (k1, v1) -> list (K2, v2)

Se encarga del mapeo y es aplicada en paralelo para cada elemento de la entrada. Esto produce una lista de pares (k2, v2) por cada llamada. Después de ese primer paso junta todos los pares con la misma clave de todas las listas y los agrupa, creando un grupo para cada una de las diferentes claves generadas.

La función *Reduce* es aplicada en paralelo para cada grupo, produciendo una colección de valores para cada dominio: Reduce (K2, list (v2)) -> list (v3)

Cada llamada a *Reduce* produce un valor v3 o una llamada vacía, aunque una llamada puede retornar más de un valor.

Por tanto, MapReduce transforma una lista de pares (clave, valor) en una lista de valores. Este comportamiento es diferente de la combinación "*map and reduce*" de programación funcional, que acepta una lista arbitraria de valores y devuelve un valor único que combina todos los valores devueltos por *map*.

Ejemplo MapReduce: Conteo de palabras

El pseudocódigo siguiente muestra la estructura básica de un programa de MapReduce que cuenta el número de ocurrencias de cada palabra en una colección de documentos.

Función Map(): Cuenta cada palabra que aparece en el documento y asocia el valor 1 a cada palabra encontrada.

```
map(String name, String document):
        // clave: nombre del documento
        // valor: contenido del documento
        for each word w in document:
                EmitIntermediate(w, 1);
```

Figura 4- Pseudocódigo. Función map.Fuente[*(Dean and Ghemawat, 2008)*]

Función Reduce (): Busca las apariciones de cada palabra en c*uentasParciales* y suma todos los valores asociados en la variable *result*.

```
reduce(String word, Iterator cuentasParciales):
        // word: una palabra
        // cuentasParciales: una lista parcial para cuentas agregadas
        int result = 0;
        for each v in cuentasParciales:
                result += ParseInt(v);
        Emit(result);
```

Figura 5- Pseudocódigo. Función reduce.Fuente[(Dean and Ghemawat, 2008)]

La Figura *6* refleja el comportamiento de MapReduce desde que el programa cliente crea la petición de trabajo hasta que se generan los archivos de salida.

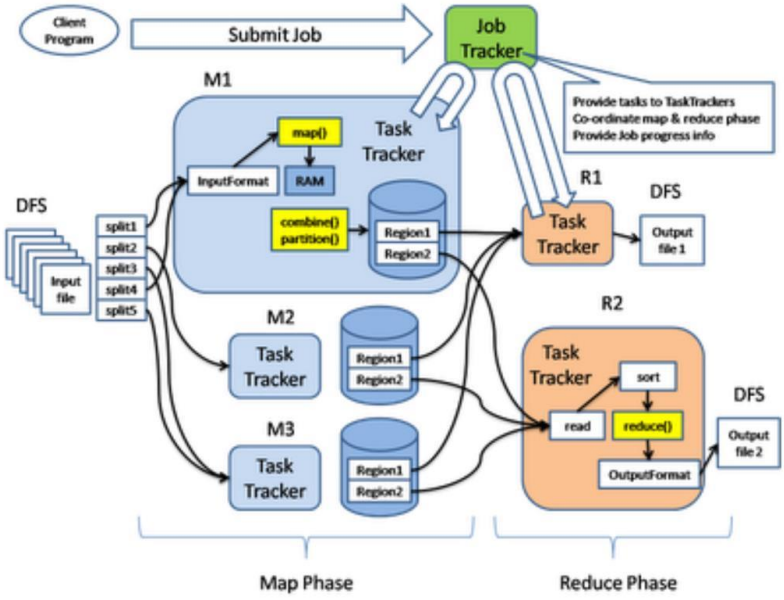


Figura 6- Flujo completo MapReduce. Fuente [(Sánchez González et al., 2014)]

Función Combinar

Combinar es un paso de procesamiento opcional que se puede utilizar para optimizar la ejecución del trabajo MapReduce. Una instancia de la clase combinador se ejecuta en cada tarea mapa y en algunas tareas de reducir (Boris Lublinsky, 2013).

1.4.2.1 Hadoop Common.

En este apartado se exponen los principales proyectos y herramientas que conforman el ecosistema Hadoop. Los cuales se han agrupado según el uso que tienen de acuerdo a la arquitectura *big data* descrita en el epígrafe Arquitectura *big data*.

- *Avro:* sistema para la serialización de datos y uno de los principales métodos para transportar información entre las herramientas y aplicaciones Hadoop (Serrat Morros 2013) .
- *ZooKeeper:* servicio centralizado que se encarga de administrar y gestionar la coordinación entre procesos en sistemas distribuidos. El objetivo de Apache con ZooKeeper es el de librar a los desarrolladores de la tarea de implementar funciones de mantenimiento entre sus procesos, como la sincronización entre estos y ofrecer alta disponibilidad a través de servicios redundantes. ZooKeeper permite la coordinación de procesos a través de un espacio de nombres compartido de estructura jerárquica, parecido a un sistema de ficheros (Serrat Morros 2013).
- *Apache Solr:* motor de búsqueda basado en Apache Lucene, escrito en Java y que facilita a los programadores el desarrollo de aplicaciones de búsqueda. Lucene ofrece indexación de información, tecnologías para la búsqueda así como corrección ortográfica, resaltado y análisis de información, entre otras características (Serrat Morros 2013).

Las herramientas descritas no se pueden clasificar en ninguna de las fases de una arquitectura *big data*. Sin embargo son proyectos esenciales ya que ofrecen soporte y funcionalidades a los desarrolladores de aplicaciones Hadoop de cualquiera de las etapas definidas y están pensadas fundamentalmente para la comunicación y sincronización de procesos.

Recolección de datos

- *Chukwa:* sistema de recolección de datos para la gestión de grandes sistemas distribuidos, herramienta principalmente pensada para trabajar sobre archivos logs y realizar análisis. Está construido a un nivel superior de HDFS y MapReduce, por lo que hereda su escalabilidad y robustez (Serrat Morros 2013).
- *Flume:* herramienta distribuida para la recolección, agregación y transmisión de grandes volúmenes de datos. Ofrece una arquitectura basada en la transmisión de datos por flujo (streaming) altamente flexible y configurable pero a la vez simple (Serrat Morros 2013).
- *Sqoop:* herramienta diseñada para transferir de forma eficiente información entre Hadoop y bases de datos relacionales (White, 2012).

Almacenamiento

- *Cassandra:* base de datos NoSQL distribuida y basada en el modelo de almacenamiento de «clave-valor» (Serrat Morros 2013).
- *HBase:* base de datos NoSQL orientada a columnas, que se ejecuta en HDFS (White, 2012).
- *Hive:* herramienta para trabajar con almacenes de datos que facilita la creación, consulta y administración de grandes volúmenes de datos distribuidos en forma de tablas relacionales. Cuenta con un lenguaje derivado de SQL, llamado HiveQL, que permite realizar las consultas sobre los datos almacenados (Serrat Morros 2013).
- *MongoDB:* sistema gestor de base de datos orientado a documentos. MongoDB guarda los documentos en el formato BSON, que no es más que una implementación binaria del conocido JSON.(Bahrami and Singhal, 2015).

Procesamiento y análisis

- *Oozie:* planificador de flujos de trabajo (*workflows*) para sistemas que realizan trabajos o procesos Hadoop. Proporciona una interfaz de alto nivel para el usuario no técnico o no experto y que gracias a su abstracción permite a estos usuarios realizar flujos de trabajo complejos (Serrat Morros 2013).

- *Pig:* Es una herramienta para analizar grandes volúmenes de datos mediante el lenguaje de alto nivel "PigLatin*"* que está diseñado para la paralelización del trabajo. Mediante un compilador se traducen las sentencias en PigLatin a trabajos MapReduce sin que el usuario tenga que pasar a programar ni tener conocimientos sobre ellos (Serrat Morros 2013).
- *Mahout: E*s una librería Java que contiene básicamente funciones de aprendizaje y que está construida sobre MapReduce. Al usar MapReduce está pensada para trabajar con grandes volúmenes de datos y en sistemas distribuidos aunque también está diseñado para funcionar en otros sistemas que no utilizan Hadoop, ni son distribuidos (Serrat Morros 2013).

En la Figura *7* se muestra la estructura del ecosistema Hadoop.

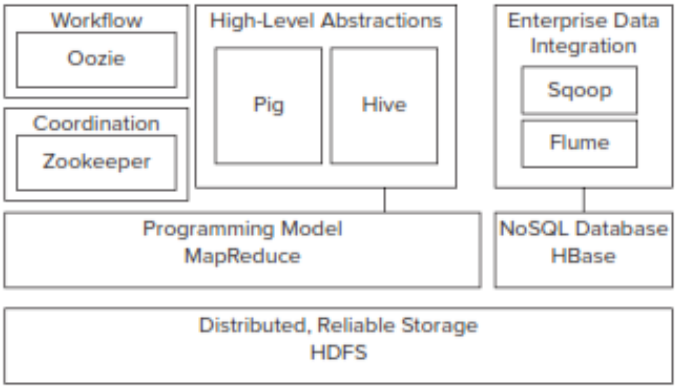


Figura 7- Ambiente Hadoop. Fuente [(Boris Lublinsky, 2013)]

1.5 Pentaho Data Integration.

Pentaho es una suite de herramientas de Inteligencia de Negocios de código abierto. Todas las herramientas que pertenecen a Pentaho están programadas en código 100% Java. Dentro de las que destaca un producto para la integración de datos llamado PDI. El cual utiliza un enfoque innovador y tiene una fuerte interfaz gráfica muy fácil de usar para el usuario. La compañía comenzó en torno a 2001 y no fue hasta 2002 cuando Kettle se integró a ella(López Burgos, 2013).

1.5.1 Características y Ventajas.

Pentaho tiene dos versiones: una comercial y otra de código abierto. En el presente trabajo se utiliza la versión de código abierto, la cual ofrece una amplia gama de herramientas orientadas a la integración de información y al análisis inteligente de los datos de su organización. Cuenta con potentes capacidades para la gestión de procesos ETL, informes interactivos, análisis multidimensionales de información OLAP y minería de datos. Todos estos servicios están integrados en una plataforma Web, en la que el usuario puede consultar de una manera fácil e intuitiva (López Burgos, 2013).

Los módulos incluidos por Pentaho pueden utilizarse de manera conjunta o de forma separada según las necesidades de cada organización(López Burgos, 2013).

Según(López Burgos, 2013), algunas de sus características más significativas son

- Entorno gráfico de desarrollo.
- Uso de tecnologías estándar: Java, XML, JavaScript.
- Fácil de instalar y configurar.
- Basado en dos tipos de objetos: transformaciones (colección de pasos en un proceso ETL) y trabajos (orquestación de transformaciones).
- Permite rápida y eficientemente extraer datos, transformarlos, limpiarlos, validarlos, cargarlos, etc. desde donde se encuentren.
- Librería de transformaciones completa con más de 100 objetos de mapeo.
- Código 100% Java, multiplataforma y soporte de una amplia cantidad de fuentes de datos, incluyendo aplicaciones integradas, sobre 30 plataformas propietarias y de código abierto, archivos planos, documentos Excel, y más.
- Soporte avanzado de Almacenes de Datos para cambios lentos y dimensiones basura.
- Herramienta gráfica de muy fácil uso (control lógico de flujo).
- Basado en repositorio, lo cual facilita el uso de componentes de transformación, colaboración y administración de modelos, conexiones, logs, etc.

- Rendimiento y escalabilidad de clase empresarial con soporte a procesamiento masivo paralelo (MPP, por sus siglas en inglés) a través de ejecución en clúster.
- Monitoreo y depurador integrado.
- Calendario programador de transformaciones y trabajos (Scheduler)
- Incluye cuatro herramientas:
    - SPOON: para diseñar trasformaciones y trabajos usando un entorno gráfico.
    - PAN: para ejecutar transformaciones diseñadas con Spoon desde la línea de comandos.
    - KITCHEN: Para ejecutar trabajos diseñados con Spoon desde la línea de comandos.
    - CARTE: para ejecutar trabajos y transformaciones en un servidor Web de forma remota.

1.5.1.1 Transformaciones.

Una transformación se compone de pasos, que están enlazados entre sí a través de los saltos. Es el elemento básico de diseño de los procesos ETL en PDI. Los pasos son el elemento más pequeño dentro de las transformaciones. Los saltos constituyen el elemento a través del cual fluye la información entre los diferentes pasos (siempre es la salida de un paso y la entrada de otro). Como se muestra en la Figura *8*, la transformación es una entidad hecha de pasos unidos por saltos. Estos pasos y saltos construyen caminos por los que fluyen los datos. Los datos entran o se crean en un paso, el paso aplica algún tipo de transformación a ella y finalmente deja paso a los datos. Por tanto, se dice que una transformación es orientada al flujo de datos (López Burgos, 2013).

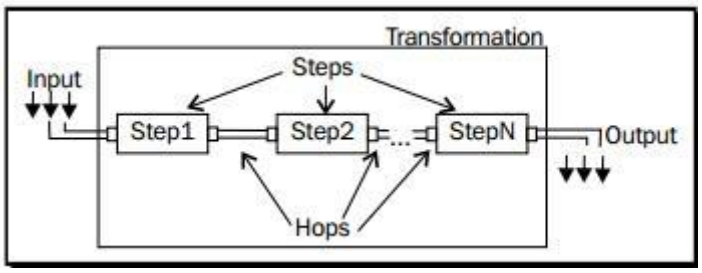


Figura 8- Estructura de una transformación. Fuente [(López Burgos, 2013)]

1.5.1.2 Saltos.

Según (López Burgos, 2013), los saltos son los componentes conectores que indican el orden de ejecución de cada paso (no empezando la ejecución del elemento siguiente hasta que el anterior no haya concluido). A través de ellos viajan los metadatos en forma de registros y tablas, como se observa en la Figura *9*.

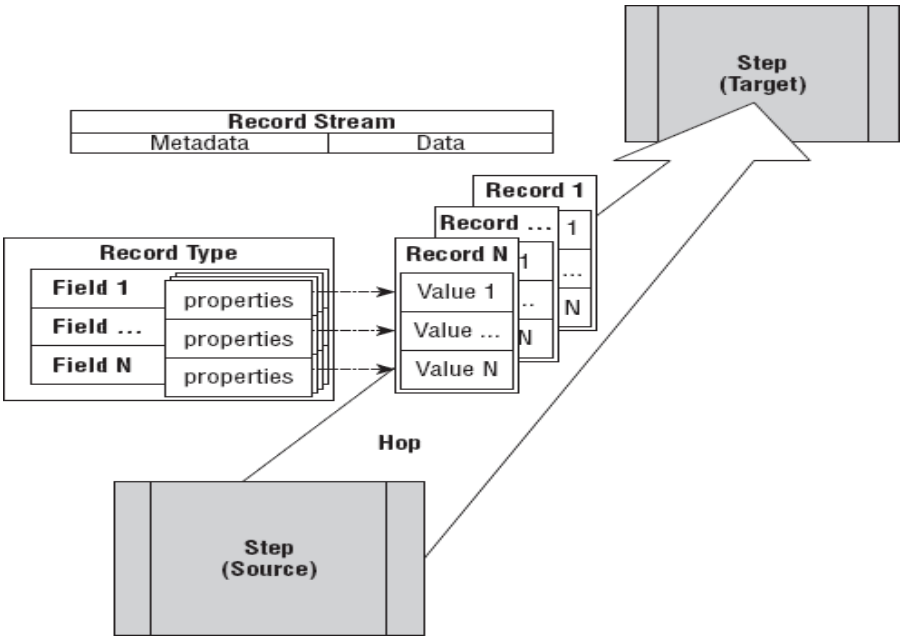


Figura 9- Estructura de los saltos. Fuente [(Roland Bouman, 2009)]

1.5.1.3 Trabajos.

Un trabajo (job) es un conjunto sencillo o complejo de tareas con el objetivo de realizar una acción determinada. En los trabajos se puede utilizar pasos específicos (que son diferentes a los disponibles en las transformaciones) como recibir un fichero vía ftp, mandar un email, ejecutar un comando, entre otros. Además, se pueden ejecutar una o varias transformaciones diseñadas previamente y orquestar una secuencia de ejecución de ellas (López Burgos, 2013).

1.5.1.4 Pasos.

Los *pasos* están agrupados por categorías y cada uno está diseñado para cumplir una función determinada. Cada paso tiene una ventana de configuración específica, donde se determinan los elementos a tratar y su forma de comportamiento.

Características generales de los pasos(Casters et al., 2010):

- Un paso es un elemento central en la transformación.
- Se representa gráficamente en forma de ícono.
- Un paso tiene que tener un nombre único en una misma transformación.
- Virtualmente, cada paso es capaz de leer y escribir filas de datos (la única excepción es el paso Generar filas, que sólo escribe los datos).
- Cuando se ejecuta una transformación, se inician una o más copias de cada paso, cada uno se ejecuta en su propio hilo. Durante la carrera, todas los pasos copias se ejecutan de forma simultánea, con filas de datos que constantemente fluyen a través de los saltos de conexión.
- Más allá de estas capacidades estándar, cada paso, tiene una funcionalidad distinta que está representado por el tipo de paso.

Existen pasos de transformaciones y otros de trabajos, estos se clasifican en categorías para hacer más fácil su manejo dentro de la herramienta.

1.5.2 Arquitectura de PDI.

En la Figura 10 se muestra la estructura interna del PDI, en la cual se puede apreciar cómo se realizan los procesos dentro de la herramienta.

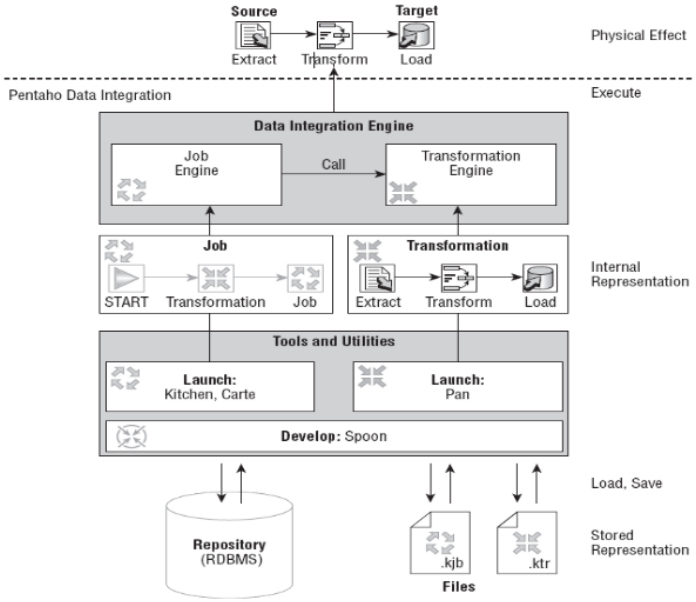


Figura 10- Estructura interna del PDI. Fuente [(Roland Bouman, 2009)]

Flujo de registros

A continuación se argumenta cómo funciona la estructura interna del PDI de abajo hacia arriba como se muestra en la Figura *10* (López Burgos, 2013):

- La representación almacenada de los repositorios desde los cuales se guardan y se cargan los archivos con extensión .kjb pertenecientes a los trabajos y los .ktr referentes a las transformaciones.
- Las herramientas y utilidades de desarrollo (Spoon) y de ejecución (Kitchen, Pan y Carte).
- La representación interna de cómo se realizan los trabajos (se hace un llamado para ejecutar una transformación) y en las transformaciones se extraen los datos desde varias fuentes, se procesan a través de diversos pasos y finalmente se exportan hacia la salida prevista que puede ser una tabla en una base de datos, un fichero de texto plano, una hoja de cálculo Excel o salidas *big data* que soporta el PDI como Hadoop file output, HBase outpat, MapReduce output, MongoDB output entre otras.

- Muestra en la parte de ejecución la ingeniería para la integración de datos exponiendo como para realizar un trabajo se realiza una llamada a la transformación pertinente. Ejemplifica a efectos físicos lo que se muestra en el Spoon como herramienta gráfica dentro del software.

1.5.3 Procesos de Extracción Transformación y Carga.

Extracción, Transformación y Carga (ETL, por sus siglas en inglés): extrae los datos desde diversas fuentes, los transforma y limpia antes de cargarlos en una base de datos, mercado o almacén de datos (Sybven, 2013).

Extracción

Extracción de datos es la selección sistemática de datos operacionales usados para poblar el componente de almacenamiento físico del almacén de datos (Nader, 2003). El paso comprende la extracción de datos desde el sistema de origen y la hace accesible para su posterior procesamiento. El objetivo principal del paso de extracción es recuperar todos los datos necesarios desde el sistema de origen como sea posible. Debe ser diseñado de manera que no afecte negativamente el sistema de origen en términos de rendimiento.

Existen diversas variantes para realizar el proceso (Javlin, 2011):

Notificación de actualizaciones: Si el sistema de origen es capaz de proporcionar una notificación de que un registro ha cambiado y describir el cambio, esta es la forma más sencilla de obtener los datos.

- Extracto incremental: algunos sistemas pueden no ser capaces de proporcionar una notificación de que una actualización se ha producido, pero son capaces de identificar los registros que han sido modificados y proporcionar un extracto de dichos registros.
- Extracto completo: algunos sistemas no son capaces de identificar qué datos se han cambiado en absoluto, por lo que un extracto completo es la única forma de poder obtener los datos del sistema. El extracto completo requiere mantener una copia del

último extracto en el mismo formato con el fin de ser capaz de identificar los cambios. El extracto completo maneja supresiones también.

Al utilizar extractos incrementales o completos, la frecuencia de retorno es extremadamente importante. Particularmente para los extractos completos, los volúmenes de datos pueden ser de decenas de gigabytes.

Transformación

Transformación de datos es el proceso para realizar otros cambios en los datos operacionales para reunir los objetivos de orientación a temas e integración principalmente (Nader, 2003). En la etapa de transformación se aplican un conjunto de reglas para transformar los datos desde el origen al destino. Esto incluye convertir los datos a una manera compatible y estándar usando las mismas unidades para que más tarde puedan ser acoplados. La etapa de transformación también requiere la combinación de datos de varias fuentes, generar agregados, la generación de claves sustitutas, clasificación, derivar nuevos valores calculados y la aplicación de reglas avanzadas de validación. Además en esta etapa se debe tener en cuenta la limpieza, calidad y conformación de los datos para lograr transformaciones eficaces.

Limpieza

La limpieza es la corrección en los datos de posibles errores, por ejemplo: datos incompletos, duplicados, formatos inconsistentes en cuanto a descripción, abreviaturas y unidades de medidas, falta de datos de entrada o datos que violen las restricciones de integridad del sistema. La etapa de limpieza es una de las más importantes, ya que garantiza la calidad de los datos en el almacén de datos (López Burgos, 2013).

En esta etapa se deben corregir las anomalías que se detecten en el proceso de la unificación de datos, a continuación se exponen algunas de las anomalías más frecuentes (López Burgos, 2013):

- No estandarización de valores

  - Esta anomalía consiste en la existencia de uno o varios campos que poseen datos escritos con formato diferente pero que significan lo mismo, o sea datos que no siguen un estándar o norma predeterminada.
  - Por ejemplo el caso típico de: (categorías de sexo Masculino / Femenino / Desconocido, M / F / null, hombre / mujer / no disponible, se convierten a la norma Masculino / Femenino / Desconocido).
- Existencia de valores nulos.
  - Como su nombre lo indica en las fuentes se encuentran valores nulos o vacíos, lo cual atenta contra un adecuado análisis de la información. La solución que se brinda es remplazarlos por un valor normalizado o sea una constante.
- Esquemas no integrados.
  - Esta anomalía se pone de manifiesto cuando se trata de integrar varias fuentes de datos y en una de ellas aparece el nombre de un campo escrito de una forma y en la otra fuente aparece escrito con otro nombre, pero los valores que tienen ambas fuentes son equivalentes. Por ejemplo: una columna denominada raza y otra llamada color de piel y en ambos casos los valores son: blanca, negra y mulata. La solución que se brinda es integrarlas a ambas en una sola columna bajo un mismo nombre.
- Existencia de duplicados.
  - Como su nombre lo indica es cuando se está en presencia de dos o más filas donde coinciden todos o casi todos los valores de sus campos. La solución que se ofrece es eliminar los valores duplicados o repetidos.

1.5.3.1 Conformación.

Cuando la información se encuentra limpia y con una calidad adecuada, esta es unificada, conformada y normalizada. Los indicadores son calculados de una forma racional, lo mismo que los atributos de las dimensiones, para que estén unificados y en todos los sitios donde aparezcan tengan la misma estructura y el mismo significado (Javlin, 2011).

Carga

El proceso de carga es la inserción sistemática de datos en el componente de almacenamiento físico del almacén de datos (Nader, 2003), o en cualquier sistema destino. Dependiendo de los requerimientos de la organización, este proceso puede abarcar una amplia variedad de acciones diferentes. En algunas bases de datos se sobrescribe la información antigua con nuevos datos; los almacenes de datos por su parte mantienen un historial de los registros de manera que se pueda hacer una auditoría de los mismos y disponer de un rastro de toda la historia de un valor a lo largo del tiempo. La integridad referencial es necesaria para asegurar la consistencia mantenida por la herramienta de ETL.

1.6 **Conclusiones Parciales.**

En este capítulo se citaron diversos conceptos de *big data* sugeridos por distintos autores. Se expusieron las principales características o dimensiones *big data* planteadas por diferentes autores: *volumen, velocidad, variedad, veracidad, valor, variabilidad y, visualización. S*e clasificaron los *big data* en base a cinco aspectos: fuente de los datos, tipo de dato, tipo de almacenamiento, organización o preparación de los datos, forma de procesamiento, resaltando la clasificación en cuanto al tipo de dato en estructurados, semi-estructurados y no estructurados. Se brindó una arquitectura donde aprecia el funcionamiento de *big data*. Además, en este capítulo se mostraron las diversas tecnologías que existen para el procesamiento de *big data*. Por último, se describieron las principales características y ventajas de utilizar la herramienta Pentaho Data Integration como solución de inteligencia de negocios.

CAPÍTULO II: CARACTERIZACIÓN DE LOS PASOS DEL PDI RELACIONADOS CON BIG DATA.

En el presente capítulo se describen las tecnologías *big data* que se pueden utilizar a través de la interfaz gráfica del PDI asociadas a los diferentes pasos para la implementación de transformaciones y trabajos. En cada caso se explica y visualiza mediante imágenes la configuración de los pasos *big data*. Finalmente, se diseña una trasformación y un trabajo que hacen uso de algunos de los pasos *big data* que ofrece el PDI.

2.1 Características de los pasos *big data* en una transformación.

Los pasos *big data* en una transformación en su mayoría son destinados para entrada y salida de datos. Entre sus principales características se encuentran:

- Permite la posibilidad de procesar datos de diferentes formatos de contenido (estructurados, semi-estructurados y no estructurados).
- Incorpora entradas para gran variedad de bases de datos NoSQL como MongoDB (orientada a documentos), Cassandra y HBase (bases de datos distribuidas orientadas a columnas), entre otras.

2.1.1 Paso Avro input.

Con Avro se puede almacenar y leer datos fácilmente desde diferentes lenguajes de programación. Está optimizado para minimizar el espacio en disco necesario para los datos. En los proyectos sobre Hadoop, suele haber grandes cuantidades de datos, la serialización se usa para procesarlos y almacenar estos datos, de forma que el rendimiento en tiempo sea efectivo. Esta serialización puede ser en texto en plano, JavaScript Object Notation (JSON) o en formato binario. (Serrat Morros 2013).

Paso Avro input:

La Figura 11 muestra el panel *Source* del paso *Avro* input.

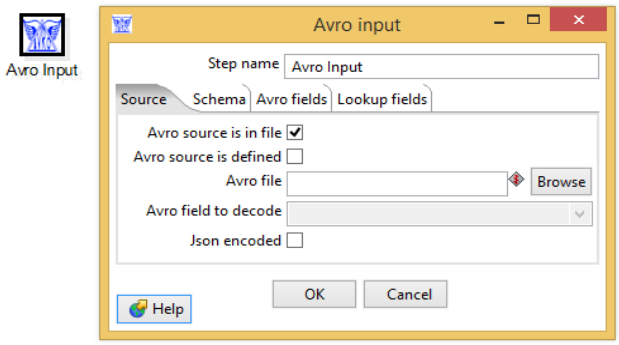


Figura 11- Panel Fuente"Source" del paso Avro input

Las opciones que se observan en el panel Source Figura 11 se describen en la Tabla 2

Opción	Definición
Avro source is in file	Indica que los datos fuentes proceden de un archivo.
Avro source is defined in a field	Indica que los datos fuente vienen desde un campo, se selecciona un campo entrante para descifrar mediante la opción *Avro field to decode from*. En este modo de operación, un archivo de esquema debe ser especificado en la opción *Schema file* del panel *Schema.*
Avro file	Dirección del archivo para descifrar.
Avro field to decode from	Especifica el campo entrante que contiene los datos de Avro para descifrar.
JSON encoded	Indica que los datos de Avro han sido codificados en JSON.

Tabla 2- Descripción del panel "*Source*" del paso *Avro input*. Fuente[(Anónimo, 2012)]

La Figura 12 muestra el panel Schema del paso Avro input.

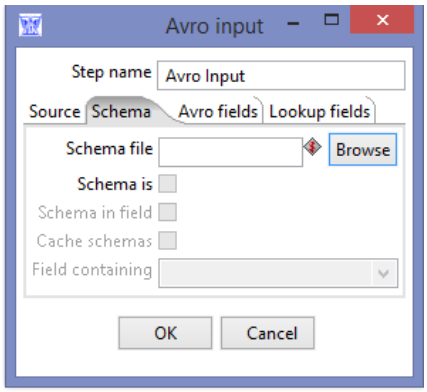


Figura 12- Panel "Schema" del paso Avro input

Las opciones que se observan en el panel *Schema* Figura 12 se describen en la Tabla 3

Opción	Definición
Schema file	Localización del esquema.
Schema is defined in a field	Indica que el esquema se encuentra dentro de un campo. Al seleccionar esta opción se activan las opciones *Schema in field is a path y Cache schemas.*
Schema in field is a path	Indica que el esquema de entrada especifica una ruta de acceso a un archivo de esquema. Si no se controla, el paso se supone que el esquema de entrada es la definición del esquema actual en formato JSON.
Cache schemas in memory	Habilitando esta opción se conserva todos los esquemas cargados anteriormente.
Field containing schema	Indica campo del esquema.

Tabla 3- Descripción del panel *Schema* del paso *Avro input*. Fuente[(Anónimo, 2012)]

La Figura 13 muestra el panel *Avro fields* del paso *Arvro* input.

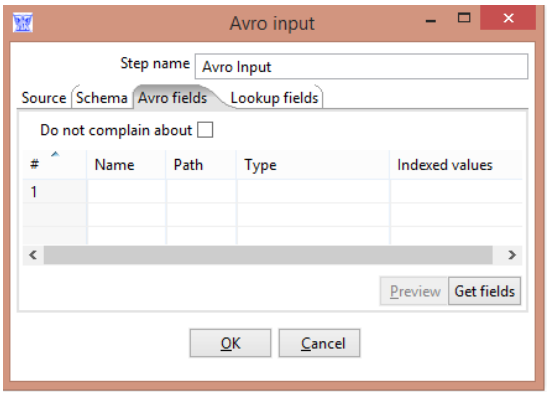


Figura 13- Panel *Avro fields* del paso *Avro Input*

Las opciones que se observan en el panel *Avro fields* Figura 13 se explican en la Tabla 4.

Opción	Definición
Do not complain about fields not present in the schema	Desactiva emitiendo una excepción cuando caminos o campos especificados no están presentes en el esquema de Avro activo. En su lugar se devuelve un valor nulo. O en cambio, el sistema devuelve un valor nulo.
Get fields	Puebla los campos disponibles del origen designado.

Tabla 4- Descripción del panel *Avro fields* del paso Avro input. Fuenta[(Anónimo, 2012)]

La Figura 14 muestra el panel *lookup fields* del paso *Avro* input. Fuente[(Anónimo, 2012)]

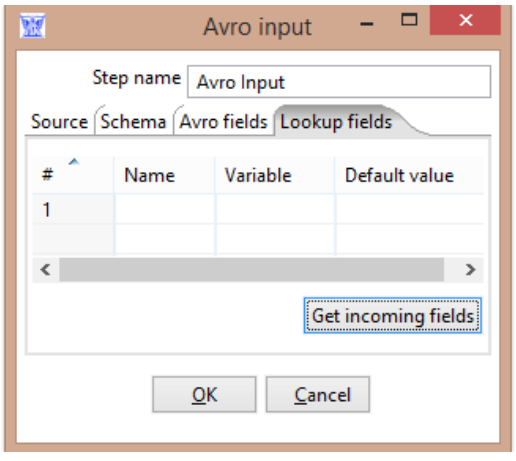


Figura 14- Panel "*Lookup fields*" del paso Avro input

Las opciones que se observan en el panel *Lookup fields* Figura 14 del paso *Avro input* se explican en la Tabla 5.

Opción	Definición
Get incoming fields	Puebla la columna Name con los nombres de los campos entrantes. La columna variable de la tabla permite que se le asigne valores de los campos entrantes a la variable. La columna valor por defecto es utilizada cuando el valor del campo entrante es nulo.

Tabla 5- Descripción del panel *lookup fields* del paso *Avro input.* Fuente[(Anónimo, 2012)]

2.1.2 Paso Cassandra Input.

El objetivo principal de Cassandra es la escalabilidad lineal y la disponibilidad. La arquitectura distribuida de Cassandra está basada en una serie de nodos iguales que se comunican con un protocolo *peer to peer* (P2P) con lo que la redundancia es máxima. Cassandra está desarrollada por Apache Software Foundation. En las versiones iniciales utilizaba un API propia para poder acceder a la base de datos. En los últimos tiempos están apostando por un lenguaje denominado CQL (Cassandra Query Language, no confundir con *Contextual Query Language*) que posee una sintaxis similar a SQL aunque con muchas menos funcionalidades. Esto hace que iniciarse en el uso de la misma sea más sencillo. Además, permite acceder en Java desde JDBC (Serrat Morros 2013).

La Figura 15 muestra el paso Cassandra input.

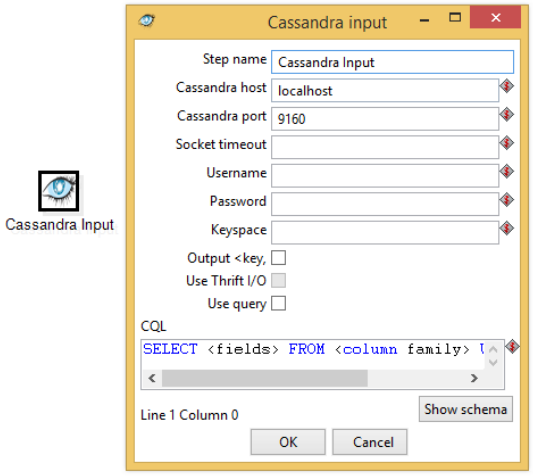


Figura 15- Paso *Cassandra input*

Las opciones que se observan en el paso Cassandra input Figura 15 input se explican en la Tabla 6.

Opción	Definición
Step name	Nombre del paso
Cassandra host	Dirección ip
Keyspace	Base de datos
Use query compression	Si está marcado, le dice al paso donde o no puede comprimir el texto de la consulta CQL antes de enviarlo al servidor.
Show schema	Abre un cuadro de diálogo que muestra los metadatos de la familia columna nombrada en la consulta CQL SELECT.

Tabla 6- Descripción del paso *Cassandra input* Fuente[(Anónimo, 2012)]

CQL:

El cuadro de texto al final permite ingresar una sentencia *Select* en CQL la cual posee la sintaxis que se muestra en la Figura 16:

```
SELECT [FIRST N] [REVERSED] <SELECT EXPR>
FROM <COLUMN FAMILY> [USING <CONSISTENCY>] [WHERE <CLAUSE>] [LIMIT N];
```

Figura 16-Sintaxis de la sentencia *Select* en CQL

2.1.3 Paso Cassandra Output.

El paso Casandra Output es un paso de salida que permite escribir en una familia de columnas de Casandra (tabla) como parte de una transformación (Anónimo, 2012).

La Figura 17 muestra el panel *Conection* del paso *Cassandra Output.*

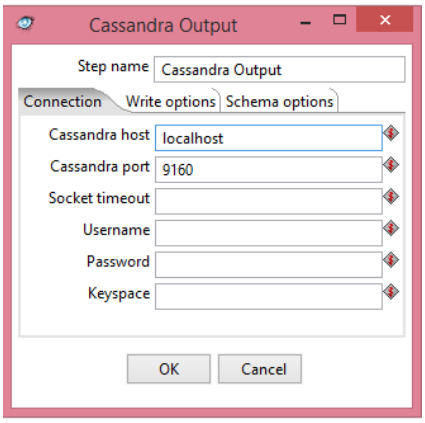


Figura 17- Panel Conexión del paso *Cassandra output*

La configuración del panel conexión del paso *Cassandra Output* es similar a la especificada en la Tabla 6.

La Figura 18 y *Figura 19* muestran el panel *Write option y Schema options* respectivamente del paso *Cassandra output.*

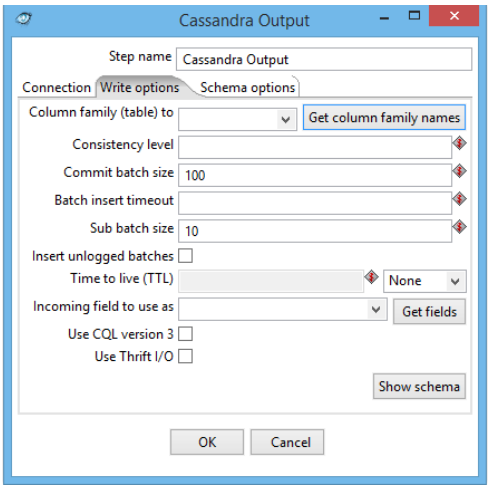


Figura 18- Panel *Write options* del paso *Cassandra Output*

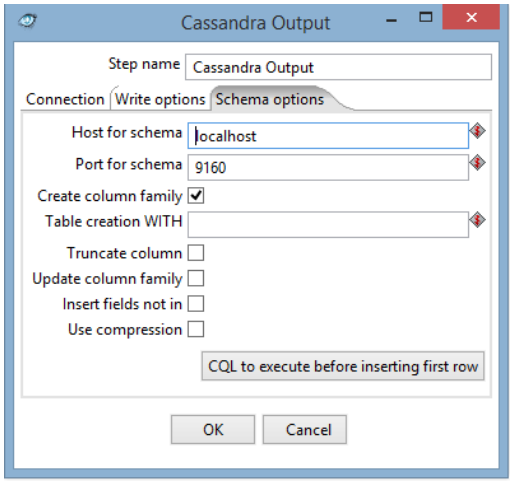


Figura 19-Panel Shema options *del paso* Cassandra Output .

Las opciones que se observan en los paneles Write option Figura 18 y Schema option Figura 19 se explican en la Tabla 7.

Opción	**Definición**
Column family (table)	Campo de entrada para especificar la familia de la columna, a la que las filas entrantes deben ser escritas.
Get column family names button	Puebla la opción *Column family* con los nombres de todas las columnas que existen en la BD especificada.

Consistency level	Especifica el nivel de coherencia, los valores válidos son cero,uno, cualquiera,todos, el valor por defecto de Cassandra es ONE.
Truncate column family	Si está activada esta opción se eliminan los datos existentes en las columnas antes de insertar datos en ellas.
Update column family metadata	Si se selecciona, se actualizan los metadatos de familia de columnas con información sobre los campos de entrada que no están ya presente, cuando se selecciona la opción. Si no se selecciona esta opción, los campos entrantes desconocidos se ignorados a menos que los campos de inserción no en opción metadatos de columna esté habilitado.
Insert fields not in column metadata	Se inserta los metadatos de la familia columna en todos los campos de entrada que no están presentes los mismos. Esta opción no tiene efecto si se selecciona la opción *Updatecolumn family metadata*.
Use compression	Opción comprimir (gzip) el texto de cada instrucción INSERTAR LOTE antes de transmitirla al nodo.
CQL to execute before inserting first row	Abre el editor CQL, donde se puede introducir una o más sentencias CQL separada por comas para ejecutar antes de insertar datos en la primera fila.

Tabla 7- Descripción de los paneles *Write options y Shema options del paso Cassandra Output*
Fuente[(Anónimo, 2012)]

2.1.4 Paso Hadoop File Input.

Hadoop es un marco de trabajo de código abierto para ejecutar aplicaciones distribuidas. Permite trabajar con miles de nodos y *petabytes* de datos. Inspirado en los documentos *Google* para *MapReduce* y *Google File System* (GFS), se trata de un proyecto de Apache que continua en fase de desarrollo (Serrat Morros 2013).

El paso *Hadoop File Input* se utiliza para leer los datos de diferentes tipos de archivos de texto almacenados en un clúster Hadoop. Los formatos más utilizados incluyen valores separados por comas (archivos CSV) y generados por hojas de cálculo Excel.

La Figura 20 muestra el panel *File* del paso *Hadoop File Input.*

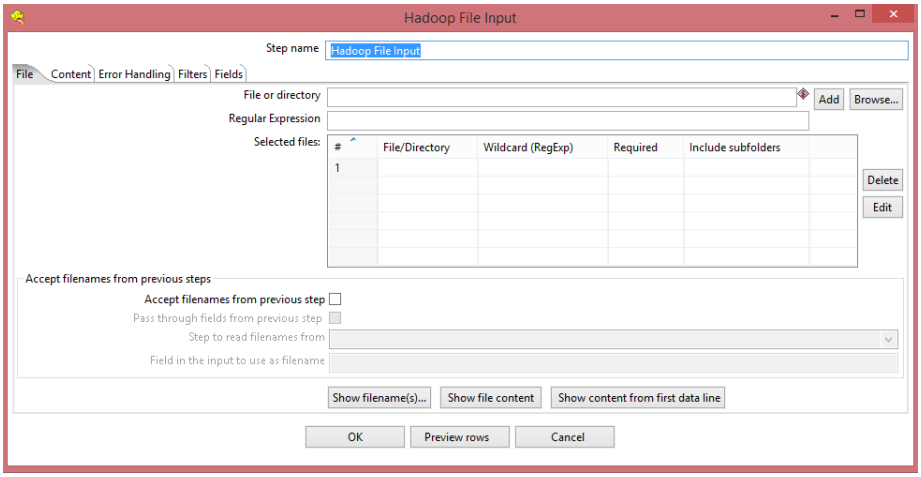


Figura 20- Panel File del paso *Hadoop File Input*

Las opciones que se observan en el panel *File* Figura 20 se explican en la Tabla 8.

Opción	Definición
File or Directory	Especifica la dirección del fichero que se desea leer. Primero hacer clic en *browse* para buscar y seleccionar el fichero, luego hacer clic en *add* para añadirlo a la lista de ficheros seleccionados.
Accept file names from previous steps	Obtener el nombre de los archivos desde los pasos anteriores.
Step to read file names from	Paso para leer el nombre de los archivos.
Field in the input to use as file name	Se ve la entrada de texto de archivos en este paso para determinar qué nombres de archivos utilizar.

Tabla 8- Descripción del panel *File* del paso *Hadoop File Input*[(Anónimo, 2012)]

La Figura 21 muestra el panel *Content* del paso *Hadoop File input.*

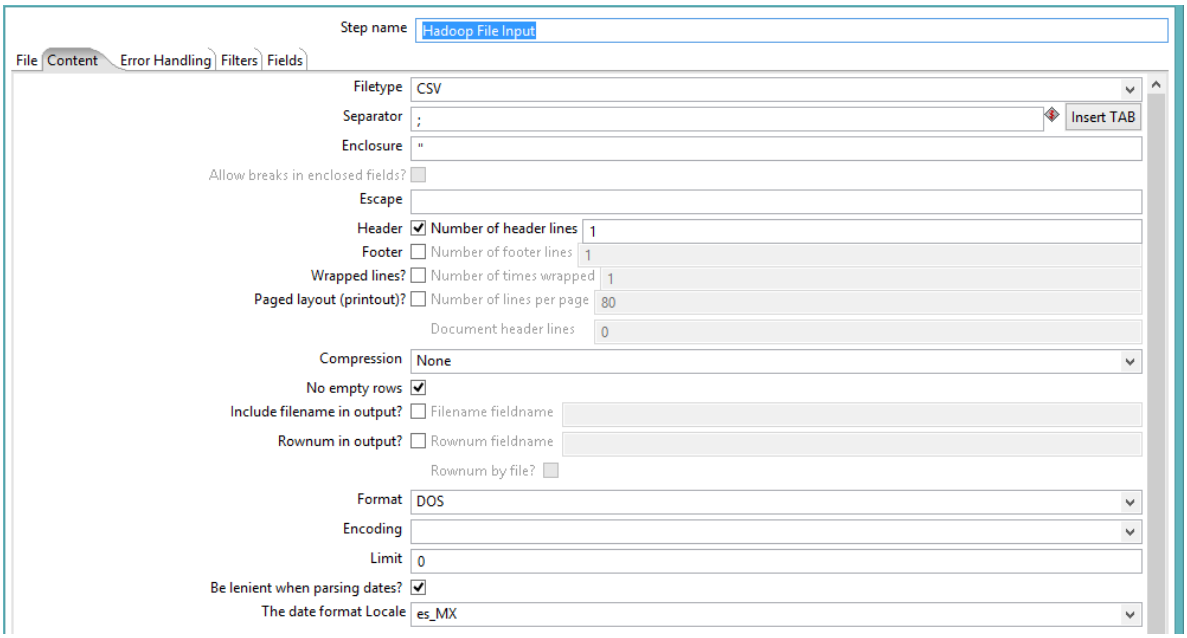


Figura 21-*Panel Content* del paso *Hadoop File Input*

Las opciones que se observan en el panel *Content* Figura 21 se explican en la Tabla 9.

Opción	Definición
File type	Tipo de fichero
Separator	Separador de campos. Separador más usado (;).
Enclosure	Separador de textos.
Compression	Se permite esta opción si el archivo está comprimido (Zip, GZip).
No empty rows	No enviar filas vacías a pasos siguientes.
Include file name in output	Si se activa esta opción el nombre del archivo forma parte de la salida.
File name field name	Nombre del campo que contiene el nombre de archivo.
Rownum in output?	Permite que el número de fila forme parte de la salida.
Row number field name	Nombre del campo que contiene el número de fila.

Format	Sistema Operativo del anfitrión.
Add filenames to result	Añadir los nombres de los archivos al resultado en una lista.

Tabla 9- Descripción Panel Content del Paso Hadoop File Input. Fuente[(Anónimo, 2012)]

La Figura 22 muestra el panel *Error Handing* del paso *Hadoop File Input*.

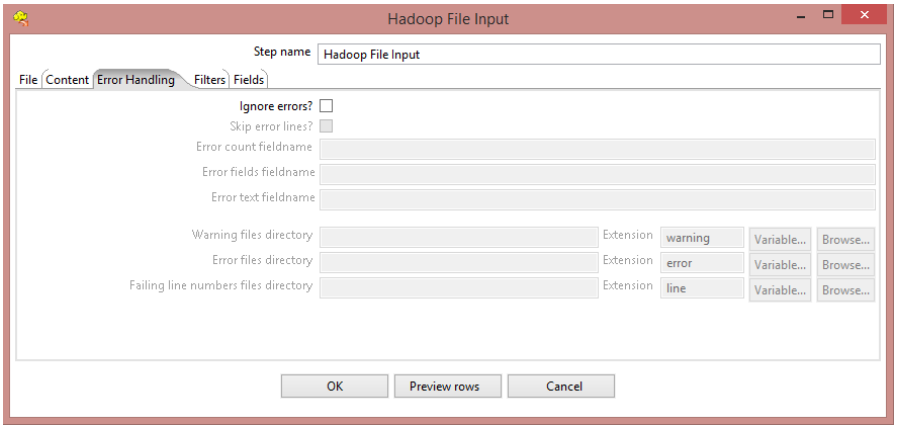


Figura 22- Panel *Error Handling* del paso *Hadoop File Input*

Las opciones que se observan en el panel *Error Handling* Figura 22 se explican en la Tabla 10.

Opción	**Definición**
Ignore errors?	Se ignoran los errores durante el análisis.
Skip error lines	Se pasan por alto las líneas que contienen errores. Se genera un archivo adicional con los errores, el cual contiene los números de línea de dichos errores.
Error count field name	Se añade un campo a la salida de las filas. Este campo contiene el número de errores por fila.
Error fields field name	Similar a *Error count field name*, el campo añadido contiene el nombre del campo con error.
Error text field name	Similar a *Error count field name*, el campo añadido contiene un análisis del error.
Warnings file directory	Dirección del directorio donde se almacenan las advertencias

	(<warning dir>/filename.<date_time>.<warning extension>).
Error files directory	Dirección del directorio donde se almacenan los errores (<errorfile_dir>/filename.<date_time>.<errorfile_extension>).
Failing line numbers files directory	Dirección del directorio donde se almacena el número de línea donde ocurre un error (<errorline dir>/filename.<date_time>.<errorline extension>).

Tabla 10- Descripción del panel Error Handling del paso Hadoop File Input. Fuente[(Anónimo, 2012)]

La Figura 23 muestra el panel *Filtrers* del paso *Hadoop File Input*.

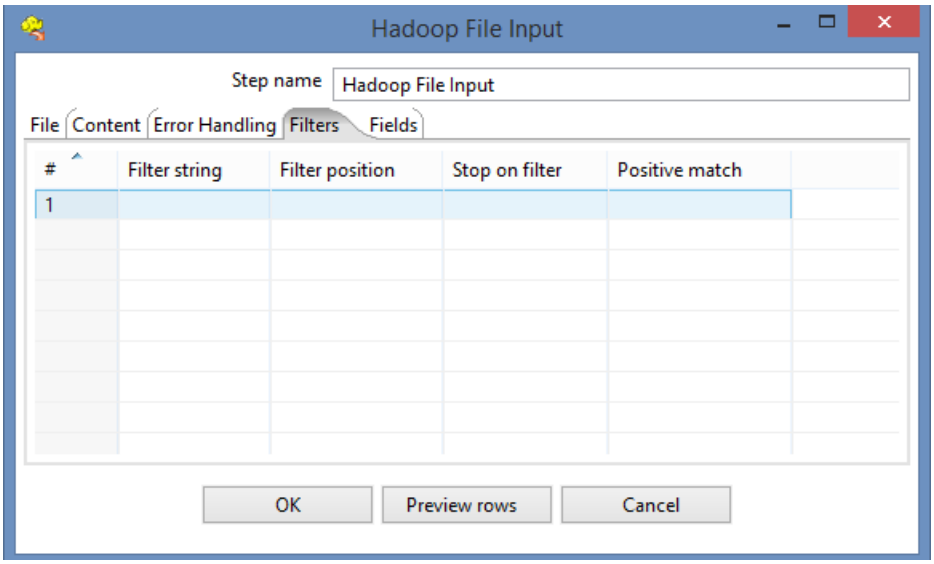


Figura 23- Panel *Filters* del paso *Hadoop File Input*

Las opciones que se muestran en panel *Filters* Figura 23 se explican Tabla 11

Opción	Definición
Filter string	Indica palabra a buscar
Filter position	La posición en la que la cadena de filtro debe ser colocada en la línea. Cero (0) es la primera posición en la línea. Si especifica un valor por debajo de cero, la cadena de filtro se busca en toda la cadena.

Stop on filter	Indica donde se desea detener el análisis.
Positive match	Activa los filtros en modo positivo cuando se enciende. Se pasarán sólo las líneas que coincidan con este filtro. Filtros negativos tienen prioridad y son inmediatamente descartados.

Tabla 11- Descripción del panel *Filters* del paso *Hadoop File Input.* Fuente[(Anónimo, 2012)]

2.1.5 Paso *Hadoop File Output.*

El paso *Hadoop File Output* se utiliza para exportar datos a archivos de texto almacenados en un clúster Hadoop (Anónimo, 2012).

La Figura 24 muestra el panel *File* del paso *Hadoop fiel Output*.

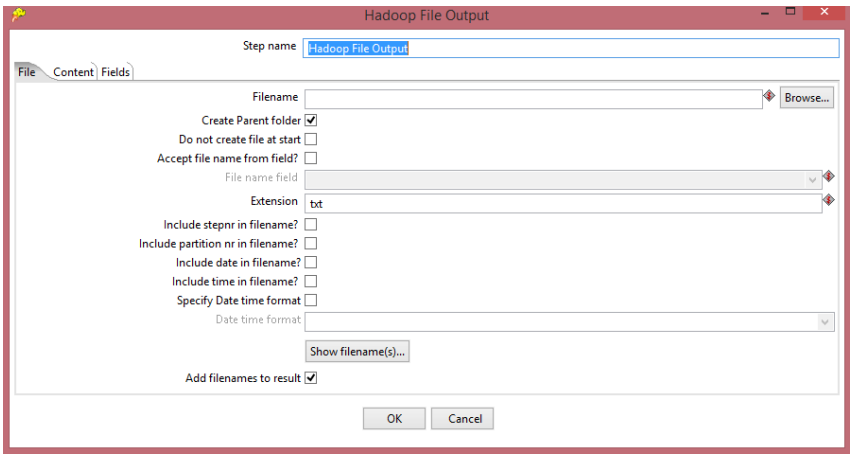


Figura 24- Panel *File* del paso *Hadoop File Output*

Las opciones que se muestran en el panel *File* Figura 24 se explican en la Tabla 12.**¡Error! No se encuentra el origen de la referencia. ¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**

Opción	Definición
Filename	Ubicación del archivo de texto donde se va a escribir el fichero, para hacerlo dar clic en *browse* y se navega hasta encontrar la dirección

	donde está el archivo.
Extension	Se indica extensión del fichero.

Tabla 12- Descripción del Panel *File* del paso Hadoop File Output. Fuente[(Anónimo, 2012)]

La Figura 25 muestra el panel *Content* del paso *Hadoop File Output*.

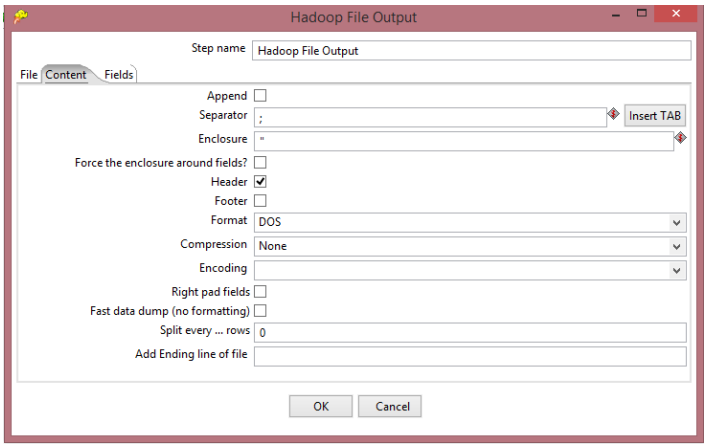


Figura 25-Panel *Content* del paso *Hadoop File Output*

Las opciones que se muestran en el panel *Content* del paso *Hadoop File Output* Figura *25* se explican en la Tabla 13.

Opción	**Definicón**
Separator	Separador de campos, usualmente (;).
Enclosure	Separador de texto

Tabla 13- Descripción del panel *Content* del paso Hadoop File Output. Fuente[(Anónimo, 2012)]

2.1.6 Paso *Hbase input.*

HBase es una base de datos clave-valor orientada a columnas (*column-family*). Capaz de manejar grandes conjuntos de datos con operaciones simultáneas de lectura y escritura. Cada tabla contiene filas y columnas como una base de datos relacional. HBase permite que muchos atributos sean agrupados en las denominadas familias de columnas, de tal manera que los elementos de una familia de columnas son almacenados en un solo conjunto (White, 2012).

El paso *HBase input* esencialmente lee datos de una tabla *HBase* de acuerdo a metadatos definidos por el usuario.La Figura 26 muestra el panel *Configure query* del paso *Hbase input.*

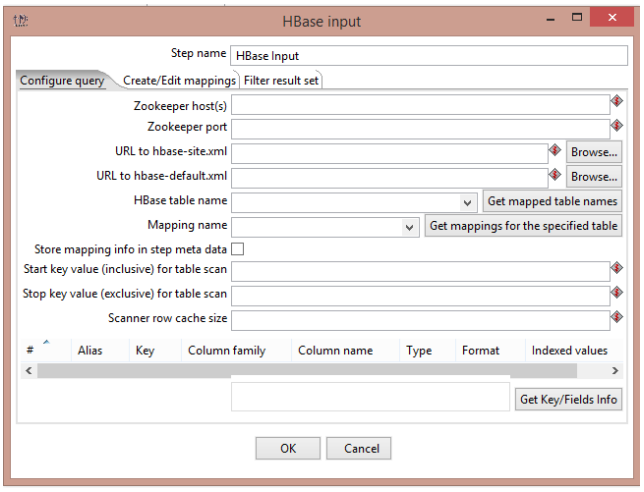


Figura 26-Panel *Configure query* del paso *Hbase input*

Las opciones que se muestran en el panel *Cofigure query* Figura *26* se explican en la Tabla 14.

Opción	Definición
URL to hbase-site.xml	Dirección del hbase-site.xml file
URL to hbase-default.xml	Dirección del hbase-default.xml file.
HBase table name	Nombre de la tabla hbase. Dar clic en *get mapped table names* para obtener las tablas Hbase disponibles.
Mapping name	Mapeo para decodificar e interpretar los valores de columna. Haga clic en "Obtener asignaciones para la tabla especificada" para rellenar la lista desplegable de asignaciones disponibles.
Scanner row cache size	Número de filas que deben ser almacenadas cada vez que un pedido de extracción es hecho a HBase.
Get Key/Fields Info	Una vez introducidos correctamente los datos

	de conexión, se pueblan los campos de la tabla que se observa.

Tabla 14- Descripción del panel *Configure query* del paso Hbase input. Fuente[(Anónimo, 2012)]

2.1.7 Paso Hbase output.

El paso *Hbase output* escribe datos en una tabla *HBase* de acuerdo a metadatos definidos por el usuario (Anónimo, 2012).

La Figura 27 muestra el panel *Configure connection* del paso *HBase output*

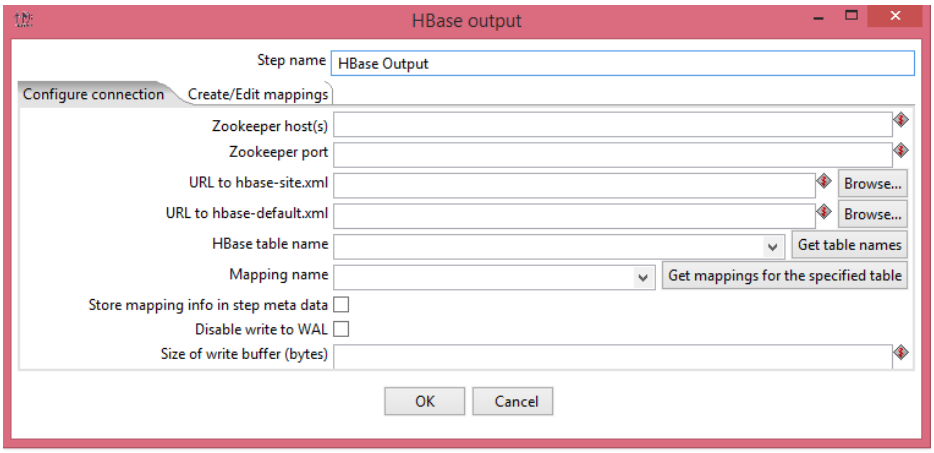


Figura 27- Panel *Configure connection* del paso *HBase output*

La configuración del panel Configure connection del paso Habase output es similar a la configuración de la Tabla 14.

2.1.8 Paso *Hbase Row Decoder.*

El paso HBase descodifica una clave entrante y el objeto resultante HBase de acuerdo con el mapeo(Anónimo, 2012).

La Figura 28 mueatra el panel *Configure fields*del paso *HBase Row Decoder*.

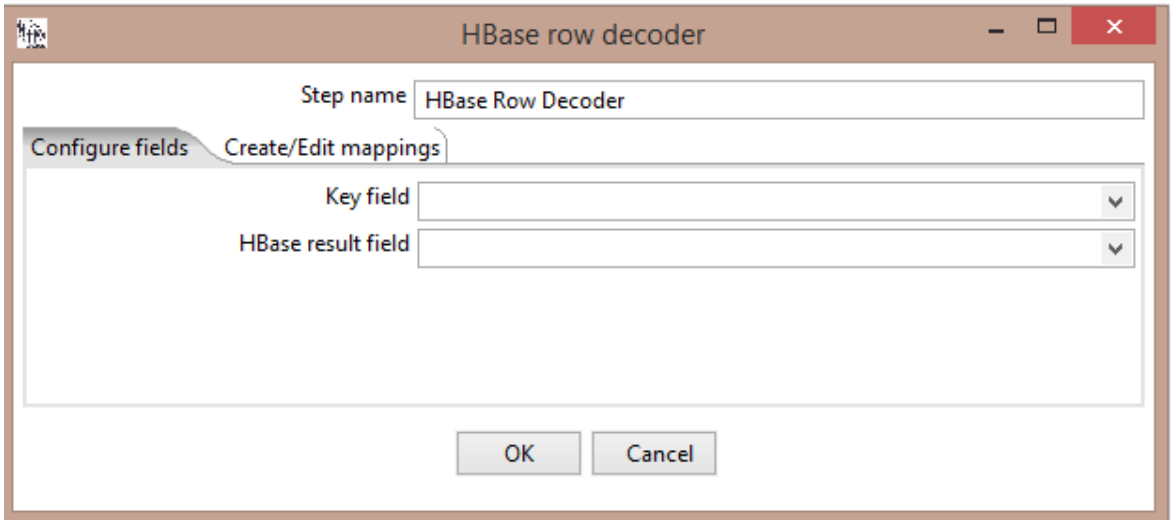


Figura 28-Panel *Configure fields* del paso *HBase Row Decoder*

Las opciones que se observan en el panel *Configure fields* Figura 28 se explican en Tabla 15.

Opción	Definición
Key field	Llave del campo
HBase result field	Campo que contiene el resultado serializado de *HBase.*

Tabla 15- Descripción del panel Configure fields *del paso* HBase Row Decoder. Fuente[(Anónimo, 2012)]

La Figura 29 muestra el panel *Creae/Edit mappings* del paso *HBase row decoder*.

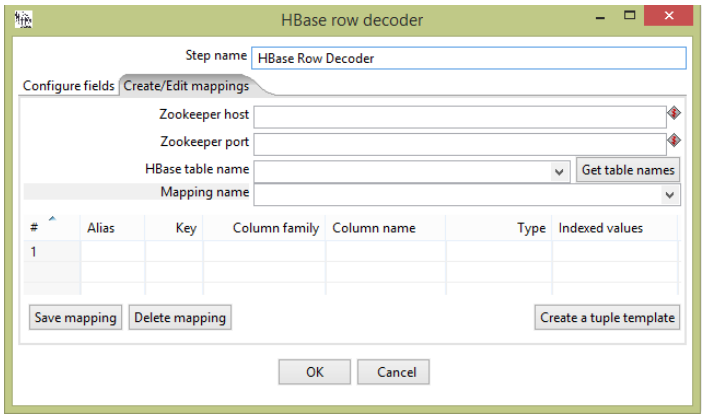


Figura 29- Panel *Create/Edit mapping* del paso *Hbase Row Decoder*

Las opciones que se muestran en el panel *Create/Edit mapping* Figura 29 se explican en Tabla 16.

Opción	Definición
HBase table name	Se muestran nombres de tablas Hbase
Create a tuple template	Parcialmente rellena la tabla con los campos especiales que definen un mapeo por tupla para su uso en el modo de salida tupla.

Tabla 16 - Descripción del Panel *Create/Edit mapping* del paso *Hbase Row Decoder*. Fuente[(Anónimo, 2012)]

2.1.9 Paso MongoDB input.

MongoDB se basa en el principio de almacenar los datos en una estructura tipo llave-valor. MongoDB se enfoca específicamente en que los valores de estas llaves (llamadas **colecciones**) son estructuras tipo JSON (llamados **documentos**), es decir objetos Javascript. MongoDB posee varias estrategias de manejo de datos tales como sus procesos de división de datos en distintos equipos físicos o también conocido como *clusterización*. En caso de documentos muy grandes que superen el límite estipulado de 16MB se aplica

una estrategia llamada GridFS que automáticamente divide el documento en partes y los almacena por separado. Al recuperar dicho documento, el *driver* se encarga de armar automáticamente el documento nuevamente. La estructura de almacenamiento es tan flexible que uno de los hechos importantes que se comparten al introducir esta base de datos es que: distintos documentos en la misma colección no deben tener obligatoriamente los mismos campos o estructura. Inclusive documentos con campos en común no tienen necesariamente que tener el mismo tipo de dato(Wiesel, 2013). La Figura 30 muestra el panel Configure conection del paso MongoDB Input.

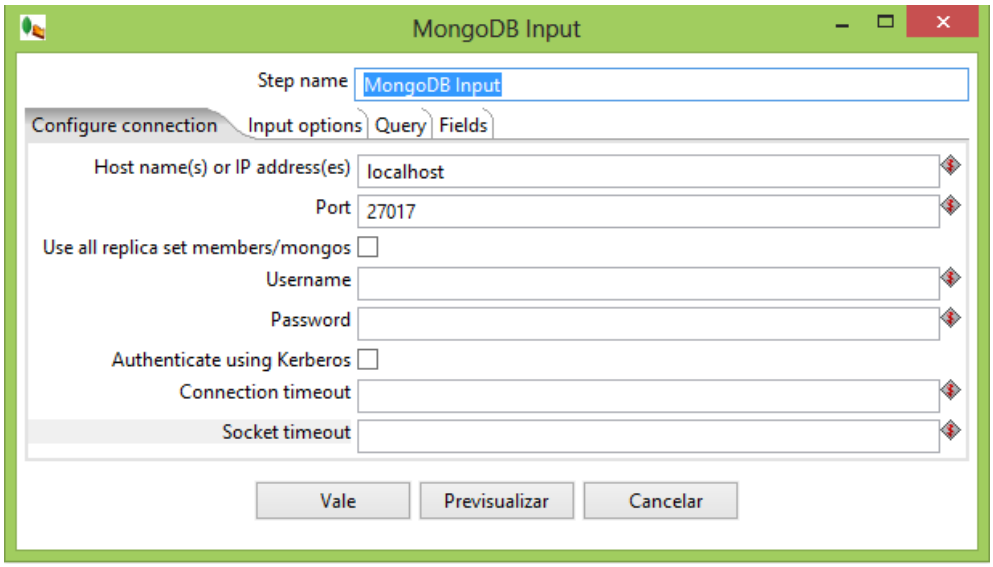


Figura 30 - Panel *Configure connection* del paso *MongoDB input*

Las opciones que se observan en el panel *Configure connection* Figura 30 se explican en la Tabla 17.

Opción	Definición
Host name(s) or IP address(es)	ip de la máquina host.
Port	Insertar el Puerto.

Tabla 17- Descripción del panel *Configure connection* del paso *MongoDB input.* Fuente[(Anónimo, 2012)]

La Figura 31 muestra el panel *Input options* del paso *MongoDB Input.*

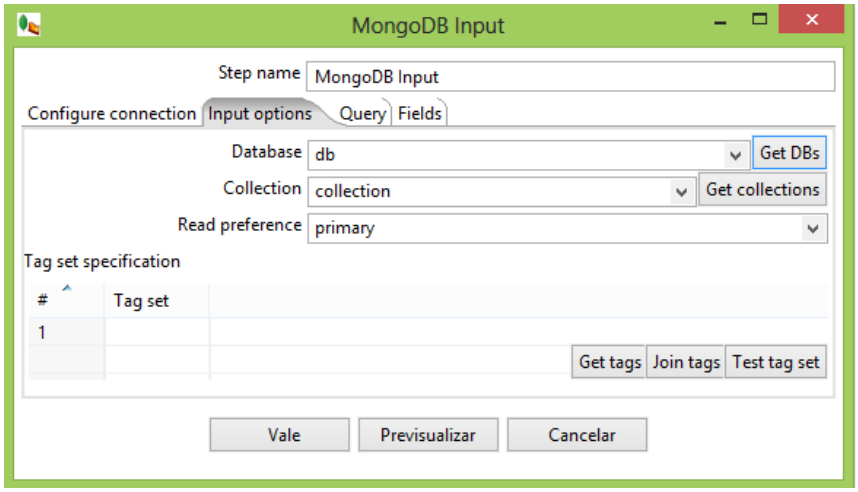


Figura 31- Panel *Input option* del paso *MongoDB input*

Las opciones que se observan en el panel *Input option* Figura 31 se explican en la Tabla 18.

Opción	Definición
Database	Se selecciona la base de datos, para esto dar dar clic en Get DBs.
Collection	Se selecciona la conexión, para esto dar clic en Get collection.

Tabla 18- Descripción del panel Input option del paso *MongoDB input.* Fuente[(Anónimo, 2012)]

La Figura 32 muestra el panel *Fields* del paso *MongoDB Input.*

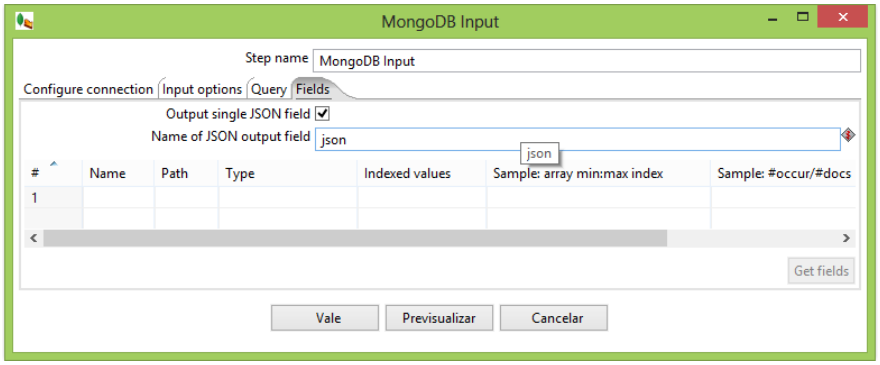


Figura 32-Panel *Fields* del paso *MongoDB input*

Las opciones que se observan en el panel *Fields* Figura 32 se explican en la Tabla 19.

Opción	Definición
Output single Json field	Se trata el documento con los campos separados o como un campo JSON.
Get fields	Se obtienen los campos y se eliminan los que no se necesitan.

Tabla 19 - Descripción del Panel *Fields* del paso *MongoDB input* Fuente[(Anónimo, 2012)]

2.1.10 Paso MongoDB output.

El paso *MongoDB output* escribe para una colección de Mongo.La Figura 33 muestra el panel *Configure connection* del paso *MongoDB Output.*

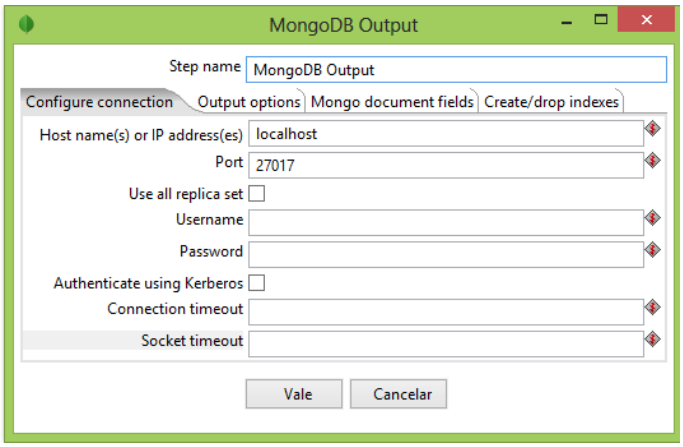


Figura 33-Panel Configure connection del paso MongoDB output

La configuración del panel *Configure connection* Figura 33 es igual que la especificada en la Tabla 17.

La Figura 34 muestra el panel panel Output option del paso MongoDB output.

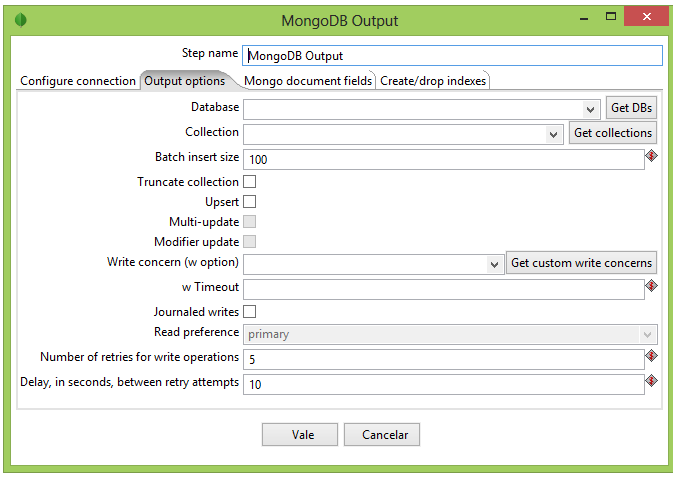


Figura 34-Panel *Output option* del paso *MongoDB output*

Las opciones que se observan en el panel *Output option* Figura 34 se explican en la Tabla 20.

Opción	Definición
Database	Se selecciona la base de datos, para esto dar dar clic en Get DBs.
Collection	Se selecciona la conexión, para esto dar clic en Get collection.
Batch insert size	Definir la cantidad de filas a tener antes de realizar la inserción.

Tabla 20 - Descripción del panel *Output option* del paso *MongoDB output* . Fuente[(Anónimo, 2012)]

2.1.11 Paso MapReduce input.

Este paso define los pares clave / valor para la entrada de Hadoop. La Figura 35 muestra el paso *MapReduce Input.*

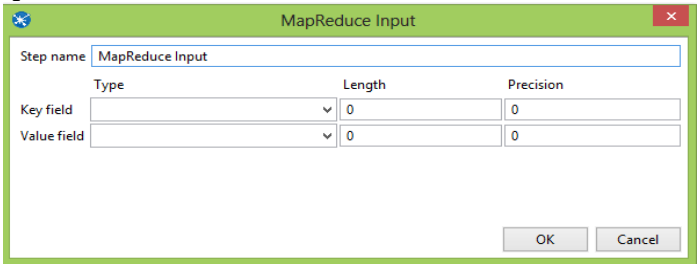


Figura 35-Paso *MapReduce Input*

Las opciones que se muestran en el paso MapReduce Input Figura 35 se explican en la Tabla 21.

Opción	Definición
Key field	Se define el tipo de dato de la llave
Value field	Se define tipo de dato del valor

Tabla 21 - Descripción del Paso *MapReduce Input.* Fuente[(Anónimo, 2012)]

2.1.12 Paso MapReduce output.

Este paso define los pares clave / valor para la salida de *Hadoop*. La Figura 36 muestra el paso MapReduce Output.

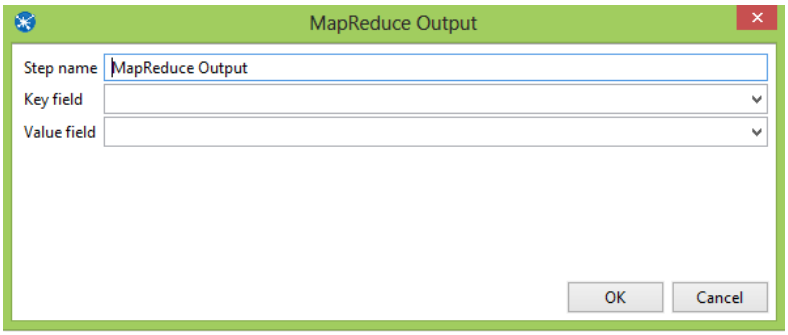


Figura 36- Paso *MapReduce Output*

Las opciones que se muestran en el paso *MapReduce Output* Figura 36 se explican en la Tabla 22.

Opción	Definición
Key field	El campo de salida Hadoop que representa la llave.
Value field	El campo de salida Hadoop que representa el valor.

Tabla 22 - Descripción del paso *MapReduce Output.* Fuente[(Anónimo, 2012)]

2.1.13 Paso CouchDB input.

Es un gestor de bases de datos de código abierto, cuya finalidad está puesta en mantener la facilidad de su uso y en ser "una base de datos que asume la web de manera completa". Se trata de una base de datos NoSQL que emplea JSON para almacenar los datos, JavaScript como lenguaje de consulta por medio de MapReduce y HTTP como API. Una de sus características más peculiares es la facilidad con que permite hacer replicaciones (Anderson et al., 2010).

La Figura 37 muestra el paso *CouchDB Input.*

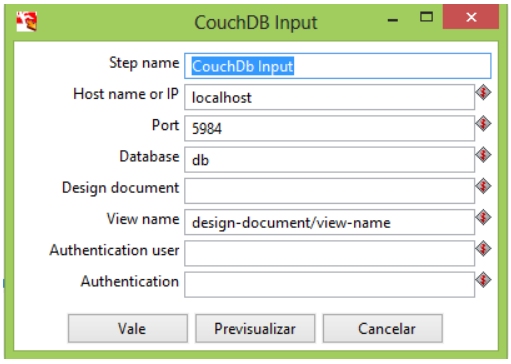


Figura 37-*Paso CouchDB Input*

Las opciones que se muestran en el paso CouchDB Input Figura *37* se explican en la Tabla 23 .

Opción	Definición
Host	Número de la dirección ip.
Port	Número del puerto
Database	Nombre de la BD
Authentication user	Nombre de usuario para acceder a la base de datos.
Authentication password	Contraseña para acceder a la BD.

Tabla 23 - Descripción del paso *CouchDB Input* Fuente[(Anónimo, 2012)]

2.1.14 Paso SStable output.

La Figura 38 muestra el paso *SSTable Output.*

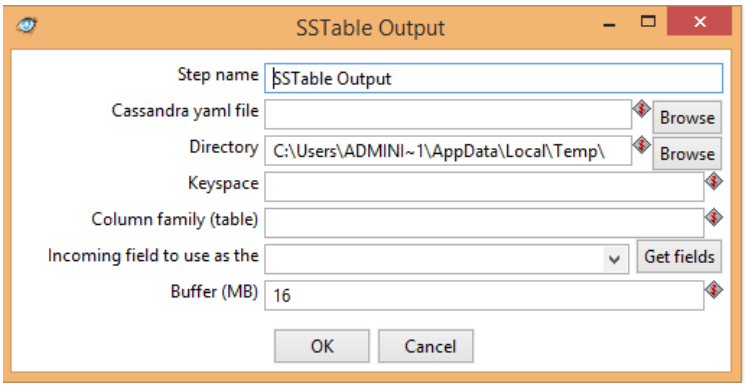


Figura 38- Paso *SStable Output*

Las opciones que se muestran en el paso *SStable Output* Figura 38 se explican en la Tabla 24.

Opciones	Definicion
Cassandra yaml file	Ubicación del archivo YAML. El archivo cassandra.yaml es el principal archivo de configuración para Cassandra y define nodo y detalles de la configuración del clúster.
Directory	Ubicación de la salida.
Incoming field to use as the row key	Permite seleccionar qué fila de entrada se utilizará como la fila clave. Este cuadro desplegable se rellena con los nombres de campos de transformaciones entrantes.

Tabla 24 - Descripción del paso *SStable Output.* Fuente[(Anónimo, 2012)]

2.2 Características de los pasos *big data* en un trabajo.

Un trabajo o *job* es similar al concepto de proceso, un conjunto sencillo o complejo de tareas con el objetivo de realizar una acción determinada. En los trabajos se pueden utilizar

pasos específicos (diferentes a los disponibles en las transformaciones) como recepción de un fichero vía ftp, envío de email, ejecución un comando, etc. Además, es posible ejecutar una o varias transformaciones y orquestarlas siguiendo un orden determinado.

2.2.1 Amazon EMR Job Executor.

Esta entrada de trabajo ejecuta trabajos de Hadoop en una cuenta de Amazon Elastic MapReduce (EMR). Para utilizar este paso, se necesitan los servicios Amazon Web (AWS), cuenta configurada para EMR, y un JAR de Java preparado para controlar el trabajo a distancia.

La Figura 39 muestra el paso Amazon EMR Job Executor.

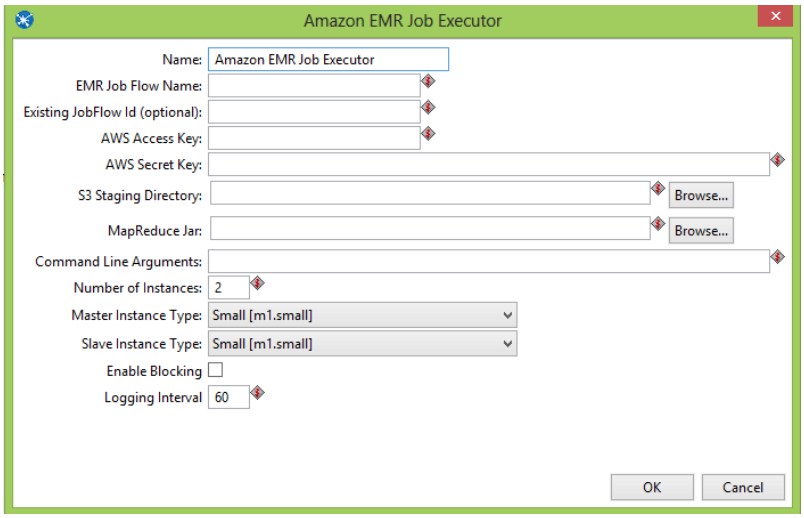


Figura 39-Paso *Amazon EMR Job Executor*

Las opciones que se muestran en el paso Amazon *EMR Job Executor* Figura 39 se explican en la Tabla 25.

Opción	Definición
EMR Job Flow Name	El nombre del flujo de trabajo de Amazon EMR que se está ejecutando.
AWS Access Key	Su clave de acceso para Amazon Web Services.
AWS Secret Key	Su clave secreta para Amazon Web Services.

S3 Staging Directory	La dirección de Amazon Simple Storage Service (S3).
MapReduce JAR	El JAR de Java que contiene las funciones Map y Reduce.
Master Instance Type	El tipo de instancia de Amazon EC2 que actuará como "maestro" en el clúster Hadoop, contiene la distribución de tareas MapReduce.
Slave Instance Type	El tipo de instancia de Amazon EC2 que actuará como uno o más "esclavos" de Hadoop en el clúster.
Enable Blocking	Obliga al trabajo que espere hasta que cada paso se complete antes de continuar con el siguiente paso.

Tabla 25 - Características del paso *Amazon EMR Job Executor*. Fuente[(Anónimo, 2012)]

2.2.2 Paso Amazon Hive Job Executor.

El Paso *Amazon Hive Job Executor* presenta características similares al paso *Amazon EMR Job Executor* solo que el primero permite configurar un script de *Hive*.

La Figura 40 muestra el paso *Amazon Hive Job Executor*.

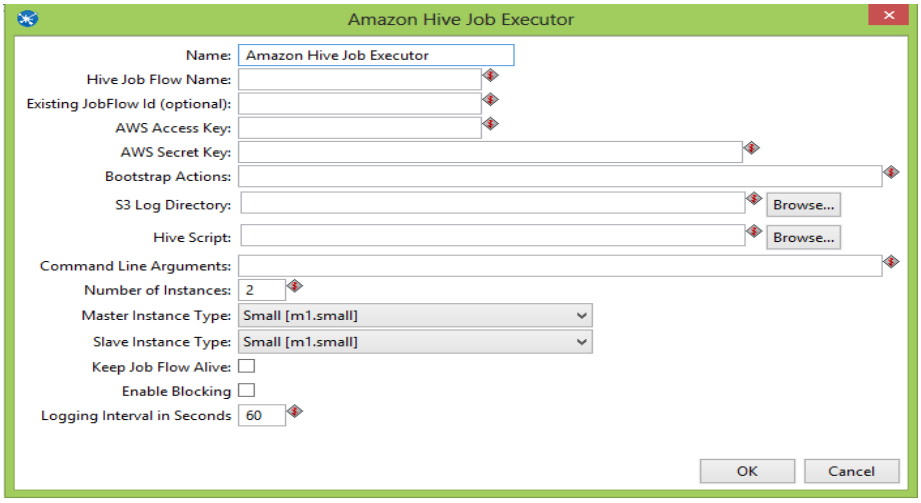


Figura 40- Paso *Amazon Hive Job Executor*

Las opciones que se muestran en el paso *Amazon Hive Job Executor* Figura 40 se explican en la Tabla 26.

Opcion	Definicion
Hive Job Flow Name	El nombre del flujo de trabajo Hive para ejecutar.
Hive Script	La URL del script Hive para ejecutar dentro de Amazon S3.
Command Line Arguments	Lista de argumentos para pasar a Hive (cadenas separadas por espacios).
Master Instance Type	Instancia de Amazon EC2 que actuará como "maestro" en el clúster Hadoop, maneja distribución de tareas MapReduce.
Slave Instance Type	El tipo de instancia de Amazon EC2 que actuará como uno o más "esclavos" de Hadoop en el clúster.
Keep Job Flow Alive	Especifica si el flujo de trabajo debe terminar después completar todos los pasos.

Tabla 26 - Descripción del paso *Amazon Hive Job Executor*. Fuente[(Anónimo, 2012)]

2.2.3 Hadoop Copy Files.

Esta entrada de trabajo copia archivos de un clúster *Hadoop* de un lugar a otro.

La Figura 41 muestra el panel General del paso *Hadoop Copy Files.*

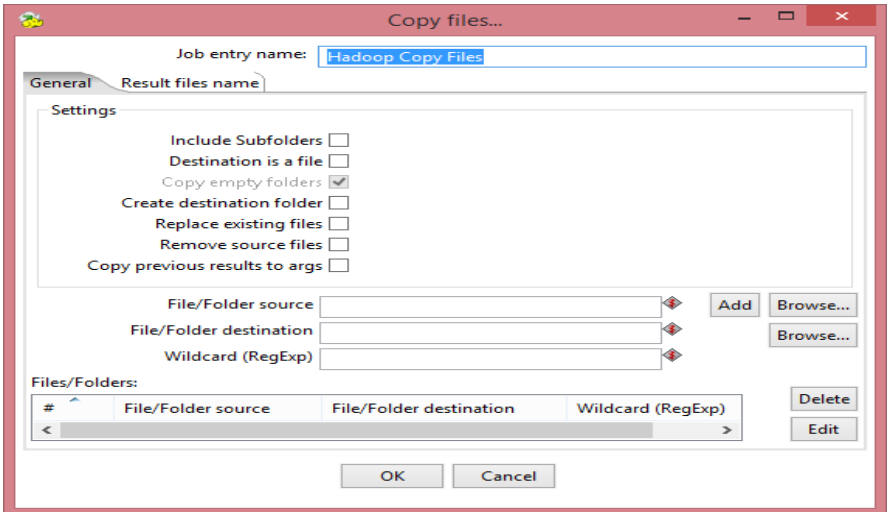


Figura 41-Paso *Hadoop Copy Files.*

Las opciones que se muestran en el panel General del paso *Hadoop Copy Files* Figura 41 se explican en la Tabla 27.

Opción	Definición
File/folder source	Carpeta que se desea tener como fuente.
File/folder destination	Fichero o directorio destino, hacer clic en Browse y seleccionar Hadoop para entrar a los detalles de la conexión del clúster Hadoop.
Wildcard (RegExp)	Expresión regular a través de la cual se van a seleccionar los archivos.

Tabla 27 - Descripción del paso *Hadoop Copy Files.* Fuente[(Anónimo, 2012)]

2.2.4 Paso Pentaho MapReduce.

Esta entrada de trabajo ejecuta transformaciones como parte de un trabajo *Hadoop MapReduce*. Esto se utiliza frecuentemente para ejecutar transformaciones que actúan como mapeos y reductores (Anónimo, 2012).

La Figura 42 muestra el panel *Mapper* del paso *Pentaho MapReduce*.

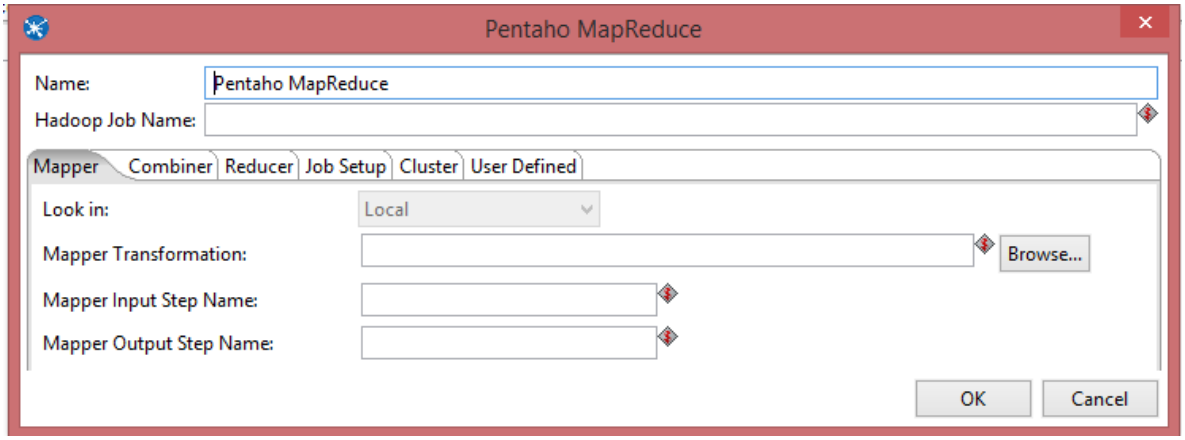


Figura 42- Panel *Mapper* del paso *Pentaho MapReduce*

Las opciones que se muestran en el panel *Mapper* Figura 42 se explican en la Tabla 28.

Opción	Definición
Mapper Input Step Name	Paso de entrada
Mapper Output Step Name	Paso de salida.

Tabla 28 - Descripción del panel *Mapper* del paso *Pentaho MapReduce.* Fuente[(Anónimo, 2012)]

La Figura 43 muestra el panel *Reducer* del paso *Pentaho MapReduce.*

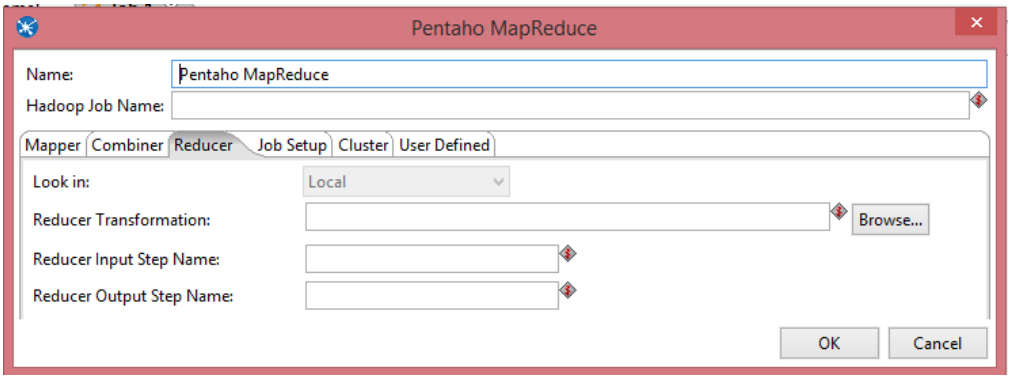


Figura 43-Panel *reducer* del paso *Pentaho MapReduce*

Las opciones que se observan en el panel *Reducer* Figura 43 se explican en la Tabla 29.

Opción	Definición
Reducer Transformation	Transformación que se utiliza como *Reducer*.
Reducer Input Step Name	Paso de entrada.
Reducer Output Step Name	Paso de salida.

Tabla 29 - Descripción del Panel *Reducer* del paso *Pentaho MapReduce*. Fuente[(Anónimo, 2012)]

La *Figura 44* muestra el panel *Job Setup* del paso *Pentaho MapReduce*.

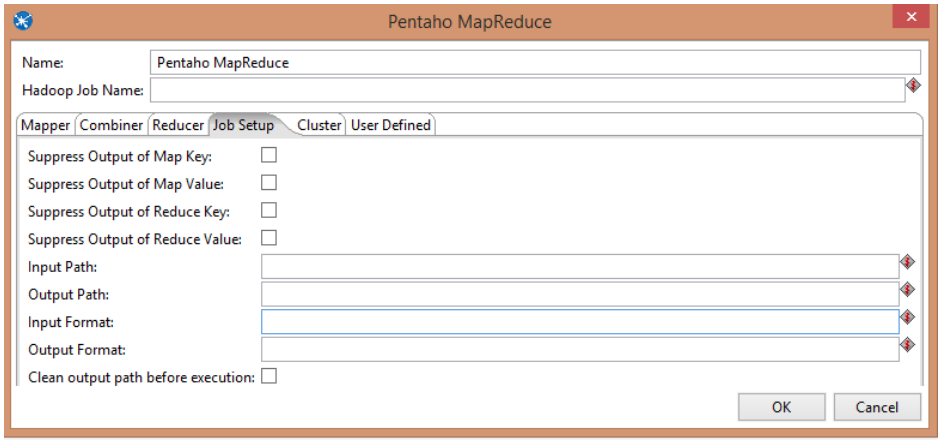


Figura 44-Panel *Job Setup* del paso *Pentaho MapReduce*

Las opciones que se muestran en el panel *Job Setup* Figura 44 se explican en la Tabla 30.

Opción	Definición
Input Path	Se Introduce el directorio donde se encuentran los ficheros de entrada.
Output Path	Se Introduce el directorio donde se guarda la respuesta (salida).
Input Format	Formato de entrada y salida.

Tabla 30 - Descripción del panel *Job Setup* del paso *Pentaho MapReduce*. Fuente[(Anónimo, 2012)]

La Figura 45 muestra el panel Cluster del paso *Pentaho MapReduce.*

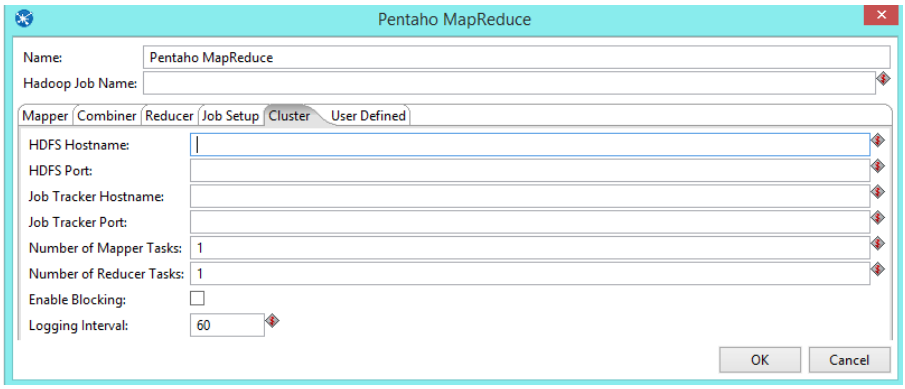


Figura 45-Panel *Cluster* del paso *Pentaho MapReduce*

Las opciones que se muestran en el panel *Cluster Figura 45* se explican en la Tabla 31

Opción	Definición
HDFS Hostname	Host donde se encuentra el clúster HDFS.
HDFS Port	Puerto.
Job Tracker Hostname	Host del JobTracker.
Job Tracker Port	Puerto del JobTracker.
Number of Mapper Tasks	Número de tareas para el Mapper.
Number of Reducer Tasks	Número de tareas para el Reducer.

Tabla 31 - Descripción del Panel *Cluster* del paso *Pentaho MapReduce*. Fuente[(Anónimo, 2012)]

2.2.5 Paso Sqoop Export.

Sqoop permite importar tablas individuales, o bases de datos enteras hacia HDFS. Además, genera clases en *Java*,desde las cuales se puede interactuar fácilmente con los datos importados. Otra de sus funcionalidades principales es la importación desde bases de datos SQL directamente hacia Hive (White, 2012).

La Figura 46 muestra el paso Sqoop Export.

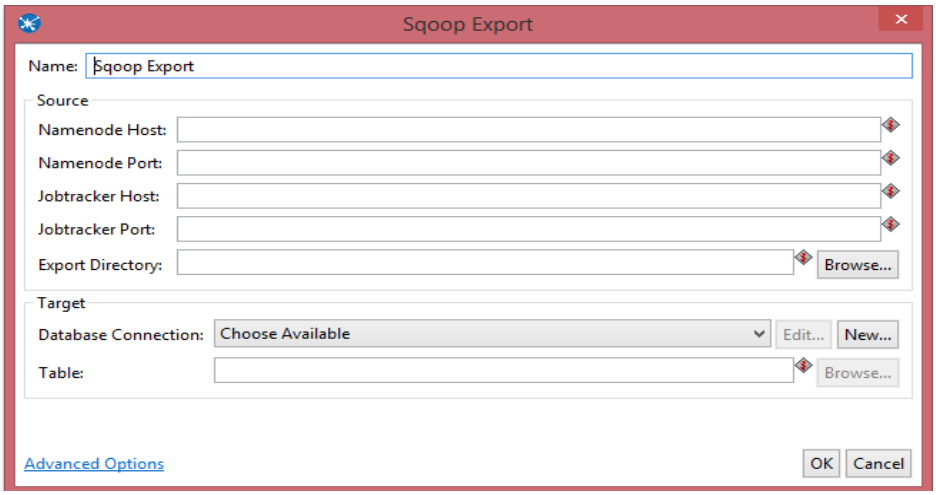


Figura 46-Paso *Sqoop Export*

Las opciones que se muestran en el paso *Sqoop Export* Figura 46 se explican en la Tabla 32.

Opción	Definición
Namenode Host	Host del Namenode.
Namenode Port	Puerto del Namenode.
Jobtracker Host	Host del Jobtracker
Job Tracker Port	Puerto del Jobtracker.
Export Directory	Directorio a exportar.
Database Connection	Conexión a la base de datos fuente.
Table	Se selecciona la tabla.

Tabla 32 - Descripción del paso *Sqoop Export.* Fuente[(Anónimo, 2012)]

2.2.6 Paso Sqoop Import.

Permite importar datos de una base de datos relacional en el Sistema de *Hadoop Distributed File* (HDFS) (Anónimo, 2012).

La Figura 47 muestra el paso *Sqoop Import*.

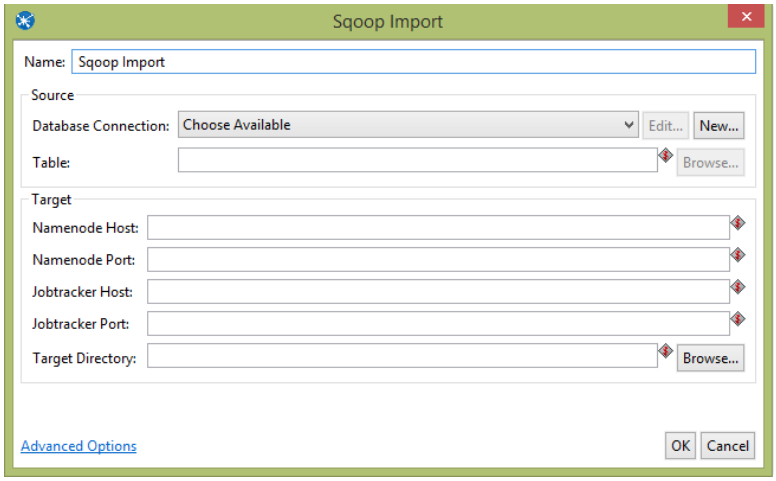


Figura 47- Paso *Sqoop Import*

Las opciones que se muestran en el paso *Sqoop Import* Figura 47 se explican en la Tabla 33.

Opción	Definición
Database Connection	Conexión a la base de datos fuente.
Table	Se selecciona la tabla.
Namenode Host	Host del Namenode.
Namenode Port	Puerto del Namenode.
Jobtracker Host	Host del Jobtracker
Job Tracker Port	Puerto del Jobtracker.
Target Directory	Directorio destino.

Tabla 33 - Descripción del paso *Sqoop Import.* Fuente[(Anónimo, 2012)]

2.2.7 Paso Hadoop Job Executor.

Esta entrada de trabajo ejecuta trabajos de Hadoop. Hay dos modos de opciones: simple (el estado por defecto), en que sólo se pasa un JAR de Java para controlar el trabajo; y avanzada, en la que usted es capaz de especificar los principales parámetros del método (pdf).

La Figura 48 muestra el paso *Hadoop Job Executor*.

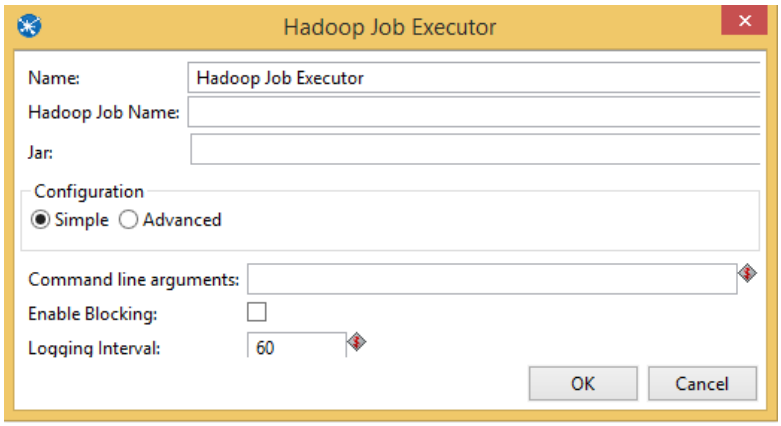


Figura 48- Paso *Hadoop Job Executor* en modo Simple

Las opciones que se muestran en el paso Hadoop Job Executor en modo Simple Figura 48 se explican en la Tabla 34.

Opción	Definición
Hadoop Job Name	Nombre del trabajo Hadoop que se está ejecutando.
Jar	Localización del JAR de Java

Tabla 34 - Descripción del Paso Hadoop Job Executor *en modo simple.* Fuente[(Anónimo, 2012)]

La Figura 49 muestra el paso *Hadoop Job Executor* en modo *Avanced*.

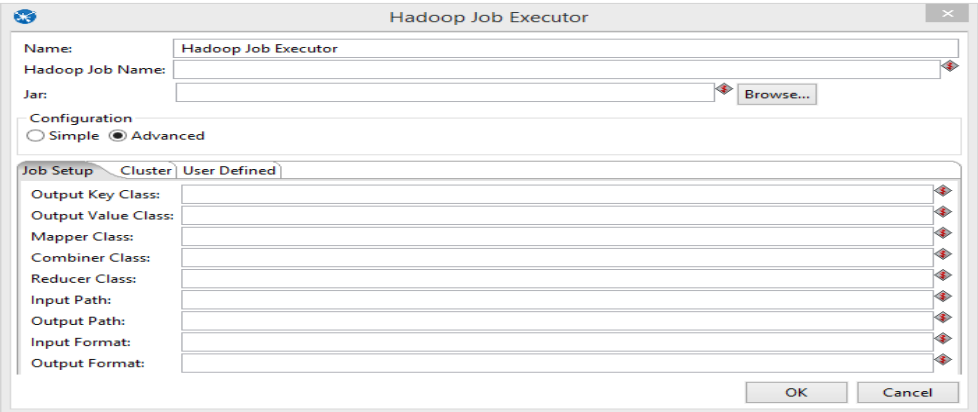


Figura 49- Panel Job Setup del paso Hadoop Job Executor en modo avanzado

Las opciones que se muestran en el panel Job Setup Figura 49 se explican en la Tabla 35.

Opción	Definición
Output Key Class	Nombre de la clase Hadoop que representa el tipo de dato de clave de salida.
Output Value Class	.El nombre de la clase Hadoop que representa el tipo de dato del valor de salida.
Mapper Class	La clase Java donde se realiza la operación de mapeo.
Combiner Class	La clase Java donde se realiza la operación de combinar.
Reducer Class	La clase Java donde se realiza la operación de reducir.
Input Path	La ruta de acceso al archivo de entrada en el clúster Hadoop.

Input Format	El nombre de la clase Hadoop que representa el tipo de dato del archivo de entrada.
Output Format	El nombre de la clase Hadoop que representa el tipo de dato del archivo de salida.

Tabla 35 - Descripción del *Panel Job Setup* del paso *Hadoop Job Executor* en modo avanzado. Fuente[(Anónimo, 2012)]

La Figura 50 muestra el panel Cluster del paso *Hadoop Job Executor* en modo avanzado.

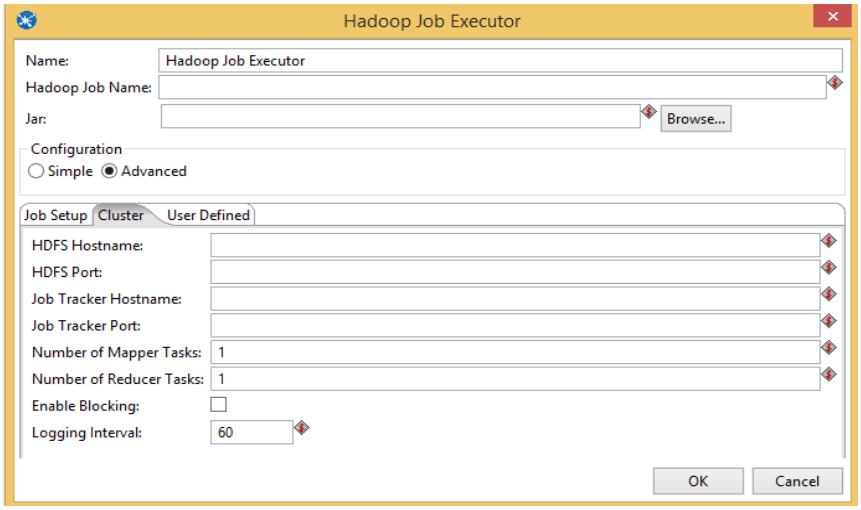


Figura 50-Panel Cluster del paso Hadoop Job Executor en modo avanzado

Las opciones que se muestran en el Panel Cluster Figura 50 se explican en la Tabla 36.

Opción	**Definición**
HDFS Hostname	Host para el clúster Hadoop.
HDFS Port	Número de puerto para el clúster Hadoop.
Job Tracker Hostname	Host del Jobtracker

Job Tracker Port	Puerto del Jobtracker.
Number of Mapper Tasks	Número de tareas para asignar a *map*
Number of Reducer Tasks	Número de tareas para asignar a *reduce*

Tabla 36 - Descripción del Panel *Cluster* del paso *Hadoop Job Executor* en modo avanzado. Fuente[(Anónimo, 2012)]

2.2.8 Paso Oozie Job Executor.

Oozie funciona como un planificador de flujo de trabajo que permite iniciar, parar, suspender y volver a ejecutar una serie de trabajos Hadoop. Los flujos de trabajo Oozie son grafos no cíclicos directos también conocidos como DAGs donde cada nodo es un trabajo o acción con control de dependencia, es decir, que una acción no puede ejecutarse a menos que la anterior haya terminado (Serrat moros 2013).

La Figura 51 muestra el paso *Oozie Job Executor*

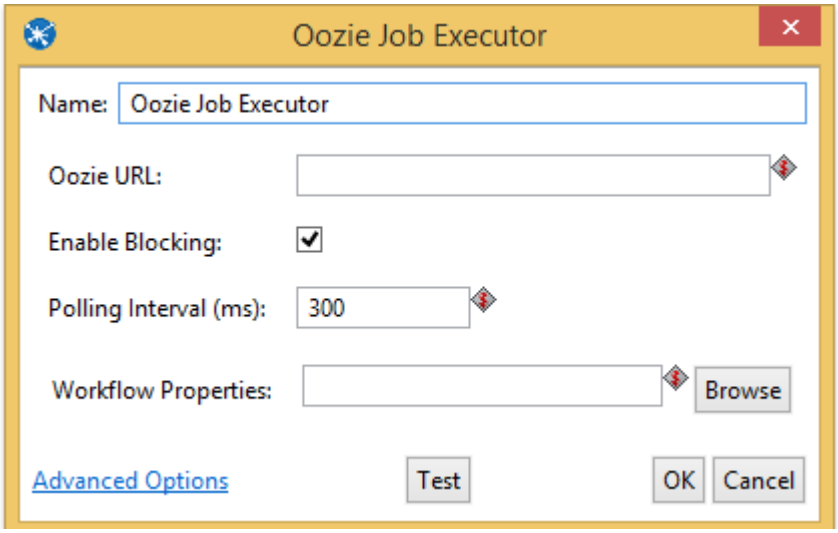


Figura 51-Paso *Oozie Job Executor*

Las opciones que se muestran en el paso Oozie Job Executor Figura 51 se explican en la Tabla 37.

Opción	Definición
Oozie URL	Campo para introducir una URL Oozie.
Workflow Properties	Campo para introducir las propiedades del archivo *Workfile*

Tabla 37 - Descripción del Paso *Oozie Job Executor*. Fuente[(Anónimo, 2012)]

2.2.9 Paso Pig script Executor.

Este paso ejecuta un script escrito en lenguaje "Pig Latin" de Apache Pig en un clúster Hadoop. Todas las entradas de registro correspondientes a esta ejecución de scripts que son generados por Apache Pig se muestran en el registro del PDI (pdf).

La Figura 52 muestra el paso *Pig script executor*.

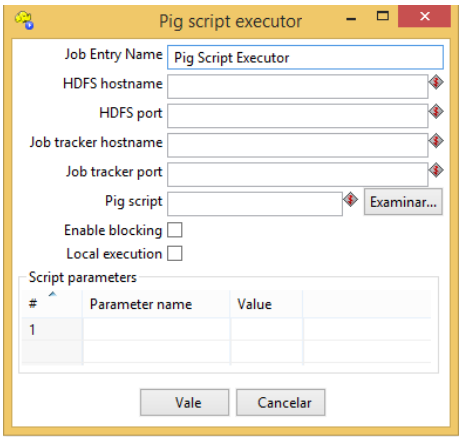


Figura 52-Paso *Pig script executor*

Las opciones que se muestran en el paso *Pig script executor* Figura 52 se explican en la Tabla 38.

Opción	Definición
HDFS hostname	El host de la máquina que opera con un

	sistema de archivos distribuido.
HDFS port	El número deb puerto de la máquina que opera con un sistema de archivos distribuido.
Job tracker hostname	Host del Jobtracker
Job Tracker Port	Puerto del Jobtracker.
Pig script	La ruta de acceso para el script Pig Latin que desea ejecutar.

Tabla 38 - Descripción del Paso *Pig script Executor*. Fuente[(Anónimo, 2012)]

2.3 Ejemplo de una transformación que usa los pasos *big data Hadoop File Input* y *Hadoop File Output*.

Se cuenta con la información referente a los vuelos efectuados por aeronaves desde un aeropuerto, dicha información se encuentra almacenada en el fichero (1987.csv) situado en un clúster Hadoop. El dato referente al tiempo de despegue se encuentra sin separar las horas de los minutos ver Figura 53.

DepTime
741
729
741
729
749
728
728
731

Figura 53 – Hora de despegue de las aeronaves

Es necesario almacenar los datos de horario de despegue de las aeronaves en el fichero *Split.txt* dentro del clúster Hadoop, donde se indique en campos diferentes la hora y los

minutos de partida. Para dar solución al problema anterior se crea la transformación **split_hour_DepTime** de la siguiente manera:

- Añadir el paso Big Data->Hadoop File Input, para extraer los datos del clúster Hadoop.
- Añadir el paso Flow->Java Filter, para dirigir por flujos diferentes cuando el campo DepTime tiene longitud 3 y 4, ver Figura 54.

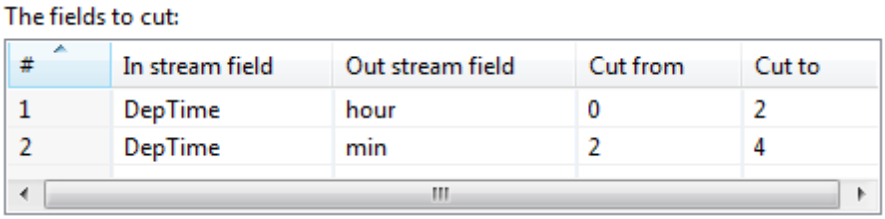


Figura 54-Configuración del paso Java Filter.

- Conectar el paso Hadoop File Input con el paso Java Filter
- Añadir dos pasos Transform->String cut. Uno para cuando el campo DepTime tiene longitud 3 y otro para cuando tiene longitud 4. En la Figura 55 se muestra la configuración de uno de ellos.

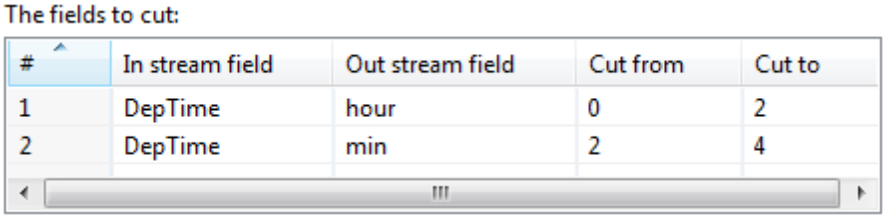


Figura 55-Configuración del paso String Cut cuando la longitud del campo DepTime es 4.

- Conectar el paso Java Filter con los pasos String Cut.
- Añadir el paso Big Data->Hadoop File Output
- Conectar los pasos String Cut con el paso Hadoop File Output

La transformación queda de la siguiente forma, ver Figura 56:

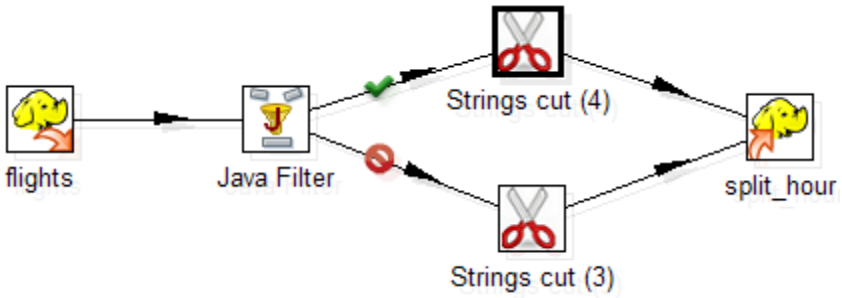


Figura 56-Transformación **split_hour_DepTime**

2.4 Ejemplo de un trabajo que usa un paso Hadoop Copy Files *big data.*

El software PDI se puede utilizar para copiar ficheros desde o hacia el clúster Hadoop de manera rápida y sencilla, con este software se puede dar solución a los siguientes problemas:

- Copiar ficheros desde un directorio local hacia un directorio del clúster Hadoop.
- Copiar ficheros desde un directorio del clúster Hadoop hacia un directorio local.
- Copiar ficheros desde un directorio hacia otro directorio del clúster Hadoop.

En PDI se solucionan los casos anteriores creando un trabajo de la siguiente manera:

- Se inserta el paso *Big Data->Hadoop Copy Files*.

En la configuración del paso *Hadoop Copy Files* se determina cuáles son los directorios o ficheros fuentes y destinos.

La Figura 57 muestra un trabajo donde se siguen los pasos anteriores.

Figura 57- Trabajo para copiar archivos desde o hacia el clúster Hadoop.

2.5 Conclusiones Parciales.

En el presente capítulo se describieron las tecnologías *big data* que se pueden utilizar a través de la interfaz gráfica del PDI asociadas a los diferentes pasos para la implementación de transformaciones y trabajos. En cada caso se explicó y visualizó mediante imágenes la configuración de los pasos *big data*. Finalmente, se diseñó una trasformación y un trabajo que hacen uso de algunos de los pasos *big data* que ofrece el PDI.

CAPITULO 3: APLICACIÓN DE PASOS DEL PDI AL CASO DE ESTUDIO ABCD

En el presente capítulo se explica en que consiste la suite ABCD (Sistema Integrado de Automatización de Bibliotecas y Centros de Documentación), se exponen los componentes principales del formato MARC21 conciso, se mencionan del total de campos de dicho formato los que se seleccionaron para primeramente detectar las anomalías de la información almacenada, para posteriormente implementar mediante el PDI transformaciones que den solución a los errores encontrados, se muestra además el tiempo de ejecución brindado por este software al ejecutarlas.

3.1 Sistema Integrado de Automatización de Bibliotecas y Centros de Documentación (ABCD).

ABCD es un sistema integrado de gestión de bibliotecas (SIGB) de código abierto y libre. La aplicación se utiliza en un entorno cliente/servidor y sus funciones son accesibles en el navegador. ABCD ofrece las funciones de catalogación, registro de publicaciones periódicas, préstamos, gestión de usuarios, adquisiciones, difusión selectiva de la información, generación de estadísticas y catálogo accesible al público. este software es de hecho una modernización de la tecnología CDS/ISIS (Computarized Documentation System - Integrated Set for Information System) originalmente desarrollada y mantenida por la Organización Internacional del Trabajo (OIT) y posteriormente por la Organización de Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO). ABCD viene pre-configurado para trabajar con algunos estándares bibliográficos tales como: MARC21, INTERMARC, UNIMARC, CEPAL, AGRIS, LILACS, entre otros (Smet, 2009).

En nuestro país el sistema ABCD se ha adoptado en diferentes centros bibliotecarios por orientaciones del Ministerio de Educación Superior. En nuestra provincia este sistema es usado por la Universidad Central “Marta Abreu” de Las Villas (UCLV), cuya instalación se encuentra en la biblioteca central de la misma, la cual sirve como servidor principal, permitiendo que mediante la web, las restantes 18 bibliotecas ubicadas en cada facultad de

la universidad, se conecten al mismo para la realización y gestión de diferentes actividades vinculadas a procesos bibliográficos.

Esta suite ABCD usada en la UCLV, cuenta con la base de datos ISIS NoSQL orientada a documentos denominada MARC, que almacena datos semi-estructurados (metadatos bibliográficos de catalogación). Además, cuenta con 10 bases de datos relacionales que manejan datos estructurados sobre: adquisiciones y copias, proveedores, sugerencias, órdenes de compra, usuarios de préstamos, objeto de préstamo, reservaciones, suspensiones y multas, transacciones de préstamo y usuarios del sistema.En la presente investigación se trabaja se trabaja con la base de datos NoSQL MARC.

Formato MARC21

Para la explicación de este formato se utiliza un ejemplo ver Figura 58 de una colección de registros bibliográficos almacenado en MongoDB y visualizado mediante la herramienta multiplataforma para la administración gráfica bases de datos Robomongo.

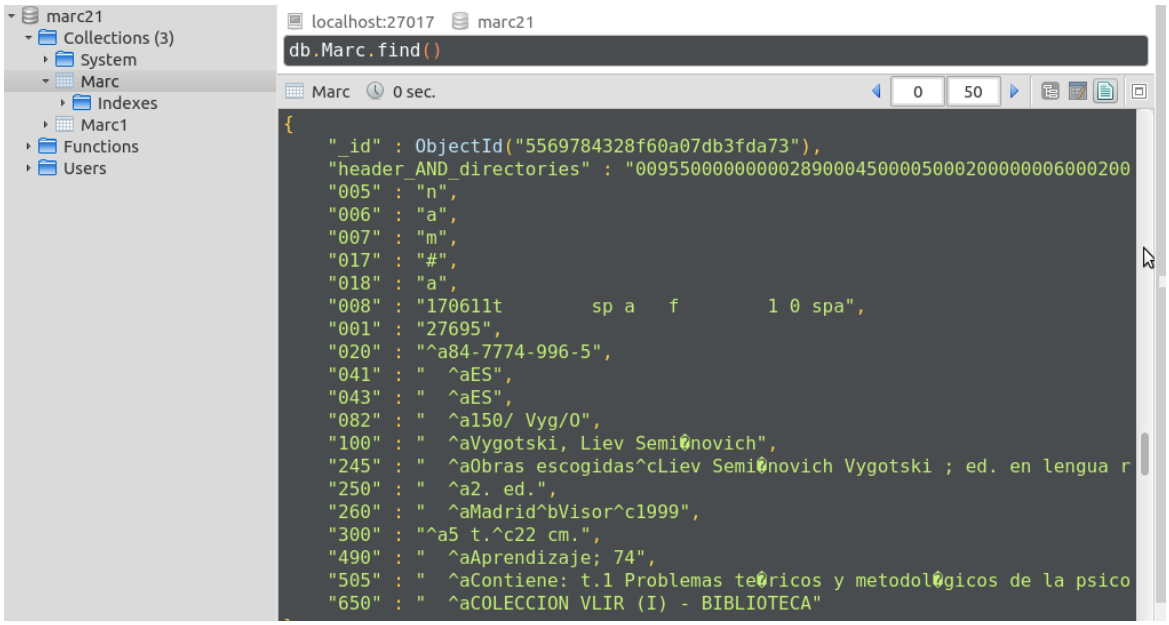


Figura 58 - Figura - Registros bibliográfico en formato MARC21

Un registro bibliográfico en formato MARC21 está compuesto por tres elementos fundamentales:

- Cabecera: Un campo fijo que comprende las primeras 24 posiciones (00-23) de cada registro y suministra información para el procesamiento del registro.
- Directorios: Una serie de entradas que corresponden a la etiqueta, la longitud y el punto de inicio de cada campo variable dentro de un registro.
- Campos: Cada registro bibliográfico se divide en unidades lógicas llamadas campos (número de tres dígitos denominado etiqueta). Los campos se clasifican en campos de longitud fija y campos de longitud variable además estos se subdividen en uno o varios sub-campos.

El formato MARC21 tiene un total de 1000 campos, en este trabajo de diploma se utiliza el formato reducido de MARC21 con un total de 205 campos. De los cuales se seleccionan para trabajar los siguientes:

- Campo 041: referido al código del idioma ver Tabla 39.

041	**CÓDIGO DE IDIOMA**
^a	Código de idioma para texto, pista de sonido o título independiente.
^b	Código de idioma del sumario o resumen
^d	Código de idioma del texto cantado o hablado
^e	Código de idioma del libreto
^f	Código de idioma de la tabla de contenido
^g	Código de idioma acompañado de otro tipo de material que no sea libreto

^h	Código de idioma de la versión original y/o traducciones intermedias del texto
^j	Código de idioma de subtítulos o leyenda o epígrafe
^6	Enlace
^8	Vínculo de campo y número de secuencia

Tabla 39 – Campo 041.

- Campo 260: publicación, distribución ver Tabla 40.

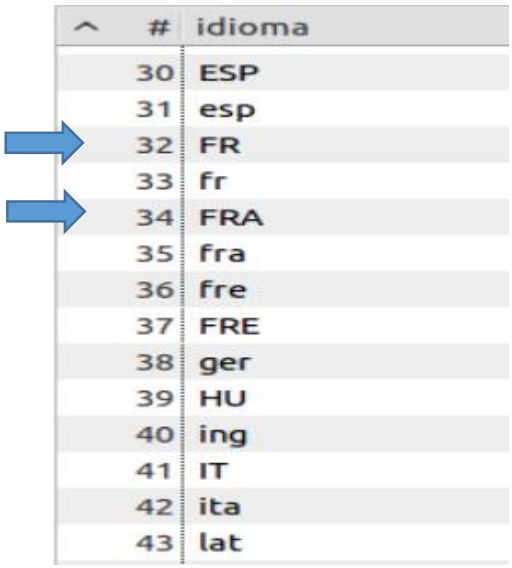
260	**PUBLICACIÓN, DISTRIBUCIÓN, ETC.**
^a	Lugar de publicación, distribución, etc.
^b	Nombre del editor, distribuidor, etc.
^c	Año de publicación, distribución, etc.
^e	Lugar de fabricación
^f	Nombre del fabricante
^g	Año de impresión
^3	Especificación de materiales
^6	Enlace
^8	Vínculo de campo y número de secuencia

Tabla 40 – Campo 260

3.2 Problemas de calidad de datos detectados en los campos 041 y 260.

Haciendo un análisis de los campos seleccionados se detectaron anomalías en la información almacenada como:

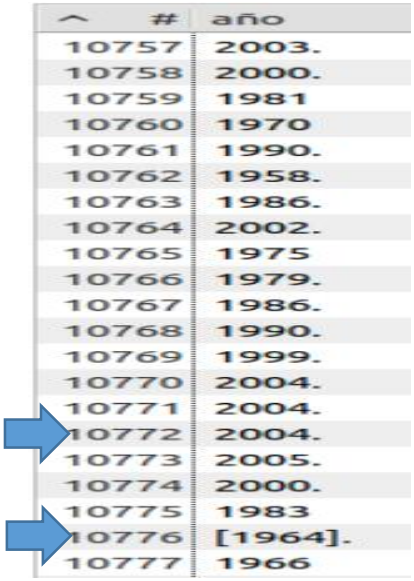
- Errores sintácticos específicamente valores no estandarizados ejemplo, haciendo una vista previa del campo 041 desde el Pentaho Data Integration (PDI) se observa que se hacen referencias a un mismo idioma con diferentes siglas ver Figura 59.



#	idioma
30	ESP
31	esp
32	FR
33	fr
34	FRA
35	fra
36	fre
37	FRE
38	ger
39	HU
40	ing
41	IT
42	ita
43	lat

Figura 59- Campo 041

- Datos incorrectos que presentan caracteres indebidos ejemplo, en el campo 260 sub-campo (c) se observa que la información del año de publicación viene acompañada de corchetes, punto y otros caracteres especiales ver Figura 60.



#	año
10757	2003.
10758	2000.
10759	1981
10760	1970
10761	1990.
10762	1958.
10763	1986.
10764	2002.
10765	1975
10766	1979.
10767	1986.
10768	1990.
10769	1999.
10770	2004.
10771	2004.
10772	2004.
10773	2005.
10774	2000.
10775	1983
10776	[1964].
10777	1966

Figura 60- Subcampo c (año) del campo 260

3.3 Solución de los problemas con el PDI.

El ecosistema Hadoop ofrece una gran variedad de herramientas para el tratamiento de grandes volúmenes de datos como se explicó en el epígrafe Hadoop Common. Pero de este conjunto de herramientas no se tiene conocimiento de alguna que maneja la integración y limpieza de datos.

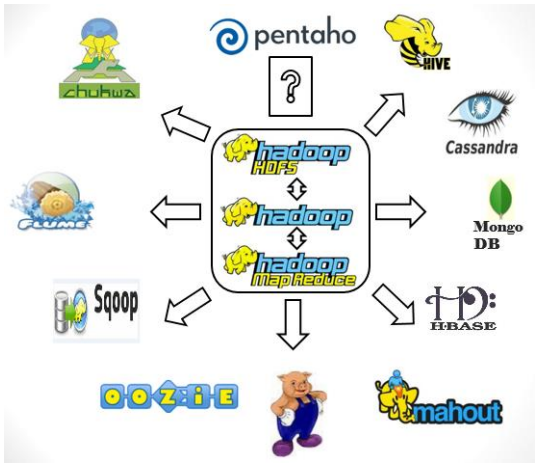
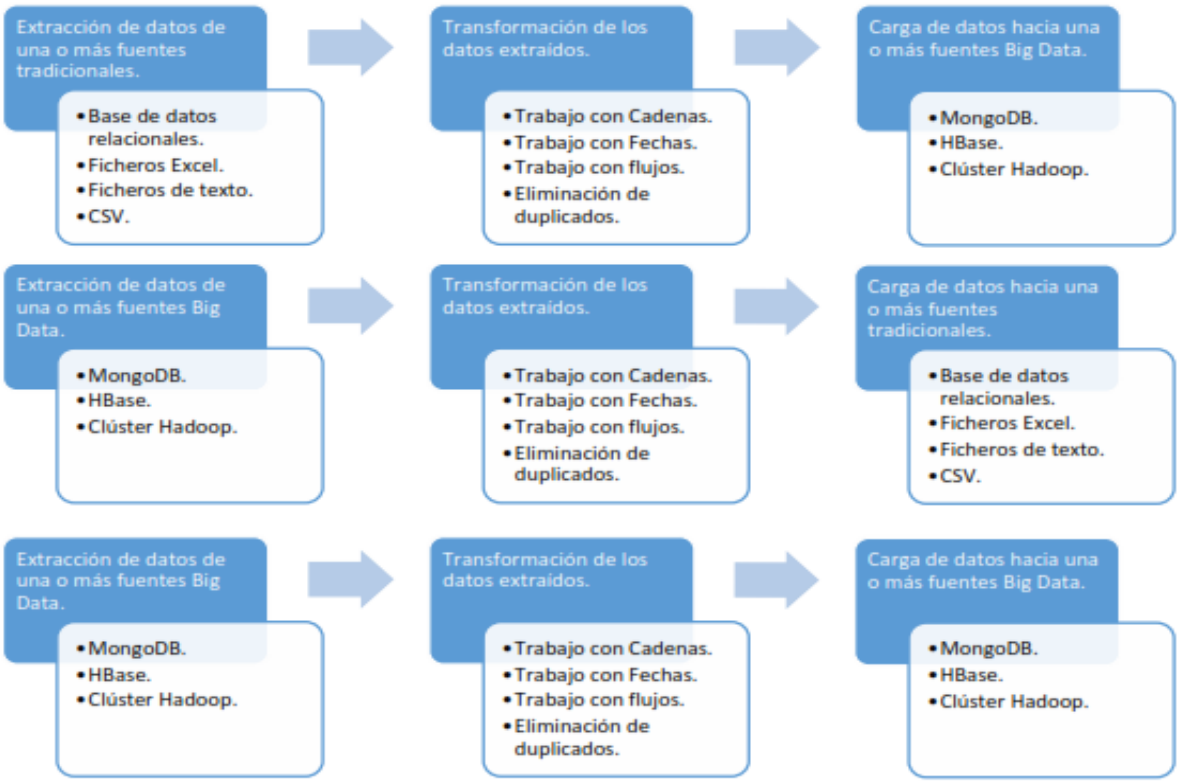


Figura 61 – Ecosistema Hadoop

Ante esta problemática se decide trabajar con el PDI aprovechando las opciones que este brinda para el procesamiento de datos *big data* mediante sus paso de limpieza e integración de datos.

Mediante el PDI se pude extraer los datos de fuentes tradicionales o relacionadas con *big data*, transformarlos o integrarlos y luego cargarlos en fuentes tradicionales o relacionadas con *big data* ver Figura *62*



*F*igura 62 – Procesos ETL utilizando tecnologías *big data*

3.3.1 Valores no estandarizados.

Para dar solución a la no estandarización de datos se sigue el siguiente modo de ejecución en la implementación de transformaciones en el PDI ver Figura 63 :

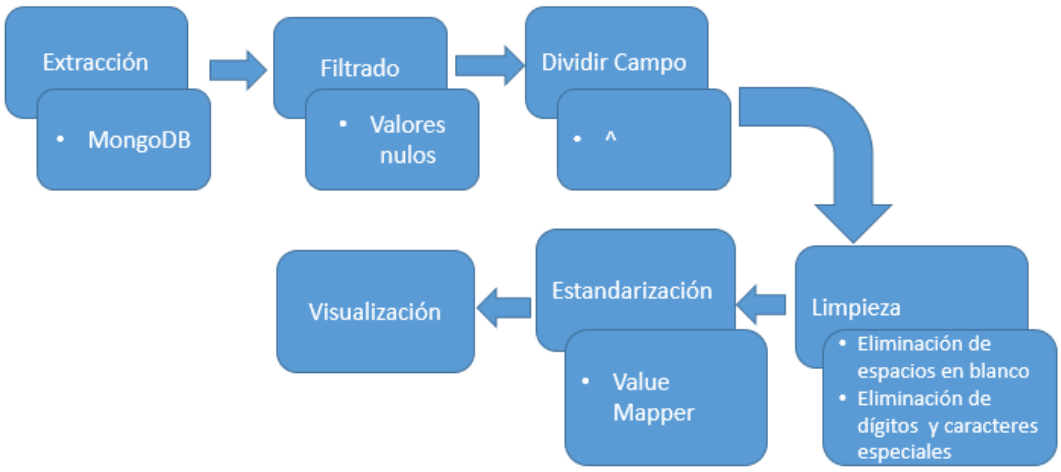


Figura 63- Proceso ETL para resolver problemas de estandarización

De acuerdo al diagrama Figura 63 se implementa la transformación idioma Figura 64 en la cual se chequea la existencia de campos nulos, se crean nuevos campos mediante el paso *Split field*, uno para cada sub-campo del campo 041. Cada sub-campo es procesado por la sub-transformación *estandarización_dioma* Figura 65, finalmente el procesamiento de datos es mostrado en la transformación idioma por el paso *visualizar*.

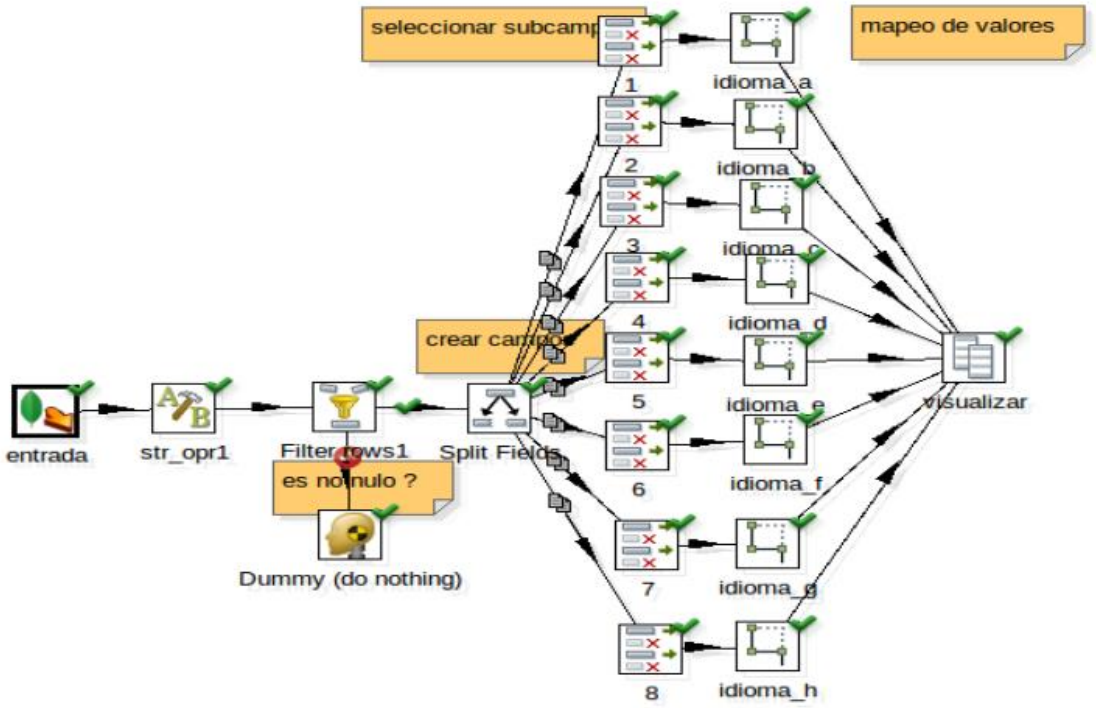


Figura 64-Transformación *idioma*

En la sub-transformación *estandarización_idioma* Figura 65 se eliminan espacios en blancos con el paso str_op1 y se logra la estandarización de los datos mediante el uso del paso Value Mapper donde se realiza un mapeo por el iso 639-1 que contiene 204 códigos de dos letras usados para identificar los principales idiomas del mundo.

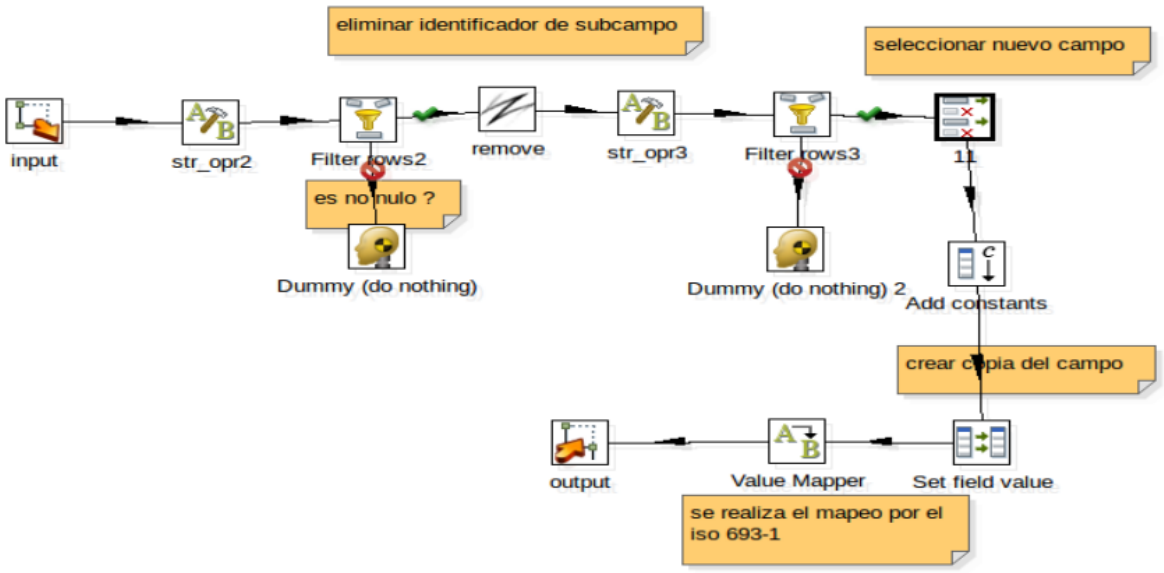


Figura 65- Sub-transformación *estandarización_idioma*

En la Tabla 41 se observa las estadísticas ofrecidas por el PDI vinculadas al tiempo de ejecución de la transformación idioma.

Nombre del Paso	**Tiempo**
Entrada	2.9ms
Str_opr1	2.9ms
Filter rows 1	2.9ms
Split Fields	2.9ms
8	2.9ms
1	2.9ms

4	2.9ms
Idioma_d	3.0ms
Dummy (do nothing)	2.9ms
2	2.9ms
6	2.9ms
Idioma_j	3.0ms
Idioma_f	3.0ms
Idioma_h	3.0ms
Idioma_a	3.0ms
Idioma_b	3.0ms
3	2.9ms
5	2.9ms
7	2.9ms
Idioma_c	3.0ms
Idioma_g	3.0ms
Idioma_e	3.0ms
Idioma_i	3.0ms
Visualizar	3.1ms

Tabla 41- Tiempo se ejecución de la transformación *idioma*

3.3.2 Datos incorrectos que presentan caracteres indebidos.

Para eliminar los datos incorrectos que presentan caracteres indebidos se sigue el siguiente modo de ejecución en la implementación de transformaciones en el PDI ver Figura 66.

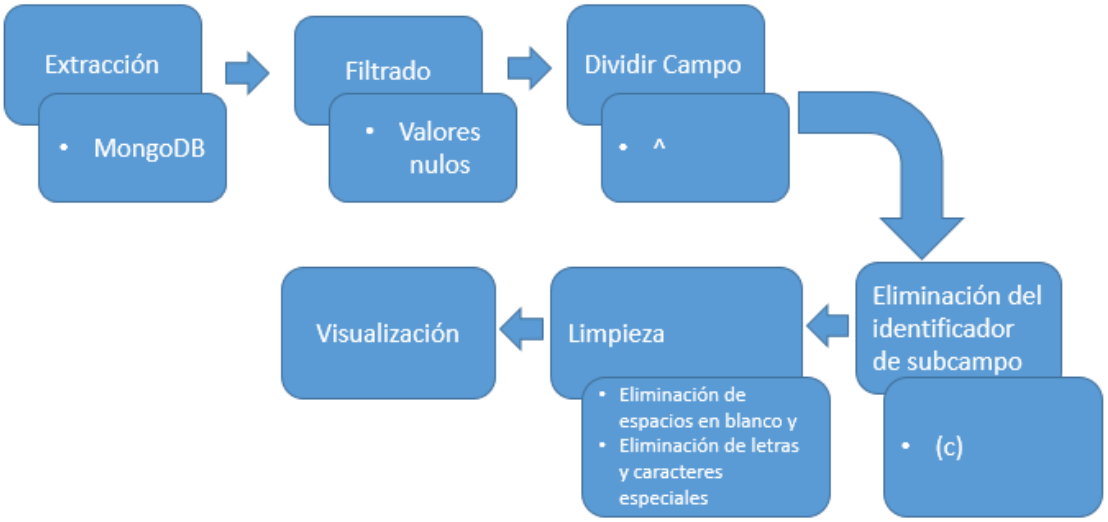


Figura 66 – Proceso ETL para eliminar datos incorrectos que presentan caracteres indebidos

De acuerdo al diagrama Figura 66 se implementa la transformación *Limpieza_año* en la que se elimina el identificador del sub-campo mediante el paso *remove id*, el uso del paso str_opr logra la limpieza de los años eliminando todo lo que no sea dígitos ver Figura 67.

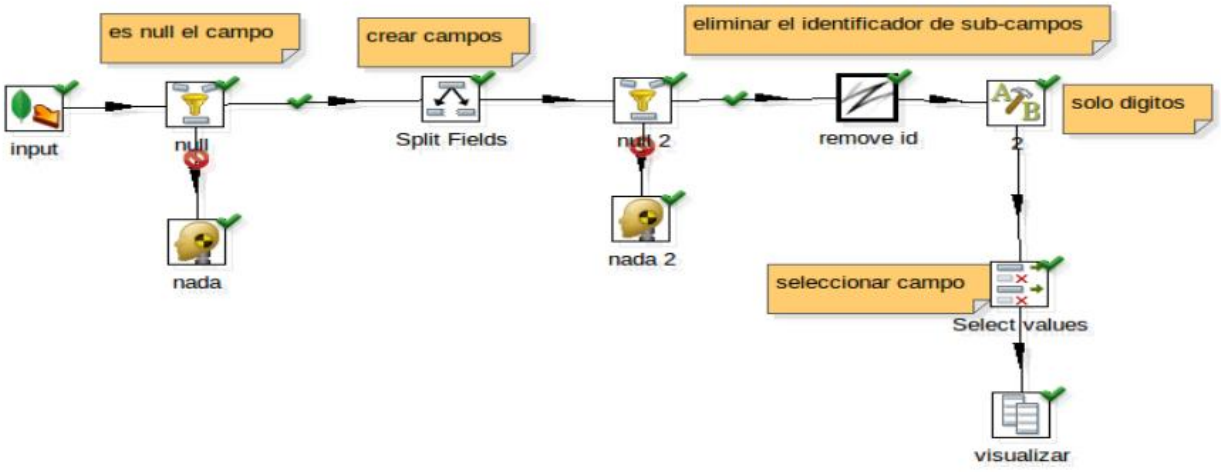


Figura 67- Transformación Limpieza_año

En la Tabla 42 se observa las estadísticas ofrecidas por el PDI vinculadas al tiempo de ejecución de la transformación *Limpieza_año.*

Nombre del paso	Tiempo
input	1.2s
null	1.2s
Split Fields	1.2s
Null 2	1.2s
remov ind	1.2s
nada 2	1.2s
Str_opr	1.2s
Select value	1.2s

nada	1.2s
visualizar	1.2s

Tabla 42-Tiempo de ejecución de la transformación *Limpieza_año*

3.4 Conclusiones parciales.

En este capítulo se explicó en que consiste la suite ABCD, los componentes principales del formato MARC21 conciso, se indicó del total de campos de dicho formato, los que se seleccionaron, para primeramente detectar las anomalías de la información almacenada en estos campos, posteriormente se implementaron transformaciones mediante el PDI, que corregían los errores encontrados, además se mostró el tiempo de ejecución de cada transformación ejecutada.

Conclusiones.

- Se describieron las tecnologías que se utilizan en cada fase de *big data*, haciendo énfasis en las que pertenecen al ecosistema Hadoop.
- Se caracterizaron los pasos del PDI relacionados con *big data* tanto en las transformaciones como en los trabajos.
- Se identificaron algunos problemas de calidad de datos referentes a la no estandarización de valores y datos incorrectos que presentan caracteres indebidos en los campos código del idioma (041) y publicación, distribución (260) del caso de estudio ABCD.
- Se implementaron transformaciones donde se ejemplificó el uso de *big data* en el PDI y se utilizan pasos de limpieza de datos para resolver los problemas de calidad de datos detectados en la suite ABCD.

Recomendaciones.

- Extender el análisis de los problemas de calidad de datos referentes a la no estandarización de valores y datos incorrectos que presentan caracteres indebidos a otros campos del formato MARC21.
- Identificar otros problemas de calidad de datos en la base de datos MARC del ABCD que puedan ser resueltos con el PDI.
- Crear otras transformaciones y trabajos donde se utilicen el resto de los pasos *big data* caracterizados en la presente investigación.

Referencias bibliográficas

ANDERSON, J. C., LEHNARDT, J. & SLATER, N. 2010. *CouchDB: the definitive guide*, " O'Reilly Media, Inc.".

ANÓNIMO 2012. The Pentaho Big Data Guide.

BAHRAMI, M. & SINGHAL, M. 2015. The Role of Cloud Computing Architecture in Big Data. *Information Granularity, Big Data, and Computational Intelligence.* Springer.

BORIS LUBLINSKY, K. T. S., ALEXEY YAKUBOVICH 2013. Professional Hadoop Solutions.

BORTHAKUR, D. 2007. The hadoop distributed file system: Architecture and design. *Hadoop Project Website,* 11**,** 21.

BORTHAKUR, D. 2013. HDFS Architecture Guide. Apache Hadoop.

CASTERS, M., BOUMAN, R. & VAN DONGEN, J. 2010. *Pentaho Kettle solutions: building open source ETL solutions with Pentaho Data Integration*, John Wiley & Sons.

CATTELL, R. 2011. Scalable SQL and NoSQL data stores. *ACM SIGMOD Record,* 39**,** 12-27.

DEAN, J. & GHEMAWAT, S. 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM,* 51**,** 107-113.

HADOOP, D. D. I. P. E. S. A. 2013. Tecnologías de Big Data para

resultados en tiempo casi real.

HASHEM, I. A. T., YAQOOB, I., ANUAR, N. B., MOKHTAR, S., GANI, A. & KHAN, S. U. 2014. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems,* 47**,** 98-115.

HOLMES, A. 2012. *Hadoop in practice*, Manning Publications Co.

JAVLIN. 2011. *Información de Integración de Datos*

Vista rápida en el mundo de los datos [Online]. Available: http://www.dataintegration.info/etl [Accessed 3/12/2012 2012].

KESTELYN, J. 2013. *Meet the Project Founder: Doug Cutting (First in a Series)* [Online]. Available: Meet the Project Founder Doug Cutting (First in a Series) _ Cloudera Engineering Blog.htm.

LANEY, D. 2001. 3-D Data Management: Controlling Data Volume. *Velocity and Variety, META Group Original Research Note*.

LÓPEZ BURGOS, D. G. A., YAUMARA 2013. *Estudio del Pentaho Data Integration en los procesos de integración de datos (ETL).*

NADER, I. J. 2003. *" Sistema de Apoyo Gerencial Universitario "* TESIS DE MAGISTER EN INGENIERÍA DEL SOFTWARE

ROLAND BOUMAN, J. V. D. 2009. Pentaho Solutions:Business Intelligence and Data

Warehousing with Pentaho

and MySQL.

REFERENCIAS BIBLIOGRÁFICAS

ROUSE, M. 2011. big data (Big Data) Retrieved March 11, 2013.

SÁNCHEZ GONZÁLEZ, J., LAURENTIU DULCEANU, A. & SAN GABINO MORENO, D. 2014. easyMahout: entorno de ejecución de algoritmos inteligentes de Mahour para Hadoop y Big Data.

SERRAT MORROS , R. S. P., JORDI. 2013. *BIG DATA- ANÁLISIS DE HERRAMIENTAS Y SOLUCIONES*

SMET, E. 2009. The abc of ABCD : the reference manual : version 1.0.

SYBVEN, P. C. 2013. Available: http://www.InfoPrimer.htm.

WHITE, T. 2012. Hadoop_ The Definitive Guide.

WIESEL, J. 2013. *MongoDB desde Cero: Introducción e Instalación* [Online]. Available: www.mongoDBdesdecero-introduccioneinstalacion.com.