Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Modelado conceptual e implementación de un proceso de carga de información genómica basado en tecnologías ETL

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Jacobo Martínez Crespín

Tutor: Oscar Pastor López

Co-tutor: Francisco Valverde Giromé

2015/2016

Resumen

Este proyecto ha sido desarrollado en el ámbito de los sistemas y tecnologías de la información, mediante estrategias propias de “Business Intelligence” y “Data Management”. Se realizará un estudio sobre las herramientas ETL, así como de las distintas tecnologías de base de datos con el fin de seleccionar el conjunto de herramientas y tecnologías que mejor se adapten a las necesidades y naturaleza de los datos sobre los que se va a aplicar el proceso ETL. El proyecto se ha realizado en colaboración con el centro de investigación PROS (Research Center on Software Production Methods), perteneciente al DSIC (Departamento de Sistemas Informáticos y Computación) de la UPV (Universitat Politècnica de València), el cual ha ofrecido tanto soporte como el modelo conceptual del sistema de información sobre el cual se ha realizado todo el proceso.

Se alinearán los objetivos y requerimientos en base al modelo conceptual, así como un análisis tanto de diseño y modelado de datos como equivalencias entre modelos conceptuales, estableciendo estas equivalencias como marco referencial a la hora de establecer e implementar las reglas de negocios a través de las cuales los requerimientos serán tácitamente implantados sobre la capa lógica y física del modelo. Por último, una vez el proceso ETL haya finalizado se evaluarán y analizarán los resultados tanto a nivel de coherencia con respecto a los datos de entrada así como a nivel de rendimiento entre las tecnologías de base de datos empleadas para distintas tallas del problema.

Palabras clave: ETL, base de datos, Sistemas de información, Business Intelligence, Modelo conceptual.

Abstract

This Project has been developed in the information systems and technologies field using Business Intelligence and Data Management strategies. A developed study about ETL tools, within distinct database technologies as well, with the objective of choosing the proper tools and technologies set which could be fixed better to the needing and nature of the data on which the ETL process will be applied. This project has been done with the support of the PROS (Research Center on Software Production Methods) research group belonging to DSIC (Departamento de Sistemas Informáticos y Computación) of UPV (Universitat Politècnica de València), which has offered both technical support and information system conceptual model on which the whole process has been realized.

Objectives and requirements will be aligned based on conceptual model, as well as both data analysis designing and modelling, and equivalences between conceptual models, establishing these equivalences as a reference at the time of setting up, and implementing the business rules through which the requirements will be tacitly implanted on the logic and physic layer of the model. Finally, when the ETL process is finished, the results will be evaluated and analyzed both coherence and performance level between database technologies.

Keywords: ETL, databases, information systems, Business Intelligence, Conceptual model.

Índice

Índice de Ilustraciones

1. Introducción

1.1. Presentación

Hoy en día la gestión de los sistemas de información se ha convertido en muchas organizaciones como un modelo de negocio, así como, seguir unas buenas prácticas y praxis el éxito en el mercado de TI (**T**ecnologías de la **I**nformación). En la actualidad diversas corrientes, entornos y paradigmas han desafiado a los modelos clásicos de SGBD (**S**istemas de **G**estión de **B**ases de **D**atos).

En el ámbito de la genómica los sistemas de información cada vez son más complejos y de mayor volumen. Actualmente los sistemas de información genómicos forman parte del fenómeno conocido como "*Big Data"* donde el volumen de datos supera la capacidad del software convencional para ser extraídos, administrados y procesados en un tiempo razonable. Por otro lado, no hay una única fuente de datos que sea capaz de proporcionar toda la información que un genetista requiera para elaborar rigurosos análisis, por ello es necesario incorporar datos de distintas fuentes para formar un escenario o sistema que pueda apoyar a los genetista en sus estudios e investigaciones. El hecho de utilizar numerosas fuentes datos conlleva a tener datos con estructuras diversas o en muchos casos sin ningún tipo de estructura.

Para los SGBD convencionales el escenario previamente descrito compromete la estabilidad y el rendimiento, por lo que muchas organizaciones buscaron alternativas tecnológicas para poder llevar a cabo sus servicios garantizando unos mínimos de calidad. En la actualidad existen numerosas herramientas y tecnologías que han tratado esta problemática como son las bases de datos NoSQL, "*B*usiness Intelligence", "*MapReduce"*, etc…

Existen herramientas conocidas como ETL (***E**xtract, **T**ransform, **L**oad*) con el objetivo de proporcionar los componentes software necesarios para poder extraer, validar, limpiar, reformatear y cargar grandes cantidades de información desde múltiples fuentes en otro sistema operacional que apoye al proceso de negocio de la organización. Estas herramientas son consideradas como parte de lo que se conoce como "*Business Intellegince"*, por lo que aparte de los objetivos tecnológicos tiene una serie de objetivos alineados con las estrategias de negocio. Estos objetivos son los siguientes:

- Accesibilidad de la información, garantizando a los usuarios el acceso a esta con total independencia de la fuente de origen.
- Apoyo en la toma de decisiones, dotando a los usuarios de herramientas de análisis que les permite interactuar solo con aquellos datos que ellos necesiten.
- Orientación al usuario final, permitiendo al usuario hacer uso de estas herramientas sin necesidad de tener conocimientos técnicos específicos.

Para que las organizaciones puedan sacar el máximo rendimiento de estas herramientas es totalmente necesario que haya un alineamiento entre las TIC y el modelo de negocio. La única forma de conseguir este alineamiento es utilizar una

arquitectura de modelado de datos que permita propagar los requerimientos a través de todas las capas de modelado.

1.2. Objetivos

EL objetivo principal constará de tres fases, la primera de ella será seleccionar una herramienta ETL adecuada para la creación de un sistema de información de datos genómicos que extraiga e integre los datos de Clinvar. Por otro lado, se seleccionará la tecnología de bases de datos NoSQL que mejor se adapte para la carga de los datos. Estos datos pertenecen a un gran archivo público que almacena información que vincula variaciones humanas y fenotipos, proporcionados por el NBCI (**N**ational **C**enter for **B**iotechnology **I**nformation). El formato utilizado es VCF (**V**ariant **C**all **F**ormat), formato de texto que se usa en Bioinformática para almacenar variaciones de la secuencia de genes y su información desarrollado de los grandes proyectos de secuenciación del ADN genotipado, como el proyecto "1000 Genomas". Dentro de los formatos convencionales, podemos asociar el formato VCF a TSV (***T**ab-**S**eparated **V**alues*).

La segunda fase, consistirá de un análisis de la arquitectura de modelado de datos basada en capas con el fin de establecer un marco referencial a la hora de llevar a cabo cualquier proyecto en el ámbito de las TIC como el que nos ocupa en este proyecto, presentando el modelado conceptual como elemento indispensable para establecer el alineamiento entre los requerimientos de la organización y los sistemas de información. Por otro lado se analizarán los modelos conceptuales de referencia con la finalidad de establecer similitudes y diferencias a todos los niveles.

La tercera fase constará en primer lugar de un análisis y estudio sobre los resultados obtenidos tras la segunda fase. Este análisis definirá claramente las equivalencias entre modelos y el conjunto de transformaciones que serán necesarias. Posteriormente se implementará el proceso ETL llevando estas transformaciones hasta el nivel físico de las bases de datos. Además de la base de datos NoSQL se utilizará una base de datos relacional, en este caso PostgreSQL ya que el centro de investigación PROS ha proporcionado un esquema bajo esta tecnología del modelo conceptual proporcionado.

Por último se hará una evaluación de los resultados obtenidos tras la carga para verificar la coherencia de las bases de datos y el rendimiento de cada tecnología de bases de datos empleada así como el comportamiento en sistemas de información genómicos.

1.3. Motivación

Este proyecto nació por la inquietud de aplicar conocimientos avanzados de ingeniería de sistemas para mejorar aquellos procesos de negocio que precisen de una transferencia masiva de información entre diversos sistemas. Tomando como referencia un modelo conceptual del sistema información a transferir, se dota de una mayor abstracción y claridad a los datos y por consiguiente una simplificación de las reglas de transformación que aplicaremos para cargar nuestros datos en la base de datos NoSQL.

Con lo cual se convierte en un proyecto muy ambicioso, queriendo aplicar técnicas de modelado de datos avanzadas en sistemas de información *Big Data* en el ámbito del *Business Intelligence*.

Por otro lado, cabe destacar el auge de las disciplinas anteriormente nombradas en los años 2010-2016, lo cual aporta un valor añadido de cara a la posible integración de la metodología aplicada en el ámbito empresarial actual.

1.4. Estructura del Trabajo

Con la finalidad de alcanzar los objetivos previamente definidos, el trabajo de fin de master se compone tal y como se va a describir a continuación. Se comenzará realizando un breve preámbulo, justificando la realización de este proyecto así como la clara y concisa definición de los objetivos y motivación de este.

A continuación, se realiza un estado del arte sobre dos temáticas, comenzando en primer lugar con herramientas ETL y abordando por último las bases de datos NoSQL. Para cerrar los capítulos más teóricos se describirá minuciosamente los procesos involucrados en la aplicación de una herramienta ETL.

Una vez finalizado este marco teórico, se irán atendiendo cada uno de los procesos correspondientes al ETL: extracción, transformaciones y carga. En primer lugar, se realizará el proceso para una base de datos SQL y posteriormente para una base de datos NoSQL.

Para finalizar el proyecto una vez concluido el proceso, se analizarán los datos y se comprobará si realmente el rendimiento ha sido mejorado utilizando tecnología NoSQL y sobre todo si el uso de un modelo conceptual como punto de partida agiliza y conlleva a mejores resultados en cuanto a tratamiento y gestión de los datos.

Por último, se incluirán algunos anexos que complementen el proyecto desarrollado con el fin de mostrar información adicional que pueda ser interesante o de ayuda al lector para comprender algunos aspectos desarrollados en la memoria.

2. Estado del Arte

2.1. Herramientas ETL

En esta sección analizaremos el amplio mercado de los ETL para hallar la herramienta más adecuada para conseguir nuestros objetivos.

2.1.1. Software comercial vs software open source

Dentro del mercado de los ETL podemos encontrar un número considerable de herramientas. Se ha comenzado estudiando y analizando las características de herramientas comerciales y herramientas *open source*. Para la realización de este proyecto hemos apostado por el uso de herramientas *open source* ya que en el ámbito y entorno de desarrollo en el que nos encontramos estas se adecúan más.

El uso de herramientas de *open source* aporta ventajas en aspectos como el coste y riesgo así como en el despliegue de la aplicación resultante. Algunas de las herramientas de *open source* tienen un coste gratuito, ofertando versiones del producto *premium*, soporte técnico o *training* por un determinado precio, en el resto de los casos las herramientas tienen un coste asociado más luego algunos suplementos para adquirir servicios adicionales.

En cuanto a los riesgos, en numerosas ocasiones al final el proceso no se ha consiguido el resultado esperado por el cliente o bien el software producido queda muy limitado al pago de licencias y no ha sido totalmente pulido, mediante el uso de herramientas *open source* el riesgo es mucho menor, además, el producto no está tan sujeto a licencias o ampliaciones del producto que ofrecen como base.

Por último, en el caso del despliegue, las herramientas comerciales suelen requerir algún tipo de infraestructura adicional o simplemente unos requisitos mayores, por lo que es una gran ventaja en herramientas de *open source,* ya que estas no suelen requerir grandes prestaciones en el sistema y aparte son menos dependientes a nivel de hardware y software.

Cabe destacar, que las herramientas comerciales tienen claras ventajas sobre aquellas de *open source* en términos de *Data quality & Monitoring*. Referente a *Data Quality,* el software *open source* ofrece algunas funcionalidades al respecto, en cambio las herramientas comerciales suelen incluir software adicional exclusivo para *Data quality*. Algo similar ocurre en el caso de las herramientas de *debugging* y *monitoring*, siendo para ETL *open source* menos sofisticadas y menos flexibles.

2.1.2. Herramientas ETL open source

Hay una amplia variedad de herramientas ETL *open source*, se ha estudiado cada una de las alternativas y haciendo una primera selección basada en la robustez y rendimiento, estas son las herramientas seleccionadas: Pentaho Data Integrator - Kettle, Talend Open Source Data Integration, Scriptella, Apatar, CloverETL.

Estas herramientas no son todas gratuitas, únicamente Kettle y Talend ofrecen productos completos de forma gratuita, el resto de herramientas ofrecen una *trial version* con lo cual las descartaremos.

No obstante, se incluirán las características básicas de las herramientas descartadas ya que son parte del mercado actual de las herramientas ETL.

Scriptella es una herramienta ETL desarrollada en Java basada en la ejecución de scripts XML (e**X**tensible **M**arkup **L**anguage). El uso de Scriptella no exige el dominio de lenguajes adicionales pero sí que es cierto que acepta lenguajes como SQL para dotar al usuario de una mayor flexibilidad a la hora de realizar transformaciones complejas. Scriptella es utilizada en entornos donde se requiera migraciones de bases de datos, operaciones "*Cross-database*" así como actualizaciones automáticas de esquemas de bases de datos. No posee interfaz gráfica, sino que la herramienta interpreta el script XML generado como en el ejemplo de a continuación:

```
<!DOCTYPE etl SYSTEM "http://scriptella.javaforge.com/dtd/etl.dtd">
<etl>
    <connection driver="$driver" url="$url" user="$user" password="$password"/>
    <script>
        <include href="PATH_TO_YOUR_SCRIPT.sql"/>
        -- And/or directly insert SQL statements here
    </script>
</etl>
```

Algunas de las características esenciales de esta herramienta son:

- Soporte de múltiples fuentes de datos.
- Soporte a "*cross-database ETL scripts*" usando elementos <dialect>.
- Gran rendimiento mediante un uso mínimo de memoria.
- Gestión de errores vía elementos <onerror>
- SPI (**S**ervice **P**rovider **I**nterface) para la interoperabilidad con fuentes de datos no JDBC (**J**ava **D**ata**B**ase **C**onnectivity).

Apatar es una herramienta ETL así como un software de integración datos. Utiliza J2EE (**J**ava **E**nterprise **E**dition) como lenguaje de programación, así como SQL, SOAP (**S**imple **O**bject **A**ccess **P**rotocol)/XML como lenguaje de consulta. Apatar utiliza "Eclipse" como herramienta de desarrollo y soporta los principales sistemas operativos.

CloverETL, es una herramienta ETL basada en Java. Puede funcionar "*Standalone*" como aplicación o bien a mediante línea de comandos o bien como herramienta integrada en otras aplicaciones como librería de Java, siendo eclipse el entorno más utilizado en este último caso.

El hecho de poderse integrar mediante el uso de librerías facilita la compatibilidad con cualquier sistema operativo donde pueda instalarse la plataforma de Java. Los recursos están almacenados en formato XML, incluyendo tanto los metadatos como los componentes que ofrece la herramienta. Cabe destacar que el lenguaje de programación es CTL (**C**loverETL **T**ransformation **L**anguage).

En cuestiones de rendimiento fomenta el (*"Massively Parallel Computing"*) ofreciendo elementos de conexión para el uso de múltiples CPUs, así como la ejecución en clusters de computadores.

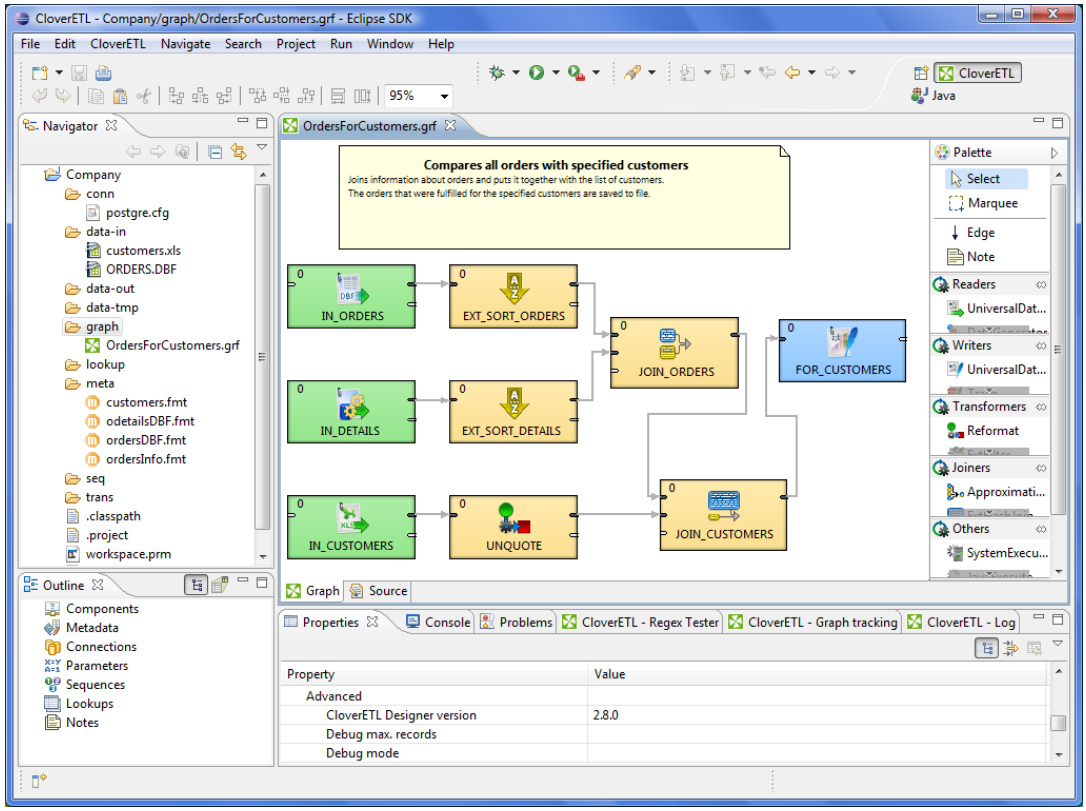


Ilustración 1 - CloverETL, entorno Eclipse

2.1.3. Pentaho Data Integration vs Talend Open Source Data Integrator

Ambas herramientas son de gran referencia en el mercado ETL *open source*. A continuación haremos un breve resumen, mostrando las características más importantes de ambas herramientas.

Pentaho Data Integration (PDI, conocida como *Kettle*) es una herramienta desarrollada por la empresa Pentaho responsable de los procesos *Extract, Transform and Load* (ETL). Su principal uso es en entornos integrados con *data warehouse* pero también es utilizado en tareas como:

- Migración de datos entre aplicaciones y/o bases de datos.
- Exportar datos desde una base de datos a un fichero.
- *Big data*.
- Filtrado de datos.
- Integración de aplicaciones.

PDI se caracteriza por su intuitiva interfaz gráfica (Spoon). Podríamos decir que PDI es *metadata oriented* permitiéndo generar los distintos procesos sin escribir código, simplemente interactuando con Spoon.

PDI puede ser utilizado como una aplicación *standalone*, o puede ser utilizado a través del *Pentaho Suite*. Es capaz de procesar gran cantidad de datos incluyendo ficheros de texto, *data sheets* y bases de datos comerciales y gratuitas.

Por último, cabe destacar la amplia disponibilidad de componentes para la transformación de datos, realzando la herramienta en aspectos como el *data quality*.

Talend Open Studio (TOS) es una herramienta ETL desarrollada por Talend, basada en Eclipse RCP (**R**ich **C**lient **P**latform). TOS se comporta como un generador de código, generando *scripts* en .java como resultado de las transformaciones realizadas sobre un conjunto de datos. A través de su *framework* puedes acceder a su repositorio de metadatos (ficheros de configuración y definiciones) o a su diseñador gráfico. Para ejecutar el código generado, puedes utilizar su *framework* o realizar un despliegue *standalone,* ejecutando directamente los *scripts* como un fichero .java corriente.

TOS *for Data integration* se utiliza principalmente para:

- Sincronización o replicación de bases de datos.
- ETL sobre *data warehouse*.
- Migración de datos.
- Transformaciones complejas y carga.
- Filtros de datos & *data quality*.
- *Big data*.

Se ha construido una tabla valorando los aspectos que se han considerado más relevantes de cara a la elección de la herramienta definitiva.

	Ease to use	**Support**	**Features**	**Perfomance**	**Deployment**
Pentaho Data Integrator - Kettle	10	10	7	9	8
Talend Open Source Data Integrator	9	9	10	10	10

Ambas herramientas tienen una interfaz amigable, no obstante, en el caso de Kettle se requiere menos tiempo de aprendizaje al inicio.

Ambas herramientas tienen una amplia comunidad de usuarios y ambas empresas ofrecen soporte vía subscripción y a través de servicios de consulta directa. Ambas empresas han tenido un rápido crecimiento debido a sus herramientas, en el caso de Kettle conforme ha ido evolucionando ha ido incluyendo componentes relacionadas con *Business Intelligence*, en cambio, TOS apostó por componentes orientadas a *Data integration, quality & management*.

En cuanto a las componentes de ambos ETL, cabe destacar que TOS es más potente a nivel de tecnología y aparte tiene más variedad de componentes, además, ofrece componentes específicos para bases de datos NoSQL y *Big data*.

En cuanto al rendimiento dependiendo de las componentes que se utilicen, el rendimiento variará en ambas herramientas favoreciendo en el caso de TOS la generación de ficheros .java, a nivel de compresión y despliegue, aportando mejores resultados que los obtenidos mediante la interpretación de los ficheros .xml generados por Kettle.

Por último, a nivel de despliegue TOS funciona como cualquier otra aplicación java, en el caso de Kettle, es necesario al desplegar la aplicación fuera del *framework* importar las librerías propias de Kettle.

Ambas herramientas son totalmente válidas para abordar este proyecto, no obstante se ha optado por Talend, sobre todo por la riqueza de componentes en el ámbito de bases de datos NoSQL y *big data* así como a nivel de rendimiento a la hora de trabajar en entornos *big data*.

2.2. Bases de datos NoSQL

NoSQL o también llamado "no solo SQL", es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado es que no usan no usan SQL como lenguaje principal de consulta. A diferencia de RDBMS estas bases de datos carecen de estructura fija en tablas, además hay operaciones como "JOIN" que no son soportadas. Otro aspecto relevante de estas tecnologías es que no garantizan completamente las características ACID (**A**tomicidad, **C**onsistencia, **A**islamiento, **D**urabilidad), aunque existen refuerzos a través de una capa intermedia como por ejemplo AppScale o CloudTPS u otros sistemas así como Precolator de Google (basado en el sistema BigTable) o Hbase (desarrollado por la Universidad de Waterloo).

Los sistemas de bases de datos NoSQL crecieron con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook, siendo el caso más emblemático la búsqueda de mensajes en la bandeja de entrada de Facebook que albergaba un total de 50TB. Con el crecimiento de la web en tiempo real tanto a nivel de volumen de datos como de usuarios, las RDBMS no eran capaces de afrontar este desafío satisfactoriamente con lo cual se precisaba de una tecnología alternativa capaz de resolver esta problemática. Analizando la naturaleza de los datos, se dieron cuenta que los grandes volúmenes de datos que necesitaban procesar tenían unas estructuras horizontales muy similares, para las cuales las tecnologías NoSQL ofrecían una escalabilidad y rendimiento muy elevado a cambio de pérdida de coherencia e integridad, así como pérdida de flexibilidad en tiempo de ejecución.

2.2.1. RDBMS vs NoSQL

Las RDBMS y bases de datos NoSQL surgieron en eras tecnológicas distintas, respondieron ante demandas completamente distintas. Mientras RDBMS se caracteriza por su robustez y nacieron por la necesidad de actuar como sistema de almacenamiento y gestor de información para grandes empresas. Por otro lado, las bases de datos NoSQL nacieron como respuesta a la demanda tan exigente tras el crecimiento exacerbado o alcista de la web, así como nuevas corrientes tecnológicas relacionadas como *"Big Data"*, *"mobile*", "IoT"(***I**nternet **o**f **T**hings)* y *Cloud Computing*, ofertando tratamiento eficiente de datos tanto estructurados como parcialmente estructurados, computación distribuída y sobre todo una alta disponibilidad (24/7).

Con la llegada del siglo XXI las organizaciones han ido evolucionando y adaptándose a las nuevas tendencias lo cual ha acarreado cambios desapacibles a nivel de los requerimientos y procesos en los que las organizaciones se veían involucrados exigiendo una gran resilencia de las bases de datos. Las RDBMS comenzaron a sufrir grandes achaques en el rendimiento ya que en este aspecto carecía de versatilidad, en cambio las bases de datos NoSQL respondieron y se adaptaron rápidamente soportando tanto cambios en los requerimientos como actualizaciones y la adición de nuevas características. Esto es debido a que las RDBMS se implementan en base a un esquema, lo cual cualquier cambio en los requerimientos provocará cambios sobre el esquema lógico no siendo así en las bases de datos NoSQL caracterizadas por no poseer esquema. A raíz de lo

anteriormente mencionado deriva otra característica esencial y sobre todo cuando se trata de IoT es el hecho de que las bases de datos NoSQL pueda gestionar datos sin estructura o semiestructurados, ya que el esquema de datos no recae sobre la propia base de datos sino sobre la aplicación.

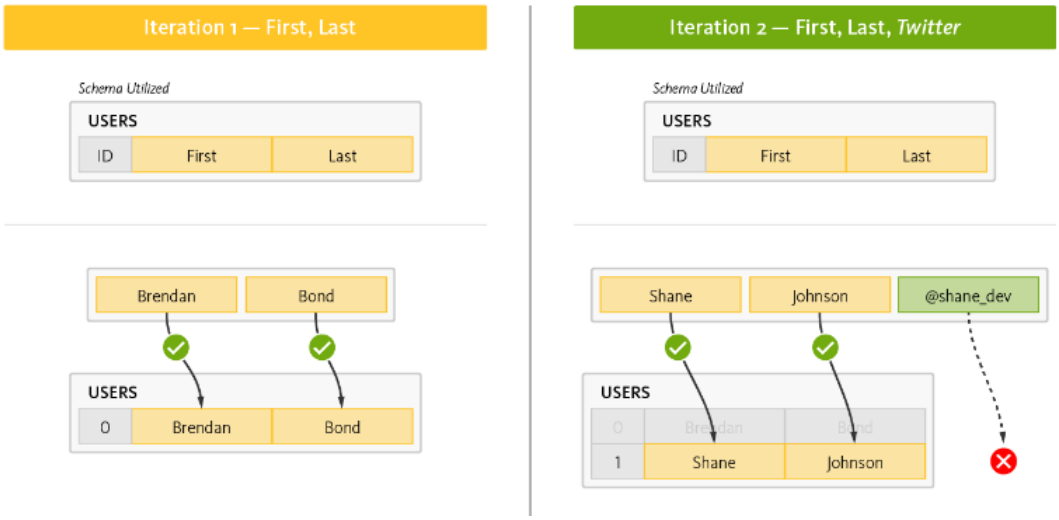


Ilustración 2 - Cambios de estructura en RDBMS

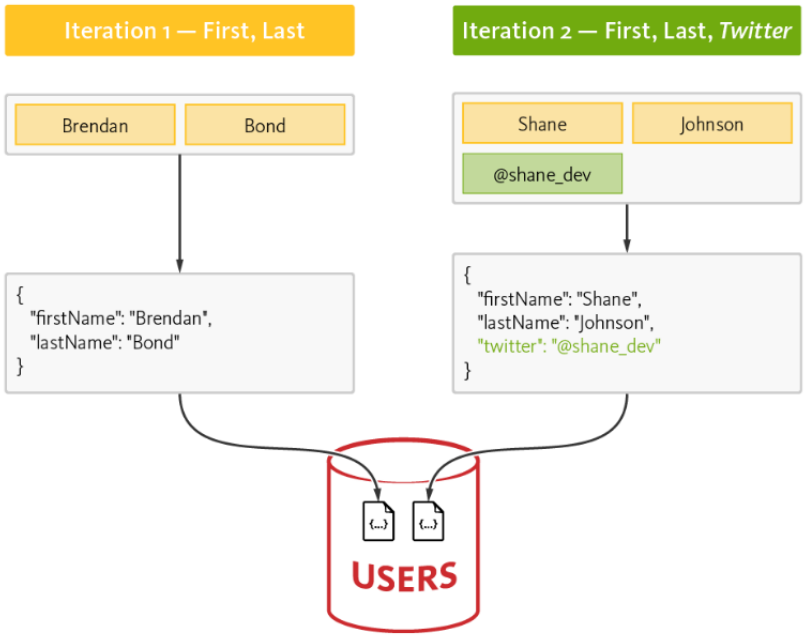


Ilustración 3 - Cambios de estructura en bases de datos NoSQL

Para continuar con el estudio ha sido necesario recurrir al teorema CAP ya que refleja un punto de inflexión entre bases de datos NoSQL y RDBMS.

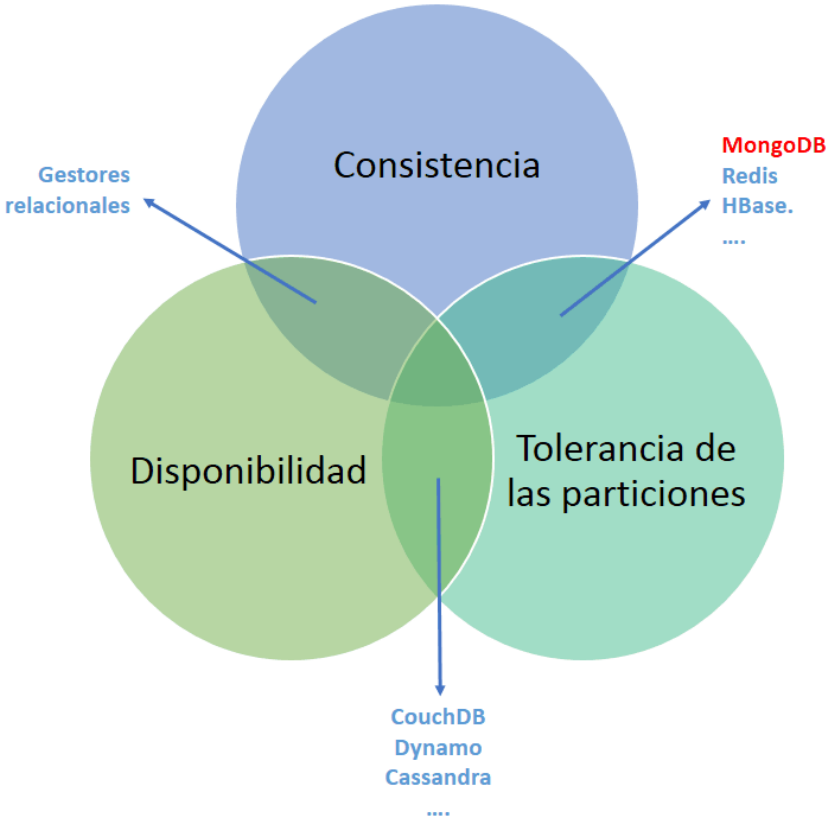


Ilustración 4 - Teorema CAP

Pese a la ilustración de la parte superior, la disponibilidad en las RDBMS en ciertamente más débil con respecto a las bases de datos NoSQL, al igual que la consistencia siempre estará de lado de las RDBMS, ya que como en todos los aspectos en estas bases de datos brinda lo estático y estable. Bien es cierto, que encontramos distintos paradigmas, la diferencia esencial reside en la arquitectura así como en la interacción y relación con los objetos. Era necesario presentar esta característica porque ha condicionado y limitado el uso de RDBMS en el ámbito web donde prima la disponibilidad de los datos, así como la computación ubicua y distribuida. La estructura y escalabilidad vertical de las RDBMS es un punto débil ya que cualquiera de los escenarios debe existir un único servidor físico con lo cual la disponibilidad y la tolerancia a las particiones se ven claramente comprometidas. La consecuencia directa de la escalabilidad en cuestiones de tolerancia y disponibilidad es que cualquier fallo bloquea totalmente el sistema, en cambio, las bases de datos NoSQL al ser "fácilmente" distribuidas y particionadas provoca que tras un fallo otra instancia de la base de datos pueda atender la petición.

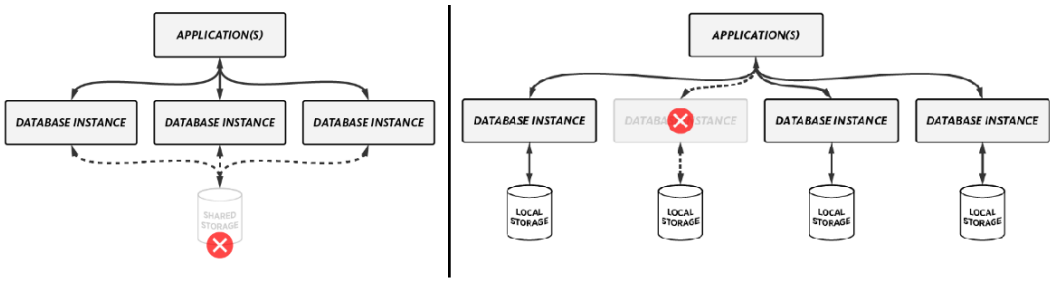


Ilustración 5 - Disponibilidad y Tolerancia en las bases de datos

El hecho de que las bases de datos NoSQL escalen horizontalmente provoca que la inversión en infraestructura sea menor y a su vez que el rendimiento sea mayor en los escenarios actuales. La gráfica siguiente muestra las repercusiones de un consumo de servicios concurrente bajo ambos tipos de base de datos:

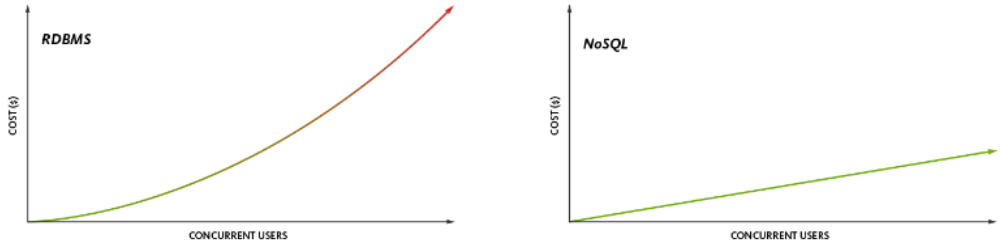


Ilustración 6 - Concurrencia en base de datos

Además de concurrencia, la escalabilidad horizontal es más eficiente ante el almacenamiento y gestión de cantidades masivas de datos.

El hecho de que las bases de datos NoSQL no sigan el modelo ORM (***O**bject-**R**elational **M**apping)* propio de las RDBMS, elimina la sobrecarga producida por el "*marshalling/unmarshalling"* de los objetos. El hecho de que para cada "*query"* se tenga que procesar y mapear cada objeto aparte de la sobrecarga propiamente dicha suele desencadenar en una serie de lecturas inevitables. En el caso de las bases de datos NoSQL no ocurre lo mismo, ya que en una misma estructura pueden alojarse distintos "objetos" (realmente no existe el concepto de objeto) accesibles en una misma petición.

2.2.2. Tecnologías NoSQL

Existen distintas tecnologías e implementaciones de estos sistemas de información, los más representativos son los siguientes:

- **Clave - Valor**: Es el modelo más popular y sencillo dentro de las bases de datos NoSQL. Cada elemento está identificado por una clave única. No es una buena opción si generalmente se precisa actualizar parte de un valor o simplemente realizar consultas. Las bases de datos más conocidas son: Dynamo, MemcacheDB, Redis.

- **Columna:** Este modelo está orientado a *big data* donde es necesario procesamiento masivo de grandes cantidades de información en este caso además, cuando el escenario incluye información completamente distribuida. También se utilizan claves, pero en este caso cada una de ellas apunta a varias columnas, estas columnas a su vez están dispuestas por familias de columnas. Se caracteriza por accesos y búsquedas rápidas. Las bases de datos más conocidas son: Cassandra, BigTable y HBase.

- **Grafos:** Esta tecnología emplea una representación particular, la información se representan como nodos de un grafo y sus relaciones como aristas del mismo. Su uso es más complejo que usando otras tecnologías. Es necesario tener ciertas nociones de teoría de grafos para poder sacar todo el rendimiento a estas estructuras. No obstante, son bases de datos muy flexibles y fácilmente escalables. Las bases de datos conocidas son: Neo4j e infoGrid.

- **Documental:** Este modelo organiza y almacena la información en documentos, utilizando algún tipo de estructura como: JSON (**J**ava**S**cript **O**bject **N**otation) o XML (e**X**tensible **M**arkup **L**anguage). Cada uno de los documentos están compuestos por una serie de registros donde se aloja la información. Permite búsquedas más precisas y exigentes que en el caso de las bases de datos por clave – valor, introduciendo metadatos u otros elementos que apoyan a una mayor eficiencia en el acceso a los datos. Esta tecnología es la más versátil, son utilizadas en numerosos proyectos de muy distintos ámbitos. Las bases de datos más conocidas son: MongoDB y CouchDB.

2.2.3. Bases de datos documentales

Una base de datos documental está constituida por un conjunto de programas que almacenan, recuperan y gestionan datos de documentos o datos de algún modo estructurados. Este tipo de bases de datos constituyen una de las principales subcategorías dentro de las denominadas bases de datos NoSQL. A diferencia de las bases de datos relacionales, estas bases de datos están diseñadas alrededor de una noción abstracta de "Documento".

Las claves son utilizadas para direccionar e identificar documentos y estas son almacenadas a través de un índice en la propia base de datos para facilitar y agilizar la recuperación de la información.

Para poder interactuar con la base de datos más allá de una simple correspondencia <clave, valor (documento)> existe un API para recuperar o consultar de la base de datos siendo ligeramente distinta en cada implementación.

La forma en las que los documentos son organizados depende de la implementación siendo las más conocidas: Colecciones, Etiquetas, Metadatos Ocultos, Jerarquías de directorios. Las implementaciones más destacadas son las siguientes: ArangoDB, CouchDB, Lotus Notes, MongoDB, Apache Cassandra y Redis. Cada una de estas implementaciones difiere en algunos detalles, pero comparten el método de encapsulamiento y codificación de los datos siguiendo un formato estándar.

MongoDB (con origen en *hu**mongo**us*, es decir, enorme) es una base de datos documental bajo un modelo de desarrollo de código abierto en lenguaje C++. El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software "10gen" formando parte de la nueva era de las bases de datos NoSQL. Como ya es sabido (véase sección 2.2) la estructura que siguen estas bases de datos viene dada por documentos tipo JSON, al que en el caso de MongoDB denomina BSON (*Binary* JSON).

Los objetos BSON están constituidos por una lista ordenada de elementos y cada uno de estos mantiene una estructura de este tipo <ID, Tipo, Valor>, el campo "ID" identifica el elemento con un *String*, "Tipo" referencia al tipo de dato almacenado en el siguiente campo, "Valor", contenido del elemento. En comparación a JSON, BSON está diseñado para tener un acceso y almacenamiento más eficiente. Cabe destacar que en casos donde hay elementos largos, hay un campo llamado "tamaño" utilizado para facilitar el escaneo que puede acarrear en un mayor uso de memoria que mediante JSON.

El principal objetivo de MongoDB fue dotar a las organizaciones de una implementación ágil y escalable para crear nuevos tipos de aplicaciones, mejorar la experiencia del cliente, acelerar el tiempo de comercialización y reducir costes.

Una de las características más destacables es la agilidad de estas bases de datos permitiendo a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan. Por otro lado, MongoDB proporciona una experiencia y funcionalidad

muy cercana en comparación con las RDBMS aportando aspectos, tales como índices secundarios, un API sólido que permite interactuar plenamente con la base de datos soportando la búsqueda por campos, consultas de rangos y expresiones regulares y algo muy importante, manteniendo una sólida y estricta consistencia como se puede observar en la ilustración 3 (véase sección 2.2.2).

Hay tres características que definen MongoDB:

1. Escalabilidad: Partiendo de una implantación en un simple servidor hasta grandes arquitecturas complejas en centros multidatos. Aparte puede escalar de forma horizontal usando el concepto de "shard". Los datos son divididos en rangos (basado en la clave de sharding) y distribuidos a través de múltiples shard. Cada shard es un replica set. MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o duplicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware.

2. Rendimiento: Tanto de lectura como en escrituras, destacando y potenciando la computación en memoria. Proporciona un framework de agregación que permite realizar operaciones similares a las que se obtienen con el comando SQL "GROUP BY". Está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los que son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado. Todo este procesado utiliza los índices existentes y se produce en memoria. Asimismo, MongoDB proporciona una función MapReduce puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación. Por otro lado, es posible realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

3. Disponibilidad: La replicación nativa de MongoDB y la automática tolerancia a fallos automática ofrece un servicio pleno y flexibilidad operativa. Soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica set 10. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad. Los secundarios tiene la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

MongoDB también presenta una serie de inconvenientes que realmente son heredados de la propia naturaleza de la tecnología en esta implementación se basa. Los problemas derivan de la no implementación de las propiedades ACID generando que la base de datos no asegure la durabilidad, la integridad, la consistencia y el aislamiento requeridos obligatoriamente en las transacciones:

- Bloquea la base de datos a nivel de documento ante cada operación de escritura. Sólo es posible hacer escrituras concurrentes si estas se realizan en documentos distintos.
- MongoDB Puede retornar cuando todavía no se ha escrito la información en el espacio de almacenamiento permanente,

devolviendo un valor que puede no satisfacer la durabilidad ni la verificabilidad.

- Las lecturas estrictamente consistentes en MongoDB no existen como tal, pudiendo mostrar datos incorrectos tras una consulta.

Llegados a este punto es conveniente establecer las equivalencias entre la terminología y estructuras entre RDBMS y bases de datos NoSQL documentales siendo en el segundo caso MongoDB como sistema de referencia, con la finalidad de comprender el proceso de carga con mayor facilidad (véase sección 4.3.2).

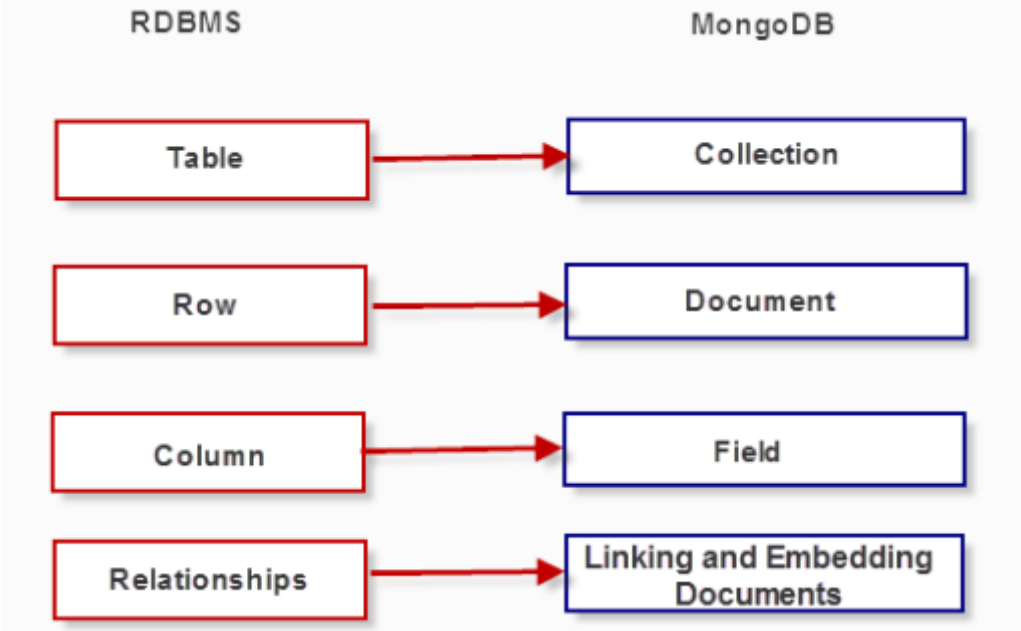


Ilustración 7 - RDBMS to MongoDB

CCabe destacar que el "*mapping*" no es completo, es decir, no son equivalencias exactas. En primer lugar, el hecho de que en MongoDB los datos carezcan de esquema evita muchas de las restricciones que en una RDBMS serían intrínsecas a la propia definición del esquema. A pesar de que a nivel de base de datos MongoDB no posea estructura, si que posee estructura a nivel de "*Collection*" siguiendo el formato BSON (más desarrollado en la sección 2.2.3). A nivel de "*Row*"/"*Document*" los datos de una RDBMS tienen que respetar la definición y restricciones que en el esquema se hayan establecido, en cambio en MongoDB, cada "*Document*" puede tener un esquema distinto, es decir, el elemento con "id = 1" puede tener tres campos y el elemento con "id = 2" puede tener cinco campos con todo tipo de sub-campos a su vez. A nivel de "*Column*"/"*Field*" los datos en las RDBMS deben cumplir con el tipo, longitud y otras especificaciones, no siendo así en el caso de MongoDB donde cada "Field" puede ser de un tipo u otro sin responder a definición alguna así como tomar el valor "null" o directamente no existir en un "Document" determinado.

Aun siendo ciertamente desconcertante lo anteriormente descrito, una de las cosas más llamativas en la tecnología NoSQL es el hecho de que no existan las relaciones, lo cual sería inconcebible en RDBMS. Las relaciones en RDBMS es una de las claves para garantizar la integridad referencial y de esta forma gestionar las claves foráneas entre las distintas tablas estableciendo sólidos y rígidos vínculos entre estas. En el caso de MongoDB, no existen las relaciones, solo un conjunto de colecciones compuestas por documentos. Para establecer de alguna forma "relaciones" entre los datos de una base de datos en MongoDB hay dos opciones:

- Embebiendo en el propio documento aquella estructura que "forme parte" o esté relacionada con el documento en cuestión. Esto puede suponer un problema en el caso de la que base de datos esté continuamente siendo actualizada o bien se trate de una base de datos con un gran volumen de datos.
- "Linkeando" el elemento en cuestión mediante identificadores. En este caso se incluiría el "id" del elemento ajeno, perteneciente a otra colección al elemento en cuestión. Hay que ser precavido ya que este campo añadido es un campo más y carece de toda integridad por defecto, por tanto, recae en la aplicación mantener la coherencia e integridad.

Como es sabido (véase sección 2.2) las bases de datos NoSQL documentales tienen un campo de aplicación extenso, concretamente MongoDB se ha visto integrado en sistemas de Almacenamiento y registro, Comercio Electrónico, Aplicaciones móviles, Proyectos de desarrollo iterativo o ágil, etc... En cuanto a disciplinas, pueden observarse en prácticamente cualquier área de conocimiento. En el ambito de las ciencias de la salud y genómica destacan varios institutos que utilizan mongoDB como base de datos, como pueden ser: "The Mount Sinai Institute for Genomics and Multiscale Biology" o "Seven Bridges Genomics", así como empresas como Gennetech.

Hay ciertos documentos que corroboran el uso de MongoDB en estas organizaciones. A continuación se muestran algunos fragmentos:

"NEW YORK—October 22, 2013—MongoDB today announced that Seven Bridges Genomics, a biotech company based in Cambridge, Mass., and Belgrade, Serbia, selected MongoDB to power data analytics for its next-generation sequencing (NGS) analysis platform. MongoDB enables researchers to collaboratively perform high-throughput genomic sequencing.

Seven Bridges simplifies data analysis for genomics by eliminating the need to manage large-scale computation and storage infrastructures that are expensive to maintain and scale," said Igor Bogicevic, co-founder and CTO of Seven Bridges Genomics. "MongoDB supports our long-term requirements for scalability and it allows us to quickly add features, without forcing us to re-think our model and architecture or invest heavily in re-engineering our infrastructure.

MongoDB has a thriving open source global community with more than 5 million downloads, 100,000 online education registrations, 20,000 user group members and 20,000 MongoDB Days attendees." (MongoDB, Seven Bridge Genomics, 2013).

El siguiente fragmento pertenece a un artículo realizado por cuatro especialistas en el ámbito de los sistemas información médicos. Laurie R. Margolies es radióloga en los siguientes hospitals: "Mount Sinai Queens, The Mount Sinai Hospital Mount Sinai St. , Luke's and Mount Sinai West". Aparte de su labor en el ámbito sanitario es profesora adjunta e investigadora en "The Mount Sinai Institute for Genomics and Multiscale Biology".

"The services that we routinely use on the Internet—social media, shopping, and financial applications—operate with a new generation of database technology that accommodates big data. Document storage databases, such as Hadoop, MarkLogic, and MongoDB, allow the quick integration and association of myriads of disparate data. These technical solutions are now migrating to the world of health care [56–58]. Discovery of associations between data elements can be expedited with this new generation of tools. For example, data mining to identify drug reactions can include, in addition to the medical literature, online self-reporting and electronic medical records [59]. Important associations that once would have been difficult, if not impossible, to imagine are now easier to substantiate with appropriate data-mining methods…" (Laurie R. Margolies, 2016).

"MongoDB is at the heart of this initiative, which captures the variety of data generated by genetic tests and integrates it with Genentech's existing Oracle RDBMS environment. MongoDB's flexible schema and ability to easily integrate with existing Oracle RDBMS has helped Genentech to reduce development from months to weeks or even days, significantly accelerating drug research. "Every day we can reduce the time it takes to introduce a new drug can have a big difference on our patients," said Doug Garrett, Software Engineer at Genentech

MongoDB also makes it possible for Genentech to load more data than in the past, which fits in well with the "collect now, analyze later" model, something he noted MongoDB co-founder Dwight Merriman has often suggested." (MongoDB, Best Of Both Worlds: Genentech Accelerates Drug Research With MongoDB & Oracle, 2014).

3. Implementación y Análisis del Modelo Conceptual

Previo al proceso ETL, es necesario realizar un análisis a nivel conceptual entre los datos de entrada y nuestro modelo conceptual, con el fin de abstraer el nivel lógico e identificar similitudes y diferencias de una forma más clara y concisa mediante un modelo de entidad-relación. Para ello, se ha implementado un modelo conceptual a partir de los ficheros .vcf que hemos utilizado como datos de entrada. Estos datos han sido proporcionados por el NBCI (***N***ational ***C***enter for ***B***iotechnology ***I***nformation).

3.1. Modelo Conceptual del Centro PROS

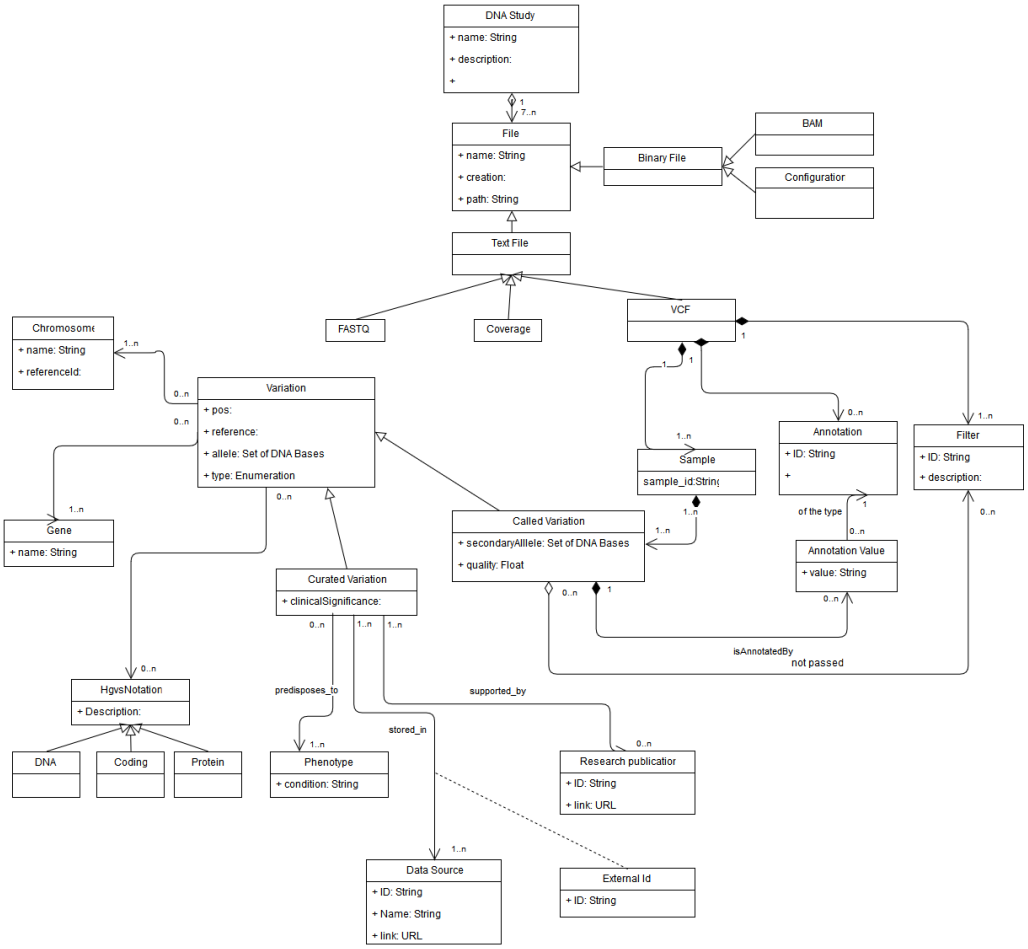


Ilustración 8 - Modelo Conceptual de referencia

El modelo conceptual mostrado en la parte superior hace referencia al modelo diseño por el Centro PROS para llevar a cabo algunas de sus tareas investigadoras. Este diseño tiene dos partes claramente diferenciadas: Secuenciación y variaciones.

Como ya se ha descrito anteriormente nuestros datos de entrada muestran información sobre las variaciones genéticas del humano, por lo que solo

trabajaremos con la parte del modelo correspondiente, tal y como se muestra en la imagen inferior:

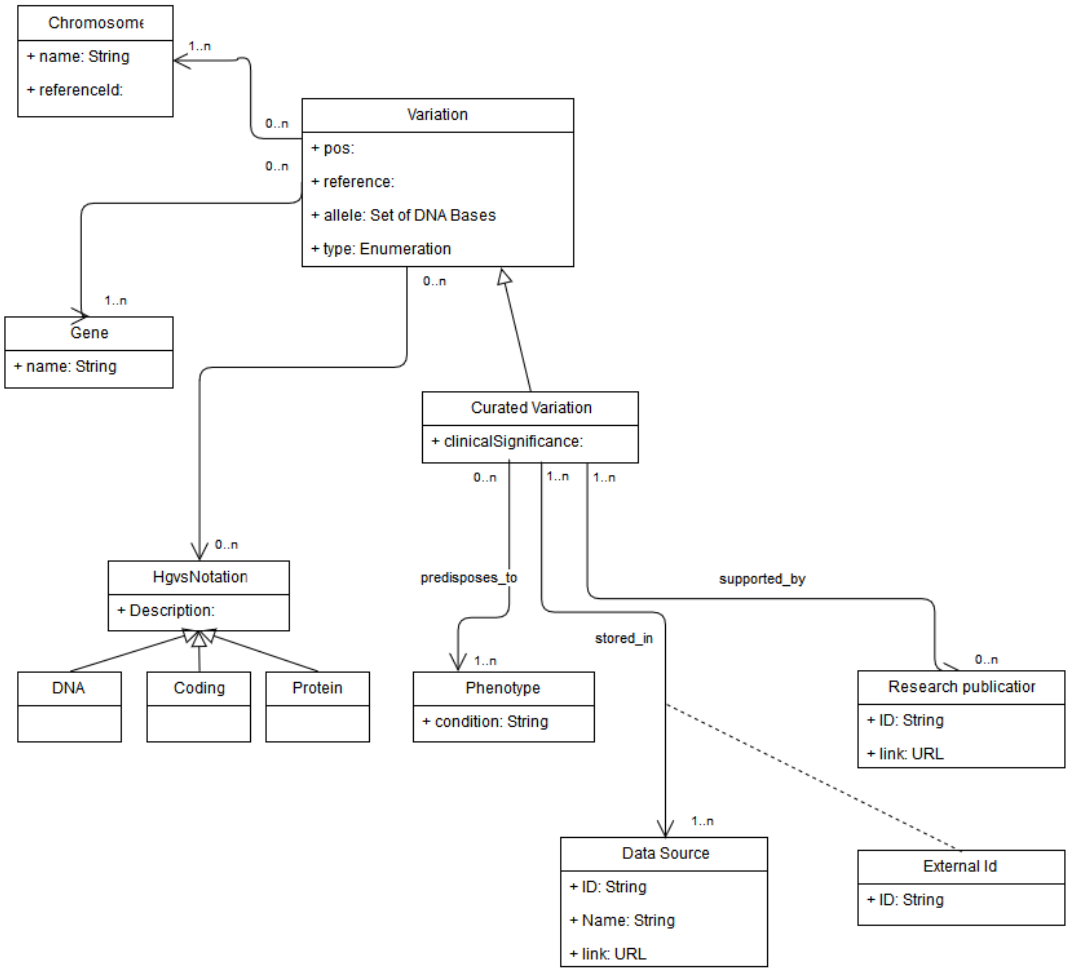


Ilustración 9 - Modelo conceptual de referencia filtrado

Para la descripción del modelo conceptual, se ha procedido en dos niveles: “Entidad” y “Relación”.

Comenzando por el nivel de entidad, se puede observar la entidad “Chromosome”, que representa una unidad de ADN que contiene el material genético. Es utilizada para situar y describir variaciones. Presenta dos atributos, “name” que representa textualmente a cada cromosoma y “referenceId” que identifica inequívocamente la secuencia de referencia en la comunidad genética. La entidad “Gene” representa una unidad de ADN que codifica una función específica en el ser humano. Gracias a este, se puede averiguar que función ha sido afectada por una variación. Presenta un atributo “name”, que representa textualmente a cada gen. La entidad “HgvsNotation” representa un estándar para la descripción de variaciones. Esta notación representa una variación en el ADN a nivel de ADN, ARN y proteína. Cada una de las entidades que heredan de esta: “ADN”, “Coding” y “protein”, se encargan de la representación en uno de los niveles que se han comentado. La entidad

“Variation” representa un cambio generado en alguna zona del ADN. Posee cuatro atributos: “Pos”, “Reference Allele”, “Alternative Allele”, “type”. El atributo “Pos” describe la posición de la variación dada por la posición de la primera base, de acuerdo con la secuencia de referencia. El atributo “Reference Allele” se corresponde a una lista de nucleótidos de la secuencia de referencia. Los posibles valores son cadenas compuestas por: A, G, T, C y “null” para inserciones. El atributo “Alternative Allele” se corresponde con una lista de los alelos alternativos a la secuencia de referencia. Los posibles valores son cadenas compuestas por: A, G, T, C, N. En el caso de no haber alelos alternativos nos encontramos con un caso de “deleción” donde no se obtendrá ningún valor. Por último, el atributo “type” representa el tipo de variación basado en el cambio sufrido con respecto a la secuencia de referencia. Como entidad heredera de esta se encuentra “Curated Variation”, a diferencia de la entidad “Variation” esta representa un tipo de variación previamente analizada por un genetista, proporcionando información sobre el efecto de la variación. Presenta un único atributo, “Clinical Significance”, correspondiéndose con una evaluación de la importancia de la variación en la práctica clínica. La entidad “Phenotype” representa información sobre el efecto de una variación. Su único atributo “condition” cita la enfermedad o condición que esta variación causa o podría causar. La entidad “Research Publication” proporciona información aquellas evidencias que demuestran la importancia de una variación y su relación con cierta enfermedad o condición. Esta entidad contiene tres atributos: “database”, “Id” y “Link”. El atributo “database” proporciona de la fuente de la que se ha extraído la publicación. El atributo “Id” identifica la publicación en cuestión dentro de la base de datos. Por último, el atributo “link” muestra la URL que contiene la información analizada. La entidad “Data Source” aporta información sobre la fuente de datos donde las variaciones analizadas son almacenadas. Esta entidad posee tres atributos: “Id”, “name”, “link”. El atributo “Id” se corresponde con una referencia textual de la fuente de datos. El atributo “name” representa el nombre completo de la fuente de datos. Por último, el atributo “link” muestra la URL de la fuente de datos. La entidad “External Id”, es una entidad débil nacida de la relación entre “Data Source” y “Curated Variation” y proporciona información adicional sobre las variaciones analizadas almacenadas en una fuente de datos. Su único atributo “Id” proporciona el identificador que identifica una variación en la fuente de datos.

A nivel de relación, cada cromosoma puede estar o no relacionado con una variación, lo cual significa que no todos los cromosomas deben contener variaciones, de forma análoga ocurre con los genes y la entidad “HgvsNotation”. Cada variación está relaciona con uno o más cromosomas, genes y elementos de la entidad “HgvsNotation”. Cada variación no tiene por qué estar relacionada con la entidad “HgvsNotation” ya que cada variación no tiene por qué estar registrada siguiendo este estándar. La entidad “Curated Variation” se relaciona con uno o más fenotipos y de forma análoga con la entidad “Data Source”. Por otro lado, la entidad “Curated Variation” no tiene por qué estar relacionada con la entidad “Research Publication”. Los fenotipos pueden estar o no relacionados con la entidad “Curated Variation” ya que no todas las enfermedades y complicaciones registradas no tienen por qué estar relacionadas con una variación analizada. Tanto la entidad “Data Source” como “Research Publication” se relaciona con una o más relaciones analizadas.

3.2. Modelo Conceptual de Clinvar

Una vez conocido el modelo de referencia vamos a exponer el modelo conceptual desarrollado con la información procedente de Clinvar. El modelado se ha basado en los requerimientos presentados por el modelo de referencia por lo que aquellas entidades o relaciones que no se contemplaban ni tenían ningún tipo de relación o equivalencia con el modelo base, han sido descartadas ya que no aportan ninguna información relevante para este proyecto. Por último, cabe destacar que como era previsible la información proporcionada por Clinvar no cubre con totalidad la información que se encuentra en nuestro modelo de referencia, no obstante el proyecto se puede llevar a cabo sin problema ya que los resultados seguirán siendo válidos el estudio que se está llevando a cabo.

El modelo realizado sobre la información de los ficheros Clinvar es el siguiente:

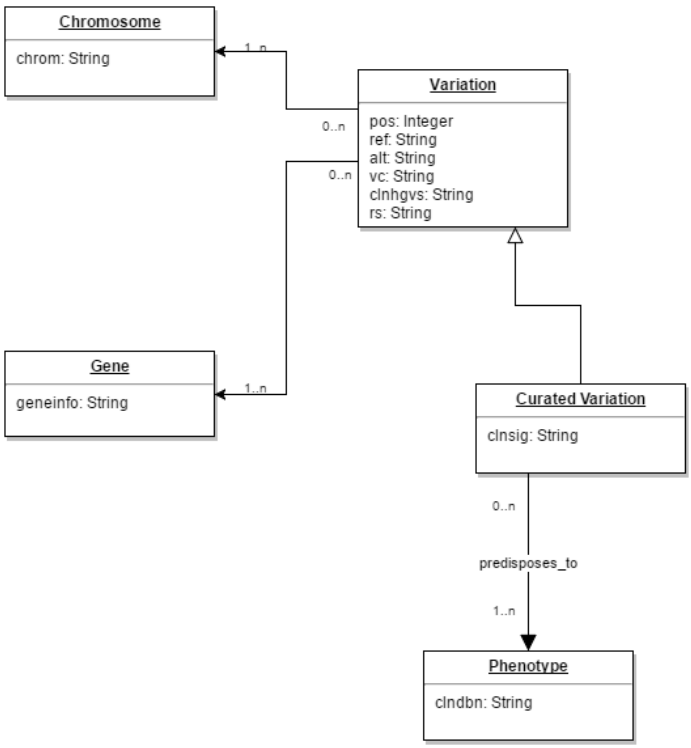


Ilustración 10 - Modelo conceptual de Clinvar filtrado

Para la descripción del modelo conceptual, se ha procedido en dos niveles: "Entidad" y "Relación".

Comenzando por el nivel de entidad, se puede observar la entidad “Chromosome”, que representa una unidad de ADN que contiene el material genético. Es utilizada para situar y describir variaciones. Presenta un único atributo, “chrom” que representa textualmente a cada cromosoma. La entidad “Gene” representa una unidad de ADN que codifica una función específica en el ser humano. Gracias a este, se puede averiguar que función ha sido afectada por una variación. Presenta un único atributo, “geneinfo”, que representa textualmente a cada gen (símbolo:geneID). La entidad “Variation” representa un cambio generado en alguna zona del ADN. Posee cuatro atributos: “pos”, “ref”, “alt”, “vc”,” clnhgvs”, “rs”. El atributo “pos” describe la posición de la variación dada por la posición de la primera base, de acuerdo con la secuencia de referencia. El atributo “ref” se corresponde a una lista de nucleótidos de la secuencia de referencia. Los posibles valores son cadenas compuestas por: A, G, T, C y “null” para inserciones. El atributo “alt” se corresponde con una lista de los alelos alternativos a la secuencia de referencia. Los posibles valores son cadenas compuestas por: A, G, T, C, N. En el caso de no haber alelos alternativos nos encontramos con un caso de “deleción” donde no se obtendrá ningún valor. Por último, el atributo “vc” representa el tipo de variación basado en el cambio sufrido con respecto a la secuencia de referencia. El atributo “clnhgvs” representa cada variación siguiendo un estándar de descripción de variaciones “HGVS”. El atributo “rs” proporciona el identificador que identifica una variación en la base de datos de dbSNP. Como entidad heredera de esta se encuentra “Curated Variation”, a diferencia de la entidad “Variation” esta representa un tipo de variación previamente analizada por un genetista, proporcionando información sobre el efecto de la variación. Presenta un único atributo, “clnsig”, correspondiéndose con una evaluación de la importancia de la variación en la práctica clínica. La entidad “Phenotype” representa información sobre el efecto de una variación. Su único atributo “clndbn” cita la enfermedad o condición que esta variación causa o podría causar.

A nivel de relación, cada cromosoma puede estar o no relacionado con una variación, lo cual significa que no todos los cromosomas deben contener variaciones, de forma análoga ocurre con los genes. Cada variación está relaciona con uno o más cromosomas, genes. La entidad “Curated Variation” se relaciona con uno o más fenotipos. Los fenotipos pueden estar o no relacionados con la entidad “Curated Variation” ya que no todas las enfermedades y complicaciones registradas no tienen por qué estar relacionadas con una variación analizada.

El aspecto es más conciso que el modelo de referencia ya que aun teniendo la información básica y esencial que constituye las variaciones, no contiene información referente a citaciones y entidades externas. La información aportada por este modelo en algunos casos no tiene una correspondencia directa con el modelo de referencia, con lo que se precisa de algunas transformaciones para que la información sea coherente, previamente a la carga, (véase sección 4.2.).

3.3. Análisis de modelado de datos

Una vez presentados ambos modelos, se debe proceder a realizar un análisis comparativo de estos con la finalidad de establecer similitudes y diferencias a la hora de trabajar con los dos modelos de datos antes de bajar a nivel lógico. Esta forma de proceder es algo que se quiere destacar en este proyecto. Como ingeniero de software antes de proceder a la implementación a nivel lógico de un conjunto de datos, es imprescindible tener implementado un modelo que nos aporte una descripción significativa del modelo de datos en una capa más abstracta. Hay diversos motivos que fundamentan esta metodología, comenzando por la naturaleza del proyecto. Un modelo conceptual representa a través de cómo puede ser el modelo de entidad-relación el modelo de datos sin entrar en un nivel técnico, fácilmente entendible por aquellos profesionales no expertos en el ámbito de la tecnología que pudieran verse inmersos en este proyecto o derivados como pueden ser aquellos expertos en el ámbito de las ciencias de la salud. Un modelo de entidad-relación es mucho más estable que un modelo lógico ya que cualquier cambio en los datos es muy probable que conlleve cambios a nivel lógico, con lo que si se procede a aplicar un proceso ETL a este nivel el proyecto es mucho más propenso a sufrir cambios que podrían tener costos temporales altos.

En la siguiente ilustración se muestran los tres niveles de modelado de datos: Conceptual, lógico y físico.

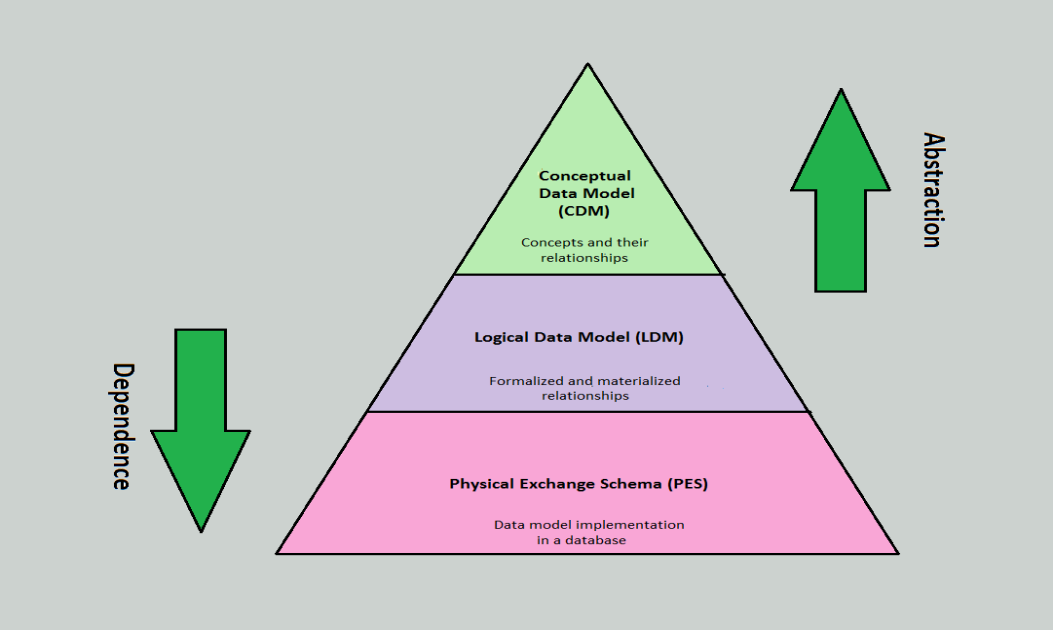


Ilustración 11 - Niveles de descripción de modelado de datos

3.3.1. Comparativa entre modelos

Antes de comenzar el análisis cabe destacar que existen entidades del modelo referencia que no están incluidas en el modelo conceptual de Clinvar, esas entidades son las siguientes: “DNA”, “Coding”, “Protein”, entidades que heredan de la entidad “HgvsNotation” ya que en el caso del modelo conceptual de Clinvar solo

se contempla una posible descripción de las variaciones, siguiendo la nomenclatura DNA, esta información está almacenada en un atributo de la entidad “Variation” y no en una entidad concreta. Por otro lado, la entidad “Research Publication” tampoco se contempla en el modelo conceptual de Clinvar, pero a diferencia del caso anterior esta información no está representada en forma de atributo de ninguna entidad ni está modelada siguiendo otra estructura. Por último, en el caso de la entidad “Chromosome”, el atributo “refenrenceId”, no se contempla en el modelo conceptual de Clinvar. Por último, el atributo “Link” de la entidad “Data Source” no está modelado en el modelo conceptual de Clinvar, pero este podrá incluirse ya que con la información proporcionada por el resto de atributos se puede conformar la de este último atributo.

A nivel de entidad, el resto de entidades coinciden entre ambos modelos, aunque cabe destacar que en algunos casos hay algunos cambios: La entidad “Gene” está estructurada de la misma forma en ambos modelos per en el caso de Clinvar se incluye más información de la que se utiliza en el modelo referencia con lo que se deberá transformar este atributo para que se corresponda con el modelo de referencia. La entidad “Chromosome” salvando lo descrito al comienzo de esta sección, ambas entidades presentan la misma estructura. En el caso, de la entidad “Variation” en el modelo conceptual de Clinvar, se constituye de un atributo adicional “clnhgvs” que como se detalló al comienzo de esta sección este se corresponde con la información representada por la entidad “DNA”→”HgvsNotation” del modelo de referencia. En el caso de la entidad “Data Source” salvando la particularidad del atributo “Link” descrita al comienzo de la sección ambas estructuras son iguales. Por último, en el caso de la entidad “ExternalID” y “Phenotype” la estructura es idéntica en ambos modelos y no precisa de ningún cambio.

A nivel de relación, el modelo es idéntico tanto a nivel de relaciones como tal y cardinalidades, por supuesto excluyendo aquellas que comprometen entidades que no están incluidas en el modelo de Clinvar.

3.3.2. Conclusiones previas al proceso ETL

Tras haber analizado ambos modelos y haber realizado una comparativa entre ambos podemos concluir con que ambos modelos son muy similares, como es lógico tienen algunas diferencias pero que a nivel conceptual son mínimas lo cual ha facilitado en gran medida tanto la compresión, análisis, así como la implementación del ETL (véase sección 4). Aunque tras este análisis parece que el proceso ETL será casi inmediato hay que tener en cuenta que el modelo conceptual de Clinvar se ha diseñado en base y bajo los requerimientos presentes en el modelo proporcionado por el Centro PROS de la UPV, lógicamente los ficheros .vcf que responden al primer modelo contienen mucha más información de la que se ha modelado así como relaciones y estructuras más complejas con lo cual la implementación es un desafío y una compleja tarea desde el comienzo del proceso.

Es destacable que la llegada de las TIC y más concretamente los sistemas de gestión y almacenamiento de datos a las organizaciones han proporcionado en muchas organizaciones el éxito estratégico y de gestión. El diseño e implementación de modelos conceptuales llevado a cabo por especialistas en sistemas ha sido el elemento clave para alinear los objetivos de la base de datos con la modelo

estratégico de negocio. En muchos casos los ejecutivos no ven la necesidad del alineamiento anteriormente descrito, pero es conveniente reiterar en la necesidad de este, así como de la importancia de analizar e identificar los requerimientos de la organización e incluir el estudio y desarrollo de modelos conceptuales como capa indispensable en la arquitectura de modelado de datos.

4. Implementación del proceso ETL

El proceso ETL consta de tres fases: Extracción, Transformación y Carga (Load). En la fase de extracción (véase sección 4.1.) se extraen los datos de la fuente principal, en este caso, Clinvar, generalmente se emplean fuentes de datos diversas, pero en este caso la fuente de datos de Clinvar responde en gran medida a los requerimientos del sistema de origen. Los datos proporcionados por Clinvar vienen representados en texto plano siguiendo un formato .tsv. En esta fase del proceso los ficheros serán filtrados, limpiados y preparados, adaptando estos a la estructura esperada, rechazando aquellos datos que no puedan ser adaptados al modelo de referencia.

La fase de transformación (véase sección 4.2) es la fase más compleja del proceso, en esta fase se hará uso del análisis realizado en el apartado anterior (véase sección 3) ya que son el punto de partida para desarrollar las reglas de negocio, manipulando los datos extraídos para que la carga sea exitosa y coherente con respecto con el modelo de referencia. En la mayoría de los casos no hará falta transformar los datos ya que hay una equivalencia directa, pero como ya se sabe (véase sección 3.3.1) habrá que hacer ciertas transformaciones en algunos casos por la naturaleza de ambos modelos y en otros casos por las estructura y disposición de los datos de entrada.

Con la fase de carga concluye el proceso ETL en este caso se dispone de dos bases de datos, SQL y NoSQL. Los datos son cargados en ambas bases de datos y se daría por finalizado el proceso. No solo se busca implementar el ETL con éxito sino obtener conclusiones sobre ciertas tecnologías como es a nivel de bases de datos para estimar su validez en estos escenarios o paradigmas.

El escenario o entorno sobre el que se implementa el ETL es el siguiente:

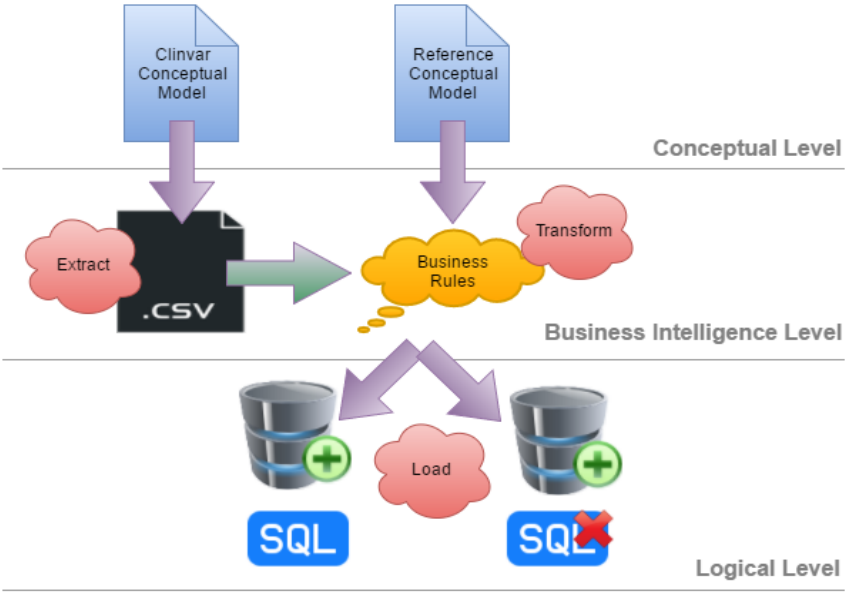


Ilustración 12 - Arquitectura de datos

4.1. Extracción de datos

En este apartado se extraen los datos de los ficheros .vcf de la base de datos de clinvar, se filtran y se limpian para que se adapten en formato a nuestro modelo de referencia.

Para la extracción de datos lo primero que se necesita es tener un "input" en este caso se trata un fichero delimitado por tabulaciones por tanto

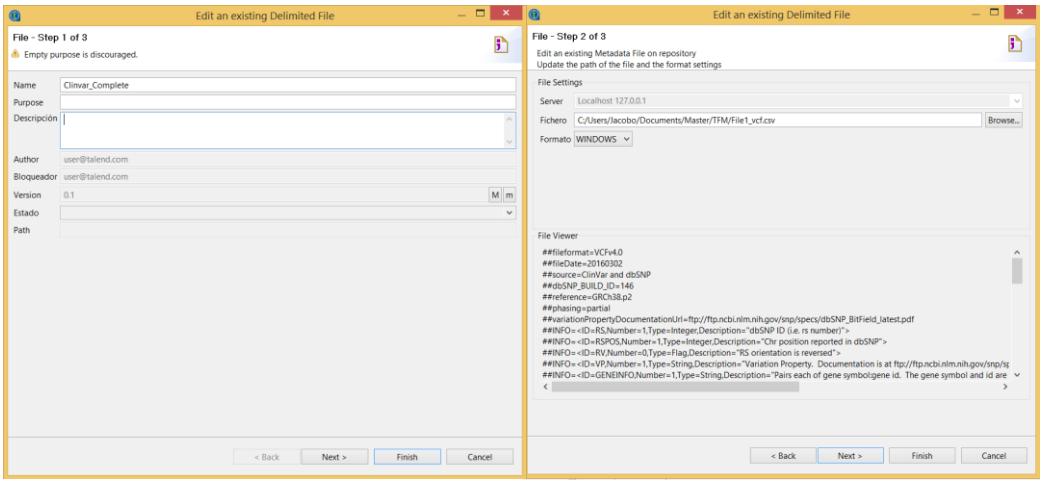


Ilustración 13 - Creación de fichero delimitado (pasos 1,2)

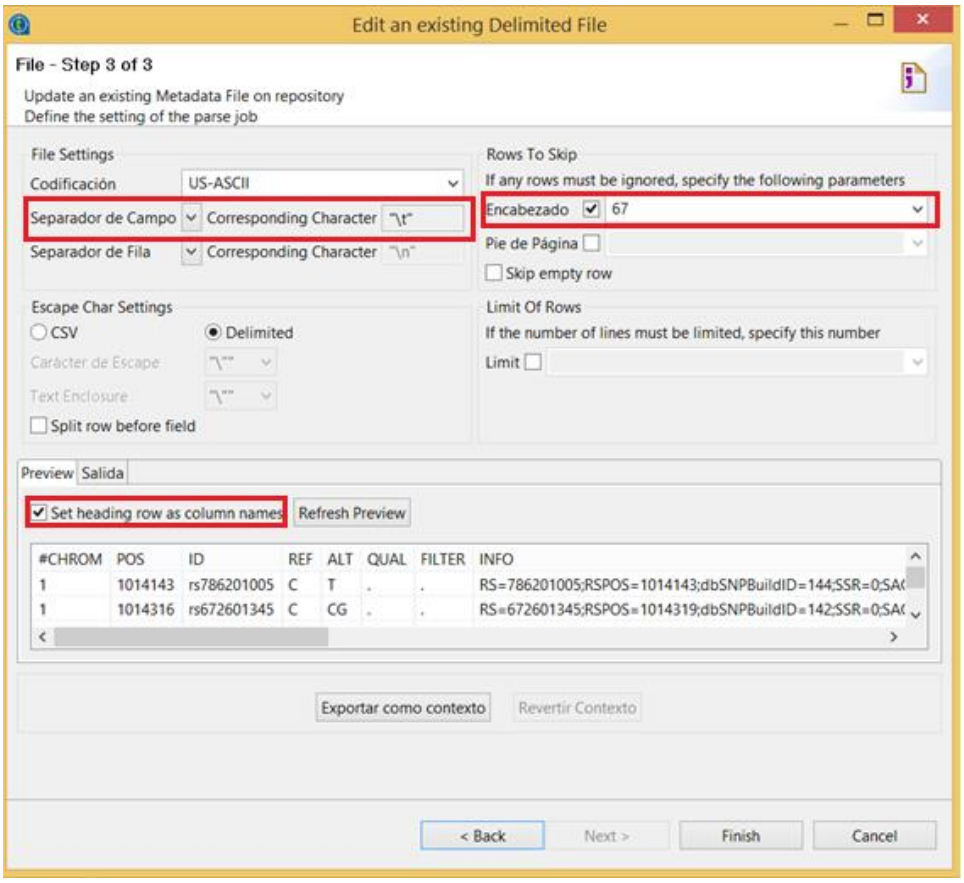


Ilustración 14 - Creación de fichero delimitado (paso final)

Se han resaltado algunos campos en la ilustración 7 ya que son aquellos que se han tenido que modificar o editar para un correcto formato de los datos. En primer lugar se ha elegido utilizar un encabezado con un valor asignado de “67”, con este parámetro se indica que el fichero introducido presenta un encabezado de sus datos. El valor “67” hace referencia a la línea donde se encuentra el encabezado, teniendo en cuenta que el contenido que se halle por encima de esta línea será descartado, en este caso se corresponde con líneas de descripción de los datos, con lo que pueden ser descartados. La siguiente opción que ha sido resaltada se corresponde con el delimitador utilizado, por defecto este ítem está preparado para ficheros .csv por tanto cambiamos el delimitador por “;” por “\t” para que los datos estén delimitados por una tabulación. Por último se ha establecido como cabecera la primera línea de datos, con esto se quiere indicar que la primera línea no se corresponde a una línea de datos, sino de cabecera.

Una vez se concluyen los tres pasos el fichero queda cargado y formateado correctamente. El siguiente paso consiste en añadir al proyecto actual este fichero arrastrándolo a la ventana de diseño.

4.2. Transformación de datos

En este apartado, se han creado las reglas de negocio, transformaciones para que los datos estén preparados para la carga en las bases de datos.

Para llevar a cabo este proceso, se ha utilizado la herramienta “tMap”. Esta herramienta permite dada una entrada y una salida, realizar una serie de manipulaciones de los datos mediante funciones proporcionadas por la herramienta o bien funciones creadas por el usuario usando java como lenguaje.

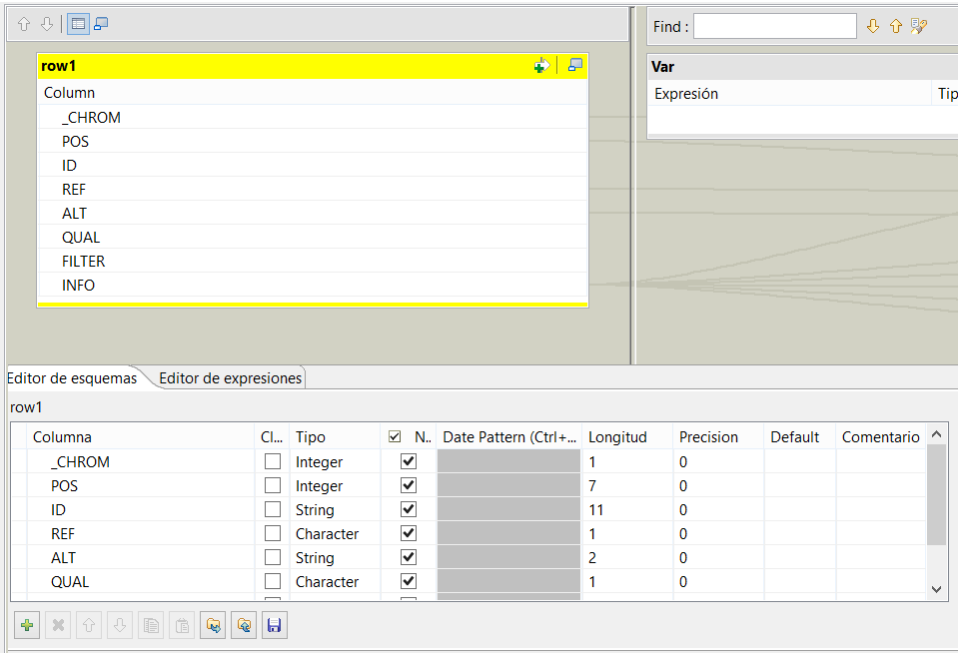


Ilustración 15 – tMap1_Input

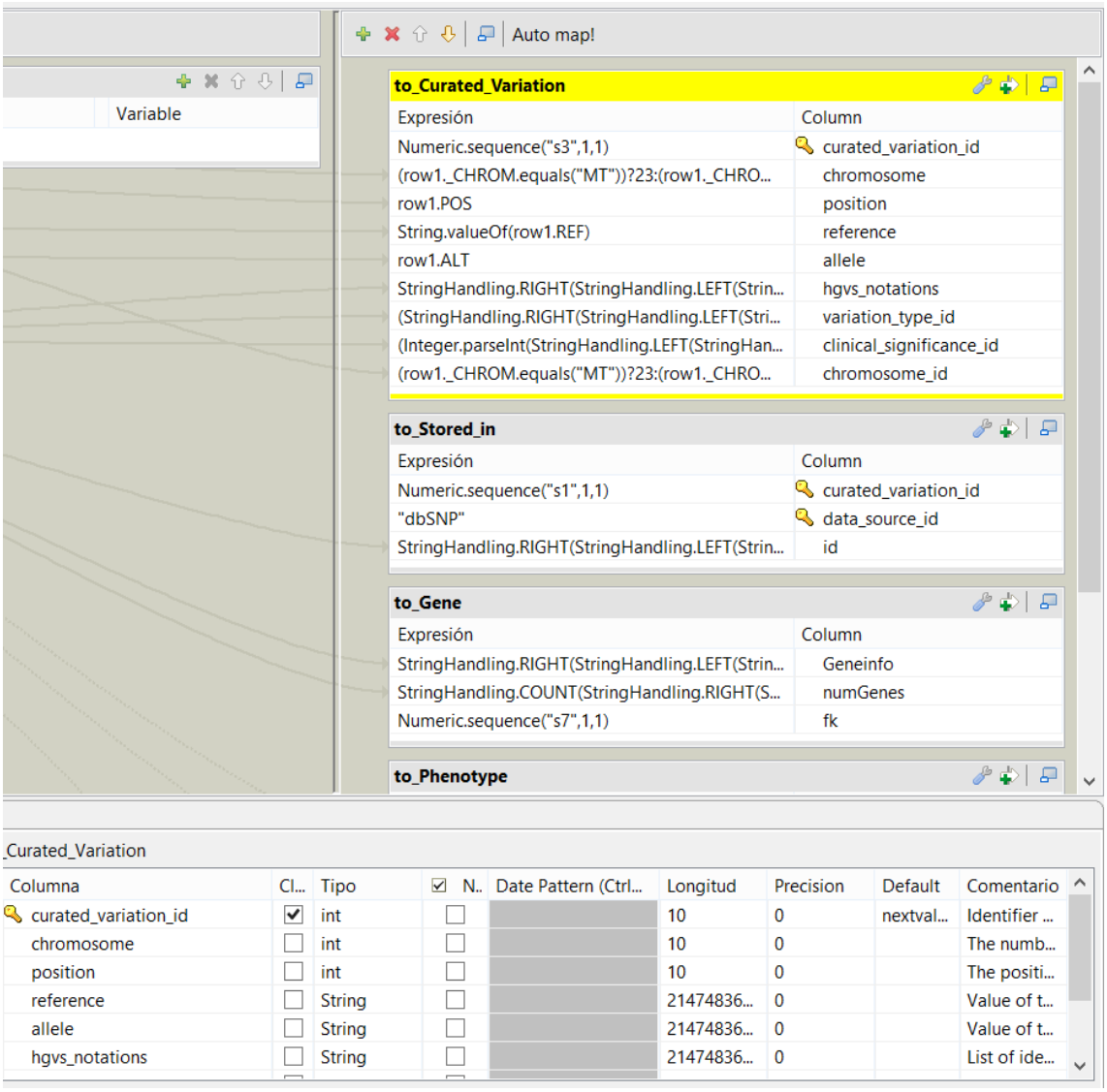


Ilustración 16 – tMap1_Output

Ambas ilustraciones se corresponden al aspecto interno del ítem “tMap” donde se puede apreciar en la parte superior los datos y las “expresiones”, transformaciones o reglas de negocios aplicadas. En la parte inferior se puede apreciar la capa física de los datos sobre la que fue montada la capa de negocio superior. Una vez observado internamente el objeto “tMap” se procede a explicar cómo han sido desarrolladas las transformaciones para al final obtener una serie de datos listos para ser cargados en las bases de datos (véase sección 4.3.).

En este caso, como entrada se ha escogido el fichero introducido en el proceso anterior (véase sección 4.1.). La problemática realmente viene debido al campo “info, ya que está compuesto por una serie de sub-campos de los cuales se ha extraído cierta información. La estructura es la siguiente:

#CHROM POS ID REF ALT QUAL FILTER INFO

1 1014143 rs786201005 C T . .
RS=786201005;RSPOS=1014143;dbSNPBuildID=144;SSR=0;SAO=1;VP=0x05
0060000605000002110100;GENEINFO=ISG15:9636;WGT=1;VC=SNV;PM;NSN;
REF;ASP;LSD;OM;CLNALLE=1;CLNHGVS=NC_000001.11:g.1014143C>T;CLNSR
C=OMIM_Allelic_Variant;CLNORIGIN=1;CLNSRCID=147571.0003;CLNSIG=5;C
LNDSDB=MedGen:OMIM;CLNDSDBID=CN221808:616126;CLNDBN=Immunode
ficiency_38;CLNREVSTAT=no_criteria;CLNACC=RCV000162196.375790

La mayoría de información está concentrada en este campo, por tanto mediante funciones proporcionadas por Talend se ha extraído la información siguiendo la siguiente estrategia:

1. Los campos fuera de la estructura del campo “info” no han necesitado ningún tipo de transformación, con lo cual la correspondencia ha sido directa salvo en el caso del campo “#CHROM”.

2. Al campo “#CHROM” se le aplica una conversión “String to Integer” y se realiza un tratamiento especial para los valores no numéricos (“X”,”Y”,”MT”) como es mostrado a continuación:
“(row1._CHROM.equals("MT"))?23:(row1._CHROM.equals("X"))?24:(row1._C
HROM.equals("Y"))?25:Integer.parseInt(row1._CHROM)”.

3. El campo “INFO” es el campo más complejo y para su transformación se han utilizado cuatro métodos: StringHandling.LEN(), StringHandling.INDEX(), StringHandling.RIGHT(), StringHandling.LEFT(). El método LEN() devuelve la longitud de una cadena de texto. El método INDEX() busca una sub-cadena dentro de una cadena de texto y devuelve la posición en la que se encuentra. El método RIGHT() dada una cadena de texto c y un entero n, devuelve los últimos n caracteres la cadena c. Por último, el método LEFT() dada una cadena de texto c y un entero n, devuelve los primeros n caracteres la cadena c. Por tanto una vez han sido presentados los métodos utilizados vamos a mostrar una traza gráfica de cómo han sido substraídos los valores de los campos interesados tomando como referencia esta cadena de ejemplo:
“x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;Campo=valor;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;”

   Paso1: Localizar el campo

“x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;Campo=valor;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;”

   Paso2: Limpiar cadena
   Para usar la función “RIGHT” es necesario antes haber aplicado el método LEN() para averiguar la longitud de la cadena.

"x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;Campo=valor;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;"

Cadena resultante: "Campo=valor;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;"

Paso3: Localizar fin del campo:

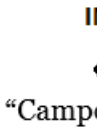
"Campo=valor;x:x:x:x:x:x:x:x:x:x:x:x:x:x:x;"

Paso4: Limpiar cadena

"Campo=valor;x:x:x:x:x:x:x:x:x:x:x:x:x:x:x;"

Cadena resultante: "Campo=valor"

Paso5: Localizar valor:



"Campo=valor"

Paso6: Limpiar cadena:
Para usar la función "RIGHT" es necesario antes haber aplicado el método LEN() para averiguar la longitud de la cadena.

"Campo=valor"

Cadena resultante: "valor"

Una vez aplicados los seis pasos ya se obtiene el valor deseado, a nivel de código este sería el resultado de una búsqueda de este tipo:

StringHandling.RIGHT(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"VC")),";")),StringHandling.LEN(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"VC")),";")))-StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"VC")),";")),"=")-1);

Los sub-campos “CLNDBN” y “GENEINFO” presentan una dificultad añadida, debido a que no había un número de elementos “GENEINFO” y “CLNDBN” concreto relacionados con cada elemento de “Curated Variation” y ha sido necesario programar una rutina en java que se encargase de transformar los datos. Antes de explicar el “pipeline” por el cual los datos han pasado, se ha incluido la siguiente imagen para facilitar la comprensión de la explicación y descripción:

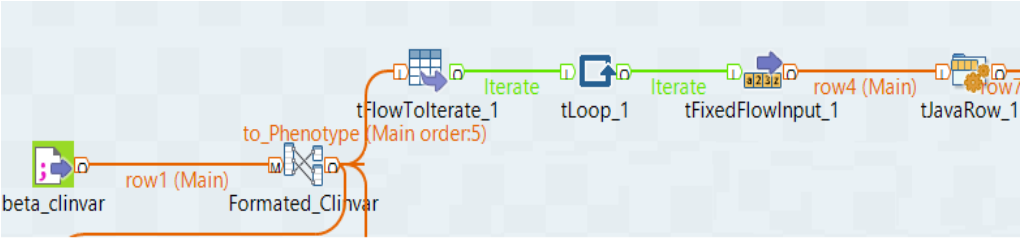


Ilustración 17 – Transformación custom

Para poder explicar este proceso hemos utilizado la entidad “Gene” para ejemplificar y explicar las siguientes etapas de la que consta este proceso:

1. Extracción de datos de “Clinvar” a través de una transformación aplicada a través de un ítem “tmap” donde se ha creado la estructura “to_Gene” donde se almacena cierta información para cada elemento de “Clinvar”. Esta estructura presenta tres atributos:

   Geneinfo: Nombre de los genes.

   Codigo: StringHandling.RIGHT(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-

StringHandling.INDEX(row1.INFO,"GENEINFO")),";")),StringHandling.LEN(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),";")))-StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),";")),"=")-1);

Numgenes: Número de genes.

Codigo:
StringHandling.COUNT(StringHandling.RIGHT(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),";")),StringHandling.LEN(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),";")))-StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"GENEINFO")),";")),"=")-1),":")

Ya que “Clinvar” utiliza el caracter “|” para delimitar sub-elementos (gene1|gene2|…) ha bastado con contar el número de “|” (+ 1).

Fk: Clave ajena. Este último dato ha sido extraído para almacenar el “id” del elemento de “Clinvar”, este dato será de gran transcendencia como se puede observar en la sección 4.3. “Carga de datos”.

Código:
Numeric.sequence("s7",1,1)

2. Bucle principal pre-código java. Se han utilizado los ítems “tFlowtoIterate” utilizado para transformar el flujo de datos en el “formato” adecuado para realizar los siguientes pasos y el elemento “tLoop” el cual nos permitirá iterar sobre cada elemento entrante. En este último utilizaremos el dato “to_Gene.numGenes” como límite superior del bucle.

3. Crear flujo de entrada pre-java. Para ello se ha utilizado el ítem “ tFixedFlowInput” el cual se ha utilizado para inyectar al componente java los datos.

4. Código java. Se ha utilizado el ítem “tJavaRow” para realizar la compleja transformación.

   Código:

```
String aux = to_Gene.Geneinfo;
String cadena;
int index = 0;
int bound = ((Integer)globalMap.get("tLoop_1_CURRENT_VALUE"));


for(int i = 0; i < bound; i++){
        index = StringHandling.INDEX(aux,":");
        cadena = StringHandling.LEFT(aux,index);
        index = StringHandling.INDEX(aux,"|");
        aux = StringHandling.RIGHT(aux,StringHandling.LEN(aux)-index-1);
        row3.name = cadena;
        row3.gene_ontology_id = Integer.toString(to_Gene.fk);
        row3.gene_id = Numeric.sequence("s6",1,1);
}
```

El hecho de que las transformaciones sean ciertamente complejas, es debido en primer lugar a que estas han sido desarrolladas de forma genérica, es decir, que sería válido para extraer cualquier tipo de información dentro del campo “INFO” y en segundo lugar a la variedad de estructuras de los datos de este campo, por lo que ha dificultado en gran medida la extracción y transformación de estos.

Por último, cabe destacar que es de vital importancia que en todo sistema de información los datos presenten una estructura bien definida. En este caso, aun tratándose de información médica hay ciertas incongruencias en cuanto a la estructuración y formato de los datos en algunos casos. Estas incongruencias dificultan notoriamente el trabajo del ingeniero y pueden acarrear problemas importantes sobre todo en el proceso de carga en la base de datos, por otro lado, este tipo de errores, según la gravedad de estos, podrían disminuir el rendimiento del proceso ETL. Un ejemplo de esto sería el mostrado a continuación donde en la cabecera del fichero de Clinvar se presenta el campo CLNSIG como un entero del 0 al 7 y el valor 255, para contextualizar, este valor representa para una variación concreta la valoración médica de esta. A continuación se muestra una variación donde vemos que se viola el formato de los datos:

1 2228866 rs387907306 G A,T . . RS=387907306;RSPOS=2228866;dbSNPBuildID=137;SSR=0;SAO=1;VP=0x050060000a05000002110100;GENEINFO=SKI:6497;WGT=1;VC=SNV;PM;NSM;REF;ASP;LSD;OM;CLNALLE=1,2;CLNHGVS=NC_000001.11:g.2228866G>A,NC_000001.11:g.2228866G>T;CLNSRC=OMIM_Allelic_Variant,OMIM_Allelic_Variant;CLNORIGIN=33,33;CLNSRCID=164780.0004,164780.0005;**CLNSIG=5|5,5|5**;CLNDSDB=MedGen:OMIM:ORPHA:SNOMED_CT|MedGen,MedGen:OMIM:ORPHA:SNOMED_CT|MedGen;CLNDSDBID=C1321551:182212:246

2:83092002|CN221809,C1321551:182212:2462:83092002|CN221809;CLNDBN=Shprintzen-Goldberg_syndrome|not_provided,Shprintzen-Goldberg_syndrome|not_provided;CLNREVSTAT=single|single,single|single;CLNACC=RCV000030819.28|RCV000200686.1,RCV000030820.27|RCV000197434.1

Aquí se observa como una de las valoraciones asignadas tiene el valor de “5,5” lo cual viola el formato establecido.

En muchas ocasiones, estas tareas no son asignadas a ingenieros informáticos y esto en ocasiones da lugar a que los sistemas de información no tengan la calidad ni la robustez esperada. La forma de evitar este tipo de problemas y construir un sistema de información robusto, es cubrir todas las capas de modelado de datos, comenzando por el diseño conceptual (véase sección 3.3).

4.3. Carga de datos

En este apartado se contemplan dos casuísticas como se mostraba en la arquitectura expuesta al comienzo de la sección. La primera de ellas consiste en realizar la carga de los datos en una base de datos relacional “PostgreSQL”, en la segunda casuística se procede a realizar la misma carga en una base de datos no relacional utilizando “MongoDB” como tecnología.

4.3.1. Base de datos relacional, PostgreSQL

La base de datos relacional utiliza PostgreSQL como tecnología. El esquema de esta base de datos al igual que el modelo conceptual de referencia ha sido proporcionado por el centro de investigación PROS del departamento del DSIC de la UPV.

Antes de proceder a la carga se muestra el esquema de la base de datos completo así como filtrado para tener una referencia:

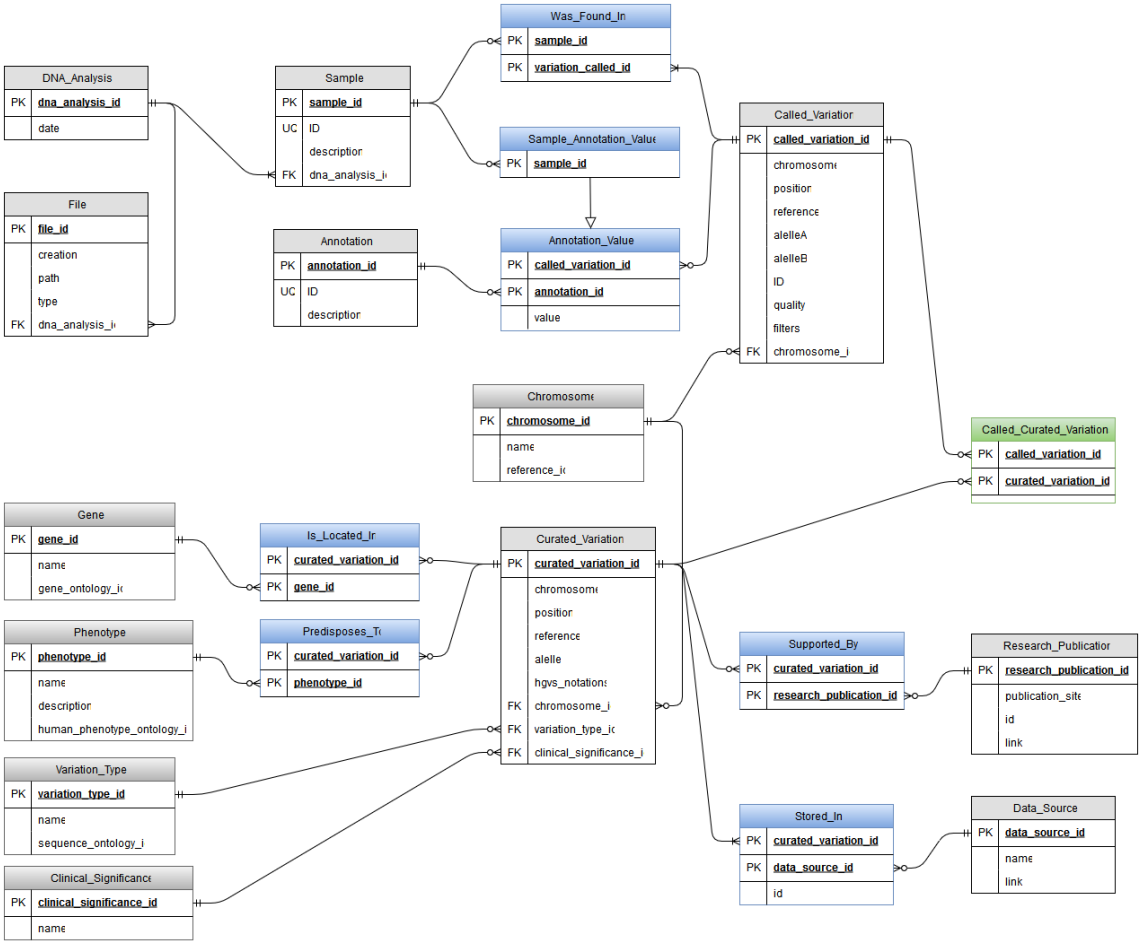


Ilustración 18 - Esquema lógico

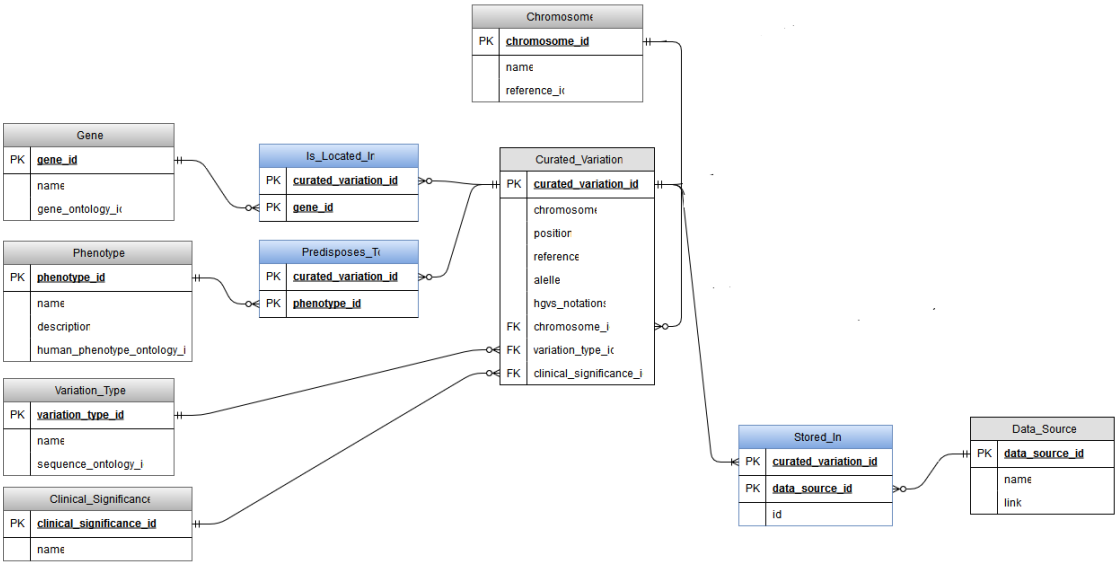


Ilustración 19 - Esquema lógico filtrado

Antes de proceder a la carga sería conveniente tener presente como y donde está constituida esta base de datos. Se ha utilizado “pgAdmin III” como gestor de base de datos en PostgreSQL. Se ha montado la base de datos en la máquina local sobre la que se ha desarrollado y desplegado el proyecto, no obstante, a la hora de trabajar con Talend, es irrelevante ya que realmente este se puede conectar o bien a una base de datos local o de forma remota a esta. La base de datos tiene el nombre de “cap” al igual que su propietario. Esta base de datos está compuesta por un esquema llamado “public” en la que es restauró el “dump” proporcionado por el centro PROS. Por último en la siguiente ilustración se puede observar la arquitectura descrita desplegada en “pgAdmin III”:

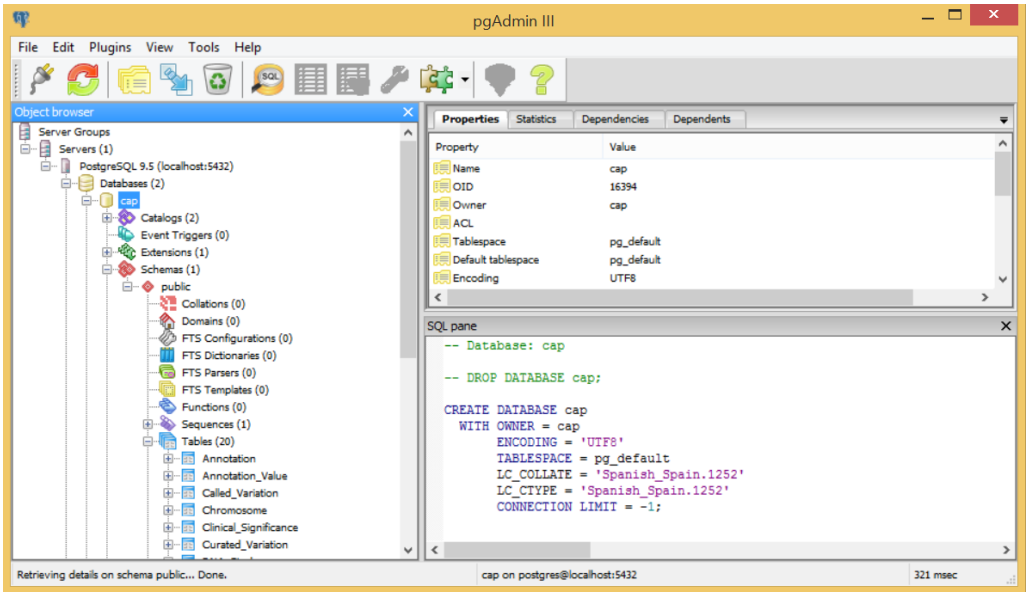


Ilustración 20 - Base de datos en PgAdmin III

Una vez sabiendo donde y como la base de datos se estructura y reside, se puede proceder a la conexión a la base de datos en Talend. Para ello se deben realizar los siguientes pasos.

En el explorador del proyecto -> Metadata -> Db Connection, se selecciona utilizando doble click sobre este último “Crear Conexión”.

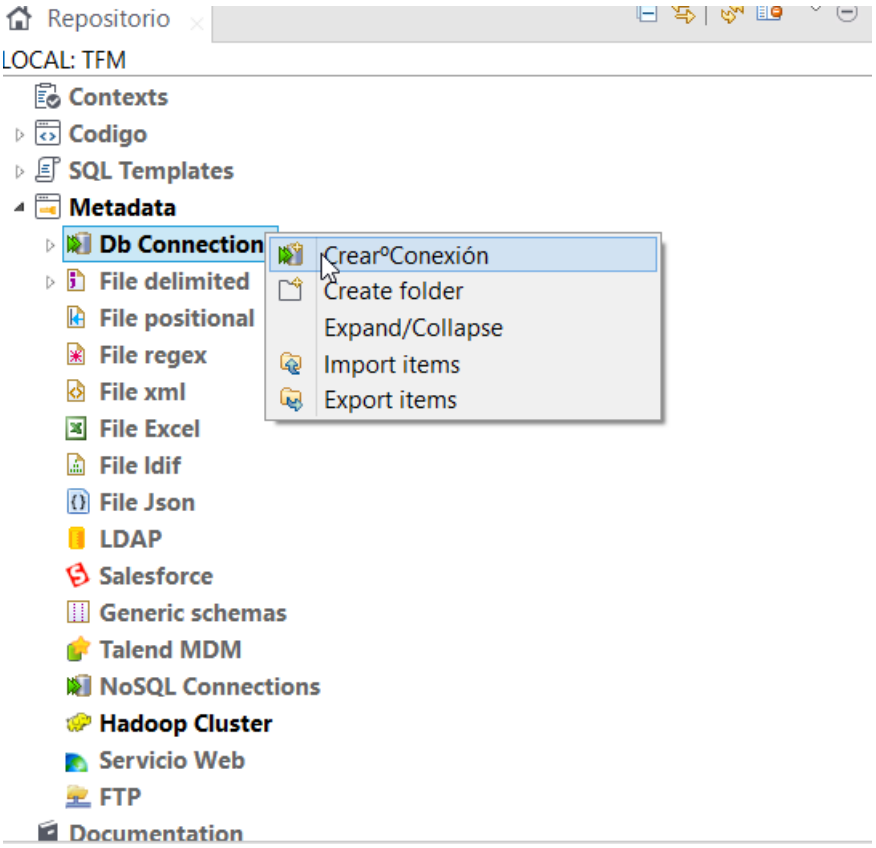


Ilustración 21 - Conexión base de datos (paso 1)

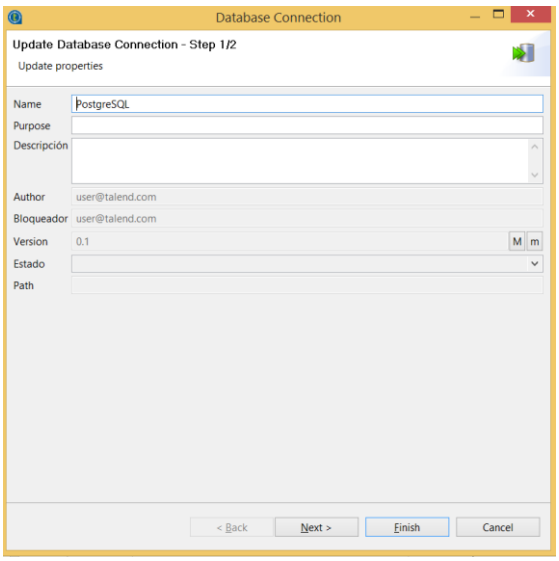


Ilustración 22 - Conexión base de datos (paso 2)

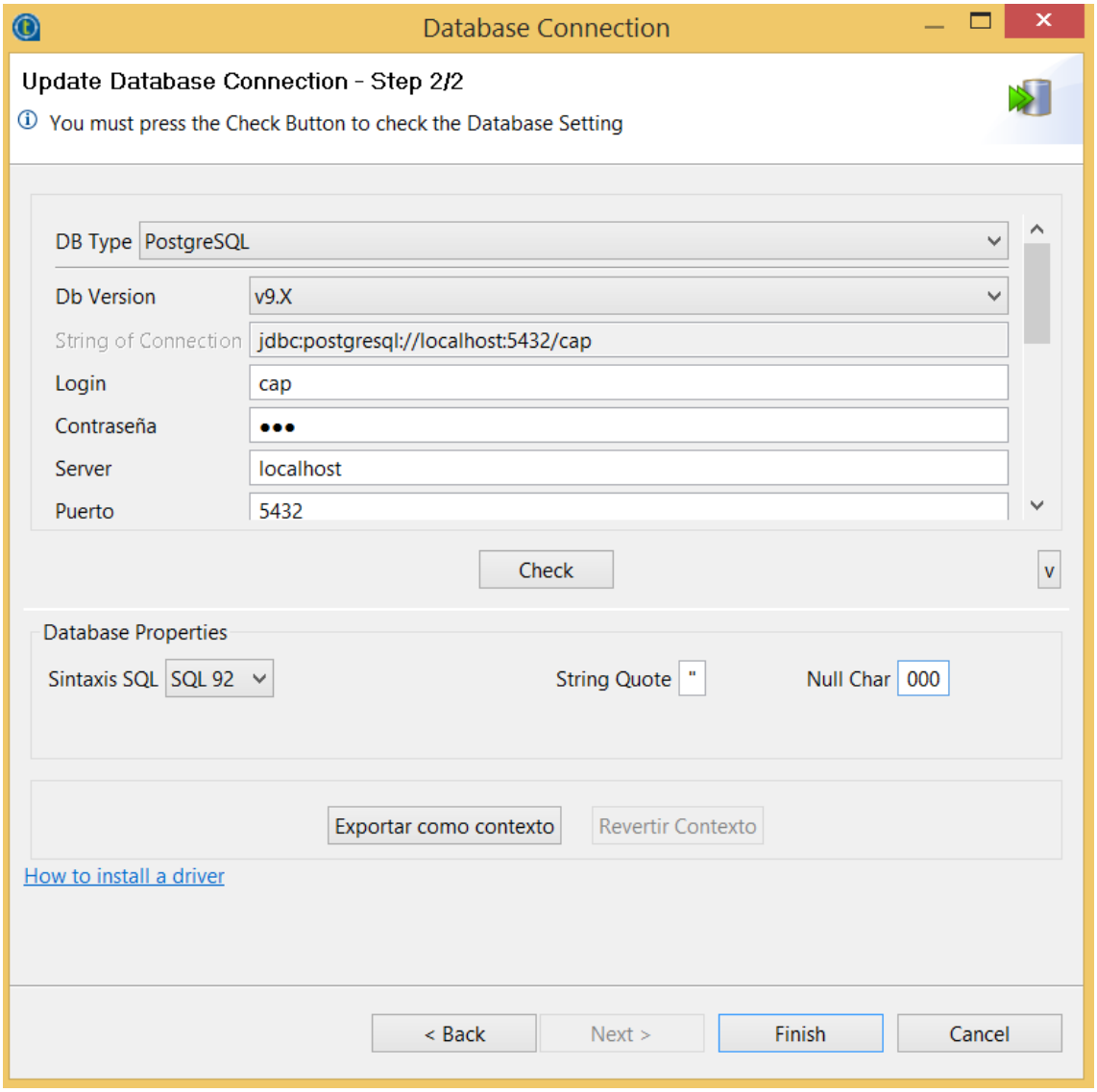


Ilustración 23 - Conexión base de datos (paso 3)

El último paso es el más importante ya que se establecen y configuran los parámetros de la conexión. En el campo "Db Type" se selecciona la tecnología de base de datos escogida, en este caso PostgreSQL, en "Db Version" se selecciona la versión utilizada, en este caso, la versión 9.X. Los dos siguientes campos se corresponden con las credenciales del usuario con el que se vaya acceder a la base de datos. Como se observó al comienzo de esta sección en el gestor "pgAdmin" la base de datos era propiedad de "cap", se utilizará este para proceder a la conexión. El puerto a utilizar es el "5432" ya que es el puerto por defecto o establecido para conexiones de base de datos bajo "PostgreSQL". Por último, se establece el nombre de la base de datos a conectarse y del esquema, que como en los parámetros anteriores se puedo observar en "pgAdmin" que recibían el valor de: "cap" y "public" respectivamente. Hay un último apartado para definir algunas propiedades de la base de datos, como puede ser la sintaxis SQL utilizada, en nuestro caso todo permanece con los valores por defecto. Es recomendable realizar un "check" para verificar que la conexión se ha realizado con éxito antes de finalizar con el proceso.

Llegados a este punto ya se ha incluido la conexión con la base de datos, no obstante, ahora quedará crear instanciaciones de esta base de datos en el proyecto haciendo uso del ítem “tpostgresqlOutput”. Este ítem permite insertar o actualizar valores de una tabla en concreto de la base de datos. Por tanto, se crearan tantos ítems como tablas a modificar. Estas instanciaciones heredan los parámetros que se configuraron al establecer la conexión con la base de datos, lo único que habrá que especificar será la tabla sobre la que se va a operar.

Una vez tenemos los datos correctamente extraídos y transformados (véase sección 4.1 y 4.2), así como, la base de datos correctamente conectada con el proyecto vamos a proceder a efectuar a realizar la carga sobre la base de datos relacional. Para contextualizar y recordar el esquema sobre el que vamos a cargar los datos, es recomendable revisar el apartado correspondiente (véase sección 3.1).

Para ello, es importante remarcar y recordar que la información que “Clinvar” ofrece es exclusivamente referente a variaciones genéticas, no obstante, como quedó reflejado en el modelo conceptual (véase sección 3.2) ha sido posible extraer a través de ingeniería inversa el resto de información.

Una vez los datos han sido transformados en muchos casos la carga es inmediata, en este caso solo algunas tablas se encuentran en esta situación, debido a que en la mayoría de los casos cómo se desarrolla en esta subsección, se va a precisar de una manipulación previa a la carga de los datos.

Por otro lado, este último proceso posee una dificultad adherida con respecto a la coherencia de los datos en la base de datos a nivel de claves (integridad). Cuando un proceso ETL es aplicado, en la mayoría de casos es necesario desactivar las restricciones de la base de datos, que como es sabido, son el conjunto de reglas relativas que definen a los valores permitidos en las distintas columnas y constituyen el mecanismo estándar para exigir la integridad.

PostgreSQL es un sistema de gestión de bases de datos relacionales que se caracteriza entre otros aspectos por la seguridad y la fuerte integridad referencial y a nivel de dominio. En otros sistemas es posible mediante “queries” desactivar las restricciones en masa, pero en PostgreSQL no es posible.

Antes de comenzar con la carga se han eliminado todas las restricciones menos de PK (“*Primary key”)* ya que la carga se ha realizado en distintas fases y por tanto, hasta que no se finaliza el proceso no es posible garantizar la integridad en la base de datos. La “*query*” utilizada en la mayoría de los casos para la generación de las restricciones de FK es la siguiente:

```
//Se selecciona esquema y tabla a modificar

ALTER TABLE public."Is_located_in"

 /* Se escoge la operación de añadir restricción estableciendo la columna “gene_id” como FK (“Foreign Key”) */

 ADD CONSTRAINT gene_id FOREIGN KEY (gene_id)
```

//Se establece la relación con el esquema, tabla y columna objetivo

REFERENCES public."Gene" (gene_id) MATCH SIMPLE

// En las acciones de "*UPDATE*" y "*DELETE*" no se permiten valores "null" en una columna FK, en este caso, se producirá un error en la "*query*".

ON UPDATE NO ACTION ON DELETE NO ACTION;

La "*query*" utilizada para la eliminación de las restricciones es la siguiente:

/*Se selecciona el esquema y tabla a modificar. Se escoge la operación borrar o eliminar restricción y por último se especifica el nombre de la restricción.*/

ALTER TABLE public."Predisposes_To" DROP CONSTRAINT phenotype_of_variation_id;

Una vez las restricciones han sido eliminadas, la base de datos ya está preparada para realizar la carga en varias fases. Cabe destacar que a partir de este punto la integridad es delegada en el ingeniero, ya que la base de datos es susceptible de aceptar datos que violen este tipo de restricciones. Por tanto, una vez la carga se haya realizado satisfactoriamente, las restricciones deben ser restablecidas para garantizar que el ingeniero no ha cometido ninguna violación de integridad.

Las distintas casuísticas que se presentan en el proceso de carga son las siguientes:

1. Datos insertados manualmente (*harcoding*).
2. Datos insertados de forma inmediata.
3. Datos insertados mediante ingeniería inversa.
4. Datos insertados en tablas intermedias.
5. Datos insertados en la tabla "Curated_Variation"

La primera de las casuísticas se ha dado en aquellas tablas de la base de datos que tienen poco contenido y este a su vez es estático como es en el caso de las tablas: "Clinical_Significance", "Data Source", "Chromosome" y "Variation Type". Para ellos se ha utilizado el componente "tFixedFlowInput", este componente te permite inyectar datos "hardcodeados". Posteriormente a través del componenten "tmap" se establecerán las correspondencias con el esquema de la base de datos y de esta forma cada dato será almacenado en la tabla y columna correspondiente.

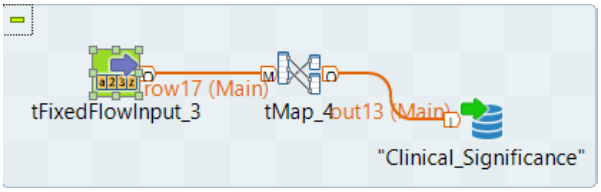


Ilustración 24 - Carga Harcoding

La segunda casuística, se corresponde con aquellas cargas que tras la transformación simplemente a través de una instanciación del componente “tmap” para establecer correspondencia de campos es suficiente. Las tablas que se van afectadas por esta casuística son: “Phenotype”, “Gene”, “Stored_In”.

La tabla “Stored_In” es una tabla intermedia con la funcionalidad de mantener las relaciones entre las tablas “Curated_Variation” y “Data Source”. Estas tablas son más complejas ya que relacionan dos tablas, en este caso, ha sido más sencillo, ya que la tabla “Data Source” solo tiene una entrada, con lo cual en este caso particular, esta tabla podría haber sido evitada.

StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"CLNSIG")),";")))-StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-StringHandling.INDEX(row1.INFO,"CLNSIG")),";")),"=")-1),1))+1:9

En este caso, se ha ejecutado una “*query*” para permitir valores “null” sobre las columnas variation_type_id y chromosome_id.

ALTER TABLE public."Curated_Variation" ALTER COLUMN "variation_type_id" DROP NOT NULL;

ALTER TABLE public."Curated_Variation" ALTER COLUMN "chromosome_id" DROP NOT NULL;

Al igual que en los casos anteriores, todas las restricciones que han sido eliminadas deberán ser restablecidas una vez los datos estén completamente cargados.

La tercera y cuarta casuística están ligadas ya que las tablas de cada una de estas son necesarias para completar el proceso de carga del resto. En el caso de la tercera casuística las tablas afectadas son “Gene” y “Phenotype”, siendo “Is_located” y “Predispose_to” en el caso de la cuarta.

En el caso de “Gene” y “Phenotype” la carga se realizó en dos fases, incluyendo una fase adicional necesaria para realizar la carga en las tablas “Is_located” y “Predispose_to”.

La primera fase de carga, es inmediata tras lo visto en la fase de transformación (véase sección 4.2). Los datos cargados en esta fase no son los datos que se esperaban, ya que inicialmente se han cargado todos los elementes “Gene” y “Phenotype” encontrados en cada variación, por lo que a través de un filtrado deberían quedar solo elementos únicos en cada una de las tablas. Este filtrado podría haber sido realizado previamente y haber evitado esta carga, pero el motivo de haber optado por esta metodología es debido a que de esta forma se mantienen

relacionadas las tablas “Curated_Variation”, “Gene” y “Phenotype”, ya que las tablas “Gene” y “Phenotype” poseen una columna que permanecerá vacía en todo momento (x_ontology_id), donde provisionalmente se ha almacenado el “id” de cada elemento de la tabla “Curated_Variation”. Por otro lado, se almacenará el “id” de cada elemento “Gene”/”Phenotype” (x_id) y por último, el elemento en sí (name). Con lo cual, las tabla “Gene”/”Phenotype” para cada fila almacenan: “id” del elemento de la tabla “Curated_Variation”, “id” del elemento y el nombre del elemento. El otro motivo por el cual no se han filtrado los datos es porque para el proceso de ingeniería inversa aplicado, aunque el flujo de datos estuviera compuesto por numerosas filas, para poder aplicar el proceso, el flujo de datos tuvo que ser reformateado (véase sección 4.2) encapsulando el flujo de datos y dejando este inaccesible a nivel de fila.

En la segunda fase de carga, se han creado dos flujos de datos, el primero de ellos se ha dedicado a la tabla “Gene”/”Phenotype” de la base de datos, para la cual se ha utilizado el componente “tUniqRow” permitiendo eliminar del flujo de datos de salida aquellos datos de entrada que estén repetidos, para ello el filtro se ha configurado para filtrar por el campo “name” lo cual en el flujo de salida no habrá ningún elemento de la tabla “Gene”/”Phenotype” repetido. Además de filtrar los datos se han eliminado los datos que estaban almacenados en la columna “x_ontology_id”, ya que no pertenecían realmente a la tabla “Gene”/”Phenotype”. El segundo flujo de datos se crea exclusivamente para las tablas intermedias “Is_located”/”Predispose_to”, almacenando los datos que habían en la tabla “Gene”/”Phenotype” mediante una tabla hash utilizando el componente “tHashOutput”. En esta estructura se ha almacenado tanto el “id” de los elementos de la tabla “Curated_Variation” y los datos de la columna “name” de “Gene”/”Phenotype”.

En la tercera fase de carga, se han cargado los datos en las tablas intermedias. Para ello se ha utilizado un componente “tmap” para realizar el “*matching*, Para entender bien este proceso, es conveniente saber que el “*input”* del tmap solo acepta un flujo principal (“*main flow”)*, los distintos flujos de datos que intervengan adicionalmente actuarán como “*lookup flow”*. En función del tipo del procesado y operaciones sobre el *“main flow”* y *“lookup flow”,* hay que configurar correctamente una serie de parámetros en el “*lookup flow”* como se puede observar en la siguiente ilustración:

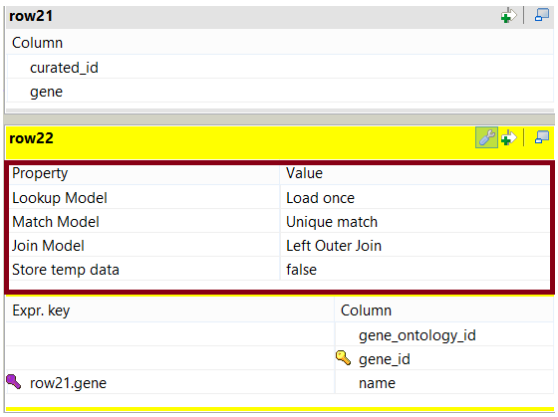


Ilustración 25 - Configuración "lookup flow"

El primero de los parámetros, *"lookup Model"* es utilizado para establecer el método de carga. En este caso, se hará una única carga de los datos para todo el *"main flow"*. El segundo de los parámetros *"Match Model"* es utilizado para establecer el método de *"matching"*. En este caso, se ha configurado para que el último dato en el que la *"Expression Key"* sobre el *"main flow"* haga *"matching"* con la *"Column"* del *"lookup flow"*. El tercer parámetro, *"Join Model"* es utilizado para establecer el tipo de unión a establecer. En este caso, se ha escogido el método *"Left Outer Join"*, retornando la pareja de todos los valores del *"main flow"* con los valores del *"lookup flow"* correspondiente. El último de los parámetros *"Store temp data"* es utilizado indicar si se desea almacenar el *"output"* del flujo de datos en memoria. En este caso, se le ha dado el valor de *"false"*.

De forma análoga a la ilustración 19, correspondiente a las tablas "Gene" y "Is_located", ocurre con las tablas "Phenotype" y "Predispose_to". En la siguiente ilustración puede apreciarse todo el flujo de datos generado en esta última fase de carga.

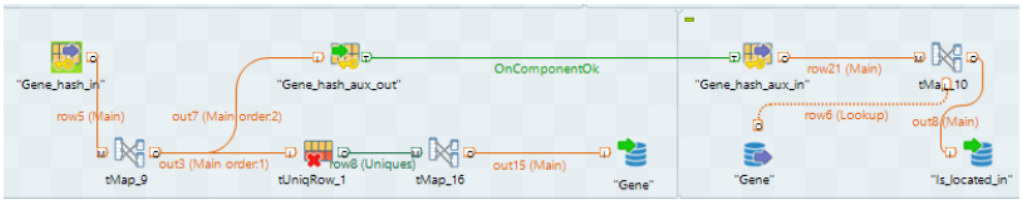


Ilustración 26 - Escenario de carga, casuística 4

Es destacable el uso de "OnComponentOk" como disparador, ya que cuando se utilizan tablas hash es necesario esta configuración para asegurar que cuando se accede a la estructura, no se está escribiendo sobre esta, de forma que la lectura sea coherente.

La lógica aplicada consiste en:

1. Utilizar las tabla hash como *"main flow"* y las tabla "Gene"/"Phenotype" como *"lookup flow"*.
2. Incluir en la "Expression Key" los valores de la columna "name" de la tabla hash sobre la *"Column"* "name" de la tabla "Gene"/"Phenotype".
3. Volcar sobre la tabla "Is_located_in"/"Predispose_to" en el caso de *"matching"* el valor de la columna "curated_id" de la tablas hash y el valor de la columna "x_id" de la tabla "Gene"/"Phenotype".

Solo queda abordar la última casuística correspondiente a la tabla "Curated_Variation". Como se ha comentado en más de una ocasión, es la tabla principal de la base de datos y está relacionada con la mayoría de tablas de esta (véase sección 4.2). Se ha querido separar de las distintas casuísticas porque era necesario concretar algunos aspectos importantes.

Todos los datos pertenecientes a esta tabla se cargaron de forma inmediata excepto el campo “chromosome” (véase sección 4.2). Una de las decisiones más importantes con respecto a esta tabla ha sido la gestión de las FK ya que el enfoque podía condicionar y afectar directamente tanto al rendimiento como a futuras limitaciones (véase sección 7). Ya que el proyecto no se ha desarrollado para un posterior uso directo se ha decidido gestionar las FK de la forma más eficiente y evitando siempre en la medida de lo posible las consultas a la base de datos. Dado que las tablas afectadas por la FK se corresponden con las tablas sobre las cuales los datos han sido “hardcodeados” nos ha aportado una gran ventaja y es que de previamente a la ejecución eran conocidos tanto los valores como las PK de estas tablas, por tanto, mediante el uso de pequeñas funciones se ha podido evitar las lecturas de la base de datos para establecer el “*matching”* entre la tabla “Curated_Variation” y estas tablas. En la mayoría de los casos siempre será más eficiente el tratamiento de cadenas de texto que la realización de operaciones sobre la base de datos. Las tablas afectadas por la FK son: “Clinical_Significance”, “Variation_Type”, “Chromosome”. Para la tabla Clinical_Significance la función utilizada ha sido la siguiente:

```
(Integer.parseInt(StringHandling.LEFT(StringHandling.RIGHT(StringHandling.L
EFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHan
dling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),";")),StringHandling.LEN(StringH
andling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO
)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHan
dling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),";")))-
StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,
StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHan
dling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),";")),"=")-
1),1))<100)?Integer.parseInt(StringHandling.LEFT(StringHandling.RIGHT(String
Handling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INF
O)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHan
dling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),";")),StringHandling.LEN(StringH
andling.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO
)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHan
dling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),";")))-
StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,
StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),StringHandling.INDEX(StringHan
dling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"CLNSIG")),";")),"=")-1),1))+1:9
```

Donde para asignar el “id” solo ha sido sumar “1” al valor recibido a través del sub-campo “CLNSIG” exceptuando en último caso (con valor 255) que se ha asignado el valor “9”.

Para la tabla “Variation_Type” la función utilizada ha sido la siguiente:

(StringHandling.RIGHT(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO
,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")),StringHandling.LEN(StringHandli
ng.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")))-
StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,
StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")),"=")-
1).equals("SNV"))?1:(StringHandling.RIGHT(StringHandling.LEFT(StringHandlin
g.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")),StringHandling.LEN(StringHandli
ng.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")))-
StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,
StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")),"=")-
1).equals("DIV"))?2:(StringHandling.RIGHT(StringHandling.LEFT(StringHandlin
g.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")),StringHandling.LEN(StringHandli
ng.LEFT(StringHandling.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")))-
StringHandling.INDEX(StringHandling.LEFT(StringHandling.RIGHT(row1.INFO,
StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),StringHandling.INDEX(StringHandling
.RIGHT(row1.INFO,StringHandling.LEN(row1.INFO)-
StringHandling.INDEX(row1.INFO,"VC")),";")),"=")-1).equals("MNV"))?3:0

En este caso el rango de datos era muy limitado (“SNV”,”MNV” y “DIV) con lo cual con una serie de sentencias “if” fácilmente las FK fueron asignadas.

Por último, en el caso de la tabla “Chromosome”, la función utilizada es la siguiente:

(row1._CHROM.equals("MT"))?23:(row1._CHROM.equals("X"))?24:(row1._CHROM.equals("Y"))?25:Integer.parseInt(row1._CHROM).

En este caso la FK ha sido fácilmente extraíble ya que el número del cromosoma se correspondía salvo en los casos en los que estos venían representados por un “char” con el “id” del elemento. Para los casos donde el elemento poseía un valor de tipo “char” se asignó el valor “id” correspondiente siendo este coherente con el “id” introducido en la tabla “Chromosome” para el elemento en cuestión. La función es exactamente la misma que la utilizada en la sección 4.2 y aunque la información esté replicada se ha decidido mantener ambos casos. El hecho de que la información esté replicada es debido a que el “id” de los elementos de la tabla “Chromosome” no solo identifica a un cromosoma en sí sino que también representa a una secuencia de referencia genómica, pero para este caso, solo se incluye una secuencia: “Grc38/Hg20”, por tanto el “id” es fácilmente extrapolable.

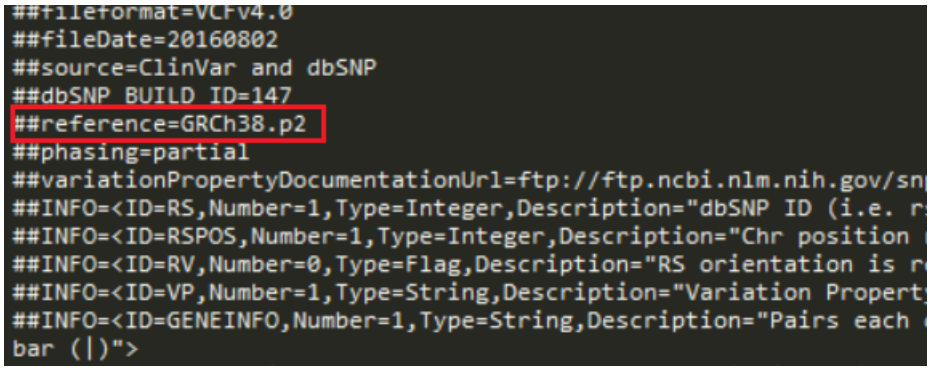


Ilustración 27 - Clinvar, secuencia de referencia

Mediante este enfoque se ha conseguido mejorar el rendimiento del proceso de carga pero sí que es cierto que en el caso en el que se produzcan cambios sobre los datos “hardcodeados” haría falta hacer cambio sobre el proceso ETL para adaptar este a las nuevas circunstancias y/o casuísticas.

4.3.2. Base de datos no relacional, MongoDB

Antes de comenzar con el proceso de carga en Talend, al igual que en el caso de PostgreSQL (véase sección 4.3.1) ha sido necesario crear, así como configurar la base de datos sobre la que se establecerá posteriormente la conexión en Talend.

En la máquina anfitriona se ha instalado el servidor de MongoDB, concretamente la versión 3.2. Antes de ejecutar el servidor es conveniente revisar el fichero de configuración para revisar que los parámetros son correctos. En este caso, se ha ejecutado con los valores por defecto, siendo el host: "localhost" y el puerto: "27017". Una vez el servidor está correctamente configurado ya puede ser lanzado quedando a la espera de recibir alguna conexión:

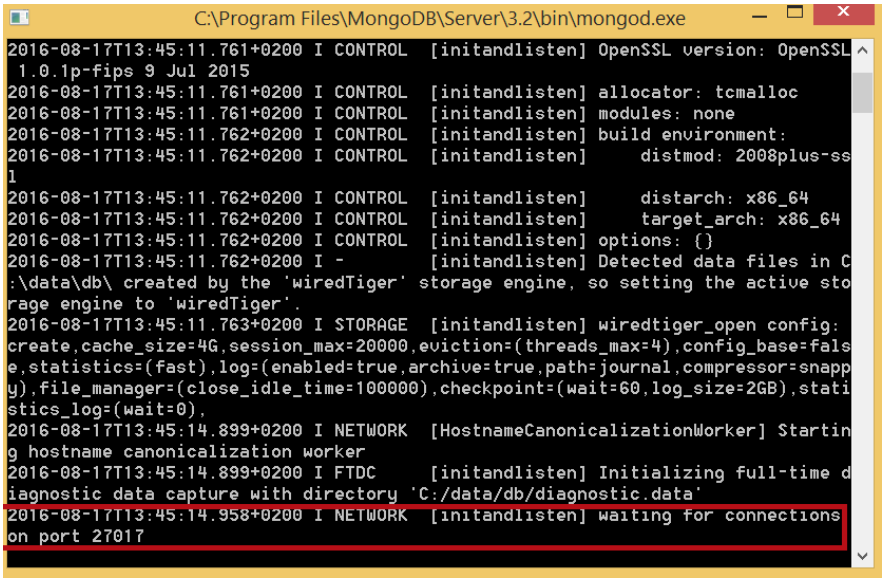


Ilustración 28 - MongoDB Server

El siguiente paso es abrir la consola cliente de MongoDb que conectará con el servidor y así crear la base de datos sobre la que se cargarán los datos. Para crear la base de datos se utiliza la siguiente query: use "nombre de la base de datos". En este caso "use tfm". Una vez la base de datos ha sido creada no es necesario crear las colecciones ya que directamente en Talend estas pueden ser creadas.

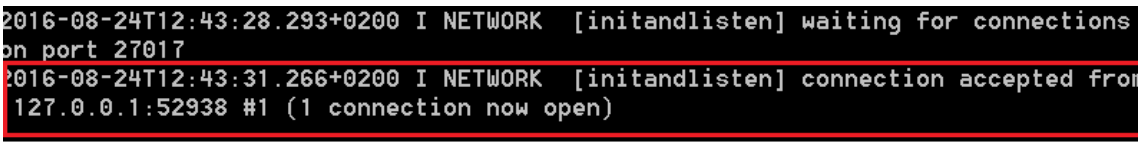


Ilustración 29 - MongoDB, cliente conectado

Para establecer la conexión desde Talend, hay que realizar los siguientes pasos:

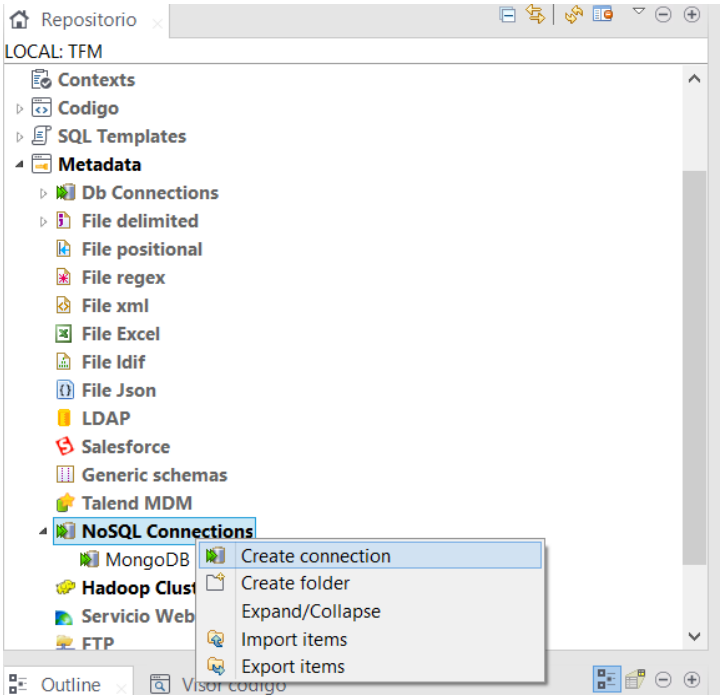


Ilustración 30 - Conexión NoSQL, paso 1

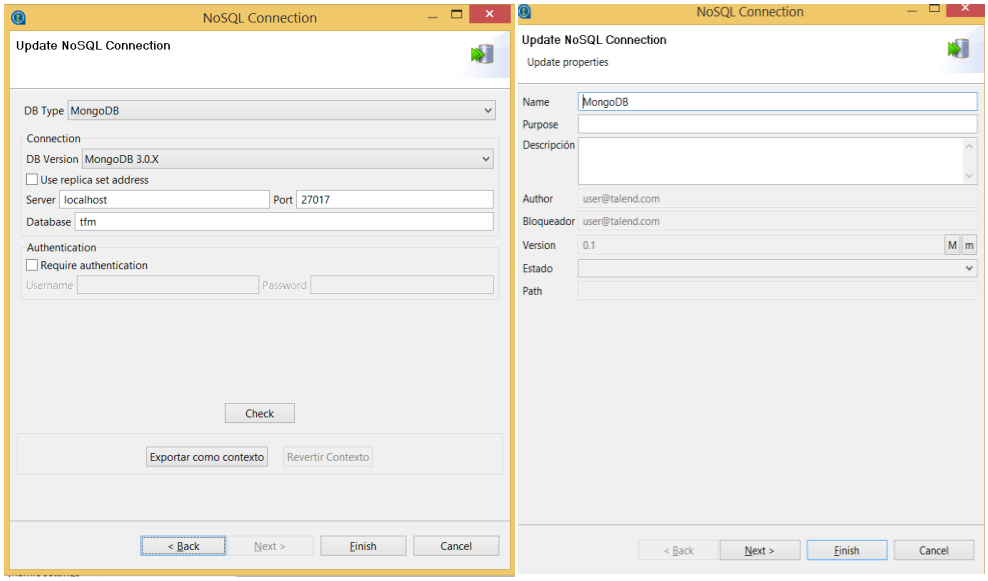


Ilustración 31 - Conexión NoSQL, paso 2, 3

Antes de seleccionar “Finish” es conveniente realizar el “Check” para verificar que los parámetros son correctos, aunque a diferencia que en PostgreSQL, si el nombre de la base de datos es incorrecto, el *check* seguirá afirmando que la conexión es correcta, no obstante, cualquier instanciación de alguna componente bajo esta conexión retornará un error.

Una vez finalizada la conexión ya pueden ser las componentes sobre MongoDB instanciadas en el proyecto. En este caso, el componente utilizado ha sido “tMongoDBOutput” únicamente. La configuración de los parámetros del componente será heredada de la propia conexión, por tanto lo único que será necesario especificar es la colección sobre la que se cargarán los datos, creando esta si no ha sido creada previamente. Para asegurar que la carga se realiza limpiamente es necesario marcar la opción “Drop collection if exist” para limpiar aquellos datos previamente almacenados. Por último, se crean las columnas que definirán el “esquema” de la base de datos para la colección en cuestión.

Si no se está familiarizado con las tecnologías NoSQL es conveniente repasar la sección 2, donde se explica con detalle cómo funcionan y estructuran este tipo de bases de datos antes de seguir con el proceso.

A la hora de realizar la carga sobre la base de datos de MongoDB había distintos enfoques e implementaciones. En este caso, se ha comenzado por generar las “colecciones” que posteriormente serán cargadas en la base de datos. Las colecciones generadas están en formato “CSV”, también hubiera sido posible generar estas en formato “XML” y JSON”, ya que el ítem “tMongoDBOutput” admite estos formatos como entrada. Aunque JSON es un formato más afín a MongoDB se ha escogido CSV debido a que los datos inicialmente estaban en formato TSV por lo que era lo más sensato.

Tanto el proceso de extracción como transformación es exactamente el mismo que en el caso de PostgreSQL. Para realizar la carga ha sido necesario el uso masivo de tablas hash para almacenar datos parciales correspondientes a las futuras colecciones a generar y cargar en la base de datos.

El escenario es muy similar al generado en el caso de la carga en PostgreSQL, aun así la siguiente ilustración muestra un ejemplo para corroborar esta afirmación. La ilustración escogida es análoga a la ilustración 27, correspondiente a la gestión de la información de la entidad “Gene” y “Is_located_in”

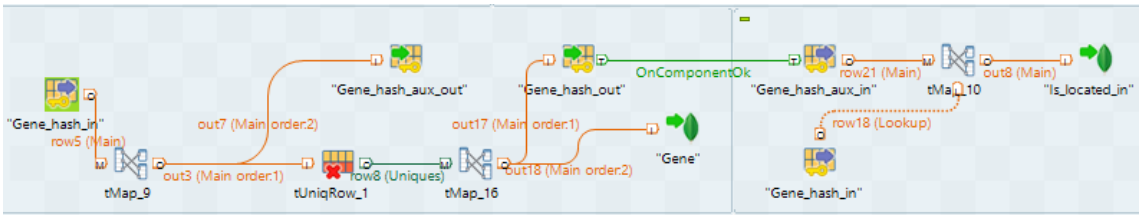


Ilustración 32 - Escemario carga NoSQL (Gene-Is_located_id)

No se ha entrado en detalle sobre las distintas casuísticas al igual que en el caso de PostgreSQL pero es cierto que como ha quedado demostrado con la ilustración anterior el procedimiento y la metodología es exactamente la misma.

5. Resultados

El resultado tras el proceso ETL ha sido la carga completa del fichero de Clinvar en la base de datos de PostgreSQL y MongoDB. Para corroborar el trabajo realizado así como la coherencia de los datos se han incluido las siguientes ilustraciones donde se puede apreciar la coherencia entre bases de datos y el fichero Clinvar de entrada, mostrando a la izquierda la base de datos de MongoDB y a su derecha la base de datos de PostgreSQL.

```
> db.Chromosome.find()
{ "_id" : 1, "name" : "chr1", "reference_id" : "hg20" }
{ "_id" : 2, "name" : "chr2", "reference_id" : "hg20" }
{ "_id" : 3, "name" : "chr3", "reference_id" : "hg20" }
{ "_id" : 4, "name" : "chr4", "reference_id" : "hg20" }
{ "_id" : 5, "name" : "chr5", "reference_id" : "hg20" }
{ "_id" : 6, "name" : "chr6", "reference_id" : "hg20" }
{ "_id" : 7, "name" : "chr7", "reference_id" : "hg20" }
{ "_id" : 8, "name" : "chr8", "reference_id" : "hg20" }
{ "_id" : 9, "name" : "chr9", "reference_id" : "hg20" }
{ "_id" : 10, "name" : "chr10", "reference_id" : "hg20" }
{ "_id" : 11, "name" : "chr11", "reference_id" : "hg20" }
{ "_id" : 12, "name" : "chr12", "reference_id" : "hg20" }
{ "_id" : 13, "name" : "chr13", "reference_id" : "hg20" }
{ "_id" : 14, "name" : "chr14", "reference_id" : "hg20" }
{ "_id" : 15, "name" : "chr15", "reference_id" : "hg20" }
{ "_id" : 16, "name" : "chr16", "reference_id" : "hg20" }
{ "_id" : 17, "name" : "chr17", "reference_id" : "hg20" }
{ "_id" : 18, "name" : "chr18", "reference_id" : "hg20" }
{ "_id" : 19, "name" : "chr19", "reference_id" : "hg20" }
{ "_id" : 20, "name" : "chr20", "reference_id" : "hg20" }
Type "it" for more
```

	chromosome_id integer	name character varying	reference_id character varying
1	1	chr1	hg20
2	2	chr2	hg20
3	3	chr3	hg20
4	4	chr4	hg20
5	5	chr5	hg20
6	6	chr6	hg20
7	7	chr7	hg20
8	8	chr8	hg20
9	9	chr9	hg20
10	10	chr10	hg20
11	11	chr11	hg20
12	12	chr12	hg20
13	13	chr13	hg20
14	14	chr14	hg20
15	15	chr15	hg20
16	16	chr16	hg20
17	17	chr17	hg20
18	18	chr18	hg20
19	19	chr19	hg20
20	20	chr20	hg20

Ilustración 33 - Resultados "Chromosome"

```
> db.Curated_Variation.find()
{ "_id" : 1, "chromosome" : 1, "position" : 1014143, "reference" : "C", "allele"
: "T", "hgvs_notations" : "NC_000001.11:g.1014143C>T", "variation_type_id" : 1
"clinical_significance_id" : 6, "chromosome_id" : 1 }
{ "_id" : 2, "chromosome" : 1, "position" : 1014316, "reference" : "C", "allele"
: "CG", "hgvs_notations" : "NC_000001.11:g.1014319dupG", "variation_type_id" :
2, "clinical_significance_id" : 6, "chromosome_id" : 1 }
{ "_id" : 3, "chromosome" : 1, "position" : 1014359, "reference" : "G", "allele"
: "T", "hgvs_notations" : "NC_000001.11:g.1014359G>T", "variation_type_id" : 1,
"clinical_significance_id" : 6, "chromosome_id" : 1 }
{ "_id" : 4, "chromosome" : 1, "position" : 1020217, "reference" : "G", "allele"
: "T", "hgvs_notations" : "NC_000001.11:g.1020217G>T", "variation_type_id" : 1,
"clinical_significance_id" : 3, "chromosome_id" : 1 }
{ "_id" : 5, "chromosome" : 1, "position" : 1020239, "reference" : "G", "allele"
: "C", "hgvs_notations" : "NC_000001.11:g.1020239G>C", "variation_type_id" : 1,
"clinical_significance_id" : 1, "chromosome_id" : 1 }
{ "_id" : 6, "chromosome" : 1, "position" : 1022225, "reference" : "G", "allele"
: "A", "hgvs_notations" : "NC_000001.11:g.1022225G>A", "variation_type_id" : 1,
"clinical_significance_id" : 6, "chromosome_id" : 1 }
{ "_id" : 7, "chromosome" : 1, "position" : 1022260, "reference" : "C", "allele"
: "T", "hgvs_notations" : "NC_000001.11:g.1022260C>T", "variation_type_id" : 1,
"clinical_significance_id" : 4, "chromosome_id" : 1 }
{ "_id" : 8, "chromosome" : 1, "position" : 1022313, "reference" : "A", "allele"
: "T", "hgvs_notations" : "NC_000001.11:g.1022313A>T", "variation_type_id" : 1,
"clinical_significance_id" : 6, "chromosome_id" : 1 }
```

Ilustración 34 - Resultados, MongoDB "Curated_Variation"

curated_variation_id integer	chromosome integer	position integer	reference character varying	allele character varying	hgvs_notations character varying	variation_type_id integer	clinical_significance_id integer	chromosome_id integer
1	1	1014143	C	T	NC_000001.11:g.1014143C>T	1	6	1
2	1	1014316	C	CG	NC_000001.11:g.1014319dupG	2	6	1
3	1	1014359	G	T	NC_000001.11:g.1014359G>T	1	6	1
4	1	1020217	G	T	NC_000001.11:g.1020217G>T	1	3	1
5	1	1020239	G	C	NC_000001.11:g.1020239G>C	1	1	1
6	1	1022225	G	A	NC_000001.11:g.1022225G>A	1	6	1
7	1	1022260	C	T	NC_000001.11:g.1022260C>T	1	4	1
8	1	1022313	A	T	NC_000001.11:g.1022313A>T	1	6	1
9	1	1040679	C	T	NC_000001.11:g.1040679C>T	1	1	1
10	1	1041249	C	T	NC_000001.11:g.1041249C>T	1	3	1
11	1	1041582	C	T	NC_000001.11:g.1041582C>T	1	6	1

Ilustración 35 - Resultados, PostgreSQL "Curated_Variation"

```
##INFO=<ID=CLNACC,Number=.,Type=String,Description="Variant Accession and Versions">
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO
1   1014143 rs786201005 C   T   .   .
RS=786201005;RSPOS=1014143;dbSNPBuildID=144;SSR=0;SAO=1;VP=0x050060000605000002110100;
T;CLNSRC=OMIM_Allelic_Variant;CLNORIGIN=1;CLNSRCID=147571.
0003;CLNSIG=5;CLNDSDB=MedGen:OMIM;CLNDSDBID=CN221808:616126;CLNDBN=Immunodeficiency_38
GENEINFO=ISG15:9636;WGT=1;VC=SNV;PM;NSN;REF;ASP;LSD;OM;CLNALLE=1;CLNHGVS=NC_000001.11:g.1014143C>

;CLNREVSTAT=no_criteria;CLNACC=RCV000162196.3
```

Ilustración 36 - Resultados, Clinvar,#1

```
> db.Is_located_in.find()
{ "_id" : 1, "curated_variation_id" : 1, "gene_id" : 1 }
{ "_id" : 2, "curated_variation_id" : 2, "gene_id" : 1 }
{ "_id" : 3, "curated_variation_id" : 3, "gene_id" : 1 }
{ "_id" : 4, "curated_variation_id" : 4, "gene_id" : 4 }
{ "_id" : 5, "curated_variation_id" : 5, "gene_id" : 4 }
{ "_id" : 6, "curated_variation_id" : 6, "gene_id" : 4 }
{ "_id" : 7, "curated_variation_id" : 7, "gene_id" : 4 }
{ "_id" : 8, "curated_variation_id" : 8, "gene_id" : 4 }
{ "_id" : 9, "curated_variation_id" : 9, "gene_id" : 4 }
{ "_id" : 10, "curated_variation_id" : 10, "gene_id" : 4 }
{ "_id" : 11, "curated_variation_id" : 11, "gene_id" : 4 }
{ "_id" : 12, "curated_variation_id" : 12, "gene_id" : 4 }
{ "_id" : 13, "curated_variation_id" : 13, "gene_id" : 4 }
{ "_id" : 14, "curated_variation_id" : 14, "gene_id" : 4 }
{ "_id" : 15, "curated_variation_id" : 15, "gene_id" : 4 }
{ "_id" : 16, "curated_variation_id" : 16, "gene_id" : 4 }
{ "_id" : 17, "curated_variation_id" : 17, "gene_id" : 4 }
{ "_id" : 18, "curated_variation_id" : 18, "gene_id" : 4 }
{ "_id" : 19, "curated_variation_id" : 19, "gene_id" : 4 }
{ "_id" : 20, "curated_variation_id" : 20, "gene_id" : 4 }
```

curated_variation_id integer	gene_id integer	locate_id integer
1	1	1
2	1	2
3	1	3
4	4	4
5	4	5
6	4	6
7	4	7
8	4	8
9	4	9
10	4	10
11	4	11
12	4	12
13	4	13
14	4	14
15	4	15
16	4	16
17	4	17
18	4	18
19	4	19
20	4	20

Ilustración 37 - Resultados, "Is_located_in"

Las consultas realizadas sobre las bases de datos siguen la siguiente sintaxis:

- Mongodb: db.”nombre de la colección”.find()
- PostgreSQL: Select * From “nombre del esquema”. “nombre de la tabla”

Una vez el proceso de carga ha finalizado en ambas tecnologías es necesario evaluar el proceso para cada una de las bases de datos así como el comportamiento de Talend.

El equipo donde se ha ejecutado el proyecto cuenta con 8Gb de memoria RAM y un procesador Intel Core I7-4710HQ a 2,5 Ghz con 4 núcleos físicos y un total de 8 junto a los virtuales.

Teniendo en cuenta las limitaciones HW de las que se disponen para ejecutar el proceso completo, cabe destacar que se han tenido que hacer varias versiones del proceso para optimizar y mejorar los tiempos. La primera versión realizaba la carga del fichero de Clinvar completo sobre la base de datos de PostgreSQL en más de cinco horas. La última versión es la incluida en la memoria, dando unos tiempos máximos de 2 minutos en procesar las más de cien mil variaciones.

Las siguientes gráficas muestran el rendimiento de cada una de las tecnologías en función de la talla del problema siendo esta el número de variaciones a procesar.

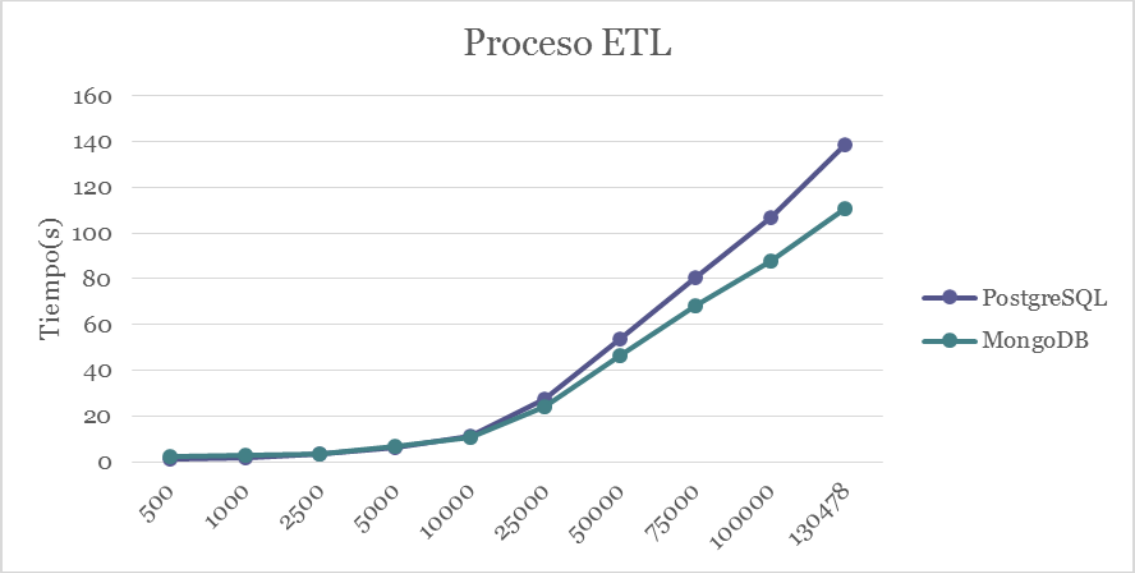


Ilustración 38 - Rendimiento PostgreSQL vs MongoDB

Cabe destacar que los resultados son ciertamente similares, pero un extenso análisis revela ciertos patrones de comportamiento lo cual puede ayudar a extrapolar y elaborar conclusiones. Realizando una interpretación del gráfico se puede observar que la base de datos PostgreSQL tiene mejores resultados cuando la talla es menor a 25000, a partir de este momento MongoDB comienza a sobrepasar en rendimiento a la base de datos de PostgreSQL.

La siguiente gráfica muestra el porcentaje de mejora de la carga de la base de datos de MongoDB frente a PostgreSQL. En esta gráfica se identifica más claramente el punto de inflexión en el rendimiento entre tecnologías. Es muy destacable que cuando la carga es muy ligera, la base de datos de MongoDB tiene un rendimiento muy inferior frente a la PostgreSQL, pero conforme la talla aumenta, la sobrecarga inicial no resulta significativa y en ambas gráficas queda totalmente reflejado.

Es destacable que conforme aumenta la talla MongoDB aumenta en rendimiento frente a PostgreSQL, aun tratándose de un fichero de entrada de 5Mb de peso y cerca de unas 150.000 entradas a procesar. Si se tratase de un entorno “big data” donde el fichero de entrada tuviese un peso de varios gigabytes sería absurdo en este tipo de sistemas recurrir a RDBMS ya que se acentuaría notoriamente la clara desventaja de estas frente a bases de datos NoSQL y más concretamente con respecto a MongoDB.

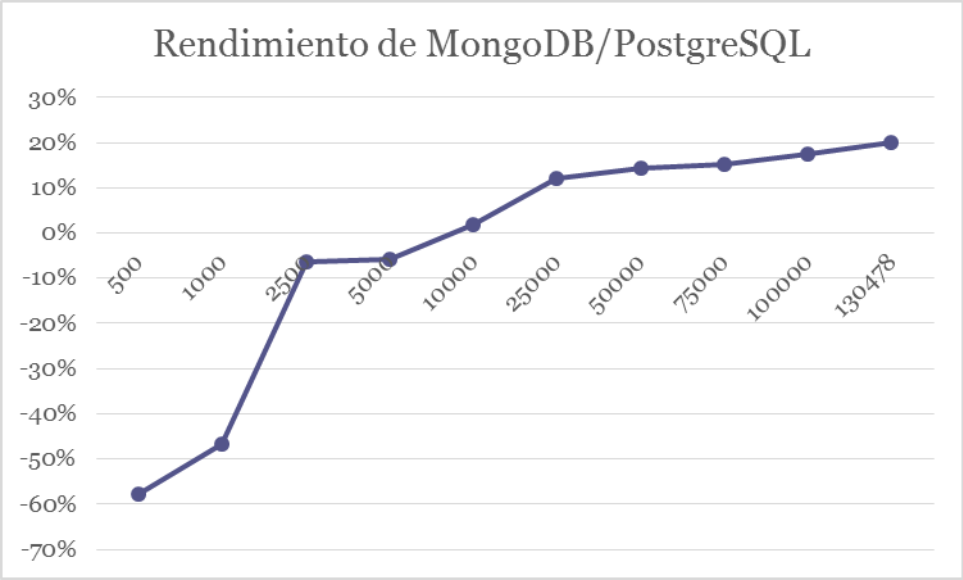


Ilustración 39 - Rendimiento MongoDB/PostgreSQL

Algo relevante y determinante ha sido el hecho de que la carga no se ha realizado en una única fase. Hay un total de 7 subprocesos en los cuales se llevan a cabo diversas cargas, afectando negativamente al rendimiento en el caso de la base de datos de MongoDB. Aun siendo así, al tratarse de tallas tan pequeñas se acentúa este hecho sobre todo en el caso de los datos insertados vía "harcoding" ya que son datos fijos que no aumentan de volumen con respecto a la talla del problema. En cambio, en aquellos subprocesos donde el volumen de datos de carga es superior a las 150.000 unidades el rendimiento de la base de datos de MongoDB es claramente superior. Ambas casuísticas quedan claramente reflejadas en las siguientes ilustraciones:

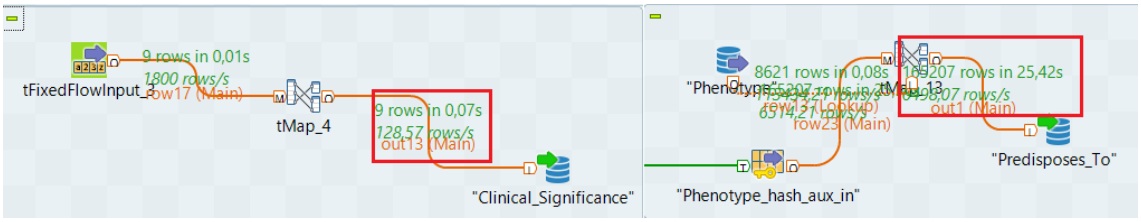


Ilustración 40 - Evaluación, casuísticas PostgreSQL

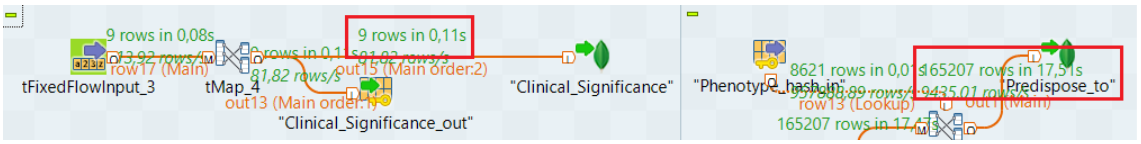


Ilustración 41 - Evaluación, casuísticas MongoDB

Cabe destacar, que cuando las tallas son pequeñas aunque la base de datos de PostgreSQL tenga un rendimiento mucho mayor la diferencia es de centésimas de segundos, pero en el caso de hablar de tallas grandes se podría hablar de minutos o incluso horas.

Es de valorar positivamente que el estudio teórico realizado en la sección 2.2 se corresponde con el resultado obtenido en el caso práctico, sobre todo en aspectos de escalabilidad y rendimiento.

6. Conclusiones

Los objetivos del proyecto han sido satisfactoriamente alcanzados tal y como quedaba descrito en los objetivos de este. Gracias a este estudio, análisis así como el hecho de haber afrontado un problema real ha ayudado a ampliar los conocimientos en el ámbito de las TIC, cada vez más presentes en las organizaciones aportando un valor añadido así como un fuerte alineamiento con los requerimientos y objetivos del modelo de negocio de la organización.

El estado del arte realizado tanto a nivel de herramientas ETL como de distintos sistemas de gestión y tecnologías de base de datos aportando un amplio conocimiento a la hora de afrontar nuevos retos y desafíos en el ámbito de los sistemas y tecnologías de la información.

En cuanto a la herramienta ETL escogida, “Talend Open Studio for Big Data” ha demostrado ser competente y completa, no restando prestaciones que afecten al rendimiento directamente en su versión gratuita. La elección de esta herramienta ha sido claramente acertada, tanto por su implementación en lenguaje Java como a por su amplia variedad de componentes, sobre todo en la versión “Big Data”, aportando componentes específicos que han resultado de gran utilidad, como los empleados en la carga de datos en MongoDB.

Los sistemas e implementaciones de bases de datos NoSQL analizados han aportado un enriquecimiento de los conocimientos en esta área. El conocimiento detallado de la arquitectura y naturaleza de las bases de datos NoSQL aportan una gran ventaja a la hora de adecuarse a proyectos futuros, que tal y como las tendencias indican, cada vez más se desarrollan en ámbitos donde la elección de la tecnología e implementación de la base de datos es un decisión crucial para asegurar un funcionamiento y rendimiento adecuado. La elección de MongoDB como base de datos NoSQL ha sido una buena elección ya que la flexibilidad estructural aportada por las bases de datos documentales ha sido idónea para almacenar información parcialmente estructurada como la presente en el fichero de entrada de Clinvar.

Tras analizar los resultados de la ejecución se puede concluir que la metodología empleada en las tareas de diseño e implementación en el proceso ETL ha sido correcta. Los valores de rendimiento calculados tras la ejecución corroboran y confirman lo antes expuesto de forma teórica. Aquellos sistemas de información que almacenen grandes volúmenes de datos son considerablemente más agiles implantados sobre base de datos no relacionales. En contraposición, es destacable que rememorando al teorema CAP, las bases de datos NoSQL son débiles en términos de consistencia pero robustas en disponibilidad y tolerancia frente a particionados. Las demandas actuales requieren de alta disponibilidad (servicios 24/7), ubicuidad de los datos y alta versatilidad estructural (mobile computing & IoT), sin olvidar la gestión eficiente de bases de datos de grandes volúmenes (Big Data). El objetivo por tanto, no ha sido concluir que tecnología es mejor o peor, sino analizar y plasmar que las necesidades tecnológicas cambian constantemente y que para ello es necesario contar con un experto en sistemas que tenga un conocimiento amplio y fresco sobre las tendencias tecnológicas que mejor se adapten en cada instante y/o escenario.

La mentalidad clásica de implantar un servidor que aloje una base de datos perpetuamente es un error, así como la idea de implantar una base de datos sin un análisis previo de requisitos.

En primer lugar, la era tecnológica actual es constantemente dinámica y totalmente susceptible a grandes cambios a nivel de tendencias. Adecuar la tecnología en cada momento es imprescindible, por ello haber explorado alternativas a sistemas clásicos de gestión y almacenamiento y haber presentado herramientas ETL como alternativa frente al desarrollo tradicional ha sido un buen punto de partida para poder iniciar una carrera como analista de sistemas en esta era cambiante.

Por otro lado, el modo de proceder en este proyecto refleja la importancia del modelado conceptual. Las TIC no están presentes en un único departamento como otra área funcional sino que están alineadas con la dirección estratégica. El modelo conceptual es el punto de encuentro entre el marco estratégico empresarial y las tecnologías que dan soporte a este.

Por tanto, los procedimientos y pautas empleados han conformado el conjunto de buenas prácticas para afrontar proyectos complejos en el ámbito de las TIC de sólida y eficientemente.

7. Limitaciones y futuras mejoras

Aunque el proceso ETL haya finalizado con éxito hay algunas limitaciones que cabe mencionar. Las limitaciones intrínsecas de las bases de datos no relacionadas deben ser tratadas, comúnmente a nivel de aplicación. En la base de datos de MongoDB no hay ningún control de integridad más que el del campo "_id" para cada elemento. Este hecho tiene ciertas repercusiones como permitir referencias a elementos repetidas o que no existen.

Por otro lado, con el fin de mejorar el rendimiento de ambas bases de datos aquella información que permanecía estática fue introducida "harcodeada" en lugar de inyectarse consultas sobre la base de datos o haber implementado alguna rutina que obtuviera los datos necesarios durante la etapa de transformación. Este hecho provoca que cualquier cambio en las estructuras afectadas devolvería un fallo al ejecutar el programa.

El hecho de contar no solo con un modelo conceptual únicamente sino con un esquema lógico y físico ha limitado las posibilidades de realizar una implementación distinta a la proporcionada. El hecho de que la carga de datos se realizaba en base a un esquema reducido es posible que otras implementaciones hubieran sido en este caso más acertadas. Por otro lado, la tecnología sobre la que está implementado el esquema es PostgreSQL. Un estado del arte sobre tecnologías RDBMS hubiera sido interesante de cara a escoger quizás una tecnología más adecuada.

El hardware ha sido otro de las limitaciones que ha impedido realizar pruebas en el ámbito de "Big Data" donde se hubiera apreciado cambios realmente sustanciales entre ambas tecnologías en cuestiones de rendimiento.

No había ningún presupuesto asignado al proyecto con lo cual no se han explorado herramientas de pago que quizás hubieran sido más adecuadas o hubieran ofertado algún tipo de servicio adicional. Como consecuencia directa, no ha habido ningún tipo de soporte técnico concreto de las herramientas utilizadas, no obstante, la comunidad de usuarios es muy amplia y los foros oficiales han sido de gran utilidad en algunos casos.

Por último, el hecho de no haber trabajado anteriormente en proyectos en el ámbito de las TIC, así como no haber utilizado herramientas ETL ha sido un gran desafío y pese al éxito del proyecto la falta de experiencia ha sido un factor realmente relevante.

En cuanto a las futuras mejoras, cabe destacar que se han planteado únicamente a nivel descriptivo, pero no hay ningún tipo de implementación adicional de estas. Hay dos vías claramente diferenciadas sobre las que se han enfocado las mejoras, siendo la primera de estas incidir sobre las limitaciones previamente expuestas y la segunda ampliar la funcionalidad con la posibilidad de ampliar los objetivos del proyecto, así como abrir nuevos horizontes sobre los que trabajar e investigar en futuros proyectos.

Comenzando por la primera casuística, la primera medida a implementar sería incluir directivas adicionales en la capa de aplicación sobre la que delegar la

integridad en el caso de la base de datos de MongoDB para evitar que en el caso de añadir nuevos elementos la base de datos acepte valores que violen la integridad y coherencia de esta.

Se podría realizar una implementación lógica distinta que se adaptase mejor al problema, por supuesto basada en el mismo modelo conceptual. Como ya se ha expuesto a lo largo del proyecto el tener un modelo conceptual implementado posibilita tener diversos esquemas que respondan a un mismo modelo y se alinee con los requerimientos de negocio.

En referencia a la segunda casuística, sería interesante en colaboración con el grupo investigador PROS poder ejecutar en algún computador de gran potencia un proceso ETL bajo un escenario "Big Data" con la finalidad de analizar la respuesta de las tecnologías de base de datos empleadas en un entorno exigente.

Sería interesante una vez los datos fueran cargados utilizar alguna herramienta en la capa de presentación de los datos como puede ser SpagoBI. La herramienta mencionada tiene la posibilidad de ser integrada con el software de Talend con lo cual sería muy interesante aunar esta de cara a un futuro proyecto. Aportaría una fuerte componente de BI (Business Intelligence) en términos de análisis ofreciendo soluciones para la presentación de informes, análisis multidimensional, así como análisis OLAP (On-Line Analitical Proccesing). A través de la amplia variedad de herramientas en al área de análisis tales como: Reportes, Gráficos, Office Documents puede mejorar la experiencia de "Data mining"

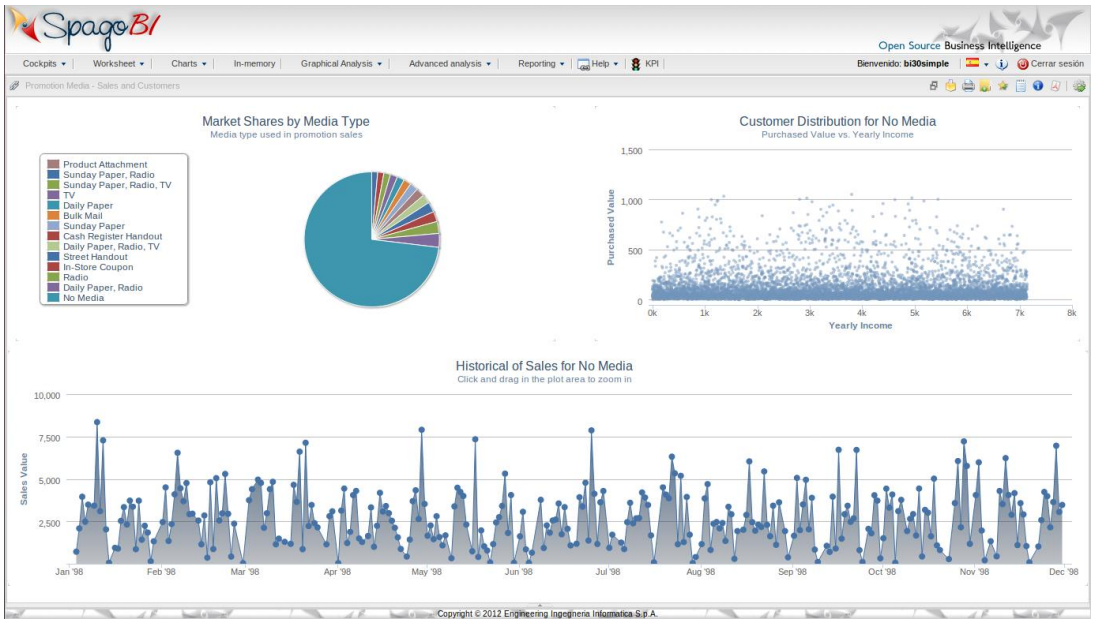


Ilustración 42 - Herramienta SpagoBI

Bibliografía

Acens. (15 de Febrero de 2014). *Acens's Nosql whitepaper*. Obtenido de http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf

Couchbase. (s.f.). *Why NoSQL?* Obtenido de http://www.couchbase.com/nosql-resources/why-nosql

Laurie R. Margolies, G. P. (2016). Breast Imaging in the Era of Big Data: Structured Reporting and Data Mining.

MongoDB. (22 de Octubre de 2013). *Seven Bridge Genomics*. Obtenido de https://www.mongodb.com/press/seven-bridges-genomics-relies-mongodb-next-generation-sequencing-analysis

MongoDB. (26 de Junio de 2014). *Best Of Both Worlds: Genentech Accelerates Drug Research With MongoDB & Oracle*. Obtenido de https://www.mongodb.com : https://www.mongodb.com/blog/post/best-both-worlds-genentech-accelerates-drug-research-mongodb-oracle

MongoDB. (26 de Junio de 2014). *Genetech's MongoDB technology*. Obtenido de http://web.archive.org/web/20150906030223/https://www.mongodb.com/blog/post/best-both-worlds-genentech-accelerates-drug-research-mongodb-oracle?c=655acadf87

MongoDB. (11 de Septiembre de 2015). *Production Deployment of MongoDB*. Obtenido de http://web.archive.org/web/20150911161929/https://www.mongodb.com/who-uses-mongodb

NBCI. (3 de Marzo de 2015). *Clinvar Archive*. Obtenido de http://www.ncbi.nlm.nih.gov/clinvar/intro/

Trivedi, A. (6 de Febrero de 2014). *http://code.tutsplus.com/*. Obtenido de http://code.tutsplus.com/articles/mapping-relational-databases-and-sql-to-mongodb--net-35650

Wikipedia. (Agosto de 2014). *https://en.wikipedia.org*. Obtenido de https://en.wikipedia.org/wiki/Business_intelligence

Wikipedia. (27 de Octubre de 2015). *https://en.wikipedia.org*. Obtenido de https://en.wikipedia.org/wiki/Apatar

Wikipedia. (30 de Abril de 2015). *https://es.wikipedia.org*. Obtenido de https://es.wikipedia.org/wiki/SpagoBI

Wikipedia. (29 de Marzo de 2016). *ETL Wiki*. Obtenido de https://en.wikipedia.org/wiki/Extract,_transform,_load

Wikipedia. (6 de Agosto de 2016). *https://en.wikipedia.org*. Obtenido de https://en.wikipedia.org/wiki/CloverETL

Wikipedia. (30 de Marzo de 2016). *https://en.wikipedia.org*. Obtenido de https://en.wikipedia.org/wiki/Scriptella

Wikipedia. (19 de Febrero de 2016). *NoSQL Wiki*. Obtenido de https://es.wikipedia.org/wiki/NoSQL

Wikipedia. (s.f.). *Base de datos documental*. Obtenido de https://es.wikipedia.org/wiki/Base_de_datos_documental

8. Anexo

En esta sección se ha incluido el mapa completo del proceso ETL para ambas tecnologías.

8.1. Escenario MongoDB

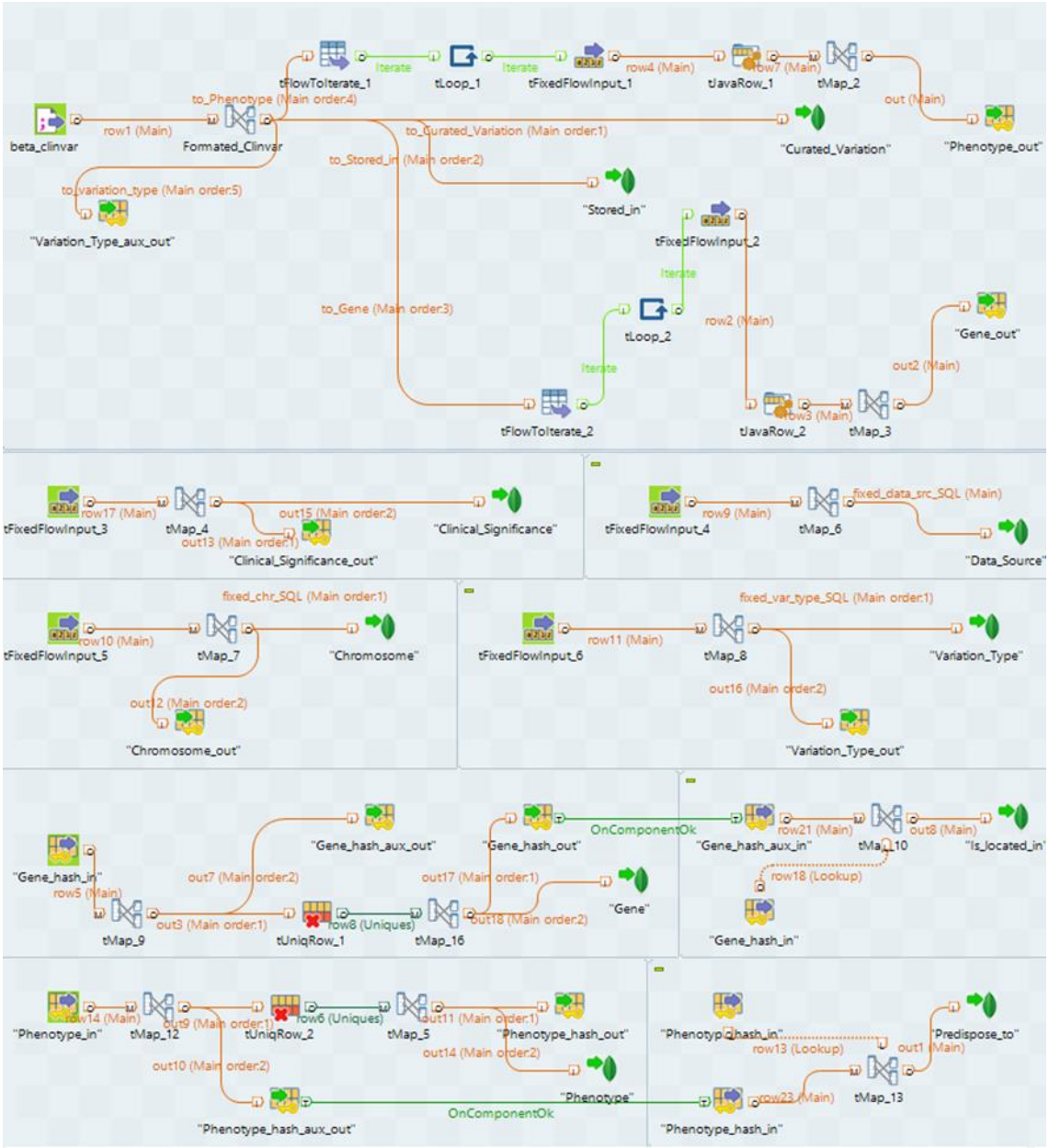


Ilustración 43 - Escenario MongoDB

8.2. Escenario PostgreSQL

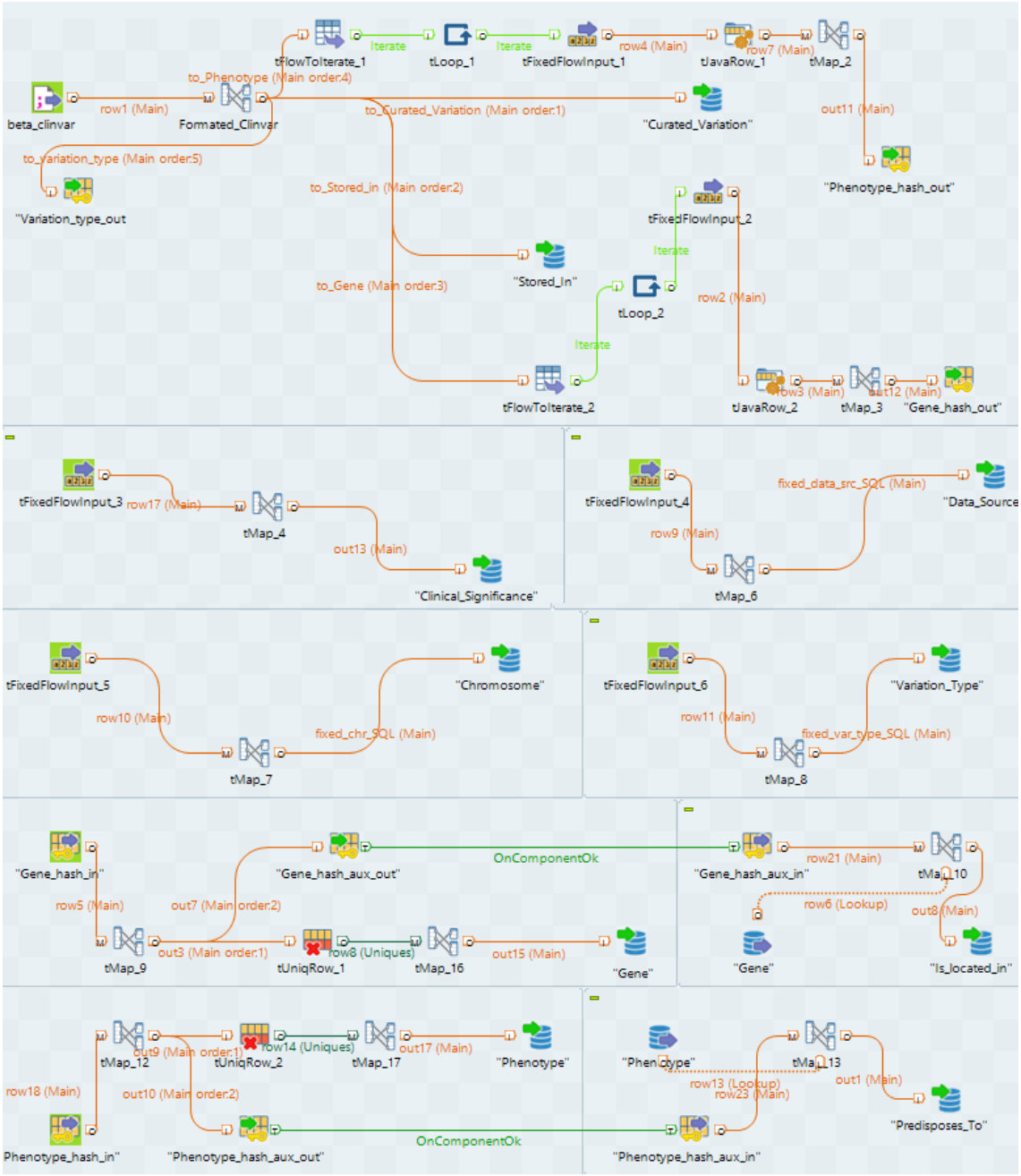


Ilustración 44 - Escenario PostgreSQL