Aspectos de calidad en la construcción de Web Warehouses con BPMS

Abril 2016

Pablo Mathías Barceló Romero
Diego Ricardo Pérez Bernardi

Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay

Tutores: Dr. Ing. Andrea Delgado y Dr. Ing. Adriana Marotta

Resumen

En el intercambio de información a través de la web entre gobiernos, ciudadanos y entre todo tipo de entes, la información compartida puede ser en muchos casos crítica, por ello, surge la necesidad de conocer, controlar y mejorar, cuando es posible, la calidad de calidad de datos sobre la información que se ofrece y consume. Además es imprescindible contar con una herramienta que facilite la integración y visualización de dichos datos logrando analizar y tomar decisiones a partir de los mismos. Estos datos no necesariamente se encuentran en una sola fuente de datos sino que provienen de distintas fuentes y/o formatos, por lo tanto es fundamental integrar los datos adecuadamente.

Un Web Warehouse (WW) nos brinda esta funcionalidad ya que consolida datos obtenidos desde la web y actúa como intermediario entre la publicación de los datos y el usuario final procesando la información y agregando un valor de calidad a los mismos. La tarea de crear un WW es una tarea compleja, que involucra integrar datos, agregarles valor, transformarlos de ser necesario, y añadir una traza de la calidad a los datos. Un Proceso de Negocio nos permite dividir y organizar dichas tareas de forma eficiente agregando restricciones entre las mismas y definiendo un flujo de ejecución óptimo para obtener el resultado esperado.

El presente proyecto, denominado **'Aspectos de calidad en la construcción de WW con BPMS'** consiste en extender un proyecto anterior, en el cual se modelaron e implementaron dos procesos para dar soporte automático a la construcción de un WW. El proceso de Configuración que permite definir la configuración de datos para la construcción del WW fue completamente modelado e implementado, y el proceso de Carga que realiza la carga del WW utilizando dicha configuración fue definido en una versión inicial para probar las definiciones de la configuración. En el presente proyecto, se incluye el modelado e implementación completa del proceso de carga, agregando el manejo de la calidad en ambos procesos y mejorando la integración en el proceso de carga. En el proceso de Configuración sedefinecalidad tanto sobre los datos recién extraídos de la web como sobre el DW resultado y se definen servicios que empleen los metadatos de calidad para mejorar la integración de entidades. El proceso de Carga realiza las mediciones de calidad y propaga las mismas a largo del proceso, así como también muestra errores al usuario y permite resolver manualmente conflictos de integración de entidades.

El prototipo realizado fue verificado a través de un caso de estudio con datos abiertos del Reino Unido, dando resultados ampliamente satisfactorios teniendo en cuenta que se trata de una prueba de concepto.

Como trabajo a futuro se plantea mejorar la amigabilidad de la interfaz del prototipo implementado y continuar trabajando con la propagación de la calidad con el fin de incluir aquellos casos que la solución no contempla por su complejidad.

Palabras Claves: Data Warehouse,Web Warehouse, Sistema de Gestión de Procesos de Negocio (BPMS), Calidad de Datos

Contenido

1 - Introducción

En el presente capítulo se presenta la motivación y contexto del proyecto, los objetivos planteados, los resultados esperados y obtenidos, y una breve descripción del desarrollo del mismo.

1.1 Motivación

Dado el gran avance en la tecnología en los últimos años, y en particular, el uso de la web como medio masivo de comunicación, la información compartida puede ser en muchos casos crítica. En el intercambio de información entre gobiernos, ciudadanos y entre todo tipo de entes, los datos son la materia prima, por ello, surge la necesidad de conocer y controlarla calidad de datos que son materia prima de la información que se ofrece y consume. Además es imprescindible contar con una herramienta que facilite la integración y visualización de dichos datos para un usuario final. De este modo el usuario final puede analizar y tomar decisiones, que de otro modo le serían imposibles. Estos datos no necesariamente se encuentran en una sola fuente de datos sino que provienen de distintas fuentes y/o formatos, por lo tanto es fundamental integrar los datos adecuadamente.

Un Web Warehouse (WW) nos brinda esta funcionalidad ya que consolida datos obtenidos desde la web y actúa como intermediario entre la publicación de los datos y el usuario final procesando la información y agregando un valor de calidad a los mismos.

La tarea de crear un WW es una tarea compleja, que involucra integrar datos, agregarles valor, transformarlos de ser necesario, y añadir una traza de la calidad a los datos. Un Proceso de Negocio (PN) nos permite dividir y organizar dichas tareas de forma eficiente agregando restricciones entre las mismas y definiendo un flujo de ejecución óptimo para obtener el resultado esperado.

Basándose en este análisis previo, se define un PN para la construcción de un WW que está compuesto por dos fases, una primera fase en la cual el usuario determina las fuentes de datos web y la calidad a medir de los mismos entre otra cantidad de definiciones necesarias para la configuración del WW, y una segunda fase en la cual se realiza la carga de los datos. Esta propuesta surge de trabajos previos [2] en donde se plantearon las bases de la construcción de un WW enfocado en la calidad utilizando un Sistema de Gestión de PN (BPMS). La propuesta se dividió en dos proyectos, correspondiendo cada uno principalmente a las fases definidas. Un primer proyecto, que se desarrolló en el año 2014 [3], orientado a definir, modelar y ejecutar la configuración del proceso, y un segundo proyecto, que es el presentado en este documento, añadiendo manejo de calidad a ambos procesos para la construcción del WW. El manejo de calidad que se agrega al WW permite fundamentalmente la evaluación de la calidad de los datos recién extraídos de la web, la evaluación de la calidad de los datos cuando ya están cargados en el DW, y la integración de datos basada en la calidad de los mismos.

Por lo mencionado anteriormente, la principal motivación de este proyecto es la construcción de un WW enfocado en los aspectos de calidad a través de una plataforma

BPMS. Además, este proyecto se contextualiza en las líneas de investigación conjuntas entre los grupos COAL [4] y CSI [5], que incluye otras propuestas como una tesis de maestría, entre otras [2].

1.2 Objetivos del proyecto

En esta sección se describen los objetivos generales y específicos del proyecto.

Objetivo general (OG) - El objetivo general de este proyecto es agregar el manejo de los aspectos de calidad al proceso de construcción de un WW mediante la utilización de BPMS.

Los objetivos específicos del proyecto son:

Objetivo específico 1 (OE1) - Estudio de los procesos existentes y de temas relacionados con Web Warehouse, Procesos de Negocio, Calidad de Datos y Servicios asociados.

Objetivo específico 2 (OE2) - Extender el prototipo (modelado e implementación) de los Procesos de Configuración y Carga para la construcción del WW enfocándose ahora en la calidad de datos del mismo y en la resolución de conflictos.

Objetivo específico 3 (OE3) - Caso de estudio de aplicación del prototipo para un dominio específico (ej. turismo, salud).

1.3 Resultados esperados

A continuación se presentan los resultados de este proyecto.

Resultado Esperado 1 (RE1) - Haber estudiado los procesos existentes y los temas relacionados.

Resultado Esperado 2 (RE2) - Extensión del proceso de Configuración de WW implementado en BPMS Activiti [6] para incluir los aspectos de calidad: el usuario interesado deberá poder definir distintas métricas de calidad ya sea en la extracción de los datos como en el WW resultado, con lo cual se pretende encontrar problemas como errores de escritura en los datos, valores perdidos, datos obsoletos o inconsistentes. Al realizar esta extensión se espera que se traduzca las interfaces del prototipo y la base de Configuración del español al inglés y se dé espacio a oportunidades de mejora del prototipo existente.

Resultado Esperado 3 (RE3) - Extensión del proceso de Carga de WW implementado en BPMS Activiti añadiendo la medición y propagación de la calidad definida por el usuario en las fuentes de datos y la medición sobre el WW resultado.

Resultado Esperado 4 (RE4) - Mejora de la integración en la Carga del WW a través de la detección de entidades duplicadas, seleccionando las más confiables a partir de los metadatos de calidad. Se espera que el proceso sea lo más automatizable posible, dejando las actividades manuales para la resolución de conflictos o decisiones que el Sistema no puede tomar.

Resultado Esperado 5 (RE5) - Ejecución de un caso de estudio, que permita mostrar la factibilidad de la construcción y carga de un WW básico utilizando los procesos implementados. Este caso de estudio debe ser en inglés y el WW resultado debe contar con medidas que no sean sólo de contar registros.

1.4 Resultados alcanzados

En la siguiente sección se desarrolla la manera en que se resolvieron los resultados esperados que fueron expuestos en la sección anterior.

Resultado Alcanzado 1 (RA1) - Se estudiaron con profundidad los procesos existentes y los temas asociados a los mismos.

Resultado Alcanzado 2 (RA2) - Se analizó, modeló e implementó la extensión del Proceso de Configuración para incluir los aspectos de calidad en el proceso de construcción del WW. El proceso resultado permite que el usuario pueda definir distintas métricas de calidad ya sea tanto sobre los datos recién extraídos de la web (fuentes de datos) como del DW obtenido. Además, se realizó la traducción de todas las interfaces del prototipo de Configuración del WW y también de la base de Configuración del mismo. Se analizaron y priorizaron posibles mejoras y correcciones al prototipo, de las cuales se implementaron las más importantes.

Resultado Alcanzado 3 (RA3) - Dado que el proceso de Carga existente es una versión inicial de tipo Draft, se debió analizar, modelar e implementar el proceso de Carga nuevamente. Por lo tanto, el proceso resultado cuenta, al igual que el proceso de Carga ya existente, con actividades automáticas, que logran realizar la extracción de los datos, la integración de los mismos y la carga del WW; pero también incluye nuevas tareas automáticas que realizan las mediciones sobre los datos recién extraídos y propagan dichos valores de calidad a lo largo del proceso, así como miden la calidad sobre el DW resultado. La resolución de conflictos de extracción e integración que no son posibles automatizar se realizó a través de actividades manuales en donde el usuario toma decisiones que mejoran la calidad del WW construido.

Resultado Alcanzado 4 (RA4) - Se implementaron los procesos desarrollados en RA2 y RA3 considerando mejorar la integración de entidades en la carga del WW. Es decir, se implementó el prototipo tal que el usuario es capaz de definir servicios de integración que se nutren de las mediciones de calidad y resuelven conflictos típicos de integración como son las entidades duplicadas. En caso que el usuario no haya definido calidad o cuando las métricas instanciadas no permiten resolver todos los conflictos de integración, igualmente se cuenta con tareas manuales en las cuales el usuario puede seleccionar los datos que desee que continúen en el proceso de construcción del WW y los datos que no.

Resultado Alcanzado 5 (RA5) - Se desarrolló un caso de estudio con datos abiertos gubernamentales en el dominio Transporte en el Reino Unido que demostró la factibilidad de la propuesta de automatización en la construcción de un WW con BPMS incluyendo aspectos de calidad mediante los procesos definidos en RA2 y RA3.

1.5 Gestión del Proyecto.

En este punto se presenta la gestión del proyecto, la cual incluye las decisiones tomadas, los riesgos a mitigar así como la organización de la documentación e implementación elegida.

Al comienzo del proyecto, se realizó el estudio del artículo [2] que plantea la idea inicial del proyecto y se profundizó en los conceptos más importantes que conciernen al proyecto que son las áreas de Data Warehouse, de Gestión de Procesos de Negocio y de Calidad de los Datos.

En un principio también se identificaron cuáles eran los riesgos del proyecto y se buscaron soluciones para mitigar los mismos.

Uno de los riesgos más importantes del proyecto fue no contar con la documentación y el código del prototipo del proyecto anterior desde un comienzo. Se mitigó el mismo estudiando con profundidad el artículo [2] y las áreas de estudio concernientes al mismo, para disminuir la curva de aprendizaje una vez que se contara con la documentación del proyecto anterior.

Otro de los riesgos del proyecto fue lograr validar a tiempo con las tutoras cada una de las decisiones tomadas. Para mitigar este riesgo, se optó por el envío de documentos vía mail de modo de tener una validación temprana y evitar re-correcciones innecesarias.

Durante todo el proyecto, se plantearon objetivos tanto a largo plazo como a corto plazo y se pautaron reuniones semanales entre los integrantes del proyecto. También se documentó durante todo el proyecto, lo que facilitó enormemente la elaboración del presente informe.

Respecto a la organización del material, se optó por la herramienta Google Drive para el manejo de documentos y por el servicio SVN para el manejo del código implementado.

Dado que se obtuvo la documentación del proyecto anterior y el prototipo cuatro meses después de comenzado el proyecto, se extendió el estudio y relevamiento de los procesos existentes y los conceptos asociados dos meses más de lo definido por las tutoras en el cronograma del proyecto ya que la documentación y el código eran imprescindibles para continuar avanzando en el proyecto.

Una vez se contó con toda la documentación necesaria y el prototipo, se comenzó a analizar en profundidad el prototipo, planteando posibles mejoras y correcciones. A medida que se realizó el pasaje de la base de metadatos de Configuración del español al inglés (RA2), se dio oportunidad a desarrollar estas mejoras y correcciones, habiendo previamente priorizado las mismas en base a cambios considerados imprescindibles o mejoras en la amigabilidad del prototipo especialmente en la interfaz. Durante esta etapa también se reestructuró el código, logrando una modularización del mismo e incluyendo comentarios que lo convirtieron en un código escalable.

Una vez culminada esta etapa, se comenzó el diseño e implementación de los aspectos de calidad del Proceso de Configuración. En un principio se dedicó menos tiempo al diseño y más al investigar la herramienta de Activiti [6] ya que no se conocían del todo las limitaciones de la misma, para luego dedicarle gran parte del tiempo al diseño de modo de evitar re-correcciones que se habían dado principalmente por el poco tiempo dedicado al diseño.

El tiempo consumido en la implementación de las mejoras y las extensiones al proceso de Configuración llevó tres meses en lugar de los dos meses planteados por las tutoras para esta etapa en el cronograma del proyecto. Esto se debió principalmente al tiempo consumido en el pasaje de la base de metadatos de Configuración del español al inglés que implicó grandes modificaciones en el código del prototipo, el cual como se mencionó antes, es poco claro y escalable.

Respecto a la implementación del proceso de Carga, si bien se contaba con una versión inicial de tipo Draft, se decidió analizar, diseñar e implementar el prototipo desde cero debido principalmente a las dificultades que se presentaron al extender el proceso de Configuración, con código dificultoso y poco escalable. La implementación del proceso de Carga llevó dos meses lo cual coincide con el tiempo planteado por las tutoras en el cronograma.

Por otro parte, para realizar la prueba de concepto del prototipo se debió encontrar un caso de estudio con datos abiertos en inglés que cumpliera con los requerimientos necesarios para poder ejecutar los procesos implementados como por ejemplo contar con medidas que no sean sólo de contar registros.

También se debieron implementar los servicios de extracción de datos, los servicios de medición de calidad y los servicios de resolución de entidades para el caso de estudio elegido; a pesar de que los mismos no formen parte de la solución central del proyecto.

En relación al tiempo empleado en la ejecución del caso de estudio llevó dos meses en lugar de un mes como fue planteado en el cronograma por las tutoras ya que para lograr la implementación de los servicios web se debieron dedicar horas a aprender sobre aplicaciones que publiquen servicios Rest[7] mediante Spring [8]y se debieron implementar todos los servicios web necesarios. Cabe destacar también, que encontrar un caso de estudio con datos reales en inglés y que además cumpla con todos los requerimientos necesarios fue una tarea difícil de solventar debido principalmente a que los datos abiertos se publican a nivel estadístico y no con el detalle que se requiere para construir y explotar un DW.

1.6 Organización del Informe.

El resto del documento se organiza de la siguiente manera: en el capítulo 2 se desarrollan los conceptos principales en los que se basa el trabajo actual. En el capítulo 3 se presenta brevemente lo resuelto por el proyecto anterior [3] para la construcción de un WW utilizando BPMS y se introducen los problemas a resolver en este proyecto. En el capítulo 4 se introduce el diseño de la solución del mismo, la arquitectura y los diseños de los modelos de

datos realizados. En el capítulo 5 se describe la implementación del prototipo realizado y se detallan las especificaciones técnicas. En el capítulo 6 se presenta el caso de estudio a aplicar sobre el prototipo y el capítulo 7 detalla el proceso de verificación realizado. Para finalizar, en el capítulo 8 se presentan las conclusiones y los trabajos a futuro propuestos.

Junto al informe se incluyen los Anexos. En el Anexo A se profundiza sobre sistemas de Data Warehousing, Procesos de Negocio y Calidad de Datos. En el Anexo B se presenta el manual de usuario, en el Anexo C, el manual técnico con algunas consideraciones de la implementación, en el Anexo D, se detalla la implementación de los servicios web que no forma parte de la solución del proyecto, y finalmente, en el Anexo E, se presentan los casos de prueba realizados durante el proceso de verificación del prototipo.

2 - Marco Teórico

En este capítulo se presentan los conceptos principales sobre Data Warehouse, Web Warehouse, Procesos de Negocio y Calidad de Datos así como la asociación de estos conceptos con el proceso de construcción de un WW planteado en trabajos previos [2][3].

2.1 Data Warehouse y Web Warehouse

Un concepto primordial a definir es el de Data Warehouse (DW), el cual según Immon[9] "es un conjunto de datos orientados a temas, integrados, no volátiles e históricos, organizados para soportar un proceso de toma de decisiones". Mientras que los Sistemas de Data Warehousing (SDW) son sistemas informáticos capaces de ofrecer información para toma de decisiones y cuya pieza fundamental es un Data Warehouse [9].

En la Figura 1 se aprecian las fases que se distinguen en el desarrollo de un Sistema de DW.

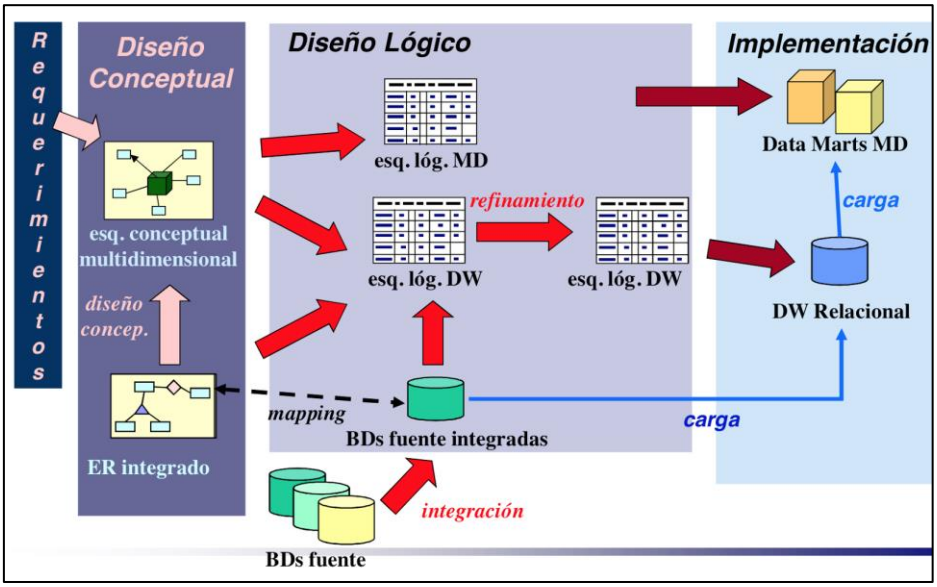


Figura 1 - Proceso de Desarrollo de un Sistema de DW [9]

A continuación, se describen brevemente cada una de estas fases.

Especificación de Requerimientos
En la primera fase se analizan los datos y se buscan relaciones o dependencias entre los mismos. Los expertos en IT deben mantener reuniones con los usuarios finales para relevar las necesidades del negocio.

Diseño Conceptual
En esta fase el objetivo es decidir qué datos son los interesantes para la organización y cómo se relacionan entre sí. El resultado de esta fase es un modelo que se denomina Modelo Multidimensional.

Un **Modelo Multidimensional** (MMD) es modelo de datos que representa los datos en forma cercana a la intuición del usuario y resuelve los problemas planteados en sistemas relacionales.

Los MMD representan los datos como una matriz, en donde los ejes se corresponden a los criterios de análisis (**dimensiones**) y en los cruces están los valores a analizar (**medidas**).

La Figura 2 ejemplifica estos conceptos. Se tiene vendedores de autos, en donde el Color y el Modelo de dichos autos son las dimensiones y sus cruzamientos en la matriz se corresponden a la medida cantidad de autos vendidos. Del ejemplo, se extrae que Ventas ("Sedan", Red)= 3. Es decir, que a partir de un valor para cada dimensión (en este caso 'Sedan' para Modelo y 'Red' para Color) se puede obtener un valor para la medida. [9]

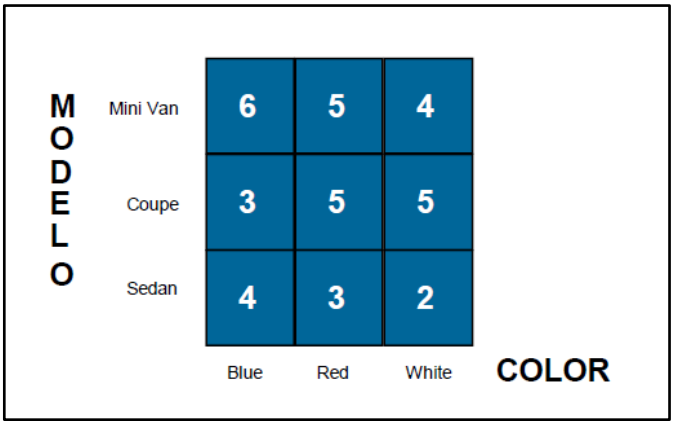


Figura 2 - Ejemplo de Medida [9]

En las dimensiones, los valores se organizan en **jerarquías** (categorías). Las jerarquías se dividen por niveles y permiten agrupar las entidades según alguna característica. En la Figura 3, se ilustra un ejemplo de este concepto. En este caso se tiene la dimensión vendedores y en la misma es posible agrupar los vendedores por Ciudad o "subir" un nivel en la jerarquía.

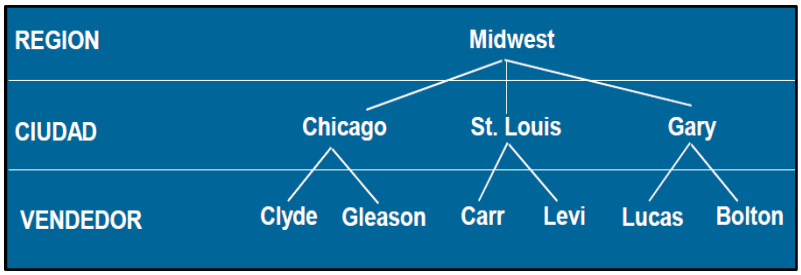


Figura 3 - Ejemplo de jerarquía de dimensiones [9]

Diseño Lógico

El objetivo de esta fase es construir un esquema lógico relacional o multidimensional del DW. Se toman en cuenta las características de las fuentes de datos, el esquema conceptual o una especificación equivalente, así como los requerimientos no funcionales del DW.

En este proyecto se trabaja con el diseño de un DW Relacional, en donde el usuario accede directamente al DW con herramientas de consulta (OLAP). Los DW Relacionales están conformados por tablas de hechos y por tablas de dimensiones.

Las **tablas de hechos** son aquellas tablas que guardan los indicadores del negocio que son de interés, es decir, las medidas del diseño conceptual. Mientras que en las **tablas de dimensiones** se guardan las descripciones de las dimensiones, pudiendo las jerarquías estar normalizadas o desnormalizadas. [9]

Las tablas de hechos y de dimensiones pueden ser organizadas de distintas maneras. Los distintos tipos de organización son Esquema Estrella, Esquema *SnowFlake* y *StarCluster*. A continuación, se detalla el Esquema Estrella que fue utilizado en este proyecto.

En particular, el **Esquema Estrella** representa los conceptos multidimensionales sobre el modelo relacional con una tabla central de hechos y un conjunto de tablas usualmente con menos registros organizados alrededor, que corresponden a las dimensiones. El modelo pretende minimizar los joins debido a que son operaciones costosas; para lograrlo emplea redundancia y tablas desnormalizadas. [9]

En el ejemplo de la Figura 4, se muestra un ejemplo de un diagrama correspondiente a un DW de ventas de productos, en donde se tiene una tabla de hechos Ventas y cinco tablas de dimensiones (Fechas, Clientes, Vendedores, Geografía y Productos).

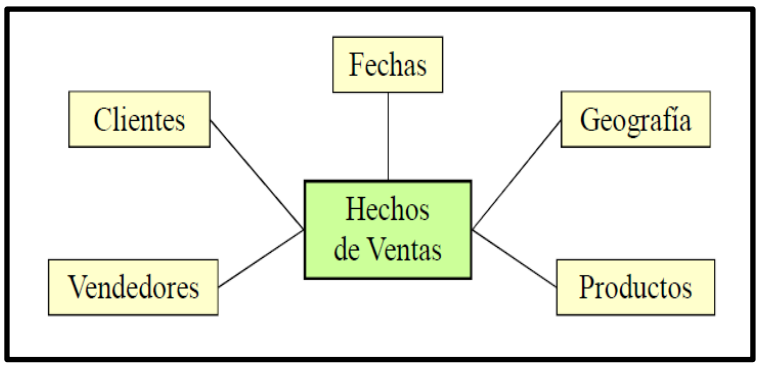


Figura 4 - Ejemplo de un Esquema Estrella[9]

Diseño Físico e Implementación

En esta fase se especifican cómo se almacenarán y accederán los datos para asegurar una *performance* adecuada del SDW y se realiza la implementación del mismo.

Luego de implementado el DataWarehouse, se utilizan herramientas para que un usuario final pueda acceder a la información de manera simple y potente. Entre esas herramientas se encuentran las herramientas OLAP.

Las Herramientas OLAP (On-Line Analytical Processing) permiten consultar modelos multidimensionales interactivamente y de forma eficiente. [9]

Un Web Warehouse (WW) se define como un Data Warehouse que consolida datos obtenidos de la web. [2] Es decir, un WW brinda la posibilidad al usuario de tomar decisiones a partir de datos extraídos de fuentes heterogéneas provenientes de la web.

El WW definido en trabajos previos [2] es un sistema orientado a servicios compuesto por varios componentes. En la Figura 5 se muestra su arquitectura, presentando cada uno de sus componentes, los cuales están encargados de una tarea particular para lograr obtener los datos de la web, integrarlos, transformarlos y presentarlos al usuario para su posterior análisis.

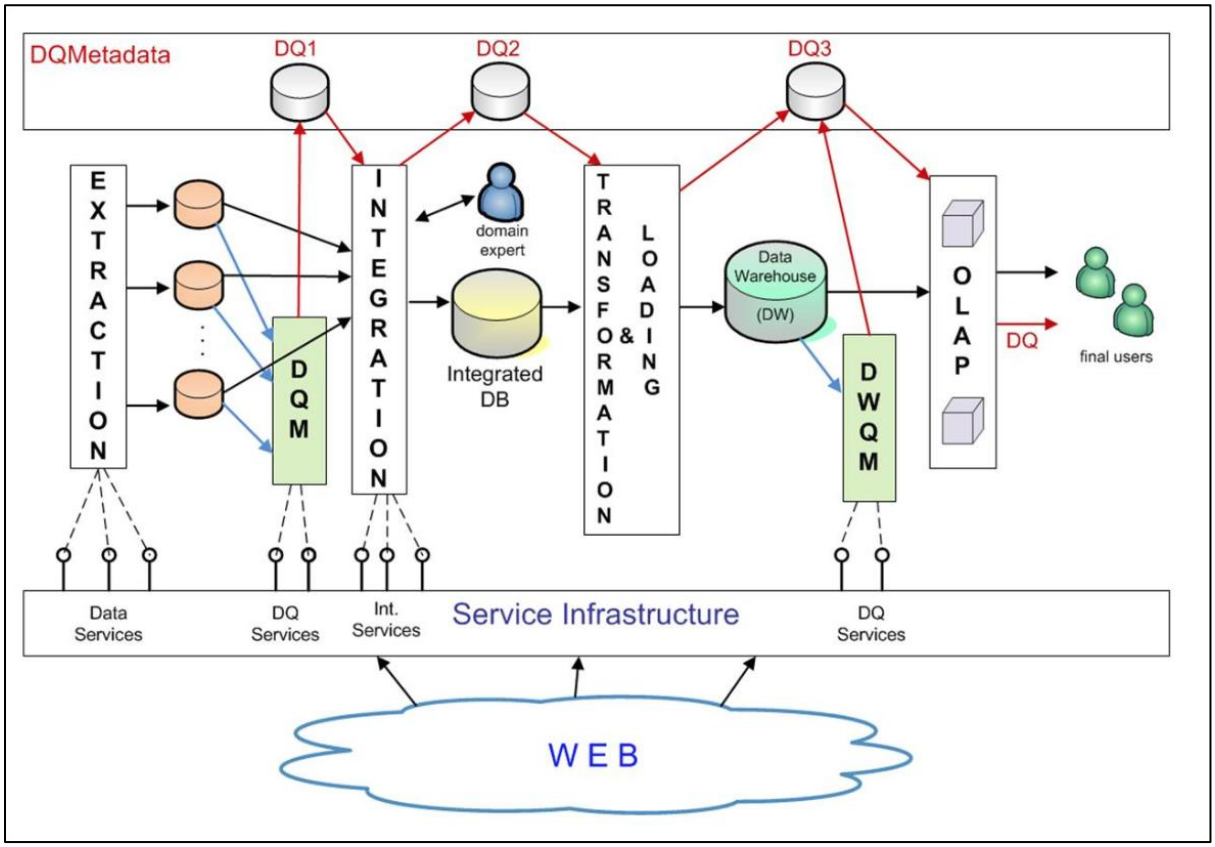


Figura 5 - Arquitectura General del WW enfocado en la Calidad [10]

Este WW es flexible ya que puede ser configurado para distintos dominios, en donde se eligen las diferentes Fuentes de Datos Web (FDW), así como también todas las características propias de un DW. Una FDW representa la ubicación web desde donde se obtiene la información requerida. Dicha fuente podrá ser accedida mediante diversos mecanismos, por ejemplo utilizando servicios web, y la manera en que dichos datos serán entregados también puede variar en formato y forma. Es por esto que la utilización de servicios facilita que el sistema pueda ser configurado para cualquier dominio sin necesidad de una implementación específica para cada caso en particular.

Como se mencionó en el capítulo 1, el proceso de construcción del WW consiste en dos fases: la Fase de Configuración y la Fase de Carga. En la Fase de Configuración se realiza un proceso de configuración del WW que incluye la selección de distintas FDW, la definición de esquemas y de mapeos entre los mismos. Luego en la Fase de Carga, a partir de la configuración definida, se realiza la carga del WW.

Los componentes del WW definido son: *Extraction*, *Integration*, *Transformation and Loading*, *OLAP (On-Line Analytical Processing)*, *Data Quality Measurement (DQM)* and *DW Quality Measurement (DWQM)*.

La función del componente ***Extraction*** es la de extraer información de las distintas FDW y guardarlos provisoriamente en una base intermedia para su posterior análisis. Mediante este paso se define un esquema común esperado para las distintas fuentes, a partir de este punto se hará referencia a este esquema como ***Expected Schema***. Por lo tanto, este componente resuelve el problema de heterogeneidad de fuentes ya que los distintos datos pueden provenir de fuentes tales como csv, html, rdf, etc.

El componente ***Integration*** se encarga de la integración de los datos que provienen de las distintas FDW. En esta etapa es crucial la resolución de posibles conflictos entre los datos provenientes de diferentes fuentes. A su vez lo que se espera es determinar un esquema común integrado y mapearlo con el esquema esperado (*Expected Schema*) definido en la etapa anterior. A este esquema se lo nombró ***Integrated Schema***.

Luego en el componente ***Transformation and Loading*** se define el esquema del DW (***DW Schema***) a partir de las tablas definidas en el esquema integrado y realizando el mapeo con las mismas de manera de obtener el modelo multidimensional. Además en esta etapa se define un proceso para transformar los datos y cargarlos en dicho modelo.

El componente ***OLAP*** es el encargado de manipular los datos del modelo multidimensional mediante la definición de distintos cubos que permiten al usuario final visualizar la información, explotando de este modo el WW.

Cabe resaltar que, a lo largo del proceso de construcción del WW, se hace énfasis en la calidad de los datos, la cual se mide y se almacena manteniendo una referencia entre los datos y su calidad para que el usuario final sea consciente de la calidad de la información que se le muestra. El componente ***DQM*** es el encargado de realizar la medición de la calidad de los datos extraídos de la web, mientras que el componente ***DWQM*** es el encargado de medir la calidad de los datos que han sido cargados al DW. Toda la información de calidad procesada es almacenada en las bases de datos: DQ1, DQ2 y DQ3 (ver Figura 5), de esta forma se permite propagar la calidad de los datos y tomar decisiones en la resolución de conflictos durante el proceso de construcción.

Finalmente en la Fase de Carga, a partir de la información obtenida en la Fase de Configuración, automáticamente se realiza la construcción del WW para el dominio especificado. Es decir, se ejecuta la extracción, integración, transformación y carga, manipulando, en todo momento del proceso, la calidad de los datos.

Es importante señalar que la Fase de Configuración consiste principalmente de la definición de los distintos esquemas que fueron mencionados anteriormente (*Expected Schema*, *Integrated Schema* y *DW Schema*) y el mapeo entre sus elementos, los cuales posibilitan la carga posterior del WW. La Figura 6 ilustra este hecho.

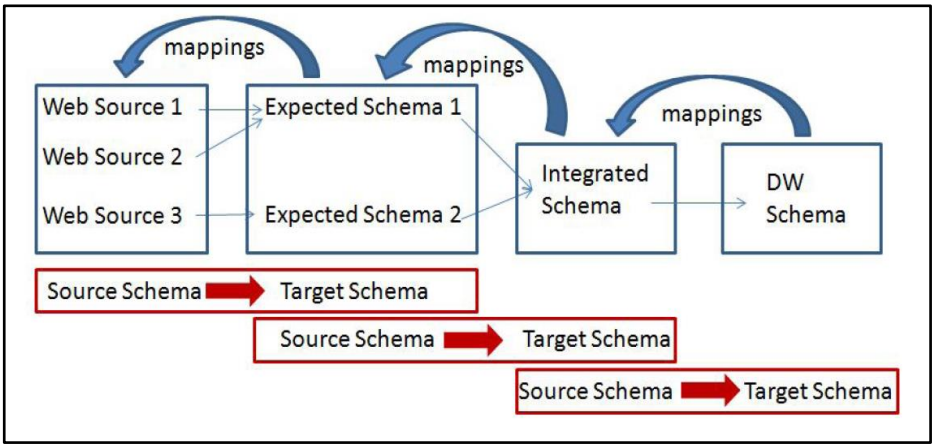


Figura 6 - Definición de Schemas y sus correspondencias[10]

2.2 Procesos de Negocio

Un **Proceso de Negocio** (Business Process) es un conjunto de actividades que son realizadas en coordinación en un entorno organizacional y técnico, para alcanzar un objetivo del negocio. [1]

Los procesos de negocio se caracterizan por ser grandes y complejos, involucrando varias secciones o áreas de una organización y organizaciones distintas. Además tienen duraciones prolongadas a lo largo del tiempo, pudiendo durar semanas, meses o años, y pueden ser tanto manuales como automatizados o ambos. [11]

Un **sistema BPM o BPMS** es un sistema de software genérico guiado por representaciones explícitas de procesos para coordinar la ejecución de procesos de negocio.[1] Es decir, un BPMS provee soporte al ciclo de vida de los PNs y su módulo principal es un motor de procesos que permite ejecutar el flujo de control de los PN.

En la Figura 7 se presenta la arquitectura típica de un BPMS la cual cuenta con diversos módulos que se describen a continuación.

- El módulo de modelado de procesos (*Process modeling tool*) es aquel que permite la creación y/o modificación de PNs. También puede tener la capacidad de la importación de los mismos.
- El módulo motor de ejecución (*Execution engine*), como se describe anteriormente, es el módulo principal de un BPMS. Es el encargado de la instanciación de procesos y toda su organización desde el comienzo al fin de su ejecución.
- El módulo de manejador de tareas (*Worklist handler*) permite a los distintos participantes del proceso confirmar una actividad del flujo del proceso. Por lo tanto es el encargado de manejar las listas de espera y asignación de actividades.

- El módulo de administración y monitoreo (*Administration & monitoring tools*) permite gestionar el BPMS, monitorear el rendimiento durante la ejecución de los procesos, entre otras características.
- El módulo de servicios externos (*External services*) contempla aquellos servicios que exponen una interfaz mediante la cual el motor de procesos puede comunicarse. Algunos ejemplos son: motores de reglas de negocio, conectores de bases de datos, notificaciones por email.

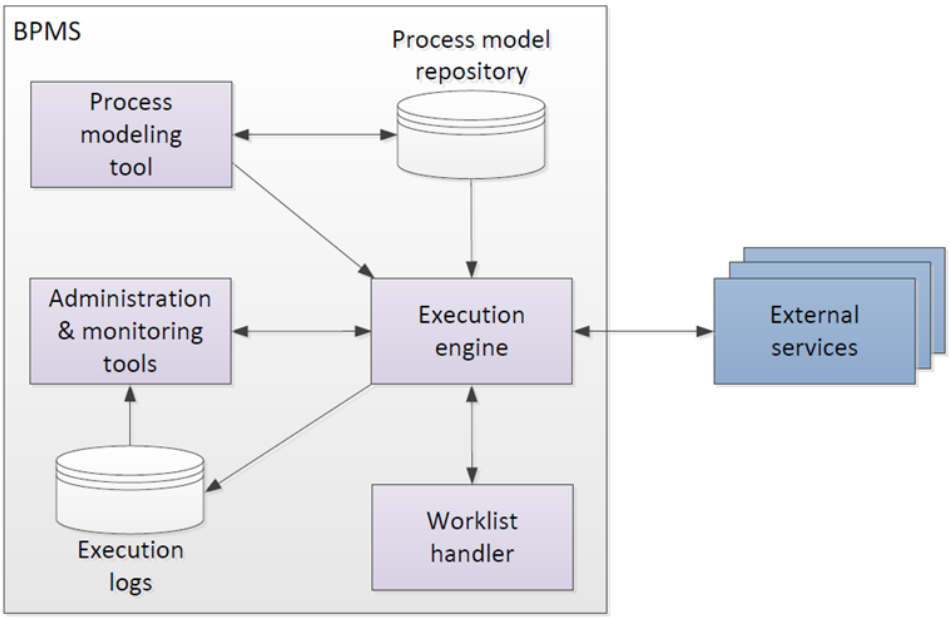


Figura 7 - Arquitectura típica de un BPMS [12]

Un **Modelo de Procesos de Negocio** es una representación abstracta de los procesos de una organización, que muestran principalmente cómo y por quién son llevadas a cabo las actividades que generan valor para la organización. [11]

Es decir, un Modelo de PN permite explicitar qué tareas y cómo (manual, automática) se realizan en la organización, especificando quiénes son los participantes del proceso (roles, secciones). Además, el modelado de PN permite reflexionar sobre la ejecución del proceso, evaluando posibles rediseños que mejoren la performance y la utilización de los recursos.

Al momento de modelar un PN, es importante acertar con la notación a utilizar ya que si todas las partes involucradas “comprenden” el proceso, se facilita la comunicación y elaboración de un proceso más eficiente. Existen muchas notaciones para modelados de PNs que provienen de distintas áreas (ej. BPMN vs. UML) e incluso existen traducciones de una a otra.

En este proyecto se utiliza el estándar de modelado OMG **Business Process Model and Notation (BPMN 2.0)**[13], el cual no solo permite representar de manera gráfica un PN sino, también, la ejecución del mismo según la semántica establecida para cada uno de sus elementos. Otra ventaja de este estándar, es que provee una notación amigable que puede

ser comprendida tanto por el área del negocio como por el área del software mitigando errores producidos por malas interpretaciones del proceso. La Figura 8 muestra los elementos principales de esta notación:

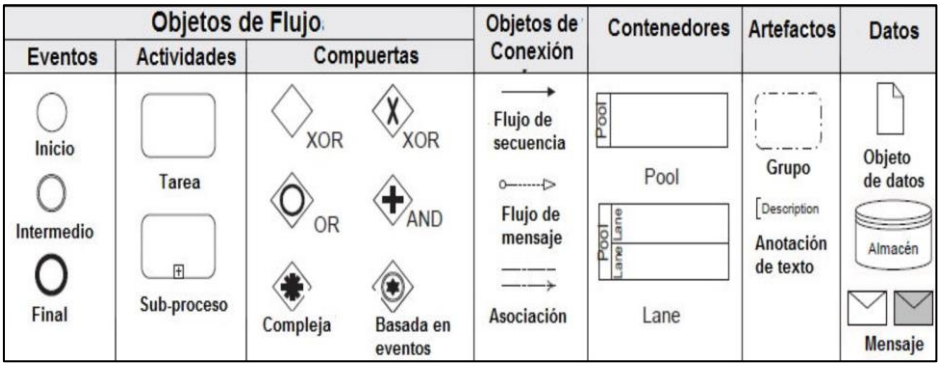


Figura 8 - Elementos principales de BPMN 2.0 [11]

En la Figura 9, se presenta un ejemplo de modelado de un PN con BPMN 2.0 para la compra y venta de un producto o servicio, en donde participan dos organizaciones cada una representada mediante un *pool* (*Buyer* y *Reseller*). El flujo del proceso es acordadoy compartido por ambas organizaciones con intercambio de mensajes.

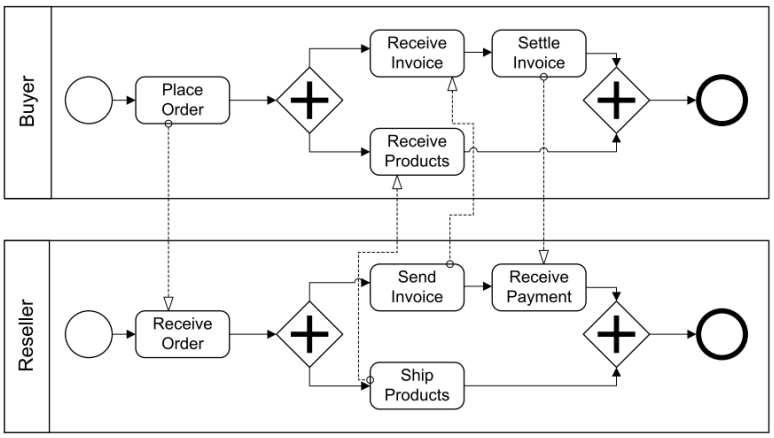


Figura 9 - Ejemplo de Proceso de Negocio[1]

Otro beneficio de utilizar PN está asociado a los roles involucrados en la construcción del WW y el uso del sistema como se aprecia en la Figura 10. Dichos roles son: Experto en Tecnología (***IT Professional***), Experto en el Dominio (***Domain Expert***) y el Usuario Final (***Final User***). El rol Experto en Tecnología participa en la Fase de Configuración del WW. En cambio el rol de usuario Experto en el Dominio seleccionado, participa en la selección de las fuentes en la Fase de Configuración y en las decisiones al momento de integrar las fuentes en la fase de Carga como es la resolución de conflictos. Por último, el Usuario Final es el encargado de explotar el WW. Con BPMN se permite de manera gráfica que cada uno de los involucrados en el proceso tenga conocimiento de cuándo y quién realiza determinada actividad. Es decir, el BPMS se encarga de asignar las tareas y estas son visibles sólo para los usuarios que poseen determinado rol.

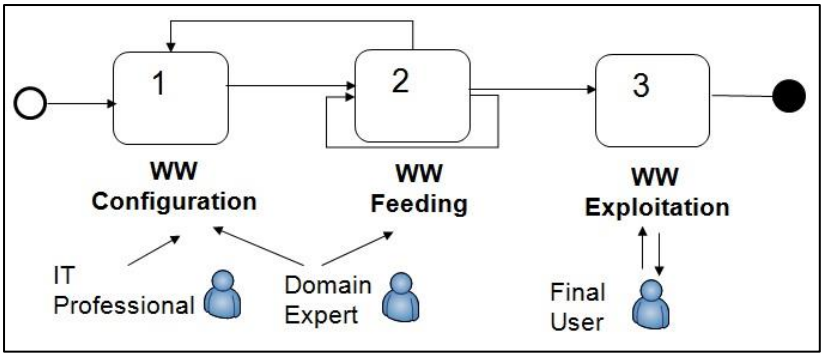


Figura 10 - Fases y Roles en la Construcción del WW [2]

Relación con la construcción del WW

Debido a la complejidad de las tareas requeridas para configurar y cargar el WW se utiliza un BPMS, y en este caso el objetivo del PN es la construcción de un WW flexible. Es decir, empleando esta tecnología se definen todos los datos necesarios para realizar el proceso de carga, registrándolos en una base de datos que es posteriormente consultada al momento de la generación del WW.

2.3 Calidad de Datos

En un WW es imprescindible establecer cuál es la calidad de la información que se ofrece y consume, por lo que se vuelve necesario considerar distintos aspectos de calidad durante el proceso de construcción del mismo. En este punto, se profundiza sobre conceptos de calidad de datos.

El concepto de calidad de datos es un término complejo que se viene estudiando hace ya varias décadas; en principio, se podría asociar con determinar qué tan correctos son los datos frente a un dominio de la realidad, si es válido en el momento entregado, o si se encuentra en el formato correcto, entre otras características que un consumidor final espera. Las consecuencias de una baja calidad de datos podrían conllevar a una gran cantidad de problemas, desde problemas de costos y/o tiempo debido a un mayor procesamiento y almacenamiento de datos duplicados o erróneos; o hasta incluso problemas de mayor índole.

Para el estudio de la Calidad de Datos se define una jerarquía de conceptos, como se puede apreciar en Figura 11 donde se definen las dimensiones múltiples. Luego, cada una de dichas dimensiones, se pueden subdividir en factores buscando con los mismos un mayor detalle del análisis. Por lo tanto, podemos ver una dimensión como un agrupamiento de factores de calidad que tienen el mismo propósito. Finalmente por cada factor se presenta una o más métricas, mediante las cuales se busca realizar una medición de la calidad de ese factor establecido planteando una semántica (cómo se mide), las unidades de medición y la granularidad de la medida.[15]

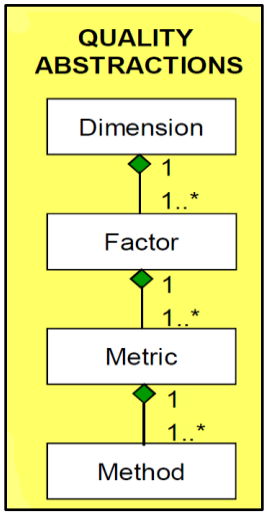


Figura 11- Jerarquía de una Dimensión [14]

Una **Dimensión** captura una faceta (a alto nivel) de la calidad. Mientras que un **Factor** representa un aspecto particular de una dimensión de calidad. [14]

También se cuenta con el concepto de **Métricas de Calidad**, las cuales son un instrumento que define la forma de medir un factor de calidad. Para instanciar una métrica se debe definir: un método para medirla (semántica), una unidad de medición (ej. tiempo de respuesta en ms, volumen en GB) y la granularidad de la medida.

Una métrica puede ser medida con distinto nivel de detalle, definiendo así el concepto de **Granularidad**. Las granularidades típicas son: celda, tupla, conjunto de celdas, atributo, tabla. Por ejemplo, si se mide una métrica a nivel de celda sólo se considera el valor del atributo de dicha celda al momento de determinar el valor de calidad.

A continuación se explican estos conceptos con un ejemplo:

La dimensión Exactitud refiere a la correctitud y la precisión con que los datos del mundo real son representados en un sistema de información. Un aspecto (factor) de esta dimensión es la Correctitud Sintáctica que indica qué tan libres de errores sintácticos se presentan los datos. Una métrica de calidad de este par dimensión-factor podría ser una que indicará si un dato es sintácticamente correcto o no. Por ejemplo si una cédula tiene el formato correcto o no. En este caso la granularidad de la medida sería a nivel de atributo (celda).

La calidad de datos obtenidos de la Web se pueden diferenciar entre calidad aplicada a los servicios que utilizan la web como fuente de datos, y la calidad propia de los datos. En este proyecto se aborda la calidad propia de los datos y algunas de sus dimensiones de calidad típicas son: completitud, exactitud, frescura, consistencia, unicidad y confiabilidad. Por más detalles sobre dichas dimensiones ver Anexo A.3.

Para considerar la calidad de un sistema de información es imprescindible definir un **Modelo de Calidad** que se adecue a las necesidades y prioridades de los usuarios finales que consumen los datos. Un modelo de calidad de datos para cierto dominio o sistema de

información define los aspectos de calidad relevantes y la forma de tratar a cada uno de ellos para lograr el control y mejora de la calidad de los datos.

Un modelo de calidad de datos debe incluir las siguientes especificaciones: dimensiones y factores de calidad relevantes, métricas de calidad junto con los datos o conjuntos de datos a los que se aplican, métodos de medición de la calidad, y acciones de mejora de la calidad tanto correctivas como preventivas. [14]

La **propagación de la calidad** consiste en determinar o estimar la calidad de un dato luego de aplicar una integración, transformación, etc. Existen dos tipos de propagación de calidad, la propagación directa y la propagación inversa. En la propagación directa se propagan los valores desde las fuentes hacia los datos resultantes pudiendo ser modificados, determinando la calidad ofrecida al usuario final. Mientras que en el caso de la propagación inversa, se fijan las restricciones de calidad para las fuentes para asegurar la calidad del dato resultante requerida por el usuario. [14]

3- Problema Planteado

En este capítulo se presenta brevemente lo resuelto por el proyecto anterior [3] para la construcción de un WW utilizando BPMS y se introducen los desafíos planteados en este proyecto.

3.1 Contexto

El proyecto anterior definió e implementó un prototipo de la fase de Configuración del proceso de Construcción de un WW, automatizando dicha fase con BPMS y definiendo metadatos que luego son utilizados por el proceso de Carga para realizar efectivamente la carga del WW. A su vez, implementó una prueba de conceptos del proceso de Carga de un WW que validó el proceso de Configuración implementado, utilizando un caso de estudio de datos abiertos del Uruguay en el dominio 'Turismo'.

A continuación se presentan los modelos de los procesos de Configuración y Carga modelados en BPMN 2.0 e implementados en Activiti BPMS [6] por el proyecto anterior junto a una descripción breve de los mismos.

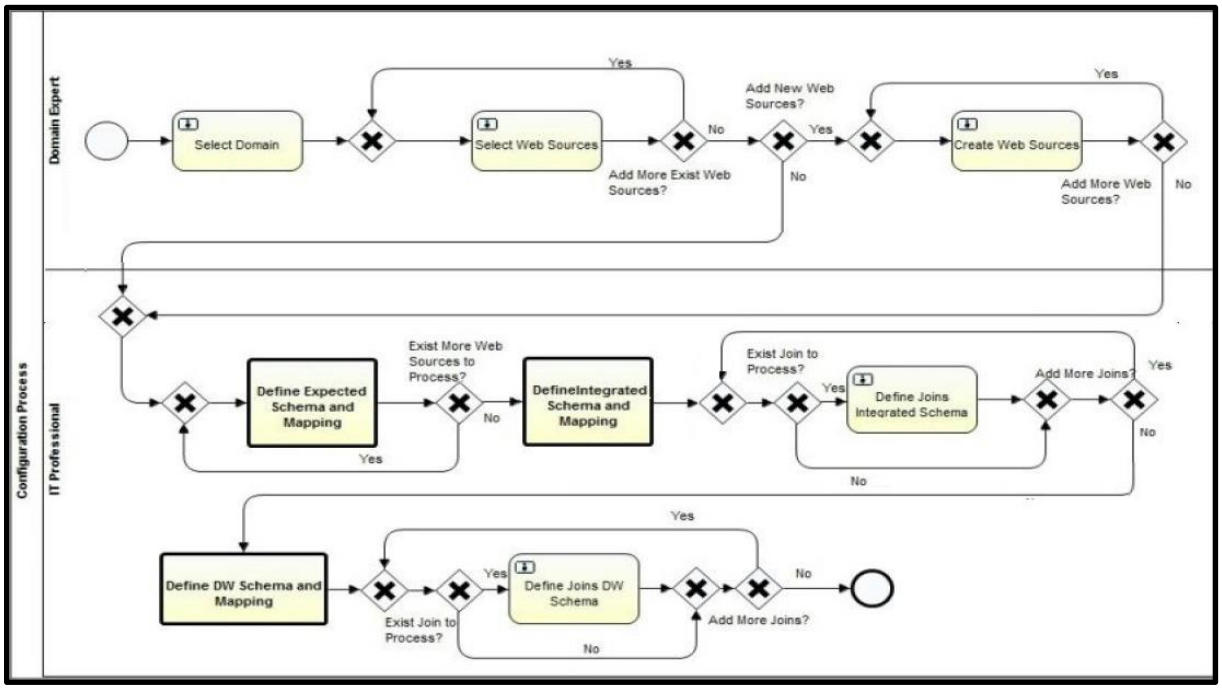


Figura 12 - Proceso de Configuración implementado por el proyecto anterior [3]

En la Figura 12 se ilustra el proceso de Configuración, el cual se divide en dos *lanes*; uno que corresponde al Experto en el Dominio (Domain Expert) y el otro al Experto en Tecnología (IT Professional). En la primeras tres tareas del proceso el usuario Experto en el Dominio, selecciona el dominio del WW a construir así como las FDW que nutrirán con datos al WW. En el resto del proceso se definen los *Expected Schemas*, el *Integrated Schema* y *DW Schema* así como el mapeo entre sus elementos para seleccionar e integrar los datos (ver Figura 6).

Todos los datos ingresados por el usuario durante el proceso de Configuración son guardados en una base denominada Base de Metadatos para luego ser utilizados en el proceso de carga para realizar la carga efectiva del DW. En la Figura 13 se aprecia dicha base, en donde se agruparon las tablas según su función; la cual que se describe a continuación:

a) En estas tablas se alojan todas las configuraciones de todos los procesos existentes en el sistema, los dominios, las FDW y sus correspondientes formatos (csv, xml, etc).
b) Tablas que almacenan la definición de los *Expected Schemas* y sus mapeos con las FDW
c) Estas tablas registran las definiciones de los *Integrated Schemas* y su asociación con los *Expected Schemas*.
d) Tablas que alojan las definiciones de los *DW Schemas* y sus mapeos con los *Integrated Schemas*

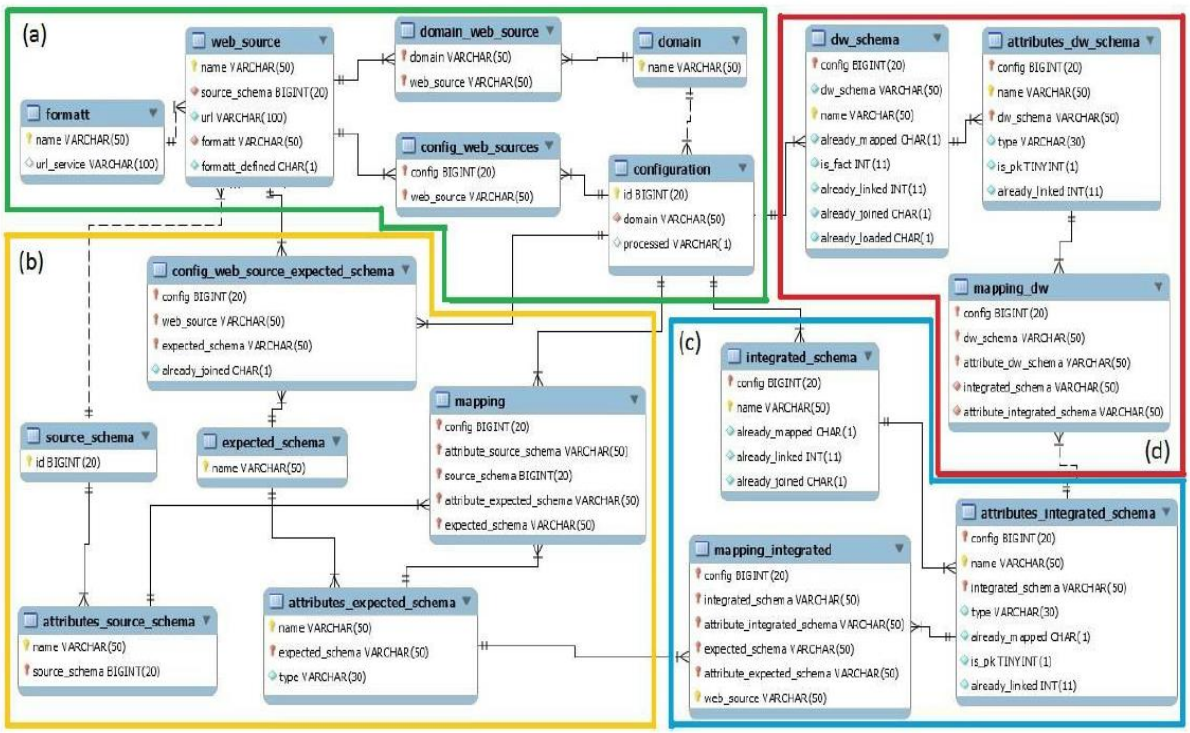


Figura 13 - Modelo de Datos de la Base de Configuración del proyecto anterior [3]

El proceso de Carga existente se puede apreciar en la Figura 14, el cual constituye una prueba de conceptos de la Fase de Carga de la construcción de un WW.

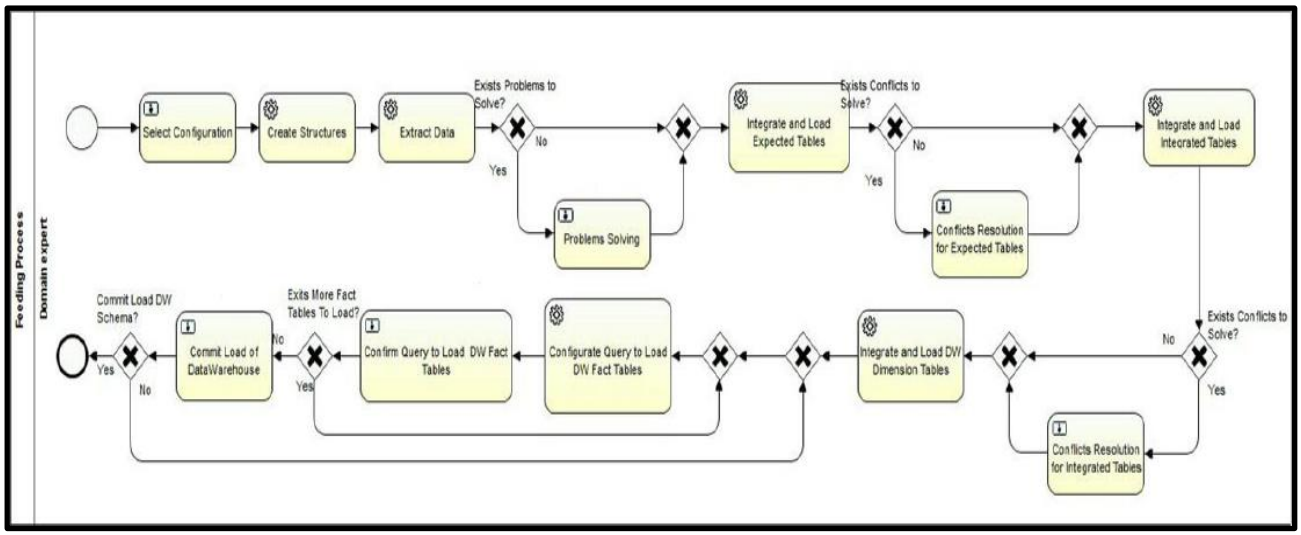


Figura 14 - Proceso de Carga implementado por el proyecto anterior [3]

En los restantes puntos de este capítulo, se pasará a desarrollar cada uno de los requerimientos que competen a este proyecto.

3.2 Aspectos de Calidad en la Construcción del WW

En este punto, se profundiza sobre los aspectos de calidad en la Construcción del WW.

La Calidad de Datos, como se mencionó antes, es imprescindible de considerar al momento de la construcción de un WW para garantizar la calidad de la información consumida por el usuario final. A continuación se detallan las actividades que debe realizar el proceso tanto en la fase de Configuración como en la de Carga para considerar la calidad en el proceso.

Las principales actividades que llevan a cabo los componentes DQM y DWQM durante la fase de Configuración son las siguientes:

1) Definir el Modelo de Calidad de los datos recién extraídos de la web
2) Definir el Modelo de Calidad del DW resultado.
3) Definir metadatos de calidad.
4) Seleccionar los servicios de calidad. [2]

Para los datos recién extraídos de la web al momento de definir el modelo de calidad se toma en cuenta principalmente aspectos de calidad generales y el dominio de las FDW que se está considerando. Mientras, para definir el modelo de calidad del DW obtenido, se debe considerar aspectos propios del modelo multidimensional definido (medidas, dimensiones y jerarquías), el contexto de análisis y reglas propias de los datos del DW y los conocimientos del usuario experto en el dominio.

Los Metadatos de Calidad, es decir los valores obtenidos en las mediciones de calidad, deben ser almacenados en una base de datos relacional a la que se pueda tener acceso en

todo momento del proceso y de esta manera permitir tomar decisiones en momentos oportunos.

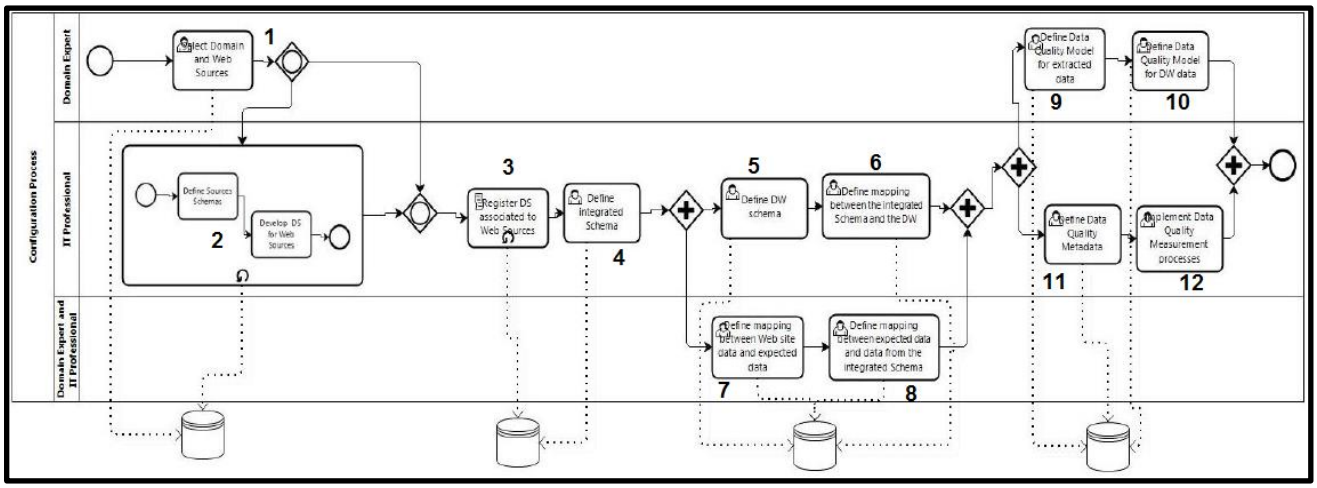


Figura 15- Proceso de Configuración inicial [2]

La Figura 15 muestra el Proceso de Configuración definido inicialmente en trabajos previos [2], el cual se utilizó en este proyecto para orientar la inclusión de la calidad al proceso. Las tareas 9, 10,11 y 12 de dicho proceso corresponden a los puntos 1, 2,3 y 4 de las actividades que realizan los componentes DQM y DWQM durante la fase de Configuración.

En el caso del proceso de Carga las actividades a realizar por ambos componentes son:

1) Realizar mediciones de calidad sobre los datos extraídos de la web.
2) Realizar mediciones de calidad sobre el DW obtenido.
3) Propagar las mediciones de calidad sobre los datos de la web durante todo el proceso de Carga.

Los puntos 1 y 2 se espera se realicen de manera automática y que las mediciones de calidad se registren en una base de metadatos de calidad a la que el usuario pueda consultar una vez finalizado el proceso.

Además, luego de realizadas las mediciones de calidad sobre los datos extraídos de la web, se desea que se propaguen las mediciones de calidad a lo largo de todo el proceso de Carga, lo cual permite mantener una correspondencia entre los metadatos de calidad y los datos del WW en los casos que los datos son combinados y transformados [2].

3.3 Mejora en la Integración del WW

Esta sección describe el problema relacionado con la integración de los datos en la Construcción de un WW, y cómo debería ser el comportamiento del sistema para resolverlo.

Existen dos distintos tipos de Integración durante la Fase de Carga: una es al momento de unificar los datos que refieren a la misma entidad y la otra es al momento de vincular

distintas entidades para conformar una nueva entidad que contenga más información. La Figura 16 ejemplifica estos dos tipos de integración.

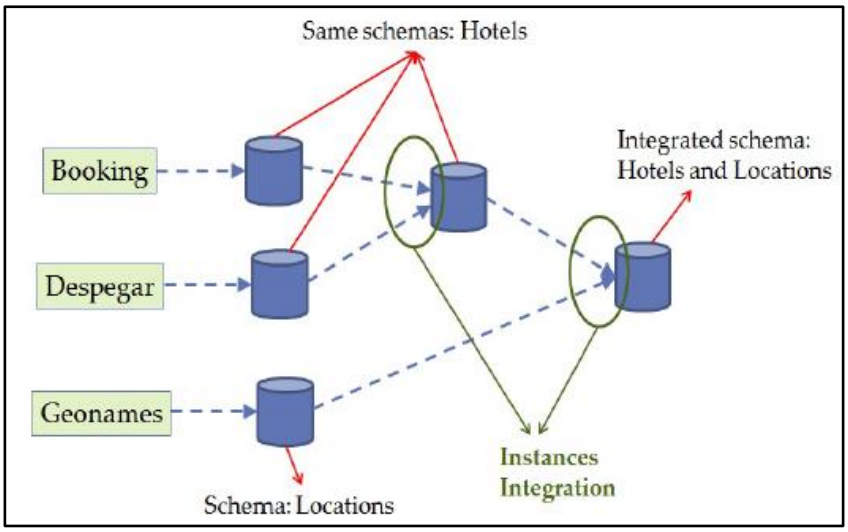


Figura 16 - Integración entre FDW e Integración entre *Expected Schemas* [2]

Como se puede apreciar en la Figura 16, se tiene un *Expected Schema* 'Hoteles', al que se le han asociado dos FDW: una de ellas es Booking y la otra Despegar. En la Fase de Carga, se debe extraer los datos de ambas FDW y registrarlos en el Expected 'Hoteles' considerando la posibilidad de que una misma entidad se encuentre tanto en la FDW Booking así como en la FDW Despegar.

Por lo tanto, el proceso de Carga implementado debe tener la capacidad de detectar las entidades duplicadas y luego seleccionar las entidades más confiables tomando en cuenta los metadatos de calidad definidos. Además, en los casos que no se defina calidad o que no se pueda resolver la integración (los valores de calidad de las entidades duplicadas son iguales), se espera que el usuario resuelva dichos conflictos a partir de tareas manuales y que se registren en un *log* las entidades duplicadas. Este tipo de integración se deberá resolver siempre que se tenga un *Expected Schema* al que se la ha asociado más de una FDW.

Por otro lado, en la Figura 16, se aprecia también que se vinculan los *Expected Schemas* 'Hoteles' y 'Locations' para conformar el *Integrated Schema* 'Hotels and Locations' añadiendo una referencia geográfica a la entidad hoteles. En este caso un determinado hotel podría no tener una referencia geográfica en el Expected 'Locations', por lo que se estaría perdiendo la posibilidad de agregar un valor de información a la entidad. Se espera que el Proceso de Carga sea capaz de detectar estos casos, generar un log con dicha información y mostrárselos al usuario.

En el proyecto anterior no se considera la primer integración (entre distintas FDW) debido a que no se define una manera de identificar las entidades en un *Expected Schema*, por lo tanto, todos los datos son cargados incluso los que referencian a una misma entidad de la realidad (quedando duplicados). Respecto a la segunda integración (entre *Expected*

Schemas), la solución no contempla los posibles errores al momento de cargar las Integrated Tables. Cabe mencionar, que dichas integraciones no fueron realizadas en el proyecto anterior ya que no estaban incluidas en el alcance del mismo.

3.4 Caso de Estudio de aplicación del Prototipo

En este punto se describen los distintos requisitos con los que debe contar el caso de estudio de aplicación del prototipo a implementar.

En primer lugar el caso de estudio debe ser con datos en inglés y pertenecer a un dominio distinto al del proyecto anterior con el fin de poder afirmar que el proceso implementado cubre distintos dominios, países y lenguajes. Además, al momento de elegir el caso de estudio se deben considerar los siguientes puntos de modo de alcanzar validar el prototipo desarrollado:

- El DW resultado debe tener una utilidad, es decir un usuario final debe poder explotar el mismo.
- Los datos de las fuentes deben ser presentados de manera simple (datos crudos).
- Se debe poder integrar al menos dos FDW (al menos dos *Expected Schemas* mapeados al mismo *Integrated Schema*)
- Debe existir un *Expected Schema* con dos FDW distintas (aplicar integración de entidades)
- El DW posea alguna medida interesante además de los conteos por cada dimensión.

4 - Diseño de la Solución

En el presente capítulo se presenta el diseño de la solución al problema planteado. Se describe la arquitectura de la solución y las extensiones que consideran los aspectos de calidad, la mejora de la integración de entidades y las mejoras al proceso de Configuración y el proceso de Carga implementado. Además, se presentan los diseños de los modelos de datos realizados.

4.1 Arquitectura de la solución

En este punto se presenta la Vista Lógica del diseño de arquitectura realizado, explicando cada uno de sus subsistemas y la Vista de Distribución (Deployment) incluyendo la explicación de cada uno de sus nodos.

4.1.1 Contexto

La solución propuesta hace uso de diversos componentes del *BPMS Activiti* [6], incluyendo su motor y la aplicación web ofrecida (*Activiti Explorer*). Por su parte el motor de Activiti se encuentra implementado en Java y es el encargado de la ejecución de todos los procesos diseñados utilizando BPMN 2.0. Por otro lado, la aplicación web es la encargada de mostrar los distintos elementos haciendo uso de un componente externo llamado Vaadin. Por lo dicho anteriormente la arquitectura del sistema implementado se basa en parte en la propia arquitectura de Activiti, esto incluye la comunicación entre lo que un usuario final puede acceder mediante un navegador web y el motor de Activiti encargado de procesar correctamente los pedidos.

En la figura 17 se ilustra un diagrama de la arquitectura del sistema. Con la misma se trata de ejemplificar las distintas comunicaciones existentes entre los componentes utilizados. De un navegador web se accede a la capa de presentación. Luego cada solicitud es enviada al servidor el cual se encarga de procesarla mediante el motor de Activiti, utilizando en caso de ser necesario las clases Java agregadas. El procesamiento puede consistir tanto en comunicarse con un motor de base de datos o con cualquier otro componente con el que sea necesario interactuar. Es oportuno mencionar que Activiti utiliza una base de datos propia para persistir datos relativos a la ejecución de los procesos.

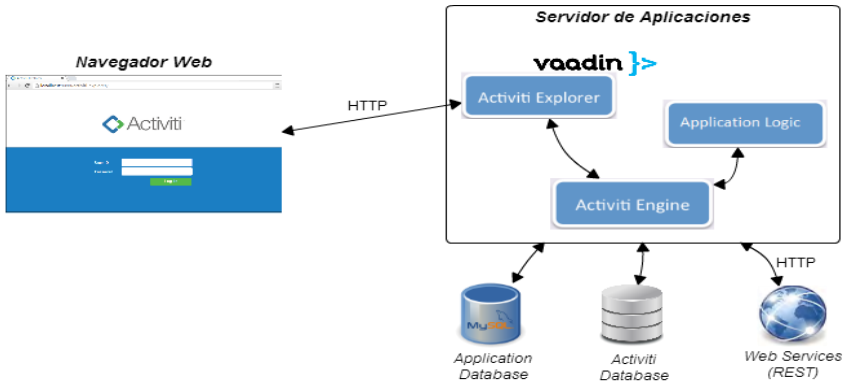


Figura 17 - Arquitectura del sistema

Al momento de diseñar los procesos se permite asignar a diversos eventos de las actividades clases Java, las cuales serán invocadas durante la ejecución de los procesos. Esto permite fácilmente agregar al proyecto Activiti clases, es decir un comportamiento personalizado al flujo del todo el proceso. En particular, para este proyecto, permite que se tenga acceso a una base de datos externa para persistir información relevante, así como también poder consumir servicios ajenos a la aplicación.

Otro aspecto destacable es la interacción con el componente visual (Vaadin), ya que es posible diseñar nuevos elementos gráficos mediante clases Java para luego ser utilizados por los procesos de negocio, pudiéndose incluir dichos elementos gráficos al sistema. Por más detalles de cómo incluir dichos elementos gráficos al sistema consultar Anexo B Manual de Usuario.

4.1.2 Vista Lógica

La vista lógica permite describir el sistema en base a abstracciones fundamentales del diseño orientado a objetos para dar soporte a los requerimientos funcionales, es decir todas aquellas funcionalidades provistas al usuario. Se trata de mostrar gráficamente cómo el sistema global se encontrará sub-dividido en subsistemas y/o paquetes para poder obtener un mayor refinamiento y observar de qué forma interactúan.

El estilo arquitectónico utilizado es el de cliente-servidor, siendo el *Activiti Explorer* la interfaz web que implementa el portal de BPMS para interacción con los diversos clientes que quieran utilizar el sistema. Por su parte, la organización del sistema cuenta con tres capas: presentación, lógica y persistencia. En la Figura 18 se presenta la vista lógica del sistema en dónde se puede observar el relacionamiento entre las distintas capas, dicha vista se realiza utilizando notación UML.[16]

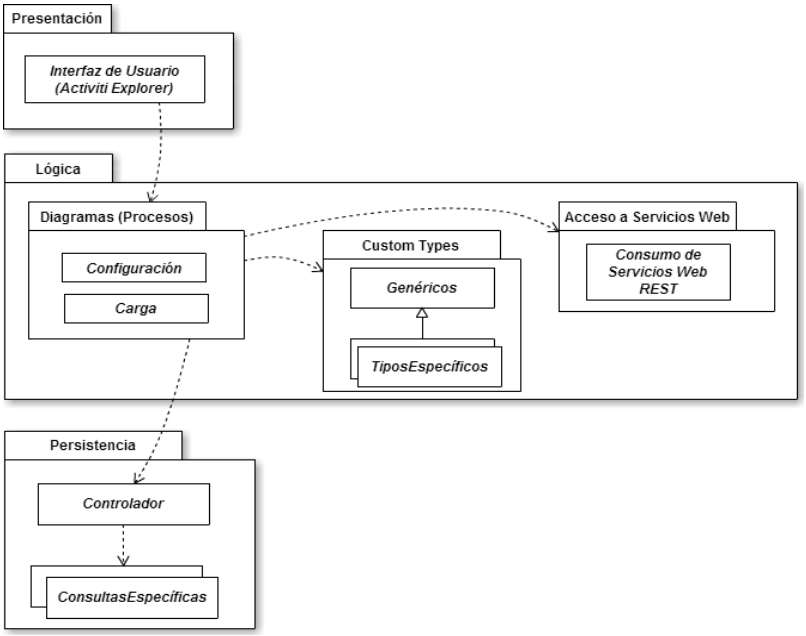


Figura 18 - Vista lógica del sistema

En cuanto a la capa de presentación, como fue mencionada previamente, es la interfaz de usuario propuesta por Activiti Explorer utilizando el componente Vaadin como forma de presentar los datos, ya sea mediante *inputs*, tablas, *combo boxes*, entre otros.

Por otro lado se tiene la capa lógica, en dicha capa se encuentra el núcleo del prototipo implementado y ofrece todas las funcionalidades del sistema, así como es la encargada de interactuar con otras capas del sistema. En particular en la capa lógica se decide encapsular el manejo para consumir servicios web externos implementados con la arquitectura REST. Dicho manejo se corresponde con la capa de servicios definida en la arquitectura del WW (ver Figura 5), y de esta forma, desde una clase asociada a un evento de una actividad se puede consumir servicios web específicos. El prototipo es capaz de consumir servicios para la extracción de datos, medición de la calidad de datos y también aquellos servicios relativos a la integración de entidades.

La capa de persistencia tiene como función principal manejar todo lo relacionado a la inserción, obtención y modificación de los datos relevantes de los procesos en ejecución en las distintas bases de datos según corresponda. Cada consulta específica encapsula un tipo de datos determinado, es decir se categorizan los pedidos según el tipo de datos y se delega a la clase que corresponda. Por lo tanto esta capa tiene la responsabilidad de comunicarse con más de una base de datos y tomar la acción que corresponda en cada una de ellas.
En la Figura 19 se presenta el diagrama de componente asociadas a subsistema de acceso a los datos, con dicho diagrama se busca clarificar la interacción del prototipo con las diversas bases de datos.

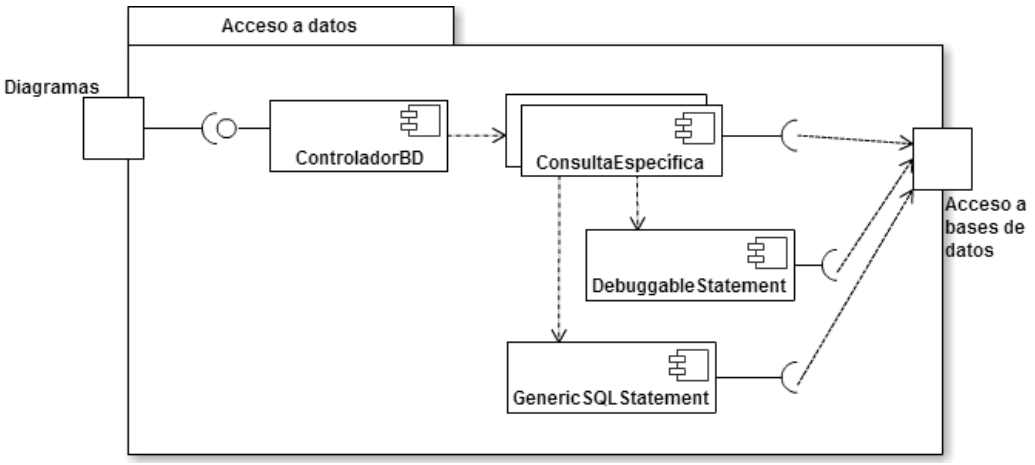


Figura 19 - Diagrama de componentes de acceso a datos

4.1.3 Vista Distribución (Deployment)

La vista de distribución o despliegue permite observar cómo cada componente será dispuesto físicamente en determinado nodo. A su vez se especifica la comunicación entre cada uno de dichos nodos. En la Figura 20 se presenta el diagrama de distribución elegido para el presente sistema.

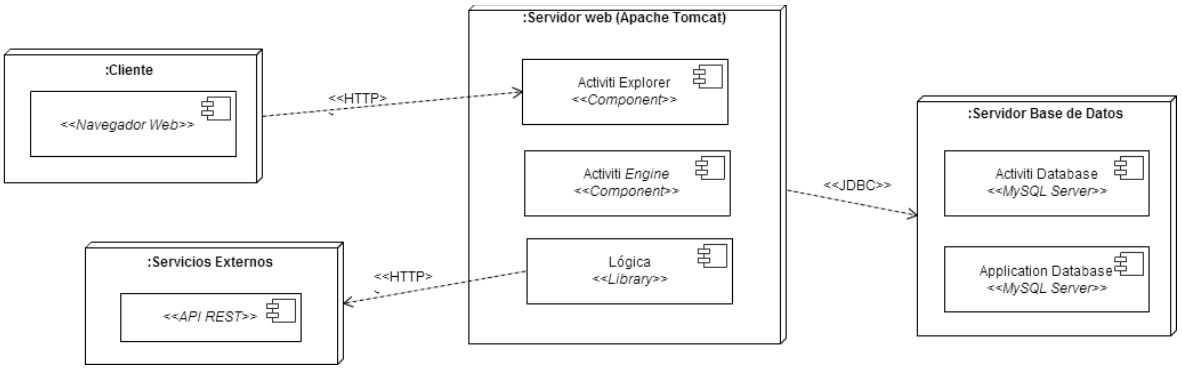


Figura 20 - Vista distribución del sistema

Como se puede observar el sistema se encuentra distribuido en cuatro nodos: Cliente, Servidor Web, Servicios Externos y Servidor Base de Datos.

El nodo Cliente representa el dispositivo físico utilizado por el cliente para acceder al sistema. Dicho acceso es realizado mediante un navegador web. En particular, el cliente interactúa con el componente "*Activiti Explorer*" provisto en el nodo Servidor Web.

Por su parte, el nodo Servidor Web, es el responsable de procesar los pedidos de los clientes y tomar las acciones correspondientes. Este nodo ejecuta un contenedor de aplicaciones (Tomcat) y encapsula tres componentes las cuales están asociadas a las capa de presentación y lógica. Un rol importante es el manejo de comunicaciones con el nodo asociado a las bases de datos, en este caso la comunicación con las mismas se realiza mediante el *driver* JDBC. El Servidor Web también se comunica con servicios web externos al sistema, tanto para realizar los pedidos como para procesar sus respuestas correctamente.

El nodo Servidor de Bases de Datos contiene, como su nombre lo indica, los servidores de bases de datos (en particular utilizando el motor MySQL) tanto de la base asociada a Activiti como de todas las otras bases de datos utilizadas por la aplicación.

Por último se tiene el nodo Servicios Externos, el cual contiene la publicación de aquellos servicios web que serán consumidos por el sistema mediante el protocolo HTTP utilizando REST. Este nodo potencialmente puede representar más de un nodo físico y ser los mismos independientes entre sí, ya que los servicios que se consumen pueden ser desarrollados por diferentes personas y/o empresas.

4.2 Proceso de Configuración Extendido

En este punto se desarrolla el diseño de la solución para el Proceso de Configuración extendiendo el mostrado en la Figura 12.

La Figura 21 ilustra el Proceso de Configuración Extendido modelado en BPMN 2.0 e implementado en Activiti BPMS; en rojo se resaltan los subprocesos y las actividades añadidos y/o modificados. El detalle de cada una de las actividades, su funcionamiento e

implementación se encuentra en el capítulo siguiente (ver Capítulo 5 - Implementación del Prototipo).

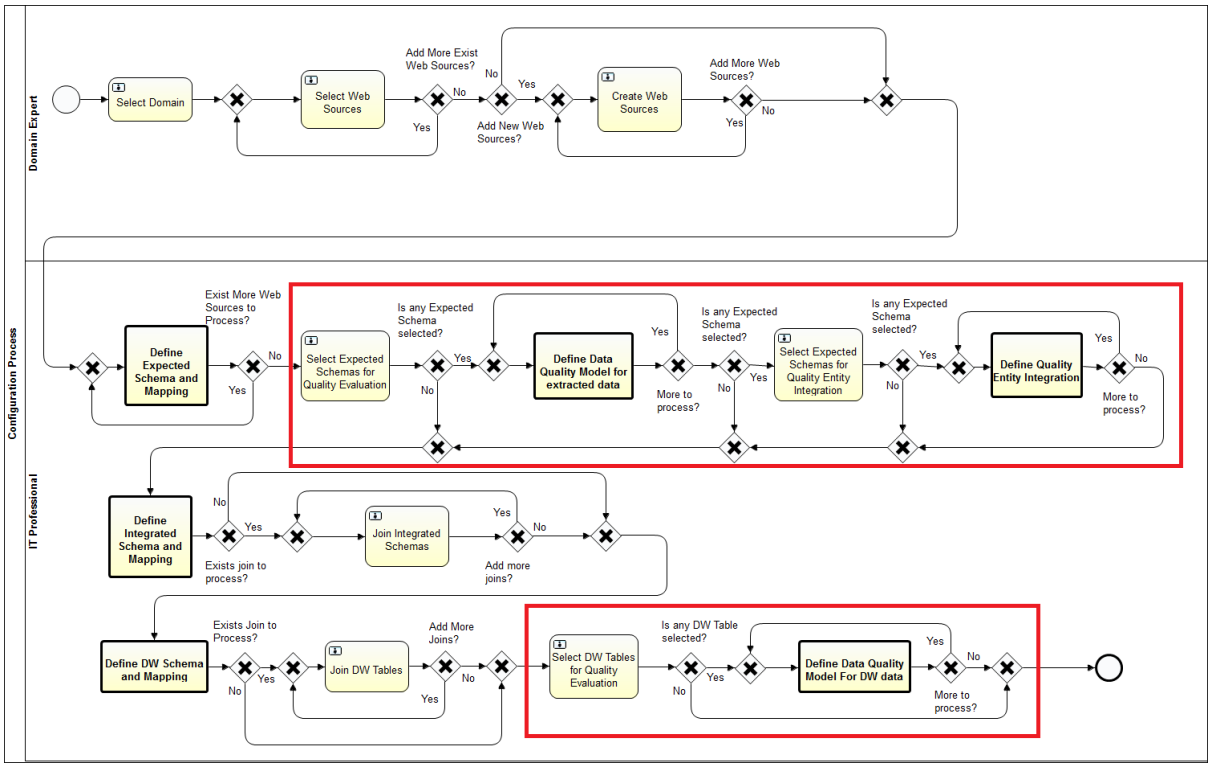


Figura 21 - Proceso de Configuración Extendido

Las figuras 22, 23 y 24 muestran subprocesos incluidos en el Proceso de Configuración que también sufrieron leves modificaciones con respecto al proyecto anterior relacionadas principalmente con la usabilidad del sistema, las cuales son mencionadas en la sección Mejoras del capítulo 5.

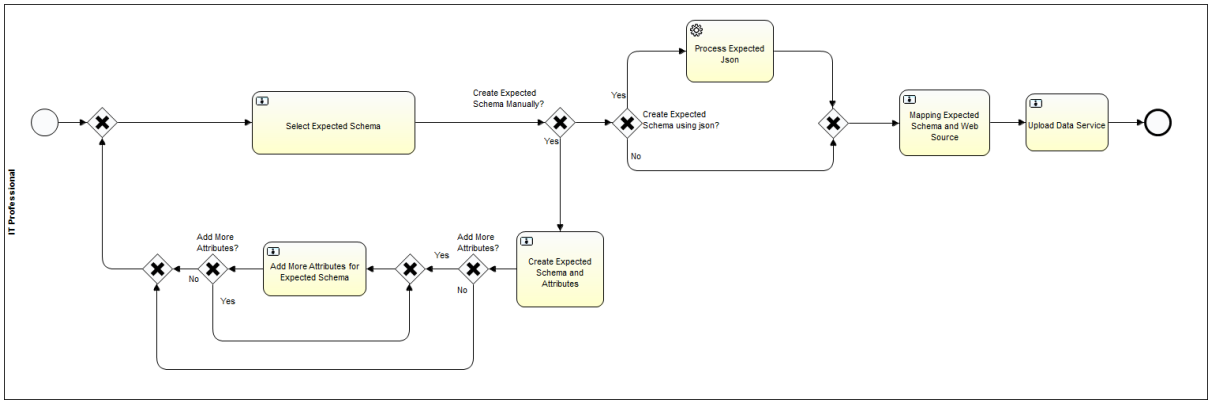


Figura 22 - Subproceso *'Define Expected Schema and Mapping'*

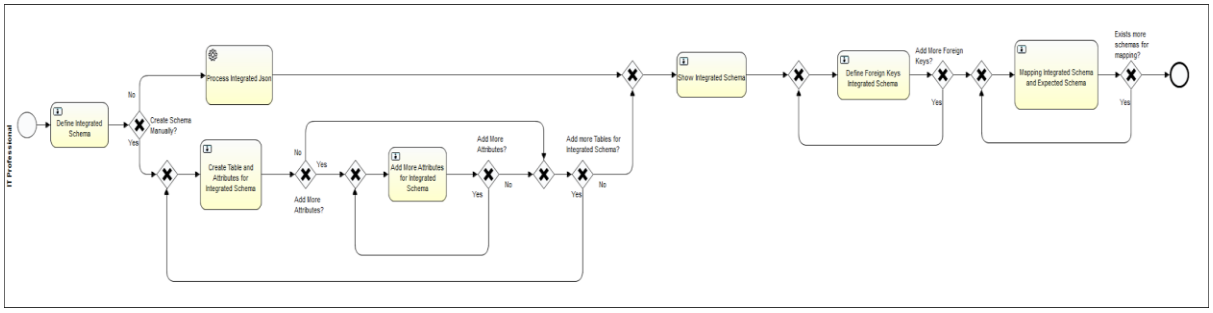


Figura 23 - Subproceso *'Define Integrated Schema and Mapping'*

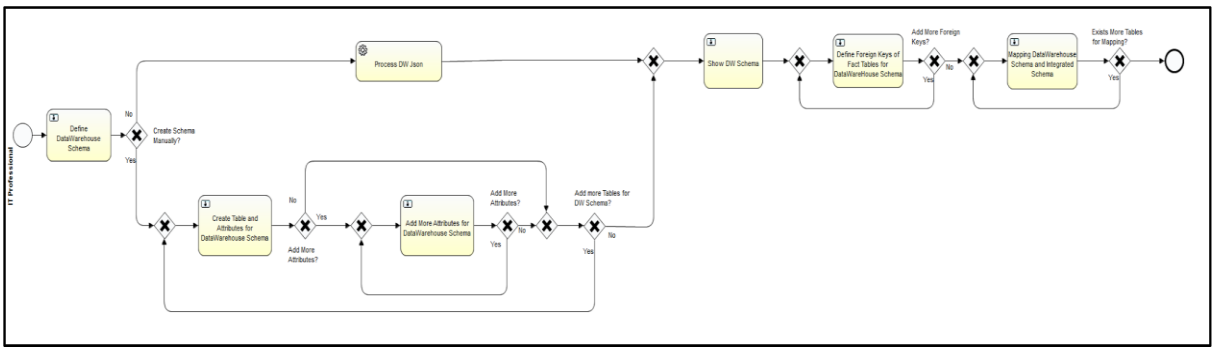


Figura 24 - Subproceso *'Define DW Schema and Mapping'*

A continuación se presenta la solución propuesta para la Fase de Configuración para cada uno de los problemas planteados en el capítulo anterior.

Definir el Modelo de Calidad para los datos recién extraídos de la web

En este punto se describe cómo se define el Modelo de Calidad de los datos recientemente extraídos de la web, es decir, sobre los *Expected Schemas*. Para definir dicho modelo se deben especificar las dimensiones y factores de calidad asociados, las métricas de calidad (método para medirla y granularidad) junto con los datos o conjuntos de datos del *Expected Schema*. En la solución propuesta se considera la granularidad por Celda, por Conjunto de Celdas y por Tabla.

Para definir cuál es el método que calcula la métrica se debe especificar un servicio que luego realice las mediciones de calidad en la fase de Carga. Estos servicios de calidad son independientes del sistema, es decir deben ser implementados por algún técnico y ser publicados por alguna aplicación externa. La unidad en la que se mide el valor de calidad queda a criterio del servicio implementado, es decir el valor de calidad podría ser un número real entre 0 y 1 en donde 1 corresponde al valor con mayor calidad y 0 al de menor calidad. En el anexo C se presenta documentación de cómo implementar un servicio de Calidad.

En el proceso de Configuración se le presenta al usuario la posibilidad de seleccionar los *Expected Schemas* a los que se quiere definir el modelo de calidad (tarea '*Select Expected Schemas For Quality Evaluation*' en Figura 21). Luego, el subproceso '*Define Data Quality Model For Extracted Data*' incluido en el proceso de la Figura 21 es el encargado de definir los modelos de calidad; el mismo se ilustra en la Figura 25.

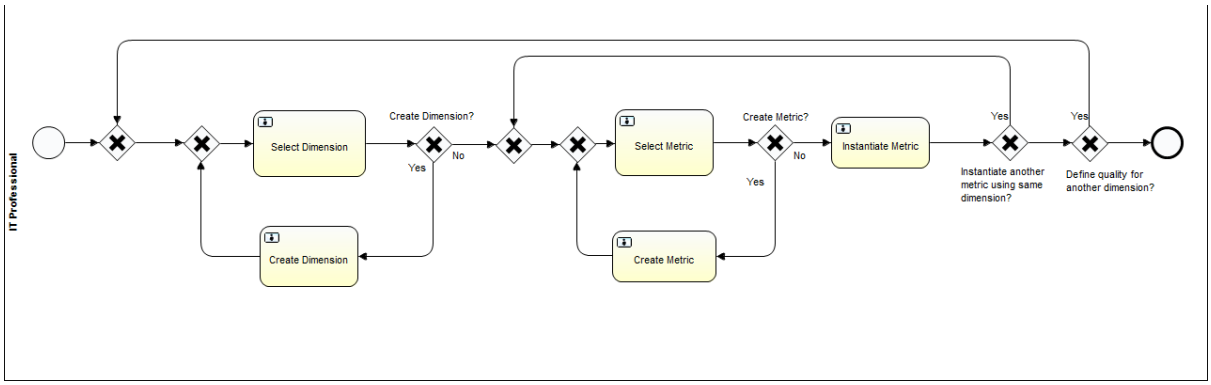


Figura 25 - Subproceso *'Define Data Quality Model For Extracted Data'*

Mejora de la Integración en la Carga del WW

En esta sección se describe las extensiones sobre el Proceso de Configuración para la mejora de la integración en la Fase de Carga.

La tarea '*Select Expected Schemas for Quality Entity Integration*' (ver Figura 21) y el subproceso '*Define Entity Integration*' que se ilustra en la Figura 26, corresponden a las extensiones realizadas para mejorar la integración en el proceso de Carga.

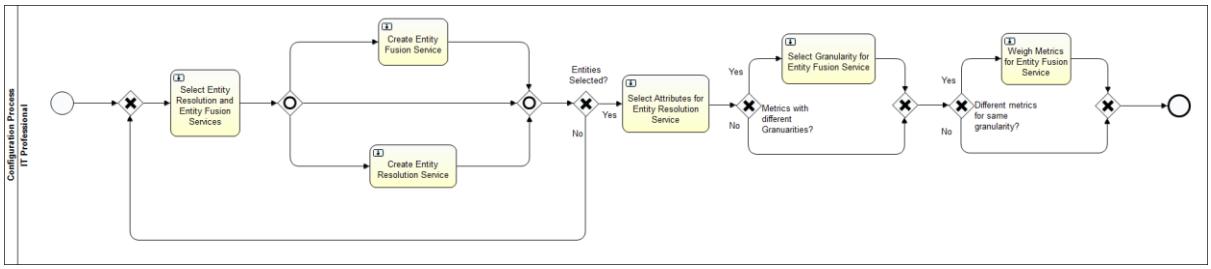


Figura 26 - Subproceso *'Define Entity Integration'*

Como se mencionó en el punto 3.3, en el caso que a un *Expected Schema* le corresponda más de una FDW se deben unificar los datos o registros que refieren a la misma entidad.
Por lo tanto, para cada *Expected Schema* se realiza una integración de Entidades (Entity Integration). Primero se detectan los duplicados (Entity Resolution) y luego tomando en cuenta los metadatos de calidad se seleccionan las tuplas más confiables (Entity Fusion).

Al igual que para el caso de la definición de los Modelos de Calidad, se decidió que el usuario primero seleccione sobre cuáles *Expected Schemas* (asociados a más de una FDW) desea definir servicios de Entity Integration (ver tarea *'Select Expected Schemas for Quality Entity Integration'* en Figura 21). Es importante señalar que, dado que el servicio Entity Fusion emplea metadatos de calidad para solucionar los duplicados, sólo se podrá definir servicios de integración sobre *Expected Schemas* a los que se les haya instanciado antes métricas de calidad. Igualmente, para las Expected Tables asociadas a más de una FDW a la que no se definió servicios de integración, el sistema cuenta con servicios de

Integración por defecto que no consideran los metadatos de calidad al momento de la integración.

Para definir los servicios de Entity Integration a aplicar sobre un *Expected Schema* específico, el usuario debe seleccionar tanto el servicio Entity Fusion como el servicio Entity Resolution (si estos ya existen en el sistema) o crear servicios nuevos que podrán ser utilizados en una futura configuración sin importar el dominio.

Al momento de definir cómo el servicio Entity Resolution detecta duplicados se tiene por defecto la opción de considerar las primary key del *Expected Schema* o la opción de elegir los atributos del *Expected Schema* que desee.

Los servicios Entity Fusion sólo consideran métricas de calidad con igual granularidad a la hora de resolver la fusión de las tuplas. Es decir, el servicio resuelve la fusión de las tuplas considerando valores de calidad calculados de métricas del mismo nivel de granularidad.Por esta razón, si el usuario instancio métricas de distinta granularidad para un *Expected Schema*, se listan dichas granularidades y el usuario debe seleccionar una.

Una vez seleccionada la granularidad o cuando solo se definieron métricas de la misma granularidad, el usuario debe indicar para las métricas de esa granularidad un valor de ponderación (0 a 1), siendo la suma total de los valores de ponderación igual a 1. De este modo al momento de seleccionar una entidad entre varias duplicadas, el valor de calidad de una métrica de mayor ponderación tendrá más peso que el valor de calidad de otra métrica de menor ponderación. En caso que se haya definido una única métrica, se tendrá un valor de ponderación por defecto igual a 1.

Cabe resaltar que los metadatos de calidad y los metadatos sobre los servicios de Entity Integration que son definidos durante la Configuración se persisten sobre la Base de Metadatos de Configuración ya que son utilizados al momento de la carga del WW. En 4.4, se profundiza sobre este punto.

Definir el Modelo de Calidad para los datos del DW

El subproceso '*Define Data Quality Model For DW Data*' (ver Figura 27) define el Modelo Calidad para los datos del Data Warehouse, el cual es análogo al proceso de definir la calidad del *Expected Schema*. La particularidad de este proceso es que interesa indicar también cuando se está instanciando una métrica para una tabla de hechos, y cuando, no. Además se le indica al usuario cuáles son los atributos que son medidas en la tabla de hechos.

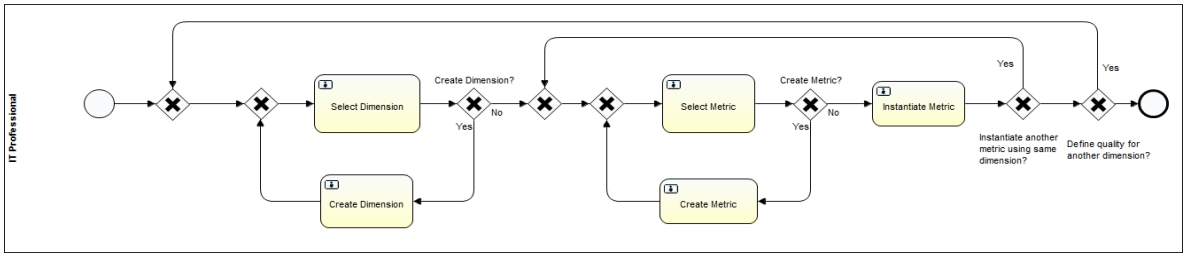


Figura 27 - Subproceso *'Define Data Quality Model For DW Data'*

4.3 Proceso de Carga

En este punto se desarrolla el diseño de la solución propuesta para el Proceso de Carga, el cual se diseñó por completo tal como se mencionó en 1.5 - Gestión del Proyecto.

La Figura 28 muestra el Proceso de Carga modelado en BPMN 2.0 e implementado en Activiti BPMS [6] y la Figura 29 exhibe el subproceso '*Do Quality Entity Integration*'incluido en dicho proceso. El detalle de cada una de las actividades, su funcionamiento e implementación se encuentra en el capítulo siguiente (Capítulo 5 - Implementación del Prototipo).

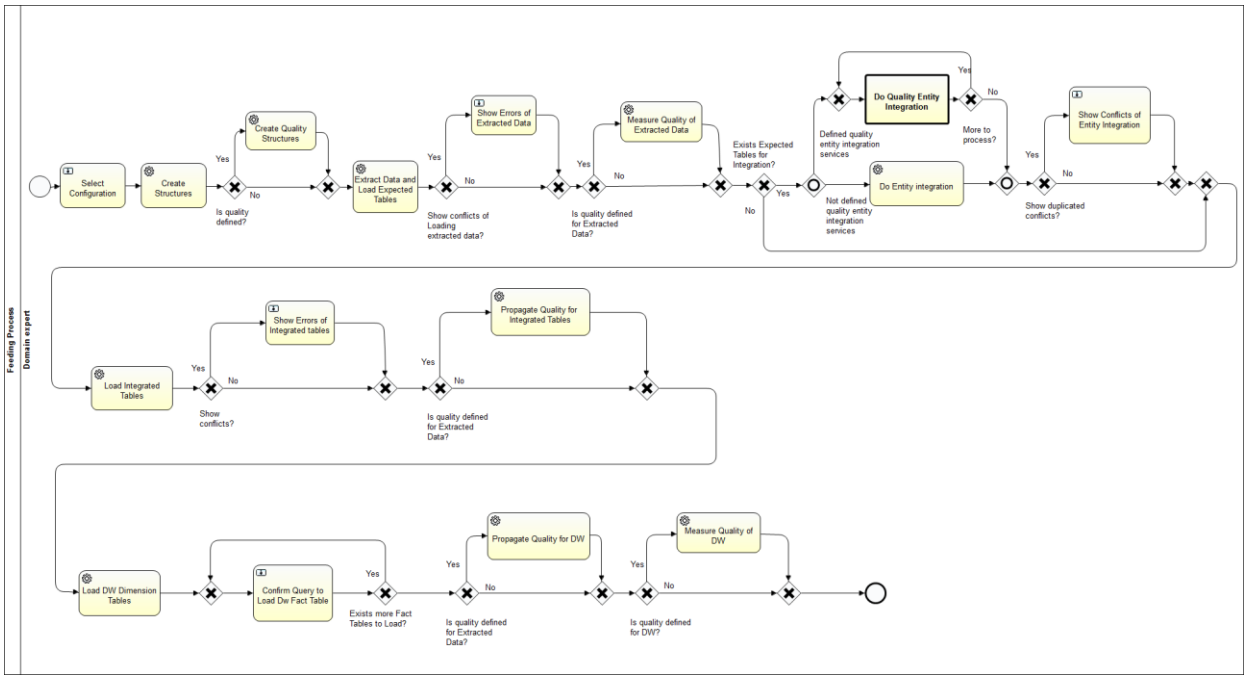


Figura 28 - Proceso de Carga

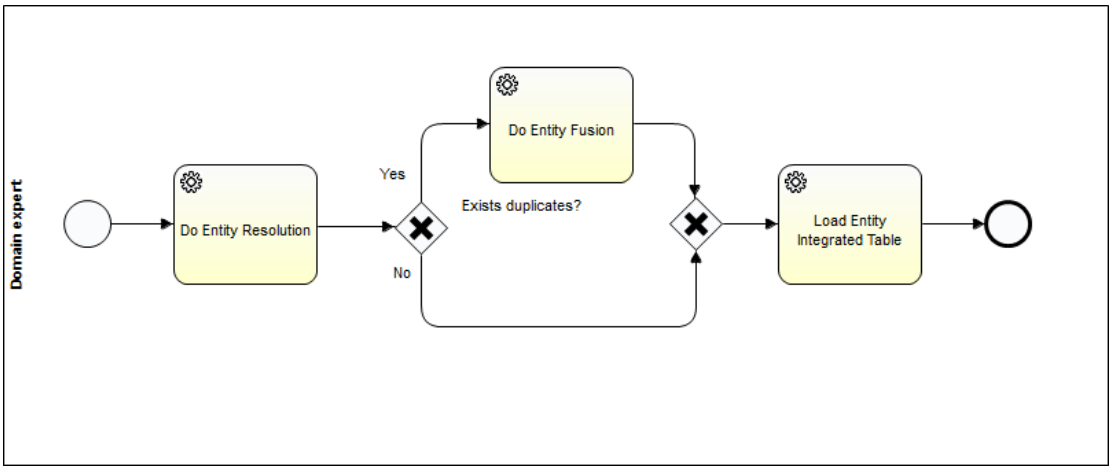


Figura 29 - Subproceso *'Do Quality Entity Integration'*

Uno de los objetivos que se planteó al momento de diseñar el Proceso de Carga es que el mismo fuera lo más automatizable posible, dejando las tareas manuales sólo para la resolución de conflictos por parte del usuario.

En la primera tarea el usuario debe seleccionar una configuración para realizar la carga del WW. En esta misma tarea el usuario puede optar por mostrar los conflictos de integración y errores (con la posibilidad de resolver los conflictos de Entity Integration) durante el Proceso de Carga o no.

Una vez seleccionada la configuración, el proceso crea las estructuras intermedias que se necesitan para extraer e integrar los datos, las estructuras del DW (tablas de hechos y tablas de dimensiones), y en caso que haya definido calidad en la configuración seleccionada, las estructuras que almacenarán los metadatos de calidad. Estas estructuras se almacenan en tres bases de datos que se describen brevemente a continuación (ver 4.4 -Diseño de los Modelos de Datos):

Base de Datos intermedia
Esta base contiene todas las estructuras intermedias necesarias para la extracción e integración de los datos. En caso que se produzcan conflictos al momento de extraer los datos o integrarlos, estos son registrados en tablas específicas en esta misma base que el usuario podrá consultar una vez finalizada la carga. También contiene la traza de los datos de los *Expected Schema*, es decir de que FDW provienen los datos.

Base de DW
Dicha base contiene las tablas de dimensiones y las tablas de hechos que conforman el DW.

Base de Metadatos de Calidad
En esta base se almacenan los metadatos de Calidad, incluyendo los metadatos generales de Calidad (métricas instanciadas) que se persistieron en la base de Configuración, las estructuras pertinentes para almacenar las mediciones de calidad que se realicen y la propagación de la calidad (sólo en caso que se haya definido calidad sobre los datos recién extraídos de la web).

En trabajos previos [2] la definición de los metadatos de calidad corresponde a una tarea manual que se realiza en la Fase de Configuración (ver 11 en Figura 15). En este proyecto se decidió que esta tarea sea automática para simplificar el proceso para el usuario final y que se ejecute en el proceso de Carga debido a que los metadatos de calidad deben referenciar a los datos de los *Expected Schemas* y del DW, según el caso, que son cargados en esta fase. Por lo tanto, todas las configuraciones utilizan una base de metadatos de calidad con las mismas características generales y se cuenta con una tarea automática que personaliza dicha base a partir de los metadatos definidos en la configuración.

Cabe destacar que en el caso que se ejecute un proceso de Carga para una configuración que ya ha sido cargada, se borran todas las bases existentes y se vuelven a generar. El Proceso de Carga draft del proyecto anterior, vacía las tablas para luego volverlas a cargar

[3], sin embargo, en este proyecto se optó por eliminar directamente las bases con el fin de mejorar la performance.

Extracción de los Datos
Luego de creadas todas las estructuras, se procede a la extracción de los datos. Para ello, se tiene una **Expected Table** en la Base de Datos Intermedia por cada Expected Schema y FDW definido en la configuración. Utilizando los metadatos de la configuración, se extraen los datos de las FDW empleando los servicios web de extracción y se cargan tomando en cuenta los mapeos entre FDW y *Expected Schemas*.

Al momento de realizar la extracción de los datos de las FDW, algunas tuplas podrían no ser cargadas a las Expected Tables debido a restricciones definidas en el *Expected Schema* (tipos de datos, largo de los campos, primary keys). Por esta razón, se cuenta con una tarea manual que despliega un log de errores, en donde se detalla la tupla que no pudo ser cargada y un mensaje de error indicando el motivo. Para poder ayudar al usuario a entender los problemas que hubo en la carga y poder registrarlos en el log de manera uniforme se decidió realizar una categorización de errores posibles que pueda luego ser extendida. Por tanto, los errores se categorizaron en:

- Tipos de datos incompatibles (excede campo, distinto tipo)
- Campo Requerido null
- Clave duplicada
- Otro tipo de error.

Una vez realizada la extracción de los datos, en caso que en la configuración se hubiera instanciado métricas de calidad sobre los *Expected Schemas*, se procede a realizar las mediciones de calidad, utilizando los servicios de medición seleccionados en la configuración, persistiendo las mismas en la Base de Metadatos de Calidad.

En este punto, se tiene por cada FDW, una tabla Expected Table que contiene los datos esperados de la FDW. Sin embargo, en caso que se haya asociado a un *Expected Schema* más de una FDW, se debe realizar la integración de entidades (Entity Integration).

Resolución de Integración de Entidades
Luego se realiza la integración de las Expected Tables asociadas a los *Expected Schemas* a los que se definió servicios de integración. Primero el servicio Entity Resolution definido encuentra las entidades duplicadas y luego el servicio Entity Fusion resuelve la integración de entidades empleando las mediciones de calidad sobre cada Expected Table.

En algunas ocasiones los metadatos de calidad pueden tener valores similares para dos entidades a integrar y el servicio Entity Fusion no es capaz de resolver dicha integración. Para estos casos se tomó la decisión que el sistema muestre al usuario en una tarea manual cuáles son las entidades duplicadas no resueltas y que el mismo usuario experto del dominio decida cuál entidad desea que continúe la carga del WW.

Para las Expected Tables asociadas a más de una FDW a las que no se definieron servicios de integración, la selección de la entidad más confiable depende del algoritmo implementado por el servicio de integración por defecto del sistema, por lo tanto, se optó por

mostrar en pantalla al Usuario Experto en el Dominio todas las entidades seleccionadas con la posibilidad de modificar la selección.

Como se puede apreciar, en el proceso de carga (ver Figura 28) primero se cargan las Expected Tables, luego se calculan las métricas de calidad definidas sobre cada una de ellas y a partir de los metadatos de calidad generados, se resuelve la integración de entidades. Por lo tanto, para que sea más intuitivo para el usuario, en el proceso de Configuración primero se definen las métricas de calidad a aplicar sobre cada *Expected Schema* y luego los servicios Entity Integration que resuelven la integración de entidades (ver Figura 29).

Mejora de la Integración entre Expected Tables y carga final del DW

Luego, se realiza la carga de las Integrated Tables, que son las tablas definidas en el *Integrated Schema* y que se crean en la Base de Datos Intermedia. En este caso también se pueden dar errores de tipos o campos requeridos al insertar los datos a las Integrated Tables (se usa la misma categorización que para los errores de extracción). También se pueden dar conflictos de integración por los joins entre las Expected Tables, por ejemplo en el caso que existan tuplas en un *Expected Schema* que no hacen join con tuplas de otro Expected. Todos estos errores son desplegados al usuario en una tarea manual con el fin de que el usuario final sea consciente de dichos errores y pueda tomar decisiones al respecto.

Una vez que se cuenta con las Integrated Tables cargadas completamente, lo que sigue es cargar las tablas del DW. Las tablas de dimensiones son cargadas directamente con los datos definidos previamente en la configuración. Respecto a las tablas de hechos, el sistema genera para cada una de dichas tablas la consulta SQL que la carga, y las muestra una por una al usuario para que las confirme o eventualmente realice alguna modificación. En caso que la modificación del usuario contenga errores de sintaxis, se despliega en pantalla el error del manejador de base de datos y el usuario tiene nuevamente la posibilidad de modificar la consulta hasta que la consulta sea correcta. Cabe resaltar que las modificaciones que se realizan sobre la consulta SQL no quedan persistidas en una futura carga de esa misma configuración ya que el sistema genera la consulta a partir de los datos de la configuración, los cuales no son actualizados en esta etapa.

Una vez culminada la carga del DW, se realizan las mediciones de calidad sobre las tablas de dimensiones y de hechos a las que se hubieran definido métricas de calidad en la fase de Configuración. Estos metadatos de calidad generados son persistidos también en la Base de Metadatos de Calidad.

Propagación de la Calidad

En esta sección se detalla el diseño de la propagación de la calidad durante la Fase de Carga.

En la fase carga, una vez realizadas las mediciones de calidad sobre los datos extraídos de la web, se realiza la propagación de los metadatos de calidad a lo largo de todo el proceso, es decir la propagación directa de la calidad, manteniendo una correspondencia entre los metadatos de calidad y los datos del WW. La propagación de la calidad se realizó de

Expected Tables (tablas a las que se mide la calidad) a Integrated Tables y de Integrated Tables a DW Tables.

Para efectuar la propagación de Expected Tables a Integrated Tables se tomaron en cuenta los siguientes puntos:

- Las mediciones de calidad de granularidad Celda se propagan siempre que el atributo sea mapeado en el Integrated.

- Si se integra más de una Expected Table para conformar una Integrated Table y en alguno de las Expected Table se midió calidad de granularidad Tabla, dichos valores de calidad no se propagan ya que ahora el Integrated no tiene ese valor de calidad a nivel de tabla. En cambio en el caso que se pase de una Expected Table a una Integrated Table de manera directa (todos los atributos del Integrated son mapeados con todos los atributos del mismo Expected), ese valor de calidad si se conserva. Por ejemplo, si para los mapeos entre Expected e Integrated de la Figura 30 se tienen definidas métricas de granularidad 'Tabla' sobre los Expected 'expected_schema1' y 'expected_schema3', el valor de calidad del Expected 'expected_schema1' no se propaga ya que el *Integrated Schema* 'integrated_schema1' contiene también atributos que provienen del Expected 'expected_schema2' y por tanto el valor de calidad a nivel de tabla cambió. Mientras que en el caso del Integrated 'integrated_schema2' como todos los atributos son mapeados del Expected 'expected_schema3', el valor de calidad se mantiene a nivel de Tabla, y por tanto, se propaga.

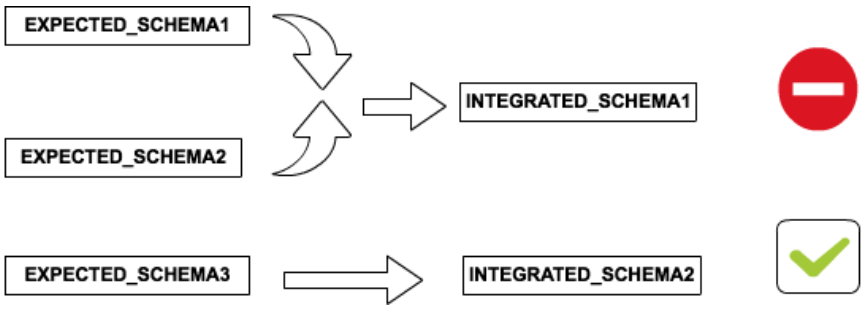


Figura 30 - Ejemplo de propagación de calidad de métricas de granularidad Tabla

- Para el caso de la calidad de granularidad Conjunto de Celdas, se razona de manera análoga al punto anterior. Es decir, si se midió una métrica de granularidad Conjunto de Celdas en una Expected Table y alguno de los atributos que conforman el Conjunto de Celdas no es mapeado en el Integrated, este valor de calidad no es propagado. En el ejemplo de la Figura 31, se tiene el Expected 'expected_personas' que mapea con el Integrated 'integrated_personas' y una métrica de granularidad 'Conjunto de Celdas' definida sobre los atributos 'edad' y 'esmayor' del Expected. Para el primer mapeo, como los dos atributos son mapeados al Integrated, el valor de calidad se conserva, y por lo tanto, se propaga la calidad. Para el segundo

mapeo, sólo se mapea el atributo ‘edad’, y dado que se pierde el atributo ‘esmayor’, el valor de calidad no se conserva.

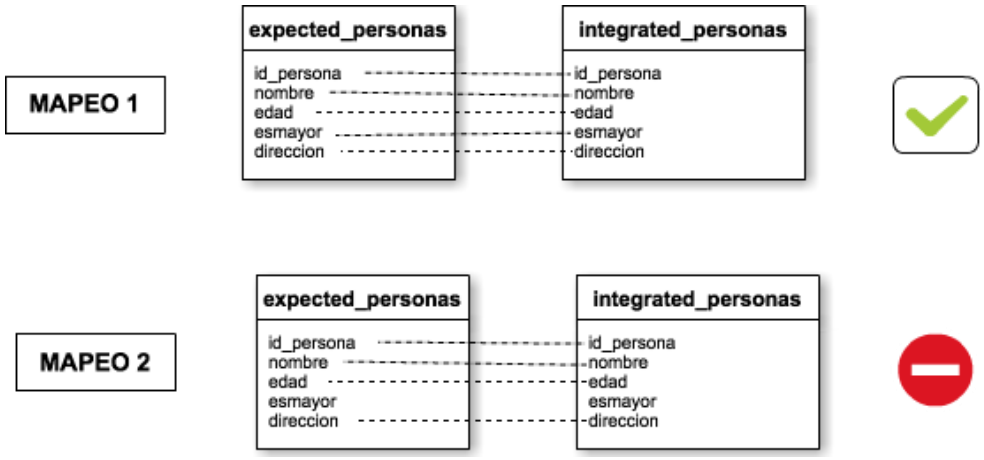


Figura 31 - Ejemplo de propagación de calidad de métricas de granularidad Conjunto de Celdas

Se debieron tomar consideraciones especiales para la propagación de calidad de las Expected Tables asociadas a más de una FDW. Esto se debe a que por cada FDW asociada al mismo *Expected Schema*, se genera una Expected Table a la cual se realizan mediciones de calidad; y en caso que existan entidades duplicadas entre estas tablas, el servicio Entity Fusion devolverá una sola tupla para cada entidad, siendo necesario elegir el valor de calidad asociado a dicha tupla. La solución propuesta para esto considera solamente el caso en que la fusión de entidades se resuelve seleccionando una de las tuplas de entrada completa. No es la mejor solución para el caso en que la fusión realiza un ‘merge’ de las tuplas duplicadas para generar la tupla resultado.

Por otro lado, para la propagación de la calidad de las Integrated Tables a las DW Tables se razonó de manera análoga al pasaje de las Expected Tables a las Integrated Tables.

También se consideró el caso en el que se tiene definida calidad para uno de los atributos de la condición del join de los *Expected Schemas* que conforman las Integrated Tables y este atributo no es mapeado en una tabla del *Integrated Schema*. En estos casos se propaga la calidad del atributo simulando que la calidad se definió en el atributo que sí es mapeado. La Figura 32 ejemplifica este caso. Como se puede apreciar, se tienen los *Expected Schemas* 'alojamientos' y 'países' y los mismos se integran para conformar el *Integrated Schema* 'alojamientos' haciendo join entre los atributos 'codigo_pais' e 'id_pais'; en el mapeo del Integrated con ambos Expected sólo el atributo 'codigo_pais' es mapeado. Por lo tanto, si se define calidad sobre el atributo 'id_pais' del Expected 'paises', se propagará la calidad en el atributo 'codigo_pais' del Integrated alojamientos.

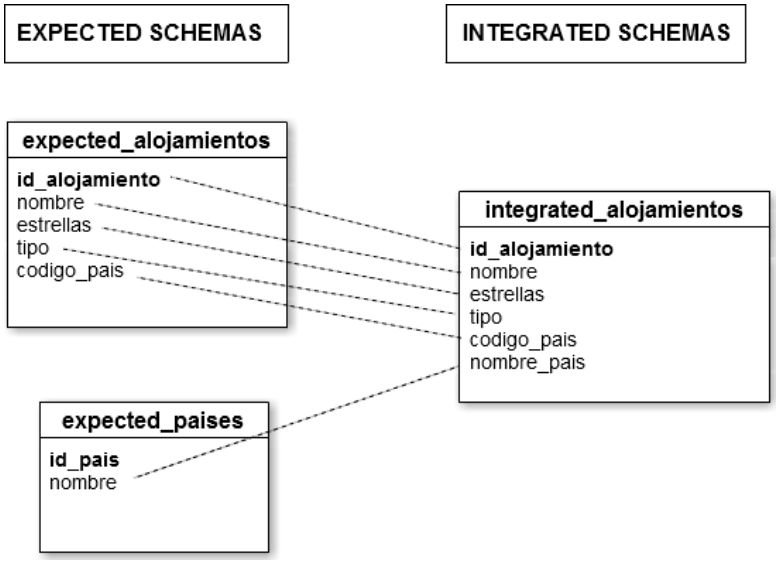


Figura 32 - Ejemplo de caso especial de propagación de calidad.

4.4 Diseños de los Modelos de Datos.

En esta sección se presentan los diseños de los Modelos de Datos de la solución propuesta, incluyendo los cambios y extensiones realizados sobre el Modelo de Datos de la Base de Configuración y de la Base de Datos del proyecto anterior así como el diseño del Modelo de Datos de la Base de Metadatos de Calidad.

4.4.1 Base de Metadatos de Configuración

La Base de Metadatos de Configuración se nutre durante el Proceso de Configuración con todas las definiciones que realiza el usuario, y luego es consultada durante todo el Proceso de Carga para lograr realizar la extracción, integración y carga de los datos. [3]

Se debió extender este Modelo de Datos para incluir los aspectos de calidad en el proceso así como las definiciones de los servicios que mejoran la integración de entidades. Cabe resaltar, como se mencionó en el capítulo 1, que se realizó el pasaje de la Base de Metadatos de Configuración del proyecto anterior del español al inglés dando oportunidades a mejoras y correcciones. En esta sección sólo se explican las tablas asociadas a las extensiones realizadas; por detalles de las demás tablas consultar Anexo C (Manual Técnico).

La Figura 33 muestra las tablas del modelo asociadas a los aspectos de calidad; en el mismo se especifican las tablas, junto a sus atributos y tipos de datos así como las relaciones entre dichas tablas. En estas tablas se guardan los conceptos generales de calidad globales a todas las configuraciones (dimensiones, factores, métricas y granularidades), y también los particulares que el usuario decidió aplicar en esta configuración. Mientras que la Figura 34 muestra las tablas asociadas a la mejora de la integración de entidades.

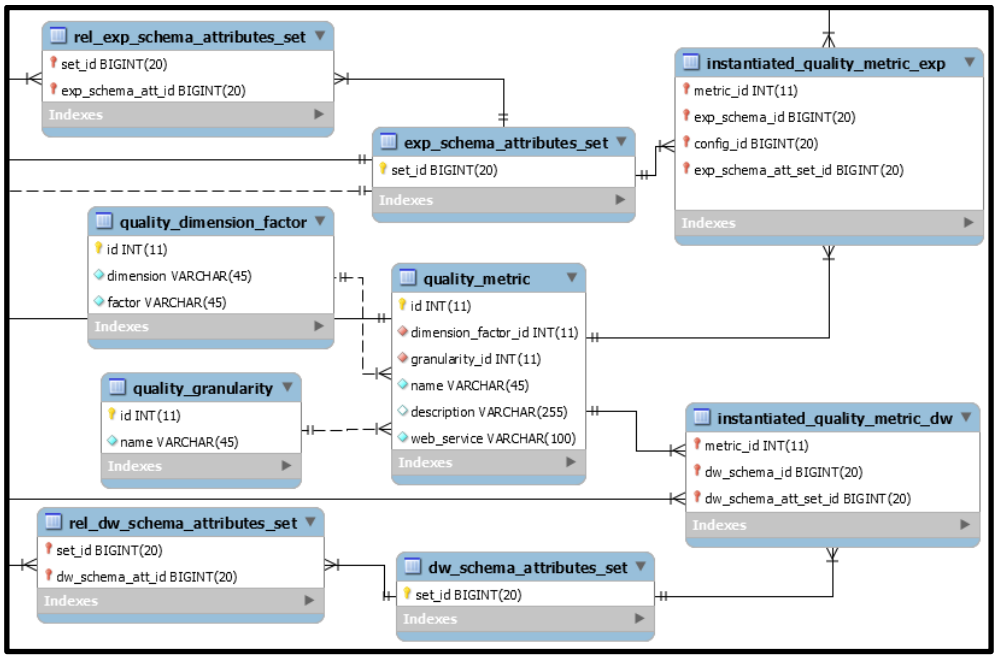


Figura 33 -Tablas de la Base de Metadatos de Configuración Extendido asociadas a los aspectos de calidad.

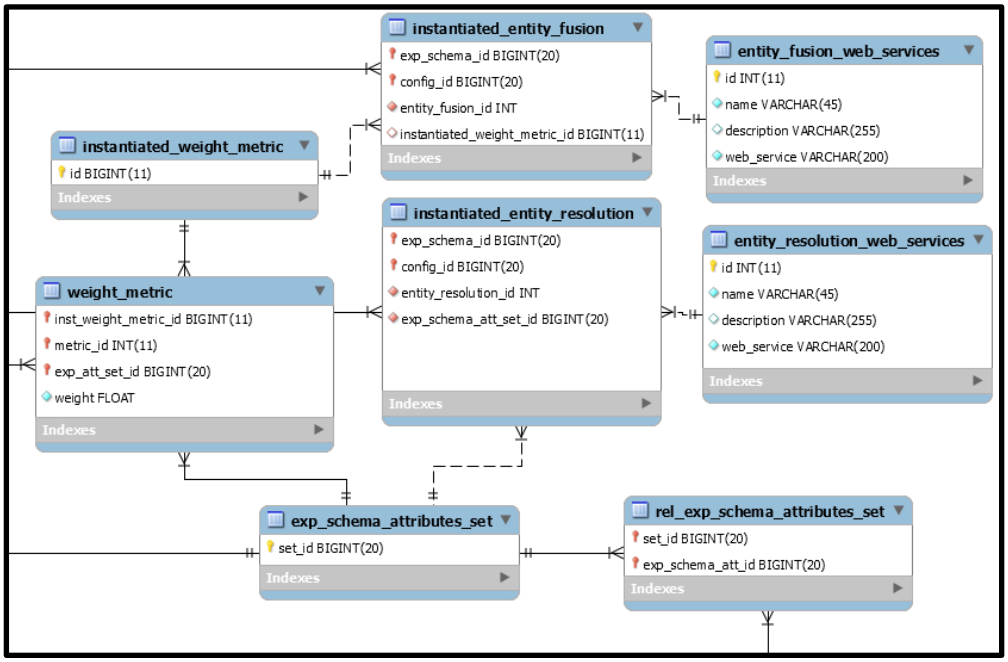


Figura 34 -Tablas de la Base de Metadatos de Configuración Extendido asociadas a la mejora de integración de entidades.

En la Tabla 1 se explican las tablas que almacenan los aspectos de calidad y en la Tabla 2, las tablas que almacenan los servicios de integración.

Tabla 1 - Descripción de las tablas referentes a los aspectos de calidad

TABLA	DESCRIPCIÓN
quality_dimension_factor	Esta tabla almacena todos los pares dimensión - factor disponibles en el Sistema. Se precargan los pares dimensión-factor más comunes y el usuario además tiene la posibilidad de añadir otros, que podrán ser utilizados en futuras configuraciones.
quality_granularity	Esta tabla contiene los distintos niveles de granularidad disponibles al momento de instanciar una métrica de calidad. Dichos valores son precargados en el sistema.
quality_metric	Esta tabla contiene todas las métricas de calidad existentes en el Sistema, es decir, las que se han agregado en ejecuciones anteriores y las que el usuario agregue durante el proceso, que podrán ser utilizadas en posteriores configuraciones.
instantiated_quality_metric_exp	Esta tabla registra todas las métricas de calidad instanciadas sobre los *Expected Schemas*. Es decir, la asociación entre una métrica y un *Expected Schema* en una configuración dada, indicando los atributos a los cuales aplica la métrica según el nivel de granularidad.
instantiated_quality_metric_dw	Esta tabla registra las métricas de calidad instanciadas sobre las tablas de los *DW Schemas*. Es decir, la asociación entre una métrica y una tabla del *DW Schema*, indicando los atributos a los cuales aplica la métrica según el nivel de granularidad.
exp_schema_attributes_set	Esta tabla contiene identificadores de conjuntos de atributos de un *Expected Schema*. El identificador -1 corresponde a la Granularidad Tabla.
rel_exp_schema_attributes_set	Esta tabla almacena la asociación entre un atributo de un *Expected Schema* y uno de los identificadores de conjuntos que contiene la tabla exp_schema_attributes_set.
dw_schema_attributes_set	Esta tabla contiene identificadores de conjuntos atributos de una tabla de un *DW Schema*. El identificador -1 corresponde a la Granularidad Tabla.
rel_dw_schema_attributes_set	Esta tabla registra la asociación entre un atributo de una tabla de un *DW Schema* y un identificador de un conjunto contenido en la tabla dw_schema_attributes_set.

Tabla 2 - Descripción de las tablas referentes a la mejora de la integración

TABLA	DESCRIPCIÓN
entity_resolution_web_services	En esta tabla se almacenan los servicios Entity Resolution ingresados por los usuarios, que pueden ser utilizados en futuras configuraciones para cualquier dominio.
entity_fusion_web_services	Esta tabla contiene los servicios Entity Resolution ingresados por los usuarios, que pueden ser utilizados en futuras configuraciones para cualquier dominio.
instantiated_entity_fusion	Esta tabla registra la asociación entre un *Expected Schema* (con más de una FDW) y un servicio Entity Fusion.
instantiated_entity_resolution	Esta tabla registra la asociación entre un *Expected Schema* (con más de una FDW) y un servicio Entity Resolution.
exp_schema_attributes_set	Esta tabla contiene identificadores de conjuntos de atributos de un *Expected Schema*.
rel_exp_schema_attributes_set	Esta tabla almacena la asociación entre un atributo de un *Expected Schema* y uno de los identificadores de conjuntos que contiene la tabla exp_schema_attributes_set.
instantiated_weight_metric	En esta tabla se almacena un identificador al momento de realizar una ponderación sobre las métricas de calidad que considera un servicio Entity Fusion para un *Expected Schema* y una configuración dada.
weight_metric	Esta tabla registra el valor de ponderación definido a una métrica de calidad para un servicio Entity Fusion, un *Expected Schema*, un conjunto de atributos de dicho Expected y una configuración dada.

Tanto la tabla 1 como la tabla 2contiene las tablas **exp_schema_attributes_set** y **rel_exp_schema_attributes_set** ya que estas tablas se utilizan tanto para registrar los atributos de un *Expected Schema* a los que se les aplica una métrica de calidad como para almacenar los atributos de un Expected que identifican una entidad para el servicio Entity Resolution.

4.4.2 Base de Datos

En la solución propuesta por el proyecto anterior en la Fase de Carga se genera una Base de Datos que contiene tanto las estructuras necesarias para la construcción del WW (extracción e integración de los datos) como las tablas propias del DW. En este proyecto, se tomó la decisión de separar las tablas del DW de todas las estructuras generadas durante la carga, para proveer al usuario final el DW resultado en forma totalmente independiente de las bases de datos intermedias y auxiliares del proceso.

4.4.2.1 Base de Datos Intermedia

En laFigura 35 se ilustra un diagrama de todas las tablas que permiten la extracción, integración y carga de los datos así como las tablas que registran los errores y conflictos en la carga. La tabla 3 describe cada una de estas tablas. Es importante destacar que las únicas tablas que están relacionadas por claves foráneas en este modelo son las Integrated Tables a las cuales se les define foreign keys en la configuración.

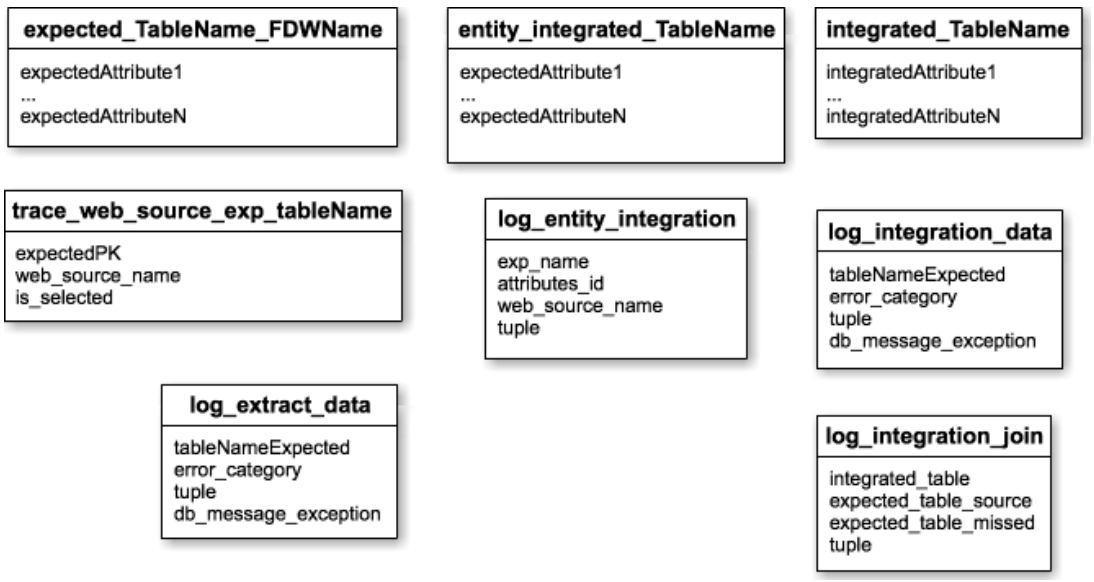


Figura 35 - Diagrama de la Base de Datos Intermedia

Tabla 3 - Descripción de las tablas referentes a la carga del WW

TABLA	DESCRIPCIÓN
expected_tableName_FDWName	Las Expected Tables tienen el formato expected_tableName_FDWName, en donde tableName corresponde al nombre que se le asignó al *Expected Schema* durante la Fase de Configuración y FDWName el nombre de la FDW asociada. Se optó por incluir la FDW en el nombre ya que un mismo *Expected Schema* puede tener asociado más de una FDW.
log_extract_data	Esta tabla almacena los errores producidos durante la extracción de los datos. En caso que no haya errores, no se crea la tabla.
entity_integrated_TableName	Esta tabla se registra el resultado del Entity Integration entre Expected Tables asociadas al mismo *Expected Schema*. 'TableName' corresponde al nombre del *Expected Schema*.
trace_web_source_exp_tableName	Esta tabla aloja la asociación entre los datos de un *Expected Schema* y la FDW de las que provienen. También indica si una entidad del *Expected Schema* continua o no siendo considerada en el proceso de carga a través del atributo '*is_selected*'.
log_entity_integration	Esta tabla se crea sólo en caso que existan más de dos FDW asociadas al mismo *Expected Schema* y almacena aquellos conflictos que el servicio Entity Fusion no alcanzó a solucionar.
integrated_tableName	Estas tablas corresponden a las Integrated Tables y '*tableName*' referencia al nombre asignado al *Integrated Schema* durante la fase de Configuración.
log_integration_join	Tabla que almacena los conflictos al momento de realizar el join entre las Expected Tables para conformar las Integrated Tables.
log_integration_data	Tabla que almacena los errores generados al cargar las Integrated Tables.

4.4.3 Base de DW

En la Tabla 4 se presentan las tablas que contiene el DW.

Tabla 4 - Descripción de las tablas referentes al DW

TABLA	DESCRIPCIÓN
dw_table_tableName	Las tablas con este formato corresponden a las tablas de dimensiones del DW resultado, en donde '*tableName*' corresponde al nombre que se le asignó a la tabla durante la Fase de Configuración.
dw_fact_table_tableName	Las tablas con este formato corresponden a las tablas de hechos del DW resultado, en donde '*tableName*' corresponde al nombre que se le asignó a la tabla de hechos durante la Fase de Configuración.

4.4.4 Base de Metadatos de Calidad

En este punto se explica el Modelo de Datos de la Base de Metadatos de Calidad, el cual permite consultar y analizar la calidad para los datos en las distintas etapas del proceso, es decir la calidad medida de las FDW, la calidad propagada y la calidad asociada al DW final.

En esta base se almacenan las métricas instanciadas en la configuración seleccionada y los datos que definen estas métricas, sus factores y sus dimensiones de calidad (estos datos fueron incluidos con el fin de independizar la Base de Metadatos de Calidad de la Base de Metadatos de Configuración). Para simplificar el modelo, se omitieron algunos de los identificadores provenientes de las tablas de la Base de Metadatos de Configuración. La Figura 36 ilustra el esquema general de la Base de Metadatos de Calidad, pero para cada instanciación, la cantidad de tablas puede variar según lo definido en el proceso de Configuración.

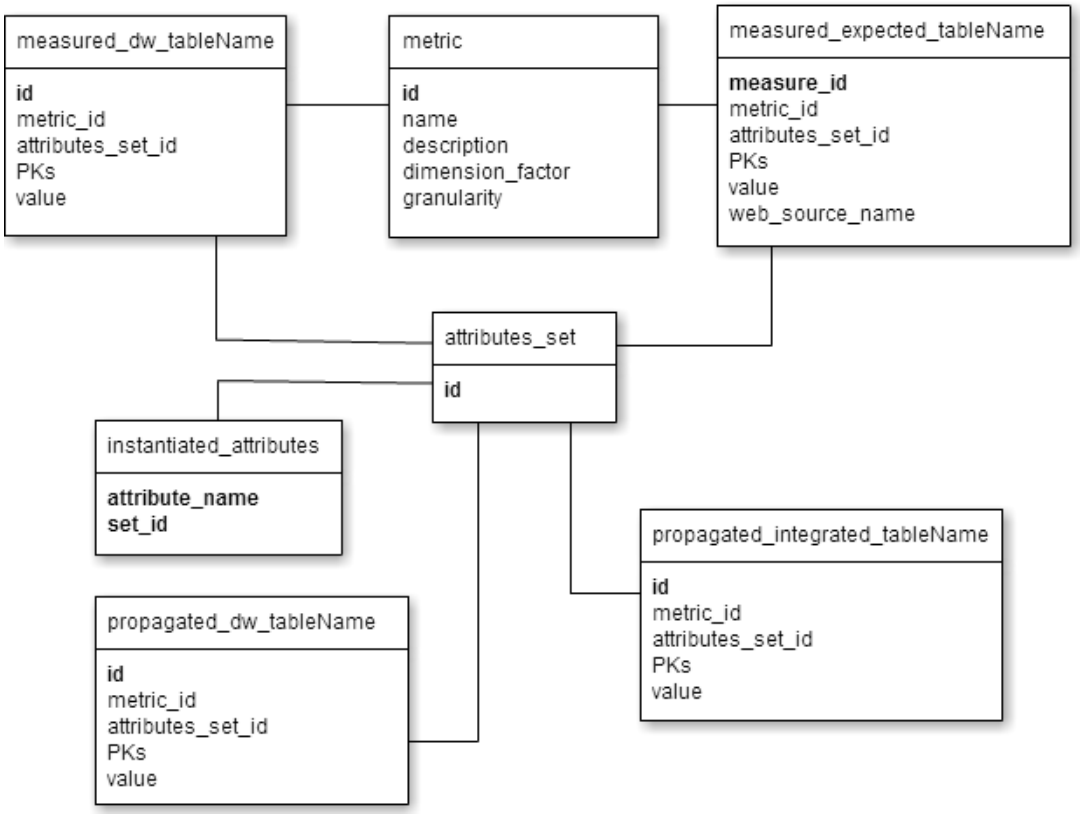


Figura 36 - Modelo de Base de Metadatos de Calidad

A continuación se presenta en la Tabla 5 la descripción de las tablas que se generan durante la Fase de Carga para almacenar los metadatos de Calidad.

Tabla 5 - Descripción de las tablas de los metadatos de calidad

TABLA	DESCRIPCIÓN
metric	Esta tabla contiene todas las métricas de calidad que fueron instanciadas en una configuración dada tanto sobre los datos recién extraídos de la web como del DW resultado.
measured_expected_tableName	Esta tabla contiene las mediciones de calidad de los datos recién extraídos de la web y se tiene una tabla de estas por cada Expected Table a la que se instancio calidad. '*tableName*' corresponde al nombre que se le asignó al *Expected Schema* durante la configuración
propagated_integrated_tableName	Esta tabla contiene los valores de calidad de las Integrated Tables que fueron propagados de las mediciones de calidad sobre las Expected Tables. 'tableName' corresponde al nombre de la Integrated Table.

propagated_dw_tableName	Esta tabla contiene los valores de calidad de las DW Tables que fueron propagados de las mediciones de calidad sobre las Expected Tables. 'tableName' corresponde al nombre de la DW Table.
measured_dw_tableName	Esta tabla almacena las mediciones de calidad sobre el DW resultado, en donde 'tableName' hace referencia al nombre de la tabla del DW a la que se evaluó calidad.
attributes_set	Para contemplar los distintos tipos de granularidad en el modelo, se definió esta tabla**,** que identifica un conjunto de atributos a los que se aplica una métrica de calidad dada**.** En el caso de granularidad Celda, este conjunto de atributos está compuesto por un sólo elemento; mientras que para la granularidad Conjunto de Celdas, se tienen varios atributos en este conjunto. Por otro lado, para el caso de granularidad Tabla se tiene el valor -1 y ninguna asociación con la tabla **instantiated_attributes**.
instantiated_attributes	Esta tabla contiene el nombre de los atributos de los conjuntos definidos en la tabla **attributes_set**. Estos nombres se corresponden a nombres de atributos de las Expected, Integrated y DW Tables.

En las tablas en las que se registran valores de calidad, se identifica una entidad a través de las primary keys definidas en la configuración. Por ejemplo, si se tiene una Expected Table **Personas** (nombre, apellido, telefono), donde los atributos nombre y apellido conforman la primary key de la tabla, su tabla de calidad asociada será **measured_expected_personas** (id, metric_id, attributes_set_id, nombre, apellido, value)

En caso que el valor de calidad corresponda a una métrica de granularidad 'Tabla', los atributos correspondientes a la primary key de la tupla que contiene el valor de calidad son nulos. En el ejemplo anterior quedaría la tupla: (id, metric_id, -1, null, null, value, false)

Los atributos ***metric_id***,***attributes_set_id***y también los atributos que conforman la PK de la Expected, Integrated o DW Table identifican una medición de calidad. Sin embargo, para un valor de calidad de granularidad tabla los atributos PKs son vacíos, por lo tanto, como no es posible definir una PK que tenga valores nulos, se optó por incluir un id autogenerado como PK de una medición de calidad.

5 - Implementación del Prototipo

En este capítulo se presenta la implementación del prototipo que se realizó como prueba de conceptos de la solución propuesta.

Como se mencionó en el capítulo anterior, se implementaron dos procesos BPM (Configuration.bpmn y Feeding.bpmn) que resuelven la Fase de Configuración y la Fase de Carga del Proceso de Construcción de un WW respectivamente. La plataforma de Construcción de WW se implementó en Java utilizando como motor de procesos a Activiti[6] que implementa el estándar *Business Process Model And Notation* (BPMN 2.0). Mientras que como interfaz de usuario se empleó la herramienta provista por Activiti denominada Activiti Explorer, la cual es una interfaz web.

5.1 Especificaciones técnicas

El prototipo implementado como un proyecto Java cuenta con todas las funcionalidades desarrolladas para la correcta ejecución de los procesos de Configuración y de Carga de un WW. El proyecto una vez compilado se utiliza como una biblioteca, un único archivo *jar*, que puede ser agregado al despliegue web de la herramienta Activiti. Como servidor de aplicación se utilizó Apache Tomcat y como motor de Base de Datos MySQL[18].

La biblioteca cuenta con la responsabilidad de manejar conexiones a diversas bases de datos, ejecutar las actividades asociadas a los procesos, y también poder consumir de forma transparente para el usuario servicios externos a la aplicación.

El proyecto Java se encuentra dividido en dos principales módulos como se puede apreciar en la Figura 37: un módulo que contiene el diseño de los procesos, llamado *resources*, y otro que contiene todo el código fuente, llamado *java*.

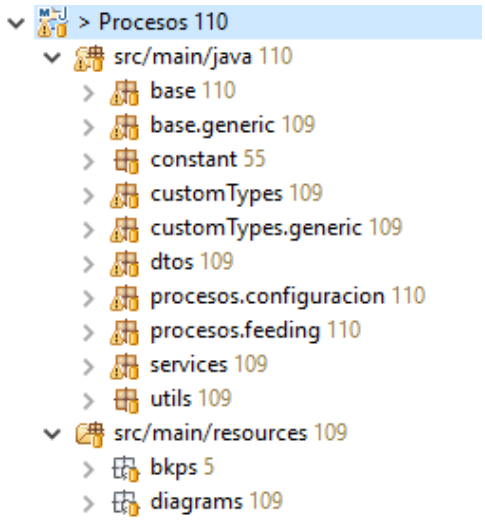


Figura 37 – Paquetes Proyecto Java

En particular para el diseño de los procesos se utiliza el *plugin* provisto por Activiti para el IDE Eclipse (*Activiti Designer*) para realizar dichos diseños de manera gráfica. El complemento ofrece una gran cantidad de funcionalidades y permite que el diseño sea más intuitivo. A su vez, mediante el mismo se pueden agregar componentes básicos a los formularios de las tareas y definir, en caso de ser necesarios, eventos sobre las actividades. En la Figura 38 se muestran algunas de las opciones ofrecidas.

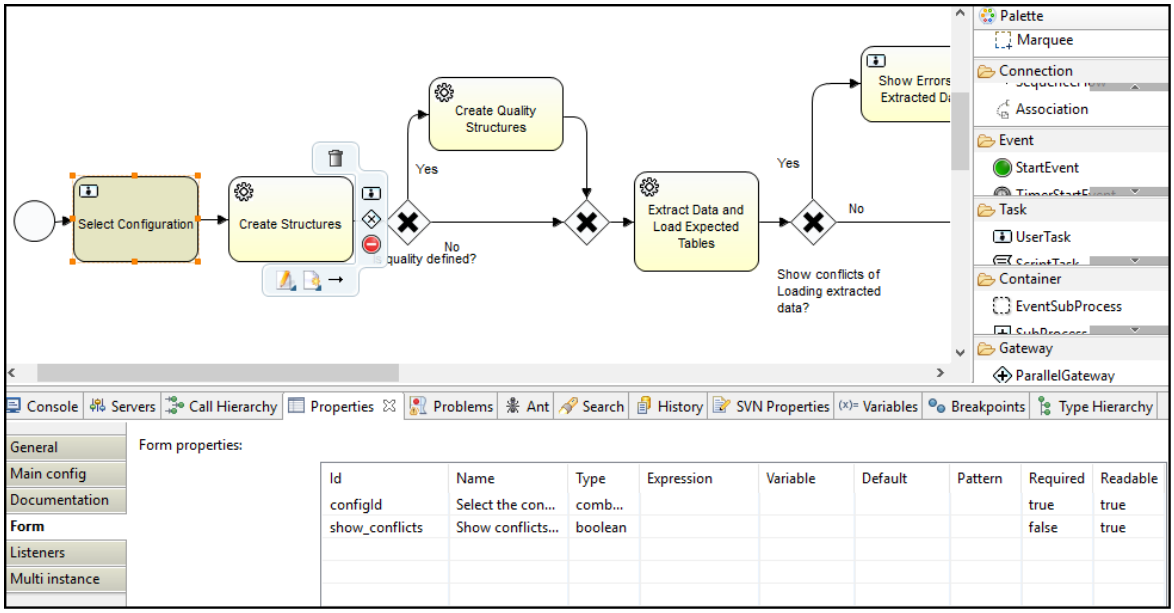


Figura 38 – Plugin Activiti Designer

Respecto al modelado de los procesos, se siguieron las guías de modelado especificadas en[17]. Un ejemplo de ellas es el empleo de verbos en infinitivo para el nombre de las tareas (‘*Select Expected Schema*’ en lugar de ‘*Selecting Expected Schema*’). También se aplicaron varios *workflow patterns*[19].

Por otro lado se tiene el módulo asociado al código fuente denominado *java.* Dicho módulo se organiza de tal forma que cada paquete que lo constituye está asociado a funcionalidades determinadas. Los paquetes principales definidos son: *base*, *constant*, *customTypes*, *dtos*, *procesos*, *services* y *utils*. La estructura de los paquetes es mantenida del proyecto anterior, pero gran parte de la estructuración interna de las clases cambia debido a las mejoras aplicadas. A continuación se detalla cada uno de los paquetes que componen este módulo.

Paquete Base

El paquete base, como su nombre lo indica, se encuentra asociado a todas las interacciones con las distintas bases de datos. El mismo se compone por una clase llamada *ControladorBD* la cual actúa como fachada para las diversas solicitudes del sistema, y luego por clases asociadas a pedidos o consultas específicas según el carácter del requerimiento. Por lo tanto desde cualquier punto del sistema, es decir desde donde sea necesario, se puede acceder a la fachada la cual redirigirá acordemente a la clase correspondiente para

que el llamado sea procesado. La subdivisión de clases establecida es la siguiente: WebSourceQueries, ExpectedSchemaQueries, IntegratedSchemaQueries, DataWarehouseQueries, QualityQueries y FeedingQueries, cada una de las mismas se encarga, según su nombre lo indica, del tipo de consultas que corresponda. La Figura 39 ilustra un diagrama de clases, con el mismo se busca clarificar la comunicación entre las distintas clases con la base de datos. En la figura, 'Event Class' representa una clase asociada a un evento de una actividad, es decir que implementa JavaDelegate o TastkListener. Mientras que 'Custom Type Class' representa clases asociadas a tipos visuales personalizados como por ejemplo combobox, tablas, entre otros.

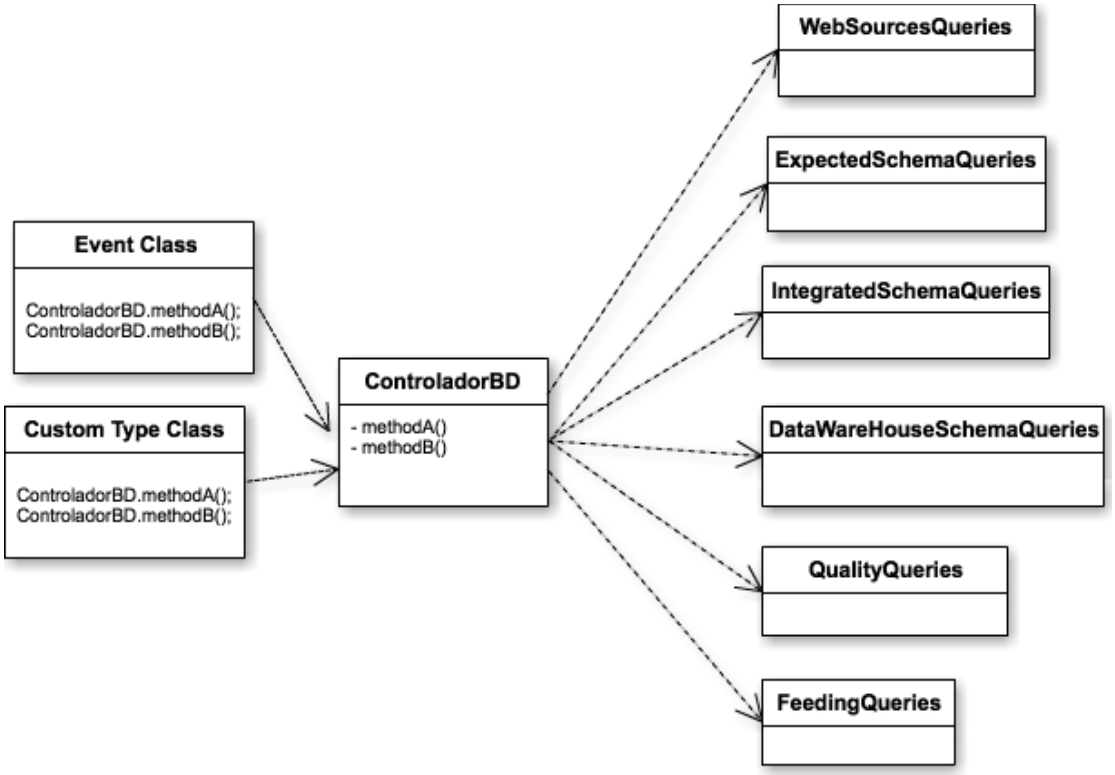


Figura 39 - Diagrama de Clases para manejar la comunicación con las bases de datos

Cabe destacar, que se tomó la decisión de independizar las consultas SQL del código. De esta forma si se quisiera utilizar otro motor de base de datos, solamente se deben, en caso de ser necesario, alterar las consultas. A continuación se puede observar en un código de ejemplo.

```
private final String SQL_GET_SCHEMA = "SELECT id, name FROM
integrated_schema WHERE config_id = ? ";

//Ejemplo de uso
public List<Table>getIntegrateSchemaAtts(Connection conn, int config) {
   PreparedStatement st = st =conn.prepareStatement(SQL_GET_SCHEMA);
   st.setInt(1, config);
   ...
}
```

Es oportuno mencionar que este paquete también contiene clases genéricas con el fin de reducir el código al momento de implementar sentencias para ser ejecutadas en la base de datos.

Otro aspecto importante que tiene este paquete es la clase llamada SQLConstructor, la cual se encarga de generar dinámicamente sentencias SQL para luego ser ejecutadas en la base de datos que corresponda. Dicha clase toma un papel crucial durante la ejecución del proceso de carga, ya que usando los datos provistos en la configuración se construyen las sentencias necesarias. En el siguiente ejemplo se muestra cómo se construye dinámicamente una sentencia SQL para insertar en la base de datos la información provista en los parámetros.

```
private static final String sql_insert_table = "INSERT INTO &1 ( &2 )
VALUES ( &3 );";

public static String insertDatabase(String tableName, List<String>
nombreColumnas, List<Valor> valoresFijos){
  String sqlInsertTable = sql_insert_table.replace("&1", tableName);
  if (valoresFijos!=null){
    for(int i=0; i<valoresFijos.size(); i++){
      sqlInsertTable = sqlInsertTable.
       replace("&2", valoresFijos.get(i).getNombre() + "&2").
       replace("&3", "'" + valoresFijos.get(i).getValor() + "'&3").
       replace("&2", ",&2").replace("&3", ",&3");
    }
  }
  for(int i=0; i<nombreColumnas.size(); i++){
    sqlInsertTable = sqlInsertTable.
      replace("&2", nombreColumnas.get(i) + "&2").
      replace("&3", "?&3");

      if (i != nombreColumnas.size()-1){
        sqlInsertTable = sqlInsertTable.
          replace("&2", ",&2").replace("&3", ",&3");
      }
  }
  return sqlInsertTable.replace("&2", "").replace("&3", "");
}
```

También se cuenta con la clase DebuggableStatement, la cual encapsula la manipulación con la base de datos cuando se pretende ejecutar varias sentencias a la vez (modo batch). Lo fundamental de ésta clase es que permite con una sola instancia mantener varias sentencias controlando los posibles errores y/o excepciones a nivel de sentencia, es decir se mantiene un registro por cada uno de los posibles errores. En el siguiente ejemplo se muestra cómo se maneja la inserción en la base de datos del resultado de una consulta SQL pasada como parámetro, en la cual se devuelve una lista con aquellas tuplas que no se pudieron insertar indicando la causa de error.

```
public List<Tupla>insertSQLQuery(String sqlQuery, ...) {
   PreparedStatement st = null;
   DebuggableStatement stBatch = null;

   List<Tupla> errors = new ArrayList<Tupla>();
   try {
      //Se ejecuta la consulta pasada por parámetro
      st = conn.prepareStatement(sqlQuery);
      result = st.executeQuery();

      conn.setAutoCommit(false);
      stBatch = new DebuggableStatement(conn, tableName,
                    SQLConstructor.insertTuple(tableName, attributes) );

      while (result.next()) {
         tuple = ...;
         stBatch.addBatch(tuple);

         count++;
         if (count > 25000 || result.isLast()){
            errors.addAll( stBatch.executeBatchWithExceptionHandler() );
            count = 0;
         }
      }
      conn.commit();

      } catch (SQLException e) {
      } finally {
        /*Se cierra conexión*/
      }
      return errors;
}
```

Paquete Constant

Por otro lado se tiene el paquete *constant* que contiene únicamente la clase llamada *FlowConstant*, la cual se encarga de almacenar la gran mayoría de las variables usadas por los flujos de los procesos y subprocesos

Paquete CustomTypes

El paquete *customTypes* contiene todas las clases asociadas a tipos visuales personalizados. Mediante *Activiti Desginer* es posible agregar a los distintos formularios tipo visuales básicos, como lo puede ser un input o un checkbox, pero si se pretende agregar un nuevo tipo más complejo es necesario programarlo.

El componente gráfico que Activiti utiliza para dibujar la interfaz se llama *Vaadin*[20], el cual es bastante completo en cuanto a los diversos tipos que se pueden construir, como los son tablas, combo boxes, listados, text area, entre otros. A su vez, agregando componentes de forma manual, permite añadir dinamismo a los formularios asociando, por ejemplo, eventos sobre los mismos.

Para agregar un nuevo tipo es necesario definir dos clases asociadas, una para definir propiamente al tipo, incluyendo el nombre mediante el cual podrá ser referenciado en los formularios de los procesos, y otra clase para especificar el tipo del mismo, es decir con qué componente visual se construirá. Por más detalles ver anexo B.1 (Manual del usuario).

Además se tiene el sub-paquete *generic*, el cual contiene clases utilizadas para la construcción de tipos genéricos. De esta forma se tiene, por ejemplo, el tipo genérico para una tabla, y luego se puede definir distintos *customTypes* que hagan uso de dicho tipo para construir una tabla personalizada.

Paquete Datos

En el paquete *dtos* (*Data Transfer Objects*) se tienen todas las clases asociadas a los tipos de datos utilizadas tanto para el manejo de objetos provenientes de las bases de datos, así como también de objetos utilizados por los pedidos y/o respuestas al momento de invocar los diversos servicios web.

Paquete Procesos

Una reestructuración aplicada sobre el proyecto, es la subdivisión de clases asociadas por un lado, a actividades del Proceso de Configuración, cuyo paquete asociado se llama '*configuracion'* y por el otro, a actividades del Proceso de Carga, cuyo paquete asociado se llama '*feeding'*. De esta manera se puede identificar de forma más sencilla clases utilizadas en un proceso o en el otro.

Para dichas clases se aplica una nomenclatura específica según el tipo de actividad, es decir si es manual, y también el evento sobre el cual se encuentra asociada, o si es de tipo servicio, es decir de carácter automática. La nomenclatura definida consiste en concatenar el nombre de la actividad conjuntamente con el tipo de evento que corresponda. A modo de ejemplo, si para la actividad 'Select Dimension' se tiene una clase asociada al evento create, el nombre de dicha clase es SelectDimensionCreate. Las clases asociadas a eventos de actividades de tipo manual implementan la clase TaskListener. La Figura 40 ilustra este ejemplo, en donde se aprecia el momento en el que el motor Activiti ejecuta los eventos asociados a la tarea 'Select Dimension'.

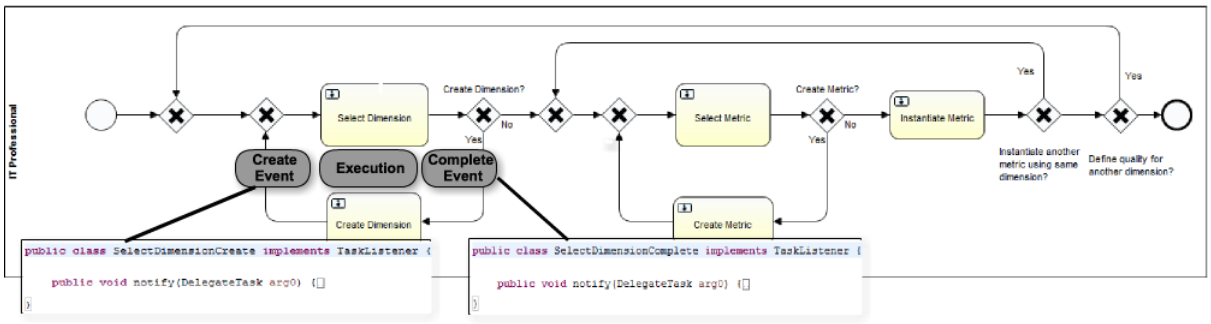


Figura 40 - Ejemplo de actividad manual

Por otro lado para el caso de actividades de tipo servicio, la nomenclatura definida consiste en concatenar el nombre de dicha actividad con la palabra 'Automatic'. Por ejemplo si se tiene la actividad de tipo servicio "*Create Structures*", la clase asociada se denomina *CreateStructuresAutomatic* (ver Figura 41) las clases asociadas a dichas actividades de tipo servicio implementan la clase *JavaDelegate*.

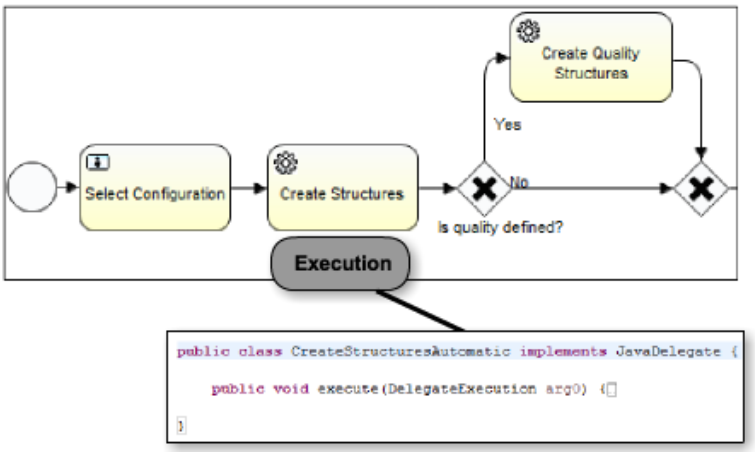


Figura 41 - Ejemplo de actividad automática

Otro aspecto que se mejoró es la unificación de código común, buscando de esta forma que aquellas actividades que tengan eventos asociados con tareas iguales o muy similares utilicen la misma clase, y por lo tanto evitando duplicación de código y de manera de volver el prototipo más tolerable a cambios. Para estos casos particulares es posible utilizar, en caso de ser necesario, una variable del proceso que distingue la actividad desde la cual se invoca la clase. Por ejemplo, se tiene la tarea *´Select Dimension´* del subproceso *´DefineDQMForExtractedData*' (ver Figura 25) y la tarea *Select Dimension´* en el subproceso *´DefineDQMForDWData´* (ver Figura 27); en ambas se selecciona el par dimensión-factor para definir el modelo de calidad de los *Expected Schemas* o DW Tables según corresponda, con la particularidad que en el subproceso *´DefineDQMForDWData´* se debe indicar si la tabla del DW es de hechos o no. Por lo tanto, ambas tareas invocan a la misma clase '*SelectDimensionCreate*' en el evento ´create´, la cual contiene el siguiente código encargado de indicar si es una tabla de hechos sólo cuando la clase se invoca desde el subproceso *´DefineDQMForDWData´*:

```
if(taskEntity.getProcessDefinitionId().startsWith("DefineDQMForDW")){
      // Código del formulario específico para DefineDQMForDW
}
```

Paquete Services

El paquete services contiene todo lo referido a la capa de servicios. Con dicha capa lo que se busca es centralizar la utilización de los servicios web ya sea tanto para la extracción de datos, medición de calidad, resolución de entidades, entre otros. Los servicios web a invocar son los definidos durante el proceso de Configuración, esta capa se encarga de que todo el trabajo de consumir dichos servicios y procesar las respuestas sea transparente para quien los invoca. En el Anexo D (Servicios Web) se especifica con más detalle la implementación de la capa de servicios.

Paquete Utils

Finalmente el paquete *utils* contiene aquellas clases de utilidad para la implementación. Se incluyen clases para la interpretación de objetos JSON y su respectiva transformación a un tipo de dato que sea manejable por la aplicación. También se incluye una clase que permite leer desde un archivo de tipo *properties,* variables como lo pueden ser relativas a la conexión a una base de datos. De esta forma una vez compilado el proyecto, se pueden obtener datos durante la ejecución del mismo.

5.2 Implementación Proceso de Configuración.

En este punto se detalla la implementación de las actividades y subprocesos que se incluyeron en el Proceso de Configuración existente. También se hace referencia a las mejoras y correcciones sobre el proceso así como las mejoras sobre la interfaz web del sistema que se aplicaron.

En la Figura 42 se muestran las actividades asociadas a la definición de la calidad de los datos recién extraídos de las FDWs y a los servicios de *Entity Integration*.

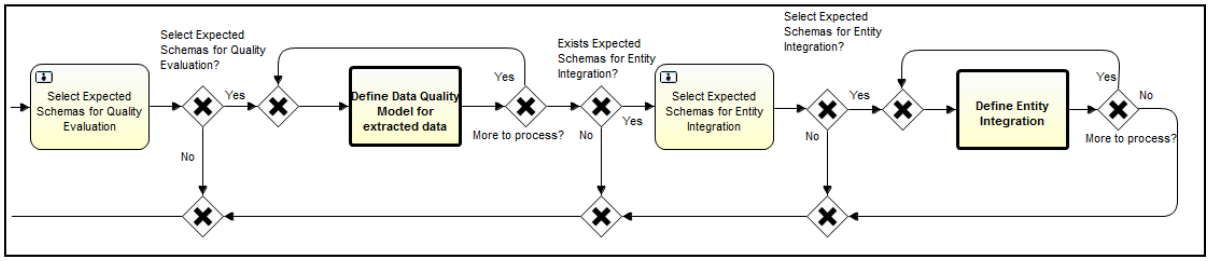


Figura 42 - Proceso de Configuración (Definición de calidad y de Entity Integration)

5.2.1 Aspectos de Calidad del Proceso

En este punto se desarrolla el funcionamiento e implementación de cada una de las actividades y sub-procesos que agregan los aspectos de calidad en el proceso.

El Proceso de Configuración extendido se muestra en la Figura 21(Capítulo 4 - Diseño de la Solución). Como se menciona en dicho capítulo, una vez definidos los *Expected Schemas*, se procede a definir las métricas de calidad a aplicar sobre ellos. A continuación se detallan las tareas asociadas a la definición de calidad de los datos recién extraídos de la web.

Select Expected Schemas For Quality Evaluation

En esta tarea se le presenta al usuario la posibilidad de seleccionar los *Expected Schemas* a los cuales desea aplicar métricas de Calidad (ver Figura 42). El usuario puede optar por no seleccionar ningún *Expected Schema*, y en ese caso, el flujo del proceso procede a la definición de los *Integrated Schemas*, ya que al no definir métricas de calidad sobre ningún *Expected* no tiene sentido definir servicios de integración que empleen metadatos de calidad para la resolución de entidades.

Subproceso Define Data Quality Model For Extracted Data

Luego que el usuario ha seleccionado los *Expected Schemas*, debe definir qué métricas de calidad aplicar a cada uno de ellos.A continuación se detalla el subproceso '*Define Data Quality Model For Extracted Data*' contenido en el Proceso de Configuración e ilustrado en la Figura 43, en donde se itera por cada uno de los *Expected Schemas* seleccionados para definir su modelo de calidad.

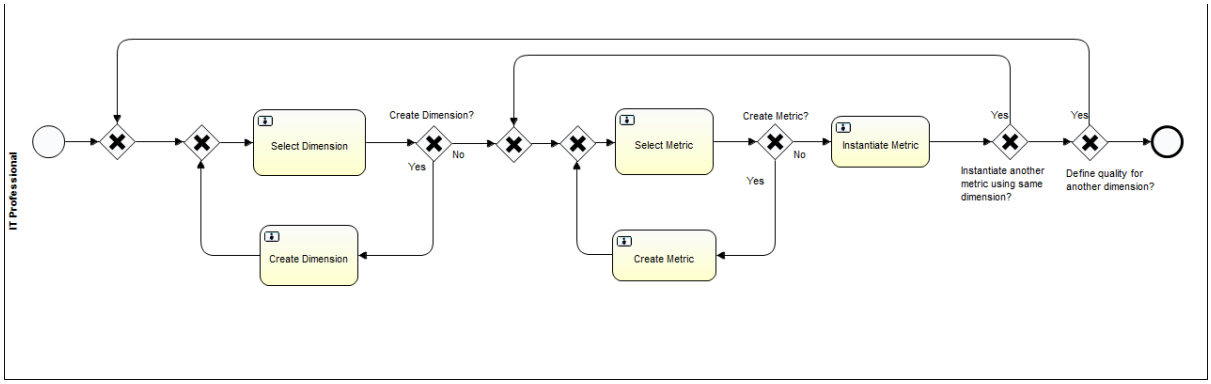


Figura 43 - Subproceso "Define Data Quality Model For Extracted Data"

Select Dimension

En la primera tarea del subproceso el usuario tiene la posibilidad de seleccionar un par dimensión - factor existente (ya precargado en la configuración) o agregar uno nuevo. Por defecto se encuentra seleccionada la opción de escoger un par dimensión- factor existente ya que las dimensiones de calidad más utilizadas vienen precargadas en la Configuración.

Create Dimension

La tarea "*Create Dimension"* se ejecuta sólo en caso que el usuario ha seleccionado en la tarea previa la opción de crear un nuevo par dimensión-factor (opción "*Create new dimension*"). En esta tarea el usuario ingresa el nombre de una nueva dimensión y un factor asociado a esa dimensión. El par dimensión-factor ingresado por el usuario luego podrá ser utilizado en una futura configuración para cualquier dominio.

Al terminar la tarea, el flujo del proceso vuelve a la tarea "*Select Dimension*", en donde el usuario tiene la posibilidad de seleccionar la dimensión recientemente agregada, seleccionar otra ya existente, o en su defecto, continuar agregando dimensiones.

Select Metric

La tarea "*Select Metric*" es análoga a la tarea "*Select Dimension*", pero en este caso el usuario selecciona entre una métrica ya existente para el par dimensión-factor elegido en la tarea previa o ingresar una nueva métrica para dicho par. Al igual que en la tarea "*Select Dimension*" por defecto está seleccionada la opción de elegir una métrica ya existente.

Create Metric

En caso que el usuario seleccione en la tarea "*Select Metric"* la opción de crear una métrica, se ejecuta esta tarea, en la cual se define una métrica de calidad para el par dimensión-factor seleccionado antes. En esta tarea el usuario debe ingresar un nombre para la métrica, una breve descripción (opcional), el servicio web que calcula la métrica y la

granularidad de la misma. Como se mencionó anteriormente, los distintos tipos de granularidad posibles son “Celda”, “Conjunto de Celdas” y “Tabla”.

Luego de confirmar la tarea, el flujo del proceso vuelve a la tarea “*Select Metric*”.

Instantiate Metric
En esta tarea el usuario selecciona cuáles son los atributos a los cuales va a aplicar la métrica de calidad que seleccionó en la tarea anterior.

La pantalla de esta tarea depende de la granularidad de la métrica previamente seleccionada. Si la granularidad es ‘*Celda*’ se permite seleccionar un atributo de los atributos del *Expected Schema*, si la granularidad es ‘*Conjunto de Celdas*’, un conjunto de atributos. Mientras que en el caso de granularidad *‘Tabla’*, se muestran todos los atributos del *Expected Schema* pero sin posibilidad de seleccionarlos.

Además en esta tarea el usuario puede decidir que quiere hacer a continuación, teniendo dos opciones:

1. instanciar otra métrica para el par dimensión-factor que se seleccionó anteriormente.
2. seleccionar otro par dimensión-factor

En caso de escoger la opción 1, el flujo del proceso continúa en la tarea “*Select Metric*”.
En caso de escoger la opción 2, el flujo del proceso continúa en la tarea “*Select Dimension*”.
Si el usuario no selecciona ninguna de las opciones, finaliza el proceso de definir métricas para el *Expected Schema* actual.

Hasta este momento se explicaron las actividades para definir calidad sobre los datos recién extraídos de la web. Para el caso de la calidad a instanciar sobre el DW resultado, al igual que para los *Expected Schemas*, primero el usuario selecciona a qué tablas quiere aplicar calidad (tarea ‘*Select DW Tables For Quality Evaluation’*) y luego instancia las métricas que desea por cada una de estas tablas. El proceso es análogo aunque se especifica al usuario si la tabla a la que se está instanciando calidad es una tabla de hechos o no. Además, si la granularidad es a nivel de Celda o Conjunto de Celda, en caso que la tabla sea de hechos, se indica cuáles atributos son medidas del DW y cuáles no.

5.2.2 Mejora para la Integración de datos en la carga

A continuación se detalla el funcionamiento e implementación de cada una de las actividades y sub-procesos que se añadieron al proceso de Configuración para mejorar la integración de los datos que se realiza en el proceso de carga en base a los metadatos de calidad.

Select Expected Schemas For Quality Entity Integration
En esta tarea el usuario debe seleccionar a qué *Expected Schemas* desea aplicar *Entity Integration*. Se listan sólo aquellos *Expected Schemas* con más de una FDW asociada y que tienen instanciadas métricas de calidad. En caso que el usuario no seleccione ningún *Expected Schema*, el flujo del proceso sigue con la definición de los *Integrated Schemas*.

Subproceso Define Entity Integration

Luego que el usuario seleccionó los *Expected Schemas* a aplicar *Entity Integration*, se procede a definir los servicios de *Entity Resolution* y *Entity Fusion*.A continuación se detalla el subproceso '*Define Entity Integration' (*ver Figura 44) contenido en el Proceso de Configuración, que itera por cada uno de los *Expected Schema* seleccionados en la tarea anterior.

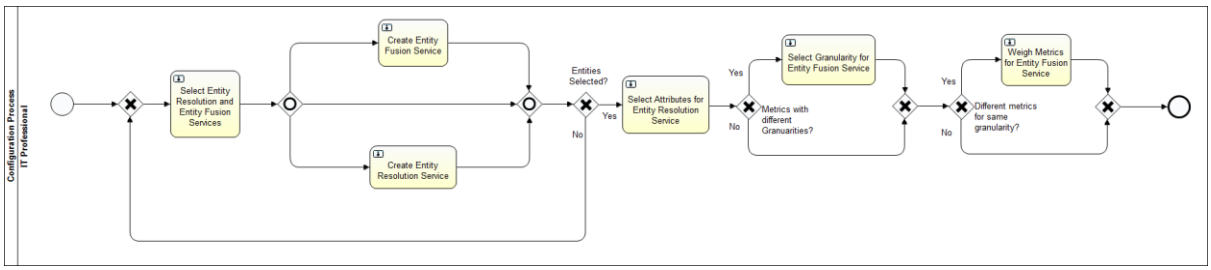


Figura 44 - Subproceso 'Define Entity Integration'

Select Entity Resolution and Entity Fusion Services

En esta tarea, el usuario tiene la posibilidad de seleccionar un servicio *Entity Resolution* y un servicio *Entity Fusion* (ya existentes), crear uno de los dos servicios, o crear ambos. En caso que no existan servicios *Entity Integration* (*Entity Resolution* y *Entity Fusion*), la única opción habilitada es la de crear ambos servicios.

Create Entity Resolution Services

Si el usuario selecciona en la tarea "*Select Entity Resolution and Entity Fusion Services*", la opción de crear un nuevo servicio *Entity Resolution* o la opción de crear ambos servicios, se ejecuta esta tarea. En la misma el usuario ingresa un nombre, una breve descripción (opcional) y el servicio web al cual debe invocar para realizar la resolución de entidades duplicadas.

Luego de confirmar la tarea, el flujo del proceso vuelve a la tarea "*Select Entity Resolution and Entity Fusion Services*", pero esta vez sólo se le permite seleccionar los servicios; las opciones de crear nuevos servicios están deshabilitadas. Esta particularidad distinta a la implementación de la creación de dimensiones y métricas se debe a que en la definición de calidad se puede instanciar más de una métrica sobre los *Expected Schemas* mientras que en la definición de servicios de integración se asocia un único servicio *Entity Resolution* y un único servicio *Entity Fusion* por cada *Expected*. Por lo tanto, para hacer más intuitivo el proceso para el usuario se permite que sólo pueda crear un servicio *Entity Resolution* y/o un servicio *Entity Fusion*.

Create Entity Fusion Service

En caso que el usuario seleccione en la tarea "*Select Entity Resolution and Entity Fusion Services*", la opción de crear un nuevo servicio *Entity Fusion* o de crear ambos servicios, se ejecuta esta tarea. En la misma el usuario ingresa un nombre, una breve descripción (opcional) y el servicio a invocar para realizar la fusión de entidades duplicadas. Luego de confirmar la tarea, el flujo del proceso vuelve a la tarea "*Select Entity Resolution and Entity Fusion Services*" pero las opciones de crear nuevos servicios están deshabilitadas por lo explicado anteriormente.

Select Attributes for Entity Resolution Service
En esta tarea el usuario determina de qué manera identificará duplicados en el *Expected Schema*. El usuario puede elegir entre utilizar su *primary key* o seleccionar aquellos atributos que considere.

Select Granularity For Entity Fusion Service
Esta tarea se presenta al usuario en caso que el *Expected Schema* actual (al que se le está definiendo los servicios *Entity Integration*) tenga asociado métricas de distinta granularidad. El usuario debe seleccionar qué granularidad desea considerar al momento de aplicar *Entity Integration* ya que el servicio *Entity Resolution*, como se mencionó antes (Capítulo 4 - Diseño de la Solución), sólo maneja métricas de la misma granularidad.

Weigh Metric For Entity Fusion Service
En esta pantalla se solicita al usuario que pondere con un valor entre 0 y 1 cada una de las métricas que tenga definidas para el *Expected Schema* actual. Las métricas que se despliegan al usuario son todas las métricas definidas para el *Expected* (caso en el que todas las métricas sean de la misma granularidad) o aquellas métricas de granularidad igual a la seleccionada en la pantalla anterior (caso que el *Expected Schema* actual tiene métricas de distinta granularidad). En caso que sólo hubiera una métrica a desplegar, esta tarea no se ejecuta ya que el único valor de ponderación posible es 1.

Uno de los controles que se realiza en esta tarea es que la suma de las ponderaciones de todas las métricas sea igual a 1, en caso que no lo sea, se despliega un mensaje de alerta al usuario.

5.2.3 Mejoras y Correcciones sobre el proceso existente

En esta sección se detallan las mejoras y correcciones realizadas sobre el Proceso de Configuración del proyecto anterior. Las mejoras sobre la interfaz del prototipo se detallan en el punto 5.2.4. Cabe destacar que estas mejoras implicaron modificaciones en el flujo de los subprocesos existentes.

Mejora 1
Dado que uno de los requerimientos del proyecto es incluir la calidad en el proceso, se debió optar por una manera de identificar los datos recién extraídos de la web para lograr propagar los valores de calidad de los mismos durante el Proceso de Carga. Por lo tanto se tomó la decisión que al momento de definir el *Expected Schema* se le solicite al usuario que indique una *primary key* para este esquema.

Esta mejora no sólo ayuda a mantener una traza de los datos sino que también facilita la definición de servicios *Entity Resolution*, ya que el usuario puede optar por identificar por defecto los duplicados a través de la *primary key*.

Mejora 2
En el mapeo entre un atributo de un *Expected Schema* y un atributo de una FDW, los atributos de la FDW no se mostraban en orden en el prototipo anterior. Se ordenaron dichos atributos para facilitar los mapeos al usuario.

Mejora 3
En el proyecto anterior una configuración se identifica sólo por un identificador autogenerado. Por lo tanto en el Proceso de Carga se selecciona la configuración a cargar seleccionado su número de configuración, lo cual resulta poco amigable para el usuario. Además, el nombre de la Base de Datos resultado del proceso de Carga, es la concatenación de la palabra "base" y dicho identificador.

Por esta razón, se decidió que el usuario comience el Proceso de Configuración ingresando un nombre para la misma, y en el Proceso de Carga, seleccione la configuración por el mismo nombre con el que la definió. También, este nombre es el que se le asigna a la Base de Datos Intermedia creada en el Proceso de Carga.

Mejora 4
En el proyecto anterior las FDW están asociadas a un dominio específico, es decir si el usuario ingresa una FDW al sistema, esta sólo puede ser utilizada en configuraciones futuras del mismo dominio. Lo mismo no ocurre para el caso de los *Expected Schemas*. Por ejemplo, si un usuario define en el dominio '*Salud*' un *Expected Schema* denominado 'enfermedades' asociado a una FDW que contenga enfermedades, este *Expected* estará disponible para seleccionar en cualquier otro dominio como es el dominio '*Turismo*'.
Por esta razón, se decide que los *Expected Schemas* también estén asociados al dominio, es decir sólo puedan ser utilizados en el dominio que los definió.

Mejora 5
Se agregó en la definición de los esquemas (*Expected*, *Integrated* y DW), que en el primer atributo el *checkbox* que indica si el atributo es *primary key* o no, esté seleccionado por defecto y el usuario no tenga posibilidad de modificar dicha selección. De este modo, se controla que todos los esquemas definidos durante la configuración tengan *primary key*.

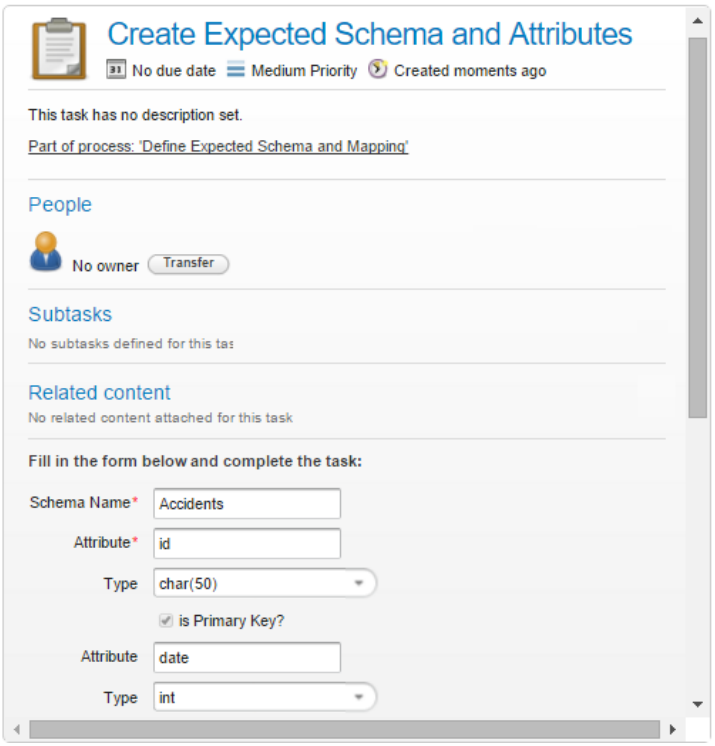


Figura 45 - Tarea '*Create Expected Schema and Attributes*'

Mejora 6

Otra restricción que se incluye en relación a las *primary keys* de los esquemas, es que en los formularios de mapeos, el usuario esté obligado a realizar los mapeos de las *primary keys*; de lo contrario, se despliega un mensaje de error.

En la Figura 45, se puede apreciar lo detallado en las Mejoras 5 y 6.

Corrección 1

El prototipo anterior no permite ingresar una FDW con el mismo nombre que una FDW que fue ingresada en otro dominio, ya que el nombre de la FDW es la clave de la tabla ***web_sources*** en la Base de Metadatos de Configuración (Ver Figura 13 de 3.1). Para solucionar esto, se agregó un identificador a la tabla ***web_sources***, por lo tanto, por ejemplo, actualmente es posible ingresar una FDW denominada 'países' en el dominio '*Salud'*, si ya fue ingresada otra FDW con el mismo nombre en el dominio '*Turismo*'.

Corrección 2

Algunas tablas de la Base de Metadatos de Configuración del proyecto anterior contienen atributos, como por ejemplo **ya_joineado** o **ya_cargado**, que se utilizan meramente como *flags* para dirigir el flujo del Proceso de Configuración. Dado que el fin de la base de Configuración es almacenar todas las definiciones que realiza el usuario, se eliminaron dichos atributos ya que estos no son definidos ni utilizados durante la carga. Por lo que se debió corregir el flujo del proceso a través de código *Java* haciendo uso de variables asignadas a una instancia de un proceso.

5.2.4 Mejoras en la interfaz

En esta sección se detallan las mejoras sobre la interfaz del prototipo sobre el Proceso de Configuración del proyecto anterior.

Mejora Interfaz 1
Durante la configuración, en varias tareas se pone al usuario "en contexto", es decir, se incluye en el formulario información que definió anteriormente, que está asociada a la definición que debe realizar el usuario en la tarea actual.

Por ejemplo, en la Figura 46 se ilustra el formulario de definición de *Expected Schema* para cada FDW, y se muestra en pantalla el nombre de la FDW definida por el usuario en pantallas anteriores. Como se puede apreciar en el formulario izquierdo, el implementado por el proyecto anterior, el nombre de la FDW, no se visualiza correctamente; mientras que en el formulario (b) (implementado en el proyecto actual) se logra modificando el estilo web (CSS).

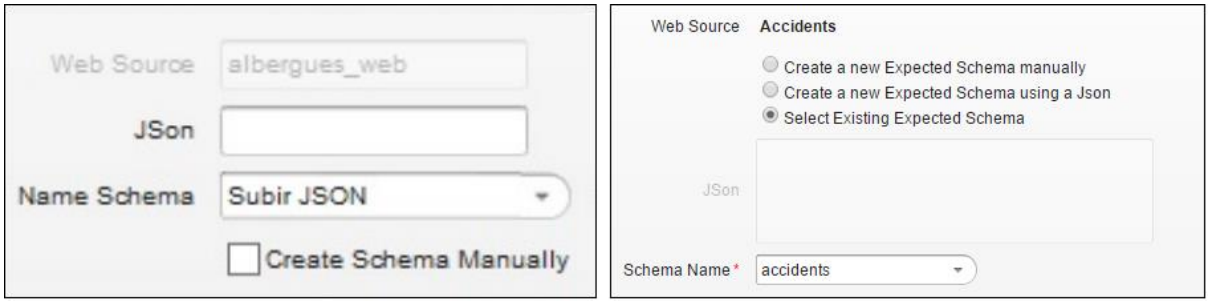


Figura 46 - Formulario de definición de *Expected Schemas*:
(a) proyecto anterior, (b) proyecto actual

Mejora Interfaz 2
Como se aprecia en la Figura 46, se modificó la implementación de los formularios de definición de todos los esquemas para hacerlos más intuitivos para el usuario. A continuación se explica el formulario de definición de *Expected Schemas*. En el formulario (a) de la Figura 46 se tiene un *combo box* con la opción de subir un *Json* y a su vez todos los *Expected Schemas* disponibles para seleccionar. En este proyecto las distintas opciones para subir un *Expected Schema* (crear, seleccionar uno ya existente o subir un *Json*) se encuentran en *radiobuttons* ((b) de la Figura 46) y al seleccionar una opción se habilita la correspondiente y se deshabilitan las demás.

Por ejemplo en el caso de la Figura 46, el usuario tiene marcada por defecto la opción de seleccionar un *Expected Schema* ya existente y el combo '*Schema Name*' habilitado donde se tienen todos los *Expected Schemas* disponibles. Si el usuario desea subir un *Json* selecciona dicha opción, e inmediatamente se deshabilita dicho *combo box* y se habilita el *textArea* para ingresar manualmente el *Json*. Cabe recordar que también el usuario puede subir el *Json* a través de un archivo, que ocurre cuando no se ingresa nada en el *textArea* y se completa la tarea, apareciendo una ventana emergente para seleccionar el archivo que contiene el *Json*.

Por último, si se quiere crear un nuevo *Expected Schema*, el usuario debe seleccionar el *radiobutton* '*Create a new Expected Schema*', que deshabilita automáticamente el *textArea* y el *combo box*, y al completar la tarea, nos dirige a la tarea '*Create Expected Schema*'.

La misma lógica se aplica para la definición de *Integrated Schemas* y DW *Schemas*.

Mejora Interfaz 3

Otra mejora de interfaz es la inclusión del *Custom Type TextArea* para los campos de descripciones y los campos en los que se ingresan los *Json* que definen los esquemas de la configuración. La Figura 47 ilustra el formulario, del lado izquierdo la implementación realizada por el proyecto anterior, y del derecho, la del proyecto actual.

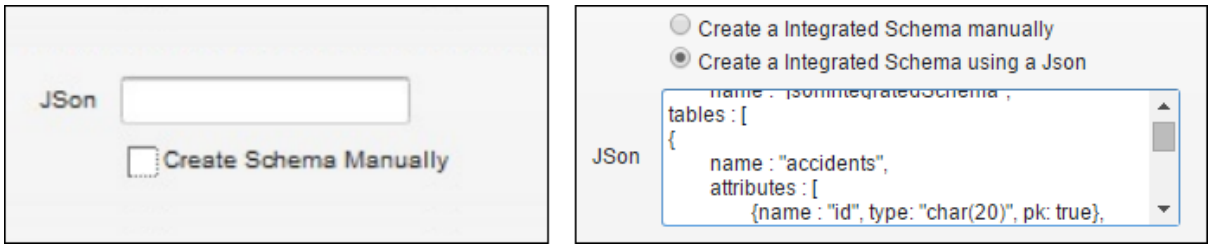


Figura 47 - Formulario para definir *Integrated Schemas*:
(a) proyecto anterior, (b) proyecto actual

Mejora Interfaz 4

En el Proceso de Configuración, una vez definido el *Integrated* o el DW *Schema*, se pasa a una tarea manual en la se muestra al usuario las tablas y los atributos del esquema recién definido. La Figura 48, muestra el formulario de Activiti de la tarea *'Show Integrated Schema'*, arriba el formulario implementado por el proyecto anterior, y abajo, el implementado por el proyecto actual. Como se aprecia en la figura, el esquema del formulario de arriba no se visualiza correctamente; por consiguiente, al igual que en la Mejora Interfaz 3, se modificó el estilo web para mejorar la visualización del mismo.

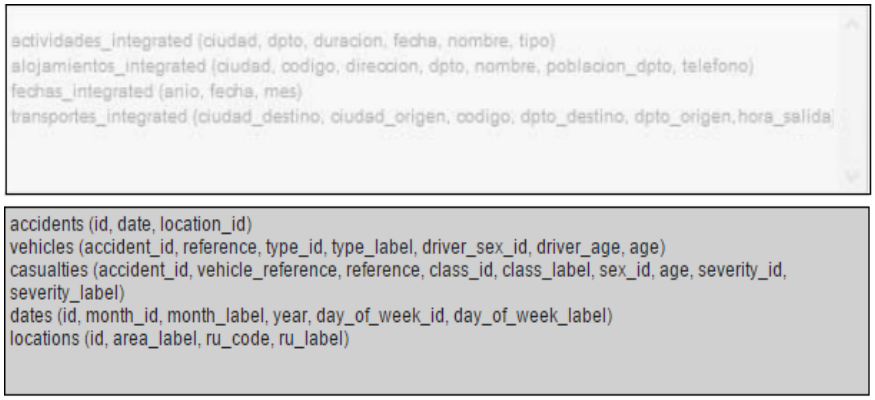


Figura 48 -Formulario de tarea *'Show Integrated Schema'*
(arriba proyecto anterior, abajo proyecto actual)

Mejora Interfaz 5

Se tomó la decisión de mejorar los mapeos realizados en la configuración utilizando tablas. Para ello, se definió un *Custom Type* específico para tablas, el cual se encarga de incluirlas en los formularios de Activiti [6]. La Figura 49 muestra el formulario de mapeo entre atributos del *Expected Schema* y atributos del *Integrated Schema*; del lado izquierdo se muestra la implementación del proyecto anterior, y del lado derecho, la del proyecto actual.

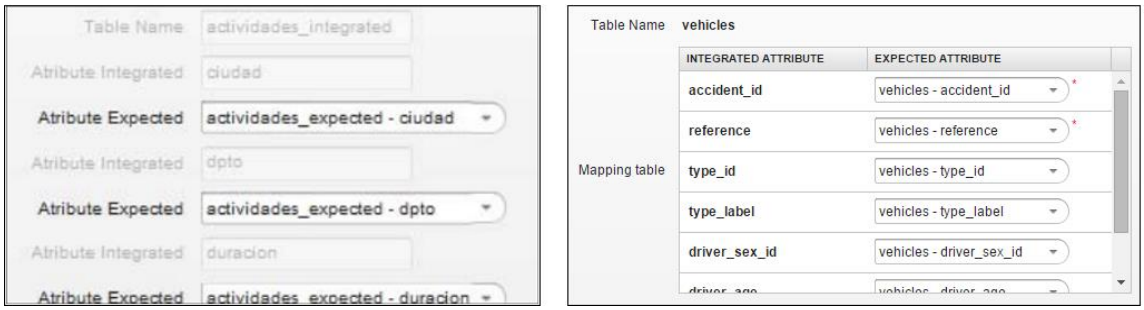


Figura 49 - Formulario de mapeo entre *Expected Schema* e Integrated: (a) proyecto anterior, (b) proyecto actual

5.3 Implementación Proceso de Carga

En este punto se presenta la implementación del Proceso de Carga. El Proceso de Carga implementado cuenta mayoritariamente con actividades automáticas y con alguna actividad manual para resolver conflictos o presentar errores. El detalle del funcionamiento e implementación de cada una de las actividades definidas se dividió en cuatro partes según el procesamiento que se debe realizar.

5.3.1 Selección de configuración y creación de estructuras

La Figura 50 muestra todas las actividades asociadas a esta etapa, las cuales se detallan en esta sección.

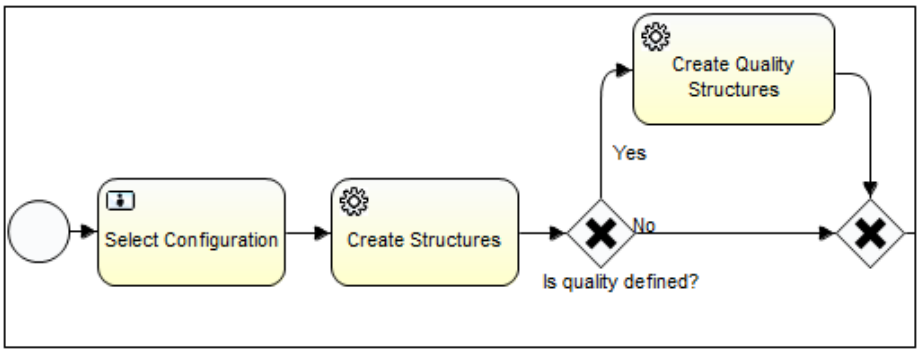


Figura 50 - Proceso de Carga (Parte 1)

Select Configuration
En primera instancia del proceso se realiza la actividad manual correspondiente a la selección de la configuración que se desea cargar, dicha actividad se llama *"Select Configuration"* (ver Figura 50).

El sistema ofrece en un *combo box* una lista con el nombre de todas las configuraciones disponibles en el Sistema, precedidas por un número que se corresponde con su identificador en la Base de Metadatos de Configuración.

También, en esta tarea se cuenta con el *checkbox* '*Show Conflicts and Errors*', que de ser seleccionado se muestran al usuario todos los errores y conflictos durante la carga, y en el caso de la integración de entidades, el usuario tendrá incluso la posibilidad de resolver dichos conflictos. De todos modos, dichos errores y conflictos son registrados en tablas de la Base de Datos Intermedia, por lo que, en caso que el usuario no desee interrumpir la carga puede no seleccionar este *checkbox*.

Create Structures
Una vez seleccionada la Configuración, se invoca a la tarea automática *"Create Structures"* (Ver Figura 50) crea todas las estructuras necesarias para realizar la carga a partir de los datos que se encuentran en la Base de Metadatos de Configuración.

El Sistema crea una base de datos para almacenar el DW. El nombre de esta base está dado por el nombre del DW *Schema* que se definió durante la configuración. Además el sistema crea una base de datos denominada Base de Datos Intermedia que aloja todas las estructuras necesarias para realizar la extracción e integración de los datos. El nombre de esta base está dado por la concatenación del identificador de la configuración y su respectivo nombre. Como se mencionó antes, en caso de ya existir estas bases (ya se realizó una carga para la configuración seleccionada) se vuelven a generar ambas bases.

Create Quality Structures
Esta tarea se ejecuta sólo en caso que el usuario haya definido en la configuración seleccionada calidad para los datos recién extraídos de la web o para el DW resultado.

La tarea se encarga de crear todas las estructuras que almacenarán los metadatos de calidad; tanto los valores de calidad de las mediciones de las métricas como los valores de la calidad propagada. Es importante destacar que el sistema crea una única tabla por cada *Expected Table* o *DWTable* sobre la que se definió calidad y allí se almacenan las mediciones de todas las métricas instanciadas para esa tabla.

Sobre las estructuras que almacenan la calidad propagada, están son creadas en este momento considerando las métricas de calidad instanciadas sobre los *Expected Schemas* y la traza de dichos datos durante todo el proceso de carga.

Es de interés mencionar en este punto que también se crean estructuras específicas para almacenar información relativa a la propagación de la calidad de los datos. Dichas estructuras se guardan en una variable del proceso y conlleva, a priori, una cierta complejidad para su creación, pero la misma facilita la propagación en el momento que se requiere. La estructura consta de mapeos entre las distintas tablas asociadas a un

Integrated Schema y las métricas que deben ser consideradas durante la propagación. De manera análoga se tiene otra estructura con la información asociada para propagar calidad desde *Integrated Tables* a *DW Tables*, para obtener esta segunda estructura que hace uso de la información de propagacióngenerada entre *Expected Tables* e *Integrated Tables*.
Con respecto a la información almacenada de la métricas consiste en mantener una traza de cómo cada atributo se transforma desde su definición en un *Expected Schema* hasta llegar al DW, es decir cómo su nombre va a cambiando durante el proceso. Para determinar si la calidad de datos asociada a una determinada métrica es necesaria propagarse se toman las consideraciones establecidas en la sección "Propagación de la Calidad" del capítulo 4.3, por lo que aquellas métricas que no cumplan con tales condiciones no se guardarán en las estructuras.
Otra utilidad de las estructuras asociadas a la propagación, es poder determinar cuáles tablas relativas a la propagación son creadas en este punto del proceso.

5.3.2 Carga y medición de calidad de datos de Expected Tables e integración de entidades

En este punto se detalla el funcionamiento y la implementación de las actividades de la Figura 51.

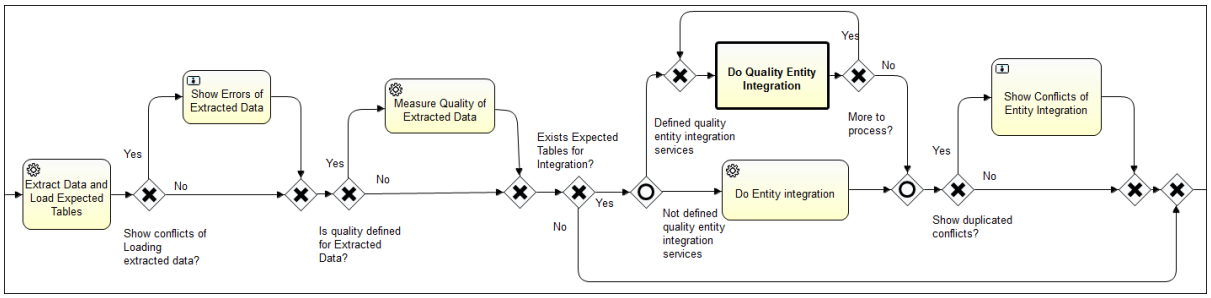


Figura 51 - Proceso de Carga (Parte 2)

Extract Data and Load Expected Tables
Luego de creadas las estructuras, se procede a ejecutar la tarea automática *'Extract Data and Load Expected Tables'* (Ver Figura 51). Dicha tarea se encarga de realizar la extracción de los datos desde la Web, empleando los servicios de extracción definidos en la configuración, y cargándolos en las *Expected Tables*.

Para cada FDW definida en la configuración se invoca el servicio de extracción asociado. Para el manejo de la inserción de los datos extraídos en la Base de Datos Intermedia se utilizan *threads* (hilos de ejecución). Por lo tanto, si se tiene un servicio que entrega los datos por partes, es decir, realiza varios pedidos para completar una extracción, dicho manejo de inserciones se efectúa de manera concurrente. Por más detalles sobre los servicios de extracción, consultar el Anexo C (*Web Services*).

En la extracción y carga de los datos, como se mencionó antes, se pueden presentar inconsistencias o problemas como por ejemplo tipos incompatibles o valores que exceden el

campo definido para el atributo. Todos estos errores se categorizan y registran en una tabla común denominada **log_extract_data**. En esta tabla se almacena la tupla entera que generó el error, el nombre de la *Expected Table* y el mensaje de error del manejador de la base de datos.

Show Conflicts of Extracted Data
Esta tarea se ejecuta en caso que el usuario haya especificado en la primera tarea manual del proceso que desea ver los errores y conflictos del proceso. En la misma se muestran al usuario los errores ocurridos durante la extracción y carga de los datos que son obtenidos de la tabla **log_extract_data.** Los errores se muestran al usuario clasificados por *Expected Table* y luego por categoría.

Con esta tarea se espera que el usuario sea consciente de que los datos que no pudieron ser extraídos y cargados a las FDW, y tanto durante como al final de la carga pueda tomar decisiones al respecto. Se espera que esta tarea sea extendida en un futuro brindando la posibilidad de que el usuario resuelva estos errores directamente en el proceso y los mismos formen parte de la carga del WW.

Measure Quality of Extracted Data
Esta tarea es la encargada de realizar las mediciones de calidad sobre las *Expected Tables* y almacenarlas en la Base de Metadatos de Calidad. La tarea se ejecuta sólo en caso que se haya definido calidad sobre al menos un *Expected Schema*.

Se itera por cada *Expected Table* a la que se le definió calidad y por cada métrica de calidad instanciada, en la misma se invoca el servicio de calidad que realiza las mediciones y se almacenan los valores de calidad en la base correspondiente. Al igual que en la tarea '*Extract Data and Load Expected Tables*' se utiliza el concepto de *threads* para el manejo de inserciones en la Base de Metadatos de Calidad.

Es importante señalar en este punto, que a los servicios de calidad se les envía la conexión a la Base de Datos Intermedia en lugar de enviar todos los datos de las *Expected Tables*. Con esto se busca minimizar el volumen de datos a transferir y el tiempo de procesamiento adicional sobre los mismos.

Por más detalles sobre los servicios de calidad consultar Anexo C (*Web Services*).

Subproceso Do Quality Entity Integration
El único subproceso del Proceso de Carga se denomina '*Do Quality Entity Integration*' (ver Figura 52) y es el encargado de iterar por cada *Expected Schema* (asociado a más de una FDW al que se instanció métricas de calidad y se asoció servicios de integración de entidades), y realizar la integración de entidades correspondiente. Al final del subproceso, se tiene por cada *Expected Schema* asociada a varias *Expected Tables*, una única tabla integrada que no contiene duplicados denominada **entity_integrated_tableName** donde *tableName* es el nombre del *Expected Schema*.

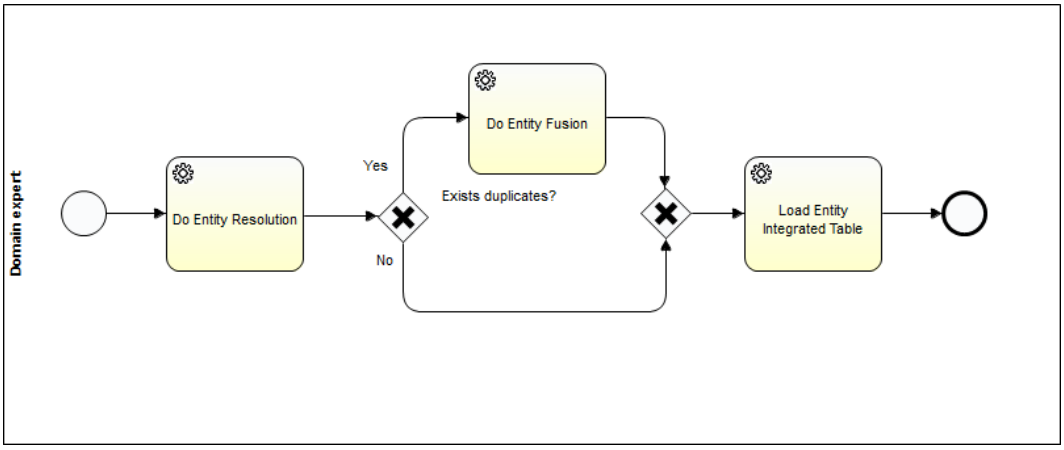


Figura 52 - Subproceso 'Do Quality Entity Integration'

A continuación se detallan las actividades que componen el mismo.

Do Quality Entity Resolution
La primera tarea automática denominada '*Do Entity Resolution*' del subproceso '*Do Quality Entity Integration*', es la encargada de detectar entidades duplicadas entre las *Expected Tables* asociadas al mismo *Expected Schema*. En esta tarea se invoca el servicio *Entity Resolution* que es obtenido de la Base de Metadatos de Configuración.

Do Entity Fusion
La tarea automática '*Do Entity Fusion'* se ejecuta sólo en caso que en la tarea anterior ('*Do Entity Resolution*') se hayan detectado entidades duplicadas. Esta tarea es encargada de invocar el servicio *Entity Fusion* que fue definido en la configuración para el *Expected Schema* sobre el que se itera. El servicio invocado se encarga de seleccionar la entidad más confiable entre cada grupo de entidades duplicadas, considerando los valores de calidad que se almacenaron para cada *Expected Table* asociada al Expected en la Base de Metadatos de Calidad.

Esta tarea también se encarga de registrar el valor 0 en el atributo **is_selected** de la tabla de traza de datos asociada al Expected (**trace_web_source_exp_tableName**) para aquellas tuplas que no fueron seleccionadas de entre las duplicadas. De ese modo, se indica que estas tuplas no serán consideradas a partir de ahora en el Proceso de Carga. Cabe recordar que inicialmente todas las tuplas tienen el valor 1 en dicho atributo.

En algunas ocasiones las mediciones de calidad de las entidades duplicadas son las mismas, y por consiguiente, el servicio *Entity Fusion* no puede determinar cuál de todas ellas es la más confiable. Para estos casos, se deja a criterio del servicio cuál duplicado escoger y se registra el conjunto de entidades duplicadas y la selección en una tabla denominada **log_entity_integration.** La razón de registrar estos conflictos es permitir que el usuario consulte dicha tabla y/o los resuelva manualmente en una tarea posterior.

Load Entity Integrated Table
La última tarea del subproceso *'Do Quality Entity Integration'* se encarga de cargar las tablas de la forma **entity_integrated_tableName.** Para realizar esta carga se realiza el

*join*entre las tuplas de la tabla de traza del *Expected Schema* actual (**trace_web_source_exp_tableName**)que contienen el valor 1 en el atributo **is_selected** con las *Expected Tables* (también asociadas al *Expected* actual), y se cargan sólo los datos de las *Expected Tables* en la tabla **entity_integrated_tableName**. De esta manera, se obtienen todas las entidades de las *Expected Tables* asociadas al mismo *Expected Schema*, que continúan en el proceso de carga.

Do Entity Integration

La tarea automática '*Do Entity Integration***'** (ver Figura 52) es invocada sólo si existen en la configuración seleccionada *Expected Schemas* asociados a más de una FDW y a los mismos no se les definió calidad o servicios de integración. Por cada uno de estos *Expected Schemas* se procede a realizar la misma lógica que en el subproceso '*Do Quality Entity Integration*' pero en este caso se utiliza un servicio *Entity Resolution* y un servicio *Entity Fusion* general del sistema que se encuentra especificado en el código del proyecto.

Dado que el usuario final desconoce el algoritmo que utiliza el servicio *Entity Fusion* de esta tarea, se decidió que todas las entidades duplicadas así como las seleccionadas fueran registradas en la tabla **log_entity_integration,** que fue explicada en la tarea anterior.

Show Conflicts of Entity Integration

En esta tarea se despliegan todos los conflictos no resueltos de la integración registrados en la tabla **log_entity_integration.** Los conflictos se muestran agrupados por *Expected Schema* y de cada conflicto se muestran las entidades duplicadas junto a la FDW de la que provienen. Cada entidad duplicada está precedida de un *radiobutton* que indica si la entidad fue seleccionada o no por el servicio *Entity Fusion*. El usuario puede modificar dicha selección.

5.3.3 Carga de Integrated Tables y propagación de calidad de los datos

En este punto se detalla el funcionamiento y la implementación de las actividades de la Figura 53.

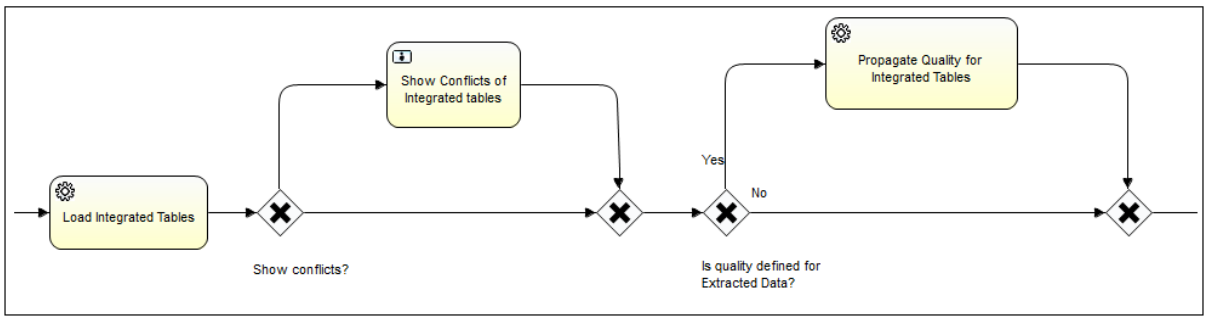


Figura 53 - Proceso de Carga (Parte 3)

Load Integrated Tables
En este punto se procede a cargar las *Integrated Tables*, ejecutando la actividad automática de nombre "*Load Integrated Tables"* (Ver Figura 53). Para realizar esto se efectúan *joins* entre las *Expected Tables* a partir de los metadatos definidos en la configuración de *joins* y de los mapeos entre atributos del *Expected Schema* con los atributos del *Integrated Schema*.

En esta tarea se pueden producir errores al momento de cargar los datos y conflictos al realizar los *joins* entre las Expected Tables. Los errores al cargar los datos se registran en la tabla **log_integration_data** y los conflictos al realizar los *joins* en la tabla **log_integration_join**. El propósito de estas tablas es, al igual que las demás tablas de *logs*, el poder ser consultadas para la toma de decisiones o mostrar los conflictos al usuario final en una tarea posterior.

Show Errors of Integrated Tables
La tarea manual '*Show Errors of Integrated Tables*'se ejecuta sólo en caso que el usuario final haya elegido en la primera tarea del proceso la opción de mostrar los errores y conflictos durante la carga. Esta tarea es la encargada de exhibir en un formulario los errores y conflictos almacenados en las tablas **log_integration_data** y **log_integration_join**.Los errores de la tabla **log_integration_data** son agrupados por las mismas categorías definidas para los errores de la extracción y los de la tabla **log_integration_join** bajo la etiqueta **'***Unmatched joins rows***'.** Todos los errores a su vez son agrupados por *Integrated Table*.

Propagate Quality of Integrated Tables
La tarea automática '*Propagate Quality of Integrated Tables*' es la encargada de propagar los valores de calidad medidos sobre las *Expected Tables* hacia las *Integrated Tables*.

Esta tarea depende fuertemente de la variable del proceso que contiene una estructura con información relativa a la propagación de la calidad mencionada en la actividad "*Create Quality Structures*". Dicha estructura permite determinar cuáles valores de calidad deben ser propagados. Se tiene por cada *Integrated Table* información de las métricas de las *Expected Tables* a considerar, además se cuenta con la información de cómo los atributos son mapeados, por lo tanto con dicha información es posible construir un sentencia SQL para obtener la calidad propagada y poder de esta manera persistir los datos en la tabla llamada **propagated_integrated_tableName**.

5.3.4 Carga del DW, propagación y medición de la calidad de los datos del DW.

En este punto se detalla el funcionamiento y la implementación de las actividades de la Figura 54.

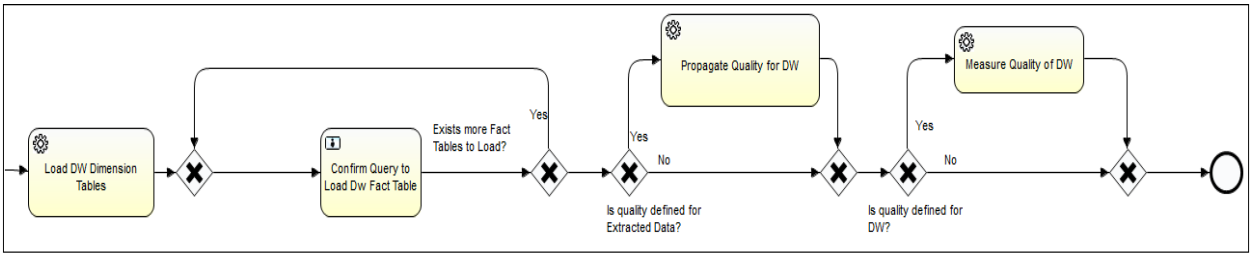


Figura 54 - Proceso de Carga (Parte 4)

Load DW Dimension Tables

En esta actividad automática*'Load DW Dimension Tables'* (Ver Figura 54) se cargan las tablas de dimensiones del WW. Esto se realiza utilizando los metadatos de *joins* y los mapeos con las tablas del *Integrated Schema* de manera análoga a la carga de las *Integrated Tables*.

Confirm Query To Load DW Tables

Esta actividad manual de nombre *'Confirm Query To Load DW Tables'* se ejecuta por cada una de las tablas de hechos que tenga el DW a construir. La tarea consiste en generar la consulta SQL que carga la tabla de hechos sobre la que se está iterando y se le muestra al usuario para que visualice cómo ha quedado su DW. El usuario puede confirmar la consulta o modificarla a su gusto. En caso que la sintaxis de la misma no sea correcta se despliega un mensaje de error con la excepción que lanzó el manejador de base de datos al ejecutarla; no se permite que el usuario avance en el proceso hasta que la sintaxis sea correcta.

Propagate Quality for DW

La tarea automática de nombre '*Propagate Quality for DW*' se encarga de propagar los valores de calidad que tienen las *Integrated Tables* hacia las *DW Tables*. La implementación de dicha tarea es análoga a la tarea automática '*Propagate Quality of Integrated Tables*'.

Measure Quality of DW

La tarea automática '*Measure Quality of DW*' es la encargada de realizar las mediciones de calidad sobre las tablas del DW y almacenarlas en la Base de Metadatos de Calidad. La tarea se ejecuta sólo en caso que se haya definido calidad sobre al menos una tabla del DW. Su implementación es análoga a la tarea '*Measure Quality of Extracted Data*'**.**

5.4 Posibles extensiones y mejoras

En este punto se mencionan posibles extensiones y mejoras sobre los procesos implementados. Las mismas se presentan ordenadas de mayor a menor prioridad según el costo/beneficio de su implementación.

Posible Mejora 1

La primera mejora que se considera necesaria realizar es respecto a los formularios de Activiti, los cuales no presentan una interfaz del todo amigable. Se podrían emplear otras herramientas como por ejemplo *Bonita*[21] para solucionar varios de los problemas que se presentan a nivel de interfaz como son los controles en campos o la imposibilidad de

implementar pantallas dinámicas Cabe destacar que aplicando esta mejora se podrían reutilizar las clases *Java* junto a las bases de datos definidas.

Posible Mejora 2
Otra de las extensiones importantes a realizar, es la de presentar al usuario, al final del Proceso de Carga, un reporte que contenga las mediciones de calidad que se realizaron durante el proceso, en especial sobre las mediciones realizadas sobre el DW resultado.

Posible Mejora 3
Otra posible mejora es la de poder modificar una configuración ya realizada, o "volver atrás" en el Proceso de Configuración y realizar modificaciones a las definiciones hechas, de modo de no tener que empezar una configuración nueva si el usuario cometió un error o desea realizar algún cambio como modificar el esquema del DW, sustituir una FDW por otra o utilizar otro servicio de integración de entidades.

Posible Mejora 4
También sería útil continuar trabajando con la resolución de conflictos por parte del usuario durante el Proceso de Carga. Es decir incluir en las tareas manuales que presentan errores y/o conflictos, la posibilidad de que el usuario pueda resolver esos errores directamente en el proceso.

Posible Mejora 5
Como se mencionó en el capítulo 2 (Marco Teórico), la calidad de los datos obtenidos de la Web se diferencia entre la calidad propia de los datos y la calidad aplicada a los servicios de extracción. En este proyecto sólo se incluyó la calidad asociada a los datos, en un futuro se podría extender el proceso para que también incluya la calidad aplicada a los servicios de extracción. De este modo se podría analizar aspectos de seguridad y/o estabilidad de las FDW así como medir con qué rapidez el servicio responde.

Posible Mejora 6
Contar con un sistema adicional en el que el usuario pueda dar alta, baja y modificación de distintos objetos de la realidad del proyecto como son dominios, FDW, granularidades, dimensiones y factores, entre otros.

Posible Mejora 7
Si en el Proceso de Configuración, se define un servicio *Entity Fusion* que identifica entidades duplicadas por atributos que no son la *primary key* del *Expected Schema*, luego en el proceso de carga, en caso de que existan dos entidades con la misma *primary key* en *Expected Tables* distintas, sólo una entidad será cargada en la tabla **entity_integration_tableName**. La otra entidad no podrá ser cargada por error de clave duplicada. Como mejora se propone que se añada una tabla **log_entity_integration_errors** que registre estos casos y que en un futuro el usuario pueda resolverlos en alguna tarea manual también.

6 - Caso de Estudio

En este capítulo se presenta el caso de estudio a aplicar sobre el prototipo realizado, lo cual incluye contexto, recursos, metadatos asociados, y el diseño conceptual y lógico del DW a construir. Además se describe el proceso de Configuración y Carga a realizar en la ejecución del mismo.

6.1 Contexto

Se seleccionó un caso de estudio de datos abiertos de Reino Unido[22] a aplicar sobre el dominio '*Transporte*'. Dicho caso de estudio cubre todos los requisitos planteados en el punto 3.4 del presente informe. Como extensiones al caso de estudio del proyecto anterior, cabe mencionar que en este se definen aspectos de calidad durante el proceso de Configuración que luego son utilizados en el proceso de Carga para la resolución de conflictos, y se incluye en el DW una medida calculada además de las medidas de conteos por cada dimensión que ya se obtenían en el anterior.

El caso de estudio cuenta con distintas FDW que proveen información detallada de accidentes de tránsito en Reino Unido entre los años 2005 y 2014, los vehículos involucrados y las víctimas consecuentes. Estos datos son públicos y sólo refieren a los accidentes de lesiones personales en la vía pública que se reportan a la policía, y posteriormente se registran en un formulario denominado '*STATS19*'[23].

En este capítulo, se pretende diseñar y construir un WW que permita explotar los datos de Seguridad Vial del Reino Unido, es decir, analizar los accidentes a lo largo de los años y diversos aspectos asociados a los mismos, entre los que se pueden destacar costos, cantidad de vehículos y víctimas involucradas.

6.2 Recursos y Metadatos

Los datos de accidentes, vehículos y víctimas provistos se encuentran en su mayoría codificados, y se cuenta a su vez con fuentes de metadatos en un archivo xls [24] compuesto por diversas codigueras que referencian a los datos de interés. A continuación se especifican los recursos y metadatos de las FDW que se utilizan en la construcción del WW.

Accidentes (data.gov.uk)

Desde la página web data.gov.uk se extraen los datos de los accidentes de tránsito del Reino Unido publicados por el *Department for Transport* (DfT)[25]. Los datos están contenidos en un archivo csv en donde cada fila representa la información asociada a un accidente específico. Se puede obtener un archivo csv por año (desde el 2005 al 2014) o la totalidad de los registros de esos años en un único archivo csv.

Recurso	Url	Nombre del Archivo	Cantidad de Registros
Accidents	http://data.dft.gov.uk.s3.amazonaws.com/road-accidents-safety-data/Stats19_Data_2005-2014.zip	accidents0514.csv	1.640.597

Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity	Number_of_Vehicles
200501BS00001	525680	178240	-0.191170	51.489.096	1	2	1
200501BS00002	524170	181650	-0.211708	51.520.075	1	3	1
200501BS00003	524520	182240	-0.206458	51.525.301	1	3	2
200501BS00004	526900	177530	-0.173862	51.482.442	1	3	1
200501BS00005	528060	179040	-0.156618	51.495.752	1	3	1
200501BS00006	524770	181160	-0.203238	51.515.540	1	3	2
200501BS00007	524220	180830	-0.211277	51.512.695	1	3	2
200501BS00009	525890	179710	-0.187623	51.502.260	1	3	1
200501BS00010	527350	177650	-0.167342	51.483.420	1	3	2
200501BS00011	524550	180810	-0.206531	51.512.443	1	3	2
200501BS00012	526240	178900	-0.182872	51.494.902	1	3	1
200501BS00014	526170	177690	-0.184312	51.484.044	1	3	2
200501BS00015	525590	178520	-0.192366	51.491.632	1	3	1
200501BS00016	527990	178690	-0.157753	51.492.622	1	3	2
200501BS00017	526700	178970	-0.176224	51.495.429	1	3	1

Figura 55- Archivo csv de Accidentes entre 2005 y 2014 en Reino Unido

Metadatos de Contenido

Nº Columna	Nombre	Tipo de Dato	Descripción
1	Accident_Index	Alfanumérico(20)	Identificador del accidente
10	Date	Date(dd/MM/YY)	Fecha del accidente
32	LSOA_of_Accident_Location	Alfanumérico(20)	Representa el código de un área geográfica del accidente

Vehículos (data.gov.uk)

Desde la misma página web que se extrajeron los accidentes de tránsito del Reino Unido, el DfT [25] publicó también los vehículos asociados a dichos accidentes. Estos datos están contenidos en un archivo csv en donde cada registro representa información asociada a un vehículo particular involucrado en un accidente determinado. De igual forma que para el caso de los accidentes, los datos provistos se encuentran en un archivo csv por año, o en un archivo único, los de todos los años.

Recurso	Url	Nombre del archivo	Cantidad de Registros
Vehicles	http://data.dft.gov.uk.s3.amazonaws.com/road-accidents-safety-data/Stats19_Data_2005-2014.zip	vehicles0514.csv	3.004.425

Accident_Index	Vehicle_Reference	Vehicle_Type	Towing_and_Articulation	Vehicle_Manoeuvre	Vehicle_Location-Restricted_Lane
200501BS00001	1	9	0	18	0
200501BS00002	1	11	0	4	0
200501BS00003	1	11	0	17	0
200501BS00003	2	9	0	2	0
200501BS00004	1	9	0	18	0
200501BS00005	1	3	0	18	0
200501BS00006	1	9	0	5	0
200501BS00006	2	3	0	18	0
200501BS00007	1	3	0	18	0
200501BS00007	2	9	0	2	0
200501BS00009	1	9	0	18	0
200501BS00010	1	9	0	18	0
200501BS00010	2	9	0	9	0
200501BS00011	1	11	0	9	0
200501BS00011	2	90	0	18	0
200501BS00012	1	9	0	18	0
200501BS00014	1	9	0	9	0
200501BS00014	2	3	0	18	0
200501BS00015	1	9	0	9	0
200501BS00016	1	9	0	18	0
200501BS00016	2	9	0	7	0

Figura 56- Archivo csv de Vehículos entre 2005 y 2014 en Reino Unido

Metadatos de Contenido

Nº Columna	Nombre	Tipo de Dato	Descripción
1	Accident_Index	Alfanumérico(20)	Identificador del accidente
2	Vehicle_Reference	Numérico(3)	Identificador del vehículo en el accidente
3	Vehicle_Type	Numérico(2)	Tipo de Vehículo
15	Sex_of_Driver	Numérico(1)	Sexo del conductor
16	Age_of_Driver	Numérico(2)	Edad del conductor
20	Age_of_Vehicle	Numérico(3)	Antigüedad del vehículo

Víctimas (data.gov.uk)

Al igual que las FDW de accidentes y de vehículos asociados, las víctimas de los accidentes se extrajeron de la página data.gov.uk y fue publicada por DfT[25]. Los datos también están contenidos en un archivo csv en donde cada registro representa a una víctima involucrada en un accidente específico. Como en ambos casos anteriores, accidentes y vehículos, los datos provistos se encuentran en un archivo csv por año o todos los años en un mismo archivo csv.

Recurso	Url	Nombre del Archivo	Cantidad de Registros
Casualties	http://data.dft.gov.uk.s3.amazonaws.com/road-accidents-safety-data/Stats19_Data_2005-2014.zip	casualties0514.csv	2.216.720

Accident_Index	Vehicle_Reference	Casualty_Reference	Casualty_Class	Sex_of_Casualty	Age_of_Casualty
200501BS00001	1	1	3	1	37
200501BS00002	1	1	2	1	37
200501BS00003	2	1	1	1	62
200501BS00004	1	1	3	1	30
200501BS00005	1	1	1	1	49
200501BS00006	2	1	1	2	30
200501BS00007	1	1	1	1	31
200501BS00009	1	1	3	2	13
200501BS00009	1	2	3	2	13
200501BS00010	1	1	1	1	35
200501BS00010	2	2	1	2	48
200501BS00011	1	1	2	2	26
200501BS00011	1	2	2	2	9
200501BS00011	1	3	2	2	40

Figura 57- Archivo csv de Víctimas entre 2005 y 2014 en Reino Unido

Metadatos de Contenido

Nº Columna	Nombre	Tipo de Dato	Descripción
1	Accident_Index	Alfanumérico(20)	Identificador del accidente
2	Vehicle_Reference	Numérico(3)	Identificador del vehículo en el accidente
3	Casualty_Reference	Numérico(3)	Identificador de la víctima en el accidente
4	Casualty_Class	Numérico(1)	Rol de víctima
5	Sex_of_Casualty	Numérico(1)	Sexo de la víctima
6	Age_of_Casualty	Numérico(2)	Edad de la víctima
8	Casualty_Severity	Numérico(1)	Severidad del daño a la víctima

Localidades (data.gov.uk y ons.gov.uk)

Las localidades se extraen de dos FDW distintas. La primera FDW es la misma que la utilizada para extraer los accidentes pero en este caso solo se obtiene la información relevante del área geográfica. Mientras que la otra FDW generada a partir del censo realizado en el año 2011 [26], constituye una clasificación de determinadas áreas en urbanas o rurales de Inglaterra y Gales (LSOA). Dicha clasificación que se denomina LSOA se basa fuertemente en la densidad de población de cada área. Esta FDW se encuentra en un archivo csv[27] en donde cada registro representa un área determinada.

Como sólo se tiene código LSOA para Inglaterra y Gales, se consideran sólo los accidentes ocurridos en estos dos países dado que no hay manera de integrar los datos de Escocia con éstas localidades.

Recurso	Url	Nombre del Archivo	Cantidad de Registros
Accidents	http://data.dft.gov.uk.s3.amazonaws.com/road-accidents-safety-data/Stats19_Data_2005-2014.zip	accidents0514.csv	1.640.597
Classification	https://geoportal.statistics.gov.uk/Docs/Products/Rural-urban_classification_(2011)_of_lower_layer_super_output_areas_(2011)_E+W.zip	RUC11_LSOA11_EW.csv	34.753

LSOA11CD	LSOA11NM	RUC11CD	RUC11
E01000001	City of London 001A	A1	Urban major conurbation
E01000002	City of London 001B	A1	Urban major conurbation
E01000003	City of London 001C	A1	Urban major conurbation
E01000005	City of London 001E	A1	Urban major conurbation
E01000006	Barking and Dagenham 016A	A1	Urban major conurbation
E01000007	Barking and Dagenham 015A	A1	Urban major conurbation
E01000008	Barking and Dagenham 015B	A1	Urban major conurbation
E01000009	Barking and Dagenham 016B	A1	Urban major conurbation
E01000010	Barking and Dagenham 015C	A1	Urban major conurbation
E01000011	Barking and Dagenham 016C	A1	Urban major conurbation
E01000012	Barking and Dagenham 015D	A1	Urban major conurbation
E01000013	Barking and Dagenham 013A	A1	Urban major conurbation
E01000014	Barking and Dagenham 013B	A1	Urban major conurbation

Figura 58- Archivo csv de Clasificación de áreas urbanas/rurales 2011 (Inglaterra y Gales)

Metadatos de Contenido

FDW 1: Accidents

Nº Columna	Nombre	Tipo de Dato	Descripción
32	LSOA_of_Accident_Location	Alfanumérico(20)	Representa el código de un área geográfica
30	Urban_or_Rural_Area	Numérico(1)	Indica si el área geográfica es rural o no.

FDW 2: Classification

Nº Columna	Nombre	Tipo de Dato	Descripción
1	LSOA11CD	Alfanumérico(20)	Representa el código de un área geográfica
2	LSOA11NM	Alfanumérico(100)	Nombre del área geográfica
3	RUC11CD	Alfanumérico(2)	Código de la clasificación
4	RUC11	Alfanumérico(100)	Nombre de la clasificación

Costos (autogenerado)

Se construye un archivo csv que contiene los costos asociados a las víctimas en un accidente. Esta fuente de datos se añadió con el fin de cumplir con el requerimiento de que el DW resultado del caso de estudio tenga una medida interesante además de los conteos por cada dimensión. El costo asociado a cada víctima se estimó a partir de las tablas estadísticas publicadas junto a las otras fuentes de datos[28].

Recurso	Url	Nombre del Archivo	Cantidad de Registros
Costs	No aplica	costs.csv	2.216.720

Metadatos de Contenido

Nº Columna	Nombre	Tipo de Dato	Descripción
1	Accident_Index	Alfanumérico(20)	Identificador del accidente
2	Vehicle_Reference	Numérico(3)	Identificador del vehículo en el accidente
3	Casualty_Reference	Numérico(3)	Identificador de la víctima en el accidente
4	Cost	Float	Costo asociado a la víctima

Accident_Index	Vehicle_Reference	Casualty_Reference	Cost
200501BS00001	1	1	17.829.772.469.810.200
200501BS00002	1	1	13.600.580.756.492.500
200501BS00003	2	1	13.980.886.403.765.400
200501BS00004	1	1	13.957.689.780.083.100
200501BS00005	1	1	13.700.789.719.853.300
200501BS00006	2	1	13.485.879.289.750.800
200501BS00007	1	1	13.182.027.319.927.700
200501BS00009	1	1	13.307.498.761.119.700
200501BS00009	1	2	13.846.411.876.549.300
200501BS00010	1	1	1.316.456.313.012.120
200501BS00010	2	2	14.138.580.051.691.800
200501BS00011	1	1	1.405.421.089.882.880
200501BS00011	1	2	13.710.314.369.802.600
200501BS00011	1	3	13.243.939.183.260.200
200501BS00011	1	4	13.943.752.837.626.900
200501BS00011	1	5	13.841.946.669.087.600

Figura 59 - Datos del archivo csv de Costos

Fechas (autogenerado)

Se construye un archivo csv que contiene todas las fechas comprendidas entre el año 2005 y el 2014 y además se agregó el mes, el año y el día de la semana.

Recurso	Url	Nombre del Archivo	Cantidad de Registros
Dates	No aplica	dates.csv	3652

Metadatos de Contenido

Nº Columna	Nombre	Tipo de Dato	Descripción
1	Id	Numérico(8)	Identificador. Formato YYYYMMDD
2	Month_id	Numérico(2)	Identificador del mes
3	Month_label	Alfanumérico(9)	Mes de la fecha
4	Year	Numérico(4)	Año de la fecha
5	Day_of_week_id	Numérico(1)	Identificador del día de la semana
6	Day_of_week_label	Alfanumérico(9)	Día de la semana

Date	Month_id	Month_label	Year	Day_of_week_id	Day_of_week_label
20050101	1	January	2005	7	Saturday
20050102	1	January	2005	1	Sunday
20050103	1	January	2005	2	Monday
20050104	1	January	2005	3	Tuesday
20050105	1	January	2005	4	Wednesday
20050106	1	January	2005	5	Thursday
20050107	1	January	2005	6	Friday
20050108	1	January	2005	7	Saturday
20050109	1	January	2005	1	Sunday
20050110	1	January	2005	2	Monday
20050111	1	January	2005	3	Tuesday
20050112	1	January	2005	4	Wednesday
20050113	1	January	2005	5	Thursday
20050114	1	January	2005	6	Friday
20050115	1	January	2005	7	Saturday
20050116	1	January	2005	1	Sunday
20050117	1	January	2005	2	Monday
20050118	1	January	2005	3	Tuesday
20050119	1	January	2005	4	Wednesday
20050120	1	January	2005	5	Thursday
20050121	1	January	2005	6	Friday

Figura 60 - Datos del archivo csv de Fechas

6.3 Diseño Conceptual del Web Warehouse

En este punto se presenta el diseño conceptual que se planteó para el caso de estudio seleccionado. La Figura 61 ilustra las dimensiones y jerarquías y la Figura 62, las relaciones dimensionales. El Modelo Multidimensional utilizado se denomina CMDM el cual se detalla en el Anexo A.

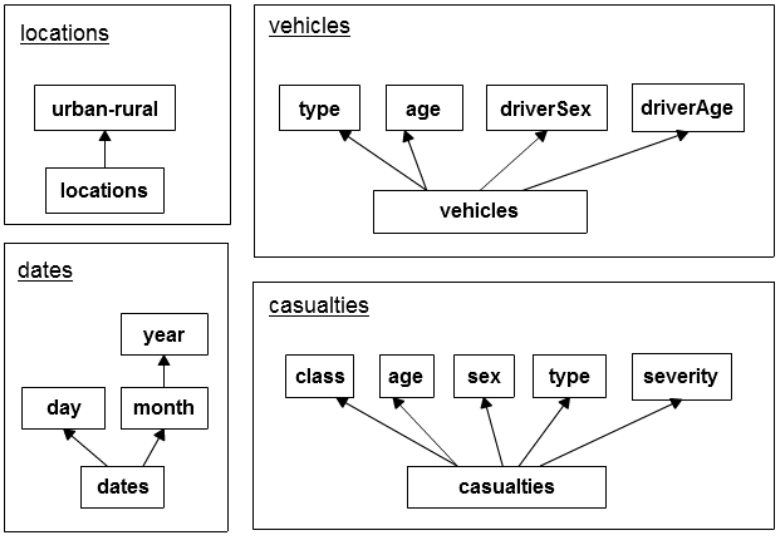


Figura 61 - Dimensiones y Jerarquías

Las dimensiones del WW a construir son:

1) *Vehicles*, con cuatro jerarquías: *Type, Age*, *DriverSex* y *Driver Age*, con cada una de ellas conteniendo un sólo nivel. En esta dimensión, *Age* hace referencia a la edad del vehículo.
2) *Casualties*, con cinco jerarquías de un sólo nivel: *Class, Age, Sex, Type* y *Severity*. La jerarquía *Class* refiere al rol de la víctima, el cual puede ser *pedestrian, driver*, o *passenger*.
3) *Location*, con una sola jerarquía de un nivel: *Urban-Rural*
4) Dates, con dos jerarquías: *Day* y *Month-year*, en donde *Day* contiene un sólo nivel y *Month-year* contiene dos niveles.

Por otra parte, las medidas del DW son *Quantity_vehicles*, *Quantity_casualties* y *cost_per_casualty*.

En comparación al DW obtenido en el proyecto anterior, este DW cuenta con una medida calculada además de los conteos por dimensión y contiene valores de calidad calculados sobre el mismo DW y propagados a partir de las FDW.

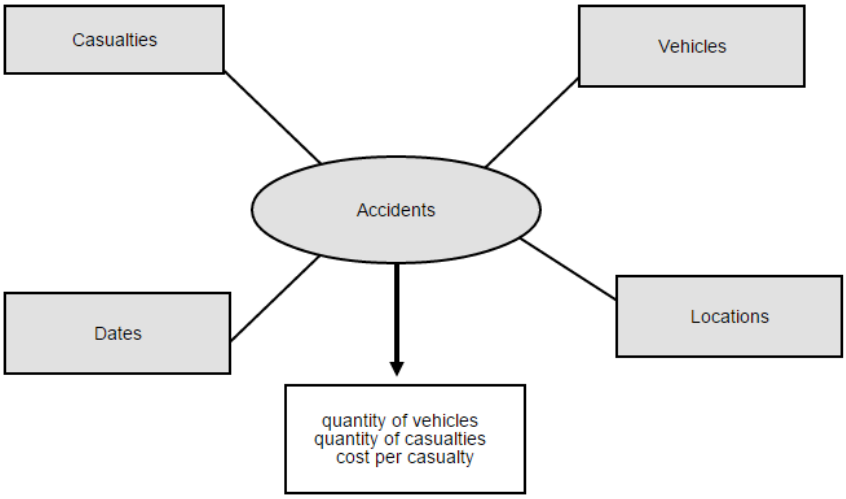


Figura 62 - Relaciones Dimensionales

6.4 Diseño Lógico del Web Warehouse

En este punto se presenta el diseño lógico del caso de estudio que consiste en un modelo estrella con una tabla de hechos y cuatro tablas de dimensiones definidas con sus respectivas claves primarias y foráneas. La Figura 63 ilustra dicho modelo en donde se tiene la tabla de hechos *Accidents* y las tablas de dimensiones *Dates*, *Casualties*, *Vehicles* y *Locations*.

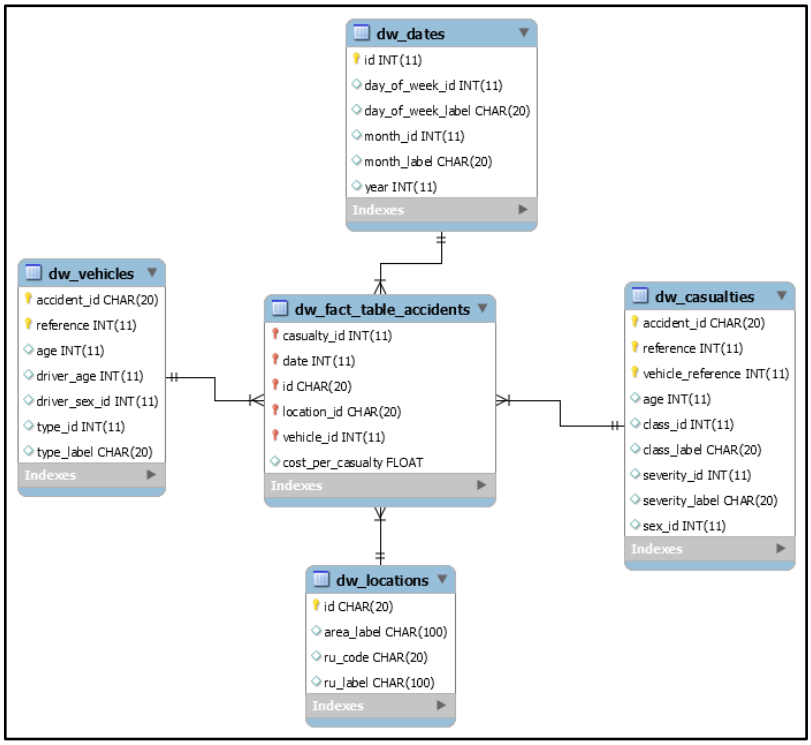


Figura 63 - Diagrama Modelo Estrella

6.5 Descripción del Proceso de Configuración

En este punto se describen los pasos necesarios para realizar la configuración del WW a construir. El detalle del uso de cada una de las actividades manuales se encuentra en el Anexo B - Manual de Usuario. La Figura 64 ilustra de manera gráfica el proceso de configuración, incluyendo la definición de las FDW (*web sources*), los *Expected Schemas*, el *Integrated Schema* y finalmente el *DW Schema*.

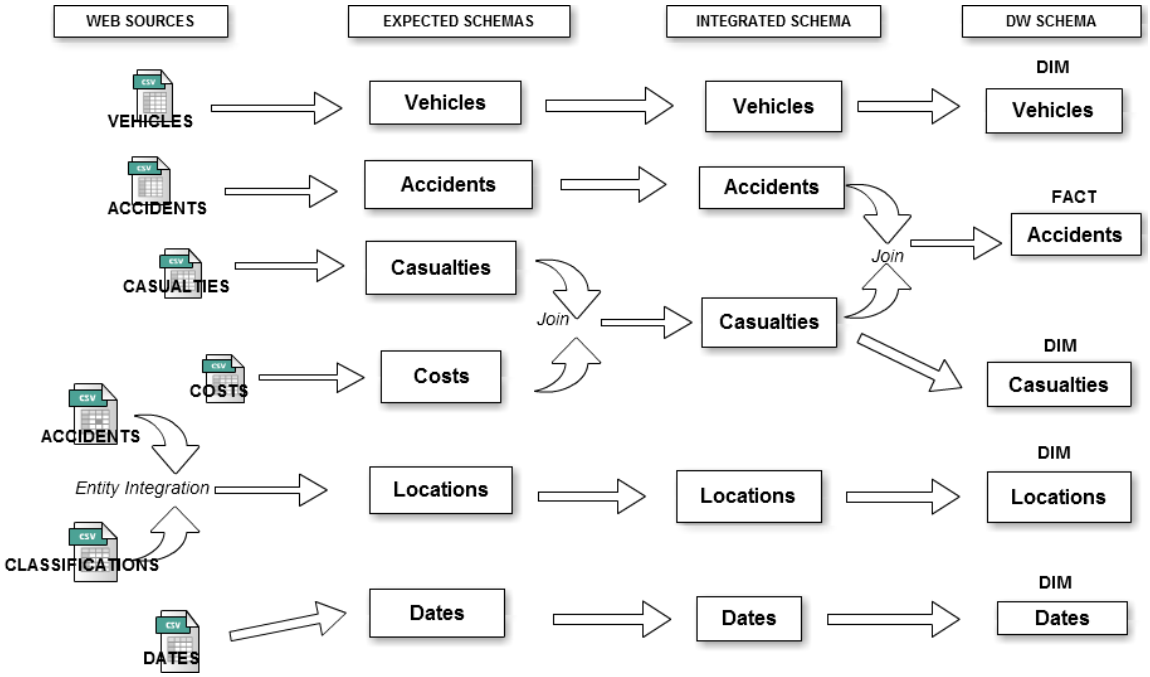


Figura 64 - Diagrama del Proceso de Configuración del Caso del Estudio

6.5.1 Dominio

El caso de estudio seleccionado corresponde al dominio '*Transport'*, por lo que al comenzar el proceso de Configuración, se selecciona dicho dominio y se ingresa el nombre "FULL_DEMO" a la configuración. Dicho nombre, como se mencionó antes, se le asignará a la Base de Datos Intermedia durante el Proceso de Carga. En la Figura 65,se muestra la actividad manual '*Select Domain*' que realiza lo descrito en este párrafo.

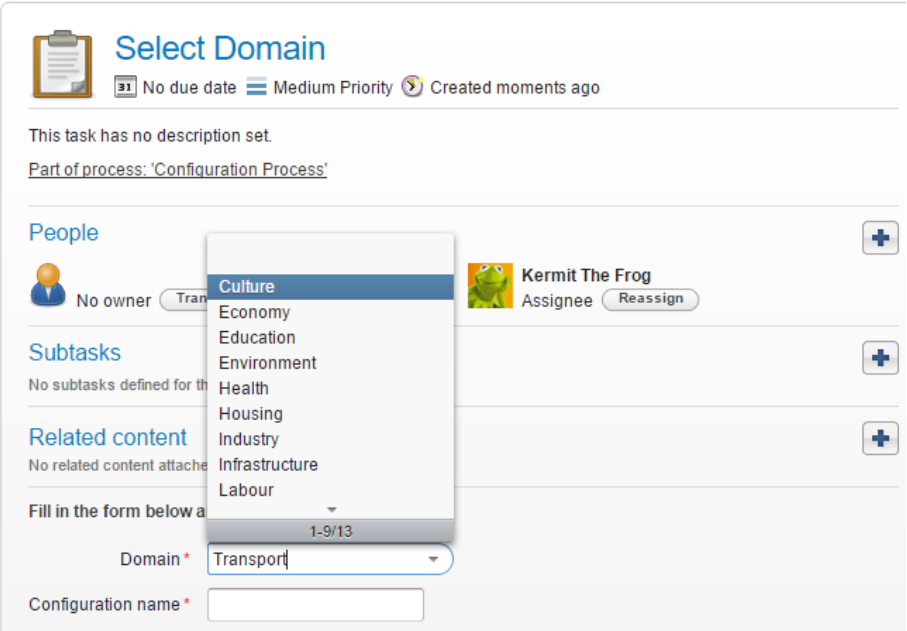


Figura 65 - Actividad manual '*Select Domain*'

6.5.2 Fuentes de Datos Web

Se comienza con definir las FDW de las que se extraerán los datos.La tabla 6 contiene todas las FDW que deben ser definidas en este caso de estudio.

Nombre	URL	Formato
Accidents	accidents0514.csv	CSV
Vehicles	vehicles0514.csv	CSV
Casualties	casualties0514.csv	CSV
Dates	dates.csv	CSV
Classification	RUC11_LSOA11_EW.csv	CSV
Locations	accidents0514.csv	CSV
Costs	costs.csv	CSV

Tabla 6 - FDW utilizadas en el caso de estudio.

6.5.3 Expected Schemas

A continuación, se presentan los datos necesarios para definir cada *Expected Schema* y su mapeo con la FDW.

Debido a que algunos de los datos de las FDW se encuentran codificados, al momento de realizar el mapeo del *Expected Schema* con la FDW, se hace uso de un servicio web

específico para obtener el valor correspondiente (en el sistema no se selecciona ningún mapeo). Para representar esta realidad en la tabla de atributos se indica 'WS' en el dato 'Mapeo con FDW'. Además si un atributo del *Expected Schema* no está definido en la FDW, en las siguientes tablas se indica 'N/A' en el dato 'Mapeo con FDW'. De igual forma que en el caso anterior, en el sistema no se selecciona ningún mapeo.

Accidents

FDW: Accidents

Servicio web: http://localhost:8080/DataServices/csv

Atributos:

Nombre Atributo	**Tipo**	**PK**	**Mapeo con FDW**
id	char(20)	✓	0- Accident_Index
date	Date		9 - Date
location_id	char(20)		31 - LSOA_of_Accident_Location

Vehicles

FDW: Vehicles

Servicio web: http://localhost:8080/DataServices/labelled_csv

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con FDW
accident_id	char(20)	✓	0- Accident_Index
reference	integer	✓	1 - Vehicle_Reference
type_id	Integer		2 - Vehicle_Type
type_label	char(20)		WS
driver_sex_id	Integer		14 - Sex_of_Driver
driver_age	Integer		15 - Age_of_Driver
age	Integer		19 - Age_of_Vehicle

Casualties

FDW: Casualties

Servicio web: http://localhost:8080/DataServices/labelled_csv

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con FDW
accident_id	char(20)	✓	0- Accident_Index
vehicle_reference	char(20)	✓	1 - Vehicle_Reference
reference	Integer	✓	2 - Casualty_Reference
class_id	Integer		3 - Casualty_Class
class_label	char(20)		WS
sex_id	Integer		4 - Sex_of_Casuality
age	Integer		5 - Age_of_Casuality
severity_id	Integer		7 - Casuality_Severity
severity_label	char(20)		WS

Costs

FDW: Costs

Servicio web: http://localhost:8080/DataServices/csv

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con FDW
accident_id	char(20)	✓	0- Accident_Index
vehicle_reference	char(20)	✓	1 - Vehicle_Reference
reference	Integer	✓	2 - Casualty_Reference
cost	float		3- Cost

Dates

FDW: Dates

Servicio web: http://localhost:8080/DataServices/csv

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con FDW
id	Integer	✓	0 - Date
month_id	Integer		1 - Month_id
month_label	char(20)		2 - Month_label
year	Integer		3 - Year
day_of_week_id	Integer		4 - Day_of_week_id
day_of_week_label	char(20)		5 - Day_of_week_label

Locations

En este caso se asocian dos FDW al mismo *Expected Schema* '*Locations'*.

FDW 1: Locations

Servicio web: http://localhost:8080/DataServices/labelled_csv

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con FDW
id	char(20)	✓	31 - LSOA_of_Accident_Location
area_label	char(255)		N/A
ru_code	char(20)		29 - Urban_or_Rural_Area
ru_label	char(100)		WS

FDW 2: Classifications

Servicio web: http://localhost:8080/DataServices/csv

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con FDW
id	char(20)	✓	0-LSOA11CD
area_label	char(100)		1-LSOA11NM
ru_code	char(20)		2-RUC11CD
ru_label	char(100)		3-RUC11

6.5.4 Métricas de Calidad a aplicar sobre los Expected Schemas

En este punto se define el Modelo de Calidad sobre los datos recién extraídos de la web. Para ello, primero se seleccionan los *Expected Schemas* a los cuales se desea instanciar métricas de calidad. La Figura 66 muestra la pantalla donde se realiza esta tarea.
Los *Expected Schemas* a los que se define calidad en este caso de estudio son:

1) *Locations*
2) *Casualties*
3) *Vehicles*
4) *Accidents*

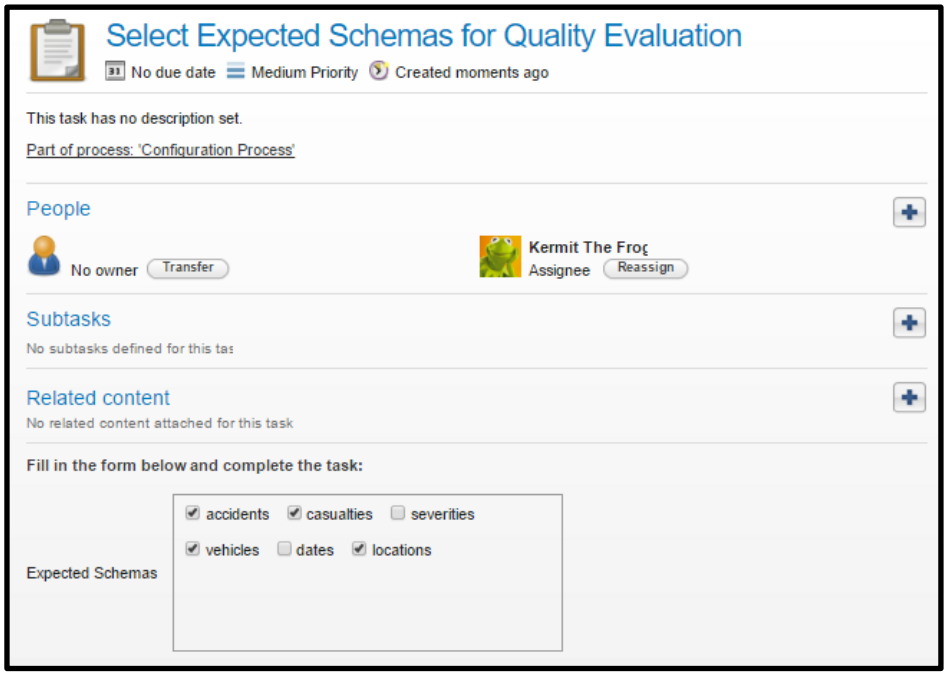


Figura 66 - Pantalla de Tarea '*Select Expected Schemas For Quality Evaluation*'

Una vez seleccionados los *Expected Schemas* a los cuales se desea aplicar calidad, se itera por cada uno de ellos, y se instancian las métricas de calidad. Las figuras 67, 68 y 69 ilustran dicha instanciación.

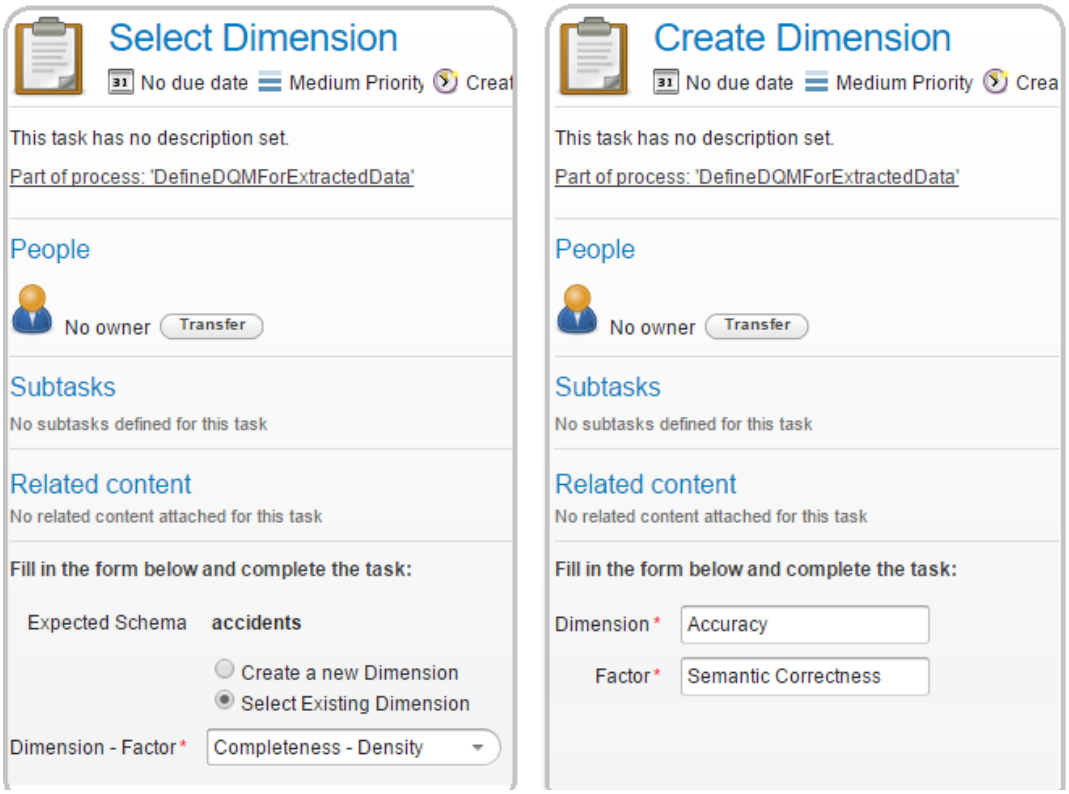


Figura 67 - Formularios para definir el par dimensión-factor: a) tarea *'Select Dimension'*, b) tarea *'Create Dimension'*

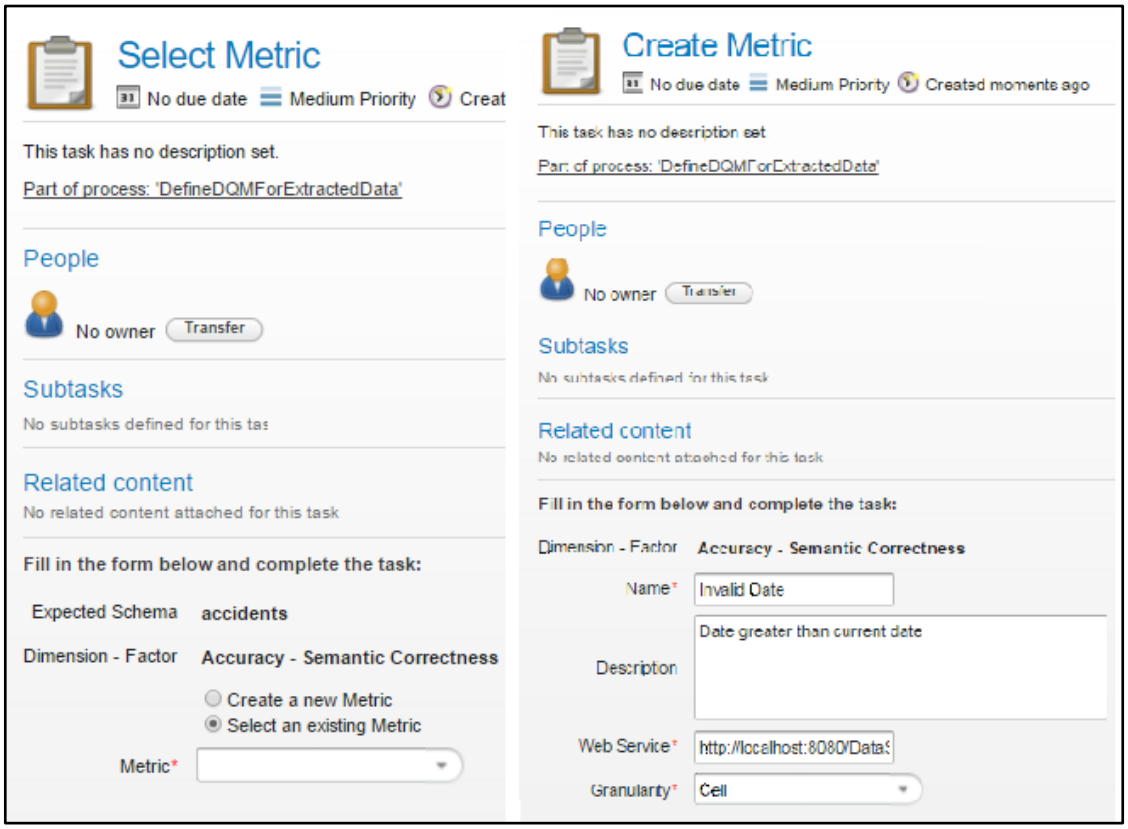


Figura 68 - Formularios para definir la métrica:a) tarea *'Select Metric'*, b) tarea *'Create Metric'*

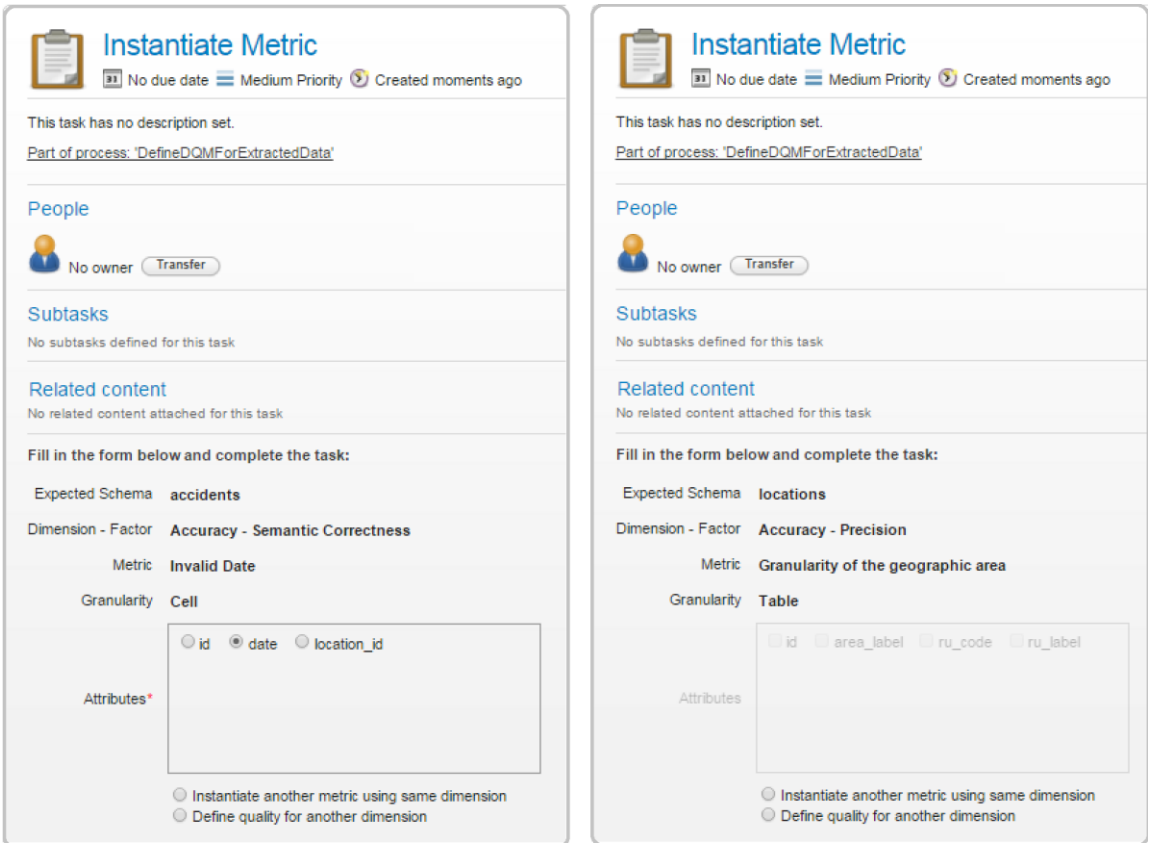


Figura 69 - Formularios para instanciar una métrica para un *Expected Schema*: a) granularidad '*Celda*', b). granularidad '*Tabla'*

Métricas a Instanciar sobre *Expected Schema 'Locations*':

1) Dimension: Accuracy
   Factor: Precision
   Metric Name: Granularity of the geographic area
   Description: Indicates how precisely is the geographical area given
   Web Service: http://localhost:8080/DataServices/areaPrecision
   Granularity: Table
2) Dimension: Completeness
   Factor: Density
   Metric Name: Density Ratio of Area Label
   Description: Indicates how many area labels are empty.
   Web Service: http://localhost:8080/DataServices/tableDensity
   Granularity: Table

Métricas a Instanciar sobre *Expected Schema 'Casualties*':

1) Dimension: Consistency
   Factor: Intra-Relation Integrity
   Metric Name: Under-Age Driver
   Description: Indicates, in case that the casualty is a driver, if the age given is valid
   Web Service: http://localhost:8080/DataServices/validAgeDriver
   Granularity: Cellset
   Attributes: class_id, age

Métricas a Instanciar sobre *Expected Schema 'Vehicles*':

1) Dimension: Consistency
   Factor: Domain Integrity
   Metric Name: Under-Age Driver
   Description: Indicates if the driver age is valid
   Web Service: http://localhost:8080/DataServices/validAgeDriver
   Granularity: Cell
   Attributes: driver_age

Métricas a Instanciar sobre *Expected Schema 'Accidents*':

2) Dimension: Accuracy
   Factor: Semantic Correctness
   Metric Name: Invalid Date
   Description: Date greater than current date
   Web Service: http://localhost:8080/DataServices/invalidDate
   Granularity: Cell
   Attributes: date

6.5.5 Entity Integration

En este punto se describe la definición de los servicios de integración de entidades para aquellos *Expected Schemas* asociados a más de una FDW y a los que se les haya instanciado métricas de calidad.

Lo primero a hacer, es seleccionar a cuáles *Expected Schemas* entre aquellos que cumplen las condiciones descritas en el párrafo anterior se les desea aplicar servicios de integración de entidades. En este caso de estudio sólo el *Expected 'Locations'* cumple tales condiciones (ver Figura 70).

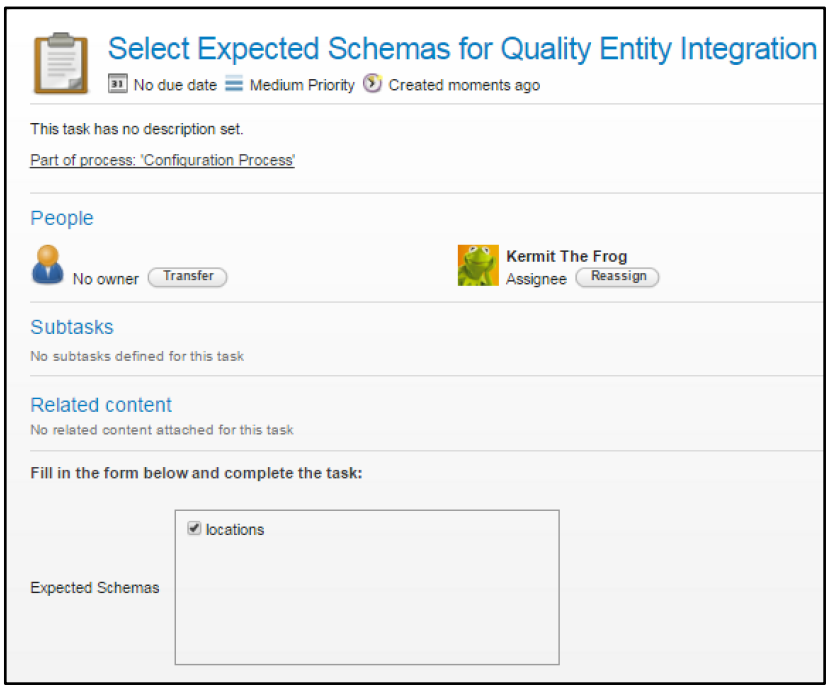


Figura 70- Tarea '*Select Expected Schemas For Quality Entity Integration*'

Una vez seleccionado el *Expected* '*Locations*', se definen los servicios de integración *Entity Resolution* y *Entity Fusion* que se describen a continuación (ver Figura 71 y 72):

Nombre Servicio	Descripción	URL
Entity Resolution	Determines records considered equals	http://localhost:8080/DataServices/entityResolution
Entity Fusion	Choose records from sets given using data quality	http://localhost:8080/DataServices/entityFusion

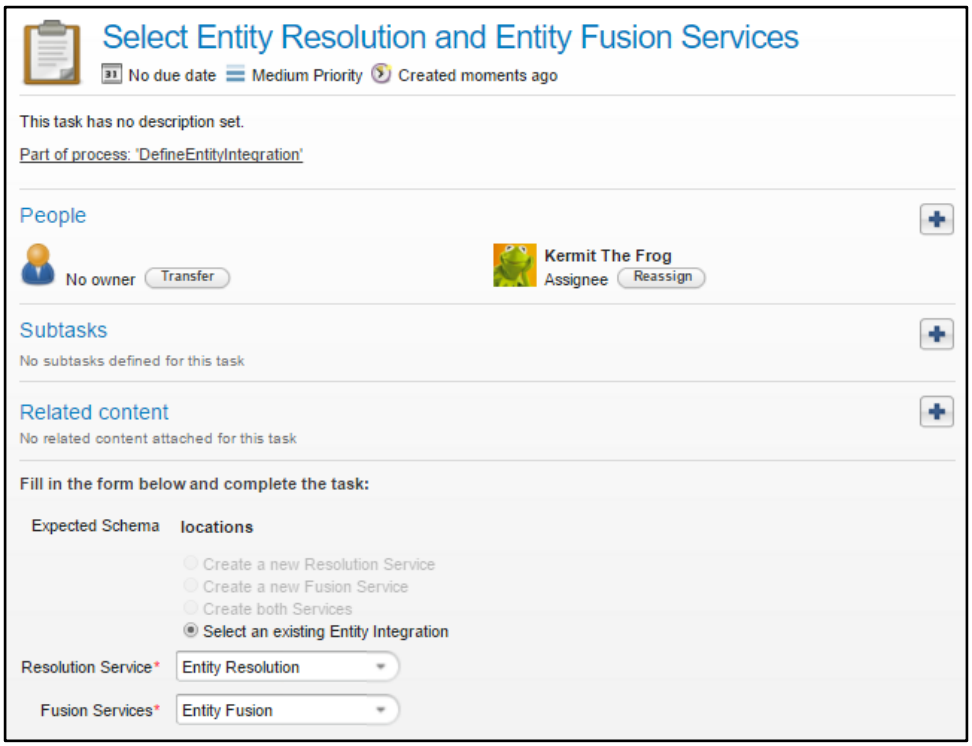


Figura 71 - Formularios para definir servicios Entity Integration

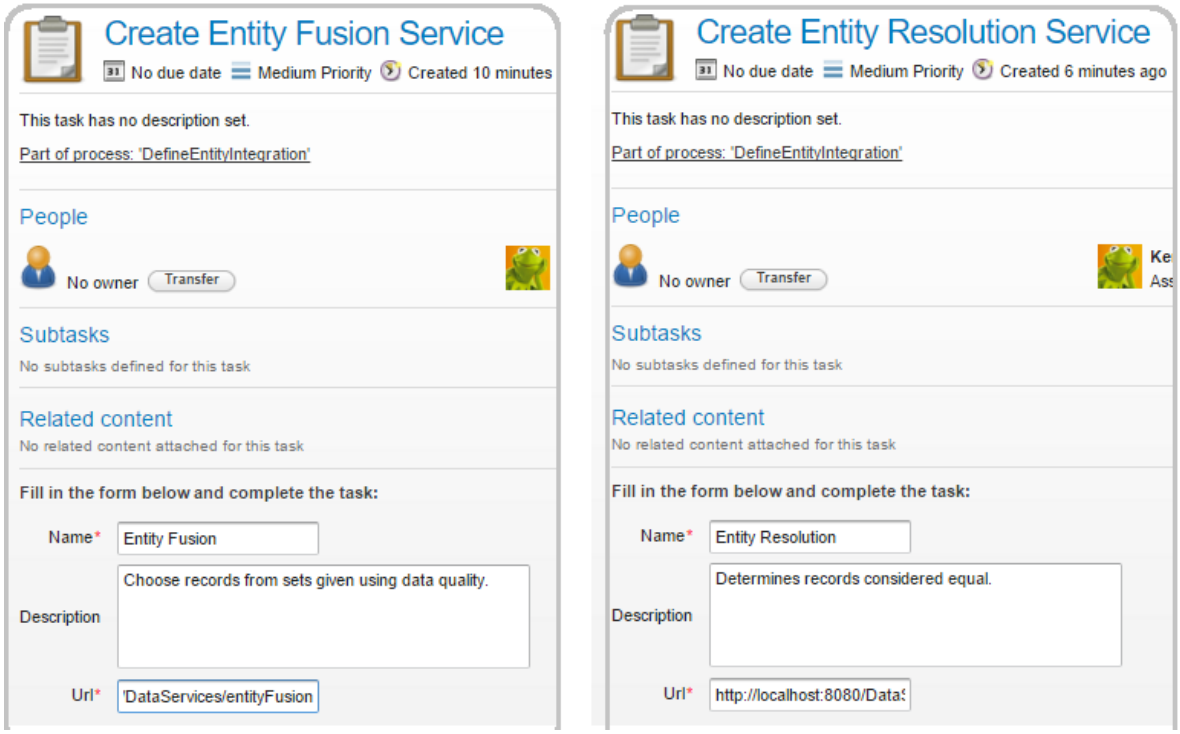


Figura 72 - Formularios para crear servicios Entity Integration:
a) tarea *'Create Entity Fusion Service',* b) tarea *'Create Entity Resolution Service'*

La Figura 73 ilustra el formulario de la tarea '*Select Attributes For Entity Resolution Service*', en donde se seleccionan los atributos del *Expected* '*Locations*' para identificar duplicados. En este caso se utilizan las *primary keys* del esquema.

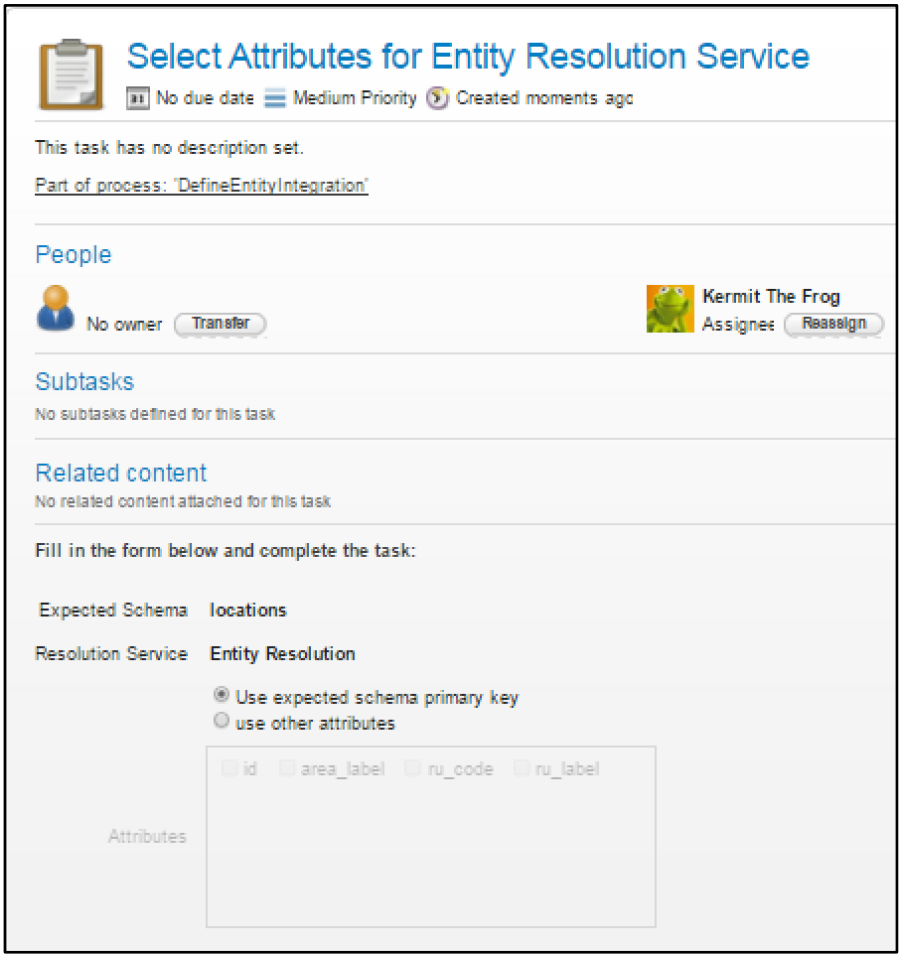


Figura 73 - Formulario tarea '*Select Attributes for Entity Resolution Service*'

Dado que el *Expected* '*Locations*' sólo tiene instanciadas métricas de granularidad '*Tabla'*, no se debe seleccionar qué granularidad va a considerar el servicio *Entity Fusion* y se pasa directamente a la ponderación de esas dos métricas. La Figura 74 muestra el formulario donde se realiza dicha ponderación y los valores de ponderación que se le indican a cada métrica instanciada.

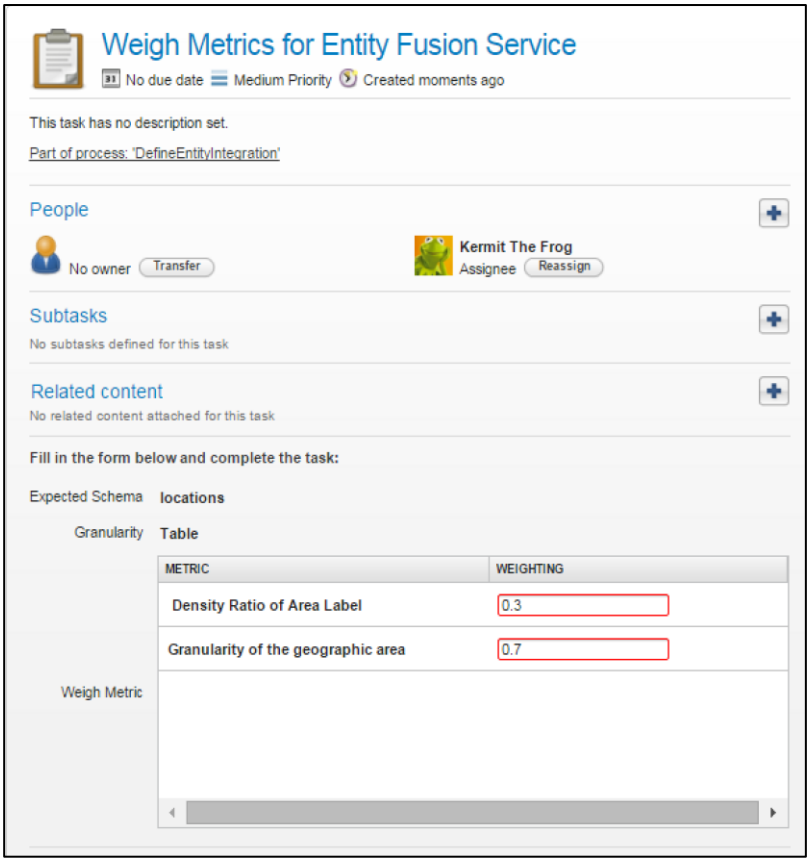


Figura 74 - Formulario tarea '*Select Attributes for Entity Resolution Service*'

6.5.6 Integrated Schemas

En este punto se describe cómo se realiza la integración de los *Expected Schemas* en el proceso de Configuración.

A continuación se presentan cada una de las tablas que componen el *Integrated Schema* las cuales incluyen la definición de sus atributos y el mapeo con los *Expected Schemas*. También se detalla las *foreign keys* entre las *Integrated Tables* y los *joins* entre los *Expected Schemas*.

Accidents

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Expected Schema*
id	char(20)	✓	Expected_Accidents -> id
date	Date		Expected_Accidents -> date
location_id	char(20)		Expected_Accidents -> location_id

Vehicles

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Expected Schema*
accident_id	char(20)	✓	Expected_Vehicles -> accident_id
reference	integer	✓	Expected_Vehicles -> reference
type_id	Integer		Expected_Vehicles -> type_id
type_label	char(20)		Expected_Vehicles -> type_label
driver_sex_id	Integer		Expected_Vehicles -> driver_sex_id
driver_age	Integer		Expected_Vehicles -> driver_age
age	Integer		Expected_Vehicles -> age

Casualties

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Expected Schema*
accident_id	char(20)	✓	Expected_Casualties -> accident_id
vehicle_reference	Integer	✓	Expected_Casualties -> vehicle_reference
reference	Integer	✓	Expected_Casualties ->reference
class_id	Integer		Expected_Casualties -> class_id
class_label	char(20)		Expected_Casualties -> class_label
sex_id	Integer		Expected_Casualties -> sex_id
age	Integer		Expected_Casualties -> age
severity_id	Integer		Expected_Casualties -> severity_id
severity_label	char(20)		Expected_Casualties-> label
cost	float		Expected_Costs->cost

Dates

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Expected Schema*
id	Integer	✓	Expected_Dates -> id
month_id	Integer		Expected_Dates -> month_id
month_label	char(20)		Expected_Dates -> month_label
year	Integer		Expected_Dates -> year
day_of_week_id	Integer		Expected_Dates -> day_of_week_id
day_of_week_label	char(20)		Expected_Dates -> day_of_week_label

Locations

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Expected Schema*
id	char(20)	✓	Expected_Locations -> id
area_label	char(255)		Expected_Locations -> area_label
ru_code	char(20)		Expected_Locations -> ru_code
ru_label	char(100)		Expected_Locations -> ru_label

Foreigns Keys

Integrated Schema Origen	*Integrated Schema* Destino
Integrated_Accidents -> date	Integrated_Dates -> id
Integrated_Accidents -> location_id	Integrated_Locations -> id
Integrated_Vehicles -> accident_id	Integrated_Accidents -> id
Integrated_Casualties -> accident_id	Integrated_Vehicles -> accident_id
Integrated_Casualties -> vehicle_reference	Integrated_Vehicles -> reference

Joins Integrated Schema

***Expected Schema* 1**	***Expected Schema* 2**
Expected_Casualties-> accident_id, vehicle_reference, reference	Expected_Costs -> accident_id, vehicle_reference, casualty_reference

6.5.7 DW Schema

En este punto se definen las tablas que componen el *DW Schema* y se le asigna el nombre "DW_DEMO" al mismo, dicho nombre será el nombre de la Base de Datos del DW que generará el Proceso de Carga.

A continuación se presenta la definición de la tabla de hechos *Accidents* y de las tablas de dimensiones *Casualties*, *Vehicles*, *Date* y *Locations*. También se describen las *foreign keys* a definir entre las tablas del DW y los *joins* entre las tablas de *Integrated Schemas* para conformar las tablas del DW.

Fact Table Accidents

Atributos:

Nombre Atributo	**Tipo**	**PK**	**Mapeo con *Integrated Schema***
id	char(20)	✓	Integrated_Accidents -> id
date	Date	✓	Integrated_Accidents -> date
casualty_id	Integer	✓	Integrated_Casualties -> reference
vehicle_id	Integer	✓	Integrated_Casualties -> vehicle_reference
location_id	char(20)	✓	Integrated_Accidents -> location_id
cost_per_casualty	float		Integrated_Casualties->cost

Dimension Vehicles

Atributos:

Nombre Atributo	**Tipo**	**PK**	**Mapeo con *Integrated Schema***
accident_id	char(20)	✓	Integrated_Vehicles -> accident_id
reference	integer	✓	Integrated_Vehicles -> reference
type_id	Integer		Integrated_Vehicles -> type_id

type_label	char(20)		Integrated_Vehicles -> type_label
driver_sex_id	Integer		Integrated_Vehicles -> driver_sex_id
driver_age	Integer		Integrated_Vehicles -> driver_age
age	Integer		Integrated_Vehicles -> age

Dimension Casualities

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Integrated Schema*
accident_id	char(20)	✓	Integrated_Casualties -> accident_id
vehicle_reference	Integer	✓	Integrated_Casualties -> vehicle_reference
reference	Integer	✓	Integrated_Casualties ->reference
class_id	Integer		Integrated_Casualties -> class_id
class_label	char(20)		Integrated_Casualties -> class_label
sex_id	Integer		Integrated_Casualties -> sex_id
age	Integer		Integrated_Casualties -> age
severity_id	Integer		Integrated_Casualties -> severity_id
severity_label	char(20)		Integrated_Casualties -> severity_label

Dimension Dates

Atributos:

Nombre Atributo	Tipo	PK	Mapeo con *Integrated Schema*
id	Integer	✓	Integrated_Dates -> id
month_id	Integer		Integrated_Dates -> month_id
month_label	char(20)		Integrated_Dates -> month_label
year	Integer		Integrated_Dates -> year
day_of_week_id	Integer		Integrated_Dates -> day_of_week_id
day_of_week_label	char(20)		Integrated_Dates -> day_of_week_label

Dimension Locations

Atributos:

Nombre Atributo	**Tipo**	**PK**	**Mapeo con *Integrated Schema***
id	char(20)	✓	Integrated_Locations -> id
area_label	char(255)		Integrated_Locations -> area_label
ru_code	char(20)		Integrated_Locations -> ru_code
ru_label	char(100)		Integrated_Locations -> ru_label

Foreigns Keys

DW Table Origen	**DW Table Destino**
DW_Accidents -> date	DW_Dates -> id
DW_Accidents -> location_id	DW_Locations -> id
DW_Accidents -> (id, casuality_id, vehicle_id)	DW_Casualities -> (accident_id, reference, vehicle_reference)
DW_Accidents -> (id, vehicle_id)	DW_Vehicles -> (accident_id, reference)

DW Joins

***Integrated Schema* 1**	***Integrated Schema* 2**
Integrated_Accidents-> id	Integrated_Casualties -> accident_id

6.5.8 Métricas de Calidad a aplicar sobre el WW resultado.

Al final del proceso de Configuración se define el Modelo de Calidad sobre el WW resultado. Por lo tanto, al igual que para el caso de la calidad definida sobre los datos recién extraídos de la web, primero se seleccionan las tablas del DW (ver Figura 75) a las cuales se desea instanciar métricas de calidad.

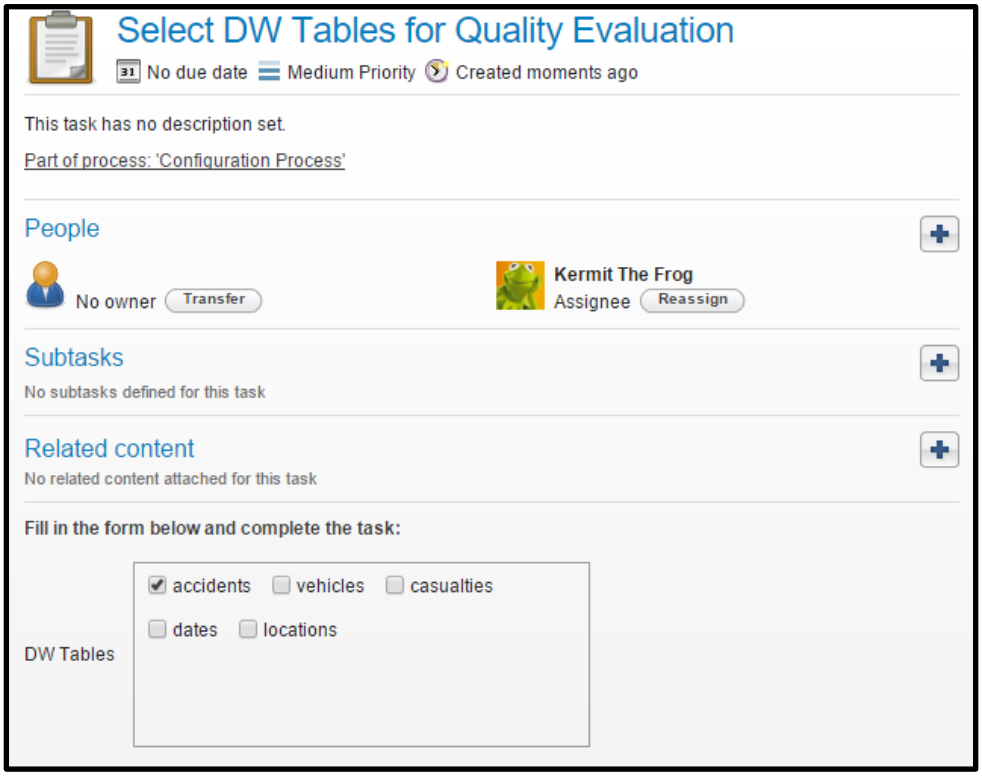


Figura 75- Formulario de Tarea '*Select Expected Schemas For Quality Evaluation*'

Métrica a Instanciar sobre la tabla del DW *Accidents*:

1) Metric Name: Relating cost to accident casualties
   Description: Indicates if the casualty cost in an accident is proper for the casualty type
   Factor: Semantic Correctness
   Dimension: Accuracy
   Web Service: http://localhost:8080/DataServices/accidentCostCorrectness
   Granularity: Cell
   Attribute: cost_per_casualty

Figura 76 - Tarea '*Instantiate Metric*' para una tabla del DW.

6.6 Descripción del Proceso de Carga

En este punto se describen los pasos necesarios para realizar la carga del WW.

Lo primero que se debe realizar es seleccionar la configuración que se desea cargar a partir del *combo box* del formulario que se ilustra en la Figura 77. Como se mencionó antes, la configuración a cargar se identifica por la concatenación del identificador que tiene en la Base de Metadatos de Configuración y el nombre que se le asignó al comienzo del Proceso de Configuración.

También, en esta tarea se selecciona el *checkbox* '*Show Conflicts and Errors*' con el fin de que durante el Proceso de Carga se desplieguen los errores y conflictos de la carga en tareas manuales.

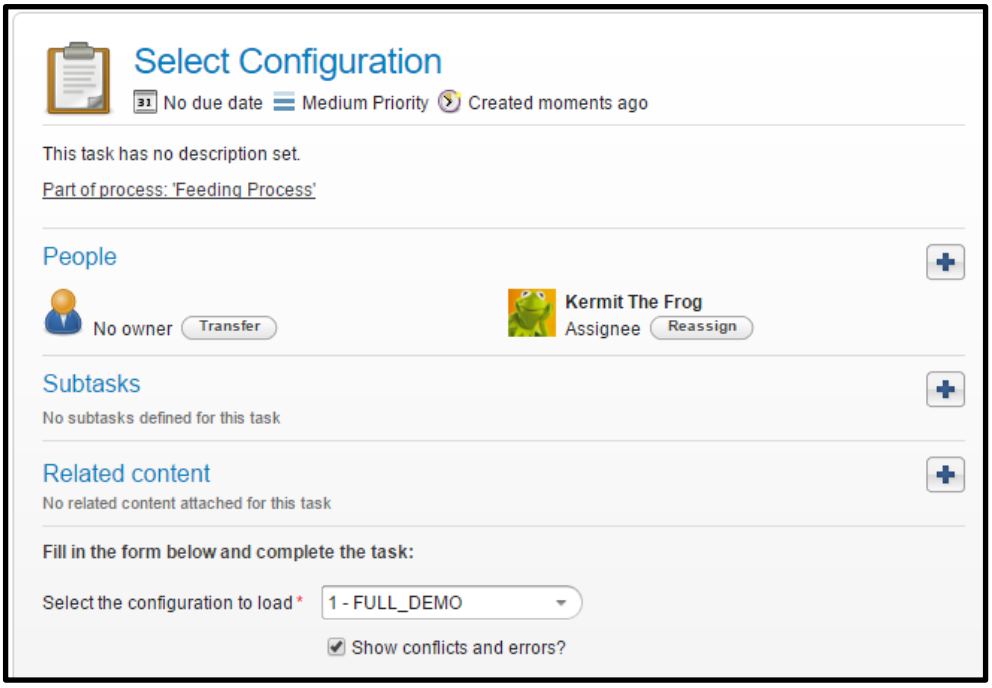


Figura 77 - Formulario de tarea '*Select Configuration*'

Una vez seleccionada la configuración, se crea la Base de Datos Intermedia y la Base de Datos del DW. El nombre de la base de datos Intermedia se corresponde con el nombre que se le asignó a la configuración (ver Figura 78) y el de la Base de Datos del WW el nombre que se le asignó al *DW Schema* (ver Figura 79). También se crea la base de Metadatos de Calidad que alojará todos los valores de calidad tanto medidos como propagados y cuyo nombre se corresponde también con el nombre que se le asignó a la configuración (ver Figura 80).

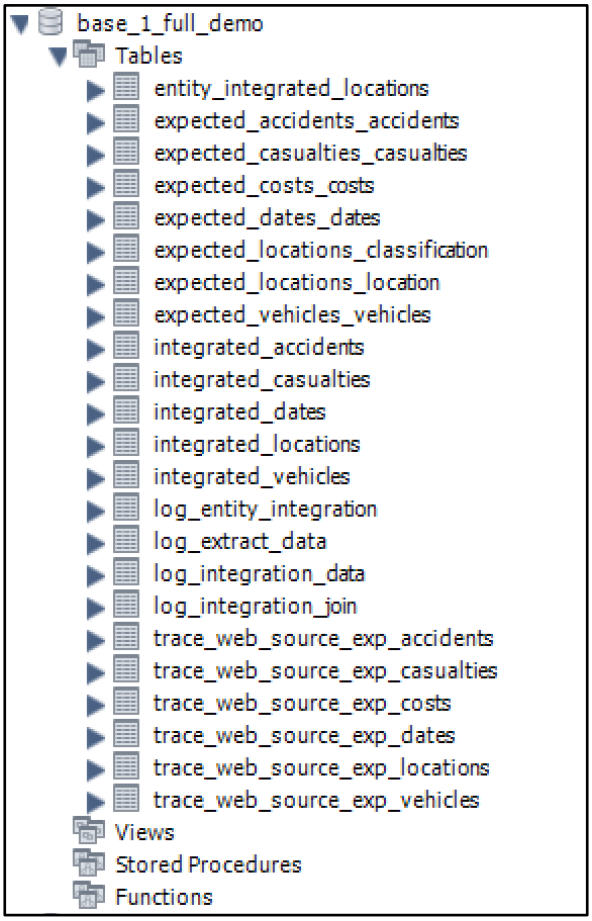


Figura 78 - Base de Datos Intermedia

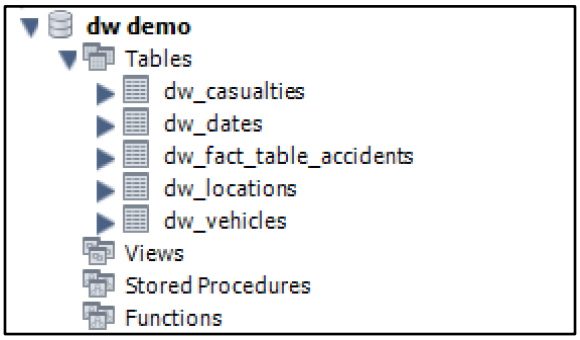


Figura 79 - Base del DW

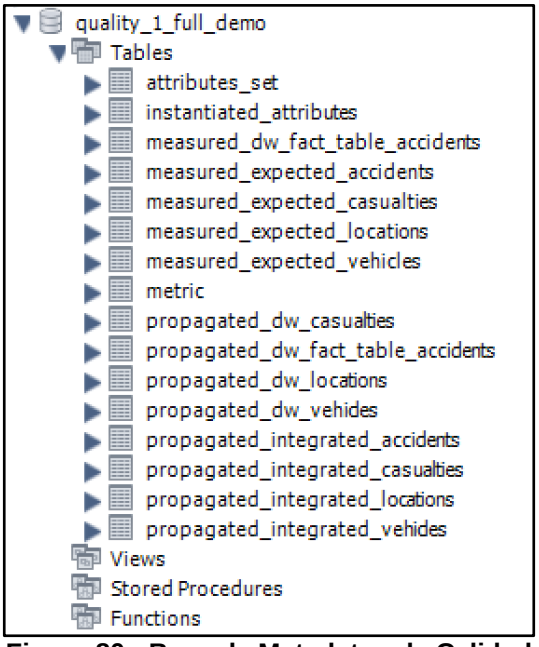


Figura 80 - Base de Metadatos de Calidad

Una vez se realizó la extracción de los datos y se cargaron los mismos en las *Expected Tables* se ejecuta la tarea '*Show Errors of Extracted Data*' en caso de que se produjeran errores al momento de la carga de los datos. La Figura 81 ilustra dicha tarea en donde se aprecia que se produjeron errores de clave duplicada en la tabla **expected_locations_locations**; esto se debe a que dicha *Expected Table* es cargada a partir de la FDW '*accidents*' en la cual distintos accidentes ocurren en la misma locación. La Figura 82 ilustra la tabla de la Base de Datos Intermedia que almacena esta información.

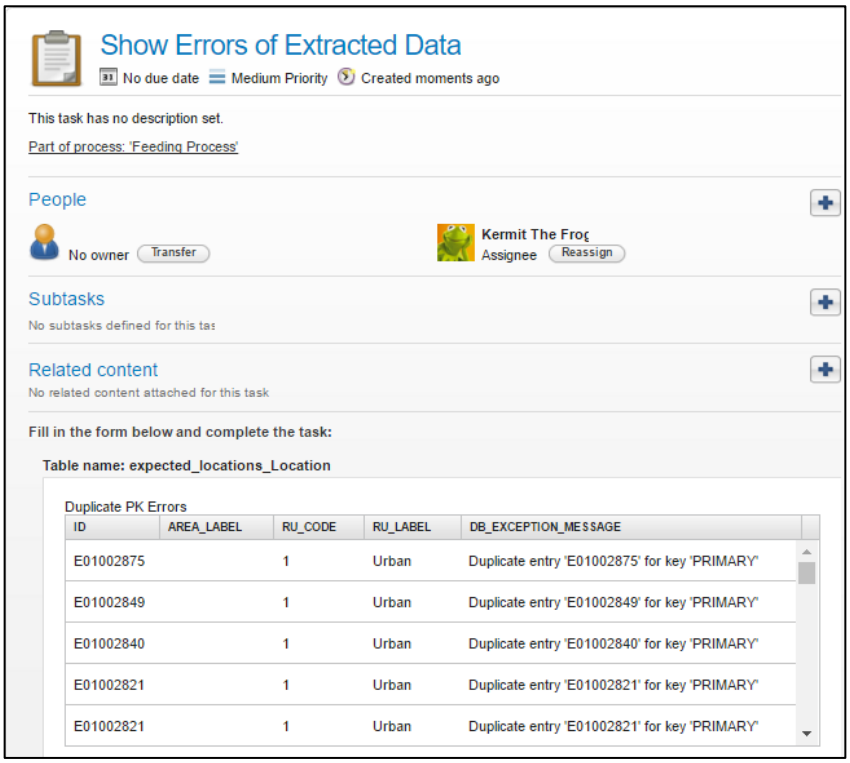
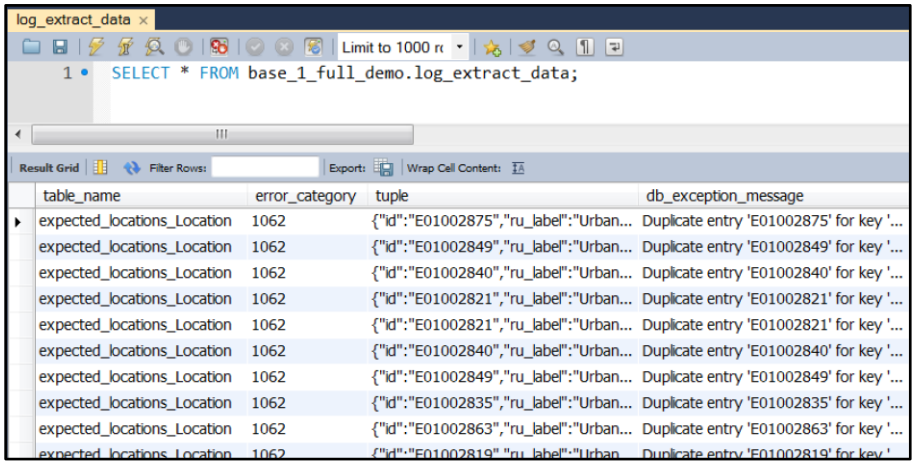


Figura 81 - Tarea '*Show Errors of Extracted Data*'



table_name	error_category	tuple	db_exception_message
expected_locations_Location	1062	{"id":"E01002875","ru_label":"Urban...	Duplicate entry 'E01002875' for key '...
expected_locations_Location	1062	{"id":"E01002849","ru_label":"Urban...	Duplicate entry 'E01002849' for key '...
expected_locations_Location	1062	{"id":"E01002840","ru_label":"Urban...	Duplicate entry 'E01002840' for key '...
expected_locations_Location	1062	{"id":"E01002821","ru_label":"Urban...	Duplicate entry 'E01002821' for key '...
expected_locations_Location	1062	{"id":"E01002821","ru_label":"Urban...	Duplicate entry 'E01002821' for key '...
expected_locations_Location	1062	{"id":"E01002840","ru_label":"Urban...	Duplicate entry 'E01002840' for key '...
expected_locations_Location	1062	{"id":"E01002849","ru_label":"Urban...	Duplicate entry 'E01002849' for key '...
expected_locations_Location	1062	{"id":"E01002835","ru_label":"Urban...	Duplicate entry 'E01002835' for key '...
expected_locations_Location	1062	{"id":"E01002863","ru_label":"Urban...	Duplicate entry 'E01002863' for key '...
expected_locations_Location	1062	{"id":"E01002819","ru_label":"Urban...	Duplicate entry 'E01002819' for key '...

Figura 82 - Tabla 'log_extracted_data'

Luego de cargadas las *Expected Tables*, se mide la calidad de datos definida sobre estas tablas y la integración de entidades empleando los valores de calidad del *Expected Schema* '*Locations*' (el único que tiene asociado dos FDW) con los servicios *Entity Resolution* y *Entity Fusion* que se definieron en la configuración. En la Figura 83, se ilustran los valores de calidad de las métricas de granularidad '*Tabla*' que se midieron sobre el *Expected* '*Locations*' para cada *Expected Table* asociada a dicho *Expected*, así como el valor de calidad ponderado para cada una, el cual se calcula a partir de la ponderación definida durante la configuración. Como se aprecia en la Figura 83, sólo las tuplas duplicadas que provienen de la tabla **expected_schema_locations_classifications** continúan en el proceso de Carga ya que dicha tabla posee mayor valor de calidad ponderado que la tabla **expected_schema_locations_location**.

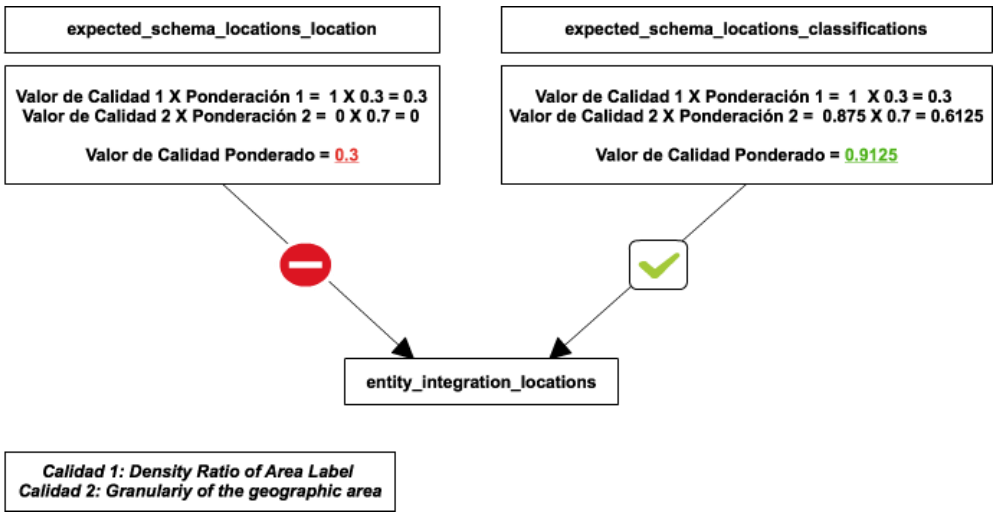


Figura 83 - Diagrama de Integración de Entidades del Expected Locations

En este caso el servicio *Entity Fusion* resuelve a partir de los metadatos de calidad todos los problemas de integración, por lo tanto, no se ejecuta la tarea manual '*Show Conflicts of Entity Integration*'.

A continuación, se ejecuta la integración de las *Expected Tables* para conformar las *Integrated Tables* y los errores producidos en la misma son mostrados al usuario en la tarea

manual '*Show Errors of Integrated Tables*' que se ilustra en la Figura 84. En este caso dos tuplas de la FDW '*costs*' no hicieron *join* con la FDW '***casualties'***, es decir dos costos de víctimas asociadas a un accidentes determinado no tienen su correspondiente víctima en el *Expected* '*casualties*'. En la Figura 84 se aprecian los dos *Expected* que no tienen una total correspondencia, en donde la columna '*Expected_Table_Missed*' indica cuál es la *Expected Table* que no contiene una tupla que se corresponda con otra en la *Expected Table* de la columna '*Expected_Table_Source*'; además se indica cual es la tupla que no hace *join* en la columna '***Tuple'***. Mientras que la Figura 85 ilustra cuál tabla de la Base de Datos Intermedia que almacena esta información.

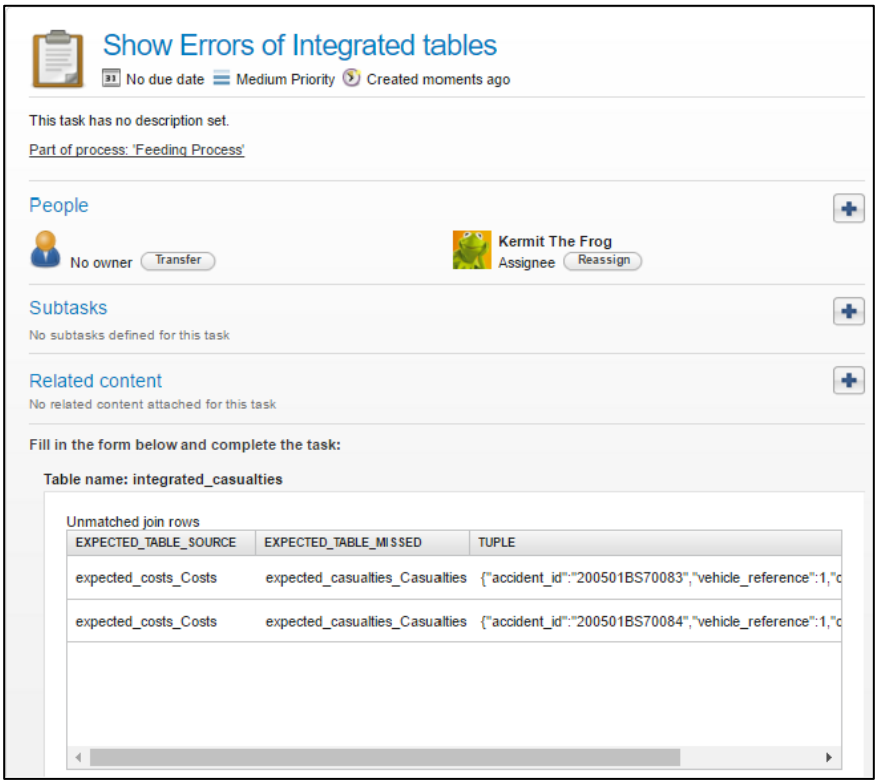


Figura 84 - Tarea '*Show Errors of Integrated Tables*'

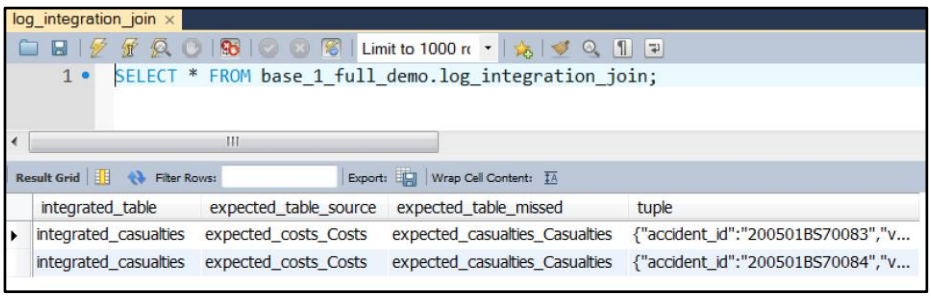


Figura 85 - Tabla '*log_integration_join*'

Ya cargadas las *Integrated Tables*, se ejecuta la carga de las tablas de dimensiones del DW. Para la carga de la única tabla de hechos *Accidents* que tiene el DW, se ejecuta la tarea '*Confirm Query To Load DW*' (ver Figura 86) en donde se muestra al usuario la consulta SQL que carga la tabla **dw_fact_table_accidents** y el usuario tiene la posibilidad de modificar dicha consulta.

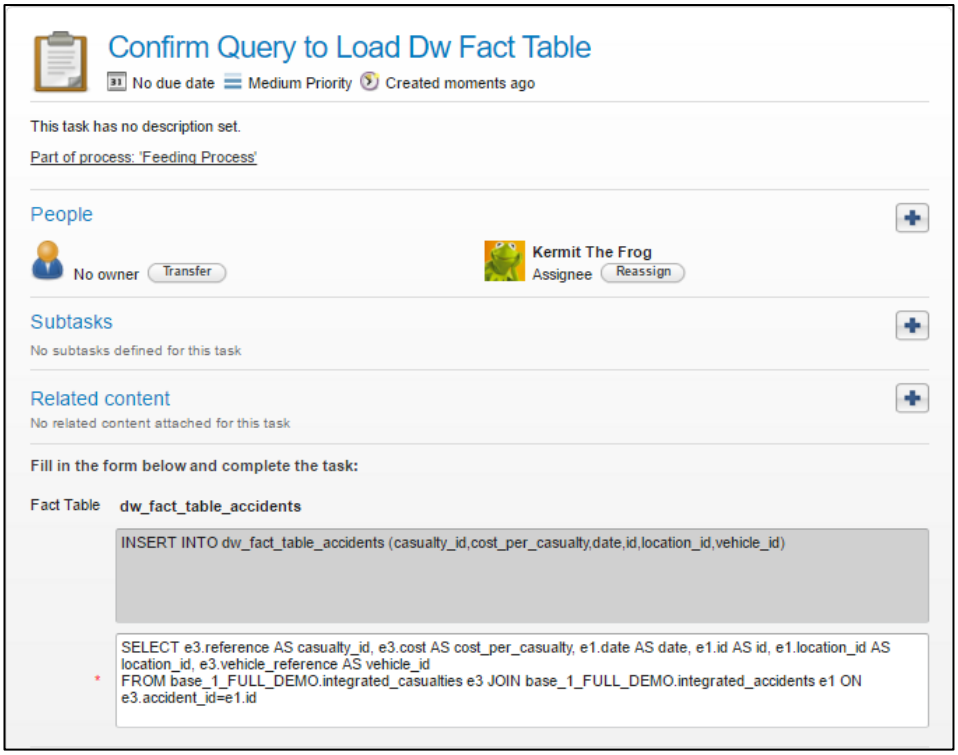
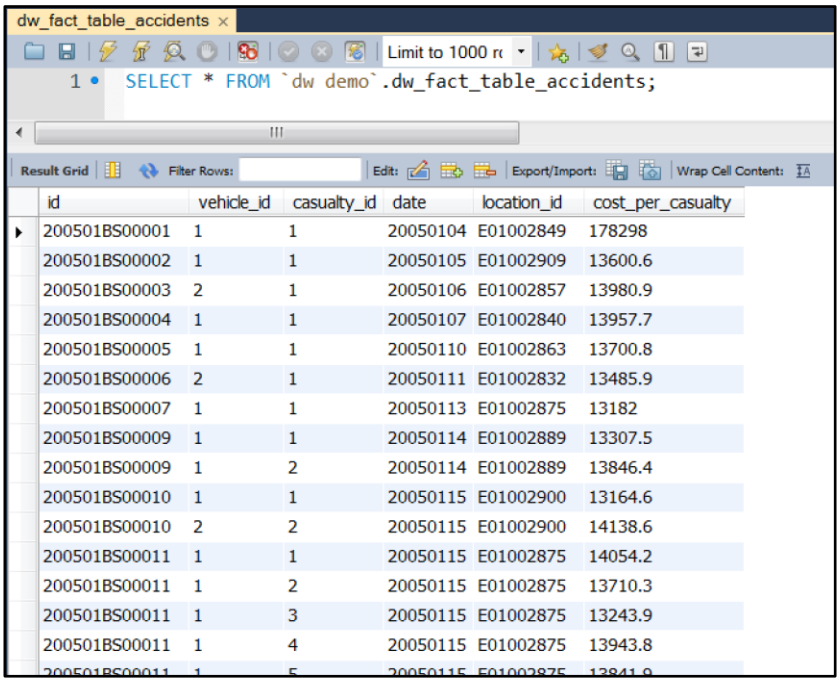


Figura 86 - Tarea 'Confirm Query to Load DW Tables'

Una vez confirmada la consulta, el DW queda completamente cargado y se realiza la medición de la calidad definida para el mismo. En la Figura 87, se pueden apreciar cómo queda cargada la tabla de hechos del DW resultado.



id	vehicle_id	casualty_id	date	location_id	cost_per_casualty
200501BS00001	1	1	20050104	E01002849	178298
200501BS00002	1	1	20050105	E01002909	13600.6
200501BS00003	2	1	20050106	E01002857	13980.9
200501BS00004	1	1	20050107	E01002840	13957.7
200501BS00005	1	1	20050110	E01002863	13700.8
200501BS00006	2	1	20050111	E01002832	13485.9
200501BS00007	1	1	20050113	E01002875	13182
200501BS00009	1	1	20050114	E01002889	13307.5
200501BS00009	1	2	20050114	E01002889	13846.4
200501BS00010	1	1	20050115	E01002900	13164.6
200501BS00010	2	2	20050115	E01002900	14138.6
200501BS00011	1	1	20050115	E01002875	14054.2
200501BS00011	1	2	20050115	E01002875	13710.3
200501BS00011	1	3	20050115	E01002875	13243.9
200501BS00011	1	4	20050115	E01002875	13943.8

Figura 87 - Consulta sobre tabla de hecho del DW obtenido

La Figura 88 ilustra los valores de calidad registrados en la tabla **measured_expected_locations** (ver (a)) y los valores de calidad propagados en la tabla

propagated_integrated_locations (ver (b)). Mientras que la Figura 89 muestra los valores de calidad calculados sobre la tabla de hechos *Accidents* del DW que indican si el costo asociado a la víctima es acorde a su tipo (*pedestrian, driver* o *passenger*).

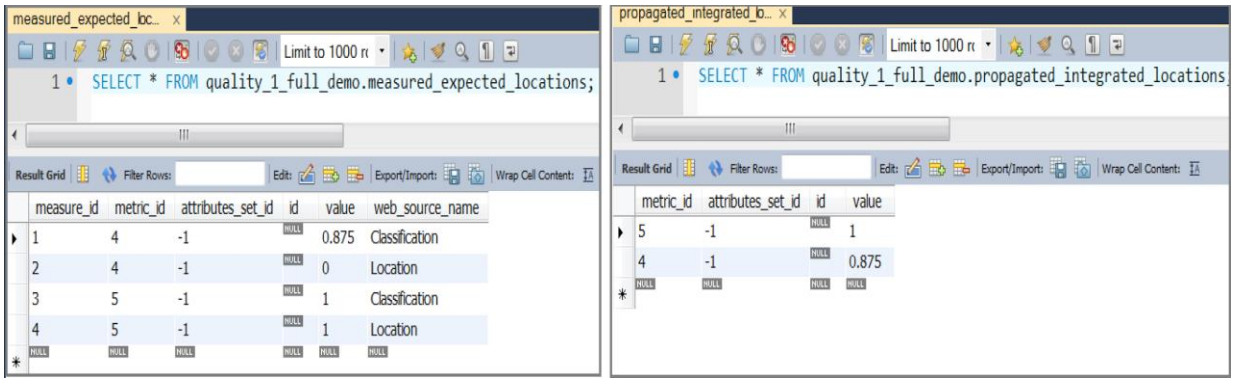


Figura 88 - Tablas que almacenan valores de calidad:
a) measured_expected_locations, b) propagated_integrated_locations

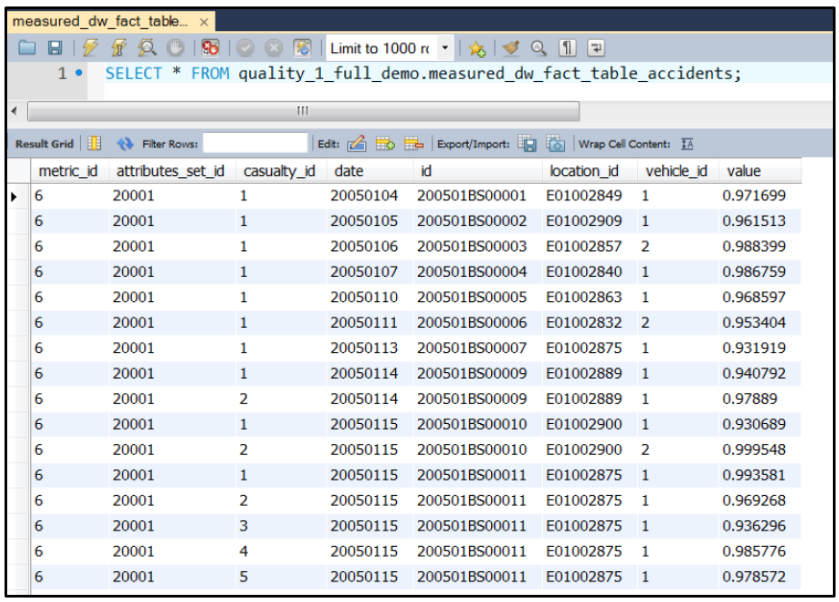


Figura 89 - Tabla que almacena valores de calidad medidos sobre el DW

7 - Verificación

En este capítulo se presenta la verificación realizada sobre los prototipos implementados, detallando algunas de las pruebas realizadas especificando en cada caso los resultados esperados y los resultados obtenidos.

7.1 Contexto

La verificación se realizó en etapas, en una primera etapa se verificó el Proceso de Configuración y en otra etapa se verificó el Proceso de Carga. En la verificación de ambos procesos se realizaron pruebas unitarias por funcionalidad de manera incremental hasta cubrir la totalidad de las funcionalidades de los mismos. Se tomó la decisión de realizar la verificación siguiendo el orden del proceso de construcción de un WW, dado que cada actividad del proceso depende de su actividad predecesora. En cada actividad se corroboró que los datos y los componentes mostrados en pantalla sean los adecuados, que los datos sean persistidos en la base y que al completar la actividad el flujo del proceso continúe según lo esperado.

En el Proceso de Configuración, se realizaron pruebas exhaustivas sobre los aspectos de calidad y la mejora de la integración de entidades ya que son las funcionalidades críticas del presente proyecto. También se realizaron más pruebas sobre aquellas actividades que incluían mejoras y/o correcciones al proceso existente para comprobar que se obtenían los resultados esperados.

Respecto a la verificación del Proceso de Carga, la metodología realizada fue análoga al proceso de Configuración realizándose pruebas más exhaustivas sobre los requerimientos críticos del presente proyecto. También se debieron realizar pruebas sobre los servicios implementados aunque los mismos no formarán parte de la solución para alcanzar a ejecutar un proceso de carga completo y el caso de estudio del presente informe.

Luego de obtener resultados satisfactorios en la verificación de ambos proceso, se verificó la solución utilizando un caso de estudio real de forma de comprobar la factibilidad de la solución.

Es importante destacar que al momento de detectar un error se evaluó la severidad del mismo y el costo/beneficio de corregirlo, priorizando en todo momento los errores asociados a los requerimientos mínimos. Además, dado que el prototipo implementado es una prueba de concepto, se dejaron como errores conocidos algunos incidentes detectados asociados a las limitaciones de la interfaz de Activiti [6] como control de campos, etc., o aquellos incidentes que para resolverse requiere complejizar el modelo agregando actividades o compuertas. A modo de ejemplo, si el usuario experto no define en el *DW Schema* ninguna tabla de hechos, se produce un error que no permite continuar la ejecución del proceso de Configuración. Una posible solución del mismo es incluir una actividad específica para la definición de las tablas de hechos que obligue al usuario a definir al menos una de ellas; pero como todo DW contiene al menos una tabla de hechos se deja al usuario experto la responsabilidad de dicho incidente.

A continuación se detallan algunas de las pruebas realizadas sobre los prototipos, separándolas entre las pruebas realizadas sobre el Proceso de Configuración y las pruebas realizadas en el Proceso de Carga. El resto de las pruebas realizadas se encuentra en el Anexo E - Casos de Prueba.

7.2 Pruebas sobre el Proceso de Configuración Extendido

Caso de Prueba 1

Caso de prueba	Instanciación de Métricas
Descripción	Verificar que se almacenen en la base correctamente los metadatos de la instanciación de una métrica de calidad de granularidad '*Celda*' y otra de granularidad '*Tabla*'. La Figura 90 muestra del lado izquierdo la instanciación de la métrica '*Wrong Type*' de granularidad '*Celda'* sobre el '*Expected Schema 1'* y del lado derecho la instanciación de la métrica 'Ratio' sobre el mismo Expected pero en este caso la métrica es de granularidad '*Tabla'*.
Resultado Esperado	Se almacenan todas las definiciones correctamente
Resultado Obtenido	Ver Figura 91

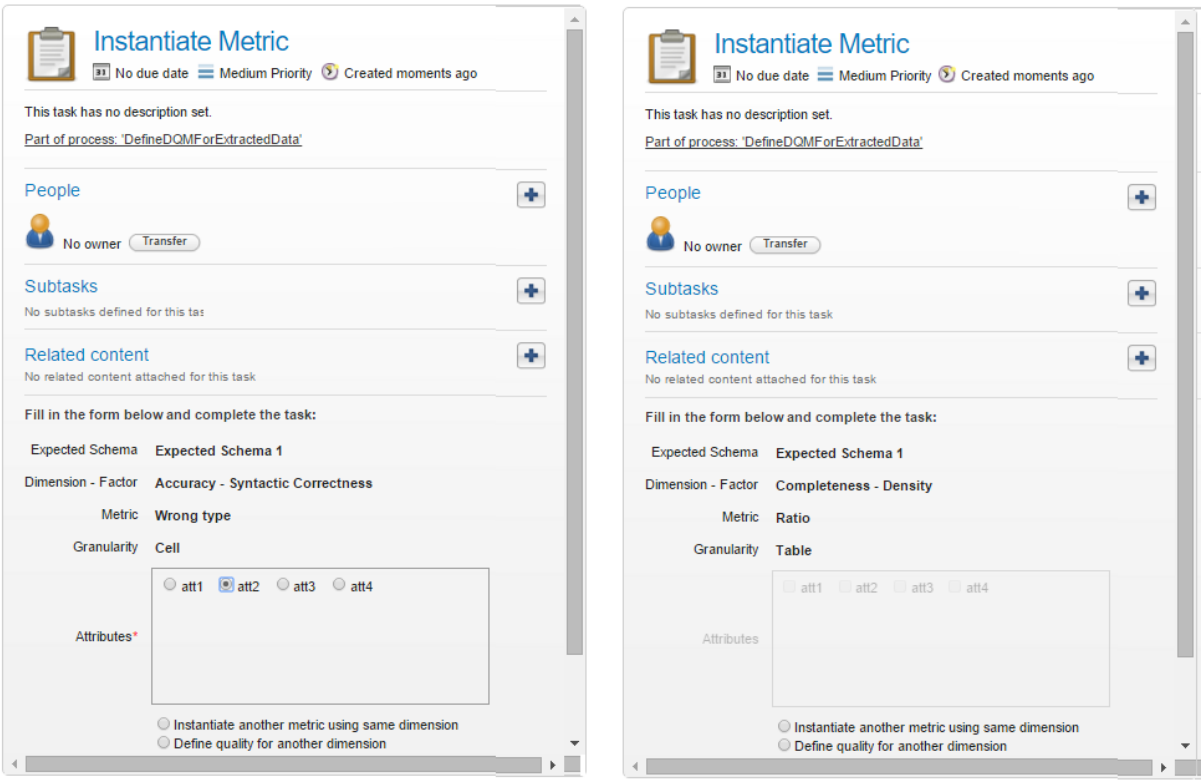
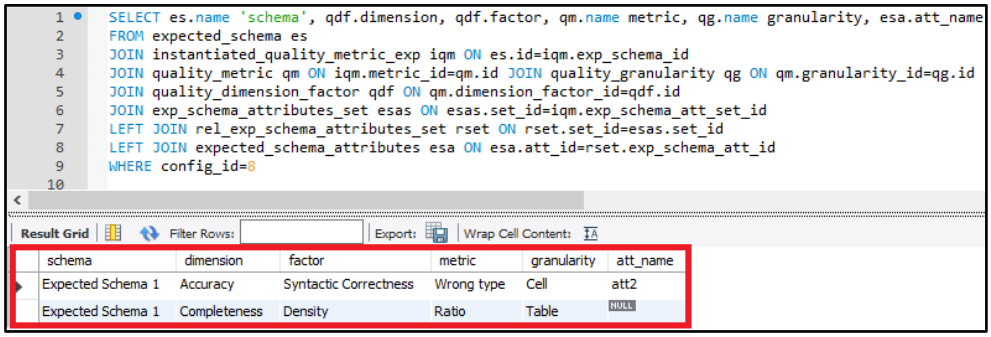


Figura 90 - Caso de Prueba 1



```
SELECT es.name 'schema', qdf.dimension, qdf.factor, qm.name metric, qg.name granularity, esa.att_name
FROM expected_schema es
JOIN instantiated_quality_metric_exp iqm ON es.id=iqm.exp_schema_id
JOIN quality_metric qm ON iqm.metric_id=qm.id JOIN quality_granularity qg ON qm.granularity_id=qg.id
JOIN quality_dimension_factor qdf ON qm.dimension_factor_id=qdf.id
JOIN exp_schema_attributes_set esas ON esas.set_id=iqm.exp_schema_att_set_id
LEFT JOIN rel_exp_schema_attributes_set rset ON rset.set_id=esas.set_id
LEFT JOIN expected_schema_attributes esa ON esa.att_id=rset.exp_schema_att_id
WHERE config_id=8
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

schema	dimension	factor	metric	granularity	att_name
Expected Schema 1	Accuracy	Syntactic Correctness	Wrong type	Cell	att2
Expected Schema 1	Completeness	Density	Ratio	Table	NULL

Figura 91 - Resultado Obtenido de Caso de Prueba 1

Caso de Prueba 2

Caso de prueba	Definición de servicios *Entity Resolution* y *Entity Fusion*
Descripción	Se define sobre el 'Expected Schema 1' asociado a más de una FDW el servicio *Entity Resolution* '*Entity Resolution Service*' en el cual se identifican las tuplas por la *primary key* del *Expected*. También se define el servicio *Entity Fusion* '*Entity fusion Service*'. (Ver Figura 92)
Resultado Esperado	Se almacenan todas las definiciones correctamente
Resultado Obtenido	Ver Figura 93

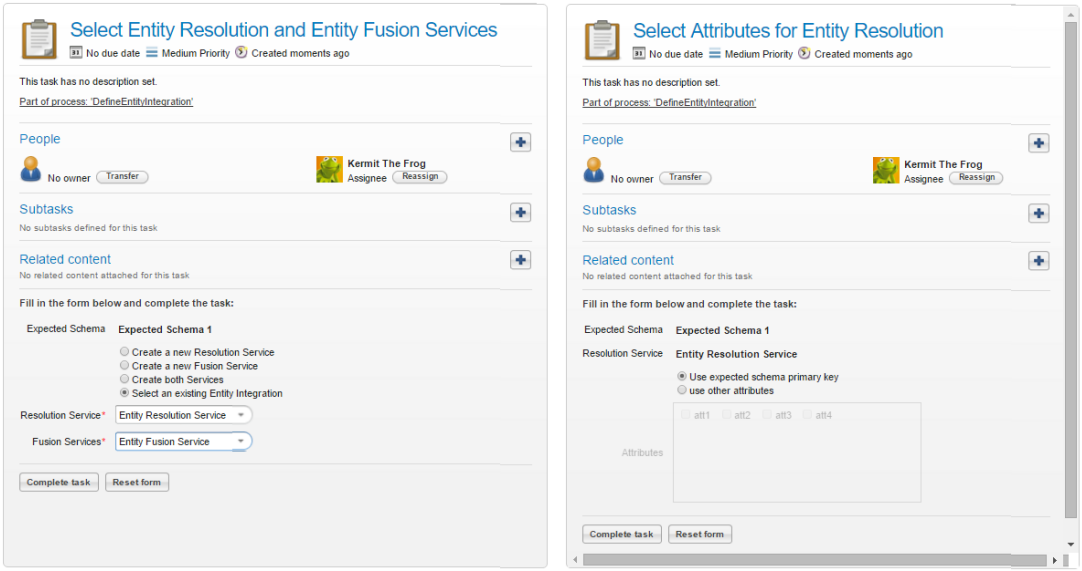


Figura 92 - Caso de Prueba 2

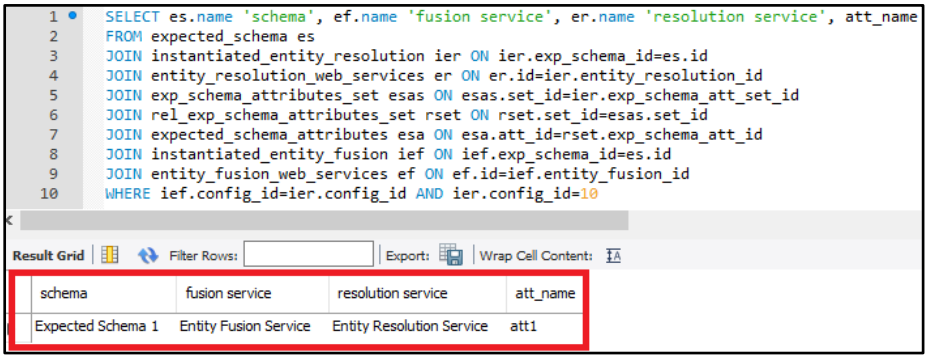


Figura 93 - Resultado Obtenido de Caso de Prueba 2

7.3 Pruebas sobre el Proceso de Carga

Caso de Prueba 3

Caso de prueba	Manejo de errores en tarea *'Confirm query To Load DW Fact Table'*
Descripción	Se modifica la sintaxis de la sentencia SQL que realiza la carga de la tabla de hechos del DW para que genere una excepción en el motor de la base de datos.
Resultado Esperado	Se lanza un mensaje de error indicando el error del motor de la base de datos.
Resultado Obtenido	La Figura 94 ilustra ejemplos de sintaxis erróneas y sus respectivos mensajes desplegados.

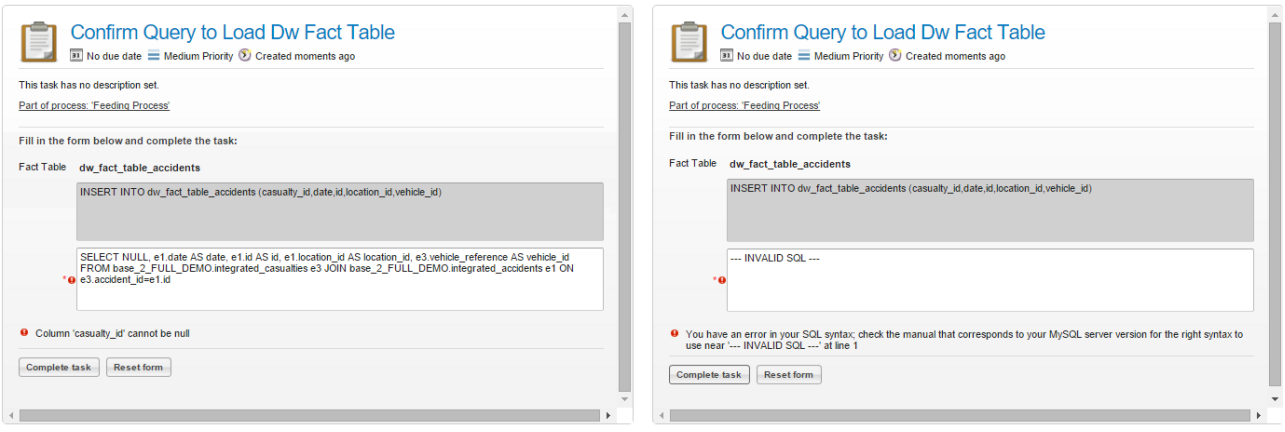


Figura 94 - Caso de Prueba 3

8 - Conclusión

En este capítulo se presentan las conclusiones finales del presente proyecto así como también el trabajo a futuro planteado y las posibles extensiones al mismo.

8.1 Conclusiones

Como conclusión se destaca que se cumplieron todos los objetivos propuestos al comienzo del proyecto. Se logró extender el Proceso de Construcción de un WW empleando BPM y enfocándose en los aspectos de calidad del mismo.

A su vez, se alcanzó a implementar un prototipo que realiza la prueba de concepto utilizando las tecnologías esperadas, demostrando que es posible construir un WW enfocado en la calidad automatizando la mayoría de las tareas que conciernen al proceso, habiendo definido previamente los metadatos que corresponden a la configuración del mismo.

Se consiguió ejecutar un caso de estudio con éxito, a pesar que se debieron generar algunas FDW con las que no se contaban. Cabe resaltar en este punto, que los datos abiertos actualmente se publican a nivel estadístico y no con el detalle que se requiere para construir y explotar un DW, por lo que se dificulta encontrar casos de estudio que puedan ser empleados en el prototipo desarrollado. Igualmente, se espera que en los próximos años se publique más información y que esto no sea un inconveniente al momento de construir un WW con BPMS.

En el aspecto académico, el proyecto requirió profundizar el aprendizaje acerca de *Data Warehouse*, Procesos de Negocio y Calidad de Datos. También implicó investigar y aprender sobre la herramienta *Activiti Explorer* para el desarrollo de la interfaz web del prototipo, y aunque no forman parte de la solución, se investigaron e implementaronaplicaciones que publican servicios Rest mediante Spring.

Durante el proyecto, se investigaron trabajos y/o herramientas similares que consideran el manejo de la calidad de datos en la construcción de un WW. En todos los casos la calidad se evalúa sobre las fuentes y/o el DW, no incluyendo la propagación de la calidad de los datos durante el proceso como por ejemplo la herramienta *Pentaho Data Integration* [33] o la herramienta *Talend*[34]. Además, dichas herramientas no cuentan con la posibilidad de resolver conflictos manualmente ni utilizan metadatos de calidad para mejorar la integración de entidades. Por lo tanto, creemos que es beneficiosa la utilización del prototipo implementado en un ambiente de producción para conocer y controlar la calidad de los datos de un WW.

Respecto a la ejecución del proyecto si bien presentó obstáculos propios de un proyecto de grado como profundizar en los conceptos antes mencionadas, resultó motivante para los integrantes del mismo dado que la temática es interesante y novedosa. Durante el proyecto se debieron emplear conocimiento obtenidos durante la carrera de ingeniería y la realización del mismo aportó la resolución de un problema de alta complejidad como es la construcción de un WW enfocado en la calidad empleando BPMS.

8.2 Trabajo a Futuro

El proyecto deja tres líneas principales a seguir trabajando en el futuro:

● **Verificación**
Si bien se realizó una verificación amplia, es necesario continuar probando la herramienta implementada con otros casos de estudio para lograr cubrir todos los casos, sobre todo la integración de los datos y la propagación de la calidad de los mismos y realizar pruebas con usuarios finales con el fin de aportar otra visión del producto realizado.

● **Mejoras en la interfaz**
A pesar de las mejoras que se realizaron sobre la interfaz del Proceso de Configuración anterior se debe seguir trabajando arduamente en esta línea evaluando la posibilidad de implementar otra interfaz web que utilice el motor de Activiti, apuntando sobre todo a la amigabilidad de la interfaz.

● **Propagación de la Calidad y Traza de los Datos**
Se debe continuar trabajando con la propagación de la calidad ya que para simplificar la solución propuesta se tomaron algunas consideraciones al momento de propagar los datos, que no incluyen todos los casos posibles.

También sería útil incluir la traza total de los datos (desde las FDW hasta las tablas de hechos y dimensiones del DW) ya que actualmente sólo se cuenta con la traza de las *Expected Tables*, que fue añadida meramente para resolver algunos aspectos de la propagación de la calidad y la mejora de la integración de entidades.

Referencias

[1] Weske M., Business Process Management: Concepts, Languages, Architectures, Springer, 2007.

[2] Delgado A., Marotta A., González L., "Towards the construction of quality-aware Web Warehouses with BPMN 2.0 Business Processes". In procs. IEEE 8th International Conference on Research Challenges in Information Science, RCIS 2014, Marrakech, Morocco, 2014.

[3] Construcción de Web Warehouse enfocados en la calidad con BPMS, Miguel Angel Mendiola, Fernando Alonso, Proyecto de Grado de la carrera Ingeniería en Computación 2015

[4] COAL - Instituto de Computación - Facultad de Ingeniería (UdelaR), https://www.fing.edu.uy/inco/grupos/coal , último acceso: Marzo 2016

[5] Concepción de Sistemas de Información - Instituto de Computación - Facultad de Ingeniería (UdelaR), https://www.fing.edu.uy/inco/grupos/csi/wiki/webspace/index.php/P%C3%A1gina_principal

[6] Activiti BPMS http://activiti.org , último acceso: Marzo 2016

[7] http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf, último acceso: Marzo 2016

[8] Spring, https://spring.io/guides/gs/consuming-rest/ , último acceso: Marzo 2016

[9] DataWarehouse, Adriana Marotta, INCO, FING, UdelaR, Curso 2014

[10] Delgado, A., and A. Marotta, "Automating the process of building flexible Web Warehouses with BPM Systems", Computing Conference (CLEI), 2015 Latin American, pp. 1-11, Oct, 2015.

[11] Taller de Gestión y Tecnologías de Procesos de Negocio (TBPM), Grupo COAL, InCo, FING, UdelaR, Curso 2015

[12] Fundamentals of BPM http://fundamentals-of-bpm.org/ , último acceso: Abril 2016

[13] BPMN 2.0, OMG, http://www.omg.org/spec/BPMN/2.0/ , último acceso: Marzo 2016

[14] Calidad de Datos, Adriana Marotta, Grupo CSI, INCO, FING, UdelaR, Curso 2013

[15] Carlo Batini - Monica Scannapieca, Data Quality Concepts, Methodologies and Techniques, Springer, 2006

[16] Larman, Craig, Applying UML and patterns, Prentice Hall, 2001

[17] Mendling J., Reijers H. A., van der Aalst W.M.P., Seven process modeling guidelines (7PMG), Information and Software Technology, 52 (2), (2010)

[18] MySQL https://www.mysql.com, último acceso: Marzo 2016

[19] van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A. "Workflow patterns". Distributed & Parallel Databases, 14 (3), (2003)

[20] Vaadin https://vaadin.com/home , último acceso: Marzo 2016

[21] BonitaSoft http://es.bonitasoft.com/ , último acceso: Marzo 2016

[22] https://data.gov.uk , último acceso: Marzo 2016

[23]https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/230590/stats19.pdf , último acceso: Marzo 2016

[24]http://data.dft.gov.uk/road-accidents-safety-data/Road-Accident-Safety-Data-Guide.xls, último acceso: Marzo 2016

[25] https://www.gov.uk/government/organisations/department-for-transport , último acceso: Marzo 2016

[26] https://www.ons.gov.uk/census/2011census , último acceso: Marzo 2016

[27]https://geoportal.statistics.gov.uk/Docs/Products/Rural-urban_classification_(2011)_of_lower_layer_super_output_areas_(2011)_E+W.zip , último acceso: Marzo 2016

[28]https://www.gov.uk/government/statistical-data-sets/ras60-average-value-of-preventing-road-accidents , último acceso: Marzo 2016

[29] van der Aalst W.M.P.,ter Hofstede A., Weske M., "Business Process Management: A Survey", In Proceedings Int. Conf. on Business Process Management (BPM), The Netherlands, 2003.

[30] BPMN Poster http://bpmb.de/poster , último acceso: Marzo 2016

[31] Marotta, A., González, L., Etcheverry, L., Rienzi, B., Ruggia, R., Serra, F., Martirena, E. Quality Management in Web Warehouses. Business Intelligence Applications and the Web: Models, Systems and Technologies, IGI Global, pp. 1-25, 2011.

[32] Anotaciones Spring, http://www.arquitecturajava.com/spring-mvc-y-anotaciones-ii/, último acceso: Abril 2016

[33] Pentaho Data Integration, http://www.pentaho.com/product/data-integration, último acceso: Abril 2016

[34] Talend, https://www.talend.com/, último acceso: Abril 2016