Instituto de Computación,
Facultad de Ingeniería,
UdelaR

2014

Construcción de Web Warehouse enfocados en la Calidad con BPMS

Informe final de Proyecto de Grado 2014

Estudiantes:

Miguel Angel Mendiola

Fernando Alonso

Tutores:

Andrea Delgado

Adriana Marotta

09/2015 Montevideo, Uruguay

Resumen

Las organizaciones hoy en día dependen fuertemente de sus Sistemas de Información que dan soporte a las actividades diarias en ellas. En los últimos años, esto incluye modelar sus procesos de negocio como parte de la Gestión por Procesos de Negocio (Business Process Management, BPM) y soportar la toma de decisiones a nivel gerencial a mediano y largo plazo mediante el estudio de datos históricos de la organización que nos provee por ejemplo un Sistema de Data Warehouse (DW).

BPM nos permite representar todas las actividades que hacen al día a día de la organización, por ejemplo, realización de tareas, responsabilidades sobre las mismas, gestión de eventos, notificaciones y alertas, entre otras, que hacen a los procesos de negocio de la entidad, sobre un flujo de control previamente establecido que dirige la ejecución del Sistema de Información basado en procesos.

Por otro lado, un DW provee soporte para la explotación de datos históricos (generados por los Sistemas de Información en la organización), de manera de entender, estudiando éstos, qué sucedió en el pasado para tomar decisiones a futuro, ya sea tendencias, detección de pérdidas, maximizar ganancias, entre otros. En particular, un Web Warehouse (WW) es un DW cuyos datos son obtenidos de la Web. Un aspecto a tener en cuenta en la construcción de cualquier DW refiere a la calidad de los datos de entrada, así como la calidad de los datos en el DW resultante.

El proyecto que se presenta, denominado "**Construcción de Web Warehouse enfocados en la calidad con BPMS"**, tendrá dos grandes pilares que son las áreas mencionadas anteriormente, pero con la motivación de poder unirlas y explotarlas simultáneamente. ¿Qué queremos decir con esto? Poder construir un WW considerando aspectos de calidad de datos, cuyo proceso de construcción (proceso de negocio) estará definido y será ejecutado, en una plataforma de soporte a Procesos de Negocio denominada Sistema BPM (BPMS).

Existen dos etapas bien definidas en la construcción de un WW que son: definir la configuración de datos para la construcción del WW y realizar la carga del mismo utilizando dicha configuración, así como también aspectos de calidad sobre los datos utilizados. Para esto se define en la primera etapa un Proceso de Configuración del WW, y en la segunda un Proceso de Carga del WW en el cual utilizando los metadatos provistos por el Proceso de Configuración, se realiza la construcción del WW. El Proceso de Configuración genera un *template* que puede ser utilizado para realizar la carga del WW tantas veces como se quiera.

Una vez finalizado el Proceso de Carga del WW, se podrá pasar a la etapa de explotación, donde el personal responsable podrá generar (mediante distintas herramientas) "cubos" que nos permitirán exponer y cruzar los datos de distintas maneras para poder analizarlos. El alcance del Proyecto no incluyó finalmente el aspecto de la consideración de la calidad de datos que se extraen de las fuentes ni del DW resultante, pero queda abierto para su extensión ya que se adjunta documentación técnica para poder continuarlo y extenderlo.

1 Introducción

En este capítulo se introduce la definición del Proyecto incluyendo su motivación y contexto, los objetivos planteados, los resultados esperados y obtenidos, así como una breve descripción del desarrollo del mismo.

Los resultados de este trabajo de Proyecto de Grado serán presentados por las tutoras en la Conferencia Latinoamericana de Informática (CLEI) a realizarse en Arequipa, Perú en octubre de este año. [15]

1.1 Motivación.

La cantidad de datos útiles que se publican en Internet es cada vez mayor, no sólo en el ámbito privado o académico, sino particularmente a nivel gubernamental. Estos datos son interesantes para la población en general y para las organizaciones o empresas que necesitan consumir servicios que provean dichos datos, por ejemplo sobre transporte, turismo, cultura, trámites, etc. El desafío más importante para las personas, organizaciones o empresas es cómo utilizar o saber explotar estos datos y es por esto que resulta importante poder brindar aplicaciones que extraigan los mismos y puedan reordenarlos de manera inteligente y práctica para poder analizarlos. [6]

Bajo este contexto, una herramienta del tipo de un Web Warehouse (WW) que permite utilizar datos que son obtenidos de la Web como por ejemplo datos abiertos gubernamentales, aporta a la toma de decisiones en diferentes áreas. El principal objetivo de estas aplicaciones es ser intermediarias entre las personas o Sistemas que publican los datos, y las personas o Sistemas que necesitan de ellos, procesándolos y agregando valor a los mismos. El procesamiento que mencionamos, incluye luego de la extracción de los datos, integración de éstos, agregado de datos, estructuración de los datos y medición de calidad de los mismos. [6]

Por otro lado, existe lo que se denomina la Gestión de Procesos de Negocio (BPM) que permite modelar, implementar y ejecutar las actividades inherentes al negocio de la organización con base en sus Procesos de Negocio. Estas actividades están relacionadas entre sí dentro de los procesos, y no sólo pertenecen a algún sector en particular de la organización, si no que trascienden a la misma, como puede ser a proveedores, clientes, etc. que participan en dicho proceso. BPM es un conjunto de metodologías orientadas a la optimización de los recursos basados en la automatización y sistematización de los procesos. [6]

Por lo dicho anteriormente, la principal motivación de nuestro proyecto es la construcción de un WW a través de una plataforma enfocada al modelado de procesos de negocio (BPM). Dicho proyecto está en el contexto de las líneas de investigación conjuntas entre los grupos COAL [2] y CSI [10], que incluye otras propuestas como una tesis de maestría, entre otras. [6]

1.2 Objetivos del proyecto

A continuación describiremos los objetivos generales y específicos del proyecto.

(OG) El objetivo general de este proyecto es explorar la utilización de un BPMS para modelar y ejecutar los Procesos de Configuración y Carga (Feeding) para la construcción de un WW enfocado en la calidad, que permitan definir e invocar fuentes de datos (utilizando por ej. Web Services), definir y realizar la integración de esquemas de datos, y además medir la calidad del proceso y los datos asociados. [6]

Los objetivos específicos relacionados se definen como:

(OE1) - Estudio de temas relacionados con Web Warehouse, Procesos de Negocio, Calidad de Datos y Servicios asociados, de la propuesta y arquitectura de WW y procesos definidos, y de plataformas BPMS *open source*.
(OE2) - Definición y modelado de los Procesos de Configuración y Carga para la construcción del WW.
(OE3) - Implementación de un prototipo de los Procesos de Configuración y Carga para la construcción del WW.
(OE4) - Caso de estudio de aplicación con ejecución del prototipo desarrollado para la construcción de un WW enfocado en la calidad para un dominio específico definido (por ej. datos abiertos de gobierno).

1.3 Resultados esperados.

En la siguiente sección vamos a presentar los resultados esperados del Proyecto de Grado.

(R1) - Obtener un proceso de Configuración de WW implementado en BPMS Activiti que soporte los distintos requerimientos: el usuario interesado debería poder parametrizar un dominio o temática en particular, sus requerimientos y propósitos específicos de análisis sobre la misma, indicar cómo y de donde realizar la extracción de los datos y de qué manera integrarlos.

(R2) - Obtener un proceso de Carga de WW implementado en BPMS Activiti que permite construir diferentes WW según las necesidades del usuario. Es deseable que dicho proceso se lleva a cabo lo más automatizado posible, dejando las actividades manuales para la resolución de conflictos o decisiones que el Sistema no podría tomar.

(R3) - Tanto en el proceso de Configuración como en el de Carga, se debería poder incorporar aspectos de calidad definidos por el usuario. Esto refiere a la calidad de los datos que gestiona y la calidad de los servicios que proporcionan los datos de origen. (Dichos datos muy comúnmente presentan inconvenientes y problemas como errores de escritura de datos, valores perdidos, datos obsoletos e incompatibles).

(R4) - Poder ejecutar un caso de estudio, que permita mostrar la factibilidad de la construcción y carga de un WW básico utilizando los procesos implementados.

1.4 Resultados alcanzados.

A continuación vamos explicar de qué manera resolvimos los resultados esperados propuestos en la sección anterior.

(RA1) - Se analizó, modeló e implementó el proceso de Configuración completo con la tecnología esperada, en una versión inicial que tiene en cuenta los principales requerimientos definidos para el mismo. El proceso de Configuración contiene actividades manuales en las cuales el usuario tiene la posibilidad de elegir con qué dominio desea trabajar, cuáles son las fuentes web de donde se van a extraer los datos, además definir la forma en cómo se extraen los mismos y cómo se vinculan e integran entre ellos. Finalmente, el sistema nos permite definir el esquema de DW final, con sus relaciones dimensionales que incluyen las tablas de hechos y las tablas de dimensiones. Como resultado de la ejecución de este proceso, obtenemos los metadatos necesarios para poder construir el DW definido.

(RA2) - Se analizó, modeló e implementó el proceso de Carga en una versión inicial de tipo *draft* sólo para mostrar la factibilidad de carga del DW utilizando los metadatos definidos en el proceso de Configuración. El

proceso cuenta con un conjunto de actividades automáticas que logran realizar la extracción de los datos, la integración de los mismos y finalmente la carga del DW. Se plantea la extensión de dicho proceso de Carga con actividades manuales para la resolución de conflictos de extracción e integración (en las secciones de mejoras y trabajo a futuro, proponemos una solución para esto).

(RA3)- Este resultado fue finalmente eliminado del alcance dado que la definición, modelado e implementación de los procesos anteriores llevo más tiempo del esperado.

(RA4) - Se realizó un caso de estudio con datos abiertos de Turismo de Uruguay que demostró la factibilidad de la propuesta de automatización de la construcción de un WW con BPMS mediante los procesos definidos en RA1 y RA2.

1.5 Gestión del Proyecto.

Al comienzo del proyecto lo prioritario fue relevar información y re-aprender los conceptos teóricos de parte del equipo sobre los conceptos más importantes del Proyecto que son las áreas de Data Warehouse (DW) y la Gestión de Procesos de Negocio (BPM).

Paulatinamente las reuniones se fueron enfocando cada vez más al tratamiento del artículo [6] que plantea la idea del proyecto analizando la solución del problema. Se decidió comenzar por trabajar inicialmente con el proceso de Configuración, sin agregados de calidad, para lograr un primer modelo e implementación detallada del mismo. Luego se agregarían las consideraciones asociadas al manejo de la calidad de datos en la extracción y en el DW resultante.
A medida que analizábamos las diferentes actividades, se iba modelando el proceso e implementando en el prototipo, evaluando el impacto en el flujo del proceso y las posibilidades que daba la herramienta elegida (Activiti) frente al modelo realizado. Una vez que el resultado obtenido era satisfactorio, íbamos profundizando en el flujo y así sucesivamente.

La mayoría de las horas dedicas al proyecto fueron invertidas en el análisis y el diseño de los procesos tanto de Configuración como de Carga. Además, el proceso de Configuración es el que tuvo más prioridad en este sentido, y nos enfocamos con respecto a la carga a poder completar una construcción básica de un WW utilizando los metadatos definidos en el proceso de Configuración.

Por otro lado, también se implementó la aplicación que nos provee y encapsula la extracción de los datos. Dicha aplicación resuelve la extracción de algunos formatos de datos pero no todos, ya que consideramos que la misma no forma parte de la solución central. En este caso fue necesario invertir tiempo en aprender sobre aplicaciones que publiquen servicios Rest mediante Spring. La invocación a este aplicación se realiza remotamente por lo que podría ejecutarse en un servidor distinto al servidor donde se ejecuta nuestro Sistema.

Con respecto al desarrollo del código de los integrantes del equipo, fue administrado por SVN de manera de independizarnos en el trabajo sobre los procesos de negocio y sobre los códigos Java.

El análisis y el diseño mencionado anteriormente fue realizado en conjunto por los dos integrantes del equipo, así como también las decisiones de implementación. Sin embargo a la hora de modificar los diagramas de BPM lo hicimos sin generar concurrencia para no tener problemas de conflictos en los mismos.

2 Marco teórico.

Este capítulo contiene los conceptos principales en los cuales se basa este proyecto de grado, estos son las definiciones de Procesos de Negocio, Data Warehouse y Web Warehouse.

2.1 Data Warehouse y Web Warehouse

Data Warehouse

Un Sistema de DW tiene como objetivo automatizar los procesos de recopilación y generación de datos para los usuarios. Para esto el sistema debe extraer los datos de las distintas fuentes, transformarlos, integrarlos, refinarlos según las necesidades y centralizar dicha información que la organización genera diariamente en sus actividades de negocio, manteniendo los datos históricos, y adicionando según sea necesario datos externos a las cuales pueda estar relacionada. [3]

Para que dichos datos sean útiles a los usuarios, el Sistema permite el acceso y la manipulación de los mismos a través del análisis multidimensional con el objetivo final de ser una herramienta para la toma de decisiones.

En las organizaciones, el análisis para la toma de decisiones presenta distintos obstáculos, algunos de ellos son la accesibilidad de las personas a la información, el tiempo de acceso a la misma, y la integración y calidad de los datos. [3]

Los Sistemas tradicionales que utilizan las empresas hoy en día, como por ejemplos los ERP (Enterprise Resource Planning), son transaccionales, esto es, están orientados a operaciones de escritura y modificación, están asociados a procesos productivos de la organización y procesan grandes cantidades de transacciones. No son adecuados para ser utilizados como medios de consultas masivas y complejas, por lo que no pueden resolver las necesidades antes mencionadas.

En contraparte, los Sistemas de DW, atacan esta problemática, generando información para la toma de decisiones, tomando como referencia algunas premisas: construir información a partir de datos objetivos, integrar diferentes fuentes de datos y ofrecer al usuario final mecanismos flexibles para el acceso a la información. [3]

Como los datos son históricos, se deben manejar con su referencia temporal y son no volátiles, los mismos deben ser estables como para permitir el análisis de larga duración, por lo que para esto es necesario realizar una hostilización y planificación de carga.

Arquitectura típica para un Sistema de Data Warehouse

Analizando la arquitectura típica de un DW como podemos ver en la Figura 1, en la capa más inferior se encuentran las distintas fuentes que nos proveerán los datos, que pueden ser históricos, archivos de texto, bases de datos relacionales, no relacionales, información geográfica almacenada en algún formato, etc.

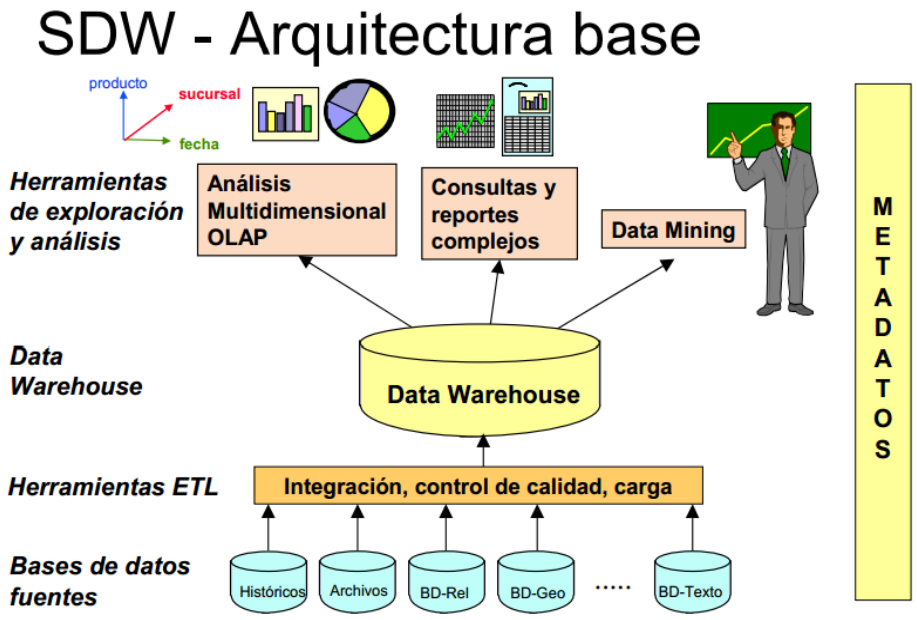


Figura 1 Arquitectura de un Data Warehouse [3]

Cuando ya definimos las distintas fuentes de donde obtendremos los datos, utilizamos las herramientas de ETL que nos permiten extraer, transformar y cargar los mismos. Una vez que extraemos dichos datos, podemos alojarlos en un medio común, realizar transformaciones necesarias para integrarlos, controlar la calidad de los mismos y cargar nuestro DW. [3]

El Sistema finalmente es un repositorio de los datos ya procesados y listos para su explotación. Se organizan en tablas denominadas de *dimensiones* y de *hechos*. Las tablas de dimensiones son las que contienen los datos correspondientes a un objeto en particular del negocio, una dimensión específica de análisis. Y las tablas de hechos, contienen registros de un hecho en particular e importante para la organización que queremos estudiar y analizar como puede ser una venta en un comercio o una reserva en una agencia de viajes, en donde participan y se cruzan las distintas dimensiones. Además, el Sistema nos provee de distintas medidas definidas por los técnicos y especialistas del negocio, que permiten realizar operaciones o análisis sobre los cruzamientos de las dimensiones y los registros de hechos. [3]

Una vez que tenemos el DW completamente cargado con sus tablas y medidas, la siguiente etapa es la que corresponde a la presentación de la información a los usuarios finales. Para esto utilizaremos herramientas dedicadas a ello que se denominan Herramientas de Exploración y Análisis. Puede ser herramientas de análisis multidimensional (Herramientas OLAP) que nos permiten armar distintos *cubos* que cruzan los datos, o también podemos obtener reportes planos de cierta complejidad o que impliquen consultas costosas.

Por último, los usuarios idóneos, no técnicos si no especialistas del negocio de la organización, podrán utilizar estas herramientas para poder realizar distintas operaciones de agrupación, cruzamiento, sumatorias, promedios, entre muchas otras, con el fin de poder tomar decisiones importantes para el futuro

de la organización en función de los datos del pasado. Estos usuarios finales son principalmente gerentes o de cargos altos cuyo trabajo refiere a la toma decisiones y a la planificación.

Con respecto al desarrollo de un Sistema de DW, se distinguen las siguientes fases [3]:

- Especificación de requerimientos: el desarrollador debe mantener reuniones con los usuarios de negocio y usuarios finales para poder entender la realidad de la organización en cuanto a los procesos de negocio que en ella existen.

- Diseño conceptual: diseñar el modelo conceptual que representa a la realidad de la organización en nuestro Sistema. Aquí tendremos representadas todas nuestras entidades que tengan importancia en el negocio, y las relaciones que se manifiestan entre ellas.

- Diseño lógico: enfocamos a la distribución de las tablas que contendrán los datos, diseñamos las tablas de dimensiones y las tablas de hechos que ya explicamos en párrafos anteriores. Otra actividad que podemos realizar en esta etapa es la definición de las medidas que nos van a permitir explotar mejor los datos. Dichas medidas pueden ser pre calculadas o definir cálculos sobre los datos.

- Proceso de extracción, transformación y carga: extracción se refiere a obtener los datos de las diferentes fuentes que nos proveerán los mismos. Transformación implica realizar determinados ajustes a los datos para por ejemplo, unificar formatos, codificaciones etc. Una vez transformados los datos seguramente queramos integrarlos entre ellos para generar información de valor, lista para ser explotada. Con respecto a la carga, se refiere al llenado de las tablas de dimensiones, de hechos, las medidas pre calculadas, etc. utilizando los datos públicos e integrados.

- Explotación: debemos preparar las herramientas que nos permitirán mostrar al usuario con útiles interfaces y operaciones de análisis, las cuales serán alimentadas por las tablas que de dimensiones y de hechos.

Diseño Conceptual

En la etapa de Diseño Conceptual se tiene por objetivo analizar la realidad del problema y obtener una descripción abstracta y completa del mismo. Para ello, comenzando con el relevamiento de requerimientos de los usuarios se identifican los objetos relevantes al problema y las relaciones entre ellos, para terminar construyendo un esquema conceptual expresado en términos de un modelo conceptual. [12][13]

"En el marco del diseño conceptual se llaman dimensiones a los criterios de análisis de los datos, es decir, a las variables independientes en la realidad planteada. En las dimensiones los valores se organizan en jerarquías." [12] Las jerarquías se dividen por niveles y nos permiten definir una o más agrupaciones de las entidades según alguna característica. Por ejemplo, si tenemos la dimensión "Vendedores", podemos "subir de nivel" en su jerarquía por Ciudad, para obtener información de los vendedores de la misma ciudad. En la Figura 2 se ilustra el ejemplo que mencionamos. [3]

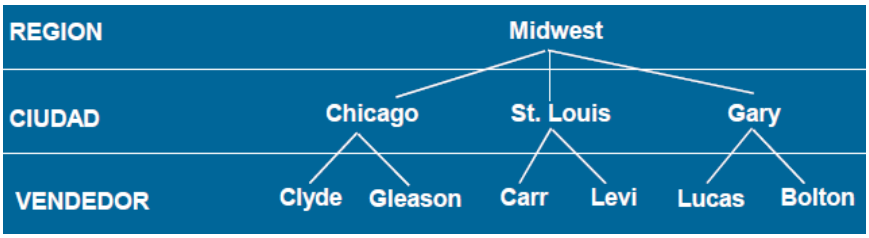


Figura 2 Ejemplo dimensión Vendedores [3]

"Los modelos de datos para especificar los datos almacenados en el DW son diferentes. A nivel conceptual se proponen modelos multidimensionales [Ken96], [Agr97], [Car00], que representan la información como matrices multidimensionales o cuadros de múltiples entradas denominados cubos. A los ejes de la matriz se los llama dimensiones y representan los criterios de análisis, y a los datos almacenados en la matriz se los llama medidas y representan los indicadores o valores a analizar." [13]
Continuando con el ejemplo de Vendedores, podrían ser vendedores de autos en donde tenemos dos dimensiones que son Modelo y Color, y sus cruzamientos en la matriz corresponden a la medida en cuestión, por lo que como se muestra en la Figura 3 sería Ventas("Mini Van","Blue")=6. [3]

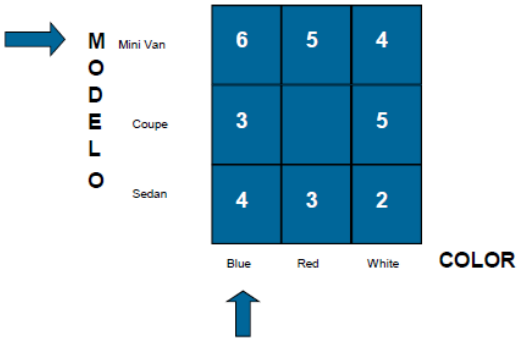


Figura 3 Ejemplo de medida. [3]

"Dado un valor para cada dimensión se puede determinar un valor para la medida". [3]

Modelo CMDM

Estructuras básicas del modelo:

- Niveles.
- Dimensiones.
- Relaciones Dimensionales.

Nivel:

"Un nivel representa un conjunto de objetos que son de un mismo tipo. Cada nivel debe tener un nombre y un tipo. Conceptualmente, no tiene diferencia con cualquier elemento del Modelo Entidad Relación que pueda ser considerado un conjunto de Entidades" [4]

Dimensiones:

"Una dimensión está determinada por una jerarquía de niveles. La instancia es una jerarquía de elementos de esos niveles". [4]

Relación dimensional:

"Una relación dimensional representa un conjunto de cubos, tomado del conjunto de todos los cubos que se pueden construir a partir de los niveles de un conjunto dado de dimensiones." [4]

"En CMDM, un cubo es una función que va del producto cartesiano de las instancias de los niveles en los booleanos. De esta forma, cualquier nivel puede cumplir el rol de medida". [4] *"Notar que, en CMDM, los cubos son elementos de las instancias de las relaciones dimensionales y no hay una estructura que los represente directamente*." [4]

Figura 4 Ejemplo Relación Dimensional

Siguiendo con el ejemplo antes mencionado, la Figura 4 ilustra la relación dimensional de Ventas, donde se pueden ver las dimensiones Colores, Vendedores, Modelos y la medida Cantidad de Unidades. [3]

Diseño Lógico

"La etapa de diseño lógico toma como entrada un esquema conceptual y genera un esquema lógico relacional o multidimensional." [13]
"El diseño lógico se refiere a expresar operaciones sobre los datos en forma abstracta para que sea viable realizar su implementación sobre ciertas herramientas concretas" [12]

Los DW con los que vamos a trabajar están conformados por tablas de hechos y tablas de dimensiones. En las tablas de hechos se alojan las medidas definidas en las relaciones dimensionales y las tablas de dimensiones contienen las dimensiones incluyendo sus atributos y sus jerarquías. [3]

Las dos formas más importantes de organizar estas tablas son el esquema estrella y el esquema copo de nieve.

El esquema estrella consiste en tener un conjunto de tablas de hechos que contienen las referencias foráneas a cada una de las tablas de dimensiones con la cual se vinculan y las tablas de dimensiones son las que alojan todos los atributos de las dimensiones y sus jerarquías (estas tablas están desnormalizadas) dibujando así algo similar a una estrella. En la Figura 5 mostramos un ejemplo de Ventas y su correspondiente diagrama estrella.

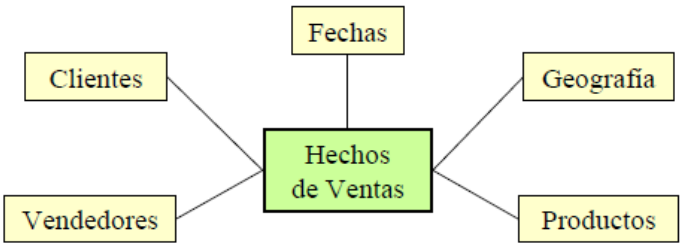


Figura 5 Ejemplo de Modelo Estrella [3]

Por otro lado, tenemos lo que se llama el modelo *SnowFlake* (Copo de Nieve), que en contraposición al modelo estrella, las tablas de dimensiones están normalizadas por lo que tendrán referencias sucesivas a otras tablas indicando la jerarquía de la dimensión. En la Figura 6 ilustramos el mismo ejemplo pero con el modelo SnowFlake.

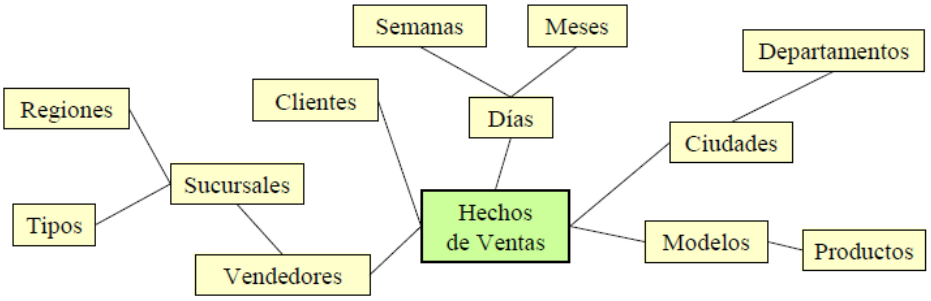


Figura 6 Ejemplo de Modelo SnowFlake [3]

Web Warehouse

El concepto más importante que plantea nuestro proyecto es de Sistema de WW. Si bien existen diferencias podemos compararlo en una gran cantidad de aspectos con los Sistemas de DW, donde la mayor diferencia radica en el origen de las fuentes de datos ya que se plantean sean provenientes de la Web.

Esto implica algunos obstáculos extras ya que las fuentes pueden cambiar en cualquier momento, dejar de estar disponibles o cambiar de formato de publicación de los datos. Además, cuando surgen nuevas fuentes Web, a priori no sabemos si nuestro Sistema soportará la extracción de esos datos o debemos realizar nuevos desarrollos para contemplarlo.

Adicionalmente, con el fin de ser una herramienta útil un WW debe tener en cuenta diferentes aspectos, como son la calidad de los datos que gestiona y la calidad de los servicios que proporcionan los datos de origen. Algunas cualidades son las que mencionamos recientemente, por ejemplo la alta disponibilidad de los datos provistos por el servicio, que sean lo menos variantes posibles en el tiempo, datos certeros y con formatos lo más convencionalmente posibles, esto es, HTML, XML, TXT, CSV, etc. Por lo que el Sistema debe estar preparado para soportar grandes cambios en las fuentes que proveen los datos, ser flexible a los mismos, o en el mejor de los casos, poder tener un control de contingencia para problemas de este tipo.

El proceso de construcción de un WW presenta grandes obstáculos, como por ejemplo los distintos formatos con los que nos podemos encontrar en los datos de la Web y protocolos de extracción de diversas naturalezas (en párrafos anteriores ya mencionamos algunos de ellos). Para complementar un poco más sobre la calidad de los mismos, observamos que los datos presentan diversos problemas como errores de

escritura en los datos, valores inexistentes, datos obsoletos o que no son compatibles con lo que esperamos. [6]

Por otro lado, una fuente Web (que definimos como Web Data Source, de ahora en adelante WDS) tiene total autonomía, que por lo general conduce a problemas de disponibilidad, costos, e inesperados cambios de formatos y protocolos de comunicación que utilizamos para obtener los datos.

Por lo que un Sistema de WW, debería permitir realizar una evaluación de calidad sobre estos WDS (y sus datos) para de alguna manera categorizarlos en "buenos" o "malos" WDS. Para ello, el Sistema debe ser flexible y configurable, para que estos servicios puedan seleccionarse dinámicamente de acuerdo a las necesidades de los datos que queremos procesar que puede variar con cada carga del WW. [6]

Ciclo de vida de un Web Warehouse

Debido a la complejidad de este tipo de Sistemas es importante especificar las diferentes etapas que componen su ciclo de vida, así como los diferentes roles que participan activamente en su instalación y operación. [6]

En la Figura 7 podemos ver un ejemplo de arquitectura de un WW en la cual se plantean tres principales componentes que son: Extracción de Datos (DSI), Integración de Datos (Integrator) y Transformación y carga de los datos al DW a través de diferentes módulos (ETL). [6]

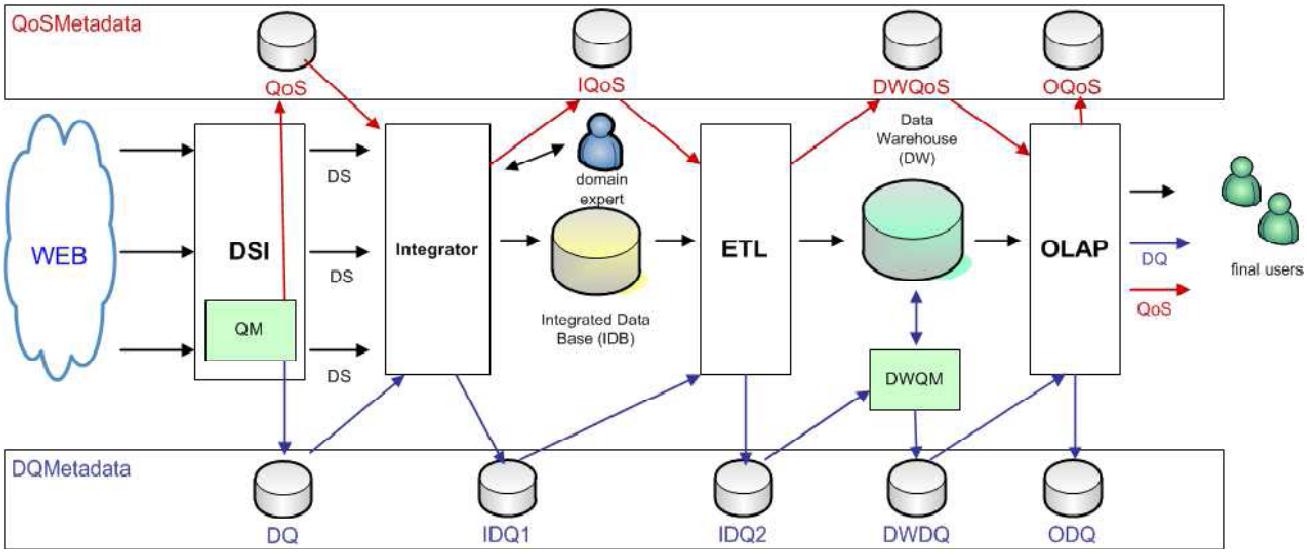


Figura 7 Arquitectura de un Web Warehouse [6]

Data Service Infrastructure:

El módulo Data Services Infrastructure (DSI) proporciona un entorno de ejecución para implementar, almacenar y ejecutar Data Services, con el fin de acceder y obtener los datos de las fuentes Web. Básicamente son servicios externos al Sistema de WW implementados por desarrolladores informáticos que conectan (por ejemplo vía Rest Services) el Sistema de WW con los WDS. [6]

Estos DSI encapsulan el conocimiento y la implementación de cómo establecer una conexión con el proveedor de los datos y como extraer los mismos, el Sistema de WW se abstrae de ello y solo confía en que

el DSI le proporcionará los datos existentes y en los formatos originales desde la fuente. Esto tiene el beneficio de que en caso de cambiar los formatos o las publicaciones de los datos, no se debe modificar el Sistema de WW, sólo debería realizarse ajustes a estos servicios que están por fuera. [6]

Tareas principales de un DSI:

- Proporcionar el servicio de extracción de datos de los diferentes WDS exponiéndolos como DS homogéneos y utilizando tecnologías estándar como son los Web Service Soap o Rest.

- Realizar un monitoreo de los DS periódicamente y evaluar la calidad de los mismos. Dicha medición será efectuada por servicios específicos que son invocados por el módulo de calidad.

- Provee de mecanismos de adaptación en tiempo de ejecución, en el proceso de extracción para reaccionar ante posibles degradaciones de la calidad de los DS.

Integrator:

El módulo Integrator es el encargado de realizar la integración de los datos provistos por el DSI y proporcionarlos en un formato normalizado, generando así la base de datos integrada que llamaremos IDB. El problema de que los datos provengan de fuentes realmente heterogéneas, permite que los datos presenten formatos y codificaciones distintas. [6]

Las siguientes tareas, son responsabilidad del módulo Integrator:

- Selección de los datos en función de la calidad particular de cada DS.

- Integrar los datos tal cual lo mencionamos anteriormente.

- Generación de metadatos de calidad asociada a los datos integrados.

Este modulo interactúa con los usuarios expertos que realizan las principales decisiones de integración de datos, tales como la identificación de objetos y resolución de conflictos. Dicha interacción deberá darse mediante una interfaz de usuario. [6]

ETL y OLAP

Los módulos de ETL y OLAP realizan las mismas tareas que en un Sistema de DW, esto es la extracción, transformación y carga y la presentación de los datos a los usuarios en cubos, a grandes rasgos respectivamente. Adicionalmente, debe construir los metadatos correspondiente a la calidad de los datos y de los DS. (DQ y DoQ en la arquitectura).

En la Figura 8 podemos ver el Ciclo de vida en la construcción de un WW, en primer lugar se deben seleccionar las fuentes Web asociadas al dominio de interés, de los cuales queremos extraer los datos. Luego, la plataforma debe ser instalada y configurada con el fin de crear una instancia para el dominio en particular. Ahora sí la carga del WW puede ser ejecutada, extrayendo los datos, integración, transformación y carga del WW. Al final, el WW puede ser explotado por el usuario final, cerrando así el ciclo de vida del mismo. [6]

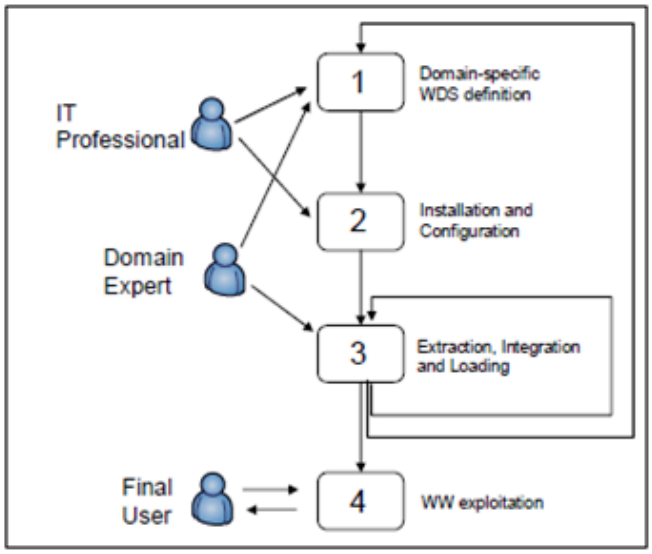


Figura 8 Ciclo de vida de un Web Warehouse [6]

2.2 Procesos de Negocio y Tecnología

Existen varias definiciones para procesos de negocio, algunas iniciales de la década de los noventa y otras más actuales que involucran nuevos elementos. Un proceso de negocios es:

- *"Un conjunto de tareas relacionadas realizadas para alcanzar una salida del negocio definida para un cliente o mercado particular"* (Davenport, 1993) [2].
- *"Una colección de actividades que toma una o más tipos de entrada y crea una salida que es de valor para el cliente"* (Hammer & Champy, 1993) [2].
- *"Un conjunto de actividades realizadas en coordinación en un entorno organizacional y técnico, para alcanzar un objetivo del negocio"* (Weske, 2007) [2].

Este conjunto de actividades tienen como objetivo ayudar a alcanzar un determinado objetivo de negocio. Cada proceso de negocio es realizado por una única organización, pero puede interactuar con procesos de otras organizaciones. En la Figura 9 podemos ver los principales elementos, son las entradas y salidas que están formadas por los requerimientos de clientes, información, servicios, productos, etc. Los recursos destinados para realizarlos con las personas, materiales y un conjunto de objetivos de la organización y los procesos de negocio específicos.

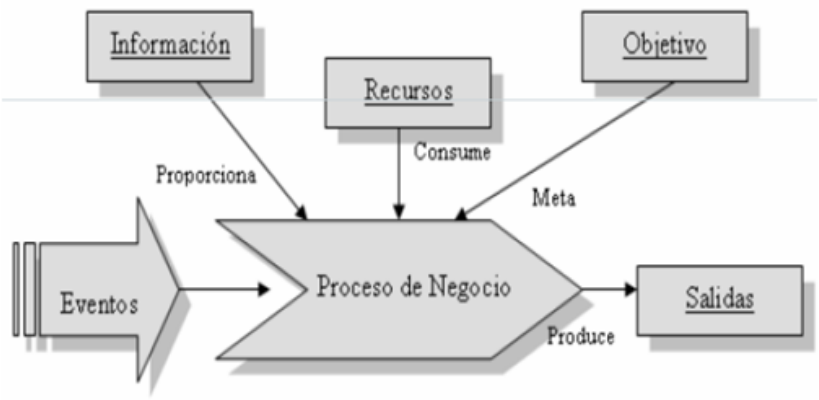


Figura 9 Elementos participantes en un Proceso de Negocio [2]

Existen también elementos específicos en la construcción de un Proceso de Negocio, por ejemplo los participantes internos de la organización (secciones, áreas, roles), participantes externos (clientes, socios, proveedores), actividades, tareas, datos y relación entre ellos (secuencia, reglas de negocio, bifurcaciones, restricciones). [2]

Los procesos de negocio se caracterizan por ser grandes, complejos y muy dinámicos, además involucran varias secciones o áreas de una organización y organizaciones distintas y tienen duraciones prolongadas a lo largo del tiempo, o sea una ejecución puede durar semanas, meses o años. Estos procesos son específicos de cada dominio: salud, bancarios, etc., pueden ser tanto manuales, automatizados o ambos y en general son difíciles de explicitar (implícitos en sistemas). [2]

Modelos de Procesos de Negocio

Son una representación abstracta (gráfica) 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. Además muestran también otro tipo de información que es de valor agregado para la empresa como ser los actores involucrados en los procesos (roles, áreas), cuáles son las actividades operativas distinguibles, qué actividades son ejecutables y por quién, cuáles son las entradas y salidas de las actividades, cuál es la secuencia de las actividades, los recursos consumidos, y los eventos que dirigen el proceso. [2]

La definición de Modelo de Proceso de Negocio es:
"Representación del conjunto de actividades y de las restricciones de ejecución entre ellas" (Weske, 2007) [2].

En la Figura 10 se muestra un ejemplo de esta representación gráfica para "hacer pan" donde se puede observar el *lane* que es el "Panadero", un evento de inicio, seguido de las tres actividades manuales por la que está formado el proceso y termina con el evento de fin. [1]

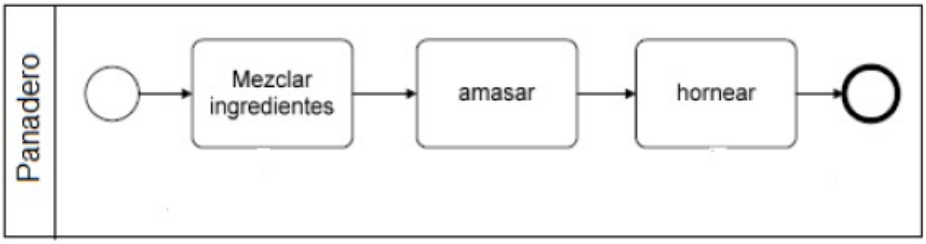


Figura 10 Ejemplo de modelo para hacer pan. [1]

Estos modelos permiten razonar sobre posibles re-diseños, evaluando distintas posibilidades (flujos, interacciones), permiten además realizar un análisis basado en simulación o técnicas analíticas, predecir performance, utilización de recursos, cuellos de botella y son la base para incorporación de un BPMS, automatización de actividades, interacción humana, envío y recepción de mensajes. [2]

Algunos posibles errores al realizar el modelado de Procesos de Negocio podrían ser: describir sólo una idealización de la realidad, concentrarse en el flujo "normal" ó "deseable" y dejar afuera flujos alternativos que puedan causar problemas, modelar a un nivel de abstracción erróneo, muy abstracto y general, no permite "ver" el proceso, muy detallado se vuelve muy complejo y difícil de entender, no elegir correctamente la notación a utilizar, no todos los involucrados pueden "entender" lo que está modelado (dificulta la comunicación). [2]

Gestión de Procesos de Negocio

Las nuevas tecnologías de gestión se han introducido en las organizaciones como una herramienta para solucionar problemas con clientes, proveedores o empleados, y además fomentan la colaboración entre los actores, la Gestión de Procesos de Negocio se podría definir como:
- *"Un conjunto de actividades que realizan las organizaciones para optimizar o adaptar sus Procesos de Negocio a las nuevas necesidades organizacionales"* (BPMI, 2005) [2].
- *"Incluye conceptos, métodos y técnicas para soportar el diseño, administración, configuración, ejecución y análisis de los Procesos de Negocio en las organizaciones"* (Weske, 2007) [2].
- *"Soporte para Procesos de Negocio utilizando métodos, técnicas y software para diseñar, ejecutar, controlar y analizar procesos operacionales involucrando humanos, organizaciones, aplicaciones, documentos y otras fuentes de información"* (Van der Aalst, 2003) [2].

Los elementos principales son: optimizar, adaptar, mejorar los Procesos de Negocio en las organizaciones, procesos alineados con objetivos organizacionales, incluye conceptos, métodos, técnicas y software (BPMS). Además soporta el diseño, análisis, configuración, ejecución, control, evaluación de los Procesos de Negocio (Ciclo de vida de los Procesos de Negocio), involucrando humanos, organizaciones, aplicaciones, documentos y otras fuentes de información. [2]

BPM

En la Figura 11 y la Figura 12 podemos ver un antes y un después en la estructuración de las organizaciones con o sin BPM respectivamente, antes la gente de negocios (procesos, roles, personas, …) y los técnicos (sistemas, máquinas, datos, …) hablaban en términos de la función que ocupan en la organización, en cambio ahora con BPM es posible que todos hablen de lo mismo, la tecnología BPMS permite salvar la distancia con los Sistemas, máquinas y aplicaciones que automatizan los PN. [1]

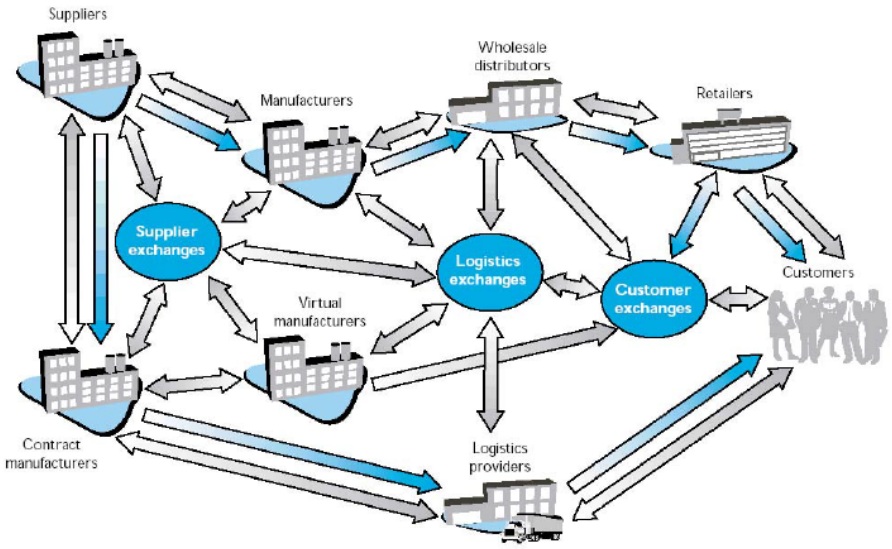


Figura 11 Organización y comunicación antes del BPM [1]

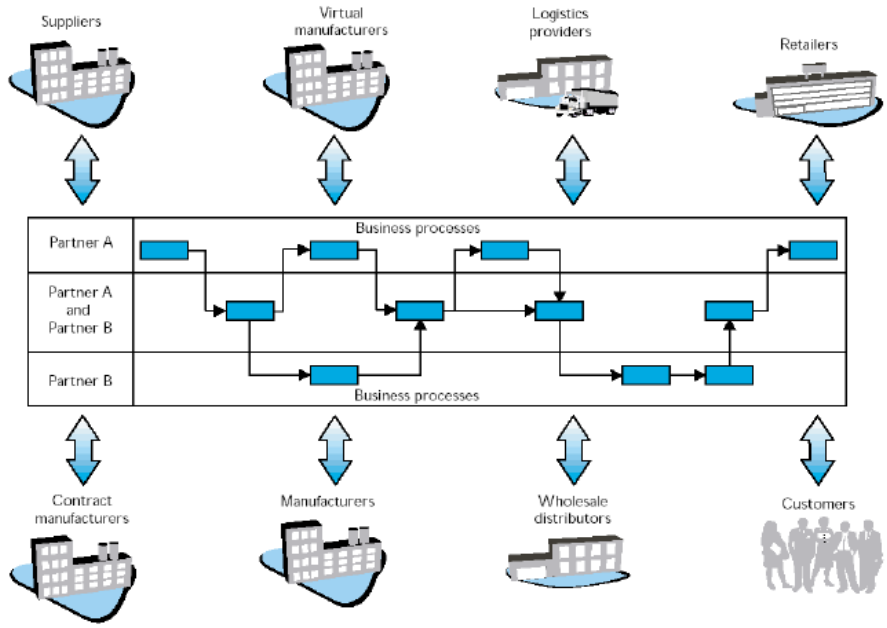


Figura 12 Estructura y comunicación de una Organización luego del BPM [1]

Ciclo de Vida de Procesos de Negocio

En la Figura 13 podemos ver las distintas etapas en el ciclo de vida de los Procesos de Negocio, en la fase Diseño y Análisis el objetivo es modelar y validar los procesos de negocio en la organización, las tareas principales en esta etapa son: analizar los procesos de negocio y su entorno organizacional y tecnológico, representarlos en modelos de procesos de negocio con notación adecuada (BPMN 2.0) y buenas prácticas (Workflow patterns) y finalmente verificar y validar los modelos especificados [2].

A continuación en la fase de Configuración, el objetivo es implementar, testear y desplegar los procesos de negocio en la organización, en la que se destacan como tareas principales seleccionar plataforma, tecnologías y lenguajes de implementación, implementar procesos de negocio y software (servicios), reglas de negocio, formularios de usuario, integración, etc., y realizar testing y despliegue del sistema en la organización para su operación (capacitación, migración de datos, etc.) [2].

Luego en la fase de Ejecución, el objetivo es ejecutar los procesos de negocio y registrar datos asociados a la ejecución, para la cual las principales tareas son ejecutar los procesos de negocio según el modelo de proceso de negocio definido y las restricciones, reglas de negocio, etc. asociadas, registrar datos de la ejecución en logs de ejecución (ejemplo: secuencia de ejecución de actividades, tiempos asociados, recursos involucrados, datos manejados, etc.) y monitorear la ejecución de los procesos de negocio, generalmente con un componente de Business Activity Monitor (BAM) [2].

Finalmente en la fase de Evaluación, cuyo objetivo es evaluar la ejecución de los procesos de negocio para mejorar los modelos y la implementación, las actividades principales son procesar los logs de ejecución de procesos de negocio y proveer información, evaluar la ejecución real de procesos de negocio con técnicas de Business Intelligence (BI), medidas de tiempos de ejecución, recursos utilizados, costos, pacientes atendidos, etc. (Key Performance Indicators, KPI) y descubrir modelos, compararlos con ejecución, extenderlos con datos reales (Process mining, minería de procesos) [2].

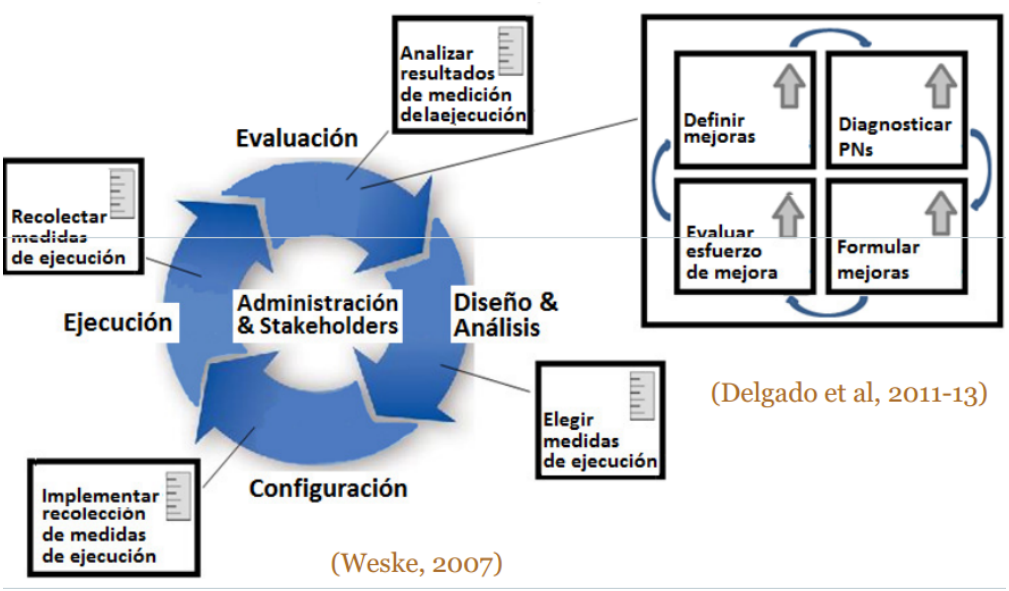


Figura 13 Ciclo de vida de los Procesos de Negocio y su Gestión [2]

BPMN (Business Process Modelling Notation)

Es una notación gráfica (diagramas) para el modelado conceptual de procesos de negocio.
Se ha desarrollado buscando proveer una notación que sea fácilmente entendida por todos los usuarios: analista de negocio, desarrollador técnico, o la propia gente del negocio, crear un puente estandarizado para el vacío existente entre el diseño del proceso de negocio y su implementación tecnológica y asegurar que los lenguajes para la ejecución de los procesos de negocio puedan ser visualizados con una notación común.

En particular BPMN 2.0 además de proveer lo mencionado anteriormente permite visualizar los procesos de negocio de la organización y todos los elementos que intervienen en el mismo (actividades, flujo, participantes, bifurcaciones, datos y subprocesos), intercambiar modelos realizados en distintas herramientas de modelado y permite la ejecución de procesos, en los nuevos motores de ejecución BPMN 2.0 (ejemplo: Activiti, JBPM5). [2]

La Figura 14 contiene todos los elementos mencionados en esta sección, como por ejemplo, Actividades, Compuertas, Eventos, etc.

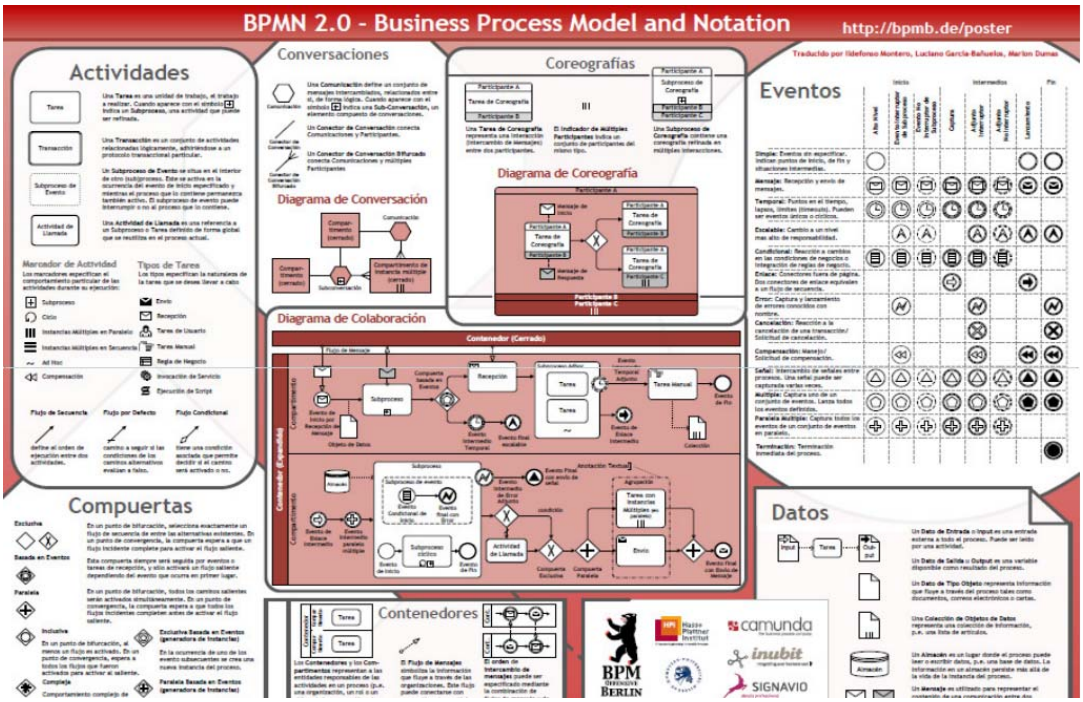


Figura 14 Todas las herramientas disponibles para BPMN 2.0 [2]

Sistemas o Plataformas BPM (BPM Systems)

Algunas de las definiciones más comunes dicen que un BPMS es:
- *"Un sistema de software genérico, guiado por representaciones explícitas de procesos con sus actividades y restricciones para coordinar la ejecución de procesos de negocio"* (Weske, 2007) [2].
- *"Un sistema de software genérico guiado por diseños explícitos de procesos para ejecutar y gestionar procesos de negocio operacionales"* (Van der Aalst 2003) [2].
- *"Un sistema de software genérico que permite coordinar la realización (ejecución) de procesos de negocio en base a representaciones de proceso explícitas (modelos)"* (Weske, 2007) [2].

O sea los BPMS son Sistemas "conscientes" de los procesos de negocio (modelos) los cuales guían su ejecución (puede ser analizados, mejorados, etc.). Un BPMS tiene que incluir un motor de procesos.
Una vez definido (modelado) un PN puede ser sujeto a análisis, mejora o realización.

Los BPMS soportan el ciclo de vida de los procesos de negocio, en los BPMS convergen y se integran diversas tecnologías (middleware), que ya están maduras a nivel de mercado como por ejemplo: servidores de aplicaciones, EAI (Enterprise Application Integration), WorkFlows, ERP, CRM, E-Business, E-Commerce, EDI (Electronic Data Interchange), servicios web, reglas de negocio (Rules Management), inteligencia de negocio (Business Intelligence), cuadros de mando (Business Activity Monitoring). Un BPMS, también llamado BPM Suite, puede incluir múltiples partes. En la Figura 15 ilustramos las distintas tecnologías que pueden estar vinculadas a los BPMS [1]

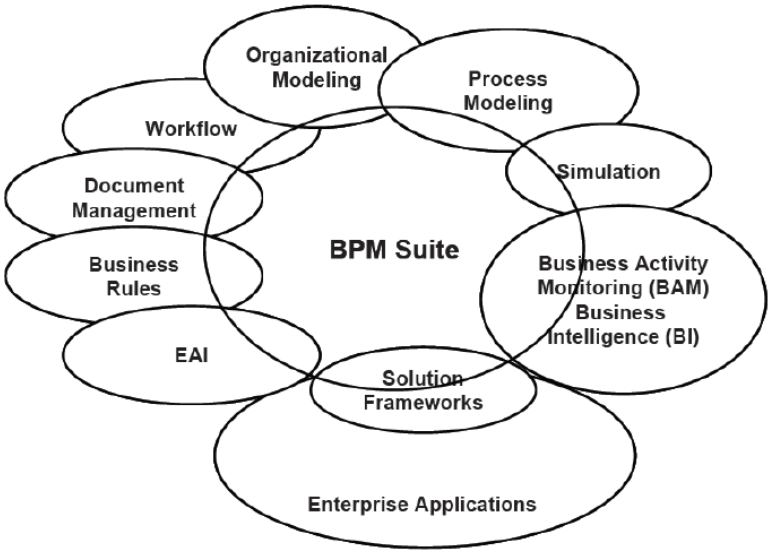


Figura 15 Posibles tecnologías que vincula un BPMS [1]

En la Figura 16 mostramos el diagrama de despliegue del motor de Activiti para el caso en que se cuenta con la herramienta Activiti Explorer y Activiti REST. En este diagrama se puede ver el componente REST que proporciona Activiti que se puede utilizar para comunicarse con el motor de Activiti desde cualquier ubicación remota. [14]

El componente REST se implementa de manera separada del Activiti Explorer y contiene un motor de Activiti independiente. En este diagrama se muestra como se puede interactuar fácilmente mediante el protocolo de comunicación REST, por ejemplo desde un dispositivo móvil o una aplicación web desde otro servidor.

Además de estos dos componentes se puede observar dos tipos de base de datos diferentes, la base de datos propia del Activiti, la cual tiene información de los procesos y el flujo de los mismos y la base de datos empresarial.

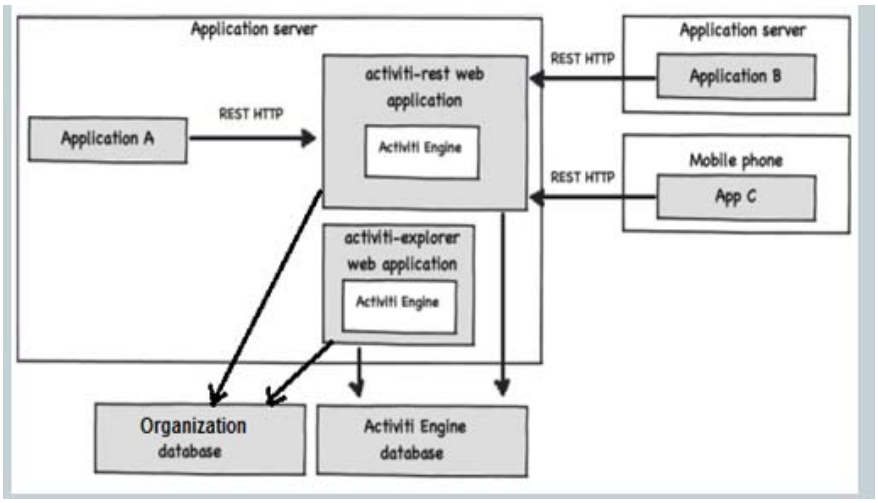


Figura 16 Diagrama de despliegue que muestra una instalación del motor de Activiti

3 Problema planteado

El problema planteado consiste en dos partes bien diferenciadas. La primera de ellas llamada "Configuración", tiene como objetivo poder definir los metadatos con los cuales vamos a construir un WW. Debemos indicar fuentes de datos, formas de extracción de los mismos e integración entre ellos. Esta Configuración definida, podrá ser utilizada cada vez que se quiera obtener un WW asociado a la misma, todos los WW obtenidos que tengan esta Configuración en común comparten la estructura, la misma temática y las mismas fuentes de datos, por lo que se podría decir que esos WW en lo único que difieren es en los datos que albergan. La segunda parte que mencionamos está destinada a (en función de la Configuración seleccionada) efectivamente construir y cargar el WW, lo que sería el conjunto de datos almacenados en un WW en un cierto momento. Por lo que la relación entre los dos módulos es 1 a n, esto es, una Configuración determina una o más cargas de WW. Esto nos permite independizar la Configuración de un WW de la estabilidad de los datos y como ellos cambian en el tiempo.

Lo mencionado en el párrafo anterior se encuentra ilustrado en la Figura 17, en la cual se encuentran las etapas de configuración del WW (WW *Configuration*), de carga del WW (WW *Feeding*), y finalmente una etapa de explotación en donde los usuarios utilizan la información procesada (*WW Explotation*).

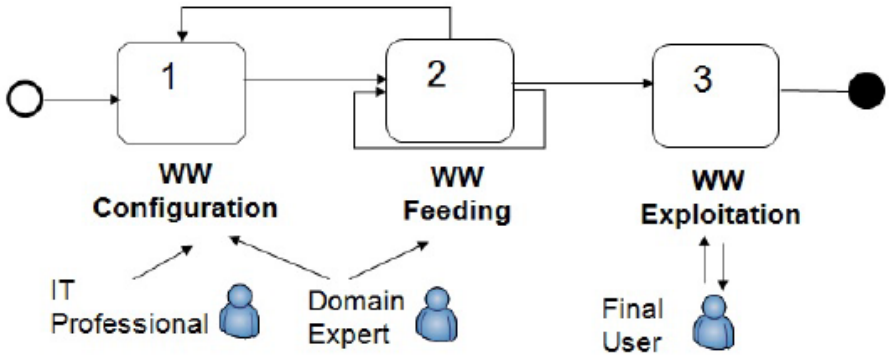


Figura 17 Fases y roles en la construcción de un WW.[6]

Nuestro trabajo abarca las etapas 1 y 2 que se muestran en la Figura 17, dejando por fuera lo que refiere a explotación de la misma. Además, podemos ver que existe una "vuelta atrás" desde la etapa 2 a la etapa 1 para realizar modificaciones a la Configuración, esta idea queda planteada como un trabajo a futuro.

El Proceso de Configuración utilizará como soporte una base de datos, que contiene los metadatos mencionados anteriormente. Lo que deberá realizar este proceso es cargar dicha base para la Configuración en particular que estamos definiendo. Por lo que, cada vez que finaliza una ejecución de un proceso de Configuración, se está generando una instancia particular de la base de metadatos. El Proceso de Carga deberá seleccionar una de estas instancias y en función de los metadatos que se definen en ella, proceder a cargar el WW. Dicho mecanismo se encuentra ilustrado en la Figura 18.

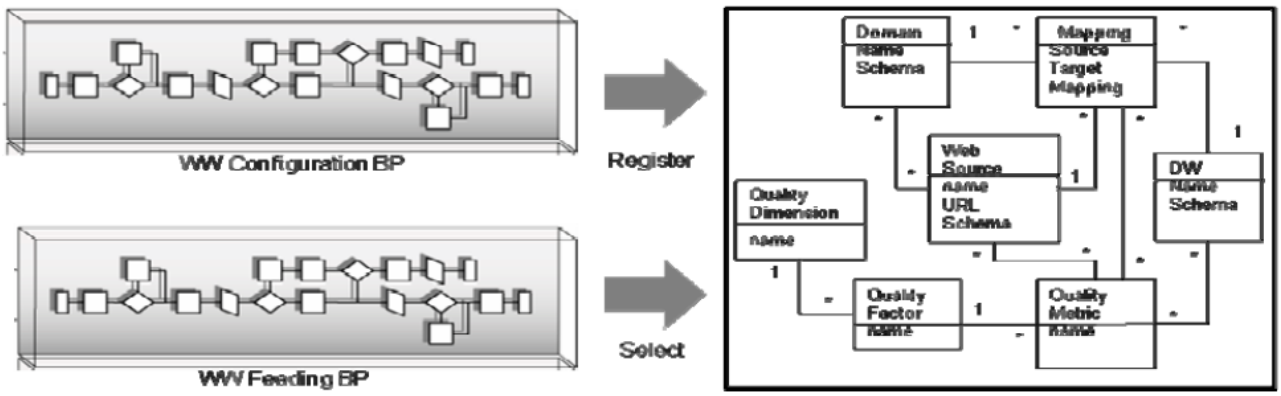


Figura 18 Procesos de Configuración y Carga y el correspondiente modelo de datos.[6]

Mapping

Un concepto que utilizaremos a lo largo de este documento ya que se aplicó en distintas oportunidades para la solución es el *mapping*. El *mapping* al cual nosotros nos referimos es establecer un vínculo entre un atributo de un esquema con uno o más atributos de otro esquema. Dicho concepto nos fue de utilidad para indicarle al Sistema como queremos resolver determinado comportamiento que deseamos se lleve a cabo. En el proceso de Configuración como se explicará más adelante, se habla de distintos *mapping*, pero a modo de ejemplo, debemos definir la relación entre los datos provistos por las fuentes y los datos que realmente queremos extraer (Source Schema y Expected Schema, respectivamente) y para ello realizamos un *mapping* entre ambos esquemas.

3.1 Proceso de Configuración

En el contexto de la definición de Configuración para la construcción de un WW, deseamos en primera instancia poder definir o seleccionar el dominio al cual refieren los datos con los que vamos a trabajar, por esto es que las fuentes que nos proveerán dichos datos deben estar relacionadas a este dominio. (Ver 1 en Figura 19)

Este dominio se refiere a un área en particular, como puede ser Turismo, Cultura, Transporte, etc., los datos con los que vamos a trabajar y el WW que vamos a construir estarán basados en esta temática.

Una vez que definimos qué dominio vamos a utilizar para el WW, el Sistema nos deberá permitir seleccionar qué Web Data Sources nos proveerán los datos, los cuales son fuentes Web que nos brindan los datos con los que queremos trabajar. Estos datos están publicados y el Sistema deberá extraerlos según las necesidades del usuario idóneo (Ver 1 en Figura 19).

Qué datos extraer de cada Web Source, deberá indicarlo el usuario experto definiendo lo que se llama Expected Schema (Ver 2 en Figura 19). Este esquema nos permitirá definir un *template* que indica al Sistema, que datos de los publicados por la Web Source deseamos utilizar y cuáles no. Por lo que definimos el primer *mapping* que se da en la solución, que refiere a vincular los datos que presenta la Web Source con los que el usuario definió en dicho template. (Ver 7 en Figura 19)

Además de indicar al Sistema de qué Web Sources vamos a extraer los datos y precisar cuáles son los que queremos, debemos definir qué Data Services implementan la extracción de los mismos. Estos Data Services

deben ser independientes de nuestro sistema y desarrollados por una persona de perfil técnico (Ver 2 en Figura 19).

No cualquier Data Service implementa la extracción de cualquier Web Source, por ello para cada Web Source debemos especificar el Data Service que nos permite extraer los datos publicados (Ver 3 en Figura 19).

Esto nos permite extraer los datos y de una cierta forma definida, es ahora que el usuario deberá especificar la forma en que desea que se realice la integración de los datos, para ello introducimos el concepto de Integrated Schema (Ver 4 en Figura 19), cuya función es integrar los distintos Expected Schema y contener los datos provenientes de las distintas fuentes en forma integrada.

El Integrated Schema es un conjunto de tablas con atributos, donde cada una de esas tablas tendrá la función de integrar distintas entidades de uno o más Expected Schemas, para poder conformar otra entidad con mayor valor agregado y guardarla en alguna tabla de las que mencionamos. Un ejemplo podría ser analizar distintos alojamientos para hacer turismo y disponemos de distintas fuentes Web que nos proveen datos sobre hoteles, otra fuente web que nos da registro de casas para alquilar u otra que nos provee datos de campings. Todas estas entidades podríamos integrarlas y conformar finalmente una entidad llamada "Albergue" a la cual le incluimos datos de los mismos que para nosotros es de interés.

Aquí se debe realizar un mapping entre cada elemento del Integrated Schema con uno o más elementos del Expected Schema. (Ver 8 en Figura 19)

Hasta aquí hemos definido con qué datos queremos trabajar, de qué manera los vamos a extraer y además, como integrarlos entre ellos. Ahora debemos indicar las características del DW que queremos construir en base a estos datos. (Ver 5 en Figura 19)

Para definir nuestro DW, es necesario indicar las tablas que pertenecen a él, las tablas de dimensiones y las tablas de hechos así como también cada atributo perteneciente a las mismas. Luego de definido el esquema de DW, debemos indicar como se cargará el mismo en base a las entidades definidas en el Integrated Schema. Por lo que nuevamente se deberá especificar un mapping pero ahora entre los elementos a cargar en el DW y los elementos que nos proveerán los registros pertenecientes al Integrated Schema. (Ver 6 en Figura 19).

En el proceso de Configuración existen tres instancias de mapping que el usuario deberá indicar al sistema, éstos son:

Mapping entre el Expected Schema y las Web Sources

En esta instancia de mapping debemos indicarle al Sistema la correspondencia entre cada atributo del Expected Schema con cada campo provisto por la Web Source para todas las Web Sources que se usen en la configuración.

Mapping entre el Integrated Schema y el Expected Schema

El Integrated Schema tiene el objetivo de tomar distintos (o en casos excepcionales uno solo) Expected Schemas y poder integrarlos, para generar información agregada que podrá ser útil más adelante cuando queramos construir un DW Schema. Para ello es necesario definir un mapping para cada atributo de cada tabla del Integrated Schema con un atributo de un Expected Schema.

Mapping entre el Data Warehouse Schema y el Integrated Schema

Finalmente este mapping se realiza para definir cómo se van a cargar las tablas del DW a través del Integrated Schema, de esta manera se deberá indicar un mapping para cada atributo de cada tabla del DW Schema con un atributo de una tabla del Integrated Schema.

En la Figura 19 podemos ver el Proceso de Configuración completo, cuyas etapas fuimos detallando en la subsección 3.1.

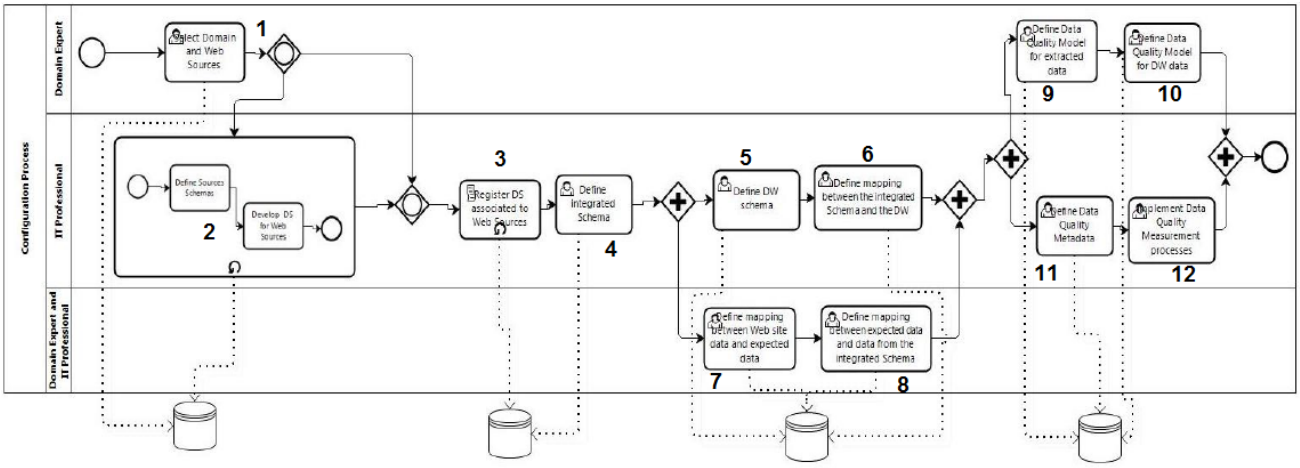


Figura 19 Proceso de Configuración especificado con BPMN 2.0 [6]

Observar que los Expected Schema y las Web Source son globales a la Configuración, mientras que el Integrated Schema y DW Schema conjuntamente con sus tablas tienen una configuración asociada. Es decir, que en estos últimos casos, el Integrated Schema y el DW Schema existen para una Configuración particular.

Otra observación que es necesaria indicar es que la Figura 19 muestra actividades pensadas para la definición de metadatos de configuración de calidad que son: *"Define Data Quality Model for Extracted Data"*, *"Define Data Quality Metadata"*, *"Define Data Quality Model for DW Data"* e *"Implement Data Quality Measurement Processes"* no fueron abordadas por este Proyecto como se ha mencionado a lo largo del documento.

3.2 Proceso de Carga

En la sección anterior pudimos obtener los metadatos de construcción de un WW, el otro problema a solucionar es especificar e implementar de qué manera utilizando dicha Configuración, vamos a lograr construir y cargar el WW.

La interacción entre el Proceso de Configuración y el Proceso de Carga se da a través de la base de metadatos, en donde el primero persistió toda la información generada y luego el segundo proceso, debe recuperar dichos metadatos, interpretarlos y proceder con la carga real del DW.

El Proceso de Carga deberá ejecutar las fases de construcción de un DW tradicional correspondientes a Extracción, Transformación y Carga (ETL) [3] La extracción estará implementada por los Data Services definidos en la etapa de Configuración para cada Web Source.

Como podemos ver en la Figura 20, el proceso de Carga obtiene datos de Hoteles desde fuentes distintas y posiblemente con distintos formatos, desde la Web Source de *Booking* y desde la Web Source de *Despegar*.

Una vez extraídos éstos, el proceso deberá integrarlos (según los metadatos definidos en el proceso de configuración) y persistirlos como una misma entidad de "Hoteles" con formato unificado. Además, realiza la extracción referente a información geográfica que también estará presente en el proceso de integración.

En las flechas marcadas con un 1, es donde se utilizan los metadatos del mapping entre el Source Schema y el Expected Schema ingresado en el proceso de configuración y las flechas indicadas con un 2 es donde se toma en cuenta los datos del mapping configurado entre el Expected Schema y el Integrated Schema.

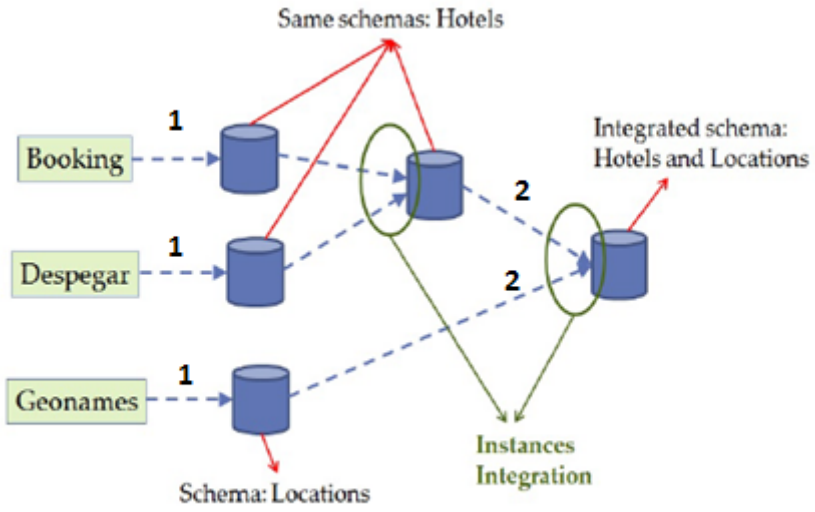


Figura 20 Ejemplo de extracción e integración de datos. [6]

Aquí hemos mencionado dos tipos de integraciones, la primera es unificar los datos o registros que refieren a la misma entidad y la segunda corresponde a agregar información de valor vinculando distintas entidades.

Se deberá profundizar en el análisis de este proceso y se espera que el mismo sea lo más automatizado posible.

4 Solución propuesta.

En este capítulo se presenta la solución propuesta del problema planteado, vamos a explicar el diseño de la solución junto con una descripción de los procesos que utilizamos, así como el entorno tecnológico utilizado esto es, la plataforma BPMS Activiti.

4.1 Proceso de Configuración.

Como mencionamos en la sección 3.1 se debe elegir un Dominio específico para el WW, lo realizamos ofreciendo al usuario la selección de un Dominio particular y las opciones estarán precargadas en el Sistema.

Una vez que ya seleccionamos el Dominio, debemos obtener las Web Sources que nos proveen datos referentes al mismo. Esto lo vamos a hacer de dos maneras, la primera es creando la Web Source en el Sistema, en donde podamos indicar un nombre para la misma, la url de donde el Sistema debe obtener los datos, etc. La segunda manera de utilizar Web Sources en nuestra ejecución de Proceso de Configuración es seleccionando Web Sources ya existentes en el Sistema que fueron creadas en ejecuciones previas, siempre y cuando éstas hayan sido creadas en el mismo Dominio que estamos utilizando en este momento.

Para la extracción de los datos de cada Web Source se debe pedir al usuario la definición del Expected Schema. Lo hará indicando el nombre de dicho esquema, y un conjunto de atributos con sus respectivos tipos de dato. Estos atributos representan los datos que el usuario desea extraer de todos aquellos publicados por la Web Source.

Una vez que el usuario define los campos que desea y le asigna un nombre propio dentro del Sistema, se deberá relacionar éstos con sus correspondientes en la publicación, esto es el mapping entre los datos provistos y los datos deseados. Cada atributo del esquema, debe referenciar a uno y solo un campo o columna provista por la Web, por lo que no puede haber más atributos de Expected Schema que atributos publicados por la Web Source.

Para resolver el problema de cómo extraer los datos, los Data Services serán independientes de nuestro Sistema y deben ser servicios publicados por alguna aplicación externa. Cada Data Service implementará la extracción de algún formato específico, por ello, para cada Web Source debemos especificar el Data Service que nos permite extraer dichos datos. Si una Web Source creada recientemente presenta un formato que no es soportado por ninguno de los Data Services disponibles, un técnico especializado deberá implementarlo para poder invocarlo en el proceso de Carga.

Como mencionamos anteriormente, el concepto de Expected Schema define la forma de extracción de los datos y además la manera de modelar los mismos en nuestro Sistema. Por lo que es importante destacar que más de una Web Source puede ser soportada por el mismo Expected Schema. Esto es, podemos desear obtener datos similares de distintas Web Sources pero le indicamos al Sistema que ambos tipos de registros obtenidos corresponden a mismas entidades. Por ejemplo, si la Web Source “A” nos provee “Hoteles” y utilizamos el Expected Schema “Expected Hoteles”, y a una siguiente Web Source le asociamos el “Expected Hoteles”, el Sistema interpretará que también estará manejando datos referente a Hoteles.

Hasta ahora hemos definido de dónde vamos a extraer los datos, cómo especificar cuáles son necesarios y cuáles no y de qué manera los vamos a extraer. Pero hasta ahora no hemos mencionado nada con respecto

a la integración, esto es, de qué manera vamos a crear un Integrated Schema que nos facilite la integración de todos los datos obtenidos. El Integrated Schema será una entidad de nuestro Sistema que estará compuesto por un conjunto de tablas y cada una de ellas con un conjunto de atributos, cuya función es relacionar distintos Expected Schemas y conformar una entidad concreta integrando distintas fuentes de datos. Será el usuario experto el encargado de ingresar en el Sistema cada una de las tablas y atributos.

Para poder realmente integrar los datos, vamos a definir un mapping entre cada atributo de cada tabla del Integrated Schema con uno o más atributos del Expected Schema, de esta manera sabremos en las tablas del Integrated Schema de donde proviene cada dato y qué es lo que se está integrando.

Una vez obtenido el Integrated Schema con todas sus tablas y atributos y realizado un mapping entre el Integrated Schema y el Expected Schema, debemos definir la estructura del DW que deseamos obtener. Para definir nuestro DW es necesario indicar las tablas que pertenecen a él, las tablas de dimensiones y las tablas de hechos así como también cada atributo junto con sus tipo de datos pertenecientes a las mismas. Luego de definido el esquema de DW, se debe volver a realizar un mapping entre cada atributo de cada tabla perteneciente al DW con uno o más atributos del Integrated Schema. Básicamente, estamos indicando al Sistema como queremos cargar dichas tablas en función de las tablas del Integrated Schema.

4.2 Arquitectura de Activiti utilizada

La Figura 21Figura 21 muestra la Arquitectura de Activiti que utilizamos para la resolución del problema planteado.

El motor de Activiti está embebido en nuestra aplicación de tipo cliente Java y para realizarlo, alcanza con incluir el motor y sus *jars* junto con el *jar* de la aplicación en el servidor de aplicaciones, que en nuestro caso utilizamos Tomcat 8. Una vez iniciado el mismo, la aplicación ya está disponible para ejecutarse.

Además, utilizamos un servidor de base de datos para alojar tanto las bases de datos de nuestra aplicación, como la base de datos que da soporte al motor de Activiti, en ambos casos utilizamos MySQL.

Para la capa de presentación de nuestro Sistema, utilizamos Activiti Explorer (aplicación Web provista por Activiti) y accedemos a la misma mediante cualquier browser por HTTP. [2]

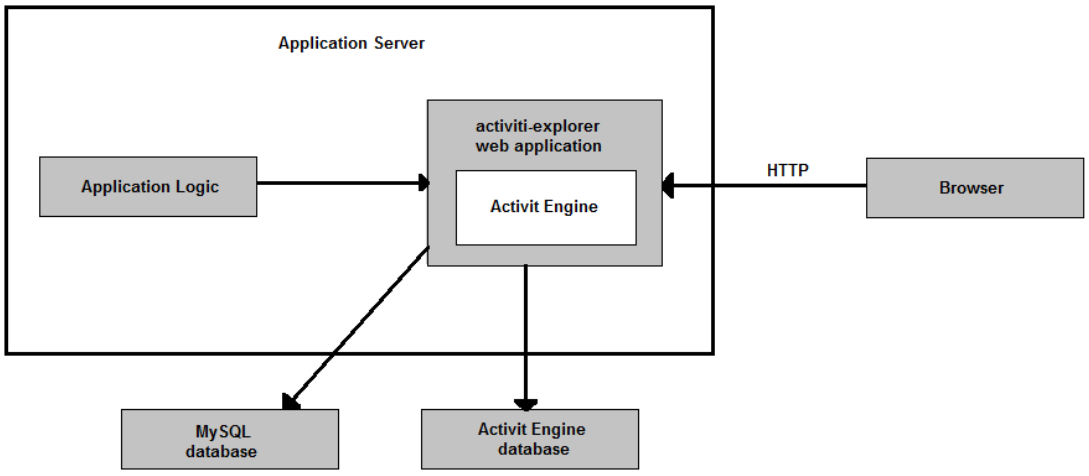


Figura 21 Arquitectura de Activiti utilizada

4.3 Descripción de la Arquitectura:

En esta sección se va a presentar la Vista Lógica de la arquitectura diseñada con todos sus subsistemas y la Vista de Distribución (*Deployment*) junto con la explicación de los nodos que forman parte de la misma.

4.3.1 Vista Lógica

La arquitectura que definimos está basada en tres capas que son la capa de presentación, lógica y persistencia. A su vez dicha arquitectura tiene un estilo Cliente – Servidor en el cual contamos con un servidor que expone servicios, en nuestro caso el portal de Activiti Explorer y los clientes que acceden a través del browser.

Como se puede ver en la Figura 22Figura 22, el prototipo está compuesto por diferentes subsistemas que tienen distintos roles:

- Subsistema Acceso a datos.
- Subsistema Servicios.
- Subsistema Lógica.

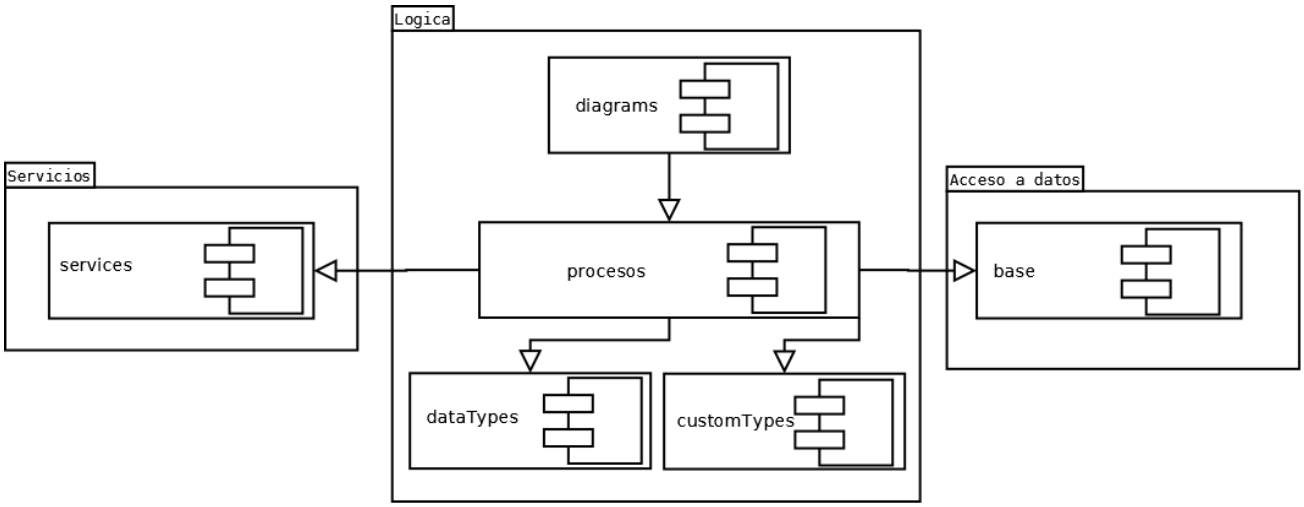


Figura 22 Arquitectura de la solución propuesta

4.3.1.1 Subsistema Acceso a datos.

El subsistema **Acceso a datos** encapsula el acceso a las bases de datos (ya sea a la base de metadatos como a otras bases de datos que se creen) y provee todos los métodos de consulta, de inserción y de actualización de registros en las mismas. Contiene una clase principal la cual implementa todos los métodos necesarios y utiliza JDBC para acceder al servidor de base de datos, en particular a la base central, que contiene las tablas de Activiti y las tablas de nuestra aplicación (base de metadatos), como también a cada base de datos generada por cada ejecución del proceso de Carga. Esto es, centraliza operaciones para el proceso de Configuración y también, centraliza las operaciones provistas a la implementación del proceso de Carga para consultar los datos de Configuración e insertar los registros procesados en las tablas de DW generado.

4.3.1.2 Subsistema Servicios.

El subsistema **Servicios** encapsula e implementa la comunicación del subsistema **Lógica** (subsistema principal de nuestra aplicación) con la/las aplicaciones encargadas de integrarse a las Web Source y extraer los datos. Estas aplicaciones no forman parte de nuestro Sistema y son quienes implementan el acceso directo a la fuente Web de datos y nos deberán proveer los mismos exponiéndolos como servicios Rest.

Servicios contiene un cliente Rest en el cual podemos configurar la url donde está publicado el servicio externo, y además podemos pasarle otros parámetros como son: la url que publica los datos y especificar qué datos queremos extraer y en qué formato están publicados.

Otro servicio que provee este módulo al Sistema, es el *parseo* de estructuras en formato *json* a objetos de negocio propios, como por ejemplo cuando creamos Expected Schemas o Integrated Schemas automáticamente.

4.3.1.3 Subsistema Lógica.

El subsistema **Lógica** es nuestro principal componente en cuanto a la lógica implementada para nuestro Sistema. Además de resolver dicha tarea, consume servicios de persistencia del subsistema **Acceso a datos** y servicios de extracción de datos del subsistema **Servicios**.

Lógica está compuesto por diferentes módulos que son **diagrams**, **procesos**, **dataTypes** y **customTypes** que a continuación procedemos a detallar.

En el módulo **diagrams** se encuentran ubicados todos los modelos de procesos necesarios para implementar los procesos de negocio tanto para el proceso de Configuración como para el de Carga.

El módulo **procesos** contiene todas las clases que implementan la lógica central de las operaciones del Sistema y además las clases que atienden a los procesos de negocio, como son los *listeners* y los *java delegate*. Este módulo contiene todas las clases que son invocadas desde **diagrams** para gestionar las actividades de Activiti, ya sean las manuales como las automáticas. Esto es, cada vez que se crea una tarea manual es posible que se ejecute una o varias clases en el evento *create* para mostrar alguna información. Cuando se ejecuta dicha tarea, también se invocará a otra u otras clases en el evento *complete* que gestionará los cambios a nivel de lógica que produjo dicha actividad. Por otro lado, cuando una actividad se ejecuta automáticamente es implementada también por una clase java. Todas estas clases que mencionamos forman parte del módulo **procesos.**

Por otro lado, además de los tipos primitivos que nos provee Activiti, debimos implementar otros tipos de datos para mostrar información más compleja en los atributos de los formularios. Dicha implementación está incluida en el modulo **customTypes**.

Finalmente, el módulo **dataTypes** provee de objetos básicos para manipular entidades del negocio y sus respectivas operaciones.

4.3.2 Vista Distribución (Deployment)

El diagrama planteado en la Figura 23 muestra la arquitectura, la cual permite tener una distribución en servidores diferentes y como se distribuyen los distintos componentes que serán descritos a continuación.

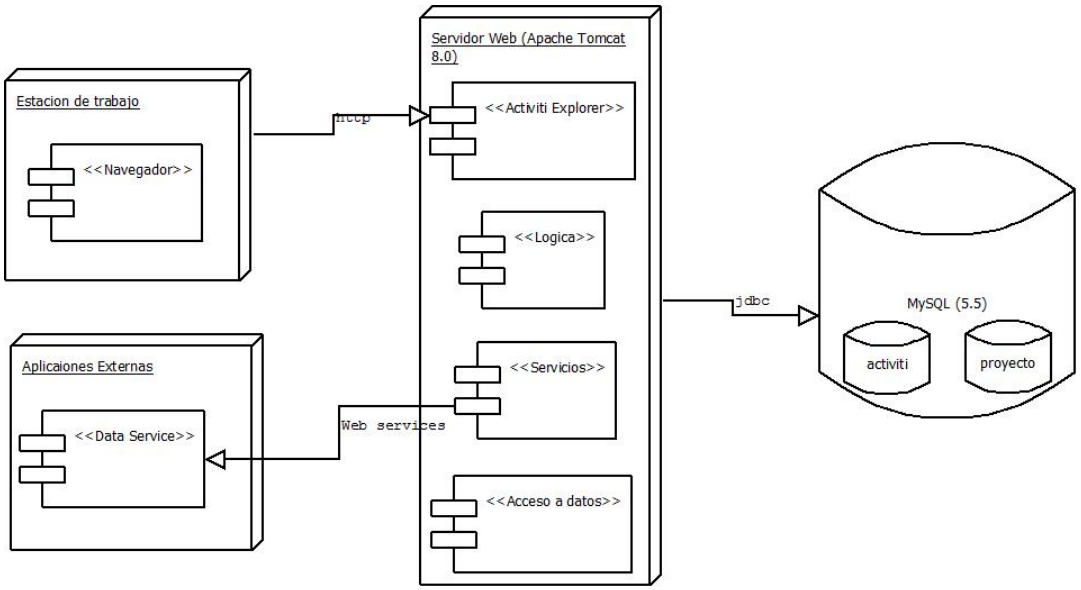


Figura 23 Diagrama de Deployment

Nodos:

Servidor Web: es el que contiene la capa de presentación, capa de servicios, capa de negocio y acceso a datos. Este nodo ejecuta un servidor de aplicaciones web Tomcat 8.0 que además de soportar los componentes ya mencionados, en él se ejecuta la aplicación web provista por Activiti llamada "Activiti Explorer" y básicamente consiste en la interfaz web que incluye todos los formularios de las actividades manuales. Por otro lado, también tenemos el motor de BPM que nos provee Activiti, el cual es *deployado* en conjunto con la aplicación web mencionada anteriormente.

Estación de Trabajo: este componente representa el navegador donde el usuario utiliza la aplicación. En particular, se conecta al Activiti Explorer, que es la interfaz web provista por Activiti publicado por el Servidor Web como mencionamos en el nodo anterior.

Aplicaciones Externas: en este nodo se ejecuta la aplicación que encapsula servicios Rest que implementan la extracción de los datos desde la Web. Estos servicios son consumidos desde el Sistema para cargar el DW utilizando los metadatos obtenidos en el proceso de Configuración. Si bien se ejecuta en el mismo servidor, es posible *deployarla* en otro servidor si fuese necesario ya que la invocación, por definición de Rest, es remota.

Servidor de base de datos: este nodo es el que contiene el servidor MySQL de base de datos. En él se encuentran: la base de datos del Activiti ("activiti"), la base de metadatos que contiene los datos generados desde el proceso de Configuración ("proyecto") y todas las bases generadas a partir de la construcción y carga de los DW. Es posible en un futuro si fuera necesario tener todas estas bases en servidores de base de datos diferentes.

4.4 Diseño del Modelo de Base de Metadatos.

La Figura 24 contiene el diagrama del Modelo de la Base de Metadatos de la solución propuesta. En el mismo se especifican las tablas pertenecientes a ella, los atributos de cada una de las tablas (con sus tipo de dato) y las relaciones entre dichas tablas. Toda esta estructura será de constante consulta cuando estemos cargando nuestro DW, ya que la forma de cargarlo estará determinada por el contenido de estas tablas, esto es, los metadatos mencionados anteriormente serán alojados en estas estructuras.

En el modelo, en el recuadro rojo se aprecian las tablas que se utilizan para las Web Sources y el Expected Schema, dichas tablas están explicadas en la Tabla 1, por otro lado en el recuadro marrón del modelo se ven las tablas que refieren al Integrated Schema y se detallan en la Tabla 2 y por último en el recuadro negro se observan las tablas que se utilizan para el DW Schema y se explican en la Tabla 3.

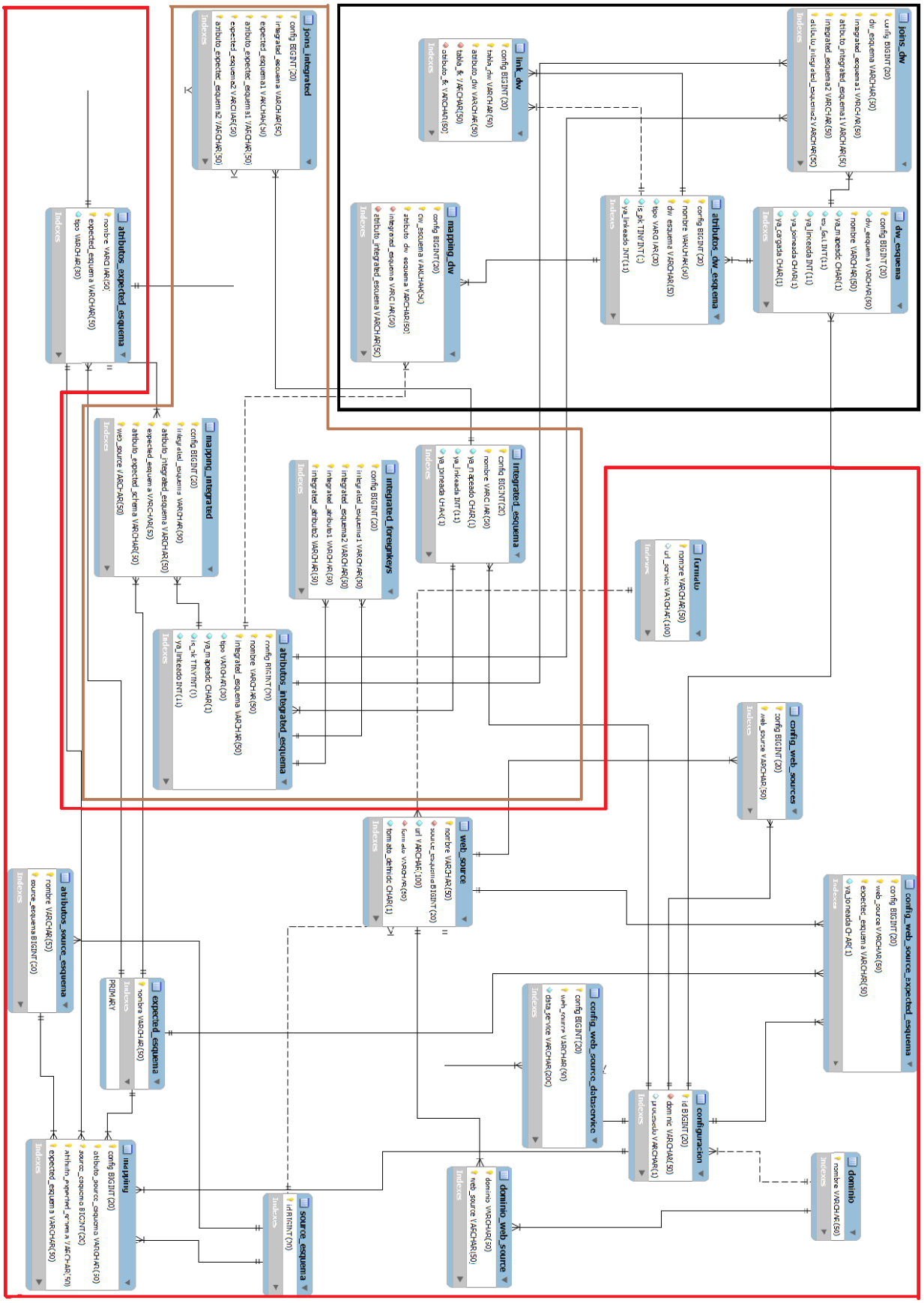


Figura 24 Modelo de la Base de Metadatos.

En la Tabla 1, Tabla 2 y Tabla 3 vamos a explicar las tablas del modelo de la base de metadatos, qué almacena cada una y en qué actividad del proceso de Configuración se utilizan.

Observar que las tablas **expected_esquema** y **web_source** son globales a las configuraciones, mientras que las tuplas de **integrated_esquema** y **dw_esquema** tienen una Configuración asociada. Es decir que, en estos últimos casos, cada tupla es una tabla perteneciente al Integrated Schema o DW Schema respectivamente, para una Configuración dada.

Tabla 1 Descripción de tablas de base de metadatos referente a las Web Sources y Expected Schema.

TABLA	DESCRIPCIÓN	ACTIVIDAD QUE LA UTILIZA
dominio	Esta tabla contiene todos los dominios disponibles en el Sistema. Estos registros son precargados en la base de metadatos.	Select Domain
configuracion	En esta tabla almacenamos las configuraciones que se ejecutan junto con el dominio correspondiente.	Select Domain
formato	Esta tabla contiene los formatos disponibles que existen en el Sistema para la extracción de los datos. Además, contiene la url del Data Service por defecto que implementa dicha extracción. Toda esta información es precargada en la base de metadatos mediante un script.	-
web_source	Esta tabla contiene todas las Web Source existentes en el Sistema, esto es, las que estamos utilizando en el proceso de Configuración y también las que se utilizaron en ejecuciones anteriores.	Create Web Source
dominio_web_source	En esta se almacena las relaciones entre los Dominios y las Web Sources	Create Web Source Select Web Source
config_web_source	Cada registro de esta tabla tiene un identificador de Configuración y un identificador de Web Source.	Create Web Source Select Web Source
config_web_source_dataservice	Esta tabla contiene la relación entre la entre la Web Source y el Data Service que implementa la extracción de los datos para una configuración dada.	Upload Data Service
source_esquema	En esta tabla almacenamos el identificador del esquema que provee web_source	Create Web Source
atributos_source_esquema	Esta tabla almacena todas las columnas que provee el Web Source a partir de una pequeña extracción del primer registro provisto (sea el titulo o la primer fila de datos)	Create Web Source

expected_esquema	Esta tabla contiene los nombres de cada uno de los Expected Schema que utiliza nuestro proceso de Configuración o los que se crearon en ejecuciones anteriores	Create Expected Schema and Atributes
atributos_expected_esquema	Esta tabla contiene los atributos asociados a cada uno de los Expected Schema.	Create Expected Schema and Atributes Add More Atributes for Expected Schema
config_web_source_expected_schema	Esta tabla almacena la relación entre la Configuración que estamos ejecutando, el Expected Schema y la Web Source. Por lo que tendremos un identificador de una Configuración, un nombre de una Web Source y un nombre de un Expected Schema.	Define Expected Schema Create Expected Schema and Atributes
mapping	Cada registro de esta tabla vincula un atributo del Source Schema con un atributo del Expected Schema para la Configuración que estamos ejecutando.	Mapping Expected Schema and Web Source

Tabla 2 Descripción de tablas de base de metadatos referente al Integrated Schema.

TABLA	DESCRIPCIÓN	ACTIVIDAD QUE LA UTILIZA
integrated_esquema	Esta tabla contiene los nombres de cada una de las tablas del Integrated Schema que utiliza nuestro proceso de Configuración	Create Table And Atributes for Integrated Schema
atributos_integrated_esquema	Esta tabla contiene los atributos asociados a cada una de las tablas del Integrated Schema.	Create Table And Atributes for Integrated Schema Add More Atributes for Integrated Schema
mapping_integrated	Cada registro de esta tabla vincula un atributo del Expected Schema con un atributo del Integrated Schema para la Configuración que estamos ejecutando.	Mapping Integrated Schema and Expected Schema
integrated_foreignkeys	Cada registro de esta tabla representa una clave foránea entre tablas del mismo Integrated Schema.	Define Foreign Keys Integrated Schema
joins_integrated	En esta tabla se almacena cómo se vinculan las tablas del Integrated Schema para cargar posteriormente las tablas del DW Schema.	Define Joins Integrated Schema

Tabla 3 Descripción de tablas de base de metadatos referente al DW Schema.

TABLA	DESCRIPCIÓN	ACTIVIDAD QUE LA UTILIZA
dw_esquema	Esta tabla contiene los nombres de cada una de las tablas del DW Schema que utiliza nuestro proceso de Configuración	Create Table And Atributes for DW Schema
atributos_dw_esquema	Esta tabla contiene los atributos asociados a cada una de las tablas del DW Schema.	Create Table And Atributes for DW Schema Add More Atributes for DW Schema
mapping_dw	Cada registro de esta tabla vincula un atributo del Integrated Schema con un atributo del DW Schema para la Configuración que estamos ejecutando.	Mapping DW Schema and Integrated Schema
link_dw	Cada registro de esta tabla representa una clave foránea entre tablas del mismo DW Schema.	Define Foreign Keys of Fact Tables for DW Schema
joins _dw	En esta tabla se almacena cómo se vinculan las tablas del DW Schema para cargar posteriormente las tablas de hechos del DW Schema.	Define Joins DW Schema

4.5 Proceso de Carga.

En la sección anterior pudimos obtener los metadatos de construcción de un DW, ahora debemos especificar de qué manera utilizando dicha Configuración vamos a lograr construir y cargar el DW.

Deseamos que este proceso sea completamente automático pero esto no es posible, ya que es muy probable que el Sistema encuentre conflictos de datos y problemas de integración que con la participación del usuario se deben gestionar.

La solución estará compuesta por un conjunto de actividades automáticas y las inconsistencias se resolverán con actividades manuales.

La primera tarea automática que definimos se encarga de crear todas las estructuras de nuestro DW. En primera instancia se creará una base de datos la cual va a contener toda la información que genere el DW, esto es, las tablas y los registros indicados en la Configuración seleccionada. Si no es la primera vez que ejecutamos un proceso de Carga para dicha Configuración, la base de datos mencionada ya existirá y el Sistema procede a borrar los datos que estén en las estructuras, pero se reutilizan la base de datos y dichas estructuras.

Una vez creada la base de datos, en ella vamos a generar todas las tablas que se necesitan para extraer los datos, integrarlos y además las tablas asociadas al DW, como se mencionaron anteriormente, las tablas de dimensiones y tablas de hechos. Por lo que una vez terminada esta tarea, vamos a contar con una tabla destinada a alojar los datos extraídos de la Web, esto es, una tabla por cada Web Source seleccionada (estas tablas que mencionamos serán utilizadas de manera temporal para poder persistir los datos extraídos e independizarnos de la disponibilidad de la Web Source). Para cada Expected Schema también tendremos una tabla, allí estarán todos los registros asociados a cada entidad representada por dicho Expected Schema. Por otro lado, también tendremos las tablas asociadas al Integrated Schema.

Luego de tener todas las estructuras creadas, necesitamos extraer los datos de la Web, para esto, decidimos extraer todos los registros provistos por cada Web Source con todos sus campos, invocando al Data Service que implementa la extracción. Estos registros se guardarán en las tablas de las Web Source correspondiente mencionadas en el párrafo anterior.

A continuación, utilizando la configuración del Expected Schema se cargarán las tablas asociados a éstos, utilizando como fuente las tablas de las Web Source que se cargaron recientemente.

Luego, en función de los metadatos de integración vamos a cargar las tablas generadas para el Integrated Schema, utilizando las tablas del Expected Schema y la relación entre ellos que se configuro inicialmente.

En este momento, contamos con todo el Integrated Schema cargado completamente, por lo tanto lo que sigue es cargar las tablas del DW.

Previamente a la carga del DW, el Sistema va a generar y mostrar al usuario la consulta SQL que utilizará para cargar cada tabla de hechos desde las tablas del Integrated Schema. Por cada una de estas SQL el usuario decide si confirma la misma o de lo contrario la modifica y la confirma posteriormente. Una vez que tenemos todas las SQL confirmadas, el Sistema procede a la carga final del DW y nos permite confirmar esta operación si es que la carga le fue satisfactoria o deshacer la misma y volver a los pasos de modificación/confirmación de las SQL.

Luego de cada tarea automática, está prevista una tarea manual que se dedique a la resolución de conflictos de datos detectados. Contamos con una tarea manual para resolver conflictos de extracción de datos, de integración y carga del Expected Schema y análogamente para el Integrated Schema, que como trabajo a futuro se podrá implementar el comportamiento de las mismas.

5 Desarrollo del prototipo.

En este capítulo presentamos la implementación del prototipo que realizamos como prueba de conceptos de la solución propuesta. Como estaba planteado, se implementaron entonces dos procesos BPM que resuelven respectivamente cada módulo, esto es: Configuration.bpmn y Feeding.bpmn.

Implementamos una plataforma de Construcción de WW realizada en Java y utilizamos como motor de procesos a *Activiti* y *Business Process Model And Notation* (BPMN 2.0), que define un formato estándar de representación e intercambio de modelos (XML) y semántica de ejecución.

Como interfaz de usuario utilizamos la herramienta provista por Activiti que se llama Activiti Explorer, la cual es una interfaz web.

5.1 Especificaciones técnicas.

Para lograr la implementación de nuestro prototipo utilizamos como servidor de aplicación Apache Tomcat 8.0. El desarrollo lo hicimos en Eclipse versión Kepler con el plugín de Activiti para Eclipse Activiti Eclipse BPMN 2.0 Designer 5.15. El motor de base de datos es MySql 5.5.

Para la implementación de los procesos utilizamos en cada actividad manual, una clase java en el evento *create* (dicho evento se ejecuta antes de que la tarea sea presentada al usuario) la cual implementa la interface *TaskListener,* para que mediante el módulo de acceso a BD pueda conectarse a nuestra base de metadatos y así presentar toda la información necesaria al usuario, como pueden ser los combos con los datos a seleccionar, etc. (esto se hace sobrescribiendo el método *notify)*. Además en el evento *complete,* que se ejecuta luego de completada la tarea invocamos otra clase java (que también implementa la interface *TaskListener* y sobrescribe el método *notify*) que se encarga de comunicarse con la capa de persistencia para lograr grabar en la base de metadatos la información ingresada por el usuario.

En la Figura 25 podemos ver un ejemplo de modelo de proceso visto desde el Eclipse, en el cual están marcados en la pestaña *General* donde se configura el *id* de la actividad y el nombre de la misma. Además es posible configurar los campos de los formularios de las actividades, parametrizar los tipos de datos de los campos, si son obligatorios o no, si son solo de lectura, y otras propiedades (en la pestaña *Form*). Otras configuraciones interesantes son la de los usuarios o grupos de usuarios que ejecutan las tareas (pestaña *Main Config*) y la configuración de las clases java que implementan las funcionalidades de las actividades y en que evento se va a ejecutar (*create*, *assignment*, *complete*,*all*).

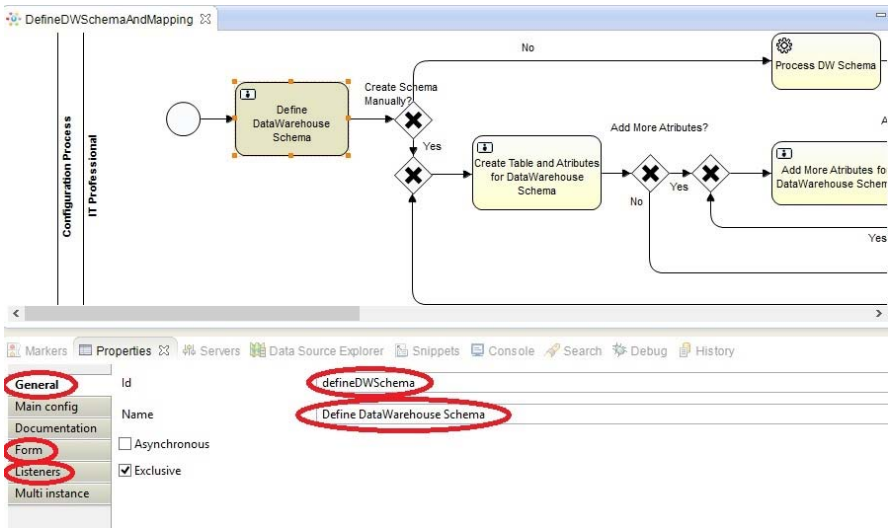


Figura 25 Muestra las configuraciones de una actividad dentro del Eclipse

Para ofrecer las columnas de las Web Sources el Sistema realiza una extracción del primer registro de los datos publicados por dicha fuente. La idea de este paso es facilitarle al usuario la realización del mapping entre los campos de la fuente Web y del Expected Schema que se definió. Para ello en el paquete *services* tenemos la clase *ExtractSourceSchema* la cual implementa la extracción desde la fuente Web del título de sus columnas o en su defecto de la primer tupla de datos mediante la invocación de un servicio Rest tomando en cuenta el formato de la fuente de datos configurado. En caso de que la fuente de datos tenga un formato que no es soportado, en vez de presentarle al usuario combos cargados con el título de las columnas o con los datos de la primer tupla, se le despliega un campo de texto en el cual ingresara manualmente dichos datos, que deben coincidir exactamente con los publicados por la Web Source.

Por otro lado, para la creación en el Sistema de los Expected Schemas, Integrated Schema y DW Schema, el usuario tiene la posibilidad de crearlos manualmente mediante un conjunto de tareas o podrá ingresarlo mediante un *json*, con esta finalidad implementamos en el paquete de *services* la clase que se encarga de *parsear* el *json* con las definiciones de los esquemas. Para el parseo de los *json* utilizamos la librería de "gson – 2.2.4". A su vez para implementar esta carga de esquemas mediante *json* tenemos un conjunto de actividades automáticas (al igual que en el Proceso de Carga) las cuales tienen un tipo de tarea (*task type*) configurada como una clase java, esta clase implementa la interface *JavaDelegate* y la lógica de dicha tarea se encuentra en el método *execute*.

Con respecto a la comunicación entre el módulo de acceso a datos y las bases de datos están implementadas con JDBC tal como lo muestra la Figura 23.

5.2 Implementación Proceso de Configuración.

A continuación presentaremos la implementación del Proceso de Configuración, las actividades que están incluidas y cada subproceso participante del mismo. Las actividades referentes a las definiciones y mappings de los Expected Schema, Integrated Schema y DW Schema fueron incluidas en sus respectivos subprocesos.

La explicación de esta sección estará fraccionada en distintas partes, una destinada a *"Selección de Dominio y Web Sources"* (finalizadas estas etapas ya contaremos con la definición del área temática en donde vamos a trabajar y de las distintas fuentes Web que nos proveerán los datos), otra que refiere a *"Definición y Mapping del Expected Schema"* (actividades que nos permiten crear los Expected Schemas para poder indicarle al Sistema qué datos de todos los publicados son relevantes para nosotros y además poder vincular

lo que nosotros esperamos con los que extraemos), la siguiente llamada *"Definición y Mapping del Integrated Schema"* (brindará actividades asociadas a la creación de tablas y atributos del Integrated Schema y además, poder relacionar cada una de éstas con los Expected Schema que necesitamos, de manera de obtener entidades con valor agregado y datos más integrados), y luego la parte de *"Definición y Mapping del DW Schema"* (en donde comenzamos a definir lo que es el DW que necesitamos, las tablas de dimensiones y de hechos con sus respectos atributos, y además, indicando cómo se cargan cada una de ellas con los datos existentes en las tablas del Integrated Schema)

En la Figura 26 se muestra el proceso de Configuración modelado en BPMN 2.0 con el plug-in de Eclipse Activiti Designer.

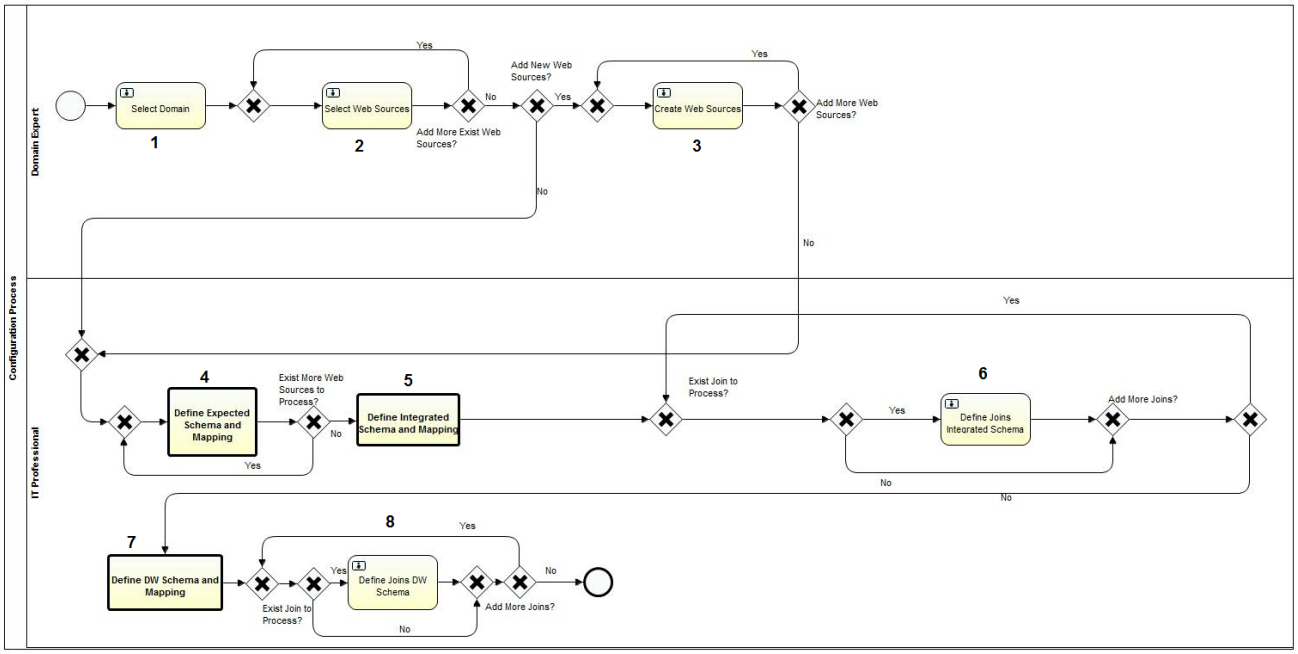


Figura 26 Proceso de Configuración

A continuación se detalla cada una de las actividades y sub-procesos definidos, su funcionamiento e implementación asociada.

5.2.1 Selección de Dominio y Web Sources.

Como podemos ver en la Figura 26, en la parte superior del proceso de Configuración tenemos tres actividades manuales que nos van a permitir mediante formularios de Activiti Explorer, crear en nuestro Sistema la Configuración con la que vamos a trabajar, asociarle un Dominio y cargar las Web Sources de las cuales queremos extraer los datos.

Estas Web Sources las podemos seleccionar desde la base de metadatos (porque ya fueron utilizadas en procesos de Configuración anteriores) o directamente darlas de alta en el Sistema. Las actividades son, *"Select Domain"* (Ver 1 en Figura 26) para seleccionar los Dominios (éstos ya están precargados en la base de metadatos), *"Select Web Source"* (Ver 2 en Figura 26) para las Web Source ya existentes y *"Create Web Source"* (Ver 3 en Figura 26) para crear nuevas fuentes de datos. Luego de ejecutar la actividad *"Select Web Source"* (Ver 2 en Figura 26), existe un XOR que permite al usuario decidir si ejecutar nuevamente la actividad para cargar nuevas Web Sources a la ejecución o si de lo contrario continuar con el flujo.

Análogamente existe otro XOR luego de *"Create Web Source"* (Ver 3 en Figura 26). Además, de esas decisiones, el usuario puede determinar directamente no ejecutar en ningún momento la actividad de *"Create Web Source"* (Ver 3 en Figura 26), ya que puede interpretar que no es necesario.

Select Domain

La actividad *"Select Domain"* está incluida en el comienzo del proceso de Configuración y es la que se encarga de (manualmente) indicar al Sistema en qué contexto deseamos construir nuestro WW. Esto es, podemos querer modelar la realidad de distintas áreas de interés como pueden ser Turismo, Transporte, Organismos Públicos, Salud, etc y debemos elegir una de ellas. Esto que llamaremos Dominio en nuestro Sistema no será necesario ingresarlos, ya existirán y nosotros debemos seleccionarlo en un combo que nos ofrecer la actividad. Seleccionar el Dominio con el cual vamos a trabajar nos va a permitir ocultar datos y registros que el Sistema ya sabe que no son relevantes para dicho contexto y ofrecernos otras que sí lo son, y un ejemplo son las Web Source que ya existen en el Sistema y este entiende que de alguna manera son relevantes para el análisis que elegimos. Esto se detallará a continuación en las actividades de selección y alta de Web Sources.

Select Web Source

En esta actividad el Sistema nos ofrecerá con un combo en el formulario, todas las Web Source que existen en la base de metadatos, que están vinculadas con el Dominio que elegimos (como mencionamos en los párrafos anteriores), podremos elegir todas las Web Source disponibles que queramos y entendamos que nos serán de utilidad.

Esto es posible, gracias a que el Sistema mantiene una relación entre el Dominio y las Web Source utilizadas para dicho contexto. La relación que nos referimos, no está precargada si no que en alguna ejecución pasada del proceso de Configuración el usuario creo la Web Source en el Sistema para el mismo Dominio que seleccionamos en este momento, de allí viene esa relación y se persiste en la base de metadatos para siempre).

Create Web Source

En esta actividad es el momento en que se dan de alta las fuentes Web en el Sistema y para eso es necesario además de indicar un nombre para identificarla, también debemos indicar la url en donde están publicados los datos y qué formato se utilizó para dicha publicación.

El usuario podrá en esta instancia crear todas las Web Source que desee, y queda a su criterio si las mismas contienen datos de interés para el contexto o Dominio que estamos configurando. Por lo que es aquí cuando se crea en el Sistema una relación entre el Dominio y cada Web Source creada. Así como se comentó en la actividad *"Select Web Source",* que el Sistema nos ofrece Web Sources relacionadas al Dominio, es dicha relación la que nos permite hacer eso.

Entonces, finalizadas las tres actividades anteriores, ya podemos decir que tenemos creados en el Sistema la Configuración (nos referimos a la entidad), la relación con el Dominio elegido, y el conjunto de Web Sources que nos proveerán los datos.

A continuación profundizaremos sobre estas actividades desde el punto de vista de la interacción con la base de metadatos y el impacto que tiene la ejecución de las mismas sobre los datos persistidos.

Interacción con Base de Metadatos:

Select Domain

Cuando se carga el formulario *"Select Domain"* se ofrece un combo con distintos dominios para que el usuario seleccione uno específico. Estos dominios están precargados en la base de metadatos y los *inserts* asociados están incluidos en el script disponible que crea todas las estructuras necesarias, por lo que no es posible para un usuario dar de alta dominios, sólo utilizará alguno de esos.

Cuando se completa el formulario antes mencionado, se crea la entidad que representa a la Configuración que estamos procesando, incluyendo un identificador y una referencia a una entidad de Dominios. Por lo que se inserta un registro en la tabla **configuracion** con un **id** autogenerado y un nombre de **dominio** que identifica al mismo.

Select Web Source

Cuando se carga el formulario "*Select Web Source*" ofrecemos al usuario seleccionar Web Sources ya existentes, realizamos una consulta en la base de metadatos sobre la tabla **dominio_web_source** y utilizamos como filtro de la consulta el campo **dominio** que referencia al nombre del Dominio con el que estamos trabajando.

Cuando se completa el formulario antes mencionado, se deberá persistir la relación entre la Configuración y las distintas Web Sources que hayan sido seleccionadas. Para ello, insertamos registros en la tabla **config_web_sources**, cargando los campos **config** y **web_source** referenciando a las tablas **configuracion** y **web_source** respectivamente.

Create Web Source

Cuando se carga el formulario *"Create Web Source"* sólo consultamos a la base de metadatos para ofrecer al usuario los distintos formatos que una Web Source puede tener, para ello obtenemos los registros de la tabla **formato** sin filtro, ya que necesitamos todos los disponibles. Estos formatos también están precargados en la base de metadatos, de la misma manera que hicimos con los dominios.

Cuando se completa el formulario antes mencionado, se deberá persistir la nueva entidad creada Web Source y además, las relaciones de ésta con el Dominio y la Configuración. Para ello, insertamos registros en las tablas **web_source**, **config_web_sources** y **dominio_web_sources**. En la primera, cargamos los campos **nombre**, **url y formato** con los datos que el usuario cargó en el formulario, este último es una referencia a la tabla **formato**. En la segunda tabla, sólo persistimos la relación entre la Configuración y la Web Source, insertando en los campos **config** y **web_source**, las referencias a las tablas **configuracion** y **web_source** respectivamente. En la tercera tabla, persistimos la relación entre el Dominio y la Web Source, insertando en los campos **dominio** y **web_source**, las referencias a las tablas **dominio** y **web_source** respectivamente. Como vimos en la actividad anterior, esta última relación será utilizada para ofrecer en futuras ejecuciones, Web Sources asociadas a dominios específicos. Otro impacto en la base de metadatos que existe en esta actividad pero es de funcionamiento interno, es la generación de una pequeña extracción de la fuente para poder saber qué datos nos provee, como se explicó oportunamente en dicha actividad. Con esas columnas extraídas, se insertan registros en las tablas **source_esquema** y **atributos_source_esquema**, cargando en el campo **id** autogenerado en la primera tabla, y los campos **nombre** y **source_esquema** en la segunda, que indican el nombre del atributo y una referencia a la tabla **source_esquema**, respectivamente.

5.2.2 Definición y Mapping del Expected Schema.

La parte inferior del modelo, está destinada a definir la forma de extracción de los datos, la integración entre ellos y los mappings correspondientes que ya explicaremos.

Para definir los Expected Schema y cómo se mapean éstos con sus Web Sources tenemos un subproceso de nombre *"Define Expected Schema and Mapping"* (Ver 4 en Figura 26) de manera de encapsular dicha funcionalidad.

Como se puede ver en la imagen anterior, este subproceso se encuentra en un bucle determinado por un XOR. Como este subproceso debe ejecutarse para cada Web Source seleccionada anteriormente, el XOR determina automáticamente si aún quedan Web Sources para procesar.

A continuación vamos a explicar dicho subproceso.

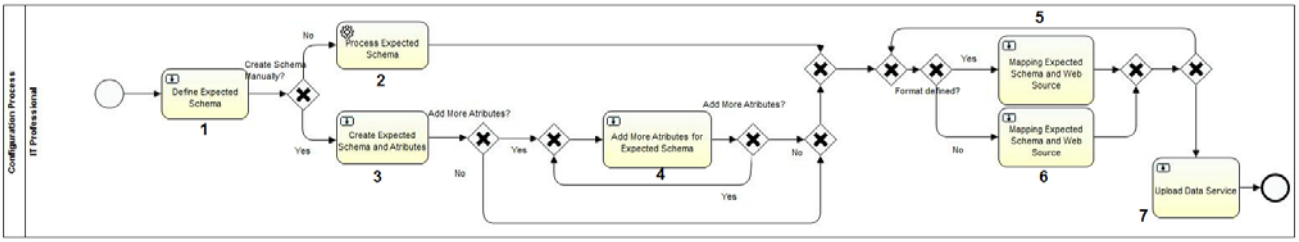


Figura 27 Subproceso Define Expected Schema and Mapping

La Figura 27 muestra el subproceso para definir un Expected Schema y además realizar el mapping correspondiente a cada atributo del mismo. Para asociar un Expected Schema a una Web Source tenemos tres maneras distintas: la primera es utilizar un Expected Schema que ya fue creado anteriormente en alguna otra ejecución, la segunda nos permite crearlo automáticamente ingresando un *json* con toda la estructura necesaria (esto es, el nombre y los atributos con su tipo) y la tercera y última es ingresando toda la información requerida de forma manual. Para poder tomar esta decisión y realizar un correcto flujo en proceso utilizamos la actividad *"Define Expected Schema"* (Ver 1 en Figura 27) y el XOR a continuación.

Define Expected Schema

Como ya dijimos está actividad nos permitirá re direccionar el flujo según como queramos crear el Expected Schema, para poder realizar esto el formulario nos presenta un campo de texto libre en donde podremos ingresar el *json* completo si es que deseamos crearlo automáticamente *parseando* un *json*. Por otro lado, también disponemos de un combo, que nos lista todos los Expected Schema disponibles en el Sistema que podremos elegir reutilizar y finalmente un check box, que activado significa pasar a la creación manual de un Expected Schema. En el flujo vemos como las dos primeras opciones se van a la actividad *"Process Expected Schema"* y la última a la actividad *"Create Expected Schema and Atributes"*.

Process Expected Schema

Si elegimos crear el Expected Schema automáticamente, ingresamos el *json* en la actividad mencionada anteriormente y luego ejecutamos la actividad automática *"Process Expected Schema"* (Ver 2 en Figura 27), que *parsea* el *json* creando los metadatos asociados al Expected Schema y persistiendo toda la información.

En el *json* deben estar presentes además del nombre del Expected Schema, el conjunto de todos los atributos que le pertenecen. Una vez que finaliza la ejecución de esta tarea, ya estará disponible en el Sistema dicho Expected Schema.

Si elegimos crear el Expected Schema utilizando alguno ya existente, el flujo también continúa por la bifurcación superior, y la actividad automática sólo realizará la vinculación del Expected Schema ya existente con la Configuración y la Web Source que estamos procesando.

Create Expected Schema and Atributes y Add More Atributes for Expected Schema

Por último, si optamos por la manera manual debemos ejecutar algunas actividades manuales para crear el Expected Schema con su nombre y atributos con sus tipos. Estas actividades son, *"Create Expected Schema and Atributes"* (Ver 3 en Figura 27), en la cual debemos indicar el nombre del Expected Schema y un conjunto de 5 atributos con su correspondiente tipo. Además, debemos indicarle al Sistema si deseamos agregar más atributos al Expected Schema mediante un checkbox (esta manera de decidir si agregar más atributos la utilizamos para otras decisiones a lo largo del proyecto y se debe a una restricción del Activiti Explorer en la cual la forma de visualizar los formularios es estática ya que los componentes que este muestra son fijos). En caso de querer agregar más atributos, el flujo nos lleva a una nueva actividad manual llamada *"Add More Atributes for Expected Schema"* (Ver 4 en Figura 27), donde cada ejecución de esta actividad nos permite crear otros 5 atributos más y además un checkbox para volver a ejecutarla y así sucesivamente.

Luego de ejecutadas las dos últimas actividades, ya está disponible en el Sistema el Expected Schema con el conjunto de atributos definidos.

Mapping Expected Schema and Web Source

Una vez que ya tenemos el Expected Schema con sus atributos asociado a la Web Source, debemos indicarle al Sistema la correspondencia entre cada atributo del Expected Schema con cada campo provisto por la Web Source, esto es lo que denominamos el mapping entre el Expected Schema y la Web Source. Esto lo vamos a realizar con la actividad manual *"Mapping Expected Schema and Web Source"* (en realidad son dos pero ya vamos a explicar por qué).

En caso de que el formato de la Web Source tenga una implementación de extracción ya establecida en el Sistema, se ejecuta la actividad de *"Mapping Expected Schema and Web Source"* (Ver 5 en Figura 27) en la cual podemos asociar a cada atributo del Expected Schema con el nombre de una columna de la fuente Web (para ofrecer las columnas de la fuente web, el sistema realiza una pequeña extracción de los datos para facilitarle al usuario la realización del mapping). Si el formato que mencionamos no es soportado por el Sistema, el comportamiento de la actividad manual cambia un poco, en vez de ofrecer el listado de columnas disponibles por la Web Source, el usuario debe ingresar manualmente cada una de ellas para realizar el mapping (Ver 6 en Figura 27). Estas columnas creadas por el usuario deben estar correctamente escritas (se deben llamar igual que la columna correspondiente en la Web), ya que el Sistema las va a utilizar a la hora de extraer los datos mediante el Data Service asociado.

Una vez que ejecutamos las actividades descritas anteriormente, ya tenemos el Expected Schema registrado en el Sistema, sus atributos y una correspondencia para cada uno de ellos con una columna de los datos publicados por la Web Source. Notar que el Expected Schema se puede repetir para distintas Web Sources, pero esta tarea de mapping se debe realizar para cada una de ellas.

Upload Data Services

La actividad *"Upload Data Services"* debe ser ejecutada manualmente y la utilizaremos para indicarle al Sistema en qué *endpoint* está publicado el Data Service que se encargará de la extracción de datos de la fuente Web. El Sistema ofrecerá en pantalla el Data Service por defecto que ya tiene asociado la Web Source (precargado en la base de metadatos para el formato seleccionado en la Web Source) y el usuario tendrá la opción de cambiarlo. Cuando nos referimos a Data Service, es la url que publica el servicio (Ver 7 en Figura 27).

Interacción con Base de Metadatos:

Define Expected Schema

Cuando se carga el formulario "*Define Expected Schema*", se consultará a la base de metadatos para ofrecer al usuario utilizar un Expected Schema ya creado anteriormente en el Sistema para alguna Configuración. Para ello consultamos la tabla **expected_esquema** sin filtros, ya que queremos ofrecer todos los Expected Schemas disponibles.

Cuando se completa el formulario antes mencionado no se altera el estado de la base de metadatos.

Process Expected Schema

Para el caso en que el usuario haya seleccionado un Expected Schema ya existente, no será necesario crearlo como entidad en la base de metadatos, pero si persistir las relaciones entre la Configuración y el Expected Schema. Para esto, insertamos un registro en la tabla **config_web_source_expected_esquema**, cuyos campos **config**, **web_source** y **expected_esquema**, se cargarán con los identificadores de Configuración, Web Source y Expected Schema respectivamente.

Para el caso en que el usuario haya ingresado un *json*, el mismo se *parseará* de manera de obtener la entidad Expected Schema y sus atributos, por lo que a nivel base de metadatos persistiremos en las tablas **expected_esquema** y **atributos_expected_esquema** y de la misma manera que el párrafo anterior, la tabla **config_web_source_expected_esquema**. En la primera insertaremos un registro que contiene el nombre de la entidad y con la lista de atributos obtenida para el mismo, insertamos en la segunda tabla los campos **nombre**, **expected_esquema** y **tipo**, que refieren al objeto atributo y corresponden con el nombre del mismo, el Expected Schema y el tipo de atributo a nivel de base de metadatos, respectivamente.

Create Expected Schema and Atributes

La interacción entre esta actividad con la base de metadatos, es exactamente igual a la mencionada con la actividad *"Process Expected Schema"* en el caso en que hayan ingresado un *json*.

Add More Atributes for Expected Schema

Cuando se carga el formulario *" Add More Atributes for Expected Schema*" no se realizan consultas a la base de metadatos.

Cuando se completa el formulario antes mencionado, se deben crear en la base de metadatos los atributos ingresados en el mismo, como el Expected Schema ya existe sólo necesitamos insertar los atributos referenciando a este último. Para ello, insertamos un registro en la tabla **atributos_expected_esquema** para

cada atributo ingresado indicando en los campos **nombre**, **expected_esquema** y **tipo**, que refieren al nombre del atributo, una referencia a la tabla **expected_esquema** y el tipo de datos del atributo.

Mapping Expected Schema and Web Source

Cuando se carga el formulario *"Mapping Expected Schema and Web Source"* se realizarán algunas consultas a la base de metadatos para mostrar información. Un caso es para el campo del formulario Atribute Expected, en ese caso obtenemos aleatoriamente algún atributo del Expected Schema que aún no haya sido mapeado, obteniendo todos los atributos de dicho Expected Schema de la tabla **atributos_expected_esquema** y buscamos cuáles de estos no están en la tabla de atributos ya mapeados que explicaremos en el siguiente párrafo. Para todos los atributos que obtuvimos, tomamos el primero. Otra consulta que hacemos en la base de metadatos es para cargar el combo Column Web Source, para esto debemos *joinear* las tablas **web_source**, **source_esquema** y **atributos_source_esquema**. En la primera tabla filtramos por nombre de Web Source en el campo **nombre** y obtenemos el campo que referencia al Source Schema **source_esquema** que lo utilizamos para *joinear* con la tabla **source_esquema**, luego de eso, *hacemos join* con **atributos_source_esquema** también *joineando* por el campo **source_esquema**, y obtenemos de esa misma tabla los nombres de los atributos que resultaron en el campo **nombre**, que hace referencia al nombre del atributo.

Cuando se completa el formulario antes mencionado, insertamos un registro por cada mapping entre atributos del Source Schema y el Expected Schema en la tabla **mapping**, cargando los campos **config**, **atributo_source_esquema**, **source_esquema**, **atributo_expected_schema** y **expected_esquema**, que indican referencias a la Configuración, el atributos del Source Schema y del Expected Schema que estamos mapeamos, y el Expected Schema respectivamente.

Observación: lo mencionado en el párrafo que indica consultas a la base de metadatos para cargar el formulario, se realizará en caso de que estemos en la actividad de *"Mapping Expected Schema and Web Source"* que nos provee una primera extracción de columnas de los datos de la fuente Web. Si vemos el modelo del proceso, existe una actividad muy similar a ésta, pero que en vez de ofrecernos atributos del Source Schema ya extraídos, nos pide ingresarlos en texto libre, ya que no pudo realizar dicha extracción primaria. En cuanto al párrafo que hace mención a los nuevos registros de la base de metadatos, para éste caso que comentamos, debemos agregar que también se insertan registros en las tablas **source_esquema** y **atributos_source_esquema**, ya que el Sistema no pudo crearlos al momento en que seleccionamos la Web Source como se explicó en su momento.

Upload Data Services

Cuando se carga el formulario *"Upload Data Services"*, se ofrece al usuario utilizar el servicio de extracción que tiene asociado por defecto el formato de la Web Source. Esto lo vemos en la tabla **formato**, tiene un campo de nombre **url_service**, a su vez, la tabla **web_source**, tiene una referencia al formato como ya se mencionó. De esta manera podemos obtener dicha url precargada para ofrecerla por defecto.

Cuando se completa el formulario antes mencionado, lo que hacemos es insertar un registro en la tabla **config_web_source_dataservice** con los campos **config**, **web_source** y **data_service** que serán una referencia a la Configuración, a la Web Source y la url del servicio, respectivamente.

5.2.3 Definición y Mapping del Integrated Schema.

Una vez finalizado el subproceso de *"Define Expected Schema and Mapping"* (Ver 4 en Figura 26), pasamos a otro subproceso que encapsula la funcionalidad de definir el Integrated Schema, con sus tablas y cada atributo de las mismas y además realizar el mapping del Integrated Schema con los Expected Schema. Esto quiere decir que debemos indicar, para cada atributo de cada tabla del Integrated Schema, cuál o cuáles atributos del Expected Schema se mapearían o se corresponderían con el atributo mencionado. Una tabla del Integrated Schema, mejor dicho, sus atributos, se pueden mapear con atributos de una tabla del Integrated Schema o de varias. Podemos ver a las tablas del Integrated Schema como tablas que unifican o integran, más precisamente, distintos Expected Schemas conformando así entidades de valor agregado.

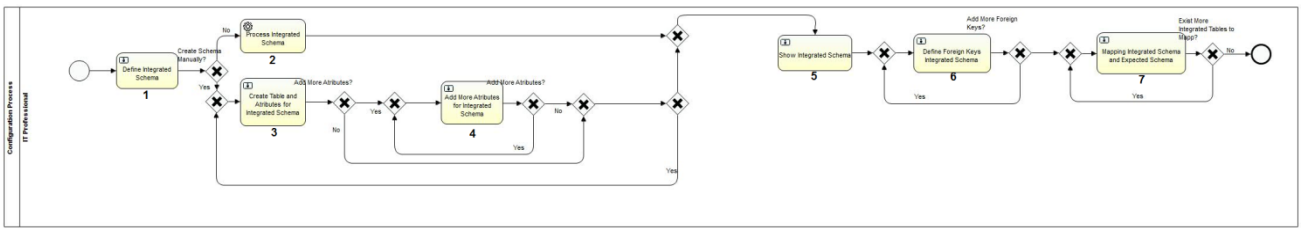


Figura 28 Subproceso de Define Integrated Schema and Mapping

La Figura 28 muestra el subproceso para definir un Integrated Schema, el mismo es muy análogo al presentado en la Figura 27 donde definíamos el Expected Schema, de todas formas presenta sus diferencias.

Define Integrated Schema

La actividad *"Define Integrated Schema"* (Ver 1 en Figura 28), nos permitirá elegir de qué manera vamos a crear las tablas y atributos del Integrated Schema, y al igual que cuando creamos el Expected Schema, tenemos distintas variantes pero esta vez, no tenemos a disposición seleccionar alguno ya existente, por lo que las opciones son dos, crear el Integrated Schema manualmente ó ingresar un *json* en el campo destinado para ello, y el Sistema lo procesará de manera automática creando el mismo en el Sistema.

Para indicar si queremos crear el Integrated Schema manualmente o por *json*, tenemos un checkbox que activado, nos llevará el flujo de la ejecución a las actividades manuales de creación de las tablas, en caso contrario, se evitarán las mismas, pasando por la actividad *"Process Integrated Schema"*.

Process Integrated Schema

Si ingresamos un *json* en el campo destinado y no activamos el checkbox de creación manual, el Sistema ejecuta automáticamente la actividad *"Process Expected Schema"* (Ver 2 en Figura 27), que *parsea* el *json* creando en el Sistema el Integrated Schema junto con cada una de las tablas definidas y sus atributos, por lo que una vez finalizada esta tarea, la entidad ya está a disposición para ser utilizada.

Create Table and Atributes for Integrated Schema

En caso de que en la actividad *"Define Integrated Schema"*, hayamos seleccionado la forma manual de crear el mismo, el flujo del proceso nos lleva a la actividad manual *"Create Table and Atributes for Integrated Schema"* (Ver 3 en Figura 28), en la cual podemos crear una tabla y asociarle los atributos que deseemos. Por lo que el formulario nos ofrecerá indicar un nombre para la tabla que estamos creando y una lista de atributos (a los cuales debemos indicar un tipo de datos asociado y además si son clave primaria en su tabla). Una vez finalizada la definición de la tabla, es posible que queramos agregar más atributos a la misma, para

ello tenemos un checkbox, que activado nos dirige a una actividad que explicaremos a continuación para crear más atributos a dicha tabla.

Add more Atributes for Integrated Schema

La actividad *"Add More Atributes for Integrated Schema"*, nos permite como ya dijimos, para la tabla que estamos creando en el Integrated Schema, seguir creando atributos a la misma, de la misma manera que lo hicimos en la actividad anterior. Nuevamente, tenemos la posibilidad de seguir agregando atributos volviendo a ejecutar la actividad *"Add More Atributes for Integrated Schema"* (Ver 4 en Figura 28).

Ya sea en esta actividad o en la anterior, contamos con un checkbox en donde el usuario puede indicar al Sistema que necesita crear más tablas del Integrated Schema, y es a continuación de esta actividad en donde el flujo define un XOR que tomará dicha información para evaluar si debe volver a ejecutar las tareas manuales de creación de tablas y atributos del Integrated Schema o si ya finalizamos la creación de las mismas para pasar a las etapas de mapping.

Show Integrated Schema

Una vez que creamos el Integrated Schema con todas sus tablas y sus atributos, pasamos a una actividad que si bien es manual, es meramente informativa y nos permite ver como ha quedado el Integrated Schema, dicha actividad se llama *"Show Integrated Schema"* (Ver 5 en Figura 28). A futuro planteamos como interesante una extensión de esto, en donde el usuario puede indicar que no ha quedado conforme con el Integrated Schema y poder volver para atrás ya sea para crearlo de nuevo o para hacer algunas modificaciones simplemente, pero de momento nos sirve para dar un resumen de cómo han quedado las tablas y su estructura.

Define Foreign Keys Integrated Schema

Es posible que en la definición del Integrated Schema hayamos creado tablas que se referencian una a la otra mediante atributos de *foreign keys*, que indique una relación entre ellas si es que existe. Es por eso que en este momento el Sistema nos brinda la posibilidad de crear relaciones del estilo entre tablas con la actividad *"Define Foreign Keys Integrated Schema"* (Ver 6 en Figura 28), en la cual podemos seleccionar repetitivamente, un campo de una tabla que referencia foráneamente a otro campo de otra tabla. Para esto el formulario correspondiente nos muestra una tabla al azar en la cual podemos querer agregar *foreign keys* o no, y tenemos la posibilidad de ingresar mediante combos, parejas de atributos donde el primer atributo es cualquiera de los pertenecientes a la primera tabla y el segundo combo es el conjunto de atributos *primary key* del resto de las tablas.

Mapping Integrated Schema and Expected Schema

Como mencionamos al comienzo de esta sección, el Integrated Schema tiene el objetivo de tomar distintos (o en casos excepcionales uno solo) Expected Schemas y poder integrarlos, para generar más información que podrá ser útil más adelante cuando queramos construir un DW Schema. Podemos ver el Expected Schema como un template de extracción de datos, un esquema que nos permite obtener de la Web los datos referentes a algo específico como puede ser hoteles. Pero podríamos querer obtener una entidad de mayor nivel, que la misma contenga a hoteles como por ejemplo albergues donde poder hospedarse. Entonces podríamos tener distintos Expected Schema que nos permitan extraer datos de distintas entidades

y luego juntarlas (o mejor dicho integrarlas) en una sola, que en nuestro Sistema se llamará tabla del Integrated Schema.

Es aquí que debemos configurar la manera de integrar esos datos obtenidos de los Expected Schemas a criterio del usuario, definiendo un mapping para cada atributo de cada tabla del Integrated Schema con atributo de un Expected Schema. Para ello contamos con la actividad *"Mapping Integrated Schema and Expected Schema"* (Ver 7 en Figura 28) en la cual el Sistema nos ofrecerá una tabla del Integrated Schema y un atributo específico, y el usuario deberá asociar al mismo algún campo de algún Expected Schema. El Sistema nos pedirá completar esta actividad hasta que no haya más atributos de tablas del esquema integrado que aún no tengan mappings.

Define Joins Integrated Schema

Con el objetivo de cargar tablas del Integrated Schema con datos provistos por los distintos Expected Schema, debemos indicar de qué manera queremos vincular estos últimos para cargar las tablas mencionadas. Para hacer esto debemos completar la siguiente actividad manual de nombre *"Define Joins Integrated Schema"*(Ver 6 en Figura 26) . En el formulario se nos presenta una tabla del Integrated Schema elegida al azar por el Sistema, y alguno de los Expected Schema que participaron del mapping de dicho esquema integrado, por lo que debemos indicar los atributos de dicho Expected Schema que se deben *joinear* con otros atributos de otros Expected Schemas para conformar el mapping de la tabla del Integrated Schema.

Nuevamente tenemos la restricción de presentación del Activiti Explorer, que no nos permite a la vez, en el mismo formulario ingresar todos los *joins* que queramos. Pero el formulario cuenta con un checkbox el cual nos permite volver a ejecutar el formulario para el mismo Integrated Schema y Expected Schema de manera de seguir agregando joins. Por el contrario, si prendemos el checkbox de Done, significa que no queremos agregar más joins entre Expected Schemas para la tabla en cuestión del Integrated Schema. Sin embargo, si no activamos ningún checkbox, el Sistema nos seguirá ofreciendo agregar joins entre Expected Schemas que participaron del mapping de la tabla del Integrated Schema.

Finalizadas las actividades comentadas anteriormente, ya podemos pasar a lo que es la definición del DW Schema, esto es, la creación de todas sus tablas de hechos y tablas de dimensiones, también debemos realizar el mapping con el Integrated Schema como comentaremos oportunamente, entre otras cosas.

Interacción con Base de Metadatos:

Define Integrated Schema

Esta actividad no hace consultas ni modificaciones a la base de metadatos, ya que simplemente toma decisiones de flujo.

Process Integrated Schema

Para el caso en que el usuario haya ingresado un *json*, el mismo se *parseará* de manera de obtener las tablas del Integrated Schema y los atributos incluidos en las mismas. Dicha información será persistida en las tablas **integrated_esquema** y **atributos_integrated_esquema**. Para la primera cargamos los campos **config** y **nombre**, que son una referencia a la Configuración y el nombre del Integrated Schema respectivamente. Para la segunda, insertamos un registro por cada atributo del Integrated Schema y los campos son **config**,

nombre, **integrated_esquema, is_pk** y **tipo**, que son: una referencia a la Configuración, el nombre del atributo, una referencia al Integrated Schema, si es clave primaria y el tipo del atributo ingresado respectivamente.

Create Table and Atributes for Integrated Schema

La interacción entre esta actividad con la base de metadatos, es exactamente igual a la mencionada con la actividad *"Process Integrated Schema"* en el caso en que hayan ingresado un *json*.

Add More Atributes for Integrated Schema

Cuando se carga el formulario *" Add More Atributes for Integrated Schema*" no se realizan consultas a la base de metadatos.

Cuando se completa el formulario antes mencionado, se deben crear en la base de metadatos los atributos ingresados en el mismo, como el Integrated Schema ya existe sólo necesitamos insertar los atributos referenciando a este último. Para ello, insertamos un registro en la tabla **atributos_integrated_esquema** para cada atributo ingresado indicando en los campos **config**, **nombre**, **integrated_esquema, is_pk** y **tipo**, que refieren a la Configuración, nombre del atributo, una referencia a la tabla **integrated_esquema**, si es clave primaria y el tipo de datos del atributo respectivamente.

Show Integrated Schema

Cuando se carga el formulario *"Show Integrated Schema*" se consulta toda la información cargada recientemente relacionada al Integrated Schema. Para esto, se consultan las tablas de **integrated_esquema** y **atributos_integrated_esquema** para mostrar el resumen. Para filtrar la información, utilizamos el campo **config** de la tabla **integrated_esquema** que hace referencia a nuestra Configuración.

Define Foreign Keys Integrated Schema

Cuando se carga el formulario *"Define Foreign Keys Integrated Schema"* se muestra en el campo Table Name, el nombre de la tabla que vamos a definir sus claves foráneas para sus atributos. Para obtener dicha tabla, consultamos la tabla en la base de metadatos **integrated_esquema**, filtrando por el campo **config** que es nuestra Configuración y el campo **ya_linkeada** en false, de todos esos registros que resultan, nos quedamos con el primero. Además, vamos a ofrecer dos tipos de combos, uno que lleva una lista de todos los atributos de la tabla del Integrated Schema que seleccionamos anteriormente para procesar y el otro, es el conjunto de atributos del resto de las tablas del Integrated Schema que sean claves primarias de las mismas. Para esto hacemos un *select* en la tabla **atributos_integrated_esquema** filtrando por **is_pk** en *true*.

Cuando se completa el formulario antes mencionado, insertamos en la base de metadatos cada una de las *foreign keys* definidas. Para esto insertamos en la tabla **integrated_foreignkeys**, que tiene los campos **config**, **integrated_esquema1**, **integrated_esquema2**, **integrated_atributo1** e **integrated_atributo2**, que llevarán referencias a: la Configuración, el Integrated Schema del atributo que es clave foránea, el Integrated Schema del atributo clave referenciado, el atributo que es clave foránea y el atributo clave referenciado, respectivamente. Además, marcamos la tabla del Integrated Schema como ya procesada en el campo **ya_linkeada** en *true* de la tabla **integrated_esquema**.

Mapping Integrated Schema and Expected Schema

Cuando se carga el formulario *"Mapping Integrated Schema and Expected Schema"*, se muestran como valores fijos tanto el nombre de la tabla como el nombre de los atributos que deben ser mapeados. Para obtener qué tabla debemos mapear y cuáles son sus atributos realizamos dos consultas que pasamos a detallar. En la tabla **integrated_esquema**, obtenemos aleatoriamente cualquier registro cuyo campo **ya_mapeado** este en false, y luego, con la clave de ese registro que es el nombre del Integrated Schema vamos a la tabla **atributos_integrated_esquema** y filtramos utilizando dicho valor en el campo **integrated_esquema** y además indicando también el campo **ya_mapeado** en *false*. De esta manera obtenemos alguna tabla que aún no hayamos mapeado y todos los atributos de esa tabla que tampoco hayan sido mapeados. Por otro lado, todos éstos atributos del Integrated Schema que debemos mapear, lo haremos con atributos pertenecientes a algún Expected Schema de nuestra Configuración, para ofrecerle todos los disponibles en los combos que se pueden ver en la actividad, consultamos las tablas **expected_esquema** y **atributo_expected_esquema** utilizando como filtro el campo **config**.

Cuando se completa el formulario antes mencionado, debemos registrar cada mapping realizado en el formulario y para ello insertamos por cada uno, un registro en la tabla **mapping_integrated** y los valores que debemos indicar son **config**, **integrated_esquema**, **atributo_integrated_esquema**, **expected_esquema**, **atributo_expected_schema** y **web_source** que son: referencia a la Configuración, al Integrated Schema, al atributo del Integrated Schema, al Expected Schema, al atributo del Expected Schema y a la Web Source, respectivamente. Además, marcamos la tabla del Integrated Schema como ya mapeada marcando en *true* el campo **ya_mapeado** de la tabla **integrated_esquema**.

Define Joins Integrated Schema

Cuando se carga el formulario *"Define Joins Integrated Schema"*, se muestran el Integrated Schema que debemos procesar, el Expected Schema (es uno de todos aquellos Expected Schemas que estén mapeados con el Integrated Schema), los atributos de dicho Expected Schema, y todos los atributos del resto de los Expected Schemas (que también están mapeados con el Integrated Schema).

Para lo primero, consultamos la tabla **integrated_esquema** y obtenemos uno de todos los registros cuyos campos **config** y **ya_joineado** tengan los valores: id de Configuración y *false* respectivamente. De esta manera obtenemos aleatoriamente alguno de los Integrated Schemas que tenemos para procesar.

Para lo segundo, consultamos en la tabla **mapping_integrated** filtrando por los campos **config** (que deberá llevar el identificador de Configuración) e **integrated_esquema** (que deberá llevar la referencia al Integrated Schema) y con esto seleccionamos el campo **expected_esquema** para obtener todos los Expected Schema que posiblemente deberían *joinearse* para conformar el Integrated Schema. El formulario iterará sobre todos estos valores obtenidos (para cada Expected Schema, para cada Integrated Schema).

Para lo tercero, una vez que definimos sobre qué Expected Schema vamos a definir los joins, con el nombre del mismo vamos a la tabla **atributos_expected_esquema** y obtenemos todos los atributos de ese Expected Schema.

Para lo cuarto y último, debemos ofrecer *joinear* el Expected Schema que estamos procesando (para el Integrated Schema en cuestión), con cualquier atributo del resto de los Expected Schemas que están vinculados con el Integrated Schema mediante mapping, y para ello consultamos dos tablas, una es la **mapping_integrated** y obtenemos todos los valores que están en el campo **expected_esquema** que

referencian a algún Expected Schema; este resultado lo *joineamos* con la tabla de atributos de Expected Schemas llamada **atributos_expected_esquemas** y así obtenemos todos los Expected Schemas que debemos ofrecer y el conjunto de atributos de cada uno de ellos.

Resumiendo, de esa manera conformamos la presentación del formulario, seleccionado un Integrated Schema, a partir de éste, un Expected Schema y sus atributos, para determinar *joinearlos* con el resto de los Expected Schemas (y sus atributos) que estén vinculados por mapping con el Integrated Schema.

Cuando se completa el formulario antes mencionado, se debe marcar el Integrated Schema como ya procesado y lo haremos en el campo **ya_joineado** de la tabla **integrated_esquema**. Además, debemos insertar cada join definido entre atributos de Expected Schemas y lo haremos insertando en la tabla **joins_integrated** cargando en los campos **config**, **integrated_esquema**, **expected_esquema1**, **atributo_expected_esquema1**, **expected_esquema2** y **atributo_expected_esquema2** los valores siguientes: las claves de la Configuración, Integrated Schema, Expected Schema que procesamos, el atributo del mismo, el Expected Schema elegido para join, y el atributo del mismo también seleccionado.

5.2.4 Definición y Mapping del DW Schema.

Hasta aquí hemos definido a lo largo del proceso de Configuración, los Expected Schemas, las tablas y atributos del Integrated Schema y el mapping entre los primeros con las tablas de este último. Además, definimos los joins que debemos realizar entre los Expected Schemas para cargar las tablas del Integrated Schema. Lo que vamos a definir ahora es como configurar los metadatos necesarios para crear el DW Schema, esto es, crear en el Sistema el conjunto de tablas que lo componen, si las mismas son tablas de hechos o tablas de dimensiones, también vamos a indicar como se mapean los atributos de estas tablas con los atributos del Integrated Schema, todo esto, entre otras cosas. La Figura 29 nos ilustra el subproceso *"Define DW Schema and Mapping* (ubicado dentro del proceso principal (Ver 7 en Figura 26)) y sobre él haremos referencias a lo largo de esta sección, explicando detalladamente cada una de las actividades.

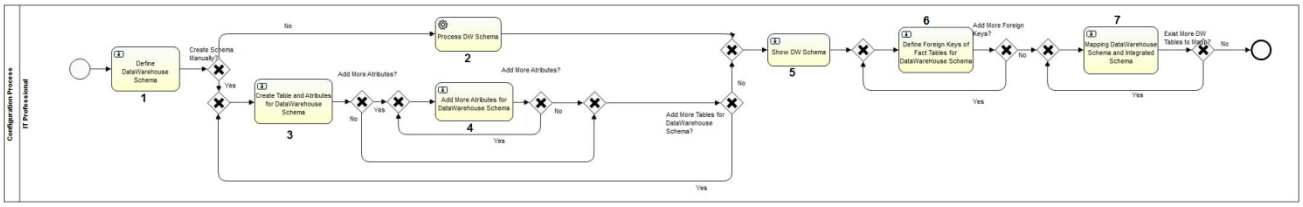


Figura 29 Subproceso de Define DW Schema and Mapping

Define Data Warehouse Schema

Al igual que como lo hicimos con las definiciones de Expected Schema e Integrated Schema, para el caso del DW Schema vamos a indicar de la misma manera, los distintos caminos que el Sistema nos ofrece para crear la configuración del DW. Nuevamente, podremos hacerlo mediante un *json* y procesándolo automáticamente o definiendo el conjunto de tablas y de atributos de manera manual y para ello contamos con la actividad manual *"Define DataWarehouse Schema"* que contiene un campo en donde ingresar un *json*, otro campo que podemos ingresar el nombre del DW Schema, ó un checkbox que indica si deseamos crear las estructuras manualmente como ya hemos visto. Así que el flujo tomará un camino u otro según qué elijamos, si ingresamos un *json* se ejecutará la actividad de arriba *"Process DW Schema"* o de lo contrario ingresamos el nombre del DW Schema y activamos el checkbox, entonces nos vamos a las actividades manuales que se encuentra por debajo en el flujo.

Process DW Schema

En caso de haber ingresado un *json* en el campo del formulario destinado para ello y no activamos el checkbox de definición manual, el Sistema ejecuta automáticamente la actividad *"Process DW Schema"* (Ver 2 en Figura 29), que *parsea* el *json* en el Sistema la entidad DW Schema con su respectivo nombre, además todas las tablas y atributos que están incluidos. Por lo que una vez que finaliza dicha ejecución, ya podremos seguir trabajando en definiciones para el DW Schema.

Create Table and Atributes for DW Schema

Como ya mencionamos en la actividad manual anterior, si el usuario eligió la forma manual para crear el DW Schema, el proceso llevará el flujo a la actividad manual *"Create Table and Atributes for DW Schema"* (Ver 3 en Figura 29), en la cual podemos crear una tabla y asociarle los atributos que deseemos. Es por esto, que el formulario nos ofrecerá ingresar un nombre para la tabla que estamos creando en el esquema y la lista asociada de atributos con su tipo y si los mismos son clave primaria en ella. Una vez finalizada la definición de la tabla, es posible que queramos agregar más atributos a la misma, para ello tenemos un checkbox, que activado nos dirige a una actividad manual que explicaremos a continuación para crear más atributos a dicha tabla.

Add More Atributes for DW Schema

De la misma manera que hicimos en la definición del Integrated Schema, al momento de crear una tabla podemos seguir agregando atributos a la misma con una nueva actividad manual *"Add More Atributes for DW Schema"*. (Ver 4 en Figura 29). Nuevamente activando el checkbox podremos realizar esta actividad repetitivamente hasta lograr los atributos deseados.

Tanto en la actividad *"Create Table and Atributes for DW Schema"* y en *"Add More Atributes for DW Schema"*, podemos indicar al momento de finalizar la creación de la tabla, que se deseamos volver a definir una nueva tabla en el esquema. Para ello tenemos un checkbox cuyo funcionamiento es idéntico que en la definición del Integrated Schema. El XOR a continuación es quién tomará la decisión de hacia donde dirige el flujo y evaluar si se debe volver a ejecutar las tareas manuales de creación de tablas y atributos del DW Schema.

Show DW Schema

A continuación, la actividad manual *"Show DW Schema"* (Ver 5 en Figura 29) no requerirá de un trabajo manual del usuario, solo deberá aceptar la misma dado que el objetivo es mostrar al usuario como ha quedado conformado el DW Schema con todas sus tablas y atributos.

Análogamente a como lo hicimos en el Integrated Schema, sería posible extender esta funcionalidad y poder volver a realizar modificaciones si el usuario no ha quedado conforme con el DW Schema.

Define Foreign Keys of Fact Tables for DataWareHouse Schema

De manera similar a cuando lo hicimos en el Integrated Schema, tenemos la posibilidad de vincular las tablas del DW Schema referenciando unas con otras mediante claves foráneas. Para ello contamos con la actividad *"Define Foreign Keys of Fact Tables for DW Schema"* (Ver 6 en Figura 29) en la cual podemos seleccionar repetitivamente, un campo de una tabla de hechos que referencia foráneamente a otro campo de alguna tabla de dimensión. El formulario correspondiente nos muestra una tabla al azar, de todas aquellas que sean

la tabla de hechos, en la cual podemos querer agregar *foreign keys* o no, y tenemos la posibilidad de ingresar mediante combos, parejas de atributos donde el primer atributo es cualquiera de los pertenecientes a la primera tabla y el segundo combo es el conjunto de atributos *primary key* del resto de las tablas de dimensiones.

Mapping DW Schema and Integrated Schema

Debemos definir de qué manera queremos cargar las tablas del DW Schema utilizando las tablas del Integrated Schema. Para ello, es necesario definir un mapping entre las primeras y las segundas y para ello contamos con la actividad *"Mapping DW Schema and Integrated Schema"* (Ver 7 en Figura 29) de manera de definir un mapping para cada atributo de cada tabla del DW Schema con un atributo de alguna tabla del Integrated Schema. El Sistema nos ofrecerá una tabla del DW Schema que aún no procesamos y secuencialmente sus atributos para que (mediante combos) elegir qué atributos de que tablas del Integrated Schema mapean con éstos. El Sistema nos pedirá completar esta actividad hasta que no haya más atributos de tablas del DW Schema que aún no tengan mapping.

Define Joins DW Schema

Como mencionamos en el párrafo anterior, es necesario vincular las tablas del Integrated Schema para poder cargar las tablas del DW Schema y para ello realizamos un mapping de manera de saber qué campos cargamos. Lo que realizamos en la actividad *"Define Joins DW Schema"* (Ver 8 en Figura 29) es definir cómo joineamos las tablas del Integrated Schema para lograr dicha carga. En el formulario se nos presenta una tabla del DW Schema que aún no procesamos, además se carga una de las tablas que participaron del mapping para dicha tabla del DW Schema, por lo que debemos especificar los atributos de esa tabla del Integrated Schema que joinean con cuáles otros atributos de tablas (sólo las que se mapearon con la tabla del DW Schema) del Integrated Schema.

Nuevamente tenemos la restricción de presentación del Activiti Explorer, que no nos permite a la vez, en el mismo formulario ingresar todos los joins que queramos, pero el formulario cuenta con un checkbox el cual nos permite volver a ejecutar el formulario para las mismas tablas del DW Schema e Integrated Schema de manera de seguir agregando joins. Por el contrario, si activamos el checkbox de Done, significa que no queremos agregar más joins entre tablas del Integrated Schema para la tabla en cuestión del DW Schema. Sin embargo, si no activamos ningún checkbox, el Sistema nos seguirá ofreciendo agregar joins siempre y cuando aún existan tablas del Integrated Schema que no aún no procesamos.

Aquí finaliza el proceso de Configuración para creación y carga del DW, podemos decir que ya contamos con los metadatos necesarios para ejecutar un proceso de Carga de manera de saber, seleccionando esta Configuración, la realidad de qué dominio deseamos analizar, de qué fuentes vamos a obtener todos los datos y dónde se alojarán los mismos. Contamos con la definición de Expected Schemas para extraer los datos con los que vamos a trabajar, cómo integrarlos en tablas de Integrated Schemas y cómo, vinculando éstas cargamos finalmente un DW Schema ya sea sus tablas de hechos y tablas de dimensiones.

Interacción con Base de Metadatos:

Define DataWarehouse Schema

Esta actividad no hace consultas ni modificaciones a la base de metadatos, ya que simplemente toma decisiones de flujo.

Process DW Schema

Para el caso en que el usuario haya ingresado un *json*, el mismo se *parseará* de manera de obtener las tablas del DW Schema y los atributos incluidos en las mismas. Dicha información será persistida en las tablas **dw_esquema** y **atributos_dw_esquema**. Para la primera cargamos los campos **config, dw_esquema, nombre y es_fact**, que son una referencia a la Configuración, el nombre del DW Schema y el nombre de la tabla, y si es tabla de hechos respectivamente. Para la segunda, insertamos un registro por cada atributo de cada tabla del DW Schema y los campos son **config**, **nombre**, **dw_esquema** y **tipo**, que son respectivamente: una referencia a la Configuración, el nombre del atributo, el nombre de la tabla del DW Schema y el tipo del atributo ingresado.

Create Table and Atributes for Data Warehouse Schema

La interacción entre esta actividad con la base de metadatos, es exactamente igual a la mencionada con la actividad *"Process DW Schema"* en el caso en que hayan ingresado un *json*.

Add More Atributes for DataWarehouse Schema

Cuando se carga el formulario " *Add More Atributes for DataWarehouse Schema*" no se realizan consultas a la base de metadatos.

Cuando se completa el formulario antes mencionado, se deben crear en la base de metadatos los atributos ingresados en el mismo, como la tabla del DW Schema ya existe sólo necesitamos insertar los atributos referenciando a este último. Para ello, insertamos un registro en la tabla **atributos_dw_esquema** para cada atributo ingresado indicando en los campos **config**, **nombre**, **dw_esquema, is_pk** y **tipo**, que refieren a la Configuración, nombre del atributo, nombre de la tabla del DW Schema, si es clave primaria y el tipo de datos del atributo respectivamente.

Show DW Schema

Cuando se carga el formulario *"Show DW Schema*" se consultan todos los datos cargados recientemente relacionados al DW Schema. Para esto, se consultan las tablas de **dw_esquema** y **atributos_dw_esquema** para mostrar el resumen. Para filtrar dichos registros, utilizamos el campo **config** de la tabla **dw_esquema** que hace referencia a nuestra Configuración.

Define Foreign Keys of Fact Tables for DataWarehouse Schema

Cuando se carga el formulario *"Foreign Keys of Fact Tables for DataWarehouse Schema"* se muestra en el campo Fact Table, el nombre de la tabla que vamos a definir sus claves foráneas para sus atributos. Para obtener dicha tabla, consultamos la tabla en la base de metadatos **dw_esquema**, filtrando por el campo **config** que es nuestra Configuración, el campo **ya_linkeada** en false y el campo **es_fact** en *true* (vamos a iterar solo en las tablas de hechos), de todos esos registros que resultan, nos quedamos con el primero. Además, vamos a ofrecer dos tipos de combos, uno que lleva una lista de todos los atributos de la tabla del DW Schema que seleccionamos anteriormente para procesar, y el otro es el conjunto de atributos del resto de las tablas (que no sean tablas de hechos) del DW Schema que sean claves primarias de las mismas. Para esto hacemos un *select* en la tabla **atributos_dw_esquema** filtrando por **is_pk** en *true*, además *joineamos* con la tabla **dw_esquema** para filtrar sólo por los registros de tablas de hechos en el campo **es_fact**.

Cuando se completa el formulario antes mencionado, insertamos en la base de metadatos cada una de las *foreign keys* definidas. Para esto insertamos en la tabla **link_dw**, que tiene los campos **config**, **tabla_dw**, **atributo_dw**, **tabla_fk** y **atributo_fk**, que llevarán referencias a la Configuración, el nombre de la tabla del DW Schema del atributo que es clave foránea, el atributo que es clave foránea, la tabla del DW Schema que contiene al atributo clave referenciado y finalmente el atributo referenciado, respectivamente. Además, marcamos la tabla del DW Schema como ya procesada en el campo **ya_linkeada** en *true* de la tabla **dw_esquema**.

Mapping Data Warehouse Schema and Integrated Schema

Cuando se carga el formulario *"Mapping Data Warehouse Schema and Integrated Schema"*, se muestran como valores fijos el nombre del DW Schema, el nombre de la tabla y el nombre de los atributos que deben ser mapeados. Para obtener qué tabla debemos mapear y cuáles son sus atributos realizamos dos consultas que pasamos a detallar. En la tabla **dw_esquema**, obtenemos aleatoriamente cualquier registro cuyo campo **ya_mapeado** este en false, y luego, con la clave de ese registro que es el nombre de la tabla del DW Schema vamos a la tabla **atributos_dw_esquema** y filtramos utilizando dicho valor en el campo **dw_esquema** indicando el nombre de la tabla del DW Schema (para obtener sólo los atributos que no hayan sido mapeados, vamos a excluir del resultado anterior los atributos que estén en la tabla **mapping_dw** que explicaremos en el párrafo siguiente). De esta manera obtenemos alguna tabla que aún no hayamos mapeado y todos los atributos de esa tabla que tampoco hayan sido mapeados. Por otro lado, todos éstos atributos de tablas del DW Schema que debemos mapear lo haremos con atributos pertenecientes a alguna tabla del Integrated Schema de nuestra Configuración, para ofrecerle todos los disponibles en los combos que se pueden ver en la actividad, consultamos las tablas **integrated_esquema** y **atributo_integrated_esquema** utilizando como filtro el campo **config**.

Cuando se completa el formulario antes mencionado, debemos registrar cada mapping realizado en el formulario y para ello insertamos por cada uno, un registro en la tabla **mapping_dw** y los valores que debemos indicar con **config**, **dw_esquema**, **atributo_dw_esquema**, **integrated_esquema** y **atributo_integrated_esquema** que son: referencia a la Configuración, a la tabla del DW Schema, al atributo de la tabla del DW Schema, a la tabla del Integrated Schema y al atributo de la tabla del Integrated Schema, respectivamente. Además, marcamos la tabla del DW Schema como ya mapeada marcando en *true* el campo **ya_mapeado** de la tabla **dw_esquema** filtrando por el campo **nombre** para obtener la tabla correspondiente.

Define Joins DW Schema

Cuando se carga el formulario *"Define Joins DW Schema"*, se muestran la tabla del DW Schema que debemos procesar, el nombre de una tabla del Integrated Schema (es una de todas aquellas tablas del Integrated Schema que están mapeadas con la tabla del DW Schema en la instancia de mapping), los atributos de dicha tabla del Integrated Schema en el formulario léase 'Integrated Atribute', y todos los atributos del resto de las tablas del Integrated Schema que también están mapeadas con la tabla de DW Schema antes mencionada, léase 'Integrated Schema and Atribute'.

Para lo primero, consultamos la tabla **dw_esquema** y obtenemos uno de todos los registros cuyos campos **config** y **ya_joineada** tengan los valores: id de Configuración y false respectivamente. De esta manera obtenemos aleatoriamente alguna de las tablas del DW Schema que aún no fueron definidos los joins que la componen.

Para lo segundo, consultamos en la tabla **mapping_dw** filtrando por los campos **config** (que deberá llevar el identificador de Configuración) y **dw_esquema** (que llevará la referencia a la tabla del DW Schema que estamos procesando) y con esto seleccionamos el campo **integrated_esquema** para obtener todas las tablas del Integrated Schema que posiblemente deberían *joinearse* para conformar la tabla del DW Schema. El formulario iterará sobre estos valores obtenidos (para cada tabla del DW Schema, para cada tabla del Integrated Schema involucrada con la primera por mapping).

Para lo tercero, una vez que definimos sobre qué tabla del Integrated Schema vamos a definir los joins, con el nombre de la misma vamos a la tabla **atributos_integrated_esquema** y obtenemos todos los atributos de esa tabla.

Para lo cuarto y último, debemos ofrecer *joinear* la tabla del Integrated Schema que estamos procesando (para la tabla del DW Schema en cuestión), con cualquier atributo del resto de las tablas del Integrated Schema que están vinculados con la tabla del DW Schema mediante mapping, y para ello consultamos dos tablas, una es la **mapping_dw** y obtenemos todos los valores que están en el campo **integrated_esquema**, que referencia a alguna tabla del Integrated Schema; este resultado lo joineamos con la tabla de atributos del Integrated Schema llamada **atributos_integrated_esquema** y así obtenemos todas las tablas del Integrated Schema que debemos ofrecer y el conjunto de atributos de cada una de ellas.

Resumiendo, de esa manera conformamos la presentación del formulario, seleccionando una tabla del DW Schema, a partir de ésta, una tabla del Integrated Schema y sus atributos, para determinar *joinearlos* con el resto de las tablas del Integrated Schema (y sus atributos) que estén vinculados por mapping con la tabla del Integrated Schema.

Cuando se completa el formulario antes mencionado, se debe marcar la tabla del DW Schema como ya procesada y lo haremos con el campo **ya_joineada** de la tabla **dw_esquema**. Además, debemos insertar cada join definido entre los atributos de las tablas del Integrated Schema y lo haremos insertando en la tabla **joins_dw** cargando en los campos **config**, **dw_esquema**, **integrated_esquema1**, **atributo_integrated_esquema1**, **integrated_esquema2**, **atributo_integrated_esquema2** los valores siguientes: las claves de la Configuración, la tabla del DW Schema y las tablas y atributos del Integrated Schema involucradas en el join.

5.3 Implementación Proceso de Carga.

A continuación presentaremos la implementación del Proceso de Carga, las actividades que la componen son mayoritariamente automáticas aunque en alguna oportunidad se pedirá la interacción del usuario para resolver alguna situación en particular. Como podemos ver en la Figura 30 se ilustra el proceso completo con sus actividades y no cuenta con subprocesos.

El proceso está divido en distintas partes según los procesamientos que debamos hacer. Primero debemos seleccionar la Configuración para la cual queremos procesar la Carga y además crear todas las estructuras que se indican en los metadatos generados en el proceso de Configuración.

Segundo, una vez que elegimos la Configuración y creamos todas las estructuras necesarias debemos consumir de alguna manera los servicios provistos para extraer los datos y persistirlos. Esto refiere a la carga de las tablas del Source Schema, que contiene todos los datos seleccionados de la web sin procesamiento alguno, pero aquí ya es donde nos independizamos de las fuentes Web porque ya nos trajimos todos los datos que necesitamos procesar.

Tercero, debemos cargar las tablas del Expected Schema y lo haremos en función de los metadatos indicados en el proceso de Configuración y utilizando como fuente las tablas que cargamos en el Source Schema. En esta etapa es posible encontrarnos con problemas de consistencia de datos o la no existencia de los mismos, dejamos una actividad para implementar a futuro que podría eventualmente levantar los datos inconsistentes a resolver e iterar sobre ellos hasta que ya no queden dichos problemas.

Cuarto, una vez que tenemos todas las tablas del Expected Schema cargadas, podemos proceder a cargar las tablas del Integrated Schema, nuevamente debemos utilizar la Configuración determinada para saber cómo se vinculan las mismas y qué datos necesitamos y de dónde los obtenemos. También aquí queda contemplada una oportunidad de resolución de conflictos de integración, que como comentamos anteriormente queda abierto como mejora a futuro.

Quinto, ya con las tablas del Integrated Schema cargadas completamente podemos proceder con la carga de las tablas del DW Schema que es nuestro objetivo final. Debemos vincular las tablas del Integrated Schema nuevamente consultando los metadatos del proceso de Configuración.

Por último existe una instancia de interacción con el usuario en donde se muestra iterativamente de qué manera se cargarán las tablas de hechos del DW, pudiendo el usuario optar por realizar modificaciones o confirmar las *querys* de carga.

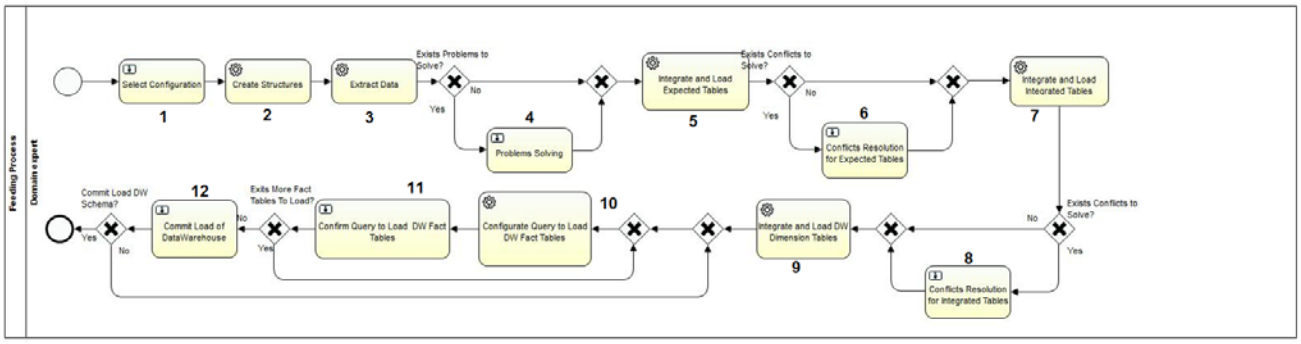


Figura 30 Proceso de Carga

Select Configuration

En la Figura 30 podemos ver el modelo correspondiente al proceso de Carga en donde en primera instancia debemos realizar la actividad manual correspondiente a la selección de qué configuración vamos a utilizar, dicha actividad se llama "*Select Configuration*" (Ver 1 en Figura 30). Aquí se ofrecerán todas las Configuraciones disponibles en el Sistema, cada una se corresponde con una ejecución del proceso de Configuración realizado anteriormente. La elección de esta configuración nos permitirá en el proceso de Carga contar con los metadatos de configuración para saber qué tablas cargar y cómo hacerlo.

El Sistema nos ofrecerá un combo con una lista de identificadores de Configuraciones y al elegir una de ellas el Sistema guardará la misma a lo largo de todo el proceso de Carga, dado que la necesitaremos en todo momento para consultar la base de metadatos.

Create Structures

Luego de seleccionada la Configuración, automáticamente se invoca a la actividad *"Create Structures"* (Ver 2 en Figura 30), que como mencionamos en la sección anterior, se encarga de crear todas las estructuras definidas en el proceso de configuración.

El Sistema crea una base de datos exclusiva para ésta, aunque es posible que la misma exista (esto es si ya realizamos una carga para dicha configuración, en este caso se vacían todas las tablas existentes sin necesidad de crear las estructuras nuevamente). El nombre para la base estará dado por "base_i" donde i corresponde al identificador numérico de la ejecución de Configuración. Esta actividad tendrá 4 etapas que son la creación de estructuras correspondientes a los Source Schemas, Expected Schemas, Integrated Schemas y finalmente al DW Schema.

Con respecto a los Source Schemas, consultamos de la tabla correspondiente a los metadatos de los mismos, los nombres y la información del Expected Schema asociado a cada uno.

Luego creamos una tabla en la base de datos de esta Configuración para cada Source Schema, cuyo nombre estará dado por "nombre_web_source"+"nombre_expected_schema" y luego el conjunto de atributos serán los del Expected Schema. Finalmente armamos una sentencia *create* de SQL utilizando los *strings* conformados.

Con respecto al resto, extraemos de las tablas de metadatos respectivas, los nombres de las tablas y atributos correspondientes a Expected Schema, Integrated Schema y DW Schema, y la creación de las estructuras es análoga a los Source Schema.

Extract Data

Finalizada la tarea anterior, pasamos a ejecutar también automáticamente la actividad *"Extract Data"* (Ver 3 en Figura 30). La misma se encargará de realizar la extracción de los datos desde la Web para cada Web Source seleccionada en la Configuración. La extracción consiste en, para cada Web Source invocar a los Data Services asociados que nos transferirán una a una cada tupla publicada por el servicio. Los servicios deben ser de tipo Rest ya que contamos en nuestro Sistema con un Cliente Rest común a cualquier tipo de Web Source. Por cada Data Service invocado, obtenemos el conjunto de datos publicados y el Sistema procederá a persistirlo en la base de datos "base_i" en la tabla correspondiente a dicho Web Source.

En la extracción de los datos que comentamos, se pueden presentar inconsistencias o problemas como por ejemplo lectura de caracteres especiales no soportados, o tamaño de campos que exceden el tipo de dato definido para dicho atributo (además, si el Sistema es extendido en cuanto a la definición de métricas de calidad en el proceso de Configuración, es aquí en donde el proceso de Carga evaluará los datos y eventualmente detectará problemas de calidad), etc. Para esto, el Sistema cuenta con una actividad manual no implementada de nombre *"Problems Solving"* (Ver 4 en Figura 30), que queda previsto sea la encargada de pedir al usuario resolver de manera manual dichas inconsistencias.

Integrated and Load Expected Tables

A continuación vamos a cargar cada tabla del Expected Schema con la actividad automática *"Integrated and Load Expected Tables"* (Ver 5 en Figura 30). Para ello utilizando los metadatos generados en el proceso de Configuración se generan los *inserts* en las mismas de la siguiente forma: realizamos un *select* a cada tabla

de Source Schema, cuyo Expected Schema sea el que estamos queriendo cargar y unimos los datos mediante *union*. Recordar que la estructura de cada tabla de los Source Schema es la misma que la estructura del Expected Schema en cuestión.

De igual forma que en la parte anterior, podríamos tener problemas al ejecutar la carga de los Expected Schemas. Para ello agregamos una actividad manual que podría resolverlos de nombre *"Conflict Resolution for Expected Tables"* (Ver 6 en Figura 30). Un ejemplo de resolución podría ser encontrar datos repetidos en diferentes Web Source, y el Sistema podría optar por la fuente más confiable. Esta parte también deja como trabajo a futuro pero dejamos el flujo extensible para ella.

Integrate and Load Integrated Tables

Ahora pasamos a cargar las tablas del Integrated Schema, ejecutando la actividad automática de nombre "*Integrated and Load Integrated Tables*" (Ver 7 en Figura 30). Para dicha carga realizamos joins entre las tablas del Expected Schema según los metadatos definidos de joins y de mapping para asociar qué campo de cada Expected Schema carga qué campo de la tabla del Integrated Schema. Cómo decíamos, en el proceso de Configuración existe una instancia en donde se define de qué manera se deben vincular los Expected Schemas para componer tablas del Integrated Schema, ya sea en la definición de mapping o en la composición de los joins, información relevante para nosotros en esta etapa que debemos consultar en la base de metadatos.

De manera análoga a las etapas anteriores, dejamos prevista una actividad manual de resolución de problemas *"Conflicts Resolution for Integrated Tables"* (Ver 8 en Figura 30). Los problemas que podrían presentarse por ejemplo es que los datos por los cuales joineamos, signifiquen lo mismo pero tengan una codificación diferente, por ejemplo, en una tabla tengamos el código de departamento MVD para representar Montevideo y otro campo en otra tabla que indique lo mismo pero con código MDEO.

Integrate and Load DW Dimension Tables

Una vez que tenemos cargadas todas las tablas del Integrated Schema, podemos comenzar a cargar el DW en cuestión, cargando las tablas de dimensiones y las tablas de hechos definidas en el proceso de Configuración y creadas en el proceso de Carga, en la actividad de creación de estructuras.

En primera instancia con la actividad automática *"Integrated and Load DW Dimension Tables"* (Ver 9 en Figura 30) vamos a cargar las tablas de dimensiones. Esto será utilizando los metadatos de joins y mapping entre las tablas del Integrated Schema, análogamente a cuando cargamos las tablas del Integrated Schema en la actividad automática anterior.

Configurate Query To Load DW Fact Tables

En cuanto a la carga de las tablas de hechos, existe una actividad automática *"Configurate Query To Load DW Fact Tables"* (Ver 10 en Figura 30), que así como la tarea anterior generar los *insert* para las tablas de dimensiones, en este caso será para las tablas de hechos. La diferencia es que este *insert* no se ejecuta en esta actividad si no que se muestra en la siguiente actividad manual *"Confirm Query To Load DW Fact Tables"* (Ver 11 en Figura 30), en donde el usuario podrá confirmar la inserción de los datos de esa forma o modificar la query. Luego de confirmar, el Sistema procede a ejecutar la sentencia *insert*.

Commit Load of Data Warehouse

Finalmente, el usuario podría mediante consultas de SQL visualizar como ha quedado su DW de manera de confirmar el mismo o volver a modificar las querys para cargar las tablas de hechos. Para notificar esto mismo al Sistema, finalizamos el proceso con una actividad manual *"Commit Load of Data Warehouse"* (Ver 12 en Figura 30).

Interacción con Base de Metadatos

Select Configuration

Cuando cargamos la actividad *"Select Configuration"* debemos consultar en la base de metadatos, todas las Configuraciones disponibles en el Sistema para ser utilizadas, como se mencionó anteriormente la Configuración que seleccionemos determinará el conjunto de metadatos para el proceso de Carga. La tabla de la base de metadatos que nos las provee es **configuracion** y nos quedaremos con el campo **id**. Cada vez que se ejecuta el proceso de Configuración se crea un registro en esta tabla, y queda disponible para utilizarse en una cantidad ilimitada de procesos de Carga que explicaremos a continuación. En este formulario ofrecemos al usuario seleccionar la referencia a la Configuración que dará soporte a nuestra Carga.

Cuando el usuario completa la actividad antes mencionada, no se alteran las bases de datos de nuestro Sistema pero si se crea una variable en el flujo de Activiti de nombre *config* a la cual accederemos de manera corriente en futuras actividades.

Create Structures

Cuando se ejecuta esta actividad (que es automática en el proceso, no requiere interacción con el usuario por lo cual no es necesario acceder a datos para presentación) lo primero que debemos definir es si es necesario crear la base de datos y las estructuras en la misma. Este funcionamiento es lo que explicaremos a continuación.

En la tabla **configuracion** existe un campo **procesado**, que nos indica si las estructuras se deben crear, o vaciar. Ambos casos se detallarán en los siguientes párrafos.

En caso de que la Configuración no haya sido procesada, se deberán crear todas las estructuras además de la base de datos que alojará las mismas. El nombre de la nueva base de datos estará compuesto por la palabra 'base_' seguida del identificador de Configuración. La creación de las estructuras estará dada en cuatro etapas, que corresponden con las estructuras de Source Schemas, Expected Schema, Integrated Schema y DW Schema.

Source Schema

Para crear las estructuras asociadas al Source Schema, primero consultamos en la base de metadatos, en la tabla **config_web_sources** de manera de obtener todas las Web Source que están participando de esta Configuración filtrando por el campo **config** con el **id** de la misma. Una vez que tenemos todas las referencias a las Web Sources vamos a la tabla **web_source** y obtenemos el nombre de las mismas.

Dado que cada Web Source, tiene una referencia a un Expected Schema, consultaremos la tabla **config_web_source_expected_esquema** filtrando por **config** y **web_source** de manera de obtener el

nombre del Expected Schema asociado a la Web Source. Los nombres de las tablas que crearemos para el Source Schema estará dado por el nombre del Expected Schema antes mencionado concatenado con el nombre de la Web Source. Con respecto a qué atributos estarán incluidos en dichas tablas, vamos a la tabla **atributos_expected_esquema** filtrando por **expected_esquema**, por lo que los atributos del Expected Schema serán los propios de cada tabla del Source Schema. Luego de eso, procedemos con sentencia *create* para crear las tablas en la base de datos de nuestro proceso de Carga.

Expected Schema

Con respecto a la creación de estructuras del Expected Schema, vamos a crear una tabla por cada Expected Schema existente, y los atributos se corresponderán con cada atributo del mismo. Por lo que consultando la tabla **expected_esquema** de la base de metadatos obtenemos los nombres de todas las tablas que debemos crear y en la tabla **atributos_expected_esquema** obtenemos los nombres de los atributos de las tablas que se corresponden con cada atributo del Expected Schema, esto es similar a la creación de estructuras para el Source Schema. Con sentencia *create* creamos todas las tablas y sus atributos con su tipo de dato.

Integrated Schema

Con respecto a la creación de estructuras para dar soporte al Integrated Schema, primero vamos a buscar toda la información relevante al mismo a la base de metadatos, en la tabla **integrated_esquema** levantamos los registros cuya **config** sea el id de Configuración que estamos procesando. Hasta ahí, tenemos la colección de tablas que deben componer dicho esquema, y para saber cuáles atributos estarán incluidos en las mismas, hacemos lo propio en la tabla **atributos_integrated_esquema** filtrando por los campos **config** e **integrated_esquema** que deberán tener los valores: id de Configuración y el nombre de la tabla del Integrated Schema que estamos queriendo crear.

Ahora ya tenemos el nombre de cada tabla del Integrated Schema, y los nombres y tipos de cada atributo que le pertenecen, por lo que ya podemos proceder con la sentencia *create* para cada una de ellas de manera de crear las estructuras que dan soporte al Integrated Schema en la base de datos del proceso de Carga que estamos ejecutando.

DW Schema

Para crear el DW Schema, con todas sus tablas de hechos y de dimensiones, debemos consultar cuales son las mismas en la base de metadatos y para ello comenzamos con la tabla **dw_esquema**, donde cada registro se corresponde con una tabla del DW Schema. Necesitamos filtrar por el campo **config** para obtener todo lo referente a nuestra Configuración. Para obtener los atributos de estas tablas que mencionamos, ahora consultamos la tabla **atributos_dw_esquema** de la base de metadatos y allí obtenemos el nombre de cada uno, el tipo de dato y si es clave primaria. Por lo que ya tenemos los nombres y atributos de cada tabla del DW Schema y ya podemos crearlas en la base de datos del proceso de Carga.

Finalmente, cuando ya tenemos todas las estructuras creadas tanto para el Source Schema, el Expected Schema, el Integrated Schema y el DW Schema, debemos marcar la Configuración de alguna manera para indicar que las estructuras ya fueron creadas y no es necesario volverlo a hacer para una futura ejecución del proceso de Carga. Para esto mismo es que marcamos con 'S' el campo **procesado** en la tabla **configuration** para el registro cuya **config** sea el id de nuestra Configuración.

Si previamente ya se ejecutó un proceso de Carga con esta misma Configuración seleccionada, tanto la base de datos del proceso de Carga como las estructuras necesarias para crear el DW ya han sido creadas y este paso debemos saltearlo. Lo que si será necesario hacer, es borrar todos los registros que estén en esas tablas, ya que los datos de las fuentes Web posiblemente hayan cambiado y esto debe actualizarse. Para esto se borrarán con sentencias *delete* todas las estructuras en cuatro etapas, las de Source Schema, de Expected Schema, de Integrated Schema y de DW Schema.

Source Schema

Para obtener las tablas de Source Schema que debemos vaciar, procedemos de la misma manera que cuando quisimos obtener los nombres al momento de crearlas exceptuando la parte de los atributos. Para cada uno de esos nombres con sentencia *delete* borramos sus registros.

Expected Schema

Para borrar los registros generados en el Expected Schema, debemos primero obtener todas las tablas que creamos en el mismo. En la tabla **config_web_source_expected_esquema** si filtramos por el campo **config**, obtenemos todos los valores en la columna **expected_esquema**, cada valor es el nombre de una tabla del mismo y para cada uno debemos ejecutar una sentencia *delete*.

Integrated Schema

Similar al Expected Schema, hacemos lo propio con las tablas creadas en el Integrated Schema, pero las tablas las obtenemos de la tabla de la base de metadatos **integrated_esquema** como siempre filtrando por el campo **config**. Nuevamente recorriendo en ellas y ejecutando *delete* para cada una.

DW Schema

Buscamos en la tabla **dw_esquema** de la base de metadatos, para saber qué tablas debemos borrar y filtramos por el campo **config** para obtener las de nuestra Configuración. El campo **nombre** es el que necesitamos nosotros para ejecutar cada una de las sentencias *delete*.

Extract Data

La actividad *"Extract Data"* será la encargada de extraer los datos de las fuentes web y tendrá como meta cargar completamente las tablas del Source Schema, por lo que antes que nada debe consultar los metadatos asociados a las Web Sources, los servicios de extracción que nos proveen aplicaciones externas, y las tablas del Source Schema que se configuro en el proceso de Configuración, ya que son estas las tablas que guardarán la totalidad de los datos. Una vez que se ejecuta esta actividad ya no dependemos de la estabilidad o disponibilidad de las fuentes Web. Por lo que a continuación procederemos a explicar cómo levantamos los metadatos y luego, como persistimos los datos obtenidos.

Consultando la tabla **config_web_source_expected_esquema** de la base de metadatos, si filtramos por **config** como lo venimos haciendo hasta ahora y joineando con las tablas **config_web_source_dataservice** y **web_source**, seleccionamos los campos **expected_esquema** (nombre del Expected Schema asociado a la Web Source), **web_source** (nombre de la Web Source), **data_service** (servicio externo que nos encapsula la extracción de datos para la Web Source) y **url** (url de la fuente Web que provee los datos) y para cada registro de esa consulta generamos la entidad en nuestro Sistema de tipo DatosConfig que entre otras cosas tiene la información necesaria para poder extraer los datos: los atributos más importantes son el nombre del

Expected Schema que nos provee los datos que queremos extraer, el nombre de la Web Source que representa en nuestro Sistema la fuente Web de datos, la url donde están publicados los datos en la Web, una lista de mapeos para indicar con qué datos nos vamos a quedar y con cuáles no (esto se refiere al mapping entre el Source Schema y el Expected Schema, que indica de todas las columnas brindadas por la Web cuáles mapeamos y decidimos extraerlas) y finalmente el campo **data_service** que es el servicio al cual debemos invocar para obtener los datos.

Para consumir el servicio lo invocamos por cliente Rest al *endpoint* indicado en el campo ya mencionado **data_service** y le enviamos nuestro objeto DatosConfig al subsistema Servicios, cuando dicho subsistema finaliza la interacción con la fuente Web nos retorna el mismo objeto pero con el atributo 'datos' que contiene la totalidad de las tuplas brindadas por la aplicación externa (nótese que si por algún motivo dicha aplicación presentó errores o filtraciones sobre la extracción de los datos nuestro Sistema no se enterará de ello).

Una vez que finalizamos la extracción de los datos y los tenemos en memoria, procedemos a persistirlos sin procesamiento. No todos los datos publicados por la fuente Web son los que finalmente extraemos, sólo aquellos indicados en el mapping entre el Source Schema (esquema que nos permite visualizar todas las columnas de datos provistos por la Web) y el Expected Schema (esquema que entre otras cosas nos permite realizar una selección de las columnas que verdaderamente queremos extraer de una fuente Web).

Nos vamos a crear una instancia de un objeto que nos provee también el subsistema Servicios, que nos permite indicar el nombre de la tabla donde vamos a insertar los datos, y el conjunto de tuplas obtenido por la extracción y que además queremos persistir y por último la entidad también nos provee de un método que se comunica con el subsistema de persistencia para cargar las tablas. A continuación, las tablas que alojan las tuplas extraídas para cada Web Source.

El nombre de la tabla que guardará los datos extraídos para esta Web Source sabemos que lleva el nombre igual a la concatenación del nombre del Expected Schema y el nombre de la Web Source, dicha tabla pertenece al Source Schema. Existe un mapping uno a uno entre los datos extraídos de la fuente Web y los datos que pertenecen a la estructura de esta tabla, recordar que la tabla fue creada con los atributos pertenecientes al Expected Schema y las columnas de los datos extraídos son exactamente los pertenecientes a la estructura del Expected Schema.

Este mecanismo será utilizado tantas veces por Web Source tengamos en nuestra Configuración asociada, como mencionamos anteriormente, por cada registro obtenido de la consulta inicial nos generamos el objeto DatosConfig, y con esa colección de DatosConfig es que invocamos determinada cantidad de veces a algún servicio externo publicado.

Finalizada esta actividad podemos decir que todo el Source Schema está cargado, o sea, todas las tablas que le pertenecen contienen todos los datos extraídos (como se mencionó anteriormente son sólo los deseados).

Integrated and Load Expected Tables

La siguiente actividad también de ejecución automática, deberá realizar la *union* de todas las tablas del Source Schema que se correspondan con el mismo Expected Schema y cargar la totalidad de las tablas de este último, esto significa que los datos extraídos corresponden a la misma entidad en el DW pero fueron extraídos de distintas Web Source. Es de interés como primer procesamiento del proceso de Carga, cargar todos estos datos en un mismo lugar. Para ello, obtenemos de la tabla

config_web_source_expected_esquema la pareja de campos **expected_esquema** y **web source**, esto es, obtenemos las referencias a cada Web Source y su Expected Schema, como siempre filtrando por el campo **config**.

Para cada pareja obtenida anteriormente, concatenamos sus nombres y obtenemos el nombre de la tabla del Source Schema que tiene todas las tuplas extraídas de esa Web Source, vamos a querer insertarla en un medio común a todas las Web Source que tengan el mismo Expected Schema. Para ello utilizaremos las tablas del Expected Schema que creamos anteriormente, cuyo nombre es igual al nombre del Expected Schema. Por lo tanto, todas las tuplas de las tablas del Source Schema cuya Web Source sean el mismo Expected Schema, se guardarán en una tabla del Expected Schema de nombre: el nombre del Expected Schema.

Finalizada esta actividad los registros de todas las tablas del Source Schema se llevaron a la tabla del Expected Schema asociado a la tabla del Source Schema (recordar que la asociación viene dada por la Web Source). Por lo que podemos decir que el Expected Schema está completamente cargado. Observación: es sencillo el traslado de los registros ya que recordar que los atributos de las tablas del Source Schema son los mismos que los atributos de las tablas del Expected Schema, ya que éstas fueron utilizadas para crear las primeras como se mencionó oportunamente.

Integrate and Load Integrated Tables

La actividad *"Integrate and Load Integrated Table"* tendrá el objetivo de cargar completamente todas las tablas pertenecientes al Integrated Schema vinculando de alguna manera las tablas del Expected Schema, que como ya vimos están cargadas completamente desde la actividad anterior. Por lo que deberá consultar en la base de metadatos para saber cuáles son las tablas del Integrated Schema que se deben cargar y cuáles son las tablas del Expected Schema (y además como debemos vincularlas para generar información de valor) que debemos consultar.

El procesamiento de esta actividad se llevará a cabo en dos partes. La primera de ellas es más simple y se da cuando la carga de una tabla del Integrated Schema requiere el traslado lineal de registros de una tabla del Expected Schema a dicha tabla del Integrated Schema. A nivel de configuración, esto implica que no hubo joins para definir la carga de esta tabla del Integrated Schema por lo que en la instancia de mapping con el Expected Schema sólo una tabla de éste estuvo involucrada. Para realizar esto, vamos a consultar a la base de metadatos, todas las tablas del Integrated Schema que cumplen que, están en la tabla **mapping_integrated** pero no están en la tabla **joins_integrated**, en ambos casos filtrando por el campo **config**. Una vez que tenemos los nombres de estas tablas, para cada una de ellas vamos a consultar cuál es la tabla del Expected Schema con la cual la mapeamos. Lo haremos de la manera siguiente: consultamos la tabla **mapping_integrated** y filtramos por los campos **config** e **integrated_esquema** con los valores id de Configuración y nombre de la tabla del Integrated Schema que obtuvimos. De esa consulta, nos quedamos con los campos **expected_esquema** y **atributo_expected_esquema** que son los que se mapearon justamente con la tabla del Integrated Schema. Nos quedamos con todas las tablas que cumplen lo mencionado.

De la tabla **mapping_integrated**, filtrando por **config** y por **integrated_esquema**, podemos obtener los campos **atributo_integrated_esquema**,**expected_esquema** y **atributo_expected_schema**, esto nos indica con qué campo de la tabla del Expected Schema cargamos qué campo de la tabla del Integrated Schema.

Luego de eso, resta armar las sentencias SQL *select* e *insert*, en las tablas del Expected Schema y de las Integrated Schema respectivamente.

Como mencionamos anteriormente, el procesamiento de esta actividad tenía dos partes distintas, ya comentamos la primera. La segunda es más compleja dado que se centraliza en el análisis de qué tablas del Expected Schema joineamos para cargar las distintas tablas del Integrated Schema.

Primero determinamos cuáles tablas del Integrated Schema son las que deben cargarse con este procedimiento. Para eso, vamos a la tabla de la base de metadatos **joins_integrated** y filtrando en **config** obtenemos los resultados para el campo **integrated_esquema**. Allí estarían el total de tablas del Integrated Schema que debemos cargar.

Luego, para cada una de las tablas que obtuvimos en el párrafo anterior, vamos a obtener todos los joins que se configuraron entre tablas del Expected Schema en el momento de definir joins para dichas tablas del Integrated Schema. Esto lo logramos consultando la tabla **joins_integrated** filtrando por Configuración y además en el campo **integrated_esquema**, el nombre de la tabla del Integrated Schema. Para obtener los joins, en dicha consulta nos quedamos con los campos **expected_esquema1**,**atributo_expected_esquema1**, **expected_esquema2** y **atributo_expected_esquema2** que significan atributos de alguna tabla de Expected Schema junto con las mismas. Con estas cuaternas de valores, nos vamos armando el join que deben componer para conformar la tabla del Integrated Schema, y una vez logrado esto, ya tenemos la parte más importante de la consulta *select* que cargara la tabla del Integrated Schema que corresponde.

Para saber qué campos de las tablas del Expected Schema nos proveerán los datos para cargar los campos de las tablas del Integrated Schema, nos fijamos en la tabla de la base de metadatos, **mapping_integrated** (filtrando por **config** y por **integrated_esquema**, que son id de Configuración y nombre de la tabla del Integrated Schema que deseamos cargar) y nos quedamos en la *select* con los campos **atributo_integrated_esquema**, **expected_esquema**, **atributo_expected_schema** (que refieren a atributo de la tabla del Integrated Schema, nombre de la tabla del Expected Schema y atributo de la tabla del Expected Schema). Esto nos permitirá obtener la parte del *from* de la consulta final que queremos obtener, realizando el producto cartesiano de todas las tablas del Expected Schema que están involucradas en el join.

Para armar la parte del *where* de la consulta, sabemos por qué campos joinean las tablas involucradas en el *from* que mencionamos más arriba, esto lo configuramos en la instancia de join para las tablas del Integrated Schema y obtenemos los registros de la tabla de la base de metadatos **joins_integrated**, que nos permite saber para cada join definido en la instancia de Configuración cuáles son los atributos por los cuáles joinean las tablas del Expected Schema.

Finalmente, para armar la consulta SQL final que cargará una tabla del Integrated Schema, basta con concatenar las distintas partes que hemos ido creando para ejecutar el *insert* en la tabla del Integrated Schema seguido de una *select* sobre tablas del Expected Schema.

Integrate and Load DW Dimension Tables

La manera en que cargamos las tablas del DW Schema es muy similar a como cargamos las tablas del Integrated Schema, ya que la definición de joins en ambas instancias es análoga. La única diferencia, es que en este caso lo haremos sólo para las tablas de dimensiones.

El resto del funcionamiento es análogo, esto quiere decir que también se hará en dos partes diferenciadas, una para tablas del DW Schema que se cargan directamente con una y sola una tabla del Integrated Schema (casos excepcionales) y la otra parte se basa en, de manera secuencial, armar por un lado el *insert* sobre cada tabla del DW Schema que queremos cargar, por otro lado obtener la parte del *select* con los campos de dichas tablas de dimensiones, luego armar el *from* con las tablas del Integrated Schema que conformaron el mapping con las tablas del DW Schema, y finalmente, la parte del *where* que se obtiene de la instancia de definición de joins entre las tablas mencionadas.

A continuación, explicaremos las tablas de la base de metadatos de donde obtendremos lo mencionado recientemente: la tabla **atributos_dw_esquema** (filtrando por los campos **config** y **dw_esquema**, que son el id de Configuración y el nombre de la tabla del DW Schema que queremos cargar respectivamente) para obtener la parte de *insert* de la consulta que deseamos obtener.

Para obtener el *select* de dicha consulta, debemos saber qué atributos de las tablas del Integrated Schema están mapeados con los atributos de la tabla del DW Schema, esto lo vemos en la tabla de la base de metadatos **mapping_dw**. Además también, podemos obtener la parte del *from* dado que son estas mismas tablas del Integrated Schema.

Para saber la parte del *where*, esto se definió en la instancia de joins entonces para ello consultamos la tabla de joins_dw y podremos obtener qué atributos de las tablas del Integrated Schema se igualan para poder obtener el resultado deseado.

Configurate Query To Load DW Fact Tables

En esta actividad, vamos a preparar la *query* que cargará en esta oportunidad las tablas de hechos, en contraparte a la actividad anterior que cargaba sólo las tablas de dimensiones joineando efectivamente las tablas del Integrated Schema. Para poder preparar la *query* debemos analizar la tabla de joins de la base de metadatos y ver como es la configuración pero sólo para las tablas que como mencionamos nos interesan. Esto lo hacemos en las tablas **joins_dw** y **dw_esquema**, de la primer tabla obtenemos los nombres de las tablas de DW Schema que debemos cargar y de la segunda tabla obtenemos cuáles están marcadas como de hechos en el campo **es_fact**. La manera de construir el *insert*, el *select* y luego las partes de *from* y de *where* es muy similar a como construimos la carga de las tablas de dimensiones del DW Schema.

Una vez obtenida la *query* final, se mostrará al usuario para confirmarla o modificarla, esto es para cada una de las tablas de hechos. Se mostrará la misma en la siguiente actividad, para ello, la concatenación de sentencias que se generó se guardará en las variables del flujo de Activiti.

Confirm Query To Load DW Fact Tables

Cuando cargamos la actividad "Confirm Query To Load DW Fact Tables" levantamos desde la variable de flujo de Activiti la query armada como se mencionó en la sección anterior y se presenta al usuario.

El usuario tendrá la posibilidad de copiar la misma, llevarla a un cliente de base de datos y poder probar la parte del *select*. De esa manera podrá ver los resultados que ésta genera.

Cuando el usuario completa la actividad antes mencionada, lo que hace el Sistema es tomar la *query* y ejecutarla, esto hará que una tabla de hechos del DW Schema se cargue con la misma.

Las últimas dos actividades que se explicaron (*"Configurate Query To Load DW Fact Tables"*, *"Confirm Query To Load DW Fact Tables"*) se ejecutarán en una iteración por cada tabla de hechos definida en el DW Schema.

Commit Load of Data Warehouse

Cuando cargamos la actividad *"Commit Load of Data Warehouse*" se habrán cargado todas las tablas del DW Schema, en especial las tablas de hechos. Pero le damos la oportunidad al usuario de poder ver los resultados sobre las mismas y poder descartar los cambios. Esto permitirá volver a ejecutar las actividades *"Configurate Query To Load DW Fact Tables"* y *"Confirm Query To Load DW Fact Tables"* con la participación del usuario dentro de un bucle que finalizará cuando el mismo determine que ya está satisfecho. Es aquí donde la el proceso de Carga finaliza completamente.

5.4 Detalles técnicos

En esta subsección se explicará cómo se podrían hacer modificaciones en el prototipo. Se pueden realizar cambios ya sea en los modelos de BPM (Proceso de Configuración y de Carga) como también en la lógica de todas las clases que implementan las actividades manuales y automáticas.

El prototipo está compuesto por un proyecto dividido en 6 paquetes como muestra la Figura 31 que son los siguientes:

1) *"base"* que contiene una clase llamada "*ControladorBD*" la cual encapsula todas las llamadas a las bases de datos (la base de metadatos del sistema y las bases de datos correspondientes a los DW construidos).
2) *"constant"* que contiene la clase llamada "*FlowConstant*", esta almacena todas las variables de los flujos de los procesos y subprocesos utilizados en la solución.
3) *"customTypes"*, en este paquete se encuentran todos los tipos de datos creados para usar como combos o como campos de textos extensos dentro del Activiti, los cuales explicaremos más en detalle.
4) *"dtos"* contiene los datatypes utilizados en todas las capas de la aplicación.
5) *"procesos"*, este paquete tiene todas las clases que resuelven las acciones realizadas por cada actividad de los procesos , es decir contiene los *listeners* y los *java delegate* que implementan las funcionalidades de las distintas actividades.
6) *"services"* que contiene los servicios utilizados en la solución, por ejemplo el cliente Rest para invocar a los Data Services provistos por terceros, como también la extracción de la primera fila de los datos publicados en la web con su tipo correspondiente (sea la fila de títulos o la primera fila de datos en caso que no existan títulos). Además, contiene una clase que nos brinda servicios de parseo de *json* a objetos del Sistema.

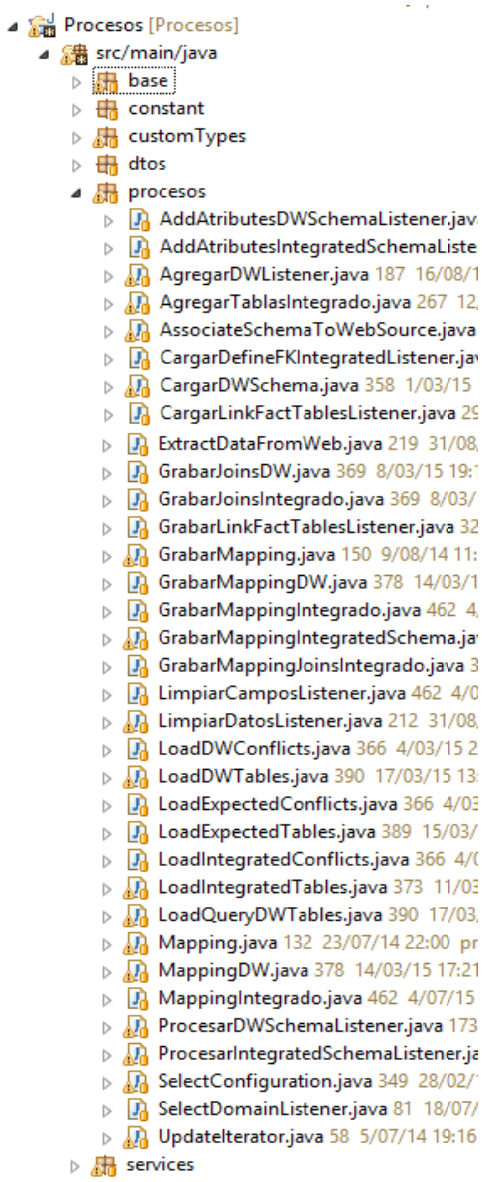


Figura 31 Vista de paquetes en Eclipse

Existe una configuración previa para los tipos de datos implementados por nosotros, la misma se encuentra en la carpeta webapps donde se *deployan* los *war* en el archivo "activiti-standalone-context.xml" (activiti-explorer\WEB-INF) en el *tag* "property" se detallan los tipos creados por el desarrollador de la siguiente manera:

<property name="customFormTypes">

<list>

<bean class="customTypes.UrlsFormType" />

</list>

</property>

Y finalmente en el archivo "activiti-ui-context.xml" (activiti-explorer\WEB-INF) en el *tag* "property" se especifica el *Renderer* del tipo *customizado* de la siguiente manera:

```
<property name="propertyRenderers">

<list>

    <bean class="customTypes.UrlsFormPropertyRenderer" />

</list>

</property>
```

En cuanto al paquete de tipos de datos (*"customTypes"*), cada uno de estos cuenta con una clase que extiende "*AbstractFormType*" el cual contiene una constante llamada "*TYPE_NAME*" que indica el nombre del tipo que será utilizado luego en el modelo del proceso. Y además contiene otra clase que de la misma forma extiende "*AbstractFormPropertyRenderer*" y sobrescribiendo el método "*getPropertyField*" de forma de *customizar* nuestro tipo deseado (estas clases son las que se parametrizan de la manera descrita anteriormente).

Otro punto a tener en cuenta, luego de generar el *jar* del proyecto y el bar del modelo (esto es desde la vista de Activiti del eclipse click derecho en el proyecto y "Create deployment artifacts") este *jar* se debe poner en activiti-explorer\WEB-INF\lib y en la sección de "Gestionar" -> "Despliegue" -> "Cargar nueva" se sube el .bar correspondiente a los modelos generados.

Teniendo en cuenta que las principales características de las actividades se encuentran en la pestañas *General* (id y nombre), *Form* (los campos con sus tipos y sus propiedades) *Listeners* (las clases invocadas y en qué momento, por ejemplo *complete*).

A continuación en la Tabla 4 explicaremos para cada actividad, al proceso al que pertenecen y que clases se ejecutan en que evento. Cada clase que implementamos se encarga de realizar una funcionalidad en particular y se ejecuta en un evento concreto, esto nos permite independizar las clases que se ejecutan en los diferentes eventos de las actividades.

Tabla 4 Descripción de clases ejecutadas en distintos eventos de las actividades de los procesos.

NOMBRE DE ACTIVIDAD	PROCESO	CLASES	EVENTO
Select Domain	Configuration.bpmn	SelectDomainListener.java	complete
Select Web Sources	Configuration.bpmn	DefineSourcesSchemas.java	complete
		LimpiarCamposListener.java	create
Create Web Sources	Configuration.bpmn	CreateWebServices.java	complete
		LimpiarCamposListener.java	create
Define Joins Integrated Schemas	Configuration.bpmn	DefineJoinsIntegrado.java	create
		GrabarJoinsIntegrado.java	complete
		LimpiarCamposListener.java	complete
Define Joins DW Schema	Configuration.bpmn	DefineJoinsDW.java	create
		GrabarJoinsDW.java	complete

		LimpiarCamposListener.java	complete
Define Expected Schema	DefineExpectedSchemaAndMapping.bpmn	CargarWebSource.java	create
Process Expected Schema	DefineExpectedSchemaAndMapping.bpmn	DefineExpectedSchemaAndMapping.bpmn	NA
Create Expected Schema And Atributes	DefineExpectedSchemaAndMapping.bpmn	DefineSchemaListener.java	complete
		LimpiarCamposListener.java	create
Add More Atributes For Expected Schema	DefineExpectedSchemaAndMapping.bpmn	DefineAtributesListener.java	complete
		LimpiarCamposListener.java	create
Mapping Expected Schema And Web Source	DefineExpectedSchemaAndMapping.bpmn	Mapping.java	create
		GrabarMapping.java	complete
Upload Data Service	DefineExpectedSchemaAndMapping.bpmn	DefineImplementation.java	complete
Define Integrated Schema	DefineIntegratedSchema.bpmn	DefineIntegratedSchemaListener.java	create
Process Integrated Schema	DefineIntegratedSchema.bpmn	ProcessIntegratedSchemaListener.java	NA
Create Table And Atributes For Integrated Schema	DefineIntegratedSchema.bpmn	CreateIntegratedSchemaListener.java	complete
		LimpiarCamposListener.java	create
		AgregarTablasIntegrado.java	create
Add More Atributes For Integrated Schema	DefineIntegratedSchema.bpmn	AddAtributesIntegratedSchemaListener.java	complete
		LimpiarCamposListener.java	create
Define Foreign Keys Integrated Schema	DefineIntegratedSchema.bpmn	GrabarMappingJoinsIntegrado.java	complete
		CargarDefineFKIntegratedListener.java	create
		LimpiarCamposListener.java	create
Mapping Integrated Schema and Expected Schema	DefineIntegratedSchema.bpmn	MappingIntegrado.java	create
		GrabarMappingIntegrado.java	complete
		LimpiarCamposListener.java	create
Define DataWarehouse Schema	DefineDWSchemaAndMapping.bpmn	DefineDWSchemaListener.java	create
Process DW Schema	DefineDWSchemaAndMapping.bpmn	ProcesarDWSchemaListener.java	NA
Create Table And Atributes For DataWarehouse Schema	DefineDWSchemaAndMapping.bpmn	CreateDWSchemaListener.java	complete
		LimpiarCamposListener.java	create
		LimpiarDatosListener.java	create
Add More Atributes For DataWarehouse Schema	DefineDWSchemaAndMapping.bpmn	AddAtributesDWSchemaListener.java	complete
		LimpiarCamposListener.java	create

Define Foreign Keys of Fact Tables for DataWarehouse Schema	DefineDWSchemaAndMapping.bpmn	CargarLinkFactTablesListener.java	create
		GrabarLinkFactTablesListener.java	complete
		LimpiarCamposListener.java	create
Mapping DataWarehouse Schema and Integrated Schema	DefineDWSchemaAndMapping.bpmn	MappingDW.java	create
		GrabarMappingDW.java	complete

5.5 Limitaciones

En cuanto a las limitaciones de la solución planteada, algunas de ellas están dadas por el Activiti Explorer en el cual las interfaces son estáticas, esto lo podemos ver en varias de las actividades por ejemplo en *"Select Web Sources"* (Ver Figura 32) en la cual tenemos 5 combos de posibles Web Sources para seleccionar y en caso de querer seleccionar más fue necesario agregar un checkbox para ello, en ese caso se le presentarán 5 combos mas.

Este criterio se mantiene para todos los formularios en los que se necesitan continuar ingresando información.

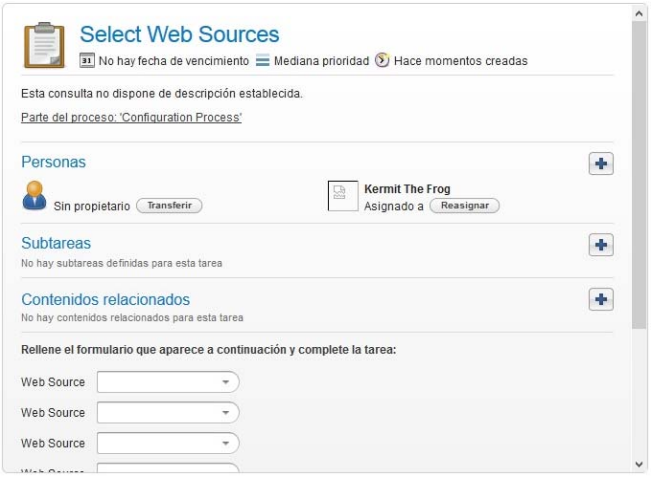


Figura 32 Ejemplo de actividad manual con controles fijos en formulario de Activiti

En cuanto a los formatos que se *parsean* desde la Web, si bien la extracción en si no es parte del problema planteado, el Sistema realiza una pequeña extracción (la primera fila, sean los títulos o la primer fila de datos) para poder facilitarle al usuario el mapping, estos solo es posible si los formatos son HTML o CSV, por este motivo implementamos una variante para realizar el mapping de un formato nuevo no soportado para que se ingrese manualmente el nombre de la columna o dato de la primer fila.

Otra limitante del Sistema es la calidad de los datos, como mencionamos anteriormente no se incluyó la calidad de los datos extraídos, esto trae aparejado por ejemplo posibles problemas de datos que signifiquen los mismo pero que tengan códigos diferentes en las distintas Web Sources (por ejemplo si tuviéramos departamentos en dos Web Sources diferentes en donde el código en uno podría ser el nombre del departamento y en otro el nombre abreviado y aún así significarían lo mismo).

Finalmente otra limitante que vemos es que las medidas de los cubos que se armen para explotar el DW son todas medidas que solamente cuentan, es decir que para obtener dicha medida se realiza la cuenta de registros que tenga la tabla de hechos para el caso que se quiere analizar.

5.6 Posibles extensiones y mejoras

Comenzando con mejoras sobre las limitaciones planteadas, con respecto a los formularios de Activiti se podría utilizar otra herramienta como por ejemplo "Bonita" [8], que solucione el problema de que el formulario sea estático a nivel de controles, pudiendo reutilizar las clases java y su lógica junto por supuesto con la base de datos y su estructura.

En cuanto a la calidad de datos, como se mencionó antes en este documento planteamos agregarla en varios puntos del Proyecto, en el proceso de Configuración se podría agregar una etapa al final en la cual se crearían los metadatos relacionados a la calidad de datos (definición de métricas de calidad, cómo medir las mismas y sobre qué aplican) o también podría agregarse luego de cada definición de los esquemas (tanto el Expected Schema, Integrated Schema y DW Schema). En relación al proceso de Carga, ya dejamos planteada la opción de identificar y resolver los problemas o conflictos. Quizás teniendo en cuenta lo definido en la Configuración y utilizando los metadatos de calidad se podría resolver en cada paso los problemas que surjan, podría ser tal vez teniendo una tabla de "errores" o "problemas" por cada tipo y en cada punto del proceso donde surja y luego consultando estos inconvenientes para que se resuelvan manualmente por parte del usuario.

Otra posible mejora que pensamos es tal vez tener un Sistema adicional en el cual se pudieran realizar las altas, bajas y modificaciones de objetos de nuestra realidad, por ejemplo los formatos, los dominios. Esto también se podría resolver creando un modelo BPM que resuelva este tipo de proceso.

Finalmente planteamos una última mejora que consiste en poder confirmar o modificar los esquemas configurados, es decir que luego de ingresarlos poder volver atrás si algo no satisface al usuario, en este sentido podría haber una tarea manual luego de las actividades de *"Show Integrated Schema"* y *"Show DW Schema"* con la posibilidad de volver a configurar los mismos.

6 Verificación

En este capítulo se presentarán algunas de las pruebas que se realizaron del Sistema junto con los resultados esperados y obtenidos de dichas pruebas.

6.1 Contexto 1.

El primer contexto que vamos a utilizar para probar nuestro prototipo estará basado en el transporte metropolitano. Contamos con una Web Source que nos provee todas las empresas y líneas que dan servicio a todos los pueblos y ciudades de los departamentos vecinos de la capital, además de cada parada y horario por donde pasan los servicios, indicando el origen y el destino. La publicación de dicha Web Source se realiza en formato CSV y lo que hicimos fue agregar a cada lugar de origen o destino el departamento al cual pertenece para darle un poco de más valor. Por lado, nos armamos un CSV con los departamentos que limitan con Montevideo y por otro lado, un CSV que tiene los tipos de días que se pueden encontrar en un listado de horarios de transporte, si son días de sábado, domingo o entre semana.

6.1.1 Diseño del Web Warehouse

En función de lo comentado anteriormente, vamos a diseñar un WW que nos permita analizar los datos referentes a todo el transporte metropolitano, desde el punto de vista de las líneas pudiendo agrupar por empresas, desde las localidades origen o destino o también según los días de la semana. En la Figura 33 se muestra un bosquejo de cómo sería el diagrama del WW.

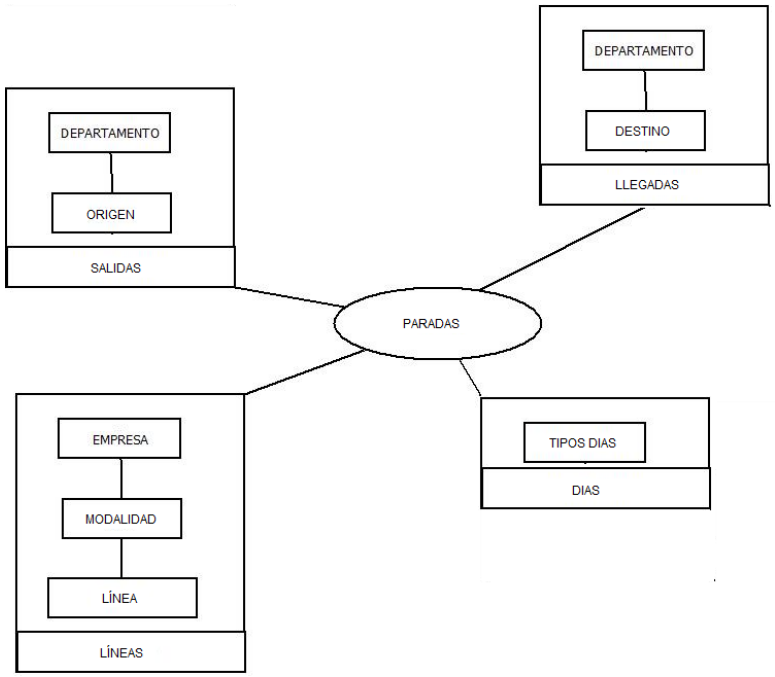


Figura 33 Diseño del Web Warehouse

Las dimensiones son las siguientes:

- Salidas.
- Llegadas.
- Líneas.
- Días.

Los datos de las dimensiones básicamente serán extraídos de la misma Web Source que mencionamos más arriba, pero en distintas instancias de extracción. En la Figura 34 mostramos el contenido de la tabla de hechos Paradas y las tablas de dimensiones de Líneas, Días, Salidas y Llegadas.

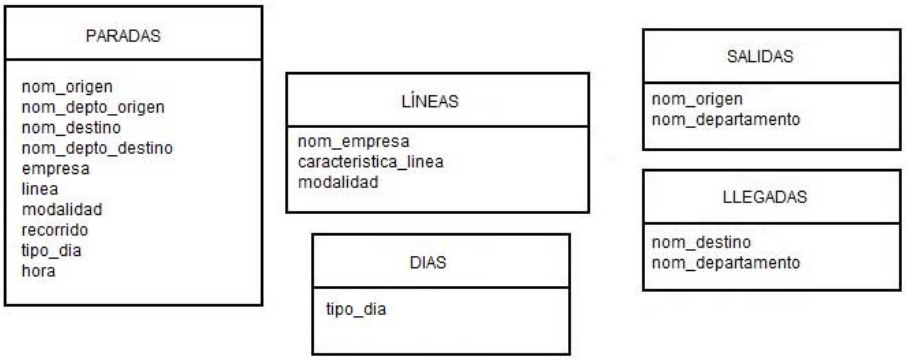


Figura 34 Tablas de dimensiones y de Hechos.

6.1.2 Recursos y Metadatos

Horarios Interdepartamentales (Dirección Nacional de Transporte)

Desde una página muy utilizada por nosotros que forma parte de la Agesic, están publicados un conjunto de catálogos que proveen datos diversos, en este caso vamos a utilizar datos referentes a Transporte en el país. Para este caso, vamos a realizar un estudio sobre todo el transporte que sale y/o llega a la capital del país desde las ciudades incluidas en la zona metropolitana, que abarca los departamentos de Canelones, Florida y San José.

A continuación, el detalle de los datos provistos.

Recurso

Recurso	**Url**	**Cantidad de Registros**
Horarios	https://catalogodatos.gub.uy/dataset/horarios-interdepartamentales-omnibus	100780

Metadatos de contenido

Nro. Columna	**Nombre**	**Tipo de Dato**	**Descripción**
1	Empresa	Alfanumérico (12)	Empresa prestadora del servicio.
2	Característica	Alfanumérico(10)	Es el código de línea que representa el servicio.

3	Origen	Alfanumérico(100)	Es el nombre de la localidad, terminal o intersección de calles donde comienza el viaje.
4	Departamento Origen	Alfanumérico(50)	Departamento de se encuentra ubicado el Origen.
5	Destino	Alfanumérico(100)	Es el nombre de la localidad, terminal o intersección de calles donde finaliza el viaje.
6	Departamento Destino	Alfanumérico(50)	Departamento de se encuentra ubicado el Destino.
7	Recorrido	Alfanumérico (100)	Secuencia de calles, avenidas o rutas más importantes en el trayecto del viaje.
8	Día	Alfanumérico(12)	Tipo de día, esto es, sábado, domingo o hábil.
9	Modalidad	Alfanumérico(50)	Tipo de viaje, si es directo, semi-directo, camino (común), etc.
10	Parada	Alfanumérico(100)	Calle, avenida, terminal, etc. en donde existe una parada en el viaje.
11	Hora	Alfanumérico(6)	Hora en la que ocurre la parada mencionada anteriormente.
12	F. Inicio	dd/mm/aaaa	Fecha de inicio de vigencia de horario.
13	F. Fin	dd/mm/aaaa	Fecha de fin de vigencia de horario.

6.1.3 Integrated Schema

En la Figura 35 podemos ver el resultado del Integrated Schema a partir de las Web Source utilizadas como explicamos en la sección Recursos y Metadatos.

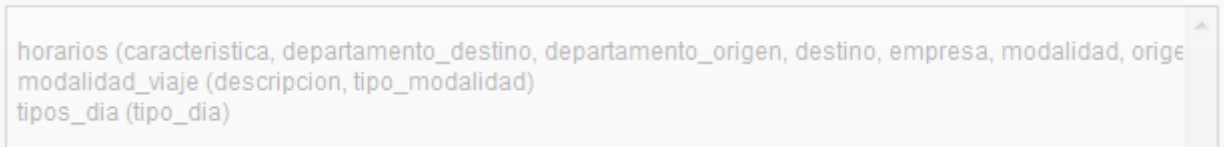


Figura 35 Resultado Integrated Schema.

6.1.4 DW Schema

Análogamente al Integrated Schema, podemos ver el resultado del DW Schema, tal cual lo muestra la Figura 36, y refleja el diseño que deseamos construir expuesto en Figura 33.

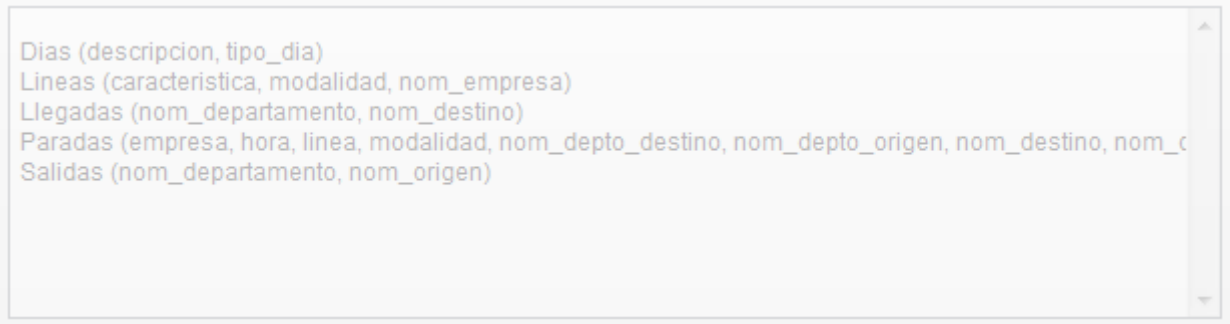


Figura 36 Resultado DW Schema.

6.1.5 Resultados esperados

Una ejecución exitosa de esta prueba sería poder extraer la totalidad de los datos y finalmente poder cargar las tablas del DW, esto es, las tablas de Salidas, Llegadas, las Líneas disponibles, el catálogo de tipos de Días y finalmente la tabla de hechos, que contiene las diferentes paradas que se pueden dar en dichos viajes.

6.1.6 Resultados obtenidos

Realizamos la verificación sobre los metadatos correspondientes a toda la configuración y no encontramos ningún problema o dato que falte. Se guardo correctamente toda aquella parametrización correspondiente a las Web Source, a los Expected Schemas, Integrated Schemas y de DW.

Además, corroboramos que determinada información que pueda utilizarse más adelante en otros procesos haya quedado disponible como por ejemplo: las Web Source utilizadas asociadas al Dominio Transporte. Los Expected Schemas creados, están también disponibles en próximas ejecuciones.

En lo que refiere a las etapas de carga, podemos ver en primera instancia como se crearon las estructuras con éxito, se crearon las tablas que guardarán los datos extraídos, las tablas de integración de Expected Schema y las tablas del Integrated Schema. Así como también el conjunto de tablas del DW.

En la Figura 37 podemos visualizar las estructuras generadas luego de ejecutar el proceso de Carga, notar que el nombre de la base de datos refiere al número de proceso de configuración.

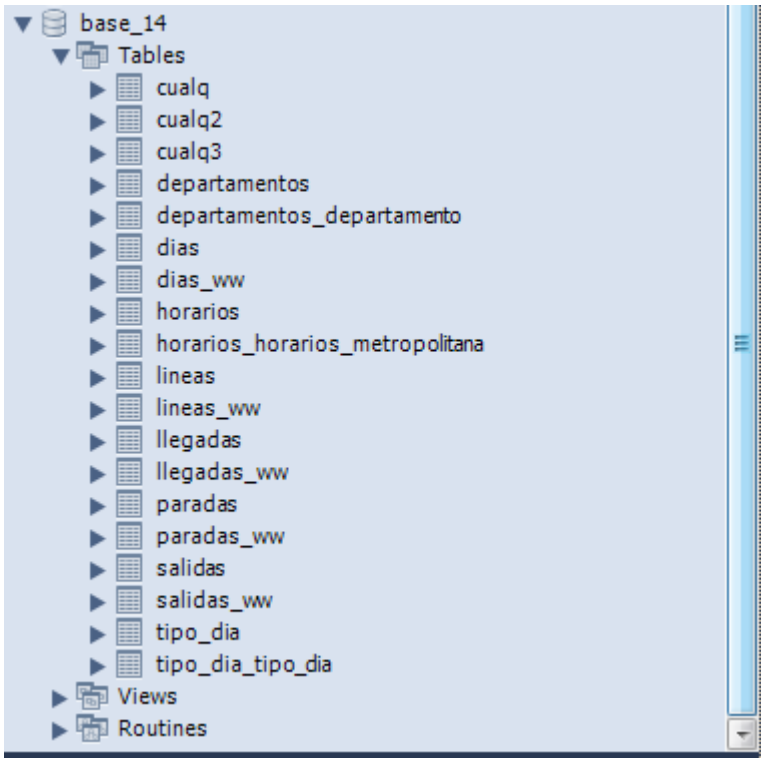
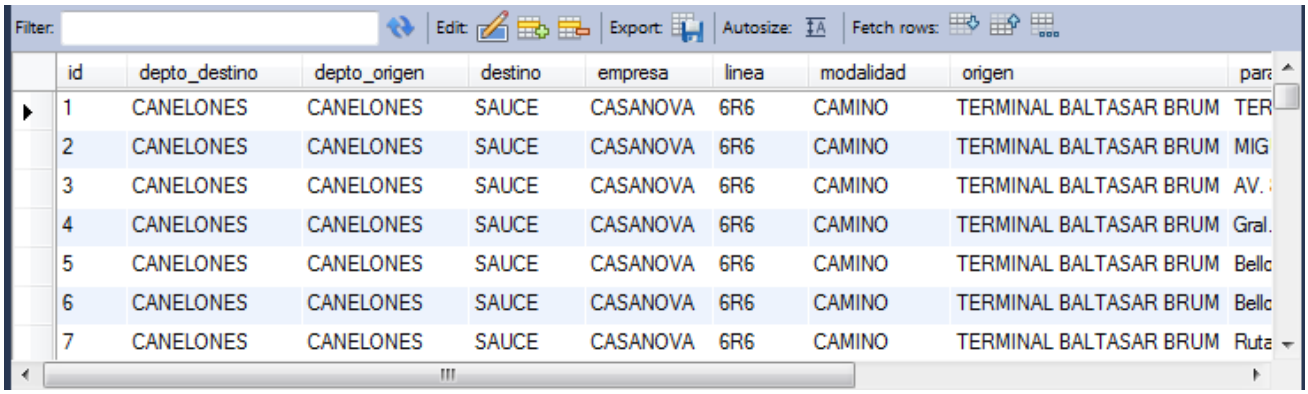


Figura 37 Resultado del Proceso de Carga en cuanto a estructuras

Con respecto a la extracción de horarios podemos decir que había un total de 100.780 registros y se extrajeron con éxito poco más de 89.000. La principal causa de esta pérdida de los datos es debido al manejo de una cantidad muy elevada de registros que mencionaremos en la sección contigua que trata sobre problemas encontrados. Otra causa es sobre la naturaliza de los caracteres especiales que pueden contener algunas palabras que provocan errores de inserción en base de datos. Los departamentos y los tipos de días se extrajeron sin inconvenientes.



id	depto_destino	depto_origen	destino	empresa	linea	modalidad	origen	para
1	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	TER
2	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	MIG
3	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	AV.
4	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	Gral.
5	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	Bello
6	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	Bello
7	CANELONES	CANELONES	SAUCE	CASANOVA	6R6	CAMINO	TERMINAL BALTASAR BRUM	Ruta

Figura 38 Resultado de la integración de los datos

Verificamos las tablas del Integrated Schema y pudimos constatar que las mismas fueron cargadas de manera correcta, esto es: departamentos, días, horarios, líneas, llegadas, paradas y salidas. Esto se ilustra en la Figura 38.

6.1.7 Problemas encontrados

La Web Source que nos provee los datos de las diferentes paradas contiene más de 100.000 registros, esto nos trajo problemas a la hora de extraer las mismas, ya que se realiza todo en memoria y con la misma invocación al servicio de extracción. Dada la cantidad de datos que manejamos en esta oportunidad vemos como el buffer donde guardamos los que vamos extrayendo sufre problemas de congestión.

Este problema podría solucionarse con dos llamadas al servicio, la primera que implemente la búsqueda de cuántos registros se van a obtener de esa Web Source, y la segunda que (según valores parametrizables) invoque al servicio de manera iterativa y controlada de manera de no exceder los recursos disponibles.

Además de la saturación del Sistema, vemos como las demoras son importantes, antes grandísimas cantidades de datos. Esto no necesariamente es un impedimento, ya que las cargas de DW se suelen hacer en horas de la noche, en donde la demanda de servicios de las base de datos es nula y los procesos de extracción son densos. Por lo que vemos aquí otra solución: que la actividad de extracción se realice también en horas nocturnas y dejando para las horas laborables las actividades manuales y de interacción humana.

6.2 Contexto 2.

El segundo contexto que vamos a utilizar para probar nuestro prototipo estará basado en las estadías de los clientes en los diferentes hoteles de Uruguay. Contamos con una Web Source que nos provee todos los hoteles, de esa misma fuente extraeremos las ciudades de nuestro WW, además para obtener los clientes utilizamos una Web Source de guías turísticos, los cuales simularemos como clientes y finalmente un CSV creado por nosotros correspondiente a las estadías de los clientes en los hoteles. La publicación de dichas Web Sources se realiza en formato HTML. Por lado, nos armamos un CSV con las estadías como comentamos anteriormente.

6.2.1 Diseño del Web Warehouse

En función de lo comentado, vamos a diseñar un WW que nos permita analizar los datos referentes a las Estadías, pudiendo posteriormente explotar el WW mostrando por ejemplo las ciudades de las cuales provienen los clientes de los hoteles.

Las dimensiones son las siguientes:

- Ciudades.
- Clientes.
- Hoteles.
- Estadías.

Los datos de las dimensiones serán extraídos de la siguiente manera, los datos de Ciudades y Hoteles de la Web Source de Hoteles, pero en distintas instancias de extracción, los datos de los clientes serán extraídos de la Web Source de guías turísticos y finalmente las estadías del CSV que creamos nosotros.

6.2.2 Recursos y Metadatos

Hoteles (Ministerio de Deportes y Turismo)

Desde una página web publicada por el Ministerio de Turismo vamos a extraer datos de Hoteles en el Uruguay. A continuación podemos ver una imagen que ilustra cómo están presentados esos datos. En este caso el formato es HTML y los datos relacionados al Hotel están separados por comas.

Recurso

Recurso	Url	Cantidad de Registros
Hoteles	http://apps.mintur.gub.uy/datosAbiertos/hoteles.php	468

Guías Turísticos (Ministerio de Deportes y Turismo)

Desde una página web publicada por el Ministerio de Turismo vamos a extraer datos de guías turísticos en el Uruguay que utilizaremos como clientes. A continuación podemos ver una imagen que ilustra cómo están presentados esos datos. En este caso el formato es HTML y los datos relacionados a los guías turísticos están separados por comas.

Recurso

Recurso	Url	Cantidad de Registros
Guías Turísticos	http://apps.mintur.gub.uy/datosAbiertos/guiasTuristicos.php	632

6.2.3 Integrated Schema

En la Figura 39 podemos ver lo que nos muestra la actividad "Show Integrated Schema" que nos permite obtener el resultado del Integrated Schema a partir de las Web Source utilizadas como explicamos en la sección Recursos y Metadatos.

Figura 39 Resultado Integrated Schema.

6.2.4 DW Schema

En la Figura 40Figura 39 podemos ver lo que nos muestra la actividad "Show DW Schema" que nos permite obtener el DW Schema planteado inicialmente.

Figura 40 Resultado DW Schema.

6.2.5 Resultados esperados

Una ejecución exitosa de esta prueba sería poder extraer la totalidad de los datos y finalmente poder cargar las tablas del DW, esto es, las tablas de Hoteles, Ciudades, Clientes, Estadías y finalmente la tabla de hechos, que contiene los hospedajes.

6.2.6 Resultados obtenidos

Realizamos la verificación sobre los metadatos correspondientes a toda la configuración y no encontramos ningún problema o dato que falte. Se guardo correctamente toda aquella parametrización correspondiente a las Web Source, a los Expected Schemas, Integrated Schemas y de DW.

Además, corroboramos que determinada información que pueda utilizarse más adelante en otros procesos haya quedado disponible como por ejemplo: las Web Source utilizadas asociadas al Dominio Turismo. Los Expected Schemas creados, están también disponibles en próximas ejecuciones.

En lo que refiere a las etapas de carga, podemos ver en primera instancia como se crearon las estructuras con éxito, se crearon las tablas que guardarán los datos extraídos, las tablas de integración de Expected Schema y las tablas del Integrated Schema. Así como también el conjunto de tablas del DW.

Verificamos las tablas del Integrated Schema y pudimos constatar que las mismas fueron cargadas de manera correcta.

6.2.7 Problemas encontrados

Al no tener una fuente de datos que nos provea las ciudades las extrajimos de la misma Web Source que hoteles y simular la fuente de datos de guías turísticos como si fueran clientes. Además para hacer el caso de prueba más interesante tuvimos que crear el CSV de estadías.

Uno de los temas más interesantes de la prueba es que se extraen datos al mismo tiempo de fuentes de datos con diferente formato. Otro motivo de esta prueba es que los datos para construir y cargar el DW fueron lineales con respecto a las Web Sources y las tablas de dimensiones y de hechos.

7 Caso de estudio.

7.1 Contexto

Para nuestro caso de estudio vamos a utilizar el dominio Turismo y contamos con algunas Web Sources que nos proveen diferentes datos. Deseamos diseñar y construir un WW que analice datos generales de Turismo.

Existen varias publicaciones que tienen relación con el turismo. Entre ellas, el propio Ministerio de Turismo y Deporte, publica los operadores turísticos habilitados. como son las inmobiliarias, alojamientos, establecimientos rurales, guías turísticos, etc. La Intendencia de Durazno publica los eventos que se realizan en el departamento, incluyendo, entre otras cosas, el nombre del evento, la localidad y época o fecha de realización.

Por otro lado, la Dirección Nacional de Transporte pública, en forma trimestral, los horarios vigentes de ómnibus regulares de carácter interdepartamental, de corta, media y larga distancia.

Estos recursos podrían ser utilizados para integrar los lugares turísticos que se pueden visitar y los eventos que se realizan, junto con los horarios de ómnibus disponibles para llegar a dichos lugares desde la localidad que uno desee.

A continuación, se especifican los recursos y metadatos de las publicaciones que vamos a utilizar, además especificamos los datos generados por nosotros que son reales.

7.2 Recursos y Metadatos

Hoteles (Ministerio de Deportes y Turismo)

Desde una página Web publicada por el Ministerio de Turismo vamos a extraer datos de Hoteles en el Uruguay. A continuación podemos ver una imagen que ilustra cómo están presentados. En este caso el formato es *html* y los datos relacionados al Hotel están separados por comas.

Recurso

Recurso	**Url**	**Cantidad de Registros**
Hoteles	http://apps.mintur.gub.uy/datosAbiertos/hoteles.php	468

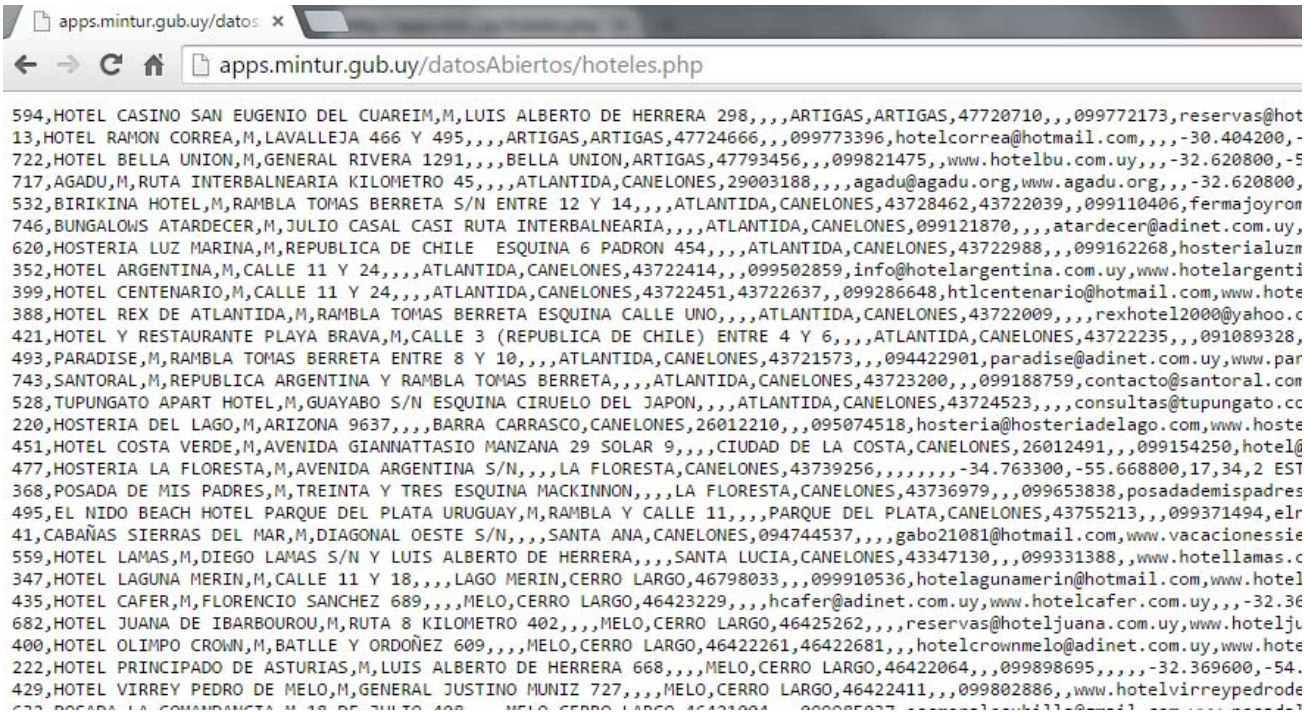


Figura 41 Web de Hoteles de Datos Abiertos

La Figura 41 nos muestra como están publicados los datos de los hoteles en la plataforma de Datos Abiertos, en este caso corresponden al Ministerio de Turismo [9].

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción
1	Identificador	Numérico(3)	Identificador numérico del Hotel en el Sistema del MinTur.
2	Nombre	Alfanumérico(100)	Nombre completo del Hotel.
3	Dirección	Alfanumérico(100)	Dirección completa del Hotel que puede tener calle, intersección, número de puerta, solar, manzana, etc.
4	Localidad	Alfanumérico(50)	Localidad en dónde está ubicado el Hotel, puede ser ciudad o balneario.
5	Departamento	Alfanumérico(50)	Departamento en dónde se ubica el Hotel.
6	Teléfono	Numérico(8)	Número de teléfono fijo.
7	Celular	Numérico(9)	Número de teléfono móvil.
8	Email	Alfanumérico(50)	Casilla de correo electrónico
9	Web	Alfanumérico(100)	Página web
10	Coordenadas	Alfanumérico(100) y Alfanumérico(100)	Coordenadas de su ubicación dadas por latitud y longitud, pueden servir para ubicar el Hotel exactamente en Google Maps
11	Categoría	Alfanumérico(20)	Categoría del Hotel expresado en cantidad de estrellas.

Campings (Ministerio de Deportes y Turismo)

Desde otra página Web también provista por el Ministerio de Turismo, vamos a obtener datos sobre campings. El formato en que están provistos los datos es igual que la Web Source de Hoteles.

Recurso

Recurso	Url	Cantidad de Registros
Campings	http://apps.mintur.gub.uy/datosAbiertos/campings.php	5

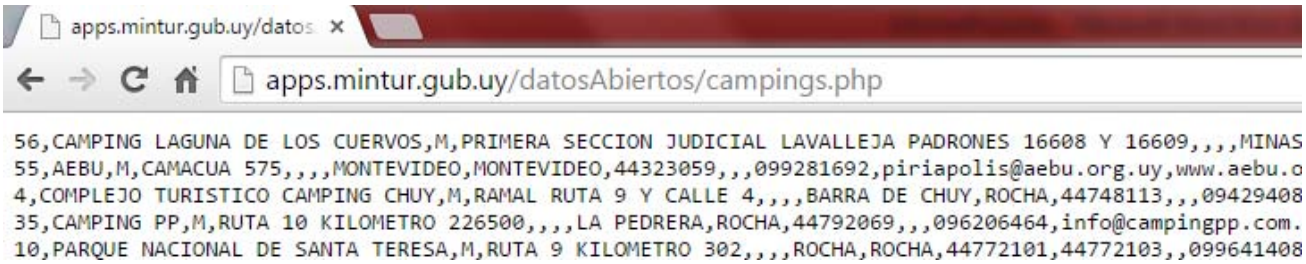


Figura 42 Web de Campings de Datos Abiertos

La Figura 42 nos muestra cómo están publicados los datos de los campings en la plataforma de Datos Abiertos, en este caso corresponden al Ministerio de Turismo [9].

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción
1	Identificador	Numérico(3)	Identificador numérico del Camping en el Sistema del MinTur.
2	Nombre	Alfanumérico(100)	Nombre completo del Camping.
3	Dirección	Alfanumérico(100)	Dirección completa del Camping que puede tener calle, intersección, número de puerta, solar, manzana, etc.
4	Localidad	Alfanumérico(50)	Localidad en dónde está ubicado el Camping, puede ser ciudad o balneario.
5	Departamento	Alfanumérico(50)	Departamento en dónde se ubica el Camping.
6	Teléfono	Numérico(8)	Número de teléfono fijo.
7	Celular	Numérico(9)	Número de teléfono móvil.
8	Email	Alfanumérico(50)	Casilla de correo electrónico
9	Web	Alfanumérico(100)	Página web
10	Coordenadas	Alfanumérico(100) y Alfanumérico(100)	Coordenadas de su ubicación dadas por latitud y longitud, pueden servir para ubicar el Camping exactamente en Google Maps

Albergues (Ministerio de Deportes y Turismo)

Otra página Web que nos provee el Ministerio de Turismo está basada en datos de Albergues del país. Nuevamente adjuntamos una imagen de cómo están distribuidos los datos, el formato es *html* al igual que los casos anteriores.

Recurso

Recurso	Url	Cantidad de Registros
Albergues	http://apps.mintur.gub.uy/datosAbiertos/albergues.php	50

Figura 43 Web de Albergues de Datos Abiertos

La Figura 43Figura 41 nos muestra cómo están publicados los datos de los albergues en la plataforma de Datos Abiertos, en este caso corresponden al Ministerio de Turismo [9].

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción
1	Identificador	Numérico(3)	Identificador numérico del Albergue en el Sistema del MinTur.
2	Nombre	Alfanumérico(100)	Nombre completo del Albergue.
3	Dirección	Alfanumérico(100)	Dirección completa del Albergue que puede tener calle, intersección, número de puerta, solar, manzana, etc.
4	Localidad	Alfanumérico(50)	Localidad en dónde está ubicado el Albergue, puede ser ciudad o balneario.

5	Departamento	Alfanumérico(50)	Departamento en dónde se ubica el Albergue.
6	Teléfono	Numérico(8)	Número de teléfono fijo.
7	Celular	Numérico(9)	Número de teléfono móvil.
8	Email	Alfanumérico(50)	Casilla de correo electrónico
9	Web	Alfanumérico(100)	Página web
10	Coordenadas	Alfanumérico(100) y Alfanumérico(100)	Coordenadas de su ubicación dadas por latitud y longitud, pueden servir para ubicar el Albergue exactamente en Google Maps

Líneas de Ómnibus Departamentales (Dirección Nacional de Vialidad - MTOP)

Recurso

Recurso	Url	Nombre del archivo	Cantidad de Registros
Líneas Departamentales	http://catalogodatos.gub.uy/dataset/aa2405be-6dde-4482-8f5f-563817154282/resource/e62a3ef8-4e00-458b-bcbe-4b293ce0ebba/download/HorariosOmnibusVerano2015.zip	Horarios_LD_DNT.csv	7791

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Id	Empresa	Origen	DeptoOriger	Destino	DeptoDestin	HSalida	HLlegada	Recorrido	Dias	Lugar	DeptoLugar	Hora	FInicio
1	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:00	06:00	1,3	Lu,Ma,Mi,Ju,	MONTEVIDEO	MONTEVIDEO	00:00	20/12/2014
2	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:00	06:00	1,3	Lu,Ma,Mi,Ju,	SALTO	SALTO	06:00	20/12/2014
3	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	MONTEVIDEO	MONTEVIDEO	00:30	20/12/2014
4	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	LIBERTAD	SAN JOSE	01:25	20/12/2014
5	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	SAN JOSE	SAN JOSE	01:45	20/12/2014
6	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	TRINIDAD	FLORES	03:00	20/12/2014
7	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	ANDRESITO	FLORES	03:40	20/12/2014
8	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	YOUNG	RIO NEGRO	04:30	20/12/2014
9	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	PAYSANDU	PAYSANDU	05:30	20/12/2014
10	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	CONSTANCIA	PAYSANDU	05:45	20/12/2014
11	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	EMP. 3 Y 26	PAYSANDU	05:55	20/12/2014
12	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	EMPALME QUEBRACHO	PAYSANDU	06:00	20/12/2014
13	AGENCIA CEI	MONTEVIDE	MONTEVIDE	SALTO	SALTO	00:30	07:00	1,3	Lu,Ma,Mi,Ju,	TERMAS DEL GUAVIYU	PAYSANDU	06:15	20/12/2014

Figura 44 Datos de .csv de Horarios de Transportes de Datos Abiertos

La Figura 44 nos muestra cómo están publicados los datos de horarios de líneas de transporte en la plataforma de Datos Abiertos, en este caso corresponden al Ministerio de Turismo [9].

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción	Observaciones
1	CodServicio	Numérico(5)	Es el código que identifica al registro.	Este dato fue provisto por nosotros sobre el Csv una vez que descargamos el mismo.
2	Empresa	Alfanumérico (30)	Nombre de la empresa que realiza el servicio.	
3	Origen	Alfanumérico (30)	Lugar origen de la línea (Localidad).	
4	Depto. Origen	Alfanumérico (30)	Departamento del lugar de origen.	
5	Destino	Alfanumérico (30)	Lugar de destino de la línea (Localidad).	
6	Depto. Destino	Alfanumérico (30)	Departamento del lugar de destino.	
7	H.Salida	Alfanumérico (5)	Hora de Salida de origen	Formato: hh:mm
8	H.Llegada	Alfanumérico (5)	Hora de Llegada a destino	Formato: hh:mm
9	Recorrido	Alfanumérico(80)	Principales rutas del recorrido	Separadas por comas (,)
10	Días	Alfanumérico (20)	Días en que se realiza el servicio	Primeras 2 letras de cada día separados por comas (,). Fe significa Feriados
11	Lugar	Alfanumérico (30)	Lugar intermedio de pasada del servicio (Localidad).	
12	Depto. Lugar	Alfanumérico (30)	Departamento del lugar	
13	Hora	Alfanumérico (30)	Hora de pasada por lo descripto en el atributo Lugar	
14	F.Inicio	Fecha(8)	Fecha de inicio de validez del servicio	Formato: dd/mm/aa
15	F.Fin	Fecha(8)	Fecha de fin de validez del servicio	Formato: dd/mm/aa

Eventos en Uruguay (Provisto por nosotros)

Para poder lograr un *csv* completo de eventos en el país tomamos como ejemplo un *csv* (http://durazno.gub.uy/portal/images/stories/Datos_Abiertos/Datos_Abiertos_Durazno/eventos.csv) provisto por la Intendencia de Durazno , que contiene datos de eventos en Durazno. Tomamos este *csv* y le agregamos eventos de todo el país que obtuvimos en distintas páginas de eventos turísticos en el país.

Recurso

Recurso	Url	Nombre del archivo	Cantidad de Registros
Eventos	No aplica	eventos.csv	143

	A	B	C	D	E	F
1	EVENTO	FECHA	CANT_DIAS	LOCALIDAD	DEPARTAMENTO	CATEGORIA
2	Festival de Folclore de Durazno	17/10/2014	1	DURAZNO	DURAZNO	Festival
3	Festejos de Blanquillo	05/12/2014	2	MONTEVIDE	MONTEVIDEO	Festival
4	Festival Folclórico y Forestal - Grandes Criollas	12/12/2014	3	LIBERTAD	SAN JOSE	Festival
5	Fiesta de la Torta Frita y el mate	23/11/2015	1	SAN JOSE	SAN JOSE	Festival
6	Movida Tropical	24/01/2015	1	TRINIDAD	FLORES	Musica
7	Festival de Tango	10/01/2015	1	ANDRESITO	FLORES	Musica
8	Elección Reinas Llamadas y Carnaval	17/01/2015	1	YOUNG	RIO NEGRO	Desfile
9	42º Festival Nacional de Folclore	06/02/2015	3	PAYSANDU	PAYSANDU	Musica
10	Encuentro Gaucho	06/02/2015	3	CONSTANCIA	PAYSANDU	Encuentro
11	Desfile Inaugural del Carnaval	14/02/2015	1	EMP. 3 Y 26	PAYSANDU	Desfile
12	26 Edición de Las Primeras Llamadas del Interior	21/02/2015	1	EMPALME Q	PAYSANDU	Desfile
13	Entierro del Carnaval	01/03/2015	1	TERMAS DEL	PAYSANDU	Desfile
14	Motoencuentro Internacional	02/03/2015	1	CHAPICUY	PAYSANDU	Encuentro
15	Semana de la Juventud	03/03/2015	1	TERMAS DEL	SALTO	Festival
16	Feria del Libro	04/03/2015	1	SALTO	SALTO	Feria
17	Festival del Bailarín	05/03/2015	1	MONTEVIDE	MONTEVIDEO	Festival
18	Festival de Coros San Pedro de Durazno	06/03/2015	1	LIBERTAD	SAN JOSE	Festival
19	Semana de Durazno	07/03/2015	1	EMP. 1 Y 3	SAN JOSE	Festival

Figura 45 Datos del .CSV de Eventos de Datos Abiertos

La Figura 45 nos muestra como están publicados los datos de eventos en la plataforma de Datos Abiertos, en este caso corresponden al Ministerio de Turismo [9].

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción
1	Evento	Alfanumérico(200)	Nombre del evento.
2	Fecha	Fecha(8)	Formato: dd/mm/aaaa Fecha de inicio.
3	Cantidad de días	Numérico(2)	Cantidad de días de duración.
4	Localidad	Alfanumérico(30)	Nombre de la Localidad donde se realiza el evento.
5	Departamento	Alfanumérico(30)	Nombre del departamento donde se realiza el evento.
6	Categoría	Alfanumérico(15)	Tipo de evento.

Departamentos de Uruguay (Provisto por nosotros)

Para poder lograr un *csv* completo de departamentos del país tomamos como ejemplo una publicación provista por la Dirección General de Catastro la cual contiene un *csv* con departamentos (http://catastro.mef.gub.uy/innovaportal/file/10251/1/zip_datosabiertos_01_2015.zip). Agregamos datos generales y reales del departamento como ser población, superficie, densidad, ciudad capital, etc.

Recurso

Recurso	Url	Nombre del archivo	Cantidad de Registros
Departamentos	No aplica	departamentos.csv	19

	A	B	C	D	E	F
1	NOMBRE	REGION	POBLACION	SUPERFICIE	DENSIDAD	CAPITAL
2	CANELONES	CENTROSUR	364248	4536	80.3	Canelones
3	MALDONADO	ESTE	94314	4793	19.7	Maldonado
4	ROCHA	ESTE	66601	10551	6,3	Rocha
5	TREINTA Y TRES	NORESTE	46869	9529	4.9	Treinta y Tres
6	CERRO LARGO	NORESTE	78416	13648	5.7	Melo
7	RIVERA	NORTE	89475	9370	9.5	Rivera
8	ARTIGAS	NORTE	75100	11928	5.8	Artigas
9	SALTO	OESTE	118100	14.163	7.7	Salto
10	PAYSANDU	OESTE	110100	13922	7.5	Paysandú
11	RIO NEGRO	OESTE	51500	9,282	5,1	Fray Bentos
12	SORIANO	OESTE	79439	9.008	8.8	Mercedes
13	COLONIA	OESTE	112717	6106	18.5	Colonia del Sacramento
14	SAN JOSE	CENTROSUR	89893	4992	18.0	San José de Mayo
15	FLORES	CENTROSUR	24739	5144	4.8	Trinidad
16	FLORIDA	CENTRO	66514	10417	6.4	Florida
17	LAVALLEJA	ESTE	61486	10016	6.1	Minas
18	DURAZNO	CENTRO	55077	11643	4.7	Durazno
19	TACUAREMBO	CENTRO	83498	15438	5.4	Tacuarembó
20	MONTEVIDEO	SUR	1338600	530	2470	Montevideo

Figura 46 Datos del .CSV de Departamentos de Datos Abiertos

La Figura 46Figura 41 nos muestra como armamos una planilla *csv* para simular la extracción de datos referentes a departamentos.

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción
1	Nombre	Alfanumérico(20)	Nombre del departamento.
2	Región	Alfanumérico (10)	Región (Sur, Norte, etc.).
3	Población	Numérico(7)	Población total del departamento.
4	Superficie	Numérico(5)	Superficie total del departamento expresada en kms cuadrados.
5	Densidad	Alfanumérico(3)	Cantidad de habitantes por km cuadrado.
6	Capital	Alfanumérico(30)	Ciudad capital del departamento.

Fechas (Provisto por nosotros)

Nos generamos un *csv* que contiene todas las fechas comprendidas del año 2000 al 2016. Además de tener la fecha en particular, agregamos mes y año.

Recurso

Recurso	Url	Nombre del archivo	Cantidad de Registros
Fechas	No aplica	fechas.csv	839

	A	B	C
1	DIA	MES	ANIO
2	02/01/2013	Enero	2013
3	03/01/2013	Enero	2013
4	04/01/2013	Enero	2013
5	05/01/2013	Enero	2013
6	06/01/2013	Enero	2013
7	07/01/2013	Enero	2013
8	08/01/2013	Enero	2013
9	09/01/2013	Enero	2013
10	10/01/2013	Enero	2013
11	11/01/2013	Enero	2013
12	12/01/2013	Enero	2013
13	13/01/2013	Enero	2013
14	14/01/2013	Enero	2013
15	15/01/2013	Enero	2013
16	16/01/2013	Enero	2013
17	17/01/2013	Enero	2013
18	18/01/2013	Enero	2013

Figura 47 Datos del .CSV de Fechas

La Figura 47Figura 46Figura 41 nos muestra como armamos una planilla *csv* para simular la extracción fechas.

Metadatos de contenido

Nro. Columna	Nombre	Tipo de Dato	Descripción
1	Día	Fecha(10)	Formato dd/mm/aaaa.
2	Mes	Alfanumérico (15)	Mes de la fecha.
3	Año	Numérico(4)	Año de la fecha.

7.3 Diseño Conceptual del Web Warehouse

A continuación se muestra el diseño conceptual que planteamos para nuestro caso de estudio. Utilizamos como base teórica el modelo CMDM que describimos en el capítulo 2 aplicado a nuestra realidad para la cual queremos analizar datos generales de Turismo.

Dimensiones y Jerarquías

Las dimensiones definidas en el caso de estudio se ilustran en la Figura 48, donde las mismas además contienen sus jerarquías.

Por un lado tenemos a la dimensión Alojamientos, que representa a los distintos establecimientos donde un turista se puede alojar. Puede incluir hoteles, campings y albergues que son los que utilizamos en nuestro estudio. Los datos relevantes para nosotros respecto a esta dimensión serán un código que representa al alojamiento, código de ciudad y de departamento, el nombre, dirección y teléfono. Además, tenemos las jerarquías de Ciudad y de Departamento en la dimensión, logradas por los códigos que mencionamos.

Por otro lado, diseñamos la dimensión de Actividades, con los atributos código, nombre y tipo. La dimensión Actividades representa la realidad de la oferta turística en cuanto a eventos, festivales y actividades en general que puede disfrutar el turista en la ciudad o centro poblado donde se aloja. La jerarquía utilizada en Actividades, es su Tipo de actividad, que sería una categorización de las actividades por ejemplo festival, música, desfile, etc.

En cuanto a la referencia temporal, incluimos una dimensión Fecha que tiene los atributos fecha, mes y año. Esta dimensión fue provista por nosotros para poder vincular el resto de las dimensiones y armar un DW de utilidad. Las jerarquías que presentamos en dicha dimensión son, Mes y Año.

Finalmente tenemos la dimensión Transporte, que nos provee datos referentes a las líneas de ómnibus de servicio interdepartamental. Por lo que con dichos datos podemos conectar los diferentes centros poblados del país. Para cada línea de ómnibus, tenemos un código de servicio que la identifica, ciudad de origen y destino, hora de salida y de llegada. La jerarquía que incluimos en este caso es la Empresa que provee el servicio.

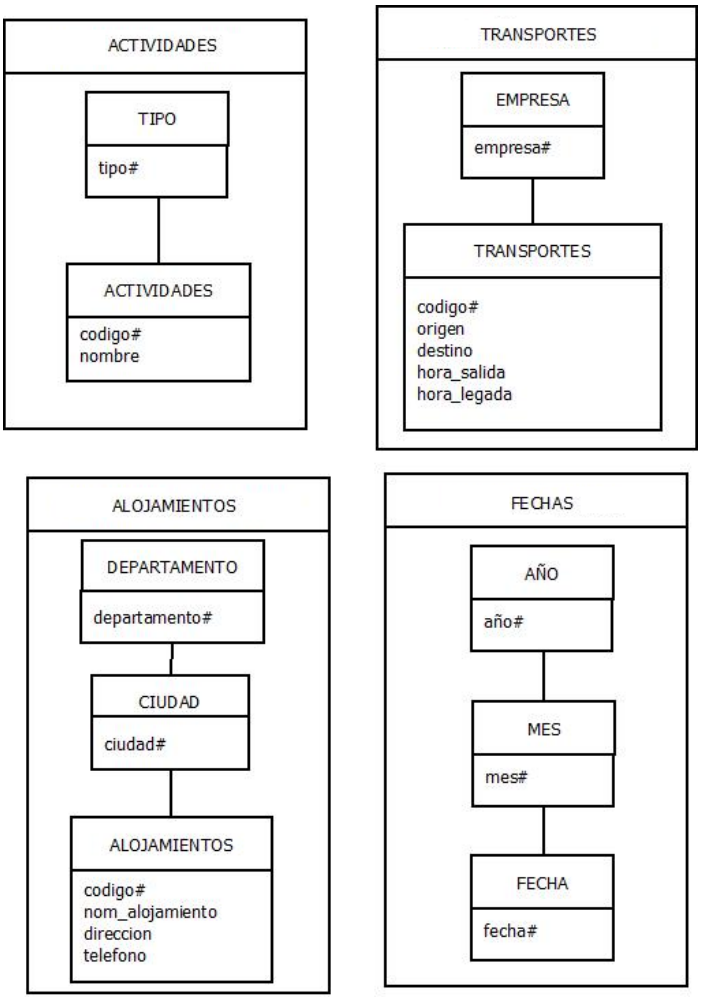


Figura 48 Dimensiones y Jerarquías

Los datos correspondientes a Alojamientos los vamos a obtener a partir de las Web Source Hoteles, Albergues y Campings. Los datos de Departamentos provienen de la Web Source Departamentos, que es un *csv* provisto por nosotros para facilitar el análisis. Con respecto a Transporte, la Web Source que nos proveerá de los datos es Líneas de Ómnibus Departamentales. Para Actividades, obtuvimos datos de la Web Source Eventos que nos brinda un *csv* como mencionamos anteriormente. Por último la dimensión Fecha contiene todos los días a partir del 2013, y como ya mencionamos es un *csv* provisto por nosotros para poder analizar la realidad con referencias temporales. Esta información la presentamos en la Tabla 5.

Tabla 5 Mapeo de dimensiones con las fuentes de datos.

DIMENSIÓN	WEB SOURCE	URL
Alojamientos	hoteles_web	http://apps.mintur.gub.uy/datosAbiertos/hoteles.php
	campings_web	http://apps.mintur.gub.uy/datosAbiertos/campings.php
	albergues_web	http://apps.mintur.gub.uy/datosAbiertos/albergues.php
	departamentos_web	Departamentos.csv
Actividades	eventos_web	Eventos.csv
Fechas	fechas_web	Fechas.csv
Transportes	transportes_web	http://catalogodatos.gub.uy/dataset/aa2405be-6dde-4482-8f5f-563817154282/resource/e62a3ef8-4e00-458b-bcbe-4b293ce0ebba/download/HorariosOmnibusVerano2015.zip

Relaciones dimensionales

En cuanto a las relaciones dimensionales, diseñamos Turismo que toma todas las dimensiones mencionadas anteriormente y para cada actividad y su fecha de comienzo obtenemos los diferentes establecimientos donde poder alojarnos y los servicios de transporte que nos llevan allí. Estos datos pueden ser analizados por localidad o departamento, por mes o por año, tipo de actividad, empresa de transporte, etc.

Las medidas que definimos para ser analizadas posteriormente con la herramienta OLAP son: cantidad de alojamientos, cantidad de servicios de transporte y cantidad de actividades. En la Figura 49 se puede visualizar la relación dimensional junto con las medidas.

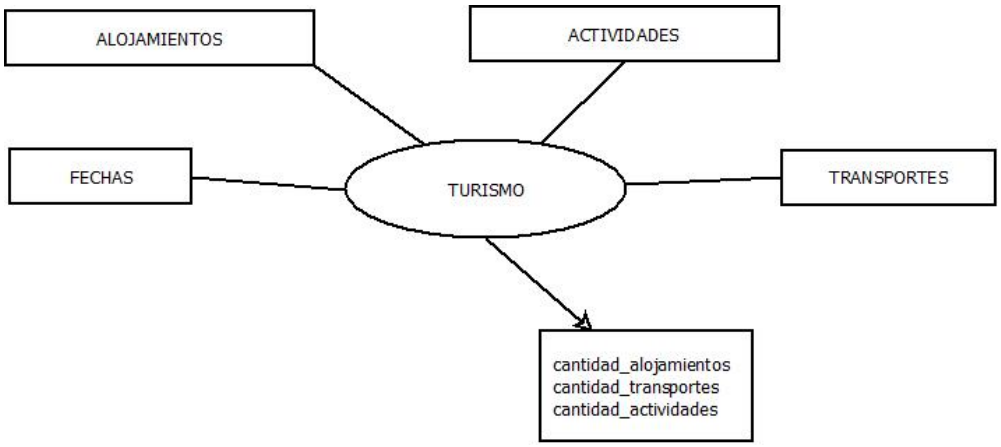


Figura 49 Relación dimensional

A continuación mostraremos paso por paso la ejecución del proceso de Configuración para arribar a este diseño de WW que explicamos.

7.4 Diseño Lógico del Web Warehouse

A continuación se muestra el diseño lógico para nuestro caso de estudio. Nuevamente utilizamos como base teórica el modelo CMDM que describimos en el capítulo 2 aplicado a nuestra realidad. Vamos a presentar un modelo estrella con una tabla de hechos y las cuatro tablas de dimensiones definidas con sus respectivas claves primarias y claves foráneas.

Modelo Estrella

Como ya mencionamos tenemos la tabla Turismo que es tabla de hechos y la misma referencia con claves foráneas al resto de las tablas de dimensiones, éstas son Actividades, Alojamientos, Transportes y Fechas que representan sus respectivas dimensiones como lo ilustramos en la Figura 50. Las medidas de la tabla de hechos no son materializadas ya que todas ellas se obtienen a partir de la cantidad de registros que cumplan la condición.

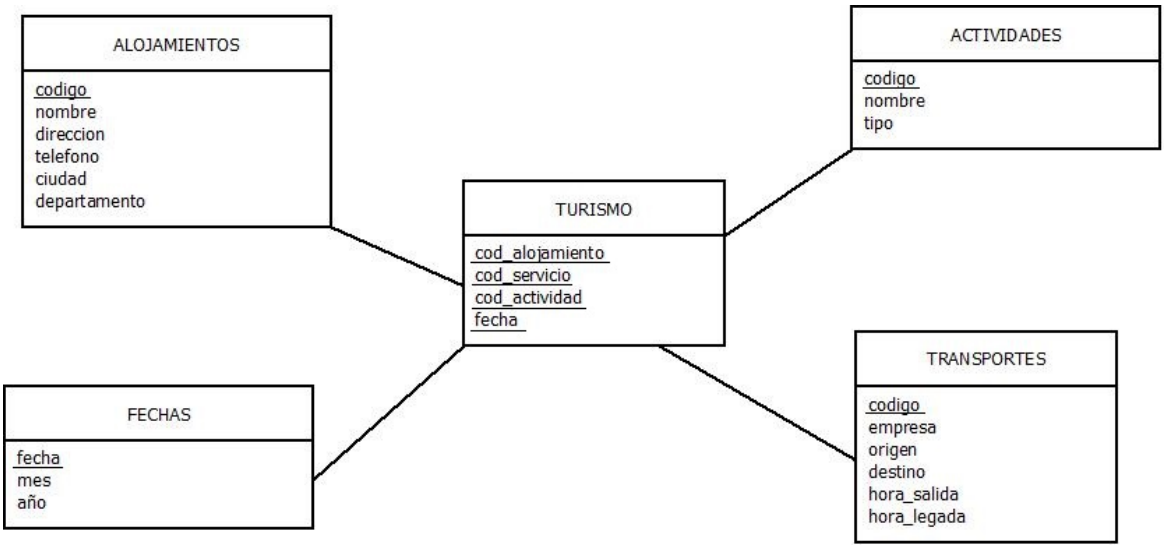


Figura 50 Diagrama del modelo estrella

7.5 Descripción del Proceso de Configuración

En esta sección vamos a ejecutar el Proceso de Configuracion completo presentado en la Figura 26 de la sección 5.2 para el caso de estudio que estamos analizando, mostrando además qué se va grabando en la base de metadatos en cada paso.

SELECT DOMAIN

Seleccionamos el dominio con el cuál queremos ejecutar la Configuración, en este caso Turismo. Esto lo ilustra la Figura 51, en la cual contamos con un combo de dominios precargados en el Sistema donde el usuario seleccionará el dominio con el cual desea trabajar, lo que permitirá luego elegir las Web Sources del dominio para esta configuración (ya dadas de alta en el sistema en alguna ejecución anterior) y/o ingresar nuevas Web Sources para el dominio seleccionado.

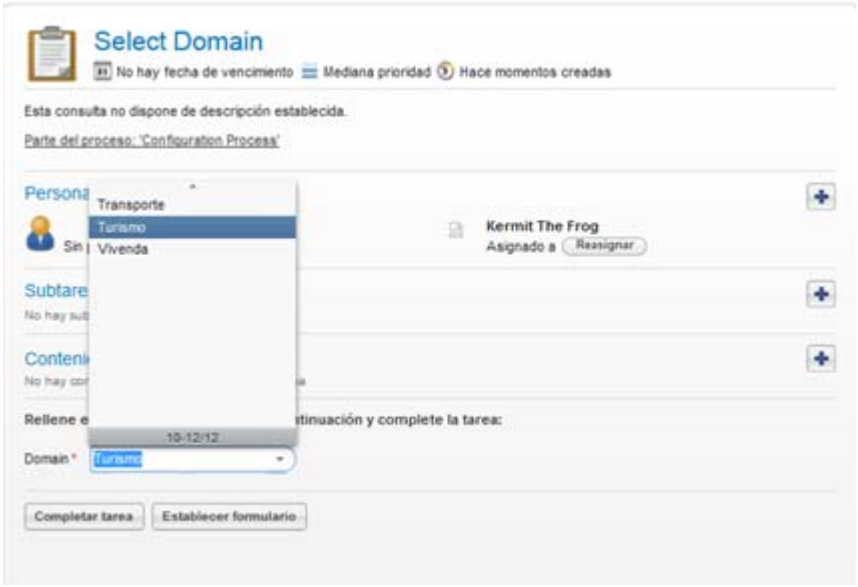


Figura 51 Actividad manual Select Domain

En la Figura 52 vemos que cuando se completa el formulario se inserta un registro en la tabla **configuracion** con un **id** autogenerado (3 en este caso) y un nombre de **dominio** (Turismo) que identifica al mismo.

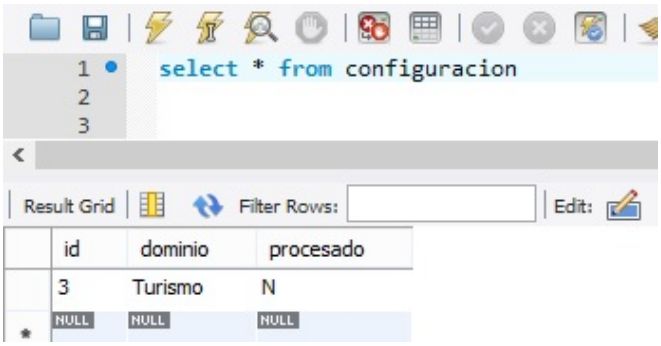


Figura 52 Contenido de tabla configuración al completar tarea

SELECT WEB SOURCE

En esta actividad, vamos a seleccionar Web Sources que ya existan y estén habilitadas para el dominio con el que estamos trabajando. Las Web Source son las fuentes web de donde vamos a extraer los datos, para nuestro ejemplo todos los recursos que explicamos anteriormente, para nuestro ejemplo ya teníamos dados de alta las fuentes de Albergues, Campings, Departamentos, Eventos, Fechas, Transportes y Hoteles. La Figura 53 nos muestra el formulario que nos permite realizar dicha actividad. En este formulario contamos con 5 combos que se cargan con todas las Web Sources del dominio seleccionado (realizamos una consulta en la base de metadatos sobre la tabla **dominio_web_source** y utilizamos como filtro de la consulta el campo **dominio**, Turismo en este caso que referencia al nombre del Dominio con el que estamos trabajando)

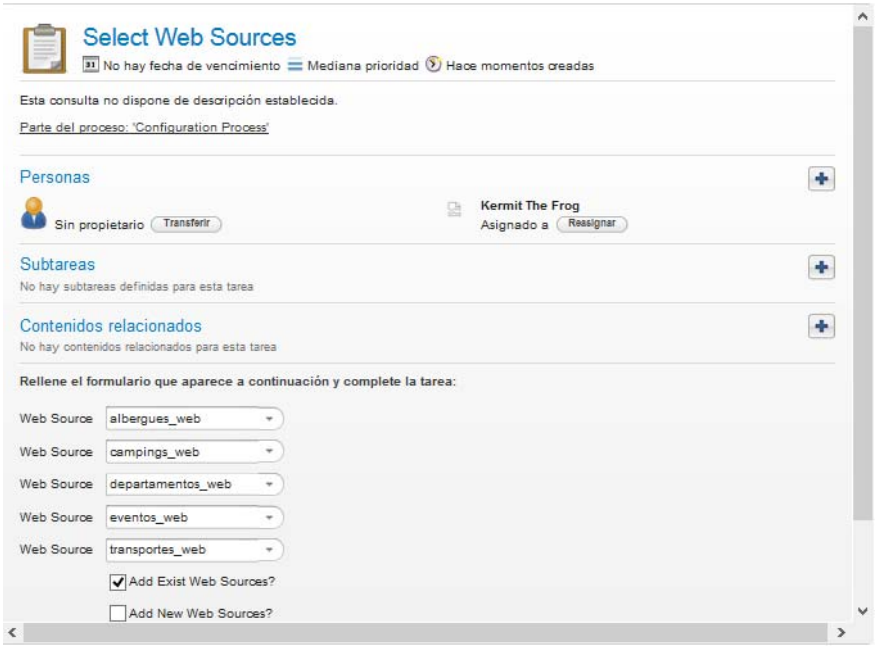


Figura 53 Actividad Manual Select Web Source

En la Figura 54 vemos que cuando se completa el formulario, se insertan registros en la tabla **config_web_sources**, cargando los campos **config** y **web_source** referenciando a las tablas **configuracion** y **web_source** respectivamente.

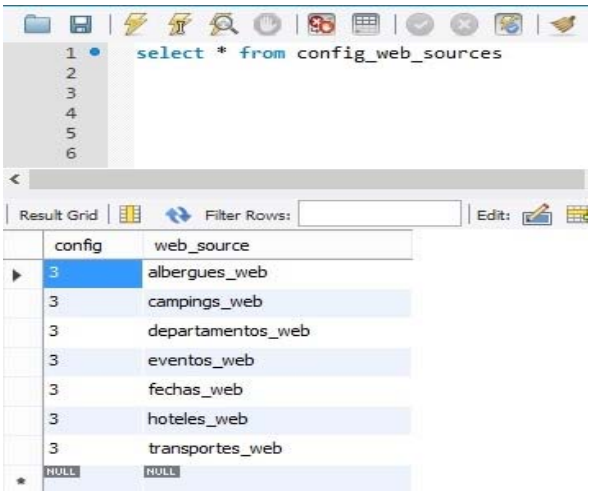


Figura 54 Contenido de tabla config_web_sources al completar tarea

CREATE WEB SOURCE

Esta tarea es muy similar a la anterior, pero presenta la diferencia de que las Web Sources a utilizar se deben dar de alta, es decir que no se usaron anteriormente. Esto nos permite en futuras ejecuciones de este proceso, tener ingresadas las Web Sources para determinado dominio, para nuestro ejemplo ya teníamos todas las fuentes de datos creadas por lo que no se selecciono la opción de agregar nuevas fuentes. Si quisiéramos crear una nueva fuente aparecería la pantalla de la Figura 55 donde se pueden ingresar los siguientes datos: nombre, url y formato de la Web Source. Estos formatos están precargados en el sistema.

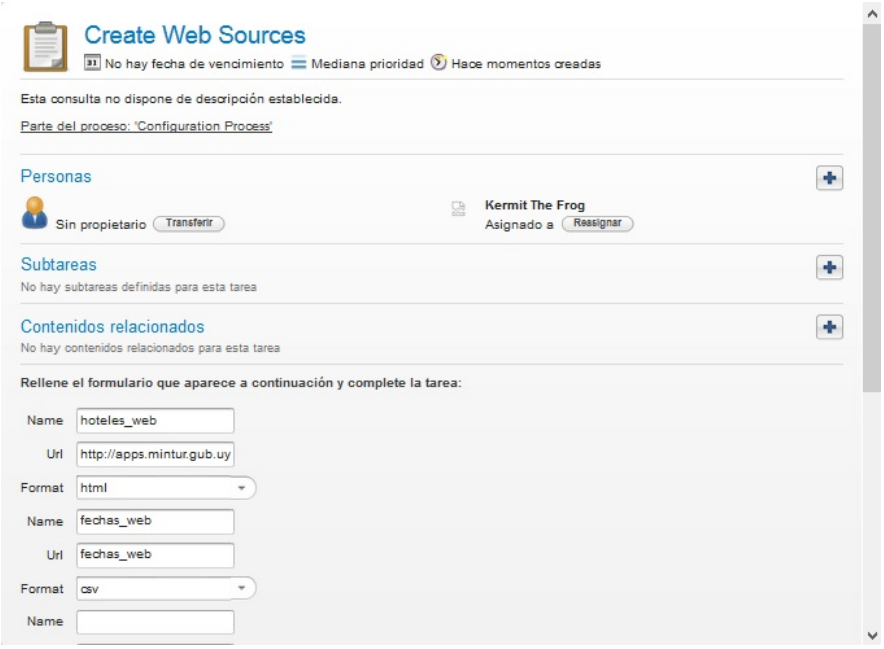


Figura 55 Actividad Manual Create Web Source

En la Figura 56 vemos que cuando se completa el formulario se insertan registros en las tablas **web_source**, **config_web_sources** y **dominio_web_sources**. En la primera, cargamos los campos **nombre**, **url y formato** con los datos que el usuario cargó en el formulario, este último es una referencia a la tabla **formato**. En la segunda tabla, sólo persistimos la relación entre la Configuración y la Web Source, insertando en los campos **config** y **web_source**, las referencias a las tablas **configuracion** y **web_source** respectivamente. En la tercera tabla, persistimos la relación entre el Dominio y la Web Source, insertando en los campos **dominio** y **web_source**, las referencias a las tablas **dominio** y **web_source** respectivamente. Otro impacto en la base de metadatos que existe en esta actividad pero es de funcionamiento interno, es la generación de una pequeña extracción de la fuente web para poder saber qué datos nos provee, como se explicó oportunamente en dicha actividad. Con esas columnas extraídas, se insertan registros en las tablas **source_esquema** y **atributos_source_esquema**, cargando en el campo **id** autogenerado en la primera tabla, y los campos **nombre** y **source_esquema** en la segunda, que indican el nombre del atributo y una referencia a la tabla **source_esquema**, respectivamente.

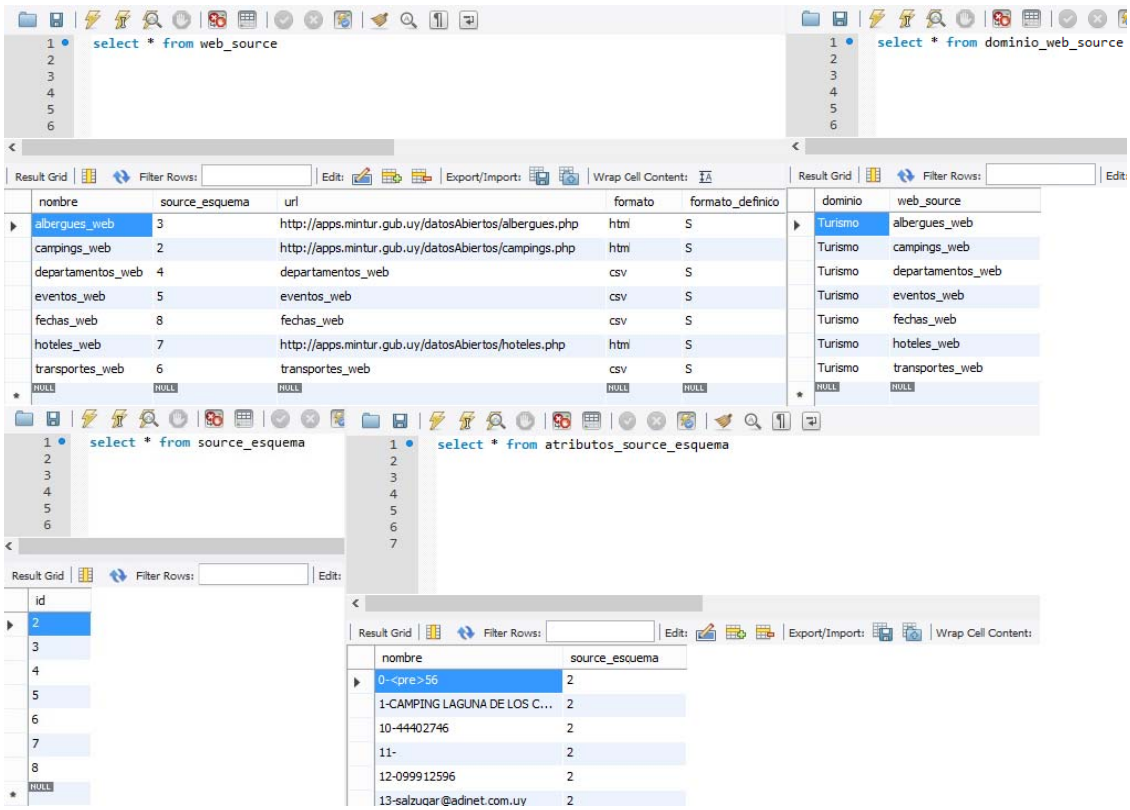


Figura 56 Contenido de tabla web_sources, dominio_web_source, source_esquema y atributos_source_esquema al completar tarea

DEFINE EXPECTED SCHEMA

La siguiente actividad determina de qué forma queremos crear o seleccionar el Expected Schema para cada Web Source.

Las opciones son: Crear el esquema manualmente, y para ello debe activar “CreateSchemaManually” que se explicará en siguientes tareas, o seleccionando un Expected Schema creado anteriormente o mediante un *json*, que nos permite de manera automática y con un formato determinado, generar toda la estructura del al Expected Schema obteniendo el mismo resultado que si lo hubiéramos creado manualmente.

La Figura 57 muestra la actividad de “Define Expected Schema” la cual cuenta con un texto informativo de la Web Source que se va a configurar, un campo donde subir el *json* que configura el Expected Schema para esa Web Source (esto en caso de seleccionar del combo “Subir JSON”), un combo donde se puede seleccionar los Expected Schemas configurados en ejecuciones anteriores o la opción “Subir JSON” y un checkbox que indica si queremos crearlo manualmente. Nosotros vamos a seleccionar esta última opción para poder mostrar las actividades manuales de creación del Expected Schema. Esta tarea no modifica nada en la base de metadatos.

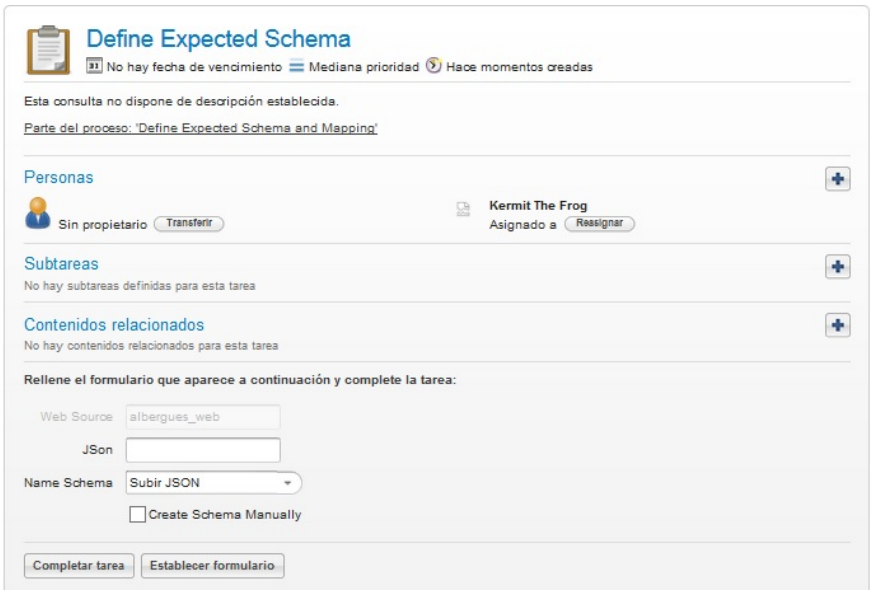


Figura 57 Actividad manual Define Expected Schema

CREATE EXPECTED SCHEMA AND ATRIBUTES

Como mencionamos en la tarea anterior, la presente actividad se ejecuta sólo si seleccionamos antes la manera manual de crear el Expected Schema. En la Figura 58 podemos observar cómo se ingresan los datos de cada "Expected Schema" (tenemos un campo para el nombre de la tabla, uno para el nombre y tipo de cada atributo). Una opción que presenta esta tarea pero no se visualiza en la imagen, es "Add More Atributes" y nos permite seguir agregando atributos para el Expected Schema que estamos creando, invocando a la actividad "Add more Atributes for Expected Schema" que se explicará más adelante.

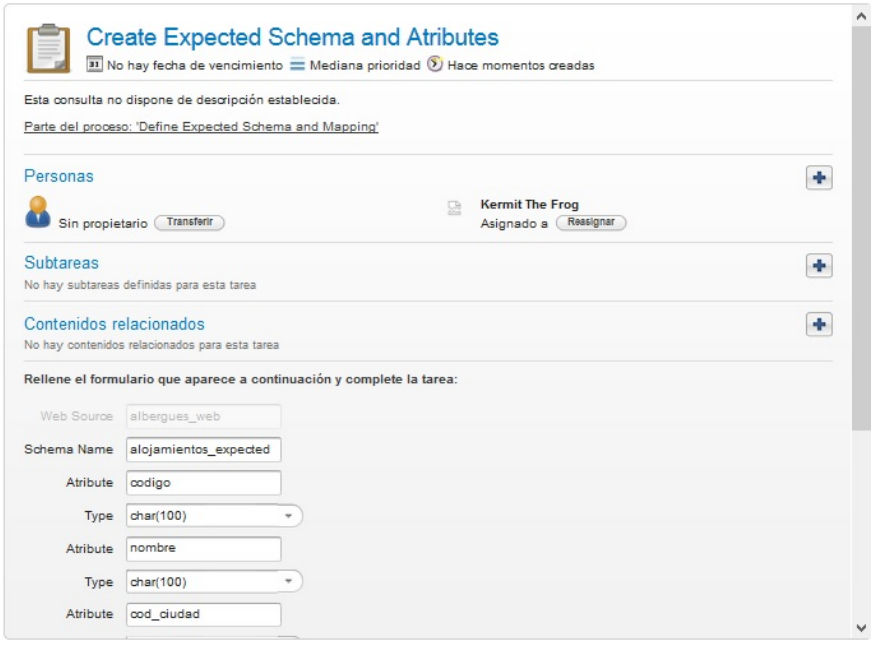


Figura 58 Actividad manual Create Expected Schema and Atributes

En la Figura 59 vemos que cuando se completa el formulario se insertan registros en las tablas **expected_esquema** y **atributos_expected_esquema**, en la primera insertaremos un registro que contiene el nombre de la entidad y con la lista de atributos obtenida para el mismo, insertamos en la segunda tabla los campos **nombre**, **expected_esquema** y **tipo**, que refieren al objeto atributo y corresponden con el nombre del mismo, el Expected Schema y el tipo de atributo a nivel de base de metadatos, respectivamente. Además se inserta un registro en la tabla **config_web_source_expected_esquema**, cuyos campos **config**, **web_source** y **expected esquema**, se cargarán con los identificadores de Configuración, Web Source y Expected Schema respectivamente. En este paso se completaron los datos de código, nombre, cod_ciudad, nom_dpto y dirección, que son los cinco combos que tenemos disponibles, el campo teléfono se completa en la actividad "Add more Atributes for Expected Schema" que se explicará más adelante.

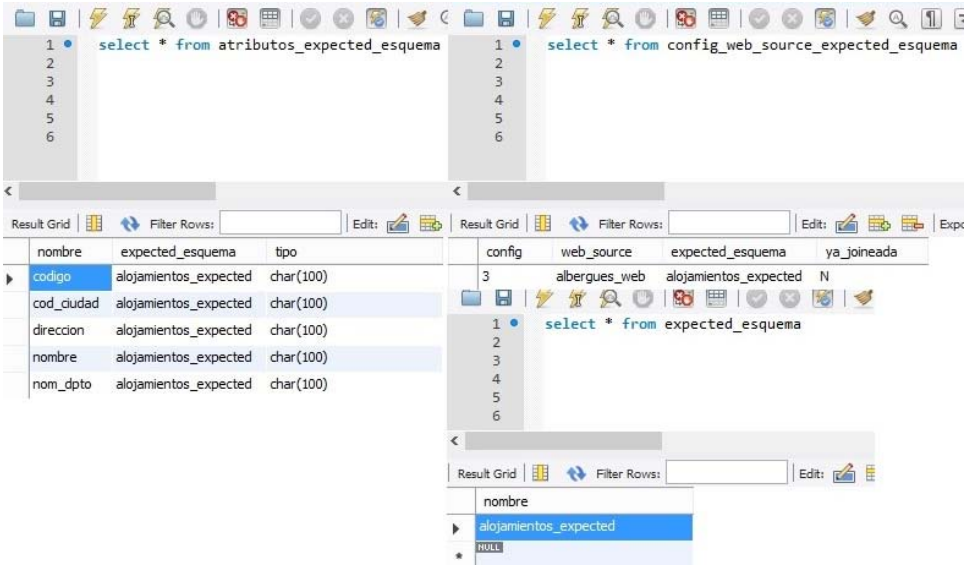


Figura 59 Contenido de tabla config_web_sources_expected_esquema, expected_esquema y atributos_expected_esquema al completar tarea

ADD MORE ATRIBUTES FOR EXPECTED SCHEMA

Esta tarea es similar a la anterior, con la diferencia que aquí ya se está trabajando con un Expected Schema en particular y lo que hacemos es agregarle atributos. Esta actividad se ejecuta sólo si seleccionamos antes la manera manual de crear el Expected Schema y luego seleccionamos el combo "Add More Atributes". En la Figura 60 podemos observar que los datos se ingresan de igual forma que en la tarea anterior con la diferencia que hay un campo informativo sobre que tabla se están agregando atributos. Esta tarea también presenta la opción de "Add More Atributes" y nos permite seguir agregando atributos para el Expected Schema que estamos creando, invocando nuevamente a la actividad "Add more Atributes for Expected Schema". Para el caso de estudio que estamos analizando se agrega el campo teléfono a alojamientos_expected.

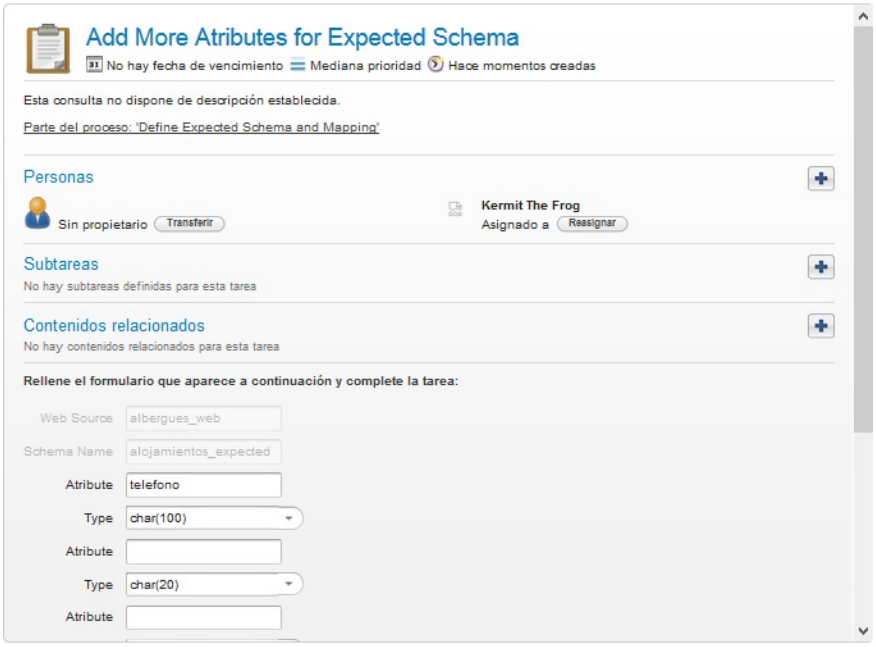


Figura 60 Actividad manual Add More Atributes for Expected Schema

En la Figura 61 vemos que cuando se completa el formulario debemos insertar los atributos referenciando al Expected Schema con el que estamos trabajando ("alojamientos_expected"). Para ello, insertamos un registro en la tabla **atributos_expected_esquema** para cada atributo ingresado indicando en los campos **nombre**, **expected_schema** y **tipo**, que refieren al nombre del atributo, una referencia a la tabla **expected_esquema** y el tipo de datos del atributo.

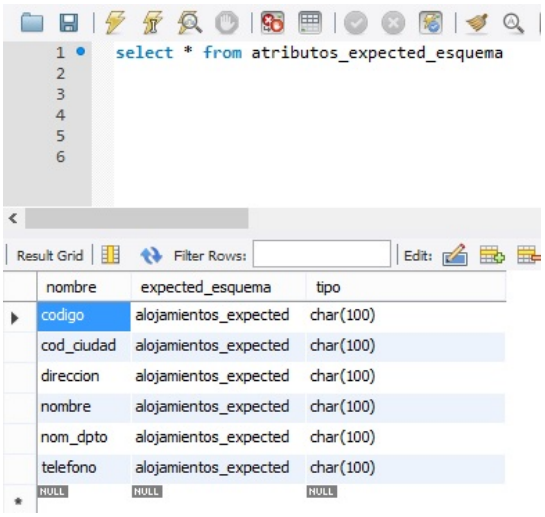


Figura 61 Contenido de tabla atributos_expected_esquema al completar tarea

MAPPING EXPECTED SCHEMA AND WEB SOURCE

Realizamos el mapeo entre los atributos del Expected Schema con las columnas de la Web Source que ya fueron extraídas como lo muestra la Figura 62. En caso de que el formato de la Web Source tenga una implementación de extracción ya establecida en el Sistema, podemos asociar a cada atributo del Expected Schema con el nombre de cada columna de la fuente web (para ofrecer las columnas de la fuente web, el sistema realiza una pequeña extracción del primer registro de datos para facilitarle al usuario la realización

del mapping). Si el formato que mencionamos no es soportado por el Sistema, el comportamiento de la actividad manual cambia un poco, en vez de ofrecer el listado de columnas disponibles por la Web Source, el usuario debe ingresar manualmente cada una de ellas para realizar el mapping. Estas columnas creadas por el usuario deben estar correctamente escritas, ya que el Sistema las va a utilizar a la hora de extraer los datos mediante el Data Service asociado. En el caso de que formato sea implementado por el sistema, se presenta un campo no editable correspondiente al atributo del Expected Schema y un combo para mapearlo con los datos de la Web Source (esto se puede realizar de a cinco campos). Esta actividad se va a ejecutar hasta que el Expected Schema no tenga atributos sin mapear.

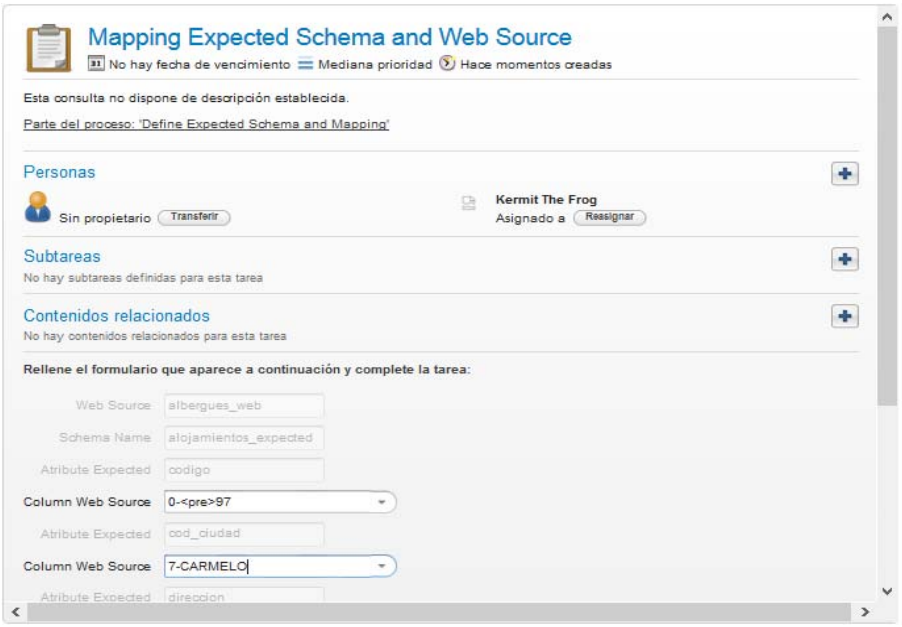
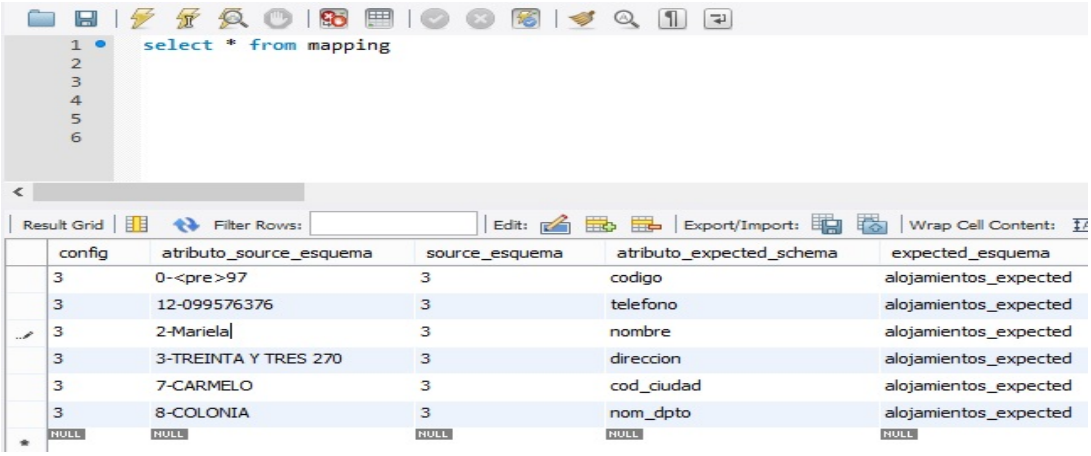


Figura 62 Mapping Expected Schema and Web Source

En la Figura 63 vemos que cuando se completa el formulario se inserta un registro por cada mapping entre atributos del Source Schema y el Expected Schema en la tabla **mapping**, cargando los campos **config**, **atributo_source_esquema**, **source_esquema**, **atributo_expected_schema** y **expected_esquema**, que indican referencias a: la Configuración, el atributos del Source Schema y del Expected Schema que estamos mapeamos, y el Expected Schema respectivamente.



config	atributo_source_esquema	source_esquema	atributo_expected_schema	expected_esquema
3	0-<pre>97	3	codigo	alojamientos_expected
3	12-099576376	3	telefono	alojamientos_expected
3	2-Mariela	3	nombre	alojamientos_expected
3	3-TREINTA Y TRES 270	3	direccion	alojamientos_expected
3	7-CARMELO	3	cod_ciudad	alojamientos_expected
3	8-COLONIA	3	nom_dpto	alojamientos_expected
NULL	NULL	NULL	NULL	NULL

Figura 63 Contenido de tabla mapping al completar tarea

UPLOAD DATA SERVICES

En el formulario que podemos ver en la Figura 64 ingresamos la url del Data Services que provee la extracción de los datos, como valor por defecto se despliega el del formato de la Web Source, en caso que el usuario desee cambiar el servicio lo podrá hacer modificando en este momento el valor del campo "Data Service Implementation".

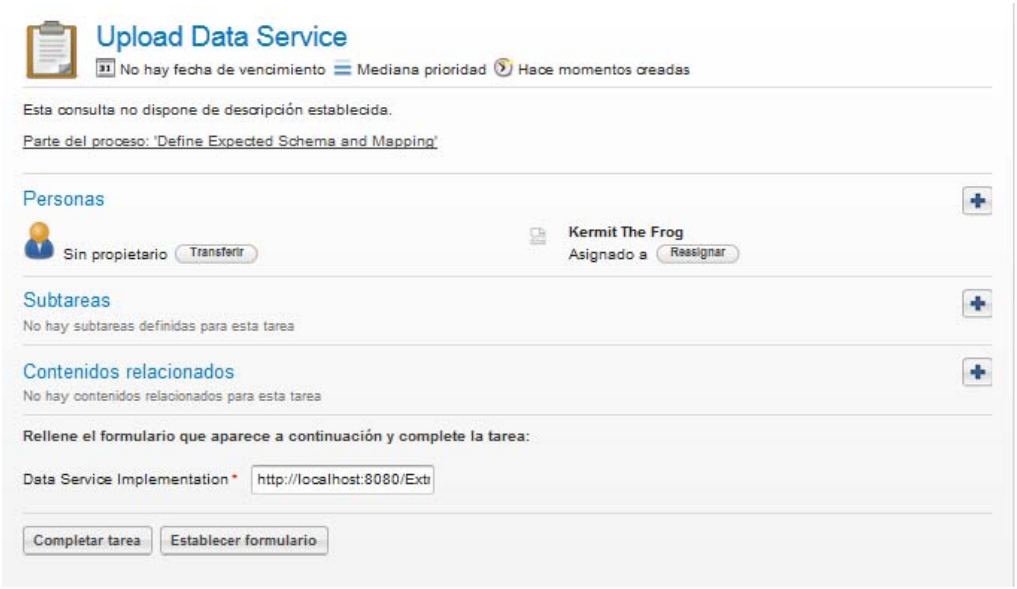


Figura 64 Upload Data Service

Este conjunto de actividades ("Define Expected Schema", "Mapping Expected Schema and Web Source" y "Upload Data Service") se repite para transportes_expected, fechas_expected, departamentos_expected y actividades_expected. Esto se puede ver en el proceso de configuración presentado en la Figura 26 de la sección 5.2 como el subproceso *"Define Expected Schema and Mapping"* (Ver 4 en Figura 26).

En la Figura 65 vemos que cuando se completa el formulario se inserta un registro en la tabla **config_web_source_dataservice** con los campos **config**, **web_source** y **data_service** que serán una referencia a la Configuración, a la Web Source y la url del servicio, respectivamente.

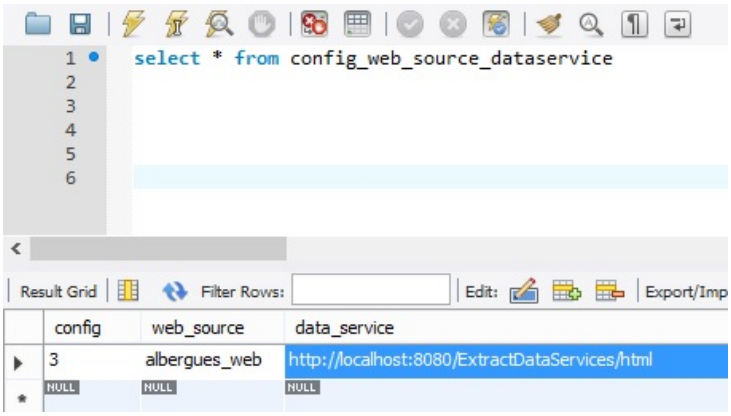


Figura 65 Contenido de tabla config_web_source_dataservice al completar tarea

DEFINE INTEGRATED SCHEMA

La siguiente actividad determina de qué forma queremos crear el Integrated Schema y la misma se visualiza en la Figura 66.

Las opciones son: crear el esquema manualmente, y para ello debe activar "CreateSchemaManually" que se explicará en siguientes tareas o mediante un *json*, que nos permite de manera automática y con un formato determinado, generar toda la estructura referente al Integrated Schema obteniendo el mismo resultado que si lo hubiéramos creado manualmente.

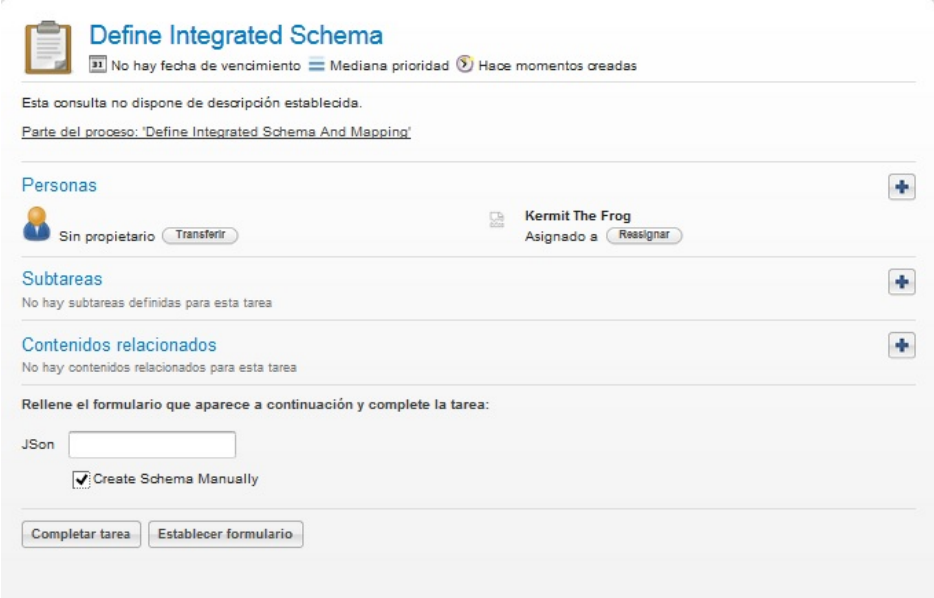


Figura 66 Define Integrated Schema

CREATE TABLE AND ATRIBUTES FOR INTEGRATED SCHEMA

Como mencionamos en la tarea anterior la presente actividad se ejecuta sólo si seleccionamos antes la manera manual de crear el Integrated Schema. En la Figura 67 podemos observar como ingresamos los datos de las tablas del "Integrated Schemas" (nombre de la tabla, nombre y tipo de cada atributo y cuál es la *primary key*). Una opción que presenta esta tarea pero no se visualiza en la imagen es *"Add More Atributes"* y nos permite seguir agregando atributos para la tabla del Integrated Schema que estamos creando, invocando a la actividad "Add more Atributes for Integrated Schema" que se explicará más adelante, así como también la opción de "Add More Tables" que permite continuar agregando tablas al Integrated Schema.

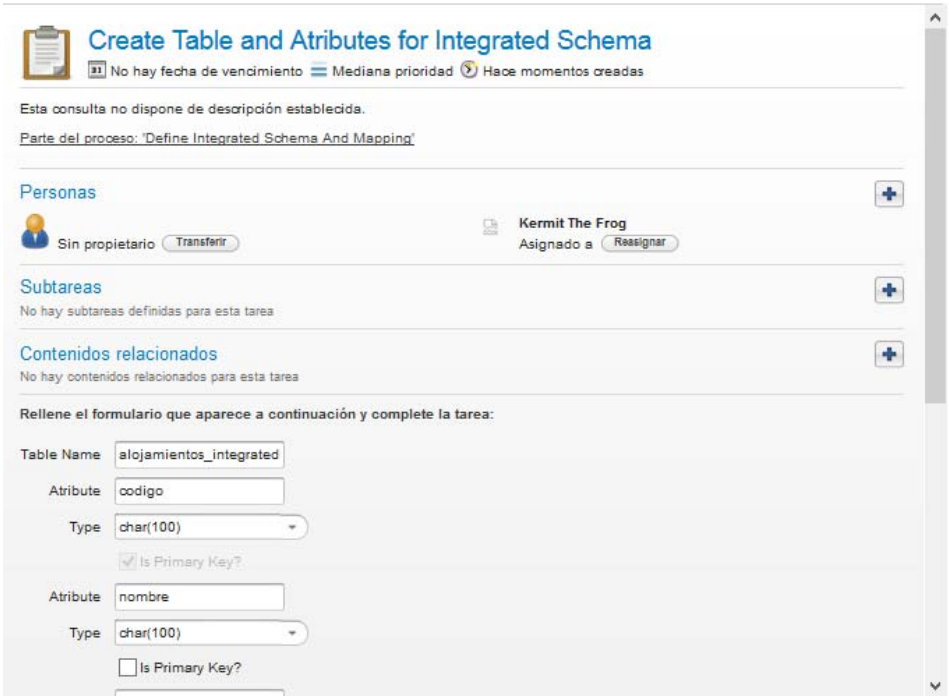


Figura 67 Create Table and Atributes for Integrated Schema

En la Figura 68 vemos que cuando se completa el formulario se insertan registros en las tablas **integrated_esquema** y **atributos_integrated_esquema**. Para la primera cargamos los campos **config** y **nombre**, que son una referencia a la Configuración y el nombre del Integrated Schema respectivamente. Para la segunda, insertamos un registro por cada atributo del Integrated Schema y los campos son **config**, **nombre**, **integrated_esquema, is_pk** y **tipo**, que son respectivamente: una referencia a la Configuración, el nombre del atributo, una referencia al Integrated Schema, si es clave primaria y el tipo del atributo ingresado respectivamente.

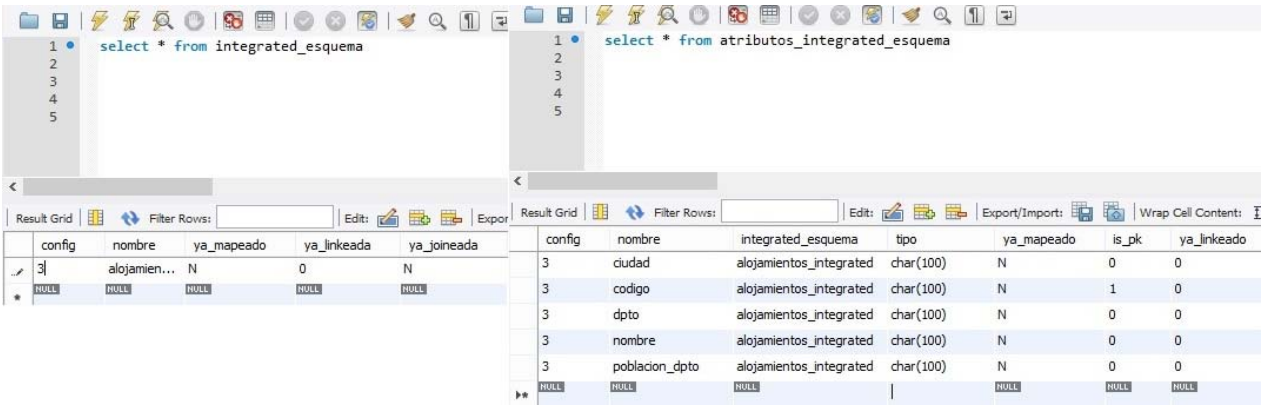


Figura 68 Contenido de tabla integrated_esquema y atributos_integrated_esquema al completar tarea

ADD MORE ATRIBUTES FOR INTEGRATED SCHEMA

Esta tarea es similar a la anterior, con la diferencia que aquí ya se está trabajando con una tabla del Integrated Schema en particular y lo que hacemos es agregarle atributos. Esta actividad se ejecuta sólo si seleccionamos antes la manera manual de crear el Integrated Schema y luego seleccionamos el combo “Add

More Atributes". En la Figura 69 podemos observar que los datos se ingresan de igual forma que en la tarea anterior con la diferencia que hay un campo informativo sobre que tabla se están agregando los atributos. Esta tarea también presenta la opción de "Add More Atributes" y nos permite seguir agregando atributos para la tabla del Integrated Schema que estamos creando, invocando nuevamente a la actividad "Add more Atributes for Integrated Schema". Además también está la opción de "Add More Tables" que permite continuar agregando tablas al Integrated Schema. Para el caso de estudio que estamos analizando se agrega el campo dirección y teléfono a alojamientos_integrated.

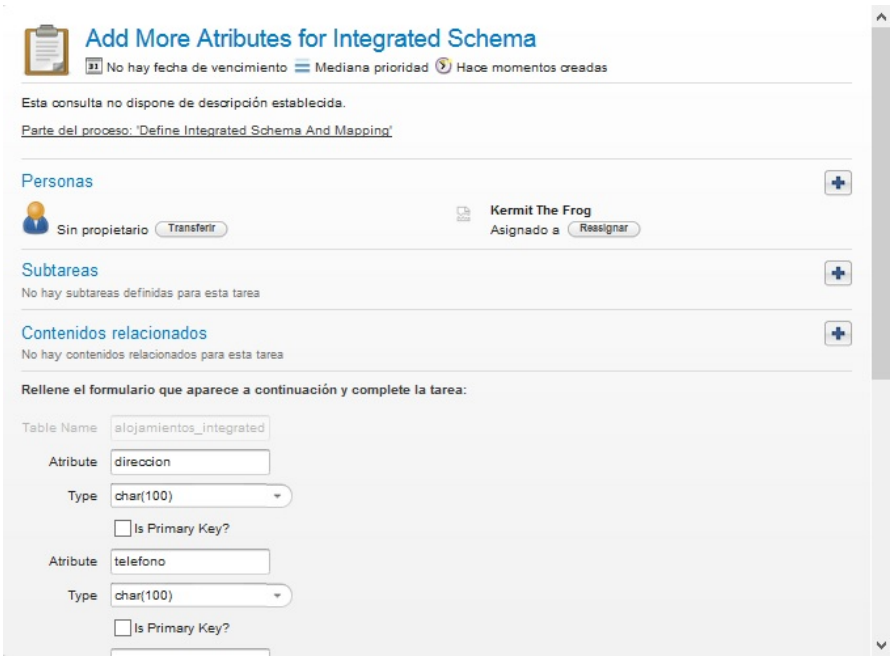
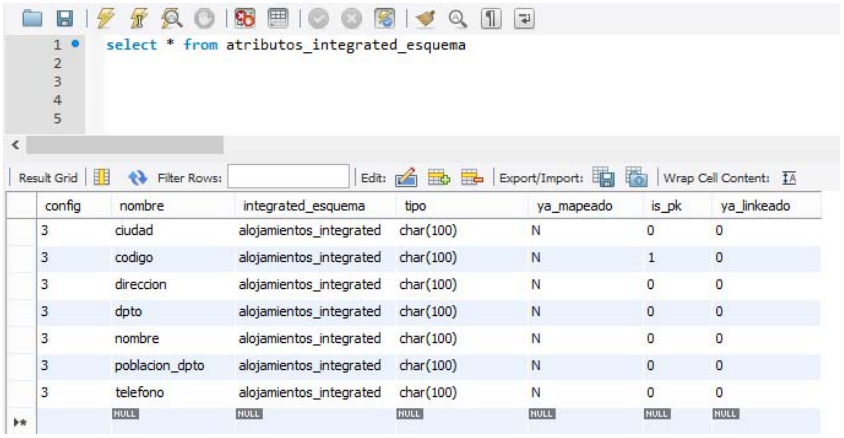


Figura 69 Actividad manual Add More Atributes for Integrated Schema

En la Figura 70 vemos que cuando se completa el formulario debemos insertar los atributos referenciando a la tabla del Integrated Schema con el que estamos trabajando ("alojamientos_integrated"). Para ello, insertamos un registro en la tabla **atributos_integrated_esquema** para cada atributo ingresado indicando en los campos **nombre**, **integrated_esquema, is_pk** y **tipo**, que refieren al nombre del atributo, una referencia a la tabla **expected_esquema**, si es clave primaria y el tipo de datos del atributo respectivamente.



```
select * from atributos_integrated_esquema
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

config	nombre	integrated_esquema	tipo	ya_mapeado	is_pk	ya_linkeado
3	ciudad	alojamientos_integrated	char(100)	N	0	0
3	codigo	alojamientos_integrated	char(100)	N	1	0
3	direccion	alojamientos_integrated	char(100)	N	0	0
3	dpto	alojamientos_integrated	char(100)	N	0	0
3	nombre	alojamientos_integrated	char(100)	N	0	0
3	poblacion_dpto	alojamientos_integrated	char(100)	N	0	0
3	telefono	alojamientos_integrated	char(100)	N	0	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 70 Contenido de tabla atributos_integrated_esquema al completar tarea

SHOW INTEGRATED SCHEMA

La siguiente actividad no requiere una acción del usuario, simplemente nos permite visualizar como ha quedado nuestro Integrated Schema como se puede observar en la Figura 71.

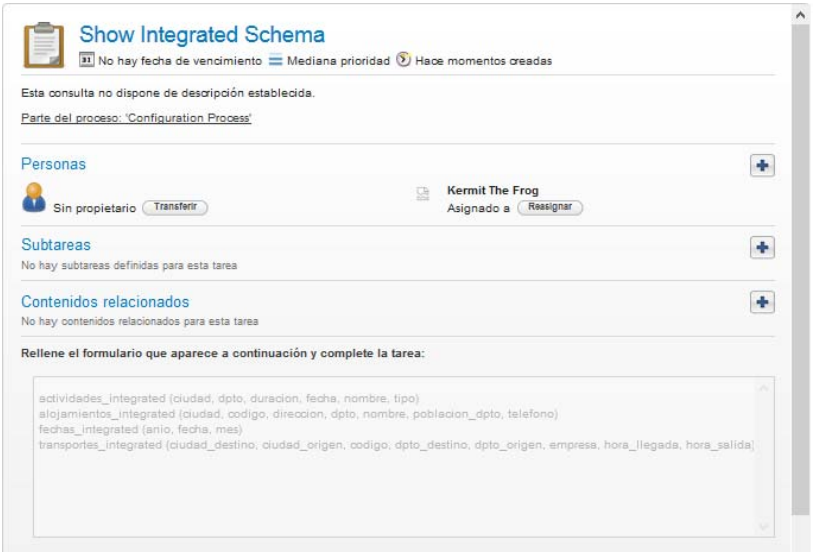


Figura 71 Show Integrated Schema

DEFINE FOREIGN KEYS INTEGRATED SCHEMA

En esta actividad vamos a determinar claves foráneas entre las tablas del Integrated Schema (Ver Figura 72).

Una vez completado el formulario, nos permite seguir agregando *foreign keys* activando "Define More Foreign Keys". Como muestra la imagen, en un combo se muestran los atributos de la tabla que estamos definiendo y en el otro las *primary keys* del resto de las tablas. Esta actividad se ejecutará para cada tabla del Integrated Schema, teniendo la posibilidad para cada una de ellas de no determinar ninguna *foreign key*.

Esta actividad se realiza para cada tabla del esquema integrado.

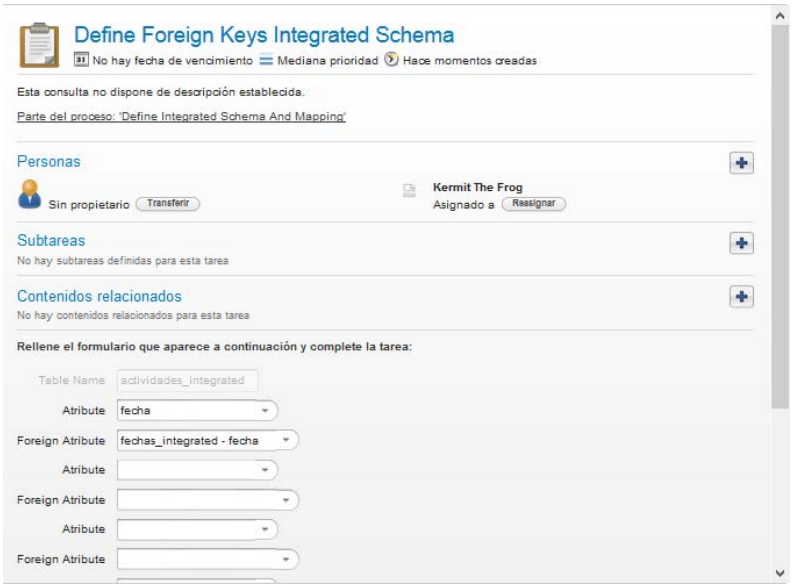


Figura 72 Define Foreign Keys for Integrated Schema

En la Figura 73 vemos que cuando se completa el formulario insertamos en la base de metadatos cada una de las *foreign keys* definidas. Para esto insertamos en la tabla **integrated_foreignkeys**, que tiene los campos **config**, **integrated_esquema1**, **integrated_esquema2**, **integrated_atributo1** e **integrated_atributo2**, que llevarán referencias a: la Configuración, el Integrated Schema del atributo que es clave foránea, el Integrated Schema del atributo clave referenciado, el atributo que es clave foránea y el atributo clave referenciado, respectivamente. Además, marcamos la tabla del Integrated Schema como ya procesada en el campo **ya_linkeada** en *true* de la tabla **integrated_esquema**.

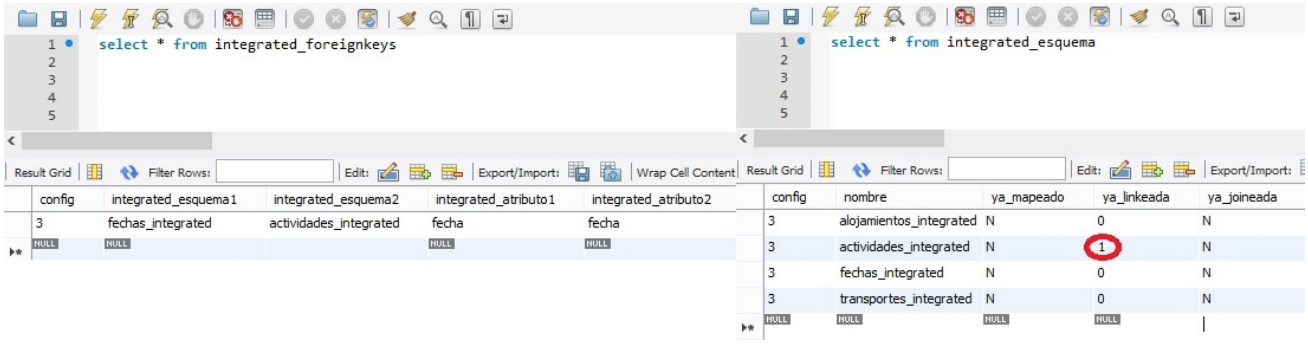


Figura 73 Contenido de tabla integrated_foreignkeys e integrated_esquema al completar tarea

MAPPING INTEGRATED SCHEMA AND EXPECTED SCHEMA

Como vemos en la Figura 74Figura 85 la actividad "Mapping Integrated Schema and Expected Schema" nos permite realizar el mapeo entre los atributos del Integrated Schema con los atributos del Expected Schema. Para realizar el mapping del Integrated Schema con el Expected Schema el Sistema nos ofrecerá la primera tabla del Integrated Schema sin mapear y un atributo en particular, y el usuario deberá asociar a éstos algún campo de algún Expected Schema (estos se presentan como combos indicando “tabla – atributo” de Expected Schema para facilitarle el mapeo al usuario). El Sistema nos indicará la realización de esta actividad hasta que no haya más atributos de tablas del Integrated Schema que aún no tengan mapeos.

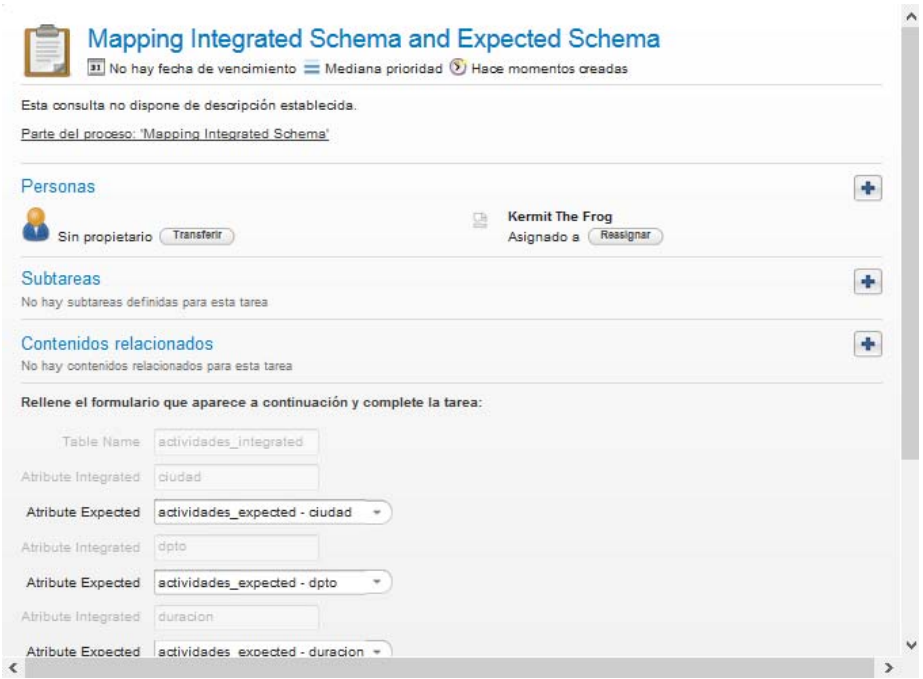


Figura 74 Mapping Integrated Schema and Expected Schema

En la Figura 75 vemos que cuando se completa el formulario se registra cada mapping realizado en el formulario y para ello insertamos por cada uno un registro en la tabla **mapping_integrated** y los valores que debemos indicar son **config**, **integrated_esquema**, **atributo_integrated_esquema**, **expected_esquema**, **atributo_expected_schema** y **web_source** que son: referencia a la Configuración, al Integrated Schema, al atributo del Integrated Schema, al Expected Schema, al atributo del Expected Schema y a la Web Source, respectivamente. Además, marcamos la tabla del Integrated Schema como ya mapeada marcando en *true* el campo **ya_mapeado** de la tabla **integrated_esquema**.

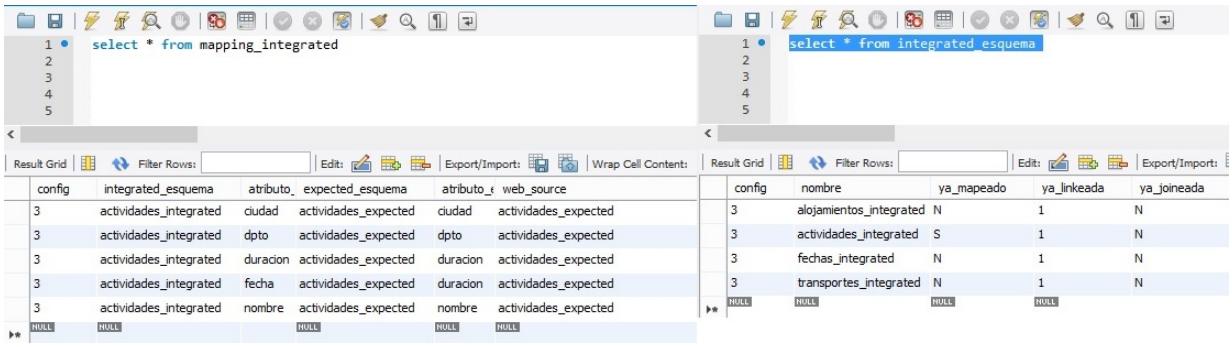


Figura 75 Contenido de tabla mapping_integrated e integrated_esquema al completar tarea

DEFINE JOINS INTEGRATED SCHEMA

En la tarea de "Mapping Integrated Schema and Expected Schema" que vemos en la Figura 76 se mapean distintas tablas del Expected Schema para conformar las tablas del esquema integrado. Esto implica que debamos indicarle al sistema como vamos a *joinear* esas tablas del Expected para cargar las tablas del Integrado. En este formulario debemos para cada tabla del Expected Schema de cada tabla del Integrated Schema, indicar cómo y con qué campo/s joinean entre ellas. Entonces seleccionamos el campo de la tabla de Expected que estamos iterando, en el combo "Expected Atribute", y luego seleccionamos un campo de las restantes tablas del Expected Schema en el combo "Expected Schema and Atribute" que ya no hayan sido procesados sus joins.

Observación: una vez que procesamos varios joins entre las tablas del Expected, es posible que nos encontremos con una tabla (que aún no procesamos) que ya fueron indicados sus joins por procesamientos de tablas anteriores. Para ellos existe el check "Done" que nos permite marcar la tabla como "ya procesada" sin indicar ningún join en la actividad.

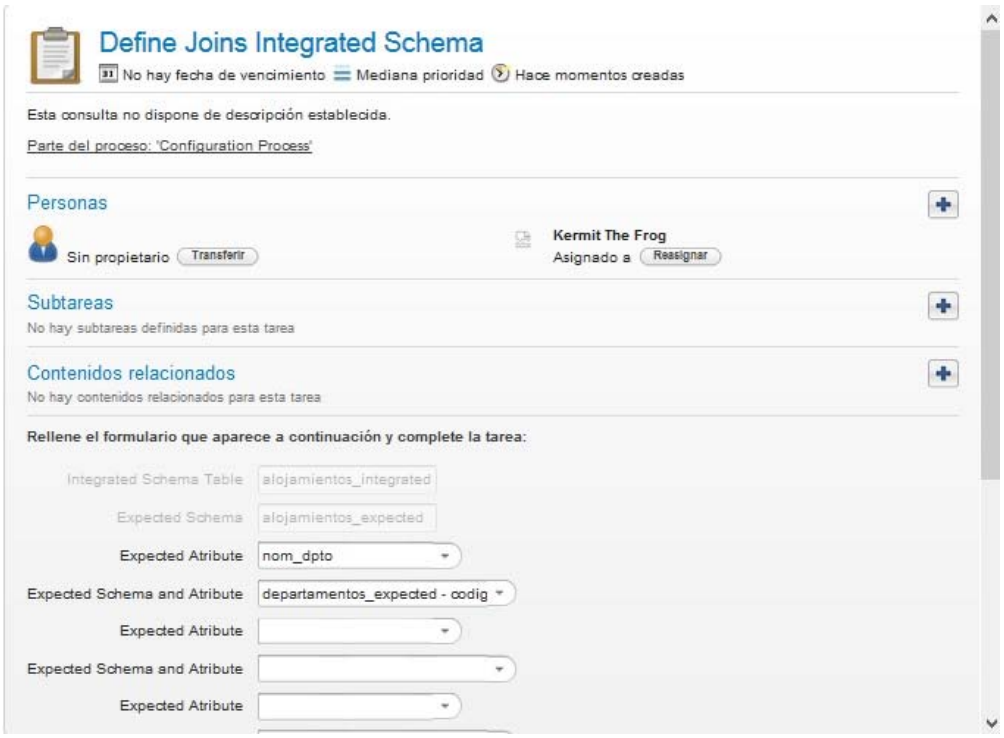


Figura 76 Define Joins Integrated Schema

En la Figura 77 vemos que cuando se completa el formulario se actualiza el Integrated Schema como ya procesado y lo haremos en el campo **ya_joineado** de la tabla **integrated_esquema**. Además, debemos insertar cada join definido entre atributos de Expected Schemas y lo haremos insertando en la tabla **joins_integrated** cargando en los campos **config**, **integrated_esquema**, **expected_esquema1**, **atributo_expected_esquema1**, **expected_esquema2** y **atributo_expected_esquema2** los valores siguientes: las claves de la Configuración, Integrated Schema, Expected Schema que procesamos, el atributo del mismo, el Expected Schema elegido para join, y el atributo del mismo también seleccionado.

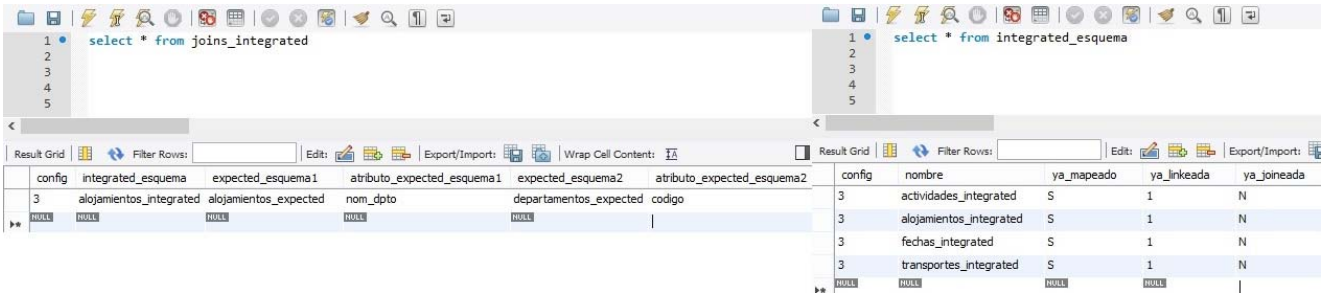


Figura 77 Contenido de tabla joins_integrated e integrated_esquema al completar tarea

DEFINE DATA WAREHOUSE SCHEMA

La siguiente actividad determina de qué forma queremos crear el DW Schema (Ver Figura 78).

Las opciones son: crear el esquema manualmente, y para ello debe activar “CreateSchemaManually” que se explicará en siguientes tareas mediante un *json*, que nos permite de manera automática y con un formato determinado, generar toda la estructura referente al DW Schema obteniendo el mismo resultado que si lo hubiéramos creado manualmente.

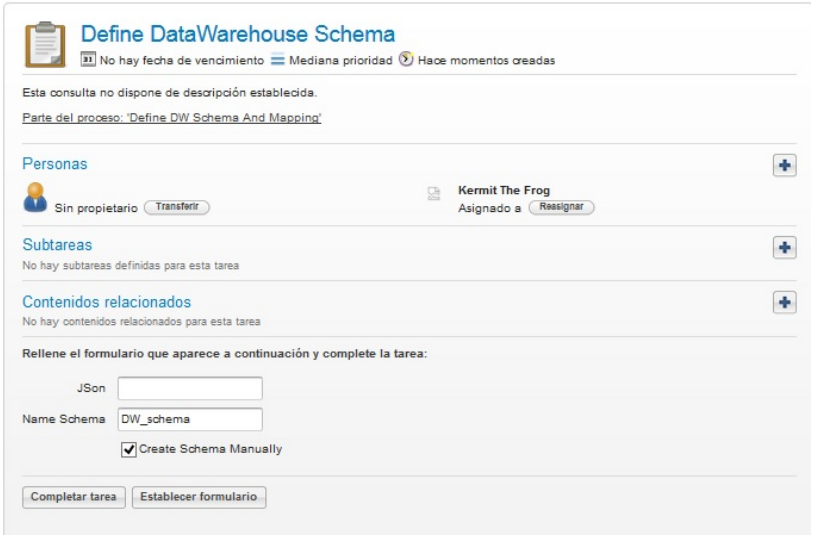


Figura 78 Define DW Schema

CREATE TABLE AND ATRIBUTES FOR DATA WAREHOUSE SCHEMA

Como mencionamos en la tarea anterior, la presente actividad se ejecuta sólo si seleccionamos antes la manera manual de crear el DW Schema. En la Figura 79 podemos ver como ingresamos los datos de las tablas del "DWSchemas" (indicando el nombre de la tabla junto con el nombre y tipo de cada atributo con su clave primaria y si es tabla de hechos o no).

Una opción que presenta esta tarea pero no se visualiza en la imagen, es "Add More Atributes" y nos permite seguir agregando atributos para la tabla del DW Schema que estamos creando, invocando a la actividad "Add more Atributes for Data Warehouse Schema" que se explicará más adelante. También tenemos la opción de seguir agregando tablas al DW Schema y volvemos a ejecutar la tarea "Create Table and Atributes for Data Warehouse Schema" seleccionando el check "Add More Tables".

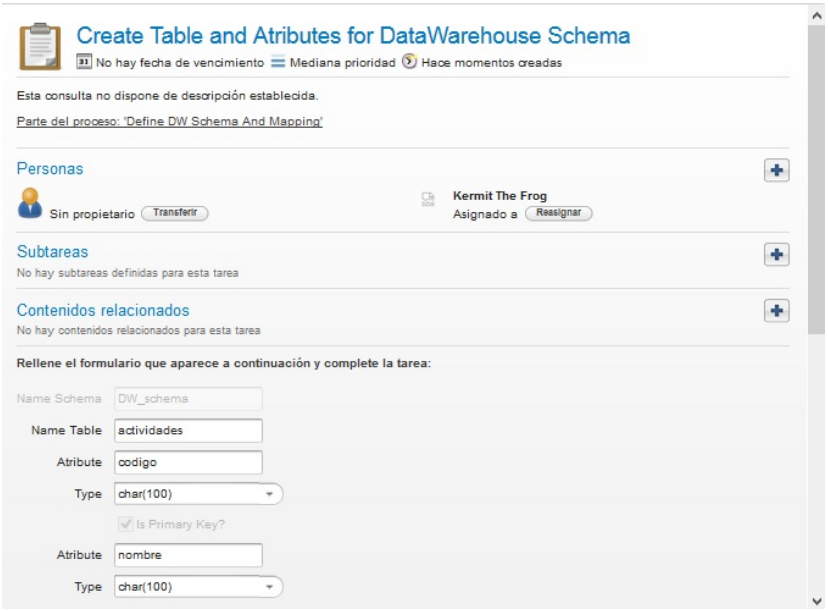


Figura 79 Create Table an Atributes for DW Schema

En la Figura 80 vemos que cuando se completa el formulario se insertan registros en las tablas **dw_esquema** y **atributos_dw_esquema**. Para la primera cargamos los campos **config, dw_esquema, nombre y es_fact**, que son una referencia a la Configuración, el nombre del DW Schema y el nombre de la tabla, y si es tabla de hechos respectivamente. Para la segunda, insertamos un registro por cada atributo de cada tabla del DW Schema y los campos son **config**, **nombre**, **dw_esquema** y **tipo**, que son respectivamente: una referencia a la Configuración, el nombre del atributo, el nombre de la tabla del DW Schema y el tipo del atributo ingresado.

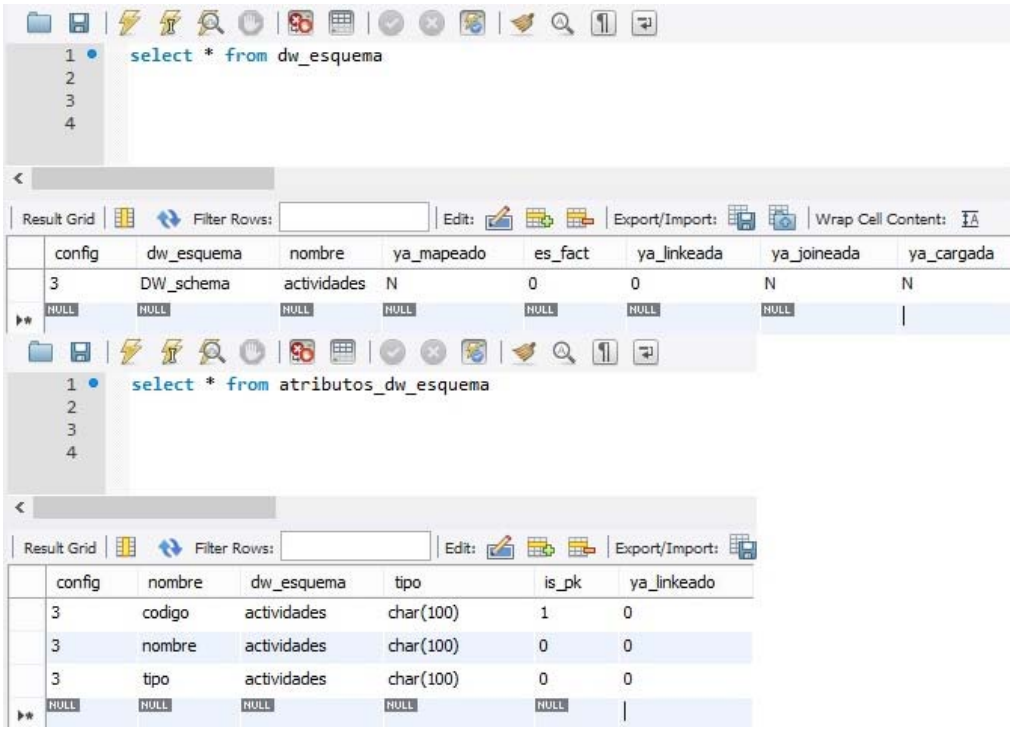


Figura 80 Contenido de tabla dw_esquema y atributos_dw_esqeuma al completar tarea

ADD MORE ATRIBUTES FOR DW SCHEMA

Esta tarea es similar a la anterior, con la diferencia que aquí ya se está trabajando con una tabla del DW Schema en particular y lo que hacemos es agregarle atributos. Esta actividad se ejecuta sólo si seleccionamos antes la manera manual de crear el DW Schema y luego seleccionamos el combo “Add More Atributes”. En la Figura 81 podemos observar que los datos se ingresan de igual forma que en la tarea anterior con la diferencia que hay un campo informativo sobre que tabla se están agregando atributos. Esta tarea también presenta la opción de “Add More Atributes” y nos permite seguir agregando atributos para la tabla del DW Schema que estamos creando, invocando nuevamente a la actividad “Add more Atributes for DW Schema”. Además también está la opción de “Add More Tables” que permite continuar agregando tablas al DW Schema. Para el caso de estudio que estamos analizando se agrega el campo departamento a alojamientos.

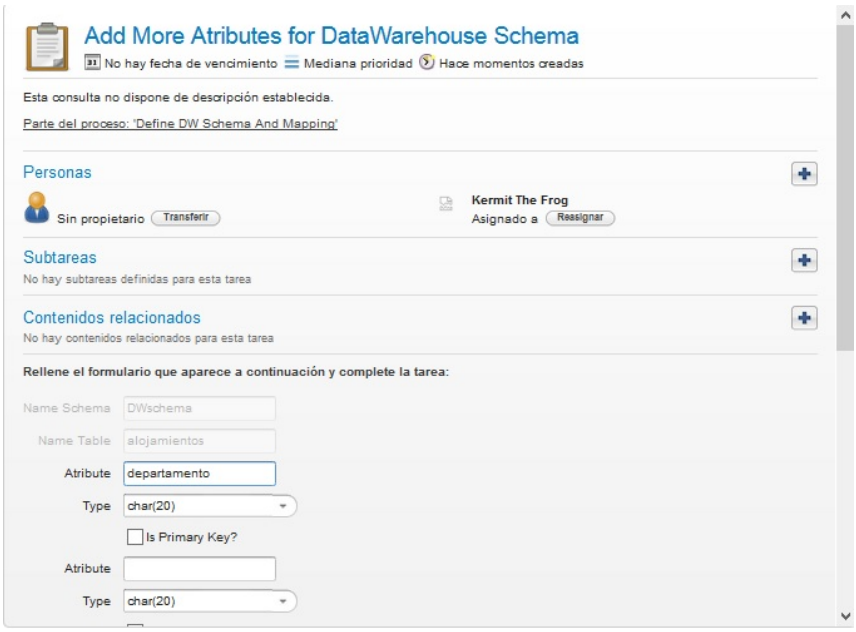


Figura 81 Add More Atributes for DW Schema

Cuando se completa el formulario, se deben crear en la base de metadatos los atributos ingresados en el mismo, como la tabla del DW Schema ya existe sólo necesitamos insertar los atributos referenciando a este último. Para ello, insertamos un registro en la tabla **atributos_dw_esquema** para cada atributo ingresado indicando en los campos **config**, **nombre**, **dw_esquema, is_pk** y **tipo**, que refieren a la Configuración, nombre del atributo, nombre de la tabla del DW Schema, si es clave primaria y el tipo de datos del atributo respectivamente.

SHOW DW SCHEMA

La siguiente actividad no requiere una acción del usuario, simplemente nos permite visualizar como ha quedado nuestro DW Schema. Esto se puede observar en la Figura 82.

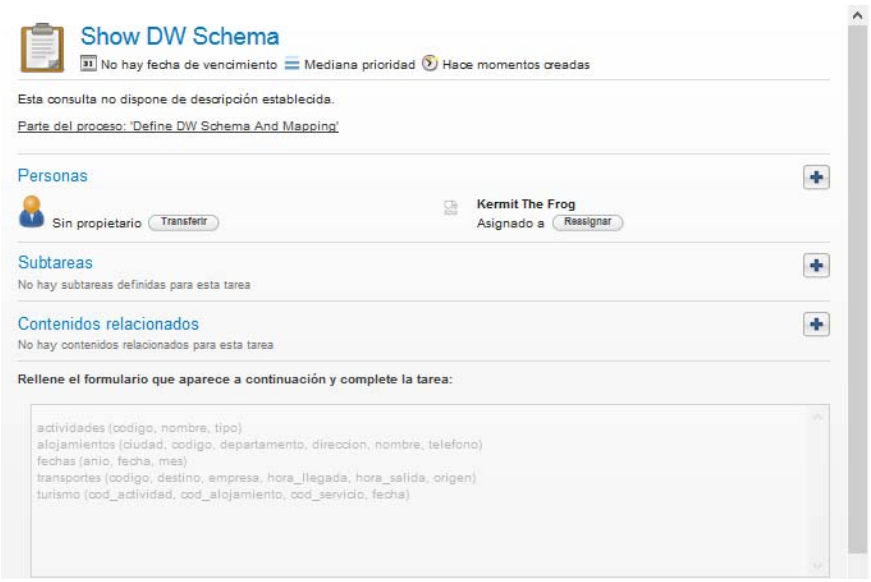


Figura 82 Show DW Schema

DEFINE FOREIGN KEYS OF FACT TABLES FOR DATA WAREHOUSE SCHEMA

En esta actividad vamos a determinar claves foráneas entre las tablas del DW Schema. Como muestra la Figura 83 en un combo se muestran los atributos de la tabla que estamos definiendo y en el otro las *primary keys* del resto de las tablas de dimensiones. Una vez completado el formulario, el mismo nos permite seguir agregando *foreign keys* activando "Define More Foreign Keys".

Esta actividad se ejecutará para cada tabla de hechos (*fact table*) del DW Schema.

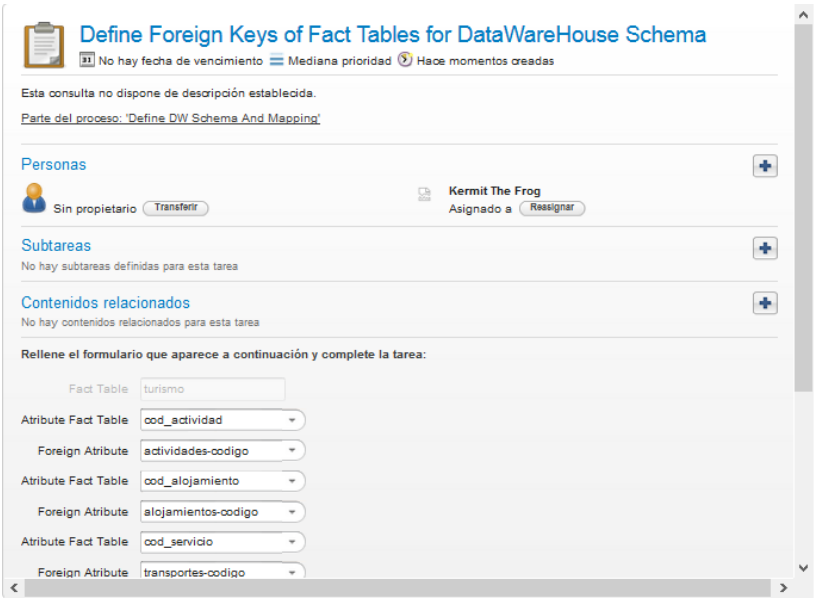
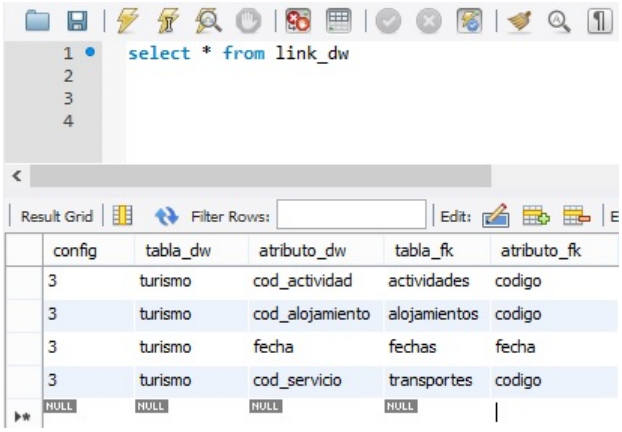


Figura 83 Define Foreign of Fact Tables for DW Schema

En la Figura 84 vemos que cuando se completa el formulario se inserta en la base de metadatos cada una de las *foreign keys* definidas. Para esto insertamos en la tabla **link_dw**, que tiene los campos **config**, **tabla_dw**, **atributo_dw**, **tabla_fk** y **atributo_fk**, que llevarán referencias a: la Configuración, el nombre de la tabla del DW Schema del atributo que es clave foránea, el atributo que es clave foránea, la tabla del DW Schema que contiene al atributo clave referenciado y finalmente el atributo referenciado, respectivamente. Además, marcamos la tabla del DW Schema como ya procesada en el campo **ya_linkeada** en *true* de la tabla **dw_esquema**.



config	tabla_dw	atributo_dw	tabla_fk	atributo_fk
3	turismo	cod_actividad	actividades	codigo
3	turismo	cod_alojamiento	alojamientos	codigo
3	turismo	fecha	fechas	fecha
3	turismo	cod_servicio	transportes	codigo
NULL	NULL	NULL	NULL	

Figura 84 Contenido de tabla link_dw al completar tarea

MAPPING DATA WAREHOUSE SCHEMA AND INTEGRATED SCHEMA

Realizamos el mapeo entre los atributos del DW Schema con los atributos del Integrated Schema. (Ver Figura 85)

Esta actividad se repite para el resto de las tablas del esquema de DW.

Para realizar el mapping del DW Schema con el Integrated Schema el Sistema nos ofrecerá la primera tabla del DW Schema sin mapear y un atributo en particular, y el usuario deberá asociar a éstos algún campo de alguna tabla del Integrated Schema (estos se presentan como combos indicando "tabla – atributo" del Integrated Schema para facilitarle el mapeo al usuario). El Sistema nos indicará la realización de esta actividad hasta que no haya más atributos de tablas del DW Schema que aún no tengan mapeos.

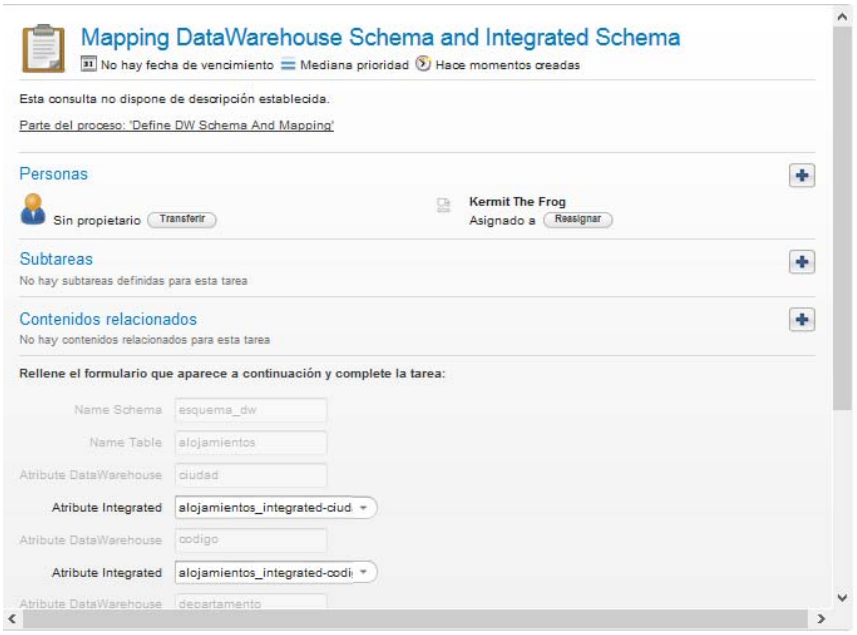
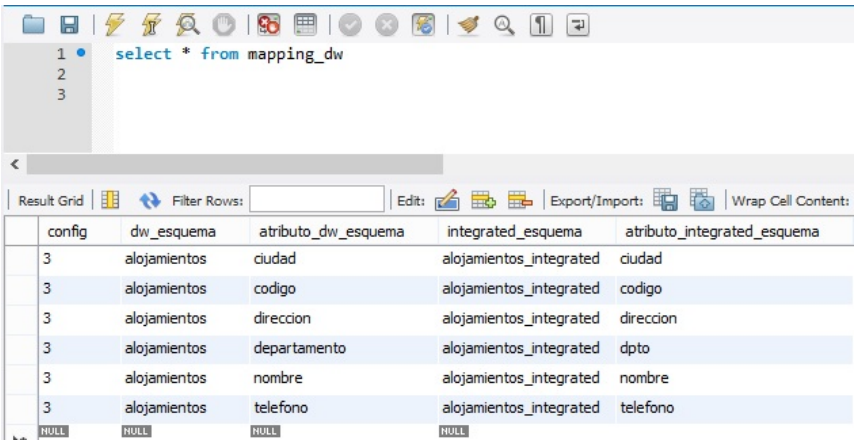


Figura 85 Mapping DW Schema and Integrated Schema

En la Figura 86 vemos que cuando se completa el formulario se inserta por cada uno un registro en la tabla **mapping_dw** y los valores que debemos indicar con **config**, **dw_esquema**, **atributo_dw_esquema**, **integrated_esquema** y **atributo_integrated_esquema** que son: referencia a la Configuración, a la tabla del DW Schema, al atributo de la tabla del DW Schema, a la tabla del Integrated Schema y al atributo de la tabla del Integrated Schema, respectivamente. Además, marcamos la tabla del DW Schema como ya mapeada marcando en *true* el campo **ya_mapeado** de la tabla **dw_esquema** filtrando por el campo nombre para obtener la tabla correspondiente.



```
1  select * from mapping_dw
2
3
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

config	dw_esquema	atributo_dw_esquema	integrated_esquema	atributo_integrated_esquema
3	alojamientos	ciudad	alojamientos_integrated	ciudad
3	alojamientos	codigo	alojamientos_integrated	codigo
3	alojamientos	direccion	alojamientos_integrated	direccion
3	alojamientos	departamento	alojamientos_integrated	dpto
3	alojamientos	nombre	alojamientos_integrated	nombre
3	alojamientos	telefono	alojamientos_integrated	telefono
NULL	NULL	NULL	NULL	

Figura 86 Contenido de tabla mapping_dw al completar tarea

DEFINE JOINS DW SCHEMA

En la tarea de “Mapping Data Warehouse Schema and Integrated Schema” se mapean distintas tablas del Integrated Schema para conformar las tablas del esquema de DW. Esto implica que debamos indicarle al sistema como vamos a *joinear* esas tablas del Integrated para cargar las tablas del DW Schema.

En el formulario de la Figura 87 debemos para cada tabla del Integrated Schema de cada tabla del DWSchema, indicar cómo y con qué campo/s joinean entre ellas. Entonces seleccionamos el campo de la tabla de Integrated que estamos iterando, en el combo “Integrated Atribute”, y luego seleccionamos un campo de las restantes tablas del Integrated Schema en el combo “Integrated Schema and Atribute” que ya no hayan sido procesados sus joins.

Observación: una vez que procesamos varios joins entre las tablas del Integrated, es posible que nos encontremos con una tabla (que aún no procesamos) que ya fueron indicados sus joins por procesamientos de tablas anteriores. Para ellos existe el check “Done” que nos permite marcar la tabla como “ya procesada” sin indicar ningún join en la actividad.

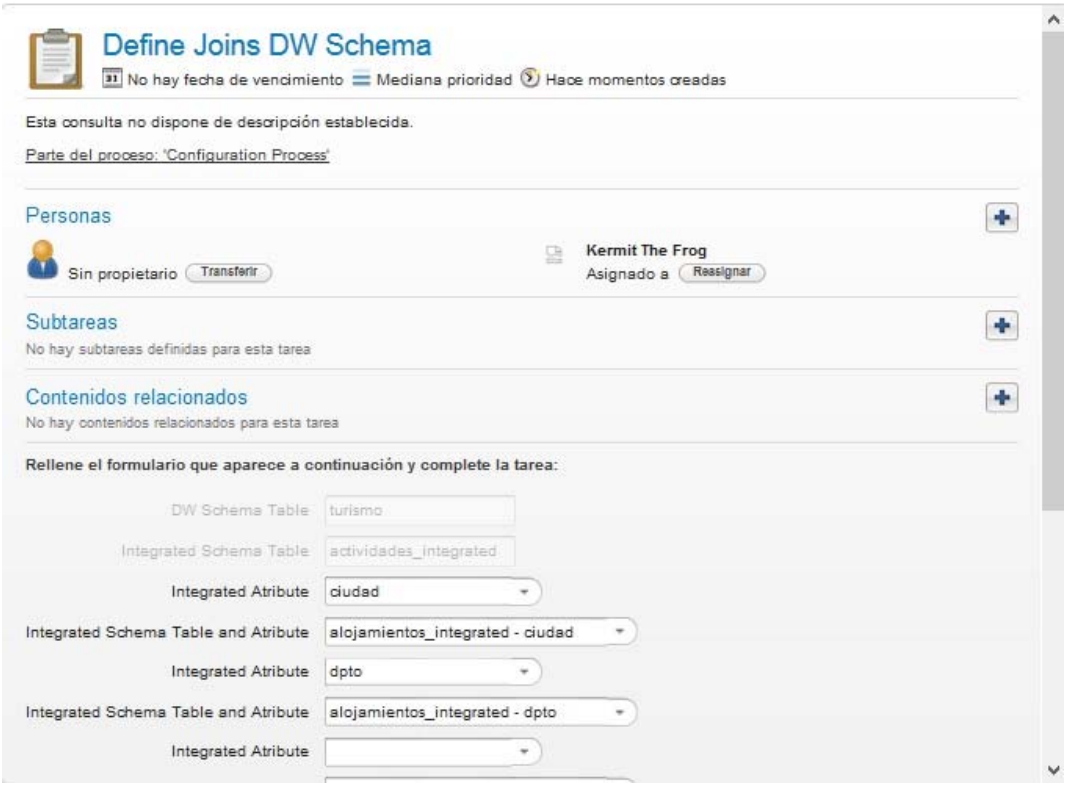
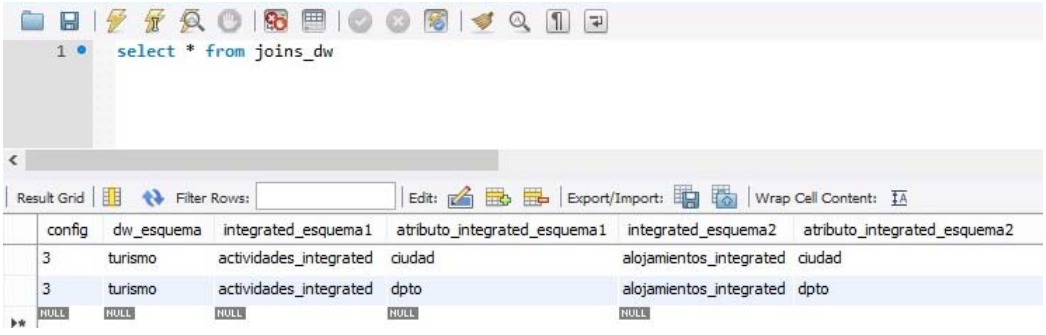


Figura 87 Define Joins DW Schema

En la Figura 88 vemos que cuando se completa el formulario se debe marcar la tabla del DW Schema como ya procesada y lo haremos con el campo **ya_joineada** de la tabla **dw_esquema**. Además, debemos insertar cada join definido entre los atributos de las tablas del Integrated Schema y lo haremos insertando en la tabla **joins_dw** cargando en los campos **config**, **dw_esquema**, **integrated_esquema1**, **atributo_integrated_esquema1**, **integrated_esquema2**, **atributo_integrated_esquema2** los valores siguientes: las claves de la Configuración, la tabla del DW Schema y las tablas y atributos del Integrated Schema involucradas en el join.



config	dw_esquema	integrated_esquema1	atributo_integrated_esquema1	integrated_esquema2	atributo_integrated_esquema2
3	turismo	actividades_integrated	ciudad	alojamientos_integrated	ciudad
3	turismo	actividades_integrated	dpto	alojamientos_integrated	dpto
NULL	NULL	NULL	NULL	NULL	

Figura 88 Contenido de tabla joins_dw al completar tarea

Finalizada esta actividad termina el proceso de Configuración y ya disponemos de los metadatos para realizar la construcción y carga del DW.

A continuación mostraremos la ejecución del proceso de Carga paso por paso nuestro caso de estudio.

7.6 Descripción de Proceso de Carga

SELECT CONFIGURATION

Todas las ejecuciones del proceso de Configuración, se guardan en la base de metadatos con un identificador. Como ya dijimos en la tabla “Configuration”.

Apenas comienza el proceso de Carga debemos seleccionar cuál de ellas vamos a utilizar, para ello utilizamos el formulario descrito en la Figura 89.

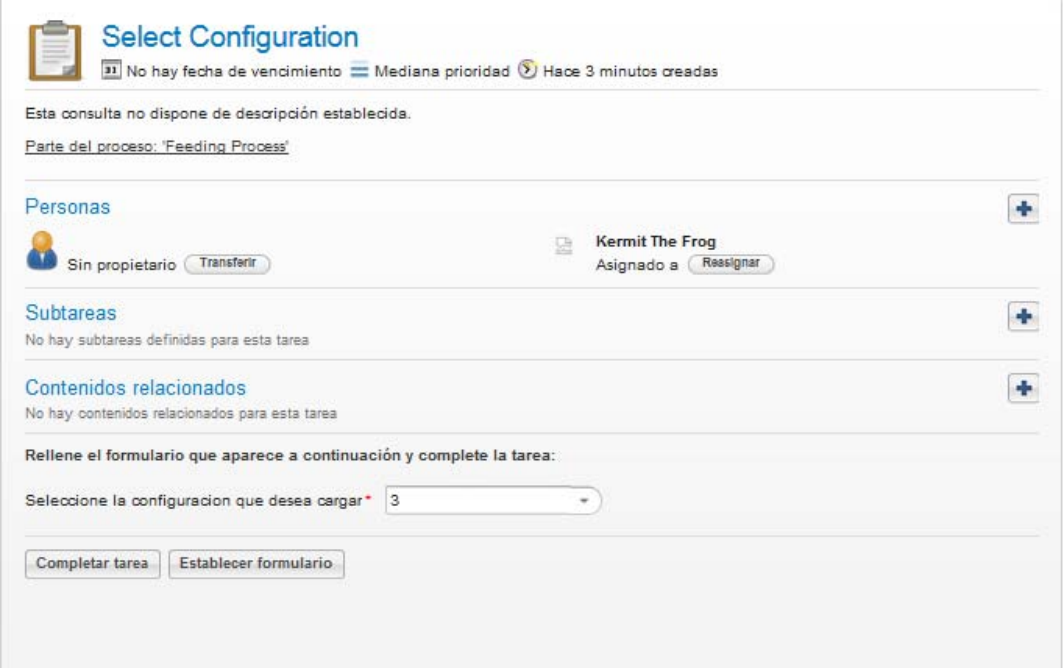


Figura 89 Select Configuration

CREATE STRUCTURES

Una vez ejecutada la tarea automática, se crean las estructuras necesarias definidas como metadatos en el proceso de Configuración como se puede ver en la Figura 99.

Como podemos ver en la Figura 90, al iniciar el proceso de Carga se crea la base de datos que alojará todos los datos del proceso de Feeding. Es importante destacar que el proceso de Configuration trabaja en una base de metadatos exclusiva, y cada proceso de Carga tendrá también una base de datos independiente. El nombre de la base de datos será “base_id” donde id corresponde al identificador del proceso de Configuración (base_3 en este caso).

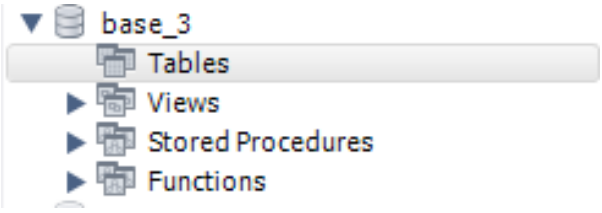


Figura 90 Creación de la base de datos de la configuración seleccionada

En la Figura 91 se puede observar como para cada Web Source seleccionada o creada en el proceso de Configuración, se creará una tabla de nombre “expectedEsquema_webSource”, donde webSource es el nombre de la Web Source y expectedEsquema es el nombre del Expected Schema asociado a la misma en el proceso de Configuración. La extracción de datos se alojará en cada una de estas tablas, para cada Web Source a procesar. Los atributos de estas tablas serán las columnas especificadas en el mapeo del Web Source.

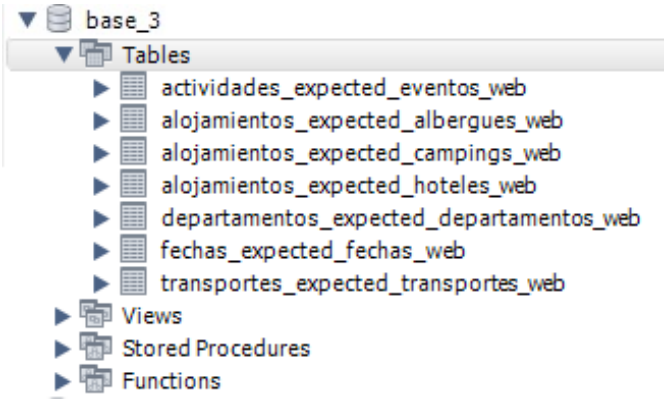


Figura 91 Estado de la base de datos de la configuración luego de creadas las tablas de web_sources

Como muestra la Figura 92 para cada Expected Schema asociados a la Configuración seleccionada, se crearán una tabla por cada uno de ellos, donde el nombre de la tabla será el nombre del Expected Schema. Los campos de estas tablas, serán los atributos definidos en el Expected Schema.

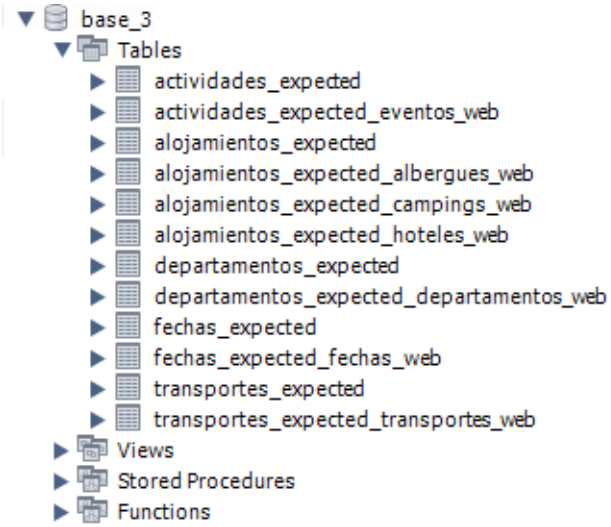


Figura 92 Estado de la base de datos de la configuración luego de creados los expected_schemas

Análogamente, la Figura 93 muestra como se crearán las tablas del Integrated Schema con sus atributos y tipos definidos en la configuración.

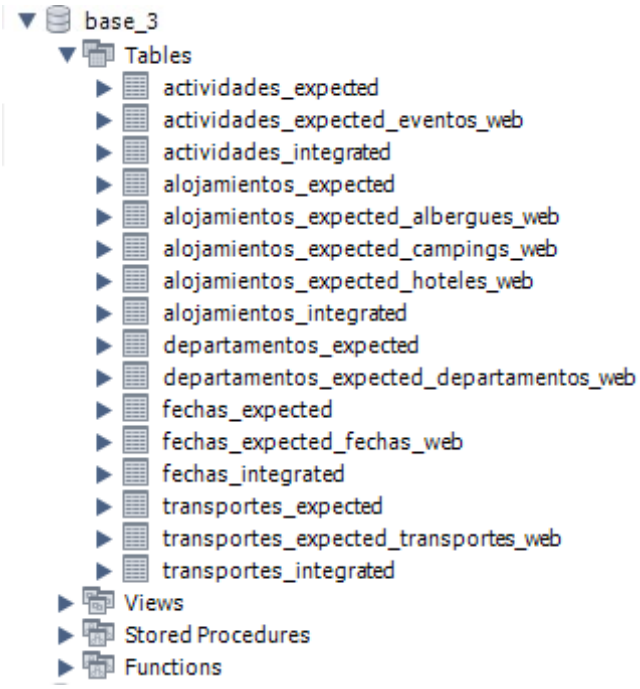


Figura 93 Estado de la base de datos de la configuración luego de creado el integrated_schema

Al igual que con la creación del Integrated Schema, de manera similar se crearán las tablas del DW Schema, con sus atributos y tipos indicados en la etapa de Configuración, esta creación se puede observar en la Figura 94 para finalmente marcar la Configuración como ya procesada. Es conveniente aclarar que si al inicio de esta tarea automática la Configuración ya figura como procesada, no se creará ni destruirá ninguna estructura, sólo se borrarán registros de las mismas. Eso sucede cuando la ejecución de la carga no es la primera (por ejemplo para una siguiente carga de la Configuración 3).

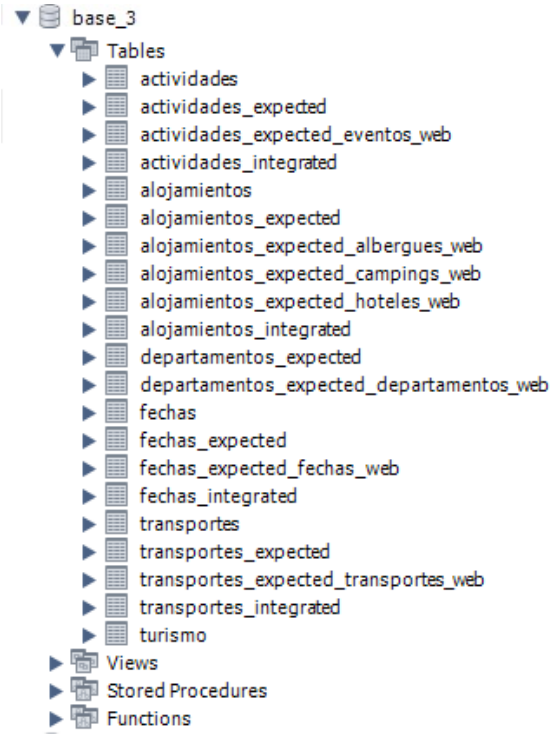
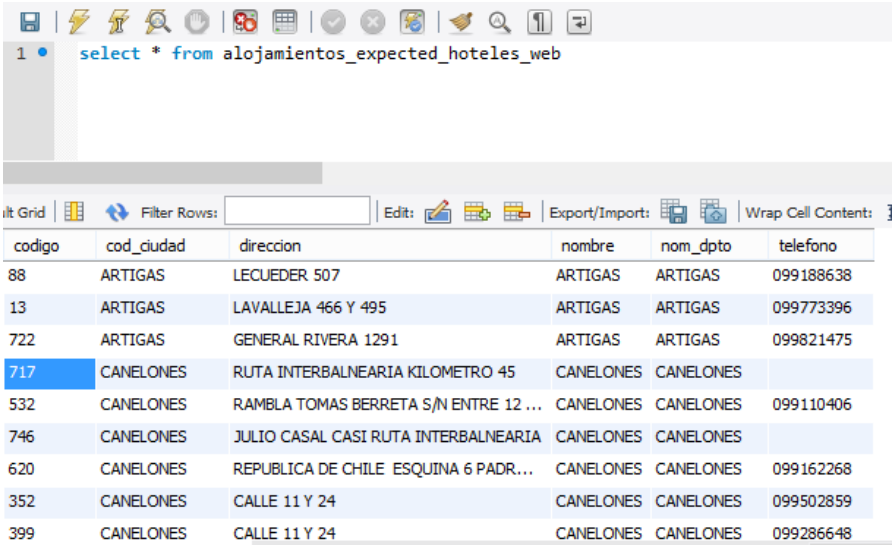


Figura 94 Estado de la base de datos de la configuración luego de creado el dw_schema

EXTRACT DATA

Esta actividad también automática, se encargará de consumir el servicio de extracción de datos para cada Web Source. Los datos obtenidos serán alojados en las tablas creadas a tales fines, mencionadas anteriormente cuyo nombre es la concatenación de los nombres del Expected Schema y del Web Source. Una vez terminada la tarea, todos los datos web habrán sido extraídos y alojados en sus correspondientes tablas, en la Figura 95 podemos ver un ejemplo de ello para la Web Source de Hoteles correspondiente al Expected Schema de alojamientos.



```
1 •  select * from alojamientos_expected_hoteles_web
```

lt Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

codigo	cod_ciudad	direccion	nombre	nom_dpto	telefono
88	ARTIGAS	LECUEDER 507	ARTIGAS	ARTIGAS	099188638
13	ARTIGAS	LAVALLEJA 466 Y 495	ARTIGAS	ARTIGAS	099773396
722	ARTIGAS	GENERAL RIVERA 1291	ARTIGAS	ARTIGAS	099821475
717	CANELONES	RUTA INTERBALNEARIA KILOMETRO 45	CANELONES	CANELONES	
532	CANELONES	RAMBLA TOMAS BERRETA S/N ENTRE 12 ...	CANELONES	CANELONES	099110406
746	CANELONES	JULIO CASAL CASI RUTA INTERBALNEARIA	CANELONES	CANELONES	
620	CANELONES	REPUBLICA DE CHILE ESQUINA 6 PADR...	CANELONES	CANELONES	099162268
352	CANELONES	CALLE 11 Y 24	CANELONES	CANELONES	099502859
399	CANELONES	CALLE 11 Y 24	CANELONES	CANELONES	099286648

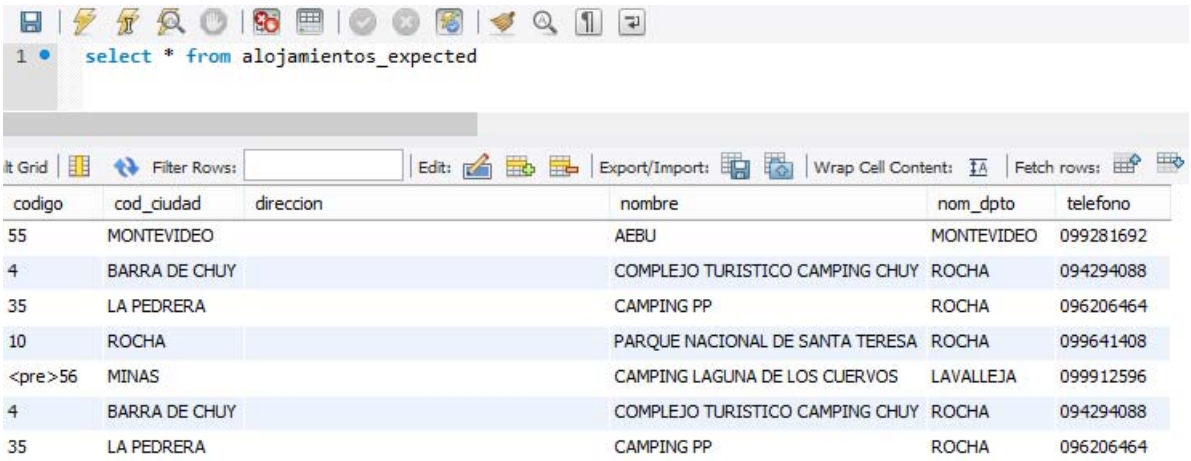
Figura 95 Tabla alojamientos_expected_hoteles_web

PROBLEMS SOLVING

Si bien esta tarea no está implementada, queda prevista en el modelo, la idea de esta tarea es resolver los problemas que surjan de la extracción tomando en cuenta los metadatos de calidad obtenidos en el proceso de Configuración. Nuestro caso de estudio está configurado para que no se ejecute esta tarea manual para no quitar el foco de la prueba que nos interesa. En la sección 5.6 que se refiere a posibles extensiones y mejoras se planteó una forma de abordar este tema.

INTEGRATE AND LOAD EXPECTED TABLES

La actividad de carga de las tablas de Expected, se encargará de unir los datos alojados en las tablas de Web Source, que corresponden al mismo Expected Schema y los cargará en las mencionadas tablas. El fin es agrupar los datos que corresponden al mismo Expected Schema en su correspondiente tabla. En la Figura 96 podemos observar un ejemplo de la carga de un Expected Schema, en este caso la tabla alojamientos_expected la cual contiene todos los datos de las tablas de Web Sources que tienen este Expected Schema (albergues, campings y hoteles).



```
1 •  select * from alojamientos_expected
```

Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

codigo	cod_ciudad	direccion	nombre	nom_dpto	telefono
55	MONTEVIDEO		AEBU	MONTEVIDEO	099281692
4	BARRA DE CHUY		COMPLEJO TURISTICO CAMPING CHUY	ROCHA	094294088
35	LA PEDRERA		CAMPING PP	ROCHA	096206464
10	ROCHA		PARQUE NACIONAL DE SANTA TERESA	ROCHA	099641408
<pre>56	MINAS		CAMPING LAGUNA DE LOS CUERVOS	LAVALLEJA	099912596
4	BARRA DE CHUY		COMPLEJO TURISTICO CAMPING CHUY	ROCHA	094294088
35	LA PEDRERA		CAMPING PP	ROCHA	096206464

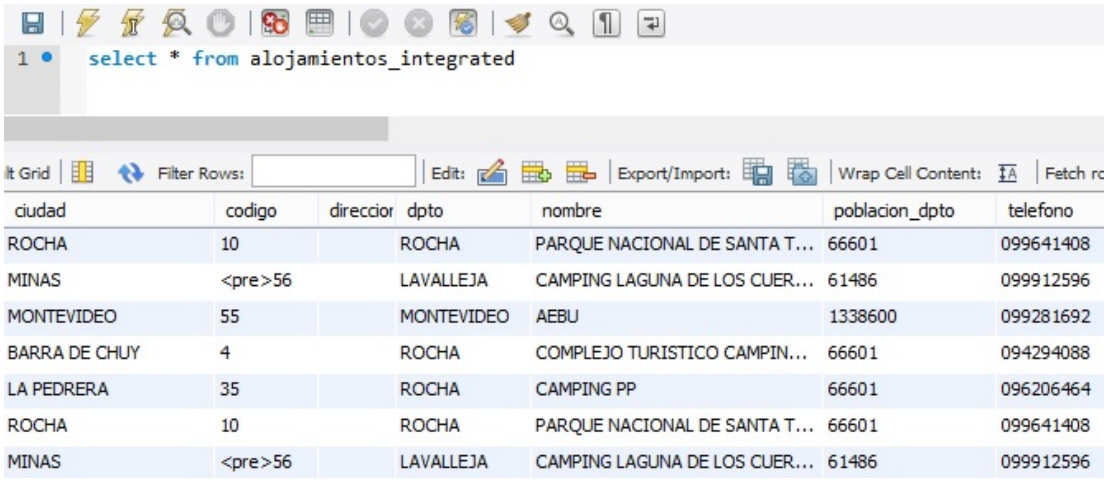
Figura 96 Tabla alojamientos_expected

CONFLICTS RESOLUTION FOR EXPECTED TABLES

Esta tarea no está implementada, queda prevista en el modelo, la idea de esta tarea es resolver los conflictos que surjan en la carga de los Expected Schemas tomando en cuenta los metadatos de calidad obtenidos en el proceso de Configuración. Al igual que explicamos en "Problems Solving" esta tarea está configurada para no ejecutarse.

INTEGRATE AND LOAD INTEGRATED TABLES

En función de los mappings de Expected Schema e Integrated Schema, y con la definición de joins correspondientes, cargaremos las tablas del Esquema Integrado, joineando las tablas correspondientes del ExpectedSchema. Como podemos observar en la Figura 97 queda cargada la tabla alojamientos_integrated con el dato además de la población del departamento como lo indicamos en la Configuración.



```
1 •  select * from alojamientos_integrated
```

Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rc

ciudad	codigo	direccion	dpto	nombre	poblacion_dpto	telefono
ROCHA	10		ROCHA	PARQUE NACIONAL DE SANTA T...	66601	099641408
MINAS	<pre>56		LAVALLEJA	CAMPING LAGUNA DE LOS CUER...	61486	099912596
MONTEVIDEO	55		MONTEVIDEO	AEBU	1338600	099281692
BARRA DE CHUY	4		ROCHA	COMPLEJO TURISTICO CAMPIN...	66601	094294088
LA PEDRERA	35		ROCHA	CAMPING PP	66601	096206464
ROCHA	10		ROCHA	PARQUE NACIONAL DE SANTA T...	66601	099641408
MINAS	<pre>56		LAVALLEJA	CAMPING LAGUNA DE LOS CUER...	61486	099912596

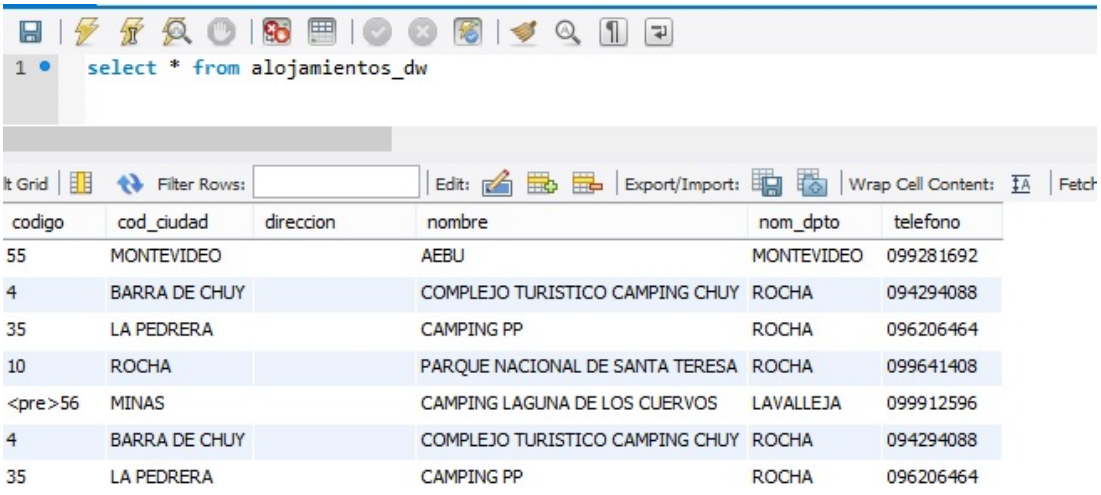
Figura 97 Tabla alojamientos_integrated

CONFLICTS RESOLUTION FOR INTEGRATED TABLES

Esta tarea no está implementada pero queda prevista en el modelo, la idea de esta tarea es resolver los conflictos que surjan en la carga del Integrated Schema tomando en cuenta los metadatos de calidad obtenidos en el proceso de Configuración. Nuevamente esta tarea está configurada para no ejecutarse.

INTEGRATE AND LOAD DW DIMENSION TABLES

En función de los mappings de Integrated Schema y DW Schema, y con la definición de joins correspondientes, cargaremos las tablas de dimensiones del Esquema DW, joineando las tablas correspondientes del IntegratedSchema. Como lo muestra la Figura 98 para el caso de la tabla de dimensiones de alojamientos (alojamientos_dw) queda cargada con los datos correspondientes a los alojamientos con la que fue configurada.



```
1 •  select * from alojamientos_dw
```

lt Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch

codigo	cod_ciudad	direccion	nombre	nom_dpto	telefono
55	MONTEVIDEO		AEBU	MONTEVIDEO	099281692
4	BARRA DE CHUY		COMPLEJO TURISTICO CAMPING CHUY	ROCHA	094294088
35	LA PEDRERA		CAMPING PP	ROCHA	096206464
10	ROCHA		PARQUE NACIONAL DE SANTA TERESA	ROCHA	099641408
<pre>56	MINAS		CAMPING LAGUNA DE LOS CUERVOS	LAVALLEJA	099912596
4	BARRA DE CHUY		COMPLEJO TURISTICO CAMPING CHUY	ROCHA	094294088
35	LA PEDRERA		CAMPING PP	ROCHA	096206464

Figura 98 Tabla alojamientos_integrated

CONFIGURATE QUERY TO LOAD DW FACT TABLES

Es una tarea automática que se encarga de generar los *insert* para cargar las tablas de hechos. Este *insert* no se ejecuta en esta actividad si no que se arma en base a los metadatos obtenidos del proceso de Configuración para poder mostrarla en la siguiente tarea manual *"Confirm Query To Load DW Fact Tables"* en donde el usuario podrá confirmar la inserción de los datos de esa forma o modificar la query.

CONFIRM QUERY TO LOAD DW FACT TABLES

Esta tarea muestra al usuario como se cargará cada tabla de hechos según una sentencia SQL, teniendo la posibilidad de modificarla. Para nuestro ejemplo no le realizamos cambios. La Figura 99 refleja lo mencionado.

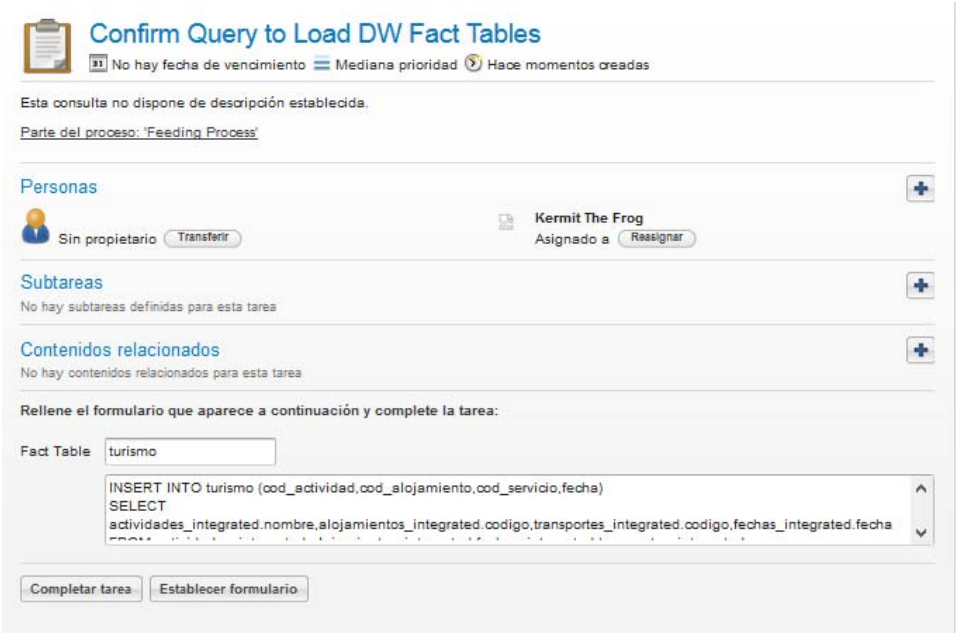


Figura 99 Confirm Query to Load DW Fact Tables

Luego de ejecutada esta actividad podemos ver si las tablas quedaron con los datos que necesitamos o si hubo algún problema en la carga. En la Figura 100 podemos ver los resultados.

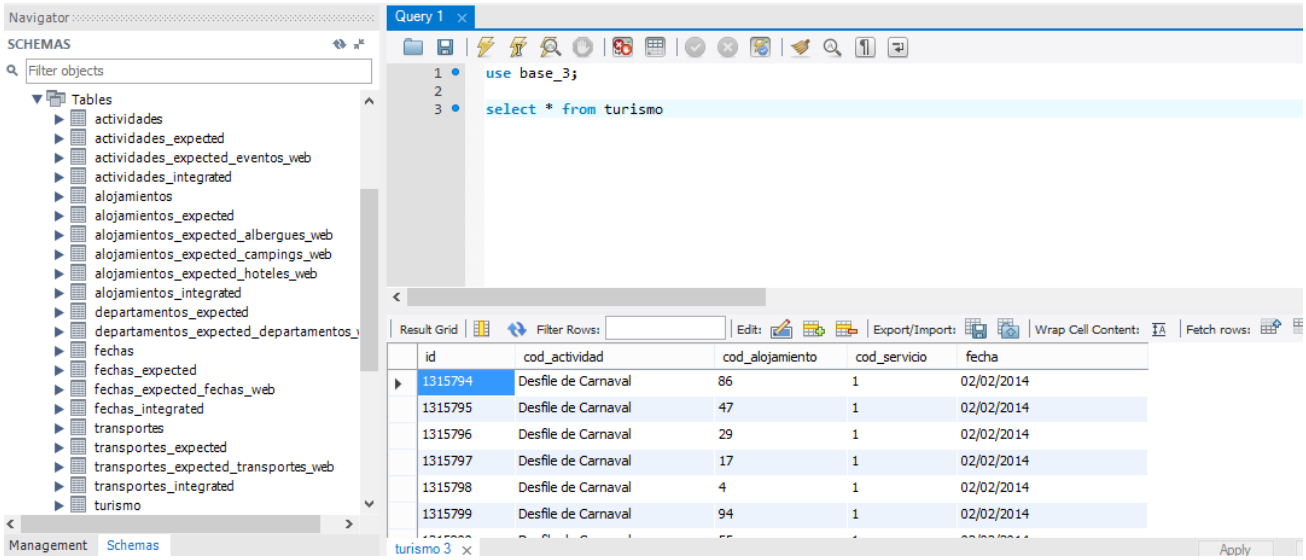


Figura 100 Estado de la base_3 con los datos de la tabla de Hechos cargada

COMMIT LOAD OF DATA WAREHOUSE

Finalmente, esta tarea permite al usuario decidir si la carga de cada tabla de hechos es correcta y desea que sea definitiva o desea cargarla nuevamente, en el formulario de la Figura 101 se muestra lo mencionado. En caso que el usuario confirme que la carga es satisfactoria se vuelve a ejecutar la tarea automática *"Configurate Query to Load DW Fact Tables"* para armar la consulta con la que se va a llenar cada una de las tablas de hechos pudiendo nuevamente modificar cada una de ellas.

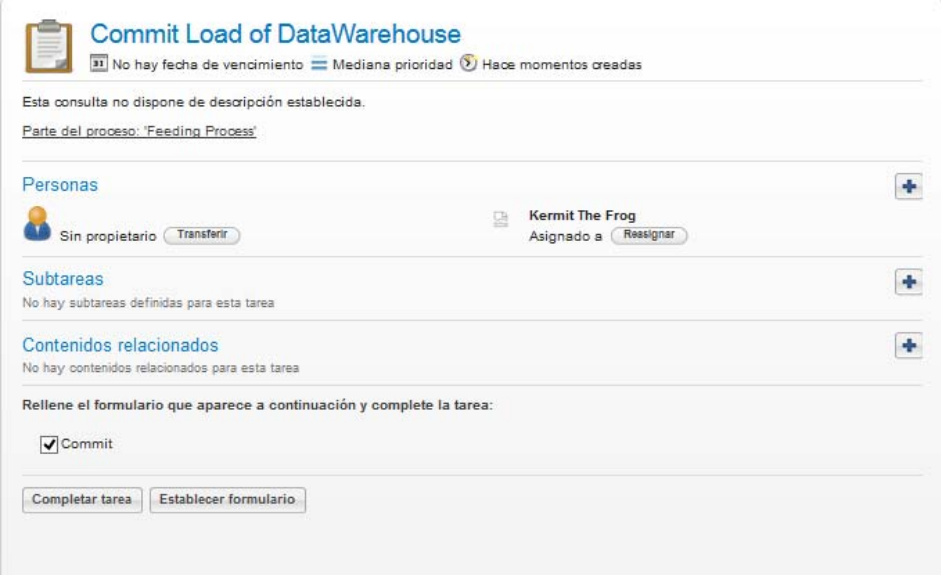


Figura 101 Commit Load of Data Warehouse

8 Conclusiones y Trabajo a Futuro.

En esta sección presentamos las conclusiones finales de nuestro trabajo así como también el trabajo a futuro planteado y las posibles extensiones al mismo.

Los resultados de este trabajo de Proyecto de Grado serán presentados por las tutoras en la Conferencia Latinoamericana de Informática (CLEI) a realizarse en Arequipa, Perú en octubre de este año. [15]

8.1 Conclusiones.

En este Proyecto hemos podido exponer todos los conceptos teóricos necesarios para poner en práctica la construcción de un WW basado en la calidad sobre un BPM. Implementamos un prototipo que realiza la prueba de conceptos utilizando la tecnología indicada previamente, pudiendo demostrar que es posible automatizar la mayoría de las actividades correspondientes a la construcción de un WW habiendo definido previamente los metadatos que corresponden a la configuración del mismo.

Se ejecutó un caso de estudio con éxito, que si bien no todas sus fuentes de datos son Web (ya que algunas son armadas por nosotros en formato *csv* por falta de datos Web), se pudo ejecutar dos procesos independientes, un proceso de Configuración del WW y otro proceso de Carga del mismo.

En cuanto a la configuración del WW, se pudieron definir todos los conceptos manejados como son Web Data Sources, Data Services y Expected Schemas para él/los Web Data Sources.

Se definieron también el Integrated Schema y el modelo estrella del DW final, con sus tablas de dimensiones y sus tablas de hechos. Además, para cada una de dichas tablas, se pudo indicar al Sistema de qué manera cargar las mismas con los datos correspondientes.

Una vez finalizado el proceso de Configuración, el Sistema tiene conocimiento en base a los metadatos generados sobre cómo debe extraer los datos de la Web y cómo transformar dichos datos e integrarlos, para finalmente poder cargar el DW deseado. Dicha carga la pudimos realizar con el proceso de Carga mencionado en este documento y el proceso se pudo hacer prácticamente automático, siempre y cuando no existan fallas o inconsistencia de datos.

Una vez obtenida la manera en que se cargarán las tablas del DW, el Sistema nos permite confirmar dicha carga o realizar algunas modificaciones de forma manual.

Una limitante fue la interfaz utilizada, la herramienta Activiti Explorer es muy limitada y rígida y debimos diseñar nuestros procesos en función de lo que la interfaz nos permitía mostrar al usuario. Como mejora a este Sistema, se podría incluir una capa Web implementada sobre el motor de procesos Activiti cuya interfaz sea a gusto del programador y contenga todo tipo de elementos que permitan mostrar y pedir al usuario para la ejecución del proceso.

Otra limitante fue la falta de publicación de datos Web, si bien existen muchos que se publican, la mayoría son archivos comprimidos que el usuario descarga y obtiene un *csv* con todos los datos. Nuestro Sistema apuntó a datos expuestos directamente en la Web. De todas formas, el Sistema que proponemos extrae los datos de la Web a archivos temporales *csv*, por lo que para esos casos que mencionamos también podría utilizarse, manualmente se deberá extraer los datos desde el comprimido a disco y ubicar los archivos *csv* en un directorio en particular.

Por último, no pudimos encontrar mucha información de otras investigaciones sobre construcción de WW sobre procesos BPM. La temática que plantea este Proyecto es muy novedosa, lo cual resultó motivante pero presentó los obstáculos propios de la falta de información.

8.2 Trabajo a futuro.

El análisis y el diseño de todo lo implementado en este prototipo han llevado un tiempo bastante extenso y no nos permitió poder satisfacer las necesidades que a priori teníamos sobre la calidad de los datos y los Data Services que nos habíamos planteado. Dejamos el sistema abierto y extensible para que futuros estudiantes puedan continuarlo y desarrollar el módulo que corresponde al tratamiento de la calidad que hacíamos mención en la propuesta del problema. Para ello, cuentan con documentación técnica muy completa desde el punto de vista de la programación y del análisis y diseño de la solución.

Otro módulo que también nos hubiera gustado implementar es el que resuelve problemas de integración de los datos, esto es, detectar automáticamente problemas de integración de diferentes datos que refieren a lo mismo pero que están publicados con diferentes formatos. Hubiera sido interesante que el sistema los detecte automáticamente y lo exponga al usuario para su tratamiento manual. Esto también queda abierto y extensible para trabajo a futuro.

Referencias:

[1] - Procesos de Negocio y Desarrollo de SW - Francisco Ruiz - Universidad de Cantabria

[2] - Taller de Gestión y Tecnologías de Procesos de Negocio - Grupo COAL - Facultad de Ingeniería (UdelaR)

[3] - Diseño y Construcción de Data Warehouse - Adriana Marotta - Facultad de Ingeniería (UdelaR)

[4] - Carpani, F.: "CMDM: A conceptual multidimensional model for Data Warehouse". Master Thesis Advisor: Raúl Ruggia. Pedeciba, Universidad de la República, Uruguay, 2000

[5] - Kimball, R.:"The Datawarehouse Toolkit". John Wiley & Son, Inc., 1996

[6] - Towards the construction of Quality-aware Web Warehouse with BPMN 2.0 Business Processes. Andrea Delgado, Adriana Marotta, Laura González. 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)

[7] - Quality Management in Web Warehouse. Adriana Marotta, Laura González, Lorena Etcheverry, Bruno Rienzi, Raúl Ruggia, Flavia Serra- IGI Global

[8] - BonitaSoft

[9] - https://catalogodatos.gub.uy/

[10] - Concepción de Sistemas de Información - Instituto de Computación - Facultad de Ingeniería (UdelaR)

[11] - Servicios Rest con Spring

[12] - Sistema de Business Intelligence para análisis de precios al consumo. Matías Morales, Sofía Maiolo, Adriana Marotta. Facultad de Ingeniería, Universidad de la República, Proyecto de Grado 2014

[13] - Diseño Lógico de Data Warehouse a partir de Esquemas Conceptuales Multidimensionales. Verónika Peralta, Facultad de Ingeniería, Universidad de la República, Tesis de Maestría 2001

[14] - Activiti in Action - Executable business processes in BPMN 2.0 - Tijs Rademakers Forewords by Tom Baeyens and Joram Barrez

[15] - A. Delgado, A. Marotta, Automating the process of building flexible Web Warehouses with BPM Systems, XLI Conferencia Latinoamericana de Informática, Arequipa-Peru, Octubre 2015