Universidad de la República
Facultad de Ingeniería
Instituto de Computación
Uruguay

Modelos y algoritmos para minería de procesos y datos

Andrés Borges

Alexis Artus

Proyecto de Grado
Ingeniería en Computación
Universidad de la República

Montevideo, Uruguay, Julio de 2021

Supervisor: Dr. Ing. Daniel Calegari
Dra. Ing. Andrea Delgado
Universidad de la República

Resumen

En entornos organizacionales de gran escala como por ejemplo e-government, las organizaciones enfrentan diversos desafíos para gestionar sus procesos de negocio y los datos que generan. El volumen de datos que un proceso genera, ha aumentado y se ha distribuido entre las diferentes tecnologías utilizadas, definiendo ecosistemas en los que se hace necesario integrar diferentes visiones, técnicas y herramientas para la gestión de la información.

La Gestión de Procesos de Negocio (Business Process Management, BPM) [1] ayuda a las organizaciones a gestionar sus procesos, desde el modelado, implementación, ejecución, evaluación y análisis, brindando soporte para la automatización y mejora de procesos. En la mayoría de los casos se utilizan Sistemas BPM (Business Process Management Systems, BPMS) [2] para modelar, implementar y ejecutar procesos, pero aún con estas herramientas es complejo identificar la conexión entre los datos de los procesos y los diferentes sistemas con los que se comunica.

Como consecuencia, recopilar los datos para obtener una visión completa de las ejecuciones de los procesos se vuelve una tarea compleja, lo que deriva en una visión partida entre datos de los procesos por un lado y datos de la organización por otro. Por este motivo, es de suma importancia lograr proporcionar a las organizaciones un todo para generar una vista unificada a la cual se pueda aplicar técnicas de minería de procesos y de minería de datos [3] para brindar a las organizaciones la inteligencia necesaria para mejorar sus procesos.

Por otra parte, los sistemas de data warehouse [4] han sido utilizados por las organizaciones para modelar sus datos y construir análisis que generan reportes para la toma de decisiones. Estos sistemas de data warehouse permitirán tener una vista unificada de los datos con la cual apoyar la toma de decisiones a la hora de optimizar los procesos.

El objetivo de este proyecto es implementar una solución que permita la integración de datos de procesos de negocio y datos organizacionales. Para ello, se implementa un metamodelo integrador [5] y a partir de él, construir un data warehouse que permita proveer a la organización de la inteligencia organizacional necesaria para mejorar su operativa diaria.

Palabras clave: Procesos de negocio, Sistemas de gestión de procesos de negocio, Minería de procesos, Extracción e Integración de datos, Data Warehouse.

Contenido

1 Introducción

Los procesos de negocio han crecido en complejidad debido a nuevas tecnologías y nuevas capacidades que permiten un mayor número de interacciones entre sistemas. Debido a ello, las organizaciones han tenido que buscar formas de organizar sus procesos y han recurrido a utilizar Business Process Management (BPM) [1]. BPM proporciona distintas reglas que son de ayuda a la hora de modelar, implementar, ejecutar, evaluar y analizar los procesos de negocio. Para la implementación y ejecución de los procesos se tiene Business Process Management Systems (BPMS) [2] que ofrece sistemas de software genéricos basados en modelos de procesos explícitos para implementarlos.

A pesar de todas las herramientas que se tienen para gestionar procesos, aún no existe una forma de identificar la conexión entre los datos de los procesos y los diferentes sistemas con los que se comunica. Recopilar los datos para obtener una visión completa de las ejecuciones de los procesos es una tarea compleja y para la cual no existen métodos definidos. Por este motivo, es de suma importancia lograr proporcionar a las organizaciones un método para generar una vista unificada a la cual se pueda aplicar técnicas de minería de procesos [3] y de minería de datos.

En un trabajo realizado previamente [5], se presenta un esquema conceptual para combatir la brecha que se tiene entre los datos de los procesos y los datos organizacionales. A partir de este diseño conceptual es posible implementar un modelo integrador en una base de datos, donde cargar los datos provenientes de los procesos y bases organizacionales. Basándose en este modelo se pretende implementar un algoritmo de matcheo consiguiendo un modelo unificado que sirve de input para sistemas de data warehouse [4] o para generar un log extendido. Las organizaciones utilizan sistemas de data warehouse para manejar grandes volúmenes de datos y generar modelos para apoyar la toma de decisiones.

El objetivo general de este proyecto es lograr la conexión entre los datos de los procesos y bases organizacionales a partir de la solución propuesta en el artículo [5]. Utilizando el diseño conceptual del metamodelo integrador como base para modelar esta conexión y a

partir de ella, construir un modelo de data warehouse que permita hacer análisis de los datos integrados. Los objetivos específicos del proyecto son:

1. Estudiar conceptos de BPM, configuración y ejecución de procesos, lenguajes BPMN 2.0 y herramientas.
2. Implementar la solución del metamodelo en un esquema de base de datos.
3. Analizar modelos y algoritmos existentes para la integración de datos de logs de eventos de procesos y datos organizacionales.
4. Implementar un algoritmo de extracción y carga del metamodelo a partir de logs y base de datos.
5. Implementar algoritmo de matcheo para generar la conexión entre los datos.
6. Desarrollar un modelo de data warehouse genérico basado en el metamodelo.
7. Aplicar la propuesta a datos reales y soporte extendido de herramientas.

Este proyecto se enmarca en el proyecto de investigación “Minería de procesos y datos para la mejora de procesos en las organizaciones” financiado por Comisión Sectorial de Investigación Científica, UdelaR en colaboración con AGESIC (Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento). En adición a este documento, también se publicaron dos artículos [6] y [7]. En el primero de ellos, se muestra un acercamiento al problema que se resuelve en este proyecto, mientras que el segundo, se centra en explicar cómo explotar la solución a partir de la implementación de un data warehouse genérico.

El presente informe está organizado de la siguiente manera. En el Capítulo 2 se desarrolla un marco teórico con las definiciones de Gestión de Procesos, extracción de logs de base de datos y data warehouse. En el Capítulo 3 se presenta el problema a resolver, presentando el metamodelo, extracción y carga del metamodelo y construcción del data warehouse. En los Capítulos 4 y 5 se explica cómo se desarrolló la solución del problema que se expuso en el Capítulo 3, mientras que en el Capítulo 6 se aplica dicha solución para un caso particular llamado Student Mobility y se muestra cómo se puede explotar la solución. Finalmente, en el Capítulo 7 se presentan las conclusiones del proyecto y se mencionan trabajos futuros que se pueden desarrollar a partir de la solución alcanzada.

2 Marco Teórico

Esta sección se centra en brindar las herramientas necesarias para entender completamente la propuesta, como también, las diferentes opciones investigadas y que, a partir de las cuales, se desarrolló la solución.

Los tópicos a explicar son:

- Gestión de Procesos de Negocio
- Extracción de bases de datos
- Data warehouse

2.1. Gestión de Procesos de Negocio

Toda organización busca identificar los distintos procesos que ayudan en las distintas operaciones que se ejecutan en su negocio. Los procesos de negocio se pueden definir como:

"Un proceso de negocio consiste en un conjunto de actividades que son realizadas en coordinación en un ambiente organizacional y técnico. Estas actividades juntas realizan un objetivo de negocio. Cada proceso de negocio es representado por una sola organización, pero puede interactuar con procesos de negocio ejecutados por otras organizaciones."[1]

Después de esta primera consideración sobre procesos de negocio, de sus componentes y de sus interacciones, se debe ampliar la vista un poco más, ya que el manejo de procesos de negocio no solo cubre la representación de procesos de negocio, sino que tiene actividades adicionales.

Tomando una nueva definición se tiene:

"El manejo de procesos de negocio incluye concepto, métodos y técnicas para soportar el diseño, administración, configuración, ejecución y análisis de los procesos."[1]

La base de la gestión de procesos es la representación explícita de los procesos con sus actividades y flujos. Una vez los procesos de negocio son definidos, pueden ser sujetos a análisis, mejoras y publicaciones.

Una mejor imagen de lo que conlleva un proceso de negocio se ve reflejado en la Figura 2.1. Este ejemplo se basa en el proceso que se lleva a cabo al momento de que se realiza una compra.

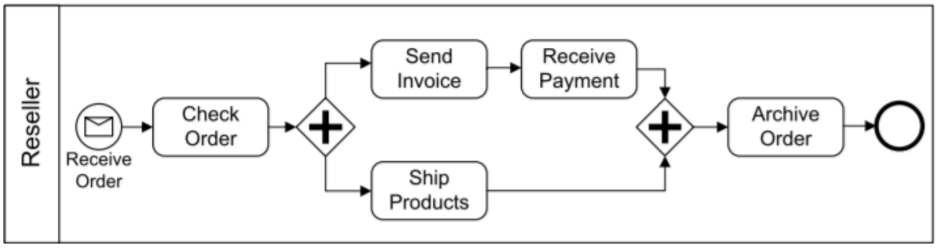


Figura 2.1: Modelo de Proceso, extraído de [1]

El primer elemento es el encargado de recibir la solicitud para que luego pueda ser chequeada. Después de este paso, se ejecutan dos actividades en paralelo las cuales se encargan por un lado manejar el pago de la orden y por el otro lado el envío. Finalmente, cuando todas estas actividades terminan, la orden termina y el proceso finaliza.

Este es un caso reducido del lenguaje de modelo de procesos introducido por la OMG y

llamada BPMN 2.0 (Business Process Model and Notation) [8].

2.1.1. Ciclo de vida de un proceso

Cada proceso tiene un ciclo de vida que describe cómo se maneja en cada fase. Se habla de un ciclo, ya que puede repetirse cuantas veces sea necesario para poder crear una mejora continua del proceso.

En la Figura 2.2 se representa el esquema básico del ciclo de vida que un Proceso de Negocio tiene, donde se distinguen cuatro grandes áreas: Diseño y Análisis, Configuración, Ejecución y Evaluación.

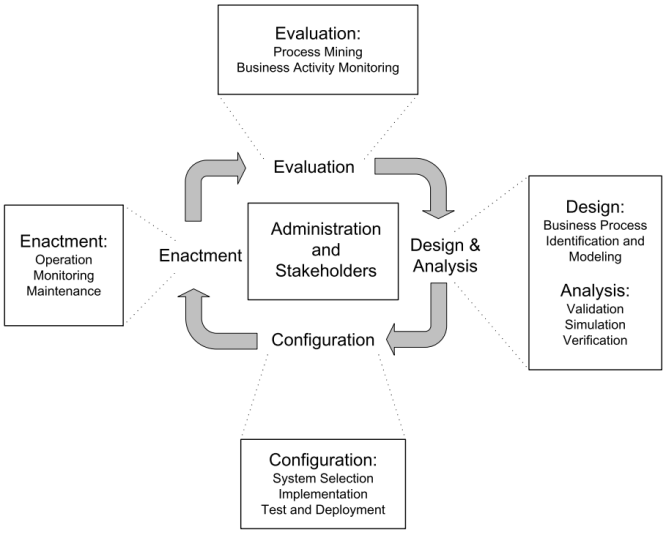


Figura 2.2: Ciclo de vida de un BPM, extraído de [1]

Las distintas partes del ciclo de vida significan:

- Diseño y Análisis: El ciclo de vida del proceso de negocio comienza en la fase de Diseño y Análisis, donde se realizan estudios sobre el proceso de negocio, su organización y detalles técnicos. A partir de dicho estudio los procesos son identificados, revisados, validados y representados por modelos de procesos de negocio. Técnicas de modelado de procesos de negocio como de validación, simulación y verificación son usadas en esta fase.

- Configuración: Una vez que el modelo del proceso es diseñado y verificado, el proceso necesita ser implementado. Hay diferentes formas de hacerlo:

- En caso de que se use un sistema de software para realizar el proceso se necesitará seleccionar una plataforma de implementación. Este sistema debe ser configurado acorde al ambiente organizativo de la empresa y al proceso de negocio que va a controlar. Este tipo de plataforma es llamado BMPS y se hablará más adelante en el documento.

- Desarrollando un set de políticas y procedimientos que los empleados de la empresa deben cumplir. Sin necesidad del soporte de un BPMS.

- Ejecución: Una vez que la fase de configuración termina, las instancias del proceso pueden ser ejecutadas. Esta fase abarca la ejecución en tiempo real del proceso. Las instancias de procesos son iniciadas para cumplir los objetivos de negocio de la organización. Por lo general son iniciadas por un evento definido, como en el ejemplo de la Figura 2.1, una compra hecha por un cliente.

- Evaluación: La fase de evaluación usa la información disponible para evaluar y mejorar el modelo e implementación del proceso de negocio. Los logs de ejecución son evaluados usando actividades de monitoreo y técnicas de minería de procesos. Estas técnicas son utilizadas para identificar la calidad del modelo y que las ejecuciones sean adecuadas.

2.1.2. Sistemas de Gestión de Procesos de Negocio

Tradicionalmente, los procesos son representados manualmente, guiados por el conocimiento del personal de la organización asistido por las regulaciones y los procesos instalados. Las organizaciones pueden lograr beneficios adicionales si usan sistemas de software para coordinar las actividades que componen un proceso de negocio. Estos sistemas de software se llaman BPMS (Business Process Management Systems).

Se puede definir como:

"Un BPMS es un sistema de software genérico que está impulsado por representaciones de procesos explícitos para coordinar la implementación de los procesos de negocio."[1]

Para mantener un mismo formato, o un formato cercano, muchos de los BPMS actuales hacen uso de la notación BPMN mencionada anteriormente. Existen múltiples herramientas que ofrecen implementaciones de procesos, entre ellas se encuentran Activiti.

Activiti

Activiti se define como:

"Activiti is the leading lightweight, java-centric open-source BPMN engine supporting real-world process automation needs"[9].

Es una de herramienta completa para modelar, implementar y ejecutar cualquier tipo de pro-

cesos que una organización necesite.

Para poder modelar los procesos usando BPMN, Activiti ofrece una herramienta web. Esta herramienta ofrece dos formas distintas de modelar, a partir de la carga de un archivo xml con la definición de los modelos o un modelador amigable al usuario donde se pueden definir los modelos gráficamente.

La primera opción debe respetar el lenguaje de modelado BPMN y la codificación que se explica en [10]. Un ejemplo de un archivo xml para un modelo se puede ver en la Figura 2.3, donde se ilustran los diferentes componentes del modelo como process, userTask y startEvent.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions id="definitions"
             targetNamespace="http://activiti.org/bpmn20"
             xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:activiti="http://activiti.org/bpmn">

  <process id="vacationRequest" name="Vacation request">

    <startEvent id="request" activiti:initiator="employeeName">
      <extensionElements>
        <activiti:formProperty id="numberOfDays" name="Number of days" type="long" value="1"
required="true"/>
        <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyy)"
datePattern="dd-MM-yyyy hh:mm" type="date" required="true" />
        <activiti:formProperty id="vacationMotivation" name="Motivation" type="string" />
      </extensionElements>
    </startEvent>
    <sequenceFlow id="flow1" sourceRef="request" targetRef="handleRequest" />

    <userTask id="handleRequest" name="Handle vacation request" >
      <documentation>
        ${employeeName} would like to take ${numberOfDays} day(s) of vacation (Motivation:
${vacationMotivation}).
      </documentation>
      <extensionElements>
         <activiti:formProperty id="vacationApproved" name="Do you approve this vacation"
type="enum" required="true">
          <activiti:value id="true" name="Approve" />
          <activiti:value id="false" name="Reject" />
        </activiti:formProperty>
        <activiti:formProperty id="managerMotivation" name="Motivation" type="string" />
      </extensionElements>
      <potentialOwner>
        <resourceAssignmentExpression>
          <formalExpression>management</formalExpression>
        </resourceAssignmentExpression>
      </potentialOwner>
    </userTask>
```

Figura 2.3: Modelo xml de un proceso

Por otro lado, para los usuarios que no son expertos en el área, se tiene un modelador con todas las definiciones necesarias para modelar un proceso. Entre las definiciones se tiene:

- Start Events.
- Activities: User task, Service Task, Mail Task.
- Gateways: Exclusive gateway, Parallel gateway, Inclusive gateway y Event gateway.
- Boundary Events: Boundary timer event, Boundary error event, Boundary signal event.
- Swimlanes: Pool y Lane.
- End Events.

Debido a estas definiciones precargadas, el modelador aporta gran valor para los usuarios de la herramienta. En la Figura 2.4 se puede ver las distintas definiciones que el modelador ofrece como también un ejemplo de un modelo construido.

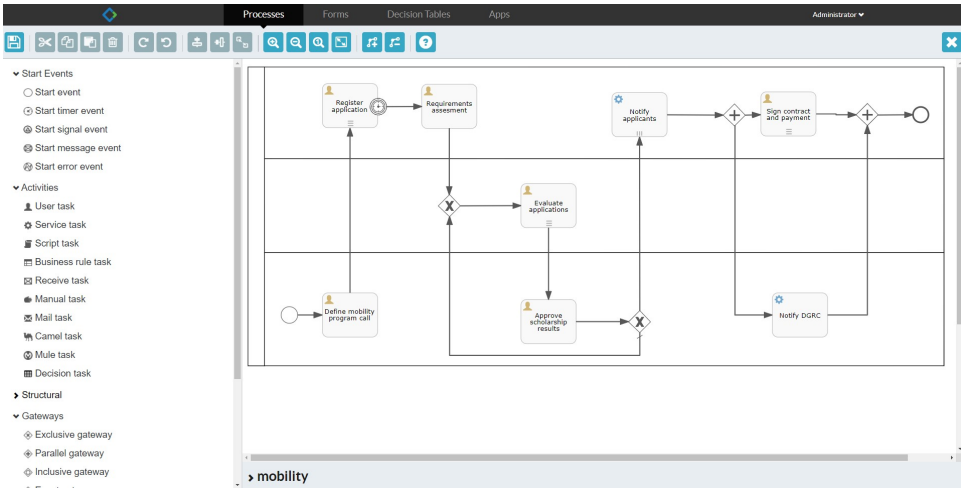


Figura 2.4: Modelador de Activiti

El núcleo de la implementación de dichos modelos está centrado en el lenguaje Java, donde Activiti ofrece una librería que permite a los desarrolladores manejar los procesos, actividades y variables. A partir del modelador se puede incluir diferentes llamados a clases de un proyecto Java para realizar diferentes operaciones como, por ejemplo, la inserción de los valores cargados en un formulario en una base de datos relacionada al proceso. Un ejemplo de este tipo de implementación se puede ver en la Figura 2.5.

Para la ejecución de los procesos definidos en Activiti se tiene una base de datos cargada con las definiciones necesarias y donde se guardarán los datos de todas las ejecuciones. La definición general de las tablas que se puede observar en la Figura 2.6 que también es utilizada por otras soluciones para mantener una norma, donde se tiene todo lo necesario para seguir el BPMN2.0.

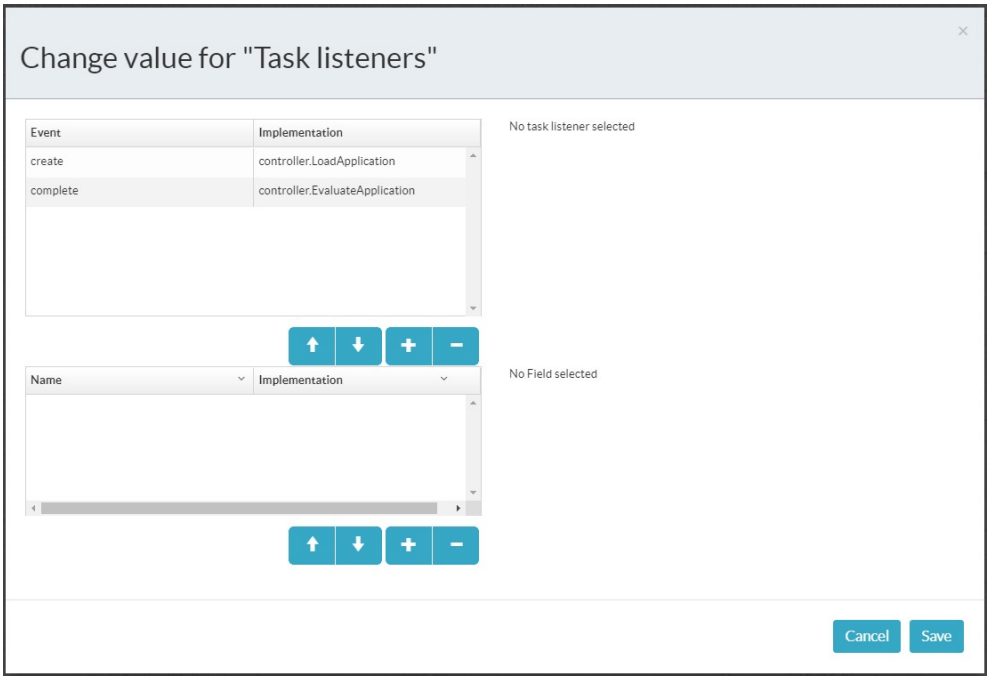


Figura 2.5: Clases Java en el modelador

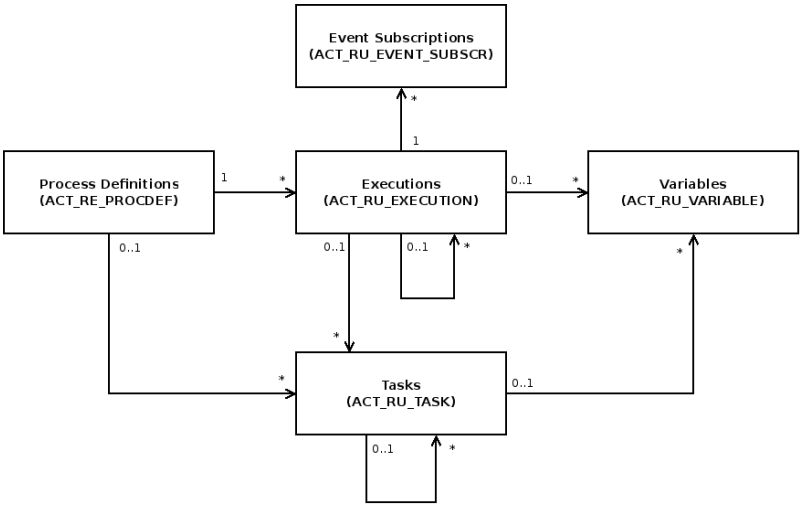


Figura 2.6: Definición de base de datos, extraído de [9]

Sus tablas hacen referencia a:

- ACT_RE_*: RE significa repositorio. Tablas con este prefijo contienen información estática como definición de procesos y recursos de procesos (imágenes, reglas, etc.).
- ACT_RU_*: RU significa runtime. Estas son tablas de tiempo de ejecución que contienen información sobre las instancias de ejecuciones de procesos, user tasks, variables, jobs, etc. Solo se guardan los datos en tiempo de ejecución, cuando finaliza una instancia la información es eliminada. Esto mantiene la tabla pequeña y con buena performance.
- ACT_ID_*: ID significa identidad. Estas tablas contienen información de identidad, como usuarios, grupos, etc.
- ACT_HI_*: HI significa historia. Estas tablas contienen los datos históricos, como instancias de procesos que ya pasaron, variables, tasks, etc.
- ACT_GE_*: Datos generales, que son utilizados en varios casos de uso.

Para poder ejecutar las instancias de procesos definidas en Activiti fácilmente, la herramienta web ofrece un manejador de ejecuciones donde se puede ingresar con diferentes usuarios para realizar las tareas que les competan. El usuario que ingresa al manejador puede seleccionar entre la lista de tareas que tiene asignadas y completarlas. Todos los datos de las ejecuciones hechas por los usuarios son almacenados en la base anteriormente definida.

Por otra parte, para manejar las ejecuciones de una forma externa a la herramienta, se ofrece una API REST que permite manejar las instancias de procesos desde, por ejemplo, un sistema externo. Entre las funcionalidades que ofrece se tiene:

- GET runtime/process-instances: Obtener las instancias de proceso que se están ejecutando en el servidor de Activiti.
- GET runtime/process-instances/{processInstanceId}: Obtener la información de una instancia de proceso en ejecución en particular.
- POST runtime/process-instances: Iniciar una instancia de proceso pasando como parámetro el identificador de definición de proceso en el body,
- GET runtime/tasks: Obtener las tasks que se están ejecutando para una instancia.
- GET runtime/tasks/{taskId}: Obtener información de una task en particular, como por ejemplo el formulario que tiene asociado pasando como parámetro el identificador de la task.
- POST form/form-data: Completar datos de un formulario a partir de un identificador de task pasado como parámetro en el body.
- POST runtime/tasks/{taskId}: Completar una task en específico con su id.

2.1.3. Registro de datos del proceso

Las organizaciones utilizan diferentes tipos de bases de datos que son impactadas durante la ejecución de los procesos de negocio y generan registros muy valiosos para análisis posteriores. Esta sección se limita a hablar de la extracción de una base de datos relacional PostgreSQL, pero los conceptos que son desarrollados pueden ser extendidos para otras soluciones de bases de datos.

Para las bases PostgreSQL se tienen archivos de logs que registran los diferentes tipos de operaciones ejecutadas y son utilizados para mantener un registro histórico de todo lo que sucede en la base.

Para comenzar a guardar estos registros, si no se tiene ya configurado por defecto, se edita un archivo de configuración llamado postgres.conf. Dentro de este archivo, se debe quitar el comentario de la propiedad "loggin_collector" como se muestra en 2.1, y cambiar "off" por "on". Luego de guardar este cambio, solo queda reiniciar el servicio y se comenzaran a escribir los logs.

Listing 2.1: Configuración Postgres

```
1 #loggin_collector = off
2 loggin_collector = on
```

De esta forma se comenzará a registrar todo lo que sucede en la base. Estos registros se almacenan dentro de la carpeta "pg_log" por default aunque también pueden ser almacenados en una ruta configurable. Entre las configuraciones se tiene:

- log_directory: Se setea una ruta para almacenar los logs.
- log_filename: Se setea un nombre para los logs.
- log_rotation_size: Se setea el tamaño de los logs. Esta configuración es muy importante para disminuir la cantidad de archivos, ya que si tienen un tamaño muy bajo se crearan más archivos para guardar las operaciones.
- log_statement: Se setea el nivel de detalle de los logs. Entre las alternativas se tienen: none, ddl, mod, all.

En un contexto donde se quiere extraer qué es lo que sucede en cada interacción con la base de datos quizá la configuración más importante es "log_statement". Si está configurada en "none" esto significa que ningún statement será almacenado en los logs, por lo que todas las operaciones como inserts, updates o deletes no se verán reflejadas en los logs. Para la extracción la mejor configuración es "mod" que permite guardar los datos de dichas operaciones y no almacena los datos de las operaciones de selects, como lo hace "all".

2.2. Data Warehouse

Un data warehouse [4] es un repositorio para datos históricos, integrados y consistentes. Está equipado con herramientas que ofrecen la oportunidad de extraer información confiable para ser usada como soporte en el proceso de toma de decisiones. Involucran procesos de extracción de datos relevantes de los sistemas de información de una organización, transformaciones de datos, integraciones y eliminación de inconsistencias para almacenarlos en un sistema que provee a los usuarios finales el acceso a los datos para generar complejos análisis de datos y consultas predictivas.

Generalmente, se comienza con una fase de diseño conceptual donde, a partir de la realidad a trabajar, se definen diferentes dimensiones que cumplen relaciones particulares para la realidad. Luego, se utilizan las relaciones entre las dimensiones para obtener métricas sobre el conjunto de datos.

2.2.1. Dimensiones

El concepto de multidimensionalidad es clave para los data warehouse. En los últimos años las bases de datos multidimensionales han generado gran cantidad de investigaciones e interés en el mercado debido a que son fundamentales para muchos sistemas de toma de decisiones, como los data warehouses.

Normalmente, cada dimensión tiene asociada una jerarquía con niveles de agregación, usualmente llamados jerarquía de roll-up. Las jerarquías de roll-up agrupan valores de los niveles de agregación de diferentes formas. En la Figura 2.7 se puede ver un ejemplo simple de jerarquías construidas a partir de productos y tiendas. Los productos son clasificados en tipos y los tipos son clasificados en categorías. Mientras que las tiendas son clasificadas por ciudad y las ciudades por estados. En el final de cada jerarquía se tiene un nivel “falso” que incluye todos los valores de las dimensiones.

2.2.2. Cubos

Cada dimensión por separado no es de gran valor para un data warehouse, lo que si genera gran aporte son las relaciones entre dos o más de ellas y las métricas que se pueden obtener dadas estas combinaciones. Cuando se combinan distintas dimensiones para obtener métricas se genera lo que es llamado cubo y es la pieza fundamental de los data warehouses. En la Figura 2.8, siguiendo con el ejemplo anterior de productos y tiendas, se puede ver un cubo formado por dichas dimensiones y la dimensión de fechas (dimensión usual en un data warehouse) que tienen como métrica cantidad y los ingresos. En el ejemplo se ve que si se toma la tienda “EverMore” con el producto “Shiny” se tienen una cantidad de 10 y unos ingresos de 25 para la fecha “04/05/08”.

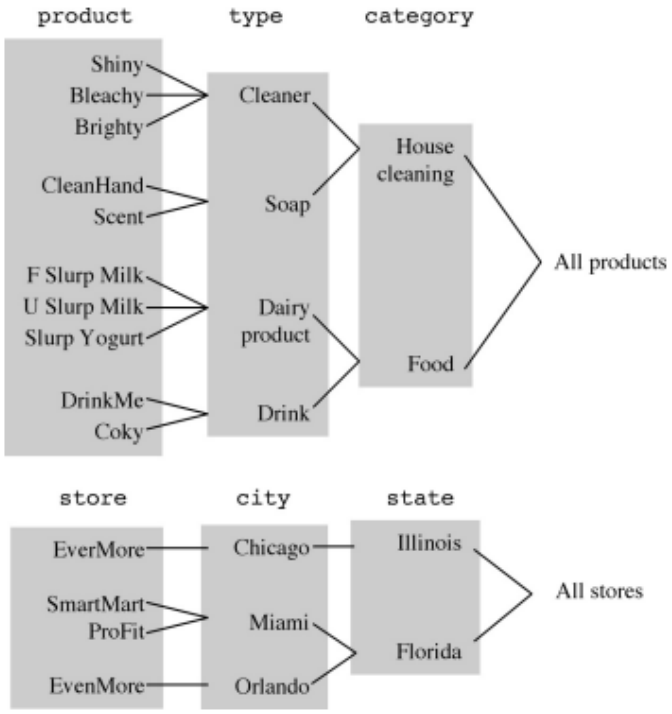


Figura 2.7: Dimensiones de un data warehouse, extraído de [4]

Para una realidad se pueden generar tantos cubos como sean necesarios, dependiendo de las combinaciones de dimensiones que se quieran utilizar y las métricas que se quieran obtener con cada una de ellas. Las combinaciones generadas deben aportar resultados que ayuden en algún sentido al análisis que se quiere realizar sobre la realidad trabajada.

2.2.3. Tabla de hechos

La tabla de hechos es, básicamente, como se construye un cubo con respecto a su implementación en una base de datos. La representación más común es denominada esquema de estrella, donde se tiene una tabla principal con foreign keys hacia las dimensiones y métricas específicas para cada combinación. Siguiendo con el ejemplo de tiendas y productos, en la Figura 2.9 se tiene un ejemplo de este tipo de esquema donde se crea una tabla llamada “Sales Facts” y que tiene foreign keys hacia las tablas de las dimensiones.

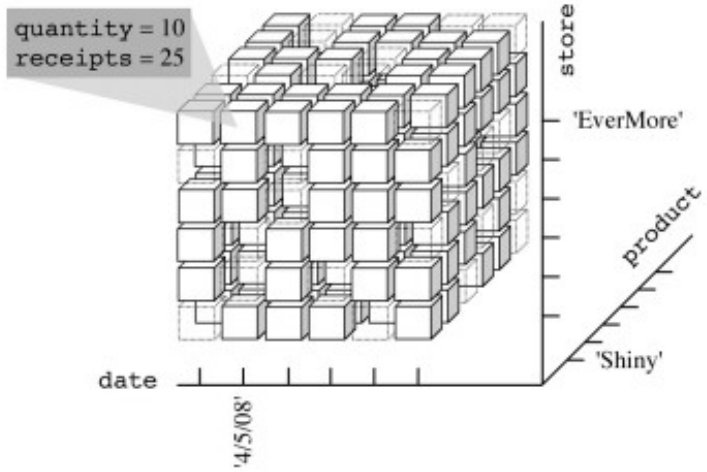


Figura 2.8: Relaciones Dimensionales, extraído de [4]

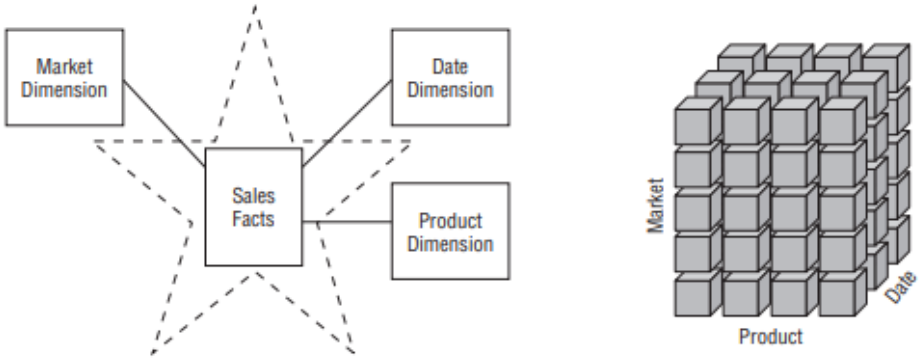


Figura 2.9: Esquema de estrella para tabla de hechos, extraído de [4]

Este esquema es la forma más simple de implementar un data warehouse y, en general, no es necesario la elaboración de estructuras más complejas. Para su implementación comúnmente se utilizan herramientas de ETL (Extrac, Transform and Load) y plataformas que permitan cargar cubos y realizar análisis.

2.2.4. Sistemas de data warehouse

Existen diferentes sistemas de data warehouse, que ofrecen una diversa gama de herramientas para el manejo de extracción, carga y visualización de los datos. Entre los sistemas de data warehouse se encuentra Pentaho, de la empresa Hitachi Ventara, que permite la integración de datos en una plataforma moderna para acceder, visualizar y explorar fácilmente los datos que se quieren analizar. Puede ser utilizado como una suite completa o como componentes individuales a los que se puede acceder localmente o en la nube. Entre sus herramientas se tiene Pentaho server y Pentaho Kettle.

Pentaho server

Pentaho server ofrece una plataforma donde se puede definir diferentes fuentes de datos y hacer análisis o crear reportes. Existen una gran cantidad de plugins que pueden ser instalados en la plataforma, algunos gratuitos y otros pagos, que permiten a los usuarios poder manejar los datos de una manera sencilla. En la plataforma se pueden crear nuevas fuentes de datos en “Manage Data Sources” o crear nuevas consultas o reportes con la opción “Create New”.

Para facilitar el análisis de datos existen múltiples plugins que pueden ser instalados en el servidor. Entre ellos se tiene el llamado Saiku Analytics, que permite analizar cubos diseñados y cargados previamente en el servidor, o cargar una nueva fuente de datos y crear definición del cubo. Luego de cargado el modelo, el plugin permite hacer diferentes cruzamientos entre las dimensiones y generar gráficas a partir de las métricas obtenidas. Un ejemplo se puede ver en la Figura 2.10 donde se analizan diferentes productos y territorios, viendo por cada combinación la cantidad que existen.

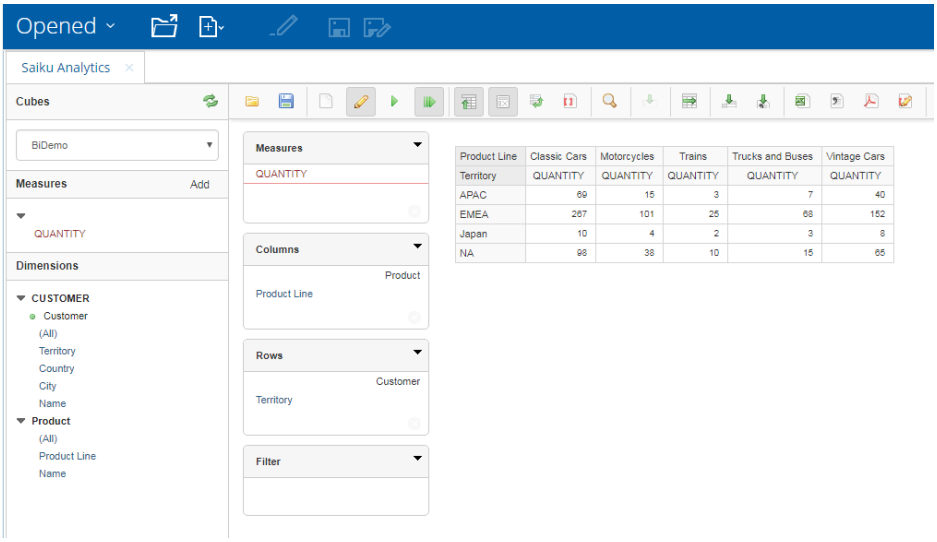


Figura 2.10: Pentaho Server: Saiku Analytics Plugin

Pentaho Data Integration

Pentaho Data Integration es una herramienta de ETL que permite acceder e integrar datos de cualquier fuente y enviarlos a diferentes destinos, como por ejemplo una base de datos, todo desde una herramienta gráfica intuitiva y fácil de usar. En la figura 2.11 se puede ver un ejemplo de una transformación simple, donde se toma como fuente de datos dos tablas distintas, de una o más bases de datos, se le aplican una serie de transformaciones y chequeos, y se hace una unión por clave de ambas fuentes para finalmente insertar los resultados en una nueva tabla.

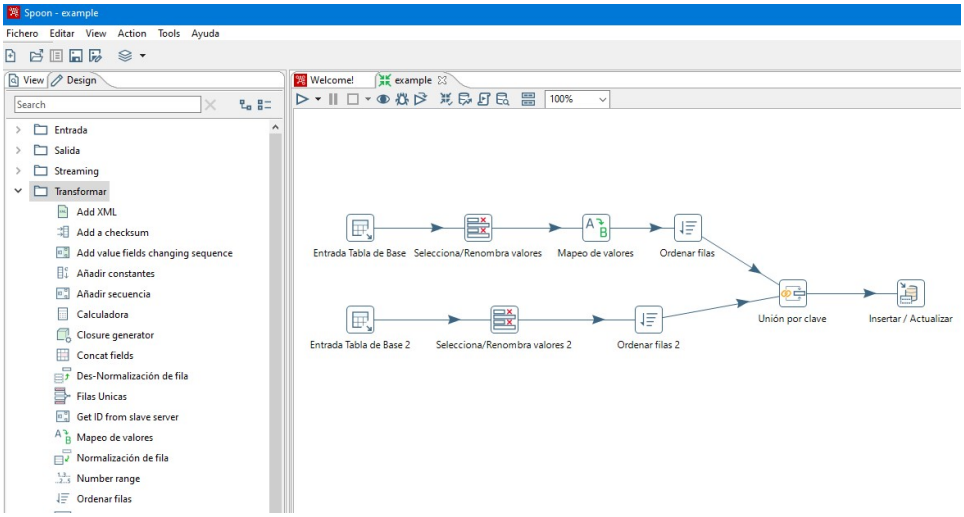


Figura 2.11: Pentaho Data Integration

Un claro ejemplo de una transformación es obtener datos de distintas fuentes para generar una dimensión del data warehouse u obtener más de una dimensión para completar la tabla de hechos con sus respectivas métricas. En el caso de los productos podrían tomarse de varias fuentes, como por ejemplo, varios supermercados. Con estas fuentes se puede obtener la lista completa de productos, transformarlos y cargarlos en la dimensión.

3 Análisis del Problema

En esta sección se analizará el problema presentado y las diferentes partes que se necesitan desarrollar para lograr la solución.

Los tópicos a explicar son:

- Introducción al problema.
- Análisis de trabajo previo.
- Análisis de implementación.
- Resumen de implementación.

3.1. Introducción al problema

Durante la ejecución de los procesos de negocios se producen interacciones con distintos sistemas, por esta razón, los datos que un proceso genera están distribuidos y son difíciles de analizar. La complejidad del problema surge al intentar extraer los datos y relacionarlos de manera que tengan sentido a la hora de realizar un análisis.

En la Figura 3.1 se puede ver un diagrama que muestra las relaciones entre los componentes que forman parte de la ejecución de un proceso de negocio. Estos componentes son:

- BPMS: Implementación y ejecución del proceso de negocio.
- Otros sistemas y servicios: Sistemas y servicios externos que interactúan con el proceso.
- Base de datos del proceso: Registro de todo lo que se realiza en el proceso, como tareas realizadas, procesos ejecutados y variables ingresadas.
- Base de datos de la organización: Registro de todos los datos de la organización, los cuales el proceso o sistemas externos puede modificar.

El BPMS registra la definición y ejecuciones del proceso en su base de datos, pero también genera información que impacta en la base de datos organizacional. A su vez, se tienen distintos sistemas o servicios externos que podrían impactar en la base de datos organizacional a partir de su participación en la ejecución del proceso.

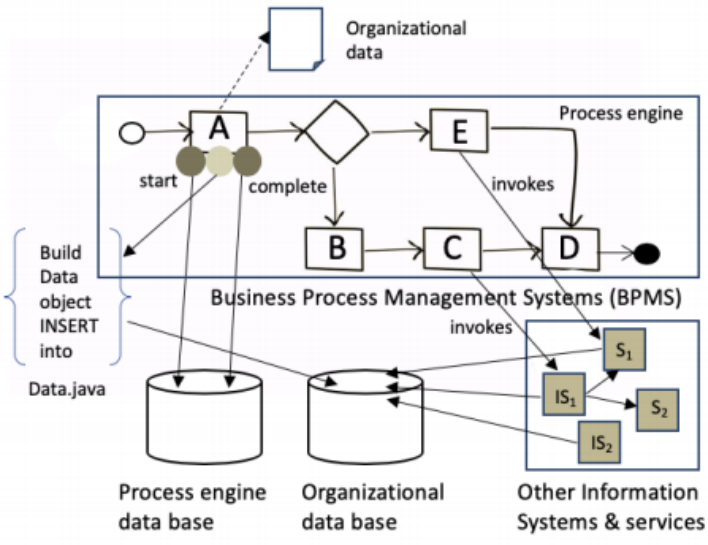


Figura 3.1: Relaciones de un proceso, extraído de [5]

Para combatir con las limitantes que se tienen al momento de relacionar los datos se plantea utilizar un modelo a partir del cual se busca integrar los datos extraídos de las bases organizacionales y de los procesos. La idea general se puede ver en la Figura 3.2, donde con la combinación de las bases organizacionales y de procesos se genera un modelo integrado que sirve como fuente de datos para la generación de un log extendido o la construcción de data warehouse.

La primera parte está compuesta por la extracción de los datos de la base organizacional y del proceso. Los datos extraídos se deben cargar en una implementación del metamodelo, donde posteriormente, se debe ejecutar un algoritmo de matcheo que los una. A partir de este metamodelo se plantean realizar dos acciones, por un lado, se puede correr un proceso de extracción y carga para obtener un data warehouse donde analizar los datos y, por otro lado, se puede generar un log extendido con el cual ejecutar algoritmos de process mining.

En este proyecto se limita a desarrollar soluciones para la extracción y carga desde las bases organizacionales y de procesos, la implementación del metamodelo en una base de datos, el algoritmo de matcheo y por último la carga y diseño del data warehouse. Como se puede apreciar, no entra en el alcance del proyecto la generación de un log extendido y su posterior análisis.

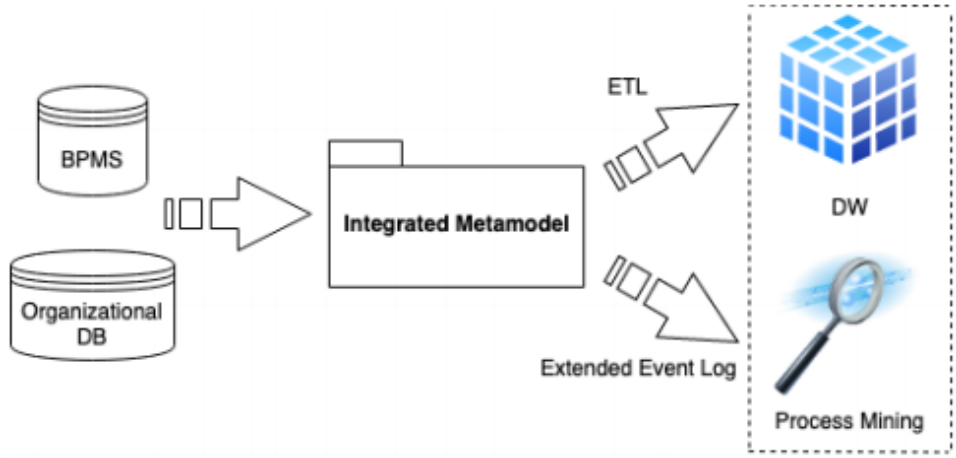


Figura 3.2: Modelo integrador, extraído de [5]

En las siguientes secciones se analizarán las partes que deben ser desarrolladas para poder llevar a cabo la solución propuesta. Las partes se separan en un análisis del metamodelo a utilizar, la extracción y carga de los datos necesaria para poder cargar dicho metamodelo correctamente y, finalmente, cómo será implementado el data warehouse.

3.2. Análisis de trabajo previo

Para comenzar, como se habló en secciones previas, es necesario un modelo bien definido para generar una conexión exitosa entre los datos. Para ello se toma como referencia el metamodelo propuesto en [5], el cual no ha sido implementado aún en ninguna base de datos y por ende, antes de este proyecto, no era posible generar un modelo como este a partir de la ejecución real de un proceso.

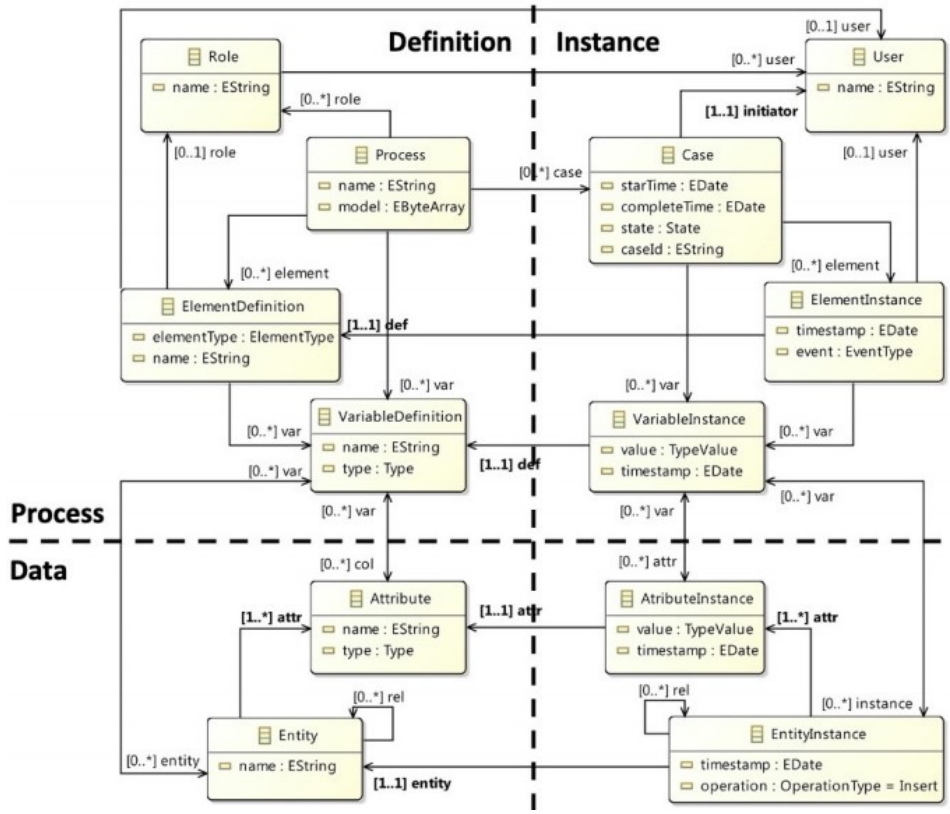


Figura 3.3: Metamodelo General

En la Figura 3.3 se puede ver la estructura del metamodelo definido. El metamodelo está seccionado en cuatro cuadrantes, separado horizontalmente por los cuadrantes de proceso y de datos, y verticalmente por definiciones e instancias. Estos cuadrantes se pueden ver más claramente en la Figura 3.4. A continuación se detalla cada cuadrante por separado para una mejor comprensión.

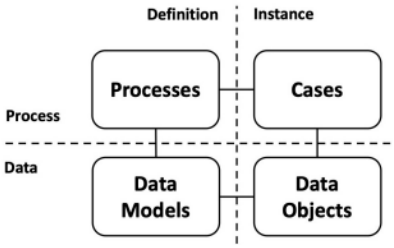


Figura 3.4: Cuadrantes de metamodelo

Definición de procesos

Por un lado, el cuadrante de definición de procesos del modelo representa todos los datos provenientes del proceso con respecto a su definición. Las diferentes partes que componen este cuadrante son:

- Definiciones de procesos.
- Definiciones de tareas, como tareas de usuario, tareas de mensajes, etc.
- Definiciones de roles.

Instancias de procesos

Por otro lado, el cuadrante instancias de procesos del modelo representa todos los datos con respecto a sus instancias. En este caso, se posee todos los datos provenientes de las ejecuciones de los procesos. Esto significa que son datos dinámicos y las diferentes partes que componen este cuadrante son:

- Instancias de procesos con su fecha de comienzo, fecha de finalización y su estado.
- Tareas ejecutadas para las instancias de procesos con su tiempo y tipo de evento.
- Valores de las variables definidas durante la ejecución.

Definición de datos

El cuadrante de definición de datos, como su denominación lo indica, refiere a la definición del esquema de la base de datos organizacional. Entre ellos se tiene:

- Definiciones de las diferentes tablas de las bases de datos.
- Definiciones de los atributos que componen las tablas.

Instancias de datos

Por otro lado, el cuadrante de instancias de datos refiere a los datos de la base organizacional,

compuestos por:

- Eventos que ocurrieron en la base, inserts, deletes o updates con sus respectivos tiempos.
- Valores de atributos insertados, actualizados o borrados sobre la base de datos.

Finalmente, entre los procesos y los datos se puede identificar una conexión. Dicha conexión es la clave para poder realizar un análisis sobre las contrapartes y relaciona las variables de un proceso con los atributos de la base de datos con respecto a su definición e instancias. Como también, las relaciones entre variables del proceso con las entidades de las bases de datos.

3.3. Análisis de implementación

En esta sección se desarrollarán las distintas partes que se implementan en este proyecto, implementación del metamodelo en base de datos, extracción y carga, algoritmo de matcheo e implementación y carga del data warehouse.

3.3.1. Implementación del metamodelo en base de datos

A partir de la definición conceptual del metamodelo presentada en la sección anterior es necesario definir un esquema de base de datos que lo implemente a partir de tablas y relaciones.

Para ello, se representa cada componente como una tabla y para modelar sus relaciones se deben tomar decisiones de diseño que sean fieles a las agregaciones representadas. Por ejemplo, para casos en que se tienen relaciones de $1...N$ se podrían agregar tablas intermedias que a priori no están representadas en el esquema conceptual.

3.3.2. Extracción y Carga

Tomando como referencia el metamodelo explicado en la sección anterior es inteligente dividir esta sección en tres partes, extraer los datos del proceso, extraer los datos de la organización y generar las relaciones entre las partes a partir de un algoritmo de matcheo. Para comenzar se analizará la extracción de los datos del proceso.

Extraer datos del proceso

Para este proyecto se optó por utilizar el sistema de gestión de procesos llamado Activiti. Como se vio en la sección 2.1.2, Activiti tiene una estructura particular en su base de datos donde guarda datos de las definiciones y las ejecuciones de los procesos. Como también para la definición del modelo se tiene un archivo de formato xml, en el cual se tienen los detalles

del modelo del proceso.

Como se puede ver en la Figura 3.5, tomando solo el cuadrante de Proccess Definition se tienen que cargar los componentes Role, Process, ElementDefinition y VariableDefinition.

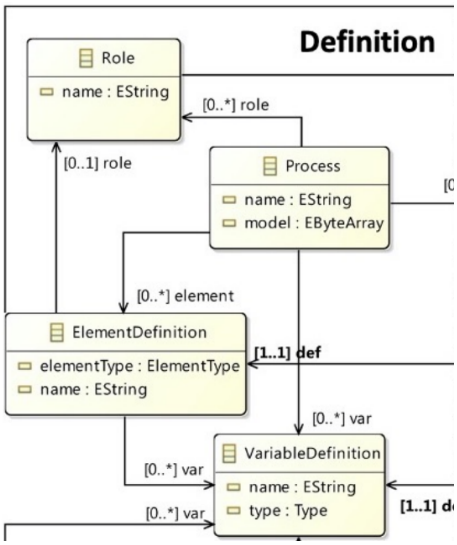


Figura 3.5: Cuadrante Process Definition

En las tablas vistas en la Figura 2.6 se puede apreciar que algunas pueden ser de utilidad para la carga, con lo que se obtiene:

- Las tablas llamadas ACT_RE_* contienen información de las definiciones de procesos, por lo que con ellas se puede completar la información del componente Process
- Las tablas llamadas ACT_ID_* contienen la información de los usuarios del sistema y sus roles. Por lo que se pueden utilizar para completar el componente Role del meta-modelo.

Con esto solo basta tener una fuente para completar ElementDefinition y VariableDefinition, cuyos datos no se encuentran entre los registros de la base de datos de Activiti. Para el primer caso se utiliza el xml del modelo visto en la Figura 2.3, este contiene todas las definiciones de tareas a ejecutar con sus respectivos responsables. Con esta información es suficiente para completar ElementDefinition y sus relaciones.

El caso de VariableDefintion es el único que no se puede obtener explícitamente a partir de las fuentes de datos que Activiti ofrece. Para llevarlo a cabo se utilizan las tablas de Activiti con el prefijo ACT_HI_* en las que se registra el histórico de todas las variables que se ingresaron en el proceso. A partir de estas tablas se agrupa por el nombre de las variables y por proceso

para saber todos los nombres de variables que participan en el proceso, y con ello se obtiene su definición.

Como se puede ver en la Figura 3.6, tomando solo el cuadrante de Proccess Instance se tienen que cargar los componentes User, Case, ElementInstance y VariableInstance.

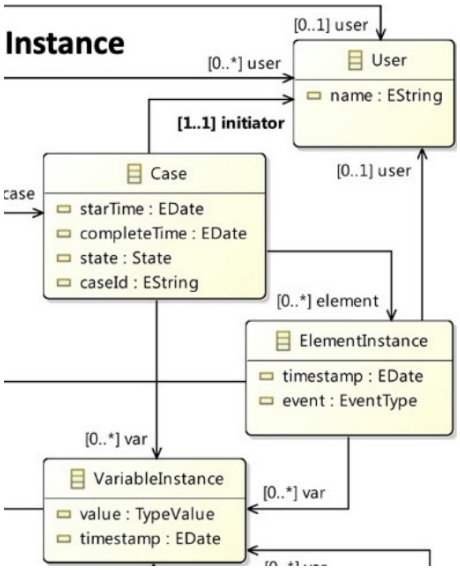


Figura 3.6: Cuadrante Process Instance

Las tablas de Activiti utilizadas para este caso son las siguientes:

- Las tablas llamadas ACT_HI_* contienen la información histórica de los procesos que se ejecutaron, por lo que se puede obtener la información necesaria para cargar los componentes VariableInstance, ElementInstance y Case.
- Las tablas llamadas ACT_ID_* contienen la información de los usuarios del sistema. Por lo que se pueden utilizar para completar el componente User.

Extraer datos de la organización

Los datos de la organización se encuentran distribuidos en tablas de la base de datos, pero estos son solo las últimas actualizaciones, por lo que se pierden los datos históricos. Para poder extraer los datos históricos, que son de suma importancia para el análisis, se pueden utilizar los logs de la base de datos.

Como se vio en la sección 2.1.3, se pueden configurar las bases de datos para registrar en los logs toda la información de todas las operaciones que se efectuaron históricamente. Los logs

detallan la hora y fecha de las operaciones, los nombres de las tablas, los atributos y el tipo de operación.

Por un lado, el cuadrante de definiciones de los datos se puede cargar a partir de analizar la metadata de la base de datos organizacional, donde se encuentra la información referente a los nombres de las tablas y sus atributos, como también todas las foreign keys que se tengan definidas.

Mientras que por otro lado, el cuadrante de instancias se puede cargar a partir de los registros que se tienen en los logs, donde se tiene registrado cada operación, las columnas afectadas y sus valores.

3.3.3. Algoritmo de matcheo

Finalmente, luego de completar la carga de todos los cuadrantes, se necesita solucionar el problema de las relaciones entre los datos del proceso y de la organización. Para ello, se propone la implementación de un algoritmo que descubra y cargue las relaciones entre los datos.

Este algoritmo de matcheo necesita identificar las relaciones entre:

- AttributeInstance - VariableInstance: Esta relación asocia cada instancia de atributos de los datos con su respectiva instancia de variable del proceso. En general, el valor de ambas debería coincidir.
- Attribute - VariableDefinition: Esta relación asocia cada definición de atributo de los datos con su respectiva definición de variable del proceso. En este caso, los nombres y tipos pueden variar.
- VariableDefinition - Entity: Esta relación asocia las definiciones de variable del proceso con las entidades en la cuales participan.
- VariableInstance - EntityInstance: Esta relación asocia las instancias de variables del proceso con las instancias de entidades que participaron.

Un algoritmo sencillo que puede dar una buena solución parte de la comparación por valor entre las instancias de variable del proceso y las instancias de atributo de los datos. Como en la mayoría de los casos estos valores van a coincidir, se pueden identificar qué valores de variables coinciden con los valores de atributos. Esta solución es válida suponiendo que los valores de los datos no se repiten, pero es fácil reconocer que el valor de los datos se van a repetir con frecuencia a lo largo de un proceso.

Para solucionar el problema de los datos con un mismo valor se propone utilizar los timestamps de ambas partes. Se define una ventana de tiempo configurable a partir del timestamp de la instancia de variable y se compara contra las instancias de atributos cuyo timestamp está

contenido dentro de ese período de tiempo. De esta forma los datos que coinciden tienen una alta probabilidad de que sean los correspondientes.

3.3.4. Data Warehouse

A partir del metamodelo se propone un data warehouse, que a priori sea genérico. Con genérico se quiere decir que no importa la realidad que se trabaje, el data warehouse puede ser cargado únicamente a partir del metamodelo y proveer un análisis productivo.

Como se vio en la sección 2.2, para la creación de un data warehouse se requiere definir las dimensiones, tablas de hechos y métricas. Como se quiere lograr una solución genérica, todas estas partes deben ser pensadas para sacar provecho sin necesidad de basarse en una realidad específica.

Para ello se propone generar dimensiones a partir de los cuatro cuadrantes del metamodelo, de los usuarios, y de tiempo. Donde tiene sentido plantear las jerarquías a partir de las relaciones que existen dentro de cada cuadrante, como por ejemplo, VariableInstance es el nivel más bajo de una dimensión, teniendo como padre a ElementInstance, la cual tiene como padre a Case. Esta dimensión se puede definir correctamente, ya que todas las instancias de variables participan en una sola instancia de elemento y cada instancia de elemento tiene una sola instancia de proceso asociada.

Por su parte, las métricas a definir tienen que ser pensadas también de forma general. Las que se proponen en este proyecto serán las de duración de un proceso, duración de una tarea, y los valores de las variables. Las dos primeras son pensadas para realizar análisis de tiempo en procesos, como por ejemplo tareas retrasadas o procesos que demoraron más de lo debido. Como, por otro lado, con la métrica de valor se intenta analizar variables un poco más particulares como puede ser el monto en un proceso de solicitud de becas. Para todas ellas se pueden definir promedios u otro cálculo que pueden aportar valor.

3.4. Resumen de implementación

En esta sección se presenta un resumen de lo que se debe implementar como parte del proyecto y referencias a las secciones en donde se detalla cómo se implementaron. En resumen, se debe realizar:

- Implementación de metamodelo en base de datos, detalles de este paso se pueden ver en la sección 4.1.
- Implementación de algoritmo de extracción y carga de los datos del proceso, detalles de este paso se pueden ver en la sección 4.2.
- Implementación de algoritmo de extracción y carga de los datos de la base organiza-

cional, detalles de este paso se pueden ver en la sección 4.3.

- Implementación de algoritmo de matcheo, detalles de este paso se pueden ver en la sección 4.4.
- Implementación de data warehouse, detalles de este paso se pueden ver en el capítulo 5.

4 Solución planteada para integración de datos

En el capítulo anterior se describió el problema a resolver y los desafíos que plantea. Por lo que en este capítulo se desarrollará a detalle la solución implementada, explicando paso a paso las decisiones que se tomaron y cómo estas afectaron a lo largo del proyecto.

Los tópicos a explicar son:

- Representación de metamodelo en base.
- Carga de proceso en metamodelo.
- Carga de base organizacional en metamodelo.
- Matcheo de datos.

A continuación se expondrá información detallada en secciones separadas por los tópicos antes mencionados.

4.1. Metamodelo en una base de datos relacional

En esta sección se detallará la implementación del metamodelo en una base de datos relacional, como se podrá observar se respeta el diseño conceptual presentado en la sección 3.2.

Con la intención de mostrar de forma clara la implementación en este documento se decide separar la presentación en tres subsecciones. Se comienza con la implementación de los cuadrantes superiores e inferiores, terminando con la implementación de la conexión entre ellos.

4.1.1. Implementación de Definición e Instancias de Procesos

En esta sección se mostrará en detalle la implementación de los cuadrantes Definición de Proceso e Instancias de Proceso.

Dicha agrupación de tablas se utilizan para la definición e instancias de los datos generados por un proceso. Como se ve en la Figura 4.1 se respeta la definición del metamodelo, por un lado se tienen tablas con nombres que terminan en “Definition” en donde, como su nombre lo indica, se guardará la definición de los datos extraídos del proceso. Por otra parte, se tienen la terminación “Instance” que serán utilizadas para guardar los valores de los datos extraídos. Asimismo, también se tienen diferentes tablas que representan datos del dominio de los procesos como son las tablas user, roles, process, y state.

Adicionalmente, pueden observarse los atributos en donde se guardarán los datos extraídos y las foreing key definidas, respetando las relaciones del metamodelo.

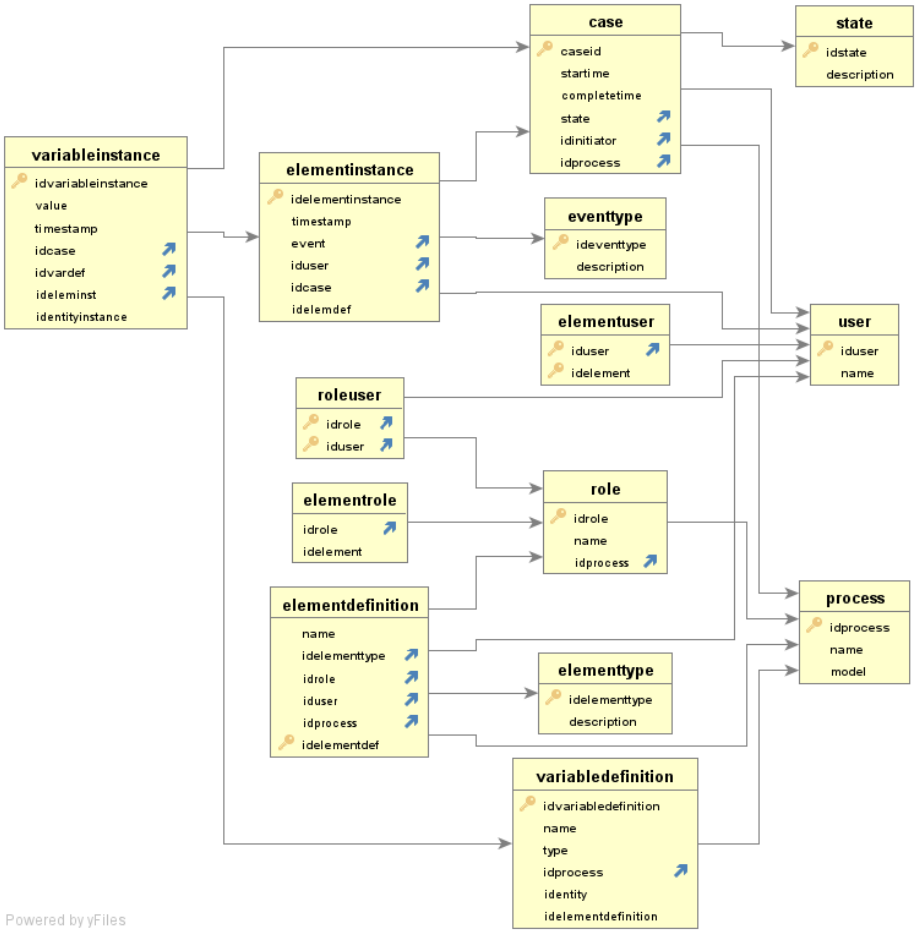


Figura 4.1: Tablas para datos del proceso

4.1.2. Implementación de Definición e Instancias de Datos

En esta sección se mostrará en detalle la implementación de los cuadrantes Definición de Datos e Instancias de Datos.

Dicha agrupación de tablas se utilizan para la definición e instancias de los datos utilizados en la organización. Como se ve en la Figura 4.2 se respeta la definición del metamodelo, por un lado se tienen tablas que contienen la definición de los datos de la organización, como lo son “entity” y “attribute”, y por otro lado se tienen las tablas que permiten almacenar las instancias de los datos, como lo son “entityinstance” y “attributeinstance”. También se tienen

diferentes tablas de relación n-m para modelar las relaciones entity-entity o entitiInstance-entityInstance.

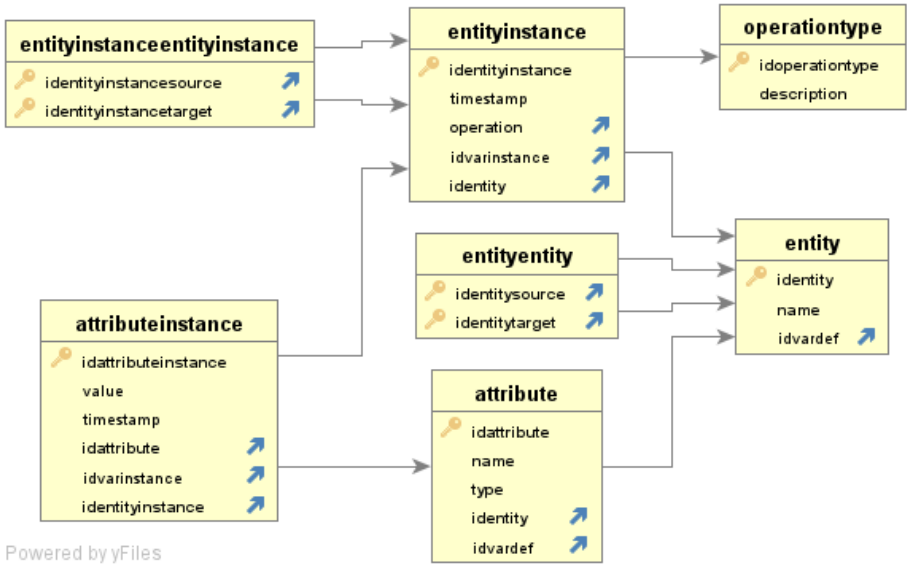


Figura 4.2: Tablas de base organizacional

4.1.3. Implementación de relaciones

En esta sección se mostrará en detalle la implementación para las tablas que contienen relaciones entre datos del proceso y de la organización.

Dicha agrupación de tablas ya fueron presentadas anteriormente, no obstante, el objetivo de esta sección es detallar minuciosamente la relación entre las dos partes. Como se ve en la Figura 4.3 se respeta la definición del metamodelo. Se tienen las siguientes relaciones:

- VariableInstance - AttributeInstance: Indica la relación entre las instancias de variables y atributos que machearon.
- VariableInstance - EntityInstance: indica la relación de la variable en el proceso y la operación que se ejecutó en la base organizacional.
- VariableDefinition - Attribute: representa la definición de la relación, o sea qué atributo se relaciona como variable. Sin considerar una instancia particular.
- VariableDefinition - EntityDefnition: indica la relación de la definición de variable en el proceso y la tabla con la que se relaciona en la base organizacional.

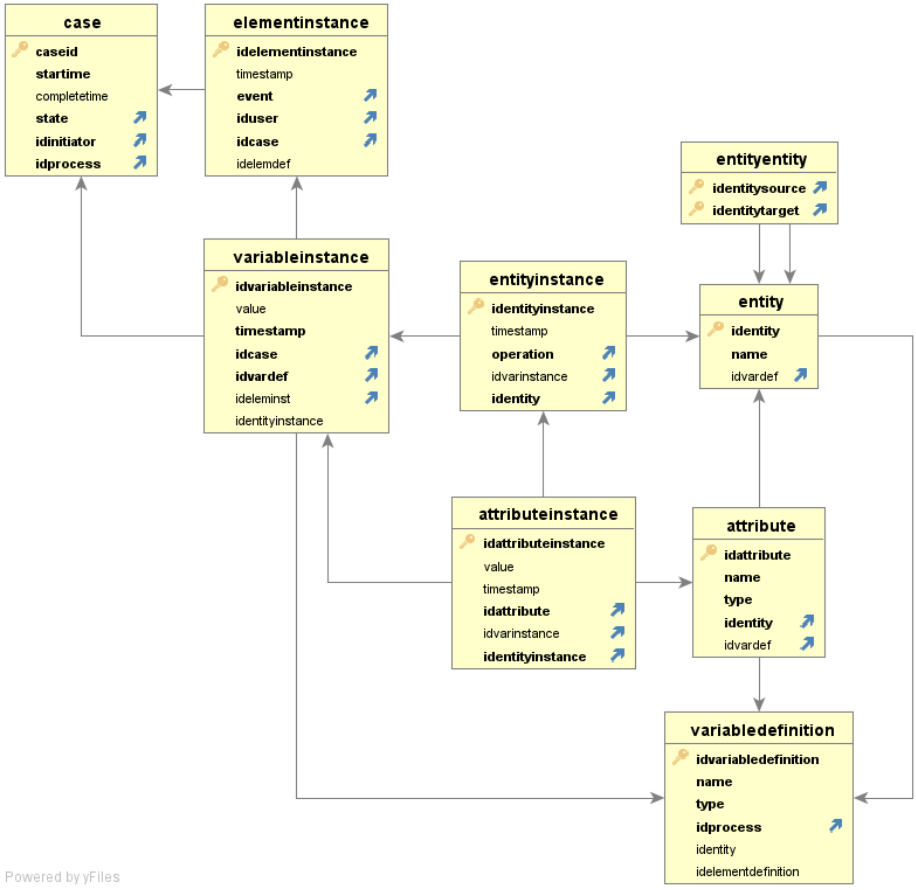


Figura 4.3: Relaciones entre tablas del proceso y la organización

Se presenta la tabla 4.1 en donde se puede ver cada una de estas relaciones, modeladas como foreing keys, mostrando la tabla de referencia junto a su columna y por último la tabla referenciada con su columna.

fktable_name	fkcolumn_name	pktable_name	pkcolumn_name
attribute	identity	entity	identity
attribute	idvardef	variabledefinition	idvariabledefinition
attributeinstance	idattribute	attribute	idattribute
attributeinstance	identityinstance	entityinstance	identityinstance
attributeinstance	idvarinstance	variableinstance	idvariableinstance
case	idprocess	process	idprocess
case	state	state	idstate
case	idinitiator	user	iduser
elementdefinition	idelementtype	elementtype	idelementtype
elementdefinition	idprocess	process	idprocess
elementdefinition	idrole	role	idrole
elementdefinition	iduser	user	iduser
elementinstance	idcase	case	caseid
elementinstance	event	eventtype	ideventtype
elementinstance	iduser	user	iduser
elementrole	idrole	role	idrole
elementuser	iduser	user	iduser
entity	idvardef	variabledefinition	idvariabledefinition
entityentity	identitysource	entity	identity
entityentity	identitytarget	entity	identity
entityinstance	identity	entity	identity
entityinstance	operation	operationtype	idoperationtype
entityinstance	idvarinstance	variableinstance	idvariableinstance
entityinstanceentityinstance	identityinstancesource	entityinstance	identityinstance
entityinstanceentityinstance	identityinstancetarget	entityinstance	identityinstance
role	idprocess	process	idprocess
roleuser	idrole	role	idrole
roleuser	iduser	user	iduser
variabledefinition	idprocess	process	idprocess
variableinstance	idcase	case	caseid
variableinstance	ideleminst	elementinstance	idelementinstance
variableinstance	idvardef	variabledefinition	idvariabledefinition

Cuadro 4.1: Tabla de foreing keys

4.2. Carga de Procesos en metamodelo

Los cuadrantes superiores del metamodelo representados en la Figura 4.4 corresponde a los datos generados en las ejecuciones del BPMS. Por lo tanto para cargar esos datos se utilizan la base de datos del BPMS y el xml de su modelo.

Se implementa un módulo en Java 1.8 que automatiza la carga de los datos del proceso. Debido a las distintas restricciones de claves foráneas del metamodelo la implementación de la carga sigue el flujo que se desarrollará a continuación.

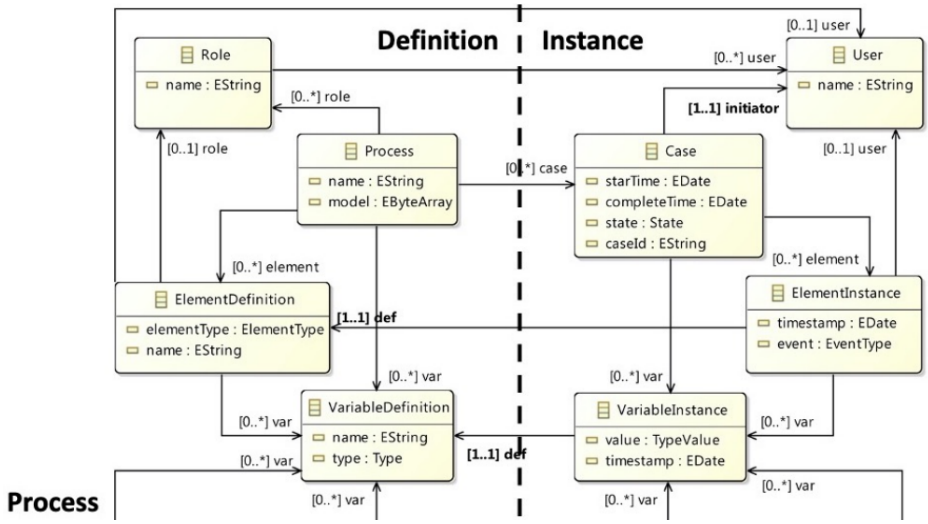


Figura 4.4: Cuadrantes de proceso del metamodelo

4.2.1. User y Role

Se comienza cargando los usuarios y sus roles, representados en el metamodelo con la Figura 4.5. Ambos datos son obtenidos a partir de la base de datos del BPMS consultando la tabla "act_id_membership" que contiene los nombres de los usuarios y sus respectivos grupos, la consulta se puede ver en 4.1.

Listing 4.1: Conulta para usuarios y roles

```
1 select * from act_id_membership;
```

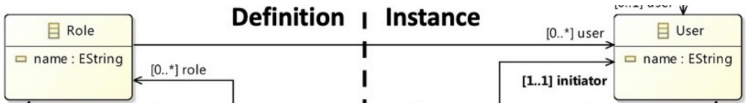


Figura 4.5: Usuarios y roles en el metamodelo

4.2.2. Process

Luego de tener los usuarios y roles se puede continuar con la carga de las definiciones de procesos. Para ello se utilizan dos tablas de la base del BPMS llamadas “act_re_procdef” y “act_ge_bytearray”, donde se tiene la información del proceso y el xml del modelo respectivamente. La consulta utilizada en este caso se muestra en 4.2:

Listing 4.2: Consulta para definición de procesos

```
select proc.id_, proc.name_, bytes.bytes_
from act_re_procdef proc
join act_ge_bytearray bytes
    on  proc.deployment_id_ = bytes.deployment_id_
    and proc.dgrm_resource_name_ = bytes.name_;
```

A partir de esta consulta se puede cargar la tabla Process del metamodelo, quedando de esta forma cargadas todas las definiciones de procesos existentes en el BPMS.

4.2.3. Case

A partir de la tabla de procesos se puede continuar con la tabla Cases, en la cual se cargan las instancias de procesos que se tienen en la base. Para ello se utiliza la tabla “act_hi_procinst” donde se tiene toda la información sobre las diferentes instancias que se han ejecutado. A partir de la consulta 4.3 se obtienen los valores necesarios.

Listing 4.3: Consulta para instancias de procesos

```
select id_,
    proc_def_id_,
    start_time_,
    end_time_,
    start_user_id_,
    delete_reason_
from act_hi_procinst;
```

Donde es importante evaluar si la instancia tiene delete_reason_ o no, con este valor, start_time_ y end_time_ se puede distinguir si el proceso está activo, completado o suspendido. Si se tiene delete_reason quiere decir que el proceso fue suspendido, en caso contrario si tiene end_time_ el proceso fue completado y por último si este valor tampoco existe el proceso se encuentra activo.

A partir de dicha consulta se carga la tabla Case con sus respectivas relaciones con su usuario de la tabla User y su definición de proceso de la tabla Process. Se puede ver esta sección del metamodelo en la Figura 4.6.

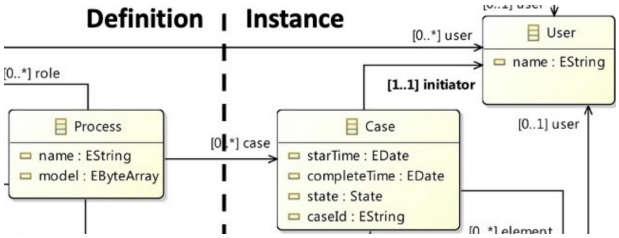


Figura 4.6: Process, case y user en el metamodelo

4.2.4. ElementDefinition

A continuación, se cargan las definiciones de tareas. Para este caso especial se utiliza el xml del modelo, ya que en la base de datos se guardan como un binario. A partir del xml se pueden obtener todos los nombres de las tareas del modelo, el proceso al que pertenecen y los usuarios que tienen acceso. Esta sección del metamodelo se puede ver en la Figura 4.7.

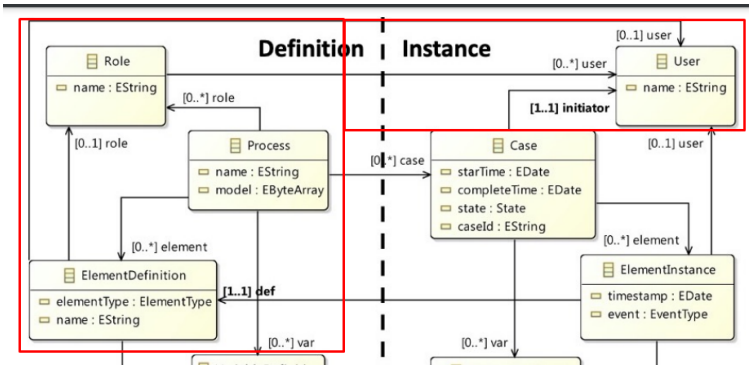


Figura 4.7: ElementDefinition en el metamodelo

En 4.4 se puede observar un ejemplo reducido de lo que es el xml que define el proceso en

BPMS. En este ejemplo se puede ver que se tienen todos los datos necesarios para recuperar las definiciones de elementos. En primer lugar, se tiene una etiqueta “process” de donde se recupera el identificador del proceso cargado previamente como se explica en la sección 4.2.2 y luego, en segundo lugar, una etiqueta “userTask” de donde se puede recuperar toda la información de una tarea de usuario. En particular interesa el identificador de la tarea, el nombre, y la lista de candidatos o roles que tienen permisos para ejecutarla.

Listing 4.4: Ejemplo de xml del proceso

```
<process id="id de proceso" name="nombre de proceso"
    isExecutable="true">
<userTask id="id de tarea" name="nombre de tarea" activiti:
    candidateGroups="lista de candidatos" activiti:formKey="
    formulario">
      <extensionElements>
        <activiti:taskListener event="create" class="
            listener1"></activiti:taskListener>
        <activiti:taskListener event="complete" class="
            listener2"></activiti:taskListener>
      </extensionElements>
    </userTask>
</process>
```

4.2.5. ElementInstance

Continuando con la carga se sigue con ElementInstance. A partir de la tabla “act_hi_actinst” se obtienen todos los valores necesarios para completar esta parte del metamodelo, a partir de la consulta 4.5.

Listing 4.5: Consulta para instancias de elementos

```
select task_id_,
    proc_def_id_,
    act_id_,
    assignee_,
    start_time_,
    end_time_,
    delete_reason_
from act_hi_actinst;
```

A partir de ellos se puede completar la tabla ElementInstance con todas sus relaciones. Para el caso de la relación con ElementDefinition se realiza la consulta que se muestra en 4.6 para obtener la definición a partir del id de instancia de proceso y el id de la actividad proveniente de dicha tabla.

Listing 4.6: Consulta para obtener una definición de elemento

```
select element
from Elementdefinition element
where element.idprocess = :processId
and element.idElementDefinition = :activitiId
```

Luego de esas consultas se puede completar satisfactoriamente el ElementInstance con User, Case y ElementDefinition. Esta sección del metamodelo se puede ver en la Figura 4.8.

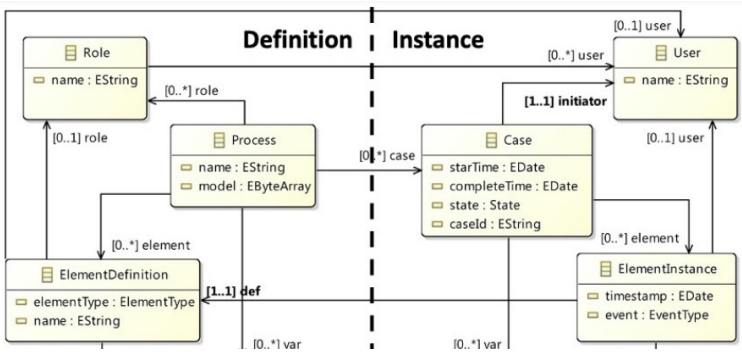


Figura 4.8: ElementInstance en el metamodelo

4.2.6. VariableDefinition

La tabla de VariableDefinition es la primera que presenta una consulta más compleja de realizar, debido a que no se tiene una tabla que tenga las definiciones de variables existentes en el proceso. Por ello, se debe utilizar la consulta que se muestra en 4.7.

Listing 4.7: Consulta para definición de variables

```
select aha.proc_def_id_,
    act_id_,
    name_,
    var_type_
from act_hi_actinst aha
join act_hi_detail ahd
    on aha.id_ = ahd.act_inst_id_
where var_type_ is not null
group by (aha.proc_def_id_, act_id_, name_, var_type_);
```

Donde, a partir de la tabla “act_hi_actinst” se obtiene la definición de proceso y a partir de la tabla “act_hi_detail” se obtienen el resto de los datos como el id de tarea, el nombre y tipo

de variable. Además, se debe realizar un group by de esos valores para obtener solo una vez cada variable, de esta forma se tiene la definición. Queda cargada la tabla VariableDefinition con sus relaciones a ElementDefinition y Process. Esta sección del metamodelo se puede ver en la Figura 4.9.

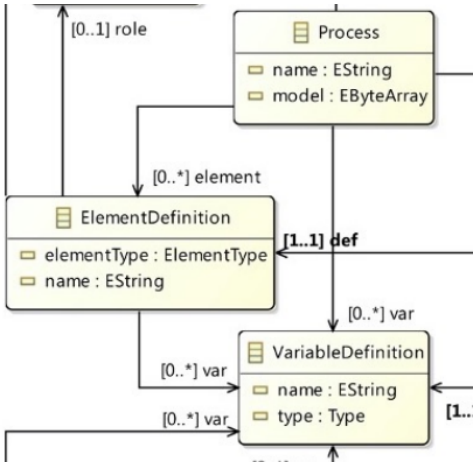


Figura 4.9: Carga de VaribaleDefinition en el metamodelo

4.2.7. VariableInstance

La tabla VariableInstance presenta un escenario más complejo a la hora de completar su relación con VariableDefinition. Esto se debe a que Activiti no provee un id para la definición de variables y, por lo tanto, en las instancias de variable no hay forma de referenciar su respectiva definición.

Para resolver este problema se comienza haciendo la consulta 4.8.

Listing 4.8: Consulta para definición de variables

```
select ahd.id_,
    proc_def_id_,
    act_id_,
    type_,
    name_,
    time_,
    text_,
    ahd.task_id_,
    ahd.proc_inst_id_
from act_hi_actinst aha
join act_hi_detail ahd
```

```
12      on aha.id_ = ahd.act_inst_id_
13  order by aha.proc_inst_id_ desc;
```

A partir de dicha consulta se pueden completar todos los campos directamente, menos el que hace referencia a su definición. Para ello se necesita la consulta 4.9.

Listing 4.9: Consulta para obtener definición de una variable

```
1  SELECT *
2  FROM variabledefinition variable
3  where variable.name = name AND
4      variable.idprocess = :idProcess AND
5      variable.idelementdefinition = :elementDef
```

Donde a partir del nombre de la variable, id de Process y el id de ElementDefinition se obtiene una única tupla correspondiente a la definición de variable que se está buscando. Se busca por estos tres valores, ya que una variable puede tener el mismo nombre para distintos procesos y tareas.

Luego de esta consulta, ya se tienen todos los datos necesarios para completar la carga de la tabla VariableInstance con sus relaciones a VariableDefinition, ElementInstance y Case. Esta sección del metamodelo se puede ver en la Figura 4.10.

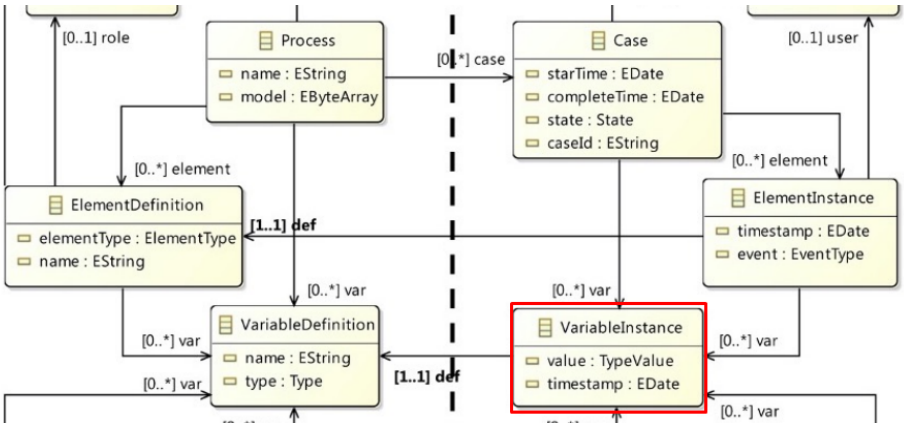


Figura 4.10: Carga de VaribaleInstance en el metamodelo

A partir de este momento se tiene toda la parte superior del metamodelo completamente cargada y se puede proseguir a completar la parte inferior a partir de los logs de la base organizacional.

4.3. Cargar de base organizacional en metamodelo

Este segundo proceso de carga es necesario para completar el resto del metamodelo, que corresponde al cuadrante de datos mostrado en la Figura 4.11. Para esta carga se utilizan los logs generados por la o las bases de datos organizacionales asociadas al proceso. Esto implica parsear un archivo de texto que posee inserts, updates y deletes con sus respectivos timestamps y atributos.

Se implementa un módulo en Java 1.8 que automatiza la carga de los datos de la base organizacional. Debido a las distintas restricciones de claves foráneas del metamodelo la implementación de la carga sigue el flujo que se desarrollará a continuación.

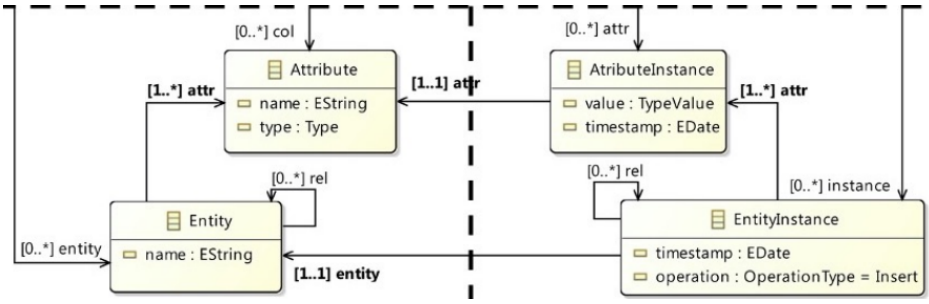


Figura 4.11: Carga de la base organizacional en el metamodelo

Lo primero a realizar es cambiar la configuración de la base de datos organizacional para que imprima el nivel de log necesario para poder extraer la información apropiada. Para ello se modifica el archivo data/postgresql.conf:

1. Descomentar la línea #loggin_collector = off y cambiar off por on. (Esto puede que ya esté hecho por defecto)
2. Descomentar la línea #log_statement = ‘none’ y cambiar ‘none’ por ‘mod’

De esta forma el log imprimirá todas las operaciones de insert, update y delete necesarias para cargar el metamodelo.

El proceso de carga comienza cargando el o los archivos de logs de la base de datos en el programa, se tiene uno o más archivos de logs dependiendo de su tamaño que está limitado a 20mb por default.

```
2021-05-13 23:38:18.459 -03 [6264] LOG:  ejecutar <unnamed>: insert into public.validation
(idapplication, idcourse, idvalidation) values ($1, $2, $3)
2021-05-13 23:38:18.459 -03 [6264] DETALLE:  parámetros: $1 = '5765', $2 = '3666', $3 = '22440'
2021-05-13 23:38:18.471 -03 [6264] LOG:  ejecutar <unnamed>: insert into public.validation
(idapplication, idcourse, idvalidation) values ($1, $2, $3)
2021-05-13 23:38:18.471 -03 [6264] DETALLE:  parámetros: $1 = '5765', $2 = '1555', $3 = '22441'
2021-05-13 23:38:47.658 -03 [17164] LOG:  ejecutar <unnamed>: insert into public.application
(amount, approved, date, notified, orden, idprogram, idstate, idstudent, idapplication) values
($1, $2, $3, $4, $5, $6, $7, $8, $9)
2021-05-13 23:38:47.658 -03 [17164] DETALLE:  parámetros: $1 = '2201', $2 = NULL, $3 =
'2021-05-13', $4 = NULL, $5 = NULL, $6 = '391', $7 = '1', $8 = '43346883', $9 = '5766'
```

Figura 4.12: Ejemplo de log

Un archivo de logs tiene una estructura particular, como se ve en la Figura 4.12. Se puede apreciar que en una línea contiene timestamp, operación, tabla asociada, los nombres de los atributos y en la siguiente línea los valores de los atributos a utilizar.

Lo primero que se hace con este formato de log es parseralo a una lista de objetos que respete la estructura presentada en 4.10.

Listing 4.10: Clase ParserLog

```
1 public class ParserLog{
2     private Date timestamp;
3     private String entityName;
4     private String entityOperationType;
5     private int entityOperationTypeId;
6     private Map<String, String> atributes;
7 }
```

Recorriendo cada línea del log se genera dicha lista de objetos donde se tienen todos los datos necesarios para completar el metamodelo. Para ello se tiene una lógica especial para diferenciar qué valores se deben insertar en cada tabla.

4.3.1. Entity y Attribute

A continuación, se cargan las entities y attributes de las tablas de la base organizacional. Esta sección del metamodelo se puede ver en la Figura 4.13

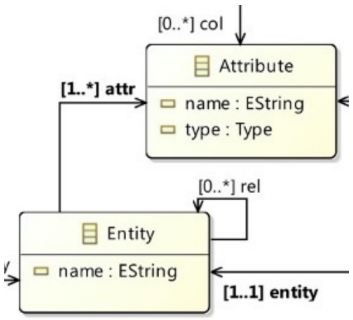


Figura 4.13: Entity y Attribute

Para realizar la carga de las entidades y sus atributos se obtiene la metadata de la conexión con la base de datos organizacional. Luego como se muestra en el pseudocódigo 4.11 se utilizan los métodos getTables(), getImportedKeys() y getColumns() de la clase DatabaseMetaData para obtener las tablas, sus columnas y las foreign keys.

Listing 4.11: Pseudocódigo carga de Entity y Attribute

```
1  //Obtener entidades de la base organizacional
2  for (table in getTables()){
3      //obtener atributos de la entidad
4      for (attribute in getColumn(table)){
5          attributeList.add(attribute);
6      }
7      //obtener relaciones entre entidades
8      for (fk in getImportedKeys(table)){
9          entityTarget = getEntity(fk);
10         fkEntityList.add(entityTarget);
11     }
12     entity = createEntity(attributeList, fkEntityList);
13 }
```

Con esto se pueden crear las entidades y atributos, como también las relaciones entre entidades, teniendo en cuenta que los datos de las tablas se mapean a una entidad, las columnas a atributos y las foreign keys a relaciones entre entidades.

4.3.2. EntityInstance y AttributeInstance

A continuación, se cargan las entities y attributes de las tablas de la base organizacional. Esta sección del metamodelo se puede ver en la Figura 4.14

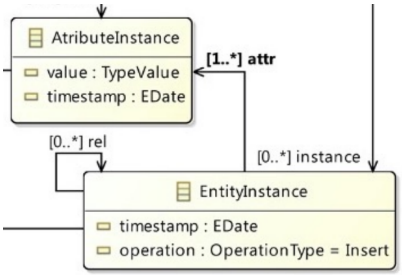


Figura 4.14: EntityInstance y AttributeInstance

Para esta carga se utiliza el log de la base organizacional. Para ello se sigue un flujo como el que se representa en la Figura 4.15.

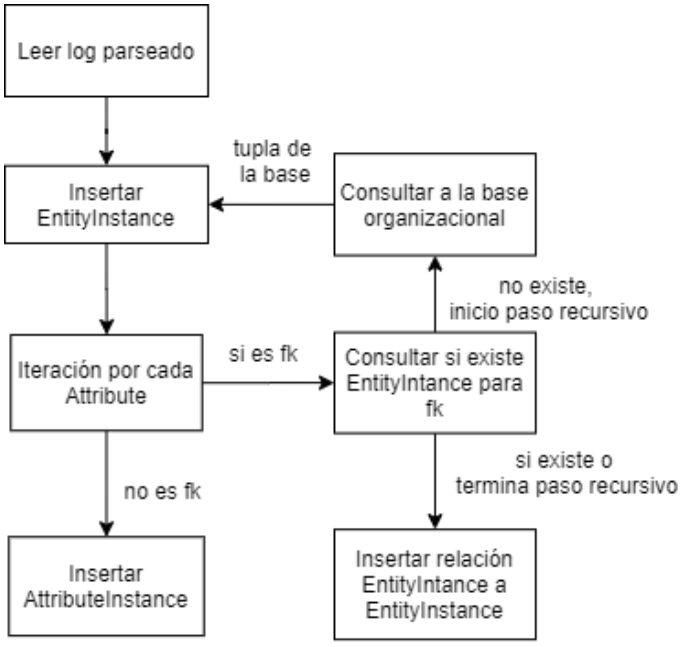


Figura 4.15: Carga del log

Se comienza leyendo un objeto de la clase ParserLog y a partir de este se inserta una nueva EntityInstance con su respectiva operación y timestamp. Además, se hace una consulta sobre la tabla Entity para obtener la relación y poder cargar la clave foránea hacia dicha tabla.

Luego de que se tiene la EntityInstance, se hace una iteración por cada uno de los Attribute que la Entity contiene. En este paso se dividen dos caminos, si el Attribute no es una clave fo-

ránea se crea y se inserta un nuevo registro de AtrributeInstance con el valor y el timestamp. En caso contrario, si es una clave foránea, se consulta si existe una EntityInstance asociada, en caso de existir se inserta la relación EntityInstance-EntityInstance. Si no existe dicha EntityInstance asociada se comienza un paso recursivo consultando por la clave en la tabla asociada de la base organizacional. Con la tupla que se obtiene por la consulta se vuelve al paso de insertar la EntityInstance y cada uno de los Attribute que esta tabla tiene. Al terminar la recursión completa las EntityInstance que antes no existían fueron ingresadas y se agrega la relación EntityInstance-EntityInstance.

Cabe destacar que la recursión solo se ejecuta una cantidad de veces configurable en una constante. Cada vez que se comienza un nuevo paso recursivo se le resta 1 a esta variable y solo se hace recursión si es mayor a 0. Esto se hace para evitar ciclos, e información innecesaria.

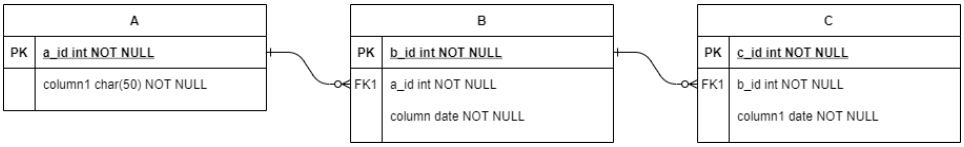


Figura 4.16: Ejemplo de recursión

Por ejemplo, si se tiene una estructura como la Figura 4.16 y se configura la recursión a un solo nivel de referencia, al cargar EntityInstance para la tabla C se cargaran los datos a los que se haga referecia de la tabla B, pero no los datos de la tabla A, ya que la tabla A se encuentra a dos niveles de referencia. Si se quiere obtener los datos de la tabla A solo hay que incrementar el límite de referencias.

De esta forma se cargan todos los valores que se obtienen por el log y, además, todos los valores asociados que se encuentran en la base organizacional conectado por claves foráneas. A partir de este mecanismo el modelo se enriquece, ya que se contemplan todos los datos asociados al proceso, se hayan insertado durante la ejecución o no.

4.4. Matcheo de datos de la organización y el proceso

Se implementa un algoritmo de matcheo con el objetivo de unir las entidades AttributeInstance del lado de los datos de la base organizacional con su respectiva VariableInstance del proceso. A partir de ese match, se agregan las relaciones entre:

- AttributeInstance y VariableInstance
- Attribute y VariableDefinition
- Entity y VariableDefinition
- EntityInstance y VariableInstance.

Todas estas relaciones están definidas en el metamodelo, como se puede apreciar en la Figura 4.17.

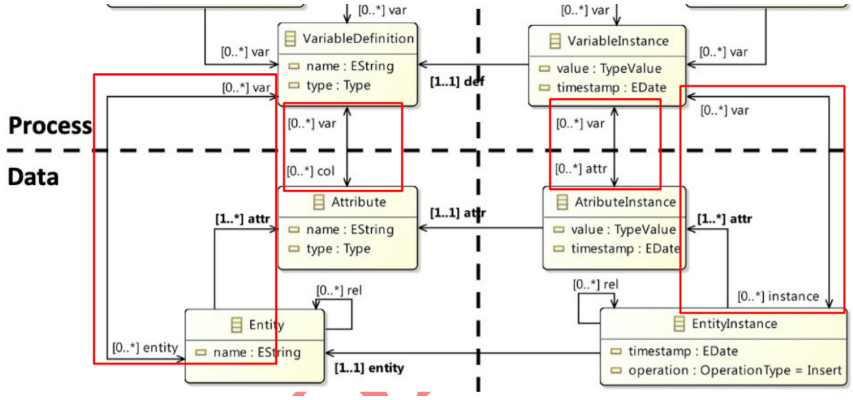


Figura 4.17: Matcheo entre los datos de la organización y el proceso

Para que los datos tengan sentido, el algoritmo de matcheo debe lograr conectar correctamente las distintas variables que se cargaron en el proceso con sus respectivas referencias en la base organizacional. Para lograr ese correcto matcheo se proponen dos condiciones:

- Matcheo por valor exacto: El valor de una variable del proceso debe coincidir exactamente con el valor del atributo de la base de datos organizacional.
- Matcheo por rango de tiempo: El timestamp de una variable del proceso debe coincidir, dentro de una ventana de tiempo configurable, con el timestamp del atributo de la base de datos organizacional.

A partir de estas condiciones se implementa un algoritmo de mathceo que realice la búsqueda y actualice las relaciones. Un pseudocódigo de como fue implementado se puede ver en 4.12.

Listing 4.12: Algoritmo de matcheo

```
const confTime = x milliseconds;

function match() {
    //Obtener todas las instancias de atributos
    listAttribute = getListAttributeInstance();

    // Recorrer todas las instancias
    for (attribute in listAttribute) {
        //Matchear atributos con variables por valor y timestamp
        matchVar = findVar(attribute.value(), attribute.timestamp()
            - confTime,
        attribute.timestamp() + confTime);

    //Setear variable matcheada y guardar en la base
        attribute.setVar(matchVar);
        attribute.save();

        //Setea el resto de las referencias del metamodelo
        setRerences(marchVar, attribute);
}
}

function findVar(value, startTime, endTime) {
    //Select en la base de datos para matchear por valor y
        timestamps
    matchVar = @Query( SELECT  vi.idVar FROM Variableinstance vi
        where vi.value=:value and vi.timestamp BETWEEN :startDate
        and :endDate)
    return matchVar;
}
```

Donde confTime define el rango de tiempo configurable. Este rango se llama ventana de tiempo y es importante encontrar un valor cercano al óptimo. Con óptimo se hace referencia al tiempo que logre la mayor cantidad de matcheos y que tengan sentido para el proceso.

A la hora de definir la ventana de tiempo usando la variable confTime, se debe considerar que latencias se manejan entre las comunicaciones. Esta consideración es importante, ya que, en ambientes locales la latencia será baja, pero en ambientes distribuidos la latencia es mayor, y por tanto la ventana de tiempo también debe serlo.

Definir la venta de tiempo no es tarea fácil, por lo que se sugieren seguir las siguientes recomendaciones:

- Recolectar un conjunto de matcheos conocidos entre las fuentes de datos y realizar un análisis estadístico tomando en cuenta promedios, ventana máxima, ventana mínima, etc.

- Realizar ejecuciones con diversas ventanas de tiempo sobre el conjunto de datos, y sacar conclusiones a partir de la cantidad y la calidad de los matcheos. Se puede analizar la calidad de los matcheos controlando que las instancias matcheen con el asignado en la definición.

Como el algoritmo solo considera valor y tiempo, se debe tener en cuenta que una ventana muy pequeña encontrará pocos matcheos, pero una ventana muy grande puede generar matcheos que no sean correctos. Esto sucede cuando un mismo valor se inserta o actualiza dos veces en una ventana de tiempo. El algoritmo, en este caso, carece de la inteligencia para definir cuál de las dos opciones es mejor, y tomará en cuenta solo el primero de ellos.

El algoritmo tiende a mostrar debilidades en escenarios donde se tienen ejecuciones de procesos en paralelo o latencias variables, pero se espera un buen rendimiento para el resto de los casos.

5 Solución planteada para data warehouse

A lo largo de este capítulo se detalla una solución para la creación del data warehouse, inspirada en el metamodelo presentado en la sección 3.2. Se toma como inspiración el metamodelo para que la solución solo dependa de su estructura y los datos que contiene, lo que permite analizar los procesos de cualquier organización.

Para describir el esquema de data warehouse se separará en tres secciones, la primera llamada diseño conceptual introducirá como se definen las dimensiones a partir de cada cuadrante del metamodelo, las relaciones entre dimensiones y finalmente las medidas que se obtendrán a partir de dichas relaciones. La siguiente sección es llamada diseño lógico y en ella se presenta el esquema de estrella que conforma la tabla de hechos. La tercera sección hablará de cómo se realiza la carga genérica del data warehouse a partir del metamodelo. Para finalizar también se describirán capacidad y limitaciones de este esquema de data warehouse.

5.1. Diseño conceptual

A continuación se muestra el diseño conceptual realizado para modelar las diferentes dimensiones del metamodelo, sus relaciones, las medidas y la tabla de roll-ups que se puede obtener del data warehouse.

Lo primero a mencionar es que las dimensiones de este data warehouse están fuertemente inspiradas en los cuadrantes del metamodelo presentado en 3.2 y los principales valores a estudiar son los propios valores de variables, como la cantidad de instancias al escalar o realizar cruzamientos entre dimensiones.

Pero también se agregan dimensiones que abren el abanico de posibilidades y permiten realizar mejores análisis como son dimensiones de tiempo, usuarios, y relaciones entre entidades.

En resumen las dimensiones que definen el data warehouse son:

- ProcessDefinition - cuadrante de definición de proceso del metamodelo.
- ProcessInstance - cuadrante de instancia de proceso del metamodelo.
- DataDefinition - cuadrante de definición de datos del metamodelo.
- DataInstance- cuadrante de instancia de datos del metamodelo.
- Time - días, meses y años.
- User - usuarios y roles del proceso.
- EntityRelation - relación entre entidades e instancias de entidades.

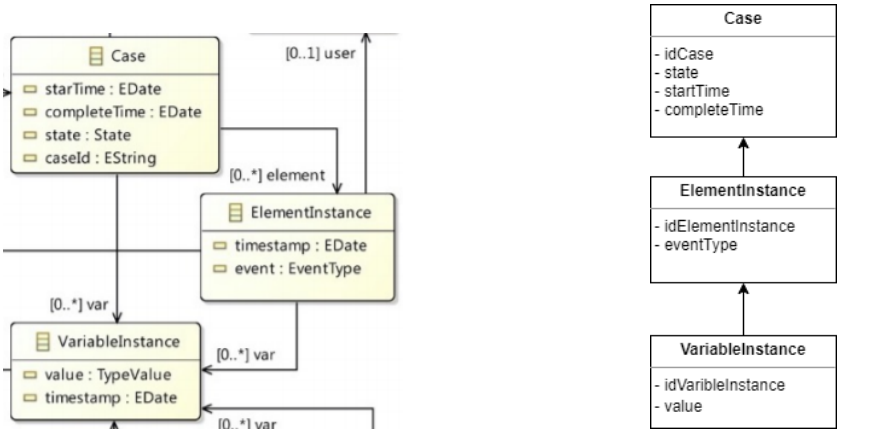
5.1.1. Dimensiones

A continuación se presentan con más detalles cada una de estas dimensiones, como también las medidas y tabla de roll-up para un mejor entendimiento del data warehouse.

Dimensión ProcessInstance

La primera dimensión llamada ProcessInstance se refiere a las instancias de variables que tiene el proceso. Para esto, se define una jerarquía como lo muestra la Figura 5.1b donde las instancias de variables son los elementos más atómicos y se puede hacer rollup hacia instancias de elementos o un nivel más arriba a instancias del proceso.

A su vez con esta dimensión se logra representar la sección del metamodelo que se muestra en la Figura 5.1a.



(a) ProcessInstance en el metamodelo

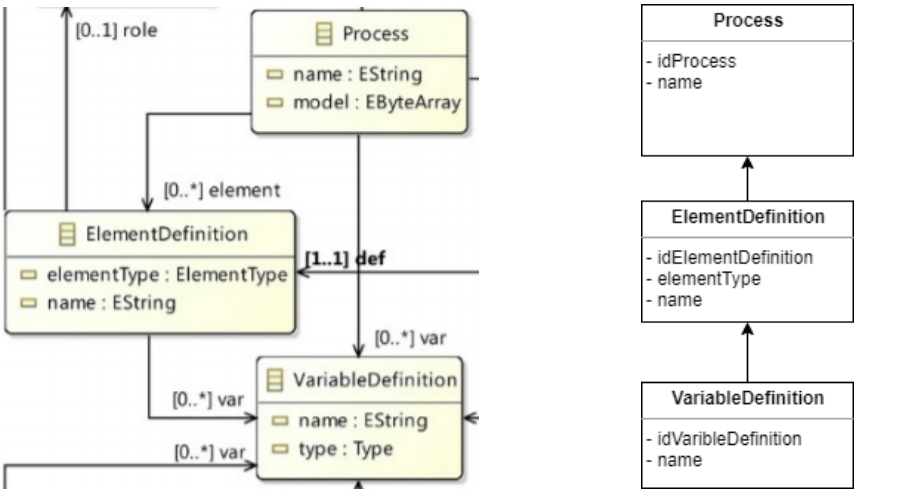
(b) Dimensión ProcessInstance

Figura 5.1: ProcessInstance

Dimensión ProcessDefinition

La segunda dimensión llamada ProcessDefinition se refiere a las definiciones de variables que tiene el proceso. Para esto, se define una jerarquía como lo muestra la Figura 5.2b donde las definiciones de variables son los elementos más atómicos y se puede hacer rollup hacia definiciones de elementos o un nivel más arriba a definiciones del proceso.

A su vez con esta dimensión se logra representar la sección del metamodelo que se muestra en la Figura 5.2a.



(a) ProcessDefinition en el metamodelo

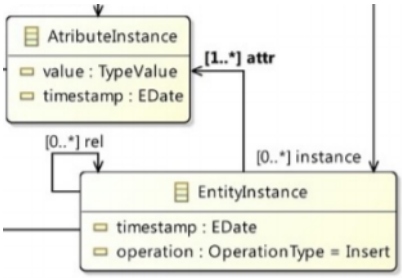
(b) Dimensión ProcessDefinition

Figura 5.2: ProcessDefinition

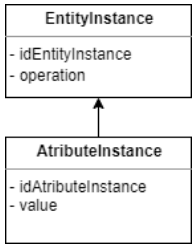
Dimensión DataInstance

Esta dimensión llamada DataInstance se refiere a las instancias de atributos que se utilizan durante el proceso en la base organizacional. Para esto, se define una jerarquía como lo muestra la Figura 5.3b donde las instancias de atributos son los elementos más atómicos y se puede hacer rollup hacia instancias de entidades.

A su vez con esta dimensión se logra representar la sección del metamodelo que se muestra en la Figura 5.3a.



(a) DataInstance en el metamodelo



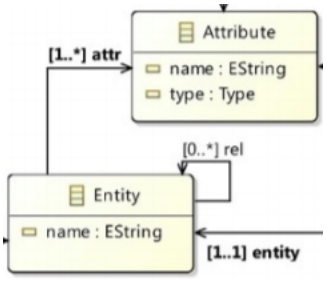
(b) Dimensión DataInstance

Figura 5.3: DataInstance

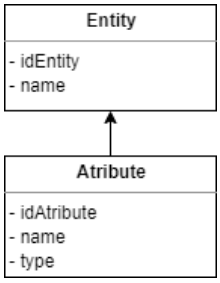
Dimensión DataDefinition

Esta dimensión llamada DataDefinition se refiere a las definiciones de atributos y entidades que se utilizan durante el proceso en la base organizacional. Para esto, se define una jerarquía como lo muestra la Figura 5.4a donde las definiciones de atributos son los elementos más atómicos y se puede hacer rollup hacia definiciones de entidades.

A su vez con esta dimensión se logra representar la sección del metamodelo que se muestra en la Figura 5.4b.



(a) DataDefinition en el metamodelo



(b) Dimensión DataDefinition

Figura 5.4: DataDefinition

Dimensión Time

Esta dimensión es definida para realizar análisis sobre distintos rangos de tiempo. Como se ve en la Figura 5.5 el nivel más atómico es día y existe la posibilidad de hacer rollup hacía mes o año.

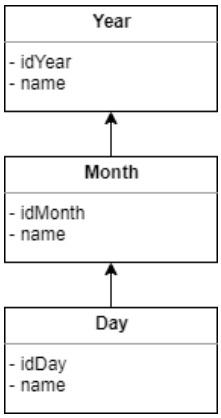


Figura 5.5: Dimensión Time

Como se puede notar, las cuatro primeras dimensiones son extraídas directamente de la estructura que el metamodelo posee, mientras que la última dimensión se considera una dimensión auxiliar para poder hacer análisis respecto a diferentes tiempos.

Dimensión User

Esta dimensión es definida para realizar análisis sobre los distintos usuarios que efectúan las tareas del proceso. Como se ve en la Figura 5.6 el nivel más atómico es el usuario y existe la posibilidad de hacer rollup hacía los roles de los usuarios.

A su vez con esta dimensión se logra representar la sección del metamodelo que se muestra en la Figura 5.7.

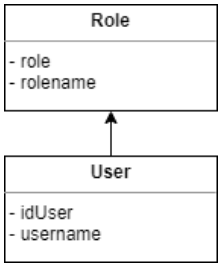


Figura 5.6: Dimensión User

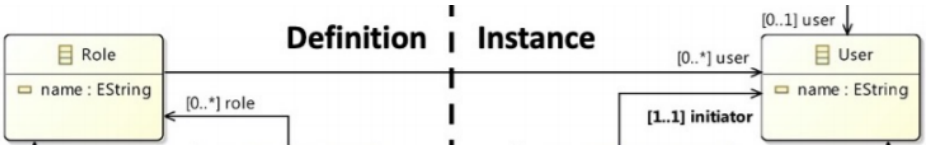


Figura 5.7: Dimensión User en el metamodelo

Dimensión EntityRelation

Esta dimensión es definida para realizar análisis sobre las relaciones entre Entities. Como se ve en la Figura 5.8 el nivel más atómico es la relación entre EntityInstances y existe la posibilidad de hacer rollup hacía la relación entre Entities.

A su vez con esta dimensión se logra representar la sección del metamodelo que se muestra en la Figura 5.9.

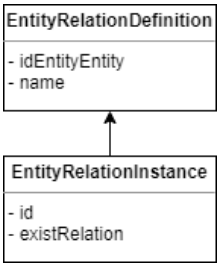


Figura 5.8: Dimensión EntityRelation

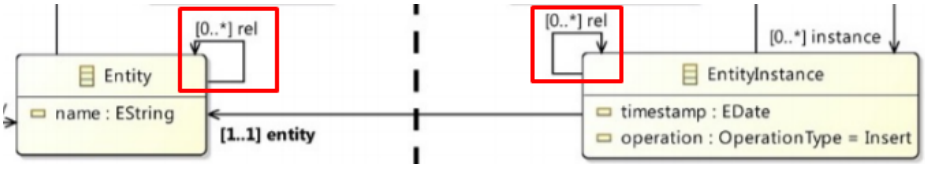


Figura 5.9: Dimensión EntityRelation en el metamodelo

Gracias a esta dimensión, es posible hacer análisis sobre las relaciones entre las tablas de la base de datos organizacional. Es decir que si se tiene dos entidades A y B que están relacionadas entre sí, y sus instancias fueron cargadas al metamodelo, con esta dimensión se podrá agrupar los atributos que estén asociados a A, pero también los que estén en B.

Muchas veces una entidad tiene atributos que conforma una foreign key a otra entidad, y conocer solo este dato en general no es suficiente. Al unir datos que están relacionados se

puede hace un análisis en donde ver fácilmente todos los datos que interesen de la entidad que se está estudiando.

Nota: El valor “existRelation” en “EntityRelationInstance” hace referencia a que alguna EntityInstance de la relación tiene un matcheo con alguna variable del proceso. Esto es importante para poder filtrar instancias de datos fácilmente y analizar solo las que tienen algún proceso asociado.

5.1.2. Medidas

- Cantidad de instancias: Indica la cantidad de instancias de un conjunto de atributos.
- Monto total: Indica los valores de tipo entero, en caso de que el valor no sea de tipo entero se asigna 0.
- Valor de los atributos.
- Process Duration: Indica la duración del proceso.
- Element Duration: Indica la duración de una tarea.

5.1.3. Relaciones dimensionales

En la Figura 5.10 se pueden observar la relación entre las dimensiones y medidas mencionadas.

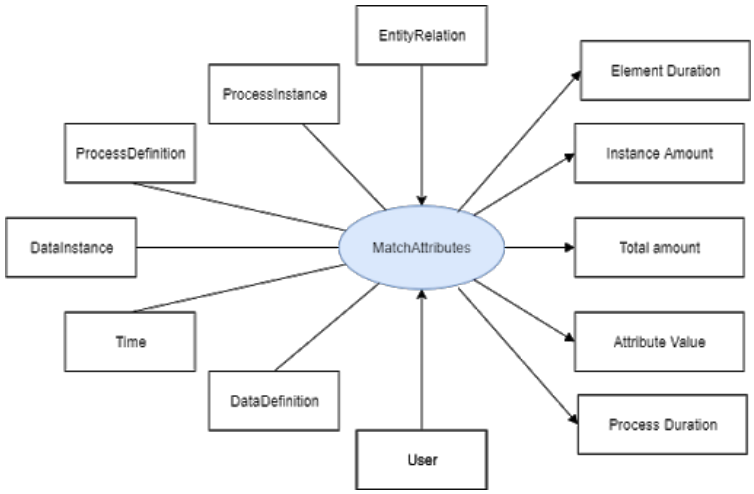


Figura 5.10: Relaciones dimensionales

Tabla de roll up

En la Figura 5.11 se puede ver la tabla de roll up, en donde se representa las agregaciones de las medidas por dimensión.

		Instance Amount	Total Amount	Attribute Value	Process Duration	Element Duration
ProcessInstance	VariableInstance -> ElementIsntance	+	+, PROM	NA	MAX	MAX
	ElementInstance -> Case	+	+, PROM	NA	MAX	MAX
	Case -> All	+	+, PROM	NA	MAX	MAX
ProcessDefinition	VariableDefinition -> ElementDefinition	+	+, PROM	NA	MAX	MAX
	ElementDefinition -> Process	+	+, PROM	NA	MAX	MAX
	Process -> All	+	+, PROM	NA	MAX	MAX
DataInstance	AttributeInstance -> EntityInstance	+	+, PROM	NA	MAX	MAX
	EntityInstance -> All	+	+, PROM	NA	MAX	MAX
DataDefinition	Attribute -> Entity	+	+, PROM	NA	MAX	MAX
	Entity -> All	+	+, PROM	NA	MAX	MAX
Time	Day -> Month	+	+, PROM	NA	MAX	MAX
	Month -> Year	+	+, PROM	NA	MAX	MAX
	Year -> All	+	+, PROM	NA	MAX	MAX
User	User -> Role	+	+, PROM	NA	MAX	MAX
	Role -> All	+	+, PROM	NA	MAX	MAX
EntityRelation	EntityInstanceRelation -> EntityRelation	+	+, PROM	NA	MAX	MAX
	EntityRelation -> All	+	+, PROM	NA	MAX	MAX

Figura 5.11: Tabla roll up

5.2. Diseño Lógico

En esta sección se desarrolla la estructura de la solución que se representará directamente en la base de datos y que será cargada dentro del data warehouse.

En la Figura 5.12 se puede ver la estructura del diseño logico, en donde se utiliza una tabla de base de datos para cada dimensión definida, y una Fact table con foreign key a cada una de ellas.

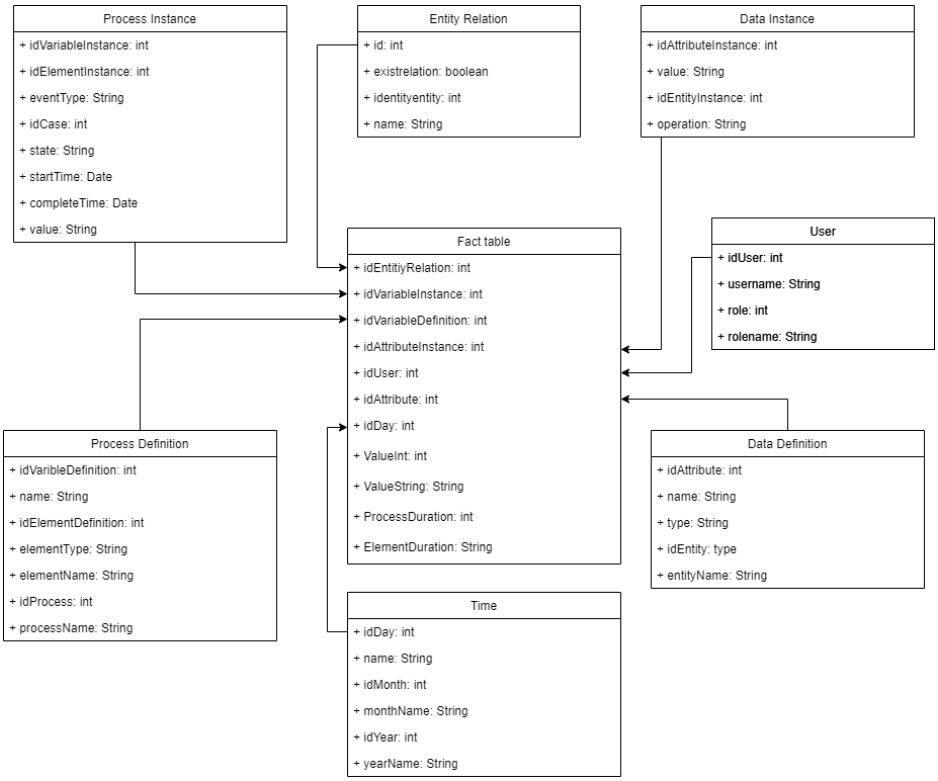


Figura 5.12: Diseño Lógico

5.3. Carga de data warehouse

En esta sección se presenta la carga genérica del data warehouse a partir del metamodelo utilizando la herramienta Kettle de Pentaho. En la Figura 5.13, se ve el Job de carga que se ejecuta para cargar completamente el data warehouse.

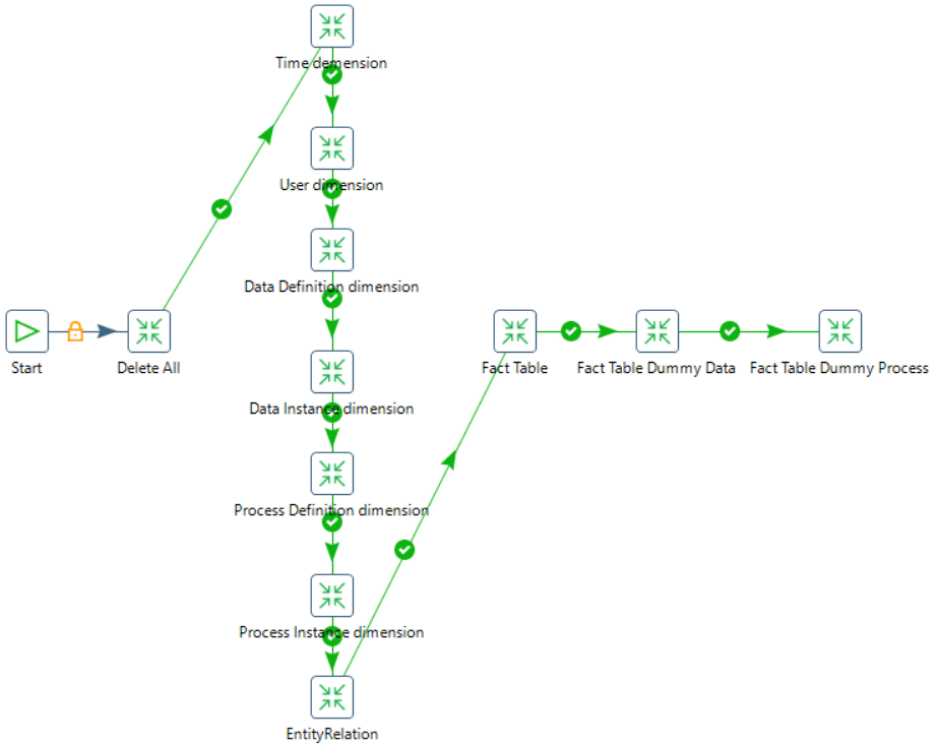


Figura 5.13: Kettle job

Este comienza cargando cada una de las dimensiones a partir de las transformations:

- Time dimension: Carga de fechas extraídas de un archivo csv que contiene todas las fechas desde 1999 al 2030.
- User dimension: Carga de la dimensión User a partir del metamodelo. Utilizando las tablas User y Role.
- Data Definition dimension: Carga de la dimensión Data Defintion a partir del metamodelo. Utilizando las tablas Attribute y Entity.
- Data Instance dimension: Carga de la dimensión Data Instance a partir del metamodelo.

Utilizando las tablas AttributeInstance y EntityInstance.

- Process Definition dimension: Carga de dimensión Process Definition a partir del metamodelo. Utilizando las tablas Process, ElementDefinition y VariableDefinition.
- Process Instance dimension: Carga de dimensión Process Instance a partir del metamodelo. Utilizando las tablas Case, ElementInstance y VariableInstance.
- EntityRelation: Carga de dimensión EntityRelation a partir del metamodelo. Utilizando las tablas EntityEntity y EntityInstanceEntityInstance.
- Fact Table: Carga de la Fact Table a partir de las dimensiones antes cargadas y utilizando solo los datos que tienen matcheo directo.
- Fact Table Dummy Data: Carga de la Fact Table a partir de las dimensiones antes cargadas y utilizando datos dummy en los procesos para poder cargar los datos de la organización que tienen relación indirecta con los procesos.
- Fact Table Dummy Process: Carga de la Fact Table a partir de las dimensiones antes cargadas y utilizando datos dummy en los datos de la organización para poder cargar los datos del proceso que no tienen relación con los datos organizacionales.

5.4. Capacidades y limitaciones de caso general

Capacidades

A través de este modelo se pueden realizar análisis sobre todos los datos del proceso, incluyendo todas sus variables y actividades con sus respectivas relaciones. Además, al contar con los datos de la base organizacional, se puede nutrir cada instancia del proceso con los datos que matchean con la base organizacional y que son invisibles para el proceso.

Lo mencionado anteriormente ofrece la capacidad de obtener más información como, por ejemplo, el nombre de un estudiante que ha participado en un proceso, siendo que este dato no existe como una variable del proceso, pero si existe en la base organizacional.

Limitaciones

Si bien este modelo de data warehouse ofrece un buen nivel de análisis tiene el problema de que es genérico, y al ser genérico se pierden relaciones que se tienen entre los datos de la base organizacional.

Como un ejemplo de esto, se puede analizar los datos de los estudiantes que se tienen para un proceso o actividad en específico gracias a la relación entre aplicaciones y estudiante, pero analizar los datos de las carreras de dichos estudiantes o qué institutos participan se vuelve una tarea complicada. Esto se debe a que las relaciones son solo de un nivel y este tipo de cruces entre los datos requerirá un análisis complicado e incluso el uso de alguna herramienta

auxiliar para lograrlo.

Principalmente estas limitaciones se deben a que los atributos están todos en el mismo nivel, lo que hace imposible la agregación. Como por ejemplo, no se pueden agrupar los datos de una aplicación para un estudiante. Para lograrlo debería hacerse una dimensión estudiante específica.

6 Ejemplo de Aplicación: Student Mobility

Para realizar un análisis sobre la solución desarrollada se toma como referencia el caso de estudio de movilidad de estudiantes [5]. Una movilidad es un programa donde estudiantes pueden postularse para estudiar y revalidar materias en universidades del extranjero. Además, como parte del programa se les otorga un monto para que puedan efectuar dichos estudios en el exterior.

Los programas son definidos y evaluados por el Consejo de la Universidad, quienes definen qué estudiantes son los aprobados para el programa definido. Los estudiantes pueden registrarse al programa a partir de la oficina de registros dentro de un plazo de 15 días desde el comienzo del programa. Luego de que se cierra el plazo para registrarse la oficina de registro descarga todas las postulaciones y confirma las que cumplen con las condiciones.

Las postulaciones confirmadas pasan al panel de evaluaciones para ser evaluadas y definir la lista de titulares y de suplentes que será enviada al Consejo para aprobar. Si el Consejo aprueba dichas listas, se notifica a cada estudiante sus resultados para luego notificar también al DGRC (Dirección General del Registro de Estado Civil de la República Oriental del Uruguay) y hacer el pago correspondiente a los estudiantes.

La realidad antes descrita se diseña como un proceso de negocios desarrollado en Activiti, y su modelado se muestra en la Figura 6.1. El primer paso consiste en que el Consejo defina el programa y se termina con las tareas de pago y notificación al DGRC. En cada paso se imita el comportamiento del proceso real y se hacen registros en la base del motor de procesos como también en una base de datos organizacional, ambas implementadas en PostgreSQL 12.

La estructura de la base organizacional se compone por datos básicos de la organización y datos de la ejecución del proceso. El esquema de la base se muestra en la Figura 6.2

Las tablas teacher, institute, course, state, student y career son parte de los datos básicos de

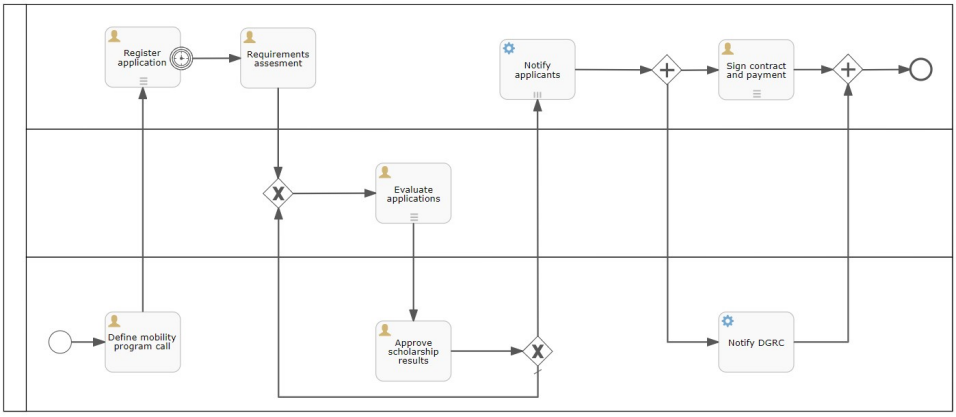


Figura 6.1: Modelo

la organización, esto implica que no se ingresan como resultado de la ejecución del proceso. Mientras que las tablas validation, mobility, application y program son cargadas durante la ejecución del proceso.

Dichas tablas refieren a:

- teacher: Todos los profesores que participan de la organización y pertenecen a un instituto.
- institute: Todos los institutos que participan de la organización.
- course: Todos los cursos de la organización que pertenecen a un instituto y a una carrera.
- career: Todas las carreras de la organización.
- student: Todos los estudiantes de la organización que pertenecen a una carrera.
- program: Todos los programas que han sido definidos en los procesos ejecutados.
- application: Todas las aplicaciones que participaron en los programas, cada una de ellas tiene solo un programa asignado, un estudiante y un estado actual.
- state: Estados que puede tener una aplicación (Initiated, Confirmed, Rejected, Holder, Substitute).
- validation: Todas las validaciones sobre cursos que hayan sido solicitados en una aplicación. Las validaciones tienen un curso asignado y una aplicación.
- mobility: Todas las movilidades que han sido aprobadas con su respectivo monto asignado y la aplicación a la que hacen referencia.

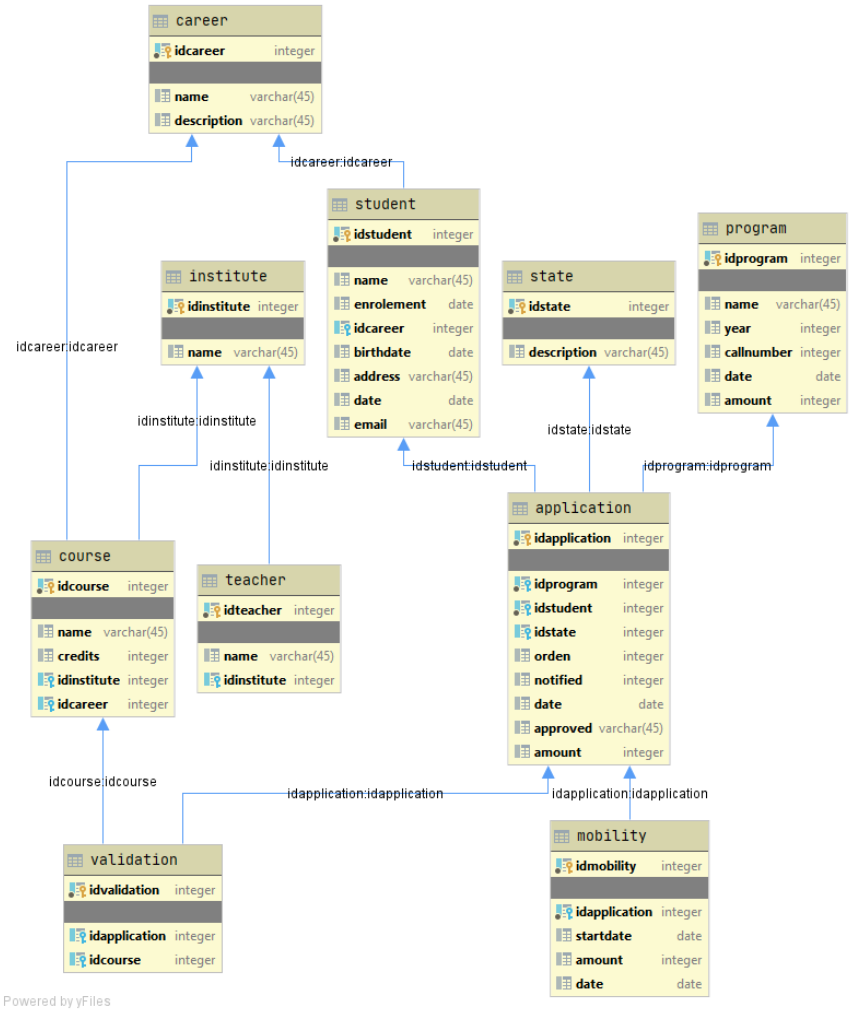


Figura 6.2: Modelo

Vale la pena aclarar que los campos amount del programa y de la aplicación no significan que el Consejo o los estudiantes tengan que definir el monto, sino que existen para poder simular la realidad de que estudiantes entran dentro del presupuesto debido al monto que van a necesitar para hacer sus cursos.

6.1. Implementación del modelo en Activiti

El modelo fue implementado en Activiti 6.0 y con una base de datos Postgres 12. A continuación se verán los detalles de cada tarea y como impactan en la base organizacional.

Define mobility program call.

En esta tarea se define el programa y se setean todos los campos que se muestran en la Figura 6.3

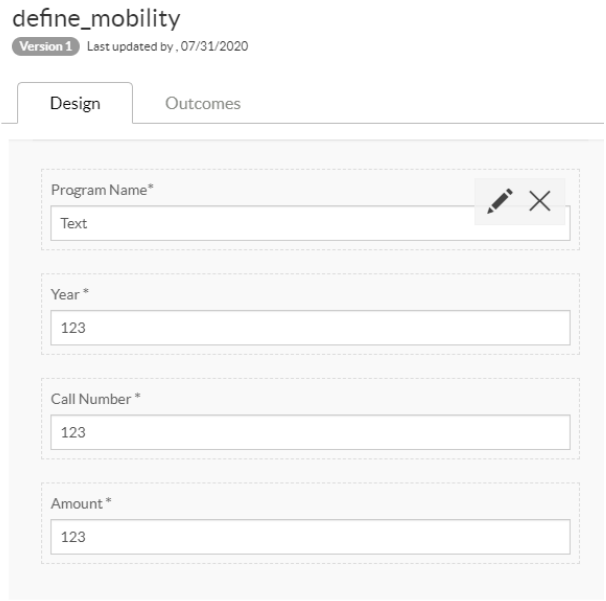


Figura 6.3: Define mobility

- Complete task listener: persist.InsertProgram. Inserta el programa con sus respectivos datos y añade al proceso la variable program_id la cual es un identificador autogenerado por la base.

Al completar esta tarea se hace una inserción en la tabla program con los datos ingresados en el formulario anterior.

Register Application

Esta tarea es la encargada de recibir las aplicaciones de los estudiantes como muestra el formulario de la Figura 6.4, cargando en la base de datos la application y sus respectivas validations para los cursos ingresados.

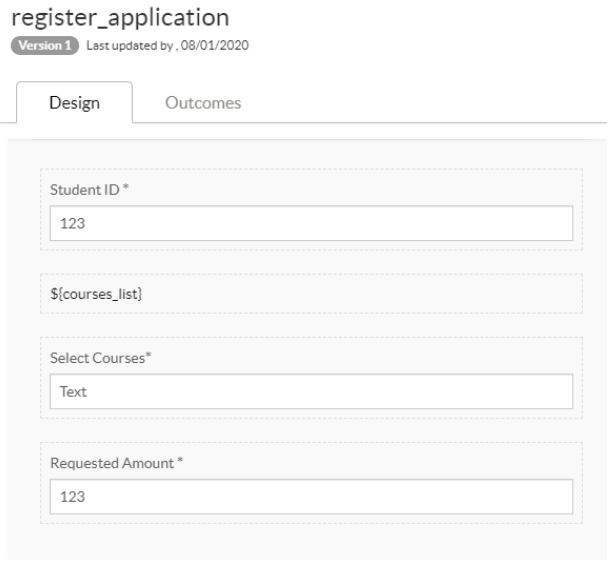


Figura 6.4: Register Application

- Create task listener: controller.GetCourses. Se consulta en la base de datos la lista de cursos y la setea en la variable courses_list del proceso.
- Complete task listener: persist.InsertApplication. Se inserta en la base de datos la aplicación con el id de estudiante ingresado. Además crea una validation por cada curso que el estudiante seleccionó.

Requirements assessment

En esta task se seleccionan las aplicaciones que se desea confirmar como se ve en la Figura 6.5.

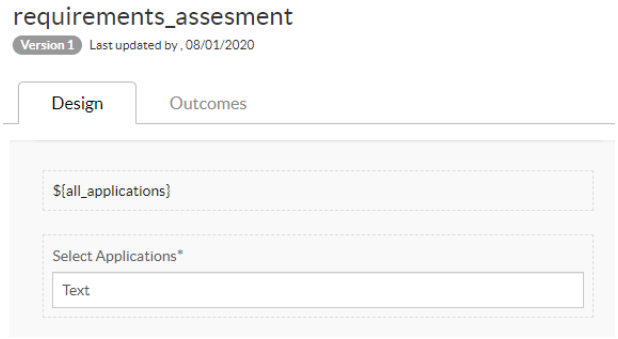


Figura 6.5: Requirements assessment

- Assignment task listener: controller.ListApplications. Se consulta en la base de datos todas las aplicaciones que se ingresaron para este programa y se las carga en la variable all_applications como un arreglo de identificadores (integers).
- Complete task listener: controller.ConfirmApplication. Se cambia el campo state en la tabla de application de la aplicación a confirmed o rejected. Además se setea una variable llamada apps_list con la lista de las aplicaciones confirmadas.

Evaluate applications

En esta user task se iteran todas las aplicaciones confirmadas y se agrega el orden para cada una. El fomulario de esta user task se puede ver en la Figura 6.6.

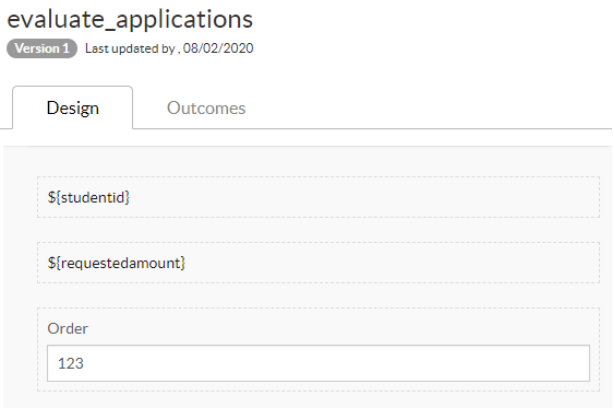


Figura 6.6: Evaluate applications

- Create task listener: controller.LoadApplication. Se carga la aplicacion actual consultado a la base por su id. Se cargan las variables studentid y requestedamount para mostrarse en el formulario.
- Complete task listener: controller.EvaluateApplication. Se guarda en la base de datos el orden asignado a la aplicación.

Approve scholarship results

En esta tarea se decide si aprobar o no el orden de las aplicaciones, se carga la variable approve_scholarship_results con true o false dependiendo de lo que se decida. El formulario de esta user task se puede ver en la Figura 6.7.

Figura 6.7: Approve scholarship results

- Create task listener: controller.GenerateHolderList. En esta task se setean los estados de las respectivas aplicaciones a titular o supletente como también se setea la variable de tipo String apps_holder_substitut indicando las aplicaciones que quedaron como titulares y las que quedaron como suplentes para luego informar al usuario. Además, se setea la variable holder_id_list, la cual es una lista de ids de los titulares (integer).

Confirmar applications

Luego de que las aplicaciones son aprobadas en approve scholarship results se setea en cada aplicación confirmada el campo approved como true. Se decidió hacer en esta flecha como lo muestra la Figura 6.8 porque la siguiente tasks son todas iteradores y se debe ejecutar una sola vez.

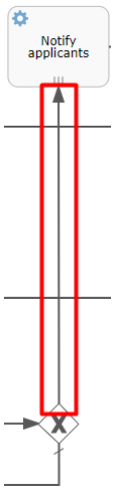


Figura 6.8: Confirmar applications

- Start execution listener: controller.ApproveScholarshipResults. Setea el campo appro-

ved como true.

Notify application

Esta tarea se cambia de mensaje a service task por problemas de performance y en los registros de la base del proceso no existe diferencia. Se envía un mail a todos los aplicantes y setea el campo notified como true.

- Class:controller.NotifyApplicant. Envía el mail al aplicante y setea el campo notified de la aplicación como true.

Sign contract and payment

Se pide al aplicante la firma y se inserta la mobility en la base de datos. El formulario de esta user task se puede ver en la Figura 6.9.

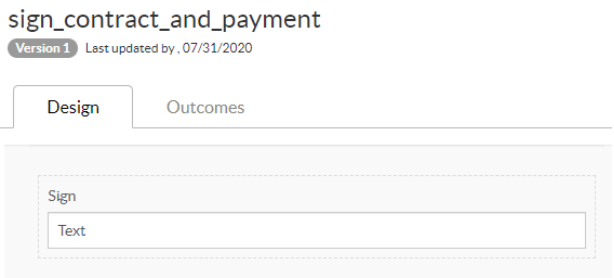


Figura 6.9: Sign contract and payment

- Complete task listener: persist.InsertMobility. Se inserta un registro en la tabla mobility para la aplicación con el monto que se le habilitó.

Notify DGRC

Se imita el envio de los resultados a la DGRC a través de un web service.

- Class NotifyDGRC. Se hace un post al servicio con el nombre del programa y la lista de títulos y suplentes con sus respectivos montos.

6.2. Generación de instancias

Se implementó una herramienta auxiliar para automatizar la generación de datos, la cual se detalla en el anexo A. Este generador es genérico para cualquier proceso, pero como la realidad trabajada tiene ciertas condiciones se decidió extenderlo para que los datos generados pertenezcan al dominio del caso de estudio.

Dichas modificaciones son realizadas a la hora de completar un campo de un formulario. Se puede identificar un campo a partir de su nombre y formulario al cual pertenece, para completarlo dependiendo de las particularidades que tenga.

Particularidades tomadas en cuenta para el caso Student Mobility:

- Los valores booleanos para el caso de la task Approve scholarship results tienen una probabilidad más alta de tomar un valor verdadero para que el loop del proceso no se repita muchas veces.
- Para el campo select courses se completa solo con identificadores de curso dentro de los que se tienen en la base de datos organizacional.
- Para el campo select application se completa solo con identificadores de las aplicaciones insertadas para el proceso actual.
- Para el campo select teachers se completa solo con identificadores de los profesores que se tienen en la base organizacional.
- Para el campo student id se completa solo con un indicador válido de estudiante consultado la base organizacional.
- Para el campo monto total se inserta un valor de 4 dígitos mientras que en el resto de los valores se insertan valores de 3 o menos dígitos. Esto se hace para que el monto total tenga sentido con respecto a los requeridos por los estudiantes.

La generación de instancias se corrió para 100 procesos. Se usaron timers que oscilan entre 15000 y 18500 milisegundos para completar una tarea, y de 10000 a 13500 milisegundos para tomar una tarea nueva. De esta forma se consigue generar un comportamiento simulado más parecido a la realidad.

A su vez, para conseguir un conjunto de datos más completo se dividen las instancias en 5 tipos de ejecuciones diferentes, los cuales fueron:

1. 10 ejecuciones en paralelo con el mismo primer estudiante.
2. 20 ejecuciones en paralelo random.
3. 25 ejecuciones secuenciales con probabilidad de aprobación de 30 %.
4. 25 ejecuciones secuenciales con probabilidad de aprobación de 70 %.
5. 20 ejecuciones secuenciales random.

Estas ejecuciones fueron realizadas sobre el siguiente conjunto de datos básicos:

1. 500 estudiantes.
2. 60 cursos.

3. 20 profesores.
4. 10 carreras.
5. 6 institutos.

Estos datos fueron generados pensando en imitar la realidad, como por ejemplo, que las cédulas de estudiantes tengan el largo correcto, que las direcciones sean reales y nombres de institutos y carreras reales.

6.3. Resultados

En esta sección se podrá ver los resultados de la generación de datos mencionada en la sección anterior, y posterior carga del metamodelo, evaluando la fidelidad de los datos cargados y del matcheo entre ellos.

6.3.1. Datos del proceso

La ejecución de las 100 instancias de llamados de movilidad generaron los siguientes datos de proceso:

1. 100 cases, que refieren a las 100 instancias que se ejecutaron
2. 22412 Instancias de elementos, o tareas completadas
3. 93586 Instancias de variables, o variables utilizadas.

En la Figura 6.10 se puede ver de forma más detallada la distribución de las tareas ejecutadas. En esta figura se puede observar los registros de ElementDefinition, las cuales corresponden con las User Tasks mencionadas en la sección 6, acompañadas de la cantidad de instancias que se crearon de cada una.

Element definition character varying (45)	cantidad de instancias bigint
Evaluate applications	6502
Requirements assesment	196
Register application	12160
Approve scholarship results	236
Sign contract and payment	3116
Define mobility program call	200

Figura 6.10: Cantidad de instancias de ElementDefinition

A modo de ejemplo en la Figura 6.11 se muestra algunos valores que fueron cargados desde el proceso. En particular se pueden ver datos que fueron ingresados al realizar la tarea "Register application" en un proceso de movilidad. Se puede ver que esta tarea tiene una variable "studenId" con valores 63843980 y 53276566 que corresponden a los identificadores de los estudiantes.

	idelementinstance integer	Nombre elemento character varying (45)	timestamp timestamp without time zone	event integer	idcase integer	Nombre processc character varyin
1	9947950	Register application	2021-05-13 21:36:50.735	1	993248	Mobility
2	9947950	Register application	2021-05-13 21:36:50.736	1	993248	Mobility
3	9947950	Register application	2021-05-13 21:36:50.736	1	993248	Mobility
4	9947950	Register application	2021-05-13 21:36:50.736	1	993248	Mobility
5	9947950	Register application	2021-05-13 21:36:50.736	1	993248	Mobility
6	9948190	Register application	2021-05-13 21:37:17.85	1	993248	Mobility
7	9948190	Register application	2021-05-13 21:37:17.851	1	993248	Mobility
8	9948190	Register application	2021-05-13 21:37:17.851	1	993248	Mobility
9	9948190	Register application	2021-05-13 21:37:17.851	1	993248	Mobility
10	9948190	Register application	2021-05-13 21:37:17.85	1	993248	Mobility

	idvariableinstance integer	value character varying (200)	Nombre variable character varying (45)
1	39619	63843980	studentid
2	39615	1555	selectcourses
3	39616	2888	selectcourses
4	39617	4555	selectcourses
5	39618	3236	requestedamount
6	39609	53276566	studentid
7	39605	21103	selectcourses
8	39606	21210	selectcourses
9	39607	21202	selectcourses
10	39608	2666	requestedamount

Figura 6.11: Ejemplo de datos del proceso

6.3.2. Datos de Mobility

La ejecución de las 100 instancias de llamados de movilidad generaron los siguientes datos:

1. 39078 Instancias de entidades, registros en las tablas de la base de datos organizacional
2. 229392 Instancias de atributos, valores de los registros manipulados en la base organizacional.

En la Figura 6.12 se puede observar los registros de Entity, las cuales se corresponden con las tablas de la base de datos organizacional que se muestran en el diagrama de la Figura 6.2, acompañadas de la cantidad de instancias que se crearon de cada una. Esta cantidad

representa las veces que se insertó un registro en alguna de estas tablas durante la generación de instancias.

Entity character varying (45)	Cantidad de instancias bigint
state	4
application	178
course	31
institute	4
student	19
career	5
validation	109
mobility	20
program	10

Figura 6.12: Cantidad de instancias de Entity

A modo de ejemplo en la Figura 6.13 se muestra algunos valores que fueron cargados desde la organización. En particular se pueden ver datos que fueron ingresados al realizar dos aplicaciones de estudiantes a un proceso de movilidad. Se puede ver que estas aplicaciones tienen un atributo “idstudent” con valores 63843980 y 53276566 que corresponden a las cédulas de estos estudiantes.

	identityinstance integer	timestamp timestamp without time zone	identity integer	Nombre entidad character varying (45)	idattribute integer	Nombre atributo character varyin
1	33986	2021-05-13 21:36:50.996	1055	application	4488	amount
2	33986	2021-05-13 21:36:50.996	1055	application	4490	approved
3	33986	2021-05-13 21:36:50.996	1055	application	4486	date
4	33986	2021-05-13 21:36:50.996	1055	application	4484	idapplication
5	33986	2021-05-13 21:36:50.996	1055	application	4491	idprogram
6	33986	2021-05-13 21:36:50.996	1055	application	4487	idstate
7	33986	2021-05-13 21:36:50.996	1055	application	4483	idstudent
8	33986	2021-05-13 21:36:50.996	1055	application	4489	notified
9	33986	2021-05-13 21:36:50.996	1055	application	4485	orden
10	33995	2021-05-13 21:37:18.037	1055	application	4488	amount
11	33995	2021-05-13 21:37:18.037	1055	application	4490	approved
12	33995	2021-05-13 21:37:18.037	1055	application	4486	date
13	33995	2021-05-13 21:37:18.037	1055	application	4484	idapplication
14	33995	2021-05-13 21:37:18.037	1055	application	4491	idprogram
15	33995	2021-05-13 21:37:18.037	1055	application	4487	idstate
16	33995	2021-05-13 21:37:18.037	1055	application	4483	idstudent
17	33995	2021-05-13 21:37:18.037	1055	application	4489	notified
18	33995	2021-05-13 21:37:18.037	1055	application	4485	orden

	type character varying (45)	idattributeinstance integer	value character varying (200)
1	int4	204396	3236
2	varchar	204397	NULL
3	date	204395	2021-05-13
4	int4	204398	5638
5	int4	204403	386
6	int4	204402	1
7	int4	204399	63843980
8	int4	204400	NULL
9	int4	204401	NULL
10	int4	204445	2666
11	varchar	204446	NULL
12	date	204444	2021-05-13
13	int4	204447	5640
14	int4	204452	386
15	int4	204451	1
16	int4	204448	53276566
17	int4	204449	NULL
18	int4	204450	NULL

Figura 6.13: Ejemplo de datos de la organización

6.3.3. Matcheo

Luego de generar las instancias y cargar los resultados en el metamodelo es hora de ejecutar el macheo entre los datos del proceso y de la organización.

EL algoritmo fue corrido en un terminal con las siguientes especificaciones:

- Procesador: Intel i5-7300HQ 2.50Hz
- Memoria ram: 16Gb
- Sistema operativo: windows 10

Como se mencionó en la sección 4.4 el matcheo se hace con un intervalo de tiempo configurable, para definir este intervalo no se utilizan técnicas especializadas, se define a partir de la experiencia. Tanto la ejecución del proceso, la base del proceso y la base organizacional se encuentran en el mismo equipo por lo que se reduce la latencia y se consiguen tiempos estables, de esta forma el rango para el matcheo se define en $\pm 1s$.

La ejecución del matcheo deja como resultado 33447 variables matcheadas, lo que representa el 35.74 % de las instancias de variable del proceso. En esta parte es importante resaltar que existen muchas variables del proceso que son necesarias a la hora de ejecutar, pero no tienen una relación con los datos del dominio, un ejemplo son las variables de loopcounter, que se utilizan para llevar a cabo las iteraciones en tareas.

En la Figura 6.14 se puede observar los atributos de las entidades que hicieron match con las variables del proceso. En este match se puede observar un claro comportamiento en donde las tareas approve scholarship results, Evaluate applications, Register application, Requirements assesment y Define mobility program call comparten valores con las tablas application, program y validation de la base de Mobility, y se puede apreciar específicamente cuáles son esos pares de atributo y variable.

Nombre de entidad character varying (45)	Nombre de atributo character varying (45)	Nombre de variable character varying (45)	Nombre de tarea character varying (45)	count bigint
application	approved	approve_scholarship_results	Approve scholarship results	20
application	amount	requestedamount	Evaluate applications	11
application	idstudent	studentid	Evaluate applications	11
application	orden	order	Evaluate applications	33
application	amount	requestedamount	Register application	34
application	idstudent	studentid	Register application	34
application	idapplication	select_applications	Requirements assesment	20
program	amount	amount	Define mobility program call	10
program	callnumber	callnumber	Define mobility program call	10
program	idprogram	program_id	Define mobility program call	10
program	name	programname	Define mobility program call	10
program	year	year	Define mobility program call	10
validation	idcourse	selectcourses	Register application	95

Figura 6.14: Cantidad matcheos de Entity

En las secciones anteriores se mencionaron ejemplos en donde se cargaba la variable de

proceso “studenId” con valores 63843980 y 53276566, y el atributo de la organización “idstudent” con los mismos valores 63843980 y 53276566. En la Figura 6.15 se puede ver que además de tomar los mismos valores fueron ingresados con timestamps que difieren en menos de un segundo, por lo tanto el algoritmo detecta un matcheo y crea la relación entre las instancias de variable y atributo correspondiente.

	idattributeinstance integer	value character varying (200)	timestamp timestamp without time zone	Nombre atributo character varying (45)	identityinstance integer
1	204399	63843980	2021-05-13 21:36:50.996	idstudent	33986
2	204396	3236	2021-05-13 21:36:50.996	amount	33986
3	204448	53276566	2021-05-13 21:37:18.037	idstudent	33995
4	204445	2666	2021-05-13 21:37:18.037	amount	33995

	idvariableinstance integer	Nombre variable character varying (45)	timestamp timestamp without time zone	idcase integer
1	39619	studentid	2021-05-13 21:36:50.735	993248
2	39618	requestedamount	2021-05-13 21:36:50.736	993248
3	39609	studentid	2021-05-13 21:37:17.85	993248
4	39608	requestedamount	2021-05-13 21:37:17.85	993248

Figura 6.15: Ejemplo de matcheo

Un caso donde el algoritmo se confunde se muestra en la Figura 6.16, en este caso se puede ver como las instancias de atributos matchean con una instancia de variable distinta. Esto se debe a que para una misma instancia de entidad dos de sus atributos se cargaron con el mismo valor y comparten el mismo timestamps, por lo tanto el algoritmo no puede distinguir cuál era el correcto. Pero como se puede ver solo sucede 5 veces, mientras que se matchean correctamente 33.442 veces. Por lo que para esta realidad solo sucede 0,014 % de todos los casos, un porcentaje que no es significativo.

idattributeinstance integer	name character varyi	value character va	timestamp timestamp without time zone	idvarinstance integer	name character varying (45)
373134	amount	3775	2021-05-18 00:06:15.465	57127	order
327038	amount	2523	2021-05-16 21:57:59.351	78148	order
238223	orden	3907	2021-05-14 16:12:20.476	118863	requestedamount
230320	orden	3070	2021-05-14 14:13:46.135	123101	requestedamount
218220	orden	2822	2021-05-14 10:39:58.304	127896	requestedamount

Figura 6.16: Ejemplo de macheo

6.4. Explotación de la solución

A lo largo de esta sección se realiza un análisis sobre la solución propuesta aplicada sobre el caso de estudio, para poder determinar su potencial y sus limitaciones. Para ello, se responde a diferentes preguntas que brindan valor sobre el caso de estudio antes propuesto a través de la solución del data warehouse.

6.4.1. Casos genéricos

En esta sección se desarrollan algunos casos que vinculan datos de la base organizacional y del proceso pero de forma general, puesto que no tienen que ver con el caso de estudio hablado en la sección anterior u otro caso.

Por cuestiones técnicas interesa saber cómo se relacionan los objetos de la organización con las tareas ejecutadas en el proceso. Para lograrlo se puede utilizar el data warehouse genérico brindando resultados como los que se ven en las Figuras 6.17 y 6.18. En la primera de ellas se ve el cruzamiento entre “Entity name” y “Element name” que refieren a los nombres de las tablas de la base organizacional y los nombres de las tareas del proceso respectivamente, de esta forma se puede descubrir qué tablas de la base organizacional tienen relación con tareas del proceso. Por otro lado, la segunda figura agrega al cruzamiento “Attribute name” y “Variable name” que refieren a los nombres de los atributos de las tablas de la base organizacional y los nombres de las variables del proceso respectivamente, con este nuevo cruzamiento se puede descubrir cuáles son las variables y atributos que se relacionan.

Entity name	Element name
application	Approve scholarship results
	Evaluate applications
	Register application
	Requirements assesment
program	Define mobility program call
validation	Register application

Figura 6.17: Nombre de tablas y tareas

A partir del anterior análisis se podría detectar problemas técnicos como, por ejemplo, datos erróneos en la base organizacional de los cuales no se tiene certeza de su procedencia. Con ello se puede saber fácilmente de dónde viene cada atributo y orientar un análisis más específico. Como también es de ayuda para comprender mejor la realidad en las que se está trabajando y poder hacer análisis más específicos como los que se hablarán en la siguiente sección.

Entity name	Attribute name	Element name	Variable name	Idvariableinstance
application	amount	Evaluate applications	order	2
			requestedamount	93
		Register application	requestedamount	5.961
	approved	Approve scholarship results	approve_scholarship_results	182
	idapplication	Requirements assesment	select_applications	187
	idprogram	Register application	selectcourses	1
	idstudent	Evaluate applications	studentid	91
		Register application	studentid	5.961
	orden	Evaluate applications	order	3.250
			requestedamount	3
program	amount	Define mobility program call	amount	99
	callnumber	Define mobility program call	callnumber	99
	idprogram	Define mobility program call	program_id	100
	name	Define mobility program call	programname	99
	year	Define mobility program call	year	99
validation	idcourse	Register application	selectcourses	17.220

Figura 6.18: Nombre de atributos y variables

6.4.2. Casos específicos

Luego de realizar los análisis genéricos, en esta sección se desarrollarán casos de análisis específicos para la realidad trabajada. Se separa por casos a trabajar en los cuales se plantea una pregunta sobre la realidad trabajada y se intenta responder a partir del uso del data warehouse.

Caso: Cursos y Carreras de procesos retrasados

En un contexto en donde se ha notado que algunas ejecuciones de procesos se retrasan más de cierta medida de tiempo, pero no se sabe con exactitud el por qué de esta situación, se intenta plantear distintas conjeturas y confirmarlas a partir de los datos que se tienen en el data warehouse.

Una pregunta que puede surgir a partir de dicha situación es la siguiente:

¿Qué cursos y de qué carreras han estado involucrados en procesos que han llevado más de 2 horas?

Para contestar a la pregunta se hace una consulta sobre el data warehouse que cruza las instancias de las relaciones validation-course a través “id”, con los identificadores de procesos “idcase”, los nombres de atributo “Attribute name” separados por su respectiva entidad “Entity name” y los valores de los atributos “Value”. Además, como se ve en la Figura 6.20, se hace un filtro sobre la medida “Process duration min” para obtener solo procesos que demoran más de 2 horas.

Se puede observar el resultado obtenido en la Figura 6.19. Estos resultados exponen todos los cursos y carreras que fueron partícipes de procesos que se retrasaron. A partir de este resultado la organización podría hacer un análisis más en profundidad para ver si existe algún tipo de problema con esos cursos y carreras que terminen generando un retraso en los procesos.

				Name - Name	validation-course
Id	Idcase	Entity name	Attribute name	Value	Process duration min
40960	1102099	validation	idcourse	173	419,40
	0	validation	idvalidation	28303	0,00
			idapplication	7630	0,00
		course	name	Bases Teóricas Del Concepto De Desarrollo	0,00
			idinstitute	2004	0,00
			idcourse	173	0,00
			idcareer	60	0,00
			credits	10	0,00
40958	1102099	validation	idcourse	189	419,40
	0	validation	idvalidation	28302	0,00
			idapplication	7630	0,00
		course	name	Adolescencia Y Seguridad Pública (Efi)	0,00
			idinstitute	2004	0,00
			idcourse	189	0,00
			idcareer	60	0,00
			credits	8	0,00
40953	1102099	validation	idcourse	1128	419,40
	0	validation	idvalidation	28301	0,00
			idapplication	7629	0,00
		course	name	Electromagnetismo	0,00
			idinstitute	2001	0,00
			idcourse	1128	0,00
			idcareer	25	0,00
			credits	10	0,00

Figura 6.19: Cursos y carreras para procesos que han llevado más de 2 horas

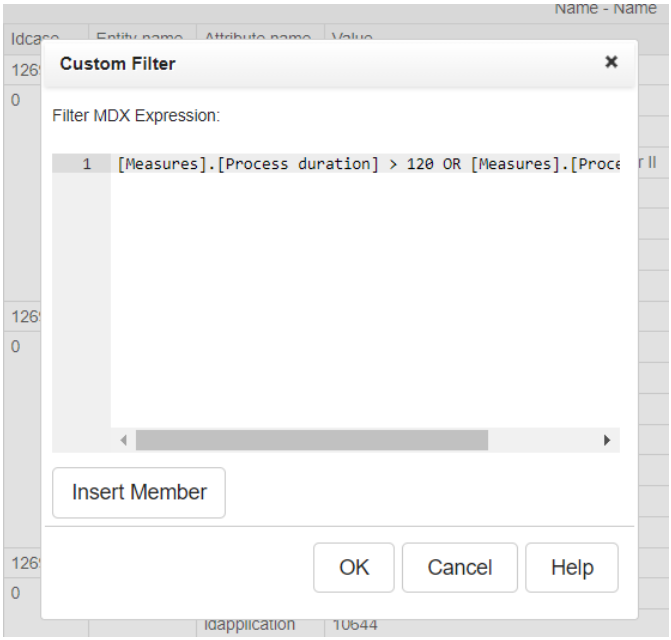


Figura 6.20: Filtro de duración de proceso

Caso: Estudiantes reiterados no aprobados

En un contexto donde se han presentado quejas de distintos estudiantes referentes a que han presentado solicitudes reiteradas veces y no han sido aprobadas, se quiere utilizar los datos para verificar la veracidad de dichas quejas.

A partir de esta situación se puede plantear la siguiente pregunta:

¿Hay estudiantes que hayan participado más de una vez en llamados que no hayan sido aprobados?

Utilizando el data warehouse se puede generar una consulta donde se cruza las instancias de las relaciones application-student a través “id”, con los identificadores de procesos “idcase”, los nombres de atributo “Attribute name”, los valores de los atributos “Value” y los nombres de las tareas en donde participaron “idelementdefinition”.

Los resultados que se obtienen de esta consulta se pueden observar en la Figura 6.21 donde se puede identificar visualmente todos los estudiantes que han participado de procesos separados por las distintas instancias de proceso. A partir del nombre de las tareas se puede saber cuáles estudiantes lograron avanzar en el proceso de evaluación y cuáles no. De esta forma se puede descubrir qué estudiantes han participado más de una vez y fueron rechazados en primera instancia.

				Name - Name	application-student
Id	Idcase	Attribute name	Value	Element name	Process duration min
9547	0	approved	NULL	dummy	0,00
		email	nahuel@gmail.com	dummy	0,00
		idapplication	5392	dummy	0,00
		idcareer	73	dummy	0,00
		idprogram	383	dummy	0,00
		idstudent	74347783	dummy	0,00
		name	Nahuel	dummy	0,00
		notified	NULL	dummy	0,00
		orden	NULL	dummy	0,00
	982502	amount	3201	Register application	53,73
		idstudent	74347783	Register application	53,73
9561	0	approved	NULL	dummy	0,00
		email	lstawella7@foxnews.com	dummy	0,00
		idcareer	23	dummy	0,00
		idprogram	383	dummy	0,00
		idstudent	53785387	dummy	0,00
		name	Lorettalorna	dummy	0,00
		notified	NULL	dummy	0,00
		orden	NULL	dummy	0,00
	982502	amount	2762	Evaluate applications	53,73
		idstudent	53785387	Evaluate applications	53,73

Figura 6.21: Estudiantes que hayan participado más de una vez en llamados que no hayan sido aprobados

Caso: Usuarios que aprobaron más créditos

En un contexto donde dentro de la organización hay problemas sobre la cantidad de créditos que algunos usuarios deben aprobar con respecto a otros, se quiere hacer un análisis de los datos de los pasados procesos para saber cómo se distribuye la aprobación de créditos y reconocer a los usuarios con la mayor cantidad de créditos aprobados.

Por lo que se puede plantear la siguiente pregunta:

¿Qué usuario del panel de evaluación aprobó el llamado con mayor cantidad de créditos a validar?

Para contestar a la pregunta se cruza las instancias de las relaciones application-student a través “id”, con los identificadores de procesos “idcase”, la tarea “Element Name” con nombre “Approve scholarship results” y el nombre de usuario “Username”. Se filtra “Element Name” por ese nombre de tarea, ya que es la encargada de aprobar las movilidades.

Se puede observar el resultado obtenido en la Figura 6.22. De esta forma se pueden ver los usuarios que aprobaron cada instancia de proceso y con una consulta extra se puede saber los créditos de cada curso que participó de dicha instancia.

					Name - Name	application-program
Id	Idcase	Attribute name	Value	Element name	Username	Process duration min
9768	982502	approved	true	Approve scholarship results	Daniel	53,73
9993	982501	approved	true	Approve scholarship results	Julio	57,49
11494	989167	approved	true	Approve scholarship results	Adriana	65,67
13290	993248	approved	true	Approve scholarship results	Raul	56,45
13716	993247	approved	true	Approve scholarship results	Raul	77,15
15183	1000998	approved	true	Approve scholarship results	Carlos	70,59
15393	1000999	approved	true	Approve scholarship results	Adriana	72,90
16952	1006371	approved	true	Approve scholarship results	Libertad	56,88
17158	1006372	approved	true	Approve scholarship results	Daniel	59,01
18740	1015002	approved	true	Approve scholarship results	Julio	57,16
19035	1015001	approved	true	Approve scholarship results	Daniel	61,05
20531	1020131	approved	true	Approve scholarship results	Raul	57,01
20755	1020140	approved	true	Approve scholarship results	Carlos	59,72
21344	1027474	approved	true	Approve scholarship results	Carlos	34,75
21904	1027466	approved	true	Approve scholarship results	Daniel	80,88
23423	1032463	approved	true	Approve scholarship results	Raul	55,61
23637	1032454	approved	true	Approve scholarship results	Raul	56,61
26958	1043791	approved	true	Approve scholarship results	Libertad	55,20
27237	1043792	approved	true	Approve scholarship results	Daniel	57,20
28743	1051071	approved	true	Approve scholarship results	Daniel	90,76

Figura 6.22: Aprobación de créditos por usuario

La consulta extra se hace cruzando las relaciones de entidades “id” con las relaciones validation-course, con las instancias de procesos “idcase”, los nombres atributos “Attribute name” y el valor de los atributos “Value”.

Se puede observar el resultado obtenido de la consulta en la Figura 6.23. A partir de este resultado se pueden ver todos los cursos y créditos que participaron en los procesos y, combinado con el resultado anterior, se puede responder a la pregunta propuesta.

			Name - Name	validation-course
Id	Idcase	Attribute name	Value	Process duration min
8445	0	credits	10	0,00
		idapplication	5270	0,00
		idcareer	59	0,00
		idcourse	4555	0,00
		idinstitute	2004	0,00
		idvalidation	20907	0,00
		name	Sujetos Colectivos y Organizacion Popular II	0,00
	982502	idcourse	4555	53,73
8450	0	credits	12	0,00
		idapplication	5271	0,00
		idcareer	59	0,00
		idcourse	5999	0,00
		idinstitute	2004	0,00
		idvalidation	20908	0,00
		name	Planificacion y Gestion I	0,00
	982501	idcourse	5999	57,49
8452	0	credits	12	0,00
		idapplication	5271	0,00
		idcareer	59	0,00
		idcourse	1555	0,00

Figura 6.23: Créditos por proceso

6.5. Comparación de la solución con solución específica

Como se vio en la sección anterior, la solución genérica tiene algunas limitantes. Debido a esto, en esta sección se pretende presentar una solución que complemente a la solución genérica y sea específica para el caso de estudio Student Mobility. Los casos a analizar son los mismos que se presentaron en la sección anterior para poder explicar que nuevos resultados se pueden obtener.

Caso: Cursos y Carreras de procesos retrasados

Para el caso descrito en la sección anterior también se puede hacer reportes con consultas específicas como se observa en la Figura 6.24.

Figura 6.24: Cursos y carreras para procesos que han llevado más de 2 horas

Como se puede notar, los procesos de ambos casos se corresponden, pero en la solución con consultas específicas la forma de agrupar los datos brinda mayor facilidad para el análisis.

Viendo los datos se observa que en muchos procesos participan más de una carrera y posiblemente se quiera investigar si esto causa problemas. Por lo que se puede plantear la siguiente pregunta:

¿Cuánto dura en promedio la resolución de un llamado con cursos de estudiantes de más de una carrera?

Para contestar esta pregunta se vuelve complicado utilizar el data warehouse genérico, ya que es difícil filtrar instancias de procesos que tengan más de una carrera por cómo están estructurados los datos. Esta pregunta se puede responder a partir de un data warehouse específico o con consultas específicas al metamodelo, donde resulta más simple hacer estos filtros.

En la Figura 6.25 se selecciona el número de carreras mínimo que participan de un proceso y en las Figuras 6.26 y 6.27 se muestra el promedio de la duración de los procesos que cumplen dicha condición de procesos en donde participan más de 1 o 10 carreras. Como también los datos específicos de cada proceso.

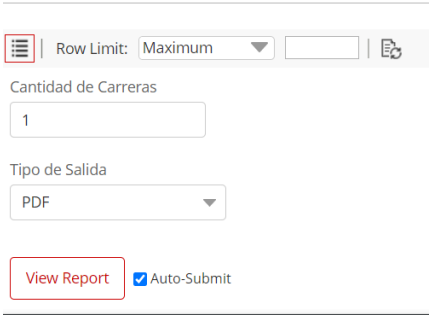


Figura 6.25: Promedio de resolución de llamados con cursos de estudiantes de más de una carrera

Figura 6.26: Promedio de resolución de llamados con cursos de estudiantes de más de una carrera

Figura 6.27: Promedio de resolución de llamados con cursos de estudiantes de 10 o más carreras

Como se puede ver el tiempo promedio varía aproximadamente en dos minutos, y en este caso los procesos con menor cantidad de carreras tienen un promedio de tiempo mayor. Por lo que en principio no se detecta una relación directa en la cantidad de carreras y la duración del proceso.

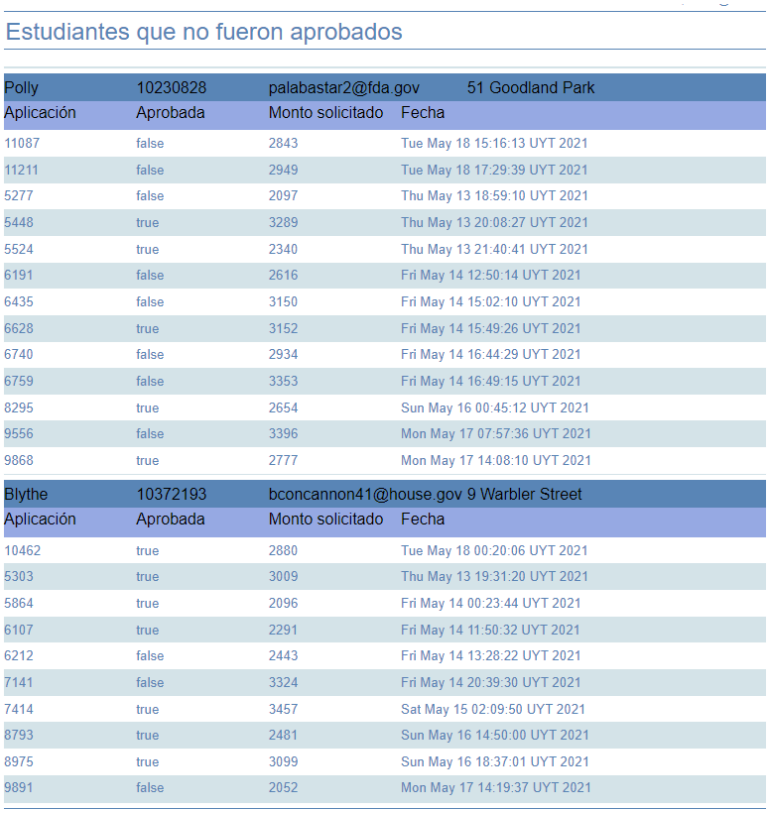
Caso: Estudiantes reiterados no aprobados

Si se quiere realizar una gráfica de estudiantes, en qué procesos participaron y si fueron rechazados se vuelve una tarea imposible para el data warehouse genérico. Esto se debe a que ya no existe la noción de "Estudiante", "aplicación" o "movilidad" sino que se tienen una serie de instancias de entidad, y para cada una de estas instancias se tienen asociadas otras instancias de atributos. Estos atributos están en el mismo nivel de jerarquía, y no se pueden agrupar, por ejemplo, las aplicaciones de un estudiante, ya que tanto estudiante como aplicación son representados como una misma definición, como una entidad. De esta forma se pierde la relación explícita de un estudiante con una aplicación, en su lugar se tiene una instancia de entidad que se relaciona con otra.

Para realizar este tipo de análisis es mejor utilizar consultas específicas y generar reportes como el que se observa en la Figura 6.28. En este reporte se pueden ver todos los estudiantes listados que se han presentado a llamados más de una vez, y han sido rechazados en alguna de estas ocasiones.

Caso: Usuarios que aprobaron más créditos

Desde la perspectiva del proceso podría interesar qué usuario aprobó la mayor cantidad de créditos en programas de movilidad. Esta consulta es muy compleja de resolver en el data



Estudiantes que no fueron aprobados

Polly	10230828	palabastar2@fda.gov	51 Goodland Park
Aplicación	Aprobada	Monto solicitado	Fecha
11087	false	2843	Tue May 18 15:16:13 UYT 2021
11211	false	2949	Tue May 18 17:29:39 UYT 2021
5277	false	2097	Thu May 13 18:59:10 UYT 2021
5448	true	3289	Thu May 13 20:08:27 UYT 2021
5524	true	2340	Thu May 13 21:40:41 UYT 2021
6191	false	2616	Fri May 14 12:50:14 UYT 2021
6435	false	3150	Fri May 14 15:02:10 UYT 2021
6628	true	3152	Fri May 14 15:49:26 UYT 2021
6740	false	2934	Fri May 14 16:44:29 UYT 2021
6759	false	3353	Fri May 14 16:49:15 UYT 2021
8295	true	2654	Sun May 16 00:45:12 UYT 2021
9556	false	3396	Mon May 17 07:57:36 UYT 2021
9868	true	2777	Mon May 17 14:08:10 UYT 2021

Blythe	10372193	bconcannon41@house.gov	9 Warbler Street
Aplicación	Aprobada	Monto solicitado	Fecha
10462	true	2880	Tue May 18 00:20:06 UYT 2021
5303	true	3009	Thu May 13 19:31:20 UYT 2021
5864	true	2096	Fri May 14 00:23:44 UYT 2021
6107	true	2291	Fri May 14 11:50:32 UYT 2021
6212	false	2443	Fri May 14 13:28:22 UYT 2021
7141	false	3324	Fri May 14 20:39:30 UYT 2021
7414	true	3457	Sat May 15 02:09:50 UYT 2021
8793	true	2481	Sun May 16 14:50:00 UYT 2021
8975	true	3099	Sun May 16 18:37:01 UYT 2021
9891	false	2052	Mon May 17 14:19:37 UYT 2021

Figura 6.28: Estudiantes que hayan participado más de una vez en llamados que no hayan sido aprobados

warehouse, por lo tanto, se presenta un reporte específico en donde ver los datos de forma amigable.

En la Figura 6.29 se pueden ver los resultados de esta consulta, mostrando los distintos usuarios con los cursos y créditos que aprobaron por proceso. Además, con una gráfica se puede ver reflejado la cantidad total de créditos por instancia de proceso que cada uno validó.

En este caso particular, que requiere de más de una relación entre las entidades de la base de datos, se muestra el poder de realizar consultas específicas sobre la realidad. Esto es posible, ya que al conocer la realidad y sus relaciones es más sencillo realizar un análisis sobre una relación en específico.

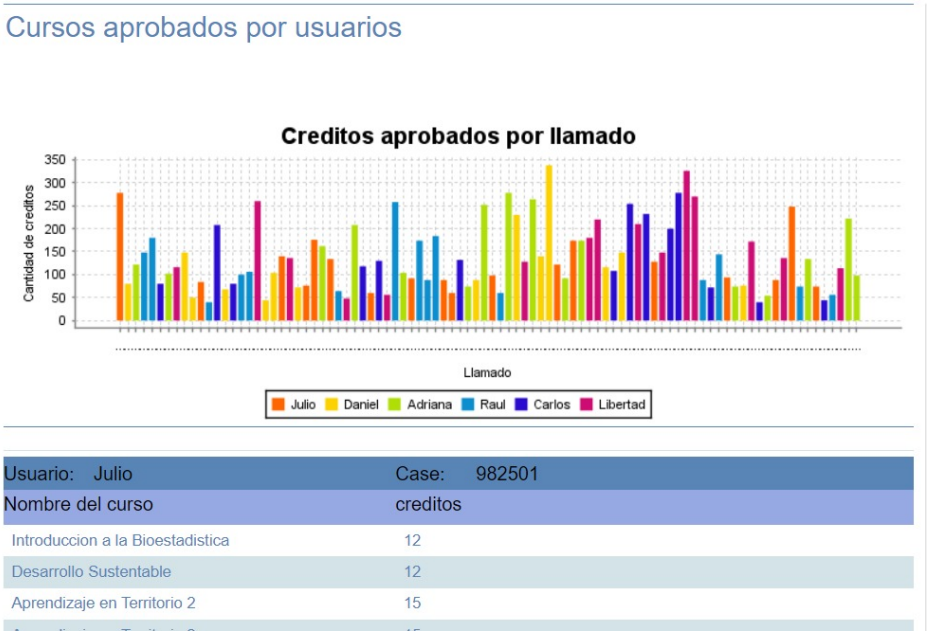


Figura 6.29: Aprobación de créditos por usuario

6.6. Conclusiones del análisis

A partir del data warehouse genérico se puede indagar las relaciones entre los datos de la base organizacional y los procesos, esto significa que se puede analizar cómo evolucionaron los datos en la base organizacional a la medida que el proceso evoluciona. Tiene gran potencia para comprender la relación del proceso con la base organizacional, pero se vuelve complejo hacer consultas específicas sobre el dominio del caso de estudio.

En la generalidad se pierde la estructura de los datos del caso de estudio, pero a partir de consultas genéricas y un post procesamiento se pueden obtener buenos resultados. Tomando un ejemplo del caso Student Mobility trabajado en la sección anterior, se puede ver que para los atributos de una aplicación la foreign key a estudiante no se diferencia de lo que es el monto de la aplicación. Esto ocurre por la estructura genérica y en donde a priori un atributo no se diferencia de otro. Para poder mantener relación entre los datos se requiere agrupar por un nivel superior y esto complica el análisis.

Por estas razones, la solución específica es un buen complemento para combatir las debilidades de la implementación genérica. Siendo posible utilizar el data warehouse genérico como base e incluir consultas y reportes específicos dependiendo de la realidad trabajada.

7 Conclusiones y Trabajo Futuro

El objetivo general de este proyecto fue lograr la conexión entre los datos de los procesos y bases organizacionales a partir de la solución propuesta en [5]. Se cumplió el flujo completo del problema planteado, abarcando la extracción y carga de los datos, la implementación del metamodelo y la generación de una herramienta de análisis. Finalmente, se verificó el correcto funcionamiento de la solución implementada aplicándola a un caso de estudio.

Se implementó el metamodelo en una base de datos PostgreSQL 12 y un algoritmo en Java para realizar la extracción y carga. Este algoritmo es totalmente genérico por lo que se puede aplicar a cualquier tipo de proceso implementado en Activiti y cualquier base organizacional PostgreSQL, para esto alcanza con tener configurada la conexión de la base del proceso y los logs de la base organizacional. Esto permite extraer y cargar los datos de una forma sencilla, y sin tener que generar configuraciones específicas para la realidad trabajada. Para trabajo a futuro se plantea extender esta solución para soportar cualquier tipo de motor de procesos y para todos los tipos de logs de los diferentes manejadores de base de datos. Una buena solución sería abstraer los formatos de las distintas bases de los procesos a un formato único para ser utilizada como input del algoritmo.

Un algoritmo de matcheo fue implementado para generar la conexión entre los datos de los procesos y de las bases organizacionales. A partir de los datos almacenados en la base del metamodelo el algoritmo descubre las conexiones existentes entre los datos del proceso y de la organización, utilizando los valores y una ventana de tiempo definida a partir de los timestamps de las variables y atributos. Este algoritmo tiene éxito y, aunque se encontraron casos de borde, permitió validar que es posible encontrar la conexión entre los datos. Para implementaciones futuras se propone brindar al algoritmo de una inteligencia para reconocer patrones de matcheo a medida que se van descubriendo. De esta forma es posible superar casos de borde en donde hoy en día el algoritmo falla, como por ejemplo cuando los timestamps entre los datos son muy cercanos y las variables tienen el mismo valor.

Se construyó un data warehouse basado en la definición de los cuadrantes y relaciones del

metamodelo. Vale destacar que gracias a esta definición el data warehouse es completamente genérico, para cualquier proceso y organización. A partir de las dimensiones y métricas definidas se pueden hacer análisis que combinen datos del proceso y de la organización, lo que provee una visión ampliada a partir de la cual las diferentes organizaciones pueden beneficiarse. A trabajo a futuro, sería interesante descubrir necesidades de diferentes organizaciones y utilizarlas para desarrollar nuevas dimensiones y métricas.

Para mostrar el uso de la solución se implementó un caso de estudio del proceso Student Mobility. En primer lugar, se implementó el proceso en la plataforma Activiti y la base organizacional en PostgreSQL 12. Se generaron datos organizacionales pensando en imitar la realidad, como por ejemplo diversos usuarios con nombres, direcciones, fechas y documentos reales. Además, se desarrolló un algoritmo para generar instancias de ejecución del proceso que permitió generar 100 instancias de proceso y más de 200.000 instancias de atributos distintas. Esto contribuyó a tener datos muy parecidos a los que un proceso real puede tener y confirmar la efectividad de la solución como también descubrir algunas falencias. A futuro, se plantea utilizar la solución con un caso de estudio real.

Como conclusión final, se puede decir que se cumplió con todos los objetivos pertinentes al proyecto. El único objetivo no cumplido es el de utilizar datos reales, pero se trabajó mucho para generar datos muy similares a la realidad y se toma este desafío como un trabajo a futuro. Se confirma la posibilidad de extraer los datos, cargarlos en un metamodelo unificador, identificar la conexión entre las partes y finalmente poder explotar estos datos a partir de un data warehouse. Además, durante el desarrollo del proyecto se publicaron dos artículos [6] y [7], en el primero se presenta una descripción del proyecto, mientras que en el segundo se centra en la solución propuesta para el data warehouse.

Referencias

[1] W. Mathias. Business Process Management - Concepts, Languages, Architectures, Third Edition. *Springer*, 2019.

[2] J. Chang. Business Process Management Systems: Strategy and Implementation. *CRC Press*, 2016.

[3] Wil M. P. van der Aalst. Process Mining - Data Science in Action, Second Edition. *Springer*, 2016.

[4] M. Golfarelli, S. Rizzi, M. Hill. Data Warehouse Design: Modern Principles and Methodologies, 2009.

[5] A. Delgado, D. Calegari. Towards a unified vision of business process and organizational data. *2020 XLV Latin American Computing Conference (CLEI)*.

[6] A. Delgado, D. Calegari, A. Artus, A. Borges. Integration of business process and organizational data for evidence-based business intelligence. *Aceptado para publicación en CLEI Electronic Journal 2021*.

[7] A. Delgado, D. Calegari, A. Artus, A. Borges. Integrated process and organizational data analysis for business process improvement. *Aceptado para publicación como short paper en 23rd International Conference on Big Data Analytics and Knowledge Discovery, DaWaK 2021*.

[8] OMG Business Process Model and Notaton (BPMN 2.0), version 2.0, Object Management Group (OMG).

[9] Activiti
https://www.activiti.org/ visitado el 07/03/2021

[10] Activiti User Guide
https://www.activiti.org/userguide/ visitado el 07/03/2021

[11] G. Roth. PostgreSQL Logging
https://wiki.postgresql.org/images/9/9d/Logging_pgopen_withnotes.pdf visita-

do el 12/03/2021

[12] Pentaho
https://www.hitachivantara.com/es-latam/products/data-management-analytics/pentaho-platform.html visitado el 15/03/2021

Anexos

A Generación de instancias

Para ejecutar instancias de proceso manualmente se requiere de algunos minutos, por lo que generar datos de esta forma sería un trabajo tedioso. Para no desperdiciar tiempo se implementa un proyecto en Java que genera un número de instancias configurables. Este generador puede ser utilizado para generar instancias de cualquier proceso, ya que su base es genérica, aunque se puede extender para agregar ciertas condiciones para los datos de la realidad a trabajar.

En la Figura A.1 se representa el flujo que realiza el generador para su caso más genérico, sin tomar en cuenta los casos especiales.

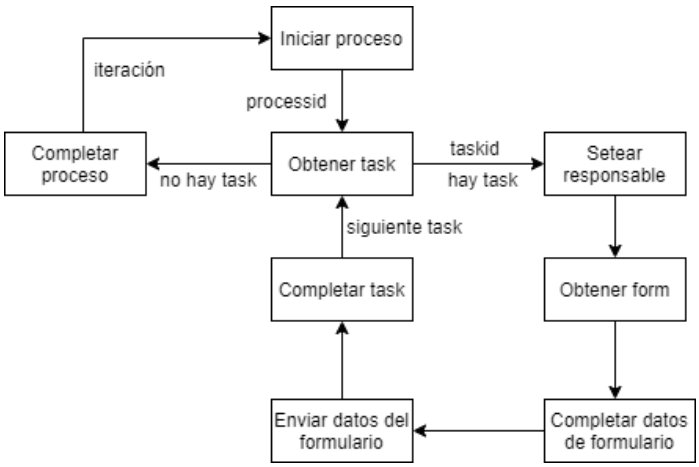


Figura A.1: Diagrama del generador

Lo primero a realizar es la inicialización de un nuevo proceso a partir de su process key. Para ello se hace un post al servicio "/activiti-rest/service/runtime/process-instances" de Activiti Rest, pasando como parámetro dentro del body el process key. Este servicio retorna un processInstanceId con el cual se trabajará el resto de la ejecución de esa instancia de proceso.

A partir del processInstanceId, comienza el loop de completado de tasks. Como primer paso se obtiene la task actual de la instancia de proceso a través del servicio "/activiti-rest/service/runtime/tasks" de Activiti Rest, pasando como parámetro el valor de processInstanceId.

En caso de que no se obtenga ninguna task nueva significa que el proceso ha terminado y se puede continuar con la siguiente iteración de proceso si es que queda alguna pendiente, en caso contrario el generador termina su ejecución.

En el supuesto de que haya una nueva task, se obtiene a partir del servicio post antes mencionado el taskId, taskDefinitionKey y formKey.

El siguiente paso a realizar es la asignación del responsable que va a completar la task. Para ello se utilizan los valores taskId y taskDefinitionKey, con el taskDefinitionKey se consulta los usuarios responsables para la task en el mapeo de usuarios con tasks. A partir de un número random se selecciona uno de ellos y se le asigna a la task. La asignación se hace a partir de un put del servicio "/activiti-rest/service/runtime/tasks/taskId" de Activiti Rest donde se pasa como parámetro el taskId.

Después del paso anterior se continúa con la obtención del formulario para la task actual, si es lo que tiene. A partir del formKey se hace una consulta a la base de datos de Activiti para obtener la información del formulario. A partir de un select como se muestra en A.1.

Listing A.1: Obtener formulario

```
1 select resource_bytes_
2 from act_fo_form_resource
3 where name_ = '"form-'+ formKey + '.form"'
4 order by pg_xact_commit_timestamp(xmin) desc limit 1;
```

De esta manera se obtiene la información de la última versión del formulario y con la cual se puede continuar con los siguientes pasos.

Siguiendo con el flujo se realiza una iteración sobre todos los campos del formulario, consultando el tipo de cada uno y completándolos con un valor random del tipo correspondiente.

Después de completar todos los campos del formulario, se prosigue a enviar los datos a partir de un post al servicio "/activiti-rest/service/form/form-data" de Activiti Rest.

Para este request hay que incluir un body con la taskId de la task actual y todos los valores del formulario.

Para finalizar el proceso de completado de una task se hace el complete a través de un post al

servicio “/activiti-rest/service/runtime/tasks/taskId” utilizando el taskId de la task actual.

A partir de este punto se comienza de nuevo con la petición de una nueva task y en caso de que no exista una nueva se completa el proceso.

B Manual de Desarrollador: Carga del metamodelo

B.1. Herramientas necesarias

- Java 8
- Maven 3.6.3
- Postgres 12

B.2. Configuración

Como primer paso se debe clonar el repositorio del proyecto desde la siguiente url:

`https://gitlab.fing.edu.uy/juan.borges/pmdatos.git`.

En este repositorio está el proyecto para la carga de metamodelo dentro del folder llamado EtlMetaModel. En este se encuentra todo el código necesario para hacer la carga total del metamodelo.

Lo siguiente es crear la base de datos del metamodelo, para esto se cuenta con el script de creación EtlMetaModel/DumpDb/dumpDb.sql

Se recomienda crear esta base de datos con el nombre "metamodelo", ya que como se verá en D es la configuración por defecto en la carga del data warehouse.

En el archivo EtlMetaModel/src/main/resources/application.properties se configuran las propiedades:

- server.port: Puerto donde se levanta la aplicación

- spring.datasource.url= Url de la base de datos del metamodelo
- spring.datasource.username= Usuario de la base de datos del metamodelo
- spring.datasource.password= Contraseña de la base de datos del metamodelo
- archivos.path= Ruta de la carpeta para tomar logs de la base organizacional
- process.datasource.url= Url de la base de datos de procesos
- process.datasource.username= Usuario de la base de datos de procesos
- process.datasource.password= Contraseña de la base de datos de procesos
- data.datasource.url= Url de la base organizacional
- data.datasource.username= Usuario de la base organizacional
- data.datasource.password= Contraseña de la base organizacional
- time.rango.milisecond = Rango del matcheo en milisegundos
- logs.path= Ruta para los logs de la base organizacional
- xml.path= Ruta para el xml del proceso
- recursive.count= Cantidad de relaciones a cargar

Luego de configurar es necesario compilar el proyecto, y finalmente levantar el servidor para la carga, para esto basta con los comandos que se muestran en B.1.

Listing B.1: Compilar y levatar servidor de carga

```
1 mvn clean install
2 mvn spring-boot:run
```

Finalmente para realizar la carga se debe hacer por separado en tres partes, y se ejecuta a través de una api rest:

- Carga de datos: GET host:puerto/carga/datos
- Carga de procesos: GET host:puerto/carga/proceso
- Matcheo de datos: GET host:puerto/match

C Manual de Desarrollador: Generador

C.1. Herramientas necesarias

- Java 8
- Maven 3.6.3

C.2. Configuración

El primer paso es clonar el repositorio desde `https://gitlab.fing.edu.uy/juan.borges/pmdatos.git`.

En este repositorio está el proyecto para la carga de metamodelo llamado GeneradorMobility.

En el archivo generadorMobility/src/main/java/constantes/Constantes.java se configuran las constantes:

- host = host donde se encuentra la aplicación de Activiti.
- processKey = clave del proceso para el cual interesa generar datos.
- user = nombre de usuario administrador del proceso.
- password = contraseña del usuario administrador.
- dbUrlProceso =url de la base de datos del proceso.
- dbUsernameProceso = usuario de la base de datos del proceso.

- dbPasswordProceso = contraseña de la base de datos del proceso.
- dbUrlDatos = url de la base de datos de la organización.
- dbUsernameDatos = usuario de la base de datos de la organización.
- dbPasswordDatos = contraseña de la base de datos de la organización.
- cantThread= cantidad de hilos simultáneos ejecutando la carga.
- cantInstancias= cantidad de instancias de procesos a cargar por hilo.

Luego de configurar el proyecto es necesario compilar el proyecto, y finalmente ejecutar la carga, para esto basta con los comandos que se muestran en C.1.

Listing C.1: Compilar y ejecutar generador

```
1 mvn clean install
2 mvn exec:java
```

D

Manual de Desarrollador: Data warehouse

En este capítulo se mostrarán todas las herramientas y pasos necesarios para la configuración y ejecución del data warehouse implementado en el proyecto.

D.1. Herramientas necesarias

Todas las herramientas utilizadas perteneces a la organización Pentaho. Las utilizadas para este proyecto fueron:

- Pentaho Kettle: Carga del las tablas de dimensiones y tabla de hecho a partir del meta-modelo.
- Pentaho Server: Data warehouse a partir de las dimensiones y tabla de hechos.
- Saiku plugin: Plugin para visualizar y analizar data warehouse en Pentaho Server.

Todas las herramientas mencionadas puede ser descargadas en el siguiente link de forma gratuita.

Pentaho

`https://sourceforge.net/projects/pentaho/`

D.2. Carga y Configuración

Los pasos para la configuración y ejecución se pueden dividir en dos etapas: Carga y configuración del data warehouse. Para la correcta comprensión se explicará detalladamente en

las siguientes secciones como ejecutar cada uno.

Para comenzar con los pasos algunos prerequisitos son necesarios:

- Proyecto descargado localmente.
- Herramientas de Pentaho.
- Base del metamodelo completamente cargada.

Si se tiene cada uno de los puntos anteriores se puede comenzar con las siguientes secciones sin ningún problema.

D.2.1. Carga de data warehouse

Para realizar la carga del data warehouse se utiliza la herramienta Pentaho Kettle a través de la ejecución de un Job que ejecuta distintas Transformations para cargar la base del data warehouse a partir de la base del metamodelo.

Los pasos para la carga son los siguientes:

1. Configurar base de datos del data warehouse. Un esquema puede ser encontrado dentro del proyecto bajo el folder “Base”. Fue implementado en una base de datos Postgres pero puede ser traducido y utilizado para cualquier tipo de base de datos relacional, ya que el tipo de base no es una restricción
2. Configurar base del metamodelo y del data warehouse globalmente en Pentaho Kettle. En la Figura D.1 se explica donde se encuentra la configuración global de las bases dentro de la herramienta. Se debe crear una para cada una de las bases y es necesario utilizar los nombres “metamodelo” y “datawarehouse” en las configuraciones. Si se desea utilizar otro nombre se deben cambiar también las configuraciones dentro de las Transformations, por lo cual sugerimos ampliamente utilizar los recomendados.
3. Ejecutar el Job llamado “insertAll”. Este job y las distintas Transformations se encuentran en el repositorio del proyecto bajo el folder “DataWarehouse”, se necesita abrir únicamente el archivo llamado “insertAll” y comenzar la ejecución.

En este punto si cada paso fue ejecutado con éxito se tiene la base de datos del data warehouse completamente cargada, el siguiente paso necesario es configurar el cubo en Pentaho Server.

D.2.2. Configuración de data warehouse

Para hacer la configuración del data warehouse se utiliza la herramienta Pentaho Server, en donde se puede hacer una conexión con la base de datos y utilizar distintas herramientas de

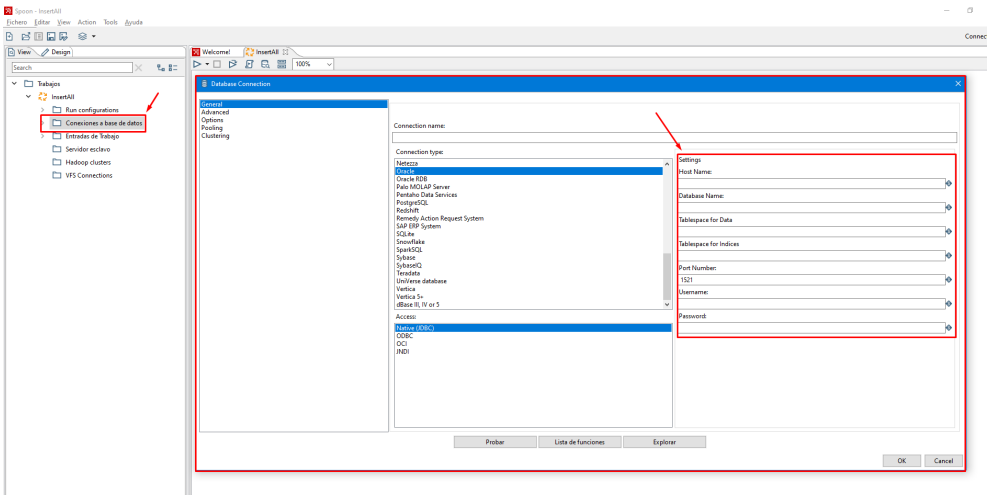


Figura D.1: Configuración de base de datos en Kettle

análisis.

Los pasos para la configuración son los siguientes:

1. Configurar la conexión con la base del data warehouse en Pentaho Server. Para ello se debe tener levantada una instancia del servidor y seguir los pasos que se ven en la Figura D.2.

2. Configurar el Data Source. Añadir un nuevo Data Source en la misma configuración de la conexión con la base y seleccionar la base de datos antes configurada. Además, seleccionar la opción “Reporting and Analysis (Requires Star Schema)” que permitirá configurar el cubo.

3. Configurar el cubo. En la configuración que continúa al paso anterior, seleccionar las tablas que pertenecen a las distintas dimensiones del cubo y en el drop down inferior seleccionar la tabla que contiene la fact table. Continuar al siguiente paso y finalizar.

4. Verificar que la configuración haya sido exitosa. Se puede utilizar la herramienta Saiku Analytics utilizando el boton “Create New” y luego “Saiku Analytics”. Luego se selecciona “Create a new query” y si la configuración fue correcta se debería ver como en la Figura D.3.

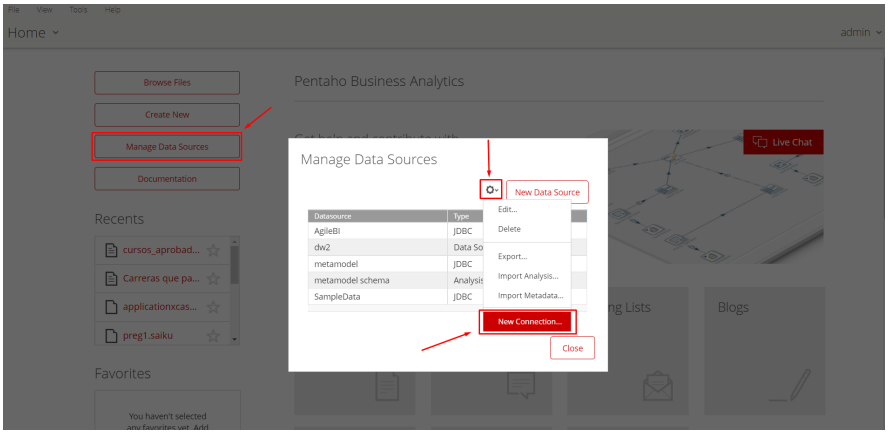


Figura D.2: Manage Data Source

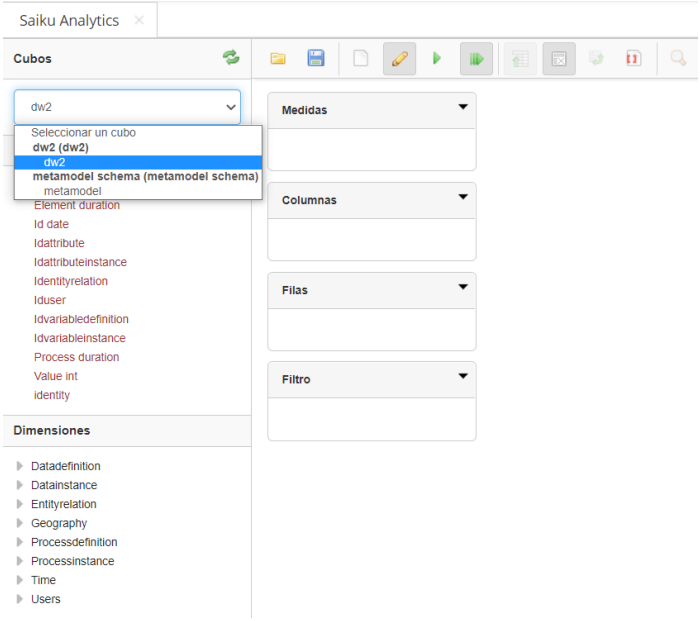


Figura D.3: Verificación en Saiku Analytics

Luego de finalizados estos pasos el cubo está listo para usarse.

E

Manual de Usuario: Data warehouse

En este capítulo se mostrará al usuario como se utiliza el data warehouse para realizar los análisis que se deseen. Para ello se explicarán los diferentes componentes de la herramienta Saiku Analytics y como se conectan para generar los análisis.

Antes de comenzar se debe tener una instancia del servidor corriendo que contenga la configuración mencionada en D. Con ese requisito se puede proseguir a crear una ventana Saiku Analytics, crear una query y seleccionar el cubo cargado.

En esta ventana se tienen de las dimensiones y métricas, utilizadas para los análisis. En la Figura E.1 se puede apreciar a la izquierda las diferentes dimensiones que puede ser seleccionadas y utilizadas como columnas, filas, o filtros. En la misma sección se tienen las métricas que se pueden seleccionar y utilizar como la medida resultante del cruce de las dimensiones antes seleccionadas.

Además, sobre el cruce y medidas seleccionas se pueden utilizar filtros predefinidos para filtrar la información resultante, como también crear filtros personalizados con mdx. En la Figura E.2 se puede apreciar un filtro personalizado para filtrar la métrica de duración de proceso con los procesos de duración mayor a 2 horas.

Los resultados se pueden obtener oprimiendo el botón de “play” de la barra superior mostrando los resultados como se puede apreciar en la Figura E.3. Se puede ver la tabla resultante del cruce de las dimensiones, columnas y filas, y en la derecha el resultado de la métrica seleccionada.

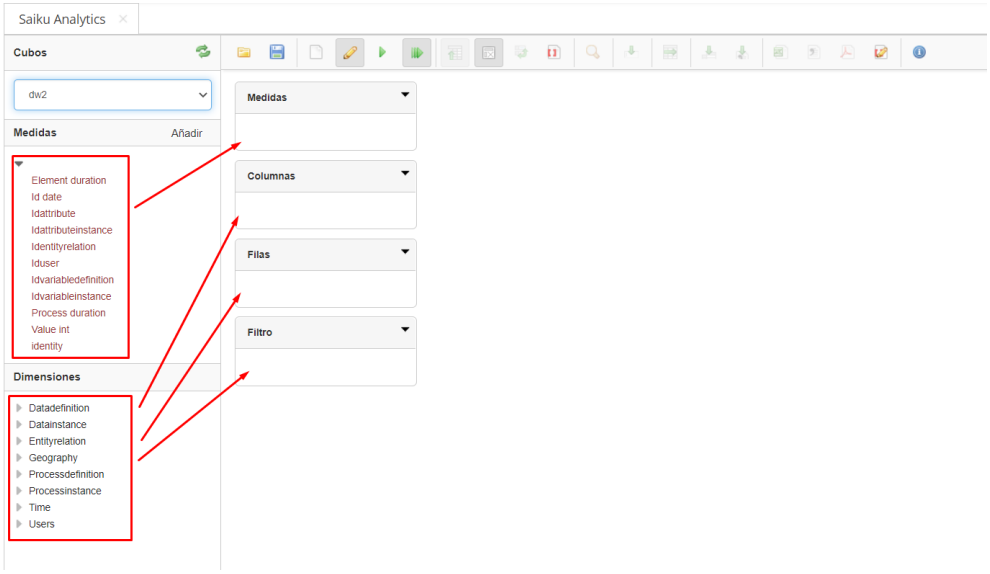


Figura E.1: Dimensiones y medidas

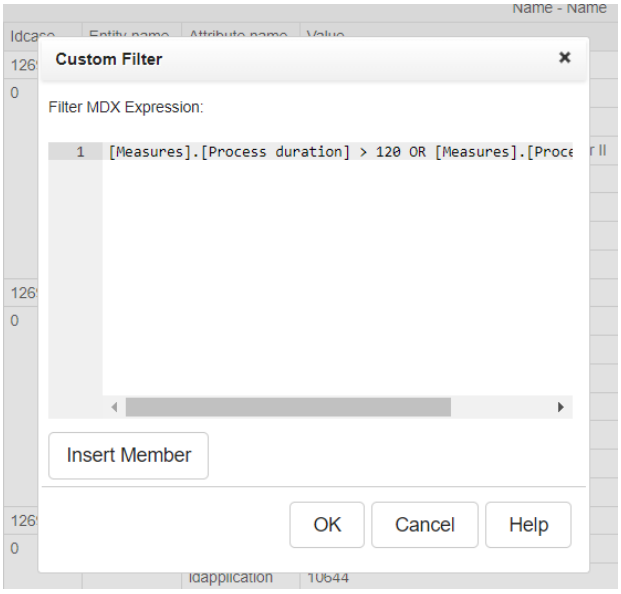
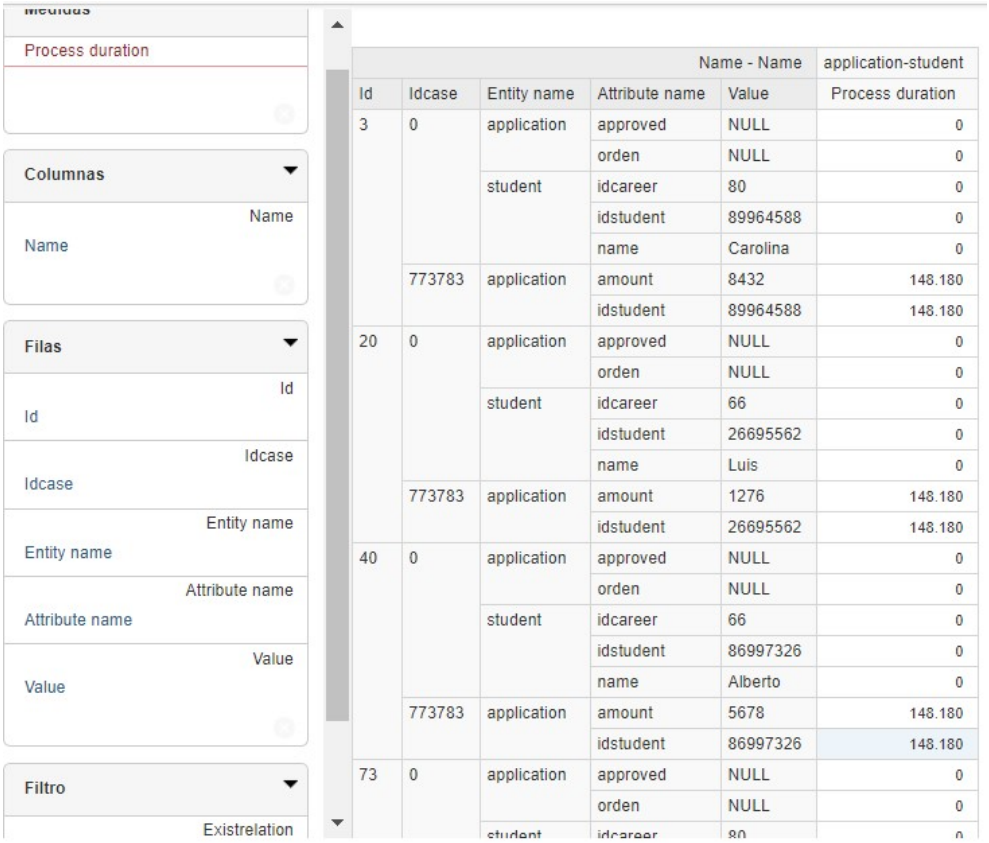


Figura E.2: Filtro con mdx



					Name - Name	application-student
Id	Idcase	Entity name	Attribute name	Value	Process duration	
3	0	application	approved	NULL	0	
			orden	NULL	0	
		student	idcareer	80	0	
			idstudent	89964588	0	
			name	Carolina	0	
	773783	application	amount	8432	148.180	
			idstudent	89964588	148.180	
20	0	application	approved	NULL	0	
			orden	NULL	0	
		student	idcareer	66	0	
			idstudent	26695562	0	
			name	Luis	0	
	773783	application	amount	1276	148.180	
			idstudent	26695562	148.180	
40	0	application	approved	NULL	0	
			orden	NULL	0	
		student	idcareer	66	0	
			idstudent	86997326	0	
			name	Alberto	0	
	773783	application	amount	5678	148.180	
			idstudent	86997326	148.180	
73	0	application	approved	NULL	0	
			orden	NULL	0	
		student	idcareer	80	0	

Figura E.3: Resultados de consulta