Universidad Autónoma de Baja California

Instituto de Ingeniería

Tesis:

“Desarrollo de una plataforma de software para la gestión de datos climáticos en malla”

Presenta:

L.S.C. Jesús Donaldo Osornio Hernández

Para obtener el grado de:

Maestría en Ingeniería

Director de tesis:

Dr. Gabriel López Morteo

Codirector de tesis:

Dr. Francisco Muñoz-Arriola

Mexicali, Baja California, Junio del 2018

Resumen

Este trabajo presenta Dataplugin, un almacén de datos desarrollado en Java con el propósito de mitigar algunas de las necesidades presentadas al trabajar con datos climáticos en malla, provenientes de fuentes heterogéneas. La arquitectura de Dataplugin se basa en conceptos de *plugins* y procesos *Extract-Transform-Load* (ETL) para importar datos al almacén, el cuál utiliza *Database Management Systems* (DBMSs) como la capa de almacenamiento. La meta final de Dataplugin es proveer una interface de acceso para obtener los datos importados al almacén de manera uniforme, y así construir clientes de software que consuman dichos datos y provean funcionalidades que satisfagan las necesidades planteadas. Dataplugin es diseñado para trabajar en un ambiente centralizado y permitir flexibilidad de tecnología, por lo que es posible utilizar distintos DBMSs o lenguajes de script para el proceso de importación. En este trabajo se presenta un caso de construcción de un *plugin* para importar una muestra del conjunto de datos *Tropical Rainfall Measurement Mission* (TRMM), distribuido en formato NetCDF, para probar la arquitectura construida utilizando PostgreSQL y su extensión PostGIS, MongoDB y Python 3. Este trabajo presenta como aportaciones la arquitectura de Dataplugin, así como la definición de los procesos de importación y exportación de datos, documentación sobre la construcción de *plugins*, un *plugin* para importar datos de TRMM y un esquema de datos para almacenar datos en malla en el DBMS PostgreSQL. Conforme a los resultados de las pruebas realizadas a la arquitectura, ésta será benéfica para aquellos usuarios e investigadores que se enfrentan con los problemas ya mencionados al tratar con datos en malla.

Abstract

This work introduces Dataplugin, a Java developed data warehouse with the purpose of mitigating some of the needs found when working with gridded climatic data, coming from heterogeneous sources. The Dataplugin architecture is based on plugin concepts and Extract-Transform-Load (ETL) processes to import data into the warehouse, which uses Database Management Systems (DBMSs) as the storage layer. Dataplugin final goal is to provide an interface to access the data imported into the warehouse in a uniform way, and then build software clients that consume said data and provide the functionality to satisfy the presented needs. Dataplugin is designed to work in a centralized environment and to allow technology flexibility. Because of that, it's possible to utilize different DBMSs or scripting languages in the import process. This work presents a plugin build case to import a sample from the Tropical Rainfall Measurement Mission (TRMM) dataset, delivered in NetCDF file format, with the purpose of testing architecture using PostgreSQL and PostGIS extension, MongoDB and Python 3. This work provides as its contributions the Dataplugin architecture, data import and export process definitions, plugin building documentation, a plugin to import TRMM data, and a data schema for storing gridded data in the PostgreSQL DBMS. According to the results of the tests applied to the architecture, it will be beneficent for those users and researchers that face the problems already mentioned when working with gridded data.

TABLA DE CONTENIDOS

Lista de Figuras

Lista de Tablas

I. Acrónimos

CPC: Climate Prediction Center

CWOP: Citizen Weather Observer Program

DBMS: Database Management System

EMA: Estación Meteorológica Automática

ESIME: Estación Sinóptica Meteorológica Automática

ETL: Extract, Transform, Load

GDAL: Geospatial Data Abstraction Library

GIS: Geographic Information System

GrADS: Grid Analysis and Display System

HDF: Hierarchical Data Format

HIH: Hydroinformatics and Integrated Hydroclimate

IMD: India Meteorological Department

JSON: JavaScript Object Notation

MSC: Meteorological Service of Canada

NASA: National Aeronautics and Space Administration

NetCDF: Network Common Data Form

NOAA: National Oceanic and Atmospheric Administration

NWS: National Weather Service (USA)

OGD: Open Government Data

SMN: Servicio Meteorológico Nacional

TRMM: Tropical Rainfall Measurement Mission

YAML: YAML Ain't Markup Language

1 Introducción

En estos últimos años, los datos y la información han cobrado fuerza como uno de los activos más importantes para las empresas, academia y gobiernos. Tal es la importancia que algunos países como Estados Unidos y México se han unido a la iniciativa de los datos abiertos, la cual consiste en que las instituciones públicas mantengan disponibles en sitios Web los datos generados en diferentes sectores para que cualquiera pueda usarlos. Entre los sectores en común que estos dos países mantienen en sus sitios Web se encuentran la economía, la educación, la salud y el clima ("data.gov", 2018a; "datos.gob.mx", 2018).

Esta iniciativa es sólo un indicio de cómo la generación de los datos, almacenamiento y disponibilidad de éstos se ha vuelto de suma importancia tanto en la industria como en la academia y el público en general. Desde 2013, el gobierno mexicano ha reconocido a los datos abiertos gubernamentales (OGD, por sus siglas en inglés) como una prioridad política, subrayando su importancia para la transformación digital del sector público mexicano. Esta política puede reforzar la reutilización de los datos, productividad del sector público, e innovación empresarial impulsada por los datos (OECD, 2018). Los OGD son importantes porque entre más accesibles, descubribles, y utilizables sean, más impacto podrán tener. Estos impactos incluyen: ahorro de costos, eficiencia, combustible para empresas, política informada, descubrimientos científicos e investigativos, transparencia, entre otros ("data.gov", 2018b). El sitio que implementa la iniciativa de datos abiertos en los Estados Unidos es https://www.data.gov/, mientras que el equivalente en México es http://datos.gob.mx/.

En la actualidad, se genera un número incalculable de datos diariamente, desde los clicks que un usuario realizó en un sitio Web, hasta la cantidad de dinero transferido en las transacciones bancarias realizadas ese día. Pese a la gran cantidad de datos que se generan, no todos éstos se encuentran disponibles para el público como los que contempla la iniciativa de datos

abiertos; por ejemplo, es razonable que una tienda departamental no comparta los hábitos de compra de sus clientes con el público, ya que la competencia podría hacer uso de esta información para su beneficio; sin embargo, existen otros tipos de datos que son de interés general y que deben mantenerse disponibles como es el caso de los datos climáticos.

Cuando se habla de datos climáticos se hace referencia a los datos históricos generados por estaciones climáticas y demás sensores remotos cuya función es medir o estimar el comportamiento de diferentes variables meteorológicas. Las estaciones climáticas son instrumentos y equipos que miden las condiciones atmosféricas para la predicción del clima y el estudio del estado del tiempo. Estos instrumentos se encuentran instalados por tierra y mar; en particular, las variables que se miden en tierra incluyen temperatura del aire, humedad, velocidad del viento, dirección del viento y precipitación, entre otras. (Weather Underground, 2016).

Las estaciones climáticas son equipos y sensores proximales instalados tanto en la tierra como en el mar que realizan mediciones directas o indirectas. También existen sensores remotos, representados por satélites (plataformas que albergan sensores de tipo pasivo y activo) de observación que se encuentran estimando constantemente el comportamiento de las variables meteorológicas en la superficie de la Tierra, a través de mediciones remotas de radiaciones emitidas por la superficie terrestre de manera directa o en respuesta de una emisión lanzada desde el espacio.

El ecosistema de estaciones climáticas y sensores remotos cuenta con un sinnúmero de opciones para medir las condiciones atmosféricas y de superficie terrestre, variando tanto en marca, como en tecnología, tamaño, condiciones de ubicación, instrumentos utilizados para la medición de variables, y el formato en que se almacenan o envían los datos generados. Existen desde simples dispositivos móviles para realizar campañas de medición, llamados *data loggers*,

hasta dispositivos mucho más complejos como los satélites, que estiman todo un conjunto de variables en base a imágenes infrarrojas. Por otro lado, las variables meteorológicas son aquellas condiciones atmosféricas y de superficie terrestre que son medidas o estimadas periódicamente por las estaciones climáticas y sensores remotos ya mencionados. Las variables más comúnmente medidas por la mayoría de las estaciones son temperatura del aire, humedad relativa, velocidad del viento y precipitación.

Así como los sitios Web oficiales de datos abiertos ya mencionados, también existen muchas otras fuentes que siguen la misma tendencia, sólo que en el sector específico de la climatología y meteorología. Distintas agencias nacionales e internacionales proveen datos propios a través de sus sitios Web. En México se tiene al Servicio Meteorológico Nacional (SMN), en Estados Unidos se cuenta con la *National Oceanic and Atmospheric Administration* (NOAA), y en otros países como la India, se cuenta también con el *India Meteorological Department* (IMD). Además de publicar los datos generados, las agencias suelen también ofrecer servicios a la población. Por ejemplo: el pronóstico del tiempo, imágenes de satélite, monitoreo y alerta de sequías, inundaciones, tornados, frentes fríos, ondas de calor, y demás inclemencias. Tal es el caso del SMN y el *National Weather Service* (NWS) de la NOAA.

En lo que respecta a la academia, existen infinidad de posibles usos y aplicaciones para los datos climáticos: como entrada para modelos hidrológicos o de otro tipo, para estudios de comparación entre observaciones en la superficie y estimaciones satelitales, análisis de habilidad de sistemas de predicciones a mediano y largo plazo, cálculo de periodos de retorno de eventos extremos, interpolación, extrapolación, re-muestreo, cálculo de variables derivadas a partir de otras, y generación de índices climáticos, por mencionar algunos ejemplos.

Ya se ha hablado un poco sobre la importancia de los datos como activo para el sector público y privado, específicamente de los datos climáticos. Se describieron los conceptos de

estaciones climáticas, sensores remotos y variables meteorológicas, así como se hizo mención de algunas agencias encargadas de mantener disponibles estos datos. Tener todo esto a disposición de cualquier investigador o interesado es sin duda un gran estímulo al desarrollo de la investigación en el área; sin embargo, esto no les exenta problemas. En la siguiente sección se habla al respecto.

1.1 Definición del problema

Contar con la iniciativa de datos abiertos es una gran ventaja, ya que un investigador interesado en estudiar un fenómeno del clima en particular, tiene a su disposición una plétora de datos de dónde elegir, siendo fuentes potenciales los sitios Web ya abordados u otras agencias climáticas que sigan la misma tendencia. Además de la disponibilidad de datos públicos, también se pueden obtener datos de estaciones climáticas locales (de una institución o de un *data logger* privado) si así se requiriera. El reto para el acceso al universo de datos mencionado arriba radica en disponer de los datos y tener los medios necesarios para procesarlos o utilizarlos. Para comprender mejor el escenario, hay que mencionar las principales características de los datos climáticos.

En computación al conjunto de datos que pueden ser manipulados como una unidad se le conoce como *dataset*. Gran parte de los *datasets* climáticos se caracterizan por su gran volumen, variabilidad y velocidad de crecimiento. El tamaño de estos *datasets* puede fácilmente llegar al orden de los terabytes debido a factores como las variables manejadas, el área comprendida, la resolución espacial, la frecuencia de actualización, lo longevo del *dataset* y el formato de archivo utilizado.

La frecuencia de actualización varía de un *dataset* a otro. Esta puede ser desde nula, hasta casi en tiempo real. Por ejemplo, estaciones climáticas como las distribuidas por la marca

Campbell Scientifics, pueden generar datos cada día, hora, media hora o incluso cada minuto. Otros productos climáticos como los *datasets* ofrecidos por la *National Aeronautics and Space Administration* (NASA) pueden ser actualizados cada mes, cada día, cada tres horas o incluso en menores periodos de tiempo. Así, el *dataset* TRMM_3B42RT de la NASA (Véase capítulo 2.9.2), el cual incluye la precipitación estimada vía satélite en el globo terráqueo, se actualiza cada tres horas teniendo su origen desde el 01/03/2000. El *dataset* de Livneh et al. (2015) contiene registros diarios desde 01/1915 hasta 12/2011 y abarca el área de Estados Unidos, además de incluir las variables de precipitación, temperatura máxima, mínima y velocidad de viento.

Por parte de las estaciones climáticas se observa que, según Weather Underground (2016), el 1ro de Diciembre del 2014 se contabilizó en Estados Unidos un total de 12,246 estaciones climáticas públicas y del NWS. El sitio Web del SMN (2016) reporta un total de 598 estaciones meteorológicas en el territorio mexicano, las cuales se dividen en Estaciones Meteorológicas Automáticas (EMAs) y Estaciones Sinópticas Meteorológicas Automáticas (ESIMEs). Según el NWS (2016), éste recolecta aproximadamente 76 mil millones de observaciones al año. El *Citizen Weather Observer Program* (CWOP) cuenta con más de 7,000 estaciones en Estados Unidos que envían entre 50,000 y 75,000 observaciones cada hora. El número de estaciones que envían datos al CWOP a nivel mundial se ha visto en aumento en las últimas décadas debido a la mayor participación de la población civil (Chadwick, 2014).

La diversidad de estaciones, sensores remotos y agencias son algunas de las fuentes de variabilidad de datos climáticos, puesto que cada fuente genera y almacena datos con la estructura y el formato de su conveniencia. Para ejemplificar: las estaciones de covarianza Eddy, generan y archivan los datos en formato .ghg. El *dataset* Pathfinder SST (Sea Surface Temperature), ofrecido por la NOAA, se encuentra en formato NetCDF. El *dataset* del *Climate Prediction Center* (CPC) de precipitación diaria, proveído también por la NOAA, se encuentra

almacenado en archivos binarios. Datos de precipitación y temperatura proveídos por el IMD se encuentran en archivos de texto plano; y la lista continúa en tanto tecnologías y procedimientos continúen siendo desarrollados.

Estas características de los datos climáticos hacen que pronto su utilización se vuelva una tarea compleja. El investigador interesado se enfrenta a una gran cantidad de archivos con formatos distintos, que van desde los más sencillos como texto plano y CSV, hasta los más complejos archivos binarios georreferenciados como NetCDF, HDF y GRD. A este problema se le conoce como heterogeneidad de datos.

La heterogeneidad de los datos climáticos representa un problema para los investigadores que usan dichos datos. Dependiendo del estudio a realizar, se necesitará consultar una o más fuentes de datos distintas teniendo el investigador que trabajar, en la mayoría de los casos, con más de un formato de archivo diferente, los cuales tienen un conjunto de variables distintas y tamaños de archivos o bases de datos también distintos. Esto constituye una limitación, debido a que es requerido tener el conocimiento y las herramientas necesarias para manipular cada formato en el que se encuentren disponibles estos datos así como tener acceso a una infraestructura computacional con capacidades suficientes para manipular estos datos.

Pero el problema no sólo termina en los formatos distintos y el conocimiento necesario para utilizarlos, sino que también se le agregan otras dificultades como las resoluciones espaciales y temporales diferentes, estructura y variables distintas e incluso discrepancia en las unidades de dichas variables. Asimismo, generalmente un estudio de investigación involucra un subconjunto de datos determinado, y no su totalidad. Ejemplo: un estudio que requiera los datos de únicamente la cuenca del Río Colorado y éstos provengan de un *dataset* que cubra todo el territorio estadounidense. Esto crea la necesidad adicional de segmentar y seleccionar un subconjunto de interés de uno o varios *datasets* existentes. Los investigadores área de

climatología, hidrología y demás ciencias de la tierra tienen la necesidad de integrar datos provenientes de distintas fuentes heterogéneas, de tal forma que puedan obtener información valiosa de estos datos, requerida ya sea para completar su estudio, realizar un pronóstico, o alguna otra aplicación. Una restricción más para el investigador es la necesidad de agregar estos datos en un formato que le sea sencillo manejar, por ejemplo, CSV o formatos soportados por software tipo *Geographic Information System* (GIS).

Ahora se tiene un panorama más amplio de la problemática: La heterogeneidad de datos climáticos, causada en gran parte por sus características de volumen, variabilidad y velocidad de crecimiento; las limitantes que esto representa y la necesidad de integrar dichos datos para explotar la información que se genere. Este trabajo presenta la necesidad de desarrollar una solución tecnológica que mitigue las dificultades asociadas a esta problemática, tomando como un caso de estudio a las necesidades de datos de los investigadores del grupo *Hydroinformatics and Integrated Hydroclimate* (HIH) del *Department of Biological Systems Engineering* de la *University of Nebraska-Lincoln* (UNL).

1.2 Justificación

Después de describir la problemática asociada a la heterogeneidad de datos climáticos, así como las necesidades y limitantes a los que se enfrentan los investigadores interesados en dichos datos, en este trabajo se presenta una propuesta para mitigar dichas necesidades y limitantes.

Se propone construir una plataforma de software capaz de almacenar o importar datos provenientes de cualquier fuente y en cualquier formato/estructura a un almacén de datos de la plataforma, logrando así el acceso a los datos importados de manera uniforme, por medio de interfaces bien definidas.

Una vez importados los datos en este almacén, la selección y extracción de la información se podrá realizar con mayor facilidad, debido a que se eliminará la necesidad de adquirir conocimiento y herramientas específicas para manipular los conjuntos de datos en sus formatos originales.

En este trabajo, se considerará únicamente a un puñado de fuentes de datos (definir con exactitud cuántas y cuales) o *datasets* de datos en malla utilizados por el HIH de la UNL, a manera de delimitar el alcance de esta investigación. Sin embargo, una vez validado el funcionamiento de la plataforma, ésta podría ser utilizada para almacenar *datasets* adicionales a los considerados en este estudio.

Esta investigación beneficiará a los investigadores del grupo HIH de la UNL, así como potencialmente a más investigadores del área de climatología, que tengan necesidades similares a las que contempla la solución propuesta.

1.3 Objetivos

1.3.1 Objetivo General

Construir una plataforma de software que ofrezca servicios homologados de almacenamiento, segmentación, recuperación (diferencia con colección? Describirlos secuencialmente) (colección: obtener los datos de las fuentes originales, especificar si la arquitectura incluye este proceso o no.) y visualización de datos climáticos en malla provenientes de distintas fuentes y conjuntos con estructura y formatos heterogéneos.

1.3.2 Objetivos Específicos

1. Desarrollar un almacén de datos donde sea posible almacenar y recuperar cualquier conjunto de datos climáticos en malla utilizado por la UNL, sin importar su estructura o formato.

2. Estandarizar el acceso al almacén de datos mediante el desarrollo de un *Application Programming Interface* (API) para consultar segmentos específicos de la información almacenada.

1.4 Preguntas de investigación

En esta sección se presenta una lista de las principales interrogantes que se tienen con este trabajo de investigación:

1. ¿Cuáles proyectos o soluciones existen en el mercado diseñadas para integrar datos climáticos en *raster*/malla? ¿Qué enfoque utilizan? (Almacén de datos, *Data hub*, Federación, Otro) ¿Qué arquitectura de software utilizan? ¿Cuáles son sus características (Almacenamiento, visualización, exportación, disponibilidad de API)?

2. ¿Cuáles manejadores de bases de datos existen para almacenar conjuntos de datos en *raster* o pueden ser utilizados para ello? ¿Cuál es la opción más factible respecto al esfuerzo de implementación, tiempo de respuesta, tiempo de carga y espacio de almacenamiento?

3. ¿Cuál esquema es el más adecuado para almacenar conjuntos de datos climáticos *raster* en una base de datos? ¿Cuáles son sus características?

2 Fundamentos

2.1 Climatología y datos

Una de las disciplinas principales con las que este trabajo de investigación intersecta es el de la climatología o la “ciencia del clima”, siendo ésta el área de conocimiento de donde provienen los datos considerados para el enfoque de este trabajo. En los siguientes apartados se definen los conceptos básicos necesarios para comprender el trabajo realizado en esta investigación, en lo referente al área de climatología. De igual modo se describen las características de los datos climáticos, algunos conjuntos de datos, variables y demás conceptos aledaños.

2.2 El clima

En lo que concierne a este tema y a lo largo de este trabajo de investigación se utilizarán frecuentemente dos términos principales: el estado del tiempo y el clima. Ambos términos difieren en significado, por lo que en esta sección se conceptualizan citando las definiciones de otros autores en el área de investigación.

El estado del tiempo o tiempo atmosférico es la condición de la atmósfera en cualquier momento y lugar en particular. Éste se encuentra siempre en constante cambio y está comprendido por los elementos de temperatura y presión del aire, humedad, nubes, precipitación, visibilidad y viento (Ahrens, 2007).

El estado del tiempo es la condición instantánea general de la atmósfera en un cierto lugar y momento. Éste es descrito mediante medición directa de propiedades atmosféricas particulares como temperatura, precipitación, humedad, dirección del viento, velocidad del viento, cobertura de las nubes, y tipo de nubes. El término se refiere a aspectos tangibles de la atmósfera (Rohli

y Vega, 2008). Ruddiman (2008) define al tiempo atmosférico como “fluctuaciones en temperatura, precipitación, y vientos en escalas de tiempo menores a un año”. Estado del tiempo es el estado de la atmósfera con respecto al viento, temperatura, nubosidad, humedad, presión, etc., en un punto en el tiempo dado (Ej. La temperatura máxima de hoy) (NWS, 2018). El tiempo desde el punto de vista climático es la suma total de las propiedades físicas de la atmósfera, o sea de los elementos en un período cronológico corto; esto es el estado momentáneo de la atmósfera (Ruiz Corral et al., 2006).

Por otro lado, el clima representa la acumulación de eventos meteorológicos (o eventos del estado del tiempo) diarios y estacionales durante un largo periodo de tiempo para una región en particular (Ahrens, 2007). Hartmann (2015) define al clima como la síntesis del estado del tiempo en una región en particular. Puede ser definido cuantitativamente usando los valores esperados de los elementos meteorológicos en una locación durante un cierto mes o estación. A estos valores esperados se les puede llamar elementos climáticos e incluir variables como la temperatura promedio, precipitación, viento, presión, nubosidad y humedad. Rohli y Vega (2008) definen al clima como el estado general de la atmósfera a largo plazo en un lugar dado, tanto en la superficie como sobre de ella, incluyendo no sólo las condiciones promedio, sino también la variabilidad y estacionalidad de dichas condiciones y factores que las causan. Clima es la condición del estado del tiempo compuesto o generalmente predominante de una región durante el año, promediada a través de una serie de años (NWS, 2018). El clima es la agregación del conjunto de fenómenos meteorológicos que caracterizan el estado medio de la atmósfera en un punto de la superficie terrestre. El clima es entonces el estado más frecuente de la atmósfera en un lugar determinado, y comprende los extremos y todas las variaciones (Ruiz Corral et al., 2006).

Para efectos de este trabajo de investigación, se utilizará el término estado del tiempo o tiempo atmosférico para referirse a la condición o fotografía de la atmósfera en un momento y lugar

dado, junto con los elementos meteorológicos que describen dicha condición (precipitación, temperatura, humedad, velocidad del viento, etc.). Por ejemplo, “El día de hoy se reportó en la ciudad de Mexicali una temperatura de 4 °C, con una humedad relativa del 33% y viento de 2 km/h”. Cuando se utilice el término clima se hará referencia al estado del tiempo promedio de un lugar, el cual puede ser descrito mediante los valores esperados de los elementos climáticos, obtenidos por la síntesis de eventos meteorológicos suscitados en dicho lugar durante un largo periodo de tiempo. Por ejemplo, Ruiz Corral et al. (2006) indican que para definir el tiempo atmosférico se hace uso de los datos meteorológicos observados en las últimas 24-48 horas. Para definir el clima de un lugar, se recurre al análisis de los datos meteorológicos observados durante los últimos 30 años después de obtener una gran cantidad de observaciones diarias del estado del tiempo durante un largo tiempo. Por ejemplo, en Mexicali se ha podido describir el clima de esta ciudad, pudiendo indicar que durante el mes de Julio la temperatura alcanza un mínimo de 23.4°C y un máximo de 42.5°C, con una precipitación acumulada promedio de 4.5 mm (Ruiz Corral et al., 2006).

Para personas con poca o nada de experiencia en este campo de conocimiento, los términos de climatología y meteorología les podrían sonar similares e intercambiables e incluso como sinónimos, pero en realidad ambos se refieren a conceptos diferentes y cada uno constituye una ciencia distinta, estrechamente relacionada a la otra, por lo que conviene conceptualizar estos términos e indicar las principales diferencias entre ellos.

La meteorología es el estudio científico de la atmósfera de la Tierra y los fenómenos de temperatura, humedad, presión y movimiento del aire, los cuales producen varios climas y condiciones en el estado del tiempo. (Hdafa et al., 2016). Meteorología es el estudio de la atmósfera y sus fenómenos (Ahrens, 2007). Meteorología es el estudio del tiempo atmosférico, la condición instantánea general de la atmósfera en un cierto lugar y momento. La meteorología involucra el análisis de propiedades atmosféricas a corto plazo para un lugar y momento dado,

por lo que ésta sólo abarca el presente, pasado inmediato y futuro muy cercano (Rohli y Vega, 2008). Según el NWS (2018), meteorología es “la ciencia que trata con la atmósfera y sus fenómenos. Se puede hacer una distinción entre meteorología y climatología, la segunda se ocupa principalmente de condiciones meteorológicas promedio, no actuales.”

Cuando se habla de climatología, se está refiriendo al “estudio científico del clima, la rama de las ciencias atmosféricas que analiza el estado a largo plazo de la atmósfera, explica las causas de las condiciones promedio, y pronostica posibles cambios en dichas condiciones en escalas de tiempo estacionales y más allá, así como también evalúa los impactos potenciales de estos cambios en los sistemas naturales y sociales.” (Rohli y Vega, 2008). La climatología es la ciencia que trata con los fenómenos de los climas o las condiciones climáticas (NWS, 2018). Seoánez Calvo et al. (2001) definen a la climatología como “disciplina científica relativa al clima, cuyo objeto es la caracterización y clasificación de los distintos tipos de clima, su localización geográfica, el estudio de las causas de su diversidad y el análisis de las oscilaciones temporales en lugares dados”.

Como se puede inferir a partir de las definiciones anteriores, la característica principal que distingue a la meteorología de la climatología es la escala temporal comprendida en su estudio. Mientras que la meteorología involucra un lapso temporal del presente, pasado inmediato y futuro muy cercano; la climatología considera un lapso temporal mucho mayor, estudiando las variaciones en el clima a través del tiempo. Otra diferencia entre la meteorología y la climatología es que “la climatología permite estudiar los procesos atmosféricos y sus impactos mucho más allá del estado del tiempo del día de hoy” (Rohli y Vega, 2008).

Por último, es conveniente mencionar que la climatología se puede subdividir en diferentes subcampos de conocimiento, así como también diferentes escalas espaciales, dependiendo del área del fenómeno a estudiar. Las escalas espaciales en las que se divide la climatología son,

en orden de área comprendida, la microescala, escala local, mesoescala, escala sinóptica y por último la escala planetaria. La microescala comprende fenómenos menores a 0.5 km, la escala local fenómenos entre 0.5 a 5 km. La mesoescala abarca fenómenos entre 5 a 100 km. La escala sinóptica comprende fenómenos de entre 100 a 10,000 km. Por último, la escala planetaria abarca fenómenos de área entre 10,000 a 40,000 km (Rohli y Vega, 2008). Rohli y Vega (2008) también indican que la climatología se puede dividir en varios subcampos, algunos de los cuales se pueden asociar a una cierta escala espacial. Entre estos subcampos se pueden mencionar la climatología de la capa límite, climatología física, hidroclimatología, climatología dinámica, paleoclimatología, climatología agrícola y climatología aplicada.

2.3 Tendencia de datos abiertos

Hoy en día existen varias opciones para que un investigador obtenga datos climáticos. Están las agencias nacionales como la NOAA y CONAGUA, organizaciones internacionales como la *World Meteorological Organization* (WMO), sitios Web gubernamentales de "gobierno abierto (*open governmen*t)" como data.gov y datos.gob.mx, estaciones meteorológicas académicas e incluso los sensores de un teléfono inteligente.

Gran parte de las fuentes mencionadas ofrecen los datos climáticos de manera gratuita y para su libre uso. Esto contribuye al desarrollo de estudios relacionados con el medio ambiente y al avance en el entendimiento de los procesos atmosféricos de la Tierra, sus causas y repercusiones, además de promover un sinnúmero de posibles aplicaciones de estos datos en otras disciplinas que se pudieran beneficiar de ellos. Por esta razón, en este estudio se considera que los datos climáticos, por lo menos aquellos que se mantienen disponibles al público de manera libre, se pueden ubicar dentro de la tendencia de los datos abiertos (*open data* en inglés).

La Open Knowledge International (2018) proporciona el significado preciso de abierto en el término de datos abiertos: “Abierto significa que cualquiera puede acceder, utilizar, modificar y compartir libremente y por el propósito que sea (sujeto, como máximo, a los requerimientos que preserven su procedencia y su carácter de abierto)”. Por consiguiente, “los datos abiertos pueden ser utilizados, modificados, y compartidos libremente por cualquiera y para cualquier motivo.”

Autores como Rohli y Vega (2008) concuerdan con que mantener los datos climáticos abiertos es un buen paso para fomentar el avance en el entendimiento de todos los procesos climáticos, lo que repercutirá en conocimiento valioso para aquellas disciplinas y obras que tomen en cuenta al medio ambiente como factor importante. Por ejemplo, la construcción de edificaciones y viviendas eficientes en el uso de energía, previsión de desastres naturales, planeación urbana, etc. Estudios climatológicos recientes se han beneficiado de la extensa disponibilidad de los datos. La mayoría de los datos se encuentran ahora accesibles vía Internet, con varios conjuntos de datos proporcionados sin costo alguno. Esta facilidad de disponibilidad ha incrementado el número de estudios independientes sobre el clima, conduciendo a un elevado entendimiento de los procesos climáticos (Rohli y Vega, 2008).

Se ha mencionado en múltiples ocasiones el término datos climáticos y fuentes de datos climáticas, así como acrónimos sin haberlos definido formalmente. En los siguientes apartados se hace un espacio para ello. A continuación, se presenta la definición de “datos climáticos” utilizada en este trabajo.

2.4 Datos climáticos

Datos climáticos es un término muy general que puede aplicar a prácticamente cualquier dato existente que tenga relación con el área de climatología, como por ejemplo, datos de fuentes

proxis utilizadas en paleoclimatología. Es conveniente por ello, delimitar lo que engloba el término para efectos de este trabajo de investigación.

Para este trabajo, los datos climáticos se pueden conceptualizar como todos aquellos registros, observaciones, *datasets*, productos, resúmenes, imágenes, o cualquier otra pieza de información referente a la medición de variables atmosféricas, siendo éstos necesariamente almacenados en algún formato digital.

2.4.1 Características

Los datos climáticos se pueden caracterizar naturalmente por medio de tres propiedades principales: volumen, velocidad de crecimiento y variabilidad. Volumen debido a que el tamaño de muchos *datasets* o bases de datos climáticas alcanzan fácilmente el orden de los miles de millones de registros, sobre todo los productos en malla.

Velocidad de crecimiento ya que al ser necesario el estudio de las propiedades atmosféricas no sólo en diferentes puntos geográficos, sino también a través del tiempo, se suelen ir generando observaciones con una frecuencia fija, llamada formalmente “resolución temporal”. Estas frecuencias pueden variar drásticamente de un *dataset* a otro, desde mensualmente hasta diariamente o más allá, por lo que entre más corto el tiempo de actualización, mayor el volumen de datos generado. En Hdafa et al. (2016) se indica que debido a que la dinámica de la atmósfera, agua y condición de la Tierra siguen una continua evolución en el tiempo, los sistemas climáticos se encuentran recibiendo datos del ambiente todo el tiempo, creando un reto de velocidad.

Variabilidad o heterogeneidad principalmente a causa de la forma en que los datos climáticos son generados / recolectados / agregados, donde existen multitudes de fuentes de datos climáticas encargadas de dichas tareas, cada una con una manera de estructurar los datos no

necesariamente igual al de las demás. Las estaciones climáticas están afinadas para diferentes fuentes de datos: radares, satélites y otros sensores. La diversidad de éstas, en número y tipo, acarrea su heterogeneidad (Hdafa et al., 2016).

El torrente de datos que fluye en los sistemas climáticos continúa creciendo en volumen, heterogeneidad y velocidad. Esto es debido, por un lado, a los avances tecnológicos que enriquecen la esfera de información con grandes cantidades de datos (Hdafa et al., 2016). En la sección 2.9: Conjuntos de datos climáticos, se presentan ejemplos de *datasets* climáticos con información detallada que permite visualizar estas tres características con mayor facilidad.

Como menciona Hdafa et al. (2016), estas tres características de los datos climáticos son compartidas con las de los *datasets big data*, por lo que se puede considerar a la ciencia de la climatología como un campo de *big data*. Como se puede deducir, esto acarrea una serie de problemas al momento de trabajar con los datos; en los siguientes apartados se discutirán los más importantes.

2.4.2 Aplicaciones y usos frecuentes

Los datos climáticos son aplicados prácticamente en cualquier estudio que involucre el análisis del comportamiento de variables climatológicas, por lo que sería imposible enumerar todos sus posibles usos y aplicaciones. Acorde con la alta influencia del componente climático sobre el quehacer humano y el desarrollo y productividad de los seres vivos en general, las aplicaciones que pueden derivarse de la información climatológica son innumerables y variadas (Ruiz Corral et al., 2006). Una de las aplicaciones principales de los datos climáticos es el cálculo de normales climatológicas. Las normales son los valores estimados que variables como temperatura, precipitación y humedad tendrían a través de las temporadas del año, resultantes de un promedio de 30 años de observaciones. (Rohli y Vega, 2008) las describen como las condiciones del estado del tiempo promedio en un lugar, calculados por periodos de 30 años.

Otra de las aplicaciones principales de las observaciones climáticas son en la generación de pronósticos del tiempo para un futuro próximo; actividad que podría involucrar previamente la ejecución de modelos que permitan simular el comportamiento de la atmósfera. Estos modelos, llamados modelos de circulación general (GCMs, por sus siglas en inglés), involucran generalmente la predicción de la circulación atmosférica global basada en condiciones dinámicas. Éstos operan usualmente en la escala planetaria (Rohli y Vega, 2008). Un ejemplo de GCM es el modelo hidrológico *Variable Infiltration Capacity* (VIC; Liang, 1994).

Otra aplicación importante es en la agricultura, en donde se realizan diversas observaciones que explican la respuesta de los cultivos a la influencia del clima (Ruiz Corral et al., 2006). La bioclimatología es otro claro ejemplo. Se han realizado estudios sobre la fisiología y productividad animal asociadas al clima. Se sabe que los niveles de confort ambiental tanto en humanos como en animales dependen en gran medida de dos elementos del clima: la temperatura y la humedad relativa. A partir de ello se han generado formas de expresar esta relación, generalmente a través de los índices de confort como el THI (índice temperatura - humedad). Éste ha sido aplicado con éxito para explicar la producción de leche, la tasa de concepción reproductiva de las vacas, y la tasa de crecimiento en pollos (Ruiz Corral et al., 2006).

Los datos climáticos también se utilizan para crear diseños arquitectónicos e ingenieriles más eficientes, generar mejoras en medicina, y para entender el impacto de los paisajes urbanos en el ambiente natural y humano (Rohli y Vega, 2008). Aquí queda claro que el área de climatología es muy rica en cuanto a la utilidad de la aplicación de sus datos en incontables estudios. Este trabajo de investigación es uno más.

2.5 Fuentes de datos climáticas

Para el enfoque de esta investigación, se considerarán como fuentes de datos climáticas a todas aquellas organizaciones, agencias, satélites, estaciones o instrumentos de medición que provean o generan datos climáticos. Los autores Rohli y Vega (2008) clasifican a estas fuentes de datos en dos categorías: fuentes de datos primarias y secundarias. Las fuentes de datos primarias consisten en los instrumentos de medición de donde provienen originalmente los datos (Ej. estaciones climáticas, sensores remotos), mientras que las fuentes de datos secundarias son una agregación de los datos generados por las primeras para generar un *dataset* distinto. En el sentido estricto, estos autores hacen específicamente la distinción de datos primarios y datos secundarios, no exactamente una distinción entre las fuentes que generan dichos datos.

Los datos primarios son definidos como los datos recolectados por un investigador usando termómetros, radiómetros, sensores de humedad, u otro equipamiento en el campo. Los datos secundarios son aquellos usados por investigadores después de ser recolectados, verificados de su calidad, y compilados en otro lugar, usualmente en formato digital (Rohli y Vega, 2008). En el caso específico de los instrumentos meteorológicos para la medición de variables, Daley (1993) presenta tres clases en las que éstos se dividen:

Clase 1: Instrumentos que realizan mediciones *in situ* de puntos. Se asume que el instrumento ocupa mucho menos volumen que el fenómeno a muestrear. Esta clase incluye los instrumentos convencionales de la superficie (termómetro, barómetro, anemómetro e higrómetro).

Clase 2: Instrumentos que toman muestra de un área (en la superficie de la Tierra) o un volumen (de la atmósfera) remotamente. Éstos pueden estar ubicados en tierra, aeronaves, o en plataformas orbitales y son de tipo activo o pasivo.

Clase 3: Instrumentos que calculan velocidades del viento a partir de trayectorias lagrangianas.

Para efectos de esta investigación, se utilizará como base la distinción que Rohli y Vega (2008) presentan sobre los datos para clasificar las fuentes de datos climáticas en dos tipos:

1. Fuentes de datos primarias: Estas fuentes engloban básicamente todo instrumento de medición que genere observaciones de una o más variables, así como estaciones climáticas de la superficie, boyas, sensores remotos de satélites, etc.

2. Fuentes de datos secundarias: Son aquellas fuentes de datos que mantienen o generan *datasets* integrados basados en datos de las fuentes primarias, utilizados a manera de datos de reanálisis o compilación histórica. En esta categoría se encuentran todas las agencias climáticas.

A su vez se utilizará también la clasificación de Daley (1993) para distinguir a las fuentes de datos climáticas de tipo instrumento cuando se dé el caso.

2.5.1 Agencias de servicios climáticos

En este trabajo se consideran agencias de servicios climáticos a aquellas organizaciones, generalmente dependientes de un gobierno, cuya función principal es ofrecer servicios de pronósticos meteorológicos, alertas de desastres naturales, recolección, re-análisis y diseminación de datos climáticos, entre otros. Estas organizaciones pueden tener un alcance tanto nacional como internacional. Por ejemplo, la WMO.

Este estudio clasifica a las agencias climáticas como fuentes de datos secundarias, debido a que se dedican a alojar *datasets* y datos climáticos generados por otras fuentes (estaciones, satélites, etc.). Respecto a la disponibilidad de registros históricos de datos climatológicos, Estados Unidos es uno de los países mejor posicionados, comparado con otras regiones (Rohli

y Vega, 2008). La agencia principal encargada de proveer datos y servicios climatológicos y meteorológicos a la población es la NOAA, a través del NWS y sus demás subagencias.

En el caso de México, se cuenta con el SMN; la dependencia de la Comisión Nacional del Agua (CONAGUA) encargada de proporcionar información meteorológica a la población mexicana. Así como México y Estados Unidos, otros países también cuentan con su propia agencia climática para proveer datos y servicios a su población. Países como la India, con el IMD; y Canadá; con el *Meteorological Service of Canada* (MSC), son ejemplos de ello. A continuación se describen las agencias climáticas consultadas en este trabajo.

2.5.1.1 NWS. Servicio Meteorológico Nacional (Estados Unidos)

El NWS es la agencia federal que supervisa el pronóstico del tiempo auspiciado por el gobierno en los Estados Unidos, establecida en 1970 (Rohli y Vega, 2008). Éste es una subagencia de la NOAA, en donde la definen como una agencia del gobierno federal dentro de la NOAA del Departamento de Comercio, la cual es responsable de proveer observaciones, pronósticos y alertas de eventos meteorológicos e hidrológicos en el interés de la seguridad y economía de la nación (NWS, 2018).

2.5.1.2 SMN. Servicio Meteorológico Nacional (México)

De acuerdo a la información presente en su página Web, “El SMN es el organismo encargado de proporcionar información sobre el estado del tiempo a escala nacional y local en nuestro país. El SMN, depende de la CONAGUA, la cual forma parte de la Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT)." (SMN, 2018). Los objetivos del SMN se concentran en la vigilancia continua de la atmósfera para identificar los fenómenos meteorológicos que pueden afectar las distintas actividades económicas y sobre todo originar la pérdida de vidas humanas. El SMN también realiza el acopio de la información climatológica nacional (SMN, 2018).

2.5.1.3 NASA. Administración Nacional de Aeronáutica y del Espacio (Estados Unidos)

La NASA es una agencia independiente del gobierno federal que supervisa el programa de vuelo espacial de los Estados Unidos (Rohli y Vega, 2008).

2.5.1.4 NOAA. Administración Nacional Oceánica y Atmosférica (Estados Unidos)

La NOAA es la extensa agencia federal en los Estados Unidos que incluye al NWS, además de encargarse de gestionar la pesca de la nación y recolectar y diseminar datos ambientales. Su misión es proteger la vida y la propiedad, así como reconocer los aspectos integrativos del sistema tierra-océano-atmósfera (Rohli y Vega, 2008). La NOAA cuenta también con otras sub agencias encargadas cada una de tareas distintas, pero la más relevante para el enfoque de esta investigación es la principal encargada de la recolección de datos: El *National Environmental Satellite, Data, and Information Service* (NESDIS). NESDIS es la oficina contenida por la NOAA en los Estados Unidos que recolecta y disemina datos atmosféricos y otros datos geofísicos (Rohli y Vega, 2008). NESDIS contiene a su vez 11 oficinas responsables de recolectar datos geofísicos de todo tipo. Éstas incluyen al *National Climatic Data Center* (NCDC), el cual es una agencia federal de los Estados Unidos que archiva y distribuye datos atmosféricos (Rohli y Vega, 2008), tales como los obtenidos por el NWS, servicios militares, la Administración Federal de Aviación, la guardia costera y observadores voluntarios.

Además de las oficinas ya mencionadas, en Estados Unidos existen también los llamados Centros Climáticos Regionales, los cuales se especializan en el archivo de datos y monitoreo de eventos atmosféricos en sus regiones respectivas (Rohli y Vega, 2008). Existen seis de

estos centros distribuidos alrededor del país. Uno de ellos es el *High Plains Regional Climate Center*, el cual está ubicado en la UNL.

2.5.1.5 IMD. India Metereological Department (India)

El IMD es la agencia climática nacional de la India, perteneciente al Ministerio de Ciencias de la Tierra y análoga al NWS en EUA y al SMN en México. Es el servicio meteorológico nacional del país y la agencia gubernamental principal en todo lo relacionado a meteorología, sismología y temas similares. Ésta fue establecida en 1875 (IMD, 2015).

2.5.1.6 WMO. Organización Meteorológica Mundial (Naciones Unidas)

La WMO, es una agencia de las Naciones Unidas que distribuye una variedad de productos de datos climatológicos para su uso en estudios sinópticos y dinámicos a gran escala, aunado al fomento de la protección del ambiente atmosférico internacional y la protección de la gente de los desastres naturales (Rohli y Vega, 2008).

2.5.2 Estaciones climáticas

Otro ejemplo claro de fuentes de datos son las estaciones climáticas. Éstas son clasificadas en este estudio como fuentes de datos primarias, así como de clase 1 según Daley (1993). Cuando se habla de estaciones climáticas se está refiriendo a todos aquellos instrumentos, junto con la infraestructura necesaria para instalarlos, para la medición de variables atmosféricas ubicados en la superficie de la Tierra, el aire superior o en los mares y océanos. Estos instrumentos deben ser fijos ya que siempre están tomando mediciones del mismo punto geográfico. Instrumentos o sensores como los de los satélites (sensores remotos) no entran en esta categoría para este estudio. Las estaciones climáticas generan series de datos sobre un mismo punto geográfico, por lo que es necesario instalar más de una estación en un área de interés si se desea cubrir lo mejor posible las condiciones atmosféricas en dicha área. Para

después cubrir el área de estudio con mediciones uniformes por medio de una malla, es necesario recurrir a técnicas de interpolación.

De acuerdo con Rohli y Vega (2008), la mayoría de los registros meteorológicos del mundo están confinados a las naciones más desarrolladas. Países en vías de desarrollo, áreas rurales y especialmente los océanos están pobremente representados en la base de datos meteorológica global. Como resultado de la carencia de datos de alta calidad, la investigación actual involucra la aplicación de mediciones derivadas de radar y satélite para suplementar la red existente de estaciones.

Debido a la imposibilidad de cubrir todo el territorio de la Tierra adecuadamente con estaciones climáticas (por costos, topografía inadecuada, etc.), los sensores remotos satelitales se han convertido en una excelente opción para cubrir uniformemente grandes áreas de la Tierra por medio de estimación de las variables climáticas.

2.5.3 Sensores remotos

Los sensores remotos son considerados en este estudio como aquellos sensores montados principalmente en satélites, cuya función es la medición de variables climáticas de manera remota en la superficie de la Tierra. Esta medición se realiza estimando sus posibles valores en base a la radiación de onda larga emanada por la superficie.

La técnica que utilizan los sensores remotos para realizar sus mediciones se le llama teledetección, la cual se emplea para obtener información a distancia sobre objetos y zonas de la superficie de la Tierra. Está basada en la medida y en el análisis de la energía radiante procedente del objeto a estudiar (Seoánez Calvo et al., 2001).

Los sensores remotos pueden ser tanto activos como pasivos. Los sensores pasivos son aquellos que se limitan a medir o analizar la energía radiante de origen natural o artificial, pero

ajena al sensor (Ej. cámara fotográfica). Los sensores activos van coordinados con una fuente de la misma radiación que registran. Emiten una onda conocida, y miden y analizan las ondas devueltas por el objeto tras el impacto con él (Ej. radar, sonar) (Seoánez Calvo et al., 2001). Se clasifican como una fuente de datos primaria, así como de la clase 2 en términos de Daley (1993).

La ventaja de los sensores remotos contra las estaciones climáticas en la superficie, es que no tienen impedimentos por la topografía de la superficie, debido a que realizan un barrido de la Tierra cada cierto tiempo, para medir las variables meteorológicas de interés en gran parte del globo, pudiendo medir áreas inalcanzables por las otras. Sin embargo, un punto en contra de estos sensores es que en vez de medir el valor real de la variable por medio de instrumentos de medición directa en la superficie, se utilizan técnicas y cálculos para inferir el valor de la variable en base a la radiación saliente de la Tierra. Esta desventaja depende del punto de vista del investigador y del enfoque de su trabajo. Normalmente los datos obtenidos de los satélites se generan y distribuyen con una estructura de malla.

2.6 Variables climáticas

El clima de la Tierra se define en términos de elementos del estado del tiempo medibles. Los elementos de mayor interés son la temperatura y la precipitación. Estos dos factores en conjunto determinan en gran parte las especies de plantas y animales que sobreviven y prosperan en una locación en particular (Hartmann, 2015).

Las variables meteorológicas pueden clasificarse en: humedad, temperatura del aire, presión atmosférica, condiciones de las nubes y movimiento de los vientos. Otras clasificaciones propuestas, como el Sistema de Observación Climático Global (GCOS), ofrece una

categorización de 50 variables clave. En realidad, el número total de variables puede aumentar a 262 (Hdafa et al., 2016).

Existe otro tipo de dato relacionado con las variables climáticas: La ubicación o localización. Todas las observaciones en el área de climatología, sin importar la variable a la que pertenezcan, se encuentran contextualizadas en una ubicación. Dicho de otra manera, se encuentran georreferenciadas. Prescindir de esta característica inhibiría la utilidad del dato observado o medido, al no poder asociarlo con ningún lugar en especial. Hdafa et al. (2016) afirman que las variables meteorológicas obtienen sus valores de posiciones espacio-temporales específicas. Sin esta información, la medición de una variable de este tipo no tiene ningún valor en absoluto.

Las variables climáticas se encuentran georreferenciadas por su ubicación en las latitudes y longitudes de la Tierra. Las latitudes son un conjunto de líneas imaginarias que van de Este a Oeste y miden distancias al Norte o al Sur del Ecuador (Rohli y Vega, 2008). Las longitudes son un conjunto de líneas imaginarias que van de Norte a Sur y miden distancias al Este o al Oeste del meridiano cero (Rohli y Vega, 2008), también conocido como el meridiano de Greenwich. A continuación se describen brevemente las principales variables climatológicas que aparecen en este trabajo de investigación.

2.6.1 Precipitación

La precipitación es el agua de “entrada” a la superficie de la Tierra y representa el suministro atmosférico de humedad a ésta, variando ampliamente en los totales presentados de lugar a lugar (Rohli y Vega, 2008). Ésta es el producto de la condensación atmosférica, que puede ser sólida o líquida y a su vez es un elemento muy importante del clima, ya que determina las condiciones del medio ambiente tales como seco y húmedo (Ruiz Corral et al., 2006). El instrumento utilizado para medir la precipitación se llama pluviómetro, el cuál puede ser desde

el más sofisticado equipo con avanzada tecnología de medición, hasta el más simple contenedor cilíndrico operado de forma manual. Éste se encarga de medir la cantidad de lluvia en un día, mientras que la intensidad de la lluvia se registra en otro instrumento llamado pluviógrafo (Ruiz Corral et al., 2006). Las unidades comunes para medir la precipitación son los milímetros (mm). El equivalente volumétrico de un milímetro de precipitación es de un litro por metro cuadrado (Ruiz Corral et al., 2006). Según Rohli y Vega (2008), los Estados Unidos cuenta con 13,000 estaciones oficiales que reportan mediciones de precipitación en tierra. Sin embargo, aunque el territorio estadounidense sea uno de los mejores servidos en lo que a datos de precipitación se refiere, la realidad es que esta cantidad de estaciones no es suficiente para cubrir eficientemente todo el territorio. Debido a la falta de datos de precipitación de alta calidad ocasionada por la limitada cobertura por parte de los pluviómetros alrededor del mundo, la investigación actual involucra la aplicación de precipitación medida por satélites para suplementar la red existente de estaciones (Rohli y Vega, 2008).

2.6.2 Temperatura

Según Ruiz Corral et al. (2006), la temperatura es el elemento climático que refleja el estado energético del aire, el cual se traduce en un determinado nivel de calentamiento, lo cual indica el grado de calor o de frío sensible en la atmósfera. Junto con la precipitación, es uno de los elementos climáticos más importantes, ya que con base en la combinación de éstos se hace la clasificación de los diferentes climas existentes. Para medir la temperatura del aire se utilizan termómetros, mientras que para registrar sus variaciones se utilizan instrumentos llamados termógrafos. Las escalas de medición más utilizadas son la Kelvin, la Celsius y la Fahrenheit (Ruiz Corral et al., 2006).

2.6.3 Velocidad de viento

El viento es el aire en movimiento; la transferencia de masa atmosférica de una locación a otra (Rohli y Vega, 2008). Es el movimiento horizontal del aire después de pasar un punto dado. El viento comienza con las diferencias de presión de aire. La presión que es más alta en un lugar dado que otra establece una fuerza desde la alta hacia la baja presión (Seoánez Calvo et al., 2001). La medición de la velocidad del viento puede expresarse en m/s, en km/h, en nudos o según la escala de Beaufort. Los aparatos que se utilizan para medir la velocidad del viento son los anemómetros, y éstos pueden ser (Seoánez Calvo et al., 2001): de molinete y de hélice, térmicos y sónicos

2.6.4 Humedad relativa

La humedad relativa es una relación o proporción de la presión de vapor dividida entre la presión de vapor de saturación. Esta relación es utilizada por los climatólogos como una forma de medir o expresar la presencia de humedad en la atmósfera. La humedad atmosférica es el componente de masa más importante en la Capa Límite de la Superficie (SBL, por sus siglas en inglés) (Rohli y Vega, 2008). La presión de vapor es la contribución de la presión atmosférica total ejercida por las moléculas del vapor de agua mientras que la presión de vapor de saturación es la presión atmosférica ejercida por el vapor de agua cuando el aire se encuentra en saturación.

La humedad relativa expresa el porcentaje de vapor de agua presente en el aire antes de alcanzar la saturación. Cuando la humedad relativa se encuentra al 100% entonces ocurre la saturación y empieza la condensación o deposición del agua. El instrumento utilizado para medir la humedad relativa se llama higrómetro.

2.7 Resolución espacial y resolución temporal

Se han utilizado también a lo largo del documento distintos términos no definidos correctamente. En esta sección se hace una breve explicación de los términos “resolución espacial y temporal”.

Resolución es un atributo que representa la nitidez o precisión de observaciones sobre el espacio (resolución espacial) y el tiempo (resolución temporal) (Rohli y Vega, 2008). Este atributo se utiliza en los conjuntos de datos climáticos en *raster* o malla, donde su estructura consiste en una serie de variables cuyas observaciones son organizadas con una resolución espacial y temporal constantes.

La resolución espacial es la distancia entre observaciones de una variable en una locación dada (Rohli y Vega, 2008). En términos ilustrativos, imagine una malla aplanada superpuesta sobre la Tierra, donde cada cuadro de ésta representa una observación de una variable dada en esa área específica de la Tierra. Estos cuadros deben tener un área uniforme en toda la malla y sus centroides deben estar a la misma distancia uno del otro. Esta distancia es la resolución espacial. Por ejemplo, una resolución espacial de 0.25° quiere decir que cada observación de la malla está separada una de la otra por 0.25° grados de latitud o longitud. Cada cuadro de esta malla equivale a un área de aproximadamente 31.25 km^2. Entre mayor sea la resolución espacial, mayor será el detalle y la cantidad de observaciones representadas en el área de la Tierra. Por ejemplo, una resolución de 0.125° es mayor que la resolución del ejemplo anterior, ya que ésta es más fina o el área que comprende cada observación es más pequeña, pudiendo una malla de esta resolución representar más observaciones en la misma área que la otra malla. Una malla de esta resolución equivaldría a un área de 15.625 km^2 por cada observación.

Es importante mencionar que las mallas no sólo se representan espacialmente en un plano de dos dimensiones (latitud y longitud), sino que también existen mallas que utilizan la tercera dimensión para añadir altura o profundidad de la superficie.

La resolución temporal es la suma de tiempo entre observaciones de una variable en una locación dada (Rohli y Vega, 2008). Dicho de otra manera, es la respuesta a ¿Cada cuánto tiempo se registra una observación nueva para la variable en cuestión? Indica el lapso que existe entre una medición y la siguiente en el eje temporal. Ejemplos de resoluciones temporales: diaria, mensual, horaria, cada tres horas, cada media hora, anual.

2.8 Formatos de archivo

Este apartado se dedica exclusivamente a describir los formatos de archivo de los datos climáticos más importantes utilizados en este trabajo. Existen muchos otros más en el área, pero la descripción se limita a cuatro: NetCDF, HDF, archivos binarios y archivos de texto plano / CSV.

Los últimos dos formatos de archivo son más generales y de uso ordinario que los primeros. No son especializados para el área de climatología y son libres de utilizar la estructura de almacenamiento de los datos que se desee, siendo mucho más variados los esquemas utilizados en estos formatos que en los que cuentan con una estructura predefinida.

2.8.1 NetCDF. Network Common Data Form

Network Common Data Form (NetCDF) es un conjunto de interfaces para el acceso a datos orientados a arreglos y una colección libremente distribuida de librerías de acceso a datos para C, Fortran, C++, Java, y otros lenguajes. Las librerías NetCDF soportan un formato independiente de la plataforma para representar datos científicos. Tanto las interfaces, como

las librerías y el formato soportan la creación, acceso, y distribución de datos científicos (Unidata, 2018).

Éste es un formato binario auto-describible (contiene cabecera con metadatos y los datos), portable y escalable. Cuenta con una estructura predefinida para almacenar los datos orientados a arreglos. Consiste a groso modo en un árbol de grupos, donde cada grupo contiene variables, atributos, u otros grupos, y cada variable puede contener a su vez más atributos y los datos propios en forma de arreglo. Esta estructura da una cierta flexibilidad para definir esquemas de variables a almacenar. Este formato es muy utilizado para *datasets* en malla.

Para poder utilizar este formato es necesario utilizar las librerías de programación que lo soporten, ya sea oficiales o de terceros. Entre las librerías más importantes se encuentran netCDF4 y scipy.io.netcdf para el lenguaje de programación Python, librerías para los lenguajes C, Java, y otros más, así como también librerías “genéricas” o capaces de leer más de un formato de archivo como GDAL y el software de visualización GrADS. Existe una distinción significativa entre las versiones de NetCDF 3.x y las 4.x. Esto radica principalmente en el motor de almacenamiento del formato. Las versiones de NetCDF 4.x están basadas en HDF 5. Al día de hoy, la última versión de NetCDF es la 4.6.0

El formato NetCDF es un formato de datos usado ampliamente por la comunidad de las geociencias para manipular datos científicos orientados a arreglos. (Lian et al., 2017). Ejemplos de *datasets* que utilizan este formato son aquellos estructurados en malla que presenta la NASA provenientes de satélites, los *datasets* de reanálsisis del NCAR y *datasets* en malla de precipitación proveídos por la NOAA.

2.8.2 HDF. Hierarchical Data Format

El *Hierarchical Data Format* (HDF) es un formato de archivo similar al NetCDF en que es orientado a arreglos, sin embargo, éste tiene más flexibilidad y características para almacenar estos datos. A decir verdad este formato, en su edición HDF5, es la superclase del formato NetCDF 4.x; es el contenedor en el que se basa esta serie de versiones.

HDF5 consiste en un formato de archivo para almacenar datos HDF5, un modelo de datos para organizar lógicamente y acceder a éstos datos desde una aplicación, y el software (librerías, interfaces de lenguaje, y herramientas) para trabajar con este formato (The HDF Group, 2018).

Características de HDF5 (The HDF Group, 2017): HDF soporta *datasets* de n-dimensiones y cada elemento en el *dataset* puede ser un objeto complejo. HDF es portable y es un formato de archivo auto-describible. No hay límite en el número o tamaño de los objetos de datos en la colección, proveyendo gran flexibilidad para *big data.*

La estructura del formato HDF5 es casi la misma que la de NetCDF 4.x, debido a que el último se basa en el primero. La diferencia está en las terminologías y la capacidad de almacenar tipos de datos distintos en un mismo objeto/variable (contrario al NetCDF). En HDF5, los dos objetos primarios de su modelo de datos son los grupos y los *datasets*. Un *dataset* en HDF5 es el equivalente de variable en NetCDF4. Un archivo HDF5 (un objeto en sí mismo) puede ser visto como un contenedor (o grupo) que sostiene una variedad de objetos de datos heterogéneos (o *datasets*). Los *datasets* pueden ser imágenes, tablas, gráficas, e incluso documentos como PDF o Excel (The HDF Group, 2017). En la Figura 1 se muestra un ejemplo de la estructura de un archivo HDF5.

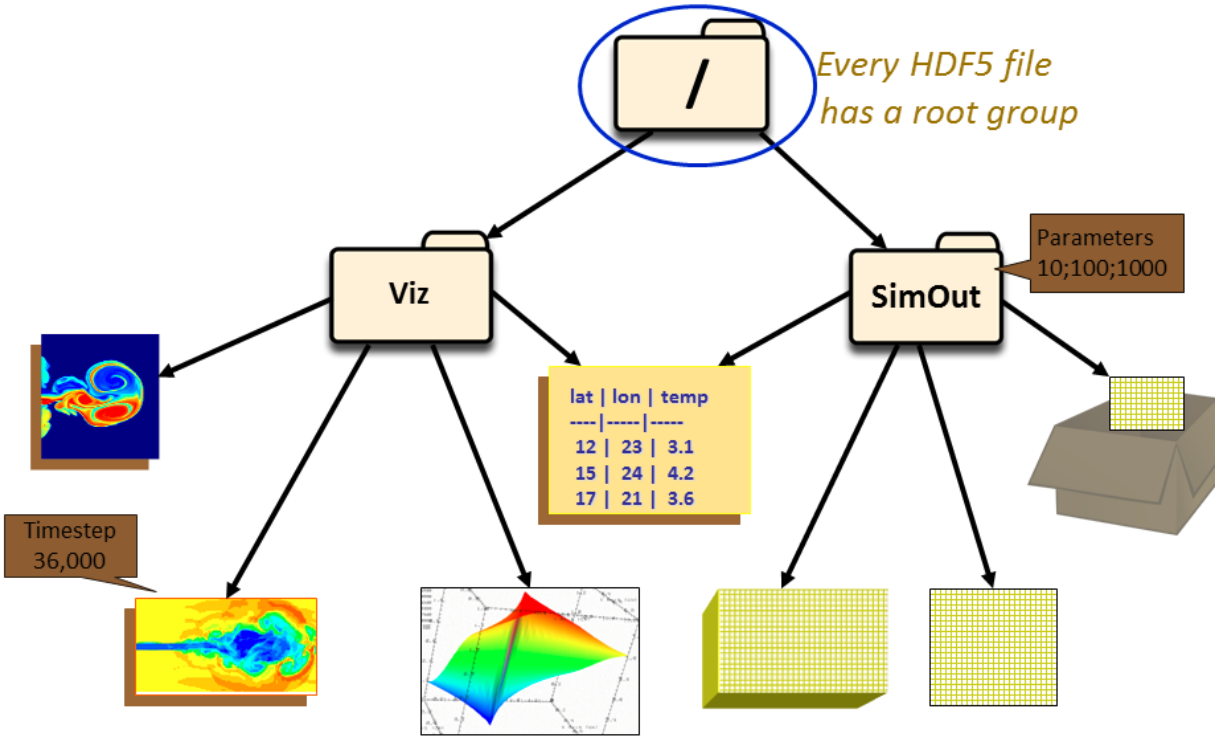


Figura 1. Ejemplo de estructura de un archivo HDF5 (Extraído de The HDF Group, 2018)

Al igual que NetCDF, HDF también cuenta con una serie de librerías y software para utilizarlo. Ejemplos de esto son los *wrappers* para C++, FORTRAN y Java que vienen incluidos con el software. También cuenta con herramientas de línea de comando para visualización gráfica y volcado de datos. La última versión de HDF5 actualmente es la 1.10. HDF ha sido utilizado exitosamente para almacenar datos de barridos de satélites proporcionados por la NASA. Un ejemplo de esto es el producto de datos *Soil Moisture Active Passive* (SMAP) (Kim et al., 2016).

2.8.3 Archivos binarios, de texto plano y CSV

No hay una definición específica para estos tipos de formatos de archivo. En esta sección, más que describir los formatos de texto y binarios, se busca indicar cómo se han llegado a utilizar para almacenar datos climáticos.

A diferencia de los formatos anteriores, los cuales están construidos para almacenar y consultar eficientemente datos climáticos o científicos, los archivos binarios y de texto plano o CSV son de uso general y no están optimizados para operaciones ES (Entrada - Salida) intensivas. Es necesario además, estructurar los archivos de tal forma que sea posible almacenar y recuperar los datos almacenados con el mayor grado de automatización posible.

En esta sección, cuando se hace referencia a archivos binarios, no se toma en cuenta toda la gama de posibilidades, dentro de las cuales caen los mismos formatos NetCDF y HDF, por ser binarios, sino que se hacer referencia a archivos binarios creados desde cero por una aplicación, para almacenar datos con algún modelo de datos sencillo (normalmente datos continuos y sin cabecera de metadatos).

A diferencia de los archivos binarios, los archivos de texto y CSV ocupan mucho más espacio de almacenamiento conforme crecen, por lo que no son los más adecuados para utilizarse en este tipo de casos. También, debido a la forma de acceso secuencial su procesamiento suele ser más tardado si se busca un conjunto de registros en especial. Ambos formatos son libres, prácticamente pueden almacenar cualquier estructura de datos que el formato les permita representar. En el caso del CSV, éste ya tiene predefinida una estructura básica: un registro en cada línea y cada campo separado por coma (de ahí su nombre). Sin embargo, así como tienen la flexibilidad de almacenar los datos como se desee, también tienen la mayor dificultad de tener que adaptar las técnicas de recuperación de datos a los archivos específicos con los que se está trabajando. Hecho que no ocurría con los dos formatos anteriores, al tener interfaces de entrada y salida y un modelo de datos bien definido.

Por lo tanto, en este trabajo se considera que, si se requieren desarrollar rutinas para automatizar la lectura de estos datos, es indispensable tener una sólida base de conocimientos de programación así como de expresiones regulares para extraer los datos que se necesiten de

archivos de texto o CSV. En el caso particular de CSV y archivos de texto con una estructura similar, se puede hacer uso de herramientas de hojas de cálculo, pero su desventaja es que no pueden trabajar con archivos muy grandes, ya que la memoria RAM se satura. En el caso de archivos binarios es más complicado, debido a que además de conocimientos de programación se requiere conocer de antemano la estructura que tienen los datos binarios y qué significa cada valor, ya que en muchos casos los archivos binarios no vienen con cabecera de metadatos para interpretar los datos almacenados. Al igual que existen infinidad de posibilidades para estructurar los datos a almacenar, también existen infinidad de casos de uso de estos archivos. Un ejemplo es la utilización de archivos de texto o CSV para la salida o entrada de modelos de simulación climática. Se puede entender después de leer este apartado que el conocimiento básico de programación es una competencia indispensable para trabajar con estos formatos de archivo si se desea automatizar las tareas. En el apartado siguiente se describen los principales *datasets* con los que se trabajó en este estudio.

2.9 Conjuntos de datos climáticos

Un conjunto de datos o *dataset* es definido por el sitio de “data.gov” (2018a) como una colección organizada de datos. La representación más básica de un *dataset* son los elementos de datos presentados en una forma tabular. Cada columna representa una variable en particular. Un *dataset* puede también presentar información en una variedad de formatos no tabulares, por ejemplo XML, NetCDF, HDF5, o archivos de imagen.

2.9.1 Malla / *raster*

Los *datasets* climáticos estructurados en malla son uno de los tipos de *datasets* más utilizados en el área científica. En este trabajo de investigación se hace distinción de sólo dos tipos de

estructura de *datasets* climáticos. Consisten en los *datasets* orientados a puntos geográficos y los orientados a mallas o *rasters*.

Se considera a los *datasets* orientados a puntos geográficos como aquellos *datasets* que almacenan registros a través de la dimensión temporal para uno más puntos geográficos distribuidos no necesariamente a la misma distancia uno del otro, sin ser absolutamente necesario tener un punto de medición cada cierta distancia de manera estricta (como una malla). Normalmente estos tipos de datos provienen de estaciones climáticas de la superficie. En cambio, los *datasets* orientados a malla son exactamente lo que se infiere: *datasets* climáticos estructurados en malla. Éstos sí deben contar con un determinado número de puntos y la distancia uno del otro debe ser la misma, los puntos de la malla son calculados en base a datos de estaciones reales mediante interpolación y otras técnicas estadísticas.

En su forma más simple, una malla consiste en una matriz de celdas (o pixeles) organizadas en filas y columnas donde cada celda contiene un valor representando información, por ejemplo temperatura. Los *rasters* son fotografías aéreas digitales, imágenes obtenidas por satélites, imágenes digitales o incluso mapas escaneados (ESRI Inc, 2016). En la Figura 2 se muestra un acercamiento a un área de un *raster* o malla. Se puede apreciar la estructura de pixeles del mismo tamaño. Cada color podría representar un valor diferente para una variable, por ejemplo.

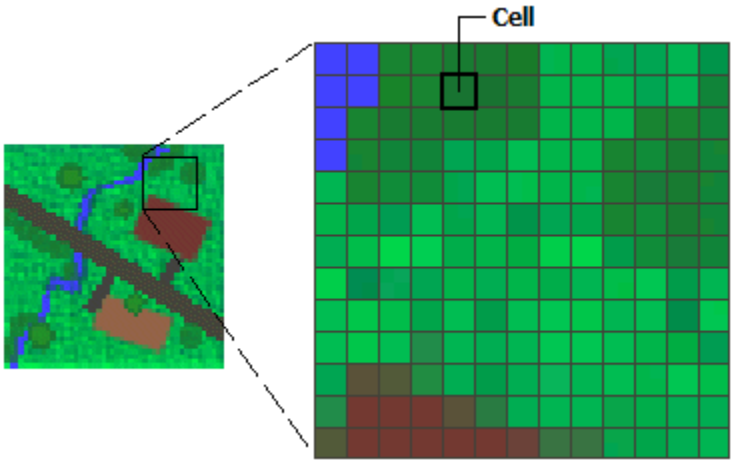


Figura 2. Acercamiento de un raster o malla. (Extraído de ESRI Inc, 2016)

2.9.2 TRMM_3B42RT

Este es el primer *dataset* con el que se trabajó durante el presente estudio de investigación. Contiene datos de precipitación estimados vía satélite en todo el globo terráqueo. Este *dataset* es proveído por la NASA y se encuentra disponible en NetCDF. TRMM_3B42RT (a partir de aquí sólo TRMM) es un *dataset* estructurado en malla. El nombre completo de este *dataset* es *3-Hour Realtime TRMM Multi-satellite Precipitation Analysis*. (Análisis de Precipitación Multi-Satélite en Tiempo Real cada 3-horas). TRMM es una abreviatura para *Tropical Rainfall Measuring Mission*, una misión en conjunto de la NASA y la agencia de exploración espacial japonesa, lanzada en 1997 para estudiar la lluvia para la investigación del clima y el estado del tiempo (NASA, 2018a).

Características: Resolución espacial: 0.25°. Resolución temporal: 3 horas. Cobertura espacial: Latitudes 60°N – S. Cobertura temporal: Marzo 2000 al presente. Formato de archivo: NetCDF. Fuente de datos: NASA. Variables: Precipitación.

Tamaño aproximado (hasta el momento): Tamaño por archivo: 1 MB. Periodo cubierto por archivo: 3 horas, 1 medición. Observaciones y variables por archivo: 480 lats x 1440 longs = 691,200 obs x 1 var. Total de archivos: 8 archivos (24 hrs) x 365 días x 18 años = 52,560 arch.

Total de observaciones: 52,560 arch x 691,200 obs = 36,329,472,000 obs x 1 var. Tamaño total: 51.33 GB.

En la Figura 3 se muestra un mapa de contorno de la precipitación (mm/h) de TRMM para el día 6 de Julio de 2005, a las 12 horas.

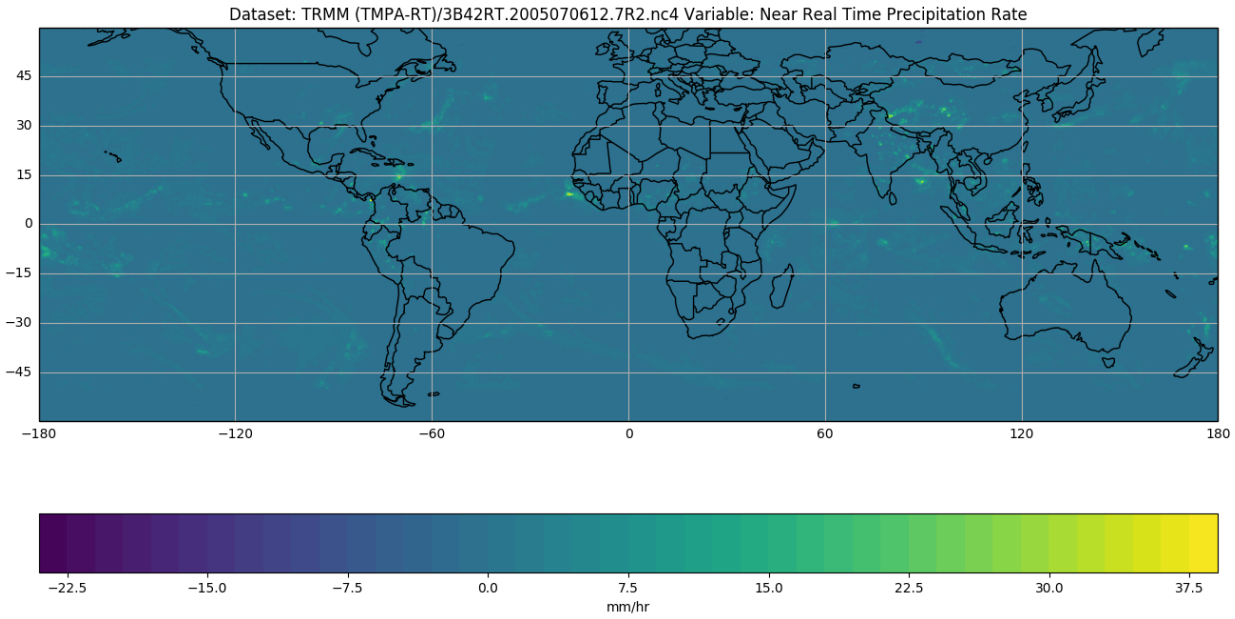


Figura 3. Visualización de precipitación de TRMM para el día 2005-07-06 a las 12 hrs.

2.9.3 Livneh

El *dataset* de Livneh (Livneh et al., 2015) es otro conjunto importante con el que se trabajó en este estudio. Es un *dataset* del tipo de reanálisis y organizado en malla que contiene variables de precipitación, temperatura máxima y mínima, y velocidad del viento. Proviene de otro *dataset* organizado en puntos geográficos en vez de malla, donde cada punto representa una estación del área del sur de Canadá, Estados Unidos y México.

Características: Resolución espacial: .0625°. Resolución temporal: diaria. Cobertura espacial: Sur de Canadá, Estados Unidos y México. Cobertura temporal: 1950 – 2013. Formato de

archivo: NetCDF. Fuente de datos: NOAA. Variables: Precipitación, temperatura máxima y mínima, velocidad de viento.

Tamaño aproximado: Tamaño por archivo: 282.6 MB. Periodo cubierto por archivo: mes, 31 - 28 mediciones. Observaciones y variables por archivo: 928 longs x 614 lats x 30 días = 17,093,760 obs. Total de archivos: 768 archs. Total de observaciones: 928 lons x 614 lats x 365 días x 64 años = 13,310,341,120 obs x 4 vars. Tamaño total: 282.6 MB x 768 = 211.95 GB.

En la Figura 4 se muestra un mapa de contorno de la precipitación (izquierda superior) (mm/h), temperatura máxima (derecha superior) (ºC), temperatura mínima (izquierda inferior) (ºC), y viento (derecha inferior) (m/s) de Livneh para el mes de Julio de 1969. Las mediciones diarias se promediaron.

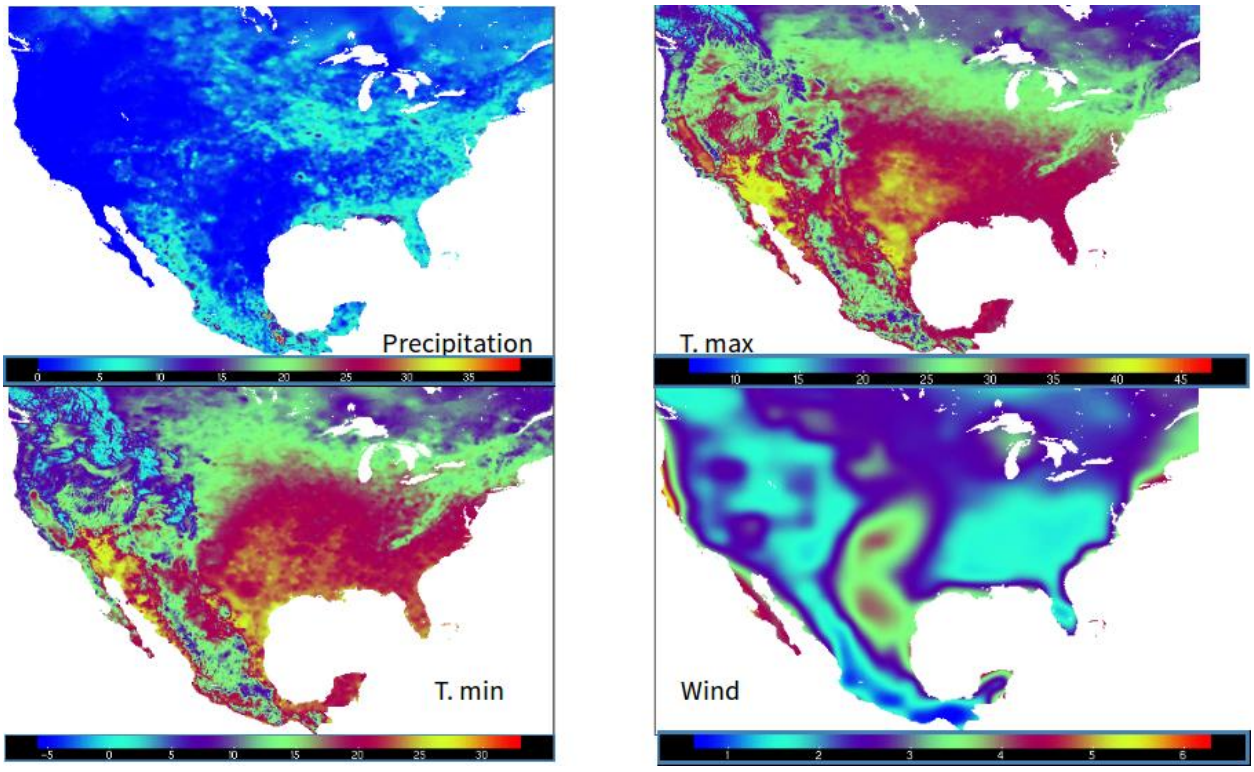


Figura 4. Visualización de las variables del dataset Livneh. *Promedio de Julio, 1969*

2.9.4 CPC

El *dataset* CPC de análisis unificado basado en pluviómetros de precipitación diaria sobre el área de CONUS (Estados Unidos Contiguo) (en adelante CPC), es un *dataset* estructurado en malla ofrecido por la NOAA y en formato NetCDF. Este *dataset* es parte de la suite de productos del Proyecto Unificado de Precipitación CPC que es puesto en marcha por el Centro de Predicción del Clima de la NOAA (CPC) (NOAA, 2018a).

Características: Resolución espacial: .25°. Resolución temporal: diaria. Cobertura espacial: 20.125 N - 49.875 N, 230.125 E - 304.875 E. Cobertura temporal: 1948 – 2006. Formato de archivo: NetCDF. Fuente de datos: NOAA. Variables: Precipitación.

Tamaño aproximado: Tamaño por archivo: 11.2 MB. Periodo cubierto por archivo: año, 365 mediciones. Observaciones y variables por archivo: 300 longs x 120 lats x 365 días = 13,140,000 obs. Total de archivos: 58 archs. Total de observaciones: 300 longs x 120 lats x 365 días x 58 años = 762,120,000 obs x 1 var. Tamaño total: 11.2 MB x 58 archs = 649.6 MB.

2.9.5 SWAT

El último *dataset* a describir es aquel al que se reconoce en este trabajo como “SWAT”. Este *dataset* no tiene el nombre de “SWAT” como tal, sino que es un nombre dado para identificar a los datos y archivos preparados para fungir como entrada para el modelo hidrológico *Soil and Water Assessment Tool* (SWAT). Estos datos comprenden el área de la India.

El modelo hidrológico SWAT es un modelo del dominio público desarrollado conjuntamente por el Servicio de Investigación Agrícola del USDA y el Texas A&M AgriLife Research. SWAT es un modelo de escala de pequeña cuenca a cuenca del río para simular la calidad y cantidad del agua de la superficie y el suelo y predecir el impacto ambiental del uso del suelo, prácticas de gestión de la tierra, y el cambio climático (SWAT, 2018).

Los datos se encuentran en formato de texto plano, y están organizados en forma de puntos geográficos, donde cada punto es representado por un archivo con una clave por nombre que apunta a la coordenada del punto. Cada archivo a su vez contiene todos los datos observados para ese punto en particular durante todo el periodo de interés. Cada renglón es un registro y los datos en cada registro están separados por coma. Las variables del *dataset* son precipitación, velocidad de viento y humedad relativa.

Características: Resolución espacial: .321°. Resolución temporal: diaria. Cobertura espacial: Latitud Sur: 12.996, Longitud Oeste: 73.29, Latitud Norte: 19.46, Longitud Este: 77.244. Cobertura temporal: 01/01/1979 a 31/07/2014. Formato de archivo: texto plano. Fuente de datos: NOAA. Variables: precipitación, velocidad de viento y humedad relativa.

Tamaño aproximado: Tamaño por archivo: 91.1 KB. Periodo cubierto por archivo: 01/01/1979 a 31/07/2014, 12,996 mediciones. Observaciones y variables por archivo: 12,996 obs. Total de archivos: 252 archs. Total de observaciones: 12,996 obs x 252 archs = 3,274,992 x 3 vars. Tamaño total: 22.5 MB.

3 Almacén de datos / Gestión de datos

3.1 Heterogeneidad de datos (Climatología)

El análisis de datos meteorológicos requiere un gran número de atributos de numerosas fuentes de datos distribuidas en espacio y tiempo. La integración de éstas puede ser una tarea que consuma mucho tiempo para los científicos (Lian et al., 2017). Diversos investigadores se han encontrado con problemas similares al tratar con datos climáticos, por ejemplo: Lian et al. (2017) indican que existe un gran número de *datasets* meteorológicos disponibles que pueden ser usados para el análisis de los eventos de tornados. Sin embargo, la mayoría de estos datos están distribuidos a través de múltiples sitios Web y no son accesibles en una locación central. Esto plantea un reto significativo para el científico que esté interesado en explorar estos eventos. Uno de los retos más importantes en el análisis de estos datos es que el dominio espacial del sistema climático es de naturaleza multi-escala, donde fenómenos meteorológicos locales son afectados por procesos climáticos globales a gran escala.

A pesar de la importancia de las observaciones meteorológicas de la superficie, su gestión se encuentra actualmente muy fragmentada. Se utilizan formatos de datos distintos, diferentes identificadores de estaciones, frecuentemente se discrepa en nombre e incluso coordenadas de la estación, y se tienen diferentes niveles de completitud de los datos (Thorne et al., 2017). Otro problema es que los registros existen archivados en una mezcla de escalas temporales promedio sub-diarias, diarias y mensuales.

Lian et al. (2017) afirman que el análisis que realizan en su investigación, requiere un número de productos de datos meteorológicos disparados y de diferentes fuentes. Estos datos

necesitan ser convertidos después a un formato adecuado para el análisis y la visualización. Una forma de solucionar la heterogeneidad de datos es mediante su integración.

3.1.1 Integración de datos (Climatología)

La integración de datos involucra combinar datos de diferentes fuentes y ofrecer una vista cohesiva de estos datos (Hdafa et al., 2016). La heterogeneidad de datos es un problema que ha existido por mucho tiempo en el campo de la meteorología y climatología, a pesar de los esfuerzos que se han realizado para mitigar este problema. Por ejemplo, el impulso por parte de la WMO para estandarizar los métodos de observación, formatos de datos y metadatos durante finales del siglo 19 y principios del 20 (Thorne et al., 2017).

Varios autores en el campo han coincidido en que la solución para este problema es por medio de la integración de datos. Thorne et al. (2017) indican que es esencial contar con registros meteorológicos del suelo completamente integrados para poder avanzar en el entendimiento del clima y del estado del tiempo. Para que sea posible investigar el alcance y las causas de los eventos climáticos de manera efectiva, es imperativo que los miles de millones de observaciones tomadas en los cientos de miles de locaciones, incluyendo métodos de observación emergentes, sean integradas y consolidadas en una base de datos coherente.

3.1.2 Almacén de datos

Un almacén de datos es definido como un repositorio o área de almacenamiento donde todos los datos de una organización son mantenidos en un sólo lugar. Esto incluye datos de diferentes fuentes así como datos actuales e históricos, los cuales pueden utilizar diferentes formatos (Spotless Data Ltd, 2018).

Un almacén de datos obtiene y almacena datos por medio de procesos de ETL. *Extract, Transform, Load* (ETL), es un proceso de almacén de datos en donde los datos son extraídos

de diferentes fuentes externas, son transformados a un formato destino, y son cargados a un almacén o repositorio de datos (Lian et al., 2017). Se le llama ETL al proceso de obtener datos de varios conjuntos de datos y otras fuentes para unirlos en un sólo conjunto de datos en el almacén de datos (Spotless Data Ltd, 2018).

3.2 Soluciones de software existentes

En esta sección, se mencionan algunas soluciones de software existentes en la literatura con cierto grado de similitud o relevancia con el presente estudio, en materia de integración de datos climáticos. En primer lugar se tiene a *FunnelCloud*, un sistema basado en la nube para explorar eventos de tornado (Lian et al., 2017). Este sistema implementa un flujo de trabajo para ayudar a científicos a analizar eventos de tornado y las variables climatológicas asociadas a ellos.

FunnelCloud es una solución que utiliza una arquitectura de *webservice* en la nube y distribuida. Utiliza flujos de trabajo ETL, los cuales están codificados en Python. Las funcionalidades generales de *FunnelCloud* son análisis de patrones espaciotemporales de eventos de tornado y condiciones ambientales asociadas. Visualización de variables relacionadas. Almacenamiento e integración de fuentes predeterminadas. Recuperación de variables y mallas completas por medio del nombre del *dataset* y un rango de fechas. Utiliza como fuentes los datos de eventos de tornados del SPC de la NOAA/NCDC, datos de reanálisis de norte américa de la NOAA y datos NEXRAD de NCEI. El esquema de datos que utiliza para el almacenamiento es una colección con documentos estáticos de las latitudes y longitudes, y otra colección con un documento por variable y día, con toda la malla en el documento en forma de arreglos. Utiliza la base de datos *NoSQL* MongoDB, junto con almacenamiento GridFS. La extracción de información o consultas comunes se realizan por nombre del *dataset* y rango de fechas.

La propuesta de solución de integración de Hdafa et al. (2016) contempla una arquitectura diseñada para procesar flujos de datos en tiempo real. Su fuente de datos es la entrada directa de distintos sensores remotos. Aplica conceptos de *Sensor Observation System* (SOS) y *Complex Event Processing* (CEP). La capa SOS es utilizada para acceder a las observaciones en forma estándar. Es un mediador entre la solución de integración y el sistema de sensores en tiempo real. CEP es utilizado para accionar eventos de procesado conforme se generen nuevas observaciones. El formato de salida utilizado es XML.

Otro ejemplo de solución de integración es *HydroCloud*, un sistema de integración de diferentes fuentes de datos hidrológicos y disposición de herramientas interactivas para el análisis exploratorio de datos y prueba de hipótesis estadística. (McGuire et al., 2016).

Finalmente, otra solución o estudio relacionado con la integración de datos climáticos es el de Thorne et al. (2017), el cual habla sobre una propuesta de banco de datos internacional para almacenar observaciones de la superficie (tierra y mar), llamada *Comprehensive Land-Based International Meteorological Observation Databank* (CLIMOD), para responder a las necesidades y problemas a los que la comunidad científica de climatología se ha enfrentado desde el comienzo.

La propuesta de solución de este estudio difiere de las soluciones o propuestas mencionadas con anterioridad. La propuesta de CLIMOD difiere de la presentada en este estudio en que su proyecto es a nivel internacional y sobre observaciones directamente a fuentes originales (primarias), mientras que éste se enfoca a una muestra de *datasets* o productos en malla, y no necesariamente observaciones puntuales o fuentes originales.

FunnelCloud está desarrollado pensando en los científicos que estudian eventos de tornado, sus causas y consecuencias. Dicha solución difiere en el enfoque del caso de estudio. Mientras que *FunnelCloud* se dirige al almacenamiento, recuperación y visualización de datos de

eventos de tornado, esta solución se enfoca en datos climáticos organizados en malla, pudiendo estos ser de cualquier variable. También la solución de *FunnelCloud* utiliza MongoDB para almacenar los datos, mientras que ésta está diseñada para utilizar cualquier base de datos para el almacenamiento.

A diferencia de la propuesta de Hdafa et al. (2016), la solución presentada en este estudio está diseñada para el almacenamiento de datos en un almacén de datos por medio de procesos ETL por lotes, y no en tiempo real ni basándose en eventos. Tampoco se limita a sólo observaciones de sensores remotos como lo hace ese estudio, sino que almacena datos climáticos provenientes de cualquier fuente, siempre que estén estructurados en malla.

3.3 Clientes de software en el área

En esta sección se relata de manera muy breve ejemplos de clientes de software en el área de climatología. Estos pueden ser tanto clientes encontrados en Internet como en estudios de la literatura. Sin embargo, antes de empezar a mencionar ejemplos de clientes, se debe de describir exactamente qué es un cliente.

Para efectos de esta investigación, un cliente de software representa toda aquella aplicación, ya sea Web, de escritorio o móvil, que implemente funcionalidades orientadas a usuarios finales y permita generalmente la interacción del usuario por medio de una interfaz gráfica. Al tratarse de clientes de software en el área de climatología, se espera encontrar funcionalidades como visualización o descarga de datos, selección, segmentación, generación de índices y algún otro tipo de análisis.

En primer lugar se encuentra el *National Centers for Environmental Information* (NCEI) de la NOAA (Antes NCDC). Este es un cliente/fuente de datos muy completo. En su sitio Web provee datos históricos de todas las estaciones, satélites y demás fuentes de datos en Estados

Unidos. El NCEI provee acceso a uno de los archivos de registro de observaciones del sistema tierra más amplios, con datos exhaustivos oceánicos, atmosféricos, y geofísicos. Desde las profundidades del océano hasta la superficie del sol y desde registros de hielo de millones de años a imágenes de satélite casi en tiempo real, NCEI es la autoridad líder en Estados Unidos para información ambiental (NCEI, 2018). NCEI Incluye un cliente Web llamado *Climate Data Online* (CDO), el cual provee libre acceso al archivo global histórico de datos meteorológicos y climáticos del NCDC, además de información histórica de estaciones. Incluye mediciones diarias, mensuales, de temporadas, y anuales de temperatura, precipitación, viento, datos de radar y normales climáticas de 30 años (NCDC, 2018b).

El cliente CDO ofrece tres herramientas para acceder a los datos, la primera es la herramienta de búsqueda, la cual permite filtrar por una serie de opciones tales como código postal, condado, ciudad, estado o país. La segunda es la herramienta de mapeo, la cual permite visualizar los datos por los mismos filtros de búsqueda que la herramienta anterior. La tercera es una colección de herramientas especializadas como búsqueda de datos por estaciones o locaciones, visualización de normales (1981-2010), acceso a registros meteorológicos diarios, datos marinos, entre otros. Los *datasets* disponibles para acceder en el cliente CDO son, por mencionar algunos: Resúmenes diarios del GHCN, datos marinos globales, resumen global del mes, resumen global del año, normales anuales/estacionales, precipitación horaria y cada 15 minutos. Además del cliente y herramientas mencionadas, el NCEI también cuenta con un API (NCDC, 2018a) basado en *webservices* para el acceso a sus *datasets*.

Algunas de las funcionalidades de este API son: Selección de datos por medio de id del *dataset*, id o coordenada de la estación. Búsqueda en *dataset* por categoría o tipo de dato (variable), por locación o estaciones. Búsqueda de datos por código postal, estados (EU) o ciudades. Filtro de datos por fecha de inicio y fin. Resultados en JSON.

Las limitantes de este API son: un límite de 1,000 registros que se pueden obtener en una petición y un periodo máximo de diez años en caso de requerir registros mensuales o anuales, un año para cualquier otro caso. Tampoco se puede seleccionar un área especificando sus coordenadas.

Además del API mencionado anteriormente (API v2), el NCEI cuenta también con el API de Servicio de Búsqueda, el cual exporta los resultados también en JSON; y el API de Servicio de Datos, el cual exporta los resultados en CSV, JSON y PDF (NCDC, 2018c). Por parte de la NOAA también existe un sitio Web con una serie de clientes o herramientas de software para acceder y visualizar los datos climáticos de dicha fuente. El sitio es Climate.gov. Este contiene una sección que presenta mapas y *datasets* reutilizables, cuyo objetivo es servir a oficiales y profesionales que necesitan datos climáticos para informar sus decisiones o compilar un reporte de adaptación al clima (NOAA, 2018b).

Climate.gov cuenta, por ejemplo, con un cliente Web llamado *Data Snapshots* (Instantáneas de datos), el cual muestra de manera visual, intuitiva y fácil de entender, mapas climáticos en una interfaz. Cada "instantánea" es una versión amigable al público de un producto de datos existente (NOAA, 2018b).

Este cliente cuenta con mapas de precipitación, temperatura, proyecciones (clima en el futuro), sequías, climas severos, y variables del océano. Cada mapa tiene una breve descripción de lo que se está presentando así como controles para elegir el mes y el año deseado para visualizar, con opción a descargar como imagen el mapa generado. La mayoría de los mapas disponibles ilustran el área de CONUS, excepto unos mapas de temperatura mundial y los mapas de océanos. El rango permitido para la selección de fechas, así como la granularidad con que se pueden elegir estas fechas, depende del producto de datos elegido. Entre los productos disponibles para visualización se encuentran: promedios mensuales de 30 años para

precipitación, precipitación total mensual, temperatura promedio mensual, monitor de sequías, temperatura máxima promedio con altas emisiones (proyección), entre otros. El sitio Web también cuenta con una "galería de *datasets*" que redirigen a descarga de datos o visualización de mapas de distinta índole, variando desde tiempo atmosférico pasado por código postal, mapa semanal de sequías, indicadores de oscilaciones del Sur - El Niño, visualizador de subida del nivel del mar, índices de climas extremos en Estados Unidos, entre otros.

Otro ejemplo de cliente Web es OSCAR/Surface (*Observing Systems Capability Analysis and Review Tool*), el repositorio oficial de metadatos de observaciones meteorológicas y climatológicas basadas en la superficie que son requeridos para el intercambio internacional. Los sistemas de observación están integrados bajo el *framework* de *WMO Integrated Observing System* (WIGOS). OSCAR/Surface es uno de los componentes del *WIGOS Information Resources* (WIR) (WMO, 2018). Este cliente Web presenta un mapa mundial de plataformas de observación del aire, superficie del suelo u océano, subsuelo, y de lagos o ríos. Es posible acceder a los metadatos de las estaciones por medio del nombre o el id WIGOS correspondiente. También es posible buscar estaciones por país o tipo.

Por último, Lian et al. (2017) presentan en su estudio un cliente Web desarrollado para asistir en el flujo de trabajo del sistema *FunnelCloud*. Este cliente consiste en varios componentes, incluyendo un mapa interactivo, una línea del tiempo dinámica, una tabla de estadísticas espaciales, y visualización de datos de radar basada en mapas. Utiliza tecnologías como d3.js y Leaflet API. El cliente Web de *FunnelCloud* puede ser utilizado para analizar eventos de tornado provenientes de un selecto grupo de fuentes climáticas. Una vez seleccionado un rango de fechas se selecciona el evento o grupo de eventos de tornado de interés. Con este cliente es posible obtener la información del tornado, visualizar variables u obtener estadísticas descriptivas. Como infieren Lian et al. (2017), construir un sistema de datos geográficos integrado es útil para que los científicos exploren y analicen patrones climáticos eficientemente.

4 Metodología

En este capítulo, se describen de manera breve las actividades en este trabajo de investigación para lograr los objetivos presentados en el capítulo 1.

El primer paso fue realizar una búsqueda de la bibliografía necesaria para sustentar esta investigación, teniendo como resultado el marco teórico de ésta. De manera similar, se definió el estado del arte del tema, la frontera, o dicho de otra forma, la revisión de los trabajos más recientes que se han realizado respecto al problema de la heterogeneidad de los datos climáticos.

Después se continuó con una investigación y aprendizaje de distintas bases de datos relacionales y *NoSQL* disponibles para uso libre, entre las que destacaban PostgreSQL, MongoDB y Cassandra. También, se obtuvo conocimiento de múltiples lenguajes de script especializados en el procesamiento de texto y con soporte de expresiones regulares, así como lenguajes de descripción de datos (o formatos de intercambio de datos), y ambientes de software como R y Matlab.

El paso 3 fue explorar los principales sitios Web o fuentes de datos climáticas utilizadas por los científicos del grupo HIH así como el adiestramiento en los diferentes formatos de archivo utilizados, tales como NetCDF y HDF, al mismo tiempo que se obtenía un entendimiento del procesamiento y casos de uso comunes de los datos climáticos del grupo.

Se definieron los requerimientos de la plataforma de software a construir con el experto del área en la UNL y se analizó la estructura de los conjuntos de datos a almacenar. Después, se continuó con la investigación de diversos tópicos para desarrollar la plataforma de software: módulos científicos de Python, herramientas ofrecidas por el *Holland Computing Center* (HCC)

y administración de trabajos, herramientas espaciales GDAL, PostGIS y GrADS, patrones de diseño arquitectónicos y esquemas de almacenamiento.

Basándose en la investigación anterior, se diseñó y construyó un prototipo de arquitectura basada en *plugins*, para el almacenamiento de datos climáticos provenientes de múltiples fuentes. Se construyeron también *plugins* múltiples para probar la funcionalidad de almacenamiento de la arquitectura, alternando entre diferentes *datasets*, lenguajes de programación y bases de datos utilizadas.

Esta arquitectura se construyó mediante desarrollo rápido basado en prototipos. Se construyeron tres grandes prototipos, cada uno renovando la base del código del anterior y expandiendo su funcionalidad.

Una vez desarrollada la arquitectura y verificado el correcto almacenamiento de datos climáticos en ella, se continuó con la implementación de un API para el acceso a dichos datos. Luego de desarrollar la arquitectura de la plataforma y los *plugins* de almacenamiento pertinentes, se documentó todo el código resultante, con el objeto de permitir a otros desarrolladores continuar expandiendo la arquitectura. Asimismo, se elaboró un diagrama del macro proceso general involucrado con la plataforma de software. Éste incluye los procesos de importación y exportación de datos. La arquitectura fue desplegada en un servidor Power ubicado en la UABC para su continua disponibilidad.

Por otra parte, y paralelo al desarrollo de la arquitectura, se construyeron también herramientas para extraer regiones de *datasets*, graficar variables, cambiar resolución de mallas, interpolar puntos, entre otros procesamientos de datos aplicados por el grupo HIH de la UNL.

En este estudio, se obtuvieron conocimientos principalmente de herramientas y formatos de archivo computacionales desarrollados en el área de climatología, lenguajes de programación, de script y módulos útiles para el procesamiento de texto y los formatos antes mencionados. En

el siguiente capítulo, se describen con mayor detalle los resultados obtenidos, las actividades realizadas y los problemas u obstáculos enfrentados en la realización de éstas.

5 Dataplugin - Arquitectura de software

La plataforma de software desarrollada para gestionar datos climáticos en este trabajo de investigación fue nombrada “Dataplugin”, haciendo alusión a la arquitectura basada en *plugins* con la que fue construido. Esta plataforma fue implementada mediante desarrollo rápido basado en prototipos, resultando en un total de tres prototipos, optando por el último desarrollado como la versión final. La arquitectura final de Dataplugin consiste en un componente de almacén de datos, un componente de *plugins*, y un componente de clientes de software o capa de visualización. A continuación, se presenta en Figura 5 la vista general de la arquitectura.

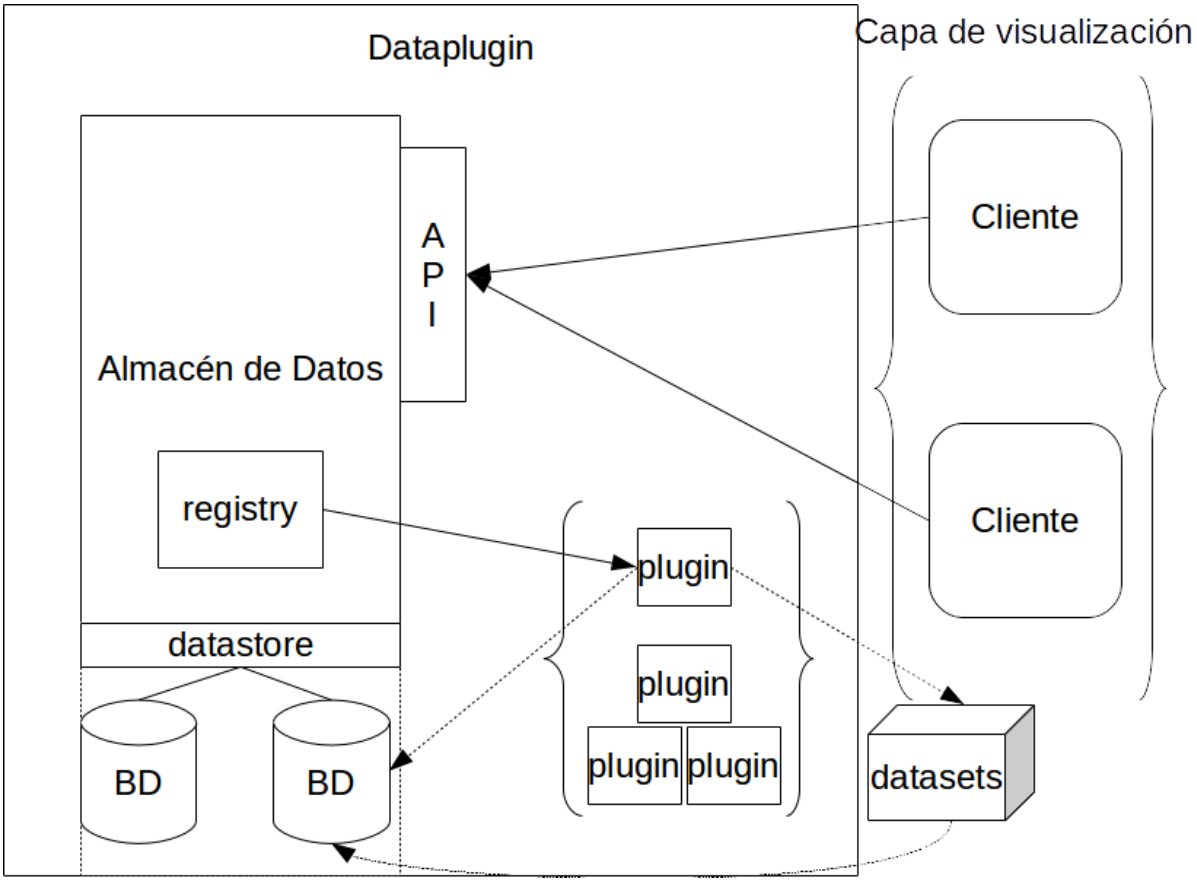


Figura 5. Vista general de la arquitectura Dataplugin

El componente principal de la arquitectura es el Almacén de Datos. Éste se encarga de gestionar y registrar los *plugins* de la plataforma, mantener el catálogo de los *datasets* almacenados, y proveer una interface para la extracción de los datos. Este componente está integrado por diferentes subcomponentes que en conjunto tienen la función de almacenar, gestionar y proveer datos climáticos. Los componentes más importantes son el *registry* (registro), las bases de datos, la capa de *datastore* para acceder a éstas, y el API de acceso externo a los datos almacenados.

Conceptualmente, la arquitectura de Dataplugin agrupa al Almacén de Datos y a los *plugins* como una sola entidad, mientras que los clientes de software y los *datasets* climáticos se mantienen como objetos externos a la arquitectura. Sin embargo, en la práctica el Almacén de Datos representa una entidad separada de los *plugins*.

Como se mencionó anteriormente, debido a la naturaleza de heterogeneidad de los *datasets* climáticos, tanto en estructura como formato, fue necesario desarrollar una solución de software capaz de lidiar con esta heterogeneidad. Se buscó diseñar una arquitectura capaz de admitir estos diferentes tipos de *datasets* sin necesidad de modificar el código fuente de ésta cada vez que se agregara un *dataset* con diferentes características.

Otra característica deseada de la arquitectura, que complementa a la anterior, es la capacidad de ejecutar código agregado recientemente de manera automática, sin necesidad de detener la ejecución de la arquitectura. Esto previendo que el Almacén de Datos fuera utilizado como una aplicación de servidor con alta disponibilidad.

Debido a estas características, principalmente, se eligió la implementación de la arquitectura del almacén de datos basada en *plugins*, donde cada uno de ellos fuera utilizado para almacenar un conjunto de datos en específico, o varios, si su estructura lo permitiera. El funcionamiento general y organización de la arquitectura es el siguiente:

Registry es el subcomponente del Almacén de Datos encargado de administrar los *plugins* existentes. Éste realiza operaciones como registrar *plugins*, borrarlos del registro, cargarlos a la arquitectura, localizarlos, proveer metadatos de los *plugins* cargados, y la operación más importante que es ejecutarlos.

Los *plugins* son la parte modular y dinámica de la arquitectura, éstos son piezas de software que empaquetan la funcionalidad necesaria para almacenar o importar uno o varios *datasets* al Almacén de Datos. Son el fragmento fundamental que permite a la arquitectura almacenar datos, sin embargo, los *plugins* no pueden funcionar por sí solos, deben ser cargados a la arquitectura para que ésta los pueda ejecutar.

Para que un *plugin* pueda ser utilizado efectivamente por el *Registry*, éste debe cumplir con una cierta estructura en lo que respecta a la organización del código. Anexo a la documentación de la arquitectura y de los *plugins* se detallan las dependencias que son necesarias utilizar, así como las interfaces de programación a implementar y otros requerimientos para desarrollar un *plugin* para Dataplugin. La arquitectura de Dataplugin, tanto el Almacén de Datos como las interfaces del *plugin*, se encuentran codificadas en el lenguaje de programación Java.

Cada *plugin* ejecuta sólo una acción de importación o almacenamiento de datos de uno o varios *datasets* climáticos. Esta acción se realiza sólo cuando el *Registry* llama a un *plugin* en especial para realizar la tarea de almacenamiento y, dependiendo del *plugin*, ésta puede ser repetida frecuentemente. (Ej. Cargar datos de un *dataset* estático contra uno que se actualiza regularmente).

El medio de almacenamiento físico o “real” del Almacén de Datos son manejadores de bases de datos (DBMS, por sus siglas en inglés) relacionales o no relacionales. Estos almacenan los datos crudos así como posibles metadatos para su recuperación posterior. La arquitectura está

diseñada para soportar distintos DBMSs, no obstante, en este trabajo se utilizan principalmente PostgreSQL y MongoDB. Dependiendo del *dataset* y el esquema elegido es la relación que tendrá con el medio de almacenamiento; en los casos implementados se ha almacenado un *dataset* por tabla de base de datos y un *dataset* por base de datos.

En este trabajo de investigación, al estar enfocado en *datasets* climáticos en malla, se utilizó el DBMS PostgreSQL para el almacenamiento de los datos "crudos" y MongoDB para el almacenamiento de los metadatos asociados. Esta aproximación se decidió basándose en una revisión de potenciales DBMSs, y durante el desarrollo de un *plugin* para el *dataset* TRMM. Para más información revisar los capítulos 7 y 8.1, respectivamente.

El Almacén de Datos accede a las bases de datos mediante la capa de programación nombrada *"datastore"*. Finalmente, el API de acceso externo a los datos es la puerta de entrada y salida del almacén. Es la interfaz que expone los métodos disponibles para consultar metadatos de los *datasets* almacenados, metadatos de los *plugins* disponibles, así como métodos para ejecutar un *plugin*, y obtener datos de los *datasets* almacenados. El API de acceso está implementado mediante servicios *REST* y estos pueden ser utilizados por aplicaciones que los soporten (clientes de software). Dicho API se desarrolló orientándolo a consultas de datos y metadatos de mallas, debido al enfoque del trabajo.

Los clientes de software son aplicaciones desarrolladas para hacer uso de los datos almacenados en Dataplugin. Estas aplicaciones son implementadas para generar información útil para usuarios finales a partir de los datos climáticos. Ejemplos de funcionalidades que se pueden implementar en un cliente de software son: visualización de variables climáticas en el espacio y el tiempo, generación de índices climáticos, consulta de observaciones agrupadas por cuencas hidrológicas, entre muchas otras funcionalidades.

A diferencia del Almacén de Datos, el cual fue desarrollado para no modificarse, los *plugins* y los clientes pueden ser desarrollados y agregados a la arquitectura por terceros, siguiendo los lineamientos definidos en esta investigación, para que funcionen correctamente en su integración con el almacén. En la Figura 6 se describe de manera más detallada los subcomponentes del Almacén de Datos, así como las interacciones principales entre ellos.

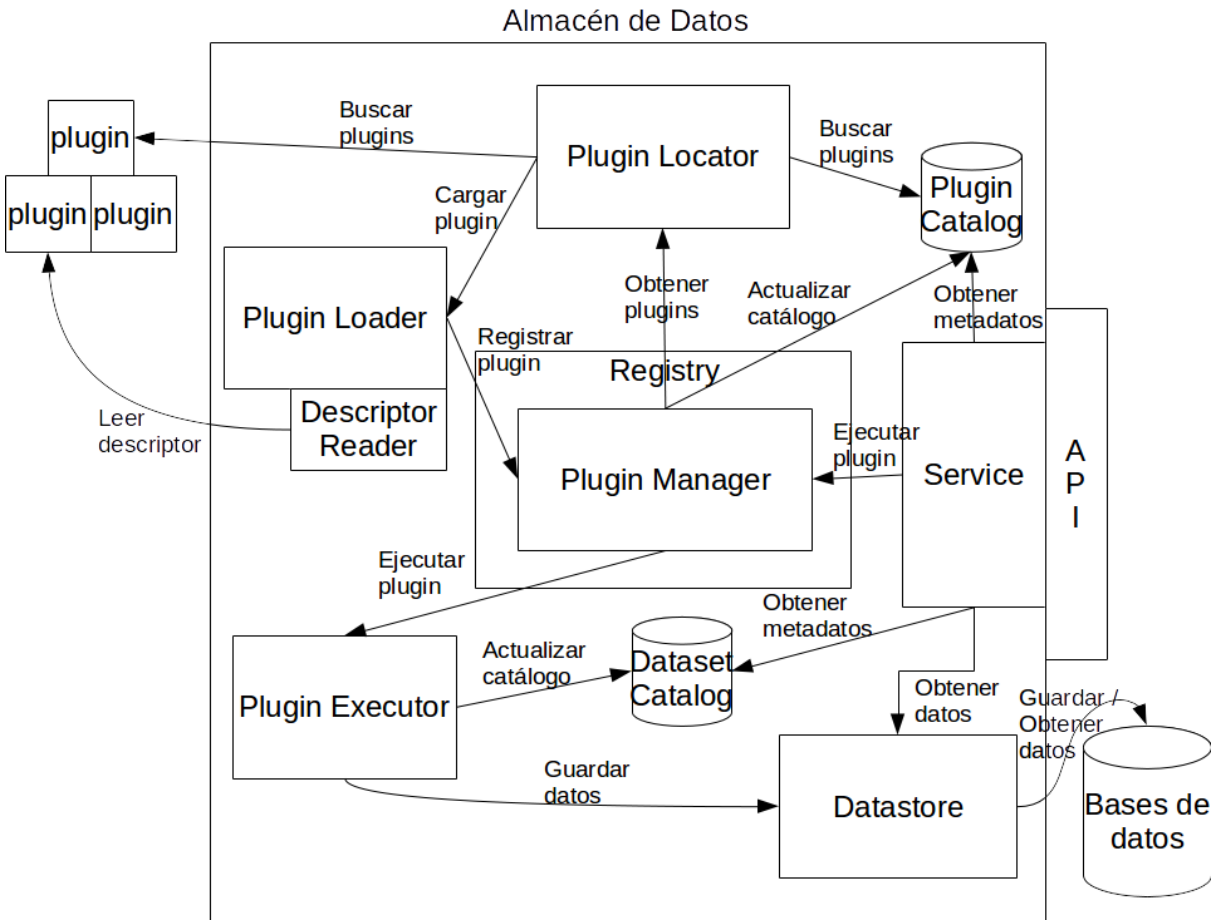


Figura 6. Arquitectura del Almacén de Datos

Los subcomponentes del Almacén de Datos son: *Plugin Manager*. *Plugin Locator*. *Plugin Loader*. *Descriptor Reader*. *Plugin Executor*. *Plugin Catalog*. *Dataset Catalog*. *Datastore*. *Service*. Bases de datos.

Para explicar la función de cada componente y las interacciones que realiza con los demás, es conveniente describir el proceso general de la ejecución de un *plugin*, así como las consultas de datos.

1. El proceso comienza cuando se detecta un nuevo archivo de *plugin* para ser agregado a la arquitectura. El *Plugin Manager*, el núcleo del componente de *Registry*, manda una señal al *Plugin Locator* para obtener el nuevo *plugin* detectado (*Obtener plugins*).

2. El *Plugin Locator* busca y localiza el archivo de *plugin* requerido en el área designada para éstos (*Buscar plugins*). En la práctica, la aplicación del Almacén de Datos destina un directorio especial para albergar los archivos de *plugin* que se utilizarán.

3. Una vez localizado el archivo de *plugin* de interés, es el deber del *Plugin Loader* cargarlo *(Cargar plugin)*. Cada *plugin* es distribuido en archivos comprimidos con extensión .plugin, cada archivo tiene una determinada estructura uniforme para todos los *plugins* de la misma versión. Cada archivo .plugin cuenta con un archivo descriptor interno que detalla los metadatos de ese *plugin*, dicho archivo es utilizado por el *Plugin Loader* para cargarlo a la arquitectura.

4. *El Plugin Loader* utiliza al *Descriptor Reader* para leer específicamente el archivo descriptor del *plugin* a cargar (*Leer descriptor*). Después de leer la información necesaria y recrear el contenido del archivo descriptor en memoria, el *Plugin Loader* termina de cargar las clases Java principales del *plugin*.

5. El último paso en este proceso es realizado de nuevo por el *Plugin Manager.* Al haber cargado todos los archivos y clases Java del *plugin* en la arquitectura, es momento de que el *plugin* se registre en el *Plugin Manager* mediante un identificador que permita referenciarlo al momento de ejecutarlo (*Registrar plugin*). Además de registrarlo en sí mismo, el *Plugin Manager* también duplica el registro del *plugin* en el *Plugin Catalog*, a

fin de contar con redundancia al momento de requerirse obtener los metadatos de un *plugin* (*Actualizar catálogo*).

El siguiente proceso importante después del registro de un nuevo *plugin* es la ejecución del mismo, esto comienza cuando el API externo recibe una llamada para ejecutar un *plugin* en específico.

1. El método del API de ejecutar *plugin* es llamado desde el exterior; entonces el componente *Service* se encarga de llamar al *Plugin Manager* para indicarle que ejecute el *plugin* solicitado, junto con los argumentos que se pudieran presentar en la solicitud (*Ejecutar plugin*).

2. El *Plugin Manager* realiza el mismo procedimiento de búsqueda que en el proceso anterior: busca el *plugin* en su propio registro, en caso de no encontrarlo llama al *Plugin Locator* para que éste se encargue de encontrarlo, ya sea en el *Plugin Catalog* o en último recurso cargándolo, debido a que no se encontraba agregado a la arquitectura.

3. Después del paso anterior el *Plugin Manager* llama al *Plugin Executor* para que realice su única función, que es ejecutar el *plugin* indicado junto con sus argumentos (*Ejecutar plugin*). En este momento el control de la ejecución pasa de la arquitectura principal al *plugin* elegido. Una vez terminada su ejecución los datos deben estar importados correctamente al almacén. El *Plugin Executor* accede a las bases de datos del almacén mediante la capa de *Datastore* (*Guardar datos*). Durante su ejecución el *plugin* se asegura de actualizar la información del *Dataset Catalog* con los nuevos registros, variables, o *datasets* agregados (*Actualizar catálogo*).

En última instancia, otras funciones que realiza la capa de *Service* es la obtención de metadatos tanto del *Plugin Catalog* como del *Dataset Catalog*. Estas consultas son accionadas cuando el API externo recibe una llamada para obtener dichos metadatos. Se realiza un

proceso similar al recibir una petición de obtención de datos climáticos (no metadatos), sólo que en esta ocasión se llama directamente a la capa *Datastore* para recuperar los datos solicitados.

Entre los metadatos que se asocian a cada *dataset* se encuentran su nombre, fecha de creación y última modificación, cobertura espacial y temporal de la malla, resolución espaciotemporal, entre otros. Los metadatos asociados a un *plugin* son aquellos incluidos en su descriptor. Para concluir este capítulo, se describe con más detalle la estructura interna de un *plugin* típico y el rol de cada componente en el proceso de ejecución. En la Figura 7 se ilustra dicha estructura.

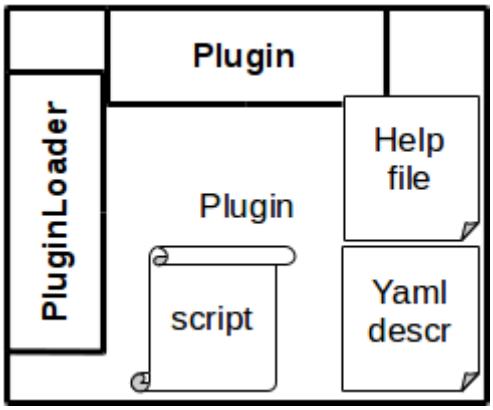


Figura 7. Estructura interna de un plugin

Los *plugins* se distribuyen en archivos comprimidos .plugin, para diferenciarlos de archivos normales. Cada uno de estos archivos debe empaquetar ficheros similares a los expuestos en la Figura 7, de lo contrario, el *plugin* no podrá ser utilizado por el Almacén de Datos. Cada archivo contiene el código Java, scripts y demás archivos necesarios para su ejecución. El fichero .plugin no es más que un .jar con la estructura y extensión modificada.

Un archivo *plugin* debe contar forzosamente con implementaciones en Java de las interfaces *PluginLoader* y *Plugin*. La primera para realizar los procedimientos necesarios y proveer un mecanismo para que el *Plugin Loader* cargue al *plugin* en cuestión, la segunda para que el

Plugin Manager y el *Plugin Executor* puedan interactuar con el *plugin* en lo que respecta a su ejecución y obtención de metadatos.

Otro archivo necesario con el que debe contar es el descriptor, el cual es codificado en YAML. Éste guarda información vital para que el *plugin* sea manejado de forma adecuada por la plataforma. El *Plugin Loader* utiliza dicho descriptor para aprender la manera de cargar cada *plugin*, usando el *Descriptor Reader* específicamente para leer este tipo de archivos. En la Figura 8 se incluye un ejemplo.

```yaml
plugin:
  name: TRMMDumper
  version: 1.0.0
  description: >
    Plugin to store the TRMM NetCDF dataset in PostgreSQL PostGIS Raster,
    for the wanted spatial and temporal space, version 1
  script:
    - /script/dumpCF.py
    - /script/loadRaster
    - /script/ncUtils.py
  help-file: /script/help/README.txt
  loader-class: plugin.Loader
  programming-languages:
    - name: Python
      version: 3
    - name: Bash
      version: 4.3
  target-OS:
    - Linux
  target-DB: PostgreSQL
  supported-files:
    - format:
        - nc
        - nc4
  dependencies:
    - name: PostgreSQL
      type: DB
    - name: psql
      type: command
    - name: python3
      type: command
    - name: raster2pgsql
      type: command
    - name: bash
      type: command
    - name: log4j-api-2.5.jar
      type: jar
    - name: log4j-core-2.5.jar
      type: jar
    - name: dataplugin.api_2.0.0.jar
      type: jar
  vendor: jdosornio
```

Figura 8. Archivo descriptor de ejemplo

El archivo “script” es la abstracción de la implementación del proceso de importación o almacenamiento de datos en sí. Éste puede ser un archivo de script real, esto es, en un lenguaje de programación distinto a Java como Python, Perl, AWK, o una clase o conjunto de clases en Java. La flexibilidad de la arquitectura permite ejecutar código en Java o en algún lenguaje de script, esto para tener la opción de reutilizar códigos de procesamiento de datos previamente desarrollados en otros lenguajes pero con algunas modificaciones para adaptarse a la estructura propuesta. También se muestra la posibilidad de agregar un archivo de ayuda con instrucciones para la ejecución del *plugin*, el cual es opcional.

Se construyeron múltiples *plugins* para probar la funcionalidad de almacenamiento de la arquitectura, alternando entre diferentes *datasets* en malla, lenguajes de programación y bases de datos utilizadas.

6 Dataplugin - Procesos involucrados

En el capítulo anterior se presenta la descripción de la arquitectura Dataplugin, en este capítulo se describe el proceso general y los subprocesos asociados a ésta. El proceso general en el que participa la arquitectura consiste en las operaciones de almacenar, segmentar, recuperar, exportar y visualizar datos climáticos.

La operación de almacenar se resume en los subprocesos y actividades asociadas a la importación y almacenamiento de datos climáticos provenientes de distintos *datasets*, en el Almacén de Datos de la arquitectura. Las acciones involucradas son la construcción de un *plugin* y el almacenamiento de un *dataset*, las cuales forman parte del subproceso de importación de datos climáticos.

La construcción de un *plugin* comprende las tareas que se esperan realizar por los diferentes roles involucrados para desarrollar un *plugin* y cargarlo a la arquitectura. El desarrollo de un *plugin* se debe realizar cuando se desee almacenar un *dataset* para el cual no exista un *plugin* para ello. Para la acción de almacenamiento de un *dataset*, se está refiriendo al hecho de almacenarlo en las bases de datos de la arquitectura, lo cual se logra una vez desarrollado el *plugin* adecuado y ejecutándolo. Por ello se considera a la construcción de un *plugin* y el almacenamiento de un *dataset* como tareas que forman parte del subproceso de importación de datos climáticos.

Segmentar consiste en realizar consultas y aplicar filtros a los datos almacenados en la arquitectura, para recuperar un subconjunto de estos. Por ejemplo, recuperar los datos de un área como una cuenca de río, un estado o un país, cuando se tienen datos de todo el mundo. La operación de segmentación está presente en los casos de uso en donde no sea requerido trabajar con la totalidad de los datos, sino una parte de éstos.

Recuperar se considera como la acción de obtener datos almacenados, segmentados o no, para cualquier propósito o caso de uso. En lo que respecta a la arquitectura Dataplugin, la recuperación puede dividirse en exportación o visualización.

La exportación consiste en entregar datos climáticos solicitados en un formato de archivo en especial, para su descarga. Los formatos de archivo disponibles dependen del alcance y funcionalidad del cliente de software utilizado para ello. Por ejemplo, se desea exportar los resultados de una consulta sobre el *dataset* Livneh en formato CSV.

La visualización involucra también la consulta de datos, pero en este caso los resultados son presentados al usuario en vez de ser enviados a descarga con un formato de archivo en específico. La presentación puede ser realizada de varias maneras: mediante gráficas de barras, de líneas, visualización de puntos en un área geográfica, mapas de contorno, tablas, resúmenes, entre otros; dependiendo de las funcionalidades desarrolladas en el cliente utilizado.

Más ejemplos de visualización gráfica podrían ser: un mapa de contorno de una variable para toda el área comprendida por el *dataset* (para una fecha o promedio de fechas), visualización de la localización de estaciones climatológicas en un mapa, visualización de una fotografía aérea (en caso de que sea un *raster* de un satélite), gráfica de líneas del comportamiento de una variable o variables en un punto a través del tiempo. Gráfica de barras de los mínimos, máximos, y normales climatológicas, de una variable para una región en un periodo particular.

En realidad cualquier tipo de visualización estadística que se le ocurra al investigador podría ser realizada por los clientes de software: por ejemplo, histogramas de los valores tomados por una variable en un área en particular (como gráfica de barras o alguna otra técnica).

Tanto la segmentación, como la recuperación, exportación y visualización forman parte del subproceso conocido como exportación de datos climáticos, y ésta se conforma por las

acciones de construcción de un cliente y la recuperación misma de los datos. A continuación se describen las actividades y los roles involucrados en este subproceso, así como también el subproceso de importación de datos climáticos.

6.1 Descripción de roles

Se identificaron diez roles que participan en los procesos anteriormente mencionados, los cuales van desde experto en la fuente de datos hasta el administrador de la plataforma. A continuación, se describe cada uno de ellos:

Experto en la fuente de datos: Puede ser un técnico de estación, el responsable de un servidor de archivos o simplemente aquel que conozca dónde obtener datos o *datasets* en malla de alguna fuente en especial, implicando que tiene cierto nivel de conocimientos en el área de climatología y los datos en malla en general. Conoce el proceso de generación de los datos.

Experto en los datos: Es el experto en entender los datos en malla presentados en un *dataset*, pudiendo generar información y conocimiento útil a partir de éstos. Se le puede llamar de otro modo, como por ejemplo el experto de en el área de climatología, ya que puede conocer los valores y su significado, además de algunos términos y definiciones que giran alrededor de los datos. Es experto en interpretar los datos en malla y en el procesamiento que se requiere aplicar a ellos para que sean utilizables.

Analista: Experto en dialogar con los usuarios y especificar requerimientos de software a partir de necesidades expuestas por ellos. Adicionalmente debe tener habilidad para el análisis de los datos y sus fuentes. Analista de software y datos.

Desarrollador: Tiene habilidades de programación y análisis de datos, debe manejar varios lenguajes de script, entre los cuales el más importante es Python, también debe manejar Java. Debe también tener conocimiento de expresiones regulares y procesamiento de archivos de texto y de otros formatos de datos. Debe poder construir código eficiente en el manejo de los recursos computacionales y contar con el conocimiento necesario para acceder a las bases de datos de la plataforma.

Diseñador: Tiene habilidades para crear diagramas de clases, modelos de diseño de software, diagramas de secuencias, diagramas de estado, etc. y todos aquellos diagramas que se requieran para describir la arquitectura y comportamiento de un *plugin* o un cliente de software. Estos diagramas están a nivel del lenguaje de implementación.

Diseñador de base de datos: Experto en modelado de datos y en construcción de bases de datos tanto relacionales como no relacionales. Es esencial que tenga entendimiento de la estructura de los datos en malla para que pueda proponer un esquema de datos robusto, basándose en su experiencia previa con varios DBMSs.

Probador: Realiza las pruebas de carga de datos directamente a la base de datos y también las pruebas de integración del *plugin* y la prueba de carga de datos desde el *plugin*. También realiza las pruebas necesarias al cliente de software.

Encargado de proyecto: Es el encargado del proyecto de desarrollo de software, ya sea la construcción de un *plugin* o la construcción de un cliente de software nuevo.

Administrador de la plataforma: Es el encargado de dar mantenimiento y administrar la plataforma. Registra nuevos *plugins*, los elimina y los ejecuta.

Usuario: Es el interesado en utilizar la plataforma de software. Su interacción es a través de los clientes de software desarrollados para extraer datos climáticos de la plataforma. El usuario

puede ser un experto en los datos (investigador, meteorólogo, etc.), o un interesado sin conocimientos profundos sobre el área de climatología. Esto depende del público al que vaya dirigido cada cliente desarrollado.

6.2 Descripción de procesos

6.2.1 Proceso de importación

El proceso de importación consiste en todas las actividades que se deben realizar para que un *dataset* nuevo quede almacenado en el Almacén de Datos, desde el levantamiento de requerimientos y la construcción del *plugin* hasta el almacenamiento de los datos. Este proceso abarca los siguientes subprocesos: Levantar requerimientos de datos, Obtener *datasets,* Limpiar datos, Analizar *datasets,* Diseñar *plugin,* Diseñar esquema, Implementar esquema, Implementar *plugin,* Realizar pruebas de carga directa, Realizar pruebas de integración, Realizar pruebas de carga por *plugin,* Validar *plugin,* Registrar *plugin, y* Ejecutar *plugin.*

Levantar requerimientos de datos: Proceso en el cual el analista, el experto en la fuente de datos y el experto en los datos llegan a un acuerdo para los requerimientos que tendrá que cumplir los datos a almacenar dentro de la plataforma. Ya sea la información o preguntas que se desean contestar o simplemente los datos que se desean almacenar, para lo cual ambos expertos deben de aportar sus conocimientos. Puede trabajarse en conjunto con levantar los requerimientos del cliente.

Obtener *datasets*: Obtener los *datasets* ya sea de forma manual o automática. Ejemplo, descargar de un servidor ftp, de una agencia, sensor, o descargarlo a manera de *stream* desde una fuente de datos. En este caso intervienen el experto en la fuente de datos (el que sabe cómo y dónde se generan), el desarrollador y el analista. Si se desea obtener los datos de

forma automática entonces el desarrollador deberá implementar una forma de realizar dicha obtención.

Limpiar datos: Este proceso consiste en procesar los datos crudos de tal forma que queden listos para ser almacenados, ya sea removiendo datos inválidos, registros erróneos, valores extremos, etc. Esta tarea le corresponde al experto en los datos. En el caso de que este proceso deba ser repetido cada vez que lleguen datos nuevos (*stream*), entonces será necesario que el desarrollador implemente una funcionalidad para aplicar el proceso de limpieza indicado por el experto a los datos nuevos.

Analizar *datasets*: El analista examinará la estructura y formato de los *datasets* obtenidos para entender cómo están compuestos y cómo es posible almacenarlos. El experto en los datos intervendrá ocasionalmente cuando su experiencia respecto a los datos sea requerida por el analista.

Diseñar *plugin*: Diseñar la estructura del *plugin* a implementar para cargar los *datasets*. Consiste en generar diagramas de clases, secuencias, paquetes, y todos aquellos diagramas necesarios para que el desarrollador pueda implementar la funcionalidad deseada. El diagrama de modelo de datos no se realiza en este proceso.

Diseñar esquema: Este proceso es exclusivo al diseño y modelado del esquema que los datos a almacenar necesitan y que se debe implementar. El modelo de datos es diagramado por el diseñador de base de datos con base a los requerimientos especificados por el analista.

Implementar esquema: Implementa el esquema de base de datos diseñado, ya sea de una base de datos relacional o una base de datos no relacional. El deber del diseñador de base de datos es crear el nuevo repositorio para los datos que se desean almacenar. Pondrá a disposición del desarrollador además la documentación y APIs necesarias para poder utilizar dicho esquema al implementar el *plugin*.

Implementar *plugin*: Proceso en el cual se construye el código que refleje lo planteado por el diseñador del *plugin*. El desarrollador debe programar todas las funciones que satisfagan los requerimientos de datos especificados por el analista. A su vez, si se trata de datos de *stream* también se debe de tomar en cuenta los mecanismos de obtención y limpieza automática de los datos en caso de que no se haya programado todavía. Documentar todo lo programado.

Realizar pruebas de carga directa: Realizar varias pruebas de carga con los datos a almacenar en una base de datos de prueba, con estructura exacta a la que se va a crear en la plataforma, esto se realiza con el objetivo de validar que el *plugin* está desarrollado correctamente. La base de datos de prueba se creará en un entorno diferente al de la plataforma. La prueba de carga se realiza directamente con el script desarrollado.

Realizar pruebas de integración: Realizar las pruebas de integración del nuevo *plugin*, registrándolo satisfactoriamente en un *Plugin Manager* de prueba diferente al entorno de ejecución de la plataforma principal.

Realizar prueba de carga por *plugin*: Prueba la carga de todos los datos a almacenar a través del *plugin* registrado en el *Plugin Manager* de prueba. Se registran todos los acontecimientos.

Validar *plugin*: El experto en los datos observa la información almacenada en la plataforma de prueba por el *plugin* y valida que ésta sea la que necesite. El experto en la fuente de datos valida los aspectos técnicos de los datos como la precisión de los decimales, etc. El encargado de proyecto entrega la documentación del *plugin* al administrador de la plataforma y a los expertos.

Registrar *plugin*: El administrador de la plataforma registra el nuevo *plugin* probado y validado en la plataforma principal para proceder a cargar los datos nuevos.

Ejecutar *plugin*: Ejecuta un nuevo *plugin* registrado para cargar los datos a la plataforma principal. El esquema debe estar creado en ella previamente, a menos que el *plugin* se encargue de crear el esquema también. Una vez terminada su ejecución los datos se almacenan en la plataforma.

6.2.2 Proceso de exportación

El proceso de exportación comprende todo el proceso de desarrollo de un cliente de software que puede ser montado en la plataforma para extraer *datasets* existentes. Este proceso abarca los siguientes subprocesos: Levantar requerimientos del cliente, Analizar procesamiento de datos, Diseñar cliente, Implementar cliente, Probar cliente, Validar cliente, Desplegar cliente, y Recuperar datos.

Levantar requerimientos del cliente: El encargado de proyecto, en conjunto con el analista levantarán los requerimientos que el experto en los datos necesita para el cliente que quiere que se desarrolle.

Analizar procesamiento de datos: Analizar si es necesario realizar algún procesamiento a los datos almacenados antes de presentarlos o ser exportados por el cliente. Para este paso se asume que ya existen los datos necesarios almacenados en la plataforma, de lo contrario es necesario comenzar con el proceso de Importación.

Diseñar cliente: Crea todos los modelos y diagramas necesarios en el lenguaje de implementación que cumplan con los requerimientos y el resultado del análisis del procesamiento a los datos que se generaron anteriormente. Esta es la entrada para que el desarrollador implemente lo generado por el diseño.

Implementar cliente: Programar el cliente diseñado. Utilizar todas las librerías y APIs necesarias, provistas por la plataforma para recuperar los datos.

Probar cliente: Realizar las pruebas de funcionalidad, integración, etc. para verificar que el cliente funciona correctamente.

Validar cliente: Prueba de aceptación por parte del experto en los datos.

Desplegar cliente: Instalar el cliente en donde sea necesario.

Recuperar datos: Este proceso comprende la utilización del cliente de software desarrollado para visualizar o exportar los datos almacenados en la plataforma. Este proceso es realizado por el usuario: el interesado en obtener información útil de los datos, ya sea directa o indirectamente del cliente de software. Por ejemplo, un agricultor podría obtener información útil directamente al utilizar un cliente que le indique la predicción de la siguiente precipitación en su área. Un investigador podría obtener información útil indirectamente del cliente, descargando datos crudos para realizar análisis por su cuenta.

En la Figura 9 y la Figura 10 se muestran diagramas de caso de uso para cada uno de los procesos descritos en este capítulo.

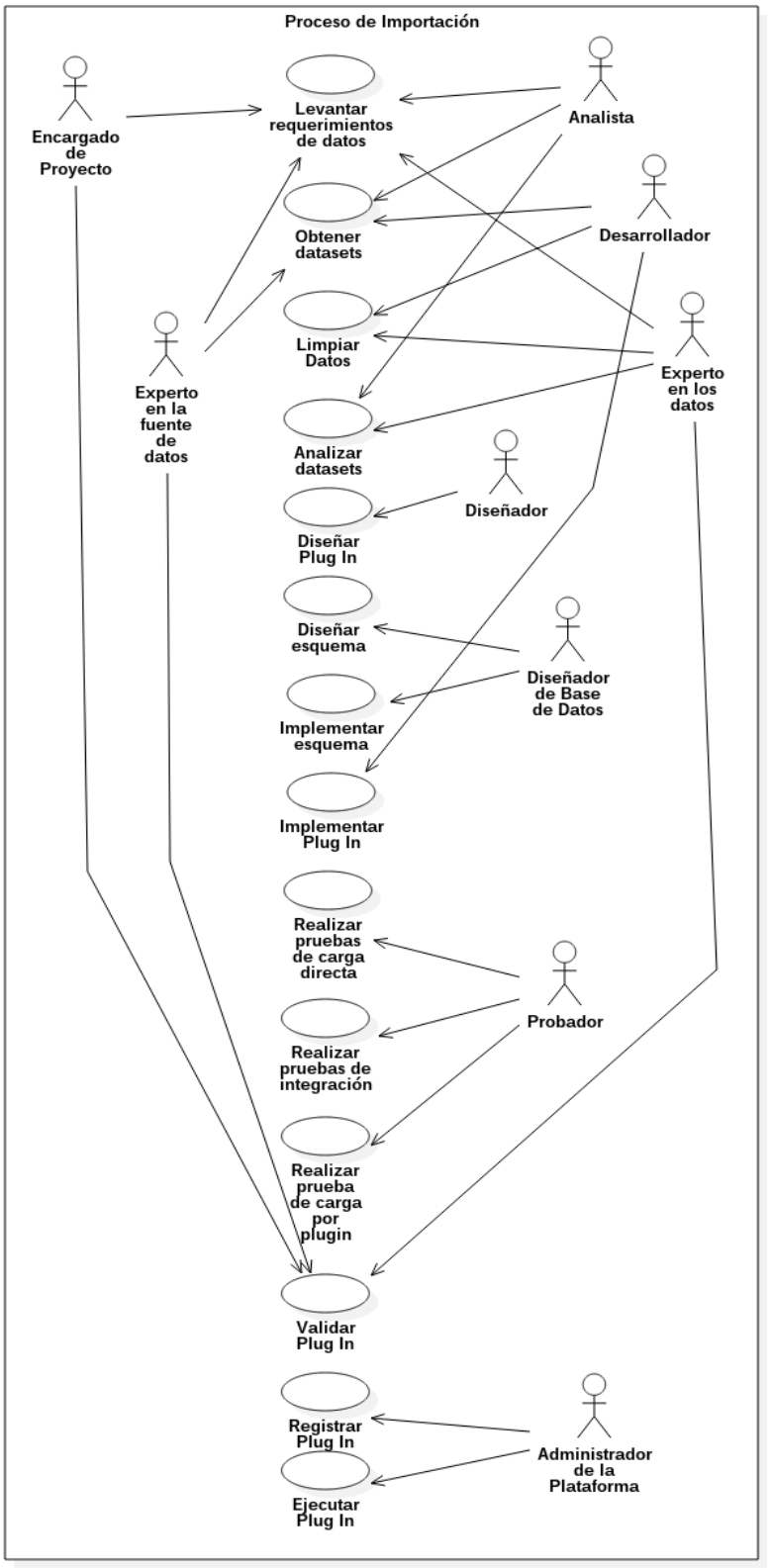


Figura 9. Proceso de Importación de datos

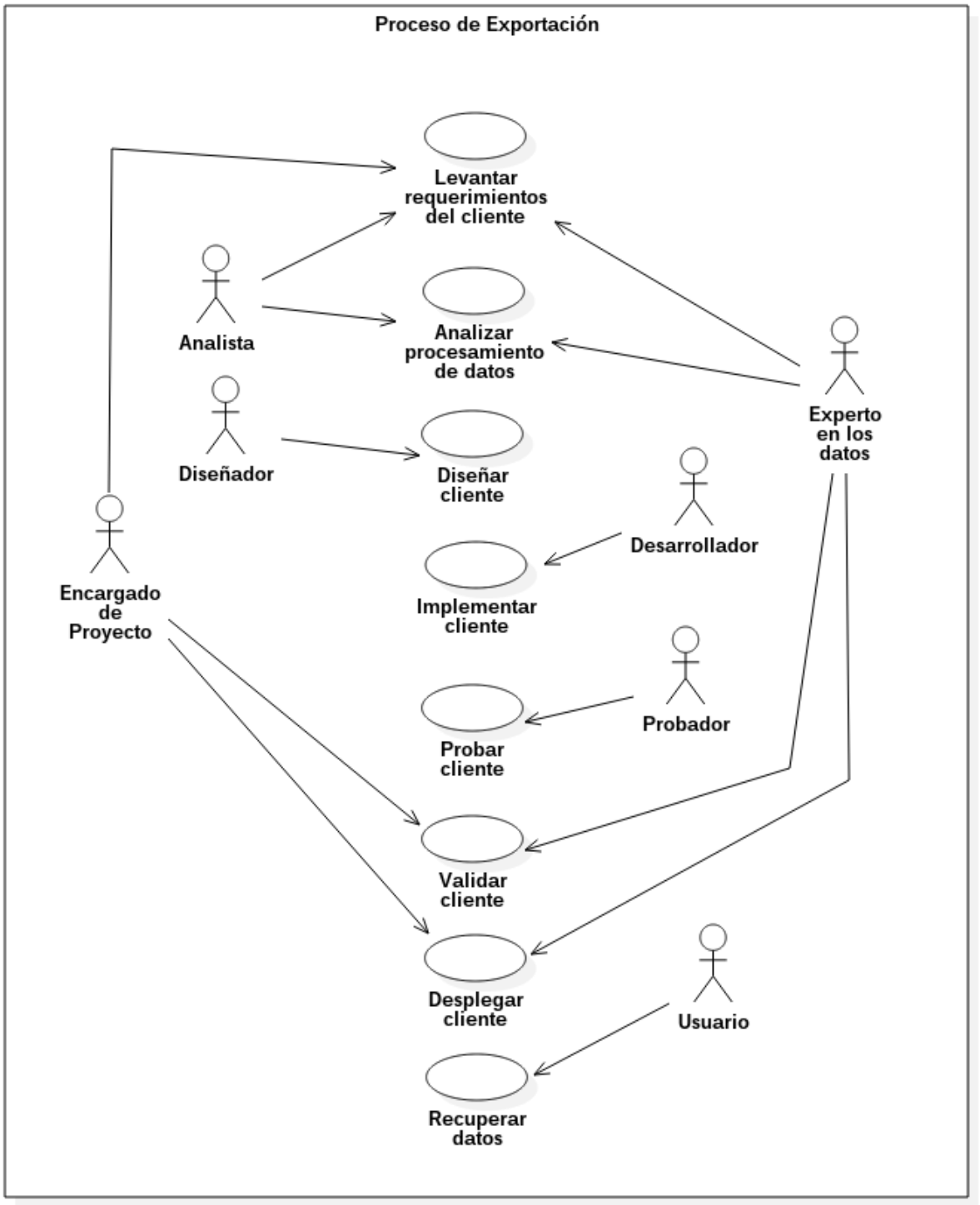


Figura 10. Proceso de Exportación de datos

7 Revisión de DBMSs para almacenar datos climáticos en malla

En diferentes etapas del desarrollo de la arquitectura de software, se prestó especial atención al o los DBMSs que la arquitectura o plataforma podría utilizar para resguardar los *datasets* climáticos. Se buscaba conocer cuál es el mejor DBMS para dicha tarea así como el esquema de datos a utilizar.

Para este trabajo se considera un esquema de datos a la representación de la configuración lógica de una base de datos. Éste puede ser expresado de forma visual o mediante lenguaje de definición de datos. Indica cómo las entidades que conforman la base de datos se relacionan entre sí (adaptada de Lucid Software, 2018).

Para elegir la mejor opción de manera objetiva, se consideraron varias características de cada DBMS: el esfuerzo requerido para implementar o construir la base de datos, el tiempo de respuesta respecto a consultas de datos, tiempo de carga del *dataset* a la base de datos, y espacio de almacenamiento utilizado.

Debido a que el enfoque principal de este estudio es el almacenamiento de *datasets* climáticos estructurados en malla, se realizaron varias pruebas con este tipo de *datasets*. Para elegir la mejor opción, se realizó una revisión o comparación de distintos DBMSs. Esta revisión se realiza debido a que no se conocían qué opciones de DBMSs existían en el mercado para el almacenamiento de estos *datasets*. En base a la experiencia previa que se tenía sobre DBMSs, y mediante una investigación más profunda acerca de sus características, se optó por elegir tres DBMSs para la comparación: PostgreSQL, MongoDB y Rasdaman. Las razones por las que se eligieron éstas son por las diferentes características de cada una.

PostgreSQL es una base de datos relacional orientada a objetos. Es *open source*, tiene detrás una comunidad muy fuerte y participativa, se actualiza con frecuencia, es utilizado por infinidad de negocios e instituciones académicas, su arquitectura está bien probada, y cuenta con varias extensiones funcionales (PGDG, 2018a).

Una extensión por la cual se eligió PostgreSQL es PostGIS, la cual es una extensión para el manejo de objetos geográficos y la integración de funciones que permiten ejecutar consultas de locación en SQL (PostGIS, 2018a). Con esta extensión es posible utilizar PostgreSQL para el almacenamiento de datos climáticos y realizar consultas en base a un área geográfica sobre ellos.

Se consideró también a la base de datos MongoDB por utilizar un paradigma de almacenamiento diferente al relacional. Esta base de datos está orientada a documentos, esto quiere decir que cualquier registro añadido se considera un documento en vez de una fila en una tabla relacional. También es *open source* y de libre uso.

Cada documento es una estructura de datos compuesta de pares de campo y valor, donde los valores pueden ser cualquier tipo de dato simple así como otros documentos anidados, arreglos, y arreglos de documentos (MongoDB Inc, 2018). Los documentos se agrupan mediante colecciones, análogas a las tablas en una base de datos relacional.

La ventaja de las bases de datos orientadas a documentos sobre otros tipos de bases de datos es su flexibilidad y libertad de esquema: no es necesario definir una estructura previamente para poder almacenar datos en este tipo de bases de datos. Esto permite evitar la rigidez del modelo relacional al requerir definir un esquema de datos previo al almacenamiento de registros, pudiendo almacenar cualquier registro o “documento”. Otras características por las que se consideró a MongoDB para esta revisión fueron su alto rendimiento, enriquecido lenguaje de consulta, alta disponibilidad, y escalabilidad horizontal (MongoDB Inc, 2018).

Por último, se eligió Rasdaman por estar enfocado directamente al almacenamiento de datos en *raster* o arreglos multidimensionales. Rasdaman (Raster Data Manager) es un motor de arreglos flexible y escalable. Permite el almacenamiento y consulta de arreglos multidimensionales masivos, tales como datos estadísticos, de simulación, imágenes y sensores, encontrados en ciencias como las de la Tierra, espacio y de la vida (Rasdaman, 2018). Rasdaman parecía la opción más adecuada y que encajaba de cierta manera en las características de almacenamiento que se estaban buscando.

También se consideró a la base de datos Cassandra como cuarta opción. Cassandra es una base de datos no relacional orientada a columnas. Se caracteriza por ser escalable y altamente disponible sin comprometer rendimiento, tolerante a fallos, permitir replicación y distribución a través de múltiples *data-centers* (The Apache Software Foundation, 2016). Sin embargo, esta opción fue descartada debido a que, después de cargar segmentos de *datasets* climáticos como prueba, se encontró que el lenguaje de consulta ofrecido no proveía la flexibilidad que se buscaba, debido a ciertas restricciones en los filtros condicionales que se podían realizar.

Estas restricciones son principalmente la inhabilidad de poder ejecutar consultas donde se utilice como condicionante un rango de valores en dos columnas distintas. Esto significa que, siguiendo un esquema similar al propuesto para PostgreSQL (Véase Figura 12), no es posible realizar las consultas requeridas.

Para ampliar la explicación, el esquema propuesto para Cassandra era una tabla con las columnas {tiempo, latitud, longitud, valor-variable}, donde tiempo es una columna que contiene fecha y hora de la medición, mientras que latitud y longitud son columnas de punto flotante que representan la ubicación, y la columna valor-variable el valor de la medición para una variable dada. Cabe mencionar que Cassandra es muy parecida al modelo relacional en que ambos

organizan los datos en tablas y columnas. De hecho, su lenguaje de consulta CQL (*Cassandra Query Language)*, es casi idéntico a SQL, salvo por algunas instrucciones.

Se eligió este básico esquema debido a la falta de tipos de datos geométricos como aquellos disponibles en PostGIS. Para resumir, no era posible buscar entre rangos de valores para la columna latitud y longitud en la misma consulta, mucho menos agregando un rango más para el tiempo. Estos rangos eran necesarios para buscar segmentos de datos tanto espacial como temporalmente. Pero fue esa restricción aunada a que, desde un principio, Cassandra no tiene soporte para operaciones espaciales, las limitantes que impidieron a Cassandra formar parte de esta revisión. A continuación se describen las pruebas realizadas y los resultados de la comparación.

7.1 Desarrollo de las pruebas

El método desarrollado para la revisión fue el siguiente:

1. Investigar diferentes DBMSs capaces de almacenar *datasets* en malla y recuperar segmentos de éstos de forma eficiente; y después seleccionar los más adecuados.

2. Elegir un *dataset* climático en malla para probar las capacidades de almacenamiento y recuperación de los diferentes DBMSs seleccionados.

3. Extraer datos de muestra del *dataset* de prueba y cargarlos en cada DBMS.

4. Ejecutar una serie de consultas a cada DBMS para evaluar el tiempo de respuesta y el número de registros recuperados.

5. Comparar los resultados, analizarlos y seleccionar el DBMS más adecuado.

Para la realización de las pruebas se utilizó un sistema Ubuntu 16.04 LTS, con 15.5 GB de RAM, procesador Intel core i7-3630QM CPU @ 2.40GHz y 8 hilos, de arquitectura 64-bit y disco duro con sistema de archivos ext 4. Se seleccionaron las versiones más actualizadas de cada base de datos al momento de la prueba, siendo éstas PostgreSQL 9.6.5 con la extensión PostGIS 2.3.3, MongoDB 3.4.9, y Rasdaman 9.5.0.

El *dataset* elegido para la prueba fue el TRMM Near Real-Time Precipitation L3 3 hour 0.25 degree x 0.25 degree V7 (Huffman, 2016). Este *dataset* contiene datos de precipitación organizados en una malla de 480 latitudes x 1,440 longitudes con una resolución de 0.25 grados. Está dividido en archivos NetCDF que comprenden un periodo de 3 horas cada uno, del año 2000 al presente.

Para el propósito de esta comparación se descargaron los archivos del 2000 al 2016, aproximadamente 49,640 archivos en total. Cada archivo tiene un tamaño aproximado de 850 KB. El tamaño total estimado del *dataset* es de 40.24 GB. Se estima un número total de 34,311,168,000 observaciones. En la Figura 11 se puede apreciar una muestra de los datos contenidos en este *dataset* así como el área comprendida.

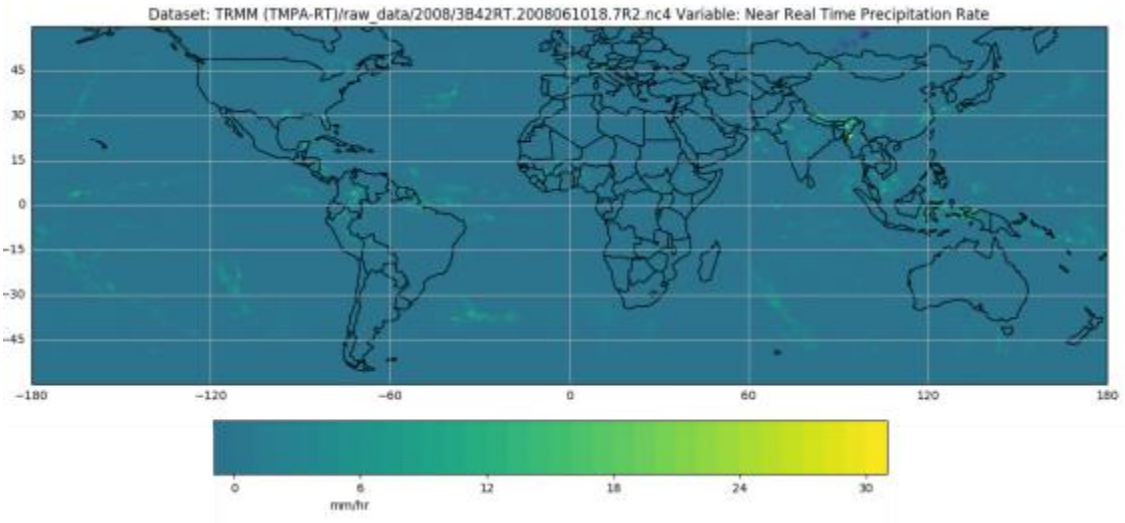


Figura 11. Mediciones de precipitación observada en el dataset TRMM para el día 10 de Junio de 2008 a las 6 pm

Para extraer y cargar una muestra de los datos descargados a cada DBMS, se desarrolló un script en Python. Este script extrae y guarda la muestra en un archivo temporal para que éste pueda ser cargado en cada DBMS. La muestra de datos elegida es un segmento desde 2016-01-01 hasta 2016-01-07. Aproximadamente 39,398,400 observaciones. Para el script en Python se utilizaron las librerías o módulos netCDF4, numpy, y re (*regular expression*).

El esquema de datos o la estructura con la que se almacena la muestra en cada DBMS varía según el paradigma del modelo de datos (relacional, documentos, arreglos), la flexibilidad del lenguaje de definición de datos (tipos y estructuras de datos), las herramientas o extensiones disponibles, y las restricciones de cada DBMS.

Para los casos de PostgreSQL y MongoDB, se decidió que la malla de muestra fuera almacenada dividiéndola en miles de registros puntuales, donde cada registro equivaldría a un punto en el espacio de la malla, para un momento específico, y acompañado del valor de medición de una o distintas variables para ese punto y momento en particular.

Por lo tanto, para el caso de PostgreSQL se generó una tabla con millones de filas, una por punto y momento en el tiempo, mientras que en MongoDB se generó una colección de millones de documentos, con el mismo principio. Es fácil vislumbrar el crecimiento exponencial de registros que esta aproximación tiene, debido a que para sólo una semana de datos en la malla global se registraron aproximadamente 39 millones de observaciones. Esto genera tablas o colecciones inmensamente grandes.

La ventaja de esta aproximación es que una vez almacenados los registros, éstos contienen tanto los datos o valores de las mediciones, así como los metadatos de éstas, como son la ubicación y momento de la medición. Se pueden almacenar distintas mediciones de variables para el mismo punto en el espacio y tiempo, agregándolas como columnas o nuevos campos, dependiendo de si el modelo es relacional u orientado a documentos, respectivamente.

Tanto MongoDB como PostgreSQL cuentan además con consultas basadas en locación y otras operaciones geográficas como intersección, cercanía a un punto, o validar que un punto se encuentra dentro de un polígono. PostgreSQL, especialmente, cuenta con la extensión PostGIS, que le permite realizar todo tipo de operaciones y utilizar varios tipos geográficos estándares como punto, polígono, multi-polígono y línea. El caso de MongoDB es más limitado en cuanto a las operaciones y tipos de datos soportados, pero es suficiente para el alcance de esta revisión.

El caso de Rasdaman es diferente. Esta base de datos utiliza otro modelo de datos o manera de organizar los datos a almacenar. Cómo está orientada a arreglos, es factible almacenar los datos de una malla de una forma más directa, suponiendo que ésta se encuentre guardada en un formato de archivo orientado a arreglos como NetCDF. En este caso no se almacenaron millones de registros en una tabla o colección representando los millones de puntos espacio-temporales de la malla, sino que se generaron colecciones de arreglos multidimensionales que representan cada uno las mediciones de una variable para toda la malla en todo el periodo de tiempo cubierto.

Estos arreglos son de tres dimensiones, donde la primera corresponde al tiempo, mientras que las últimas dos corresponden a la latitud y la longitud, resultando en un arreglo de momentos en el tiempo x latitudes x longitudes. El valor de cada celda en este arreglo representa la medición de una variable para ese momento, latitud y longitud en particular. Este esquema tiene la ventaja de no manejar una cantidad inmensa de registros, como en los casos anteriores. Debido a la organización en arreglos, los datos se pueden recuperar prácticamente de forma directa, utilizando las operaciones de arreglos permitidas por el lenguaje de consulta.

Aquellos programadores que han utilizado arreglos alguna vez durante su carrera, pueden corroborar la facilidad con la que se recuperan los valores de un arreglo en la mayoría de los

lenguajes de programación: Sea un arreglo A de forma (3, 2, 2), que contenga tres momentos, dos latitudes y dos longitudes para una variable en especial. Es posible recuperar la observación en el segundo momento, primer latitud y segunda longitud mediante la instrucción A(2, 1, 2). Existen otras operaciones de arreglos posibles en Rasdaman como los *slices*, pero el objetivo de esta revisión no es ahondar en los aspectos técnicos.

Sin embargo, es importante mencionar que así como almacenar los datos en arreglos representa una ventaja importante, ese mismo hecho representa a su vez una desventaja. Los mismos programadores recordarán que los arreglos son estructuras de datos que no soportan de manera nativa los metadatos. Dicho de otra manera, el tener un arreglo tridimensional de distintos valores reales no aporta ningún valor si éstos no se pueden interpretar, y para ello es necesaria una estructura aparte que indique qué momento, qué latitud y qué longitud representa una posición en el arreglo. A diferencia de los casos anteriores, este esquema de datos no cuenta con los metadatos necesarios para interpretar los datos (ubicación y momento en el tiempo), por lo que es necesario contar con un repositorio de metadatos aparte que complemente a los datos almacenados en Rasdaman.

Tampoco cuenta propiamente con funciones o expresiones para realizar consultas basadas en la locación o cualquier otra operación geográfica. Al representar los datos como simples arreglos, el lenguaje de consulta se limita a las operaciones de arreglos estándar.

En la Figura 12 se muestra de manera resumida el proceso de exportación realizado y los esquemas de almacenamiento utilizados.

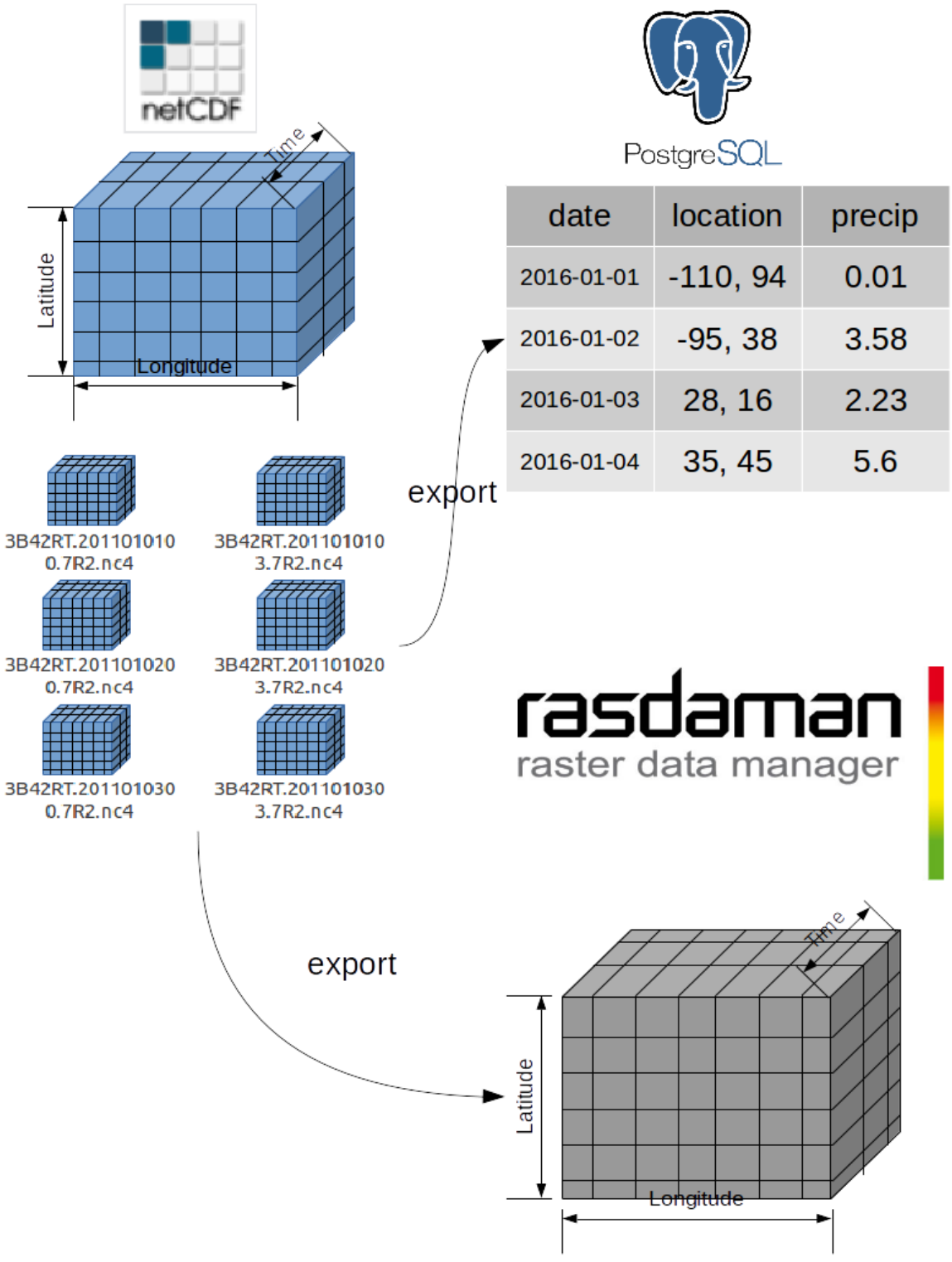


Figura 12. Proceso de exportación de la muestra de TRMM a los esquemas de datos implementados.

Una vez cargados los datos en cada DBMS, el paso inmediato fue definir las consultas a realizar sobre los datos cargados, para ello se definieron siete consultas como se describen enseguida:

1. Obtener los datos en malla dentro de un rectángulo delimitador arbitrario en EU. Desde Utah hasta el sur de Minnesota. Un rectángulo con esquina inferior izquierda en -110 longitud, 38 latitud (-110, 38), y esquina superior derecha en -95 longitud, 44 latitud (-95, 44) (Q1).

2. Obtener los datos en malla dentro de un rectángulo delimitador que cubra el territorio de México. Un rectángulo con esquina superior izquierda en (-117.905, 32.689) y esquina inferior derecha en (-86.315, 12.659) (Q2).

3. Obtener los datos en malla dentro de un rectángulo delimitador que cubra el territorio de EU. Un rectángulo con esquinas en (-125.746, 49.484) y (-66.359, 23.378) (Q3).

4. Obtener los datos puntuales dentro de una aproximación al área de la cuenca del Río Grande, representada por un polígono de 12 vértices (Q4).

5. Obtener los datos en malla dentro de la misma área considerada para la consulta 1, pero delimitando los registros del 2016-01-02 al 2016-01-04 (Q5).

6. Obtener los datos en malla dentro de la misma área considerada para la consulta 3, pero delimitando los registros del 2016-01-02 al 2016-01-04 (Q6).

7. Obtener los datos puntuales dentro de la misma área considerada para la consulta 4, pero delimitando los registros del 2016-01-02 al 2016-01-04 (Q7).

En la Figura 13 se muestra de manera gráfica las siete consultas definidas.

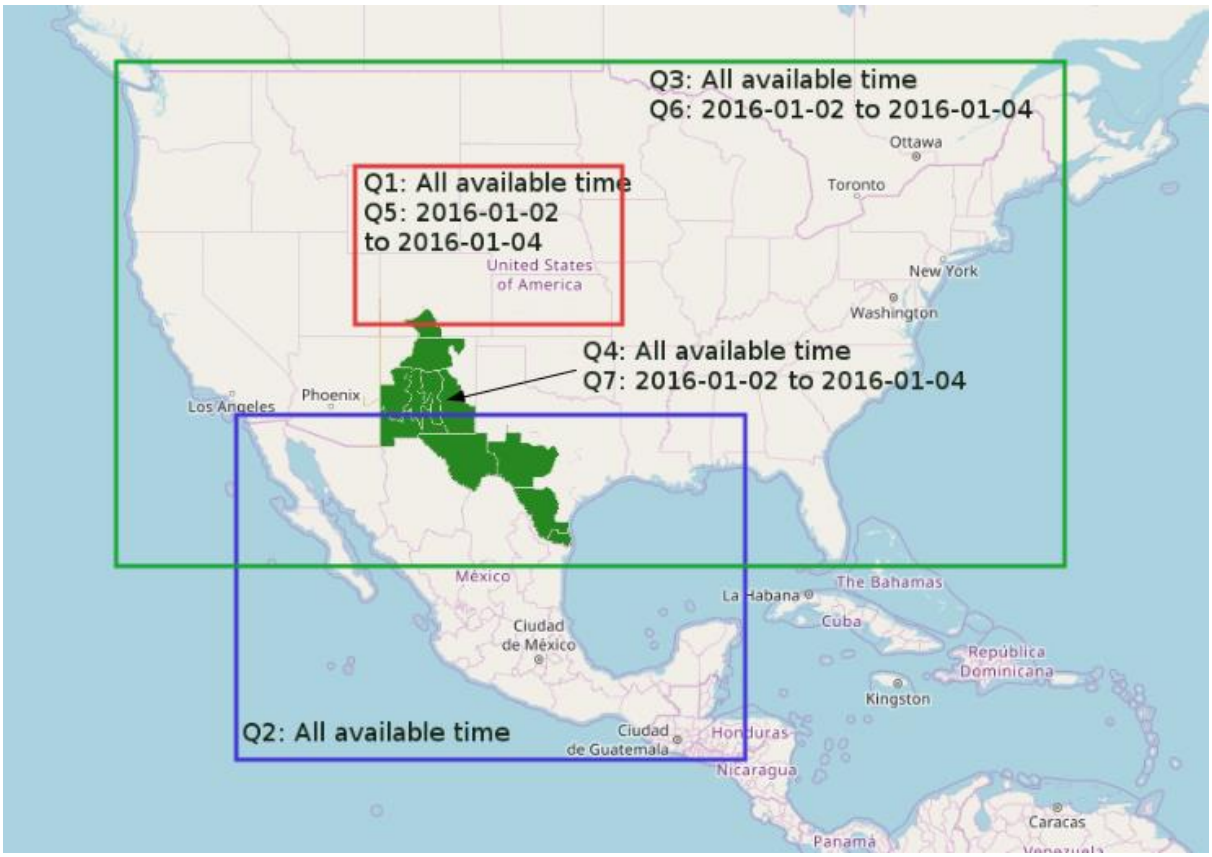


Figura 13. Consultas de prueba definidas involucrando restricciones espaciales y temporales

En la Tabla 1 se presenta de forma resumida los resultados obtenidos en cada consulta de prueba y cada base de datos.

Tabla 1. Resultados de consulta para cada DBMS*, mostrando el tiempo de respuesta y el número de observaciones recuperadas. El tamaño de la base de datos y el tiempo de ingestión también se muestran*

	PostgreSQL	MongoDB	Rasdaman
Q1	1.082 s 82,080 obs	1.308 s 82,080 obs	0.089 s 82,080 obs
Q2	5.953 s	2 m 50.769 s	0.244 s

	579,120 obs	579,120 obs	579,120 obs
Q3	14.079 s 1,410,864 obs	4 m 55.211 s 1,410,864 obs	0.278 s 1,410,864 obs
Q4	0.750 s 38,988 obs	6.847 s 41,382 obs	No fue posible
Q5	0.636 s 24,480 obs	42.138 s 23,040 obs	0.067 s 24,480 obs
Q6	6.569 s 420,784 obs	1 m 36.255 s 396,032 obs	0.146 s 420,784 obs
Q7	0.440 s 11,628 obs	2.732 s 11,616 obs	No fue posible
Tamaño DB	5.84 GB	1.66 GB	300.58 MB
Tiempo de ingestión	30 m 57.28 s	15 m 1.6 s	6.995 s

Se puede observar en la tabla que la base de datos con el mayor tiempo de ingestión es PostgreSQL, seguido de MongoDB y por último Rasdaman. El tiempo de ingestión indica el tiempo transcurrido para almacenar los datos de muestra en la base de datos.

La ingestión comprende la extracción de la muestra del *dataset* de prueba, la transformación de los datos a un archivo intermedio, y la importación del archivo intermedio a cada base de datos. Incluye también el tiempo transcurrido para la generación de los índices necesarios para agilizar la consulta; éstos sólo fueron generados en PostgreSQL y MongoDB.

Para el caso de PostgreSQL se generó un archivo intermedio CSV para cargarlo mediante su herramienta de importación COPY. Para MongoDB se generó un archivo intermedio JSON y se cargó mediante su herramienta de importación análoga. Para el caso de Rasdaman, no fue necesario transformar los datos a otro formato de archivo, sin embargo, fue necesario agregar los múltiples archivos NetCDF en los que se encontraba distribuido el *dataset*, de tal manera que Rasdaman pudiera cargar directamente un sólo archivo NetCDF. Una vez terminada la carga, se crearon índices en las columnas o campos de fecha y posición tanto para PostgreSQL como para MongoDB.

Como se puede observar también, debido a los mecanismos de almacenamiento propio a cada base de datos, el espacio de almacenamiento requerido por cada una varía, siendo PostgreSQL la que requiere más espacio y Rasdaman la que menos ocupa. Sin embargo, aun siendo la base de datos que más requiere espacio y más tarda en cargar los datos, PostgreSQL mantiene tiempos de respuesta mucho menores en todas las consultas realizadas respecto a MongoDB, donde la última tarda hasta 6 veces más. Esta relación queda minimizada al observar que Rasdaman resulta ser mucho más rápida en casi todas las consultas, excepto en aquellas que no son posibles realizar.

Para tratar de explicar por qué ocurre este comportamiento vale la pena describir cómo es que se obtuvieron estas mediciones. En el caso del espacio de almacenamiento, se sumaron los tamaños de los datos y de cualquier índice aplicable a ellos. En esta característica el motor de almacenamiento de cada base de datos jugó un papel importante.

Para la medición del tiempo de respuesta de cada consulta, se realizó la medición en base al volcado de todos los datos resultantes de la consulta en el formato de mayor conveniencia para cada base de datos. Esto quiere decir que el tiempo de respuesta no representa el tiempo transcurrido para que la base de datos regrese el puntero al resultado, sino el tiempo transcurrido para que dicha base de datos escriba el resultado en disco, en el formato de archivo normalmente utilizado. CSV para PostgreSQL, JSON para MongoDB y NetCDF para Rasdaman.

Se consideró tomar esta aproximación debido a que las consultas de datos se realizan para transferir o procesar sus resultados, no sólo para encontrar un puntero a ellos. Se optó por el almacenamiento en disco local debido a que es más rápido que enviarlo a través de una red o a la salida estándar.

Es importante notar que no fue posible ejecutar las consultas Q4 y Q7 a Rasdaman debido a las limitantes del lenguaje de consulta y a la estructura del esquema de datos utilizado. Estas consultas involucran la obtención de datos o puntos dentro de un polígono irregular (cuenca del Río Grande). Este tipo de operación es trivial para la funcionalidad que MongoDB y PostgreSQL ofrecen, pero este no es el caso para Rasdaman.

Rasdaman no cuenta con funciones geográficas similares a PostgreSQL y MongoDB, sino que se enfoca al manejo de arreglos multidimensionales masivos. Debido a la estructura con la que se almacenaron los datos de la muestra, fue posible para Rasdaman ejecutar todas las demás consultas, por medio del uso de *slices* en las dimensiones del arreglo. Pero, en el caso de las consultas Q4 y Q7, le resultó imposible debido a las restricciones del lenguaje de consulta y a la falta de metadatos de ubicación.

Se consideró una forma de atacar este problema mediante la adición de un repositorio de metadatos en una base de datos que soporte operaciones geográficas, pero esta aproximación

necesitaría del desarrollo de código “pegamento” que actúe como intermediario entre el cliente y Rasdaman, por lo que la opción fue descartada para esta prueba.

7.2 Conclusiones

A primera vista, Rasdaman podría ser visto como el DBMS más adecuado para el almacenamiento de *datasets* en malla, y la recuperación de segmentos de éstos de forma eficiente, pero éste también tiene algunas desventajas no encontradas en los otros dos DBMSs.

Rasdaman es un DBMS orientado a arreglos, así que todos los datos son almacenados en arreglos de n-dimensiones con un sólo tipo de dato base (real, entero, etc). Se necesita un catálogo de metadatos separado para interpretar los datos almacenados en estos arreglos.

El lenguaje de consulta no es muy flexible ni permite formular consultas complejas similares a otros lenguajes como SQL. Está especializado en obtener secciones contiguas de arreglos, especificando un rango de índices por dimensión (*slices*). Es por eso que no fue posible formular todas las consultas de prueba, específicamente aquellas que involucran obtener datos de puntos dentro de un polígono irregular, ya que Rasdaman no cuenta con esta característica como los otros DBMSs.

Otra desventaja es que la comunidad de Rasdaman no es tan activa como en los otros DBMSs, por lo que el proyecto no se actualiza frecuentemente. Además, la curva de aprendizaje es alta y la documentación del proyecto no es tan extensiva como podría ser, así que aprender a cómo usar Rasdaman puede ser más difícil que en los otros DBMSs.

Estas desventajas son abordadas por los otros DBMSs con un lenguaje de consulta más expresivo, extensa documentación y comunidades activas, y soporte de diferentes tipos geográficos y operaciones espaciales.

Las desventajas con MongoDB y PostgreSQL son que el espacio de almacenamiento, el tiempo de ingestión de datos y el tiempo de respuesta de las consultas son mucho mayores que en Rasdaman, siendo PostgreSQL el que requiere más espacio de almacenamiento y tiempo de ingestión.

Una ventaja con Rasdaman contra los otros DBMSs es que éste tiene una funcionalidad para exportar directamente los datos consultados en diferentes formatos tales como CSV, NetCDF, GeoTIFF, entre otros. PostgreSQL también cuenta con esta característica, mediante la extensión PostGIS, pero es necesario instalar los *drivers* GDAL de los formatos de archivo deseados.

En conclusión, no existe una panacea para almacenar datos en malla en un DBMS. Depende del propósito de los datos y las consultas aplicadas a ellos. Si las consultas serán principalmente obtener datos en malla dentro de un rectángulo o sección del arreglo y las exportaciones a formatos de archivo *raster* serán frecuentes, entonces Rasdaman es el camino a seguir.

Si el *dataset* a ser almacenado es suficientemente pequeño para ser manejado, se requieren consultas y tipos de datos más complejos, o el tiempo de respuesta no es tan importante, entonces PostgreSQL o MongoDB son buenas opciones.

En general, Rasdaman es suficiente para la mayoría de los *datasets* climáticos en malla, donde los datos son almacenados en arreglos de n-dimensiones. No obstante, se debe realizar un análisis de las consultas aplicadas al *dataset* antes de elegir este DBMS, ya que las restricciones de su lenguaje de consulta es una de sus mayores desventajas. Si las consultas

necesarias van más allá de simple recuperación de datos dentro de un rectángulo espacial y sobre un rango de tiempo, entonces será necesario implementar procedimientos externos para superar esta falta de capacidad en el lenguaje de consulta, potencialmente incrementando el tiempo de recuperación de los datos.

Debido a la naturaleza de la plataforma de software desarrollada, sería incorrecto concluir que un DBMS es el más adecuado para almacenar datos climáticos en malla. La respuesta correcta es que depende del *dataset* a almacenar y las consultas a realizar. No obstante, gracias a esta revisión se cuenta con una mejor idea de cuál DBMS usar para cada caso.

“Cuando se almacenan series de tiempo, es necesario diseñar un esquema de datos apropiado, que depende de la granularidad de los datos y los escenarios de consultas.” (Lian et al., 2017).

8 Construcción de *plugins*

Para comprobar el funcionamiento de la arquitectura en cuanto al proceso de Importación de datos, se construyó un *plugin* para la carga del *dataset* TRMM llamado “TRMMDumper”. A continuación se describe de manera general el proceso seguido para su construcción, el esquema utilizado para almacenar los datos en malla, y el proceso de importación específico del *plugin*.

8.1 *Plugin*: TRMMDumper

La parte medular en el proceso de importación de un *dataset* es el diseño del esquema de datos que lo almacenará. Una vez diseñado e implementado el esquema, los pasos restantes consisten en implementar las interfaces del *plugin* correspondientes y programar el proceso de extracción, transformado y carga de datos.

En el caso de nuevos tipos de fuentes de datos, el diseño del esquema de datos cobra mayor importancia, al no existir previamente esquemas en el almacén en los que basarse. Tal fue ese el caso para la construcción de este *plugin*. Fue por eso que se hizo una revisión previa de las posibles bases de datos para almacenar datos climáticos en malla, para conocer sus características y en base a ello diseñar un esquema para la opción más factible.

En base a la revisión de bases de datos realizada con anterioridad, se terminó descartando la opción de utilizar Rasdaman para almacenar los datos en malla. Esto en parte porque era necesario añadir funcionalidad con la que no contaba el lenguaje de consulta, además de implementar un repositorio de metadatos que almacenara la información del tiempo y el espacio que representaba cada cubo de datos.

Otra limitante que desalentó su uso es la falta de documentación para extender el sistema con la funcionalidad requerida, o incluso simplemente conocer los parámetros o argumentos posibles para cargar archivos en malla como NetCDF. Aunque Rasdaman muestra el potencial para almacenar grandes arreglos multidimensionales, la falta de la funcionalidad requerida en el lenguaje de consulta es su mayor limitante.

Para el almacenamiento de *datasets* en malla, específicamente para TRMM, se definieron como requerimientos la capacidad de consultar los datos tanto en el eje espacial como temporal, por un rango de tiempo o un área espacial requerida, ya sea un rectángulo arbitrario o algún polígono definido, como la cuenca de un río.

Con el esquema de cubo contemplado para Rasdaman, es posible consultar datos en base a un rango de tiempo o áreas espaciales rectangulares, más no de cualquier otro polígono. Esto ya que la primera y segunda dimensión representan el eje espacial, y puede ser recuperado mediante *slices* contiguos de cada dimensión (de aquí el rectángulo), pero no hay forma de recuperar los puntos o celdas que se encuentran dentro de cualquier otro polígono definido, ya que la funcionalidad del lenguaje de consulta se basa en operaciones de arreglos, no en operaciones espaciales de esta complejidad.

No obstante, desacorde con los resultados de la revisión para los otros DBMSs, no se pudo elegir uno de las dos para almacenar los datos en malla, pues se consideraba que la aproximación tomada sería demasiado ineficiente para *datasets* de gran tamaño; tal es el caso que en las pruebas sólo se almacenó una semana de datos en estos DBMSs, para evitar tiempos de importación largos. Por dicha razón se decidió cambiar el enfoque y el esquema de datos utilizado, pero utilizando uno de los dos DBMSs restantes.

Se optó por utilizar PostgreSQL junto con su extensión PostGIS, pero usando el tipo de dato *raster* disponible en dicha extensión, por lo que se realizó una revisión extensiva de su API de

referencia (PostGIS, 2018b). *Raster* es un tipo de dato especial añadido a la extensión PostGIS en su versión 2.0. Fue creado con el fin de extender la funcionalidad del DBMS PostgreSQL y facilitar el manejo de datos en *raster*. Se construyó con el propósito de implementar un tipo de dato *raster* tan similar al tipo *geometry* como fuera posible, y ofrecer un sólo conjunto de funciones SQL de superposición, que operen sin problemas en coberturas de *raster* y vectores (PostGIS, 2018c).

La estructura del tipo de dato *raster* consiste en una malla, matriz, o arreglo de dos dimensiones que puede albergar cualquier tipo de dato numérico, pero siendo éste homogéneo en cada *raster*. Este tipo de dato funciona como cualquier otro en PostgreSQL, en el sentido de que puede ser asignado a cualquier columna de una tabla. Cada *raster* es almacenado uno por fila como datos binarios, o de manera externa en forma de archivos.

Cada *raster* almacenado tiene también asociados metadatos que lo describen: largo y alto en píxeles, tamaño geoespacial en X y Y del pixel, origen (coordenadas X y Y de la esquina superior izquierda), rotación en X y Y, *Spatial Reference ID* (SRID), y ruta del archivo, en caso de ser un *raster* externo. Gracias a estos metadatos es posible representar los datos del *raster* espacialmente, lo que permite recuperar segmentos del *raster* mediante coordenadas directamente, en vez de índices en un arreglo (aunque también es posible de esta forma).

Cada *raster* puede contener a su vez una o más bandas. Cada banda contiene propiamente los datos del *raster* o malla. Es una especie de instantánea que contiene los datos de una cobertura definida, la cual puede representar un momento en el tiempo, un nivel de altura, etc.

Como cada *raster* puede contener múltiples bandas, se puede decir que éste puede tener múltiples momentos o niveles de altura. No obstante, cada banda comparte los mismos metadatos geoespaciales del *raster*, por lo que puede cada una contener diferentes datos, pero para una misma cobertura espacial. Si se compara con un cubo en Rasdaman podría verse a

cada banda como primera y segunda dimensión, mientras al conjunto de bandas como la tercera.

El hecho de incluir el tipo de dato *raster* en la extensión PostGIS facilita la integración de éste con las funciones existentes de dicha extensión, por lo que se obtiene un extenso soporte de operaciones espaciales, así como la capacidad y expresividad de consulta del lenguaje SQL. Es posible transformar los datos *raster* a polígonos o puntos, y viceversa, mediante funciones SQL incorporadas al DBMS.

En base a este tipo de dato, a sus características y a las funciones ofrecidas por la extensión, se diseñó un nuevo esquema de datos, en el cual los datos de TRMM se organicen en *rasters* o mallas completas, en vez de dividirlas en puntos individuales.

Después de realizar varias pruebas con el nuevo tipo de dato, diferentes alternativas de organización, y analizando los escenarios de consulta, se propuso el siguiente esquema (Figura 14):

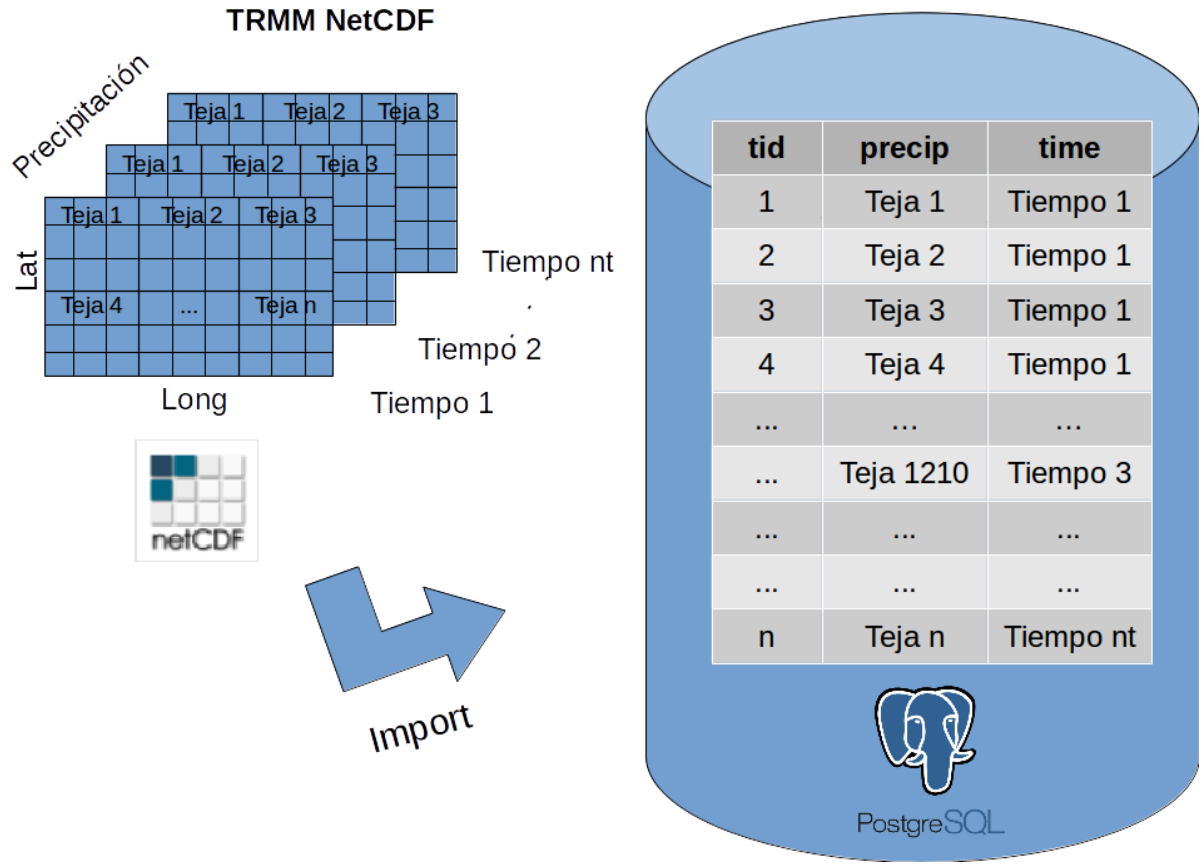


Figura 14. Esquema de datos implementado en PostgreSQL para TRMM. Se muestra la relación entre el dataset NetCDF (izquierda) y el esquema diseñado (derecha)

En la Figura 14 se puede apreciar el esquema de datos propuesto para almacenar TRMM. A la izquierda se muestra la estructura general del *dataset* en NetCDF. Un conjunto de mallas o *rasters* representando cada una un tiempo o momento específico, y cada *raster* representa la misma cobertura espacial que los demás. Cada celda de la malla representa una medición de precipitación para un punto dado, y cada malla tiene una dimensión de 1,440 longitudes x 480 latitudes. En el esquema de datos utilizado en la revisión anterior, se siguió el enfoque de almacenar cada punto y cada tiempo en una fila de una tabla, a manera de: tiempo, punto (longitud, latitud), y medición de precipitación. Esto significa que para este esquema, por cada malla almacenada, se estarían guardando 691,200 filas (1,440 longs * 480 lats) para un sólo tiempo. En una semana se estarían guardando aproximadamente 38,707,200 filas.

Este enfoque obviamente no es manejable conforme la serie de tiempo crece, por lo que, dada la funcionalidad ofrecida por el *raster* de PostGIS, se optó por almacenar las mediciones de precipitación, junto con sus ubicaciones o puntos, en tejas. En este caso se está refiriendo a una teja como una parte de la malla original, la cual es un *raster* o malla en sí misma, pero de un tamaño menor y más manejable que la malla original.

En la parte derecha de la Figura 14 se presenta el esquema de datos elegido para TRMM: Una tabla con tres columnas, donde cada fila representa una teja. La columna *tid*, la cual es un entero que funciona como identificador de la teja. La columna *precip*, que almacena la teja en sí, y la columna *time*, la cual indica el momento en el que se midieron los datos representados por la teja. La columna *precip* es de tipo *raster*.

Para cada tiempo en el *dataset*, se divide la malla original en tejas de igual tamaño, para después almacenarlas una por fila en la base de datos. Por ejemplo: al almacenar la malla del "tiempo 1" en la tabla, la malla sería dividida en t tejas, convertidas en t filas en la tabla, las cuales tendrían "tiempo 1" como valor en la columna *time*, con los datos de las tejas en la columna *precip*. El siguiente conjunto de tejas para la malla del "tiempo 2", tendrían a su vez asignado dicho valor en la columna *time*.

Esta aproximación se tomó para, además de evitar el excesivo número de filas del esquema anterior, dividir cada malla en piezas de igual tamaño, que pudieran distribuir la cobertura de la malla original entre ellas. De esta forma, es posible recuperar sólo las tejas que sean solicitadas, en lugar de toda la malla cada vez que se requiera, y también se evitan los gastos (tamaño de índices, tiempo, etc.) de realizar búsquedas a través de billones de filas en la tabla, ya que su número se reduce considerablemente.

Para este *dataset*, se dividió la malla original de 1,440 x 480 en tejas de 30 x 30, resultando en un total de 768 tejas por malla, lo que equivale a un total de 768 filas por cada malla, contra las

691,200 del esquema anterior. Se está hablando de una reducción de 900 veces el número de filas.

Este esquema también tiene la posibilidad, en caso de que se requiera, de añadir más variables como nuevas columnas. Además, si fuera necesario añadir también niveles de altura a la malla, se podrían añadir fácilmente por medio de bandas al *raster* de cada teja (una banda por nivel).

Debido a que el número de filas seguirá siendo grande dado el número de tiempos comprendido por este *dataset*, aún con la división por tejas, es necesario añadir índices que optimicen el tiempo de respuesta para las consultas de datos consideradas. Por este motivo, se añadieron dos índices a la tabla después de cargarla con los datos:

1. Índice *BTree* en la columna *time*, con *clustering* de la tabla en dicho índice.
2. Índice *GIST* en el *casco convexo* de la columna *precip*.

El índice *BTree* es la opción por defecto de PostgreSQL al crear índices. *BTree* puede manejar consultas de igualdad y rangos en datos que pueden ser ordenados de alguna manera (PGDG, 2018b). Básicamente un índice *BTree* implementa la búsqueda binaria para encontrar los valores buscados para una columna o columnas. La complejidad del índice *BTree* es O(ln(n)).

Precisamente por estas características fue que se eligió un índice de este tipo para la columna *time*, debido a los requerimientos de consultar datos de precipitación en un rango de tiempo. Para mejorar el rendimiento de las consultas aún con un gran número de registros, se optó también por aplicar *clustering* a la tabla basado en éste índice. De acuerdo a la documentación de PostgreSQL: “Cuando una tabla es agrupada (*clustered*, en inglés), ésta es reordenada físicamente en base a la información del índice” (PGDG, 2018c). El objetivo de este reordenamiento es mantener los datos similares agrupados físicamente, para que el DBMS pueda recuperarlos en menos accesos al disco.

El segundo y último índice considerado es el *GIST* aplicado al *casco convexo* de la columna *precip*. *GIST* (Generalized Search Tree) es otro tipo de índice utilizado por PostgreSQL; técnicamente estos índices no son un sólo tipo, más bien son una infraestructura dentro de la cual diferentes estrategias de indexación pueden ser implementadas (PGDG, 2018b). Tal es el caso de la estrategia utilizada por PostGIS para indexar los tipos de datos geométricos y de *raster* ofrecidos en su extensión. Como la otra parte de los requerimientos contemplaba consultas en base al espacio, tanto en simples rectángulos como en polígonos más complejos, se agregó este tipo de índice a la columna *precip*.

Al indexar el *casco convexo* de cada teja, lo que se está utilizando en realidad es solamente la caja que delimita el área de la teja, en vez de la teja completa. El *casco convexo* también soporta rotación, pero esa característica no aplica para este *dataset* en particular. A continuación se muestra una imagen para ampliar la explicación (Figura 15).

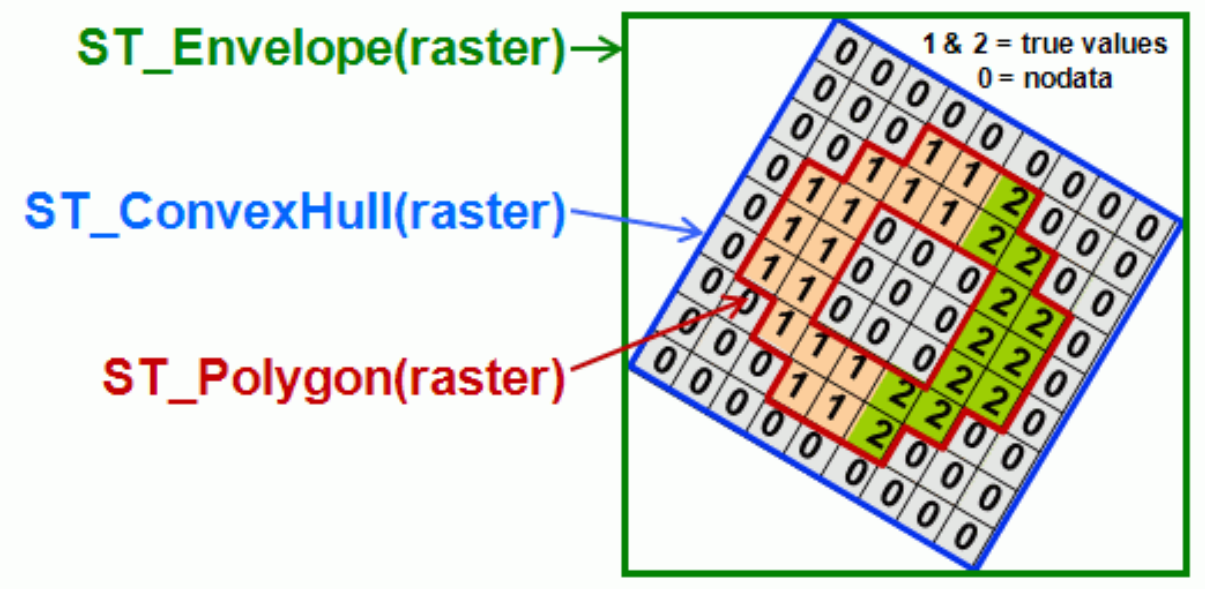


Figura 15. Ejemplo del casco convexo (convex hull) de un raster. (Extraído de PostGIS, 2018d)

De esta manera, es posible realizar consultas eficientes que filtren registros en base a la columna *precip*. Por ejemplo, con condiciones relacionales como intersección, contiene, es contenido, etc.

Ejemplificando en la consulta de “obtener las mediciones de precipitación contenidas en el polígono P. El primer paso sería encontrar todas las tejas de la tabla cuyo *casco convexo* intersectara con el área del polígono P, esto se logra de manera eficiente con el índice *GIST*. Una vez descartado un gran número de tejas innecesarias, se puede aplicar la *operación de cortado* en las tejas restantes para obtener las mediciones requeridas. En general, éste es el esquema propuesto para almacenar los datos de TRMM y los datos en malla con características similares a este *dataset*. A continuación se describe el proceso de construcción del *plugin*.

Para el momento de la construcción de este *plugin*, ya se habían descargado los archivos NetCDF pertenecientes al periodo del 2000 al 2016, del *dataset* TRMM. Detalles sobre el *dataset* en la sección 2.9.2 (TRMM_3B42RT). Se tomó como base el esqueleto de un prototipo de *plugin* desarrollado anteriormente para construir este *plugin*. En la Figura 16 se muestra el diagrama de clases del *plugin*:

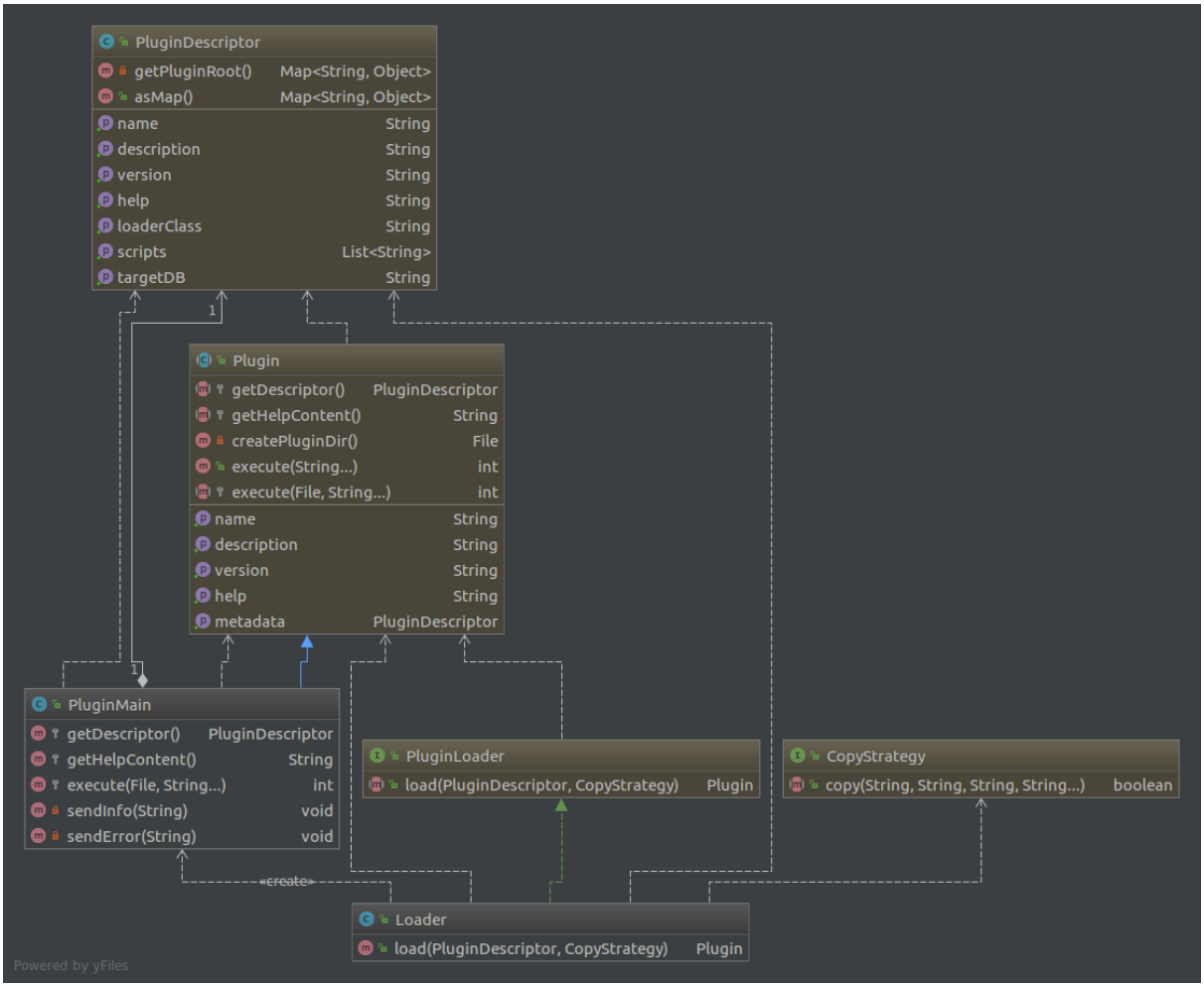


Figura 16. Diagrama de clases de TRMMDumper

Una vez implementado el esquema de datos en PostgreSQL, se continuó con las pruebas de carga directa. Estas pruebas consisten en cargar una muestra del *dataset* directamente a una base de datos PostgreSQL, para validar que el script o código Java implementado importa los datos correctamente al esquema diseñado.

Durante estas pruebas se utilizó la misma muestra de datos utilizada en la revisión de DBMSs anterior (datos desde 2016-01-01 hasta 2016-01-07). El proceso de importación se resume en la siguiente figura (Figura 17):

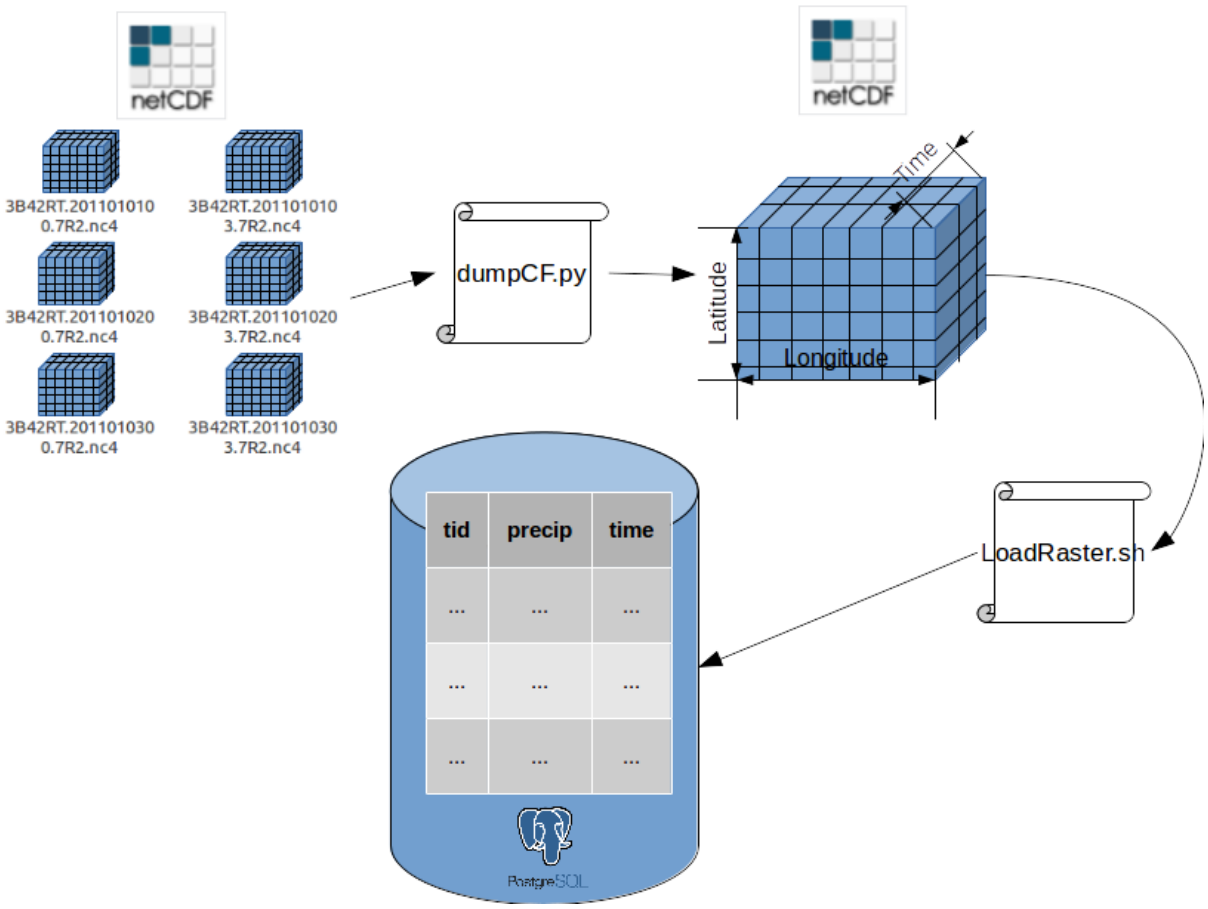


Figura 17. Proceso de importación de datos TRMM a PostgreSQL

El proceso de importación se implementó mediante dos scripts. El primero, llamado "dumpCF.py", es un script en Python utilizado para agregar las mallas de varios archivos NetCDF de TRMM en un sólo archivo, para que de esta forma se realizara la carga de un único archivo NetCDF a PostgreSQL. El script acepta parámetros opcionales como las coordenadas de las esquinas de un cuadrado, o la fecha inicial y final de un periodo de tiempo, para segmentar los datos espacial y temporalmente, respectivamente.

Cada archivo NetCDF original contiene una sola malla para un sólo tiempo, y éste se encuentra indicado en el nombre del archivo. El script dumpCF.py lee cada archivo perteneciente al periodo de tiempo solicitado y almacena su malla añadiéndola al arreglo del archivo agregado,

en la tercera dimensión. También agrega el tiempo que representa ese nuevo índice de la tercera dimensión en los metadatos del archivo.

El segundo script, llamado “loadRaster.sh”, utiliza la salida del primer script como su entrada. Su función es cargar el archivo NetCDF agregado al esquema diseñado previamente en PostgreSQL. Debido a ciertas limitaciones de la herramienta de carga de *rasters* a PostgreSQL, se tuvo que implementar el script como sigue:

Primero se requirió crear una tabla temporal con la misma estructura del esquema de datos diseñado, después se procedió a cargar los datos del archivo NetCDF mediante la herramienta raster2pgsql (PostGIS, 2018e), incluida con PostGIS. Esta herramienta fue la única manera que se encontró para incorporar datos de archivos externos a PostgreSQL. Las alternativas revisadas no ofrecían beneficios sustanciales para el proceso que se buscaba implementar, por ejemplo, la librería GDAL.

La herramienta raster2pgsql está configurada para cargar un archivo completo como un sólo *raster* en PostgreSQL. Esto significa que, si se tiene un archivo NetCDF con un arreglo de 1,440 x 480 x 2928, la herramienta carga este arreglo como un sólo *raster* de 1,440 x 480 con 2,928 bandas, toda esta información almacenada en una sola columna (*precip*) en una sola fila. Ésta también se puede configurar para dividir automáticamente el *raster* en tejas y cargarlas una por fila. Esto es una aproximación al enfoque que se estaba buscando, pero aun así la tercera dimensión del arreglo en NetCDF era tratada como las bandas de las tejas (2,928 bandas por teja).

Este no era el enfoque que se deseaba seguir. No obstante, la herramienta también ofrece un parámetro para indicar la banda o el conjunto de bandas que se desean cargar de un archivo. Con esta posibilidad, se implementó un script en Bash para repetir la ejecución de la herramienta para cargar todas las bandas del archivo, una banda a la vez. En cada ejecución

de la herramienta, se añadía una fila conteniendo una nueva malla o banda del NetCDF agregado.

Después de cargar los datos en la tabla temporal, ésta tiene nt filas, donde nt es el número de tiempos o bandas en el archivo NetCDF agregado, y cada fila contiene la malla completa de un tiempo dado. El siguiente paso del script fue cargar los datos de los tiempos faltantes en la columna *time*, ya que la herramienta no permitía ningún tipo de configuración respecto a otras columnas diferentes de la del tipo *raster* (*precip*, en este caso).

Una vez completada la información de la columna *time*, y teniendo cada malla el tiempo que le correspondía según el archivo NetCDF, se procedió a la creación de las tejas. Para ello se creó otra tabla con la misma estructura de la temporal, y se ejecutó seguidamente una sentencia SQL que seleccionara todas las filas de la tabla temporal, dividiéndolas cada una en tejas de 30 x 30 para cargarlas en la tabla creada hace unos instantes.

Por último, se eliminó la tabla temporal y se crearon los índices mencionados previamente en la tabla de las tejas. Se realizó la creación de las tejas de esta manera debido a que, si se hubiera realizado por medio de la herramienta, el proceso de importación sería más largo y más complicado de continuar en los pasos siguientes.

Ésta fue la explicación técnica del proceso de importación de TRMM. El *plugin* fue desarrollado para reproducir este proceso mediante su ejecución, utilizando los mismos scripts para ello. A continuación se muestra la estructura interna del *plugin* TRMMDumper:

El archivo .plugin no es más que un archivo .jar con la extensión diferente, por lo que en realidad es un archivo comprimido .zip. Este fichero puede contener las clases y los archivos que le sean pertinentes para la correcta importación de un *dataset*. Sin embargo, debe cumplir con estos requisitos fundamentales para poder integrarse al Almacén de Datos:

1. Archivo descriptor en lenguaje YAML llamado “desc.yaml” en la raíz del archivo .plugin, con las configuraciones mínimas necesarias.
2. Clase en Java que implemente la interface *PluginLoader* de la librería dataplugin.api
3. Clase en Java que implemente la clase abstracta *Plugin* de la misma librería.
4. Archivo de ayuda.

En la Figura 18 se presenta la estructura del *plugin* TRMMDumper.

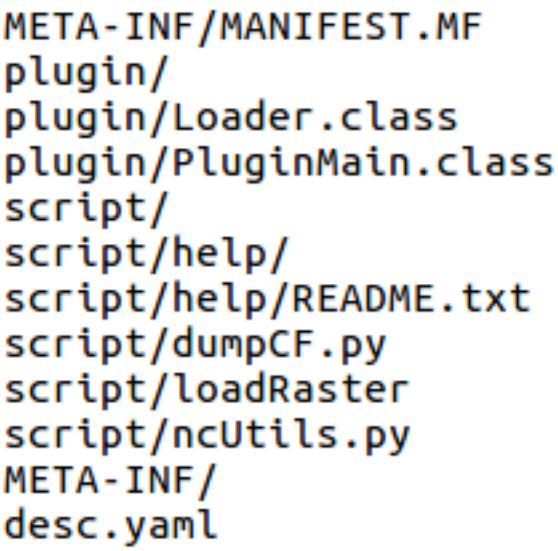


Figura 18. Estructura interna del plugin TRMMDumper

Este *plugin* contiene los elementos comunes de un archivo .jar, además del archivo descriptor en la raíz, un archivo de ayuda, y los scripts dumpCF.py y loadRaster (loadRaster.sh en la Figura 17) necesarios para el proceso de importación. PluginMain.class contiene la codificación de dicho proceso e implementa la clase abstracta *Plugin*, mientras que Loader.class implementa la interface *PluginLoader* y contiene la lógica para cargar el *plugin* la primera vez que se requiera. El archivo ncUtils.py contiene funciones utilitarias requeridas por dumpCF.py, por eso la importancia de distribuirlo en el archivo .plugin. En la Figura 19 se muestra el contenido del archivo desc.yaml:

```yaml
plugin:
  name: TRMMDumper
  version: 1.0.0
  description: >
    Plugin to store the TRMM NetCDF dataset in PostgreSQL PostGIS Raster,
    for the wanted spatial and temporal space, version 1
  script:
    - /script/dumpCF.py
    - /script/loadRaster
    - /script/ncUtils.py
  help-file: /script/help/README.txt
  loader-class: plugin.Loader
  programming-languages:
    - name: Python
      version: 3
    - name: Bash
      version: 4.3
  target-OS:
    - Linux
  target-DB: PostgreSQL
  supported-files:
    - format:
        - nc
        - nc4
  dependencies:
    - name: PostgreSQL
      type: DB
    - name: psql
      type: command
    - name: python3
      type: command
    - name: raster2pgsql
      type: command
    - name: bash
      type: command
    - name: log4j-api-2.5.jar
      type: jar
    - name: log4j-core-2.5.jar
      type: jar
    - name: dataplugin.api_2.0.0.jar
      type: jar
  vendor: jdosornio
```

Figura 19. Contenido del archivo descriptor desc.yaml

Cada descriptor de *plugin* debe tener una estructura similar a ésta. La mayoría de la información presentada en este descriptor es opcional, excepto por los siguientes campos:

name: Denota el nombre del *plugin*, es la clave con la que se podrá acceder a él una vez registrado en el *Plugin Manager* de la plataforma.

version: Número de versión del *plugin*. Se utiliza para localización de *plugins* en conjunto con su nombre.

description: Una breve descripción sobre el funcionamiento del *plugin*

script: Lista de las rutas de los scripts utilizados por el *plugin*

help-file: Ruta del archivo de ayuda del *plugin*.

loader-class: Nombre completo de la clase que implementa la interface *PluginLoader*.

target-DB: DBMS destino al que se cargarán los datos que el *plugin* importe.

Por el momento, los demás atributos presentes en el descriptor son opcionales, pero se incluyen por si se llegan a requerir en una futura implementación. Este *plugin* fue desarrollado en Java 8, excepto por los scripts mencionados, que fueron implementados en Python 3 y Bash 4.3.

Por último se realizaron las pruebas de integración y de carga por *plugin* para finalmente registrar TRMMDumper en la plataforma principal. Para las pruebas de carga por *plugin* se utilizó un periodo de tiempo más largo para los datos a almacenar, debido a que el nuevo esquema implementado permite un proceso de ingestión más rápido.

Esta vez se importaron los datos desde 2011-01-01 hasta 2012-01-01, siendo éstos un total de 2,920 mallas, una por cada 3 horas transcurridas. En el esquema anterior esto resultaría en un total de (2,920 mallas x 1,440 longs x 480 lats =) 2,018,304,000 filas a importar, pero para este esquema resulta en un total de (2,920 mallas x 768 tejas =) 2,242,560 filas. En la siguiente tabla (Tabla 2) se muestra brevemente la comparación entre los resultados del esquema actual contra el anterior:

Tabla 2. Pruebas de consultas del esquema actual contra el anterior

	Esquema actual	Esquema anterior
Q1	0.262 s	1.082 s
Q2	0.800 s	5.953 s
Q3	1.441 s	14.079 s
Q4	0.328 s	0.750 s
Q5	0.246 s	0.636 s
Q6	0.622 s	6.569 s
Q7	0.238 s	0.440 s
Tamaño DB	1.13 GB	5.84 GB
Tiempo de ingestión	22 m 52.09 s	30 m 57.28 s
Total de filas	2,242,560	38,707,200
Periodo	2,920 tiempos	56 tiempos

Nota: Como los datos importados no pertenecen al mismo periodo ni son la misma cantidad de tiempos importados, para las consultas Q1 - Q4 se consultó el periodo 2011-07-01 a 2011-07-07 (7 días) y para las consultas Q5 - Q7 el periodo 2011-07-02 a 2011-07-04 (3 días), para el esquema actual; En aras de una comparación justa.

Como se puede observar, hubo una mejora tanto en la reducción del número de filas como en el tiempo de respuesta de las consultas en general. También se redujo considerablemente el espacio ocupado por la base de datos, contando también los índices. El tiempo de ingestión se mantuvo relativamente igual entre ambos esquemas. Aunque se importaron más filas en el esquema anterior, el periodo que éstas representan es mucho más corto que en el esquema actual.

9 Descripción del API

En la siguiente sección, se describen a grandes rasgos los métodos disponibles en el API de acceso de la plataforma.

registerRasterDataset(dataset_name, plugin_name, plugin_version, variables, dbms)

Este método registra los metadatos del *dataset* en malla especificado en el *Dataset Catalog*, obteniéndolos del *dataset* con el nombre especificado, para el DBMS y la(s) variable(s) dada(s). Está diseñado para ser llamado por el *plugin* al finalizar el proceso de importación, por ello los argumentos del nombre y versión del *plugin*.

getDatasetMetadata (dataset_name) : metadata

Este método obtiene los metadatos de un *dataset* dado. Requiere como argumento el nombre del *dataset*, y regresa una cadena JSON con metadatos como el nombre del *dataset*, el *plugin* que lo importó al almacén, esquema de datos, entre otros.

getPluginMetadata (plugin_name, plugin_version) : metadata

Este método obtiene los metadatos de un *plugin* dado. Requiere como argumento el nombre del *plugin* y la versión a buscar, y regresa una cadena JSON con metadatos como el nombre del *plugin*, versión, descripción, rutas de archivos de script, clase cargadora, entre otros.

getDatasets(newer_than_date = None, older_than_date = None): dataset_list

Este método retorna una lista de los *datasets* almacenados. Tiene dos argumentos opcionales: newer_than_date, para solicitar todos los *datasets* modificados después de la fecha indicada, y older_than_date, para recuperar los *datasets* modificados antes de la fecha indicada. Retorna un arreglo JSON con el nombre y fecha de última modificación de cada *dataset*.

getPlugins(): plugin_list

Su función es obtener todos los *plugins* registrados en la plataforma. Regresa un arreglo JSON con el nombre y versión de cada *plugin*.

executePlugin(plugin_name, plugin_version, args)

Como su nombre lo indica, ejecuta el *plugin* con el nombre y la versión dada. El argumento *args* es un arreglo con los parámetros de ejecución del *plugin*.

getRasterMetadata(dataset_name) : raster_metadata

Este método recupera los metadatos de la malla de un *dataset* dado. Requiere como argumento el nombre del *dataset* en cuestión, y retorna un JSON con los metadatos de la malla: origen x, origen y, largo en pixeles, alto en pixeles, escala x, escala y, SRID y número de bandas.

getTemporalCoverage(dataset_name) : temporal_coverage

Obtiene información de la cobertura temporal de un *dataset*. Requiere el nombre del *dataset* a buscar y retorna un JSON con la fecha inicial y final de la cobertura.

getVariables(dataset_name) : variable_list

Método para recuperar las variables disponibles en un *dataset*. Requiere el nombre del *dataset* a buscar y retorna un arreglo JSON con los metadatos de las variables existentes en dicho *dataset*.

getTotalTimes(dataset_name) : total_times

Recupera el total de tiempos existentes en el *dataset* dado. Requiere como argumento el nombre del *dataset* y retorna el total de tiempos existentes. Tiempos se refiere en este caso a los momentos o fechas disponibles en un *dataset*. Es el largo de la tercera dimensión, si se quiere ver como un arreglo.

getTotalLatitudes(dataset_name) : total_lats

Recupera el total de latitudes existentes en un *dataset* dado. Requiere como argumento el nombre del *dataset* y retorna el total de latitudes existentes en la malla. Dicho de otra manera, este método retorna el alto de la malla o la segunda dimensión, si se quiere ver como un arreglo.

getTotalLongitudes(dataset_name) : total_longs

Recupera el total de longitudes existentes en un *dataset* dado. Requiere como argumento el nombre del *dataset* y retorna el total de longitudes existentes en la malla. Dicho de otra manera, este método retorna el largo de la malla o la primera dimensión, si se quiere ver como un arreglo.

getTotalHeightLevels(dataset_name): total_levels

Este método retorna el total de niveles de altura disponibles en un *dataset* dado. Requiere como argumento el nombre del *dataset* buscado, y retorna el total de niveles. Por defecto el total de niveles de una malla es uno.

getOrigin(dataset_name): origin

Este método recupera el origen de la malla del *dataset*. Requiere como argumento el nombre del *dataset*, y recupera un punto con la longitud y latitud del origen de la malla (esquina superior izquierda).

getSpatialResolution(dataset_name): spatial_resolution

Este método obtiene la resolución espacial del *dataset* solicitado. Requiere como argumento el nombre del *dataset* y retorna la resolución en longitud y latitud. La resolución espacial es la diferencia entre una posición de longitud/latitud con respecto a la siguiente.

getTemporalResolution(dataset_name): temporal_resolution

Este método recupera la resolución temporal de un *dataset*. Requiere como argumento el nombre del *dataset* y retorna la resolución temporal de éste. La resolución temporal es la diferencia que existe entre un tiempo y el siguiente.

getRasterData(dataset_name, variable_name, start_time = None, end_time = None, withinPolygon = None, output_format = raw, aggregate_operation = MEAN) : raster_data

Este método recupera los datos de una variable y *dataset* datos. Requiere como argumento el nombre del *dataset* y la variable. Los argumentos opcionales son el tiempo de inicio, tiempo de fin, polígono delimitador, formato de salida (por defecto binario) y operación de agregación (por defecto promedio). Retorna los datos solicitados para el *dataset* y la variable indicada, opcionalmente dentro del rango de tiempo y dentro del polígono solicitados. Si se consulta más de un tiempo (o *raster*) a la vez, éstos son agregados en uno sólo utilizando la operación de agregación indicada. Los formatos de salida soportados dependen de los *drivers* GDAL instalados con PostGIS. Ejemplos de formatos GDAL: NetCDF, GeoTIFF, HDF5, entre otros.

getDataFromPoint(dataset_name, variable_name, point, start_time = None, end_time = None) : data_array

Este método recupera los datos de un punto en específico para una variable y *dataset* dados. Requiere como argumentos el nombre del *dataset*, nombre de la variable, y punto geoespacial, con los argumentos opcionales de tiempo de inicio y fin. Retorna un arreglo con todos los valores de la variable en el punto dado, para toda la cobertura temporal del *dataset*. En caso de especificar un tiempo inicial o final, se retornarán sólo los valores que cumplan con dichas restricciones.

10 Herramientas implementadas

Paralelo al desarrollo de la plataforma de software, se construyeron distintas herramientas para auxiliar en los procesamientos de datos aplicados comúnmente en el grupo HIH. Estas herramientas se implementaron en forma de scripts en Python, debido principalmente a su agilidad en el desarrollo, módulos científicos disponibles (incluyendo netCDF4), y su presencia en los *clusters* HCC de la UNL.

Entre las funciones realizadas por estos scripts se encuentran la extracción de regiones de *datasets*, la visualización gráfica de variables, el cambio de resolución de mallas y la interpolación de datos puntuales. Enseguida se presenta una tabla (Tabla 3) resumiendo los scripts desarrollados más representativos, su función y los datos procesados.

Tabla 3. Lista de los scripts desarrollados para el grupo HIH

Nombre del script	Función	Datos de entrada (Formato)	Datos de salida (Formato)
plotCF.py	Muestra un mapa de contorno en el área comprendida por el archivo NetCDF de entrada, para una variable elegida. Opcionalmente para un área rectangular y rango de fechas especificado	Archivo / *dataset* en malla (NetCDF)	Visualización de datos (Mapa de contorno)

dumpTRMM.py	Vuelca todos los datos del *dataset* TRMM en archivos CSV, para una variable elegida	*dataset* TRMM (NetCDF)	Archivo orientado a filas, con un punto y momento por fila. (CSV)
TRMMtoDaily.py	Agrega los datos TRMM de cada 3 horas a diario, para una variable elegida	*dataset* TRMM (NetCDF)	Archivo orientado a filas, con un punto y momento por fila. (CSV)
dumpCPC.py	Vuelca todos los datos del *dataset* CPC en archivos CSV, para una variable elegida	*dataset* CPC (NetCDF)	Archivo orientado a filas, con un punto y momento por fila. (CSV)
extractRectData.py	Segmenta los datos para un área geográfica rectangular y un rango de fechas dado, de una lista de archivos del *dataset* IMD	Datos de lluvia y número de pluviómetros del *dataset* IMD en archivos anuales (Texto)	Archivo orientado a filas, con un punto y momento por fila. (CSV)
extractRectData_2.py	Segmenta los datos para un área geográfica rectangular y un rango	Datos de temperatura máxima, mínima y	Archivo orientado a filas, con un punto y momento por fila.

	de fechas dado, de una lista de archivos del *dataset* IMD	promedio del *dataset* IMD en archivos anuales (Texto)	(CSV)
colsWiseToMonthly.py	Transforma un archivo CSV orientado a columnas diario a mensual	Archivo orientado a columnas, con puntos por columnas y momentos por fila, registros diarios (CSV)	Archivo orientado a columnas, con puntos por columnas y momentos por fila, registros mensuales (CSV)
getDataByCoord.py	Segmenta los datos para una lista de puntos dada	Archivo orientado a columnas, con puntos por columnas y momentos por fila (CSV)	Archivo orientado a columnas, con puntos por columnas y momentos por fila, segmentado con los puntos elegidos (CSV)
resample.py	Remuestrea o cambia la resolución de los datos a una más fina, de la cual la original sea un múltiplo	Archivo orientado a columnas, con puntos por columnas y	Archivo orientado a columnas, con puntos por columnas y momentos por fila,

	(Ej. de 0.5° a 0.25°)	momentos por fila (CSV)	con resolución cambiada (CSV)
swatInt.py	Interpola datos de SWAT organizados en un archivo por punto, usando substitución del punto más cercano, la interpolación se realiza de los puntos reales a una lista de puntos solicitada, para una variable elegida	Archivos de datos SWAT (Texto)	Archivos con las nuevos puntos solicitados, una por archivo (Texto)

Estas herramientas fueron desarrolladas para funcionar con *datasets* específicos, y otras para trabajar con datos que cumplieran con cierta estructura y formato. Algunas fungen como pasos intermedios en una secuencia de procesamiento mayor, otras como herramientas de visualización exploratoria.

11 Discusiones y Conclusiones

La plataforma de software desarrollada “Dataplugin” es una posible solución para lidiar con la heterogeneidad de datos climáticos en malla mediante la integración de éstos en un almacén. Su construcción representa el primer paso en un proceso de actualización de características y optimización que promueva la elaboración de herramientas más sofisticadas, para la explotación de los datos integrados en el almacén, permitiendo sentar las bases de una metodología más amplia para la construcción de nuevos clientes y *plugins*. Dataplugin permite solucionar en cierta medida los impedimentos y problemas presentados a lo largo de este documento.

Por medio de clientes de software desarrollados para la plataforma, usuarios interesados pueden acceder a los datos climáticos almacenados y procesarlos, visualizarlos, segmentarlos, o exportarlos a algún formato de archivo. La utilización de estos clientes se puede dar sin la necesidad de adquirir conocimientos sobre los formatos de archivo originales y sus herramientas, incluso sin requerir la adquisición de recursos computacionales importantes en caso de que se acceda a la plataforma a través de un servidor remoto. Desde el punto de vista del desarrollador, éste puede seguir la documentación referente a la construcción e integración de un *plugin* al importar un nuevo *dataset*, pudiendo preparar los datos para ser utilizados por los clientes existentes. Del mismo modo, un desarrollador de un nuevo cliente puede enfocarse en el esquema de datos del almacén para la funcionalidad, dejando que otros se encarguen de importar y adaptar nuevos *datasets* a dicho esquema mediante nuevos *plugins.*

El usuario interesado en los datos climáticos no necesitará enfrentarse más a archivos de datos complejos y heterogéneos, tanto en estructura, como en formato. De eso se encargarán los desarrolladores que importen los datos a la plataforma. Con el desarrollo de clientes de software se tiene la posibilidad de agregar y exportar los datos almacenados en un formato que

le resulte sencillo de manejar al usuario, si así lo requiere. Se vislumbra que en una futura actualización de la plataforma se implemente también la consulta de múltiples *datasets* y variables del almacén, así como procesos de interpolación y transformación que permitan la consulta y procesamiento de mallas con diferentes resoluciones espaciotemporales, variables y unidades. Esto resulta en una integración de datos de distintas fuentes heterogéneas para la obtención de información valiosa para el usuario.

Un inconveniente con la forma en la que se desarrolló la arquitectura es el manejo de *datasets* de gran tamaño, lo que se traduce en requerimientos de almacenamiento considerables y posible deterioro en el rendimiento debido a la centralización de la plataforma en un solo nodo. Por ello, se recomienda como trabajo futuro explorar medios de almacenamiento complementarios a los DBMSs, como archivos binarios comprimidos o incluso recuperación de datos sobre demanda y de manera remota sobre las fuentes originales mediante cosechado de metadatos.

Respecto al objetivo principal de éste trabajo se considera que fue cubierto por medio del desarrollo de Dataplugin, los servicios mencionados son cubiertos por diferentes componentes de la arquitectura. El almacenamiento se logra por medio del Almacén de Datos y los *plugins* desarrollados, mientras que la segmentación, recuperación y visualización es un trabajo en conjunto entre el API de acceso del Almacén de Datos y los clientes de software desarrollados (Véase capítulo 5).

El primer objetivo específico consiste en “Desarrollar un almacén de datos donde sea posible almacenar y recuperar cualquier conjunto de datos climáticos en malla utilizado por la UNL, sin importar su estructura o formato”. Este objetivo se cumplió con el desarrollo de Dataplugin, específicamente con el componente del Almacén de Datos y los *plugins* construidos. El segundo objetivo específico consiste en “Estandarizar el acceso al almacén de datos mediante

el desarrollo de un API para consultar segmentos específicos de la información almacenada". Este objetivo se logró con la construcción del API de acceso mencionado durante el capítulo 5 y descrito más a fondo en el capítulo 9: Descripción del API, donde se indican los métodos implementados para el acceso a los datos y metadatos de los *datasets* en malla.

Al principio de este trabajo se plantearon también preguntas de investigación. En la primera de ellas se buscaba conocer las soluciones existentes diseñadas para integrar datos climáticos en malla. En la revisión de literatura realizada no se logró encontrar una solución de integración que fuera exactamente igual a la propuesta, sin embargo, se encontraron tecnologías que pueden ayudar a la integración de datos, además de trabajos similares.

Respecto a las tecnologías se encontraron a DBMSs orientados al almacenamiento de arreglos o *rasters* como Rasdaman y PostgreSQL, por medio de su extensión PostGIS y el tipo de dato *raster*. También se encontró GDAL, una librería traductora codificada en C++ para el manejo de archivos *raster* en múltiples formatos geoespaciales, tales como NetCDF, HDF, GeoTIFF, etc. (GDAL/OGR contributors, 2018).

Las siguientes interrogantes planteadas buscaban conocer los DBMSs existentes para el almacenamiento de *datasets* en malla, la opción más factible y el esquema de datos más adecuado. Se puede concluir que, después de realizar una revisión de DBMSs (Véase capítulo 7) e importar un *dataset* a la plataforma mediante la construcción de un *plugin* (Véase capítulo 8), el DBMS más adecuado para almacenar datos en malla en este trabajo de investigación es PostgreSQL, mediante el tipo de dato *raster* de PostGIS. Las opciones consideradas fueron PostgreSQL, MongoDB y Rasdaman, pero debido a ciertas limitantes de cada uno se tuvieron que descartar. Claro que esta elección depende de las características del *dataset*, las consultas aplicadas, requerimientos, y el esquema de datos utilizado. Para este trabajo PostgreSQL + PostGIS resultó adecuado debido a que se buscaba un DBMS capaz de almacenar datos en

malla y que además soportara y entendiera los tipos de dato y operaciones geoespaciales. El esquema de datos elegido se explica en el capítulo 8.1.

Durante el proceso de investigación de este trabajo se encontraron varios problemas, tanto con los datos como en general. A continuación se mencionan los más importantes:

Problemas con la instalación de paquetes en la arquitectura *ppc64le* (Arquitectura de IBM Power). Fue una tarea ardua, los paquetes de software como MongoDB y Python 3, que normalmente se encuentran en binarios listos para descargar en muchas distribuciones de Linux, no se encontraban de la misma manera en el servidor IBM Power donde se instaló la plataforma. Se tuvo que recurrir a recursos alternativos.

Los problemas inherentes a las características de los datos climáticos. Formatos de archivo heterogéneos, de gran tamaño y/o grandes cantidades de archivos, y crecimiento rápido. Específico al *dataset* trabajado es la distribución de los datos, donde las mediciones son generadas una cada tres horas y distribuidas en un archivo NetCDF nuevo. Para poder extraer datos de este *dataset* es necesario acceder a múltiples archivos en una sola consulta, la mayoría del tiempo.

Respecto a la obtención del *dataset*, se tuvo problemas con su descarga. Se tuvo que implementar un script en Bash que permitiera la descarga automatizada de los archivos TRMM pertenecientes a un periodo solicitado, para evitar así descargar los archivos de manera manual. El script se basaba en repetidas llamadas al sitio que aloja TRMM (NASA, 2018b) y en el procesamiento de HTML para descargar los archivos de cada página.

Se incurrieron también en varios problemas durante el desarrollo de la arquitectura y los *plugins*: problemas con la ejecución de procesos externos debido al uso indebido de su API, dificultad en la carga dinámica de *plugins* y clases, problemas con el uso y transferencia de archivos dentro del fichero del *plugin*, agotamiento del tiempo máximo de una petición Web en

GlassFish, dificultad al desplegar la plataforma en el servidor GlassFish, problemas de portabilidad entre el script original y su ejecución mediante un *plugin*, y flujo de trabajo y depuración más complicada que en sistemas de un solo lenguaje y paquete.

Además se tuvieron problemas con la carga de los datos al *raster* de PostGIS por medio de su herramienta (raster2pgsql). También hubo ciertos inconvenientes en la integración de los drivers de GDAL con PostGIS. Estos drivers son utilizados para leer y escribir datos a formatos de archivo *raster*, por ejemplo NetCDF, HDF5, GeoTIFF, e incluso PNG y JPG. El no poder integrar estos drivers a PostGIS equivale a no poder exportar los datos a dichos formatos. GDAL es la misma librería utilizada por Rasdaman para exportar datos a formatos de archivo *raster*.

En el desarrollo de la arquitectura quedaron algunos cabos sueltos referente a la optimización de la arquitectura. Estos son la ejecución interna de los *plugins* mediante el *Plugin Executor*, donde éste podría utilizar un *pool* de hilos u otro API de ejecución más sofisticado. Con respecto del acceso a las bases de datos del almacén (PostgreSQL), sería recomendable que se realizara mediante un *pool* de conexiones y no de la manera actual, que es manteniendo una conexión siempre abierta. También es necesario implementar métodos para la descarga (eliminación del registro) de *plugins* registrados en el *Plugin Manager*, y no sólo el registro y carga al arranque de la aplicación. También sería prudente implementar una funcionalidad que permita registrar el *plugin* sobre demanda, por ejemplo, al agregar el archivo .plugin al directorio de *plugins* de la plataforma. El método para registrar un *plugin* ya existe, sólo falta adjuntarlo al evento correcto (Ej. Directorio modificado). Del mismo modo es importante implementar mecanismos de respuesta asíncronos al momento de realizar peticiones HTTP al API de acceso. Debido a la posible larga duración para que un procesamiento se realice, es probable que la petición se cancele debido al tiempo transcurrido sin respuesta. La exportación a diferentes formatos de archivo es posible mediante la librería GDAL, pero ésta puede

limitarse a un número reducido de formatos, dependiendo de la instalación e integración con PostGIS. Aunque el formato que se requiere sea soportado, será necesario implementar subrutinas en PostgreSQL que exporten el archivo correctamente con la estructura que se requiera, en caso de solicitar más de una malla (o tiempo). Actualmente el *plugin* desarrollado no soporta la actualización de datos, esto quiere decir que cada vez que se ejecute un *plugin* se estará reemplazando los datos almacenados con el resultado de la nueva ejecución. Los procesos de importación tampoco son transaccionalmente seguros en caso de múltiples ejecuciones del mismo *plugin* a la misma base de datos.

En la plataforma convergen distintas tecnologías, y dependiendo del rol involucrado, los conocimientos necesarios son diferentes. Debido al diseño de la plataforma, ésta podría requerir experiencia sobre una tecnología no mencionada a continuación, pero para el enfoque de este trabajo de investigación, los conocimientos necesarios hasta el momento son los siguientes:

El código base de la arquitectura es Java 8, por lo que el desarrollador de *plugins* debe de ser experto en este lenguaje. Debe tener también experiencia adicional en la ejecución de procesos, manejo de archivos, patrones de diseño, cargadores de clases, programación funcional, servicios *REST*, uso de expresiones regulares, procesamiento de texto, NetCDF, HDF5, archivos binarios, y conocimiento acerca de la librería dataplugin.api..

También debe ser capaz de crear documentos en lenguaje YAML, conocimiento avanzado en el lenguaje de script Python 3, conocimiento básico en el lenguaje Bash o shell de Linux, dominio del lenguaje JSON y la creación de este tipo de documentos (programática o manualmente).

Además de contar con conocimientos en Python, es necesario también dominar ciertas librerías y módulos científicos para trabajar con datos en malla y archivos NetCDF. Los módulos

requeridos son: netCDF4, para manipular archivos del mismo formato, matplotlib.pyplot, para elaborar gráficas para visualización de datos, cartopy, para elaborar mapas de contorno y otro tipo de mapas geográficos para visualizar datos, y numpy, para manipular arreglos numéricos eficientemente.

Para el caso en el que el desarrollador deba construir un cliente de software, éste deberá dominar tecnologías Web básicas como Javascript, HTML5, CSS3, librerías y *frameworks* como JQuery, Angular, Leaflet, Mapbox, Android SDK, Swift 4, y cualquier otra tecnología que sea requerida para desarrollar un cliente dado, ya sea para extracción, procesamiento o visualización de datos.

El desarrollador, así como el diseñador de base de datos y el administrador de la plataforma, deben también contar con conocimiento sólido sobre PostgreSQL: creación de bases de datos, optimización de recursos, manejo de transacciones, procesos de carga masiva, creación de índices, optimización de consultas, y por supuesto SQL. Además es indispensable contar con conocimientos de la extensión PostGIS y sus tipos de datos geométricos, como *Point* y *Polygon*, pero también el tipo de dato *Raster*, junto con las funciones disponibles para su manipulación. También es necesario saber operar la herramienta/comando raster2pgsql distribuido con PostGIS. El administrador de la plataforma debe dominar también la creación y administración de bases de datos en MongoDB, manejo de credenciales para control de acceso y lenguaje de consulta. El administrador de la plataforma debe ser experto en el manejo del servidor GlassFish o algún otro contenedor para *JavaEE* como Tomcat. Experto en administración de servidores Linux, administración de DBMSs como PostgreSQL y MongoDB, y administración de la plataforma Dataplugin.

Además de los conocimientos en programación, bases de datos, librerías y formatos de archivo, es también indispensable para el desarrollador tener conocimiento básico de temas de

climatología como: qué es un *raster* o malla, resolución espaciotemporal, variables climáticas y fuentes de datos principales, además de operaciones básicas como: interpolación de mallas, remuestreo y agregación temporal. También es importante conocer, sino es que dominar, librerías como GDAL, para manipular *rasters* en múltiples formatos, y utilerías como nco, ncdump, ncview, grads, y h5dump para el manejo exploratorio, visualización básica y operaciones con archivos NetCDF y HDF5.

Los perfiles mencionados son conocimientos basados en el *plugin* desarrollado TRMMDumper. Dependiendo del *plugin* a construir, los formatos de archivo a utilizar, los lenguajes de programación y los DBMSs utilizados, son los conocimientos que se requerirán tener. El conocimiento mínimo requerido para un desarrollador es Java 8, YAML, y expresiones regulares. Python 3 es altamente recomendado. Además, se recomienda tener dominio de los archivos en malla NetCDF y HDF5, y adherirse en lo posible al DBMS y el esquema de datos utilizado al almacenar *datasets* en malla. La razón es debido a que PostgreSQL es una base robusta y bien probada, utiliza el lenguaje SQL, un lenguaje de consulta fácil de utilizar y es fácil encontrar desarrolladores con competencias en SQL.

Mientras se almacenen los *datasets* en malla con el mismo esquema de datos propuesto, será posible obtener los datos de manera automática mediante el API o cualquier cliente que haga uso de ese esquema. En el futuro se plantea utilizar los metadatos del *dataset* para “mapear” un esquema general de malla con cualquier esquema implementado por algún DBMS, pudiendo recuperar de manera transparente datos de múltiples DBMSs y esquemas mediante un API orientado a mallas.

El proceso para la carga de un *dataset* a la arquitectura es manual. Tiene que intervenir en gran parte el desarrollador para poder construir un *plugin* que se adapte a las características de un *dataset* en especial. No obstante, una vez almacenados los datos es posible recuperarlos

programáticamente mediante el API disponible y algún cliente construido para procesar dicho *dataset*. El proceso de desarrollo del *plugin* y el cliente es un proceso vital que no puede ser removido ni automatizado, sin embargo, se espera que en el futuro se pueda hacer uso de los metadatos tanto del *plugin* como del *dataset* para automatizar en cierto grado la búsqueda y ejecución de un *plugin*. Por ejemplo, para un nuevo *dataset* en HDF5, determinar automáticamente, mediante metadatos, si es posible cargar los datos utilizando *plugins* HDF5 existentes o es necesario construir uno desde cero.

La plataforma desarrollada no fue construida con el propósito de eliminar todos los problemas de heterogeneidad de datos climáticos. No obstante, existen casos de uso para los cuales es más adecuado utilizar la plataforma que en otros. Es posible almacenar en la plataforma cualquier tipo de *dataset*, implementando el *plugin* de importación y el esquema correspondiente. Dicho esquema puede ser incluso implementado en un DBMS diferente a PostgreSQL, y el *plugin* puede utilizar código Java completamente o algún otro lenguaje diferente a Python. Sin embargo, el API de acceso y el catálogo de metadatos implementados en este trabajo están orientados a datos climáticos en malla o *raster*. Los casos de uso adecuados para utilizar esta plataforma son:

El almacenamiento de *datasets* en malla o *raster* en un DBMS, donde se tenga mediciones de variables del tipo XY[Z]T, donde X y Y representan longitud y latitud, Z opcionalmente altura, y T tiempo, para una o más variables.

Aplicaciones en las que sea necesario procesar mallas de datos para extraer subconjuntos tanto en el espacio como en el tiempo, donde se requiera realizar agregaciones en tiempo, cambio de resolución espacial, exportación a diferentes formatos, visualización o algún otro tipo de procesamiento, para *datasets* almacenados en cualquier formato de archivo.

Casos en los que sea necesario proveer un medio de acceso uniforme, y con capacidades de consulta y otras características de un DBMS, a los *datasets* en malla almacenados en distintos formatos de archivo que no cuenten con esas capacidades, o cuyo manejo se vuelve complicado debido a la cantidad de archivos generados.

También cuando se requiera añadir algún cliente de software que consuma datos existentes en el almacén de datos, o cuando se desee utilizar algún cliente existente en la plataforma para datos en malla aún no almacenados.

Cuando se desee contar con un almacén de datos climáticos centralizado, con documentación bien definida para el proceso de importación de datos al almacén, y con soporte de distintas fuentes de datos y DBMSs. Con un API de acceso adaptado para recuperar datos climáticos en malla y sus metadatos, que permita la construcción de clientes de software para consumir la información almacenada.

Los casos de uso en los cuáles esta plataforma no es adecuada son todos aquellos que no caen en los casos anteriormente mencionados. A continuación se hace mención de algunos de ellos:

Cuando se requiera almacenar *datasets* en un DBMS cuya estructura sea distinta a la de un *raster* o malla. Por ejemplo, datos climáticos puntuales, donde cada punto representa una estación que registra mediciones de múltiples variables a través del tiempo. Las mallas en esencia también se componen por datos puntuales, pero éstos se encuentran distribuidos uniformemente en todo el espacio mediante una resolución constante. Este no es el caso de los datos puntuales, que pueden estar separados uno del otro en distancias variables, y no se organizan espacialmente de ninguna manera en particular. Aunque es posible almacenar nuevos tipos de datos diferentes a las mallas, el API de acceso actual de la plataforma está orientado a recuperar *datasets* en malla, por lo que será necesario implementar otro API que se

especialice en el tipo de *dataset* que se desee almacenar. En un futuro podría ser posible agrupar ambos APIs mediante un API más general que acceda a ambos tipos de *datasets*, creando así una familia de APIs para el acceso a los datos en vez de uno sólo orientado a mallas.

Otro caso es cuando se busque rendimiento sobre funcionalidad. Esto quiere decir que se prefiere que el tiempo de respuesta de una consulta o un procesamiento dado sea lo más rápido posible, sacrificando la funcionalidad o flexibilidad que la plataforma hubiera proveído a los datos almacenados. Por ejemplo, la integración del *dataset* con clientes de visualización existentes o simplemente el uso de SQL sobre éste. Aunque la plataforma tiene sus ventajas, también tiene sus limitantes. Aunque el tiempo de respuesta es pequeño para ciertas consultas, puede que existan procesos analíticos que tarden más tiempo en la plataforma que directamente de archivos binarios. La plataforma sacrifica una parte de rendimiento por mayor soporte funcional, tanto con operaciones espaciales en lenguaje SQL, como con la fácil integración con otras tecnologías o con clientes existentes. Si se busca el rendimiento sobre todo eso, es mejor almacenar el *dataset* en algún tipo de archivo binario con compresión como NetCDF o HDF5, y utilizar las librerías existentes para codificar el procesamiento que se desea realizar. No obstante, todo esto depende también del formato de salida de los datos y el costo de codificar dicho procesamiento.

Otro en el que no es recomendable utilizar la plataforma es cuando se busque explotar el paralelismo de un *cluster* o ambiente distribuido para procesar datos. Este caso también está relacionado con el anterior en el sentido de buscar rendimiento por medio del cómputo paralelo. Debido a que la arquitectura está desarrollada para instalarse en un sólo nodo, es imposible utilizarla en ambientes distribuidos como en un *cluster* o en la nube. Modificar este comportamiento requeriría hacer una reingeniería a la arquitectura de software del almacén, desde la ejecución de *plugins* y clientes distribuidos hasta la fragmentación de las bases de

datos debajo. Además, normalmente los recursos de DBMSs y servidores de aplicación son más costosos en un servicio de nube que un simple sistema de archivos, por lo que el uso de la plataforma en un ambiente como la nube tendría un alto costo, en caso de existir la posibilidad.

Un caso más es cuando se deseen almacenar mallas de muy alta resolución (gran tamaño) en la plataforma. Tomando como ejemplo el *dataset* TRMM utilizado, su malla se puede considerar relativamente pequeña, cubriendo el área desde -180° a 180° longitud y -60° a 60° latitud, a una resolución de 0.25°, lo que equivale a un tamaño de 1,440 x 480. Esta malla pesa no más de 3 MB sin comprimir. Para este *dataset*, no es problema almacenar la malla en el esquema de datos propuesto: el número de tejas por malla y el total de registros en la tabla se mantiene manejable, pero no se puede decir lo mismo para mallas de mayor resolución. Por ejemplo, el *dataset* SMAP (Kim et al., 2016) ofrece una malla de humedad de suelo con una resolución espacial de 3 x 3 km, lo que equivaldría a aproximadamente .024°. Cubriendo un área de -180° a 180° longitud y -85.044° a 85.044° latitud, la malla resultante tiene un tamaño de 15,000 x 7,087, y un peso de aproximadamente 400 MB. Esta malla ya no resulta tan manejable como la otra, especialmente tomando en consideración que todos esos datos pertenecen a un sólo tiempo de medición. Es conveniente entonces considerar si será adecuado el mismo esquema de datos utilizado anteriormente para este *dataset*. Una opción podría ser aumentar el tamaño de las tejas. Todo depende del *dataset*, las consultas a realizar y el rendimiento del DBMS. En realidad no se ha probado hasta el momento el límite del esquema de datos propuesto ni la cantidad de registros que puede soportar, pero se espera que su rendimiento se degrade conforme aumente la resolución espaciotemporal de las mallas.

Tampoco es adecuado utilizar la plataforma cuando no se busque nada más allá de un medio de almacenamiento. Cuando no se busca explotar las ventajas de un DBMS como poder y expresividad del lenguaje de consulta, tareas de mantenimiento y respaldo, control de acceso, comunidad y soporte detrás de él, alta disponibilidad, arquitectura robusta, e integración con

otras tecnologías y soluciones (en el caso de SQL). Si este fuera el caso sería simplemente mejor almacenar los *datasets* en el formato de archivo original, directamente en el sistema de archivos. Si el problema es el espacio en disco se puede aplicar compresión.

Por último está el caso de procesamiento *ad-hoc*: cuando la ejecución de un proceso es única y no se volverá a repetir. No tiene sentido pasar por el ciclo de desarrollo de un *plugin*, la importación de un *dataset* completo y la creación de un cliente de software sólo para realizar un proceso que se aplicará una vez: para los resultados de un estudio o para responder una pregunta, por ejemplo. En esos casos es mejor sólo ejecutar las instrucciones necesarias o implementar un script o dos para realizar el procesamiento. Sin embargo, si el procesamiento a realizar a los datos se ejecutará más de una vez, y la repetición del proceso basada en simples scripts se vuelve poco práctica, entonces convendría pasar por los procesos de importación y exportación de datos a la plataforma. De esta manera, construyendo el cliente adecuado, será posible repetir el procesamiento múltiples veces sin necesidad de recurrir al procesamiento *ad-hoc* creado. Inclusive puede ser posible que alguna funcionalidad similar ya exista en un cliente de software y sólo sea necesario cargar los datos, o que ya exista un *plugin* capaz de cargar los datos y sólo sea necesario implementar un cliente que los procese. De cualquier modo, gracias al diseño de los *plugins*, es posible portar los scripts creados en la solución *ad-hoc* para ser ejecutados por el *plugin* en el proceso de importación, si así se desea.

Tomando en consideración los casos de uso recomendados, se considera que la arquitectura de software desarrollada será benéfica para aquellos usuarios e investigadores que se enfrentan con los problemas ya mencionados al tratar con datos en malla. Una vez entendiendo los procesos involucrados en la plataforma y la documentación asociada, éstos podrán ser replicados con cada *dataset* y fuente de datos que se desee agregar. Se espera que el esfuerzo requerido durante los procesos de importación de *datasets* se vea recompensado al

poder aplicar análisis y procesamientos en los datos almacenados en cuanto éstos estén disponibles, habiendo construido los clientes pertinentes con anterioridad, por supuesto.

Si ya se tiene la plataforma instalada en un servidor local o remoto, junto con clientes de software para proveer funcionalidades comúnmente requeridas, uno o varios grupos de investigación pueden reutilizar rutinariamente los clientes para satisfacer sus necesidades de información climática. En caso de requerir una nueva característica, se puede colaborar con un desarrollador que agregue dicha característica a algún cliente existente o cree uno nuevo. A su vez, si se requiere añadir una nueva fuente de datos para trabajar con ella, un desarrollador puede implementar un *plugin* siguiendo la documentación relacionada para que los datos importados se adapten al esquema del almacén. El beneficio de la plataforma se puede ver a largo plazo en base a las funcionalidades de los clientes y el número de *datasets* almacenados. Por ejemplo, en el caso de requerir implementar 18 funcionalidades diferentes para un *dataset* en un formato X, se requeriría desarrollar dichas funcionalidades una vez, pero si se requiere trabajar además con tres *datasets* en formato Y, Z y F, se tendría también que desarrollar las 18 funcionalidades tres veces más, una para cada formato. Incluso es probable que se requieran implementar por cada *dataset* nuevo aunque éste se encuentre en un formato ya soportado, en caso de que su estructura varíe y esto no sea contemplado. Esto no sería un problema con la plataforma, debido a que en el peor de los casos se requeriría implementar un *plugin* por cada *dataset* nuevo, más las 18 funcionalidades que soporten el esquema de datos del almacén una sola vez.

También cabe la posibilidad de que en un futuro se cree una especie de repositorio público de *plugins*, para que diferentes grupos que hayan adoptado la arquitectura los puedan integrar a ésta, y lo mismo podría aplicar también para los clientes. Claro que para ello sería necesario que la comunidad tenga una respuesta positiva a la arquitectura y que su adopción se expanda lo suficiente, además de contar con mecanismos de control de versiones, comunicación y

distribución de paquetes de software sofisticados, e incluso estándares para el desarrollo de *plugins* y clientes. El esfuerzo colectivo de la comunidad podría entonces permitir un inicio mucho más acelerado para los grupos o investigadores que vayan comenzando con el uso de la plataforma.

11.1 Trabajo futuro

Se consideran las siguientes actividades para una futura versión de la plataforma:

1) Poder utilizar una fuente de datos indiferentemente entre datos almacenados en un DBMS, en archivos binarios (como NetCDF) o de manera remota (fuente de terceros), mediante uso de APIs y cosechado de metadatos.
2) Mejorar la especificación y el uso de metadatos para automatizar procesos como búsqueda de un *plugin* apto para importar un *dataset* dado, o consulta uniforme de datos *raster* de múltiples fuentes dentro de la plataforma (distintos DBMSs).
3) Implementar una familia de APIs de acceso para distintos tipos de *datasets* climáticos.
4) Acceder de forma homogénea a variables de múltiples *datasets* en una misma consulta.
5) Implementar procesos de interpolación que permitan la consulta de mallas con diferentes resoluciones espaciotemporales.
6) Estandarizar un esquema de datos para cada tipo de *dataset* (malla, datos puntuales, etc.).
7) Mejorar el control y la seguridad de la ejecución de los *plugins*.

12 Referencias

"data.gov". (2018a). Glossary of Terms – Data.gov. Recuperado el 15 de Marzo del 2018, de https://www.data.gov/glossary

"data.gov". (2018b). Impact – Data.gov. Recuperado el 16 de Mayo del 2018, de https://www.data.gov/impact/

"datos.gob.mx". (2018). Datos Abiertos de México. Recuperado el 16 de Mayo del 2018, de https://datos.gob.mx/

Ahrens, C. D. (2007). *Meteorology today: An introduction to weather, climate, and the environment*. (8a ed.). Belmont, California, USA: Thomson Brookes/Cole. https://doi.org/10.1017/CBO9781107415324.004

Chadwick, R. (2014). Citizen Weather Observer Program. Recuperado de http://wxqa.com/

Daley, R. (1993). *Atmospheric Data Analysis* (1a ed.). New York, USA: Cambridge University Press. https://doi.org/10.1016/0160-9327(91)90140-7

Earth Server. (2018a). Recuperado de http://earthserver.eu/

ESRI, Inc. (2016). Environmental Systems Research Institute, Inc. What is raster data? Help | ArcGIS for Desktop. Recuperado el 15 de Marzo del 2018, de https://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/what-is-raster-data.htm

GDAL/OGR contributors (2018). GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation. Recuperado de http://gdal.org

Hartmann, D. L. (2015). *Global physical climatology: Second Edition* (2a ed.). University of Washington, Seattle, USA: Elsevier. https://doi.org/10.1016/C2009-0-00030-0

Hdafa, M., Zouhairi, Y., Lemoudden, M., y Ziyati, E. (2016). An Approach for Meteorological Data Integration and Stream Processing. En *PROCEEDINGS OF 2016 THIRD INTERNATIONAL CONFERENCE ON SYSTEMS OF COLLABORATION (SYSCO)* (pp. P96–P100).

Huffman, G. (2016). TRMM (TMPA-RT) Near Real-Time Precipitation L3 3 hour 0.25 degree x 0.25 degree V7. Greenbelt, MD. Goddard Earth Sciences Data and Information Services Center (GES DISC), Recuperado el 27 de Septiembre del 2017, de https://disc.gsfc.nasa.gov/datacollection/TRMM_3B42RT_7.html

India Meteorological Department. (2015). IMD Mandate. Recuperado el 13 de Marzo del 2018, de http://www.imd.gov.in/pages/about_mandate.php

Kim, S., Van Zyl, J., Dunbar, R. S., Njoku, E. G., Johnson, J. T., Moghaddam, M., y Tsang, L. (2016). SMAP L2 Radar Half-Orbit 3 km EASE-Grid Soil Moisture, Version 3. Boulder, Colorado USA. NASA National Snow and Ice Data Center Distributed Active Archive Center. doi: https://doi.org/10.5067/J8SGO1E0Y9XZ. Recuperado el 2 de Mayo del 2018.

Lian, J., McGuire, M. P., y Moore, T. W. (2017). FunnelCloud: A cloud-based system for exploring tornado events. *International Journal of Digital Earth*, 10(10), 1030–1054. https://doi.org/10.1080/17538947.2017.1279235

Liang, X., Lettenmaier, D. P., Wood, E. F., y Burges, S. J. (1994). A simple hydrologically based model of land surface water and energy fluxes for general circulation models. *Journal of Geophysical Research*, 99(D7), 14415. https://doi.org/10.1029/94JD00483

Livneh, B., Bohn, T. J., Pierce, D. W., Munoz-Arriola, F., Nijssen, B., Vose, R., … Brekke, L. (2015). A spatially comprehensive, hydrometeorological data set for Mexico, the U.S., and Southern Canada 1950-2013. *Scientific Data*, 2, 1–12. https://doi.org/10.1038/sdata.2015.42

Lucid Software. (2018). Qué es un esquema de base de datos. Lucidchart. Recuperado el 9 de Abril del 2018, de https://www.lucidchart.com/pages/es/qué-es-un-esquema-de-base-de-datos

McGuire, M. P., Roberge, M. C., y Lian, J. (2016). Channeling the water data deluge: A system for flexible integration and analysis of hydrologic data. *International Journal of Digital Earth*, 9(3), 272–299. https://doi.org/10.1080/17538947.2015.1031715

MongoDB, Inc. (2018). Introduction to MongoDB. MongoDB Manual 3.6. Recuperado el 10 de Abril del 2018, de https://docs.mongodb.com/manual/introduction/

National Aeronautics and Space Administration. (2018a). TRMM Data Downloads | Precipitation Measurement Missions. Recuperado el 15 de Marzo del 2018, de https://pmm.nasa.gov/data-access/downloads/TRMM

National Aeronautics and Space Administration. (2018b). Data Calendar for Data Set Near Real-Time Precipitation L3 3 hour 0.25 degree x 0.25 degree V7. Recuperado 1 de Mayo del 2018, de https://mirador.gsfc.nasa.gov/cgi-bin/mirador/presentNavigation.pl?tree=project&*dataset*=%20Near%20Real-Time%20Precipitation%20L3%203%20hour%200.25%20degree%20x%200.25%20degree%20V7&project=TRMM&dataGroup=Gridded&version=7

National Centers for Environmental Information. (2018). About the National Centers for Environmental Information. Recuperado el 17 de Marzo del 2018, de https://www.ncei.noaa.gov/about

National Climatic Data Center. (2018a). Web Services API (version 2) Documentation | Climate Data Online. Recuperado el 17 de Marzo del 2018, de https://www.ncdc.noaa.gov/cdo-Web/webservices/v2

National Climatic Data Center. (2018b). Climate Data Online. Recuperado el 17 de Marzo del 2018, de https://www.ncdc.noaa.gov/cdo-Web/

National Climatic Data Center. (2018c). NCDC Web Services | Climate Data Online. Recuperado el 17 de Marzo del 2018, de https://www.ncdc.noaa.gov/cdo-Web/webservices

National Oceanic and Atmospheric Administration. (2018a). CPC Unified Gauge-Based Analysis of Daily Precipitation over CONUS. Recuperado el 16 de Marzo del 2018, de https://www.esrl.noaa.gov/psd/data/gridded/data.unified.daily.conus.html

National Oceanic and Atmospheric Administration. (2018b). About | NOAA Climate.gov. Recuperado el 17 de Marzo del 2018, de https://www.climate.gov/about

National Weather Service. (2016). About NOAA's National Weather Service. National Oceanic and Atmospheric Administration. Recuperado de http://www.weather.gov/about

National Weather Service. (2018). Glossary | NOAA's National Weather Service. Recuperado de http://w1.weather.gov/glossary/

OECD. (2018). Organisation for Economic Co-operation and Development. *Open Government Data in Mexico: The Way Forward* (1a ed.). OECD Digital Government Studies. París: OECD Publishing. http://dx.doi.org/10.1787/9789264297944-en

Open Knowledge International. (2018). The Open Definition | Defining Open in Open Data, Open Content and Open Knowledge. Recuperado el 9 de Marzo del 2018, de http://opendefinition.org/

PGDG. (2018a). The PostgreSQL Global Development Group. PostgreSQL: About. Recuperado el 9 de Abril del 2018, de https://www.postgresql.org/about/

PGDG. (2018b). The PostgreSQL Global Development Group. Documentation: 9.2: Index Types. Recuperado el 27 de Abril del 2018, de https://www.postgresql.org/docs/9.2/static/indexes-types.html

PGDG. (2018c). The PostgreSQL Global Development Group. Documentation: 9.2: CLUSTER. Recuperado el 27 de Abril del 2018, de https://www.postgresql.org/docs/9.1/static/sql-cluster.html

PostGIS. (2018a). Recuperado el 9 de Abril del 2018, de https://postgis.net/

PostGIS. (2018b). Chapter 9. Raster Reference. Recuperado el 25 de Abril del 2018, de http://postgis.net/docs/manual-2.2/RT_reference.html

PostGIS. (2018c). WKTRaster. Recuperado el 26 de Abril del 2018, de https://trac.osgeo.org/postgis/wiki/WKTRaster

PostGIS. (2018d). WKTRaster/SpecificationWorking01. Recuperado el 27 de Abril del 2018, de https://trac.osgeo.org/postgis/wiki/WKTRaster/SpecificationWorking01

PostGIS. (2018e). Chapter 5. Raster Data Management, Queries, and Applications. Recuperado el 30 de Abril del 2018, de http://postgis.net/docs/manual-dev/using_raster_dataman.html

Rasdaman. (2018). Rasdaman, the raster array database. Recuperado el 11 de Abril del 2018, de http://rasdaman.org/

Rohli, R. V., y Vega, A. J. (2008). *Climatology* (1a ed.). Sudbury, Massachusetts, USA: Jones and Bartlett Publishers.

Ruddiman, W. F. (2008). *Earth's climate: Past and future* (2a ed.). New York, USA: W.H. Freeman and Company.

Ruiz Corral, J. A., Díaz Padilla, G., Guzmán Ruiz, S. D., Medina García, G., y Silva Serna, M. M. (2006). *Estadísticas climatológicas básicas del estado de Baja California (Período 1961-2003). Libro Técnico Núm. 1* (1a ed.). Obregón, Sonora, México: INIFAP-CIRNO

Seoánez Calvo, M., Bellas Velasco, E., Casabella Cabana, E., Díaz Perdomo, A., López García, A., Martín Romero, M., … Vílchez Martínez, M. del C. (2001). *Tratado de climatología aplicada a la ingeniería medioambiental: Análisis climático. Uso del análisis climático en los estudios medioambientales* (1a ed.). Madrid, España: Ediciones Mundi-Prensa.

Servicio Meteorológico Nacional (2016). Estaciones Automáticas. Comisión Nacional del Agua. Recuperado de http://smn.cna.gob.mx/emas/

Servicio Meteorológico Nacional. (2018). Funciones y objetivos. Recuperado el 27 de febrero del 2018, de http://smn.cna.gob.mx/es/smn/funciones-y-objetivos

Spotless Data Ltd. (2018). Exploring data warehouses and data quality. Recuperado el 19 de Marzo del 2018, de https://spotlessdata.com/blog/exploring-data-warehouses-and-data-quality

SWAT | Soil and Water Assessment Tool. (2018). Recuperado el 16 de Marzo del 2018, de https://swat.tamu.edu/

The Apache Software Foundation. (2016). Apache Cassandra. Recuperado el 11 de Abril del 2018, de https://cassandra.apache.org/

The HDF Group. (2017). HDF5®. Recuperado el 15 de Marzo del 2018, de https://www.hdfgroup.org/solutions/hdf5/

The HDF Group. (2018). Introduction to HDF5. Recuperado el 15 de Marzo del 2018, de https://portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5

Thorne, P. W., Allan, R. J., Ashcroft, L., Brohan, P., Dunn, R. J. H., Menne, M. J., … Worley, S. J. (2017). Toward an integrated set of surface meteorological observations for climate science and applications. *Bulletin of the American Meteorological Society*, *98*(12), 2689–2702. https://doi.org/10.1175/BAMS-D-16-0165.1

Unidata. (2018). NetCDF: FAQ. Recuperado el 14 de Marzo del 2018, de https://www.unidata.ucar.edu/software/netcdf/docs/faq.html#whatisit

Weather Underground (2016). Personal Weather Station Network, Overview. The Weather Company, LLC. Recuperado de https://www.wunderground.com/weatherstation/overview.asp

World Meteorological Organization. (2018). OSCAR. Recuperado el 18 de Marzo del 2018, de https://oscar.wmo.int/surface//index.html/

13 Glosario

Humedad: Medida de la cantidad de vapor de agua en el aire (Ahrens, 2007).

Nubes: Una masa visible de pequeñas gotas de agua y/o cristales de hielo que están sobre la superficie de la Tierra (Ahrens, 2007).

Precipitación: Toda forma de agua, ya sea líquida o sólida, que cae desde las nubes y alcanza el suelo (Ahrens, 2007).

Presión del aire: La fuerza del aire sobre un área (Ahrens, 2007).

Reanálisis: Involucra la compilación y asimilación de datos observados de una amplia variedad de fuentes, como estaciones meteorológicas de la superficie, boyas, satélites, aviones y navíos, en un sólo *dataset* para cada variable analizada (Rohli y Vega, 2008).

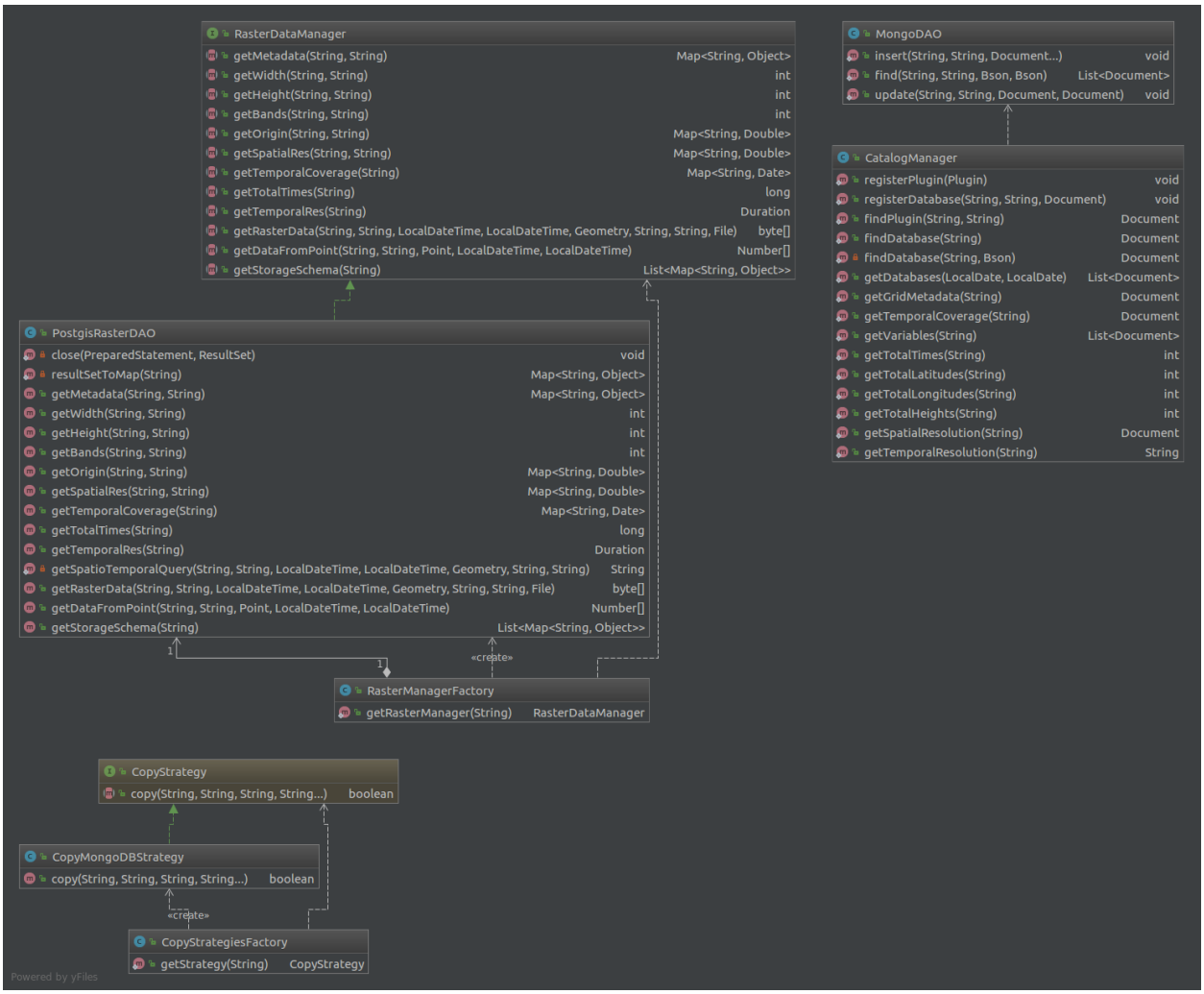
Temperatura del aire: El grado de calor o frialdad del aire (Ahrens, 2007).

Viento: El movimiento horizontal del aire (Ahrens, 2007).

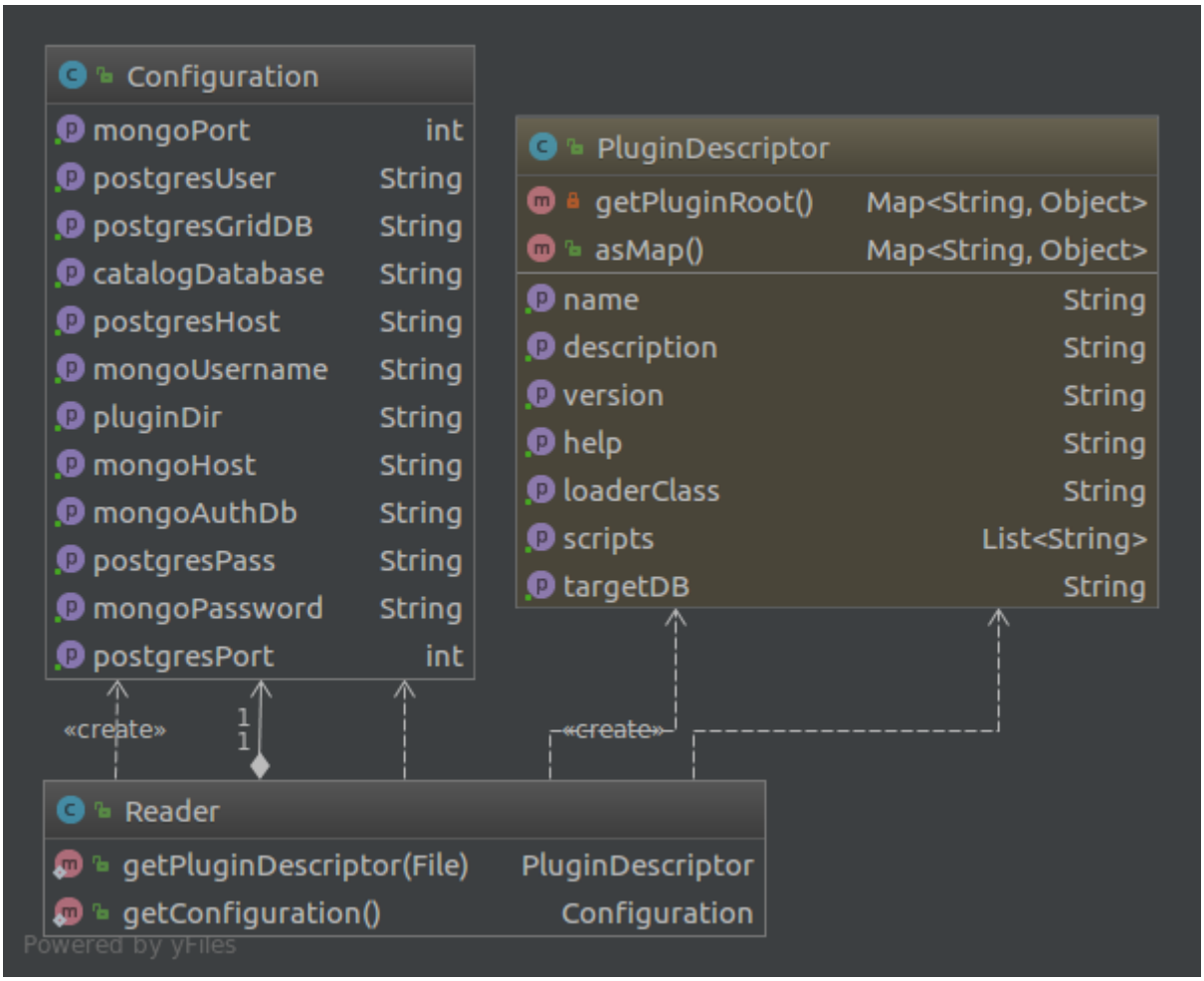
Visibilidad: La mayor distancia que uno puede ver (Ahrens, 2007).

14 Anexos

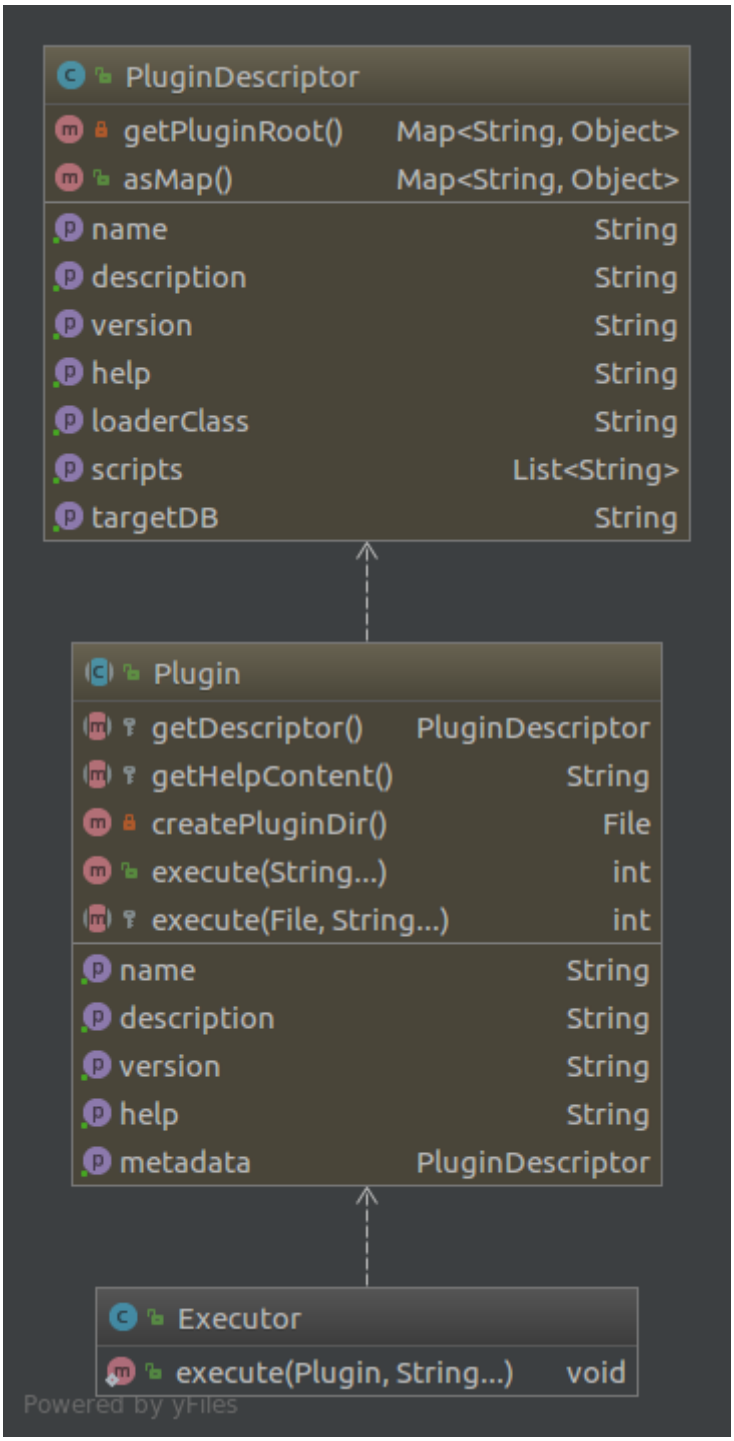
14.1 Anexo A. Diagrama de clases del paquete "*datastore*" de Dataplugin



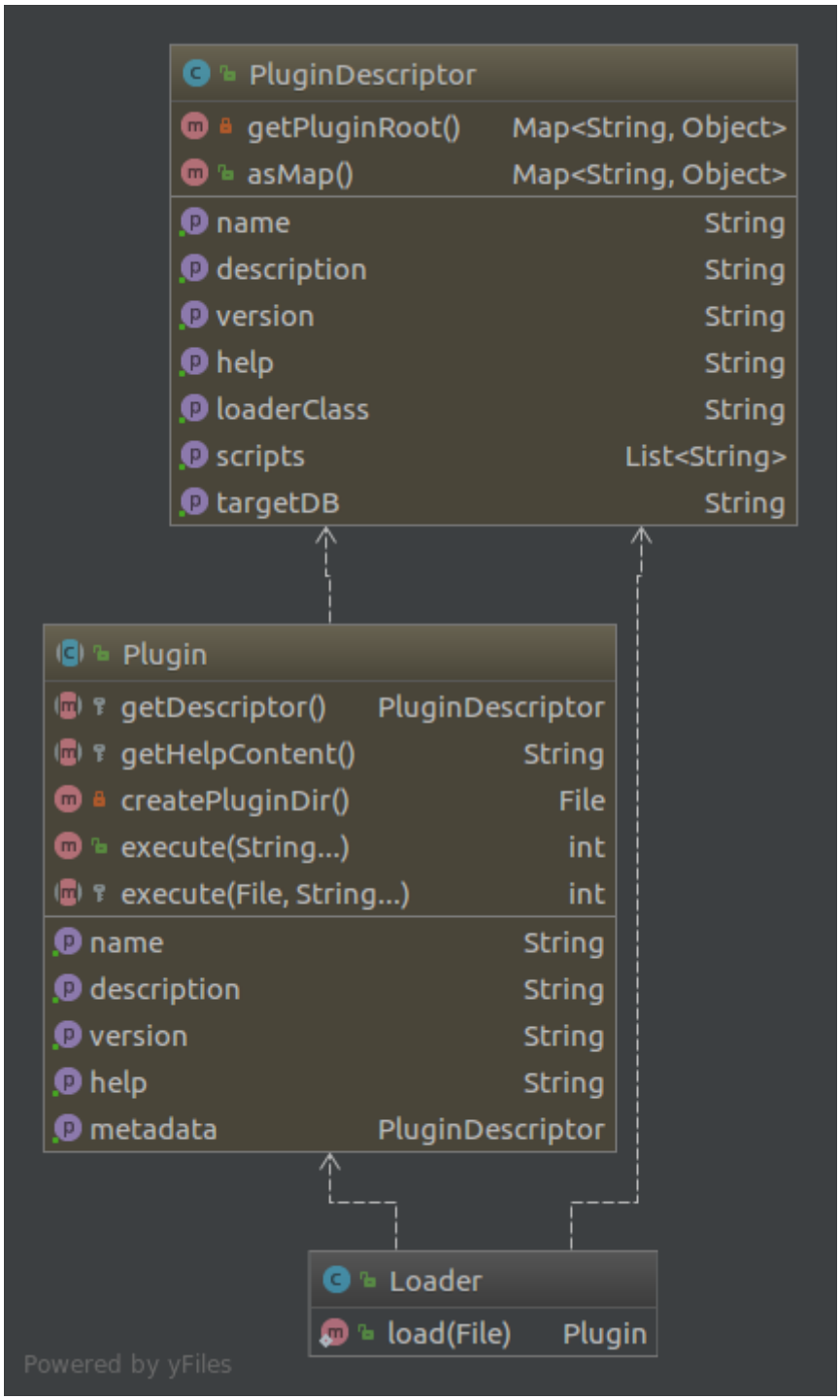
14.2 Anexo B. Diagrama de clases del paquete "*descriptor*" de Dataplugin



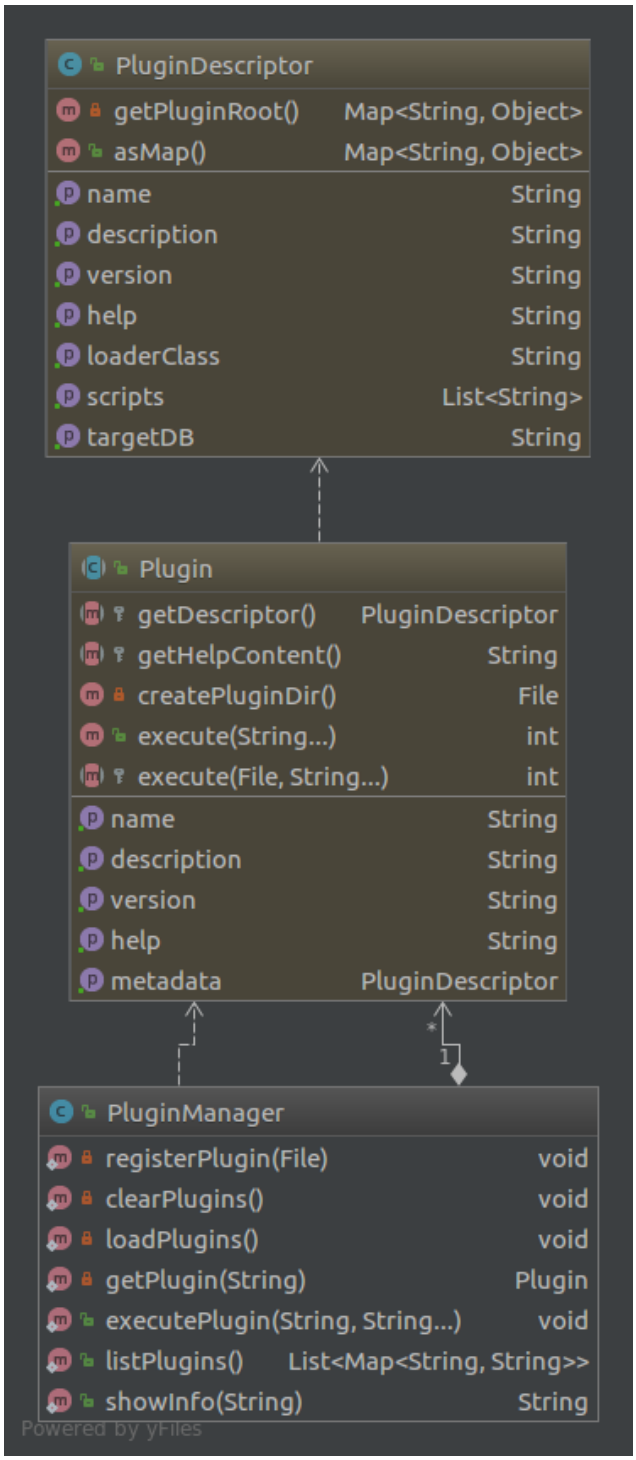
14.3 Anexo C. Diagrama de clases del paquete "*execution*" de Dataplugin



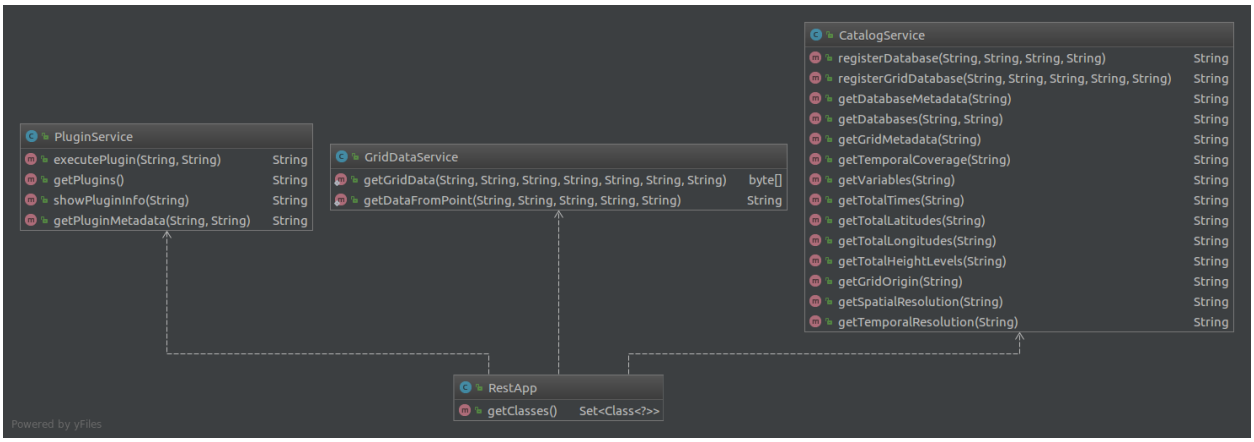
14.4 Anexo D. Diagrama de clases del paquete *"load"* de Dataplugin



14.5 Anexo E. Diagrama de clases del paquete *"registry"* de Dataplugin



14.6 Anexo F. Diagrama de clases del paquete *"service"* de Dataplugin



14.7 Anexo G. Diagrama de clases de la librería "dataplugin.api"