Masters Program in Geospatial Technologies

Array-database Model (SciDB) for Standardized Storing of Hyperspectral Satellite Images

Eias Hausen

Dissertation submitted in partial fulfilment of the requirements for the Degree of *Master of Science in Geospatial Technologies*

NOVA IMS Information Management School
Universitat Jaume I
Westfälische Wilhelms-Universität Münster

Array-database Model (SciDB) for Standardized Storing of Hyperspectral Satellite Images

Dissertation supervised by

Dr. Ignacio Guerrero

Co-supervised by

Prof. Edzer Pebesma

Prof. Marco Painho

February 2016

Author's Declaration

I hereby declare that I am the sole author of this Master Thesis which is entitled "Array-database Model (SciDB) for Standardized Storing of Hyperspectral Satellite Images".

I declare that this thesis is submitted in support of candidature for the Master of Science in Geospatial Technologies and that it has not been submitted for any other academic or non-academic institution.

Eias Hausen

Castellón de la Plana, 26.02.2016

ACKNOWLEDGMENTS

I would like to express my uttermost appreciation to Dr. Ignacio Guerrero for his supportive help and supervision. I would also like to thank Prof. Edzer Pebesma and Prof. Marco Painho for their co-supervision of this thesis.

I am grateful to the coordinating staff at the three universities in Münster, Castellón de la Plana, and Lisboa for giving me the opportunity to be part of this program and be awarded the Erasmus Mundus scholarship.

I appreciate the help of the academic and administrative staff who made the integration and attendance of this Master a straightforward process for all of the Master students.

My greatest gratitude to my family and friends who stood by me through this Master generally and also provided me with advice and support whenever needed.

Array-database Model (SciDB) for Standardized Storing of Hyperspectral Satellite Images

ABSTRACT

Hyperspectral Imaging is a technique that collects information from the electromagnetic spectrum, storing the value of the spectrum band for each pixel of the image. This technique stands out for the contiguous wide range of wavelengths it covers; leading to the ability of accurate surface and material distinction. The big volumes of Hyperspectral Images datasets, which are called data cubes as the band value represent the third dimension, have been a barrier against exploiting the full potential of these images where there is no standardized way in storing them. On top of that, the classical relational databases proved to be an inconvenient storage space for such images.

Array databases have been a serious choice for storing scientific and big volumes of data, and they represent a promising suitable environment for hyperspectral images. We aim to study the efficiency of storing hyperspectral images on an array-database by suggesting a convenient data model. Furthermore, in order to examine the feasibility of this model, we make a comparison with two relational databases using specific measurements in performance and query complexity.

KEYWORDS

Array Database

Database Comparison

GeoTIFF Images

Hyperspectral Imaging

Hyperspectral Satellite Images

Raster Database

Satellite Images Storage

SciDB

ACRONYMS

AQL – Array Query Language

CSV – Comma Separated Values

GDAL – Geospatial Data Abstraction Library

HDR – High Dynamic Range

NoSQL – Not only SQL

OS – Operating System

OGC – OpenGIS Consortium

RAM – Random Access Memory

RasDaMan – Raster Data Mangement

SciDB – Scientific Database

SQL – Structured Query Language

TIFF – Tagged Image File Format

UFI – Universal File Interface

VM – Virtual Machine

Table of Content

Index of Tables

Index of Figures

1. Introduction

This chapter serves as an introduction for this dissertation. We introduce the problem and what motivated us to address it. Furthermore, we introduce the approach and the steps followed in the attempt to reach our goal. This chapter also includes how this dissertation is organized.

1.1. Motivation

Hyperspectral images are like spectral images; they represent information of different bands of the electromagnetic spectrum. But unlike spectral images, they convey a wide range of fine wavelengths leading to obtaining the wavelength of each pixel of the image taken [Plaza et al. 2009]. These satellite images form together a cube of images where the third dimension represents the spectral wavelength. This results in collecting huge datasets of information, especially that this type of datasets is subject to increase over time due to technological advances. The current database engines lack the ability to store and process such enormous amounts of data efficiently [Stonebraker et al. 2007], besides the fact that there is still no standardized format for this type of images [Griffith et al. 2012].

A relatively new database model called array-database has been drawing the attention to. This model depends on storing data in multidimensional arrays. SciDB is an *array-database* from Paradigm4 and it has been proving efficiency with big data applications. Array-databases, like SciDB, have recently become a serious choice for storing scientific data like hyperspectral satellite images [Griffith et al. 2012].

Hyperspectral images datasets still lack a unified environment regarding storage and reuse, and they show a great deal of "Heterogeneity regarding their formats" [Sevilla and Plaza 2014]. There has been some effort to develop a standardized repository for hyperspectral images, but only on a classical relational database [Sevilla and Plaza 2014]. A standardized format for hyperspectral images and their

metadata will improve data reusability and integration, and ease the management and design of their datasets.

There has been some work on studying the performance of different array-databases [Liu 2014] as well as comparing it with other types of databases; with NoSQL [Ramakrishnan et al. 2013], and with relational database [Cudre-mauroux et al. 2011]. Nevertheless, there haven't been significant studies on comparing the different storage environments of hyperspectral images, particularly between relational and array databases. This has led to a lack of evidence that favors the option of array-database when it comes to select the storage environment of a big data enterprise like Hyperspectral imagery.

1.2. Aim and Objective

The main goal of this Master thesis is to study the feasibility of using array-database in storing hyperspectral satellite images. In order to think of array-database as a featured environment for storing scientific data, we need to reach tangible results that justify this choice over other well-known and widespread databases.

We suggest a data model for storing hyperspectral images on array-database (SciDB) and evaluate its performance in a comparison with two relational databases. For this purpose, we will be using a real world dataset of hyperspectral satellite images.

1.3. General Methodology

The general methodology of this thesis consists of three main parts:

- SciDB Data Model

  This part encounters studying the model and structure of SciDB in order to put a convenient data model for hyperspectral images and load a real world dataset in terms of the model put.
- Relational Database Storage

This part encounters storing the same dataset that was loaded onto SciDB on a classical relational database.

- Database Comparison

  Finally, we write several queries that are designed to examine the performance of both array and relational databases.

1.4. Dissertation Organization

This dissertation is organized as follows. Chapter 2 discusses the literature review and the background of this dissertation. Chapter 3 addresses in detail the methodology we followed, besides the implementation and the tools used. Chapter 4 encounters the results. Chapter 5 is about the conclusion of this dissertation and future work. Finally, data and query samples are provided in the appendices section.

2. State of the Art

This chapter addresses the background behind this dissertation, the related work done so far, and the concepts that have led to the implemented work and the concluded results accordingly.

2.1. Hyperspectral Images

The definition of hyperspectral satellite images, their specifications and usage, and the obstacles in fully exploiting their potential.

2.1.1. Introduction

Hyperspectral Images are spectral images (images that collect information of the electromagnetic spectrum) that capture the spectral-band value of each pixel of the image taken [Plaza et al. 2009]. Figure 1 shows the electromagnetic spectrum and the range of visible light.

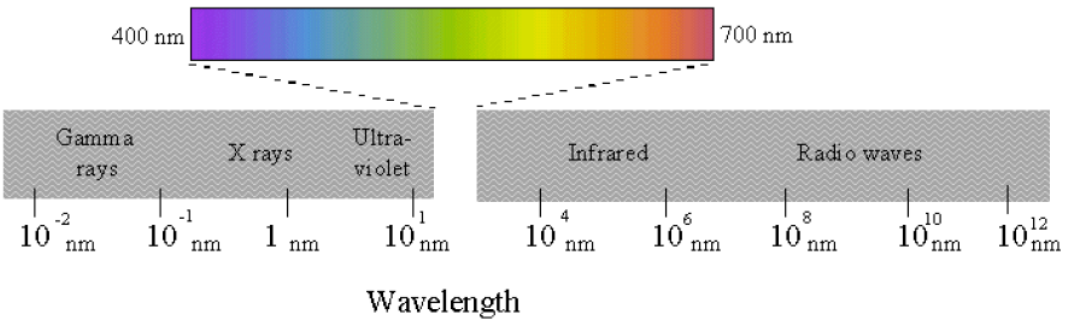

Figure 1 Electromagnetic Spectrum [Shippert 2003].

What makes these images different is that they are collected for a bigger number of bands than the normal spectral or multispectral ones. These bands cover a wide range of wavelength resolution leading to narrow, adjacent wavelengths that can be hundreds in some cases [Smith 2006]. The range and wavelengths values of the hyperspectral images depend on the sensors used. But commonly, hyperspectral imaging covers values from the visible range (400-700 nm) to near-infrared (2400 nm) with a width of 10 nm between contiguous bands [Manolakis and Shaw 2002]. Table 1 shows a list of some hyperspectral images sensors [Thenkabail et al. 2002],

some of which are out of service and others are still operational. Figure 2 shows the differences between the diverse kinds of spectral images. It also shows the ultraspectral images which surpass the hyperspectral ones in the number of bands and lead to more accurate and unambiguous material identification; they are out of our scope but worth mentioning, though. Multispectral images collect discrete narrow-band images whereas hyperspectral ones collect images over a continuous spectral range.

Table 1 Hyperspectral Sensors and Data Providers [Shippert 2003].

Satellite Sensor	**Manufacturer**	**Number of Bands**	**Spectral Range**
FTHSI on MightySat II	Air Force Research Lab	256	0.35 to 1.05 μm
Hyperion on EO-1	NASA Goddard Space Flight Center	220	0.4 to 2.5 μm
Airborne Sensor	**Manufacturer**	**Number of Bands**	**Spectral Range**
AVIRIS (Airborne Visible Infrared Imaging Spectrometer)	NASA Jet Propulsion Lab	224	0.4 to 2.5 μm
HYDICE (Hyperspectral Digital Imagery Collection Experiment)	Naval Research Lab	210	0.4 to 2.5 μm
PROBE-1	Earth Search Sciences Inc.	128	0.4 to 2.5 μm
CASI (Compact Airborne Spectrographic Imager)	ITRES Research Limited	up to 228	0.4 to 1.0 μm
HyMap	Integrated Spectronics	100 to 200	Visible to thermal infrared

EPS-H (Environmental Protection System)	GER Corporation	VIS/NIR (76), SWIR1 (32), SWIR2 (32), TIR (12)	VIS/NIR (.43 to 1.05 μm), SWIR1 (1.5 to 1.8 μm), SWIR2 (2.0 to 2.5 μm), and TIR (8 to 12.5 μm)
DAIS 7915 (Digital Airborne Imaging Spectrometer)	GER Corporation	VIS/NIR (32), SWIR1 (8), SWIR2 (32), MIR (1), TIR (6)	VIS/NIR (0.43 to 1.05 μm), SWIR1 (1.5 to 1.8 μm), SWIR2 (2.0 to 2.5 μm), MIR (3.0 to 5.0 μm), and TIR (8.7 to 12.3 μm)
DAIS 21115 (Digital Airborne Imaging Spectrometer)	GER Corporation	VIS/NIR (76), SWIR1 (64), SWIR2 (64), MIR (1), TIR (6)	VIS/NIR (0.40 to 1.0 μm), SWIR1 (1.0 to 1.8 μm), SWIR2 (2.0 to 2.5 μm), MIR (3.0 to 5.0 μm), and TIR (8.0 to 12.0 μm)
AISA (Airborne Imaging Spectrometer)	Spectral Imaging	up to 288	0.43 to 1.0 μm

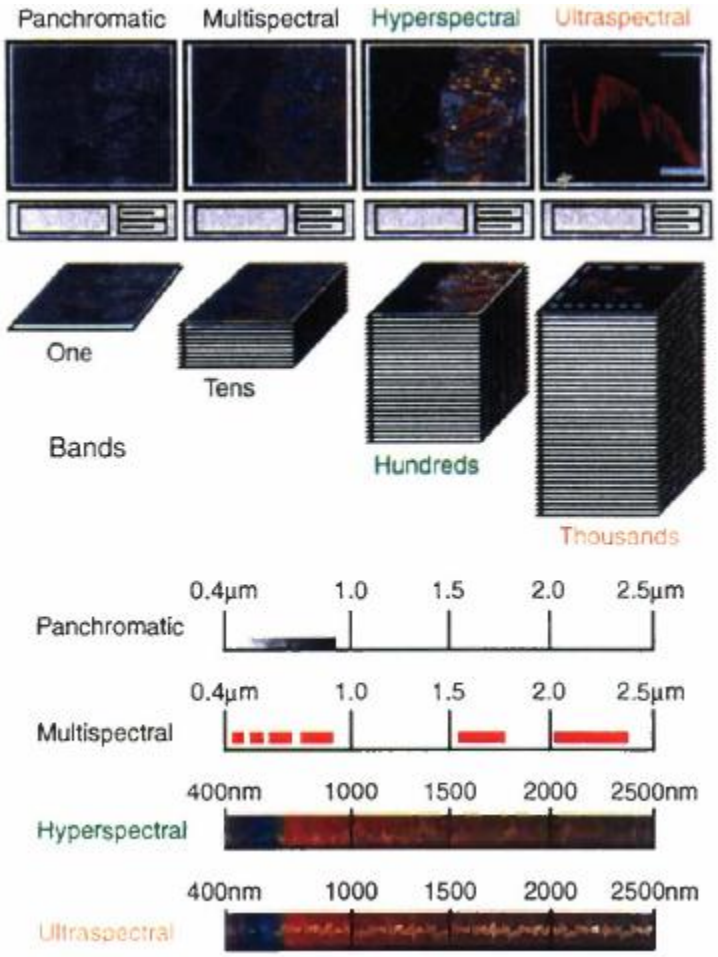

Figure 2 Evolution of remote sensing spectroscopy with respect to spectral resolution [Belokon 1997].

The consecutive images representing the different spectrum-band values form a cube of images (Figure 3); the width and height being the first two dimensions, and the band value being the third dimension. This representation leads to huge volumes of datasets, especially if we are dealing with images of high resolution and hundreds of bands.

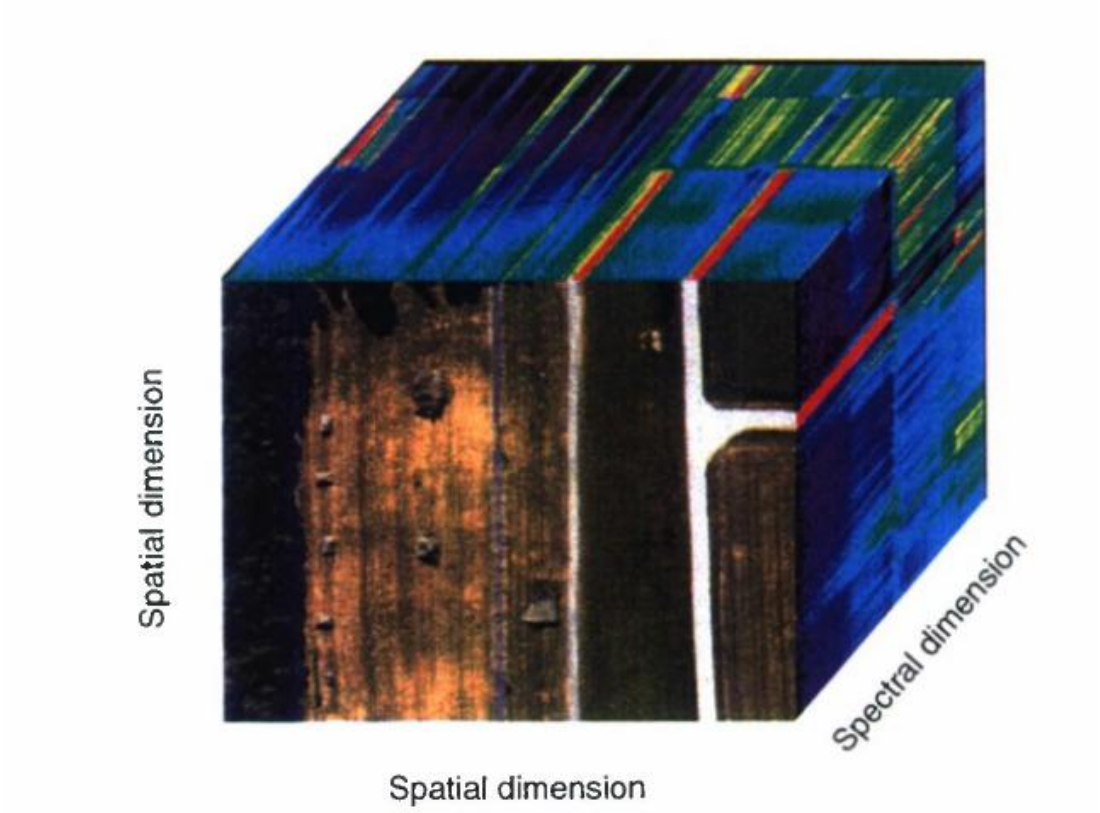

Figure 3 Hyperspectral Cube [Manolakis and Shaw 2002].

There is currently a wide range of applications that depend on hyperspectral images [Chakrabarti and Zickler 2011]. Different operating providers (Table 1) serve for different purposes and fields [Fauvel et al. 2013]:

- Ecological Science. Study of land "cover change" or estimation of biomass [Azadeh Ghiyamata and Shafria 2010], [Cochrane 2000].
- Geological Science. Recovery of "physicochemical mineral properties" [Cloutis 1996].
- Hydrological Science. Study of "wetland" aspects [Schmid et al. 2005].
- Precision Agriculture. Classification of vegetation [Lanthier et al. n.d.], [Boggs et al. 2003].
- Military Applications. Detection of military target [Manolakis and Shaw 2002], [Renhorn et al. 2012].

Despite the usefulness of hyperspectral images and their broad range of applicability, their usage has some drawbacks like any technology which will be addressed in the following section.

2.1.2. Problem Statement

The spectrometers that produce the hyperspectral satellite images use thousands of detectors where each detector measures the energy of only one band. This sophistication leads to the ability to obtain band measurements as narrow as 0.01 micrometers [Smith 2006]. The small differences between the wavelengths give high potential in discovering the type of the surface captured, regardless of the application in use.

The high density of the adjacent wavelengths in the hyperspectral images makes the normal procedures in dealing with satellite images infeasible, especially that raster images such as hyperspectral ones are considered “the most voluminous data type encountered in remote sensing applications” [Gutierrez and Baumann 2007]. The rapid advancements in remote sensing technology will make the current situation only worse; spatial resolution will increase as well as the number of bands representing the spectrum values (Ultraspectral Images). New standardized data models shall be considered in order to cope with the new challenges. Despite a huge load of work done on capturing Hyperspectral Satellite Images, these images’ datasets are still stored in different storage environments and lack a unified format and standardized metadata [Sevilla and Plaza 2014].

2.1.3. Hyperspectral Imaging Importance

The energy reflectance of the electromagnetic spectrum differs depending on the wavelength. These differences let us identify materials on the surface of the Earth; because different kinds of materials reflect light in different intensities for specific wavelengths. For example, vegetation has a high reflectance at a wavelength range between 0.7 and 1.4 micrometers (Figure 4).

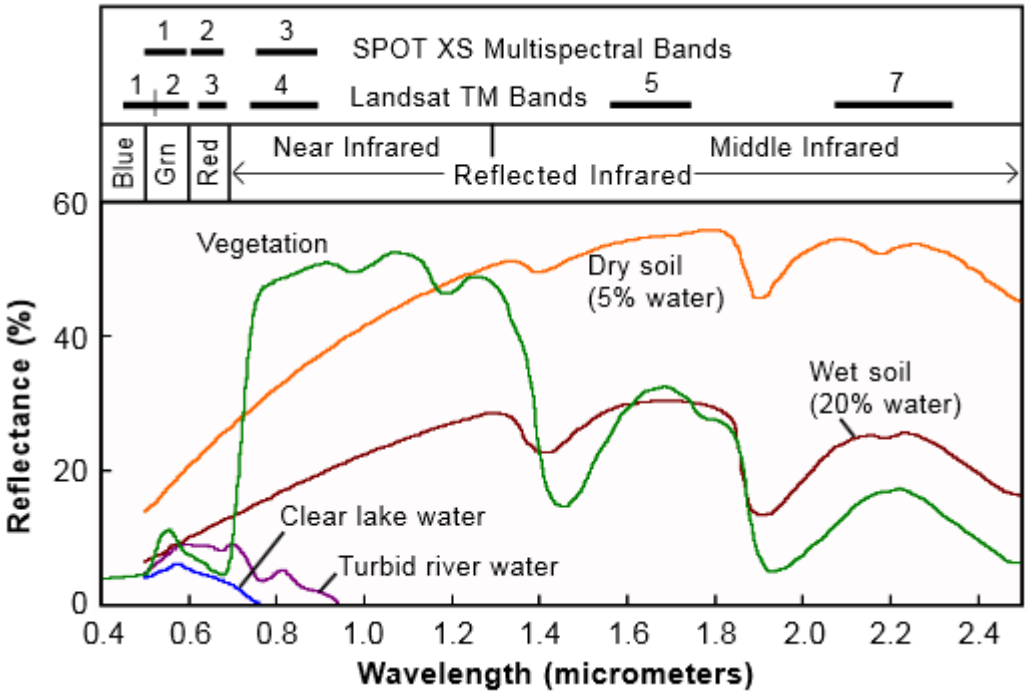

Figure 4 Energy Reflectance of the Electromagnetic Spectrum [Smith 2006].

This phenomenon gives us the chance to distinguish materials and detect spatial structures with high precision because of the accuracy of the spectrum-band values of the hyperspectral images [Fauvel et al. 2013]. Therefore, dealing with this type of images in a convenient and facilitated manner may lead to important results and highly demanded applications i.e. image classification, which is a very common process in many systems, can be very accurate thanks to the precision of hyperspectral imaging [Tarabalka et al. 2010]. Figure 5 shows a representative sample of hyperspectral images and how the difference in energy reflectance can lead to the type-of-surface detection.

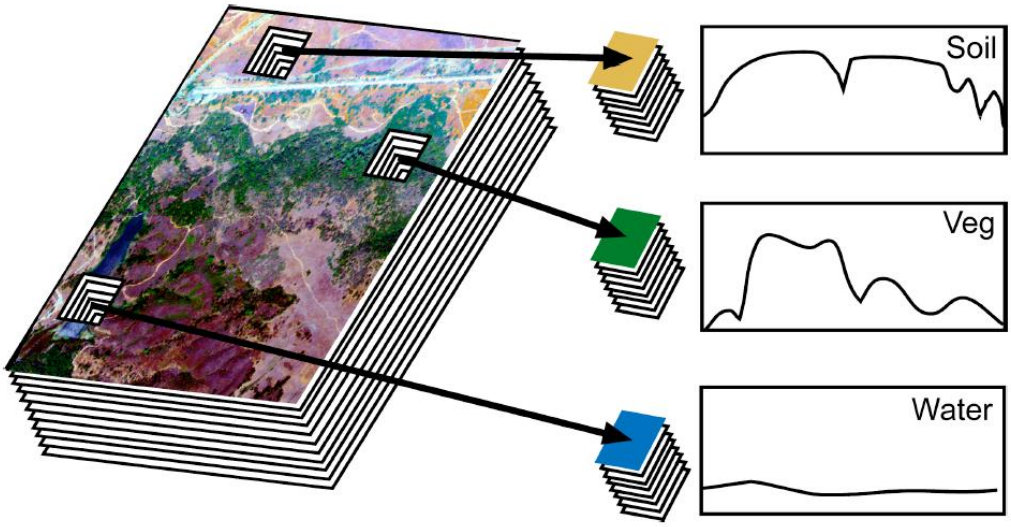

Figure 5 Reflectance Spectrum for different surfaces [Shippert 2003].

2.1.4. Storage of Hyperspectral Images

Conventional relational database management systems have been reliable for storing and retrieving data for a long time. But the sophistication and massiveness of data in the recent years made those traditional management systems unable to be dependent on [Planthaber et al. 2012]. The traditional method of storing an image in a relational database is to store it in a tabular cell as a big object [Ogle and Stonebraker 1995], or store the path of its directory as a file. These methods in most cases make no feasible solution for accessing large scale databases [Baumann 2001].

In the meantime, hyperspectral images are available in many formats, some of which are:

- GeoTIFF: it is a Tiff file format that enables embedding georeferencing information. It can bear geo-tags like the *coordinate system*, *map projection*, *datum*, and other more. It is wide used for hyperspectral images because it is a public standard format. on the other hand, it stores attribute information that concerns geo-images [Ritter et al. 1995].
- HDR: A file format that follows an imaging technique for producing higher dynamic ranges of luminosity. As this technique allows collecting a greater high-

detailed range of luminance levels, it is convenient for storing hyperspectral images that contain adjacent spectrum-band values [Myszkowski et al. 2005].

- MAT: A MATLAB binary format. It is not so convenient for storing hyperspectral images because of its exclusivity inside MATLAB software only [The MathWorks 1999].

These formats serve either a specific purpose or application (like MAT), or add some sort of an attribute layer to enrich the content of the image (like GeoTIFF). Apart from that, PostGIS, which is a spatial database extender for PostgreSQL [PostGIS 2014], defines a function with the signature "raster2pgsql"[1]. This function loads raster images of GDAL supported formats like GeoTIFF and HDR, and stores them in a PostGIS raster table. The raster is stored inside a tabular cell as one of two OGC standards, WKT (Well-Known Text) and WKB (Well-Known Binary). The aim of this functionality is to apply geometric SQL functions on raster images similarly to vectors ones, yet it still follows the main methodology of storing images in relational tables. Therefore, this database will be one of the relational ones that we will compare array-database with.

Sevilla and Plaza (2014) made a repository where one can upload hyperspectral images (mostly .hdr) and fill and edit the metadata fields. But it is only a digital repository to share and maintain the images and basic metadata about them. It is not concerned with the content of the images; it is like a search engine to retrieve images based on their content. They store the images, the metadata, and other supplementary relevant information in a relational database, and the image as it is in a tabular cell of the database.

We need a new methodology for storing the raw data of hyperspectral images so that we can retrieve specific information in an efficient manner. Furthermore, metadata about the images should be accompanied with its raw data and structured in a standardized method. Part of this dissertation aims to tackle this issue and facilitate a storage space for hyperspectral images.

[1] http://postgis.net/docs/manual-2.2/using_raster_dataman.html

2.2. Array-database

We present here the definition of array-databases, their specifications, and the different array-database management systems in the market. In addition, we go deeper in studying SciDB, the array-database management system that is being used in this dissertation.

2.2.1. Definition

Array-database management systems depend on arrays rather than tables in defining its storage structure and in manipulating the data [Baumann and Holsten 2012]. Array-databases represent, at the meantime, a serious choice for storing scientific data in the fields of remote sensing, astrophysics, statistics…etc. [Cudre-mauroux et al. 2011]. They are an intuitive choice for raster images where the cells of 2D arrays are the most appropriate space for storing the pixel values of rasterized images [Baumann and Holsten 2012].

Each cell of an array-database consists of one or more attributes that may differ in data types. To give an illustration of what an array looks like, Figure 6 shows a 3D array, where each cell may contain different attributes and even another array within.

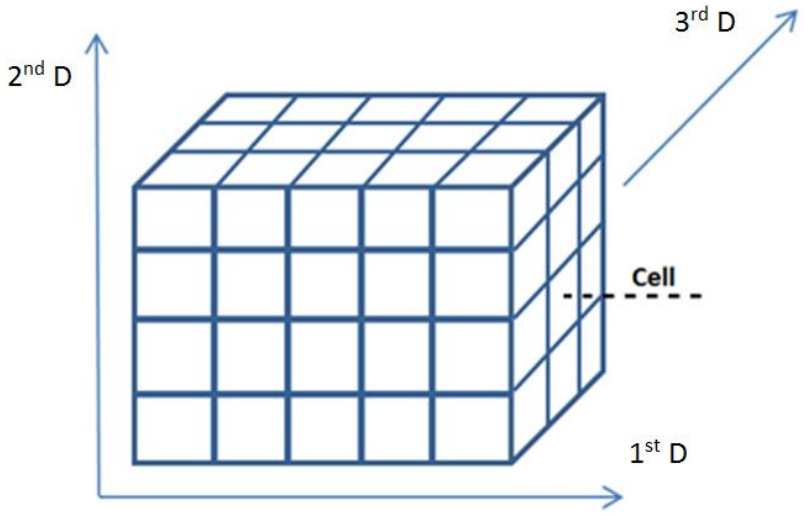

Figure 6 Example 3D array.

2.2.2. Array-database Importance

Some scientific data need a more convenient storage environment than the classical relational database. The latter has proved a high level of both performance and reliability over the last decades, but scientists tend to deal with arrays and vectors rather than tables when it comes to storage space [Cudre-Mauroux et al. 2009]. Scientific data made relational database queries that contain mathematical operations complex to build and revise, let alone the complexity of writing SQL queries that encounter operations of linear algebra [Liu et al. 2014].

The storage of multimedia resources, images in particular, lack information of the resource itself. Most traditional image databases retrieve only “feature vectors” linked to the resource [Baumann and Holsten 2012]. The profound advantage of array-databases is that they operate on the content level of the multimedia resources.

Alternatives to relational databases started to appear. Some of which is NoSQL, column-oriented database, and one which we are concerned with the most, *array-database*. These alternatives have tried to provide optimized performance, less complicated queries, and real-time data manipulation [Ramakrishnan et al. 2013].

Satellite images, for example, are array-based in nature and it makes more sense to store them in an array rather than a relational table. Let alone the timestamp which can be added as a new array dimension, whereas it has to be added artificially to a relational database [Stonebraker et al. 2013].

2.2.3. Array DBMS

Since the advent of the array-database concept, many implementations have been developed revolving around this idea. The market currently offers many array-databases [Liu 2014]:

- RasDaMan;
- SciDB;
- MonetDB;

- Essbase;
- InterSystems Caché;
- Oracle GeoRaster;
- UFI.

All of these databases differ either slightly or significantly in their data model and structure. Three of them are considered more popular for scientific data than the rest. Whereas, Oracle GeoRaster follows a different approach with object-relation scheme [Qingyun et al. 2007], RasDaMan and SciDB are the most similar.

On paper, both databases provide similar operations and services except for a significant difference that is SciDB allows nested arrays where RasDaMan does not. As for licenses, both databases have a commercial as well as an open-source version. Neither the aim nor the scope of this dissertation is about an extensive comparison between two array-databases, but between a relational one and an array-database. We think that any results concluded in this dissertation will give a general impression of array-databases despite the name of one specific implementation or commercial name.

2.2.4. SciDB

Paradigm4, the producer of SciDB, defines it as follows: “SciDB is an open-source database that organizes data in n-dimensional arrays. SciDB features include ACID transactions, parallel processing, distributed storage, efficient sparse array storage, and native linear algebra operations.” [Lewis 2014]. SciDB is also defined as follows: “SciDB is an open-source analytical database oriented toward the data management needs of scientists. As such it mixes statistical and linear algebra operations with data management ones, using a natural nested multi-dimensional array data model” [Stonebraker et al. 2001]. Figure 7 shows an example of a SciDB array and explains the nested array concept.

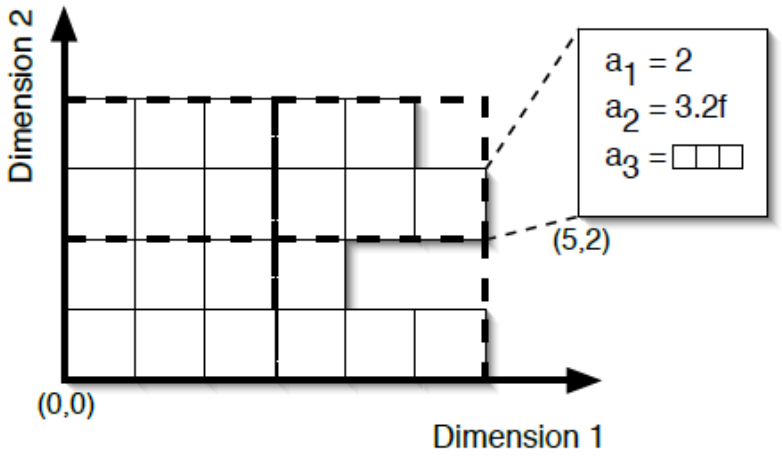

Figure 7 SciDB array example, each cell contains an integer, a floating point number, and 1D array [Cudre-mauroux et al. 2011].

SciDB proved to be faster than a relational database with about "2 order of magnitude on a typical science workload" [Stonebraker et al. 2001]. Figure 8 shows an example of how the same data of a raster image could be stored on a classical relational database and on SciDB.

Relational Database

x	y	value
0	0	5
0	1	24
0	2	77
1	0	32
1	1	13
1	2	34
2	0	81
2	1	52
2	2	19

27 cells

SciDB

5	24	77
32	13	34
81	52	19

9 cells

Figure 8 Storage needed in a relational database and SciDB.

3. Modeling and Implementation

In this chapter, we talk about the main approach followed in order to achieve the goal we aim at in studying the efficiency and performance of array-database, besides the tools that were used.

3.1. Methodology

The work of this thesis consists of several independent processes. The first encounters loading a real world dataset in GeoTIFF format into SciDB management system after establishing a data model for this purpose. The second is about loading the same dataset into two relational databases, PostgreSQL and PostGIS Raster. Finally, the last procedure encounters making a comparison of performance and query complexity between the different databases (Figure 9).

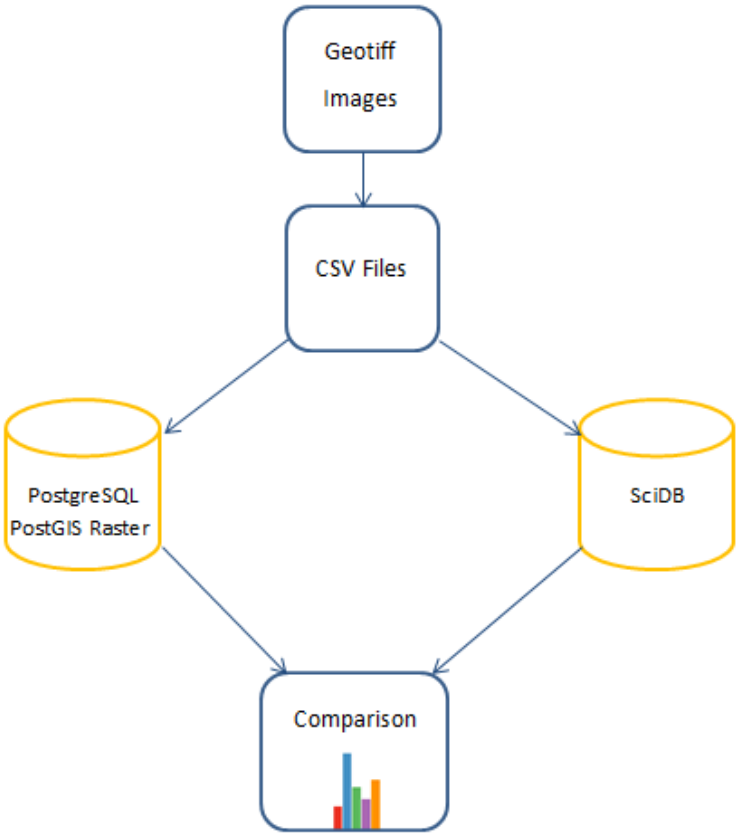

Figure 9 Main Approach.

3.1.1. Array-database (SciDB)

The main part of this procedure is about loading the data into SciDB, which is alone a lengthy process. SciDB provides three main options for data loading depending on the format of the file that is being loaded; CSV, binary, and SciDB format besides two options of opaque or TSV loading [Inc. 2012]. Each loading technique of them includes some advantages and disadvantages.

In order to load the data into SciDB, we must have the ability to initially produce the files that are going to be loaded in a specific structure and byte-granularity order (except for the .csv files). Both SciDB format and the binary format demand a high understanding of how SciDB management system reads the files and how it disassembles the file bit by bit. For example, “A string data type that disallows nulls is always preceded by four bytes indicating the string length”; this is a rule out of many others that must be followed while generating the binary file before loading it into SciDB.

Files of CSV format are another possible option for loading the data. The structure and the granularity of the data are known and standardized, which is values separated by a comma. Nevertheless, we encounter two shortcomings of this choice. Firstly, the relatively larger size of the stored files comparing, for example, to the binary ones. Secondly, the slower process of loading the files into SciDB. On the other hand, taking this option let us avoid the complicated process of writing the binary files and a high possibility of reaching a dead end in loading the data 100% correctly in SciDB arrays.

Planthaber et al. (2012) faced the same issue concerning the choice of the file format with the MODBASE system. “An impediment to the ease of use of each SciDB load format for multidimensional arrays is the requirement that data in the load files be partitioned and organized by chunk. Meeting this requirement can entail significant preprocessing of the data and forward knowledge of the dimensions and chunk sizing of the target array. This approach may prove infeasible for many real-world big data applications” [Planthaber et al. 2012]. This does not

only make us prefer a loading option rather than another, but also urges us to think thoroughly in the convenience of choosing SciDB depending on the application in hand.

Consequently, the raw data of the images, which represent the spectrum-band values of the pixels, were stored using the CSV format using a tool we developed. In addition to the raw data of the images, a separate CSV file was generated containing four metadata tags as principal attributes of the hyperspectral images.

3.1.2. Relational-database (PostgreSQL and PostGIS Raster)

In parallel with loading the data into an array-database, the same dataset was loaded into two relational-database management systems which are PostgreSQL and its PostGIS Raster extension.

Most typical geodatabases deal with data in a relational model, either implicitly or explicitly. Storing an image in a relational database is typically a binary large object “Blob” in a tubular cell, PostGIS follows this approach under a structured formula transparent to the user. Therefore, we decided to examine an image database (PostGIS Raster) and manually build another model on PostgreSQL. Consequently, besides the raster storage of PostGIS, the spectrum-band values of all images were stored in relational tables in PostgreSQL. This way, we can compare SciDB with two different relational databases, one of which is dedicated for image storage.

3.1.3. Comparison

After loading the same dataset into the three databases, a comparison was performed by writing specific queries that tackle different performance aspects, and running these queries on the different management systems. The aim of this comparison is to detect the most convenient database for hyperspectral images and give an overview of the feasibility of following this new trend in changing from relational database to array-database for scientific data. Performance is the key factor in making this comparison concluded with tangible results. Other important factors are the length and complexity of the retrieval commands.

3.2. Implementation

The implementation encountered different processes and diverse tools in order to make the previously explained approach work. Starting from preparing the data, and ending with final results.

3.2.1. Installation and Hardware

Theoretically, the comparison between the two databases (array and relational) is not quite equitable. SciDB was installed on an Ubuntu Machine (Table 3) where PostgreSQL and PostGIS (Table 2) were installed on a Windows machine with little differences of resources. Unfortunately, we cannot measure the difference caused by this inequity. But we have to expect a result that does not totally match with the one that would be concluded within an ideal equal environment.

- PostgreSQL and PostGIS

Table 2 PostgreSQL and PostGIS installation environment specifications.

Operating System	Windows 7 Ultimate
Processor	Intel(R) Core(TM) i5-3337U CPU @ 1.80 GHz 1.80
Processors	4
Hard Disk	750 GB
RAM	6.00 GB
System Type	64-bit

- SciDB

Table 3 SciDB installation environment specifications.

Operating System	Linux Ubuntu 14.04.3 LTS

Processor	Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
Processors	4
Hard Disk	300 GB
RAM	8.00 GB
System Type	64-bit

Besides the different operating system, the “Ubuntu” machine mainly surpasses the “Windows” machine in two more Gigabytes of RAM and a bit higher processor speed.

3.2.2. Dataset

The data we worked with is “E01 Hyperion” from the explorer of the US Geological Survey. This dataset is a 242 band hyperspectral satellite images in the format of GeoTIFF (.tif) (Table 4).

Table 4 Dataset specifications.

Name	USGS EARTH OBSERVING-1 (EO-1) - HYPERION
Coordinate System	UTM Zone 43 Northern Hemisphere (WGS 84)
File Format	GeoTIFF
Number of Images	242
Page per Image	1
Width/Height	961/3501
Bits per Sample	16-bit
Size (Original Images)	1.6 GB

Size (As CSV files)	5 GB

3.2.3. File Writer

As previously explained, SciDB loading process demands to prepare the files for loading and storing them as arrays. The user should be fully aware of the format and structure of the data files in order to correctly *migrate* the data onto the arrays of SciDB. The dataset in hand is formatted as GeoTIFF files. Therefore, a tool was implemented for the purpose of converting the GeoTIFF files into CSV files in a specific order and structure.

The tool is implemented using C# .Net framework and Libtiff.Net (a .Net library for manipulating tiff files), and is capable of the following features:

- Reading .tif files (both types: single band per image – multiple bands per image) and showing all specifications and metadata of the files.
- Reading .csv files.
- Running a Python script which is responsible for extracting the spectrum-band values from the GeoTIFF files and saving them in CSV files; each image is saved in a separate CSV file.
- Taking the row data needed from either the .tif or .csv files which are the spectrum-band values and saving the data as follows:
    1. One CSV file for four metadata tags including (width – height – bits per sample – Geodetic System)
    2. One CSV file for all raw data (all images). Each column of the file contains all the values of one image line by line consecutively.

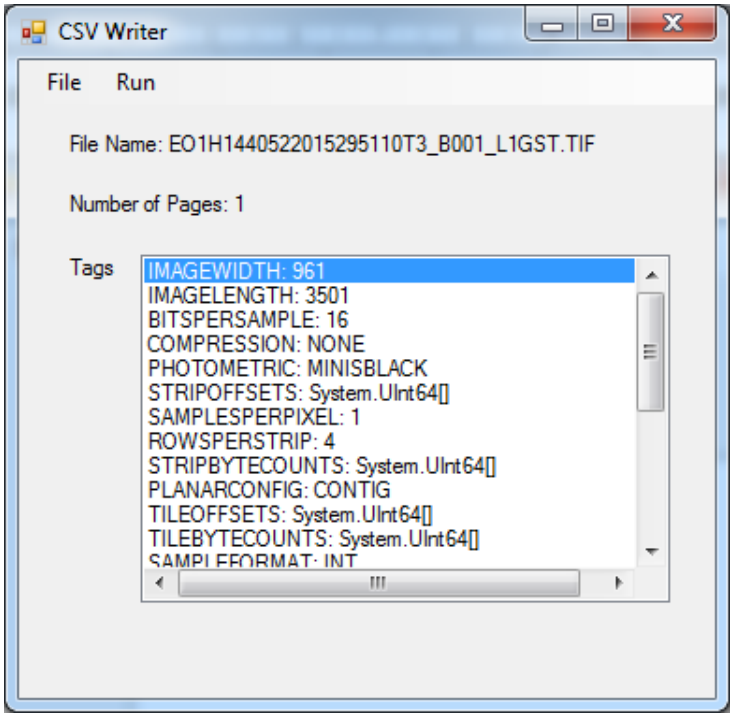

Figure 10 File Writer interface showing the details of one GeoTIFF.

Figure 10 shows the interface after loading one image from the Hyperion dataset. It shows the file name, the number of pages per GeoTIFF image, and all the metadata tags the image embeds.

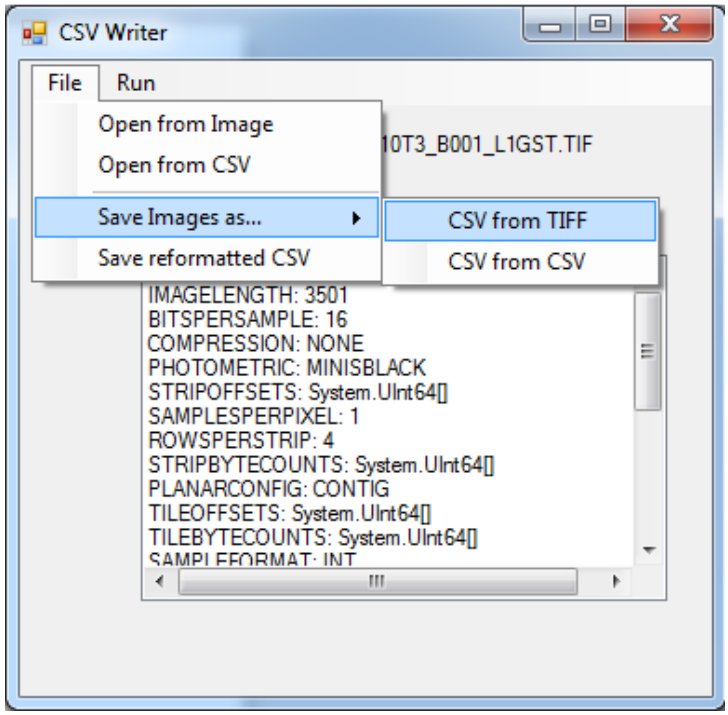

Figure 11 File Writer interface showing the different options available.

Figure 11 shows the main functionalities as they appear on the interface, including all capabilities in dealing with both CSV and GeoTIFF files.

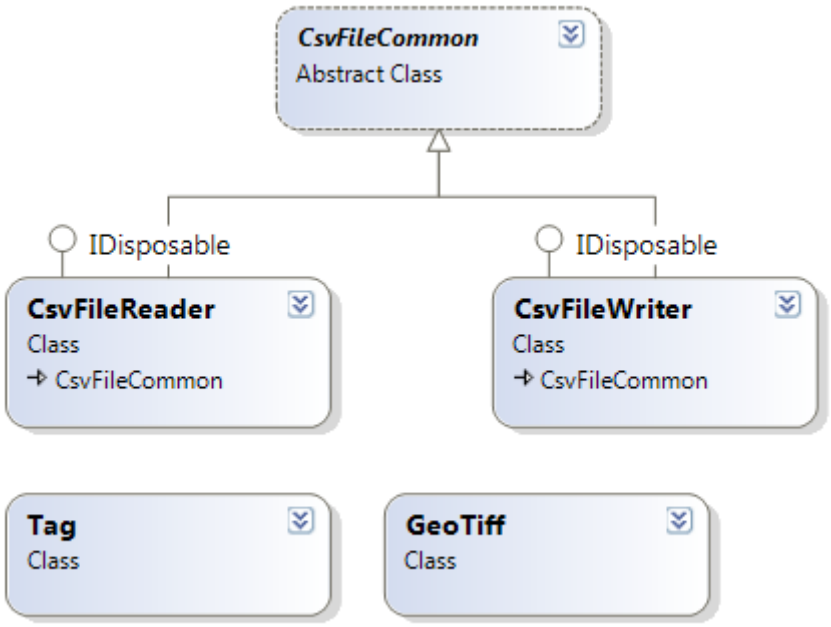

Figure 12 Class Diagram of the File Writer

Figure 12 shows the class diagram of the tool generated by Microsoft Visual Studio 2010. The tool shows a high level of flexibility in terms of the type of the files being processed and the written files as well. It can edit the header of the CSV files according to the query command of SciDB. The result eventually is .csv files ready to be loaded into SciDB.

3.2.4. SciDB Loading

SciDB runs on a Linux Operating System. For this purpose, a Linux Ubuntu OS was obtained through Amazon Virtual Server, and sufficient memory and processing resources were given along (technical details available in Table 3). We ran SciDB 14.12, which is the latest open version of SciDB.

Two data models were put in order to load the data into SciDB. The approach of dealing with SciDB was followed in two parallel processes. The first encounters loading each image into a 2D array, where the second encounters loading all of the

images together into one 3D array. These two different methods enable us examine the performance of SciDB according to diverse storage approaches.

Loading data into SciDB follows only one way, which is loading the CSV file into a 1D array and then re-dimension this array, if needed, into the intended n-dimension array. Unfortunately, there is not a loading process that fetches the data directly into an array of a pre-specified number of dimensions. Table 5 shows the values of the attributes that were changed in the configuration file in order to adapt to the size of the data loaded.

Table 5 SciDB memory attributes configuration (all values in MB).

sg-send-queue-size = 100
sg-receive-queue-size = 100
smgr-cache-size = 1024
mem-array-threshold = 1024
merge-sort-buffer = 128
network-buffer = 512
replication-send-queue-size = 500
replication-receive-queue-size = 500
max-memory-limit = 5000

The result of this process for the first approach is one 1D array containing the metadata tags, and two hundred forty-two 2D arrays containing the spectrum-band values. Whereas, the result of this process for the second approach is similarly one 1D array containing the metadata tags, and one 3D array of the spectrum-band values.

3.2.5. PostgreSQL and PostGIS Loading

Depending on the previously explained tool and running the Python code, we obtained CSV files representing the metadata and the images of the dataset.

Concerning PostgreSQL, the CSV files were migrated into relational tables where each file was stored in a different separate table. This process was accomplished

using a trial version of a database-transferring tool called “Navicat Premium”. The result is one table containing the metadata tags, and 242 tables containing the band values where the table attributes represent the (x-axis) and the table rows represent the (y-axis).

As for PostGIS, GeoTIFF images were stored in one PostGIS raster table using a PostGIS function called “Raster2pgsql”. The result is one table containing all of the images where each image is stored in one single row. It is noticeable here that there was no need to store the metadata tags explicitly because PostGIS can access such type of data within its structured storage of the raster.

3.2.6. Impediments

- Machine Inequity

  SciDB can be installed on a Linux OS, not Windows. Consequently, the comparison between the different databases was not performed on a single machine. Nevertheless, the specifications of the two machines were not that far from each other (detailed numbers are available in Table 2 and Table 3).

  Naturally, a comparison between two identical machines and even identical operating systems will give more accurate and precise results.

- Dataset

  The obtained dataset was fairly convenient to assess the two databases with, in regards to both number of images and the image resolution. Unfortunately, because of some technical problems that had been rising through the writing of this thesis, we did not have the time to test the different databases with other datasets of different nature. Repeating the whole process with different datasets would give more accurate and dependable results.

4. Results and Discussion

In this chapter, we present and discuss the results of our work which encounters the data models of SciDB, and the comparison between SciDB, PostgreSQL and PostGIS.

4.1. Data Model

As explained previously, two approaches will be followed within SciDB. One is concerned with multiple 2D arrays, and another with one 3D array. In this regard, each approach will depend on a conceptual data model as well as a physical data model.

4.1.1. Conceptual Data Model

The first conceptual data model is designed to eventually load the data into 2D arrays. It consists of one CSV file headed with the four metadata tags, and a number "n" of other CSV files as many as we have in the dataset, headed with the (x-axis) of number "m" (Figure 13).

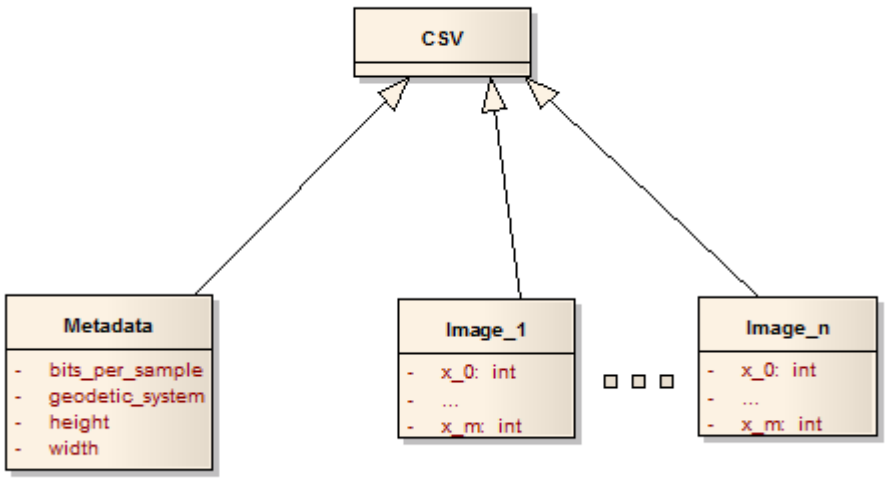

Figure 13 First conceptual data model.

The second conceptual data model gathers all the CSV files in one file headed with the image number. The values of each image are stacked line by line from the

original file into one column in the target file, and the metadata file stays without any changes (Figure 14). This model takes into consideration the two methods in storing band values: band-sequential where all values of one image are followed by all values of the next image, and band-interleaved where the values of all images of one specific point are consequent; followed by the values of all images of the next point.

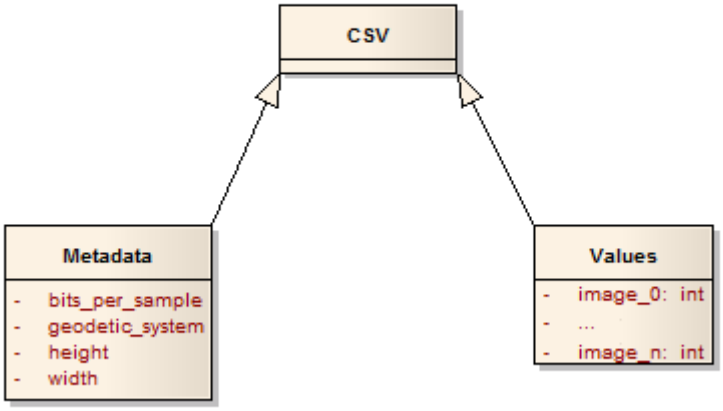

Figure 14 Second conceptual data model.

4.1.2. Physical Data Model

The physical data model represents the interpretation of the conceptual data model into SciDB arrays. The first physical model corresponds to the first conceptual model. The metadata array is a 1D array where each cell consists of 4 attributes which represent the 4 tags (Figure 15 - A). Each image is represented in a 2D array where each cell is one integer attribute which is the spectrum-band value of the respective pixel (Figure 15 - B).

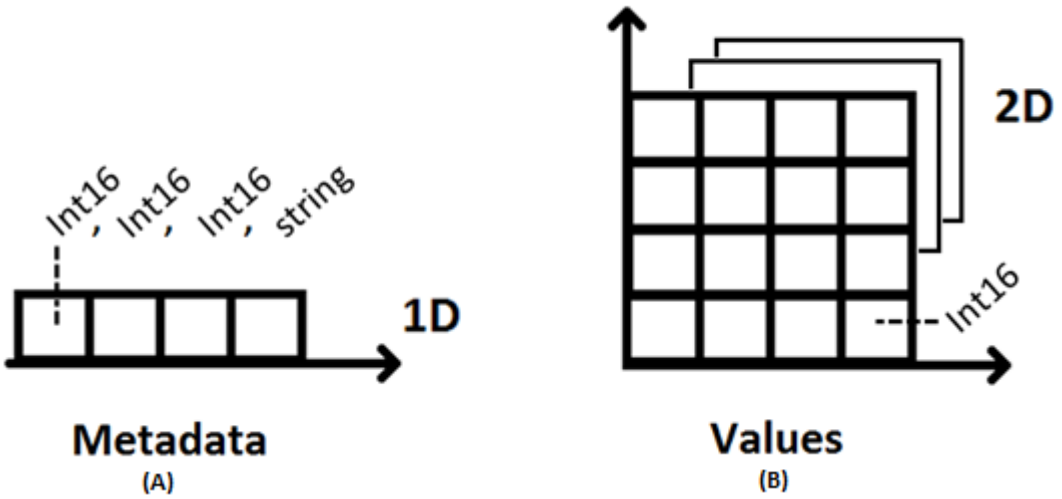

Figure 15 First physical data model, (A) Metadata array, (B) Values arrays.

The second physical model corresponds to the second conceptual model. The metadata array, as the first model, is a 1D array where each cell consists of 4 attributes which represent the 4 tags (Figure 16 - A). All images are represented in one 3D array (cube) containing the spectrum-band values (Figure 16 - B).

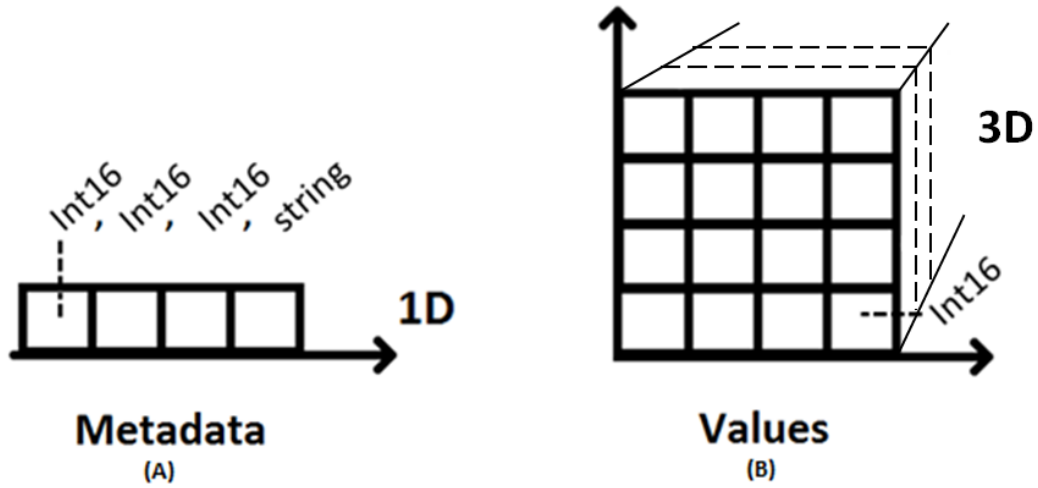

Figure 16 Second physical data model, (A) Metadata array, (B) Values array.

4.2. Query Comparison

The comparison encounters asking specific questions about the dataset in hand. These questions are translated into AQL, SQL commands in SciDB and

PostgreSQL/PostGIS respectively. The queries show diversity in the retrieved data, dimensionally in specific, to tackle the different aspects of each of the databases. Furthermore, they represent common retrieval commands of image datasets. One query retrieves a 1D line of values, the second retrieves a 2D plane of values, and last but not least, the third retrieves a 3D cube of values.

All the queries were about the raw data of the images, not the metadata tags. The real challenge in our case is huge volumes of data, and the four tags we stored serve only in the completion and integrity of the model but not for comparison purposes. The values of "performance" are average numbers taken after multiple attempts of the same query, and full samples of the queries are available next to each other in the appendices. Unfortunately, the queries were applied using only two different sets of values. A third value would have made the performance tendency of each database clearer, especially in the charts below. But any significant change in the values would make the results of one specific database either too fast or too slow which made it unpractical to include it in the comparison results.

- First Query

Fetch the spectrum-band values of the point (x=100, y=100) for images (n to m).

Table 6 First query comparison.

	SciDB (1st Model)	**SciDB (2nd Model)**	**PostgreSQL**	**PostGIS Raster**
(n=1 , m=10)				
Length (char)	230	84	1388	80
Performance (sec)	39	1.8	0.4	0.15
(n=1 , m=50)				
Length (char)	949	84	7193	80
Performance (sec)	1120	7.8	19	0.8

Table 6 shows that PostGIS retrieves data for this kind of queries much faster than the others followed by the second model of SciDB. Nevertheless, the second model of SciDB shows a tendency in performing best for a larger set of retrieved data. The numbers also show that the more images we retrieve, the higher pace of speed regression PostgreSQL will show in comparison to SciDB; the second model to be precise. This is demonstrated in Figure 17 where we had to apply a logarithmic scale on the time axis to distinctly show the performance differences. As for query length, PostGIS and the second model of SciDB show a significant reduction in the number of characters.

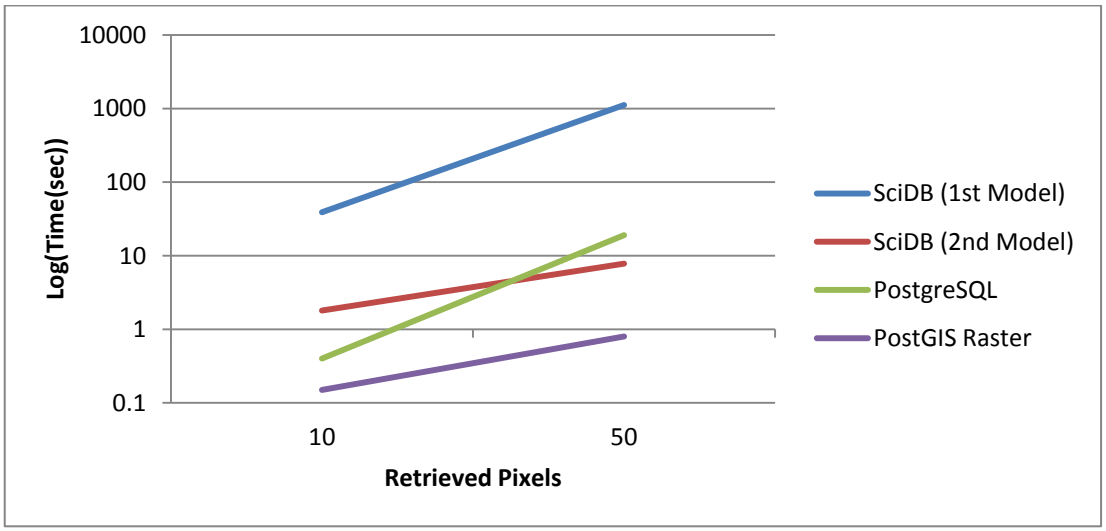

Figure 17 First query chart of performance.

Figure 18 shows a representation of what the first query is intended to retrieve, which is the pixels of a specific point of a set of images.

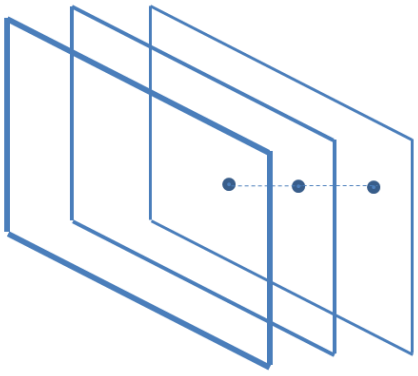

Figure 18 First query representation.

- Second Query

Fetch all the spectrum-band values of Image_30 for the following axes values (x1<x<x2)/(y1<y<y2) .

Table 7 Second query comparison.

	SciDB (1st Model)	SciDB (2nd Model)	PostgreSQL	PostGIS Raster
(x1=400 , x2=900) / (y1=1000 , y2=2000)				
Length (char)	89	92	6160	167
Performance (sec)	20	34	1.8	690
(x1=0 , x2=900) / (y1=0 , y2=2000)				
Length (char)	90	93	8224	167
Performance (sec)	67	89	2.1	1642

Table 7 shows that PostGIS is by far the slowest database in accomplishing such type of queries. PostgreSQL shows high performance against SciDB, but this performance is at the expense of query length which is way longer than the others.

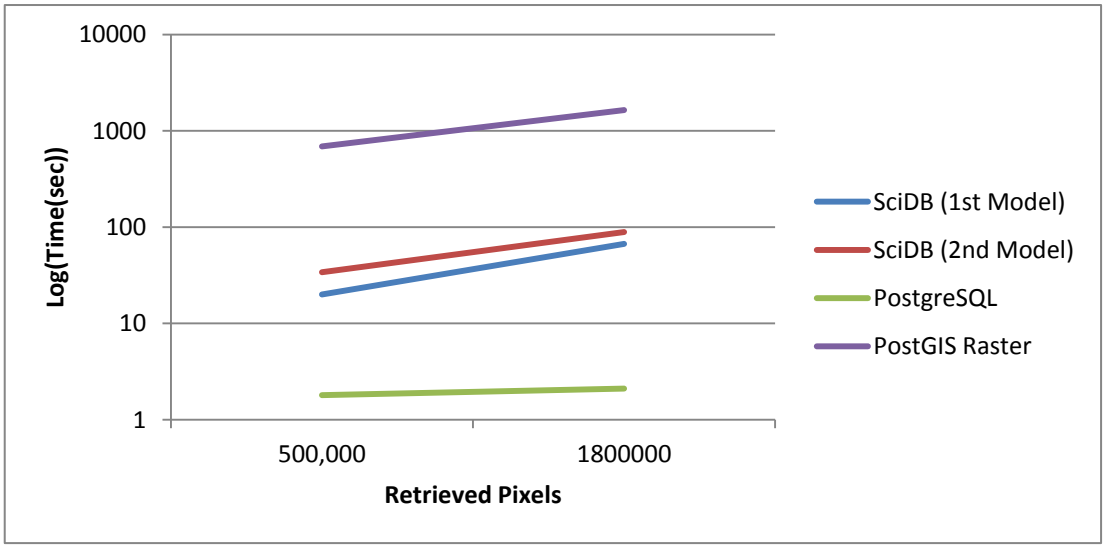

Figure 19 Second query chart of performance.

Figure 19 shows that PostgreSQL has a relatively more stable and better performance than the others. Once again, an algorithmic scale was applied on the time axis to show a clear comparison. Figure 20 shows a representation of what the second query is designed for.

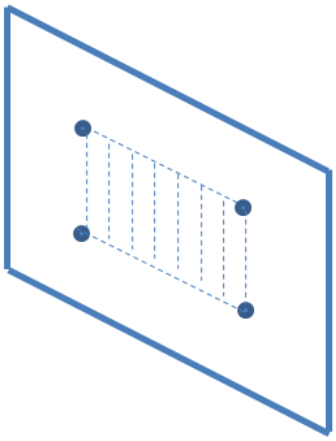

Figure 20 Second query representation.

- Third Query

Fetch all the spectrum-band values of images (n to m) for the following axes values (x1<x<x2)/(y1<y<y2).

Table 8 Third query comparison.

	SciDB (1st Model)	**SciDB (2nd Model)**	**PostgreSQL**	**PostGIS Raster**
(n=1 , m=5) , (x1=0 , x2=50) / (y1=0 , y2=50)				
Length (char)	166	103	7349	178
Performance (sec)	16	13	0.3	173
(n=1 , m=10) , (x1=0 , x2=900) / (y1=0 , y2=900)				
Length (char)	252	104	92880	181
Performance (sec)	96	112	343	1037

Table 8 shows that SciDB, with both models, outperforms the relational databases when the number of retrieved pixels gets higher. As for query length, the second model of SciDB surpasses all other models.

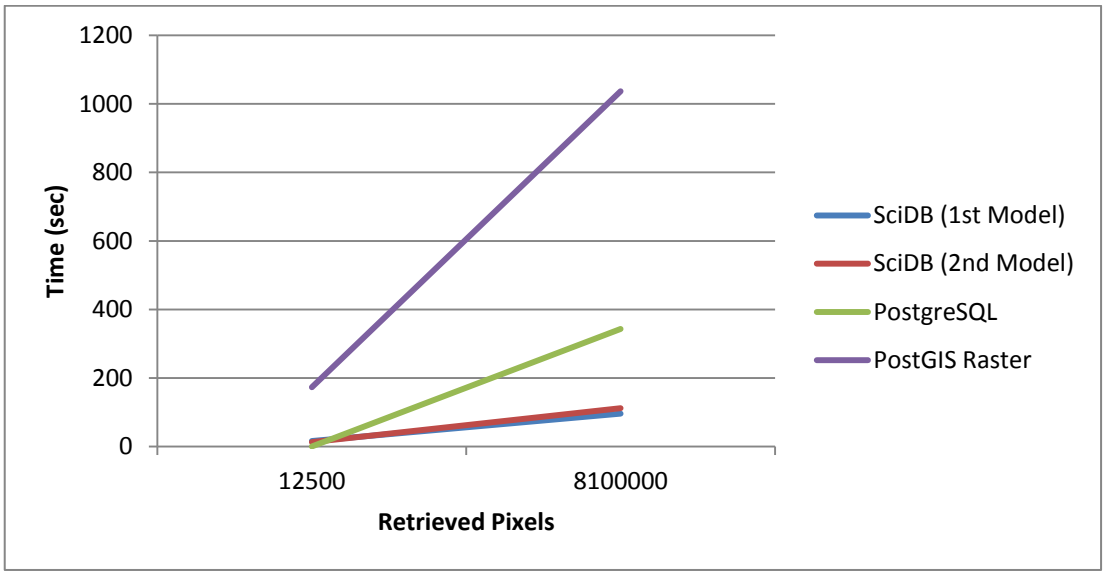

Figure 21 Third query chart of performance.

Figure 21 clearly shows a stable performance of SciDB; such queries demonstrate the advocacy of an array-database over a relational one. We did not have to apply an algorithmic scale on the time axis for this comparison as the differences are clear with the original scale and values. Figure 22 shows a representation of the third query.

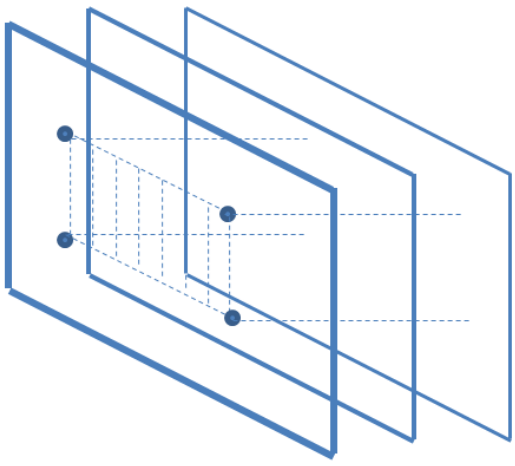

Figure 22 Third query representation.

4.3. Discussion

The results we ended up with do not give us the ability to make a final judgment of SciDB and array-databases in general. Nevertheless, they put us on the right track to do so.

As SciDB needs a clear and known structure of the files that are going to be loaded within, this constricts the feasibility of its usage. Each source of scientific data needs to put a uniform model to prepare the data for SciDB. The data model we put was an attempt to make this process mode standardized.

The query comparison between SciDB, PostgreSQL and PostGIS showed a significant difference in query length in favor for SciDB. This is a true indicator that SciDB simplifies queries for scientific data. As for performance, the more complicated the query was, the higher performance SciDB showed against the other databases, and this applied to both data models of SciDB. Furthermore, SciDB showed a tendency in a better performance when the data retrieved gets larger.

For the first query that retrieves a specific point from multiple raster images, PostGIS was the best in both performance and length, but the values show that the second model of SciDB would perform better when we deal with larger amounts of data.

The second query that retrieves a plane from a single image gave the advantage of performance to PostgreSQL where we queried a single relational table. This demonstrates that a single table query, with no “joins” involved, can make a relational database outperforms any other type of databases. But it is also worth to mention that we had to compromise the length of the written query.

The third query, which retrieves a cube and is considered the most complicated, gave the advantage to both models of SciDB with a tendency to perform way better with larger amounts of data. As for length, the query of the second model of SciDB was the shortest and the least complicated. This query is a proof that SciDB excels in such type of image queries with large datasets.

The comparison results show a featured performance of SciDB with large scientific datasets. However, SciDB is not designed to be deployed on personal computers nor used with any dataset regardless of its nature. Before deciding for SciDB, we need to answer critical questions, such as: How adequate is my dataset for SciDB? What are the minimum hardware resources I must obtain? And what type of queries I am going to apply?

5. Conclusions and Future Work

In this dissertation, we studied what hyperspectral images are, and mentioned what purpose they can serve. We spotted the relatively new concept of array-database and used SciDB as a case study of such databases.

We addressed the feasibility of the current storage methodologies of hyperspectral images and reached a conclusion that there is no wide acceptable environment of the sort. In this regard, we suggested two data models for storing hyperspectral images on SciDB.

We stored a real world dataset of hyperspectral images on SciDB and on two relational databases which are PostgreSQL, and its extension PostGIS Raster. We put three different queries and ran them on the different databases. The results were generally in favor of SciDB, especially when the amounts of data get larger. It is crucial to mention that the inequity between the two machines that ran the databases gave a slight advantage of processing capabilities and memory to SciDB.

SciDB proved to be a serious choice for storing scientific data such as hyperspectral images. It mostly showed higher performance and fewer query characters in comparison with the relational databases. This work spots a light on the promising future of array-databases and their effectiveness in storing scientific and big volumes of data.

The work done in this thesis can be carried on with to generalize the study and broaden the scope of hyperspectral images. A unified data model can be studied for general satellite images. The comparison between SciDB, PostgreSQL and PostGIS can be extended with more datasets and more complicated queries in order to more accurately study the advantages and disadvantages of the two types of databases.

Appendices

Data sample

Metadata Tags

Below is a sample of metadata tag CSV file.

```
IMAGEWIDTH,IMAGELENGTH,BITSPERSAMPLE,SYSTEM
961,3501,16," UTM Zone 43 - Northern Hemisphere|WGS 84|"
```

Spectrum-Band Values

Below is a sample of single image spectrum-band value CSV file.

```
x0,x1,x2,x3,x4,x5,x6,x7,…
1458.00,1449.00,1336.00,1290.00,1341.00,1360.00,1314.00,…
1401.00,1353.00,1377.00,1527.00,1539.00,1303.00,1278.00,…
1361.00,1348.00,1402.00,1343.00,1291.00,1260.00,1200.00,…
1522.00,1499.00,1423.00,1361.00,1379.00,1413.00,1356.00,…
1315.00,1269.00,1299.00,1224.00,1251.00,1288.00,1289.00,…
1337.00,1335.00,1326.00,1267.00,1299.00,1388.00,1415.00,…
1348.00,1418.00,1518.00,1393.00,964.00,791.00,747.00,…
…
```

Query sample

First Query

Below is a sample of the first query on SciDB 1st Model (x=100, y=100, image_001 - image_010).

```
time iquery -q  "SELECT * FROM image_001 as i1, image_002 as i2, image_003 as i3, image_004 as i4, image_005 as i5, image_006 as i6, image_007 as i7, image_008 as i8, image_009 as i9, image_010 as i10 WHERE i1.x=100 and i1.y=100"
```

Below is a sample of the first query on SciDB 2nd Model (x=100, y=100, image_001 - image_010).

```
time iquery -q  "SELECT * FROM all as i WHERE i.x=100 and i.y=100 and i.z>0 and i.z<10"
```

Below is a sample of the first query on PostgreSQL (x=100, y=100, image_001 - image_010).

```
select x1,x2,x3,x4,x5,x6,x7,x8,x9,x10 from
(WITH m1 AS (SELECT H001.x105 x1,  row_number() over() as r1 FROM "H001" H001) SELECT x1 FROM m1 WHERE r1 =2956) q1,
(WITH m2 AS (SELECT H002.x105 x2,  row_number() over() as r2 FROM "H002" H002) SELECT  x2 FROM m2 WHERE r2 =2956) q2,
(WITH m3 AS (SELECT H003.x105 x3,  row_number() over() as r3 FROM "H003" H003) SELECT x3 FROM m3 WHERE r3 =2956) q3,
(WITH m4 AS (SELECT H004.x105 x4,  row_number() over() as r4 FROM "H004" H004) SELECT x4 FROM m4 WHERE r4 =2956) q4,
(WITH m5 AS (SELECT H005.x105 x5,  row_number() over() as r5 FROM "H005" H005) SELECT x5 FROM m5 WHERE r5 =2956) q5,
(WITH m6 AS (SELECT H006.x105 x6,  row_number() over() as r6 FROM "H006" H006) SELECT x6 FROM m6 WHERE r6 =2956) q6,
(WITH m7 AS (SELECT H007.x105 x7,  row_number() over() as r7 FROM "H007" H007) SELECT x7 FROM m7 WHERE r7 =2956) q7,
(WITH m8 AS (SELECT H008.x105 x8,  row_number() over() as r8 FROM "H008" H008) SELECT x8 FROM m8 WHERE r8 =2956) q8,
(WITH m9 AS (SELECT H009.x105 x9,  row_number() over() as r9 FROM "H009" H009) SELECT x9 FROM m9 WHERE r9 =2956) q9,
(WITH m10 AS (SELECT H010.x105 x10,  row_number() over() as 10
  FROM "H010" H010) SELECT x10 FROM m10 WHERE r10 =2956) q10)
```

Below is a sample of the first query on PostGIS Raster (x=100, y=100, image_001 - image_010).

```
SELECT ST_Value(rast, 100, 100, 1) As val FROM postgis.all where rid>=1 and rid<=10;
```

Second Query

Below is a sample of the second query on SciDB 1st Model (x=100-150, y=100-150, image_001).

```
time iquery -q  "SELECT * FROM image_001 as i1
 WHERE i1.x>100 and i1.x<150 and i1.y>100 and i1.y<150"
```

Below is a sample of the second query on SciDB 2nd Model (x=100-150, y=100-150, image_001).

```
time iquery -q  "SELECT * FROM all as i
 WHERE i.x>100 and i.x<150 and i.y>100 and i.y<150 and i.z=1"
```

Below is a sample of the second query on PostgreSQL (x=100-150, y=100-150, image_001).

```
select
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 from
(WITH m1 AS(SELECT H001.x100 x100,H001.x101 x101,H001.x102
x102,H001.x103 x103,H001.x104 x104,H001.x105 x105,H001.x106
x106,H001.x107 x107,H001.x108 x108,H001.x109 x109,H001.x110
x110,H001.x111 x111,H001.x112 x112,H001.x113 x113,H001.x114
x114,H001.x115 x115,H001.x116 x116,H001.x117 x117,H001.x118
x118,H001.x119 x119,H001.x120 x120,H001.x121 x121,H001.x122
x122,H001.x123 x123,H001.x124 x124,H001.x125 x125,H001.x126
x126,H001.x127 x127,H001.x128 x128,H001.x129 x129,H001.x130
x130,H001.x131 x131,H001.x132 x132,H001.x133 x133,H001.x134
x134,H001.x135 x135,H001.x136 x136,H001.x137 x137,H001.x138
x138,H001.x139 x139,H001.x140 x140,H001.x141 x141,H001.x142
x142,H001.x143 x143,H001.x144 x144,H001.x145 x145,H001.x146
x146,H001.x147 x147,H001.x148 x148,H001.x149 x149,H001.x150
```

```
x150,   row_number() over() as r1 FROM "H001" H001) SELECT
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 FROM m1 WHERE r1 >100 and r1<150) q1
```

Below is a sample of the third query on PostGIS Raster (x=100-150, y=100-150, image_001 - image_010).

```
SELECT ST_Value(rast, 1, x, y) As val FROM postgis.all CROSS
JOIN    generate_series(100,150)    As    x    CROSS    JOIN
generate_series(100, 150) As y WHERE rid =  1;
```

Third Query

Below is a sample of the third query on SciDB 1st Model (x=100-150, y=100-150, image_001 - image_010).

```
time iquery -q  "SELECT *
FROM image_001 as i1, image_002 as i2, image_003 as i3,
image_004 as i4, image_005 as i5, image_006 as i6, image_007
as i7, image_008 as i8, image_009 as i9, image_010 as i10
 WHERE i1.x>100 and i1.x<150 and i1.y>100 and i1.y<150"
```

Below is a sample of the third query on SciDB 2nd Model (x=100-150, y=100-150, image_001 - image_010).

```
time iquery -q  "SELECT * FROM all as i
WHERE i.x>100 and i.x<150 and i.y>100 and i.y<150 and i.z>1
and i.z<10"
```

Below is a sample of the third query on PostgreSQL (x=100-150, y=100-150, image_001 - image_005).

```
select
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
```

```
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 from (WITH m1 AS (SELECT H001.x100 x100,H001.x101
x101,H001.x102 x102,H001.x103 x103,H001.x104 x104,H001.x105
x105,H001.x106 x106,H001.x107 x107,H001.x108 x108,H001.x109
x109,H001.x110 x110,H001.x111 x111,H001.x112 x112,H001.x113
x113,H001.x114 x114,H001.x115 x115,H001.x116 x116,H001.x117
x117,H001.x118 x118,H001.x119 x119,H001.x120 x120,H001.x121
x121,H001.x122 x122,H001.x123 x123,H001.x124 x124,H001.x125
x125,H001.x126 x126,H001.x127 x127,H001.x128 x128,H001.x129
x129,H001.x130 x130,H001.x131 x131,H001.x132 x132,H001.x133
x133,H001.x134 x134,H001.x135 x135,H001.x136 x136,H001.x137
x137,H001.x138 x138,H001.x139 x139,H001.x140 x140,H001.x141
x141,H001.x142 x142,H001.x143 x143,H001.x144 x144,H001.x145
x145,H001.x146 x146,H001.x147 x147,H001.x148 x148,H001.x149
x149,H001.x150 x150, row_number() over() as r1 FROM "H001"
H001)SELECT
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 FROM m1 WHERE r1 >100 and r1<150) q1

union

select
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 from (WITH m1 AS (SELECT H002.x100 x100,H002.x101
x101,H002.x102 x102,H002.x103 x103,H002.x104 x104,H002.x105
x105,H002.x106 x106,H002.x107 x107,H002.x108 x108,H002.x109
x109,H002.x110 x110,H002.x111 x111,H002.x112 x112,H002.x113
x113,H002.x114 x114,H002.x115 x115,H002.x116 x116,H002.x117
x117,H002.x118 x118,H002.x119 x119,H002.x120 x120,H002.x121
x121,H002.x122 x122,H002.x123 x123,H002.x124 x124,H002.x125
x125,H002.x126 x126,H002.x127 x127,H002.x128 x128,H002.x129
x129,H002.x130 x130,H002.x131 x131,H002.x132 x132,H002.x133
x133,H002.x134 x134,H002.x135 x135,H002.x136 x136,H002.x137
x137,H002.x138 x138,H002.x139 x139,H002.x140 x140,H002.x141
x141,H002.x142 x142,H002.x143 x143,H002.x144 x144,H002.x145
x145,H002.x146 x146,H002.x147 x147,H002.x148 x148,H002.x149
x149,H002.x150 x150, row_number() over() as r1 FROM "H002"
H002)SELECT
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
```

```
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 FROM m1 WHERE r1 >100 and r1<150) q1

union

select
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150  from  (WITH  m1  AS  (SELECT  H003.x100  x100,H003.x101
x101,H003.x102  x102,H003.x103  x103,H003.x104  x104,H003.x105
x105,H003.x106  x106,H003.x107  x107,H003.x108  x108,H003.x109
x109,H003.x110  x110,H003.x111  x111,H003.x112  x112,H003.x113
x113,H003.x114  x114,H003.x115  x115,H003.x116  x116,H003.x117
x117,H003.x118  x118,H003.x119  x119,H003.x120  x120,H003.x121
x121,H003.x122  x122,H003.x123  x123,H003.x124  x124,H003.x125
x125,H003.x126  x126,H003.x127  x127,H003.x128  x128,H003.x129
x129,H003.x130  x130,H003.x131  x131,H003.x132  x132,H003.x133
x133,H003.x134  x134,H003.x135  x135,H003.x136  x136,H003.x137
x137,H003.x138  x138,H003.x139  x139,H003.x140  x140,H003.x141
x141,H003.x142  x142,H003.x143  x143,H003.x144  x144,H003.x145
x145,H003.x146  x146,H003.x147  x147,H003.x148  x148,H003.x149
x149,H003.x150 x150,  row_number() over() as r1 FROM "H003"
H003)SELECT
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 FROM m1 WHERE r1 >100 and r1<150) q1

union

select
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150  from  (WITH  m1  AS  (SELECT  H004.x100  x100,H004.x101
x101,H004.x102  x102,H004.x103  x103,H004.x104  x104,H004.x105
x105,H004.x106  x106,H004.x107  x107,H004.x108  x108,H004.x109
x109,H004.x110  x110,H004.x111  x111,H004.x112  x112,H004.x113
x113,H004.x114  x114,H004.x115  x115,H004.x116  x116,H004.x117
x117,H004.x118  x118,H004.x119  x119,H004.x120  x120,H004.x121
x121,H004.x122  x122,H004.x123  x123,H004.x124  x124,H004.x125
x125,H004.x126  x126,H004.x127  x127,H004.x128  x128,H004.x129
x129,H004.x130  x130,H004.x131  x131,H004.x132  x132,H004.x133
x133,H004.x134  x134,H004.x135  x135,H004.x136  x136,H004.x137
x137,H004.x138  x138,H004.x139  x139,H004.x140  x140,H004.x141
```

```
x141,H004.x142  x142,H004.x143  x143,H004.x144  x144,H004.x145
x145,H004.x146  x146,H004.x147  x147,H004.x148  x148,H004.x149
x149,H004.x150 x150,  row_number() over() as r1 FROM "H004"
H004)SELECT
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 FROM m1 WHERE r1 >100 and r1<150) q1

union

select
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 from (WITH m1 AS (SELECT H005.x100 x100,H005.x101
x101,H005.x102  x102,H005.x103  x103,H005.x104  x104,H005.x105
x105,H005.x106  x106,H005.x107  x107,H005.x108  x108,H005.x109
x109,H005.x110  x110,H005.x111  x111,H005.x112  x112,H005.x113
x113,H005.x114  x114,H005.x115  x115,H005.x116  x116,H005.x117
x117,H005.x118  x118,H005.x119  x119,H005.x120  x120,H005.x121
x121,H005.x122  x122,H005.x123  x123,H005.x124  x124,H005.x125
x125,H005.x126  x126,H005.x127  x127,H005.x128  x128,H005.x129
x129,H005.x130  x130,H005.x131  x131,H005.x132  x132,H005.x133
x133,H005.x134  x134,H005.x135  x135,H005.x136  x136,H005.x137
x137,H005.x138  x138,H005.x139  x139,H005.x140  x140,H005.x141
x141,H005.x142  x142,H005.x143  x143,H005.x144  x144,H005.x145
x145,H005.x146  x146,H005.x147  x147,H005.x148  x148,H005.x149
x149,H005.x150 x150,  row_number() over() as r1 FROM "H005"
H005)SELECT
x100,x101,x102,x103,x104,x105,x106,x107,x108,x109,x110,x111,x1
12,x113,x114,x115,x116,x117,x118,x119,x120,x121,x122,x123,x124
,x125,x126,x127,x128,x129,x130,x131,x132,x133,x134,x135,x136,x
137,x138,x139,x140,x141,x142,x143,x144,x145,x146,x147,x148,x14
9,x150 FROM m1 WHERE r1 >100 and r1<150) q1
```

Below is a sample of the third query on PostGIS Raster (x=100-150, y=100-150, image_001 - image_010).

```
SELECT ST_Value(rast, 1, x, y) As val    FROM test.all CROSS
JOIN    generate_series(100,150)    As    x    CROSS    JOIN
generate_series(100, 150) As y WHERE rid>=1 and rid<=10;
```

References

Azadeh Ghiyamata and Helmi Z.M. Shafria. 2010. A review on hyperspectral remote sensing for homogeneous and heterogeneous forest biodiversity assessment. *Int. J. Remote Sens.* 31, 7 (2010).

P. Baumann. 2001. Web-enabled raster GIS services for large image and map databases. *12th Int. Work. Database Expert Syst. Appl.* (2001), 870–874. DOI:http://dx.doi.org/10.1109/DEXA.2001.953165

Peter Baumann and Sönke Holsten. 2012. A Comparative Analysis of Array Models for Databases. *Int. J. Database Theory Appl.* 5, 1 (2012), 89–120. DOI:http://dx.doi.org/10.1007/978-3-642-27157-1_9

William F. Belokon. 1997. *Multispectral Imagery Reference Guide*, Logicon Geodynamics, Incorporated.

J.L. Boggs, T.D. Tsegaye, T.L. Coleman, K.C. Reddy, and Ahmed Fahsi. 2003. Relationship Between Hyperspectral Reflectance, Soil Nitrate-Nitrogen, Cotton Leaf Chlorophyll, and Cotton Yield: A Step Toward Precision Agriculture. *J. Sustain. Agric.* 22, 3 (2003), 5–16. DOI:http://dx.doi.org/10.1300/J064v22n03_03

Ayan Chakrabarti and Todd Zickler. 2011. Statistics of real-world hyperspectral images. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (2011), 193–200. DOI:http://dx.doi.org/10.1109/CVPR.2011.5995660

E.A. Cloutis. 1996. Review Article Hyperspectral geological remote sensing: evaluation of analytical techniques. *Int. J. Remote Sens.* 17, 12 (1996), 2215–2242. DOI:http://dx.doi.org/10.1080/01431169608948770

M.A. Cochrane. 2000. Using vegetation reflectance variability for species level classification of hyperspectral data. *Int. J. Remote Sens.* 21, 10 (2000).

P. Cudre-Mauroux et al. 2009. A Demonstration of SciDB: A Science-Oriented DBMS. *Proc. VLDB Endow.* 2, 1 (2009), 1534–1537. DOI:http://dx.doi.org/10.14778/1687553.1687584

Philippe Cudre-mauroux et al. 2011. SS-DB : A Standard Science DBMS Benchmark. *Byte* (2011).

M. Fauvel, Y. Tarabalka, J.A. Benediktsson, J. Chanussot, and J.C. Tilton. 2013. Advances in Spectral-Spatial Classification of Hyperspectral Images. *Proc. IEEE* 101, 3 (2013), 652–675. DOI:http://dx.doi.org/10.1109/JPROC.2012.2197589

Caitlin a. Griffith et al. 2012. Possible tropical lakes on Titan from observations of dark terrain. *Nature* 486, 7402 (2012), 237–239. DOI:http://dx.doi.org/10.1038/nature11165

Angelica Garcia Gutierrez and Peter Baumann. 2007. Modeling fundamental geo-raster operations with array algebra. *Proc. - IEEE Int. Conf. Data Mining, ICDM* , 7 (2007), 607–612. DOI:http://dx.doi.org/10.1109/ICDMW.2007.53

SciDB Inc. 2012. *SciDB User' s Guide Version 12.3*,

Y. Lanthier, A. Bannari, D. Haboudane, J.R. Miller, and N. Tremblay. Hyperspectral Data Segmentation and Classification in Precision Agriculture : a Multi-Scale Analysis. *Environment*, 4–5.

Bryan W. Lewis. 2014. The scidb Package. (2014), 1–26.

Feng Liu et al. 2014. GPU Accelerated Array Queries: The Good , the Bad , and the Promising. *HP Lab.* (2014).

Haicheng Liu. 2014. Comparing NetCDF and a multidimensional array database on managing and querying large hydrologic datasets : a case study of SciDB. , September (2014).

Dimitris Manolakis and Gary Shaw. 2002. *Detection algorithms for hyperspectral imaging applications*,

Karol Myszkowski, Wolfgang Heidrich, Michael Goesele, Bernd Hofflinger, Grzegorz Krawczyk, and Matthew Trentacoste. 2005. *High Dynamic Range Techniques in Graphics : from Acquisition to Display*,

Virginia E. Ogle and Michael Stonebraker. 1995. Chabot: retrieval from a relational database of images. *Computer (Long. Beach. Calif).* 28, 9 (1995), 40–48.

DOI:http://dx.doi.org/10.1109/2.410150

Gary Lee Planthaber, Computer Science, Computer Science, Michael R. Stonebraker, Computer Science, and Thesis Supervisor. 2012. MODBASE: A SciDB-Powered System for Large-Scale Distributed Storage and Analysis of MODIS Earth Remote Sensing Data. (2012).

Antonio Plaza et al. 2009. Recent advances in techniques for hyperspectral image processing. *Remote Sens. Environ.* 113 (2009), S110–S122. DOI:http://dx.doi.org/10.1016/j.rse.2007.07.028

PostGIS. 2014. PostGIS 2.1.2dev Manual SVN Revision (12349). , 12349 (2014).

Xie Qingyun, Xu Weisheng, and Siva Ravada. 2007. GEORASTER PHYSICAL DATA MODEL FOR STORING GEOREFERENCED RASTER DATA. 2, 12 (2007).

Lavanya Ramakrishnan, Pradeep K. Mantha, Yushu Yao, and Richard S. Canon. 2013. Evaluation of NoSQL and Array Databases for Scientific Applications. (2013).

Ingmar Renhorn et al. 2012. Detection in urban scenario using combined airborne imaging sensors. 8353, c (2012), 83530I–83530I–9. DOI:http://dx.doi.org/10.1117/12.921473

Niles Ritter, Mike Ruth, and …. 1995. ►*GeoTIFF format specification GeoTIFF revision 1.0*,

Thomas Schmid, Magaly Koch, and Jose Gumuzzio. 2005. Multisensor approach to determine changes of wetland characteristics in semiarid environments (Central Spain). *IEEE Trans. Geosci. Remote Sens.* 43, 11 (2005), 2516–2525. DOI:http://dx.doi.org/10.1109/TGRS.2005.852082

J. Sevilla and a Plaza. 2014. A New Digital Repository for Hyperspectral Imagery With Unmixing-Based Retrieval Functionality Implemented on GPUs. *Sel. Top. Appl. Earth Obs. Remote Sensing, IEEE J.* PP, 99 (2014), 1. DOI:http://dx.doi.org/10.1109/JSTARS.2014.2314601

Peg Shippert. 2003. Introduction to hyperspectral image analysis. *Online J. Sp. Commun.* (2003).

Randall Smith. 2006. Hyperspectral Imaging. *MicroImages, Inc.* (2006), 1–24. DOI:http://dx.doi.org/10.1016/j.quaint.2007.05.011

Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman. 2001. The Architecture of SciDB. 167 (2001), 138–167. DOI:http://dx.doi.org/10.5465/AMR.1993.3997509

Michael Stonebraker, Jennie Duggan, Leilani Battle, and Olga Papaemmanouil. 2013. SciDB DBMS Research at MIT. (2013), 1–10.

Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland. 2007. The End of an Architectural Era (It's Time for a Complete Rewrite). *Vldb* 12, 2 (2007), 1150–1160. DOI:http://dx.doi.org/10.1080/13264820701730900

Yuliya Tarabalka, Mathieu Fauvel, Jocelyn Chanussot, and Jón Atli Benediktsson. 2010. SVM-and MRF-based method for accurate classification of hyperspectral images. *IEEE Geosci. Remote Sens. Lett.* 7, 4 (2010), 736–740. DOI:http://dx.doi.org/10.1109/LGRS.2010.2047711

Inc The MathWorks. 1999. *MATLAB The Language of Technical Computing. Mat-File Format*,

Prasad S. Thenkabail, Ronald B. Smith, Eddy De Pauw, and E. De Pauw. 2002. Evaluation of Narrowband and Broadband Vegetation Indices for Determining Optimal Hyperspectral Wavebands for Agricultural Crop Characterization. *Photogramm. Eng. Remote Sensing* 68, 6 (2002), 607–621. DOI:http://dx.doi.org/0099-111210216806-60

Masters Program in **Geospatial Technologies**

Supported by:
Education and Culture
ERASMUS MUNDUS