Geographical Delocalization of FITS Data in MongoDB Environment

Bruno Luigi Martino* [1], **Fabio Guglietta**[2], **Francesco Reale**[3], **Giorgio Patria**[4]

[1] *CNR-IASI: Istituto di Analisi dei Sistemi ed Informatica*
Via dei Taurini 19, 00185 Roma, Italy
bruno.martino@iasi.cnr.it

[2] *CNR-IASI: Istituto di Analisi dei Sistemi ed Informatica*
Via dei Taurini 19, 00185 Roma, Italy
fabio.gu@gmail.com

[3] *CNR-ISC: Istituto per i Sistemi Complessi*
Via Fosso del Cavaliere 100, 00133 Roma, Italy
francesco.reale@sic.rm.cnr.it

[4] *ITIS G. Galilei*
Via Conte Verde, 51, 00185 Roma, Italy
giorgio.patria@gmail.com

The astrophysics community prefers to use the FITS format for storage of data of interest. This article compares this classic approach and one related to the use of non-relational databases highlighting their many similarities. Among the main advantages of using non-relational database stands out the possibility of moving and replicate automatically your data minimizing the risks related to their loss or corruption. Once you realize that a header FITS is nothing more than a sequence of key-value pairs, the jump to the NoSQL world becomes almost natural.

Accretion Processes in Cosmic Sources - APCS2016 -
5-10 September 2016,
Saint Petersburg, Russia

*Speaker.

1. The FITS format

Flexible Image Transport System (FITS) [1] is an open standard defining a digital file format useful for storage, transmission and processing of scientific data and images. FITS is the most commonly used digital file format in astronomy. Unlike many image formats, FITS is designed specifically for scientific data and hence includes many provisions for describing photometric and spatial calibration information, together with image origin metadata. A major feature of the FITS format is that image metadata is stored in a human-readable ASCII header, so that an interested user can examine the headers to investigate a file of unknown origin. The information in the header is designed to calculate the byte offset of some information in the subsequent data unit to support direct access to the data cells. Each FITS file consists of one or more headers containing ASCII card images (80 character fixed-length strings) that carry keyword/value pairs. FITS support is available in a variety of programming languages used for scientific work, including C, C++, C#, Fortran, IDL, Java, LabVIEW, Mathematica, MATLAB, Perl, Python, R, and other. Image processing programs such as ImageJ, GIMP, Photoshop, etc. can generally read simple FITS images, but frequently cannot interpret more complex tables and databases. The FITS Liberator software [2] is used by imaging scientists at the European Space Agency, the European Southern Observatory and NASA.

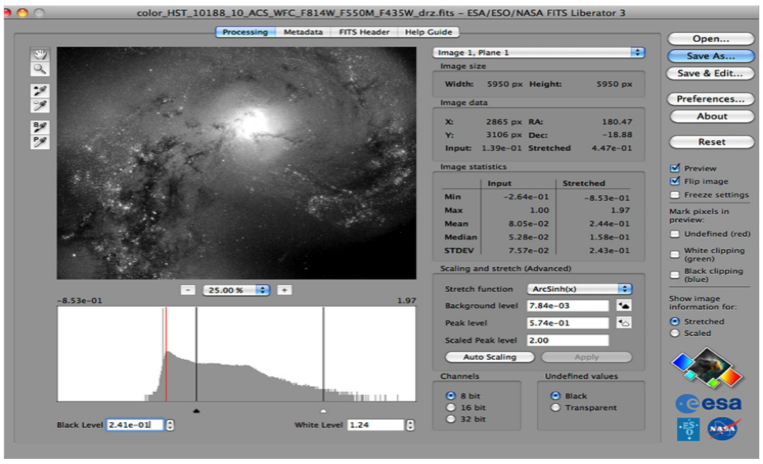


Image 1, Plane 1

Image size

Width: 5950 px Height: 5950 px

Image data

X: 2865 px RA: 180.47
Y: 3106 px Dec: -18.88
Input: 1.39e-01 Stretched 4.47e-01

Image statistics

	Input	Stretched
Min	-2.64e-01	-8.53e-01
Max	1.00	1.97
Mean	8.05e-02	2.44e-01
Median	5.28e-02	1.58e-01
STDEV	7.57e-02	2.43e-01

Scaling and stretch (Advanced)

Stretch function: ArcSinh(x)
Background level: 7.84e-03
Peak level: 5.74e-01
Scaled Peak level: 2.00

Channels: [✓ 8 bit] 16 bit 32 bit

Undefined values: [✓ Black] Transparent

25.00 %

-8.53e-01 1.97

Black Level 2.41e-01 White Level 1.24

✓ Preview
✓ Flip image
☐ Freeze settings

Mark pixels in preview:
☐ Undefined (red)
☐ White clipping (green)
☐ Black clipping (blue)

Show image information for: [✓ Stretched] Scaled

Figure 1: FITS liberator screenshot

2. NoSQL and MongoDB

NoSQL is a database technology [3] designed to support the requirements of cloud applications and architected to overcome the scale, performance, data, and data distribution limitations of relational databases (RDBMS's). A NoSQL database typically offers the following benefits over other database management systems:

- Continuously Available: A database that stays online even in the face of the most devastating infrastructure outages.

- Geographically Distributed: active data, everywhere you need it.
- Operationally Low Latency: Response times fast enough for your most intense operational applications.
- Linearly Scalable: Predictably scale to meet the current and future data needs of cloud applications.
- Operationally Mature: Enterprise-ready data management for cloud applications.
- Low TCO: No requirements for specialized hardware or ancillary software.

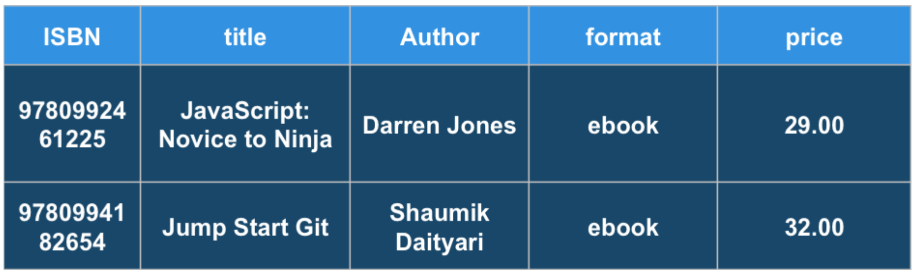
The primary way in which non-relational databases differ from relational databases is the data model. Although there are dozens of non-relational databases, they primarily fall into one of the following three categories:

- Document Models
- Graph Models
- Key-Value and Wide Column Models

We are going to investigate on the Document Models

3. NoSQL data model: Document Model

Whereas relational databases store data in rows and columns, document databases store data in documents. These documents typically use a structure that is like JSON (JavaScript Object Notation). Documents provide an intuitive way to model data aligned with object-oriented programming; each document is effectively an object. Documents contain one or more fields, where each field contains a typed value, such as a string, date, binary, or array. For example, if you run an online book store sw, the informations can be added to a table named "book" as shown in figure 2:

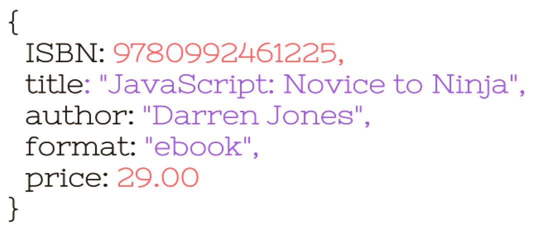


ISBN	title	Author	format	price
97809924 61225	JavaScript: Novice to Ninja	Darren Jones	ebook	29.00
97809941 82654	Jump Start Git	Shaumik Daityari	ebook	32.00

Figure 2: "book" table

Every row is a different book record. The design is rigid; you cannot use the same table to store different informations or insert a string where a number is expected.

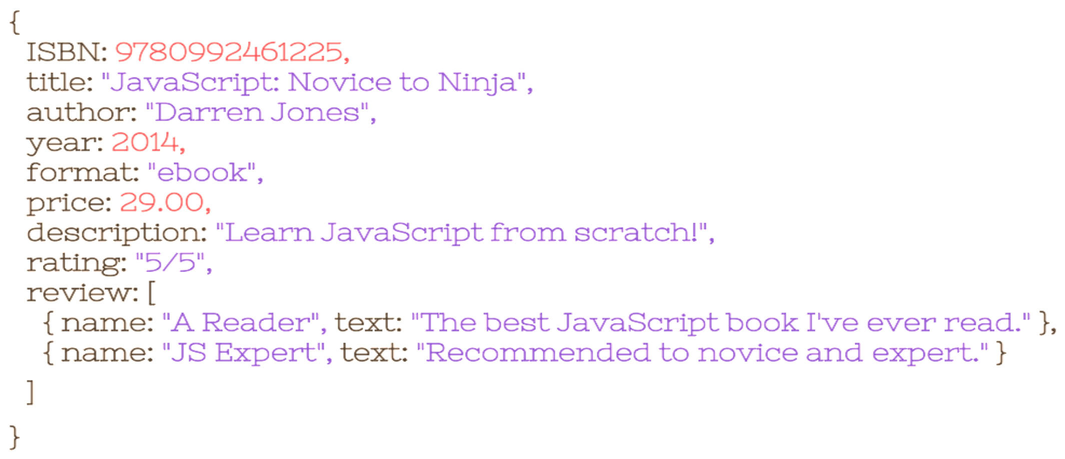
A noSQL DB typically stores data in JSON-like field-value pair documents, (e.g. figure 3): Similar documents can be stored in a collection, which is analogous to an SQL table.



```
{
  ISBN: 9780992461225,
  title: "JavaScript: Novice to Ninja",
  author: "Darren Jones",
  format: "ebook",
  price: 29.00
}
```

Figure 3: JSON-like field-value pair document

A property of NoSQL database is to be free of pattern (schemaless) [4], with consequent advantages and disadvantages like ease of deployment, but sometimes, more difficulties to construct complex queries. *Then in the same collection*, you can store the data as you like in any document (figure 4):



```
{
  ISBN: 9780992461225,
  title: "JavaScript: Novice to Ninja",
  author: "Darren Jones",
  year: 2014,
  format: "ebook",
  price: 29.00,
  description: "Learn JavaScript from scratch!",
  rating: "5/5",
  review: [
    { name: "A Reader", text: "The best JavaScript book I've ever read." },
    { name: "JS Expert", text: "Recommended to novice and expert." }
  ]
}
```

Figure 4: JSON-like field-value pair with a different pattern

4. MongoDB scalability by sharding

Concerning the scalability and looking for an affordable solution we tested MongoDB because it provides horizontal scale-out for databases, using a technique named sharding, which is transparent to applications. Sharding distributes data across multiple physical partitions called shards. Sharding allows MongoDB deployments to address the hardware limitations of a single server, such as bottlenecks in RAM or disk I/O, without adding complexity to the application. Unlike relational databases, sharding is automatic and built into the database. Operations teams don't need to deploy additional clustering software or expensive shared-disk infrastructure to manage process and data distribution or failure recovery [5]. Moreover, multiple sharding policies are available to enable developers and administrators to distribute data across a cluster according to query patterns or data locality. As a result, we can achieve much higher scalability across a diverse set of workloads:

- Range-based Sharding. Documents are partitioned across shards according to the shard key patterns or data locality. This approach is well suited for applications that need to optimize range-based queries

- Hash-based Sharding. Documents are uniformly distributed. This approach guarantees a uniform distribution of writes across shards, but is less optimal for range-based queries
- Location-aware Sharding. Documents are partitioned according to a user-specified configuration that associates shard key ranges with specific shards and hardware. Users can continuously refine the physical location of documents for application requirements such as locating data in specific institute data center or on different storage types (optimized for big storage or, for frequent queries and so on...)

5. Continous Data Availability

A replica set is a group of mongoDB demons (mongod) instances that maintain the same data set. and contains several data bearing nodes and optionally one arbiter node. Of the data bearing nodes, one and only one member is deemed the primary node, while the other nodes are deemed secondary nodes [6]. The primary node receives all write operations. A replica set can have only one primary capable of confirming writes with w: "majority" write concern.

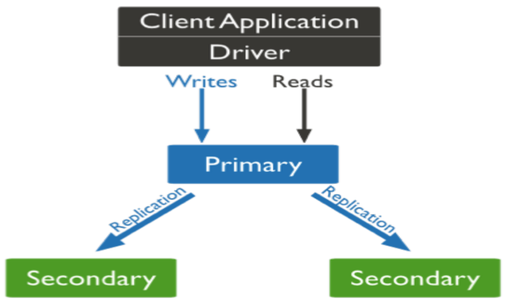


Figure 5: Primary node behavior

The secondaries replicate the primary's Operations Log (oplog) and apply the operations to their data sets such that the secondary data sets reflect the set of the primary.

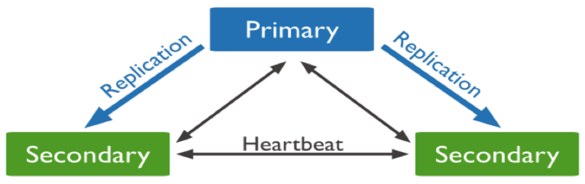


Figure 6: Data replication

If the primary node became unavailable, an eligible secondary will hold an election to elect itself the new primary.

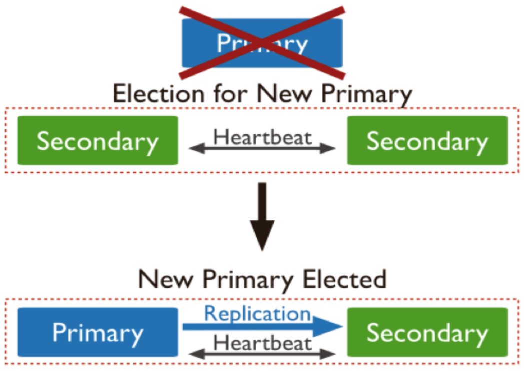


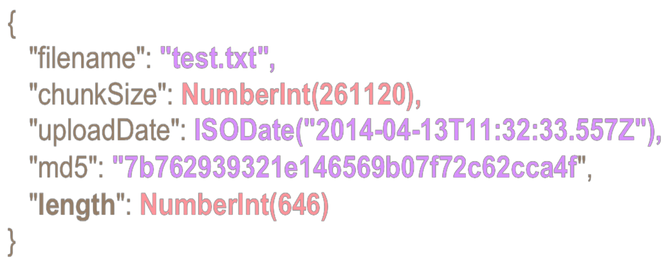
Figure 7: New primary election

Combining Sharding and Replica Set you can obtain the followiong features:

- Endless horizontal scalability
- Automagically store the data spread them geographically where there are mainly accessed (but they can be stored as replica where you like more)
- Continous Data Availability

6. MongoDB Filesystem

You'll notice that the header of a FITS file is in the key-value format, exactly like a MongoDB document; then these formats are very close relatives! You may have heard about the 16 MB document size limit. What this means is that everything is working just fine, your system is screaming fast, until you try to create a document that's 16.001 MB and ... nothing works any more. A good solution is to store your data in a series of documents (let's call them "chunks") of limited size. Then you can tie them all together with another document that specifies all the file metadata. That's exactly what GridFS does [7]. GridFS is the MongoDB specification for storing and retrieving large files. It is kind of a file system to store files but its data is stored within MongoDB collections.



```
{
  "filename": "test.txt",
  "chunkSize": NumberInt(261120),
  "uploadDate": ISODate("2014-04-13T11:32:33.557Z"),
  "md5": "7b762939321e146569b07f72c62cca4f",
  "length": NumberInt(646)
}
```

Figure 8: A typical fs.files document

GridFS by default uses two collections (*fs.files* and *fs.chunks*) to store the file's metadata and the chunks. Each chunk is identified by its unique *_ObjectId* field. The *fs.files* serves as a parent document. The *files_id* field in the *fs.chunks* document links the chunk to its parent. Now for example, we will store an image file using GridFS using the put command. For this we will use the *mongofiles* utility:

mongofiles -d gridfs put image.jpeg

Here, gridfs is the name of the database in which the file will be stored. If the database is not present, MongoDB will automatically create a new document on the fly; image.jpeg is the name of the file uploaded.

7. Putting all together

The Astropy Project [8] is a collection of software packages written in the Python programming language and designed for use in astronomy. The software is a single, free, core package for astronomical users. In the myriad of things it does in astronomy, what interests us is the *astropy.io.fits* package and its API. Using this package we can open, read and create FITS files quite easily. We wrote a very small (and lightweight) app in python which performs the following tasks: scans the directories where FITS files are stored, reads FITS files one by one and generates as output a structure like this:

- FITS file name
- FITS file position inside the FS
- metadata obtained by headers in key-value format
- metadata that we want to eventually add

The structure so created (or a file or whatever we want) is then received and processed by python's MongoDB driver (pymongo). The driver allows to upload the files in MongoDB GridFS; when a file is uploaded, it is automatically broken up into chunks (standard size is 255k). Uploading files with pymongo, in a distributed and sharded environment, you talk with a demon called Mongos (MongoShard) which, in a transparent fashion:

- Select, according to the indices, on which physical machine and in which geographical location the file will be loaded.
- it breaks it up and distributes among the shard
- replicates all the shard in the replication set

We did our testing using INTEGRAL / IBIS data. The INTEGRAL data are analyzed using the OSA package distributed by ISDC (Geneva). OSA is written using various programming languages and it heavily uses the FITS format. In OSA FITS files are handled by the FITSIO library; this library known only the standard FITS format.

8. Conclusions

Our goal is now to write stub procedures to allow the use of the same functions calls of the library FITSIO replacing, where needed, file system I/O operations with non-relational database MongoDB accesses. This activity will allow us to migrate through a simple recompilation of the software already written and tested towards the proposed approach for an efficient and reliable management of astronomical data. Summarizing, the proposed approach:

- allows fast access and efficient management of data
- protects against hardware failures and data corruption
- does not require complex directory hierarchies
- simplifies the synchronization of data from the central repository
- does not require to know exactly the location of the working files
- teams that deals with an instrument can have their local data set, increasing processing speed

Acknowledgments

Pietro Ubertini, Franco Giovannelli, Dmitry Bisikalo and the lovely russian Organizing Committee. A very special thanks to our friend Memmo Federici, who has never really left us but it is always with us. Thank you Memmo!

References

[1] Pence, W. D. et al., *Definition of the Flexible Image Transport System (FITS), version 3.0. A&A 524, A42*, [2010]

[2] Nielsen, L. H. et al., *The ESA/ESO/NASA Photoshop FITS Liberator v.3: Have your Say on New Features. Communicating Astronomy with the Public, Athens (October 8-12)*, [2007]

[3] Kanwar, R. et al., *NoSQL, a Solution for Distributed Database Management System. International Journal of Computer Applications (0975 âĂŞ 8887) Volume 67âĂŞ No.2*, [2013]

[4] Martino B. L., Federici M., *Non Relational Models for the Management of Large Amount of Astronomical Data, Acta Polytechnica CTU Proceedings 2(1): 293?296,*, [2015]

[5] Chodorow, K. et al., *Scaling MongoDB, O' Reilly Media,*, [2011]

[6] Yong-Shin, K. et al., *MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data, IEEE sensors journal, vol. 16, NO. 2,*, [2016]

[7] Yunhua, G. et al., *Analysis of Data Storage Mechanism in NoSQL Database MongoDB, International Conference on Consumer Electronics-Taiwan,*, [2015]

[8] Robitaille, T. P. et al., *Astropy: A Community Python Package for Astronomy Astronomy & Astrophysics manuscript no. astropy* , [2013]