it's about time

Technical Whitepaper

Kdb+ in Astronomy

Author:

Andrew Magowan is a kdb+ consultant who has developed data and analytic systems for some of the world's largest financial institutions. Andrew is currently based in New York where he maintains a global tick capture application across a range of asset classes at a major investment bank.

James Neill works as a kdb+ consultant for one of the world's largest investment banks developing a range of applications. James has also been involved in the design of training courses in data science and machine learning as part of the First Derivatives Capital Markets Training Programme.

Content

1. Introduction 1

2. Astronomy Data 2

2.1 ESA Gaia 2

2.2 Sloan Digital Sky Survey (SDSS) 2

2.3 FITS Files 2

3. Processing the data – linking FITS and kdb+ using C 3

3.1 The Shared Object 3

3.2 Importing table data to kdb+ 5

3.3 Building tables in kdb+ 7

3.4 Calculating Recessional Velocity 8

4. Conclusion 11

1. Introduction

The field of Observational Astronomy has always been data driven, but like many other fields, technological advancements are causing something of a paradigm shift, and - according to experts – a bit of a headache! Currently under construction are new infrastructures that have the potential to record volumes of data that have not been seen before in the field. The Large Synoptic Survey Telescope (LSST) and Square Kilometer Array (SKA) are set to record such huge amounts of data that experts are concerned about their ability to make sense of this data, purely due to its sheer size.

The LSST is expected to be fully operational and start recording data in 2021, producing 15TB every night. It will take an image of half the sky every 3 nights [1], and so the same objects will be photographed again and again. This gives the data from the LSST a time dimension - how an object moves over time is something that will be studied from this data. In the 2020s, the SKA will produce 160TB of data per second [2]. This is in the region of exabytes per night, and zettabytes per year [3]. If this data were processed in real time, any large differences in objects (e.g. brightness/position) could then be investigated immediately. Use cases of the data on a non-real time basis include discovering the formation and structure of our solar system, investigating distant galaxies, and the evolution of the universe.

There is no single choice of programming language in astronomy, but C is used for many astronomical applications. C is one of the most popular and commonly used programming languages in the world with a wide range of uses varying from powering operating systems to building application software. Kdb+ has the ability to extend its functionality through dynamically loaded C/C++ modules, so we have the ability both to make use of existing utilities, and to create our own.

We believe that due to the amount of data collected, its time series nature, and the potential need for both real-time and historical based analysis, kdb+ would be very well suited to the data collected in the astronomy industry, and would be an ideal fit for many future astronomy projects.

This paper serves to provide an example of how this data collected in astronomy can be processed using the power and speed of kdb+. With future projects collecting data at increasing scales, this is an ideal time for the power of kdb+ to be applied to the field of astronomy. To demonstrate, we have loaded some raw astronomical data in to kdb+. In addition, we make use of kdb+'s versatility in extending with C to show how it takes a relatively small amount of q code to load some raw astronomical data in to kdb+ and run some simple analysis on the dataset.

2. Astronomy Data

2.1 ESA Gaia

The European Space Agency’s (ESA) Gaia project is a mission to chart a three-dimensional map of the Milky Way galaxy. Gaia is a billion-pixel camera that will take images of the galaxy to eventually record over one billion stars across the next five years, with also the anticipation of making other discoveries along the way. It recently released its first set of data to the public in the form of FITS files. This can be downloaded from ESA’s website at http://gea.esac.esa.int/archive/.

2.2 Sloan Digital Sky Survey (SDSS)

The Sloan Digital Sky Survey (SDSS) - based in New Mexico - began surveying the sky in 2000. It covered roughly one third of the sky, and observed around 500 million objects. It has produced more than 100TB of data, of which all is available free to the public through its website at http://www.sdss.org. This has resulted in the SDSS often being described as the project that genuinely brought astronomy in to big data territory, and so its significance in this field cannot be underestimated. The data can be accessed through SQL queries, or through the raw FITS files that are also publicly available. We describe FITS files below.

2.3 FITS Files

Flexible Image Transport System (FITS) is a digital file format used for storing scientific data. It is the format most widely used by astronomers and astrophysicists for transporting, analyzing and archiving scientific data since the early 1980s. The data in the SDSS database is archived in FITS files. These files are primarily designed to store images in the form of multidimensional numerical arrays or data in the form of tables. One FITS file can contain multiple tables and images.

FITS files consist of segments called Header Data Units (HDUs). Every FITS file has a primary HDU which is usually an image and optional extension HDUs which may contain images or tables. Each HDU contains a Header Unit and a Data Unit. The Header Unit contains meta-data pertaining to the information stored in the Data Unit and the Data Unit is the image or table referenced by the meta-data. A full description of FITS data can be found at http://fits.gsfc.nasa.gov/fits_primer.html.

3. Processing the Data – linking FITS and kdb+ using C

Kx provide a header file, k.h, for interacting with C from kdb+. It provides the link between kdb+ and C by converting the different data types and structures between the two languages. Using this header file, we created a shared object that could natively parse a FITS file and load the data into a kdb+ database. This C extension can read meta-data in HDUs and extract columns from binary tables, converting the information into a kdb+ usable format. The C functions are loaded into kdb+ from the shared object by using the `2:` function, which is described in more detail later.

Please see James Neill's GitHub Repository for kdb+ in Astronomy here https://github.com/jpneill/fitsToKdb

3.1 The Shared Object

There are 5 functions for extracting meta-data about the FITS file and the tables. Each of these takes in one or more K objects from a q process and returns a K object to that process.

```
 // Function which prints a list of the number of HDUs and their types

 // x – q symbol that is the FITS file `$”example.fits”

 K listHDUs(K x)


 // Function which returns number of rows in a table from a FITS file

 // x - q symbol that is the FITS file `$"example.fits"

 // y – q int or long representing the HDU of the binary table

K num_rows(K x, K y)


 // Function which returns number of columns in a table from a FITS file

 // x - q symbol that is the FITS file `$"example.fits"

 // y – q int or long representing the HDU of the binary table

K num_cols(K x, K y)

 // Function which returns a list of columns in a binary table in a FITS
file

 // x - q symbol that is the FITS file `$"example.fits"

 // y - q int or long representing the HDU we're looking for

K cols(K x, K y)
```

```
 // Function which returns a column type from a binary table in a FITS
file

 // x - q symbol for name of FITS file

 // y - q symbol for column name

 // z - q int or long representing the HDU we're looking at

 K getColType(K x, K y, K z)
```

There are 4 functions for extracting columns of different types:

```
// All of the following take in the same parameters:

// x - q symbol for FITS file name

// y - q symbol for column name

// z - q long for number of rows to extract

// h - q int or long representing the number of the HDU we're looking at

K readLongCol(K x,K y,K z,K h)

K readIntCol(K x,K y,K z,K h)

K readDoubleCol(K x,K y,K z,K h)

K readSymCol(K x,K y,K z,K h)
```

3.2 Importing table data to kdb+

In kdb+, the 2: operator is a dyadic function used to dynamically load in C functions from a shared object. The left argument is the library from which the function is loaded (of type symbol), and the right argument is a list containing the function name (type symbol), and the number of arguments that function takes (type integer). Dynamically loaded functions have the datatype value 112h. Below is an example of how the `listHDUs` function is loaded:

```
q).astro.listHDUs:`fitsToKdb 2:(`listHDUs;1);
```

Once the function is loaded into the q process it can be called in the same way as any q function:

```
q)file:`$"specObj-dr12.fits"

q).astro.listHDUs[file]

Number of HDUs: 2

HDU 1: IMAGE_HDU

HDU 2: BINARY_TBL
```

The other functions can then be loaded in in the same manner.

```
q).astro.getFitsRowCount:`fitsToKdb 2:(`num_rows;2);

q).astro.getFitsColCount:`fitsToKdb 2:(`num_cols;2);

q).astro.getFitsColNames:`fitsToKdb 2:(`cols;2);

q).astro.getFitsColType:`fitsToKdb 2:(`getColType;3);

q).astro.readLongCol:`fitsToKdb 2:(`readLongCol;4);

q).astro.readDoubleCol:`fitsToKdb 2:(`readDoubleCol;4);

q).astro.readIntCol:`fitsToKdb 2:(`readIntCol;4);

q).astro.readSymCol:`fitsToKdb 2:(`readSymCol;4);
```

The source file for these example analytics is the specObj-dr12.fits file from the SDSS database, found at http://www.sdss.org/dr12/spectro/spectro_access. This is a 2.9GB file containing the redshifts and classifications of all 4 million+ objects observed, including galaxy, quasar, and stellar spectra. We are able to use the functions defined above to create a kdb+ database from this FITS format file.

```
q).astro.getFitsRowCount[file;2]

4355200i

q)c:.astro.getFitsColNames[file;2] // columns in table

q)t:.astro.getFitsColType[file;;2]each c // types of columns

q)n:1000000 // number of rows to extract from the table

q)icols:c where t=`I // get only the int columns

// build a dictionary containing the int columns

q)icols:lower[icols]!.astro.readIntCol[file;;n;2]each icols

// repeat for each of the long, float and sym cols
```

For more information on extending kdb+ with C/C++ please visit http://code.kx.com/wiki/Cookbook/ExtendingWithC

3.3 Building tables in kdb+

Once the data is in dictionary form, combine and build the table:

```
q)specObj:flip raze(icols;jcols;fcols;scols)

q)specObj

nspecobs spectrographid bluefiber nturnoff boss_specobj_id ..

--------------------------------------------------------------

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

2        1              -1        -1       0               -1

3        1              -1        -1       0               -1

..

q)count specObj

1000000
```

At this point, we have successfully loaded the data from a FITS file in to an in-memory kdb+ table. From here we can run some sample queries to take a look at the data and what it contains.

Obtaining a breakdown by class of the data:

```
q)select count i by class from specObj
class | x
------| ------
GALAXY| 678098
QSO   | 112701
STAR  | 209201
// Time(ms) and memory(bytes) taken to compute across 1m rows
q)\ts select count i by class from specObj
18 16777872
```

3.4 Calculating recessional velocity

Calculating the recessional velocity (recessional velocity is the rate at which an object is moving away from Earth) of each object using the observed redshift (z column):

```
q)rv:{[z](z-1)%1+z*:z+:1}
q)rv 1.6
0.742268
q)select class,subclass,plug_ra,plug_dec,z,recVel:rv z from specObj
class  subclass    plug_ra  plug_dec   z            recVel
--------------------------------------------------------------
GALAXY             146.7142 -1.041304  0.02127545   0.02104918
GALAXY             146.9195 -0.9904918 0.2139246    0.1914661
QSO    BROADLINE   146.9023 -0.9849133 0.6521814    0.4637643
GALAXY             146.8598 -0.8089017 0.1265536    0.1186022
..
```

```
// Time(ms) and memory(bytes) taken to compute above query

 q)\ts select class,subclass,plug_ra,plug_dec,z,recVel:rv z from
specObj

 30 33555104

 // breakdown of average recessional velocity by class

 q)select recVel:avg rv z by class from specObj

 class | recVel

 ------| -------------

 GALAXY| 0.1374245

 QSO   | 0.6371058

 STAR  | -6.876509e-05

 q)\ts select recVel:avg rv z by class from specObj

 53 53478144
```

The kdb+ function fby aggregates values from one list based on groups in another list. It is commonly used to extend the functionality of the where clause in a select statement. Placing fby in a where clause allows an aggregate function (e.g. sum, max, avg) to be used to select individual rows across groupings. The left argument is a list containing two items – the first being the aggregation function, the second being the data vector (list/column) – and the right argument is the 'group by' vector. Below we make use of the fby function to obtain a breakdown by class of the objects with above average recessional velocity.

```
 q)select count i by class from (select class, recVel:rv z from
specObj) where recVel>(avg;recVel) fby class

 class | x

 ------| ------

 GALAXY| 242519

 QSO   | 68750

 STAR  | 116152

 q)\ts select count i by class from (select class, recVel:rv z   from
specObj) where recVel>(avg;recVel) fby class

 84 36701520
```

We can infer from these results that quasi-stellar objects (QSOs) have the greatest recessional velocities of the 3 classes in this data set on average, whereas stars' negative recessional velocity suggests they are moving towards us on average. Only roughly one third (242,519 / 678,098) of the galaxies in this data have an above average recessional velocity by class, while over half of stars, and closer to two thirds of QSOs are above average in their respective classes.

As previously mentioned, the positive impact of having the SDSS data available to the public has been massive. Since it was released, over 3000 papers have been written on a range of different topics in the field, based on data from the SDSS[4]. The SDSS is just one example of several projects, but it shows the benefit of making the data accessible to all interested parties, both professional and amateur. This impact throughout the wider astronomical community has pushed leaders in the field to continue this development – to further promote sky surveys by building bigger, more powerful telescopes. However, this doesn't come without its problems, with data storage and computational processing power being pushed to the limits. Kdb+ has the ability to scale to these extremes. Sean Keevey's paper 'A Natural Query Interface for Distributed Systems' in the Kx Technical Whitepaper series discusses how data distributed over several processes and machines can be seamlessly accessed from a single starting point by the end user. For a more detailed discussion go to http://code.kx.com/mkdocs/db/DB_A_Natural_Query_Interface_for_Distributed_Systems.pdf

We previously mentioned how astronomy data can often come with a time domain, which would be well suited to kdb+ and how the data would be stored on disk. The data set in this example does not have a time domain, as it just provides information on given objects recorded once, but this certainly does not mean that it is not suited to kdb+. We could apply an attribute to this data for optimization, such as the sorted attribute to the class column in the example data set. The benefits of this would become more apparent as more files were being loaded in. Ciaran Gorman's paper "Columnar Database and Query Optimization" in the Kx Technical Whitepaper series gives an in-depth explanation as to how they can be applied. For a more detailed discussion go to http://code.kx.com/mkdocs/db/DB_Columnar_Database_and_Query_Optimization.pdf

4. Conclusion

In this paper, we have taken one sample of astronomy data that came in a format which was initially unfamiliar to us, and subsequently analyzed it in kdb+ with relative ease. We made use of the flexibility that kdb+ has in extending to other languages; in this case C, which effectively only required using the 2: function. This approach can be replicated where C utilities have been created for interacting with data sources, or for other reasons.

We were then able to build functionality to calculate the recessional velocity of different object classes, which took less than a tenth of a second to run on a table of one million rows. In doing this we made use of kdb+'s efficient fby syntax to make use of an aggregate function as a filter criteria. It is apparent that kdb+ is extremely performant, both in storing and analyzing this data. The nature and expected volumes of data, and the way it will be used, go hand in hand with kdb+ and we have found it to be an ideal fit. We believe this will be a very attractive solution to the field of astronomy going forward, coping with the predicted expansion of data volumes in the field due to the new projects that are scheduled to begin in the near future.

References

[1] http://www.lsst.org/

[2] https://www.skatelescope.org/news/raeng-grant-to-engage-with-ska-engineering/

[3] Newman, R. and Tseng, J. (2011). Cloud Computing and the Square Kilometre Array.
http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002195

[4] Feigelson, E. D. and Babu, G. J. (2012), Big data in astronomy. Significance, 9: 22–25.
doi: 10.1111/j.1740-9713.2012.00587.x (link to online version:
http://onlinelibrary.wiley.com/doi/10.1111/j.1740-9713.2012.00587.x/pdf)

Further reading

http://www.theatlantic.com/technology/archive/2012/04/how-big-data-is-changing-astronomy-again/255917/

http://www.research.ibm.com/news.shtml

http://www.skatelescope.org/uploaded/8762_134_Memo_Newman.pdf

EMEA

Head Office
3 Canal Quay,
Newry,
BT35 6BP
N. Ireland
Tel: +44 (0)28 3025 2242

Belfast
11-13 Gloucester Street,
BT1 4LS
N. Ireland
Tel:+44 (0)28 9023 3518

Dublin
Fleming Court,
D04 N4X9
Rep. of Ireland
Tel: +353 (0)1 630 7700

London
Cannon Green Building,
1 Suffolk Lane,
EC4R 0AY
United Kingdom
Tel:+44 (0)207 3371210

Americas

New York
45 Broadway,
New York,
NY 10006
USA
Tel:+1 (212) 447 6700

Toronto
1599 Hurontario Street
Mississauga, On,
L5G 4S1
Canada
Tel: +1 289-329-0636

Ottawa
300 Terry Fox Drive,
Kanata, On,
K2K 0E3
Canada
Tel: + 1 (613) 216 9095

Palo Alto
#375
555 Bryant Street,
CA 94301
USA
Tel: +1 (650) 798 5155

APAC

Sydney
22 Pitt Street,
Sydney,
NSW 2000
Australia
Tel: +61 (0) 2 9236 5700

Singapore
55 Market Street,
048941
Singapore
Tel: +65 6592 1960

Hong Kong
Two Exchange Square,
8 Connaught Place,
Central
Tel: +852 2168 0715

Tokyo
Sanno Park Tower,
2-11-1 Nagata-cho,
Chiyoda-ku,
100-6162
Japan
Tel:+81 (0)36 205 3494

www.kx.com