Exploring Tabular Datasets

Clive Page
University of Leicester.
cgp@star.le.ac.uk

SC4DEVO: 2004 December 2

Raw and Reduced Data

- Raw data formats depend on the waveband
  - Radio – complex visibilities in u-v plane
  - IR/optical – CCD frames
  - X-ray/Gamma-ray – photon event lists.
- Data reduction requires specialised software, little scope for generalised data mining tools.
- Reduced data much more homogeneous:
  - Source lists – tables of numbers
  - Spectra – tables of numbers
  - Light-curves (time-series) – tables of numbers.
- Tabular datasets are probably the most important form of data for data exploration and mining – but tools lacking.

Astronomical Data Deluge

We hear much about the astronomical data deluge (tidal wave, tsunami, avalanche, …) which arises from:

- Growing importance of sky surveys (WFCAM, VISTA)
- Use of large-format detectors (OmegaCAM, MegaCAM)
- Interest in high-time resolution data (SWIFT, SuperWASP)
- Lots of new facilities: XMM-Newton, Chandra, INTEGRAL, eMERLIN, GAIA, ALMA, Planck, Eddington, JWST, XEUS…

- Much of the expansion is still to come
  - Source catalogues have up to 10^9 rows, the largest tables we have, but many of these come from old technology - scans of photographic plates dating back to the 1950s.

Examples of Source Catalogues

SDSS DR3 has around 1 Terabyte of tabular data.

Note that tables are growing not only longer, also wider:

Table	Number of Columns
USNO-B catalog	50
2MASS point sources	61
1XMM source catalogue	379
SDSS DR3 PhotObjAll	446

It's all the fault of Moore's Law
Can we keep up?

- Astronomical data volume – doubling maybe every 2 years.
- Processor power – doubling every 2 years or less.
- Disc storage per unit cost – doubling every 2 years or less.

So we can process and store the data.

- I/O bandwidth and disc seek times are improving much more slowly (~10% per year)

Hence: I/O more of a bottleneck – avoid if at all possible.

Storing and Retrieving Tabular Data

- Relational DBMS
  - Designed for large tabular datasets.
- Object-oriented DBMS
  - Support complex structures, and can avoid joins by merely following links through the schema.
  - But: astronomical users have mostly had their fingers burned.
- Statistical packages (SAS, SPSS, MINITAB, BDMP, MATLAB, S-PLUS, etc.)
  - Designed to handle datasets which fit in memory
- Data visualisation packages (IDL, PVWave, AVS, etc.)
  - Limited scalability: designed for millions, not billions.

Nine out of Ten Archives use RDBMS

- Can handle large datasets, well beyond the 2 GB boundary of 32-bit filing systems.
- Can use index for fast searching (find any row in ~30 ms)
- Have SQL - a powerful query language.
- Can handle null (missing) values properly (3-way logic)
- Mature technology with reliable software products.
- Cheap, because Open Source products like MySQL and Postgres are generally “good enough” for data archives.
- But: sequential scans are slow, because tables are invariably stored row-wise, because
  - Row-based storage makes transactions efficient.
  - Lots of other features mis-matched to needs of astronomy.

And the tenth?

- Some home-grown database software is still in use:
  - BROWSE from ESOC/ESTEC –
    - originated at ESOC in 1970s, written in Fortran66.
    - used by several astronomical data archives, and in a few recent papers
    - being phased out by archives as hard to support.
  - WCStools from Harvard-Smithsonian CfA
    - used by many archive sites as it supports fairly efficient cone-searches etc.
  - SIMBAD at CDS (Strasbourg)
    - the world's most comprehensive astronomical bibliographical database
    - was written in-house.

What should a data explorer package support?

- Basic database operations: select subsets, compute new columns, sort, group, equi-joins with other tables.
  - RDBMS does these fairly well.
- Statistical operations: find means, medians and other quantiles, find outliers, compute regressions, etc.
  - RDBMS can do simpler operations at least.
- Cross-matching: spatial join - needs spatial indexing
  - Some RDBMS can do this, but not all.
- Graphics and visualisation: histograms, scatter-plots, image overlays, density maps, multi-dimensional plots, etc.
  - Nearly all beyond the scope of RDBMS.
- Advanced mining algorithms (clustering, classification, etc)
  - Cannot do in RDBMS - data have to be exported.

Main Types of Query

- Indexed:
  - Fast: typically 20 to 30 milliseconds to select row.
- Sequential scans:
  - Slow: may take 30 to 60 minutes to scan a large table
- Combined - involving scanning and indexed access – e.g. joins.
  - Also slow, depending on the table sizes.

Main Types of Query

- Indexed:
  - Fast: typically 20 to 30 milliseconds to select row.
- Sequential scans:
  - Slow: may take 30 to 60 minutes to scan a large table
- Combined - involving scanning and indexed access – e.g. joins.
  - Also slow, depending on the table sizes.

Hence: avoid sequential scans if at all possible.

Unfortunately it isn't always possible.

Queries which usually involve a sequential scan:

- CREATE INDEX (*obviously)*
- Create/populate a new column, e.g.
  - UPDATE table SET column = expression;
- SELECT * FROM table WHERE (bmag-vmag) > 0.5
  - Column index won't help selection on *expression*
- SELECT AVG(col) FROM table;
  - Note: sampling to get approx answer often no faster.
- Finding largest N or smallest N from some column
  - DBMS usually do this by sorting.
- Joining one table with another (must scan smaller table)
- SELECT * FROM table WHERE ABS(glat) < 5.0;
  - May not use index unless selectivity >1000, because random seeks so much slower than sequential reads.

Data exploration in practice

- Data mining must be preceded by data exploration:
  - Understanding the data and instrument characteristics
  - Data cleaning – removing spurious values.
  - Finding the optimum parameters by repeated trials.
- All these operations need to be done interactively, often using graphical methods.
- Astronomical data explorations will nearly always include some sequential scans.
  - These will account for nearly all the elapsed time

Hence: need to speed up scans so they can be done interactively and do not require batch jobs – hugely more productive.

Three ways to speed up sequential scans

- Data compression:
  - gain of factor of ~2 easily
  - Hard to implement while keeping indexed access.
- Parallel I/O e.g. using Beowulf cluster or similar:
  - Many clusters available to astronomers.
  - Large potential large gains in speed.
  - Not yet adequately exploited for data mining.
- Column-oriented storage:
  - Most queries only involve a few columns out of many, so this will greatly reduce I/O.

The Story So Far

- Tables are getting longer and wider
- I/O is an increasingly serious bottleneck
- Queries which scan a whole table are an unavoidable part of any extensive data exploration process
- Most queries only involve a few columns out of dozens or hundreds in a typical astronomical table
- RDBMS use row-based storage which means the whole table has to be read from disc even if only a single column is accessed
- Column-based storage would be much more efficient for most queries, no less efficient for any archive query.

Column-oriented Tables

Several existing examples:

- Sybase-IQ best known commercial product - from makers of Sybase-ASE relational database. Fast, but:
  - No Linux version yet
  - No spatial indexing
  - Very expensive licence costs
- ESO/MIDAS table system
- STSDAS tabled (part of IRAF written at STScI)

Column-oriented table in FITS

- FITS binary and ASCII tables are normally row-orientated, but I managed to get column-based storage in two ways:
  1. Table of one row, but each field is a vector of length N (where N is the required number of "rows").
  2. FITS file with one column per header-data-unit.
- Both of were handled well by the cFITSIO library, much faster as expected, but
  - cFITSIO library still has some 32-bit limits.
  - Files conformed to letter but not spirit of Standard: incompatible with other FITS tools, so rather pointless to use FITS format.
  - High CPU overhead from need for byte swaps, and handling 2880-byte blocks.

Hierarchical Data Format 5

- HDF5 has flexible structure, supports tables, arrays, groups of objects.
- Comes from NCSA, well-supported library and tools.
- Compatible with Globus, GridFTP.
- Designed for efficient access to big files (no 2 GB limit).
- Simple API (bindings to C, Fortran90, Java, Python).
- Can attach unlimited metadata to tables and columns.
- Prototype using HDF5 was built and benchmarked:
  - Used query parser and evaluator originally written for Starlink's CURSA about 1992.
  - A sample of 2MASS (~10% of full catalogue) ingested into HDF5 column-based format.

Performance Gain

```
mysql> select count(phi_opt),min(phi_opt),max(phi_opt),avg(phi_opt) from
   twomass;
+---------------+-------------+-------------+-------------+
| count(phi_opt) | min(phi_opt) | max(phi_opt) | avg(phi_opt) |
+---------------+-------------+-------------+-------------+
|       33971487 |             0 |          360 |     195.9702 |
+---------------+-------------+-------------+-------------+
1 row in set (4 min 30.73 sec)


hydra:~/hdf> hstats phi_opt
 --Column--      points minimum     maximum     average
     phi_opt  33971487  0.0000      360.00      195.97
 8.37 seconds
```

32 times faster than MySQL, other tests show gains 13 to 400.

```
hydra:~/hdf> hstats phi_opt
 --Column--      points minimum     maximum     average
     phi_opt  33971487  0.0000      360.00      195.97
 3.51 seconds
```

Even faster when results in disc cache.

How to build an HDF5-based data explorer

- HDF5 library – reliable, efficient, easy to use, allows unlimited metadata to be attached to column or table.
- Query parser and evaluator – CURSA, or JEL (Java)
- Graphics package – great variety of them available
- Statistics routines – many available
- Indexing – many B-tree libraries available
- Spatial indexing – free R-tree code exists, or use pixel-code methods (based on HTM or HEALpix)
- Integration with VO for authentication, MySPACE, etc.
- Provide as stand-alone package, and as a Web Service.
- User interface – allowing iterative filtering, undo, etc.
  - Probably the most difficult part of the design.

Additional Advantages

- Makes distributed cross-match more feasible:
  - only transfer (ra, dec, err) columns across the network.
- Simple API so users with their own data mining code (clustering, classification, etc) can read HDF5 datasets directly.
- Data explorer can easily access astronomical tables in other formats such as FITS, VOTable, CSV, or (using JDBC) tables within other DBMS.
- Can provide user with integrated graphics, statistics, etc. and a host of other desirable features missing from commercial RDBMS.

Data explorer *complements* RDBMS

Appears to be feasible

- would speed up many slow data operations by 10 to 100 times,
- would overcome all the problems of the RDBMS

The HDF5-based data explorer is not intended to *replace* the existing DBMS-based archives but to *supplement* them

Cost of using both is mainly that of using twice as much disc space – no longer an expensive commodity.

The Data Deluge Revisited

From the introduction to the AstroGrid-1 Grant Application:

> A tidal wave of data is approaching astronomy, requiring radical new approaches to database construction, management, and utilisation.