Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

Federating Datasets with Spatial Joins

Clive G. Page

Dept. of Physics & Astronomy, University of Leicester

Abstract. Federating or combining astronomical datasets frequently leads to new discoveries, for example when cross-matching source catalogs. In database terms this requires a spatial join between tables. Two algorithms for spatial joins (pixel-code and R-tree join) were compared, and the performance of three DBMS (DB2, MySQL, and PostgreSQL) was evaluated using large source catalogs.

1. The Problem

Valuable information often arises when one combines or federates astronomical datasets. An important case is that of cross-matching two (or more) source catalogs to find information about each source in different wavebands or at different epochs. The usual matching criterion is the coincidence of celestial position (within limits of the error regions), but additional criteria may be required where there is still ambiguity. In database terms cross-matching requires a *spatial join* between tables, which requires spatial indexing. Relational database management systems (DBMS) are designed to execute joins on integer or string types, and only a few cope with joins on approximate celestial positions. We have compared two spatial join algorithms:

- Join using the true spatial indexing built into some DBMS. These mostly use the R-trees of Guttman (1984), the astronomical value of which was noted by Baruffolo (1999).
- Join using the pixel-code[1] algorithm (Page, 2003). This allows any DBMS to be used, as it only requires an equi-join of integers, but requires several more processing steps.

We also compared the performance and ease of use of three DBMS. Our short-list arising from earlier work[2] was:

- **DB2** (from IBM) - a commercial product (but free to many academic users), with an optional Spatial Extender using a multi-level grid file indexing system. Version 8.1 was used in this investigation.
- **PostgreSQL** - an open-source object-relational DBMS with R-tree indexing built-in, which is becoming more widely used in astronomy. Version 7.3.4 was used.

[1] `http://wiki.astrogrid.org/bin/view/Astrogrid/SkyIndexing`

[2] `http://wiki.astrogrid.org/bin/view/Astrogrid/DbmsEvaluations`

- **MySQL** - another open-source relational DBMS already widely used by astronomical data archive sites. The latest release, V4.1.0, supports spatial indexing using R-trees.

Notes:

1. These tests ran on a 2.2 GHz Intel Xeon system running Linux.
2. Each DBMS was optimized only as suggested in the introductory documentation. Large books on performance tuning are available for DB2, and perhaps expert tuning would have produced faster times.

2. Indexing: 1-d versus 2-d

Until recently only a few DBMS supported true spatial indexing. As a result many astronomical data archives, where the most advanced service is a simple cone-search, have managed by indexing just one coordinate. This is usually declination as it helps to avoid the wrap-around problem at 0 hours right ascension. Indexing on one axis becomes increasingly inefficient as dataset size grows. The results below show the elapsed times when using PostgreSQL to perform a cone search on the entire 2MASS point-source catalog of 470,992,970 rows using either one or two-dimensional indexing and with two different areas of sky:

Indexing method	**0.01 deg^2**	**2.0 deg^2**
Declination index (B-tree)	156.7 secs	1639.7 secs
Spatial index (R-tree)	1.2 secs	2.2 secs

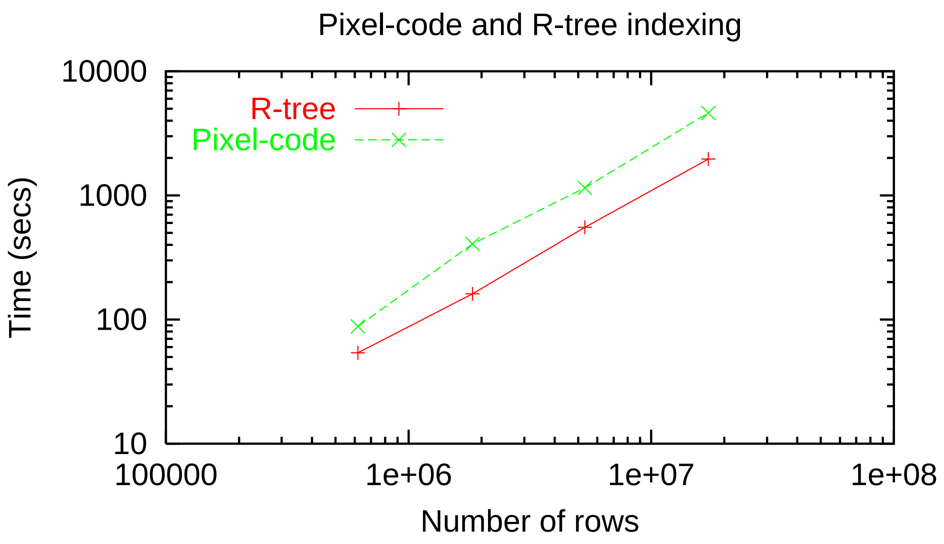
It was also notable that repeating the same query reduced the R-tree times to a few milliseconds, while those using B-trees were almost unchanged (because so much of the table had to be scanned sequentially). Since a join between tables is conceptually similar to a performing a large number of cone-searches, true two-dimensional indexing is clearly vital.

3. Pixel-code versus R-tree Join

Using PostgreSQL, we were able to compare the pixel-code algorithm with a join based on R-trees. In the new tests we matched varying chunks of the USNO-B catalog with sections of 2MASS covering a similar part of the sky.

Rows	**Pixel-code**	**R-tree index**
618,275	88 secs	54 secs
1,833,202	406 secs	161 secs
5,322,501	1150 secs	553 secs
17,202,108	4623 secs	1964 secs

The R-tree is seen to be about twice as fast as the pixel-code method for the largest datasets, and (as shown in Figure 1) the speed gap widens with increasing dataset size.



4. DBMS Ease of Use

These DBMS all claimed to support at least the entry level of the SQL-92 standard, but in practice many differences were found, e.g. in the data types they supported, the formats accepted for bulk loading of data, and in SQL syntax, especially for handling spatial data. In particular:

- **DB2** installation and configuration was exceptionally hard, one really needs a DB2 guru on hand. There are three user interfaces, all very user-hostile. No satisfactory way of representing nulls in an external data file could be found: if none exists this is a rather severe drawback when ingesting astronomical datasets. The spatial indexing system required the data range at each level of the grid file to be specified, in contrast to R-trees, which adapt automatically to any data range.
- **MySQL** was easy to install and use, but conformed less to the SQL standards. There appeared to be no external format for spatial data, so all data files had to be converted into very verbose `INSERT` statements, which also made data loading very slow.
- **PostgreSQL** was easy to install and use, and had good standards conformance, but a few irritating features, for example an index will be ignored if one fails to execute an `ANALYZE` statement after it is created, or if there is data type mismatch e.g. between a *double* constant in a selection expression and a *real* column index.

5. Relative Performance

These three DBMS appeared to run at similar speeds on most simple SQL commands, but diverged somewhat on complex operations. The table below shows the total time to perform the pixel-code join (load data, create indices, perform 2-way and 3-way joins).

DBMS	Time
DB2	383 secs
MySQL	427 secs
PostgreSQL	355 secs

Spatial joins using R-tree indexing were also tested on larger datasets (3.6 million rows of USNO-B, 5 million rows of 2MASS):

DBMS	Time
MySQL	5223 secs
PostgreSQL	1645 secs

Here PostgreSQL was significantly faster; its spatial data facilities are also much easier to use. We intended to test DB2's more complex multi-level grid file as well, but the intractability of the product and its licensing system prevented completion in the limited time available.

6. Conclusions

- PostgreSQL handled astronomical spatial data very well, and was in all respects a full-featured DBMS with good performance and ease of use.
- MySQL's features were more basic, but it was also easy to use. Although it performed well on simple queries, it was slower on complicated ones. Its spatial data facilities were hard to use and showed poor performance.
- DB2 was a large package with a wealth of features, but was exceptionally difficult to install, configure, and use. Its performance (without expert tuning) was generally similar to PostgreSQL.

The R-tree algorithm was not only faster and simpler, it was more flexible as the matching radius can be changed more easily. Since high quality DBMS supporting R-tree indexing are available from both free and commercial sources there seems no reason to adopt a slower more complicated algorithm for the spatial join. With join speeds over 10,000 rows/second, it is feasible to join an entire X-ray catalog (such as 1XMM) with a large optical or infra-red catalog in just a few seconds, once the necessary index has been created.

For further details of our DBMS evaluations see `http://wiki.astrogrid.org/bin/view/Astrogrid/DataDocs`

References

Baruffolo, A. 1999, in ASP Conf. Ser., Vol. 172, ADASS VIII, ed. D. M. Mehringer, R. L. Plante, & D. A. Roberts (San Francisco: ASP), 375

Guttman, A. 1984, Proc. of SIGMOD'84 Ann. Meet., ed. B. Yormark (Boston: ACM Press), 47

Page, C. G. 2003, in ASP Conf. Ser., Vol. 295, ADASS XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco: ASP), 39