Astronomical Data Analysis Software and Systems XV
ASP Conference Series, Vol. 351, 2006
C. Gabriel, C. Arviset, D. Ponz and E. Solano, eds.

Q3C, Quad Tree Cube – The new Sky-indexing Concept for Huge Astronomical Catalogues and its Realization for Main Astronomical Queries (Cone Search and Xmatch) in Open Source Database PostgreSQL

Sergey Koposov[1], Oleg Bartunov

Sternberg Astronomical Institute, Universitetskiy pr. 13, 111992, Moscow, Russia

Abstract. In this paper we present Q3C (Quad Tree Cube), a new sky indexing scheme and its implementation for the open source database PostgreSQL. We have implemented our sky-partitioning scheme as a loadable module for PostgreSQL and developed a very simple SQL interface for main astronomical queries: cone search, various spatial searches on the sphere and cross-matches of catalogues. The performance is very high and allows to work easily with the largest existing catalogues (USNO-A/B, 2MASS, SDSS). We propose PostgreSQL and our sky-indexing scheme as open source solution for any VO-services dealing with huge catalogues.

1. Introduction

In last few years, several projects have provided to community with a set of very large catalogues consisting of up to billion of objects. Simple access methods are no longer suitable to work with that amount of data and databases must be used instead. But even the most advanced databases are lacking for methods to effectively work with spherical coordinates and astronomical queries like cone searches and cross-matches. So, special astronomical indexes and sky-partitioning schemes were invented for this purpose. The first successful attempt was HTM – hierarchical triangular mesh (Kunszt 2000). This sky partitioning scheme has been used in MS SQL to provide the interfaces for the astronomical queries in the SDSS project. However, HTM has several disadvantages (O'Mullane 2000): first, HTM is a too complicated scheme which significantly limits its performance (in particular, in those cases with a high depth of segmentation); second, HTM was specifically developed for Microsoft SQL server, which is not open source. That was the motivation for the development of a new simple, fast and powerful sky-indexing scheme for open source database PostgreSQL.

[1]Max Planck Institute for Astronomy, Königstuhl 17, D-69117, Heidelberg, Germany

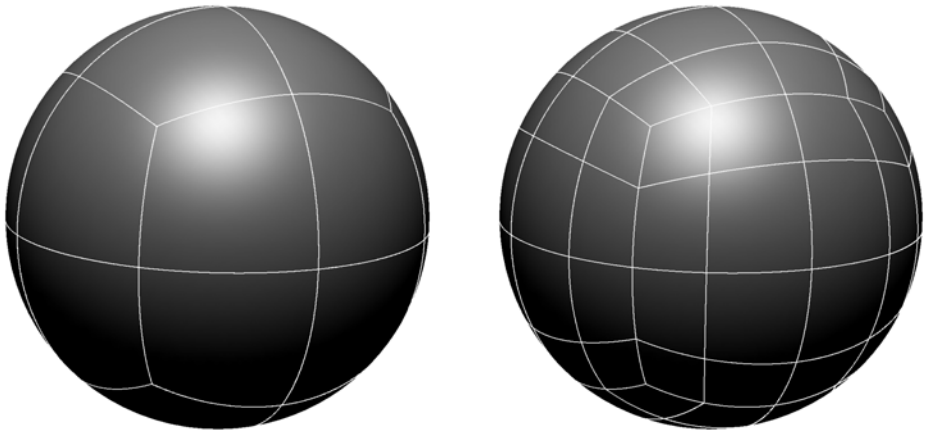

Figure 1. The sphere segmentation in Q3C.

2. Pixelization Scheme Details

The idea is similar to other sky-indexing/sky pixelization schemes. The base of the scheme is the cube inscribed in the sphere. On each face of the cube a quad tree is constructed. The quad tree structure creates the mapping of the 2D coordinates in the square to the bitmask (or just integer number). Since there are 6 faces, the 3 bits indicating the face number are appended. Thus the mapping of the cube to the integer numbers is established. And since one can easily do the central projection of the surface of the cube to the sphere, the quad tree structure is automatically inherited by the sphere. The final sky pixelization of the sphere with different depths is shown in Figure 1. We would like to stress two important points of this scheme. On one hand, the scheme as well as all the mathematical calculations are extremely simple (the quad tree in the square is very simple to work with and the trigonometric operations are not too numerous since the mapping of the sphere and the cube surface is just the central projection). On the other hand, the computations are much simpler than in HTM and HEALPIX (Górski 2005) and thus they do not limit the performance of the queries. Due to the usage of the quad tree in the square, special look-up tables speeding up the computations are utilized making it faster than HTM in the case of high depth of segmentation. The individual pixels of our pixelization scheme do not have equal areas (as HTM) since the property of equal areas of the pixels is absolutely unnecessary when the scheme is used for database indexing.

As we have seen, Q3C provide the mapping of each point of sphere to the integer number (we call it IPIX value) fulfilling that nearby points on the sphere have nearby IPIX values. This is the value used to create the index to allow the fast searches on the sphere. To effectively use the index, every spatial query is firstly segmented on pixels (the example for the circular query is on Figure 2) and since each pixel represents the continuous range of IPIX, the data for that part of the sphere can be easily and quickly retrieved from the database.

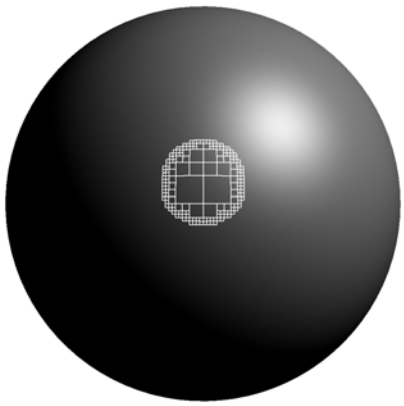

Figure 2. This example shows how the cone query is handled using Q3C

3. Realization of Q3C in PostgreSQL and Main Spatial Queries

The sky-indexing scheme Q3C has been programmed in C to achieve an optimum performance. Specific functions to integrate Q3C in PostgreSQL have been written. In the Q3C package we have written the set of SQL functions for the main spatial queries: cone search, rectangle query, cross-matches and others. We would like to emphasize that, at the database level, Q3C does NOT require adding special columns to the database, and only one index must be created! The example here shows how easy is to deploy Q3C from the PostgreSQL databases/tables (assuming that you have the table *usnob* with the RA, DEC columns)

```
db# CREATE INDEX usnob_idx ON usnob (q3c_ang2ipix(ra,dec));
db# CLUSTER usnob_idx on usnob;
db# ANALYZE usnob;
```

Now all the special queries from Q3C can be used. For example, to perform the cone search around the point (10°, 30°) with 1°radius:

```
db# SELECT * FROM usnob WHERE q3c_circle_query(ra,dec,10,30,1);
```

or, to perform the cross-match of *usnob* with other table *2mass* with error radius of 1 arcsec (0.00027°):

```
db# SELECT * from 2mass, usnob WHERE
q3c_join(2mass.ra, 2mass.dec, usnob.ra, usnob.dec, 0.00027);
```

with the same syntax you can do OUTER join cross-matches, or cross-matches with variable error radius. Q3C also supports rectangular queries, and will support polygonal area queries soon.

4. The Performance of Q3C with PostgreSQL

Q3C is not only the first open source sky-indexing scheme but it is also extremely fast. The main factors producing such a performance are: a) the ordering of the data induced by Q3C guarantees the optimal I/O performance of the data retrieval from the database; b) due to the simplicity of the sky-partitioning

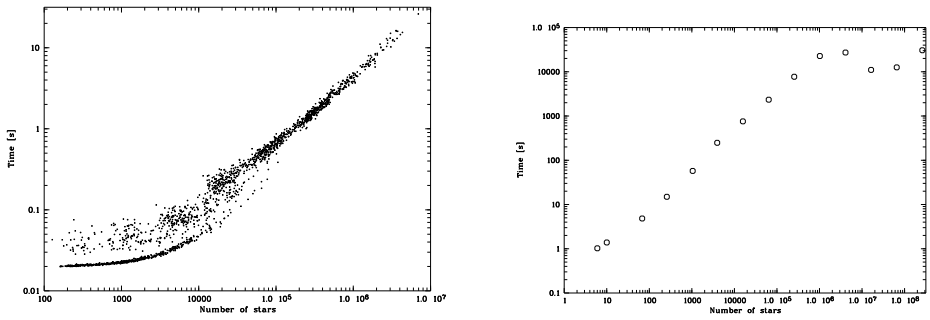

Figure 3. The performance of cone search and cross-match queries. On the left panel, the time to perform the cone search query versus the number of stars returned is shown. On the right panel, the time to perform a 1 arcsecond cross-match with USNO-B is plotted versus the number of stars of the catalogue.

scheme the computations (CPU time) does not limit the performance of Q3C even on high depth of segmentation; c) PostgreSQL works very effectively with the complex queries produced by Q3C. Figure 3 shows the performance of Q3C for the main astronomical queries (cross-match and cone-search).

5. Conclusions

- Q3C with PostgreSQL are the first open source solutions for spatial queries on the sphere for large databases.
- Q3C provides a very simple interface to the main astronomical spatial queries.
- Q3C does not alter the schema of the tables. Only one B-tree index should be created.
- Q3C is very fast and works perfectly even with the largest catalogues.
- The depth of the segmentation is limited to 30 (although it can be easily extended). Thus the size of the smallest pixel in the segmentation is $\approx 1mas \times 1mas$).
- ADQL to SQL translation for cone searches and Xmatches is very straightforward and it is very easy to build Skynodes based on Q3C.
- Q3C can be downloaded from `http://q3c.sourceforge.net`

References

Kunszt, P. Z. et al. 2000, in ASP Conf. Ser., Vol. 216, ADASS IX, ed. N. Manset, C. Veillet, & D. Crabtree (San Francisco: ASP)

O'Mullane, W. et al. 2000, 20 MPA/ESO/MPE Joint Astronomy Conference, Garching

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., HansenF. K., Reinecke, M., Bartelmann, M. 2005, ApJ, 622, 759