1. **Background**

MongoDB’s 2dshpere index actually combines the strength of discrete global grids and B+-tree structures, which first partitions the earth surface into cells at multiple resolution levels and then applies B+-tree to index geographical features approximated as one or multiple cells. It means that 2dsphere index is limited to accept spatial data of geodetic coordinate system (i.e., latitude and longitude) and calculate geometries on the earth surface (note hat 2dindex, supporting only point data of Cartesian coordinate system, was no longer advocated by MongoDB). However, there exist a large number of applications particularly at the city/country scale, such as land-use data, road network data and administrative division data, which are used to apply planar Cartesian coordinate to measure spatial data because of convenient acquiring and straightforward calculation. Planar spatial data, collected by these applications, is usually stored in object relational databases, like Oracle and PostgreSQL. One may argue that it is possible to transform planar spatial data under geodetic coordinate system and then import into MongoDB. However, such a transformation is neither necessary nor proper for spatial data of small scales, because position locating and geometry calculating under Cartesian coordinate system is more precise, and moreover, Cartesian computation is more simple and efficient than spherical computation.

1. **Introduction**

It is well known that a widely used index structure for planar spatial data is R-tree, which has already become an essential and indispensable module in modern spatial databases. In order to take both advantages of MongoDB and R-tree, we investigates how to effectively integrate R-tree index into MongDB, and therefore be capable of managing planar spatial data. The core idea is to flatten hierarchical R-tree structures into tabular MongoDB collections, in which R-tree nodes are represented as collection documents and pointers between nodes are expressed as document identifiers for foreign referencing. Firstly, data schema about planar spatial data and flattened R-trees is designed; after that, a module that manages planar spatial data by consuming and maintaining flattened R-trees is developed, which is then seamlessly plugged into MongoDB’s routing nodes. With this design, not only planar spatial data but also flattened R-trees could be distributed among MongoDB’s storage nodes, and moreover, planar spatial data could be loaded and queried through the existed interfaces that manipulate spherical spatial data. The experimental evaluation using real world data shows that planar spatial data can be efficiently organized by MongoDB with our plug-in R-tree index. As a result, the application areas of MongoDB could be greatly enlarged.

As far as implementation is concerned, our R-tree module resides in router server. By taking over message handling and reusing system commands, a set of built-in commands developed for accessing geographical data also applies to planar spatial data, and therefore, the learning cost of manipulating projected data will be greatly reduced. In addition, one new command, i.e., *registerGeometry*, is added to register layers of geometries, and the command of *createIndex* is overridden to accept parameters of R-trees, e.g., branching factor, minimum entries.

1. **Usage**

One can use almost the same way in operating geographical data to manipulate planar spatial data. The only difference is that before loading planar spatial data into MongoDB database, one should call the command of *registerGeometry* to add a layer to populate the same type of geometries. After that, users access planar spatial data with the commands exactly same to those applying to geographical data. It means one can use *insert* to add data, *createIndex* to build index, *find* to fetch data, *update* to modify data, and *remove* to delete data. However, it should be noted that the command of *createIndex* is overridden to have different parameters, and it could be invoked before or after data is loaded.