Analysis of Spatial Indexing Mechanism and Its Application in Data Management: A Case Study on Spatialite Database

Chengkun Liu123, Rui Ma123 and Li Zhang123
1 Changjiang Survey, Planning, Design and Research Co., Ltd, Wuhan, China
2 Changjiang Spatial Information Technology Engineering Co., Ltd, Wuhan, China
3 Water Resources Information Perception and Big Data Engineering Research Center of Hubei Province, Wuhan, China
Email: liuchengkun, marui, zhangli6@cjwsjy.com.cn

Abstract. Lightweight spatial database Spatialite is playing an increasing role in the current geographic information applications and other fields. This paper discusses the data organization and spatial indexing mechanism of Spatialite database in detail, studies the use of Spatialite database for data organization and index establishment, and utilizes the retrieval and spatial analysis methods supported by spatial index. The experiment takes urban pipeline data management as an example, through the visual application of urban pipeline data management in the 3DGIS environment, it provides a good solution for data management application of lightweight spatial database.

1. Introduction
In recent years, with the popularization of embedded and mobile devices, traditional complex GIS systems are gradually moving towards lightweight. In the GIS system, the storage and management of spatial data is a complex task. The emergence of embedded SQLite databases provides ideas for data storage of lightweight GIS applications with its simple and efficient characteristics. Therefore, early lightweight GIS applications usually stored the attribute information of spatial data in SQLite data table in a structured manner, and stored the spatial information in binary form as a field.

SQLite is an open source embedded relational database that implements a self-contained, zero-configuration, transaction-supporting SQL database engine. It is highly portable, easy to use, compact, efficient, and reliable. It has been widely used in recent years in various types’ Mobile software or embedded solutions. This mode can solve the problem of spatial data storage to a certain extent, but it is not conducive to the spatial query and spatial analysis of GIS.

In order to better meet the needs of spatial data management in lightweight applications such as tablet computers and mobile phone clients, developers of open source communities have expanded the SQLite database in accordance with the OGC standard to form a Spatialite database, which can access spatial data according to the OGC standard. Supports spatial indexing, querying and spatial analysis.

This article will explore the data organization and spatial indexing mechanism of the Spatialite database, and take the urban pipeline data management as an example to study the methods of using the Spatialite database for data organization and index establishment, as well as retrieval and spatial analysis supported by the spatial index.

2. Spatial Expansion in Spatialite Database
Thanks to its advantages such as light weight, convenience and high efficiency, Spatialite database has been widely used in various mobile geographic information engineering projects since its release, and
has become one of the research hotspots in GIS in recent years. It has strong advantages in terms of magnitude and data processing speed. It also has extensive market prospects in network and desktop GIS applications.

The Spatialite database supports the operation of SQL language and geometry type fields under the OGC standard, and also integrates the following open source libraries:

- GEOS library, which is used to support various spatial predicates and implement spatial analysis operations;
- PROJ.4 library for coordinate transformation between different coordinate reference systems;
- LIBICONV library for supporting multiple languages;
- SQLite library for implementing SQL data engine.

The use of Spatialite spatial database can conveniently and effectively organize and manage data, realize seamless organization of data, ensure data integrity, reduce data redundancy, so as to improve data query speed and reduce data storage space.

For the storage of spatial coordinate information, Spatialite uses a spatial field that specifically represents geographic coordinate information. It includes a Geometry data type using a flat rectangular coordinate system and a Geography data type using a geographic coordinate system. The former is represented by plane coordinates XY, and the latter is represented by latitude and longitude.

According to different spatial entities, Spatialite supports several spatial data types: Point, LineString, Polygon, MultiPoint, Multi-LineString, Multi-Polygon, etc. These different types of spatial entity records constitute complex spatial information data. Each spatial record corresponds to its specific ellipsoid-based spatial reference system according to the spatial reference identifier (SRID).

Spatialite uses a metadata mechanism to more efficiently retrieve and manage geospatial data. Its metadata table stores the data table name of the spatial data, the spatial field name, the geometry type of the spatial entity, the coordinate dimensions, and the coordinate reference information. Spatialite spatial metadata consists of two sets of tables, geometry_column and spatial_ref_sys. The spatial index is the key technology of a database system that supports spatial expansion. It is an important indicator for quickly and efficiently querying, retrieving and displaying geospatial data.

3. Analysis of Spatial Index Mechanism

Unlike ordinary relational databases, there are many difficulties in using spatial databases to store and manage data. On the one hand, due to the high dimensionality of the data processed by spatial data and the large number of spatial elements that need to be stored, the amount of data is very large; on the other hand, the types of spatial data are complex. Includes some special data types that are closely related to geographic location.

In order to realize the management of spatial data from multiple fields such as geographic research, meteorological information, road planning, municipal pipe network, etc., and to meet its requirements for efficient query, retrieval, and update, the design of the spatial index becomes particularly critical. The following will discuss the advantages and disadvantages of various types of spatial indexes, and focus on analysing the spatial index mechanism and characteristics of the Spatialite database.

3.1. Common Spatial Indexing Method

Spatial index is a data structure that sorts spatial objects in a certain order according to the location information, shape, attribute information, or spatial relationships between objects when the database stores data. It contains the object's identification information, objects Minimum Bounding Rectangle (MBR) and a pointer to a spatial object entity. The spatial index is a data structure between the operation algorithm and the actual spatial object. Through searching in the spatial index, quickly locate the object, filter a large amount of data that does not meet the conditions, reduce the amount of calculation that needs to be accurately calculated, and improve the query. Efficiency, so spatial index is a very effective structure, which is becoming the focus of people's research and attention.

In order to adapt to different spatial data storage requirements, researchers have proposed multiple spatial index structures for different problems. The existing spatial index structures can be roughly
divided into four types of spatial indexes: binary tree-based index structures, B-tree-based index structures, hashing-based index structures, and spatial filling curve-based index structures.

3.1.1. Index Structure Based on Binary Tree
This index structure is a binary index tree structure. The most typical representative is the KD tree, which is mainly used to index multi-dimensional data points. For more complex spatial objects, such as lines, areas, and volumes, the objects must be approximated first. Or use spatial mapping, so its query efficiency is low.

3.1.2. Index Structure Based on B-tree
The B-tree and its variants have strong search performance and are the most widely used index structure. At present, most of the index structures are based on the B-tree and the B-tree variant R-tree. Variations with improved features, including: R+ tree, R* tree, TPR tree, Hilbert R tree, R-link tree, etc.

3.1.3. Index Structure Based on Hashing
The idea is to divide the entire space into non-uniform small squares. Each small square corresponds to a certain disk page, and then the objects in each grid are also associated with a specific disk page. Correspondence can be calculated through a hash function. Typical index structures are Grid-file, R-tree, etc. This type of index structure is mainly applicable to point objects.

3.1.4. Index Structure Based on Space Filling Curve
The object space is divided into several grids, and each grid is represented by a unique number. If an object occupies several grids, then this object can obtain another unique code based on the numbers of the intersecting grids. This type of filling curve-based index structure essentially maps high-dimensional space objects to one-dimensional space. These values can be sorted and retrieved in one-dimensional space. The most common indexing algorithms are Z-ordering, Location Keys and Hilbert Curve, etc.

3.2. Spatial Indexing Method of Spatialite Database
Spatialite usually uses the R* tree spatial index mechanism to improve the speed of spatial retrieval and data analysis. The establishment of R* tree indexes can be implemented using SQL extension functions: CreateSpatialIndex (TableName, ColumnName).
When indexing, each node of the R* tree contains an index code for a rectangular area, which is composed of the smallest containing rectangle of all the child nodes of the corresponding node. Therefore, the geometry to be queried is represented by the smallest bounding rectangle. To determine the spatial extent of the collection graphic. The process of indexing the space division is shown in Figure 1.
3.3. Spatial Metadata of Spatialite Database

Spatial data generally contains metadata information that reflects the data area, format, and projection information. Using this information can further improve the retrieval efficiency of spatial data. Spatialite uses a metadata mechanism. The metadata table stores the table names, field names, spatial entity geometry types, coordinate dimensions, and coordinate reference information of spatial data. The relationship diagram of the metadata table is shown in Figure 3.

**Figure 1.** Spatialite Database R* Tree Spatial Structure Partition

Based on the above-mentioned spatial division, the R* tree structure index shown in Figure 2 can be constructed, and the spatial data required by the user can be efficiently retrieved.

**Figure 2.** Spatialite Database R* Tree Hierarchical Index

**Figure 3.** Relationship Diagram on Metadata Tables of Spatialite Database
4. Spatial Data Management of Pipelines based on Spatialite Database

Urban pipeline data is a type of typical spatial data, it has the characteristics of large data volume and rich attribute information. In view of this, how to establish an efficient storage and retrieval mechanism is a problem that pipeline information management systems need to solve. In this paper, Spatialite is used to manage the pipeline data, and the spatial information of the pipeline is stored in the LineString type. At the same time, the information such as the layer, name, and attributes of the pipeline for display and query are recorded in the attribute information, as shown in Table 1.

Table 1. Pipeline Data Table Design

| Field  | Type      | Description                              |
|--------|-----------|------------------------------------------|
| 1 PK_UID | Integer   | unique identifier                        |
| 2 layer | Text      | layer of pipelines                       |
| 3 name  | Text      | name of pipelines                        |
| 4 vis   | Bool      | visibility of pipelines                  |
| 5 colour| Integer   | colour of pipelines                      |
| 6 note  | Text      | additional note info of pipelines        |
| 7 texture| Text     | texture file path of pipelines           |
| 8 unit  | Double    | unit of the texture file                 |
| 9 direct| Integer   | flow direction of the pipelines          |
| 10 size | Text      | size of the pipelines                    |
| 11 Geometry | LineString | spatial coordinates and shape of the pipeline |

By storing vector data files of various pipelines into the Spatialite database and indexing the spatial information fields, it is possible to perform rapid retrieval and query of spatial objects. At the same time, with the help of the spatial analysis extension function of Spatialite database, it is also possible to calculate the length and area of spatial objects and improve the flexibility of pipeline data management. The following will introduce the application of Spatialite in pipeline data management, query, and spatial analysis.

4.1. Data Manipulation Design and Data Management

Spatialite provides C language API interface functions for operation. In order to perform data operations more conveniently, these functions are encapsulated, and a unified data structure is provided to meet the operational requirements of addition, deletion, modification, and query. The main operation classes after encapsulation are as follows:

- CppGaiaGeometry class. Encapsulates the Point, LineString, Polygon data types provided in Spatialite, and provides a unified interface for data operations. This data type is used in other data operation classes, which facilitates unified storage management and addition, deletion, modification, and review of spatial data.

- CppSpatialiteDB class. Encapsulates functional functions for database file operations, including opening or closing data connections, and performing operations such as DML and DDL on the database.

- CppSpatialiteQuery class and CppSpatialite Table class. Encapsulate a function that returns the results of a SQL query. The former uses a cursor to traverse the query results, while the latter directly returns the complete query content.

- CppSpatialite Statement class. Parameter binding is used to execute SQL commands to improve the efficiency of data operations on complex spatial data objects. Where bind is used to bind
the specified spatial data object to the parameters in the SQL statement, reset is used to reset the statement, and finalize is used to destroy the statement.

- CppSpatialiteAnalysis class. Encapsulates common spatial analysis methods, such as: distance calculation between space objects, buffer calculation, finding the intersection point between two objects, length calculation, area calculation, etc.
- CppSpatialiteBinary class. Encapsulates the conversion of a spatial data object into a binary data store.

4.2. Spatial Query
Spatial query is an important part of GIS application. By selecting a custom query range and combining constraints such as keywords, you can quickly retrieve the information the user needs. Based on the spatial index created by the Spatialite database, each spatial feature generates its corresponding minimum bounding rectangle (MBR), and when the user selects the query range, he will also give a rectangular range to be queried through the selection of the mouse frame, and its corresponding MBR is itself. Assume that the query rectangle specified by the user is Geom1, and the MBR of the spatial feature is Geom2. The possible spatial relationship between the two is shown in Figure 4.

![Figure 4. The Spatial Relationship of Different Geometric Objects](image)

In the seven spatial relationships listed in FIG. 4, there are mainly two types of spatial relationships commonly used for spatial query, that is, the query area contains the elements to be queried (Contains) and the query area and the elements to be queried (Intersects).

The query process is performed using spatially extended SQL language. Its input and output are in accordance with the spatial data organization specification formulated by OGC. The type information and coordinate information of the object are given in the text format of the Well-Known Text (WKT). In order to improve the cheapness of use, users do not need to understand the complex format of WKT, in the encapsulated CppSpatialiteQuery class, the CppGaiaGeometry geometric object class is used to define the input and output conditions of the spatial object, and the MBR methods involved in spatial query are encapsulated as corresponding function that User calls.

4.3. Additional Spatial Analysis Method
Spatialite integrates GEOS, Proj.4 and other libraries, and can implement commonly used spatial analysis functions, such as: spatial relationship judgment, spatial distance calculation, spatial predicate operation, coordinate conversion, etc. Table 2 summarizes these commonly used spatial analysis methods, and gives examples of corresponding typical functions according to classification. By encapsulating these functions in the CppSpatialiteAnalysis spatial analysis operation class, common analysis functions can be provided for pipeline data.
| Spatial Analysis Function                                                                 | Spatial SQL Encapsulated Examples                                                                 |
|------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Spatial relationship judgment, returns whether it conforms to a specific spatial          | Disjoint (geom1 Geometry, geom2 Geometry): Integer                                                |
| relationship                                                                              |                                                                                                  |
| Spatial distance calculation                                                              | Distance (geom1 Geometry, geom2 Geometry): Double                                                 |
| Spatial operators such as buffers, unions, etc., return spatial objects after performing  | GUnion (geom1 Geometry, geom2 Geometry): Geometry                                                 |
| a spatial operation                                                                        |                                                                                                  |
| Coordinate transform, according to the given coordinate system                           | Transform (geom Geometry, newSRID Integer): Geometry                                              |
| Other spatial analysis functions, such as: Delaunay triangulation generation, Voronoi     | DelaunayTriangulation (geom Geometry, edges_only Bool, tolerance Double): Geometry                |
| diagram construction                                                                      |                                                                                                  |

5. Applications

Taking the Panlong City underground pipeline data in Wuhan as an example, the pipeline data is imported into the Spatialite database, and attributes such as colour, size, flow direction are given in the data table fields for different types of pipelines. A three-dimensional management information platform for pipeline data is constructed. The pipeline data is displayed in three dimensions.

At the same time, the encapsulated Spatialite operation class can be used to add, delete, and modify data as needed. Figure 5 shows the query result of the pipeline data obtained through the pull-frame query. The query pipeline is highlighted in yellow and its attribute information is displayed on the left panel.

![Spatial Query of Panlong City Underground Pipeline](image)

6. Conclusion

In recent years, traditional complex GIS systems are gradually moving towards lightweight, and the emergence of Spatialite database has filled the gap of lightweight spatial database. This paper discusses the data organization and spatial indexing mechanism of the Spatialite database. Taking the urban pipeline data management as an example, this paper studies the methods of using the Spatialite database for data organization and index establishment, as well as retrieval and spatial analysis supported by the spatial index.
The test results of pipeline data examples show that the use of Spatialite database can improve the efficiency of spatial query and spatial analysis to a certain extent, and provide effective support for the development of GIS-related program functions. With the in-depth application of Spatialite database in the future, it can further provide practical pipeline analysis functions such as tube burst analysis and connectivity analysis.

7. Acknowledgments
We would like to thank the constructive comments from the anonymous referees, and we appreciate the financial supports from the National Key R&D Program of China (No. 2017YFC1501206 and 2017YFC1502601).

8. References
[1] Qing Zhu, Jun Gong, Yeting Zhang. An efficient 3D R-tree spatial index method for virtual geographic environments[J]. Isprs Journal of Photogrammetry & Remote Sensing, 62(3):217-224.
[2] Zhao B, Bian F L. Dynamic Quadtree Spatial Index Algorithm for Mobile GIS[J]. 2007, 33(15):86-88.
[3] Zenon Pylyshyn. The role of location indexes in spatial perception: A sketch of the FINST spatial-index model[J]. Cognition, 32(1):1-97.
[4] P.J.M. van Oosterom, C.H.J. Lemmen. Spatial data management on a very large cadastral database[J]. Computers Environment & Urban Systems, 25(4-5):509-528.
[5] Francesco Napolitano. repo: an R package for data-centered management of bioinformatic pipelines [J]. Bmc Bioinformatics, 2017, 18(1):112.
[6] Donghui Zhang, Tian Xia. A novel improvement to the R*-tree spatial index using gain/loss metrics[C]// 12th ACM International Workshop on Geographic Information Systems, ACM-GIS 2004, November 12-13, 2004, Washington, DC, USA, Proceedings. ACM, 2004.
[7] Qing Zhu, Jun Gong. An Improved Full 3D R-Tree Spatial Index Method[J]. Geomatics & Information Science of Wuhan University, 2006, 31(4):340-343.