Managing Spatial Indexes

Spatial indexing is used to promote the query and access speed of geospatial data.

Spatial indexes of GIS are on the basis of field indexes. Since traditional field indexes cannot manage with the fast locating the spatial data that is spatially related, spatial indexes are necessary to improve the access speed of spatially related data.

Spatial Index Types Supported

Introduces the spatial index types supported by SuperMap, including Q-tree index, R-tree index, Tile index and Dynamic index.

Spatial indexes of GIS are on the basis of field indexes. The field index is a data structure which can locate key values in a fast way, which is very popular in database systems and computer systems. In addition, the field index is able to access certain specific keys in a fast way. Generally, the field indexes employ the B-tree index or B-tree index derivative data structure. Since traditional field indexes cannot manage with the fast locating the spatial data that is spatially related, spatial indexes are necessary to improve the access speed of spatially related data.

SuperMap provides four types of spatial indexes applicable to vector datasets: Q-tree index, R-tree index, tile index and dynamic index.

A dataset can have only one type of spatial index at the same time. The current spatial index of a dataset will be automatically maintained if the dataset is being edited. The indexing efficiency might be negatively affected after several times of editing. Therefore, you need to rebuild the spatial index.

<table>
<thead>
<tr>
<th>Engine Type</th>
<th>R-tree index</th>
<th>Q-tree index</th>
<th>Dynamic index</th>
<th>Tile index</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDB</td>
<td>Y (Applicable to all types of datasets)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SQL Plus</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>OraclePlus</td>
<td>Y (Applicable to all types of datasets)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>OracleSpatial</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>KingBase</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>MySQL</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>DB2</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>BeyondDB</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>GBase</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
<tr>
<td>Alibase</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to 2D/3D point and tabular dataset)</td>
<td>Y (Not applicable to CAD dataset)</td>
<td>Y (Not applicable to route dataset)</td>
</tr>
</tbody>
</table>
**R-tree index**

Introduces the concept, characteristics, and use of R-tree index.

**Principles**

R-tree is an index structure based on disk, which is an natural extension to the B-tree (one dimension) in the multi-dimension space. It is convenient to integrate the current database system, such as Oracle, SQL Server, etc, and it supports diversified spatial query operation. In fact, it has been widely used and is one of the most popular spatial indexes.

The R-tree spatial index defines several rectangles to contain geometric objects, that is, a rectangle contains some objects which have the close spatial locations. So the rectangle is regarded as the spatial index, which includes the pointers of the geometric objects. As shown in the figure, the rectangle named A includes the geometric objects such as D, E, F. The rectangle named B includes the geometric objects such as H, I, J and K. The rectangle named C includes the geometric object such as M, L. An example of R-tree spatial index displays as below.

![Figure: R-tree Index Diagram](image)

If you perform the spatial index, you should first determine which rectangles are in the search window, then determine which geometric objects are in those rectangles. In this way, the query will be speed up.

**Features**

- The spatial retrieval efficiency of R-tree index is very high, and the accuracy of the retrieval is also very high.
- R-tree index data and geographic data is not stored together, and the update of the R-tree index is related to its balance and split, so it's complex, and the update speed is slow, the query efficiency is reduced if many editing is done, it will be needed to rebuild the spatial index. R-tree index is suitable for the read-only data or the spatial data which doesn’t be edited often.
- R-tree index supports file type datasource and database type datasource, the specific types can be seen in Table 1.

**Available**

R-tree index is suitable for static data, e.g., the base map data and the data which doesn’t often need to be edited (except the attribute data editing) are recommended to establish the R-tree index.

**Note**

It supports the engine types: UDB, SQLPlus, PostgreSQL, OraclePlus, OracleSpatial, KingBase, MySQL, BeyonDB, GBase and Altibase.

**Q-tree index**

Introduces the concept, and use of Q-tree index.

**Principles**

Quadtree index is an effective 2D spatial index, and its basic idea is: the entire working space is encoded to 0, with horizontal and vertical lines divided into four parts, which are encoded to 00, 01, 02 and 03 as the order, the upper left corner is 00, the lower right corner is 03; the same way to divide each grid, e.g., the grid 00 is divided into 000, 001, 002 and 003. According to the need, you can also continue to divide them. The dividing layers depend on the number and the average size of the objects in the research area.
If the object shown above is objects contained by the entire dataset, then its quadtree index structure is established as shown below:

- Available

  It's for high concurrency editing of the data with small data volume.

- Note

  It supports the engine types: SQLPlus, OraclePlus, KingBase, MySQL, DB2 and GBase.
**Tile index**

Introduces the concept, and use of Tile index.

**Principles**

In SuperMap SDX+, the spatial objects are classified according to one attribute field or a given range (sheet length and width). The spatial objects are managed by the index to improve the query speed. As shown in the figure below shows the map of the administrative boundaries of the Chinese municipal administrative boundaries, according to the given range to create the tile spatial index, showing h4708. The tile spatial index is very useful for the display and query of the vast amounts of data (i.e., the number of objects is more than one million). After the establishment of the index, it only shows what you need to see so as to greatly shorten the display time.

![Tile spatial Index Diagram](image)

In particular, the tile spatial index can also be cached to the local (in the form of binary file), so that it can further improve the speed of query and retrieval. File caching is an intelligent distributed storage scheme, which is provided by SuperMap SDX+ to balance the network and server load and improve the whole performance. After you open the file cache option, when the application is accessing data stored in the spatial database, it will firstly check whether the
data exists in the local cache library, if there is no corresponding cache data or the cache data is not the latest version, it will read the data from the server and update the local cache data, so the next time it can read the local cache data. If the local cache has the latest version of the corresponding data, it does not need to use the network to access the server to request data, but directly read the local cache data to complete the display or analysis. Through this solution, the database server load and network load can be reduced greatly, and the whole performance of the application is greatly improved.

**Available**

For the datasets stored in the database and with the standard scale (such as 1:25 million, 1:10 million, 1:5 million), it can provide a very good query performance after the tile spatial index is built, with the provided caching option, it can achieve better map browsing speed.

**Note**

- The algorithm of building the tile spatial index has made a substantial improvement in version 5.3. According to the data quantity, the index time consuming can be reduced 4-10 times. To build the tile spatial index for the dataset whose records are more than 50000, it's recommended to use the version 5.3 and above.
- The tile spatial index is currently only supported by the point, line, region, text and CAD data sets in the database datasource.
- It supports the engine types: SQLPlus, OraclePlus, KingBase, MySQL, GBase.

**Dynamic index**

Introduces the concept, characteristics, and use of Dynamic index.

**Principles**

This multi-level grid index uses multi-level grids to organize and manage data. The basic method of the dynamic index is that it divides the dataset into equal size or unequal size grid, and records the locations of the grids which are occupied by the features. The regular grid is usually used in GIS. When performing spatial query, first the cells that contain the queried features will be calculated, then, the features will be queried rapidly. This multi-level grid index is used to optimize spatial query performance.
The spatial grid index has three grid levels, namely one, two, and three grid levels in current version, and each level has its own distinct cell size. The first grid level has the smallest cell size, and the data of 50% belongs to the first level. The second and third cell size must be larger than that of the preceding.

**Features**

- The dynamic index will improve the speed when browsing the data.
- The ability to index updates and concurrency is good;
- The spatially search accuracy is high.

Note: this index type supports dynamic concurrency editing, supports database datasource.

**Available**

Dynamic indexing is a new type of index provided by SuperMap 5.3. It combines the advantages of R tree index and quaternary tree index to provide a very good concurrent editing support. It has good universality, which is the default spatial index type.
If you cannot determine which spatial index is applied to the data, it is recommended to build a dynamic index.

**Note**

- The engine types which support for building a dynamic index include: SQLPlus and OraclePlus.
- After the establishment of a dynamic index, the user can make a local cache according to the map range, which can greatly improve the speed of query and browse, especially for the GB level data. For the data that is stored sheet by sheet, such as the national basic scale topographic map, the dynamic index can be set according to the size of the map sheet.

**Index Storage Sample**

Introduces the storage structures of the four spatial indexes.

The following examples are based on the Oracle datasource, according to the index type to introduce the maintenance ways and the storage of spatial index, to facilitate advanced users to understand the structure of the index in the database.

**Quadtree index.**

It’s for high concurrency editing of the data with small data volume.

Maintenance: No related index table, if you build this index, only need to modify the SmKey value of the attribute table.

**Index Storage Sample:**

<table>
<thead>
<tr>
<th>SMID</th>
<th>SmKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>268435557</td>
</tr>
<tr>
<td>26</td>
<td>268435557</td>
</tr>
<tr>
<td>27</td>
<td>268435557</td>
</tr>
<tr>
<td>28</td>
<td>268435557</td>
</tr>
<tr>
<td>29</td>
<td>268435557</td>
</tr>
<tr>
<td>30</td>
<td>268435557</td>
</tr>
<tr>
<td>31</td>
<td>268435557</td>
</tr>
<tr>
<td>32</td>
<td>268435557</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>134217734</td>
</tr>
</tbody>
</table>
R-tree spatial index.

R-tree index is suitable for static data, e.g., the base map data and the data which doesn’t often need to be edited (except the attribute data editing) are recommended to establish the R-tree index.

Maintenance:

- The corresponding index table name: sm_idx_dataset table name, used to store the R-tree index data.
- When you create the index and do not modify the objects in the dataset, its SmKey=-2.
- SmIndexLevel in the registry table smregister is 0.
- If you add or modify the records, the SmKey may be larger than -2 (determined by the R-tree index algorithm).
- If there is a record whose SmKey is greater than -2, then its SmIndexLevel is 5 in the registry table smregister.

Index Storage Sample:

Tile spatial index

Available:

- Field index: that is, according to the field to classify the objects, through the index to manage the classified spatial objects, in order to improve the query and retrieval speed.
- Range index: that is, according to a specific range to classify the objects, through the index to manage the classified spatial objects, in order to improve the query and retrieval speed.

Maintenance:

- Index table name: Lib_dataset table name.
- The SmLibTileID value in the attribute table corresponds to SmID in the index table.
- The row and column number of the SMLIBTILENAME field value(The row and column number (0,0) is calculated from the lower left corner of the dataset range.).
- The SmLibTileID field in the dataset table and the SMID field in the index table is the relationship of 1 to N.
- Data query process: first, get the current map window range; second, according to the sheet size in the range and tile spatial index, determine which sheets will be displayed in the current view; third, in the corresponding index table (Lib_ dataset table name) to find the corresponding SMID values; fourth, find the objects to be displayed through the relationship between SMID and SmLibTileID. Through the above four steps, you can quickly locate the objects to be displayed.
- For the newly added object, the object is added to the position where the SMID=1 in the index table if the object is outside the range of the original dataset.
Index Storage Sample:

Dynamic index

Applicable scope: combines the advantages of R-tree index and quaternary tree index, provides a very good concurrent editing support, and can be widely used.

Maintenance:

- The corresponding index table name: SM_GDX_ dataset ID.
- The SmDynamicIndex table records the dynamic index information of each dataset.

Index Storage Sample:

Note

After the data is edited, the index will be automatically updated, but here in order not to affect the efficiency of display, it will not rebuild the index. Therefore, after a large number of editing operations, it is recommended to manually rebuild index, in order to achieve the index updating maintenance to ensure the query efficiency.
Rebuilding Spatial Index

Introduces the concept, characteristics, and use of R-tree indexes.

Instructions

Rebuilds the spatial index for a vector dataset. A data set can only use one index in one moment, but the index can be switched. For example, the data set in the early editing operation more, you can create a dynamic index of the data set, at the end of the editing, only to static display, you can rebuild the spatial index R-tree index.

When the data set is in the edit state, the system automatically maintains the current index. In particular, when the data has been edited several times, the efficiency of the index will be affected to varying degrees, the need to re-establish the spatial index.

Basic Steps

1. Right click on a vector dataset node in the workspace manager, and select Rebuild Spatial Index, the Spatial Index Manager dialog box is displayed.
2. In the "index type to be built", select the appropriate spatial index type to rebuild the spatial index for the dataset.

Managing Spatial Indexes

Introduces how to manage the indexes for one or multiple datasets, including checking the index info and building indexes, etc.

Instructions

The Spatial Index command allows you to manage the spatial indexes for the selected datasets.

Basic Steps

1. In the Workspace Manager, select the datasets whose spatial indexes you want to view or modify. You can select several datasets using Shift + click or Ctrl + click.
2. On the Start tab, click Data Processing > Spatial Index to display the Spatial Index Manager.
The Original Index indicates the original index type of a dataset. To create a new type of spatial index for a dataset, change its Target Index in the Spatial Index Manager.

To know more about the Spatial Index Manager, please refer to the introduction to the Spatial Index Manager in this page.

3. If you want to create a new type of spatial index for a dataset, you can set the Target Index for it, and then click Create at the bottom of the Spatial Index Manager.
4. After creating or viewing the spatial indexes for datasets, you can click Cancel to close the Spatial Index Manager.

5. **Introduction to Toolbar**
   - **Add**: Click "Add" the "Select" dialog box pops up, the spatial index information of other dataset can be added to the "Spatial Index Manager".
   - **Select All**: Select all datasets in the list box.
   - **Select Inverse**: Select the inverse datasets in the list box.
   - **Remove**: Remove datasets form the list box.

**The Spatial Index Manager**
The list box on the left side of the Spatial Index Manager lists all datasets selected for spatial index management. The information about each dataset are also displayed in the list box.

- **Dataset**: Displays the type, alias of the vector dataset and the name of the datasource which contains the dataset.
- **Datasource**: Display the data source of the vector data set.
- **Original Index**: The original spatial index type of the dataset.
- **Original Index**: The original spatial index type of the dataset.

**Target Index**: The type of the index you want to create for the dataset.

- The Target Index combo box on the right side of the Spatial Index Manager is bound to the selected items in the list box. If one or more datasets have been selected in the list box, you can set the the Target Index for them together with the Target Index combo box.

Under the Index Type combo box is the description of the spatial index type selected in the combo box.

- **Create**: Create spatial indexes for datasets in the list box according to the specified types of Target Index. Remember to remove the datasets you don’t want to create spatial indexes for before clicking Create.
- **Close**: Close the Spatial Index Manager.

**Parameter Settings**

**Tile index**: Click the dropdown button on the right of "Index Type", and select "Tile index". You need to set parameters for a Field Index or a Grid Index.

- **Field Index**: Sorts the dataset objects by the values of a specific field.
- **Grid Index**: Sorts the dataset objects by dividing the dataset into grid cells with a specified width and height. The unit of the width and height is the same with that of the dataset.

**Dynamic index**: Click the dropdown list of "Index Type", and select "Dynamic index". You need to set parameters for a Field Index or a Grid Index.

- **Center Point**: Sets the center of the index, usually the center of the dataset.
- **Level 1 Width**: Set the cell size if the first level of grids for the dynamic index. This parameter cannot be set to 0. The unit of this parameter is the same with that of the dataset.
- **Level 2 Width**: Set the cell size if the second level of grids for the dynamic index. The unit of this parameter is the same with that of the dataset.
- **Level 3 Width**: Set the cell size if the third level of grids for the dynamic index. This parameter cannot be set to 0. The unit of this parameter is the same with that of the dataset.

**Note**: Dynamic index is supported in the following engine types: SQLPlus, OraclePlus, DB2, MySQL.
Note

1. SuperMap iDesktop 9D supports r-tree index, quadtree index, dynamic index and tile index.
2. SDBPlus, UDB and POSTGRESQI datasource only support r-tree index.
3. DB2 datasource only support dynamic index.
4. Other database datasources support r-tree index, quadtree index, dynamic index and tile index.
5. Network dataset does not support any spatial index.
6. Related Topics
7. For more information on different types of spatial indexes, please refer to Spatial Index Types.
8. The asterisk * in the title of this page indicates that this function is supported by SuperMap Express .NET and SuperMap Deskpro .NET, not SuperMap Viewer .NET.