Многомерни и битови индекси. Дървовидни структури за многомерни данни в MySQL

Валентина Динкова, ф.н.71112

3 юни 2010 г.

1 GIS и разширението на MySQL за пространствени данни

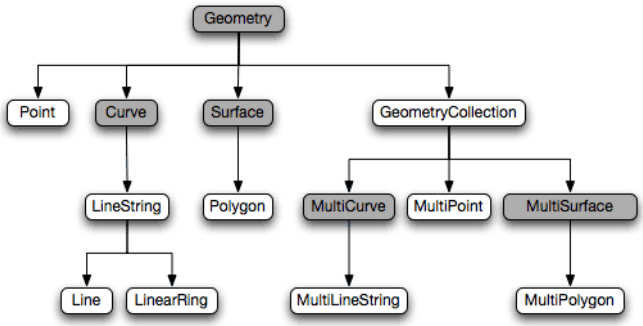
GIS означава Географска Информационна Система и е един от най-очевидните примери за пространствени данни.
OGC (Open Geospatial Consorcium) е организация, която работи по стандартизирането на различни области на GIS. Един такъв стандарт е и спецификацията за SQL, която определя разширението на SQL базирани релационни бази данни, което да използва GIS обекти и операции.

OGC работи в 4 важни области:

- типове данни;
- операции;
- възможност да се подават като вход и да се извеждат GIS данни;
- индексиране на пространствени данни.

Друга важна област са метаданните.

2 Стандартът, използван от почти всички SQL бази данни с пространствено разширение, включително и MySQL



Типовете, отбелязани в сиво са абстракти и обекти от тези типове не могат да се създават.

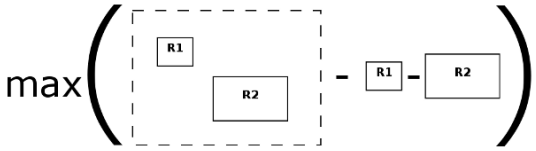
3 Пространствени индекси

Пространствените данни могат да се индексират също както останалите данни в MySQL. Но за да бъде индексирането ефективно, се използва пространствен тип индексиране, реализирано чрез R-дървета. MySQL използва R-дървета с квадратично разделяне.
Не всички engine-и поддържат многомерни индекси.

3.1 R-дървета с квадратично разделяне

Добавяме нов запис. Нека сме намерили листото, където трябва да добавим новия запис и нека $M =$ *"брй региони в листо"*.

1. Избираме 2 от $M + 1$ записа да бъдат първите елементи на двете нови листа, като избираме двойката,която би заела най-много място ако и двата елемента се постават на едно място (двойката при която покриващия регион ще е най-голям). Намираме тази двойка като от областта покриваща двата записа изваждаме самите записи и искаме тази разлика да е най-голяма.



2. Останалите записи разделяме в двете листа един по един. На всяка стъпка разширяването, необходимо за добавянето на всеки от оставащите записи към всяко листо се изчислява и добавеният запис е този, който е показал най-голяма разлика спрямо двете листа.

3.2 Примери с *MySQL*

Създаваме таблицата *map_test*, **където** *loc* **е пространствен атрибут**

```
mysql> create table map_test
    -> (
    ->   name varchar(100) not null primary key,
    ->   loc  geometry not null,
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Добавяме данни

```
mysql> insert into map_test values ('One Two', point(1,2));
Query OK, 1 row affected (0.00 sec)

mysql> insert into map_test values ('Two Two', point(2,2));
Query OK, 1 row affected (0.00 sec)

mysql> insert into map_test values ('Two One', point(2,1));
Query OK, 1 row affected (0.00 sec)
```

Ето как изглежда *map_test* сега:

```
mysql> select name, AsText(loc) from map_test;
+---------+-------------+
| name    | AsText(loc) |
+---------+-------------+
| One Two | POINT(1 2)  |
| Two Two | POINT(2 2)  |
| Two One | POINT(2 1)  |
+---------+-------------+
3 rows in set (0.00 sec)
```

Заявка за проверка коя точка се съдържа в полигона

```
mysql> SELECT name, AsText(loc) FROM map_test WHERE
    -> Contains(
    -> GeomFromText('POLYGON((0 0, 0 1, 1 1, 2 0, 0 0))'),
    -> loc) = 1;
+---------+-------------+
| name    | AsText(loc) |
+---------+-------------+
| Two One | POINT(2 1)  |
+---------+-------------+
1 row in set (0.04 sec)
```

Сега създаваме пространствен индекс по атрибута *loc*

```
mysql> create spatial index ps_index on map_test(loc);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

И отново правим същата заявка

```
mysql>  SELECT name, AsText(loc) FROM map_test WHERE
    ->  Contains(
    ->  GeomFromText('POLYGON((0 0, 0 1, 1 1, 2 0, 0 0))'),
    ->  loc) = 1;
+---------+-------------+
| name    | AsText(loc) |
+---------+-------------+
| Two One | POINT(2 1)  |
+---------+-------------+
1 row in set (0.00 sec)
```