寻找附近的人解决方案初探

主讲人：郭 栋
时　间：2014-06-23

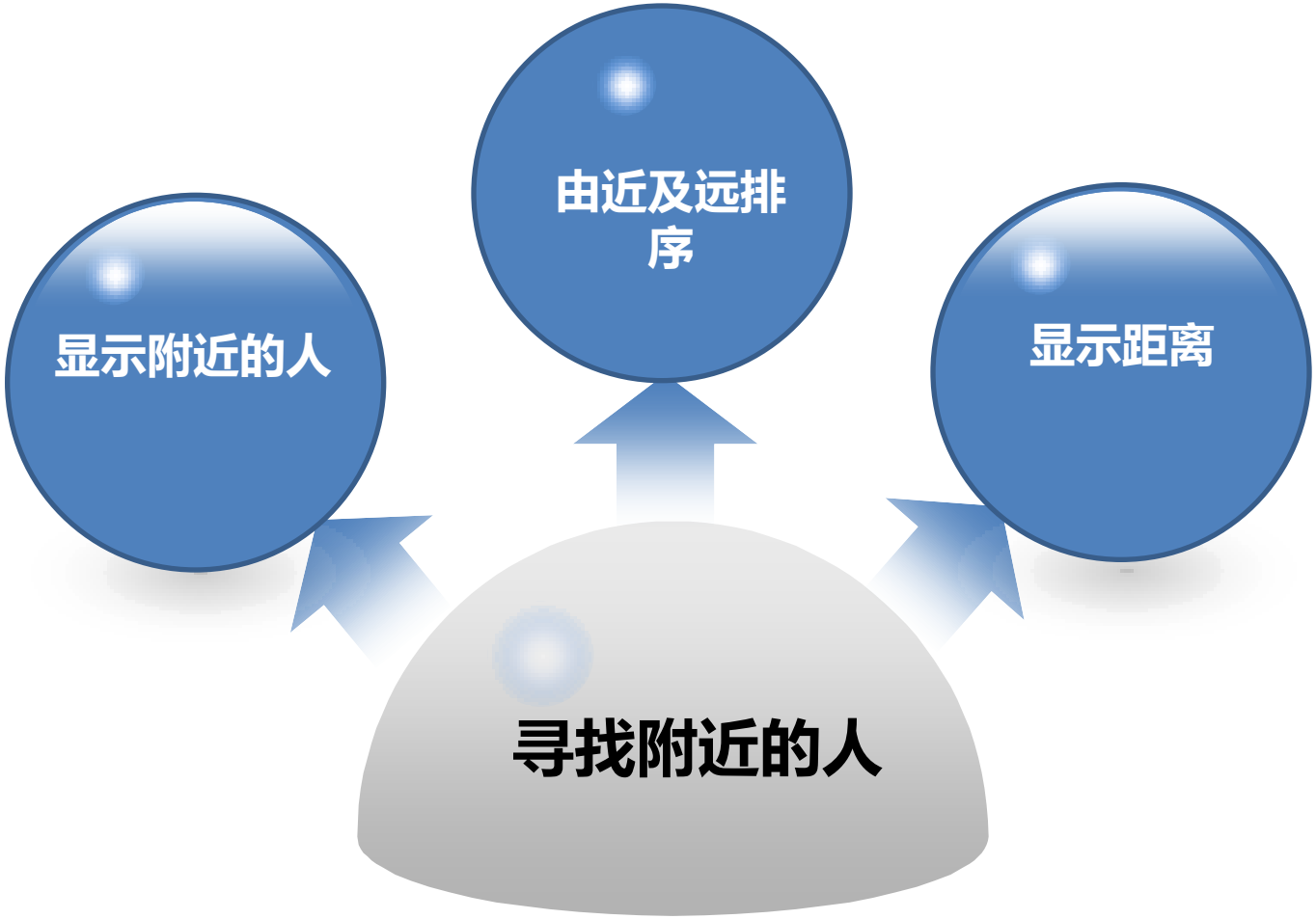
主要内容

1. 开发背景
2. 功能需求
3. 解决方案
4. 总结与展望

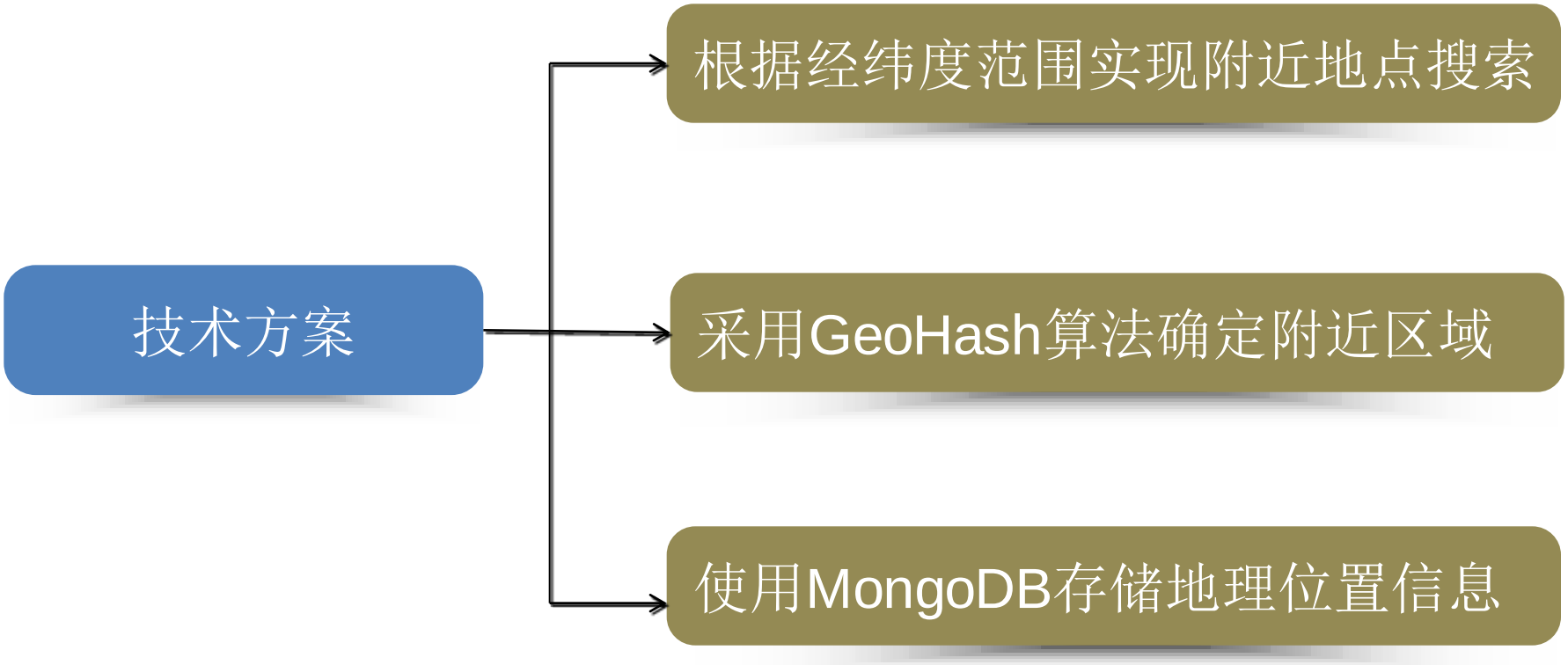
开发背景

- 公司需要开发一款基于基于位置的服务(LBS)的软件，其中包含的功能有：
  1、我的足迹
  2、我的轨迹
  3、附近的人（重点）

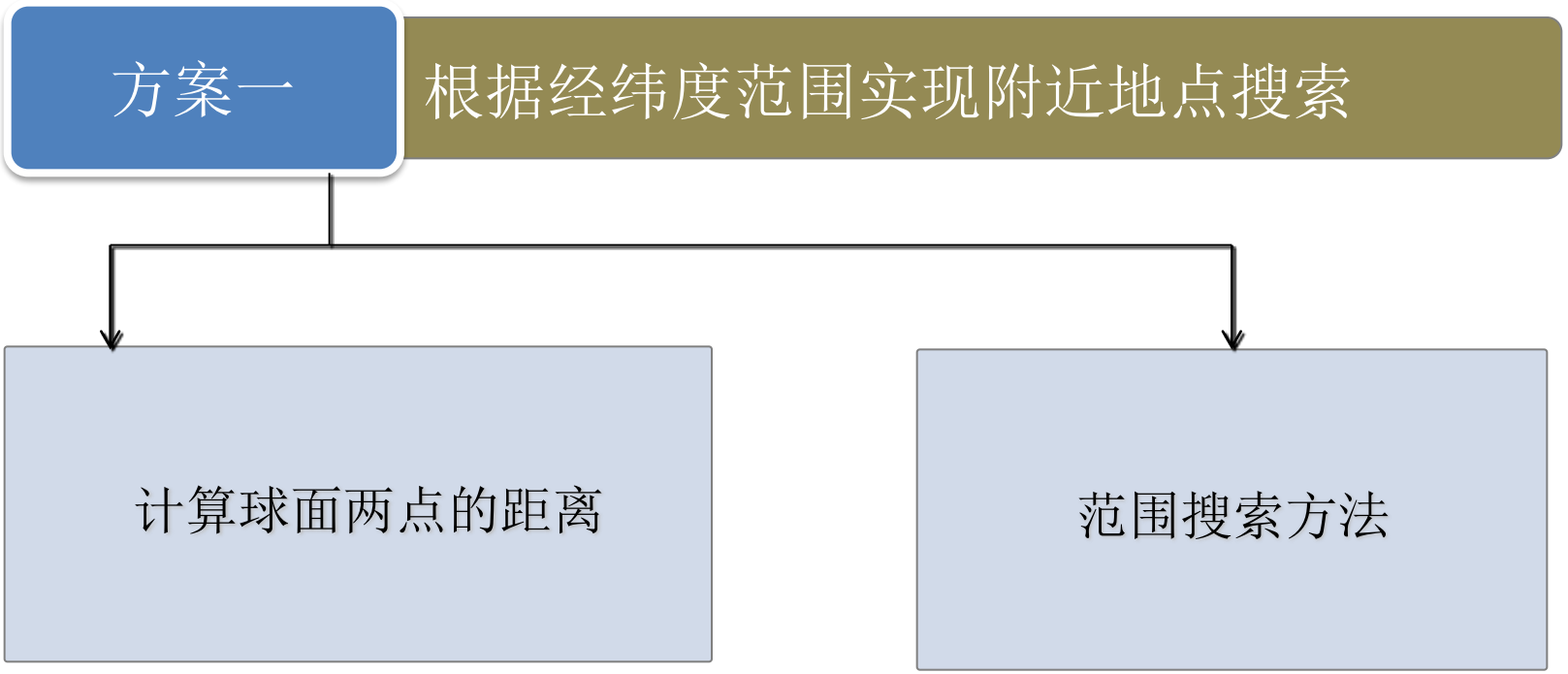
功能需求



技术方案


技术方案
根据经纬度范围实现附近地点搜索
采用GeoHash算法确定附近区域
使用MongoDB存储地理位置信息

方案一



范围搜索方法

优缺点

- **优点：**

1、将距离转化成经纬度范围，利用经纬度上的索引，提高查询效率。

- **缺点：**

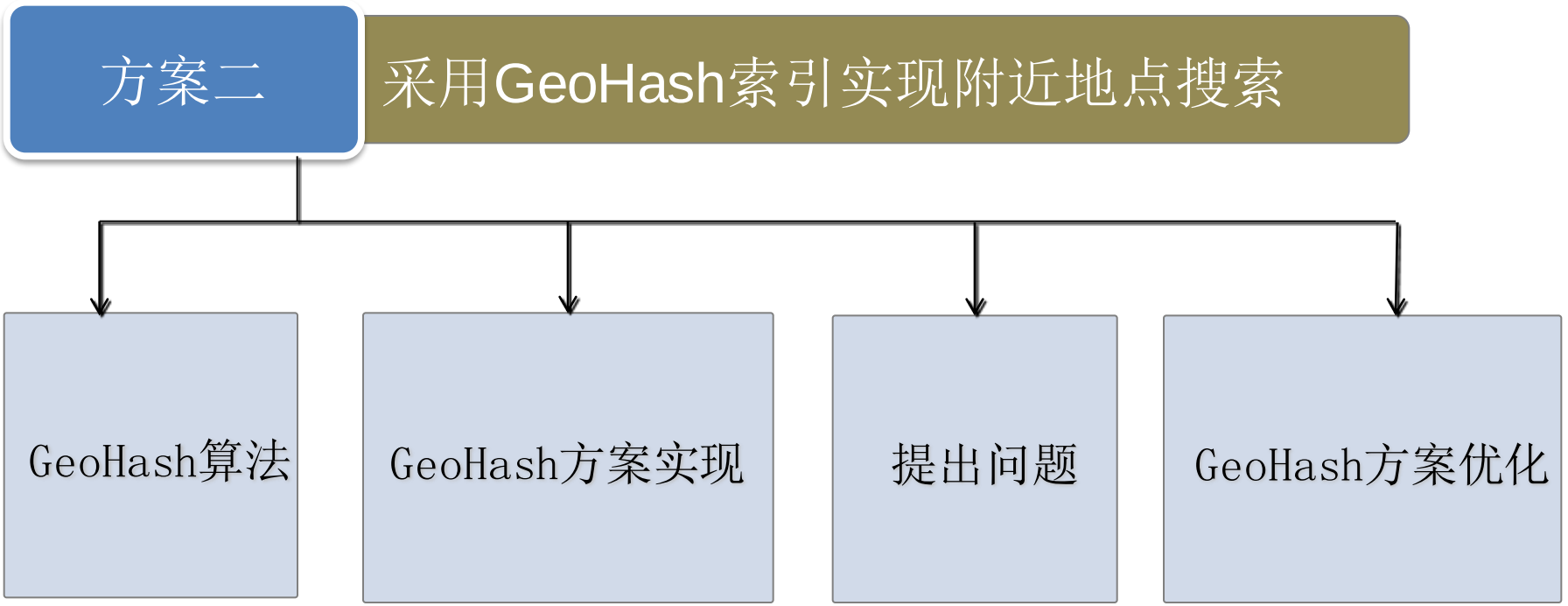
1、数据库支持添加双索引;

2、支持一次查询中同时使用两个索引;

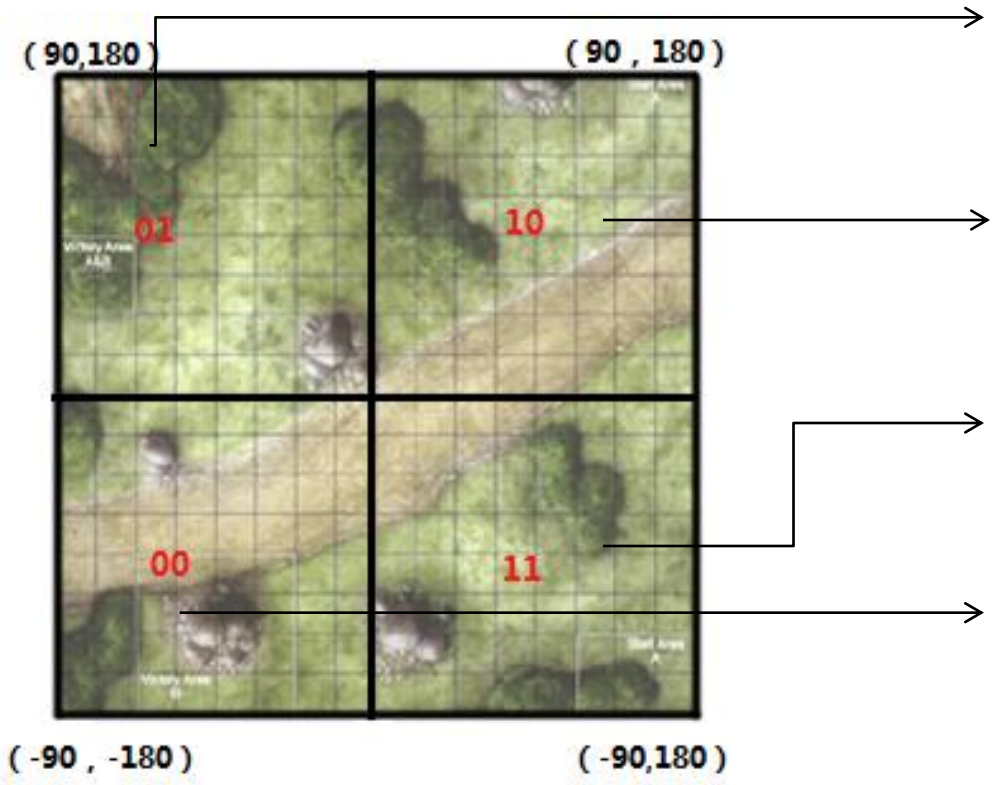
3、SQL语句极其不稳定，查询结果很难缓存;

```
SELECT * FROM places WHERE ((lat BETWEEN ? AND ?) AND (lng BETWEEN ? AND ?))
```

方案二


方案二
采用GeoHash索引实现附近地点搜索
GeoHash算法
GeoHash方案实现
提出问题
GeoHash方案优化

GeoHash算法



01

经度（-180,0）
纬度（0,90）

10

经度（0,180）
纬度（0,90）

11

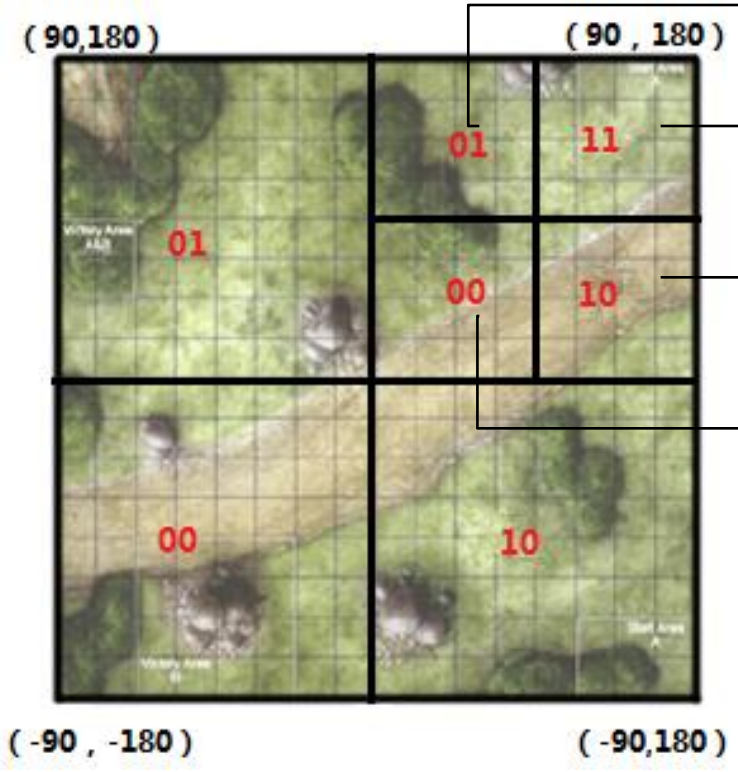
经度（0,180）
纬度（-90,0 ）

00

经度（-180,0）
纬度（-90,0）

示例：(39.92324, 116.3906)

GeoHash算法



11 01

11 11

11 10

11 00

经度（0,90）
纬度（45,90）

经度（90,180）
纬度（45,90）

经度（90,180）
纬度（0,45）

经度（0,90）
纬度（0,45）

示例：(39.92324, 116.3906)

划分纬度

纬度范围	划分区间0	划分区间1	39.92324所属区间
(-90, 90)	(-90, 0.0)	(0.0, 90)	1
(0.0, 90)	(0.0, 45.0)	(45.0, 90)	0
(0.0, 45.0)	(0.0, 22.5)	(22.5, 45.0)	1
(22.5, 45.0)	(22.5, 33.75)	(33.75, 45.0)	1
(33.75, 45.0)	(33.75, 39.375)	(39.375, 45.0)	1
(39.375, 45.0)	(39.375, 42.1875)	(42.1875, 45.0)	0
(39.375, 42.1875)	(39.375, 40.7812)	(40.7812, 42.1875)	0
(39.375, 40.7812)	(39.375, 40.0781)	(40.0781, 40.7812)	0
(39.375, 40.0781)	(39.375, 39.7265)	(39.7265, 40.0781)	1
(39.7265, 40.0781)	(39.7265, 39.9023)	(39.9023, 40.0781)	1

划分经度

经度范围	划分区间0	划分区间1	116.3906区间
(-180, 180)	(-180, 0.0)	(0.0, 180)	1
(0.0, 180)	(0.0, 90.0)	(90.0, 180)	1
(90.0, 180)	(90.0, 135.0)	(135.0, 180)	0
(90.0, 135.0)	(90.0, 112.5)	(112.5, 135.0)	1
(112.5, 135.0)	(112.5, 123.75)	(123.75, 135.0)	0
(112.5, 123.75)	(112.5, 118.125)	(118.125, 123.75)	0
(112.5, 118.125)	(112.5, 115.312)	(115.312, 118.125)	1
(115.312, 118.125)	(115.312, 116.718)	(116.718, 118.125)	0
(115.312, 116.718)	(115.312, 116.015)	(116.015, 116.718)	1
(116.015, 116.718)	(116.015, 116.367)	(116.367, 116.718)	1

Base32编码

- **合并经纬度：**

  经度度和纬度的编码合并，经度在前，纬度后，得到编码11100 11101 00100 01111 00000 01101 01011 00001。

- **Base32编码：**

  用0-9、b-z（去掉a, i, l, o）这32个字母进行base32编码，得到(39.92324, 116.3906)的编码为wx4g0ec1。

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
base32	0	1	2	3	4	5	6	7	8	9	b	c	d	e	f	g
十进制	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
base32	h	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z

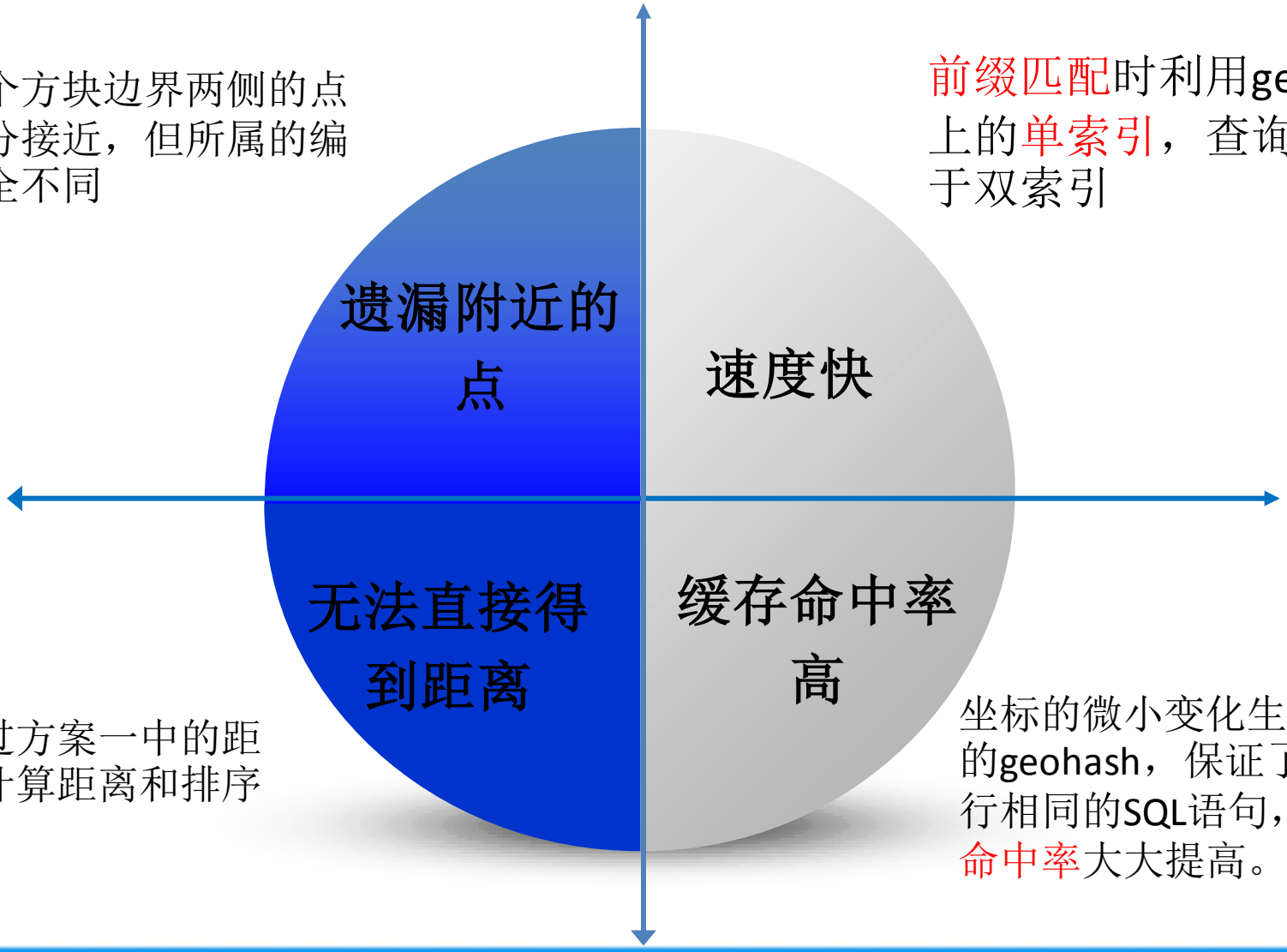
GeoHash方案实现

id	user_id	longitude	latitude	time	geohash
18	351565053248644	114.3480907	30.5269561	2014-05-15 09:37:29	wt3mdj38
19	358472045454005	114.3380907	30.5169561	2014-05-16 15:53:45	wt3m9g6q
20	358512032147722	114.3200000	30.5169561	2013-06-17 14:30:08	wt3m97mq
21	359836040135646	114.3367890	30.5169561	2013-06-17 14:30:41	wt3m9g3w
22	359836040135647	114.4597704	30.4169561	2013-06-17 15:39:59	wt3mh95p
23	359836040135648	114.3380907	30.5002608	2013-06-17 16:03:14	wt3m9b6k
37	359836040135649	114.3980907	30.5169561	2014-05-12 16:07:30	wt3me5qn
38	359836040135650	114.0334300	30.5152960	2014-05-12 16:08:25	wt3jdghs
39	359836040135651	114.3360000	30.5169561	2014-05-15 09:35:35	wt3m9g2y
40	359836040135652	114.3350000	30.5169561	2014-05-16 09:36:01	wt3m9g2n
41	359836040135653	114.4000000	30.5169561	2014-05-16 09:36:26	wt3me5rw
42	359836040135654	114.3480907	30.5169561	2014-05-15 09:36:43	wt3md53w
44	861344022478010	114.3580907	30.5169561	2014-05-15 10:28:01	wt3md73n
245	359836040135700	114.4764828	30.6675901	1970-01-01 00:00:00	wt3qkyz5
246	359836040135701	114.4688087	30.6728357	1970-01-01 00:00:00	wt3qkzcd
247	359836040135702	114.4621483	30.6759528	1970-01-01 00:00:00	wt3qs8ku
248	359836040135703	114.4213348	30.6934155	1970-01-01 00:00:00	wt3qeex4
249	359836040135704	114.4907538	30.6980434	1970-01-01 00:00:00	wt3qtk3t
250	359836040135705	114.4858278	30.6789798	1970-01-01 00:00:00	wt3qt0yn

优缺点

靠近每个方块边界两侧的点虽然十分接近，但所属的编码会完全不同

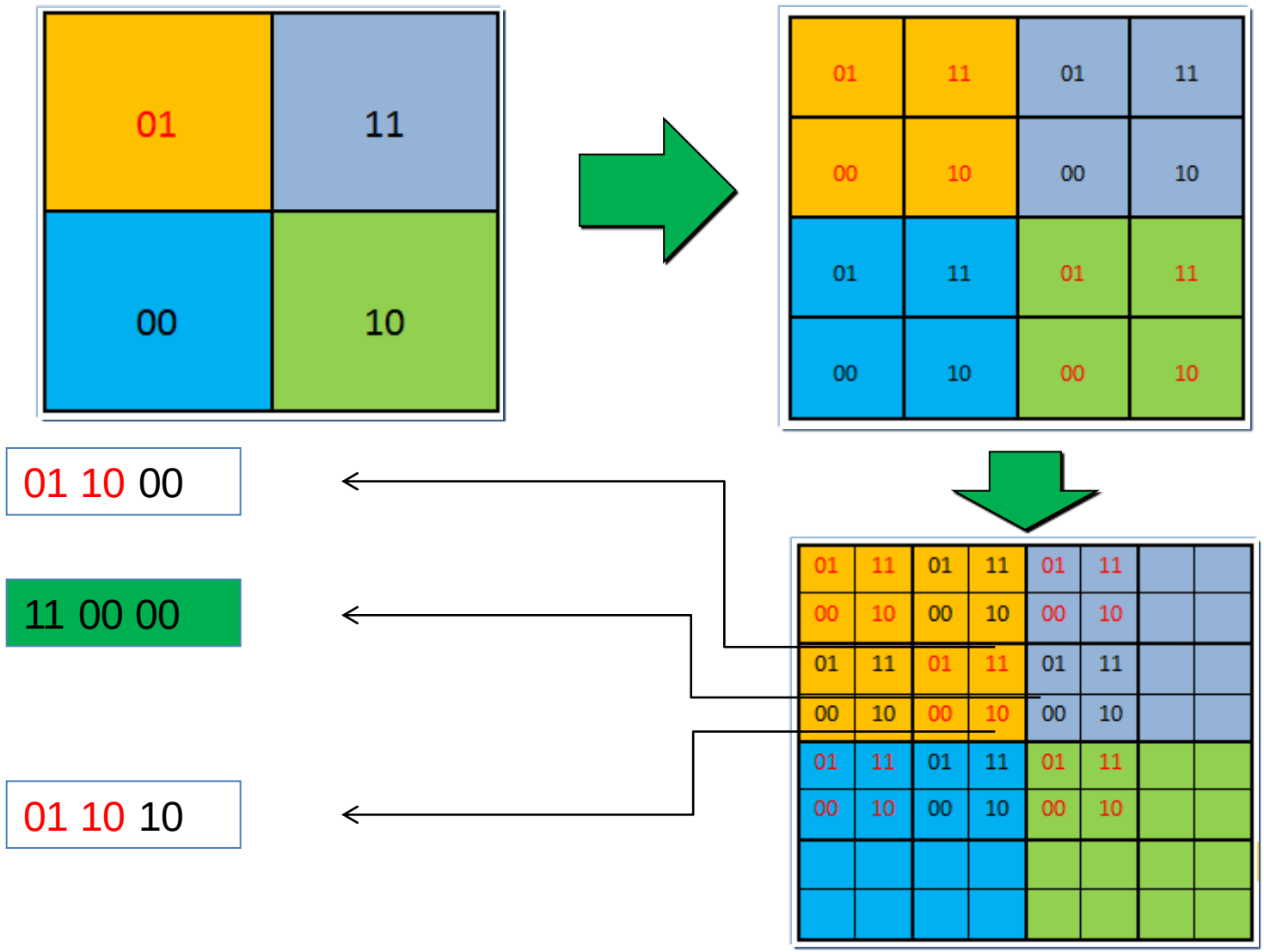
前缀匹配时利用geohash列上的单索引，查询效率高于双索引



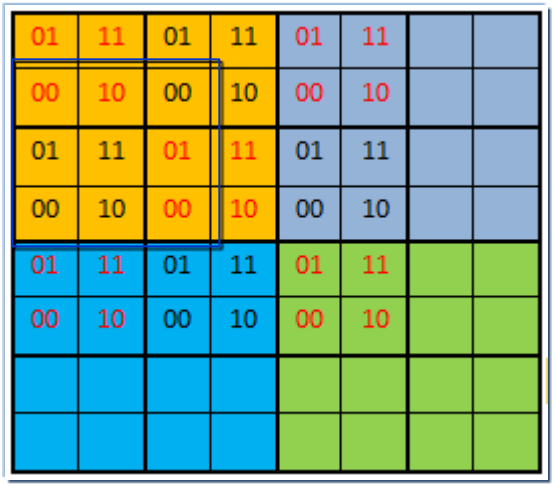
需要通过方案一中的距离公式计算距离和排序

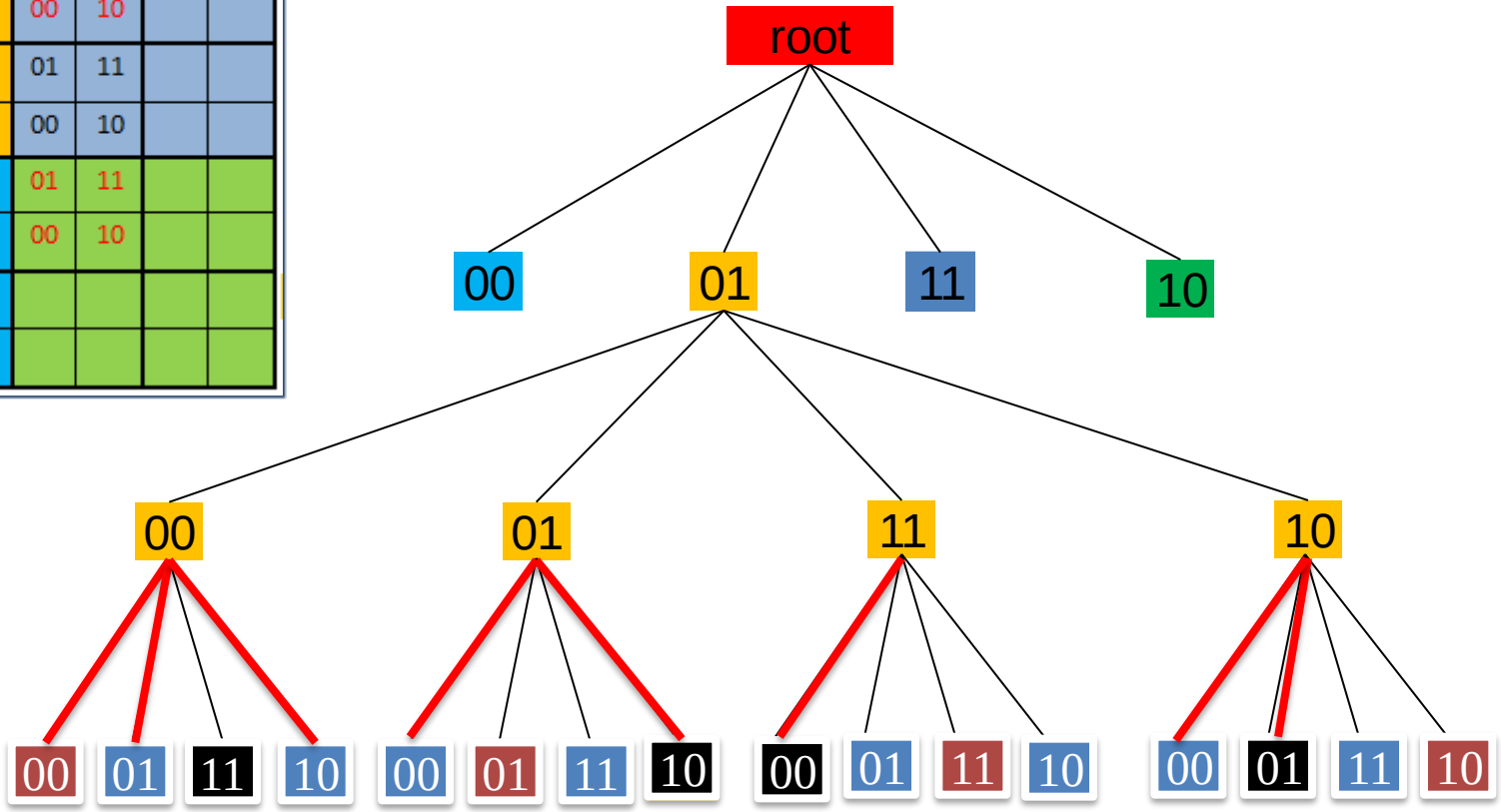
坐标的微小变化生成相同的geohash，保证了每次执行相同的SQL语句，缓存命中率大大提高。

提出问题


01
11
00
10
01 11 01 11
00 10 00 10
01 11 01 11
00 10 00 10
01 10 00
11 00 00
01 10 10

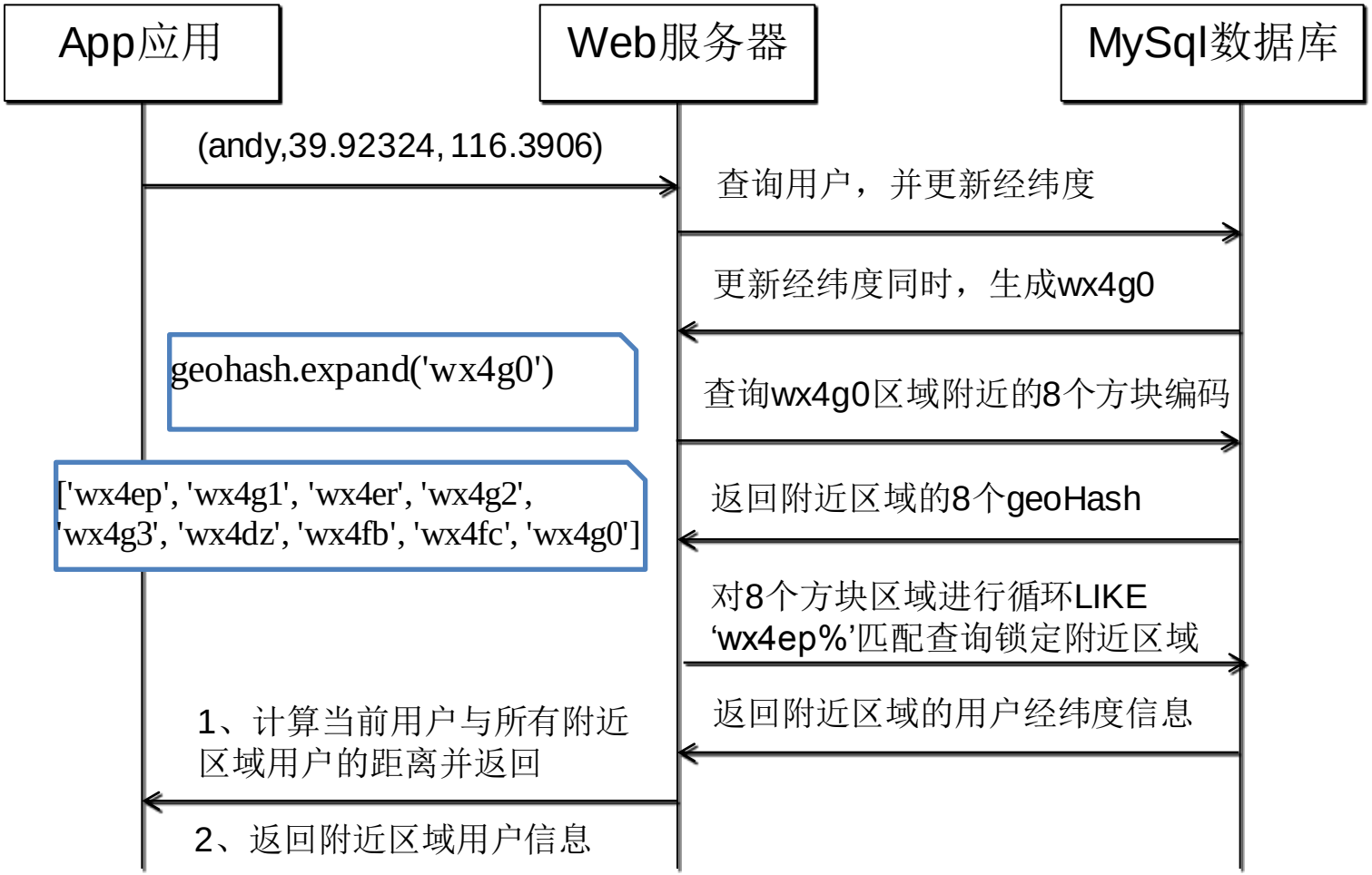
寻找附近8个方块





[GeoHash获取8个方块演示]

GeoHash方案优化



总结与展望

1. 找到了几种解决方案并进行了比较
2. 重点对GeoHash算法进行了分析
3. 对GeoHash算法寻找附近8个方块算法实现还需琢磨
4. 使用MongoDB存储地理位置信息有待尝试

THANK YOU !