Geo/Spatial Search with MySQL

Alexander Rubin

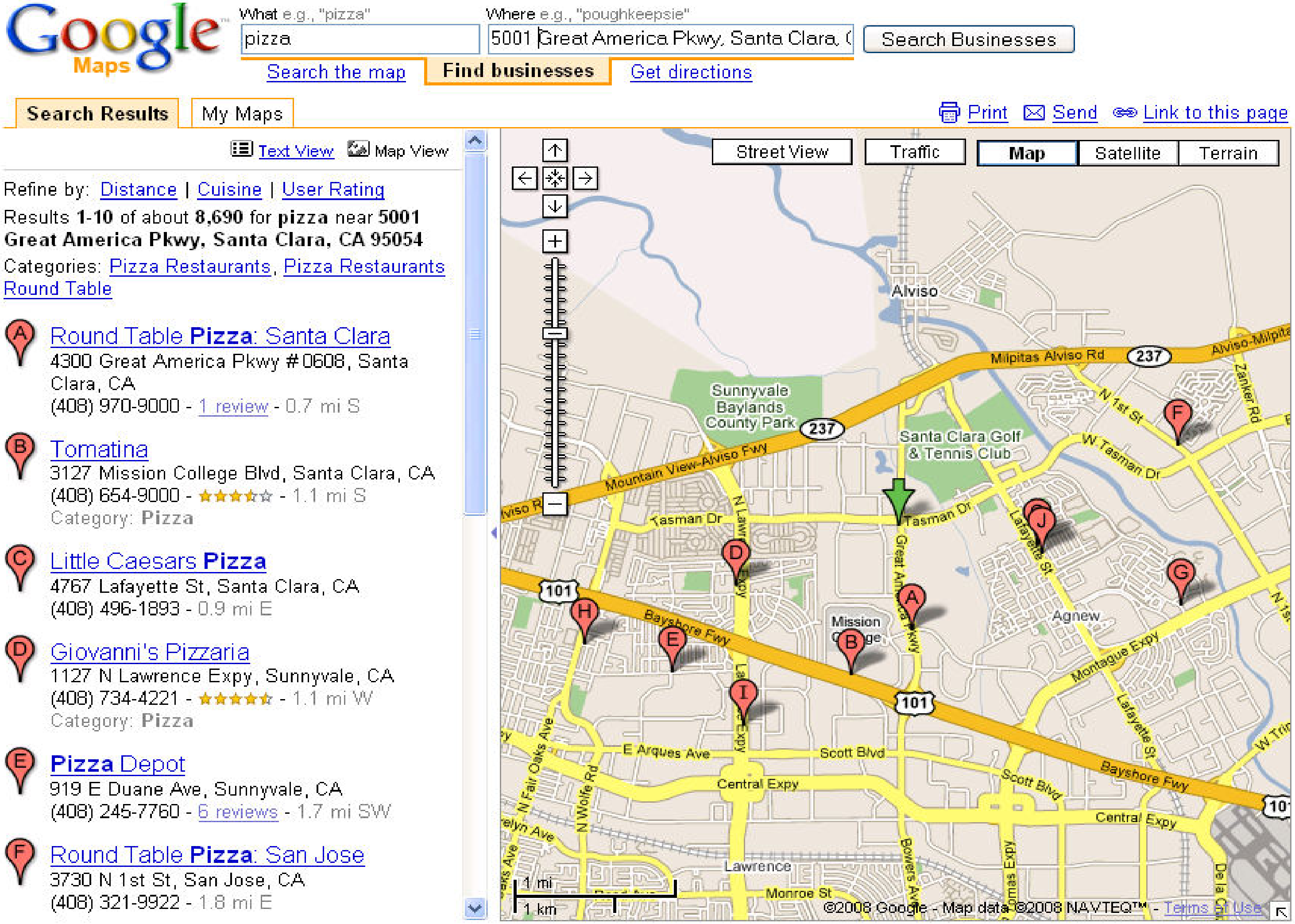
Senior Consultant, MySQL AB

Copyright 2006 MySQL AB

Why Geo Search?

- Stores: find locations new you
- Social networks: find friends close to you
- Online maps: find points of interest near your position
- Online newspapers/yellow pages: find show times next to you home.

POI Search Example



What e.g., "pizza": pizza

Where e.g., "poughkeepsie": 5001 Great America Pkwy, Santa Clara, (

Refine by: Distance | Cuisine | User Rating

Results **1-10** of about **8,690** for **pizza** near **5001 Great America Pkwy, Santa Clara, CA 95054**

Categories: Pizza Restaurants, Pizza Restaurants Round Table

A. Round Table **Pizza**: Santa Clara
4300 Great America Pkwy #0608, Santa Clara, CA
(408) 970-9000 - 1 review - 0.7 mi S

B. Tomatina
3127 Mission College Blvd, Santa Clara, CA
(408) 654-9000 - ★★★★☆ - 1.1 mi S
Category: **Pizza**

C. Little Caesars **Pizza**
4767 Lafayette St, Santa Clara, CA
(408) 496-1893 - 0.9 mi E

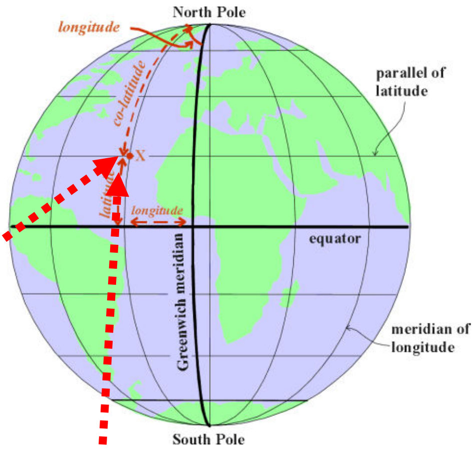
D. Giovanni's Pizzaria
1127 N Lawrence Expy, Sunnyvale, CA
(408) 734-4221 - ★★★★★ - 1.1 mi W
Category: **Pizza**

E. **Pizza** Depot
919 E Duane Ave, Sunnyvale, CA
(408) 245-7760 - 6 reviews - 1.7 mi SW

F. Round Table **Pizza**: San Jose
3730 N 1st St, San Jose, CA
(408) 321-9922 - 1.8 mi E

Common Tasks

- Task: Find 10 nearby hotels *and sort by distance*



- What do we have:

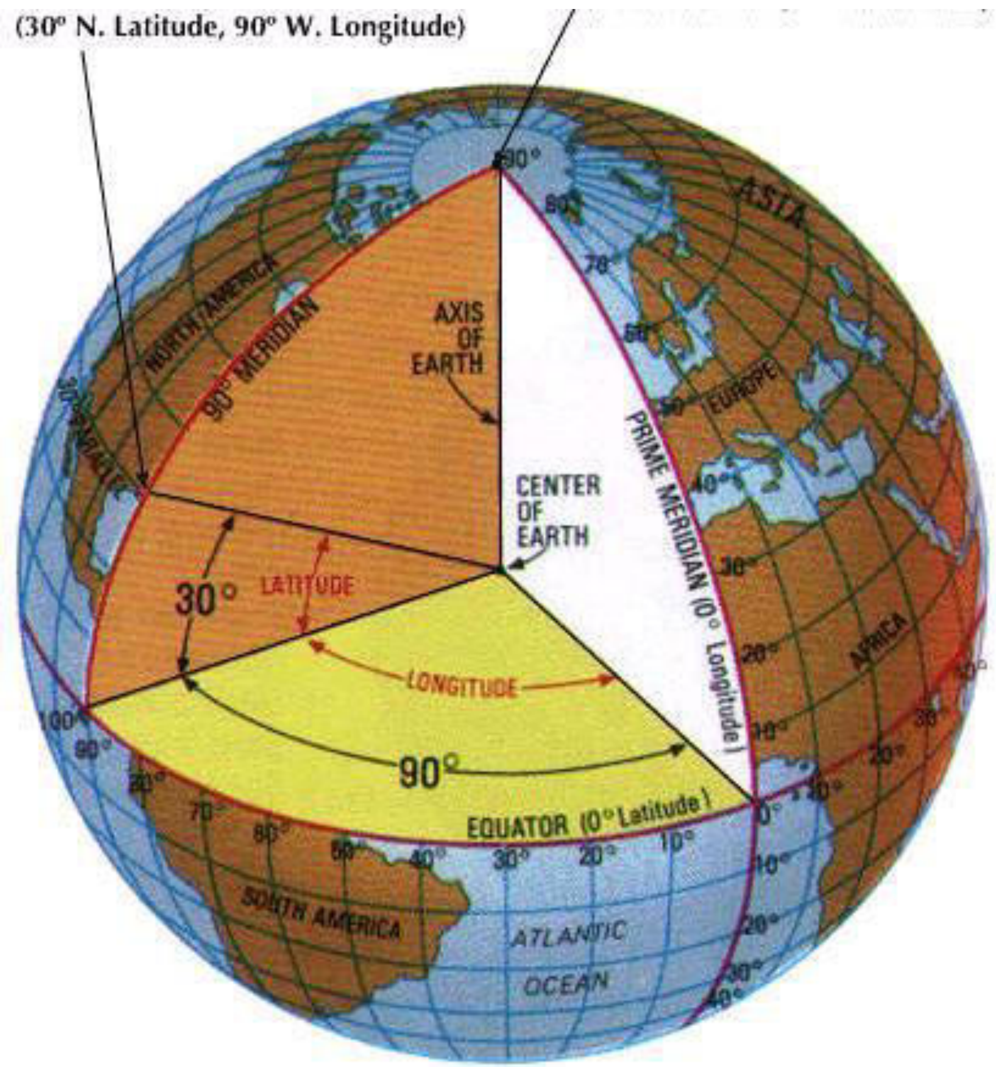
1. Given point on Earth: Latitude, Longitude
2. Hotels table:

Hotel Name	Latitude	Longitude

- ***Question: How to calculate distance between us and hotel?***

Copyright 2006 MySQL AB

Latitudes and Longitudes



Distance between 2 points
The Haversine Formula

For two points on a sphere (of radius R) with latitudes φ1 and φ2, latitude separation Δφ = φ1 − φ2, and longitude separation Δλ the distance d between the two points:

$$\boxed{\operatorname{haversin}\left(\frac{d}{R}\right) = \operatorname{haversin}(\Delta\phi) + \cos(\phi_1)\cos(\phi_2)\ \operatorname{haversin}(\Delta\lambda)}$$

$$\operatorname{haversin}(\theta) = \frac{\operatorname{versin}(\theta)}{2} = \sin^2\left(\frac{\theta}{2}\right)$$

$$\operatorname{versin}(\theta) = 1 - \cos(\theta) = 2\sin^2\left(\frac{\theta}{2}\right)$$

Copyright 2006 MySQL AB

The Haversine Formula in MySQL

R = earth's radius

Δlat = lat2− lat1; Δlong = long2− long1
*a = sin²(Δlat/2) + cos(lat1) * cos(lat2) * sin²(Δlong/2)*
*c = 2*atan2(√a, √(1−a)); d = R*c*

angles need to be in radians

3956 * 2 * ASIN (SQRT (POWER(SIN((orig.lat - dest.lat)*pi()/180 / 2), 2) + COS(orig.lat * pi()/180) * COS(dest.lat * pi()/180) * POWER(SIN((orig.lon - dest.lon) * pi()/180 / 2), 2))) as distance

Copyright 2006 MySQL AB

MySQL Query: Find Nearby Hotels

```
set @orig_lat=122.4058; set @orig_lon=37.7907;
set @dist=10;

SELECT *, 3956 * 2 * ASIN(SQRT(
POWER(SIN((@orig_lat - abs(dest.lat)) * pi()/180 / 2),
2) +  COS(@orig_lat * pi()/180 ) * COS(abs(dest.lat) *
pi()/180) *  POWER(SIN((@orig_lon – dest.lon) *
pi()/180 / 2), 2) )) as  distance
FROM hotels dest
having distance < @dist
ORDER BY distance limit 10;
```

Lat can be negative!

Find Nearby Hotels: Results

```
+-----------------+---------+--------+--------+
| hotel_name      | lat     | lon    | dist   |
+-----------------+---------+--------+--------+
| Hotel Astori..  | 122.41  | 37.79  | 0.0054 |
| Juliana Hote..  | 122.41  | 37.79  | 0.0069 |
| Orchard Gard..  | 122.41  | 37.79  | 0.0345 |
| Orchard Gard..  | 122.41  | 37.79  | 0.0345 |
...
+-----------------+---------+--------+--------+
10 rows in set (4.10 sec)
```

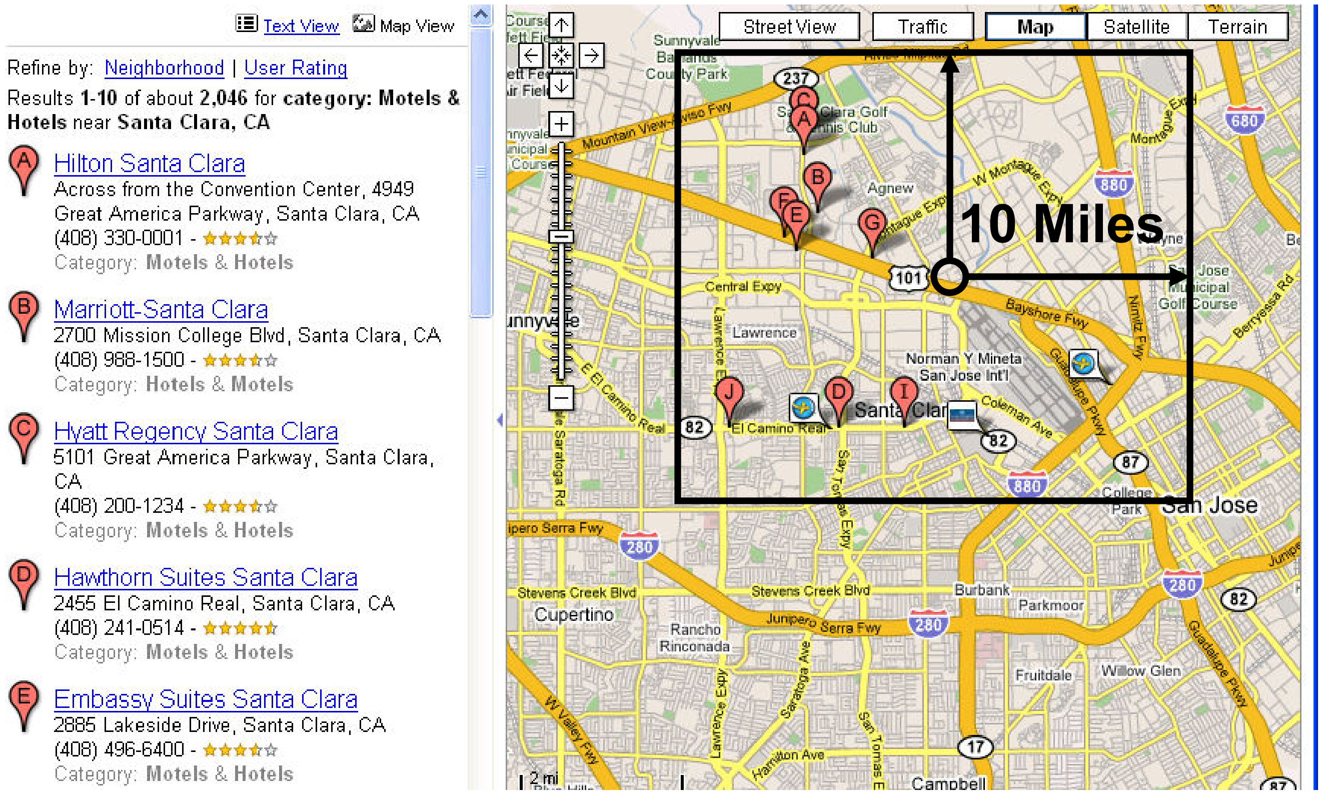
- 4 seconds - very slow for web query!

MySQL Explain query

```
Mysql> Explain …
select_type: SIMPLE
        table: dest
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1787219
        Extra: Using filesort
1 row in set (0.00 sec)
```

How to speed up the query

- We only need hotels in 10 miles radius
  - no need to scan the whole table



Text View | Map View

Refine by: Neighborhood | User Rating

Results **1-10** of about **2,046** for **category: Motels & Hotels** near **Santa Clara, CA**

A [Hilton Santa Clara](#)
Across from the Convention Center, 4949 Great America Parkway, Santa Clara, CA
(408) 330-0001 - ★★★★☆
Category: **Motels & Hotels**

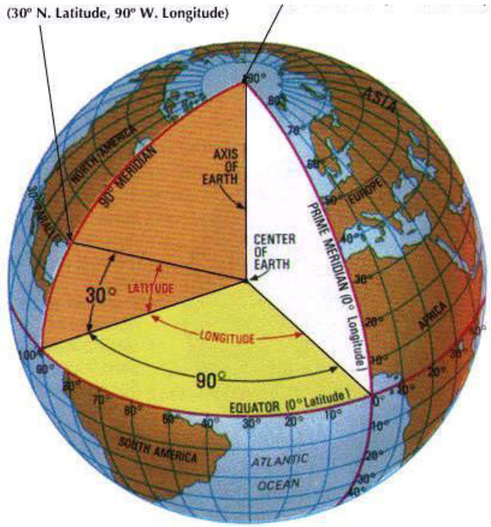
B [Marriott-Santa Clara](#)
2700 Mission College Blvd, Santa Clara, CA
(408) 988-1500 - ★★★★☆
Category: **Hotels & Motels**

C [Hyatt Regency Santa Clara](#)
5101 Great America Parkway, Santa Clara, CA
(408) 200-1234 - ★★★★☆
Category: **Motels & Hotels**

D [Hawthorn Suites Santa Clara](#)
2455 El Camino Real, Santa Clara, CA
(408) 241-0514 - ★★★★★
Category: **Motels & Hotels**

E [Embassy Suites Santa Clara](#)
2885 Lakeside Drive, Santa Clara, CA
(408) 496-6400 - ★★★★☆
Category: **Motels & Hotels**

How to calculate needed coordinates



- 1° of latitude ~= 69 miles
- 1° of longitude ~= cos(latitude)*69
- To calculate lon and lat for the rectangle:

```
set lon1 = mylon-
dist/abs(cos(radians(mylat))*69);
set lon2 =
mylon+dist/abs(cos(radians(mylat))*69);
set lat1 = mylat-(dist/69);
set lat2 = mylat+(dist/69);
```

Copyright 2006 MySQL AB

Modify the query

```
SELECT  destination.*,
3956 * 2 * ASIN(SQRT(  POWER(SIN((orig.lat - dest.lat)  *
pi()/180 / 2), 2) +
COS(orig.lat * pi()/180)  *  COS(dest.lat * pi()/180)  *
POWER(SIN((orig.lon -dest.lon)  * pi()/180 / 2), 2)  )) as
distance
FROM  users destination,  users origin
WHERE  origin.id=userid
and  destination.longitude
between lon1 and lon2
and destination.latitude
between lat1 and lat2
```

Stored procedure

```
CREATE PROCEDURE geodist (IN userid int, IN dist int)
   BEGIN
declare mylon double;  declare mylat double;
declare lon1 float;  declare lon2 float;
declare lat1 float; declare lat2 float;
-- get the original lon and lat for the userid
select longitude, latitude into mylon, mylat from users5
   where id=userid limit 1;
-- calculate lon and lat for the rectangle:
set lon1 = mylon-dist/abs(cos(radians(mylat))*69);
set lon2 = mylon+dist/abs(cos(radians(mylat))*69);
set lat1 = mylat-(dist/69);  set lat2 = mylat+(dist/69);
```

Stored Procedure, Contd

```
-- run the query:
SELECT  destination.*,
3956 * 2 * ASIN(SQRT(  POWER(SIN((orig.lat - dest.lat)
* pi()/180 / 2), 2) +
COS(orig.lat * pi()/180) *  COS(dest.lat * pi()/180) *
POWER(SIN((orig.lon -dest.lon) * pi()/180 / 2), 2)  )) as
distance FROM  users destination,  users origin
WHERE  origin.id=userid
and  destination.longitude between lon1 and lon2
and destination.latitude between lat1 and lat2
having distance < dist ORDER BY Distance limit 10;
END $$
```

Copyright 2006 MySQL AB

Speed comparison

- Test data: US and Canada zip code table, 800K records
- Original query (full table scan):
  - 8 seconds
- Optimized query (stored procedure):
  - 0.06 to 1.2 seconds (depending upon the number of POIs/records in the given radius)

Stored Procedure: Explain Plan

```
Mysql>CALL geodist(946842, 10)\G
```

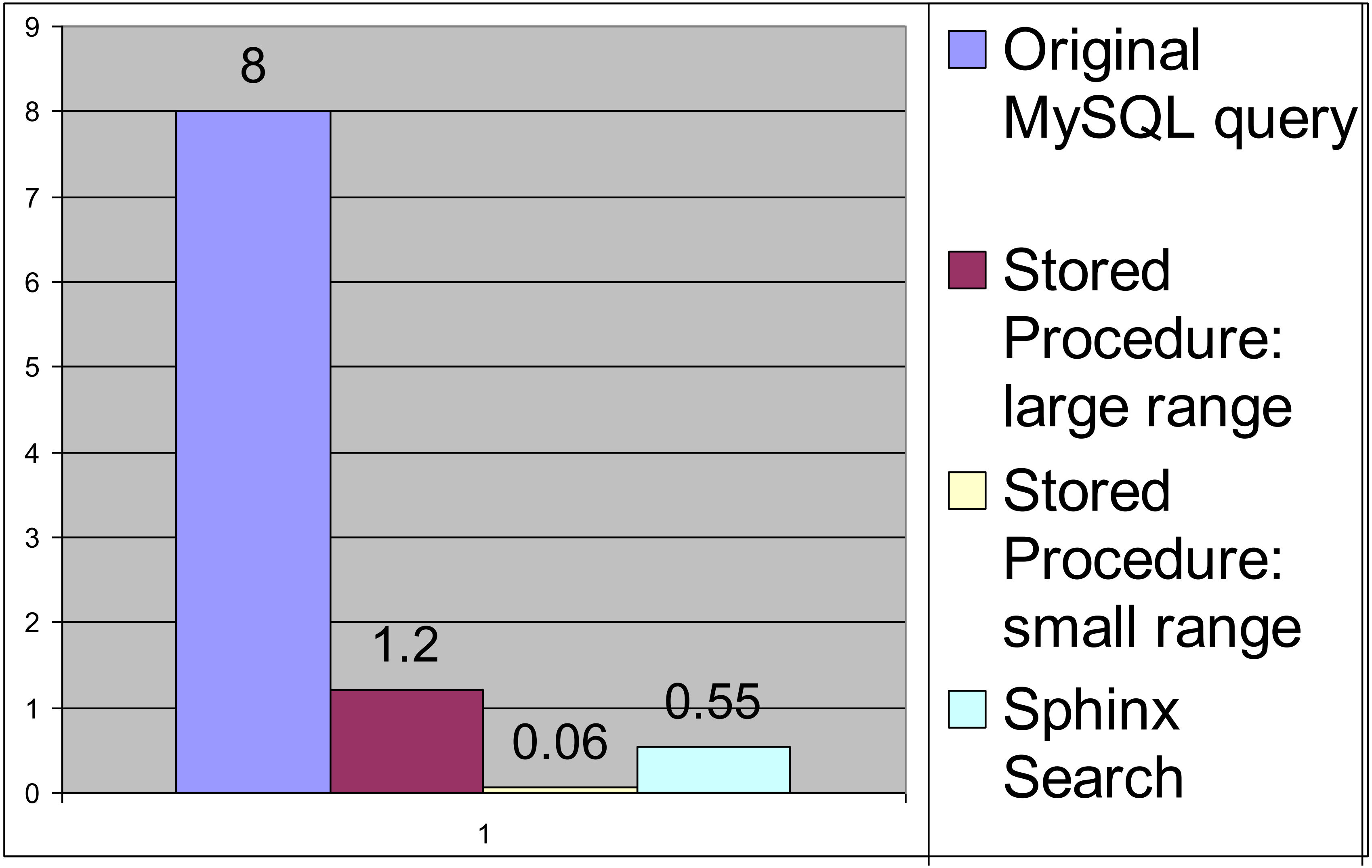
```
table: origin
type: const
key: PRIMARY
key_len: 4
ref: const
rows: 1, Extra: Using filesort

table: destination
type: range
key: latitude
key_len: 18
ref: NULL
rows: 25877, Extra: Using where
```

Geo Search with Sphinx

- Sphinx search (www.sphinxsearch.com) since 0.9.8 can perform geo distance searches
- It is possible to setup an "anchor point" in the api code and then use the "geodist" function and specify the radius.
- Sphinx Search returns in 0.55 seconds for test data regardless of the radius and zip

```
$ php test.php -i zipdist -s
  @geodist,asc Query '' retrieved
  1000 matches in 0.552 sec.
```

Speed comparison of all solutions



Different Type of Coordinates

- Decimal Degrees (what we used)
  - 37.3248 LAT, 121.9163 LON
- Degrees-minutes-second (used in most GPSes)
  - 37°19′29″N LAT, 121°54′59″E LON
- Most GPSes can be configured to use Decimal Degrees
- Other

Converting between coordinates

- **Degrees-Minutes-Seconds to Decimal Degrees**:
  - degrees + (minutes/60) + (seconds/3600)

```
CREATE FUNCTION `convert_from_dms`
(degrees INT, minutes int, seconds int)
RETURNS double DETERMINISTIC
BEGIN
RETURN degrees + (minutes/60) +
   (seconds/3600);
END $$
mysql>select convert_from_dms (46, 20,
  10) as DMS\G
dms: 46.3361111
```

Geo Search with Full Text search

- Sometimes we need BOTH geo search and full text search
- Example 1: find 10 nearest POIs, with “school” in the name
- Example 2: find nearest streets, name contains “OAK”
- Create FullText index and index on LAT, LON
  - `Alter table geonames add fulltext key (name);`
  - MySQL will choose which index to use

Geo Search with Full Text search: example

- Grab POI data from www.geonames.org, upload it to MySQL, add full text index

```
Mysql> SELECT  destination.*,
3956 * 2 * ASIN(SQRT(POWER(SIN((orig.lat
  - dest.lat) * pi()/180 / 2), 2) +
  COS(orig.lat * pi()/180) *
  COS(dest.lat * pi()/180) *
  POWER(SIN((orig.lon -dest.lon) *
  pi()/180 / 2), 2)  )) as distance
FROM  geonames destination
WHERE  match(name)
against (‘OAK’ in boolean mode)
having distance < dist ORDER BY Distance
  limit 10;
```

Geo Search with Full Text search: Explain

```
mysql> explain SELECT  destination.*,
3956 * 2 * ASIN(SQRT(POWER(SIN(…
```

```
table: destination
type: fulltext
possible_keys: name_fulltext
key: name_fulltext
key_len: 0
ref:
rows: 1
Extra: Using where; Using filesort
```

DEMO

DEMO: Find POI near us

- –Use GPS
- –All POIs near GPS point
- –Match keyword

Using MySQL Spatial Extension

```
CREATE TABLE `zipcode_spatial` (
`id` int(10) unsigned NOT NULL
  AUTO_INCREMENT,
`zipcode` char(7) NOT NULL, …
`lon` int(11) DEFAULT NULL,
`lat` int(11) DEFAULT NULL,
`loc` point NOT NULL,
PRIMARY KEY (`id`),
KEY `zipcode` (`zipcode`),
SPATIAL KEY `loc` (`loc`)
) ENGINE=MyISAM;
```

Zipcode with Spatial Extension

```
mysql> select zipcode, lat, lon,
  AsText(loc) from zipcode_spatial
  where city_name = 'Santa Clara'
  and state ='CA' limit 1\G

****** 1. row*********
 zipcode: 95050
        lat: 373519
        lon: 1219520
AsText(loc): POINT(1219520 373519)
```

Spatial Search: Distance

Spatial Extension: no built-in distance function

```
CREATE FUNCTION `distance`
(a POINT, b POINT)
RETURNS double DETERMINISTIC
BEGIN
RETURN round(glength(linestringfromwkb
(linestring(asbinary(a),
asbinary(b)))));
END $$
```

（forge.mysql.com/tools/tool.php?id=41）

Spatial Search Example

```
SELECT DISTINCT
 dest.zipcode,
distance(orig.loc, dest.loc) as sdistance
FROM
 zipcode_spatial orig,
 zipcode_spatial dest
WHERE
orig.zipcode = '27712'
having sdistance < 10
ORDER BY
sdistance limit 10;
```