

## [Jeremy Cole](#)

Geek, electronics nerd, database nerd, father of three.

« [On Hiring a MySQL DBA/Architect](#)  
[Proven Scaling goes global](#) »

# On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS

Geo-referencing IPs is, in a nutshell, converting an IP address, perhaps from an incoming web visitor, a log file, a data file, or some other place, into the name of some entity owning that IP address. There are a lot of reasons you may want to geo-reference IP addresses to country, city, etc., such as in simple ad targeting systems, geographic load balancing, web analytics, and many more applications.

This is a very common task, but I have never actually seen it done efficiently in MySQL in the wild. There is a lot of questionable advice [on forums, blogs, and other sites](#) out there on this topic. After working with a [Proven Scaling](#) customer, I recently did some thinking and some performance testing on this problem, so I thought I would publish some hard data and advice for everyone.

Unfortunately, R-tree (spatial) indexes have not been added to InnoDB yet, so the tricks in this entry only work efficiently with MyISAM tables (although they should work with InnoDB, they will perform poorly). This is actually OK for the most part, as the geo-referencing functionality most people need doesn't really need transactional support, and since the data tables are basically read-only (monthly replacements are published), the likelihood of corruption in MyISAM due to any server failures isn't very high.

## The data provided by MaxMind

[MaxMind](#) is a great company that produces [several geo-referencing databases](#). They release both a commercial (for-pay, but affordable) product called [GeoIP](#), and a free version of the same databases, called [GeoLite](#). The most popular of their databases that I've seen used is GeoLite Country. This allows you look up nearly any IP and find out which country (hopefully) its user resides in. The free GeoLite versions are normally good enough, at about 98% accurate, but the for-pay GeoIP versions in theory are more accurate. In this article I will refer to both GeoIP and GeoLite as "GeoIP" for simplicity.

GeoIP Country is available as a CSV file containing the following fields:

- ip from, ip to (text) — The start and end IP addresses as text in dotted-quad human readable format, e.g. "3.0.0.0". This is a handy way for a human to read an IP address, but a very inefficient way for a computer to store and handle IP addresses.
- ip from, ip to (integer) — The same start and end IP addresses as 32-bit integers<sup>1</sup>, e.g. 50331648.
- country code — The 2-letter ISO country code for the country to which this IP address has been assigned, or in some cases other strings, such as "A2" meaning "Satellite Provider".
- country name — The full country name of the same. This is redundant with the country code if you have a lookup table of country codes (including MaxMind's non-ISO codes), or if you make one from the GeoIP data.

## A simple way to search for an IP

Once the data has been loaded into MySQL (which will be explained in depth later), there will be a have a table with a range (a lower and upper bound), and some metadata about that range. For example, one row from the GeoIP data (without the redundant columns) looks like:

| ip_from  | ip_to    | country_code |
|----------|----------|--------------|
| 50331648 | 68257567 | US           |

The natural thing that would come to mind (and in fact the solution [offered by MaxMind themselves](#)<sup>2</sup>) is BETWEEN. A

simple query to search for the IP 4.2.2.1 would be:

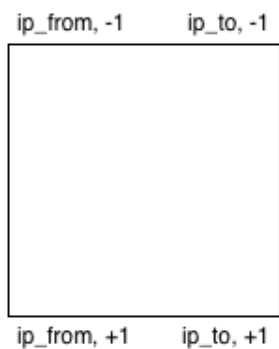
```
SELECT country_code
FROM ip_country
WHERE INET_ATON("4.2.2.1") BETWEEN ip_from AND ip_to
```

Unfortunately, while simple and natural, this construct is extremely inefficient, and can't effectively use indexes (although it can use them, it isn't efficient). The reason for this is that it's an open-ended range, and it is impossible to close the range by adding anything to the query. In fact I haven't been able to meaningfully improve on the performance at all.

## much better solution

While it probably isn't the first thing that would come to mind, [MySQL's GIS support](#) is actually perfect for this task. Geo-referencing an IP address to a country boils down to "find which range or ranges this item belongs to", and this can be done quite efficiently using [spatial R-tree indexes](#) in MySQL's GIS implementation.

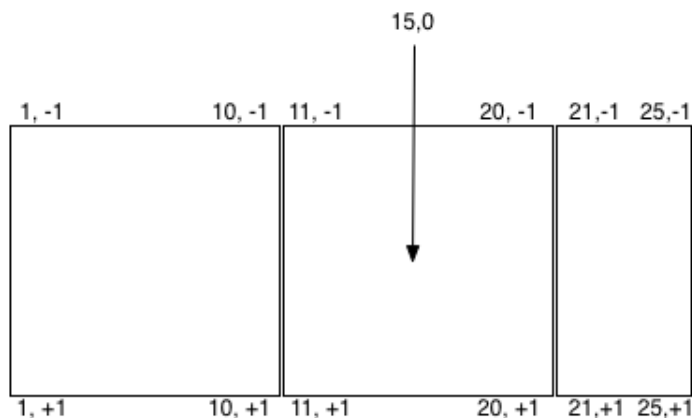
The way this works is that each IP range of (*ip\_from*, *ip\_to*) is represented as a rectangular [polygon](#) from (*ip\_from*, -1) to (*ip\_to*, +1) as illustrated here:



In SQL/GIS terms, each IP range is represented by a 5-point rectangular `POLYGON` like this one, representing the IP range of 3.0.0.0 – 4.17.135.31:

```
POLYGON ((
  50331648 -1,
  68257567 -1,
  68257567 1,
  50331648 1,
  50331648 -1
))
```

The search IP address can be represented as a [point](#) of (*ip*, 0), and that point will have a relationship with at least one of the polygons (provided it's a valid IP and part of the GeoIP database) as illustrated here:



It is then possible to search these polygons for a specific point representing an IP address using [the GIS spatial relationship function](#) `MBRCONTAINS` and `POINT`<sup>3</sup> to search for "which polygon contains this point" like this:

```
SELECT country_code
FROM ip_country
WHERE MBRCONTAINS(ip_poly, POINTFROMWKB(POINT(INET_ATON('4.2.2.1'), 0)))
```

Pretty cool huh? I will show how to load the data and get started, then take look at how it performs in the real world, and compare the raw numbers between the two methods.

## Loading the data and preparing for work

First, a table must be created to hold the data. A `POLYGON` field will be used to store the IP range. Technically, at this point the `ip_from` and `ip_to` fields are unnecessary, but given the complexity of extracting the IPs from the `POLYGON` field using MySQL functions, they will be kept anyway. This schema can be used to hold the data<sup>4</sup>:

```
CREATE TABLE ip_country
(
  id            INT UNSIGNED NOT NULL auto_increment,
  ip_poly      POLYGON      NOT NULL,
  ip_from      INT UNSIGNED NOT NULL,
  ip_to        INT UNSIGNED NOT NULL,
  country_code CHAR(2)      NOT NULL,
  PRIMARY KEY (id),
  SPATIAL INDEX (ip_poly)
);
```

After the table has been created, the GeoIP data must be loaded into it from the CSV file, `GeoIPCountryWhois.csv`, downloaded from MaxMind. The `LOAD DATA` command can be used to do this like so:

```
LOAD DATA LOCAL INFILE "GeoIPCountryWhois.csv"
INTO TABLE ip_country
FIELDS
  TERMINATED BY ","
  ENCLOSED BY ""
LINES
  TERMINATED BY "n"
(
  @ip_from_string, @ip_to_string,
  @ip_from, @ip_to,
  @country_code, @country_string
)
SET
  id            := NULL,
  ip_from      := @ip_from,
  ip_to        := @ip_to,
  ip_poly      := GEOMFROMWKB(POLYGON(LINESTRING(
    /* clockwise, 4 points and back to 0 */
    POINT(@ip_from, -1), /* 0, top left */
    POINT(@ip_to,   -1), /* 1, top right */
    POINT(@ip_to,    1), /* 2, bottom right */
    POINT(@ip_from,  1), /* 3, bottom left */
    POINT(@ip_from, -1) /* 0, back to start */
  ))),
  country_code := @country_code
;
```

During the load process, the `ip_from_string`, `ip_to_string`, and `country_string` fields are thrown away, as they are redundant. A few GIS functions are used to build the `POLYGON` for `ip_poly` from the `ip_from` and `ip_to` fields on-the-fly. On my test machine it takes about 5 seconds to load the 96,641 rows in this month's CSV file.

At this point the data is loaded, and everything is ready to go to use the above SQL query to search for IPs. Try a few out to see if they seem to make sense!

## Performance: The test setup

In order to really test things, a bigger load testing framework will be needed, as well as a few machines to generate load. In my tests, the machine being tested, `kamet`, is a [Dell PowerEdge 2950](#) with Dual Dual Core Xeon 5050 @ 3.00Ghz, and 4GB RAM. We have four test clients, `makalu{0-3}`, which are [Apple Mac Mini](#) with 1.66Ghz Intel CPUs and 512MB RAM. The machines are all connected with a [Netgear JGS524NA](#) 24-port GigE switch. For the purposes of this test, the disk configuration is not important. On the software side, the server is running [CentOS](#) 4.5 with kernel 2.6.9-55.0.2.ELsmp. [The Grinder](#) 3.0b32 is used as a load generation tool with a [custom Jython script](#) and [Connector/J](#) 5.1.5 to connect to [MySQL](#) 5.0.45.

There are a few interesting metrics that I tested for:

- The latency and queries per second with a single client repeatedly querying.
- Does the number of queries handled increase as the number of clients increases?

- Is latency and overall performance adversely affected by many clients?

The test consisted of an IP search using the two different methods, and varying the number of clients between 1 and 16 in the following configurations:

| Clients | Machines | Threads |
|---------|----------|---------|
| 1       | 1        | 1       |
| 2       | 1        | 2       |
| 4       | 1        | 4       |
| 8       | 2        | 4       |
| 16      | 4        | 4       |

Each test finds the country code for a random dotted-quad format IP address passed in as a string.

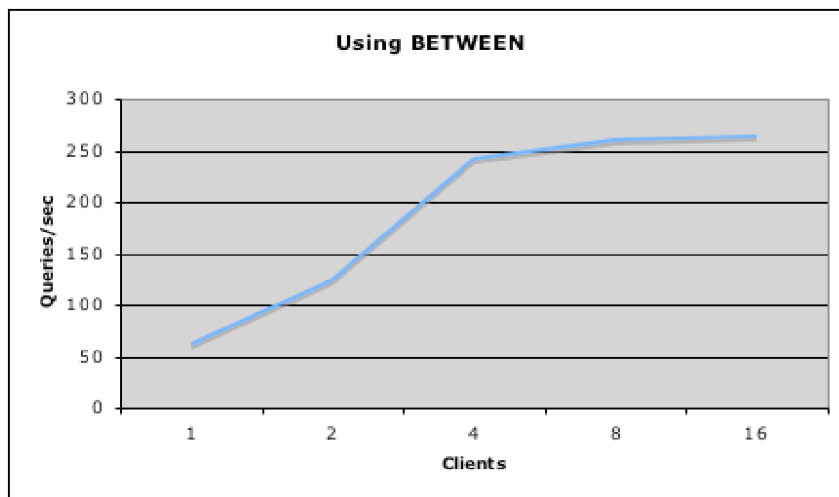
How does it perform? How does it compare?

There are a few metrics for determining the performance of these searches. If you tried the `BETWEEN` version of this query, you may have noticed that, in terms of human time, it doesn't take very long anyway: I pretty consistently got 1 row in set (0.00 sec). But don't let that fool you.

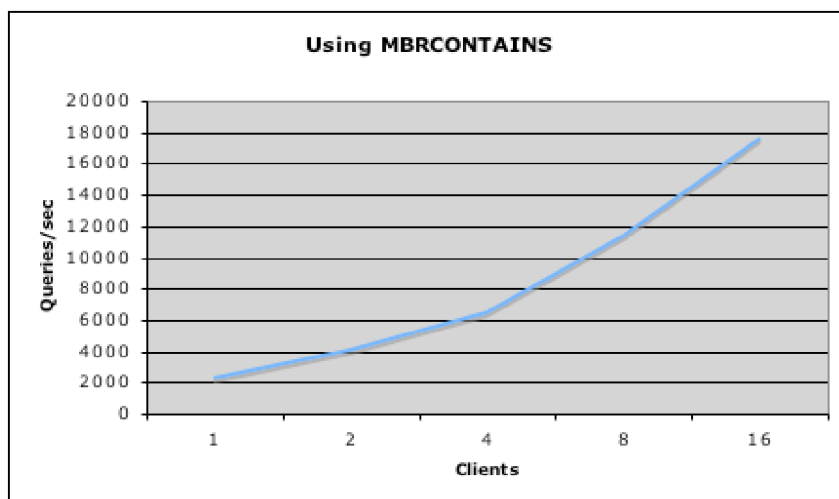
It's clear that `I` wins hands down.

First, a look at raw performance in terms of queries per second.

Using `BETWEEN`, we max out at 264q/s with 16 clients:

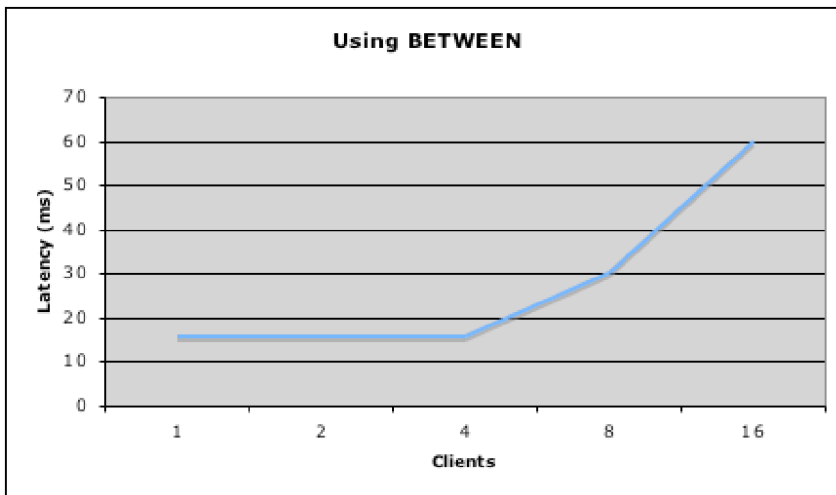


Using `MBRCONTAINS`, we max out at 17600q/s with 16 clients, and it appears that it's the test *clients* that are maxed out, not the server:

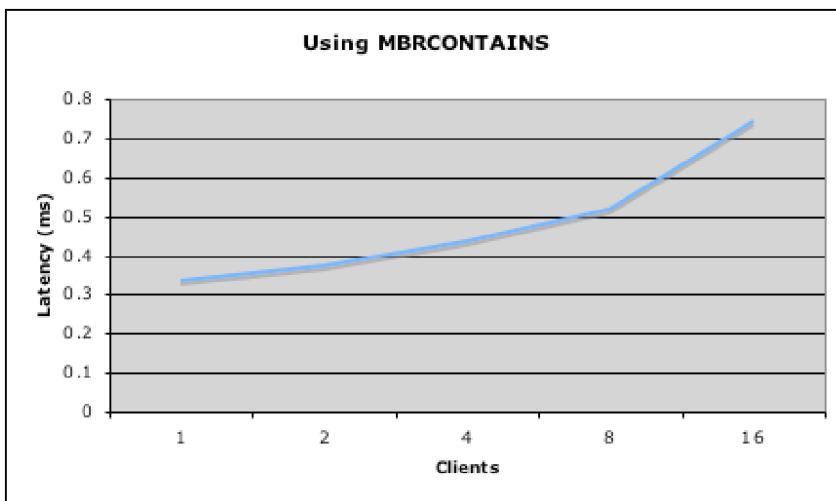


Next, a look at latency of the individual responses.

Using `BETWEEN`, we start out with a single client at 15.5ms per request, which is not very good, but still imperceptible to a human. But with 16 clients, the latency has jumped to 60ms, which is longer than many web shops allocate to completely construct a response. As the number of test clients increases, the latency gets much worse, because the query is so dependent on CPU:



Using `MBRCONTAINS`, we start out with a single client at 0.333ms per request, and even with 16 clients, we are well under 1ms at 0.743ms:



## Conclusion

Definitely consider using MySQL GIS whenever you need to search for a point within a set of ranges. Performance is fantastic, and it's relatively easy to use. Even if you are an `allInnoDB` shop, as most of our customers are (and we would recommend), it may very well be worth it to use MyISAM specifically for this purpose.

## Update 1: Another way to do it, and a look at performance

[Andy Skelton](#) and [Nikolay Bachiyiski](#) left a comment below suggesting another way this could be done:

```
SELECT country_code
FROM ip_country
WHERE ip_to >= INET_ATON('%s')
ORDER BY ip_to ASC
LIMIT 1
```

This version of the query doesn't act *exactly* the same as the other two — if your search IP is not part of any range, it will return the next highest range. You will have to check whether `ip_from` is `<=` your IP within your own code. It may be possible to do this in MySQL directly, but I haven't found a way that doesn't kill the performance.

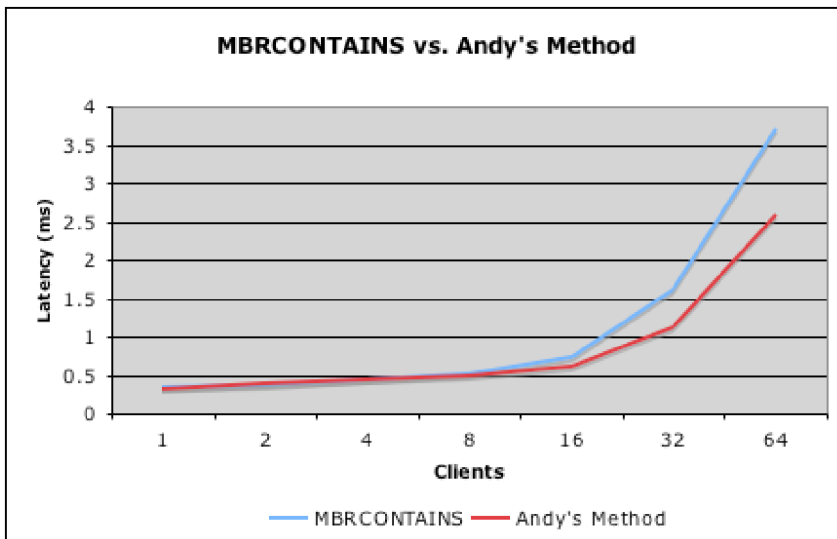
Andy's version actually performs quite well — slightly faster and more scalable than `MBRCONTAINS`. I added two new performance testing configurations to better show the differences between the two:

| Clients | Machines | Threads |
|---------|----------|---------|
|         |          |         |

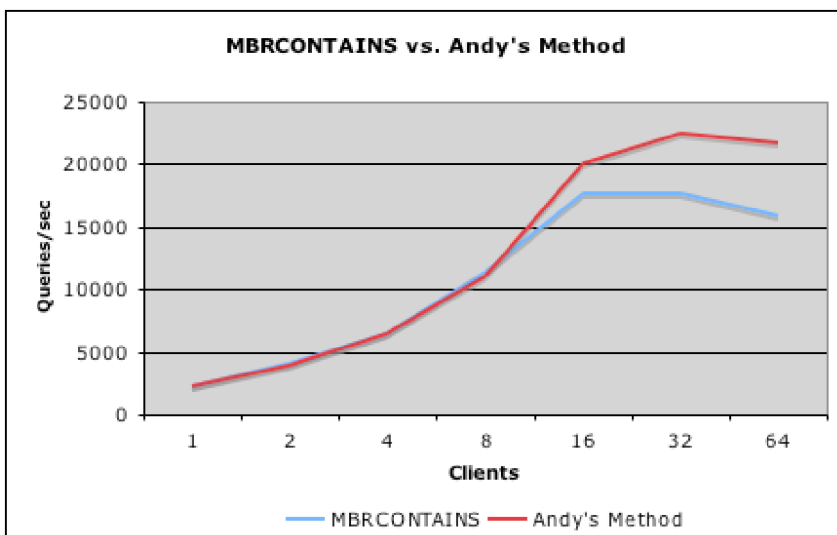
|    |   |    |
|----|---|----|
| 32 | 4 | 8  |
| 64 | 4 | 16 |

Here's a performance comparison of `MBRCONTAINS` vs. Andy's Method:

Latency (ms) — Lower is better:



Queries per second — Higher is better:



Once I get some more time to dig into this, I will look at why exactly `BETWEEN` is so slow. I've also run into an interesting possible bug in MySQL: If you add a `LIMIT 1` to the `BETWEEN` version of the query, performance goes completely to hell. Huh?

Thanks for the feedback, Andy and Nikolay.

## ootnotes

<sup>1</sup> MySQL provides the `INET_ATON()` and `INET_NTOA()` functions for converting back and forth between dotted-quad strings (`CHAR(15)`) and 32-bit integers (`INT UNSIGNED`). You can also use the equivalent functions, if they exist, if your favorite programming language so that you can just feed an integer to MySQL. I haven't tested the (positive) performance implications of doing that.

<sup>2</sup> Although, strangely they offer a different solution specifically for MySQL using `<=` and `>=` operators instead of `BETWEEN`. I don't find that that difference has any effect on MySQL. Maybe it was for a really old version of MySQL that didn't have `BETWEEN`?

<sup>3</sup> Pet peeve: Why does MySQL require you to pass the output of its own `POLYGON`, `LINestring`, `POINT`, etc., functions through `GEOMFROMWKB` in order to use them? It makes life suck that little bit more than necessary.

<sup>4</sup> Note that if you're looking to play around with the `BETWEEN` version of things, you will want to add some indexes on `ip_from` and `ip_to`. I would recommend `INDEX (ip_from, ip_to)` and `INDEX (ip_to, ip_from)` as those two seemed to perform the best that I could find (given its poor efficiency to start with).

Share this:

Gefällt mir 1

This entry was posted on November 24, 2007 at 05:07 and is filed under [GIS and Cartography](#), [MySQL](#), [MySQL Tips, Technology](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.

## 66 Responses to On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS

1. [DbRunas - On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS](#) Says: [November 24, 2007 at 09:32](#) | [Reply](#)

[...] On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS [...]



2. [Mark Robson](#) Says:

[November 24, 2007 at 10:54](#) | [Reply](#)

In fact another way of doing it is to use a conventional index on `ip_from` and then just do:

```
SELECT * FROM ipcountry WHERE ipfrom
```



3. [Baromedia](#) Says:

[November 24, 2007 at 11:30](#) | [Reply](#)

Hi Jeremy,

That's a well researched and written post that has taught me a lot about speed and efficiency regarding GeoIP lookups in MySQL and will inspire me to further create documented and researched material for educational purposes.

I have written an article about how to install the Maxmind GeoIP Country database you can read at [Maxmind GeoIP setup tutorial using phpMyAdmin](#).

Kind Regards

Bart



4. [Jeremy Cole](#) Says:

[November 24, 2007 at 12:53](#) | [Reply](#)

Hi Mark,

I'm not sure I follow. What do you mean? Do keep in mind we're looking for an IP (x) within a range (m – n). Neither m nor n is necessarily x itself.

Regards,

Jeremy



5. [Nikolay Bachiyiski](#) Says:

[November 24, 2007 at 15:26](#) | [Reply](#)

While trying to improve the `BETWEEN` query performance, a colleague of mine — [Andy Skelton](#) — devised a very simple and fast query:

```
SELECT country_code
FROM ip_country
WHERE ip_to >= INET_ATON('4.2.2.1')
ORDER BY ip_to ASC LIMIT 1
```

Of course this trick works only if we have contiguous intervals, which cover all the IPs. Luckily most of the geoip databases conform to this rule.

My simple tests showed that Andy's query is slightly faster than yours, but you may feed it into your benchmarks, so that we can

see if there's any substantial difference.



6. [Andy Skelton](#) Says:

[November 24, 2007 at 15:39](#) | [Reply](#)

I don't know the first thing about GIS but I have found an efficient solution and put it to use "in the wild"—on WordPress.com.

Possibly similar to the solution from Mark, whose comment I assume was cut off due to an unescaped HTML entity, we typically get results from a 4,900,000-row MyISAM table in under 0.5ms using this query:

```
SELECT * FROM ip2location WHERE 123456789 <= ip_to LIMIT 1
```

All we had to do was add an index on ip\_to and verify that there are no overlapping ranges in our table. We can quickly compare against ip\_from in software so there is no need for a double-range query.

Until I discovered this simple query, our ip2location table was useless in production. Now we use it all the time. It doesn't even need its own server; it coexists on a server with dozens of other busy tables.



7. [Jeremy Cole](#) Says:

[November 24, 2007 at 18:19](#) | [Reply](#)

Hi Andy,

This is pretty interesting. I'll be updating the post shortly with your additions and the performance of that version of the query.

Regards,

Jeremy



8. [Tom Allender](#) Says:

[November 25, 2007 at 04:31](#) | [Reply](#)

Andy Skelton said:

```
SELECT * FROM ip2location WHERE 123456789
```

Do you not need an ORDER BY there? Or do you ALTER TABLE ip2location ORDER BY ip\_to?



9. [Mark Robson](#) Says:

[November 25, 2007 at 08:38](#) | [Reply](#)

My post originally said

```
SELECT * FROM ipcountry WHERE ipfrom < 123456789 LIMIT 1
```

But yes, it seems to have been cut off.

My experiments suggested that this was very good. It wouldn't work with overlapping ranges, but nor would any other algorithm. It would be straightforward to check that there were no overlapping ranges.

You would have to check the row returned to make sure the IP was really within it (this check is simple and efficient). If the IP wasn't in the range of the returned row however, you could be sure it was nowhere else either (i.e. not found)

Mark



10. [Mark Robson](#) Says:

[November 25, 2007 at 08:50](#) | [Reply](#)

Further correction:

```
Should have said WHERE ipfrom <= 123456789 ORDER BY ipfrom DESC LIMIT 1
```

The "ORDER BY ipfrom DESC" is essential. You're finding the highest ipfrom which is less than or equal to the one you're checking. This means you will find the range which the IP is in, if any, in a simple query using a conventional index.

Mark




11. [Andy Skelton](#) Says:

[November 25, 2007 at 11:28](#) | [Reply](#)



The index on ip\_to lets us get away without any ORDER BY because the index is naturally scanned in ascending order. That's also why I didn't use ip\_from.

12.  [Jeremy Cole](#) Says:  
[November 25, 2007 at 11:34](#) | [Reply](#)


Hi Andy,

You could actually do the same trick with either ip\_from or ip\_to, as MySQL is capable of scanning in either direction. I wouldn't leave the ORDER BY out; as long as MySQL does what you expect (and it should) the ORDER BY is a no-op. If an optimizer change is made at some point, the entire intent of your query can change without the ORDER BY, and that would be Bad(tm) — you would start getting random rows back (but still 1) instead of the one you want.


At least for me, getting back the wrong row “sometimes” is bad enough, the possibility of the behaviour changing on upgrade and completely breaking doesn't make me comfortable at all. 😊

Regards,

Jeremy

13.  [Andy Skelton](#) Says:  
[November 25, 2007 at 11:43](#) | [Reply](#)

Nikolay's version with ORDER BY is what we use in production because we haven't observed any difference in performance when adding ORDER BY, nor accuracy, nor EXPLAIN SELECT. It is as you said. Better safe.

14.  [Nikolay Bachiyki](#) Says:  
[November 25, 2007 at 14:34](#) | [Reply](#)

Jeremy,


I forgot to mention that the GeLite Country database actually contains lots of gaps. In order to make sure the query gives the correct result I wrote a [simple perl script](#), which takes the csv file and fills in the gaps with dummy rows. Now if the entry doesn't exist you will get “-” instead of the next IP's country.

15. [links for 2007-11-27 : Bob Plankers, The Lone Sysadmin](#) Says:  
[November 26, 2007 at 23:17](#) | [Reply](#)

[...]jcole's weblog » On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS [...]

16. [Geolocalisation : MySQL GIS Ã la rescousse](#) Says:  
[November 28, 2007 at 12:49](#) | [Reply](#)


[...] On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS (0 visite) [...]

17.  [Callum](#) Says:  
[November 30, 2007 at 10:05](#) | [Reply](#)


Forgive me if this is a stupid question, but how does = perform in comparison to BETWEEN?

```
SELECT country_code
FROM ip_country
WHERE ip_to >= INET_ATON('4.2.2.1') AND ip_from
```

I'm not a MySQL expert by any means, but could the performance gain be the LIMIT 1? Perhaps after finding the first result, MySQL can stop scanning the rest of the table. If that is the case, I'm guessing the >= AND

18.  [Joaquim](#) Says:  
[November 30, 2007 at 10:19](#) | [Reply](#)

¿have you tried the binary file version that maxmind provides? It has a optimized format for this especific search, and libraries to open it in varius languages (java, .net, ruby and more.)

19.  [Jeremy Cole](#) Says:  
[November 30, 2007 at 12:57](#) | [Reply](#)

Joaquim,

No, I haven't done any performance testing on MaxMind's solution yet. It would be interesting to do so.

Regards,

Jeremy

20. [links for 2007-12-01 « Donghai Ma](#) Says:

[November 30, 2007 at 21:17](#) | [Reply](#)

[...] On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS (tags: blog geography network gis geoip performance database) [...]



21. [hk](#) Says:

[November 30, 2007 at 22:27](#) | [Reply](#)

I had to insert a slash into your SQL to make it work:

```
LINES
TERMINATED BY "n"
```

Or it errored out with "Cannot get geometry object" etc. Worked fine after that change.

Thanks for the great article.

22. [Sho Fukamachi Online » Blog Archive » Dead simple IP to Country in Rails](#) Says:

[December 1, 2007 at 00:55](#) | [Reply](#)

[...] these instructions to get MaxMind's GeoLite IPCountry table into your [...]



23. [Jeremy Cole](#) Says:

[December 1, 2007 at 01:54](#) | [Reply](#)

Hi hk,

Ack, you're quite right. I guess WordPress escaped/changed it somewhere, as it was "n" in the edit box. Changing it to "\n" corrected it in display. Thanks for the note! Nobody else caught that. 😊

Regards,

Jeremy

24. [Web 2.0 Announcer](#) Says:

[December 1, 2007 at 03:20](#) | [Reply](#)

jcole?s weblog: Jeremy Cole?s take on life.    [Blog Archive](#)    [On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS...](#)

[...][...]



25. [Nathan Schmidt](#) Says:

[December 1, 2007 at 04:03](#) | [Reply](#)

Seconded on Joaquim's hint about using the binary file version of MaxMind's product — it's quite compact and really the right way to do things if you've got a very large number of queries to run — and this is important — at page view time, when latency as opposed to aggregate throughput is a factor. When post-processing for stats you should of course use whatever is closest to your input dataset. We use the binary file api in php at PBwiki for a number of things, and page content can vary based on nation of origin. There are many use cases where an RDBMS shines and this isn't really one of them.



26. [Joshua Paine](#) Says:

[December 1, 2007 at 20:33](#) | [Reply](#)

This should do the in-range check in the query cheaply (but I haven't tested at all):

```
select * from (SELECT * FROM ip2location WHERE 123456789 = ip_from;
```



27. [Jeremy Cole](#) Says:

[December 1, 2007 at 20:51](#) | [Reply](#)

Joshua,

Did your post get truncated? I don't think I get it.

Regards,

Jeremy

28. [links for 2007-12-03 at but i forgot my pen](#) Says:

[December 3, 2007 at 15:21](#) | [Reply](#)

[...] jcolea€™s weblog: Jeremy Colea€™s take on life. A» Blog Archive A» On efficiently geo-referencing IP... (tags: mysql geoip database)[...]

29. [Weird Wonderful Web Links for a cold December | False Positives](#) Says:

[December 4, 2007 at 18:37](#) | [Reply](#)

[...] On efficiently geo-referencing IPs with MaxMind GeoIP and MySQL GIS – [...]

30.  [Fast MySQL Range Queries on MaxMind GeoIP Tables « Andy Skelton](#) Says:

[December 16, 2007 at 03:48](#) | [Reply](#)

[...] MySQL Range Queries on MaxMind GeoIP Tables A few weeks ago I read Jeremy Cole’s post on querying MaxMind GeoIP tables but I didn’t know what all that geometric magic was about so I dropped a comment about how we [...]

31.  [Tim](#) Says:

[January 9, 2008 at 06:01](#) | [Reply](#)

We are using IP2Location instead of Maxmind.

You can find similar SQL syntax from their FAQs.

<http://www.ip2location.com/faqs-ip-country.aspx>

32.  [Jon](#) Says:

[January 10, 2008 at 12:54](#) | [Reply](#)

So I tried the original example out of curiosity but I found many issues with accuracy and such. So I looked at it and changed it around a bit and this is what I came up with.

Import like this:

```
TRUNCATE ip_country;
LOAD DATA LOCAL INFILE "GeoIPCity.csv"
INTO TABLE ip_country
FIELDS
TERMINATED BY ","
ENCLOSED BY "\""
LINES
TERMINATED BY "n"
IGNORE 1 LINES
(
@startIpNum,@endIpNum,@country,@region,
@city,@postalCode,@latitude,@longitude,
@dmaCode,@areaCode
)
SET
id := NULL,
ip_from := INET_ATON(@startIpNum),
ip_to := INET_ATON(@endIpNum),
ip_poly := GEOMFROMWKB(POLYGON(LINESTRING(
/* clockwise, 4 points and back to 0 */
POINT(INET_ATON(@startIpNum), -1), /* 0, top left */
POINT(INET_ATON(@endIpNum), -1), /* 1, top right */
POINT(INET_ATON(@endIpNum), 1), /* 2, bottom right */
POINT(INET_ATON(@startIpNum), 1), /* 3, bottom left */
POINT(INET_ATON(@startIpNum), -1) /* 0, back to start */
))),
country_code := @country,
city := @city,
zipcode := @postalCode,
lat := @latitude,
lon := @longitude,
```

```
areacode := @areaCode
;
```

Then use it like this:

```
SELECT city,country_code FROM ip_country WHERE MBRCONTAINS(ip_poly,
POINTFROMWKB(POINT(INET_ATON('12.199.160.34'), 0)));
```

33.  [Jon](#) Says:

[January 10, 2008 at 12:58](#) | [Reply](#)

I forgot to mention, that while this was rather fast assuming you already have a connection pool or something similar, when you get to an installation like ours (1500+ servers) the time taken to create a MySQL connection, run the query and return the recordset turned out to be longer than the time to use the binary file locally.

I had figured it would be close to on par if not faster. But because of the way Linux caches files on local file systems it's faster to use the binary file for us.

34.  [Stan van de Burgt](#) Says:

[February 6, 2008 at 02:13](#) | [Reply](#)

Another \*very\* fast solution is to use a hash based on the class B network of the IP address your looking for.

First add a hash field 'bucket' and create a key for it:

```
ALTER TABLE ip_country ADD bucket smallint unsigned NOT NULL AFTER ip_to;
ALTER TABLE ip_country ADD KEY bucket(bucket);
```

Then fill the hash (this took 90 seconds on my development machine):

```
UPDATE ip_country SET bucket=ip_from >>16;
```

This sets the new field to the first 16 bits of the IP address of the ip\_from field.

And from now on add the following to the WHERE clause of every query. Here \$ip is the dotted notation of the IP address your are looking for.

```
... AND bucket=INET_ATON($ip)>>16
```

Example:

```
SELECT * FROM ip_country
WHERE bucket=INET_ATON('72.14.207.99')>>16
AND INET_ATON('72.14.207.99') BETWEEN ip_from AND ip_to;
```

Adding the extra condition brought down the time from 3-5 seconds to 0.0 😊

Would be great if you could run the performance tests on this one too, so we can see some stats on this one.

- Stan

PS: The geo data (lat / long) in these databases is \*really\* bad.

35.  [Andrew Droffner](#) Says:

[February 6, 2008 at 15:40](#) | [Reply](#)

Converting IP Addresses to Integer Database Fields: Endian Match

---

There is a SQL/GIS POLYGON example, representing the IP range of [3.0.0.0; 4.17.135.31]. The unsigned integer versions are 50331648 & 68257567 respectively on an Intel architecture. This is confusing, since the "network" architecture is the opposite.

In order to store IP addresses in INTEGER database fields on Intel (x86), you \*must\* get the endianness right. See inet\_aton(), ntohl(), etc.

36.  [Andrew Droffner](#) Says:

[February 7, 2008 at 12:38](#) | [Reply](#)

Why does this use a POLYGON (rectangle) rather than a LINESTRING? An IP Address range is a line-segment.

I'd expect a 1D point on a line-segment to be an easier calculation than an MBR around a 2D point. Is the MBR/R-Tree solution really any faster than a GIS LINESTRING?

37.  *Andrew McLetchie* Says:  
[February 11, 2008 at 16:06](#) | [Reply](#)


Nothing to add, just want to say how much I appreciate this thread!! I am analyzing large volumes of data on people \*trying\* to access our site, but getting turned away b/c no access. We want to profile them geographically, and my big, fat Sun MySQL server was dragging.

After I converted into polygonal data, I was able to process, using only a dual-core MacBook Pro, almost 6 million IP addresses to identify country of origin in 5 minutes!!

Thanks y'all!

38.  *Dave K* Says:  
[February 26, 2008 at 14:15](#) | [Reply](#)

This is a very helpful thread. We are using a database from IP2Location (www.ip2location.com). Some of the approaches discussed in this thread assume that the IP ranges in the database are not overlapping. Has anyone confirmed that this is actually the case (either for IP2Location or any other IP database)? Thanks.

39.  *Dave K* Says:  
[February 28, 2008 at 11:07](#) | [Reply](#)

Well, I have tried some of the suggestions in this tread but the results were not what I expected. Here are my results:

QUERY 1

This was my initial query, which I am trying to optimize.

```
mysql> select SQL_NO_CACHE * from IP_TO_LOCATION where 1234567890 between IP_FROM and IP_TO LIMIT 1;
1 row in set (1.62 sec)
```

---

QUERY 2

This is my first attempt at optimizing the query. However, this did not work so well... see the results below. Can anyone explain why this is the case? Based on this thread, I would not have expected the ORDER BY clause to be a performance hit.

```
mysql> select SQL_NO_CACHE * from IP_TO_LOCATION where IP_TO >= 1234567890 order by IP_TO asc LIMIT 1;
1 row in set (3.84 sec)
```

---

QUERY 3

Removing the ORDER BY improved performance. Now the performance is slightly better then the initial query...but still not as fast as I was expecting. Again, can anyone help explain what is happening here and why I a not seeing they type of performance suggested by the comments on this thread?

```
mysql> select SQL_NO_CACHE * from IP_TO_LOCATION where IP_TO >= 1234567890 LIMIT 1;
1 row in set (1.52 sec)
```

Note: In the example queries above, the 1234567890 is a placeholder for the actual IP address used in these queries.

40.  *Kim* Says:  
[March 11, 2008 at 02:51](#) | [Reply](#)

Useful thread indeed. I got lots of valued information from it. Similar to Andrew McLetchie, our company wanted to profile people visiting our website and I was the guy to make a solution for it. After some research I found this blog and it gave me a few ideas.

I ended up using the free data from MaxMind (they update it monthly), but I used all their free products (country, city, region and timezone) instead of just their Country or City data. When doing that there were of course duplicates, but other then that no major problems.

I combined it all into a single database giving me access to not only look-up IPs but also to reverse look-up based on one or more of the information stored.

Data is found usually within 0.1sec.

Thus we can find users who are in a certain timezone, country, city or region. Pretty neat.

When MaxMind updates their data I only have to download the new csv files, load up my installation script and about 15-30mins

later (when run on my laptop) the database is updated.

I made it so easy to install, that its ready to be published for any to use (sorry, not my call). Written in PHP5 to MySQL5.



41. [ck](#) Says:

[March 17, 2008 at 10:57](#) | [Reply](#)

I've actually used this technique for a couple years now after realizing not only does it speed up the search but reduces the db table to just over 1mb which can be cached far better (and dropping the unneeded columns)

However I discovered it's a little more accurate and gives better "missing" results if you do it backwards using the ENDING column and descend – searching backwards essentially. MySQL does it just as fast, and if not found, the next lower result is better.

I'd had to manually patch the maxmind ranges about two dozen times now. The free db has several holes and inaccuracies, especially with ISPs like AOL. It also lists EU for several spots that should be more country specific. We should group together to share the patches for holes.



42. [Yannick](#) Says:

[April 15, 2008 at 09:21](#) | [Reply](#)

@ Dave K.

Query 2: You need an index(key) on field IP\_TO



43. [Tod Landis](#) Says:

[April 23, 2008 at 14:08](#) | [Reply](#)

Thank you for the great thread. We have a GPLed tool you can use to make plots of GeoIP data called Entrance. We were at the MySQL Conference and I was wowed by the world map Jonathan Schwartz used to show MySQL and Solaris downloads in his keynote. So I came home and added something similar to Entrance.

Its based on world map images by David Pape (which probably what the Sun guys used). Once you have lat, long calculated you do this:

```
PLOT EarthChart
x, very small filled yellow circle
WITH
gray gridlines
SELECT lon, lat
from ACCESS_IPS;
```

The details are on my page: <http://todlandis.com/> and the Entrance downloads are on <http://dbentrance.com/> To get EarthCharts you'll need version 1.2.70 or greater, and either the GPL or IDE version.

To get a black background do this:

```
PLOT EarthChart
x, very small filled yellow circle
WITH
gray gridlines
no bitmap
background black
SELECT lon, lat
FROM ACCESS_IPS;
```

... then you can flip between them with Window | Go back... Window | Go forward.

The black background is pretty handy to have.



44. [Michal Szewczyk](#) Says:

[May 6, 2008 at 03:04](#) | [Reply](#)

Previou post was truncated, seemc to be the < problem...

```
SELECT SQL_NO_CACHE country_code FROM ip_country WHERE INET_ATON("4.2.2.1")>=ip_from AND INET_ATON("4.2.2.2")
<=ip_to;
```

with primary key on (ip\_from, ip\_to).

It gives the same result (queryTime = 0.0003 s) for me as Andy's method and it wont't give you incorrect results when ip not in range.

It's also much faster than BETWEEN condition.

45.  [Paul Hirsch](#) Says:  
[August 18, 2008 at 12:24](#) | [Reply](#)

Curious to know if anyone has tried these out with myisampack'ed tables. while the country database isn't too big, I wonder what affect this has with the city database – quite a bit larger and the overlapping numbers again.

On another note (slightly off-topic): have any of you guys tried out the apache mod\_geoip API way of doing things? Just installed it (pretty painless) and it works pretty darn quickly. Haven't done any serious load testing with it, but with that API its pretty easy and PHP gives you variables through \$\_SERVER you can use (heck just about any language you use with Apache)... Food for thought to save a little pain and database wear and tear. No 30 min. data build, etc, etc, etc. Once installed (took me about 5 minutes, maybe less) you just have to download once a month their compressed data file and (perhaps) restart apache. There may be a better way to update it than that, and it surely can be automated. Food for thought....

[http://www.maxmind.com/app/mod\\_geoip](http://www.maxmind.com/app/mod_geoip)


46.  [Rolf](#) Says:  
[October 1, 2008 at 10:34](#) | [Reply](#)

It gives the same result (queryTime = 0.0003 s) for me as Andy's method and it won't give you incorrect results when ip not in range.

Per query it may be faster but when doing an updtde on a table with 977k rows, this took 25seconds. Using BETWEEN ran for over 12 hours before I stopped it, I let the primary key method run for several minutes. Spatial Indexing FTW

47. [GeoTarget database setup](#) Says:  
[October 16, 2008 at 03:38](#) | [Reply](#)

[...] format and querying is not very efficient at all, and there are two excellent posts (here and here) which I intend to one day impliment myself, but for now the above method works fine for low to [...]

48.  [Perry M](#) Says:  
[December 8, 2008 at 19:08](#) | [Reply](#)

I've tried to expand on Andy's method above with regards to having to check the ip\_from at the software level.

I haven't exactly benchmarked this yet, but it seems as though it would not hurt performance. Basically I needed a reliable way to do the test for ip\_from <= result\_row to determine if the IP was in the range that was returned. With PHP being loosely typed and lack of support for unsigned int's I tried to return a Boolean value from MySQL to check for instead of doing a mildly more complex comparison at the software level.

[CODE]

```
SELECT country_code, ip_from, IF(INET_ATON(%s)>=ip_from,1,0) AS bool_inrange
FROM ip_country
WHERE ip_to >= INET_ATON('â€™%sâ€™')
ORDER BY ip_to ASC
LIMIT 1;
[/CODE]
```

This allows me to simply check if (\$result\_array['bool\_inrange']) { â€! } instead of having to worry about the signed/unsigned oddities; additionally, it ensures all LONG numbers are in correct endianness by using MySQL's INET\_ATON.

In short, you still need to check at the software level against the 'bool\_inrange', but this should make it easier for some languages and require less code to do those checks at the software/app level.

[This is a correction post due to the blog assuming a < and > combination in a post is an HTML element, heh.]

49.  [Chris](#) Says:  
[August 10, 2009 at 06:27](#) | [Reply](#)

Hi,

I need to work out the country name for 3 million IPs stored in a table called ips. I have already converted these to their integer forms. When I try to run the following query:

```
SELECT * FROM IP2Country, ips
WHERE ip >= ip_start AND ip <= ip_end
ORDER BY ip_start ASC
```

LIMIT 1

It just says it is executing for ages and never finishes. Does anyone have any idea how I am going to manage this? 3million is just the tip of the iceberg as I have 182 million to convert in total.

Cheers

50.  [GameGape.com](#) Says:  
[August 18, 2009 at 09:53](#) | [Reply](#)

Thanks for the amazing MySQL tune article.

Here is my benchmark:

#1: Traditional way:

```
SELECT a.ip, b.code FROM tbl_online AS a
INNER JOIN tbl_ip AS b
ON INET_ATON(a.ip) BETWEEN b.ipfrom AND b.ipto
GROUP BY a.ip ORDER BY a.time DESC
—> 14.6 seconds
```

#2 : Using polygon:

```
SELECT a.ip, b.code FROM tbl_online AS a
INNER JOIN tbl_ip AS b
ON MBRCONTAINS(polygon, POINTFROMWKB(POINT(INET_ATON(a.ip), 0)))
GROUP BY a.ip ORDER BY a.time DESC
—> 0.0028 second
```

Someone said: “when you upgrade the hardware, the speed will be multiply by 10, but when you OPTIMIZE your code, the speed will be multiply by 1,000”


That’s exactly TRUE.

Thanks again for the tip !!!

51.  [Cezar VANT](#) Says:  
[December 5, 2009 at 00:22](#) | [Reply](#)

Thanks a lot Jeremy for this thread. I’m using MaxMind GeoIP stucture to find visitors location in my project and your spatial method is amazing.


Cezar

52.  [Sam](#) Says:  
[December 28, 2009 at 17:05](#) | [Reply](#)

How about this:


```
SELECT if(ip_from = INET_ATON('%s')
ORDER BY ip_to ASC
LIMIT 1
```

select and check in one query...


53.  [Sam](#) Says:  
[December 28, 2009 at 17:09](#) | [Reply](#)

How about this:

```
SELECT if(ip_from = INET_ATON('%s')
ORDER BY ip_to ASC
LIMIT 1
```

54.  [Sam](#) Says:  
[December 28, 2009 at 17:10](#) | [Reply](#)

```
SELECT if(ip_from = INET_ATON('%s')
ORDER BY ip_to ASC
LIMIT 1
```

55.  [Ilja](#) Says:  
[April 14, 2010 at 02:04](#) | [Reply](#)



This is the way i query the ip-table:

```
SET @qip = INET_ATON("213.128.135.37");
SELECT @qip as ip, a.*, g.* FROM (
( SELECT i1.start, i1.end, i1.loc
FROM geo_ip i1
WHERE i1.start >= @qip
ORDER BY i1.start ASC LIMIT 1)
UNION ALL
( SELECT i2.start, i2.end, i2.loc
FROM geo_ip i2
WHERE i2.start = a.start AND @qip <= a.end;
```

56.  *Ijja* Says:

[April 14, 2010 at 02:06](#) | [Reply](#)

[CODE]

```
SET @qip = INET_ATON("213.128.135.37");
SELECT @qip as ip, a.*, g.* FROM (
( SELECT i1.start, i1.end, i1.loc
FROM geo_ip i1
WHERE i1.start >= @qip
ORDER BY i1.start ASC LIMIT 1)
UNION ALL
( SELECT i2.start, i2.end, i2.loc
FROM geo_ip i2
WHERE i2.start = a.start AND @qip <= a.end;
```

[/CODE]

57.  *Ijja* Says:

[April 14, 2010 at 02:07](#) | [Reply](#)

please ignore my posts, since they are not displayed correctly

58.  *joao* Says:

[September 21, 2010 at 13:31](#) | [Reply](#)

Try the following and you will be amazed as I was.

Add a primary index to (ip\_to, country), assuming your table has 3 fields (ip\_from, ip\_to, country), and execute the following query:

```
SELECT country FROM ip2country WHERE 123456789 BETWEEN `ip_from` AND `ip_to` LIMIT 1;
```

Note that the index has to be on ip\_to and not on ip\_from... and by adding country to the same index mysql will use the "Using index" optimization.

The other trick is to add the LIMIT 1 together with BETWEEN!

59.  *joao* Says:

[September 21, 2010 at 13:36](#) | [Reply](#)

My bad, in order to use the "Using index" optimization the index has to be on all 3 fields (ip\_to, ip\_from, country)... or just (ip\_to) if you don't care about the optimization or are using MEMORY tables which don't use that optimization anyway. Btw, in case you are using MEMORY tables make sure the index is of type "BTREE" and not "HASH" as that's the default for memory tables.

60.  *joao* Says:

[September 22, 2010 at 06:00](#) | [Reply](#)

I just posted a more detailed explanation on my blog.

61.  *kidbrando* Says:

[January 26, 2011 at 18:24](#) | [Reply](#)

I have to say that this blog has helped me a lot. The solution posted wasnt the exact solution I needed, but guided me in the right direction!

HERE IS MY EXAMPLE FOR SOMEONE WHO HAS A TABLE OF IPs and wants to join against GeoIP data for the location id from GeoLiteCity-Blocks.csv.

Step 1: Create Table For Loading The Raw File

```
drop table ip_country;
```

```
CREATE TABLE ip_country
(
id INT UNSIGNED NOT NULL auto_increment,
ip_poly POLYGON NOT NULL,
ip_from INT UNSIGNED NOT NULL,
ip_to INT UNSIGNED NOT NULL,
locId bigint NOT NULL,
PRIMARY KEY (id),
SPATIAL INDEX (ip_poly)
);
--
```

\*\*\* Note I am loading from a remote location Hence No LOAD DATA LOCAL \*\*\*\*\*

```
LOAD DATA INFILE '/GeoLiteCity-Blocks.csv'
INTO TABLE ip_country
FIELDS
TERMINATED BY ','
ENCLOSED BY '"'
LINES
TERMINATED BY "\n"
IGNORE 2 LINES
(
@ip_from, @ip_to,
@locId
)
SET
id := NULL,
ip_from := @ip_from,
ip_to := @ip_to,
ip_poly := GEOMFROMWKB(POLYGON(LINESTRING(
/* clockwise, 4 points and back to 0 */
POINT(@ip_from, -1), /* 0, top left */
POINT(@ip_to, -1), /* 1, top right */
POINT(@ip_to, 1), /* 2, bottom right */
POINT(@ip_from, 1), /* 3, bottom left */
POINT(@ip_from, -1) /* 0, back to start */
))),
locId := @locId
;
```

Step 2: Query your Table using ipcountry

Table Example:

```
drop table if exists geo_test;
create table geo_test (ipvalue varchar(255),locId bigint );
insert geo_test select '4.22.141.200',NULL ;
insert geo_test select '4.42.246.68',NULL ;
insert geo_test select '4.59.148.141',NULL ;
create index idx_inet on geo_test(ipvalue);

update geo_test a , ip_country b
set a.locID = b.locID
WHERE MBRCONTAINS(ip_poly, POINTFROMWKB(POINT(INET_ATON(ipvalue), 0)));


/**BENCHMARKING ****/
```

Updating a table of 1K records takes .110 ms..

Updating a table of 100K records takes 3.37 ms..

62. [How to build an efficient GeoIP SQL table | dopefish.de](http://www.dopefish.de) Says:  
[August 22, 2011 at 12:20](http://www.dopefish.de) | [Reply](#)

[...] while back I found a very interesting posting at <http://www.jcole.us> that described how to use Spatial Indexes together with MySQL's GIS to speed up the [...]

63.  *Jonathan* Says:  
[October 22, 2011 at 01:39](#) | [Reply](#)

anyone else looking at the GeoLiteCity data from MaxMind – also now free but a different format?

64.  *Vovan* Says:  
[November 23, 2011 at 07:47](#) | [Reply](#)

Hello, Jeremy.

I'm trying to JOIN table with logged ips and geoip table with country codes. I created polygons for ip\_country (same, like in your's tutorial) and points for logged ips:

```
CREATE TABLE `ips` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `ip` int(10) unsigned NOT NULL,
  `ip_point` point NOT NULL,
  PRIMARY KEY (`id`)
)
```

If I do:

```
SELECT *
FROM ips
LEFT JOIN ip_country ON MBRCONTAINS(ip_country.ip_poly, ips.ip_point)
WHERE ips.id = 2;
```

Everything works good – index ip\_poly in tbl ip\_country is used and everything is fast. But when I want to join several rows from logged\_ips everything breaks down – index is not used:

```
SELECT *
FROM ips
LEFT JOIN ip_country ON MBRCONTAINS(ip_country.ip_poly, ips.ip_point)
WHERE ips.ip IN (1,2,3,4,5);
```

I tried to solve this problem for many days, but still have no idea, why it is not working properly. Please, give me any clue how to fix it.

65.  *Christian Fazzini* Says:  
[November 25, 2011 at 22:35](#) | [Reply](#)

Great article!

I am trying the following query against my table and its returning no results:

```
SELECT country_name
FROM ip_to_countries
WHERE ip_address_to >= INET_ATON('4.2.2.1')
ORDER BY ip_address_to ASC LIMIT 1
```

Nikolay Bachyiski mentioned that the GeoLite Country database actually contains lots of gaps.

I am using the GeoLite Country csv version rather than the \$50 paid. Does this problem go away automatically if I use the paid version (\$50)?

Or would I still need to find a way to fill in the “gaps”? If so, I am on Ruby On Rails, is there a way to do this with Ruby?

66. [MySQL Performance Tips | i++](#) Says:  
[December 19, 2011 at 20:48](#) | [Reply](#)

[...] Abuse the system for optimization you're using with system dependant features like RTREE's for optimized range queries [...]

hat do you thin

Enter your comment here...

Fill in your details below or click an icon to log in:



Email (required) (Address never made public)

Name (required)

Website

Notify me of follow-up comments via email.

**Post Comment**

Notify me of new posts via email.

Theme: [Kubrick](#). [Blog at WordPress.com](#). [Fonts on this blog](#).  
[Entries \(RSS\)](#) and [Comments \(RSS\)](#).

