Postgres OnLine Journal: January / February 2010

An in-depth Exploration of the PostgreSQL Open Source Database

Table Of Contents

A Product of Paragon Corporation
http://www.paragoncorporation.com/
http://www.postgresonline.com/

Looking forward to PostgreSQL 8.5

Ah a new year, a new PostgreSQL release in the works. Beware -- this post is a bit sappy as we are going to highlight those that have made our lives and lives of many a little easier.

These are people we consider the most important because they provide the first impression that newcomers get when first starting off with PostgreSQL. The newcomer that quickly walks out the door unimpressed, is the easy sale you've lost. Make your pitch short and sweet.

As always Hubert does a really good job of taste testing the new treats in the oven and detailing how cool they are. I highly suggest his posts if people have not read them already or are looking at PostgreSQL for the first time. You can catch his *Waiting for PostgreSQL 8.5 series* which is in progress. Surely gives us a list of things to test drive.

Then there are those that document, the volumes of PostgreSQL documentation which are just great, up to date and rich with content. Probably too many of these people to call out, and sadly we don't know them by name.

Of course its not just enough to announce releases, document them and talk about them, you must make it really easy for people to try them out. If people have to compile stuff, especially windows users, forget about it. You won't hear complaints, you won't hear whispers, you'll hear dust blowing. The biggest audience you have is the one you just lost because you didn't make it easy for them to try your stuff. The apple hit me on the head one day when a very dear friend said to me and here is a slight paraphrase. *You don't actually expect me to compile this myself do you? How much time do you think I have? It is not about you, it is about me.*. This was especially surprising coming from a guy I always thought of as selfless. This I realized is the biggest problem with many open source projects, that they are lost in the flawed mentality that its about scratching their own itch and the rest will come. It is not. Always concentrating on your own itch and scratching it is a sure way of guaranteeing that no one will scratch your itch for you. Think of it like a pool game. Do you target the aim at the ball you are trying to hit, or balls near by that will knock down the others. So in short don't be a complete wuss that people can walk all over, but look past your nose and choose your balls wisely; make sure all your balls are not focused on software development.

So in the spirit of scratching someone else's itch, We would also like to thank greatly the efforts of:

- Devrim for his work with the PostgreSQL Yum repository which has personally made our lives a lot easier.
- David Page, EnterpiseDb, and the PgAdmin crew for making an easy to use one-click installer (MacOSX,Linux, Windows) and an easy to to use Database IDE that is very inviting to use for inexperienced users. It is scary to imagine how many people would not have used PostgreSQL without the existence of these tools.
- Bruce Momjian -- for making those very useful webcasts, and presentations and being one of the great undying hearts of PostgreSQL.
- Tom Lane -- not just for scrutinizing patches, but answering all those endless questions on the PostgreSQL user newsgroups, and still managing to find time to shower us in the PostGIS developer's group with his very welcomed surprise appearances. I think the jury is still out as to whether Tom is exploiting some feature of the wave theory to accomplish what he does.
- To Josh Drake, Josh Berkus, Selena Deckelmann for organizing all those PostgreSQL conferences and making available all the videos and podcasts for us to enjoy afterward.
- To all those countless testers, documenters, other packagers, and bloggers that make PostgreSQL a *wow its fast, wow its cool, and I don't see any bugs* experience.

We haven't started to play with PostgreSQL 8.5 yet but plan to in the next month or two. Surprisingly we are more interested in the smaller things than the big Hot-Standby feature everyone is talking about. The things we are most excited about trying out so far

- Grant All this is just a great usability feature. Its one of the reasons why new users just run everything under postgres account or whine about how MySQL security is so much easier, cause its too damn much effort to secure things correctly in PostgreSQL. It also comes close and perhaps better than feature of SQL Server that has built-in data reader/data writer roles you can assign to users/groups to give them the rights and all rights in future to created tables.
- Exclusion constraints. I'm really hoping this will help serve a somehwat nagging problem that we have that you can't use constraint exclusion inheritance benefits to partition data by geometric space. Well we'll see. It is a dream.
- Drop IF EXISTS. Its about freaking time
- Named function arguments, this is one of the reasons I love VB as a language, because VB has named arguments and why I find C# and all those C like knock-offs frustrating to work with at times because they lack these nuggets of syntactic sugar.

Back to Table Of Contents Looking forward to PostgreSQL 8.5 Reader Comments

Forcing the planner's hand with set enable_seqscan off WTF *Intermediate*

UPDATE: Thanks all for the suggestions. For now we ended up increasing the seq_page_cost from 1 to 2 in the database. That has gotten us back to our old much much faster speeds without change in code and seems to have improved the speeds of other queries as well, without reducing speed of any.

```
ALTER DATABASE mydb SET seq_page_cost=2;
```

As Jeff suggested, we'll try to come up with a standalone example that exhibits the behavior. The below example was more to demonstrate the construct. Table names and fields were changed to protect the innocent so that is why we didn't bother showing explain plans. The behavior also seems to do with the distribution of data and gets worse when stats are updated (via vacuum analyze). Didn't see this in PostgreSQL 8.3 and this was a system recently upgraded from 8.3 to 8.4

---ORIGINAL ARTICLE HERE --

This is a very odd thing and I think has happened to us perhaps once before. Its a bit puzzling, and we aren't particularly happy with our work around because its something that looks to a casual observer as a bit bizarre. The hack is setting the enable_seqscan setting off for a particular query to force the planner to use indexes available to it.

What is particularly troubling about this problem, is that it wasn't always this way. This is a piece of query code we've had in an application for a while, and its worked shall I say *really fast*. Response times in 300 ms - 1 sec, for what is not a trivial query against a not so trivially sized hierarchy of tables. Anyrate, one day -- this query that we were very happy with, suddenly started hanging taking 5 minutes to run. Sure data had been added and so forth, but that didn't completely explain this sudden change of behavior. The plan it had taken had changed drastically. It just suddenly decided to stop using a critical index it had always used. Well it was still using it but just on the root table, not the children. Though querying a child directly proved that it still refused to use it, so it didn't seem to be the hierarchy at fault here.

We checked our dev server which has the same PostgreSQL 8.4.2 install as production (both running on Windows though production is a Windows 2008 and dev Windows 2003), and data that was probably about two weeks older on dev. Dev was working perfectly and production was not.

Then we vacuum analyzed some critical tables on Dev and Dev started to behave in the same slow stupid way with the same stupid plan. Vacuum analyzing is supposed to help the situation? After analyzing the plans, we realized what had changed before vacuum analyze and after was that, it stopped using this critical index on the child nodes.

Our query reduced down to what we think is the problem area. In a very watered down form it looked something like this:

```
SELECT item_name, geom
   FROM items
   WHERE items.feature_type
      IN(SELECT f.feature_type
         FROM lufeature_types As f
         WHERE f.grouping IN('G','K') );
```

That used to use the btree index we had on feature_type for all the tables but then suddenly stopped. Changing it to an explicit IN list where the IN list is what the subquery would return made it use the btree index again, but that was a very undesirable thing to do because we like our code as short as possible, and such a stupid solution would require writing a query -- parsing it into an array and then feeding it to a second query. And no even making the subquery a CTE did not help.

```
SELECT item_name, geom
   FROM items
   WHERE items.feature_type IN('A', 'B','C');
```

We finally ended up changing the code to do this which allowed us to keep the SQL statement as we liked it and still use the index. But the idea of giving the planner these kinds of hints especially when it never needed it before, I've always found a bit unpleasant.

```
set enable_seqscan = off;
SELECT item_name, geom
   FROM items
WHERE items.feature_type
   IN(SELECT f.feature_type
      FROM lufeature_types As f
      WHERE f.grouping IN('G','K')
      );
```

For those unfamiliar with the enable_seqscan setting, it doesn't disable sequential scanning completely to set it to off since well in some cases there is no alternative. It does make the planner use sequential scanning as a last resort when there is no index it can possibly use. Sequential scans are not always a bad thing even if you do have indexes since an index scan is not a completely free strategy.

Back to Table Of Contents Forcing the planner's hand with set enable_seqscan off WTF Reader Comments

Regular Expressions in PostgreSQL ***Beginner***

Every programmer should embrace and use regular expressions (INCLUDING Database programmers). There are many places where regular expressions can be used to reduce a 20 line piece of code into a 1 liner. Why write 20 lines of code when you can write 1.

Regular expressions are a domain language just like SQL. Just like SQL they are embedded in many places. You have them in your program editor. You see it in sed, grep, perl, PHP, Python, VB.NET, C#, in ASP.NET validators and javascript for checking correctness of input. You have them in PostgreSQL as well where you can use them in SQL statements, domain definitions and check constraints. You can mix regular expressions with SQL. When you mix the two domain languages, you can do enchanting things with a flip of a wrist that would amaze your less informed friends. Embrace the power of domain languages and mix it up. PostgreSQL makes that much easier than any other DBMS we can think of.

For more details on using regular expressions in PostgreSQL, check out the manual pages Pattern Matching in PostgreSQL

The problem with regular expressions is that they are slightly different depending on what language environment you are running them in. Different enough to be frustrating. We'll just focus on their use in PostgreSQL, though these lessons are applicable to other environments.

Common usages

We are going to go backwards a bit. We will start with demonstrations of PostgreSQL SQL statements that find and replace things and list things out with regular expressions. For these exercises we will be using a contrived table called notes, which you can create with the following code.

```
CREATE TABLE notes(note_id serial primary key, description text);
INSERT INTO notes(description)
  VALUES('John''s email address is johnny@johnnydoessql.com.  Priscilla manages the http://www.johnnydoessql.com site.
    She also manages the site http://jilldoessql.com and can be reached at 345.678.9999
    She can be reached at (123) 456-7890 and her email address is prissy@johnnydoessql.com or prissy@jilldoessql.com.');

INSERT INTO notes(description)
  VALUES('I like ` # marks and other stuff that annoys militantdba@johnnydoessql.com.
    Militant if you have issues, give someone who gives a damn a call at (999) 666-6666.');
```

Regular Expressions in PostgreSQL

PostgreSQL has a rich set of functions and operators for working with regular expressions. The ones we commonly use are **~**, **regexp_replace**, and **regexp_matches**.

We use the PostgreSQL **g** flag in our use more often than not. The g flag is the greedy flag that returns, replaces all occurrences of the pattern. If you leave the flag out, only the first occurrence is replaced or returned in the regexp_replace, regexp_matches constructs. The ~ operator is like the LIKE statement, but for regular expressions.

Destroying information

The power of databases is not only do they allow you to store/retrieve information quickly, but they allow you to destroy information just as quickly. Every database programmer should be versed in the art of information destruction.

```
-- remove email addresses if description has email address
UPDATE notes SET description = regexp_replace(description,
    E'[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]{2,4}',
     '---', 'g')
     WHERE description
     ~ E'[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]{2,4}';

-- remove website urls if description has website urls
-- matches things like http://www.something.com or http://something.com

UPDATE notes SET description = regexp_replace(description,
    E'http://[[[:alnum:]]+.]*[[:alnum:]]+[.][[:alnum:]]+',
     E'--', 'g')
     WHERE description
     ~ E'http://[[[:alnum:]]+.]*[[:alnum:]]+[.][[:alnum:]]+';

-- remove phone numbers if description
-- has phone numbers e.g. (123) 456-7890 or 456-7890 or 123.456.7890
UPDATE notes SET description = regexp_replace(description,
    E'[\(]{0,1}[0-9]{3}[\).-]{0,1}[[:space:]]*[0-9]{3}[.-]{0,1}[0-9]{4}',
     '---', 'g')
     WHERE description
     ~ E'[\(]{0,1}[0-9]{3}[\).-]{0,1}[[:space:]]*[0-9]{3}[.-]{0,1}[0-9]{4}';

-- set anything to single space that is not not a \ ( ) & * /,;. > < space or alpha numeric
UPDATE notes set description = regexp_replace(description, E'[^\(\)\&\/,;\*\:.\>\<[:space:]a-zA-Z0-9-]', ' ')
 WHERE description ~ E'[^\(\)\&\/,;\*\:.\<\>[:space:]a-zA-Z0-9-]';

-- replace high byte characters with space
-- this is useful if you have your database in utf8 and you often need to use latin1 encoding
-- and you have a table that shouldn't have high byte characters
-- such as junk you get from scraping websites
-- high byte characters don't convert down to latin1
UPDATE notes SET description = regexp_replace(description,E'[^\x01-\x7E]', ' ', 'g')
WHERE description ~ E'[^\x01-\x7E]';
```

Getting list of matches

These examples use similar to our destroy but show us in a table, a list of stuff that match. Here we use our favorite PostgreSQL **regexp_matches** function.

```
-- list first email address from each note --
SELECT note_id,
  (regexp_matches(description,
   E'[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+'))[1]  As email
FROM notes
WHERE description ~ E'[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+'
ORDER By note_id, email;

-- result
 note_id |             email
---------+-------------------------------
       1 | johnny@johnnydoessql.com
       2 | militantdba@johnnydoessql.com
```

```
--list all email addresses
-- note this uses the PostgreSQL 8.4 unnest construct
```

```
-- to convert the array returned to a table
SELECT note_id,
  unnest(
    regexp_matches(description,
    E'[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+', 'g')
  ) As email
FROM notes
WHERE description ~ E'[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+'
ORDER By note_id, email;


-- returns
 note_id |             email
---------+-------------------------------
       1 | johnny@johnnydoessql.com
       1 | prissy@jilldoessql.com
       1 | prissy@johnnydoessql.com
       2 | militantdba@johnnydoessql.com
```

Parts of a Regular Expresssion

Here we will just cover what we consider the parts you need to know if you don't have the patience or memory to remember more. Regular expressions are much richer than our simplified view of things. There is the great backreferencing feature we won't get into which allows you to reference an expression and use it as part of your replacement expression.

Part	Example
Classes []	Regular expression classes are a set of characters that you can treat as interchangeable. They are formed by enclosing the characters in a bracket. They can also have nested classes. For example [A-Za-z] will match any letter between A-Z and a-z. [A-Za-z[:space:]] will match those plus white spaces in PostgreSQL. If you need to match a regular expression character such as (then you escape it with \. so [A-Za-z\(\)] will match A thru Z a thru z and ( ). Classes can contain other classes and expressions as members.
.	The famous . matches any character. So the infamous .* means one or more of anything.
Quantity {} + *	You denote quantities with {}, +, * + means 1 or more. * means 0 or more and {} to denote allowed quantity ranges. [A-Za-z]{1,5} means you can have between 1 and 5 alpha characters in and expression for it to be a match.
()	This is how you denote a subexpression. A subexpression can be composed of multiple classes etc and can be backreferenced. They define a specific sequence of characters.
[^members here]	This is the NOT operator in regular expressions so for example [^A-Za-z] will match any character that is not in the alphabet.
Special classes	[[:alnum:]] any alphanumeric, [[:space:]] any white space character. There are others, but those are the most commonly used.

Back to Table Of Contents Regular Expressions in PostgreSQL Reader Comments

Making backups of select tables *Beginner*

Every once in a while, especially if you have a fairly large database, you may find the need to do select backups of certain tables. Your criteria might be based on name or how relatively recently data has changed in the table. Below are some of the tricks we use. Some use our favorite hack of scripting command line scripts with SQL.

Backup specifically named tables - no tricks

The simple case is when you know exactly what tables you need backed up and there aren't too many that its easy enough to type them out. Here you just use the -t option for as many tables as you need to backup.

```
pg_dump -h localhost -p 5432 -U postgres -F c -b -v -f "/pgbak/somedb_keytbls.backup" -t
someschema.sometable1 -t someschema.sometable2 somedb
```

Generate script to backup specifically named tables based on table name filter using SQL

We use this approach if say we have multiple schemas with same similarly named critical tables. We can't use the -n dump option because the tables cross schemas, but we know they all have similar names. You can get fairly fancy with this and even use information_schema.columns to get even fancier. The below will generate a pg_dump command to backup any table not in public or pg_catalog that has notes as part of the table name.

```
--- Simple go by name use information_schema.tables
-- but note you can use pg_stat_user_tables instead
SELECT 'pg_dump ' || ' -h localhost -p 5432 -U postgres  -F c -b -v -f "/pgbak/somedb_keytbls.backup" ' ||
  array_to_string(ARRAY(SELECT '-t ' || table_schema || '.' || table_name
        FROM information_schema.tables
        WHERE table_name LIKE '%_notes' AND table_schema NOT IN('pg_catalog','public' )
    ), ' ') || ' somedb';
```

Backup recently changed tables using stats view

Sometimes we want to backup just tables that have recently changed. There doesn't seem to be an absolutely perfect way of doing this, so we use the vacuum stats as the next best thing. This trick only works if you have autovacuum on. It uses the assumption that the vacuum process will try to go around and vacuum tables, where enough data has changed since the last vacuum run. The vacuum run setting you can tweak for each table for at least 8.4 and up. The below example will generate a pg_dump command to backup all tables in **somedb** that have been auto analyzed in the past 24 hours.

```
-- Backup recently updated tables based on auto analyze date
SELECT 'pg_dump ' || ' -h localhost -p 5432 -U postgres  -F c -b -v -f "/pgbak/somedb_keytbls.backup" ' ||
  array_to_string(ARRAY(SELECT '-t ' || schemaname || '.' || relname
        FROM pg_stat_user_tables
        WHERE last_autoanalyze > ( CURRENT_TIMESTAMP - (INTERVAL '1 day') ) )
    , ' ') || ' somedb';
```

Back to Table Of Contents

PostGIS 1.5.0 out and PLR working on Windows 8.3-8.4 installs

PostGIS 1.5.0 is finally out

I'm happy to report that after a long haul, we have finally released PostGIS 1.5.0. Two months late, but there it is, and its a really great release I think; Perhaps the best release ever.

Details on what makes this release so special. The geodetic support.

Summary excerpted from Paul's slightly premature announcement

```
February 4, 2010

The PostGIS development team has, after a long course of reflection
and a detailed self-examination of our various personal failings,
decided to release PostGIS 1.5.0 to the public.

  http://postgis.org/download/postgis-1.5.0.tar.gz

This new version of PostGIS includes a new "geography" type for
managing geodetic (lat/lon) data, performance-enhanced distance
calculations, GML and KML format readers, an improved shape loading
GUI, and other new features as well.

Especial thanks to:

* Dave Skea for algorithms and mathematics necessary to support
spherical geometry
* Nicklas Avén for the new performance enhanced distance calculations
and other distance-related functions
* Sandro Santilli for new buffering features (end caps and style options)
* Olivier Courtin for GML/KML input functions
* Guillaume Lelarge for support for the upcoming PgSQL 9.0
* George Silva for an example implementation of history tables
* Vincent Picavet for Hausdorff distance calculations
* The maintainers of GEOS, Proj4, and LibXML, without whom we would
have less of a spatial database

Love, the PostGIS project steering committee,

Mark Cave-Ayland
Kevin Neufeld
Regina Obe
Paul Ramsey
```

Yes evidentally we still have some personal failings to work out :).

More extensive details at http://www.postgis.org/news/20100204/

PostGIS 1.5.0 Windows binaries and StackBuilder install will be out soon

Leo and I are currently working on the Windows stackbuilder version and hope to have that released sometime next week.

PL/R Now works on PostgreSQL Windows 8.3/8.4 VC++ compiled versions

More great news. Joe Conway has gotten PL/R to compile under VC++ and thus allowing it to work with the standard Windows One-click installers. It is not an option in stack builder yet, but we are hoping there will be one to ease its use. You can download the windows pl/r binaries from the new PL/R Wiki. Once we've gotten caught up in our other activities, you can expect to see on this site a PL/R cheatsheet and examples similar to our PL/Python series.

The final chapters of PostGIS in Action

We are down to writing the last 3 chapters of our book. We are currently working on Chapter 10: PostGIS Add-Ons and ancillary tools and Chapter 11: Using PostGIS in web applications. Chapter 13 on WKT Raster we hope to get to within the next 2 weeks. Joe Conway coming to our rescue is very timely for us for Chapter 10.

Chapter 10 covers PL/R, PL/Python, PgRouting and the new Shapefile Tiger Geocoder reworked by Stephen Frost. We think of Chapter 10 as the *Hidden Gems Chapter* because it covers features that no other database management system (DBMS) we can think of can compete with PostgreSQL on. Our windows audience is fairly large, so we didn't want to leave them out on being able to exercise any of these rich pieces. Chapter 10 is a really exciting chapter. We've had a lot of fun preparing the examples and writing it so far. We hope people will enjoy reading it and playing with the sample data and exercises.

Back to Table Of Contents PostGIS 1.5.0 out and PLR working on Windows 8.3-8.4 installs Reader Comments

Reader Comments

Looking forward to PostgreSQL 8.5

Jeff Davis
"...problem that we have that you can't use constraint exclusion inheritance benefits to partition data by geometric space"

Exclusion Constraints have nothing to do with constraint_exclusion. That may be unfortunate naming.

Why does constraint_exclusion not allow you to partition data by geometric space?

Regina
Jeff,

I didn't think they did. I guess I was thinking more along the idea that it would give me ideas :)

If you were to put a constraint in for geometric data you would have a constraint that sort of bounds the region in the table with something like

&& BOX(....)

Most PostGIS relationship function have the && .. built in to utilize indexes and also it is common practice when pulling data for a map to do a && ... box region -most map servers do that automagically. If the planner could recognize that if a box does not overlap the table constrained box, then there would be no reason to search the table, and that I would suspect be a big performance boost for very large spatial dbs.

But && is not a range constraint (well it sort of is), but its not seen as such by the planner. So although you can in theory do this, it does you no good when doing queries against an inherited table hierarchy because it still scans the tree.

Well I haven't tried it in a while so I should give it a try again before I open my mouth :).

So general work around is that you still partition by space using some btree code like region_name and tack these on to your queries. This is generally what we do.

Sami Kuhmonen
Don't take this as a total rant, but...

I'd like to NOT thank the people who decided the "one-click installer" (that requires lots of clicks, doesn't work on correctly secured Win7/WS2008R2 and is unbearable for administrators to use) is a lot better than the old MSI based thing (which has NONE of the aforementioned problems). It's like telling Debian users to install Pgsql from tar packages and run random scripts to rape the package system to think it's installed.

I'd like also to NOT thank you two for your strange page where you complain about the problems compiling PostGIS etc on Windows. I find it strange that one would complain about a platform not being like Linux with myriads of compiling thingies. Platforms are different, which should be quite obvious to anyone in the SW industry. And I really don't understand why you'd want to compile PostGIS with mingw.

You know, I took the sources of PostGIS, GEOS and PROJ (never ever having seen them before) and within 15 minutes I had a Visual Studio solution compiling the whole thing, the resulting binary is smaller than with mingw and even works slightly faster. So is the problem with Windows or the people using wrong compilation tools? And a hint: Visual C++ compiler is free.

But anyway, I'd like to thank you for PostGIS, since it's a wonderful tool and I really couldn't live without it.

www.dbrunas.com.ar

Regina
Sami,

Did you report these problems. I haven't run into the issues you described running on my windows 7 or windows 2008 boxes even with windows firewall and virus software installes. Could be a virus scanner conflict problem.

Right now for Windows PostGIS really only supported compiled under MingW. This is on our todo list to change , but we have higher priorities at the moment. For the most part people don't need to compile PostGIS under Windows anyway since we do the compiling. Its always better to have a vocal critical user than a silent frustrated one, so we welcome

your rants :).

I would be interested in your PostGIS compile instructions and how the performance is for you though (if you would like to document them on the PostGIS wiki -- we would be very delighted :).

http://trac.osgeo.org/postgis
Which version were you compiling?

Mateusz did do some cleanup to make it compile under VC++
http://mateusz.loskot.net/2009/03/29/building-postgis-using-visual-cpp/
so the ability you can compile at all is probably of great thanks to him
, but sadly he didn't have time to maintain it, and we are still getting our feet wet with maintenance so not quite ready to rock the boat yet. VC++ has never been supported for PostGIS so didn't want to risk it for current go around without extensive testing. We are taking over the task of building the PostGIS installers so we are still feeling our way thru the process. Eventually we would like to have it compiled in VC++. Anyrate any support you can provide is very welcome :).

Mateusz Loskot
Sami, you mean 15 minutes? Tell me first when you did that. I assume it was after March 2009. Previously, PostGIS did not compile using Visual C++ due to use of incompatible features not available in C++ implementation from Visual C++.

Here is the story:

http://mateusz.loskot.net/2009/03/29/building-postgis-using-visual-cpp/

Also, check number of commits to PostGIS.

Mateusz

www.pythian.com

Forcing the planner's hand with set enable_seqscan off WTF

Lance French
We noticed a similar type of problem with 8.4.1 after moving from 8.3.5. Our 'fix' was to increase the default_statistics_target by a factor of 10 (it's set to 500 now, I think). Analyze is a bit slower now ...

Chris
I hope you had an extra line after that query to re-enable seq_scan .. otherwise for the rest of the queries in the script it'd be off (which may or may not be what you want). :)

Regina
Lance,

Great thought, but sadly doing that and reanalyzing the tables, did not seem to do the trick and slowed the analyze down considerably (and I ran out of memory) for the table I have that has large geometries in it. though I think you are right its the stats target that is off in some way.

I ended up just increasing the seq_page_cost in the database to 2

ALTER DATABASE somedb SET seq_page_cost=2;

And that seemed to fix my issue. I suspect that I have some other queries that are suffering from the same fate but not as noticeable as this one, so I'm going to retest them with this new setting and compare my old speeds with the new.

gregj
just use JOIN instead, and it will be fast.
SELECT i.item_name, i.geom
FROM items i JOIN lufeature_types f ON i.feature_type = f.feature_type
WHERE f.grouping IN('G','K') ;

job done....

Regina
Greg,

We usually do. The reason We don't in this case, is that that is just one of many conditions used to filter data in this application, its easier to tack these on into the WHERE condition because some of these if joined would create duplicated data and I really don't need anything from these tables since they are just used as filters.

Anryate the speed of these has always been just as fast if not faster than doing the JOIN approach because the filtered subqueries are standalone and return very few records. For example the above example really only returns 4 records in the IN.

Jeff Davis
You didn't post EXPLAIN ANALYZE plans so it's difficult to give useful advice.

You should probably post to -performance, including explain analyze results, to get to the bottom of the problem.

Regina
Jeff,

Thanks for the suggestion. Unfortunately I can't for this piece without divulging snippets of code under NDA agreements. I'll try to formulate a standalone example that exhibits the same behavior. It would probably be easier to follow with a simplified reproducible standalone example anyway.

As Lance experienced too. Didn't run into this particular problem when running in 8.3 and the production one I recently upgraded to 8.4 from 8.3 (taking the 8.4 config and making the custom adjustments I had made to the 8.3).

Also as mentioned -- which was weird -- the development one only started to exhibit the same bizarre behavior after I vacuum analyzed the tables where as I imagine the production undergoes much more vacuuming since data is constantly being added to it.

Jeff Davis
It sounds like it may be a planner or stats bug. If multiple people are having problems, it would be nice to find a case that's free of sensitive data, code and queries. I understand that's not always possible though.

Regular Expressions in PostgreSQL

David Fetter
Great post!

Modern regex engines, including PostgreSQL's, allow for an expanded regex syntax like:

```
SELECT 'foo123' ~
'(?x)
f
o{2}
[[:digit:]]{3}
';
```

can make complex expressions less confusing :)

uberVU - social comments
This post was mentioned on Twitter by planetpostgres: Leo Hsu and Regina Obe: Regular Expressions in PostgreSQL http://tr.im/O7p9

The Pythian Blog
You have found the 179th edition of Log Buffer, the weekly review of database blogs. Welcome. Enjoy your stay. We begin with . . . SQL Server Merrill Alrich gets going with a fresh juxtaposition–his thoughts on motorcycles a...

PostGIS 1.5.0 out and PLR working on Windows 8.3-8.4 installs

topsy.com

www.pythian.com

www.pythian.com
