Postgres OnLine Journal: October 2009

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/

Beyond Nerds Bearing Gifts: The Future of the Open Source Economy

This week is a busy week for events. While PostgreSQL is having its PostgreSQL West conference in Seattle, the biggest Open Source GIS conference of the year is happening in Sydney, Australia FOSS4G 2009. Sadly given our schedule and the distances of the commutes, we couldn't make either conference.

Mateusz Loskot pointed out that the video is out for Paul Ramsey's FOSS4G 2009 Keynote speech on *Beyond Nerds Bearing Gifts: The Future of the Open Source Economy*. I think its a very important distinction Paul makes between selling software and selling a product, that a lot of people miss when trying to evaluate the solvency of open source software.

For those who don't know Paul, he's one of the co-founders of the PostGIS project and Refractions Research.

Back to Table Of Contents

Allocating People into Groups with Window aggregation *Intermediate*

This is along the lines of more stupid window function fun and how many ways can we abuse this technology in PostgreSQL. Well actually we were using this approach to allocate geographic areas such that each area has approximately the same population of things. So you can imagine densely populated areas would have smaller regions and more of them and less dense areas will have larger regions but fewer of them (kind of like US Census tracts). So you have to think about ways of allocating your regions so you don't have a multipolygon where one part is in one part of the world and the other in another etc. Using window aggregation is one approach in conjunction with spatial sorting algorithms.

The non-spatial equivalent of this problem is how do you shove people in an elevator and ensure you don't exceed the capacity of the elevator for each ride. Below is a somewhat naive way of doing this. The idea being you keep on summing the weights until you reach capacity and then start a new grouping.

To test this idea out lets create a table of people and their weights. The capacity of our elevator is 750 kg.

```
CREATE TABLE passengers(passenger_name varchar(20) PRIMARY KEY, weight_kg integer);
INSERT INTO passengers(passenger_name, weight_kg)
VALUES ('Johnny', 60),
       ('Jimmy', 120),
       ('Jenny', 50),
       ('Namy', 20),
       ('Grendel', 500),
       ('Charlie', 200),
       ('Gongadin', 400),
       ('Tin Tin', 10),
       ('Thumb Twins', 10),
       ('Titan', 600),
       ('Titania', 550),
       ('Titan''s Groupie', 55);

SELECT carriage,COUNT(passenger_name) As cnt_pass,
    array_agg(passenger_name) As passengers, SUM(weight_kg) As car_kg
FROM (SELECT passenger_name, weight_kg,
  ceiling(SUM(weight_kg)
    OVER(ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)/750.00) As carriage
FROM passengers) As congregation
GROUP BY carriage
ORDER BY carriage;
```

The result looks like

```
 carriage | cnt_pass |                 passengers                 | car_kg
----------+----------+--------------------------------------------+--------
        1 |        5 | {Johnny,Jimmy,Jenny,Namy,Grendel}          |    750
        2 |        4 | {Charlie,Gongadin,"Tin Tin","Thumb Twins"} |    620
        3 |        1 | {Titan}                                    |    600
        4 |        2 | {Titania,"Titan's Groupie"}                |    605
```

Back to Table Of Contents Allocating People into Groups with Window aggregation Reader Comments

Allocating People into Groups with SQL the Sequel *Advanced*

In our prior story about allocating people with the power of window aggregation, we saw our valiant hero and heroine trying to sort people into elevators to ensure that each elevator ride was not over capacity. All was good in the world until someone named Frank came along and spoiled the party. Frank rightfully pointed out that our algorithm was flawed because should Charlie double his weight, then we could have one elevator ride over capacity. We have a plan.

What was wrong with our first try? Well we failed to realize that while the number of rides you need at each increment of weight has to be at least the ceiling of the sum divided by the capacity of the car, it can exceed that number. This approach we think works fine if you are simply counting people. If we had normally weighted people instead of some dwarfs and giants that may work fine. So if we wanted to fit at most 5 people per car we would do this. Note we also changed our window to something that is a bit shorter but equivalent in meaning.

To test this idea out lets recreate our table of characters (Charlie being a little fatter now). We want at most 5 people per car run.

```
CREATE TABLE passengers(passenger_name varchar(20) PRIMARY KEY, weight_kg integer);
INSERT INTO passengers(passenger_name, weight_kg)
VALUES ('Johnny', 60),
           ('Jimmy', 120),
           ('Jenny', 50),
           ('Namy', 20),
           ('Grendel', 500),
           ('Charlie', 400),
           ('Gongadin', 400),
           ('Tin Tin', 10),
           ('Thumb Twins', 10),
           ('Titan', 600),
           ('Titania', 550),
           ('Titan''s Groupie', 55) ;

SELECT carriage,COUNT(passenger_name) As cnt_pass,
     array_agg(passenger_name) As passengers
FROM (SELECT passenger_name, weight_kg,
  ceiling(COUNT(passenger_name)
     OVER(ROWS UNBOUNDED PRECEDING)/5.0) As carriage
FROM passengers) As congregation
GROUP BY carriage
ORDER BY carriage;
```

The result looks like

```
  carriage | cnt_pass |                    passengers
----------+----------+-----------------------------------------------
         1 |        5 | {Johnny,Jimmy,Jenny,Namy,Grendel}
         2 |        5 | {Charlie,Gongadin,"Tin Tin","Thumb Twins",Titan}
         3 |        2 | {Titania,"Titan's Groupie"}
```

Of course this is not the problem we were trying to solve. We want to ensure that our car weights are at most 750 kg with giants in mind. So what to do. Sadly the power of window aggregates in PostgreSQL appears to fail us here but we can write a recursive query AKA (Recursive common table expression (CTE) with PostgreSQL 8.4. Sorry it looks kind of convoluted. The translation to this into spoken tongue is actually closer to what we said we were doing in the first place. If anyone can come up with a simpler way of achieving this, we'd like to know. I think there is a way to do this with a self-

join, but haven't thought that far into it.

```
WITH RECURSIVE p1 As
(SELECT 1 As carriage, passenger_name, weight_kg, weight_kg As car_weight
FROM passengers
        ORDER BY passenger_name LIMIT 1
),
congregation AS
(
        SELECT carriage, passenger_name, weight_kg, car_weight
FROM  p1
UNION ALL
SELECT np.carriage, np.passenger_name, np.weight_kg,  np.car_weight
 FROM
(SELECT CASE WHEN c.car_weight + p2.weight_kg <= 750
                        THEN c.carriage ELSE c.carriage + 1 END As carriage,
        p2.passenger_name, p2.weight_kg,
                CASE WHEN c.car_weight + p2.weight_kg <= 750
                        THEN c.car_weight + p2.weight_kg
                                ELSE p2.weight_kg END As car_weight
FROM passengers As p2 INNER JOIN
        (SELECT carriage, passenger_name, weight_kg, car_weight
                FROM congregation ORDER BY passenger_name DESC LIMIT 1)
                        As c ON (p2.passenger_name > c.passenger_name)
ORDER BY p2.passenger_name LIMIT 1) As np
)
SELECT carriage, COUNT(passenger_name) As cnt_pass,
        array_agg(passenger_name) As passengers, SUM(weight_kg) As car_kg
 FROM congregation
GROUP BY carriage;


 Result
carriage | cnt_pass |                    passengers                       | car_kg
----------+----------+-----------------------------------------------------+------
               1 |        1 | {Charlie}                                           |    400
               2 |        1 | {Gongadin}                                          |    400
               3 |        5 | {Grendel,Jenny,Jimmy,Johnny,Namy}                   |    750
               4 |        4 | {"Thumb Twins","Tin Tin",Titan,"Titan's Groupie"} |    675
               5 |        1 | {Titania}                                           |    550
```

Back to Table Of Contents Allocating People into Groups with SQL the Sequel Reader Comments

Enable and Disable Vacuum per table ***Beginner***

Vacuuming and analyzing is the process that removes dead rows and also updates the statistics of a table. As of PostgreSQL 8.3, auto vacuuming (the process that runs around cleaning up tables), is on by default. If you are creating a lot of tables and bulk loading data, the vacuumer sometimes gets in your way. One way to get around that is to disable auto vacuuming on the tables you are currently working on and then reenable afterward. You can also do this from the PgAdmin III management console.

Question:

How do you disable and reenable auto vacuuming for a select table?

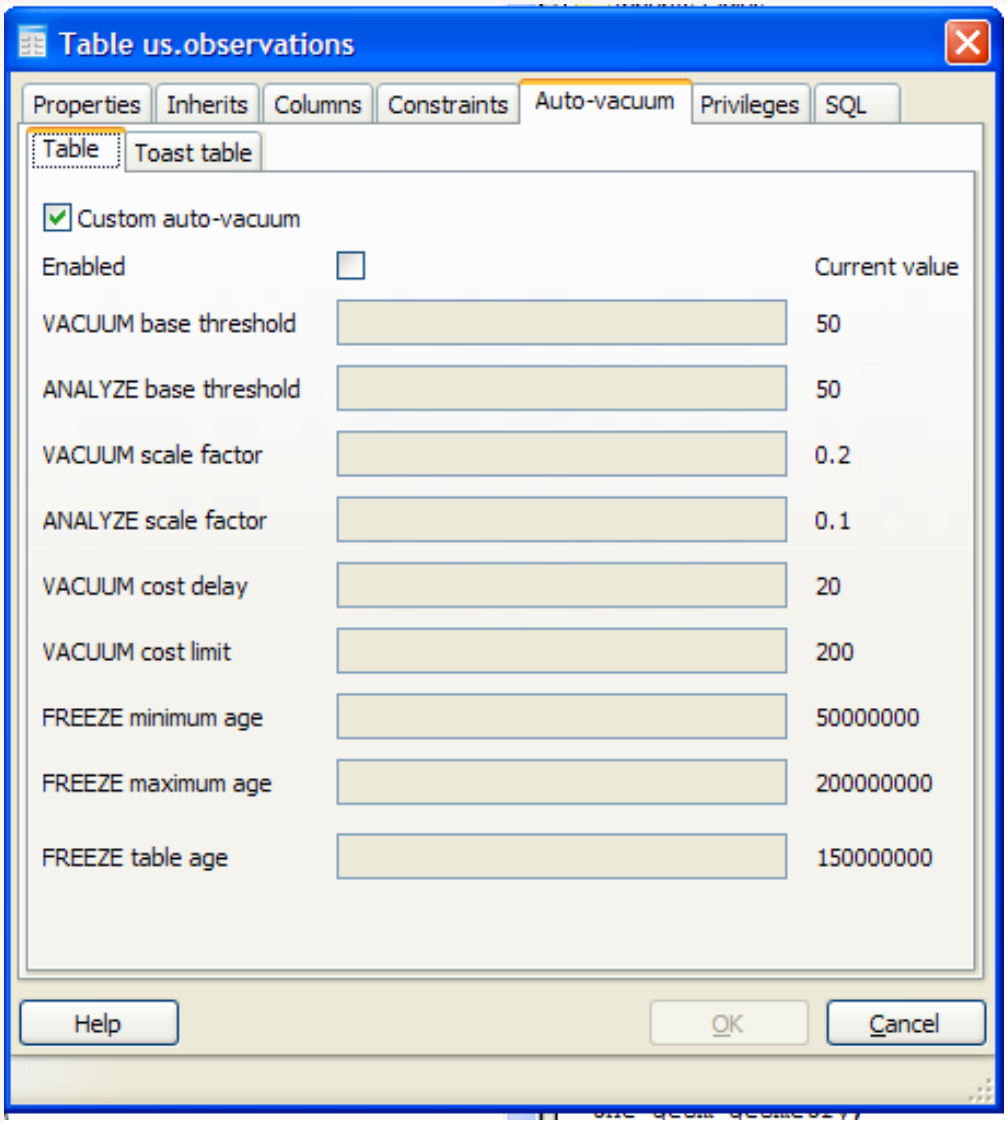
Solution: The SQL way

```
--disable auto vacuum
ALTER TABLE sometable SET (
  autovacuum_enabled = false, toast.autovacuum_enabled = false
);

--enable auto vacuum
ALTER TABLE sometable SET (
  autovacuum_enabled = true, toast.autovacuum_enabled = true
);
```

Solution: The PgAdmin III way

Navigate to the table in PgAdmin III explorer -> Select Table --> Properties and Click to go to Autovacuum tab



Note the other settings you can set in PgAdmin III. These can be set in SQL as well, and to see the SQL equivalent, simply set the settings, and switch to the PGAdmin III SQL tab. It will show the ALTER TABLE statement to run to make the change.

Back to Table Of Contents Enable and Disable Vacuum per table Reader Comments

Lowercasing table and column names *Beginner*

This is an unfortunate predicament that many people find themselves in and does cause a bit of frustration. You bring in some tables into your PostgreSQL database using some column name preserving application, and the casings are all preserved from the source data store. So now you have to quote all the fields everytime you need to use them. In these cases, we usually rename the columns to be all lower case using a script. There are two approaches we have seen/can think of for doing this one to run a script that generates the appropriate alter table statements and the other is to update the pg_attribute system catalog table directly.

Lowercase column names

This is our preferred way, because it doesn't assume anything about the underlying structure of the pg_catalog tables and therefore less likely to cause damage. It also gives you history about what will be changed so in a sense is self-documenting. We are really big on self-documenting structures.

```
--This generates SQL you can then run - lower case column names - this puts the sql in a single record
-- suitable for running in a single EXECUTE statement or cut and paste from PgAdmin query window
SELECT array_to_string(ARRAY(SELECT 'ALTER TABLE ' || quote_ident(c.table_schema) || '.'
  || quote_ident(c.table_name) || ' RENAME "' || c.column_name || '" TO ' || quote_ident(lower(c.column_name))
|| ';'
  FROM information_schema.columns As c
  WHERE c.table_schema NOT IN('information_schema', 'pg_catalog')
     AND c.column_name <> lower(c.column_name)
  ORDER BY c.table_schema, c.table_name, c.column_name
  ) ,
   E'\r') As ddlsql;

-- As David Fetter kindly pointed out, this looks cleaner, returns one record per ddl and is more psql friendly
SELECT  'ALTER TABLE ' || quote_ident(c.table_schema) || '.'
  || quote_ident(c.table_name) || ' RENAME "' || c.column_name || '" TO ' || quote_ident(lower(c.column_name))
|| ';' As ddlsql
  FROM information_schema.columns As c
  WHERE c.table_schema NOT IN('information_schema', 'pg_catalog')
     AND c.column_name <> lower(c.column_name)
  ORDER BY c.table_schema, c.table_name, c.column_name;
```

Lowercase table names

It goes without saying, that you may run into the same issue with table names. A similar trick works for that case.

```
--lower case table names - the PgAdmin centric way, single EXECUTE way
SELECT array_to_string(ARRAY(SELECT 'ALTER TABLE ' || quote_ident(t.table_schema) || '.'
  || quote_ident(t.table_name) || ' RENAME TO ' || quote_ident(lower(t.table_name)) || ';'
  FROM information_schema.tables As t
  WHERE t.table_schema NOT IN('information_schema', 'pg_catalog')
     AND t.table_name <> lower(t.table_name)
  ORDER BY t.table_schema, t.table_name
  ) ,
    E'\r') As ddlsql;

-- lower case table names -- the psql friendly and more reader-friendly way
SELECT 'ALTER TABLE ' || quote_ident(t.table_schema) || '.'
  || quote_ident(t.table_name) || ' RENAME TO ' || quote_ident(lower(t.table_name)) || ';' As ddlsql
```

```
  FROM information_schema.tables As t
  WHERE t.table_schema NOT IN('information_schema', 'pg_catalog')
     AND t.table_name <> lower(t.table_name)
  ORDER BY t.table_schema, t.table_name;

--generates something like this
ALTER TABLE public."SPRINT" RENAME TO sprint;
```

Back to Table Of Contents Lowercasing table and column names Reader Comments

Essential PostgreSQL DZone RefCardz is out

A while ago we wrote about DZone RefCards cheatsheets and how its a shame there isn't one for PostgreSQL. They are a very attractive and useful vehicle for learning and brushing up on the most important pieces of a piece of software or framework. Since that time we have been diligently working on one for PostgreSQL to fill the missing PostgreSQL slot. The fruits of our labor are finally out, and a bit quicker than we expected. The cheatsheet covers both old features and new features introduced in PostgreSQL 8.4. We hope its useful to many old and new PostgreSQL users.

The Essential PostgreSQL Refcard can be downloaded from Essential PostgreSQL http://refcardz.dzone.com/refcardz/essential-postgresql?oid=hom12841

Back to Table Of Contents Essential PostgreSQL DZone RefCardz is out Reader Comments

Reader Comments

Allocating People into Groups with Window aggregation

sysadmin.camerblog.net

Frank Heikens

This query has a bug, a simple update will break it:

UPDATE passengers SET weight_kg = 400 WHERE passenger_name = 'Charlie';

Execute the SELECT-query again and this is the result:
1;5;"{Johnny,Jimmy,Jenny,Namy,Grendel}";750
2;3;"{Gongadin,"Tin Tin","Thumb Twins"}";420
3;1;"{Titan}";600
4;3;"{Titania,"Titan's Groupie",Charlie}";1005

The fourth car_kg has to much kg, 1005.

The problem lies in the calculation of "carriage" and the order of the tubles on the pages, but so far I don't see a solution.

I did have a situation where a VACUUM fixed the result, but that's not acceptable.

Regina

Frank,

Thanks for the observation. Guess I have to think this out a little more.

Allocating People into Groups with SQL the Sequel

sysadmin.camerblog.net

Enable and Disable Vacuum per table

Bernd Helmle

Note that the described ALTER TABLE statements works since PostgreSQL 8.4 only. In earlier versions the DBA has to populate a system table called pg_autovacuum itself (or using PgAdmin III and the dialog you've described above). You might also find those settings not to be included in pg_dump before 8.4.

sysadmin.camerblog.net

Lowercasing table and column names

David Fetter

Wouldn't what's below, which is your code with the array_to_string(array())
peeled off, Just Work(TM)?

```
SELECT

    'ALTER TABLE ' || quote_ident(c.table_schema) || '.' ||

    quote_ident(c.table_name) || ' RENAME "' || c.column_name ||

    '" TO ' || quote_ident(lower(c.column_name)) || ';'

FROM information_schema.columns As c

WHERE

    c.table_schema NOT IN('information_schema', 'pg_catalog')

AND c.column_name <> lower(c.column_name)
```

```
ORDER BY c.table_schema, c.table_name, c.column_name;
```

Anyhow, one trick I've used when using SQL to generate SQL is to do it in psql.

* Invoke psql with -At (noalign, tuples only), or set same during the session.
* Check that the output looks right. What's below can shoot your whole leg off, not just put holes in your foot.
* \o |psql myuser mydb
* \g
* \o

Cheers,
David.

Regina
David,

I've been contemplating writing it the way you have it just so its easier to read. That works fine if you are working in psql, but most of the time we are working in PgAdmin III and so do most of our customers.

So the problem is that that approach gives you a record for each statement so when you copy and paste it into the query window it has all these annoying quotes around it that you then have to macro replace out.

Thanks,
Regina

David Fetter
I hadn't known about the requirement that this be done through pgAdminIII, which seems like a bizarre requirement at first blush. Anyhow, \r is a Windows-only thing which pretty well breaks on other platforms. It also doesn't quite handle best practices, which would be putting all DDL in your source code management system.

It seems to me that piling a kludge on top of the fact that you're using an unsuitable tool (pgAdminIII) for the job is a great way to waste a lot of time and effort where simply choosing a more appropriate tool is a much better way to approach things.

Of course, if yet another unstated requirement is to make the system obscure by creating secret knowledge, which in turn bumps up the hours you'll be charging to deal with it, your idea makes a large amount of sense.

Regina
David,

I take it I must have offended you with my windows/PgAdmin III centric lifestyle to deserve such a backlash or you are just a person who likes to start debates.

I have taken your advice and included the single row ddl way, but I still prefer my way better for my general usage.

Now getting back to your comment about bashing my favorite tool of choice and how I treat "my customers". For most day to day use I prefer PgAdminIII
(not everything) I like psql for a lot of things like automated testing and so forth. This is not necessarily one I feel one tool is greatly stronger than another.
We can't all be power psql users like you and not all people want to source control everything in their life.

I have come to accept the proposition that the factors that determine the best tool for the job are the job, the person doing it, and the current mood of that person. If I want my customers to be self-reliant, I need to let them use tools they feel most comfortable with -- e.g. ArcGIS this, Safe ETL that, and PgAdmin III whether or not I prefer those tools myself and to some extent, that does
mean I do need to understand them, and work around their short-comings etc. Trust me I have tried to get customers introduced to psql -- and often I get "Where are the lights?"

As Leo likes to say when he bought his first stick shift car "I buy a stick-shift because I feel more in control with it, its $2000 cheaper, and I know that since most of my friends can't drive
stick, they won't ask to play with my toy. Sometimes when I'm sitting on a long drive with lots of stops -- I really don't care to think that hard, I just want an automatic."

So to me psql is a stick -- sure you can do a lot of things with it you can't with PgAdmin III, but with PgAdminIII -- I click a

bunch of fairly intuitive buttons look at the sql it generates if I care, and go on with my business.
My customers don't care about learning the ins and outs of how to write ddl statements, they just want to create their damn tables and foreign keys and what not.

David Fetter
Regina,

You seem to be under the misapprehension that I dislike pgAdminIII and/or Windows. While I seldom use either of those tools, nothing could be further from the truth.

What you left out of your original article was that the procedure added code which specifically ties it to both. I understand that sometimes we forget to put in "the obvious," but in this case, you:

* Obfuscated the code, and in the process

* Broke how this would work in any toolchain other the two pieces of software you had in mind.

You stated no reason, and this puzzled at least me, and very likely others. To put it at its most generous, which I'm inclined to do, your code looks a lot like an entry for an obfuscated coding contest. If that is what it is, please do the rest of us the courtesy of so labeling it next time.

P.S. Please let people format code using "pre" (or other--Serendipity has lots of options) tags so our code doesn't flatten into illegibility when we post it.

Regina
David,

My point is I'm not going to jump into PSQL for something as trivial as this just because PSQL might be a slightly better tool for this. Ruins my train of thought is all constantly jumping from editing tool to editing tool. But you are correct -- I was oblivious to the fact that this may not behave as nicely in PSQL and other tools people commonly use for this. So thanks for that observation.

Hmm the code looks okay to me in my IE and Firefox browser. Which browser are you looking at? This is the first time I'm hearing its illegible.

Now the reason I don't use Serendipity for my code formatting is as you may say another "obvious to me but not to you", thing and that is because I don't display the content just in serendipity. I need to flip it into a PDF among other formats and I'm very particular about how that is done.

Essential PostgreSQL DZone RefCardz is out

Mark Lawrence
From the so-called "Privacy Agreement" at DRef:

"Postal addresses, and other personally identifying information and data, will be used to promote DZone and other DZone companies products and services, and may be rented and/or licensed to selected outside firms for promotional purposes..."

Even something as worthwhile as a PostgreSQL cheatsheet is not enough motivation for me to be putting all kinds of personal information into a site with a business model based on marketing.

Any chance of a less-encumbered download somewhere?

gregj
yeah, it is the pain. how about direct link to pdf somewhere ?

Sven Sporer
Full support to this proposal.

Regina
To all of you. Unfortunately no we can't provide a direct pdf. The next cheatsheet we do we might do in our standard format available for free download. it won't look as nice but awe well.

Normally if I am paranoid about the intentions of a site, I usually provide my junk email address or fake mail address. I personally haven't had issues with DZone FWIW.

Now why we chose this avenue -- for this cheatsheet in case anyone is wildly curious.

a) To capture more of a developer audience like those that frequent DZone. The audience is somewhat different from those that frequent most PostgreSQL sites and one we feel is needed to spread the word about PostgreSQL beyond its current

reach.

b) to have a professionally looking cheatsheet without having to pay a fortune for an art designer.

c) and forthright -- as an experimental avenue to get the word out there about our upcoming book. yes we have evil marketing needs as well :).

Its nice they paid us for our work, but that's of minor consequence to the aforementioned reasons.
