Postgres OnLine Journal: January 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/

How to determine what elements are in your enum ***Beginner***

This question was asked on the PostgreSQL novice newsgroup recently and Tom Lane fielded the question. Its a common thing to want to know if you are using the new 8.3 Enum feature. So we felt it useful to restate it.

Question: Given an enum, is there a query you can run to list out all the elements allowed by the enum?

Answer: Yes.

Below is just a regurgitation of the news item http://archives.postgresql.org/pgsql-novice/2008-12/msg00043.php

```
CREATE TYPE myenum as enum ('red','green','blue');
SELECT enumlabel
   FROM pg_enum
   WHERE enumtypid = 'myenum'::regtype
   ORDER BY oid;

 enumlabel
-----------
 red
 green
 blue
(3 rows)

http://www.postgresql.org/docs/8.3/static/catalog-pg-enum.html
```

Back to Table Of Contents

How to require all checked conditions are met by a result *Intermediate*

Problem You are developing a consultant search application where by a user looking for a consultant can check all the items in the list they require a consultant to have. You have 2 tables. **consultant** and **consultant_qual**. The consultant_qual has primary key formed by 2 fields *consultant_id* and *qual_id*. qual_id contains a constrained list with things like 'OpenLayers', 'PostGIS', 'Python', 'C#' etc.

How do you devise a query that given a list of checked options by the user, only returns consultants with not 1 but all of those qualifications?

Solutions

When looking at a problem like this, I am always amazed at the number of ways you can solve this in SQL. Some very convoluted and slow, some very convoluted looking and fast, and some pretty fast and pretty simple. How can that be when SQL is a declarative language? It all boils down to what angle you look at the problem or rather how you restate the problem and also the fact that like in other languages there are a myriad of ways to say the same thing and some databases and dataset patterns do better with one than another.

To demonstrate:

1. The divide and conquer approach: Treat this as an n step repeating problem and only solutions that are solutions to each step will be the solutions to the final problem.
2. The lets just do this in one gulp approach: The number of checked options by the user must equal the number of qualifications that a consultant satisfies that are in the list of checked options by the user.

Divide and Conquer Solutions

```
-- This will work in databases that support ANSI INTERSECT (e.g. PostgreSQL and SQL Server 2005/2008)
--but not in MySQL, SQL Server 2000 for example
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
      INNER JOIN consultant_qual AS Q ON C.consultant_id = Q.consultant_id
   WHERE Q.qual_id = 'OpenLayers'
INTERSECT
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
      INNER JOIN consultant_qual AS Q ON C.consultant_id = Q.consultant_id
   WHERE Q.qual_id = 'PostGIS';


--This will work in any database that supports EXISTS (which is a lot)
-- and uses several correlated subqueries (that is just variant -- I can count at least one more)
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
   WHERE EXISTS(SELECT Q.consultant_id
               FROM consultant_qual As Q
                  WHERE Q.qual_id = 'OpenLayers' AND C.consultant_id = Q.consultant_id)
      AND EXISTS(SELECT Q.consultant_id
               FROM consultant_qual As Q
                  WHERE Q.qual_id = 'PostGIS' AND C.consultant_id = Q.consultant_id);


--This will work in any database that supports IN (which is even more than EXISTS)
-- and uses several non-correlated subqueries
```

```
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
   WHERE C.consultant_id IN(SELECT Q.consultant_id
               FROM consultant_qual As Q
                  WHERE Q.qual_id = 'OpenLayers')
      AND C.consultant_id IN (SELECT Q.consultant_id
               FROM consultant_qual As Q
                  WHERE Q.qual_id = 'PostGIS');

-- This will work in any database that supports subselects in JOINS
-- which is a lot
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
      INNER JOIN
         (SELECT Q.consultant_id
            FROM consultant_qual As Q
                  WHERE Q.qual_id = 'OpenLayers') AS Q1
                  ON C.consultant_id = Q1.consultant_id
      INNER JOIN
         (SELECT Q.consultant_id
            FROM consultant_qual As Q
                  WHERE Q.qual_id = 'PostGIS') AS Q2
                  ON C.consultant_id = Q2.consultant_id ;
```

One Gulp solutions

The nice thing about the gulp solutions is that regardless of what the user checks, your query is more or less the same and always has the same number of sub queries. There are lots of permutations we can think of here too, but these are just 3 to demonstrate.

```
-- This will work in pretty much any relational database I can think of
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C INNER JOIN consultant_qual As Q
         ON C.consultant_id = Q.consultant_id
   WHERE  Q.qual_id IN('OpenLayers', 'PostGIS')
   GROUP BY C.consultant_id, C.consultant_name
   HAVING COUNT(Q.qual_id) = 2;


-- This will work in pretty much any relational database I can think of
-- If it supports subselects in WHERE it will work
SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
   WHERE 2 = (SELECT COUNT(Q.qual_id)
         FROM consultant_qual As Q
         WHERE Q.consultant_id = C.consultant_id
            AND Q.consultant_qual IN('OpenLayers', 'PostGIS')) ;

--This on first glance looks redundant,
-- but takes advantage of the fact that the first condition is easy for most relational dbs to optimize
-- and the second will never be run if the first check fails

SELECT C.consultant_id, C.consultant_name
   FROM consultant AS C
   WHERE EXISTS(SELECT Q.qual_id FROM consultant_qual As Q
         WHERE Q.consultant_qual IN('OpenLayers', 'PostGIS')
            AND Q.consultant_id = C.consultant_id) AND
            (2 = (SELECT COUNT(Q.qual_id)
         FROM consultant_qual As Q
         WHERE Q.consultant_id = C.consultant_id
            AND Q.qual_id IN('OpenLayers', 'PostGIS') ) ) ;
```

Back to Table Of Contents

SQL Coding Standards To Each His Own

I was reading Josh Berkus last blog post and was intrigued by his last post Writing Maintainable Queries Part I.

He is right that lots has been said about coding standards in other languages and even right out holy wars have been launched on the subject, but as far as SQL goes, not quite enough has been said on the subject for us to have a great war to end all wars.

I was also happy to see that we agreed with all his points except his first one. Yes I felt dissed, and thought hmm if someone as important as Josh thinks our aliases should be very descriptive and we should use the table name rather than the alias where possible, surely there must be something wrong with me for not believing in this fundamental philosophy.

In the rest of this excerpt I shall make fun of Josh and also set forth some of our own SQL Coding guidelines. Hopefully Josh won't take too much offense at this small jibe.

Getting back to Josh. Imagine a world filled with Josh Berkus wannabies taking his advice to its logical conclusion and what you get is this.

```
SELECT foot_from_left.foot AS the_foot_I_stuck_in_mouth, foot_from_right.foot AS
the_foot_from_the_right, mouth AS the_mouth_I_stuck_my_left_foot_in FROM foot AS foot_from_left
INNER JOIN foot AS foot_from_right ON foot_from_right.person = foot_from_left.person INNER JOIN
mouth ON ( mouth.person = foot_from_right.person) WHERE mouth.location = foot_from_left.location
AND foot_from_left.foot_position ='L' AND foot_from_right.foot_position = 'R'
```

I found a great cartoon depicting this somewhere, but don't seem to find this at the moment.

Yes we would be those disrespectful people who would be guilty of writing that statement as follows because those extra characters are not only more to type, but they actually get in the way of targetting join mistakes and where is the indentation. INDENTATION is the most important thing:

```
SELECT fl.foot AS fl_foot, fr.foot AS fr_foot, mouth.mouth
        FROM foot AS fl INNER JOIN foot AS fr ON fl.person = fr.person
                INNER JOIN mouth ON (mouth.person = fr.person)
        WHERE mouth.location = fl.location AND fl.foot_position = 'L' AND fr.foot_position = 'R'
```

Indenting, uppercasing SQL keywords, and using AS are the most important conventions we try to hold to.

Of course this is no offense to Josh - just a little nit-pick. Kind of reminds me of working on PostGIS, and one day to my disbelief I discovered the philosophies of 2 people I have great respect for look like this.

Paul

```
if (   lots_of_spaces_and_lots_of_under_scores && !camels  ){
        yeah();
}
```

Mark

```
if(!one) scream();
```

and of course all I could think -- *What barbarians! How could I even share the same planet with these people?*

When all is said and done, probably the most important thing when working on a team, is that everyone grudgingly agrees to follow the same sane standard and the code created done by 20 programmers looks like it was done by the same person.

Back to Table Of Contents SQL Coding Standards To Each His Own Reader Comments

SQL Coding Standards To Each His Own Part II

Both Josh Berkus and Hubert made blog entries about our last excerpt. In general I will start off by saying that we are more or less in agreement on what is good SQL coding format.

Here are the things I think we can all agree on

- SQL Keywords should be upper cased
- Field names should be prefixed with their tables especially when you have a multi-join statement involved
- Use JOIN syntax instead of stuffing everything in the WHERE though we are highly against just saying JOIN. We prefer INNER JOIN

The major points of contention I think are

- Should you use aliases over tables and if you use aliases should you keep them short or more descriptive. Josh thinks table names should be used where possible and when aliases are used they should be longer than a few characters and Hubert agrees with us that short aliases are fine and in fact desirable. I think we all agree aliases should be meaningful when used, but our idea of what it means to be meaningful is a little different.
- In use of JOIN syntax -- we prefer using INNER JOIN instead of using just JOIN and in fact find it quite irritating that PostgreSQL rewrites our INNERs as JOIN. I suspect Hubert and Josh and many other PostgreSQL folk are on the other side of the fence on this. The reason we feel strongly about this is there are so many kinds of JOINS - INNER JOIN, RIGHT JOIN, LEFT JOIN, CROSS JOIN, FULL JOIN, and the horrid NATURAL JOIN (that should be shot and put out of its misery). To just say JOIN to us is just confusing.
- While you can write LEFT OUTER JOIN, the OUTER is kind of pointless because no one goes around writing LEFT INNER JOINS
- Use well supported standards where possible. This means CURRENT_TIMESTAMP instead of now(). now() is not in all databases, but most relational databases support CURRENT_TIMESTAMP

So let us start by looking at Josh's last SQL statement and compare to how we would write it. Josh I hope you didn't come up with this table structure. We personally hate the use of **id** as a field identifier for every table, but that's another story. We prefer something like forum_id, post_id etc. and whats the deal with giving a field name of date. Stay away from data types as field names.

```
-- Josh
SELECT persons.id, persons.first_name, persons.last_name, forums.category,
  COUNT(DISTINCT posts.id) as num_posts,
  COALESCE(MAX(comments.rating), 0) AS highest_rating,
  COALESCE(MIN(comments.rating), 0) AS lowest_rating
FROM persons JOIN posts ON persons.id = posts.author
  JOIN forums on posts.forum = forums.id
  LEFT OUTER JOIN comments ON posts.id = comments.post
WHERE persons.status > 0
  AND forums.ratings = TRUE
  AND comments.date > ( now() - INTERVAL '1 year')
GROUP BY persons.id, persons.first_name, persons.last_name, forums.category
HAVING count(DISTINCT posts.id) > 0
ORDER BY persons.last_name, persons.first_name;


-- Us

-- Leo tends to prefer upper case alias for table lower case or proper case for fields.
-- Leo also likes to see his SELECT fields across one line regardless how many because
-- he has 2 wide screen ubber huge monitors and wants to take full advantage of them
-- I couldn't demonstrate this without messing up the format of the page.
```

```
SELECT P.id, P.first_name, P.last_name, F.category, COUNT(DISTINCT PO.id) as num_posts,
  COALESCE(MAX(C.rating), 0) AS highest_rating, COALESCE(MIN(C.rating), 0) AS lowest_rating
FROM persons AS P INNER JOIN posts AS PO ON P.id = PO.author
   INNER JOIN forums As F ON PO.forum = F.id
   LEFT JOIN comments AS C ON PO.id = C.post
WHERE P.status > 0
   AND F.ratings = TRUE
   AND C.date > ( CURRENT_TIMESTAMP - INTERVAL '1 year')
GROUP BY P.id, P.first_name, P.last_name, F.category
HAVING COUNT(DISTINCT PO.id) > 0
ORDER BY P.last_name, P.first_name;


-- Regina flip flops and often depends on database
-- she is working with that day or moment
-- whether she upper cases or lower cases or proper cases.
-- It must be noted that Regina also has 2 wide  ubber screens to work with
-- because Leo believes every programmer should have at least 36 inches to work with.
-- Once she finally gets adjusted to the enormity of these things,
-- she will use the space to keep 20 applications open at the same time and train her mind
-- for massive parallel concentration and get 20 times more work done
SELECT pe.id, pe.first_name, pe.last_name, f.category, COUNT(DISTINCT po.id) AS num_posts,
   COALESCE(MAX(c.rating), 0) AS highest_rating, COALESCE(MIN(c.rating), 0) AS lowest_rating
FROM persons AS pe INNER JOIN posts AS po ON pe.id = po.author
   INNER JOIN forums AS f ON po.forum = f.id
   LEFT JOIN comments AS c ON po.id = c.post
WHERE pe.status > 0
   AND f.ratings = TRUE
   AND c.date > ( CURRENT_TIMESTAMP - INTERVAL '1 year')
GROUP BY pe.id, pe.first_name, pe.last_name, f.category
HAVING COUNT(DISTINCT po.id) > 0
ORDER BY pe.last_name, pe.first_name;
```

I will start off by saying for the above example -- Josh is lucky because his table names are naturally short and we are disadvantaged in this example because two tables start with the same letter. Imagine if the table names were much more descriptive how repetitive and noisy Josh's scene would become.

The reason we prefer short aliases over long is that it allows you to fit more fields on the same line and scan them quickly and still prefix with the table names. Leo prefers upper case his table because he can quickly spot where a table name ends and field begins. Generally speaking you usually have 4 tables or fewer in a statement, the FROM clause is not that far away as a legend, and the tables usually start with different letters, so its not a large mind cramp for us to remember *F is for Forum and C is for Comments*.

Unlike other languages, you then need to keep on repeating this variable over and over again which is much more typing as well as just makes your statement that much longer to digest.

As a last gripe to what Josh said. Just because you have 255 characters at your disposal as opposed to the 8 you once had, doesn't mean you should feel compelled to use them all. Long live Fortran. It is older than any language I can think of including COBOL and has stood the test of time and is still undergoing innovation.

Back to Table Of Contents SQL Coding Standards To Each His Own Part II Reader Comments

Quick Intro to PLPython *Intermediate*

We have mentioned time and time again, one of the great selling points of PostgreSQL is that it has so many languages to choose from for writing database stored functions and the code you write in those stored functions is almost exactly the same as what you would write when writing in that language's environment. The reason for that is that PostgreSQL applies a thin layer around the environment the language lives in, so your code is really running in that environment. The down-side of this approach is you must have that environment installed on the server. This is a bit different from the Microsoft SQL Server model where code you write in VB.NET, C#, IronPython etc. gets translated into Common Runtime Logic (CLR) so your code is not really running in the environment it would normally breathe in and if you have dependencies you have to enable them in the SQL Server GAC which is different from the Server's .NET GAC.

In this section we shall introduce PL/Python - which is a PL language handler for Python that allows you to write PostgreSQL stored functions in Python. First of all I should start off by saying that we are not proficient Python programmer's so if anyone sees anything wrong with what we say feel free to embarass us.

We are also taking this opportunity to test-drive PostgreSQL 8.4 on both Linux (OpenSUSE) and Windows, using the EnterpriseDB PostgreSQL 8.4 beta that Dave Page recently announced on his blog. This install is great if you are running Windows, MacOSX or Linux Desktop, but sadly does not have PostGIS as part of the stack builder option.

For pure Linux Server CentOS/Redhat EL/Fedora no desktop installs or if you just feel more comfortable at the command-line, PostgreSQL Yum repository generously maintained by Devrim is the one to go for.

We haven't tested this one out, but I presume the steps are pretty much what we outlined in Using PostgreSQL Yum repository.

Installing PostgreSQL 8.4 beta

Some things to watch out for which may not be entirely obvious if Linux is new to you.

- Before you can run the Linux .bin installs, you must make them executable by either chmod 777 thebinfile or in explorer GUI right-click and mark as executable. Windows users can skip this step.
- If you are running another PostgreSQL on your box, give this a different port when the wizard asks, say 5434 or 5433.

Alas the taste of the serpent: Installing Python

One thing I find very intriguing about the language Python is that it seems that every hot shot GIS programmer programs in it and prefers it to any other language. In fact it is almost a tautology, *If you don't program in Python, you must not be a hot shot GIS programmer, though you could be a hot shot Spatial Database Programmer.* I haven't used Python enough to figure out what these people see in this language, but there must be a reason for its strong following particularly in the GIS industry. Even ESRI applications install Python which seems kind of odd to me if you look at the strong .NET/Servlet infrastructure underneath the ESRI architecture. Their programmers must have been throwing severe temper tantrums for ESRI to allow this to happen.

Can I use PL/Python under PostgreSQL Windows?

Yes. Though in general Python is not preinstalled so you must install it.

- Install **Python 2.5** by getting from http://www.python.org/download/releases/ or using Linux distro. It must be Python 2.5 since that is what the PostgreSQL 8.4 beta builds are compiled against. As of this writing Python 2.5.4 is the latest of the 2.5 series.

Can I use PL/Pyton under PostgreSQL Unix

Of Course. What PostgreSQL thing can you not use under Unix? Mac OSX. In fact a lot of Linux installs have Python already loaded so your life is surprisingly easy here.

Installing PL/Python: Our gateway to the serpent

To install PLPython simply run the following on your favorite database. By either using the PgAdmin III or with psql

```
CREATE PROCEDURAL LANGUAGE 'plpythonu' HANDLER plpython_call_handler;
```

However if you get a message when installing it that it couldn't be loaded most likely you do not have Python 2.5 installed or it can not be found.

Our first PLPython stored function

PLPython is an untrusted language which means you can do dangerous things with it if you want and you should be more careful about what accounts you allow to write these functions.

Simple finding if a file exists

```
CREATE OR REPLACE FUNCTION fnfileexists(IN afilename text) RETURNS boolean AS
$$
   import os
   return os.path.exists(afilename)
$$
LANGUAGE 'plpythonu' VOLATILE;


--testing the function --
SELECT fnfileexists(E'C:\\test.htm')


fnfileexists
-----------
t
```

PLPython and default parameters

Now we shall test drive PL/Python with a new feature introduced in 8.4 called *default parameters*. As a side note, in an unrelated article entitled Chocolate and Peanut Butter Cross-Breeding with PostgreSQL, SQL Server 2008, and Oracle on our BostonGIS site, we griped a little bit about how PostgreSQL has no default parameters like Oracle does and alas in 8.4 it has it and better yet you can even use it in Python.

```
CREATE OR REPLACE FUNCTION  fndumencoder(randstring text,
   mapfrom text DEFAULT 'abcdedfhijklmnopqrstuvwxyz',
   mapto text DEFAULT 'bcdefghijklmnopqrstuvwxyza'  )
RETURNS text AS
$$
   import string
   mapt = string.maketrans(mapfrom, mapto)
   return randstring.lower().translate(mapt)
$$
  LANGUAGE 'plpythonu' VOLATILE;


CREATE OR REPLACE FUNCTION  fndumdecoder(randstring text,
   mapfrom text DEFAULT 'abcdedfhijklmnopqrstuvwxyz',
   mapto text DEFAULT 'bcdefghijklmnopqrstuvwxyza')
RETURNS text AS
```

```
$$
   import string
   mapt = string.maketrans(mapto, mapfrom)
   return randstring.lower().translate(mapt)
$$
  LANGUAGE 'plpythonu' VOLATILE;


--Testing the functions using default values
SELECT fndumencoder('Johnny thinks too much');
fndumencoder
------------
kpiooz uijolt upp nvdi

SELECT fndumdecoder('kpiooz uijolt upp nvdi');
fndumdecoder
------------
johnny thinks too much

--Testing using our own trivial mapping
SELECT fndumencoder('Johnny thinks too much', 'abcdefghijk', '11234567890');

fndumencode
------------
9o7nny t78n0s too mu27

SELECT fndumdecoder('9o7nny t78n0s too mu27', 'abcdefghijk', '11234567890');
fndumdecoder
------------
johnny thinks too much
```

Back to Table Of Contents

Fusion Charts and PostgreSQL Part 2: ASP.NET Dashboard *Intermediate*

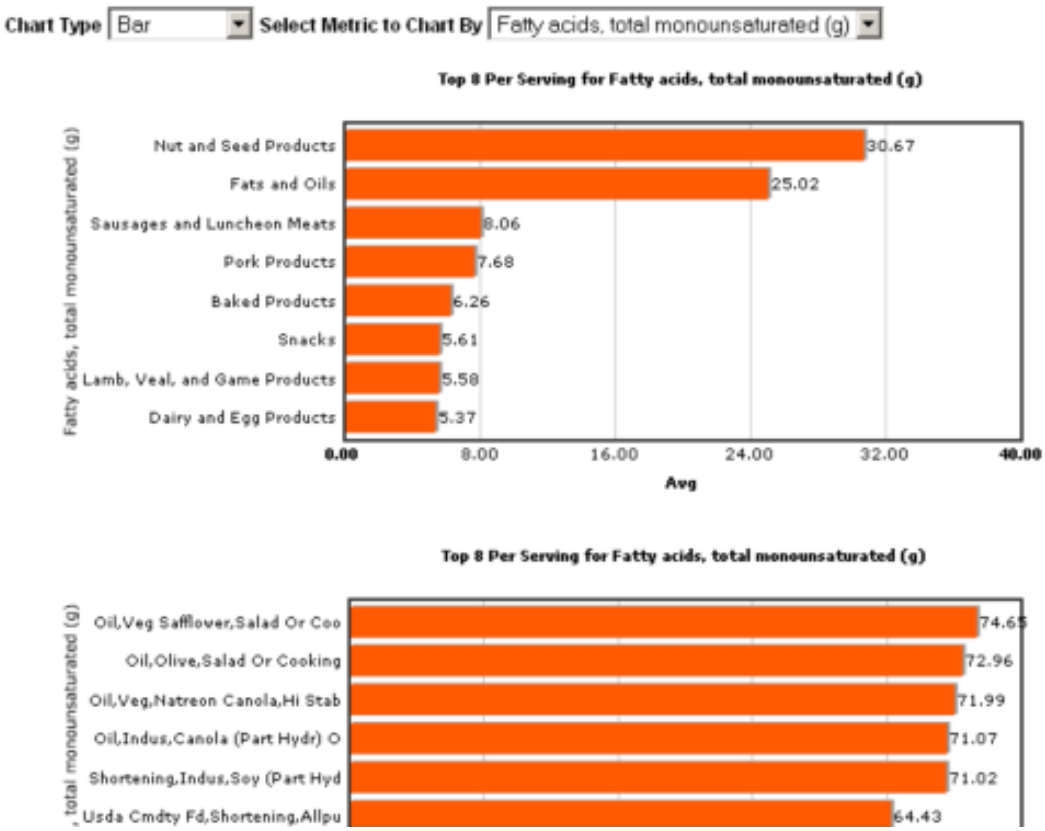
In the first part of this series Fusion Charts and PostgreSQL Part 1: Database Analysis of USDA DB in our November/ December 2008 issue, we did some back-seat analysis of a database we had never seen before and formulated some thoughts of things that would be useful to see in a dashboard as well as starting to develop some views to support our Dashboard.

In this part, we start the fun off by building an ASP.NET app in both VB and C#. In the next part of this series, we shall perform the same feat with PHP.

We are going to create a simple dashboard that has the following features:

1. A drop downlist to allow the user to pick the kind of chart to display the data in (Bar, column, funnel etc)
2. A drop downlist that allows the user to pick the metric to explore -- e.g. Cholestrol, Vitamin K, Caffeine etc.
3. 2 charts -- one chart showing the top 5 food groups for our metric and another showing the top 5 foods for our metric

Our final product will look like this:



You can see the app in action - USDA Food Stats and discover some interesting things about the food you eat or were considering eating.

Our tools and Structure

What we will need for this exercise will be the following:

1. PostgreSQL USDA database setup from Part 1 (8.1+)
2. FusionCharts Free version will do which you can download from - Fusion Charts Free
3. ASP.NET 2.0 with Framework 3+ or Mono.NET equivalent
4. Npgsql.net 2.0.2 libraries for connectivity to PostgreSQL db, which you can download from http://pgfoundry.org/frs/?group_id=1000140&release_id=1273

For this exercise we shall also be using a homegrown Db wrapper to wrap the npgsql classes. This just makes it a little easier to swap one driver/database platform for another

The VB App
VB code behind - ViewChartsVB.aspx.vb

```
Imports InfoSoftGlobal
Partial Class ViewChartsVB
    Inherits System.Web.UI.Page
    Public DbPg As PC.Data.DbAccessor = PC.Data.DbAccessor.GetDbAccessor(System.Configuration.
ConfigurationManager.ConnectionStrings("DSN").ConnectionString, _
        "PC.Data.PGSQLDbAccessor")


    Public Function CreateChart(ByVal aChartName As String, ByVal aViewName As String) As String
        'strXML will be used to store the entire XML document generated
        Dim strXML, sql As String
        Dim outputTag As String = "</chart>"
        Dim topn As String = "8"

        'Generate the graph element
        strXML = ""
        Select Case Me.DropDownListChartType.SelectedValue
            Case "FCF_Funnel"
                strXML = "<chart isSliced='1' slicingDistance='4' decimalPrecision='2' subcaption='Top " & topn & " By " &
_
                    Me.DropDownListAttribute.SelectedItem.Text & "'>"
                outputTag = "</chart>"

            Case "FCF_Column3D", "FCF_Doughnut2D", "FCF_Pie3D", "FCF_Bar2D"
                strXML = "<graph showNames='1'  decimalPrecision='2'  formatNumberScale='0' rotateNames='1'
caption='Top " & topn & _
                    " Per Serving for " & Me.DropDownListAttribute.SelectedItem.Text & "' xAxisName='" & _
                        Me.DropDownListAttribute.SelectedItem.Text & "' yAxisName='Avg' >"
                outputTag = "</graph>"
        End Select

        sql = "SELECT Substr(item_name,1,30) As item_name, avgnutritionalvalue As tot, units " & _
            " FROM   " & DbPg.sb(aViewName) & _
            " WHERE metric = COALESCE(" & DbPg.StringToSQLNull(Me.DropDownListAttribute.SelectedValue, PC.Data.
DataTypes.DbVarChar) & ", 'Unknown') " & _
            " ORDER BY avgnutritionalvalue DESC " & _
            " LIMIT " & topn

        Using dr As System.Data.IDataReader = DbPg.GetDR(sql)
            While dr.Read()
                strXML = strXML & "<set name='" & Server.HtmlEncode(dr("item_name").ToString().Replace("&", " "))
& "' value='" & dr("tot").ToString() & "' />"
            End While
        End Using

        strXML = strXML & outputTag

        'Create the chart - Pie 3D Chart with data from strXML
```

```
        Return FusionCharts.RenderChart("FusionCharts/" & Me.DropDownListChartType.SelectedValue & ".swf", "",
strXML, aChartName, "650", "300", False, False)
    End Function

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        If Not Page.IsPostBack Then
            Me.DropDownListAttribute.DataSource = _
                DbPg.GetDataSet("SELECT metric,  metric || ' (' || units || ')' As display FROM vwfdgrpstats GROUP BY
metric, units ORDER BY metric")
            Me.DropDownListAttribute.DataBind()
        End If
        If Me.DropDownListAttribute.SelectedValue > "" Then
            Me.litChartFoodGroup.Text = Me.CreateChart("chartFoodGroup", "vwfdgrpstats")
            Me.litChartFood.Text = Me.CreateChart("chartFood", "vwfoodstats")
        End If
    End Sub
End Class
```

ASPX display interface ViewChartsVB.aspx

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="ViewChartsVB.aspx.vb" Inherits="ViewChartsVB" %
>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Charts</title>
    <SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js"></SCRIPT>
    <style type="text/css">
    <!--
    body {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 12px;
    }
    .text{
        font-family: Arial, Helvetica, sans-serif;
        font-size: 12px;
    }
    -->
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <b>Chart Type</b>
        <asp:DropDownList ID="DropDownListChartType" runat="server" AutoPostBack="True">
            <asp:ListItem Text="Bar" Value="FCF_Bar2D" />
            <asp:ListItem Text="Funnel" Value="FCF_Funnel" />
            <asp:ListItem Text="Pie 3D" Value="FCF_Pie3D" />
            <asp:ListItem Text="Column 3D" Value="FCF_Column3D" />
            <asp:ListItem Text="Doughnut" Value="FCF_Doughnut2D" />
        </asp:DropDownList>
        <b>Select Metric to Chart By</b>
        <asp:DropDownList ID="DropDownListAttribute" DataTextField="display" DataValueField="metric"
runat="server" AutoPostBack="True" />
        <asp:Literal ID="litChartFoodGroup" runat="server" />
```

```
    <asp:Literal ID="litChartFood" runat="server" />
  </div>
  </form>
</body>
</html>
```

The C# App
CS code behind - ViewChartsCS.aspx.cs

```
using InfoSoftGlobal;
partial class ViewChartsCS : System.Web.UI.Page {
  public PC.Data.DbAccessor DbPg = PC.Data.DbAccessor.GetDbAccessor(System.Configuration.
ConfigurationManager.ConnectionStrings["DSN"].ConnectionString,
      "PC.Data.PGSQLDbAccessor");



public string CreateChart(string aChartName, string aViewName)
{
  //strXML will be used to store the entire XML document generated
  string strXML = null;
  string sql = null;
  string outputTag = "</chart>";
  string topn = "8";

  //Generate the graph element
  strXML = "";
  switch (this.DropDownListChartType.SelectedValue) {
    case "FCF_Funnel":
      strXML = "<chart isSliced='1' slicingDistance='4' decimalPrecision='2' subcaption='Top " + topn + " By "
        + this.DropDownListAttribute.SelectedItem.Text + "'>";
      outputTag = "</chart>";
      break;
    case "FCF_Column3D":
    case "FCF_Doughnut2D":
    case "FCF_Pie3D":
    case "FCF_Bar2D":
      strXML = "<graph showNames='1'  decimalPrecision='2'  formatNumberScale='0' rotateNames='1'
caption='Top " + topn + " Per Serving for " +
        this.DropDownListAttribute.SelectedItem.Text + "' xAxisName='" + this.DropDownListAttribute.
SelectedItem.Text + "' yAxisName='Avg' >";
      outputTag = "</graph>";
      break;
  }

  sql = "SELECT Substr(item_name,1,30) As item_name, avgnutritionalvalue As tot, units " + " FROM   " + DbPg.sb
(aViewName) + " WHERE metric = COALESCE(" +
    DbPg.StringToSQLNull(this.DropDownListAttribute.SelectedValue, PC.Data.DataTypes.DbVarChar) + ",
'Unknown') " +
      " ORDER BY avgnutritionalvalue DESC LIMIT " + topn;

  using (System.Data.IDataReader dr = DbPg.GetDR(sql)) {
    while (dr.Read()) {
      strXML = strXML + "<set name='" + Server.HtmlEncode(dr["item_name"].ToString().Replace("&", " ")) +
"' value='" + dr["tot"].ToString() + "' />";
    }
  }

  strXML = strXML + outputTag;

  //Create the chart - Pie 3D Chart with data from strXML
```

```
   return FusionCharts.RenderChart("FusionCharts/" + this.DropDownListChartType.SelectedValue + ".swf", "",
strXML, aChartName, "650", "300", false, false);
}




   protected void Page_Load(object sender, System.EventArgs e)
   {
      if (!Page.IsPostBack) {
         this.DropDownListAttribute.DataSource = DbPg.GetDataSet("SELECT metric,  metric || ' (' || units || ')' As
display " +
               " FROM vwfdgrpstats GROUP BY metric, units ORDER BY metric");
         this.DropDownListAttribute.DataBind();
      }
      if (! (this.DropDownListAttribute.SelectedValue == "")) {
         this.litChartFoodGroup.Text = this.CreateChart("chartFoodGroup", "vwfdgrpstats");
         this.litChartFood.Text = this.CreateChart("chartFood", "vwfoodstats");
      }
   }
}
```

CS ASPX - ViewChartsCS.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ViewChartsCS.aspx.cs" Inherits="ViewChartsCS" %
>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Charts</title>
   <SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js"></SCRIPT>
   <style type="text/css">
   <!--
   body {
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
   }
   .text{
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
   }
   -->
   </style>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <b>Chart Type</b>
      <asp:DropDownList ID="DropDownListChartType" runat="server" AutoPostBack="True">
         <asp:ListItem Text="Bar" Value="FCF_Bar2D" />
         <asp:ListItem Text="Funnel" Value="FCF_Funnel" />
         <asp:ListItem Text="Pie 3D" Value="FCF_Pie3D" />
         <asp:ListItem Text="Column 3D" Value="FCF_Column3D" />
         <asp:ListItem Text="Doughnut" Value="FCF_Doughnut2D" />
      </asp:DropDownList>
      <b>Select Metric to Chart By</b>
      <asp:DropDownList ID="DropDownListAttribute" DataTextField="display" DataValueField="metric"
runat="server" AutoPostBack="True" />
      <asp:Literal ID="litChartFoodGroup" runat="server" />
```

```
      <asp:Literal ID="litChartFood" runat="server" />
   </div>
   </form>
</body>
</html>
```

Back to Table Of Contents Fusion Charts and PostgreSQL Part 2: ASP.NET Dashboard Reader Comments

Fusion Charts and PostgreSQL Part 3: PHP Dashboard *Intermediate*

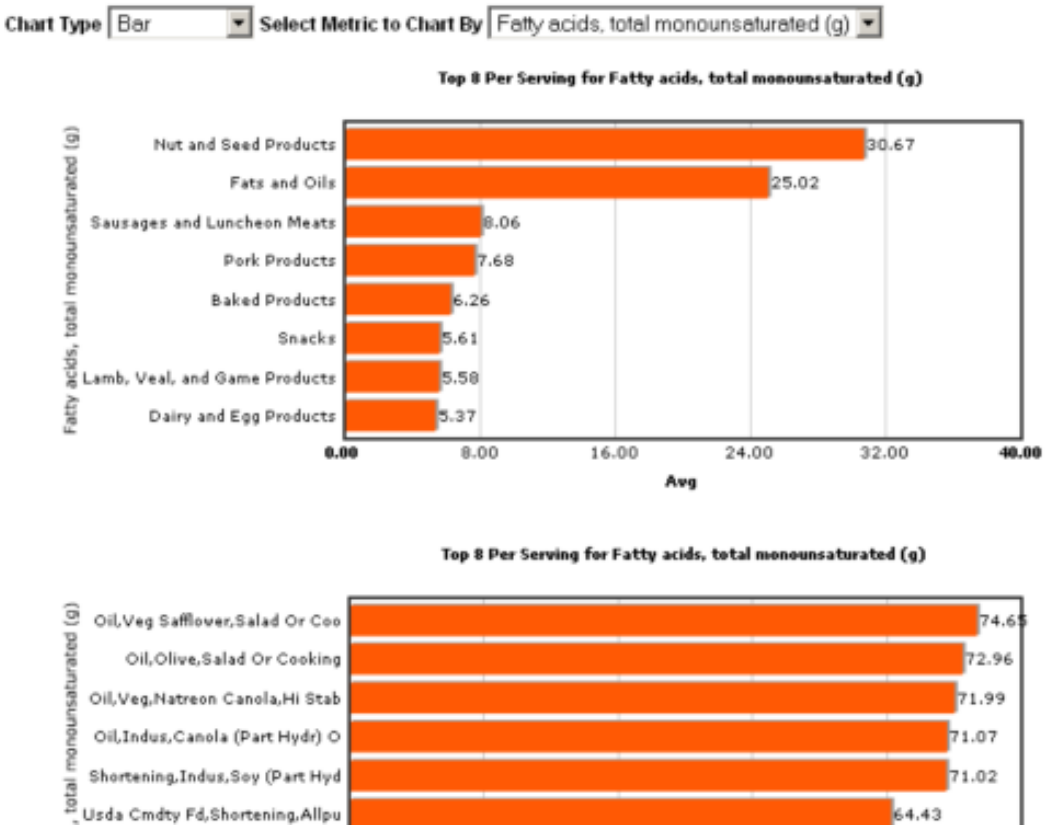
In the first part of this series Fusion Charts and PostgreSQL Part 1: Database Analysis of USDA DB in our November/ December 2008 issue, we did some back-seat analysis of a database we had never seen before and formulated some thoughts of things that would be useful to see in a dashboard as well as starting to develop some views to support our Dashboard. In the second part of our Fusion Charts series, we covered creating a Dashboard application in ASP.NET that demonstrated both VB.NET and C# variants using the database we loaded and prepped in Part 1.

In this part three of our series, we shall conclude by demonstrating the same application we did in ASP.NET in PHP.

We are going to create a simple dashboard that has the following features:

1. A drop-down list to allow the user to pick the kind of chart to display the data in (Bar, column, funnel etc)
2. A drop-down list that allows the user to pick the metric to explore -- e.g. Cholestrol, Vitamin K, Caffeine etc.
3. 2 charts -- one chart showing the top 5 food groups for our metric and another showing the top 5 foods for our metric

Our final product will look like this:



You can see the app in action - USDA Food Stats and discover some interesting things about the food you eat or were considering eating.

Tools and Structure

What we will need for this exercise will be the following:

1. PostgreSQL USDA database setup from Part 1 (8.1+)
2. FusionCharts Free version will do which you can download from - Fusion Charts Free
3. PHP 5+
4. PHP Smarty Templating Engine which you can get from www.smarty.net -

   Long aside: Paragon forcing their twisted philosophies on poor unsuspecting readers. You can ignore this if you want, unless you wish to be enlightened :).
   We have to add, that those who develop PHP without a templating engine should try one. For very simple applications, it might be overkill, but as your applications become more complex, it becomes extremely useful. You just can't imagine the value of a templating engine until you use one. The main benefits to us have been:
   1. *It allows you to easily cater to different devices and produce numerous output layouts without making your basic logic difficult to understand. Think for example if you have a report aht rquires you ca output in wordml, open office xmla nd html format that needs to confirm to an extremely defined design spec. It is just so much easier to do it with a template than try to generate different variants with standard PHP syntax.*
   2. *When those designers give you their vision of what a page is supposed to look like, its easier to integrate their vision into the overall architecture without screwing up your vision of how the logic should flow.*
   3. *Display concerns lend themselves to a different style of programming than what you would expect directly from a business flow tier. It keeps your sanity if these two styles aren't swashed together.*
5. PHP ADODB Database Abastration Layer (Note: Pear DB and others are equally good. The main benefit of a DB Abstraction layer is you don't need to remember the different function calls each db platform uses and can concentrate on just writing SQL. It also makes it a little easier to swap one db out for another.) which you can download from http://adodb.sourceforge.net/ and PHP pgsql driver enabled in php.ini

The PHP App

Begin by creating folders in your web app directories called

1. **templates** - this will be where our .tpl file resides.
2. **templates_c** - this needs to be writeable by your web server process and is where Smarty compiles the templates into php files
3. **libs** - extract adodb and smarty into this directory and put the FusionCharts/FusionCharts.php such that you have 3 folders -- adodb, smarty, FusionCharts
4. Create another folder called FusionCharts on root of webfolder and put the FusionCharts flash and .js files there.

The Config file

```
<?php
define("DSN", 'postgres://usda_app:foodie@localhost:5432/usda?persist');
?>
```

Controller Logic - view_charts.php

```
<?php
require_once('config.inc.php');
require_once('libs/smarty/Smarty.class.php');
require_once('libs/adodb/adodb.inc.php');
require_once('libs/FusionCharts/FusionCharts.php');
class _view_charts extends Smarty{

   protected $db;
   protected $charttypes = array('FCF_Bar2D' => 'Bar', 'FCF_Funnel'=>'Funnel',
         'FCF_Pie3D' => 'Pie 3D', 'FCF_Column3D' => 'Column 3D',
         'FCF_Doughnut2D' => 'Doughnut');
   protected $rs_attributes;
   function __construct() {
      $this->db = NewADOConnection(DSN);
      if (!$this->db) {
         die("Connection failed");
```

```
    }
    $this->rs_attributes = $this->db->Execute("SELECT metric,  metric || ' (' || units || ')' As display
            FROM vwfdgrpstats GROUP BY metric, units ORDER BY metric")->GetAssoc();
    $this->page_load();
  }

  function page_load(){
    $this->assign('rs_attributes', $this->rs_attributes);
    $this->assign('rs_charttypes', $this->charttypes);

    $this->assign('lit_chart_food_group', $this->create_chart("chartFoodGroup", "vwfdgrpstats"));
    $this->assign('lit_chart_food', $this->create_chart("chartFood", "vwfoodstats"));

    $this->display('view_charts.tpl');
  }

  function create_chart($achart_name, $aview_name){
    //str_xmlwill be used to store the entire XML document generated
    $str_xml = null;
    $sql = null;
    $output_tag = '';
    $topn = '8';
    $charttype = key($this->charttypes);

    //default attribute to first element in list
    $att_display = reset($this->rs_attributes);
    $att_value = key($this->rs_attributes);



    if (!empty($this->rs_attributes[$_REQUEST['dropdownlist_attribute']])){
      $att_display = $this->rs_attributes[$_REQUEST['dropdownlist_attribute']];
      $att_value = $_REQUEST['dropdownlist_attribute'];
    }

    if (!empty($_REQUEST['dropdownlist_charttype'])){
      $charttype = $_REQUEST['dropdownlist_charttype'];
    }

    //Generate the graph element
    $str_xml = "";
    switch ($charttype) {
      case "FCF_Funnel":
        $str_xml = "<chart isSliced='1' slicingDistance='4' decimalPrecision='2' subcaption='Top $topn  By "
          . $att_display . "'>";
        $output_tag = "</chart>";
        break;
      case "FCF_Column3D":
      case "FCF_Doughnut2D":
      case "FCF_Pie3D":
      case "FCF_Bar2D":
        $str_xml = "<graph showNames='1'  decimalPrecision='2'  formatNumberScale='0' rotateNames='1'
caption='Top " . $topn . " Per Serving for " .
          $att_display .
          "' xAxisName='" . $att_display  . "' yAxisName='Avg' >";
        $output_tag = "</graph>";
        break;
    }

    $sql = "SELECT Substr(item_name,1,30) As item_name, avgnutritionalvalue As tot, units
          FROM   " . $aview_name . " WHERE metric = COALESCE(" .
      $this->db->qstr($att_value) . ", 'Unknown') " .
```

```
            " ORDER BY avgnutritionalvalue DESC";

        $dr = $this->db->SelectLimit($sql, $topn);


        while (!$dr->EOF) {
            $str_xml = $str_xml . "<set name='" . preg_replace('/[^A-Za-z0-9\s\s+]/',' ',$dr->fields
("item_name")) . "' value='" . $dr->fields("tot") . "' />";
            $dr->MoveNext();
        }
        $dr->Close();
        $dr = null;

        $str_xml = $str_xml . $output_tag;

        //Create the chart with data from strXML
        return renderChart("FusionCharts/" . $charttype . ".swf", "", $str_xml, $achart_name, "650", "300", false,
false);
    }
}
new _view_charts();
?>
```

Display interface templates/view_charts.tpl

```
        <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Charts</title>
    <SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js"></SCRIPT>
    <style type="text/css">
    {literal}
    <!--
    body {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 12px;
    }
    -->
    {/literal}
    </style>
    <SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js"></SCRIPT>
</head>
<body>
    <form id="form1" action="{$smarty.request.url}" method="post">
    <div>
        <b>Chart Type</b>
            <select id="dropdownlist_charttype" name="dropdownlist_charttype" onchange="this.form.submit()">
                {html_options options=$rs_charttypes selected=$smarty.request.dropdownlist_charttype}
            </select>
        <b>Select Metric to Chart By</b>
            <select id="dropdownlist_attribute" name="dropdownlist_attribute" onchange="this.form.submit()">
                {html_options options=$rs_attributes selected=$smarty.request.dropdownlist_attribute}
            </select>
        {$lit_chart_food_group}
        {$lit_chart_food}
    </div>
    </form>
</body>
</html>
```

Back to Table Of Contents Fusion Charts and PostgreSQL Part 3: PHP Dashboard Reader Comments

On the topic of Cheat Sheets: DZone RefCardz

This is a product that has nothing to do with PostgreSQL yet, so it does seem kind of odd that we are listing this in our product showcase section.

Well it is not even really a product per se, but I was just so enamored by the beauty of the layout and the usefulness of these cheat sheets, that I felt it was worthy of being listed in our showcase section since some of these would be useful to the PostgreSQL community of programmers.

What is this product you ask? It is DZone RefCardz. When I discovered this array of cheat sheets, I became a glutton and downloaded about 10 of them. It made me even want to start learning Ruby and also made me realize how little I know about CSS.

I was disappointed to find out that while they have an *Essential MySQL*, which by the way is extremely useful, they don't also have an *Essential PostgreSQL* yet. Someone should really write one of these things for PostgreSQL, but I guess it would be best for that someone to be a published author since it seems the main focus of RefCardz is as a publicity card for an author's book.

As a side note, it did get me thinking about the format of Postgres OnLine Journal cheat sheets and as many people have mentioned, perhaps we do try to cram too much information on one page. The layout of RefCardz cheat sheets seems to provide a good balance between amount of content and brevity and made me realize that having a multi-page cheat sheet is not such a bad thing. Their choice of colors, layout and diagrams is just mesmerizing.

Back to Table Of Contents On the topic of Cheat Sheets: DZone RefCardz Reader Comments

PostgreSQL 8.3 PSQL Cheatsheet Overview

PSQL is the command line administration/query tool for PostgreSQL. It is simple, light-weight and packed with a lot of charm. Unfortunately we can't quite capture all its charm in a single pager cheat sheet, but hopefully this will give you a sense of its usefulness.

Below is a Thumbnail view of the PSQL cheat sheet. This is by no means comprehensive, but are the features we use most often. It is also pretty much what you get from the help screen with just some added color.

PDF Portrait version 8.5 x 11" of this cheatsheet is available at PSQL8.3 Cheatsheet 8.5 x 11 and also available in PDF A4 format and HTML.

Hubert has some neat tips for making more effective use of psql

- keyboard shortcuts in psql
- Suppressing CONTEXT lines in psql

Back to Table Of Contents

Reader Comments

SQL Coding Standards To Each His Own

Carl T.
In the MS SQL world, most of the examples I've seen use lower case letters as aliases. Unfortunately for me, I've gotten in the habit of copying this convention:

"FROM sometable a
INNER JOIN someothertable b"

What was even weirder was the convention of some Teradata people I worked with from the data warehouse group that used numbers.

Josh B. was right about one thing - there don't appear to be any set conventions established.

Ants Aasma
Possibly you meant the Far Side cartoon brought out by Steve Yegge in
http://steve-yegge.blogspot.com/2006/10/egomania-itself.html

Regina
That wasn't the one I was thinking of, but that is pretty funny.

Pythian Group Blog
Welcome, readers, to the 129th edition of Log Buffer, the weekly review of database blogs. Welcome also to 2009, so fresh it still has that wonderful new year smell. Let's take 'er out on the road and see what she can do.
Starting with Or...

fsilber
Who cares whether everyone on the project makes the same choice between underscores versus camel style?

What is important is that the name describes the content of the variable well enough that the reader need not read huge tracts of code to figure it out.

Regina
I have to agree with you that variable naming is pretty important, but I also think consistency in basic style is important too.

Have you ever worked on a code-base where half the people indent one way and other half indent another? For me it doesn't matter one way or another but seeing 2 different styles confuses new people coming into the project (should I do it this way or that way?) and just reading the code requires a sort of non-sensical context switching (at least for me anyway).

I guess its kind of hard to explain and maybe I'm just more sensitive to those kinds of things.

Baloo
The styling of the comments hurts my eyes.

SQL Coding Standards To Each His Own Part II

alan
In regards to the table.id argument, why not consider semantics? table.id is semantic. table.table_id is just plain redundant and insulting to coding intelligence.

Why do the extra work for no explicit benefit? Note how you guys state, "Field names should be prefixed with their tables." You might as well go on to say, "... and id columns should also be prefixed with their tables." Read that back to yourselves and let your mind wobble. Thanks for helping to push non beneficial "standards" to make my day-to-day tasks more of a pain in the ass.

Regina
Alan,

First of all these are just opinions and part of the point of this article is that standards are not quite as cut and dry as some people think. Josh I'm sure feels strongly about his style just as much as we feel strongly about our style.

The 2 problems I see with id is

1) You always have to alias it in the SELECT because who wants to see id in a multi join output when you can't see the tables that are being joined.

In many cases we need to output the primary key for example if the query is a view to be joined later with other things and if you are going to have to alias to a meaningful name, you might as well do it in the table.

2) Its just distracting to see all fields named the same unless they mean the same thing (yah we can argue an id is an id, but an id is not always an integer so why should it always have the same name and that's a whole other argument as to whether we should always use surrogate keys). first_name is a first_name and always means the same thing okay I get it.

When you are using short aliasing of tables its especially annoying.

PO.id is just not quite as useful to see as PO.post_id. In Josh' model its not quite as bad since he always spells out the table name. So I guess in Josh's model it works if you never need to output the primary key in the query.

RDL
Hi. Just a small nitpick.

Not once did you mention the word "column", but used the word "field" 8 times.

Columns are not fields.

Here is a great explanation:

http://www.developersdex.com/gurus/articles/118.asp

Regards.

Regina
RDL,

I'll have to think about this a bit more. You are probably right. Rereading Celko's paper I could read it a couple of ways.

The way I have always understood it, a column defines the whole set of data of a specific slot (column) in a table. A field defines a specific element of data that lives in a specific row in a specific column.

So when I think of writing SQL statements -- I think of them as fields, because I am not selecting the whole column 0 table (I am only selecting a set of fields that live in a column). But yes I am selecting from columns so perhaps I should call it columns.

So when I am defining a table structure I think columns. When I am selecting data, I think fields.

The other reason I just assume not bother talking about columns is we do a lot of financial consulting and when you start talking columns - the first crazy thought that pops into finance peoples minds is (Ah so a database is a spreadsheet - we have columns too). I just assume not get into that mindset of thinking of databases as spreadsheets even though they both have columns and I have partaken in the guilty pleasure of simulating spreadsheet behavior in databases :)

Fusion Charts and PostgreSQL Part 2: ASP.NET Dashboard

Anton
Hi - please take a look at AnyChart - this flash chart solution will also work for your case.
http://www.anychart.com

Postgres OnLine Journal
In the first part of this series Fusion Charts and PostgreSQL Part 1: Database Analysis of USDA DB in our November/ December 2008 issue, we did
some back-seat analysis of a database we had never seen before and formulated some thoughts of things that woul

---

Fusion Charts and PostgreSQL Part 3: PHP Dashboard

santiago-ve
Interesting article!

may be there will be a 4th part with python example?

On the topic of Cheat Sheets: DZone RefCardz

Josh Berkus
Regina, Leo,

I'm trying to get in touch with you about pgcon. Please e-mail me. Thanks!

POSTGRESQL 8.3 PSQL CHEAT SHEET

psql is located in the bin folder of the PostgreSQL install and PgAdmin III install.

This is psql 8.3.5, the PostgreSQL interactive terminal.

Usage: **psql** [OPTIONS]... [DBNAME [USERNAME]]

General options:

```
  -c COMMAND                                run only single command (SQL or internal) and exit
  -d, --dbname=NAME                         specify database name to connect to (default: "logged in username here")
  -f, --file=FILENAME                       execute commands from file, then exit
  --help                                    show this help, then exit
  -l, --list                                list available databases, then exit
  -v NAME=VALUE                             set psql variable NAME to VALUE
  --version                                 output version information, then exit
  -X                                        do not read startup file (~/.psqlrc)
```

Interactive Console:

```
TYPE:  \copyright                           for distribution terms
       \h for help with SQL commands        for help with SQL commands
       \? for help with psql commands       for help with psql commands
       \g or terminate with semicolon to execute query      or terminate with semicolon to execute query
       \q to quit                           to quit
GENERAL:
  \c[onnect] [DBNAME|- USER|- HOST|- PORT|-]        connect to new database
  \cd [DIR]                                         change the current working directory
  \encoding [ENCODING]                              show or set client encoding
  \h [NAME]                                         help on syntax of SQL commands, * for all commands
  \set [NAME [VALUE]]                               set internal variable, or list all if no parameters
  \timing                                           toggle timing of commands (currently off)
  \unset NAME                                       unset (delete) internal variable
  \prompt [TEXT] NAME                               prompt user to set internal variable
  \! [COMMAND]                                      execute command in shell or start interactive shell

QUERY BUFFER:
  \e [FILE]                                         edit the query buffer (or file) with external editor
  \g [FILE]                                         send query buffer to server (and results to file or |pipe)
  \p                                                show the contents of the query buffer
  \r                                                reset (clear) the query buffer
  \w FILE                                           write query buffer to file

INPUT/OUTPUT:
  \echo [STRING]                                    write string to standard output
  \i FILE                                           execute commands from file
  \o [FILE]                                         send all query results to file or |pipe
  \qecho [STRING]                                   write string to query output stream (see \o)

INFORMATIONAL:
  \d [NAME]                                         describe table, index, sequence, or view
  \d{t|i|s|v|S} [PATTERN] (add "+" for more detail) list tables/indexes/sequences/views/system tables
  \da [PATTERN]                                     list aggregate functions
  \db [PATTERN]                                     list tablespaces (add "+" for more detail)
  \dc [PATTERN]                                     list conversions
  \dC                                               list casts
  \dd [PATTERN]                                     show comment for object
  \dD [PATTERN]                                     list domains
  \df [PATTERN]                                     list functions (add "+" for more detail)
  \dF [PATTERN]                                     list text search configurations (add "+" for more detail)
  \dFd [PATTERN]                                    list text search dictionaries (add "+" for more detail)
  \dFt [PATTERN]                                    list text search templates
  \dFp [PATTERN]                                    list text search parsers (add "+" for more detail)
  \dg [PATTERN]                                     list groups
  \dn [PATTERN]                                     list schemas (add "+" for more detail)
  \do [NAME]                                        list operators
  \dl                                               list large objects, same as \lo_list
  \dp [PATTERN]                                     list table, view, and sequence access privileges
  \dT [PATTERN]                                     list data types (add "+" for more detail)
  \du [PATTERN]                                     list users
  \l                                                list all databases (add "+" for more detail)
  \z [PATTERN]                                      list table, view, and sequence access privileges (same as \dp)

FORMATTING
  \a                                                toggle between unaligned and aligned output mode
  \C [STRING]                                       set table title, or unset if none
  \f [STRING]                                       show or set field separator for unaligned query output
  \H                                                toggle HTML output mode (currently off)
  \pset NAME [VALUE]                                set table output option
                                                      (NAME := {format|border|expanded|fieldsep|footer|null|numericlocale|recordsep
                                                      |tuples_only|title|tableattr|pager})
  \t                                                show only rows (currently off)
  \T [STRING]                                       set HTML <table> tag attributes, or unset if none
  \x                                                toggle expanded output (currently off)

COPY, LARGE OBJECT
  \copy ...                                         perform SQL COPY with data stream to the client host
  \lo_export LOBOID FILE                            LOBOID FILE
  \lo_import FILE [COMMENT]                         FILE [COMMENT]
  \lo_list
  \lo_unlink LOBOID                                 large object operations
```

Connection options:

```
  -h, --host=HOSTNAME      database server host or socket directory
  -p, --port=PORT          database server port number
  -U, --username=NAME      connect as specified database user
  -W, --password           force password prompt (should happen automatically)
  -e, --exit-on-error      exit on error, default is to continue
  -d DBNAME                some database
```

psql automated shell examples

restore w hole server

```
psql --host=localhost --username=someuser -f /path/to/pgdumpall.sql
```

Run an sql batch script against a database

```
psql -h localhost -U someuser -d somedb -f /path/to/somefile.sql
```

Run an sql batch script against a database and send output to file

```
psql -h localhost -U someuser -d somedb -f /path/to/scriptfile.sql  -o /path/to/outputfile.txt
```

Run a single statement against a db

```
psql -U postgres -d pagila -c "CREATE TABLE test(some_id serial PRIMARY KEY, some_text text);"
```

Output data in html format

```
psql -h someserver -p 5432 -U someuser -d somedb -H -c "SELECT * FROM sometable" -o mydata.html
```

psql Interactive mode

Launch Interactive session

```
psql -h localhost -U postgres -d somedb
```

View help for SELECT * LIMIT

```
\h  SELECT * LIMIT
```

List all tables in db w ith descriptions

```
\dt+
```

List all tables in db w ith **s** in the name

```
\dt *s*
```

Cancel out of MORE screen

```
:q
```