Postgres OnLine Journal: May / June 2008

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/

What can PostgreSQL learn from MySQL

There has been a lot of talk lately about PostgreSQL and what MySQL can learn from the PostgreSQL clan. We would like to look at the reverse of that. This article is a bit of a complement to Joshua Drake's What MySQL (and really, Sun) can learn from PostgreSQL.

First of all a lot of staunch advocates of PostgreSQL wonder what exactly is it that MySQLers see in that beast of a database or as Martin Mickos likes to call it The Ferrari of databases?

For example, as Magnus Hagander pointed out in A new MySQL gotcha and we pointed out in SQL Math Idiosyncracies, MySQL's casting behavior is shall we say *odd*. I won't even recount our frustrations with using their ODBC driver with the MySQL 5.0 incarnations, but that could be ignorance on our part. Zack Urlocker of Sun has espoused, the upcoming MySQL 5.1 will have no bugs so perhaps this along with other bug complaints is a moot point.

So why do people choose MySQL time and time again over PostgreSQL and why is PostgreSQL sometimes a hard sell? Some of what we are going to say is a bit tongue-in-cheek so please don't take offense.

MySQL is pervasive and ubiquitous

Being pervasive and ubiquitous is a huge selling point.

- It means you are already in the **In crowd**.
- It means System Admins already feel comfortable with you.
- It means common folk have heard of you.
- It means people feel comfortable enough with you that they are willing to build an ecosystem around you and ecosystems make you more ubiquitous and pervasive.

How did MySQL get to that point and what tricks can PostgreSQL borrow from that experience?

1. It ran natively on Windows before PostgreSQL did and Windows is a huge user-base especially for beta-testing. PostgreSQL has caught up there and is beginning to see the fruits of that labor.
2. Everywhere you look for Mac etc. there are already pre-compiled binaries. Again PostgreSQL is catching up there with the Yum repository, pre-compiled windows binaries and valiant efforts of people making binaries for Mac, Debian, SUSE available. More work needs to be done there to insure binaries/rpms for latest releases are available for most of the OSes PostgreSQL supports and popular add-ons.
3. MySQL is an easier install and easier to upgrade. As Joshua Drake pointed out in the above article, you can do in-place upgrades with MySQL, but really can't with PostgreSQL. Again PostgreSQL needs more work there. This is a huge plus for many ISPs and we know when many projects start out, they can't afford dedicated boxes so they rely on their ISPs to have these things already installed.
4. You can run MySQL on a USB stick (or at least for windows) easily without having to reboot your pc. Portability is huge. Take a look at Server2GO its quite a slick run on USB that allows you to run Apache, MySQL, PHP and edit data on USB as well. It is all nicely packaged and allows a brain-dead user to point and click to use it. We need that for PostgreSQL and it seems there are efforts going on in that area which is good. LiveCD comes close but doesn't quite make the cut.
5. A lot of people who used older versions of PostgreSQL have a bad taste in their mouth. People need to be reminded that PostgreSQL 6/7 of the past is vastly different from the PostgreSQL 8 series. You would think MySQL would have similar issues, but they seem to have less of that. That could be because official install repositories/ISPs have more updated versions of MySQL than PostgreSQL. Also MySQL has always been a relatively easy install and less threatening looking database for good or bad.
6. Of course there is safety in crowds. People don't like to stand-alone because its okay to be wrong if everyone is wrong, but really lonely if you stand wrongly alone. MySQL's dominance makes it more dominant except in situations such as Open Source GIS where PostgreSQL/PostGIS is still a clear winner over MySQL and the dominant database for that growing niche market.

PostgreSQL has a lot of selling points that MySQL lacks, but this is a talk about why MySQL is so great, so we'll save that talk for another day.

PostgreSQL people look like a gang of geeks and Martin Mickos looks polished

PostgreSQL clan is heavily loaded with geeks and the first thing that comes to at least my mind is geek when trying to summarize the generic face of a PostgreSQL user. This is a good thing from a development standpoint and for attracting great developers, but is bad when a large constituency you are trying to sell to are not geeks and they perceive your database as a *thing that only a geek can use effectively*. While we would like to think otherwise and at least think that its in vogue to be a geek, most people are not geeks. We need to sell more to the otherside of the fence - because like it or not - they are often the ones that make the decisions.

Martin Mickos is riding on what Leo likes to call *People who look alike play together* rule of social behavior. He is forced to wear a suit like most money-decision making people and looks comfortable in that outfit. He looks like a CEO. He might be a geek inside but he covers it well, and he gives you the polished feel that he can endure the torture of a CEO recounting his golf game or bragging about his greatness. Why is this important? Isn't it all an act? Yes **it is an act**, but it sends the message - *I know how to act right - when to open my mouth and when to keep it shut*. This is incredibly comforting to a generic CEO, CFO or CIO type. Yes it does often come down to *he got the contract* not because his technology was better but because he had a better suit on, got drunk on his own cool-aid and could finish a sentence without saying "Uhmm".

Martin Mickos balances Michael "Monty" Widenius disposition well, whereas a lot of PostgreSQL folk look and sound like Monty with very few balancing Martin's to level the see-saw.

We don't have a very strong prominent non-geek face to offset this imbalance. We need a *CEO friendly* imposter. I would say the closest are some of the EnterpriseDb folk, but EnterpriseDb is a commercial offering so that can only get the clan so far.

It would help if core PostgreSQL had a prim and proper looking figure or a comedian like Steve Jobs, Larry Ellison, Marc Fleury (founder of JBOSS), or geekesses in disguise like Kim Polese (the woman behind SpikeSource, Marimba, and Java) who exudes subtle persuasion abilities or a quietly confident woman such as Diane Greene, CEO and CoFounder of VMWare. Some PostgreSQL folks come to mind that can be molded into some of those type figures, but they still have that geek suit on (the T-shirt and the long hair that appeals to geeky sysadmins who are not always allowed to make decisions). Steve Jobs is a geek, but one with style. His *brat geek kid who never grew up and with an obsession with clean interfaces that look good* is great marketing. Similarly Larry Ellison's farcical obsession with wardrobe, racing, womanizing and destroying competitors is equally entertaining. Marc Fleury seems to be a compromise between the Steve and Larry personalities and some others mixed in. He has an air of mystique, but he knows when to behave as well.

Kim Polese perfectly complemented the advanced geekism of James Gosling who wanted to call Java a stupid name like *Oak*.

Diane Greene is an equally neat woman. She is not pushy, doesn't quite look like everyone else, but enough to make one feel comfortable, and she has that look that just screams - *I can lead an army of ships into war.* She balances out the more behind the scenes personality of her husband, Mendel Rosenblum, co-founder and chief scientist of VMWare.

Bill Gates (please don't throw stones at me for listing Bill Gates as one of those we respect). We for the most part like Bill Gate's and admire him. Both he and Diane Greene are what I shall refer to as anchors. Another rule of social behavior is that *people like to hang around anchors*. Anchors are those people who have a horned or natural instinct for feeling out what makes their audience comfortable and becoming it. They don't quite dress like their audience, but are not vastly different enough to be scary. They stand out just enough to look different, but not too much. They seem as content being by themselves as they do with being with other people. You only need to look at video of Bill Gates last full day to see what an anchor he is.

A lot of PostgreSQL folk, from a cursory observation, are suffering from some form of geekism and while it is not something that is easy to cure or should be cured - we need to offset it more with at least people who can hide their disorder or highlight those who have it to such a ridiculous level that it serves as a natural parody.

Symptoms of this disorder include

- Preferring to discuss merits of technology approaches and insisting on silence over discussion of the daily weather and other idle banter.
- Telling very obscure jokes and laughing at your own jokes.
- Saying what is on your mind and making criticisms in a matter of fact way without thinking about how it affects other's feelings. Basically lacking social situational awareness.
  e.g.
  That was a really idiotic thing you did. You better have backed up your data or you are probably screwed.

  instead of the less harsh

  The best way I think to handle that would have been .... Did you make a backup?

  Keep in mind quite a few people are obsessed with what other people think and feel and spend a lot of time tailoring their words, gestures, and dress codes to that. They get quickly insulted when they come across someone who seems fairly oblivious to that concern unless that someone can add a lot of comedy to insult.
- Unable to feel or at least look comfortable in anything else but a T-Shirt.
- Uneasiness in non-geek crowds.

MySQL has a pluggable storage architecture

Now this one we wouldn't suggest bothering with, although we should probably dissect this thoroughly to understand better what value people see in this and what substitutions can be made and pointed out that PostgreSQL offers. Granted the camp is divided here. Personally the fact that some things are supported on one MySQL storage engine and not another MySQL storage engine makes MySQL somewhat unpleasant to work with. As Tom mentioned in his The Value of MySQL Storage Engines I suppose there is some charm that we just can't appreciate.

Back to Table Of Contents What can PostgreSQL learn from MySQL Reader Comments

PostgreSQL 8.4 goodies in store

The PostgreSQL 8.4 planned release is March 1, 2009 and is outlined in the PostgreSQL 8.4 Development plan. It has just passed its May 2008 commit fest milestone and is currently in its July 2008 Commit Fest. Lots of PostgreSQL Planet bloggers have started showcasing some of the new features in store. We will briefly list our favorite planned and already committed patches.

Things that seem likely

- **Built-in simple replication** - As Bruce Momjian mentioned in his blog PostgreSQL 8.4 is planning to have built-in log shipping master/slave replication. More details of that in Core Team on Replication thread. Mostly we are excited about this because it will be one less thing MySQLers and SQL Servers can hold over why they will never consider PostgreSQL because it has no built-in replication.
- **RETURN QUERY EXECUTE** This one already made the commit. While it seems small to some, we are very excited about the RETURN QUERY working with EXECUTE. That was one of the things that made RETURN QUERY of limited use in 8.3. Thanks to Pavel Stehule for that one.
- **Function Stats** - Hubert Lubaczewski has been highlighting a lot of new nifty features. Many psql enhancements. Function stats is our favorite of his list so far and is detailed in Waiting for 84 Function Stats
- **With RECURSIVE** This is another one that hasn't quite made it in yet but is in the July 2008 commit fest. With Recursive. For people familiar with SQL Server 2005 - this is similar in style to SQL Server Common Table Expressions (CTE). Not sure the equivalent in Oracle - I presume Corresponding by CONNECT BY or something along that line. IBM DB2 also has something called common table expressions, but not quite sure how that works or if it is the same.
- **ANSI SQL 2003: table function support** This one hasn't made it in yet but is under review. For those who use set returning functions extensively and are frustrated by having to create a type for each set returning function you create, you will appreciate this one. ANSI SQL 2003: table function support. Thanks go to Pavel again for this one.

Things not so likely

Now there are two other things we are looking forward to that sadly we fear may not make it into 8.4, but with the cycles of PostgreSQL we'll probably only need to wait an additional year as opposed to 3-5 years (with other DB product release cycles) to see these.

- **Windowing functions** - for things like moving averages, cumulative sums and other stats across large amounts of data this is important. Lots of people have talked about this and its one of those things that sticks out like a sore thumb for high-end users.
- **MERGE for PostgreSQL 8.4** as noted here - arguably this is just a check-off item for us because once SQL Server 2008 comes out, PostgreSQL will be the only database we commonly work with that doesn't have this functionality.

Back to Table Of Contents PostgreSQL 8.4 goodies in store Reader Comments

Setting up PostgreSQL as a Linked Server in Microsoft SQL Server 64-bit *Intermediate*

We would like to thank Jeff Crumbley of IILogistics for providing many of these steps and informing us that Microsoft has finally released a 64-bit OLEDB for ODBC driver.

For those who have not experienced the torture of this situation - let me start with a little background. First if you are running SQL Server 2005 32-bit and wished to create a linked server to a PostgreSQL server, everything is hunky dory. If however you had a SQL Server 2005 64-bit server, you ran into 2 very annoying obstacles.

1. **Obstacle 1:** There for a long-time was no 64-bit ODBC driver nor native driver for PostgreSQL. This obstacle was somewhat alleviated when Fuurin Kazanbai made experimental compiled 64-bit PostgreSQL ODBC drivers available which work for AMD and Intel based processors.
2. **Obstacle 2:** All looked good in the world until you tried this in SQL Server 2005 64-bit and low and behold - you needed a 64-bit OLEDB provider for ODBC to use it in SQL Server 2005 64-bit. Yes we waited patiently for years for this piece to be available. We still love you Microsoft. Then as Jeff Crumbley pointed out - Microsoft released an OLEDB 64-bit provider for ODBC in early April 2008.

Below are the steps to get a PostgreSQL linked server working in SQL Server 2005 64-bit.

1. Run WindowsServer2003.WindowsXP-KB948459-v2-x64-ENU.exe - (Available as of 4/4/2008 from: http://www.microsoft.com/downloads/details.aspx?FamilyID=000364db-5e8b-44a8-b9be-ca44d18b059b&displaylang=en)
2. Make the folder C:\Program Files\PostgreSQL\8.1\AMD64bin (you can try the 8.3 if you are running that) and place the dlls from psqlodbc_AMD64 available from http://www.geocities.jp/inocchichichi/psqlodbc/index.html
3. Run the psqlodbcwAMD64.reg file
4. Create a System DSN in the 64-bit Data Source (ODBC) - alternatively you can skip this and use and embedded file DSN in SQL Server 2005 that we will outline in the next step.
5. Create a Linked Server in SQL Server - below is a sample script that creates a PostgreSQL Linked Server in Microsoft SQL Server 2005 64-bit.

```
EXEC master.dbo.sp_addlinkedserver @server = N'NAMEOFLINKEDSERVERHERE', @srvproduct=N'PostgreSQL
AMD64A',
   @provider=N'MSDASQL', @provstr=N'Driver=PostgreSQL AMD64A;uid=pguser;Server=pghost;
database=pgdatabase;pwd=somepassword'
 /* For security reasons the linked server remote logins password is changed with ######## */
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname=N'NAMEOFLINKEDSERVERHERE',
      @useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL
```

   After that you should see the linked server in SQL Server 2005 Management ->Server Objects ->Linked Server and from there you can fiddle further with the settings. You should also be able to expand the PostgreSQL linked server and see the tables and views.
6. To test out the linked server - you can run the sample query below in SQL Server:

```
SELECT *
   FROM
      OpenQuery(NAMEOFLINKEDSERVERHERE,
         'SELECT * From information_schema.tables')
```

Keep in mind that the PostgreSQL 64-bit ODBC is marked as experimental, but we have had good success with it on an Intel processor based 64-bit Windows 2003 running SQL Server 2005 64-bit.

Back to Table Of Contents

How to calculate Running Totals and Sums in SQL *Intermediate*

People have asked us how to calculate running totals a number of times; not a lot but enough that we feel we should document the general technique. This approach is fairly ANSI-SQL standard and involves using SELF JOINS. In a later article we shall describe how to calculate moving averages which follows a similar technique but with some extra twists.

Note that the below examples can also be done with a correlated sub-select in the SELECT clause and in some cases that sometimes works better. Perhaps we shall show that approach in a later issue. We tend to prefer the look of the SELF JOIN though and in practice it is generally more efficient since its easier for planners to optimize and doesn't always result in a nested loop strategy. Just feels a little cleaner and if you are totaling a lot of columns (e.g number of items, products) etc, much more efficient.

Question 1: Calculate running total for a customer by order but don't include in the total the current order amount?

Solution 1:

This is one of the cases where the use of a SELF JOIN comes in handy. For this particular example we shall assume we have a table of orders and for each order, we would like to know for that given customer the total price of goods they have purchased prior to date of order. For sake of argument we shall assume the order_datetime has full timestamp of order so it is a fairly rare or non-existent situation that a customer will have 2 orders with the same timestamp.

```
SELECT n.customer_id, n.order_id, n.order_total,
    SUM(o.order_total) As running_total
FROM orders n LEFT JOIN orders o
    ON (o.customer_id = n.customer_id
            AND n.order_datetime > o.order_datetime)
GROUP BY n.customer_id, n.order_datetime, n.order_id
ORDER BY n.customer_id, n.order_datetime, n.order_id;
```

Question 2: How to calculate running total for each day?

In this case we want to know the total profit of the company for each day and running total for each day including current day. In this case we can use the more efficient INNER JOIN since we know that the prior and including current will have the current order date as well. Its debateable if we need the ORDER BY in the subselect. That is mostly there to try to force the planner to materialize the subselect which would tend to be faster. The below query should work fine in MySQL 5 and above as well.

NOTE: if you tried such a thing in Microsoft SQL Server 2005 and below the below would not work for 2 reasons.

1. SQL Server 2005 and lower does not like order bys in subselects so you will need to remove that OR use the (SELECT TOP 100 PERCENT * FROM orders ORDER BY order_datetime) hack. Anyrate forcing an order would probably not change the plan in SQL Server 2005. Haven't tried on beta of SQL Server 2008
2. SQL Server 2005 and below have no Date data type. Date and Datetime with Timestamp were introduced in SQL Server 2008 so the CAST part will probably work in SQL Server 2008. In prior versions there is some messy code you have to write involving subtraction of time. This is a PostgreSQL blog so we will not go into that. But here is a good link that covers that messiness - http://weblogs.sqlteam.com/jeffs/archive/2007/01/02/56079.aspx.

To give SQL Server 2005 and above due credit - this would be done more efficiently using the windowing functions introduced in SQL Server 2005 and above and so would the above example.

Solution 2:

```
SELECT n.order_date, n.order_total, SUM(o.order_total) As running_total
FROM (SELECT CAST(order_datetime As date) As order_date,
            SUM(order_total) As order_total
        FROM orders
            GROUP BY CAST(order_datetime As date)
        ORDER BY CAST(order_datetime As date)) n INNER JOIN
     (SELECT CAST(order_datetime As date) As order_date,
        SUM(order_total) As order_total
     FROM orders
        GROUP BY CAST(order_datetime As date)
        ORDER BY CAST(order_datetime As date)) o
        ON (n.order_date >= o.order_date)
GROUP BY n.order_date
ORDER BY n.order_date;
```

Back to Table Of Contents How to calculate Running Totals and Sums in SQL Reader Comments

Cross Compare of SQL Server, MySQL, and PostgreSQL

Comparison of Microsoft SQL Server 2005, MySQL 5, and PostgreSQL 8.3

The below is by no means an exhaustive comparison of these 3 databases and functionality may not be necessarily ordered in order of importance. These are just our experiences with using these 3 databases. These are the databases we use most often. If we left your favorite database out - please don't take offense. Firebird for one has some neat features such as its small footprint and extensive SQL support, but we have not explored that Db.

People ask us time and time again what's the difference why should you care which database you use. We will try to be very fair in our comparison. We will show equally how PostgreSQL sucks compared to the others. These are the items we most care about or think others most care about. There are numerous other differences if you get deep into the trenches of each.

For those looking to compare MySQL and PostgreSQL you may want to also check out http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL

If you really want to get into the guts of a relational database and the various parts that make it up and how the various databases differentiate in their implementations, we suggest reading Architecture of a Database System by Joseph M. Hellerstein, Michael Stonebraker, and James Hamilton. Architecture of a Database System focuses mostly on Oracle, DB2, and SQL Server but does provide some insight into MySQL and PostgreSQL.

People have pointed out things we omitted and things we got wrong, so we have corrected some of these and will be slowly adding updates.

A lot of people have been making comments on the related Reddit Cross Compare of SQL Server, MySQL, and PostgreSQL thread in addition to this blog. I guess it shows people are really passionate about their databases. Lots of good discussion.

Feature	Microsoft SQL Server 2005	MySQL 5	PostgreSQL 8.3
OS - Why is this important? Why would you even dream of not running on Windows? If you decide one day that Microsoft is not your best friend in the whole wide world, you can ditch them or at least on your DB Server (could that ever happen?). On a side note, Microsoft can't compete with Oracle on Linux/Unix anyway. If Microsoft has a non-Microsoft DB running on a customer's box, I wonder which database they would prefer - Oracle, IBM DB2, Sun MySQL or PostgreSQL?	Windows XP, Windows 2000+	Windows (even down to 98?), Linux, Unix, Mac	Windows 2000+, Linux, Unix, Mac
Licensing	Commercial - Closed Source, Various levels of features based on version, Free Crippleware	GPL Open Source, Commercial. Here is an interesting blog entry on the subject MySQL free software but not Open Source. The comments are actually much more informative than the article itself.	BSD Open Source

Install/Maintenance Process	Hardest most time-consuming and biggest hog of resources of the 3 even when its not doing anything	Easiest	Medium
Drivers already installed on Windows	Yes - when you have a windows shop this is huge especially when you are not allowed to install stuff on client desktops and you need to integrate seamlessly with desktop apps. This is why using SQL Server Linked Server to get at yummy features of PostgreSQL comes in handy.	No	No
ODBC, JDBC, ADO.NET drivers available	Yes	Yes	Yes
Read-Only Views	Yes	Yes	Yes
Open Source products available for it	Few except CodePlex/.NET	Many	Few but ramping up and in PHP more than SQL Server
Commercial	Many?	Moderate	Few but ramping up
Updateable Views	Yes - even for 2 table views will automatically make them updateable if they have keys and update does not involve more than one table. You can write instead of triggers against more complex views to make them updateable	Yes - Single one table views are automatically updateable, some 2 table views are updateable if they don't have left joins and don't involve update of more than one table. If you have more complex views you want to make updateable - good riddance - no support for triggers or rules on views.	Yes, but not automatic. You have to write rules against views to make them updateable but can make very complicated views updateable as a result
Materialized/Indexable Views	Yes but varies slightly depending on if you are running SQL Express, Workgroup, Standard, Enterprise and numerous restrictions on your views that makes it of limited use	No?	No, but there are I think 2 contrib modules e.g. matviews that are simple and basically rebuild the materialized view
Can add columns and change names, data types of views without dropping	Yes	Yes	No - and extremely annoying if you have views that depend on other views.
Can drop tables, (drop, change size, data type of columns), and views used in views - this is a arguably a misfeature but sometimes it comes in handy when you are an EXPERT user :)	Yes - yikes! (but if you schema bind your tables and views, you can not drop dependent objects)	Yes - yikes!	No
Graphical View Designer (e.g. you can see tables and select fields drag lines to do joins) included no additional charge	Yes via SQL Management Studio and Express?	No	No
Computed Columns	Yes - but we still like using Views more except when we really need the computed column indexed and often we just do triggers. Computed columns are of very limited use since they can't hold roll-ups.	No - but looks like its slated for future release	No - but PostgreSQL has functional indexes so just use a view.

Functional Indexes - indexes based on a function	No - but you can create a computed column and create an index on it	No	Yes
Partial Indexes - e.g. you want to create a unique index but only consider non-null values	No - but as pointed out you can achieve similar results with an indexed view	No	Yes!
ACID compliance - do I dear say this is sometimes over-rated - not all data is created equal and sometimes bulk-insert speed is more important than ACID	Yes	Some storage engines e.g. InnoDB and not MyISAM	Yes
Foreign Key - Cascade Update/Delete	Yes	InnoDB and not MyISAM	Yes
Multi Row value insert	No - but SQL Server 2008 will have it	Yes	Yes
UPSERT logic - where you can simultaneously insert if missing and update if present	No - but SQL Server 2008 will have it via MERGE UPDATE	Yes - via INSERT IGNORE and REPLACE	No
Replication - haven't used much except for SQL Server so this is mostly hear-say	Yes - all sorts - log shipping, mirroring, snapshot, transactional and merge etc. and can even have non-SQL Server windows-based subscribers. Its still a bear to get working the way you want it and makes making structural changes difficult. Built-In	Yes - including master-master (built-in) See comments below and from numerous reports a big selling point of MySQL.	Yes but from reports seems to be the least polished of the bunch, although numerours third-party options to choose from that are both free and non-free. PostgreSQL 8.4 or higher is slated to have built-in replication - see core team notes - http://archives.postgresql.org/pgsql-hackers/2008-05/msg00913.php
Can program stored procs/functions in multiple languages	Yes - any language that complies with CLR -e.g VB.Net, C#, IronPython - but you need to compile into a dll first - so kind of cheating since you can't see python code right in your db. Upside you don't need IronPython etc. hosted on server, but you can't use the rich environment of Python either and need to have all dependent libraries explictly loaded into SQL Server GAC, a real PITA if you have lots of these dependencies.	No (except C and Pl/SQL) but they are working on it	Yes - PostgreSQL just does it the cool way - we like having our code right there where we can see what it is doing. Downside server must host the language environment.
Can define custom aggregate functions	Yes - any .NET language, but not TRANSACT SQL. Why is Transact-SQL thrown out to dust like this?	Yes but only in C as UDF	Yes - any PL language and built-in C, SQL, PLPgSQL.
Triggers	Yes	Yes	Yes

Table Partitioning	Yes - only Enterprise version - functional, range	Yes? (only applied to NDB cluster), 5.1 will be vastly improved	via Table Inheritance, Constraint Exclusion, RULES and Triggers - basically RANGE. Issues with using foreign-key constraints with inherited tables (plans to improve for 8.4?)
Can write Set/Table returning functions that can be used in FROM clause	Yes	No	Yes
Support creation of functions - e.g. CREATE FUNCTION	Yes	Yes	Yes
Support creation of stored procedures - e.g. CREATE PROCEDURE	Yes	Yes	Sort-Of - CREATE FUNCTION serves the same need
Dynamic and action SQL in functions	No - but you can in Stored procedures but you can't call stored procs from SELECT statements so much more limiting than PostgreSQL	No, but can in Stored procedures which aren't callable from SELECT statements so more limiting than PostgreSQL	Yes! - you can do really cool things with action functions in SELECT statements
Graphical Explain Tool - no additional charge	Yes - SQL Management Studio/Express	No	Yes - PgAdmin III
Job Scheduling Agent controllable from DB Manager client, for running batch sql and shell jobs - no additional charge (not CronTab)	Yes - SQL Agent (not for Express)	No - but upcoming 5.1 will	Yes - PgAgent
Access tables from other databases on same server	Yes - server.db.schema. table, can even access disparate data sources via linked server or open query	Yes - db.table, but not easily across servers	Sort of - via Dblink, but much less elegant than MSSQL and MySQL way and much less efficient. Can also access disparate data sources via DBI Link
Case-Insensitivity - e.g. LIKE 'abc%' and LIKE 'ABC%' mean the same thing	By default its not case sensitive, but can change this down to the column level.	It is not case-sensitive by default	By default is case-sensitive and a pain to make it not so. Sure you can do ILIKE, but its not indexable and just not the same since an ODBC driver doesn't expose it and is not ANSI compliant. This makes it annoying in environments like MS Access, PHP Gallery where MySQL/MSSQL Server default case insensitivity is more user expected.
Date Time support	SQL Server 2005 and below are just really lame. Only have Datetime (no support of timezone or just plain DATE). SQL Server 2008 will have these.	Less lame, has Date and DateTime but none with Timezone	Best - Date, TimeStamp and TimeStamp with Timezone (not to be confused with MySQL's timestamp which autoupdates or SQL Server's deprecated timestamp which is a binary)

Authentication	Standard Db security and NT /Active Directory Authentication	Standard Db with table-driven IP like security	Extensive - standard, LDAP, SSPI (can tie in with Active Directory if running on NT server, but still not quite as nice as SQL Server seamless integration), PAM, trust by IP, etc.
DISTINCT ON	No	No	Yes
WITH ROLLUP	Yes	Yes	No
WITH CUBE	Yes	No	No
Windowing Functions OVER..PARTITION BY	Yes	No	No
COUNT(DISTINCT), AGGREGATE(DISTINCT)	Yes	Yes	Yes
OGC Spatial Support - for the My dad is better than your dad fight in the GIS world between SQL Server and PostgreSQL/ PostGIS check out A look at PostgreSQL and ArcSDE	No - well Open source MSSQL Spatial add-on has basic support but not as good as PostGIS. Numerous commercial vendors provide spatial extensions for SQL Server 2005 - e.g. Manifold.net, MapDotNet, ESRI ArcSDE come to mind. SQL Server 2008 will have built-in, and geodetic but a lot of functions that PostGIS has will be missing.	Yes - MBR mostly and spatial indexes only work under MyISAM. Limited spatial functions. Some commercial (MapDotNet, Manifold.net), Open source GIS tools gaining steam but still more behind PostGIS.	Yes - PostGIS is great and lots of spatial functions and fairly efficient indexing and lots of open source and commercial support and upcoming ESRI ArcGIS 9.3 supports it too.
Schemas	Yes	No	Yes
CROSS APPLY	Yes	No	No but can for the most part simulate by putting set returning C/SQL functions in SELECT clause and wrapping more complex functions in an SQL function body.
LIMIT .. OFFSET	No - has TOP and ROW_NUMBER() which is much more cumbersome to use	Yes	Yes
Advanced Database Tuning Wizard	Yes - SQL Management Studio recommends indexes to put in etc. Very sweet. NOT available for Express or Workgroup.	No	No
Maintenance Plan Wizard	Yes via SQL Management Studio - Workgroup and above. Very sweet. Will walk you thru creating backup plan, reindexing plan, error checking and schedule these for you via SQL Agent	No	No
Pluggable Storage Engine	No	Yes	No
Correlated Subqueries	Yes	Yes	Yes
FullText Engine - all 3 have it, but we don't feel right comparing since we haven't used each enough to make an authoritative comparison. Its annoying there is no set standard for doing Full Text SQL queries	Yes	Yes	Yes

Sequences /Auto Number	Yes - via IDENTITY property of int field	Yes - via AUTO_INCREMENT of int field	Yes - via serial data type or defaulting to next Sequence of existing sequence object - this is better than MySQL and SQL Server simple auto_increment feature. The reason it is better is that you can use the same sequence object for multiple tables and you can have more than one per table. In the past PostgreSQL sequence was a pain but now you just create it with data type serial if you want it to behave like SQL Server and MySQL and it will automatically drop the sequence if you drop the table it is bound to.

Back to Table Of Contents Cross Compare of SQL Server, MySQL, and PostgreSQL Reader Comments

Choosing the right Database Procedural Language PL *Beginner*

One of the great selling points of PostgreSQL is its pluggable PL language architecture. MySQL is known for its pluggable storage and PostgreSQL is known for its pluggable PL language architecture. From Monty's notes on slide 12 looks like MySQL may be working on a pluggable PL language architecture of their own. The most common of these languages are the all-purpose languages SQL and C (these are built-in and not really PLs like the others, but we'll throw them in there), PLPgSQL which is also built-in but not always enabled, PL/Perl, PL/Python, and the domain specific languages PL/R, PL/SH and gaining popularity Skype released PL/Proxy. There are others in the family such as PL/Tcl, PL/PHP, PL/Ruby, PL/Scheme (a dialect of Lisp), PL/Java, PL/Lua and PL/LOLCode (for kicks and as a reference implementation. Think of LOLCode as PostgreSQL Pluggable PL equivalent of MySQL's BLACK HOLE storage engine.) .

The other interesting thing about the PostgreSQL PL language architecture is that it is a fairly thin wrapper around these languages. This means the kind of code you write in those languages is pretty much what you would write if you were doing general programming in those languages minus some spi calls. Since the handler is a just a thin wrapper around the environment, the language environment must be installed on the database server before you can use the PL language handler. This means you can have these functions utilized in your SQL statements and you can write in a language you feel comfortable with if you can get the darn PL compiled for your environment or someone has already kindly compiled it for your environment or that it is even compilable for your environment. The pluggable PL architecture means you can write a PL Handler for your favorite language or invent your own language that you can run in the database. In the end the barrier between code,data, and semantic constructs is more of a constraint imposed by compilers. If you have any doubts about the above statement, you need only look at some javascript injection attacks to bring the statement home. One of my fantasies is developing a language that morphs itself, that utilizes the database as its morphing engine and its OS and that breaks the illusion of data being data, code being code, and lacks rigid semantics. Of the languages we have worked with, SmallTalk comes closest to a language that satisfies these ideals and Lisp to a much lesser extent. Lisp lacked the semantic elegance of SmallTalk among other things.

Most people are used to having their procedural language push their data around. PL code living in PostgreSQL allows your data to push your procedural code around in a set-based way. This is a simple but pretty powerful feature since data is in general more fluid than code. For interpretated/just-in time compiled languages it can live in the database, for compiled it has to call compiled functions.

Now I shall stop here and say there are consequences to a thin wrapper that are both good and bad.

1. Good/Bad - you are writing code you are used to. This is good because it makes people just getting used to relational database concepts feel at home. This is bad because it gives one the false confidence that the Romans will be happy when you impose your cultural bad habits on their perfect society. It is great to bring new ideas into the database, but try not to destroy the sanctity of the database by forgetting that **you are in a database**. Similar things have been said by DB programmers when SQL Server 2005 introduced .NET code in the database and you had all these reckless programmers doing things in .NET code that would have been more efficient in Transact-SQL. Just because you can do it doesn't mean you should. More on that later.
2. Good - you can leverage all the goody libraries in your language of choice that others have written or you have written by calling the libraries from your database. With some caveats e.g. markings as safe and unsafe.
3. Bad - PostgreSQL is doing a context switch to push the code into the environment your code is comfortable in. Generally the bigger and more complex the environment the more context switching that is happening.
4. Bad - In order to manipulate data, except for the built in languages SQL, PlPgSQL, and C, PostgreSQL functions generally need to push the data into the language's environment and pull it out. This makes most languages somewhat suboptimal for set returning functions or functions that consume a lot of data.

As we mentioned in a prior article Trojan SQL Function Hack - A PL Lemma in Disguise not all languages are created equal as far as PostgreSQL is concerned and PostgreSQL has its favorites. Just as annoying as the MySQL storage engine idiosyncracies where you can have foreign keys in one storage engine and they are ignored in another, similar can be said with PL languages in PostgreSQL - they all handle sets differently and set returning functions are easier to write in some PL's than in others. Not to mention each programming environment has certain idiosyncracies of its own which make this useful yet still *Leaky abstraction* apparent.

Even if PostgreSQL did not have its favorites, one must keep in mind that languages are designed for a particular reason. They are designed to satisfy a particular language designers philosophies and goals. This means that Perl is optimized for the certain kinds of problems that Larry Wall liked to solve (e.g. string manipulation) and to solve them the way that Larry thought was fitting. Similarly R, S, S-Plus were designed for scientific, statistical processing, graphing. R takes some effort to get used to its terminology of data frames and factors and its way of pushing data into arrays and defining functions, but its well-worth it for what it does. Generally speaking the PL languages are not optimized for pushing data in a SET-oriented way and if you try to use them for something they were not really designed well for, you may feel comfortable but your database will suffer for your illusion of comfort. This false comfort leads people to write otherwise simple SET code in PlPython when it could have been done more efficiently and simply in PostgreSQL SQL function language or PLPgSQL language. Some people further like to encapsulate things in functions that shouldn't be encapsulated in functions in the first place because it gives them a false sense of comfort to shove stuff that they couldn't figure out how to write in a set-based way into a loop-di-loop blackbox. **It may be amusing to write needlessly complicated code and kill cockroaches with hammers, but it is not a terribly efficient way of occupying your time.** In fact the most impressive programmers are just clear thinkers. The real geniuses in programming are those who can restate an unsolvable problem into a solvable one or don't get caught up in the mob thinking that causes groups of people to simultaneously come up with the wonderful idea of solving the same wrong problem.

General rule of thumb when deciding which language to program a particular functionality

- Can you do the same thing in plain old SQL language? If you can you should. SQL functions are generally inlined which makes them more efficient from a planner perspective
- Most people don't feel comfortable writing in C - so you may want to throw this out of the equation although it is close in priviledge to SQL functions and for intense processor functions more efficient
- Does the function require some intensive calculation or functionality that doesn't exist in PostgreSQL or that is just faster to write and process in PLPerl, PL/R or that these languages have built-in already etc.? Here you probably want to do some benchmarks to be sure to make sure the lose in context switching is less than the gain in efficiency.
- Will you have to run this database in an environment that doesn't support your poison of choice?
- Do you feel comfortable in X language and does it look like something you can feel comfortable with.

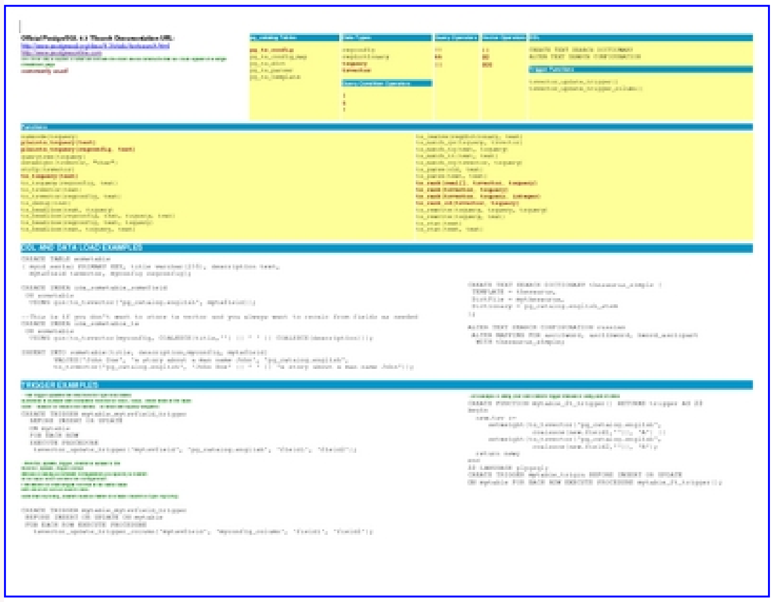
Below are links to various articles that demonstrate some uses of PLs :

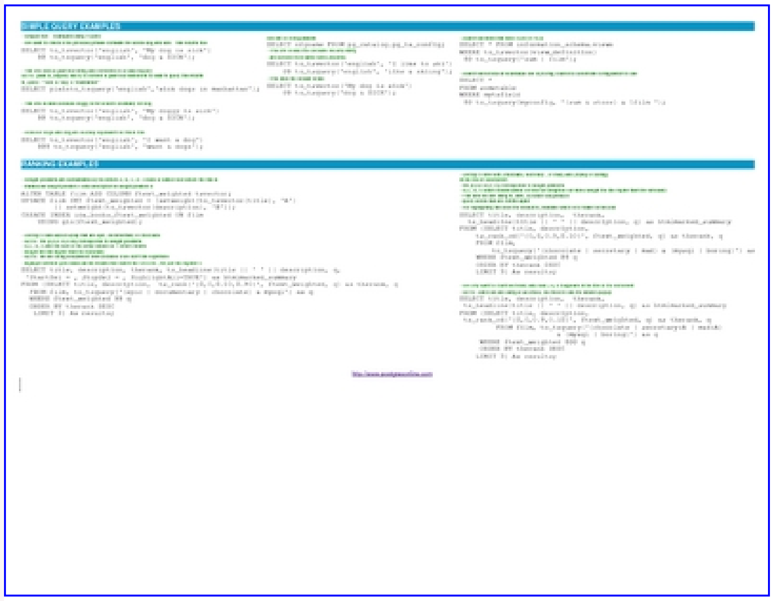
- David Fetter's A Babel of PLs PG 2007 presentation - complete list of presentations and link to audio file can be found at http://www.postgresqlconference.org/2007/talks/
- Joe Conway's 2003 Oscon presentation on PL/R - this is dated but still well worth a read.
- Prepared statements gotcha - (not quite intention of this article) - Hubert provides a PLPerl function called random_username() which is a very good use case of using PLPerl.
- Wiki PL Matrix provides status of existing PostgreSQL PLs.

Back to Table Of Contents Choosing the right Database Procedural Language PL Reader Comments

PostgreSQL 8.3 TSearch Cheat Sheet Overview *Intermediate*

Below is a Thumbnail view of a PostgreSQL 8.3 TSearch Cheat Sheet that covers PostgreSQL 8.3 Full Text search engine constructs.





This one we broke into two pages so its a bit more readable than our PostgreSQL 8.3 cheat sheet.

PDF landscape version 8.5 x 11" of this cheatsheet is available at PostgreSQL 8.3 TSearch Full-Text Search in PDF 8/12 by 11 and also available in PDF A4 format and HTML.

Back to Table Of Contents PostgreSQL 8.3 TSearch Cheat Sheet Overview Reader Comments

REST in PostgreSQL Part 3 A - Simple REST Client in Adobe Flex 3 *Intermediate*

In prior articles of this series, we covered the following:

1. Showcasing REST in PostgreSQL - The PreQuel we went over what REST is and isn't
2. REST in PostgreSQL Part 1 - The DB components we loaded the Pagila database and created a db plpgsql search function to support our rest server service
3. REST in PostgreSQL Part 2 A - The REST Server service with ASP.NET we demonstrated a REST web service using Mono. NET, MS.NET both in C#, VB.Net/Monobasic
4. REST in PostgreSQL Part 2 B - The REST Server service with PHP 5 we demonstrated a REST web service using PHP 5

What is Adobe Flex?

Adobe Flex is the development API for developing standard Flash and Adobe Air applications which Adobe calls Rich Internet Applications (RIA). Adobe Air run as desktop apps and regular Flex Flash apps run in a browser. At its core is ActionScript 3 which one can think of as a language that is a cross between Javascript and Java. It is more type sensitive than JavaScript but less so than Java. Flex is basically Adobe's effort at minimizing the *a tool for building bouncing balls and other pointless but pretty graphic effects* image of Flash and having it do real work that would cater more to application developers.

Its a fairly direct competitor to Microsoft's SilverLight, and is a more mature platform than Microsoft's SilverLight. Granted it is less ambitious. For example you can only program in ActionScript on the client (which is compiled into a binary) and interact via JavaScript where as Microsoft's SilverLight 2 incarnation promises ability to program in numerous languages on both Client and Server. Both strive to bring the responsiveness of a desktop app to the web. Flex also has an advantage in that most user's have the Flash Player installed which is supported on most OS. SilverLight will require an additional 3-5 MB download and is supported on Windows/Mac by Microsoft and will be deployable on Linux/other Unix/PDAs Platforms via Novell's MoonLight implementation.

Tools for developing Adobe Flex apps

You can develop Adobe Flex with notepad, VIM, emacs or any other favorite editor and compile with the freely available Adobe Flex 3 SDK. If you prefer having a more advanced IDE, there are 2 options that come to mind that we have tried.

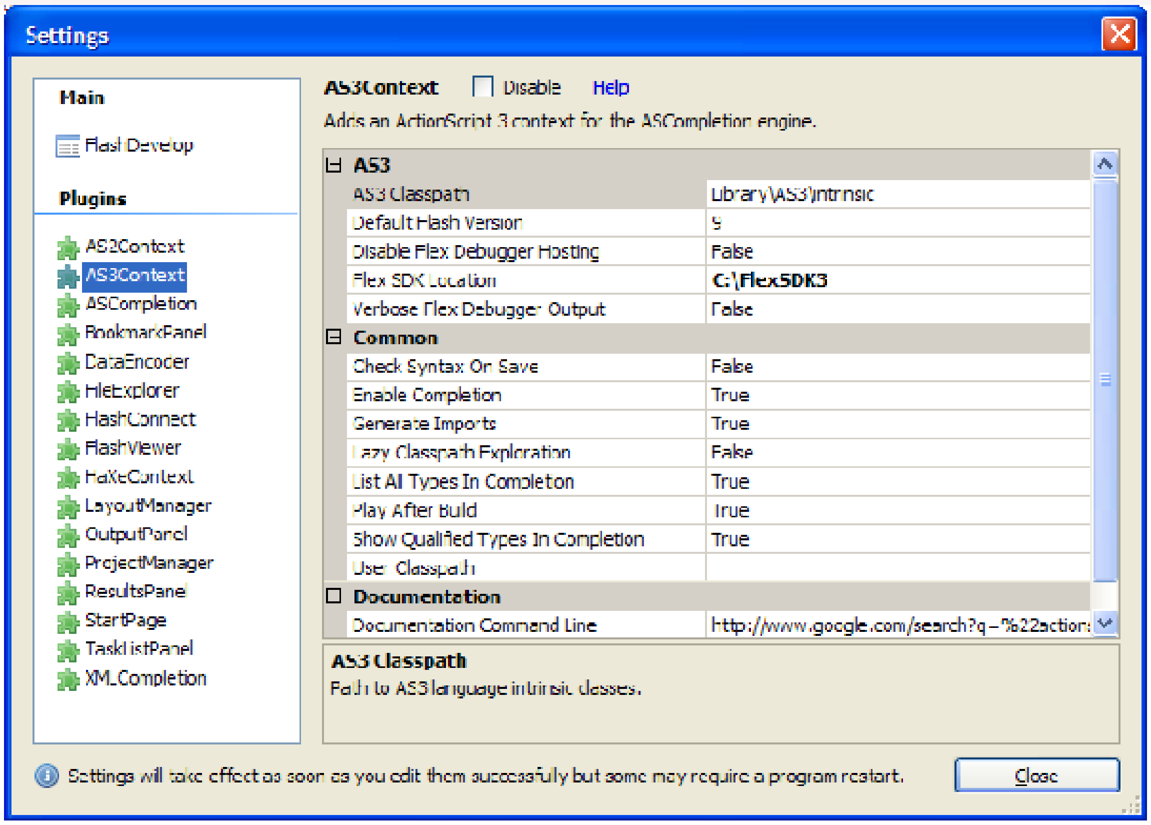
- Flex Builder download from http://www.adobe.com/cfusion/entitlement/index.cfm?e=flex3email - this is Adobe's IDE for Flex development. Flex Builder is built using the Eclipse Framework and can be installed as a standalone or as part of the Eclipse IDE. Flex sports *What you see is what you get (WYSIWIG) functionality*, intellisense, on the fly error handling, compile on save. It costs about $800, but has a 60-day trial version. We presume this should work anywhere you have a Java runtime 1.6 or above.
- FlashDevelop - download from http://www.flashdevelop.org/community/viewforum.php?f=11 - This is a freely available Open Source Flex IDE. It lacks the WYSIWIG functionality of Flex Builder, but similar to Flex Builder, sports intellisense for both action script and Adobe application XML and simple compile with click of a button, error handling response. It is built using .NET Framework 2.0 and Sharp Develop. Haven't tried this in Linux so not sure if it works under Mono. It does work on MacOSX according to reports.

Setting up FlashDevelop IDE Environment

For this exercise, we will be using FlashDevelop. WYSIWIG IDEs always make us feel like we are giving up responsiveness to WYSIWIG and in many cases we find WYSIWIG to be a distraction from our core motive. For a simple app - WYSIWIG is not needed. For more decorative apps, you may be better going with Flex Builder.

To setup the development environment - do the following:

1. Install Java 1.6 if you don't have it already - Flex3 SDK needs it
2. Download FlashDevelop and install http://www.flashdevelop.org/community/viewforum.php?f=11. We are using the 3.0.0 Beta 7 version.
3. Download Flex 3 SDK from http://www.adobe.com/cfusion/entitlement/index.cfm?e=flex3email. Keep in mind you do not need Flex Builder which is also downloadable from this page and over 300 MB.
4. Extract flex3sdk into some folder.
5. Install FlashDevelop (Note you will need .NET Framework 2 for this)
6. Launch FlashDevelop
7. Go to Tools -> Program Settings -> Select AS3Context and set FlexSDK Location to where you extracted the SDK as shown



Adds an ActionScript 3 context for the ASCompletion engine.

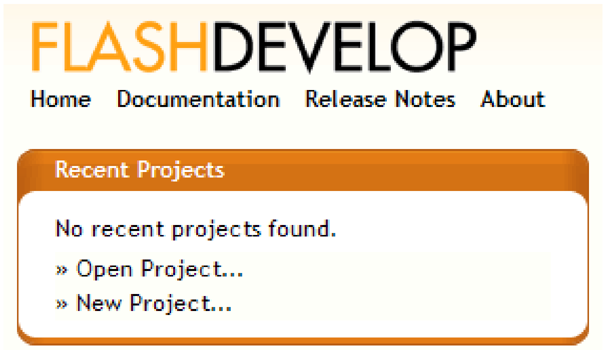
AS3	
AS3 Classpath	Library\AS3\intrinsic
Default Flash Version	9
Disable Flex Debugger Hosting	False
Flex SDK Location	**C:\Flex SDK3**
Verbose Flex Debugger Output	False
Common	
Check Syntax On Save	False
Enable Completion	True
Generate Imports	True
Lazy Classpath Exploration	False
List All Types In Completion	True
Play After Build	True
Show Qualified Types In Completion	True
User Classpath	
Documentation	
Documentation Command Line	http://www.google.com/search?q=%22action:

AS3 Classpath
Path to AS3 language intrinsic classes.

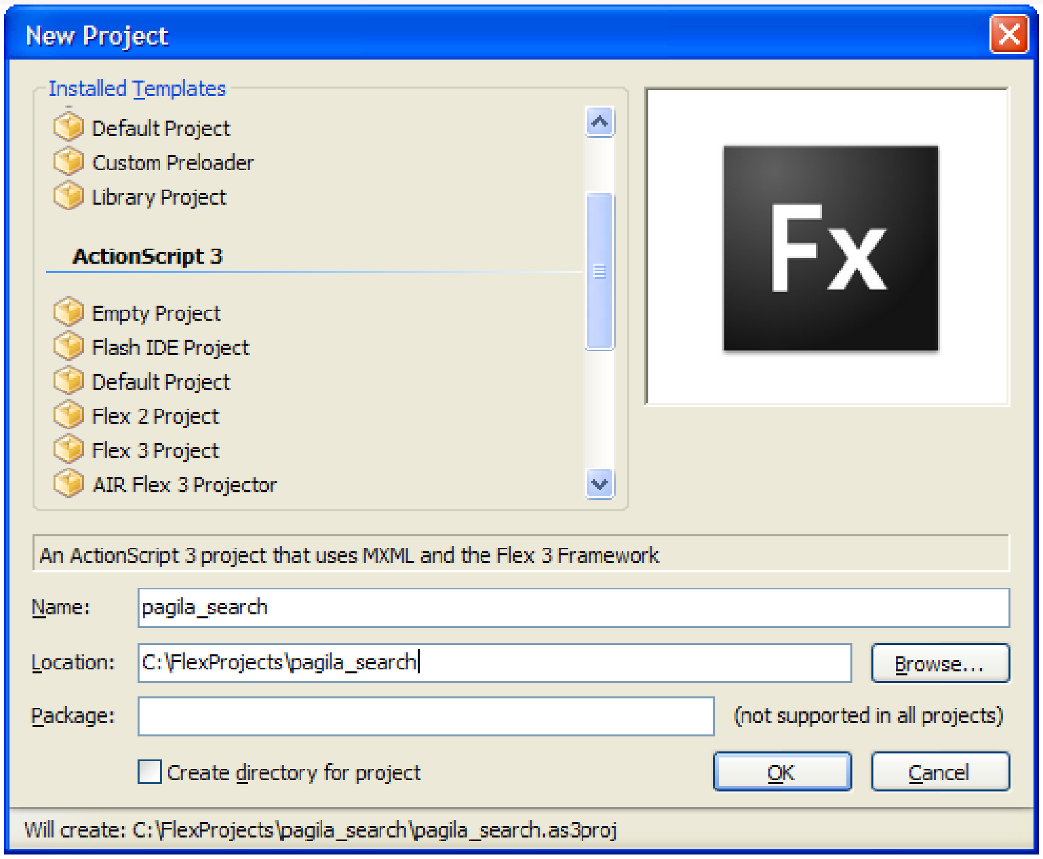
Settings will take effect as soon as you edit them successfully but some may require a program restart.

Building Pagila Search Adobe Flex Client in FlashDevelop

To build a Pagila Search client that will use our REST Server service, we do the following:



1. Click on New Project from FlashDevelop Recent Projects dashboard -
2. Select Flex 3 Project and specify the path to store the project files as shown in the picture -



An ActionScript 3 project that uses MXML and the Flex 3 Framework

Name: pagila_search

Location: C:\FlexProjects\pagila_search

Package: (not supported in all projects)

Create directory for project

Will create: C:\FlexProjects\pagila_search\pagila_search.as3proj

3. On menu tab select -> Project -> Properties -> Test Movie - set to Play in Flash Viewer

The IDE creates a rudimentary *Main.mxml* file in the src folder and sets it to always compile. We shall for simplicity put all our code in this file.

1. Open up this file and replace the contents with the below.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical">
  <mx:Script>
          <![CDATA[
        import mx.collections.ArrayCollection;
        import mx.rpc.events.ResultEvent;
        import mx.rpc.events.FaultEvent;
        import mx.controls.Alert;
        [Bindable]
        private var sresults:ArrayCollection;
        private var numresults:String;
        public function handleXml(event:ResultEvent):void
        {
           numresults = event.result.results.resultsummary.count;
           txtResultStatus.text = numresults + " items fit your search criteria";
           if (numresults == "0")
           {
              sresults = null;
           }
           else
           {
              sresults = event.result.results.table.row;
           }
        }

        public function handleFault(event:FaultEvent):void
        {
          Alert.show(event.fault.faultString, "Error");
        }
      ]]>
  </mx:Script>
```

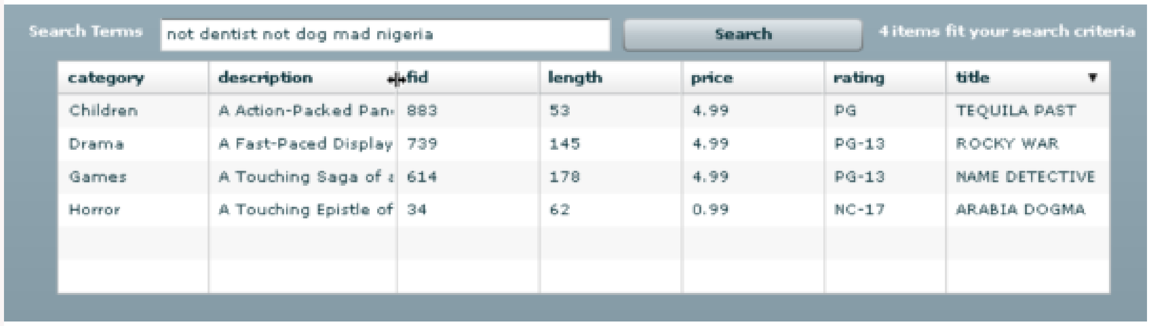
```
    <mx:HTTPService id="xmlRpc"
        url="http://localhost:8080/pagila_php.php"
        result="handleXml(event)"
        fault="handleFault(event)">
        <mx:request>
            <query>{search.text}</query>
            <maxrecs>20</maxrecs>
        </mx:request>
    </mx:HTTPService>

    <mx:VBox>
        <mx:HBox id="HBoxUser" width="100%">
            <mx:Label text="Search Terms" textAlign="right" fontWeight="bold" color="white" />
<mx:TextInput id="search" width="300" height="22" />
            <mx:Button x="130" y="95"
            label="Search"
            click="xmlRpc.send()"
            width="160" height="22" />
            <mx:Label id="txtResultStatus" fontWeight="bold" color="white" />
        </mx:HBox>
    </mx:VBox>
    <mx:DataGrid id="grdResult" dataProvider="{sresults}" editable="false" variableRowHeight="true"/>
</mx:Application>
```

2. Click F8 to build the project (this is optional since F5 will automatically build and play).
3. Click F5 to run the generated flash file.

The above is a fairly brain-dead implementation. What it does is the following:

- Calls our REST Service. Which is hard-coded - you may want to dynamically set that.
- It looks at our XML resultsummary portion ... and checks to see if there are records. It puts the number of records in the results label.
- If results are returned it loads them in a variable called sresults which then gets bound to our Flex grid. If none, then it sets sresults to null.
- Flex Grid provides out of the box grid sorting, column dragging, dynamic width change, and if you don't explicitly specify the columns it infers it from the datasource.

Below is a screen shot of our brain-dead implementation in action.



Search Terms
not dentist not dog mad nigeria
Search
4 items fit your search criteria

category	description	fid	length	price	rating	title
Children	A Action-Packed Pan	883	53	4.99	PG	TEQUILA PAST
Drama	A Fast-Paced Display	739	145	4.99	PG-13	ROCKY WAR
Games	A Touching Saga of	614	178	4.99	PG-13	NAME DETECTIVE
Horror	A Touching Epistle of	34	62	0.99	NC-17	ARABIA DOGMA

Deploy on webserver

To deploy the application on your webserver

1. Copy the generated flash file from the bin folder of your FlashDevelop project to webserver
2. Create an html file that looks something like this: Note for brevity we left out all the check for flash version etc.

```
<html lang="en">
<head>

<title>My Pagila Search Client</title>


<style>
body { margin: 0px; overflow:hidden }
</style>
```

```
</head>

<body scroll="no">

   <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
         id="pagilasearch" width="100%" height="100%"
         codebase="http://fpdownload.macromedia.com/get/flashplayer/current/swflash.cab">
         <param name="movie" value="pagilasearch.swf" />
         <param name="quality" value="high" />
         <param name="bgcolor" value="#ffffff" />
         <param name="allowScriptAccess" value="sameDomain" />
         <embed src="pagilasearch.swf" quality="high" bgcolor="#ffffff"
            width="100%" height="100%" name="pagilasearch" align="middle"
            play="true"
            loop="false"
            quality="high"
            allowScriptAccess="sameDomain"
            type="application/x-shockwave-flash"
            pluginspage="http://www.adobe.com/go/getflashplayer">
         </embed>
   </object>
</body>
</html>
```

Back to Table Of Contents

REST in PostgreSQL Part 3 B - The REST Client in Adobe Flex 3 with Paging *Intermediate*

In prior articles of this series, we covered the following:

1. Showcasing REST in PostgreSQL - The PreQuel we went over what REST is and isn't
2. REST in PostgreSQL Part 1 - The DB components we loaded the Pagila database and created a db plpgsql search function to support our rest server service
3. REST in PostgreSQL Part 2 A - The REST Server service with ASP.NET we demonstrated a REST web service using Mono.NET, MS.NET both in C#, VB.Net/Monobasic
4. REST in PostgreSQL Part 2 B - The REST Server service with PHP 5 we demonstrated a REST web service using PHP 5
5. REST in PostgreSQL Part 3 A - Simple REST Client in Adobe Flex 3 we demonstrated a basic REST client in Adobe Flex

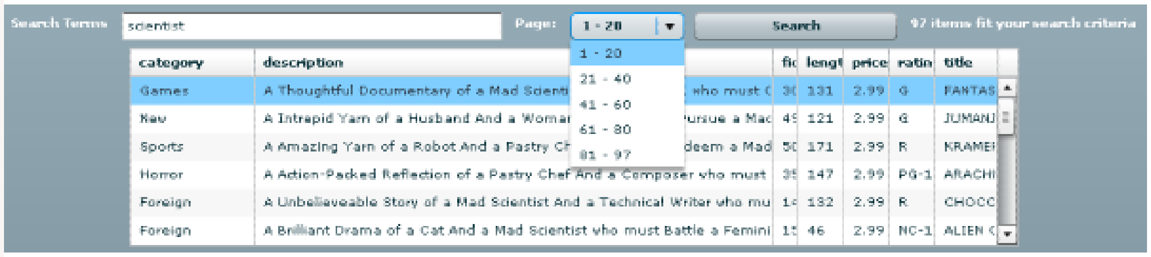
In this article we shall continue where we left off by adding paging functionality to our Adobe Flex REST grid client.

Put in Paging Drop Down

We have revised our code to deal with dynamic paging. The below code does the following:

1. In the display portion, we have drawn a combo box to hold paging and initialize it to invisible.
2. We have changed our webservice offset to be bound to a function of combo box index. Note the index denotes the page we want to pull.
3. We need to do a bit more when the search button is clicked, so we have created a function called **search_click()** which will reset our offset and paging back to one. We do this because clicking Search triggers a new search
4. In the combo box page drop down we have it to trigger the webservice call.
5. During the webservice call, if the page combo is at 1 (meaning a new search), we redraw the combo as function of number of results.

Our revised application view and code is shown below.



category	description	fi	lengt	price	ratin	title
Games	A Thoughtful Documentary of a Mad Scienti who must (	3(	131	2.99	G	FANTAS
New	A Intrepid Yarn of a Husband And a Womar ursue a Mac	4!	121	2.99	G	JUMANJ
Sports	A Amazing Yarn of a Robot And a Pastry Cr deem a Mad	5(	171	2.99	R	KRAMEI
Horror	A Action-Packed Reflection of a Pastry Chef And a Composer who must	3!	147	2.99	PG-1	ARACHI
Foreign	A Unbelieveable Story of a Mad Scientist And a Technical Writer who mu	1<	132	2.99	R	CHOCC
Foreign	A Brilliant Drama of a Cat And a Mad Scientist who must Battle a Femini	1!	46	2.99	NC-1	ALIEN (

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical">
  <mx:Script>
    <![CDATA[
                  import mx.collections.ArrayCollection;
      import mx.rpc.events.ResultEvent;
      import mx.rpc.events.FaultEvent;
      import mx.controls.Alert;
      [Bindable]
      private var sresults:ArrayCollection;
      private var numresults:String;


      public function handleXml(event:ResultEvent):void
      {
        var i:int;
        var npages:Array;
        numresults = event.result.results.resultsummary.count;

        txtResultStatus.text = numresults + " items fit your search criteria";
        if (numresults == "0")
```

```
            {
               sresults = null;
               cboPages.visible = false;
               lblPage.visible = false;
               npages = new Array(1);
               npages[0] = '';
               cboPages.dataProvider = npages;
            }
            else
            {
               sresults = event.result.results.table.row;
               cboPages.visible = true;
               if (cboPages.selectedIndex == 0){ //new search request
                  npages = new Array(int(int(numresults) / 20));
                  for (i = 0; i <= int(int(numresults) / 20); i++)
                  {
                     if ((i + 1) * 20 > int(numresults))
                     {
                        npages[i] = (i * 20 + 1) + ' - ' + numresults;
                     }
                     else {
                        npages[i] = (i * 20 + 1) + ' - ' + (i + 1) * 20;
                     }
                  }
                  cboPages.dataProvider = npages;
               }
               lblPage.visible = true;

            }


         }

         public function handleFault(event:FaultEvent):void
         {
            Alert.show(event.fault.faultString, "Error");
         }

         public function search_click():void {

            xmlRpc.request.offset = 0;
            cboPages.selectedIndex = 0;
            xmlRpc.send();

         }]]>
   </mx:Script>

   <mx:HTTPService id="xmlRpc"
      url="http://localhost:8080/pagila_php.php"
      result="handleXml(event)"
      fault="handleFault(event)">
      <mx:request>
         <query>{search.text}</query>
         <maxrecs>20</maxrecs>
         <offset>{int(cboPages.selectedIndex)*20}</offset>
      </mx:request>
   </mx:HTTPService>

   <mx:VBox>
      <mx:HBox id="HBoxUser" width="100%">
         <mx:Label text="Search Terms" textAlign="right" fontWeight="bold" color="white" /> <mx:TextInput
id="search" width="300" height="22" />
         <mx:Label text="Page:" id="lblPage" textAlign="right" fontWeight="bold" color="white" visible="false" />
         <mx:ComboBox id="cboPages"  width="90" visible="false" change="xmlRpc.send()">
            <mx:dataProvider>
               <mx:Array><mx:String></mx:String></mx:Array>
            </mx:dataProvider>
         </mx:ComboBox>
         <mx:Button x="130" y="95"
         label="Search"
         click="search_click()"
         width="160" height="22" />
```

```
            <mx:Label id="txtResultStatus" fontWeight="bold" color="white" />
        </mx:HBox>
    </mx:VBox><mx:DataGrid id="grdResult" dataProvider="{sresults}" editable="false" variableRowHeight="true"/
>
</mx:Application>
```

Back to Table Of Contents

PHP Gallery 2 for Picture Storage and Simple Document Management *Intermediate*

What is PHP Gallery 2?

PHP Gallery 2 is a web-based management system for storing pictures and other documents such as movies and flash files. While it is not designed for storing documents such as Microsoft Word or PDF, it serves as a simple storage container for those as well and will even automatically create thumbnails for PDFs if you have ImageMagick installed. It is similar to Gallery 1 except unlike Gallery 1, the meta data of documents is stored in a database as opposed to the file system. Documents are still stored in the file system. Gallery is Open Source software licensed under GPL. Details here.

We've been using Gallery 2 for various projects over the past year or so because it has been fairly easy to integrate into our PHP applications. Below is the list of features we like most about it:

1. Supports one of our favorite databases and those other 2 - PostgreSQL, MySQL, Oracle. Minor gripe - you can tell from the docs that there is a MySQL bias.
2. Cross-Platform - will work anywhere PHP works.
3. It uses PHP ADODB as the database abstraction layer.
4. It uses Smarty Templating engine.
5. Lots of Plugins to choose from - we'll go over our favorites later
6. When you upload a high-res image it automatically creates 2 other sizes (thumbnail and regular web view)

People may ask when you've got Flickr and Picasa and all that other stuff, why would you ever go with Gallery 2. We haven't tried Flickr or those others, so we can't really speak of their merits or downfalls, but the reasons we prefer Gallery 2 over those other options is the same reason we prefer Serendipity Blogging engine over something like Google Blogger. We have more control, more seamless integration with our other database applications, and if you loose internet connection and are running an intranet, you are not out of luck.

Gallery 2 Gotchas when using PostgreSQL: Case Sensitivity when doing Search

Gallery 2 for our purposes has performed very nicely with PostgreSQL except for case-sensitivity when doing searches. People must think we are broken records by now. Here is a fine occasion where the case-sensitivity of PostgreSQL and as I recall Oracle - bites you. We have come up with 2 ways of overcoming this obstacle.

1. Make PostgreSQL non-case sensitive - this we outlined in Using MS Access with PostgreSQL
2. Make Gallery 2 compensate for Case Sensitivity - Hack the /modules/core/classes/GalleryCoreSearch.class -> search as described in this thread we posted to http://gallery.menalto.com/node/18076

Cool Plugins

A lot of the plugins require the following Graphics Toolkits

- Image Magick which has binaries available for Unix, Windows and Mac OSX. Image Magick is a very cool free open source graphics package that can be used within PHP, .NET, Perl etc. It resizes images, flips them, removes EXIFs, convert in-between various formats, even deals with CMYK images. Very cool. If you haven't used it and do a lot of graphical manipulation in your applications, I highly suggest you look into the 100s of features it offers.
- FFMPEG is used by Gallery for doing things like putting water marks on MPEG files and other MPEG manipulation
- Zip/Unzip is used by the shopping cart feature to allow a user to download a whole album or selected documents as a zip and the unzip is used by Gallery upload to allow a user to upload a batch of documents as a single zip that are then extracted. Each folder in the zip becomes a sub album. Linux/Unix machines already have this pre-installed. For windows users you'll need cygwin1.dll and the zip and unzip.exe that come with cygwin

Below are some of our highly recommended Plug-ins in addition to the Graphics Toolkits and pre-installed:

1. Cart and Zip Download: Makes it easy for users to pick pictures they want and download High-Res versions
2. Keyword Album: Allows you to have dynamic albums created based on Keywords you type in
3. Flash Video, MP3 Audio: Allows these to be played right on the page and for Flash set the desired size
4. Archive Upload: Allows uploading zip files and having them explode into albums and album items
5. Numerous other upload plugins: XP Upload (for uploading from Windows XP Explorer), Picasa, etc.
6. Javascript Slideshow

Back to Table Of Contents

Reader Comments

What can PostgreSQL learn from MySQL

Jay Pipes
Regarding "Zack Urlocker of Sun has espoused, the upcoming MySQL 5.1 will have no bugs so perhaps this along with other bug complaints is a moot point."

Huh? Software with no bugs? Sounds like a marketing gesture to me. Where did you get this quote from?

-jay

Regina
Its in the eWeek Ferrari of Databases article we listed above. The exact phrase used was "This version now has zero bugs," Urlocker told eWEEK. I think what he meant was all bugs in MySQL bug tracker have been corrected.

David Fetter
> PostgreSQL people look like a gang of geeks and Martin Mickos looks polished

I'm curious just exactly which "PostgreSQL people" you're referring to here. The vast majority of the people who travel and speak keep both their hair and their words short and tidy. There are two people I can think of offhand who wear ponytails, and neither of them otherwise fits any "geek" stereotype.

Anonymous
This post reads a bit like a joke -- it's hard to take it seriously. Maybe that was done on purpose?

Anyhow, I'd appreciate it if you could elaborate on "...PostgreSQL has caught up there and is beginning to see the fruits of that labor." -- what kinds of fruitds of that labour is PG seeing? Could you describe that a bit more, please? Thank you.

Chris Mellish
MySQL is popular for the same reason Visual Basic is popular, it is simple to get started with and you see instant results.

Take a fresh download of Postgres, the steps to get TCP/IP configured are not straight forward or obviously documented - MySQL just works.

Some of the non-standard features that make MySQL so poor in the eyes of a professional, actually help soften the learning curve of the database for those with little experience who aren't used to thinking like a DBA. As you get started on your first MySQL based project and start creating a schema, things like auto_increment are easy to get your head around.

And just as with many Microsoft extensions to standards, those features end up being a barrier to entry with other databases as you progress, keeping people hooked on MySQL and finding ways to make it work for them, rather than having to discard everything they've learnt to date.

Postgres documentation for those who are already literate in DB speak is very comprehensive and covers all the features. However there are two distinct gaps in the documentation:

1. Beginners guides, written in plain English, that explain the basic concepts and technical terms that people are going to come across, as well as taking them step by step through their first example project. This guide should really be database agnostic, but provide examples for Postgres.

2. A concept migration guide - there are lists of features that Postgres has that MySQL doesn't, and the like, but I have not found a guide that explains in simple terms what each of those features is, how to use them, why you should use them, and how they all fit together. Finding out the syntax for a particular statement is easy, but finding out how and why you should be using that facility is not covered.

I think that both these issues help reinforce the outsiders view that PG people tend to have an elitist attitude to non-PG people. I write this as a recent convert, having finally had enough of MySQLs limitations with an enterprise class project, that had started out small and grown beyond the DB's capabilities. I always looked at the feature set in PG with envy, but until recently I found the DB just too different for me to want to invest in making the switch.

Jay Pipes
Hi David,

I tend to agree with you that I think the pony-tail/geek argument doesn't hold too much water. I mean, Jonathan Schwartz, CEO of Sun, has a ponytail and bridges the professional and the geek quite gracefully. :)

However, I *do* think that the authors of this article are correct in that there is certainly an attitudinal difference between MySQL "advocates" and PG "advocates". The former, in my experience, tend to me more moderate and less dogmatic about theoretical things than their PG counterparts. The dogmatism and ideology of some PG advocates, especially in the arena of licensing (BSD vs GPL) and the arena of "pure relational DBMS" arguments have been a turn-off to elements of the broader technology field that are not DBAs by trade.

To put it in a one-liner, you will rarely find a MySQL user/advvocate say that "PG is crap". Instead, often they will say "PG is great, but different, and more complex". But the opposite, im my experience, is not true. I often run across blog posts and PG advocates that generalize "MySQL is crap. Use PG, it's a *real* database".

This attitude is a turn-off to a large segment of people.

Just my two cents. Cheers,

Jay

LewisC
I would have to agree with Jay's thought about attitude. While it's not true for everyone in the pg community it is true for many. It's not even *just* against mysql. It's against every other database. It seems that if it's not done the pg way, it's not done the right way.

LewisC

Dave
It's all about the name.

Well, no, it's not all about the name. But I wish it would change. It's not that it's really bad (not compared to the gimp anyway), it's just that it stumbles off the tongue, and it's not very memorable. "Howdyou say that now? Post Rest QL?"

Quite honestly I've recommended the use of PostgreSQL quite a few times when it was the appropriate tool but the name always seemed to be a bit of a stumbling block. When you're trying to pitch to a non-techy crowd, the name is more important than it ought to be.

It started as Ingres. Why couldn't it have been called Egress and saved me all this hassle? :-)

Leo
I agree too. Even Oak as stupid as it is slides off the tongue more easily. Saying PostgreSQL just doesn't make you sound like a smooth talker in meetings.

Regina
Okay I admit, the ponytail pick was a bit unfair. I guess I was thinking of the comic store owner guy in Simpsons whenever I hear someone in PostgreSQL groups make a snide remark about MySQL and other things. I admit I am not immune to this fault so part of it is a pick at me too. It just irritates me immensely when I see the reflection of my worst inner faults in others. :)
- e.g. "ah you were using that pedestrian database to do real work? No wonder you have problems." "STOP TOP POSTING!"

You have to admit that comparing being terse and to the point vs. being mean is like comparing apples and oranges. You can be very mean and to the point at the same time.

Regina
This is just from personal experience. We meant that before we'd have clients where PostgreSQL was a better option for them than say MySQL or SQL Server and it was a really really hard-sell to pitch PostgreSQL. For lots of things it really doesn't matter if you run MySQL or PostgreSQL or SQL Server.

Now we are seeing clients (even those running on Windows) that are already sold on PostgreSQL before they come to us.

Toba
I would like to raise one MySQL feature to the list, working integrated replication. PostgreSQL replication is a joke compared to MySQL one. Not to even get started with non-working PgCluster compared to very proven MySQL Cluster. Building high available, scalable systems is quite hard with Postgres, and that is a big problem if one wants to use Postgres for commercial applications.

Yes it's possible, I have setup DRBD + Slony replication for my company, but I used to maintain large MySQL replication and damn it was easy and better working.

Tonci Grgin
Having no experience with PostgreSQL I will not engage in technical details but rather speak of my journey with MySQL. In the late 90's, my former boss decided to switch our SW to true SQL server. After many a discussion, choice fell on MySQL. At first, we were amazed with speed and ease of use. Later on, when problems surfaced, I was always able to get Monty or Sinisa on the phone and ask for advice. This was a great asset to us! Following on this, MySQL got from us one of the first Windows GUI client and numerous replies on windows mailing list once it was established. My point being you need quality response from community and to get that you need to invest a lot! Take a Jay Pipes for example (btw. Hi Jay!), MySQL community team consists of great professionals lead by one of the most important persons in MySQL. No compromise, community get's the best.
This brings me to, along with Chris's comment "1. Beginners guides, written in plain English", another point I wish to make. Although above seems unrelated, it is! What Marten brought to MySQL, besides looking sharp ;-), was enough money to be spent on community, support & docs team. It is my opinion that all three teams are of top quality in business. And only support makes money.
So, it doesn't seem right to attack MySQL over commercializing part of it's offers as most of the money goes back to where it should, to community.

To the authors, minority of one doesn't mean you're crazy! Although Nietzsche was well aware how hard that might be. Do not antagonize people for being fond of "anchoring", it is our nature. I have never met a vegetarian that didn't try to "convert" my eating habits ;-).

Finally, I'm looking forward to checking into PostgreSQL deeper, now that we're all part of SUN.

Tom Briggs
Thanks for the link. I'm happy to see others using S9Y. :)

I'm actually more a fan of PostgreSQL than MySQL. In fact, the only thing I like about MySQL is the pluggable storage engine option. If there's a way to do something similar with PostgreSQL I'd love to learn about it.

One comment concerning the lack of polished geeks promoting PG: Martin Mickos fits in with the business types because he runs a business. And that's really the key - the simple fact that there's a business behind MySQL makes it immensely more acceptable in the corporate world. There's a contract to sign and an entity to sue (if necessary), and that makes it palettable. Not so with PostgreSQL, at least not as far as I know. And until or unless that changes, PostgreSQL will never be adopted, officially or unofficially, as much as MySQL.

Regina
You are welcome Tom. I remember someone mentioning that way back when (e.g. maybe in the 6+ era) PostgreSQL had a pluggable storage engine, but the feature was discarded because it was too much trouble to maintain.

As far as contracts to sign and entity's to sue, I always thought most organizations are over that or at lest they are into suing consultants rather than software vendors. I can't really imagine suing Oracle for anything and their contracts seem more like customer chains. Still I think a lot of people read a lot into a suit and just because we are open source hackers doesn't mean we can't look good too :).

I never think of an entity to sue behind apache, but that is largely an accepted webserver by corporate types although I never paid much attention to the Apache model or the face of apache so maybe there is something behind that.

Berend de Boer
And we have to add that MySQL is vastly easier to program than PostgreSQL. Just try to write a procedure that returns a result set in PostgreSQL.

The weird distinction between PostgreSQL and PlSQL means programming PostgreSQL requires more skills.

For most applications MySQL is good enough. And easier to program.

Note that I myself prefer PostgreSQL, but would hardly ever recommend it to a customer.

Regina
Berend,

I would agree that for most applications MySQL is good enough, but not necessarily easier to program. I too have recommended it over PostgreSQL for other reasons - e.g. it was an application that they needed well supported by many ISPs, they already had a MySQL server and were comfortable with it and they weren't doing anything complicated. But NEVER because it was easier to program.

The easier to program is I think more a perception than reality. If you are doing something trivial it will be trivial in either. Just because people tend to do simpler things in MySQL doesn't make it easier to program.

If you look at even Monty's criticisms of MySQL http://www.scribd.com/doc/2575733/The-future-of-MySQL-The-Project on slide 12. You can not use set returning stored procs of MySQL in the from clause which I find to be a huge disadvantage for most of the applications we write and makes many SQL Server to MySQL job ports painful. For simple set returning functions - nothing is easier than a PostgreSQL SQL set returning function. The MySQL PL is NOT easier.

I have had really bad experiences in MySQL where I wrote a marginally complicated View for one 5..release. something version and in a minor point newer release the view just didn't work at all. The same view works perfectly in SQL Server and PostgreSQL with no change. For MySQL prior to 5 - I even refused to work on many such jobs or charged a 20% premium because I can't imagine living without views for a marginally complex project.

Pythian Group Blog
This is the 96th edition of the weekly review of database blogs, Log Buffer...

PostgreSQL 8.4 goodies in store

pabloj
WITH RECURSIVE should be the implementation of CTE in PostgreSQL.
It's something already available in Firebird, SQLServer 2005, IBM DB2 and Oracle (CONNECT BY synthax).
A much needed synthax improvement to handle hierarchical queries.

Slava
I *really* wish the Windowing functions could make in the upcoming 8.4 release.
We use Postgres to handle analytical requests in 1TB+ Datawarehouse and it makes a big difference if we could get the valuable information by doing just one table scan.
Thank you.

Leo
Did you see this hacker post just in today by H. Harada. Looks like he might take up the challenge to get it into 8.4 in some shape or form.

http://archives.postgresql.org/pgsql-hackers/2008-06/msg00380.php

Leo
Pablo,
Thanks for the correction to our Oracle typo. Nice blog you have by the way. We really should explore Firebird more. Sounds like a nice database.

Slava

Thanks for the link to the post. This is great if the windowing function will be able to make in 8.4. I have used them quite a lot in Oracle and Teradata, but can barely remember a single time I needed the "Frame" functionality. I personally consider the clause ".. ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW" quite a cumbersome thing to understand. So if it doesn't make in first release - it wouldn't be much problem for me.

How to calculate Running Totals and Sums in SQL

depesz
There is also a way to do it without subqueries, using pl/pgsql.

I described it in a blogpost here: http://www.depesz.com/index.php/2007/08/17/rownum-anyone-cumulative-sum-in-one-query/

Leo
Depesz,
That looks pretty neat.
Thanks

duke
The sub-select solution doesn't look very good for few reasons:
1) it's not readable
2) subqueries will use lots of temporary space when materialize during execution and it's generally not efficient for big datasets.

What postgres desperately needs to compete with commercial database in Datawarehousing is a oracle-like window analytical functions. I hope 8.4 will bring us closer to this!

Cross Compare of SQL Server, MySQL, and PostgreSQL

PJ
There are plans for MERGE for PostgreSQL 8.4 (if Simon Riggs writes it in time) + there is an example in docs on how to do UPSERT using PL/pgSQL function.

jon
cost: 1 million dollars! / free / free

BLAHBLAH
I'm sorry, but if you are going to present a case with such clear bias as "few but ramping up" on your underdog, while shitting bricks on any of the smallest flaws of the guy you dislike, you might as well be done with the fancy html tables and put a picture of a chimp flinging feces.

As an aside, I have to take your bait on the installation: if you call the setup for MSSQL difficult, then you are either outright lying or do not have the mental capacity to hit "Next>>" 5 times and enter a few names when prompted. You seriously saying this is difficult?

Oh, and dropping tables being a misfeature? Are you fucking retarded??

Get a life man, seriously, get a life and be done with your stupid bitterness.

PJ
You should learn to read. He clearly states why it is misfeature and he is right, dropping table which has objects (views in this case) depending on it is clearly a misfeature.
Also installation/maintenance is more then just hitting Next five times (all three dbs have installation wizard like that), you want to setup db for your specific use pattern, you want to make backups etc.

Regina
Honestly BlahBLAH if that is your real name :) SQL Server is one of those databases we particularly love to work with. So if you think this is a complete bash, you are dead wrong. We are just pointing out things we would like changed in it.

You are forgetting that as PJ said there is more to supporting and maintaining a database than 5 keystrokes and if you think that is all, I feel pity for you. SQL Server 2005 requires you have .NET Framework 2 preinstalled, among other prerequistes. The amount of time you need to wait in between those 5 steps (and if you happen to be on a box that doesn't have those things downloading them) is a LOT MORE than you have to with PostgreSQL and MySQL. Troubleshooting when it actually fails is more work too (at least on Windows).

As we said there is nothing as sweet as the maintenance plan or tuning wizard that comes with SQL Server 2005 and those are things PostgreSQL and MySQL from our experience are sorely lacking.

If truth be known we make a ton more dough supporting SQL Server than any other database and we don't go around throwing poo at our cash cow for no good reason.

Our general philosophy if you care about something you should point out its faults as well as its strengths because otherwise you can't be taken seriously.

Joe
The row for "Case-insensitivity" is confusing. What exactly do you mean by PostgreSQL "By default is case-sensitive and a pain to make it not so?" If you're talking about it being case-sensitive about identifiers that have been created as quoted-identifiers, e.g., CREATE TABLE tableX vs. CREATE TABLE "tableX" vs. CREATE TABLE "TableX", then MySQL and MSSQL (I presume) are not following the ANSI/ISO SQL Standard. PostgreSQL isn't following it entirely because the first CREATE should instantiate a table

named "TABLEX" in its catalogs, but instead creates one named "tablex". What I found painful (but I haven't used it for a while) is MySQL's use of non-standard backticks for quoting identifiers.

Can you create a table named "user" or "USER" (or some other SQL reserved keyword) in either MySQL or MSSQL? In PostgreSQL you can create both and they can co-exist.

Leo
Joe,

Actually what we were talking about was text comparisons like

somefield LIKE '%abc%'

vs.

somefield LIKE '%ABC%'

In many cases you want those to be treated the same. Sure you could do ILIKE in PostgreSQL, but it doesn't use indexes and is not cross platform. Having to do upper upper is a pain too. Our point is that it should be controlled at the column level and SQL Server lets you do that. PostgreSQL does not let you do that.

mgroves
Is this some sort of joke?

Tom Bille
I must out of painfull experience speak against the point that mysql is case insensitive - it relies on the operating system, so importing and exporting between linux and windows tends to break everything.

Bill
"CREATE TABLE tableX ... should instantiate a table named TABLEX in its catalogs"

Like Oracle does it, then? If that's what ISO says, then the standard is wrong. It's a nuisance and strongly encourages AWFUL_NAMES_LIKE_THIS, since GoodTableNames become unreadable unless you quote absolutely everything.

Nils
MySQL supports CREATE FUNCTION for SQL Functions. MySQL also supports User defined Functions and Aggregate Functions coded in C. Replication is Master->Slave but one server can take 2 roles. 5.1 also ships an Event Scheduler, but current version don't so it's ok to say "no".

You're missing some criteria, like Price, Sequence support, Online Backup...

Artacus
Another good row might be XML/XPath support.

Regina
Nils - thanks for the correction, we'll update to reflect.

Kurt
"Partial Indexes - e.g. you want to create a unique index but only consider non-null values"

You can do these in SQL Server with indexed views, which work even on the Express edition.

Indexes on views are enforced even when modifying the underlying table directly.

TC
I gather that some of these products have some restrictions in their SQL; eg. no correlated subqueries? I'd like to see some info on that. Otherwise, the article is a good start. #2 is a foul mouthed idiot.

Leo
Actually all 3 support correlated subqueries so we didn't think to add that as a deciding point. Although since you mention it, I suppose its something that people wouldn't know and that is an important factor.

anonymous
What data type support? I'm more familiar with oracle on these, but I find it ridiculous that oracle still forces you to store data out of band (i.e., clobs) for more than 4000 characters, compared to pg's simple straightforward 'text' type. How do mysql and sql server stack up here?

Also, what about full-text search? Oracle has intermedia which is naturally extra cost but is pretty powerful. What do any of these three have?

Ries
PostgreSQL has tsearch2 for full text, at least more pwoerfull then MySQL's full text.

About case sensitive, most languages I know are case sensitive, unless you tell them not to do so. This goes for perl, php, java etc etc etc.... So I find it natural that in a DB it's also case sensitive.

The examples that Leo are given (LIKE '%ABC%' etc) are not anchored and will never be indexed by any RDMB, as far as I know. So Saying that ILIKE '%abcd%' cannot use an index is true for all databases. I strongly feel in such situations full text needs to be used unless table size is small.

Looking at the tabl,e I don't find this particular usefull... Just my opinion...

Leo
Ries - I have to disagree with you on this one. In SQL Server 2005 - right down to the column level you can specify the collation, language and the sort order (dictionary, binary etc) and when you do a LIKE search it if you specify case insensitive dictionary (which is default), it will still use the index.

There are a lot of things databases do differently from programming languages e.g the way they short-circuit.

From a portability standpoint this makes SQL Server apps that utilize this feature (which is a lot) harder to port to PostgreSQL.

Leo
Slight correction - LIKE 'ABC%' LIKE 'abc%' are index searchable but LIKE '%ABC%' etc are not. So you are right that was a bad example.

Mike McNally
Clustered indexes?

Adam Zochowski
In MS-SQL / Transact SQL, you cannot drop a column, table or view if it is schema bound.

```
create table test_table
( a int not null )

create view test_view
with schemabinding
as
select a from dbo.test_table


drop table test_table


Msg 3729, Level 16, State 1, Line 1
Cannot DROP TABLE 'test_table' because it is being referenced by object 'test_view'.
```

cheers

PAul
My opinion, given this comparasion is that SQL Server is the best because it is the most comprehensive and easier to use

Joe
Maybe it's because I'm more comfortable on Unix (even though I've used VMS, DOS, Windows, etc.), but treating LIKE '%abc%' the same as LIKE '%ABC%', by default, seems unnatural to me.

Mark Stock
I think text comparisons in SQL databases are painful.

My solution is to create two columns where one would would suffice. The first column is the DISPLAY text, and the second column is the SEARCH_SORT text. The display text is whatever was entered. The search_sort text is a copy of the text stored as all UPPER or all lower case characters, pick one. My preference is lower case, but historically UPPER case is used. There are minor reasons to go with lower case, but now I'm getting off topic. If your SQL database likes indexes, then you might want to put one on the SEARCH_SORT text column. When you do a comparison, convert the desired search text to the same case as you are using in your search_sort column. Do the search using the text in matching cases. For example:

The search text might be "Katie" and so it gets converted to "katie". "Starts with" was chosen from the user interface so a % gets tacked onto the end of the LIKE search text. In my example, the text in the search_sort_name column is always in lower case, even though the name column retains any mixed case characters.

```
SELECT person_id, name
FROM person
WHERE search_sort_name LIKE 'katie%'

50016, Katie
```

There are even more gymnastics involved with the LIKE cause. The wildcard characters, different ways various SQL databases handle wildcard characters and embedded characters in the search text play havoc, but perhaps this is another topic also.

I hope that my techno-babble is useful to someone. ;-)

tene
And why would there not be a truly free DBMS - FireBird?

Milos Babic
And why you don't have locking mechanism comparison?

Regina

Well PostgreSQL is truly free since it is BSD License too - you can modify and do not need to give back to the community if you don't want to and it is community controlled as opposed to single company controlled. I suspect FireBird is the same. Reading looks like FireBird is IPL licensed (InterBase Public License) which looks similar in spirit to BSD and it is controlled by a non-Profit Firebird Foundation.

We mentioned FireBird even though we haven't investigated it because of its choice of installation features - (SuperServer, Classic and Embedded). This puts it in an interesting spot because we see it as something with almost the lightness of SQLite but that can be spinned into a true server side db, but that is much more powerful than SQLite (e.g. it isn't type impoverished etc). As I recall it is also a one file db which makes it useful from a portability stand-point similar to SQLite. The embedded not requiring installation step is very alluring.

MySQL is just confusing in terms of licensing although arguably less confusing now that they are a part of Sun. They still need to work more on their community image since I get the impression its hard for non-MySQL employees to make direct contributions to the MySQL core code base.

Regina
Milos,

We thought about that too, but we thought locking mechanism would open up a whole can of worms that not everyone would be able to appreciate unless they were deep into the trenches of the DB infrastructure. Comparing Locking mechanisms of the various dbs we felt to do it justice really needs a whole article dedicated to just that. Perhaps in another article we shall cover just that.

Along similar lines there is also the issue of how threads are spurn, page level compression etc.

Regina
We may add that to the pile but that may be a long discussion.

Both PostgreSQL and Microsoft SQL Server offer some form of clustered indexes (MySQL sort of does too, but I think it varies depending on storage engine and for example InnoDb, its always clustered by the primary key which neither PostgreSQL nor SQL Server require - so MySQL clustering works in a similar fashion to MSSQL). MSSQL's maintains the cluster index at a price (price is lower for SQL Server 2005 than it was for SQL Server 2000), while PostgreSQL requires a CLUSTER operation to maintain cluster. So on short side - MSSQL incurs a penalty during update/insert and in some cases having a CLUSTERED index on MSSQL may actually make non-covered index queries slower since all indexes are tagged against the clustered index. Dropping a clustered index also requires rebuild of all indexes in SQL Server since all are referenced to that.

PostgreSQL incurs a penalty for maintenance. There are also other differences such as MSSQL's use of covering indexes (which in many cases MSSQL will not need to go back to the actual table record - can read straight from an index if the query fields can be satisfied by the index).

So in short while PostgreSQL's clustering is less sophisticated it often performs better than SQL Server's. There are pros and cons to both.

K. Brian Kelley
"install/maintenance process"

SQL Server can be tuned to take up less resources. The idea behind SQL Server is different than say, MySQL. SQL Server was architected to be the only thing on the box. It was designed to consume memory and handle scheduling on its own, so it could be more responsible. As a result, it will grab the resources unless you tell it not to. Now, is it is configurable as something like Oracle? No, it's not. But you can make SQL Server pretty low resource impact. SQL Server Express is actually designed around that concept.

"Can drop tables"

For SQL Server 2005 and above, you have more than schemabinding. You also have DDL triggers which can prevent ANY schema change.

Kevin
MySQL has MASTER-MASTER replication in addition to MASTER-SLAVE replication. I have set this (M-M) up recently in my production environment and it works well. Clustering is also available.

Postgres OnLine Journal
The PostgreSQL 8.4 planned release is March 1, 2009 and is outlined in the PostgreSQL 8.4 Development plan.
It has just passed its May 2008 commit fest milestone and is currently in its July 2008 Commit Fest. Lots of PostgreSQL Planet bloggers have star

Choosing the right Database Procedural Language PL

RW
Isn't it PL/Scheme, not PL/Schema? Regarding your comment about Smalltalk vs Lisp when looking for a language that "morphs itself" (and saying that Lisp is less fitting), mistaking "schema" for "scheme" makes it sound like you simply have more experience with Smalltalk and are less familiar with Lisp. It may not be true, but it could be construed that way. Please proofread to avoid confusion.

Regina
Thanks for the typo catch :).

I'm used to thinking of schemas when talking about databases so that was a bit of a Freudian slip. Regarding Lisp - wasn't trying to imply it can't morph itself. Mostly I just don't care much for the syntax of Lisp and that was my big gripe with it. You are right though that I have more experience with SmallTalk than Lisp and haven't programmed in Scheme since college.

Java Languages
hubs about Java Languages to **...** and the domain specific **languages** PL/R, PL/SH and gaining popularity Skype released PL/Proxy. There are others in the family such as PL/Tcl, PL/PHP, PL/Ruby, PL/Schema (a dialect of Lisp), PL/**Java**, PL/Lua and PL/LOLCode (for kicks **...**

Procedural Languages
hubs about Procedural Languages to Most people are used to having their **procedural language** push their data around. PL code living in PostgreSQL allows your data to push your **procedural** code around in a set-based way. This is a simple but pretty powerful feature since **...**

eggyknap
One nit to pick: It's PL/LOLCODE, not PL/LOLCode :)

Also, regarding point 4, above, specifically "[pushing data between PostgreSQL and some language interpreter] makes most languages somewhat suboptimal for set returning functions or functions that consume a lot of data." I'm not sure I buy this. Sure, converting data into a format the language backend likes takes time and cycles, but less time and cycles than it would to push all that data into some front-end application, process it there, and push it back. One of the huge advantages of a PL is that you can keep database-like functions (data integrity, for instance, or massive data crunching) in the database, where the data don't have to move too far between storage and processing.

Regina
couldn't agree with you more. Regarding the note about set returning functions being sub optimal. for the most part you are right its probably more efficient to have the function in the db than call the function from native code. Our comment was more along the lines of writing set returning functions in something like SQL/PLPGSQL that in general writing set returning functions in those built in languages is more efficient than writing in say PLPython or PL/R. Of course some things you just can't do easily in SQL/PLPGSQL so then you would resort to another language and efficiency gains may actually be higher when you consider other things like processing speed in that language and just more efficient functions you can piggy back on.

PostgreSQL 8.3 TSearch Cheat Sheet Overview

Rauan
Hi. Thanks a lot. Just read your article on Tsearch and have been waiting for cheatsheet. :)

Rauan
Hmm... You've forgotten plainto_tsquery(text). I think it's very important function.

Leo
Thanks we'll add that one to the list.

Official PostgreSQL 8.3 TSearch Documentation URL:
http://www.postgresql.org/docs/8.3/static/textsearch.html
http://www.postgresonline.com
We cover only a subset of what we feel are the most useful constructs that we could squash in a single cheatsheet page
commonly used

pg_catalog Tables

pg_ts_config
pg_ts_config_map
pg_ts_dict
pg_ts_parser
pg_ts_template

Data Types

regconfig
regdictionary
tsquery
tsvector

Query Condition Operators

|
&
!

Query Operators

!!
&&
||

Vector Operators

||
@@
@@@

DDL

CREATE TEXT SEARCH DICTIONARY
ALTER TEXT SEARCH CONFIGURATION

Trigger Functions

tsvector_update_trigger()
tsvector_update_trigger_column()

Functions

numnode(tsquery)
plainto_tsquery(text)
plainto_tsquery(regconfig, text)
querytree(tsquery)
setweight(tsvector, "char")
strip(tsvector)
to_tsquery(text)
to_tsquery(regconfig, text)
to_tsvector(text)
to_tsvector(regconfig, text)
ts_debug(*text*)
ts_headline(*text*, *tsquery*)
ts_headline(*regconfig*, *text*, *tsquery*, *text*)
ts_headline(regconfig, text, tsquery)
ts_headline(text, tsquery, text)
ts_lexize(regdictionary, text)
ts_match_qv(tsquery, tsvector)
ts_match_tq(text, tsquery)
ts_match_tt(text, text)
ts_match_vq(tsvector, tsquery)
ts_parse(oid, text)
ts_parse(text, text)
ts_rank(real[], tsvector, tsquery)
ts_rank(tsvector, tsquery)
ts_rank(tsvector, tsquery, integer)
ts_rank_cd(tsvector, tsquery)
ts_rewrite(tsquery, tsquery, tsquery)
ts_rewrite(tsquery, text)
ts_stat(text)
ts_stat(text, text)

DDL AND DATA LOAD EXAMPLES

```
CREATE TABLE sometable
( myid serial PRIMARY KEY, title varchar(255), description text,
  mytsfield tsvector, myconfig regconfig);

CREATE INDEX idx_sometable_somefield
 ON sometable
  USING gin(to_tsvector('pg_catalog.english', mytsfield));

--This is if you don't want to store ts vector and you always want to recalc from fields as needed
CREATE INDEX idx_sometable_ts
 ON sometable
  USING gin(to_tsvector(myconfig, COALESCE(title,'') || ' ' || COALESCE(description)));

INSERT INTO sometable(title, description,myconfig, mytsfield)
        VALUES('John Doe', 'a story about a man name John', 'pg_catalog.english',
        to_tsvector('pg_catalog.english', 'John Doe' || ' ' || 'a story about a man name John'));
```

```
CREATE TEXT SEARCH DICTIONARY thesaurus_simple (
 TEMPLATE = thesaurus,
 DictFile = mythesaurus,
 Dictionary = pg_catalog.english_stem
);

ALTER TEXT SEARCH CONFIGURATION russian
 ALTER MAPPING FOR asciiword, asciihword, hword_asciipart
  WITH thesaurus_simple;
```

TRIGGER EXAMPLES

--This trigger updates the field tsvector type field called
mytsvfield in mytable with combined tsvector of field1, field2,..fieldn fields in the table.
Note -- number of fields is not limited. All fields are equally weighted.

```
CREATE TRIGGER mytable_mytsvfield_trigger
  BEFORE INSERT OR UPDATE
  ON mytable
  FOR EACH ROW
  EXECUTE PROCEDURE
   tsvector_update_trigger('mytsvfield', 'pg_catalog.english', 'field1', 'field2');
```

--tsvector_update_trigger_column is similar to the
tsvector_update_trigger except
instead of taking a constant configuration,you specify a column
in the table which defines the configuration.
This allows for multi-lingual records in the same table
with different full text search rules.
Note that myconfig_column must be name of a table column of type *regconfig*

```
CREATE TRIGGER mytable_mytsvfield_trigger
 BEFORE INSERT OR UPDATE ON mytable
 FOR EACH ROW EXECUTE PROCEDURE
   tsvector_update_trigger_column('mytsvfield', 'myconfig_column', 'field1', 'field2');
```

--An example of using your own custom trigger instead of using built-in ones

```
CREATE FUNCTION mytable_ft_trigger() RETURNS trigger AS $$
begin
  new.tsv :=
     setweight(to_tsvector('pg_catalog.english',
                coalesce(new.field1,'')), 'A') ||
     setweight(to_tsvector('pg_catalog.english',
                coalesce(new.field2,'')), 'B');
  return new;
end
$$ LANGUAGE plpgsql;
CREATE TRIGGER mytable_trigiu BEFORE INSERT OR UPDATE
ON mytable FOR EACH ROW EXECUTE PROCEDURE mytable_ft_trigger();
```

SIMPLE QUERY EXAMPLES

```
--Snippet two  - examples using TQuery
--We want to check if the provided phrase contains the words dog and sick.  This returns true
SELECT to_tsvector('english', 'My dog is sick')
    @@ to_tsquery('english', 'dog & SICK');

--This one uses a plain text string and convertes to a valid tsquery
NOTE: plain to_tsquery will try to convert a plain text statement to valid ts query this returns
ts_query - "'sick' & 'dog' & 'manhattan'"
SELECT plainto_tsquery('english','sick dogs in manhattan');

--This one is false because doggy is not a word boundary for dog
SELECT to_tsvector('english', 'My doggy is sick')
    @@ to_tsquery('english', 'dog & SICK');

--However dogs and dog are lexically equivalent so this is true
SELECT to_tsvector('english', 'I want a dog')
    @@@ to_tsquery('english', 'want & dogs');
```

```
See list of configurations
SELECT cfgname FROM pg_catalog.pg_ts_config;
--This one is also true because ski and skiing
--are derived from same word (lexeme)
SELECT to_tsvector('english', 'I like to ski')
    @@ to_tsquery('english', 'like & skiing');
--This uses the default locale
SELECT to_tsvector('My dog is sick')
    @@ to_tsquery('dog & SICK');
```

```
--Search all views that have SUM or FILM
SELECT * FROM information_schema.views
WHERE to_tsvector(view_definition)
 @@ to_tsquery('sum | film');

--Search all records in sometable use myconfig column to determine configuration to use
SELECT *
FROM sometable
WHERE mytsfield
 @@ to_tsquery(myconfig, '(sum & store) & !film ');
```

RANKING EXAMPLES

```
--Weight positions are demarcated by the letters A, B, C, D. Create a fulltext field where the title is
--marked as weight position A and description is weight position B
ALTER TABLE film ADD COLUMN ftext_weighted tsvector;
UPDATE film SET ftext_weighted = (setweight(to_tsvector(title), 'A')
        || setweight(to_tsvector(description), 'B'));
CREATE INDEX idx_books_ftext_weighted ON film
    USING gin(ftext_weighted);

--List top 3 films about Mysql that are epic, documentary or chocolate
--NOTE: the {0,0,0.10,0.90} corresponds to weight positions
--D,C, B, A and the sum of the array should be 1 which means
--weight the title higher than the summary
--NOTE: we are doing a subselect here because if we don't the expensive
--highlight function gets called all the results that match the WHERE, not just the highest 3
SELECT title, description, therank, ts_headline(title || ' ' || description, q,
 'StartSel = , StopSel = , HighlightAll=TRUE') as htmlmarked_summary
FROM (SELECT title, description,  ts_rank('{0,0,0.10,0.90}', ftext_weighted, q) as therank, q
  FROM film, to_tsquery('(epic | documentary | chocolate) & mysql') as q
  WHERE ftext_weighted @@ q
  ORDER BY therank DESC
   LIMIT 3) As results;
```

```
--List top 3 films with (chocolate, secretary , or mad) and (mysql or boring)
in the title or description
--the {0,0,0.90,0.10} corresponds to weight positions
--D,C, B, A which means based on how we weighted our index weight the title higher than the summary.
--This time we are using ts_rank_cd which will penalize
--query words that are further apart
--For highlighting this uses the default ts_headline which is to make terms bold
SELECT title, description,  therank,
  ts_headline(title || ' ' || description, q) as htmlmarked_summary
FROM (SELECT title, description,
    ts_rank_cd('{0,0,0.9,0.10}', ftext_weighted, q) as therank, q
    FROM film,
        to_tsquery('(chocolate | secretary | mad) & (mysql | boring)') as
    WHERE ftext_weighted @@ q
    ORDER BY therank DESC
    LIMIT 3) As results;

--We only want to count secretary and mad (:A) if it appears in the title of the document
--NOTE: Since we are using a GIN index, we need to use the slower @@@
SELECT title, description,  therank,
 ts_headline(title || ' ' || description, q) as htmlmarked_summary
FROM (SELECT title, description,
 ts_rank_cd('{0,0,0.9,0.10}', ftext_weighted, q) as therank, q
         FROM film, to_tsquery('(chocolate | secretary:A | mad:A)
                        & (mysql | boring)') as q
     WHERE ftext_weighted @@@ q
     ORDER BY therank DESC
    LIMIT 3) As results;
```

http://www.postgresonline.com