

The Postgres Application Server

Copyright 2008
David Fetter
All Rights Reserved.

Large Applications In Postgres

- PostGIS
- Bucardo
- DBI-Link
- etc., etc., etc.



Your First Postgres Application

```
CREATE SCHEMA math;
```

```
CREATE TABLE math.fibonacci_memoize(  
    n numeric PRIMARY KEY,  
    fib_n numeric NOT NULL  
);
```

```
CREATE OR REPLACE FUNCTION math.memoize_fib(  
    n numeric,  
    fib_n numeric  
)  
RETURNS numeric  
LANGUAGE SQL  
AS $$  
    INSERT INTO math.fibonacci_memoize  
        VALUES ($1, $2);  
    SELECT $2;  
$$;
```

```
CREATE OR REPLACE FUNCTION math.fibonacci(numeric)
RETURNS numeric
LANGUAGE SQL
AS $$
    SELECT COALESCE(
        (SELECT fib_n FROM math.fibonacci_memoize WHERE n=$1),
        memoize_fib(
            $1,
            CASE WHEN $1 < 2
                THEN $1
            ELSE
                math.fibonacci($1-2) + math.fibonacci($1-1)
            END
        )
    );
$$;
```

Dia_Postgresql_Campinas_2008=> SELECT math.fibonacci(1360);

7476627396809555566|73648400|802236553|893853972|16|938685659756|9247|45|34|4|08222446|886460|29
49850679252203985|397|8940206737|20793|104|167403866|855|7|8928|2907625|16606|668342|77|7654|745
809926607375829704760249067|933273|4248785049678|78037490773433878|053032404|722007526|14395
(1 row)

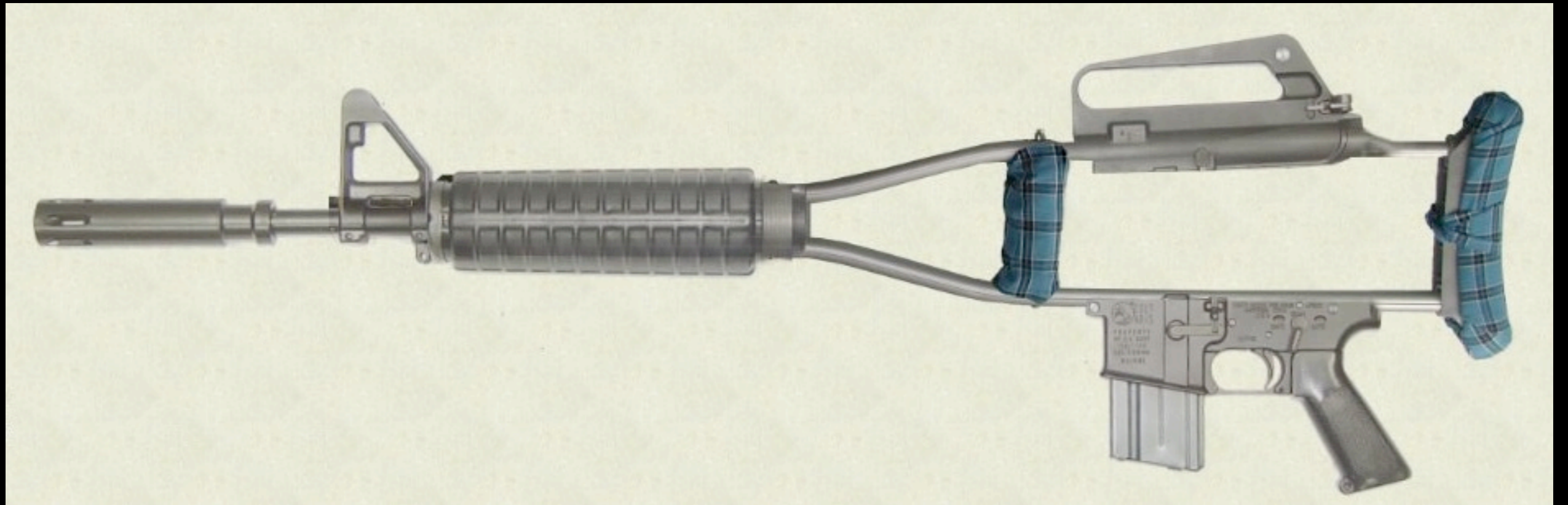
Time: 649.425 ms

Dia_Postgresql_Campinas_2008=> SELECT math.fibonacci(1360);

7476627396809555566|73648400|802236553|893853972|16|938685659756|9247|45|34|4|08222446|886460|29
49850679252203985|397|8940206737|20793|104|167403866|855|7|8928|2907625|16606|668342|77|7654|745
809926607375829704760249067|933273|4248785049678|78037490773433878|053032404|722007526|14395
(1 row)

Time: 0.758 ms

Postgres Tools



Programming Languages

- SQL
- PL/PgSQL
- PL/Lua(U)
- PL/Perl(U)
- PL/PHP(U)
- PL/Parrot(U)?
- etc., etc., etc.

Trust

(Security Backwards)

- Functions in untrusted languages can:
 - Open Filehandles
 - Open Pipes
 - Do Dangerous Things™.
- Functions in trusted languages
 - Cannot

Lua!

Você pode usar a linguagem Lua dentro Postgres!



<http://pgfoundry.org/projects/pllua/>

Perl



Ruby



<http://rubygems.org/>

Debuggers

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

Kernighan

Building and Installing PL/Lua

Build and install:

```
make && make install
```

```
psql -f pllua.sql your_database
```

A Custom Data Type

Bar Codes:

- * UPC
- * EAN
- * GTIN
- * Nossa!

How to do it

Luckily, they are all GTINs

And they have a checksum

Creating The GTIN Domain

```
CREATE DOMAIN gtin AS BIGINT  
    CHECK ( is_gtin(VALUE) );
```

The Function

```
CREATE OR REPLACE FUNCTION is_gtin(bigint)
RETURNS BOOLEAN
IMMUTABLE
LANGUAGE ?
AS $$
...
$$;
```

How to do it in SQL

```
CREATE OR REPLACE FUNCTION is_gtin(bigint)
RETURNS BOOLEAN
LANGUAGE sql
STRICT IMMUTABLE
AS $$
    SELECT ( sum(dgt) % 10 ) = 0
    FROM (
        SELECT substring($1 from idx for 1)::smallint AS dgt
        FROM    (SELECT generate_series(length($1), 1, -2) as idx) AS foo
        UNION ALL
        SELECT substring($1 from idx for 1)::smallint * 3 AS dgt
        FROM    (SELECT generate_series(length($1) -1, 1, -2) as idx) AS foo
    ) AS bar;
$$;
```

How (not!) to do it in PL/Perl

```
CREATE OR REPLACE FUNCTION is_gtin(bigint)
RETURNS BOOLEAN
LANGUAGE plperl
AS $$
    my $i = 0;
    my $total = 0;
    for my $c (reverse split //, shift) {
        $i++;
        $total += $i % 2 ? $c : $c * 3;
    }
    return $total % 10 ? 'false' : 'true';
$;
```

How to do it in PL/Ruby

```
CREATE OR REPLACE FUNCTION isa_gtin(bigint)
RETURNS BOOLEAN
LANGUAGE plruby
AS $$
    chars = args[0].to_s.split('').reverse
    total = 0
    chars.each_index { |i|
        total += i % 2 == 0 ? chars[i].to_i : chars[i].to_i * 3
    }
    return !(total % 10 == 0)
$$;
```

A New Data Type

```
CREATE DOMAIN email AS TEXT  
CHECK(is_email(VALUE));
```

The Function

```
CREATE OR REPLACE FUNCTION is_email(text)
RETURNS BOOLEAN
IMMUTABLE
LANGUAGE plperl
AS $$
...
$$;
```


How Perl Programs Start:

```
use strict;  
use warnings;
```

More of the Program

```
use Email::Valid;
my $address = shift;
my $checks = {
    -address    => $address,
    -mxcheck    => 1,
    -tldcheck   => 1,
    -rfc822     => 1,
};
```

```
if (defined Email::Valid->address( %$checks )) {  
    return 'true'  
}
```

```
warn "address failed $Email::Valid::Details check."  
return 'false';
```

The Email Domain in Action

```
Dia_Postgresql_Campinas_2008=> SELECT  
'david@fetter.org'::email;  
email
```

```
david@fetter.org  
(1 row)
```

```
Dia_Postgresql_Campinas_2008=> SELECT  
'david@fetter.con'::email;
```

```
ERROR: value for domain email violates check constraint  
"email_check"
```

A Domain Maker

```
use Proc::ProcessTable;
my $t = new Proc::ProcessTable;
my $p = shift ( @{ $t->table } );
my @fields = ( );
```

More Domain Maker

```
foreach my $f ($t->fields) {
    if ($p->{$f} =~ m/-?\d*\.\d+/) {
        push @fields, "$f FLOAT8";
    }
    elsif ($p->{$f} =~ m/^\d+$/) {
        push @fields, "$f INT8";
    }
    else {
        push @fields, "$f TEXT";
    }
}
```

Done!

```
print <<CREATE_TYPE;  
CREATE TYPE process_table_type AS (  
    @ {[ join(",\n", @fields) ] }  
);  
CREATE_TYPE
```

Dynamic Custom Type?!

```
$ psql Dia_Postgresql_Campinas_2008
Welcome to psql 8.3.1, the PostgreSQL interactive terminal.
...
Dia_Postgresql_Campinas_2008=> \set mptt `./make_process_table_type.pl`
Dia_Postgresql_Campinas_2008=> :mptt
CREATE TYPE
```

The PL/PerlU Function

```
CREATE OR REPLACE FUNCTION get_ps ()  
RETURNS SETOF process_table_type  
LANGUAGE plperl  
AS $$  
    use Proc::ProcessTable;  
    my $proctab = new Proc::ProcessTable;  
    return $proctab->table;  
$$;
```

A Query

```
SELECT *  
FROM  
    pg_class "c1",  
    pg_class "c2",  
    pg_class "c3";
```

What is happening?!?

```
SELECT
    a.current_query,
    MAX(p.size) AS "size"
FROM
    pg_stat_activity AS "a"
JOIN
    get_ps() AS "p"
    ON (a.procpid = p.pid)
GROUP BY a.current_query;
```

Aha!

current_query	size
SELECT *	
FROM	
pg_class "c1",	
pg_class "c2",	
pg_class "c3";	195290000

(1 row)

Perguntas?

Comentários?

Joguem as pedras?

Muito Obrigado!

Copyright David Fetter 2008

All Rights Reserved

<http://fetter.org/>

david@fetter.org