



PL/pgSQL Debugging

Who Am I?



- Jim Mlodgenski
 - jimm@openscg.com
 - @jim_mlodgenski
- Director
 - United States PostgreSQL (www.postgresql.us)
- Co-organizer of
 - Philly PUG (www.phlpug.org)
 - NYC PUG (www.nycpug.org)
- CTO, OpenSCG
 - www.openscg.com

Why?



- More web developers using PostgreSQL
 - Use programming techniques inside the database
- Wrong assumptions
 - Using stored procedures prevent SQL Injection attacks
- Migrations
 - Its how Oracle has been advocating database development for years







Example



- Find the youngest rookie to rush for 100 yards against the Dallas Cowboys

first_name	last_name	p_year	p_week	p_age
Edgerrin	James	1999	8	21
Jamal	Lewis	2000	12	21

(2 rows)



Database Developer



```
CREATE OR REPLACE FUNCTION get_youngest_rookie_backs_against_team_min_yards_simple(
    p_team VARCHAR, p_yards integer, OUT first_name VARCHAR, OUT last_name VARCHAR,
    OUT p_year integer, OUT p_week integer, OUT p_age integer)
    RETURNS SETOF record AS
$$
BEGIN
    RETURN QUERY
        WITH rookies AS (
            SELECT p.first_name, p.last_name, r.player_key,
                g.year, g.week, (g.year - p.birth_year) AS age,
                first_value(g.year - p.birth_year) OVER w AS youngest
            FROM games g, rushing r, player p
            WHERE g.game_id = r.game_id
                AND r.player_key = p.player_key
                AND (g.team = p_team OR g.opponent = p_team)
                AND r.yards >= p_yards
                AND g.year = p.debut_year
                AND r.player_key NOT IN (SELECT s.player_key
                                        FROM seasons s
                                        WHERE s.year = g.year
                                        AND s.team = p_team)
            WINDOW w AS (ORDER BY (g.year - p.birth_year))
            SELECT k.first_name, k.last_name, k.year, k.week, k.age
            FROM rookies k
            WHERE k.age = k.youngest;
        END;
    $$ LANGUAGE plpgsql;
```

Web Developer



```
CREATE OR REPLACE FUNCTION get_backs_against_team_min_yards (
    p_team VARCHAR, p_yards integer, OUT p_player_key VARCHAR,
    OUT p_year integer, OUT p_week integer)
    RETURNS SETOF record AS
$BODY$
DECLARE
    g record;
    r record;
BEGIN
    FOR g IN SELECT game_id, year, week
              FROM games
              WHERE team = p_team
                 OR opponent = p_team
    LOOP
        FOR r IN SELECT player_key, yards
                  FROM rushing
                  WHERE game_id = g.game_id
        LOOP
            IF r.yards >= p_yards AND
               NOT is_on_team(p_team, r.player_key, g.year) THEN
                p_player_key := r.player_key;
                p_year := g.year;
                p_week := g.week;
                RETURN NEXT;
            END IF;
        END LOOP;
    END LOOP;

    RETURN;
END;
$BODY$
LANGUAGE plpgsql;
```

Web Developer



```
CREATE OR REPLACE FUNCTION is_on_team(  
    p_team VARCHAR, p_player_key VARCHAR, p_year integer)  
    RETURNS boolean AS  
$BODY$  
DECLARE  
    r record;  
BEGIN  
    FOR r IN SELECT team  
                FROM seasons  
                WHERE player_key = p_player_key  
                AND year = p_year  
    LOOP  
        IF r.team = p_team THEN  
            RETURN true;  
        END IF;  
    END LOOP;  
  
    RETURN false;  
END;  
$BODY$  
LANGUAGE plpgsql;
```


Web Developer



```
CREATE OR REPLACE FUNCTION get_rookie_backs_against_team_min_yards(  
    p_team VARCHAR, p_yards integer, OUT p1_player_key VARCHAR,  
    OUT p1_year integer, OUT p1_week integer)  
    RETURNS SETOF record AS  
$BODY$  
DECLARE  
    r record;  
BEGIN  
    FOR r IN SELECT p_player_key, p_year, p_week  
        FROM get_backs_against_team_min_yards(p_team, p_yards)  
    LOOP  
        IF is_rookie_year(r.p_player_key, r.p_year) THEN  
            p1_player_key := r.p_player_key;  
            p1_year := r.p_year;  
            p1_week := r.p_week;  
            RETURN NEXT;  
        END IF;  
    END LOOP;  
  
    RETURN;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

Web Developer



```
CREATE OR REPLACE FUNCTION is_rookie_year(  
    p_player_key VARCHAR, p_year integer)  
    RETURNS boolean AS  
$BODY$  
DECLARE  
    l_year integer;  
BEGIN  
    SELECT debut_year  
        INTO l_year  
        FROM player  
        WHERE player_key = p_player_key;  
  
    IF l_year = p_year THEN  
        RETURN true;  
    END IF;  
  
    RETURN false;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

Web Developer



```
CREATE OR REPLACE FUNCTION get_youngest_rookie_backs_against_team_min_yards (
...
BEGIN
  l_age := 99;

  FOR r IN SELECT pl_player_key, pl_year, pl_week
             FROM get_rookie_backs_against_team_min_yards(p_team, p_yards)
  LOOP
    IF get_player_age_in_year(r.pl_player_key, r.pl_year) < l_age THEN
      l_age := get_player_age_in_year(r.pl_player_key, r.pl_year);
    END IF;
  END LOOP;

  FOR r IN SELECT pl_player_key, pl_year, pl_week
             FROM get_rookie_backs_against_team_min_yards(p_team, p_yards)
  LOOP
    IF get_player_age_in_year(r.pl_player_key, r.pl_year) = l_age THEN
      SELECT firstname, lastname
         INTO l_fn, l_ln
         FROM get_player_name(r.pl_player_key);

      first_name := l_fn;
      last_name := l_ln;
      p_year := r.pl_year;
      p_week := r.pl_week;
      p_age := l_age;

      RETURN NEXT;
    END IF;
  END LOOP;

  RETURN;
END;
$BODY$
LANGUAGE plpgsql;
```


Web Developer



```
CREATE OR REPLACE FUNCTION get_player_age_in_year(  
    p_player_key character varying, p_year integer)  
    RETURNS integer AS  
$BODY$  
DECLARE  
    l_year integer;  
BEGIN  
    SELECT birth_year  
        INTO l_year  
        FROM player  
        WHERE player_key = p_player_key;  
  
    RETURN p_year - l_year;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

Web Developer



```
CREATE OR REPLACE FUNCTION get_player_name(  
    p_player_key VARCHAR, OUT firstname VARCHAR,  
    OUT lastname VARCHAR)  
    RETURNS record AS  
$BODY$  
BEGIN  
    SELECT first_name, last_name  
        INTO firstname, lastname  
        FROM seasons  
        WHERE player_key = p_player_key  
        LIMIT 1;  
  
    RETURN;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

How?



- The code rarely starts so complex, but growing functional requirements and short time lines contribute to short cuts
- Developers know enough PostgreSQL to be dangerous



Fantasy Football



- Fantasy football is a statistical game in which players compete against each other by managing groups of real players or position units selected from American football teams.





Business Rules



- Passing
 - Touchdown (4 points)
 - Every 25 yards (1 point)
 - Interception (-1 point)
- Rushing
 - Touchdown (6 points)
 - Every 10 yards (1 point)
- Receiving
 - Touchdown (6 points)
 - Every 10 yards (1 point)

Business Rules (Hockey Translation)

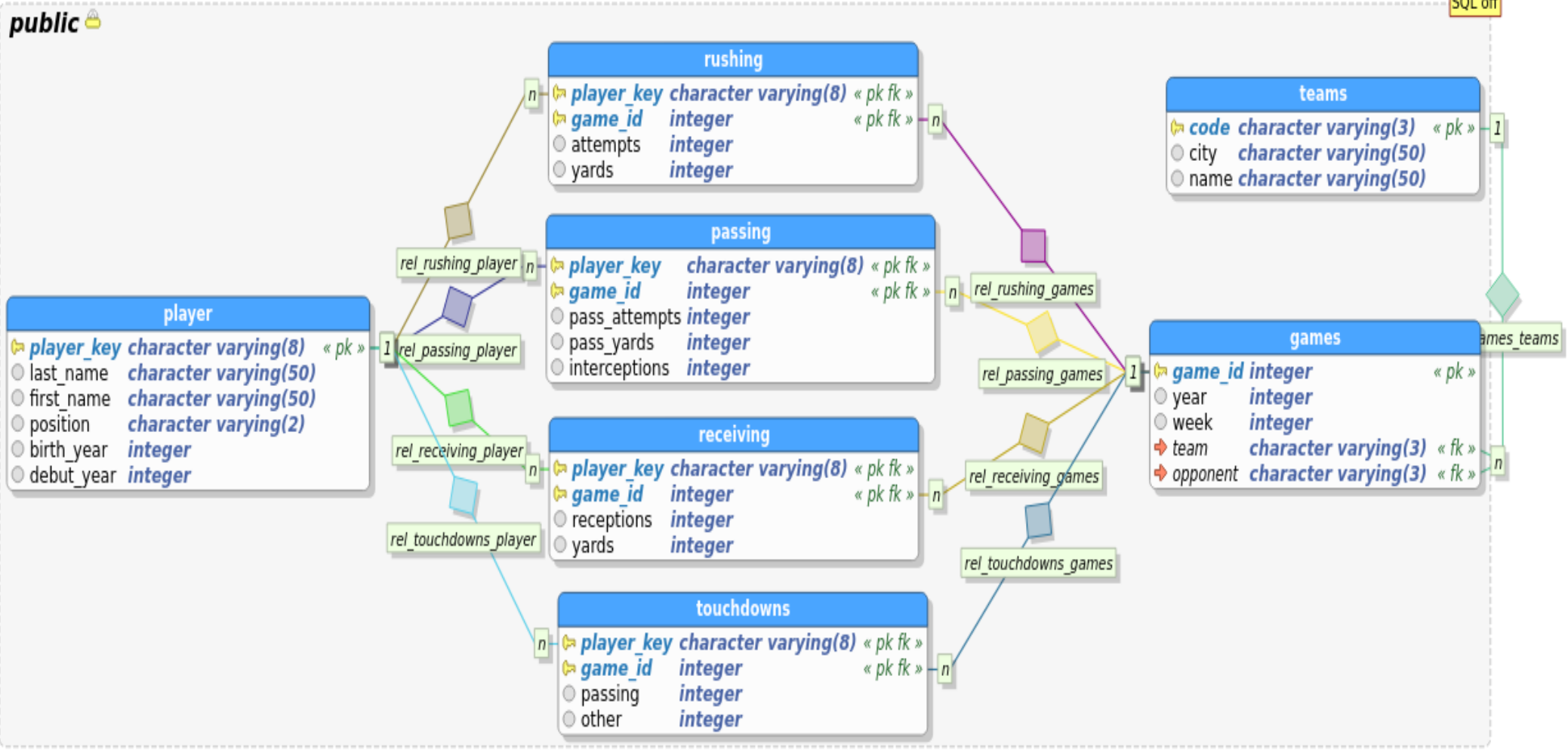


- **Goal**
 - Wings & Centers (3 points)
 - Defensemen (5 points)
 - Goalie (5 points)
- **Assist**
 - Wings & Centers (2 points)
 - Defensemen (3 points)
 - Goalie (3 points)

Sample Schema



SQL off



Procedures



```
CREATE OR REPLACE FUNCTION rushing_score(p_player_key VARCHAR,
                                         p_year int,
                                         p_week int)

    RETURNS INT AS
$$
DECLARE
    score          INT;
BEGIN
    -- 1 point for every 10 yards rushing
    SELECT r.yards/10
        INTO score
        FROM rushing r, games g
        WHERE r.game_id = g.game_id
              AND g.year = p_year
              AND g.week = p_week
              AND r.player_key = p_player_key;

    IF score IS NULL THEN
        RETURN 0;
    END IF;

    RETURN score;
END;
$$ LANGUAGE plpgsql;
```

Procedures



```
CREATE OR REPLACE FUNCTION player_game_score(p_player_key VARCHAR, p_year int, p_week int)
...
BEGIN
    score := 0;

    -- Get the position of the player
    SELECT position
        INTO l_position
        FROM player
        WHERE player_key = p_player_key;

    IF l_position = 'qb' THEN
        score := score + passing_score(p_player_key, p_year, p_week);
        score := score + rushing_score(p_player_key, p_year, p_week);
        score := score + td_score(p_player_key, p_year, p_week);
    ELSIF l_position = 'rb' THEN
        score := score + rushing_score(p_player_key, p_year, p_week);
        score := score + td_score(p_player_key, p_year, p_week);
    ELSIF l_position = 'wr' THEN
        score := score + receiving_score(p_player_key, p_year, p_week);
        score := score + td_score(p_player_key, p_year, p_week);
    ELSIF l_position = 'te' THEN
        score := score + receiving_score(p_player_key, p_year, p_week);
        score := score + td_score(p_player_key, p_year, p_week);
    ELSE
        return 0;
    END IF;

    return score;
END;
$$ LANGUAGE plpgsql;
```

Procedures



```
CREATE OR REPLACE FUNCTION avg_yearly_score(  
    p_player_key VARCHAR, p_year int)  
    RETURNS REAL AS  
$$  
DECLARE  
    score          INT;  
    i              INT;  
BEGIN  
    score := 0;  
  
    FOR i IN 1..17 LOOP  
        score := score + player_game_score(p_player_key,  
p_year, i);  
    END LOOP;  
  
    RETURN score/16.0;  
END;  
$$ LANGUAGE plpgsql;
```

Debugging: RAISE



```
CREATE OR REPLACE FUNCTION passing_score(p_player_key VARCHAR, p_year int, p_week
int)
  RETURNS INT AS
  $$
  ...
BEGIN
  score := 0;

  -- 1 point for every 25 yards passing
  SELECT p.pass_yards/25
     INTO yardage_score
     FROM passing p, games g
     WHERE p.game_id = g.game_id
           AND g.year = p_year
           AND g.week = p_week
           AND p.player_key = p_player_key;

  IF yardage_score IS NULL THEN
    yardage_score := 0;
  END IF;

  RAISE NOTICE 'Passing Yards Score: %', yardage_score;
```


Debugging: RAISE



```
nfl=# select passing_score('BreeDr00',  
2005, 5);
```

```
NOTICE: Passing Yards Score: 8
```

```
NOTICE: Passing TD Score: 4
```

```
NOTICE: Interception Score: -2
```

```
passing_score
```

```
-----
```

```
10
```

```
(1 row)
```

Debugger



```
git://git.postgresql.org/git/pldebugger.git
make USE_PGXS=1
make install USE_PGXS=1
```

- OR -

www.bigsql.org/postgresql/installers.jsp

```
shared_preload_libraries =
'$libdir/plugin_debugger'
```

```
CREATE EXTENSION pldbgap;
```

Debugger



The screenshot displays the pgAdmin III interface. On the left, the 'Object browser' shows a tree view with 'nfl' expanded to 'public' and then 'Functions (31)'. The 'passing_score' function is selected, and a context menu is open with 'Debugging' highlighted. A sub-menu is also open, showing 'Debug' as the selected option. The 'Properties' pane on the right shows details for the 'passing_score' function, including its name, OID, owner, argument count, arguments, signature arguments, return type, language, and whether it returns a set. The 'SQL pane' at the bottom shows the SQL code for the function, with a green comment line: `-- 25 yards passing` on line 10. The status bar at the bottom indicates 'Retrieving details on function passing_score... Done.' and '0.01 secs'.

Property	Value
Name	passing_score
OID	99158
Owner	jim
Argument count	3
Arguments	p_player_key character varying, p_year integer, p_week integer
Signature arguments	character varying, integer, integer
Return type	integer
Language	plpgsql
Returns a set?	No
Source	DECLARE

```
CREATE OR REPLACE FUNCTION passing_score(p_player_key character varying, p_year integer, p_week integer)
RETURNS integer AS
$BODY$
DECLARE
    score INT;
    yardage_score INT;
    yardage INT;
    game_id INT;
    game_year INT;
    game_week INT;
    game_player_key INT;
    score := 0;
-- 25 yards passing
    SET breakpoint /25
    INTO yardage_score
FROM passing p, games g
WHERE p.game_id = g.game_id
AND g.year = p_year
AND g.week = p_week
AND p.player_key = p_player_key;
```

Debugger



View Data Options

Properties

Enter the required values for each parameter:

	Name	Type	Null?	Expression?	Value	Use default?	Default Value
1	p_player_key	character varying	<input type="checkbox"/>	<input type="checkbox"/>	BreeDr00	<input type="checkbox"/>	<No default val
2	p_year	integer	<input type="checkbox"/>	<input type="checkbox"/>	2006	<input type="checkbox"/>	<No default val
3	p_week	integer	<input type="checkbox"/>	<input type="checkbox"/>	5	<input type="checkbox"/>	<No default val

Debug package initializer?

Debug Cancel

Debugger



Debugger - public.passing_score

```
DECLARE
  score      INT;
  yardage_score INT;
  td_score   INT;
  int_score  INT;
BEGIN
  score := 0;

  -- 1 point for every 25 yards passing
  SELECT p.pass_yards/25
  INTO yardage_score
  FROM passing p, games g
  WHERE p.game_id = g.game_id
  AND g.year = p_year
  AND g.week = p_week
  AND p.player_key = p_player_key;

  IF yardage_score IS NULL THEN
    yardage_score := 0;
  END IF;

  -- 4 points for every passing TD
```

Stack pane
passing_score(character varying,integer,integer)

Output pane

Name	Type	Value
score	integer	0
yardage_sc	integer	6
td_score	integer	NULL
int_score	integer	NULL

Parameters Local Variables

Waiting for target (step over)... Done 0:00:00.200 Ln 19 Col 1 Ch 352

Debugging Triggers



The screenshot shows the pgAdmin III interface. On the left, the Object browser displays a tree view of database objects, including Trigger Functions. The 'teams_trigger()' function is selected, and a context menu is open over it, with 'Set breakpoint' highlighted. The Properties pane on the right shows details for the 'teams_trigger' function, including its name, OID, owner, and source code. The SQL pane displays the SQL code for the function, which includes a DROP statement, a CREATE OR REPLACE statement, and an ALTER statement.

Property	Value
Name	teams_trigger
OID	20984
Owner	jim
Argument count	0
Arguments	
Signature arguments	
Return type	trigger
Language	plpgsql
Returns a set?	No
Source	BEGIN...

```
-- Function: teams_trigger()
-- DROP FUNCTION teams_trigger();

CREATE OR REPLACE FUNCTION teams_trigger()
  RETURNS trigger AS
$BODY$
BEGIN
  IF (TG_OP = 'DELETE') THEN
    RAISE NOTICE 'How is the NFL losing a team?';
    RETURN OLD;
  ELSIF (TG_OP = 'UPDATE') THEN
    RAISE NOTICE 'Is a team finally moving to LA?';
    RETURN NEW;
  ELSIF (TG_OP = 'INSERT') THEN
    RAISE NOTICE '$_Move expansion? Isn't 32 teams enough?$_';
    RETURN NEW;
  END IF;
  RETURN NULL;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION teams_trigger()
  OWNER TO jim;
```


Debugging Triggers



Global Debugger

```
BEGIN
IF (TG_OP = 'DELETE') THEN
  RAISE NOTICE 'How is the NFL losing a team?';
  RETURN OLD;
ELSIF (TG_OP = 'UPDATE') THEN
  RAISE NOTICE 'Is a team finally moving to LA?';
  RETURN NEW;
ELSIF (TG_OP = 'INSERT') THEN
  RAISE NOTICE '$_Move expansion? Isn't 32 teams enough?$_';
  RETURN NEW;
END IF;
RETURN NULL;
END;
```

Stack pane

teams_trigger()@3

Output pane

Name	Type	Value
tg_name	name	teams_trigg
tg_when	text	AFTER
tg_level	text	ROW
tg_op	text	UPDATE
tg_relid	oid	20588
tg_relname	name	teams
tg_table_n	name	teams
tg_table_sc	name	public

Local Variables

Waiting for another session to invoke the target - "public.teams_trigger" Done

0:00:18.400 Ln 3 Col 1 Ch 8

Debugging: RAISE



```
nfl=# SELECT p.first_name, p.last_name,  
           p.position,  
           avg_yearly_score(p.player_key, 2006) AS score  
FROM player p  
WHERE p.player_key IN (SELECT *  
                       FROM yearly_player(2005))  
   AND avg_yearly_score(p.player_key, 2006) > 10  
ORDER BY 4 DESC;
```

Debugging: RAISE



```
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing TD Score: 8
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Interception Score: -2
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing Yards Score: 9
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing TD Score: 4
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Interception Score: 0
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing Yards Score: 10
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing TD Score: 8
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Interception Score: -4
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing Yards Score: 7
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Passing TD Score: 4
CONTEXT: PL/pgSQL function player_game_score(character varying,integer,integer) line 15 at assignment
PL/pgSQL function avg_yearly_score(character varying,integer) line 9 at assignment
NOTICE: Interception Score: -2
...
```

Track Functions



```
set track_functions = 'PL';
```

```
nfl=# SELECT * FROM pg_stat_user_functions;
```

funcid	schemaname	funcname	calls	total_time	self_time
20564	public	avg_yearly_score	547	7011.25	13.33
20565	public	passing_score	1666	1551.862	1551.862
20566	public	player_game_score	9299	6997.92	188.718
20567	public	receiving_score	4811	2465.982	2465.982
20568	public	rushing_score	4488	2303.934	2303.934
20569	public	td_score	9299	487.424	487.424
20570	public	yearly_player	1	14.139	14.139

```
(7 rows)
```


Profiler



```
https://bitbucket.org/openscg/plprofiler.git  
make USE_PGXS=1  
make install USE_PGXS=1
```

- OR -

```
http://www.bigsql.org/postgresql/installers.jsp
```

```
shared_preload_libraries =  
'$libdir/plprofiler.so'
```

```
CREATE EXTENSION plprofiler;
```

Profiler



```
$ plprofiler help
```

```
usage: plprofiler COMMAND [OPTIONS]
```

```
plprofiler is a command line tool to control the plprofiler extension  
for PostgreSQL.
```

```
The input of this utility are the call and execution statistics, the  
plprofiler extension collects. The final output is an HTML report of  
the statistics gathered. There are several ways to collect the data,  
save the data permanently and even transport it from a production  
system to a lab system for offline analysis.
```

Use

```
plprofiler COMMAND --help
```

```
for detailed information about one of the commands below.
```

Profiler



```
$ plprofiler help
```

```
...
```

```
GENERAL OPTIONS:
```

```
All commands implement the following command line options to specify the target database:
```

```
-h, --host=HOST      The host name of the database server.

-p, --port=PORT      The PostgreSQL port number.

-U, --user=USER      The PostgreSQL user name to connect as.

-d, --dbname=DB      The PostgreSQL database name or the DSN.
plprofiler currently uses psycopg2 to connect to the target database. Since that is based on libpq, all the above parameters can also be specified in this option with the usual conninfo string or URI formats.

--help               Print the command specific help information and exit.
```

Profiler



```
$ plprofiler help
```

```
...
```

```
TERMS:
```

The following terms are used in the text below and the help output of individual commands:

`in-memory-data` The plprofiler extension collects run-time data in per-backend hashtables (in-memory). This data is only accessible in the current session and is lost when the session ends or the hash tables are explicitly reset.

`collected-data` The plprofiler extension can copy the in-memory-data into global tables, to make the statistics available to other sessions. See the "monitor" command for details. This data relies on the local database's system catalog to resolve Oid values into object definitions.

`saved-dataset` The in-memory-data as well as the collected-data can be turned into a named, saved dataset. These sets can be exported and imported onto other machines. The saved datasets are independent of the system catalog, so a report can be generated again later, even even on a different system.

Profiler



```
$ plprofiler help
```

```
...
```

```
COMMANDS:
```

```
run          Runs one or more SQL statements with the plprofiler
             extension enabled and creates a saved-dataset and/or
             an HTML report from the in-memory-data.

monitor      Monitors a running application for a requested time
             and creates a saved-dataset and/or an HTML report from
             the resulting collected-data.

reset-data   Deletes the collected-data.

save         Saves the current collected-data as a saved-dataset.

list         Lists the available saved-datasets.

edit         Edits the metadata of one saved-dataset. The metadata
             is used in the generation of the HTML reports.

report       Generates an HTML report from either a saved-dataset
             or the collected-data.

delete       Deletes a saved-dataset.

export       Exports one or all saved-datasets into a JSON file.

import       Imports the saved-datasets from a JSON file, created
             with the export command.
```

Profiler



```
nfl=# SELECT p.first_name, p.last_name, p.position,  
nfl-#         avg_yearly_score(p.player_key, 2006) AS score  
nfl-# FROM player p  
nfl-# WHERE p.player_key IN (SELECT * FROM  
yearly_player(2005))  
nfl-# AND avg_yearly_score(p.player_key, 2006) > 10  
nfl-# ORDER BY 4 DESC;
```

first_name	last_name	position	score
LaDainian	Tomlinson	rb	22.5
Peyton	Manning	qb	18.875
Larry	Johnson	rb	17.875
Michael	Vick	qb	15.875
Drew	Brees	qb	15.75
Marc	Bulger	qb	15.5625
Steven	Jackson	rb	15.1875
Carson	Palmer	qb	15.125
Willie	Parker	rb	14.9375

Profiler



```
$ plprofiler run -h localhost -d nfl \
  -c "SELECT p.first_name, p.last_name, p.position, \
      avg_yearly_score(p.player_key, 2006) AS score \
      FROM player p WHERE p.player_key \
      IN (SELECT * \
          FROM yearly_player(2005)) \
      AND avg_yearly_score(p.player_key, 2006) > 10 \
      ORDER BY 4 DESC" \
--name="Run 1" --title="PGDay" \
--desc="PGDay Russia" \
--output="run1.html"
```

Profiler



PGDay

file:///home/jim/postgresql/pg951/bin/run1.html

PGDay Russia

PL/pgSQL Call Graph

PGDay

The call graph shows a sequence of function calls. The top call is `public.passing_score() oid=16386` (yellow bar). This is followed by `public.receiving_score() oid=16388` (orange bar), `public.rushing_score() oid=16389` (yellow bar), and `public.t.` (red bar). Below these are `public.player_game_score() oid=16387` (orange bar) and `public.avg_yearly_score() oid=16385` (red bar). The bars are stacked vertically, indicating a call sequence.

List of functions detailed below

- [public.avg_yearly_score\(\) oid=16385](#)
- [public.passing_score\(\) oid=16386](#)
- [public.player_game_score\(\) oid=16387](#)
- [public.receiving_score\(\) oid=16388](#)
- [public.rushing_score\(\) oid=16389](#)
- [public.td_score\(\) oid=16390](#)
- [public.yearly_player\(\) oid=16392](#)

All 7 functions (by self_time)

Function `public.receiving_score() oid=16388` [\(show\)](#)

self_time = 2,454,817 µs
total_time = 2,454,817 µs

```
public.receiving_score (p_player_key character varying,  
                       p_year integer,  
                       p_week integer)  
    RETURNS integer
```

Function `public.rushing_score() oid=16389` [\(show\)](#)

Profiler



PGDay x
file:///home/jim/postgresql/pg951/bin/run1.html#A16388

All 7 functions (by self_time)

Function public.receiving_score() oid=16388 (hide)

self_time = 2,454,817 µs
total_time = 2,454,817 µs

public.receiving_score (p_player_key character varying,
p_year integer,
p_week integer)
RETURNS integer

Line	exec_count	total time	longest time	Source Code
0	4,811	2,454,817 µs (100.00%)	1,296 µs	-- Function Totals
1	0	0 µs (0.00%)	0 µs	
2	0	0 µs (0.00%)	0 µs	DECLARE
3	0	0 µs (0.00%)	0 µs	score INT;
4	0	0 µs (0.00%)	0 µs	BEGIN
5	0	0 µs (0.00%)	0 µs	-- 1 point for every 10 yards receiving
6	4,811	2,442,218 µs (99.49%)	1,284 µs	SELECT r.yards/10
7	0	0 µs (0.00%)	0 µs	INTO score
8	0	0 µs (0.00%)	0 µs	FROM receiving r, games g
9	0	0 µs (0.00%)	0 µs	WHERE r.game_id = g.game_id
10	0	0 µs (0.00%)	0 µs	AND g.year = p_year
11	0	0 µs (0.00%)	0 µs	AND g.week = p_week
12	0	0 µs (0.00%)	0 µs	AND r.player_key = p_player_key;
13	0	0 µs (0.00%)	0 µs	
14	4,811	6,188 µs (0.25%)	32 µs	IF score IS NULL THEN
15	2,667	859 µs (0.03%)	12 µs	RETURN 0;
16	0	0 µs (0.00%)	0 µs	END IF;
17	0	0 µs (0.00%)	0 µs	
18	2,144	226 µs (0.01%)	1 µs	RETURN score;
19	0	0 µs (0.00%)	0 µs	END;
20	0	0 µs (0.00%)	0 µs	

Function public.rushing_score() oid=16389 (hide)

self_time = 2,254,588 µs
total_time = 2,254,588 µs

Real Use Case



PL Profiler Report for [redacted]-1-run-1

Domestic 1 segment, fails for Day/Time, took 1 minute on production.
Took 15 seconds on OpenSCG's [redacted].
Original schema from customer backup.

PL/pgSQL Call Graph

PL Profiler Report for [redacted]-1-run-1

Function: aff.get_new_sequence() oid=3584385247 (5,682,038 samples, 39.45%)

List of functions detailed below

- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)
- [aff. \[redacted\]](#)

Real Use Case



PL Profiler Report fo x

35	0	0 μs (0.00%)	0 μs	
36	99	1,672 μs (0.01%)	51 μs	
37	99	1,081 μs (0.01%)	51 μs	
38	0	0 μs (0.00%)	0 μs	
39	99	397 μs (0.00%)	28 μs	
40	0	0 μs (0.00%)	0 μs	
41	0	0 μs (0.00%)	0 μs	
42	0	0 μs (0.00%)	0 μs	
43	99	13,101,821 μs (99.94%)	702,573 μs	FOR i in 1..maxLevel LOOP
44	113	2,462 μs (0.02%)	217 μs	sql:='SELECT new seq number,
45	0	0 μs (0.00%)	0 μs	
46	0	0 μs (0.00%)	0 μs	
47	0	0 μs (0.00%)	0 μs	
48	0	0 μs (0.00%)	0 μs	
49	0	0 μs (0.00%)	0 μs	
50	0	0 μs (0.00%)	0 μs	
51	0	0 μs (0.00%)	0 μs	
52	0	0 μs (0.00%)	0 μs	
53	0	0 μs (0.00%)	0 μs	
54	0	0 μs (0.00%)	0 μs	
55	0	0 μs (0.00%)	0 μs	
56	113	146 μs (0.00%)	13 μs	
57	0	0 μs (0.00%)	0 μs	
58	0	0 μs (0.00%)	0 μs	
59	0	0 μs (0.00%)	0 μs	
60	0	0 μs (0.00%)	0 μs	--raise notice 'sql=%', sql;
61	113	13,097,663 μs (99.91%)	144,152 μs	EXECUTE sql INTO
62	113	748 μs (0.01%)	56 μs	
63	0	0 μs (0.00%)	0 μs	
64	0	0 μs (0.00%)	0 μs	
65	113	446 μs (0.00%)	87 μs	
66	19	66 μs (0.00%)	42 μs	
67	19	29 μs (0.00%)	9 μs	
68	19	33 μs (0.00%)	7 μs	
69	19	28 μs (0.00%)	7 μs	
70	19	46 μs (0.00%)	18 μs	
71	0	0 μs (0.00%)	0 μs	
72	94	16 μs (0.00%)	1 μs	
73	0	0 μs (0.00%)	0 μs	
74	0	0 μs (0.00%)	0 μs	
75	99	133 μs (0.00%)	8 μs	
76	99	422 μs (0.00%)	13 μs	

Real Use Case



PL Profiler Report for [redacted]-1-run-2

Domestic 1 segment, fails for Day/Time. took 1 minute on production.
Took 1.6 seconds on OpenSCG's [redacted].
Added Index [redacted]

PL/pgSQL Call Graph

The call graph is a heatmap where the vertical axis represents the call stack and the horizontal axis represents time. The color scale ranges from yellow (low time) to red (high time). The most significant activity is shown in the bottom-most layer, with several long, dark red horizontal bars. Labels for these bars include 'aff.check_r...', 'a.', 'aff.check_', and 'aff.ch...'. There are also some smaller, less intense yellow and orange bars in the upper layers of the call stack.

List of functions detailed below

- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]
- [aff.](#) [redacted]

Summary



- Be careful running these tools on production systems
 - They do have some performance impact
- Building the extensions are simple, but still require a development environment
- The debugger and the profiler use the same hooks so at the moment they can not be used at the same time



Questions?

jimm@openscg.com
[@jim_mlodgenski](https://twitter.com/jim_mlodgenski)