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


SMITH


EADS
PEUGEOT
Groupama
9
UNITED COLORS OF BENETTON
TERRA SERENA


6
2
7
3
11

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 VARCHR, 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 p1_player_key, p1_year, p1_week
               FROM get_rookie_backs_against_team_min_yards(p_team, p_yards)
    LOOP
        IF get_player_age_in_year(r.p1_player_key, r.p1_year) < l_age THEN
            l_age := get_player_age_in_year(r.p1_player_key, r.p1_year);
        END IF;
    END LOOP;

    FOR r IN SELECT p1_player_key, p1_year, p1_week
               FROM get_rookie_backs_against_team_min_yards(p_team, p_yards)
    LOOP
        IF get_player_age_in_year(r.p1_player_key, r.p1_year) = l_age THEN
              SELECT firstname, lastname
                INTO l_fn, l_ln
                FROM get_player_name(r.p1_player_key);

              first_name := l_fn;
              last_name := l_ln;
              p_year := r.p1_year;
              p_week := r.p1_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.


CRUZ

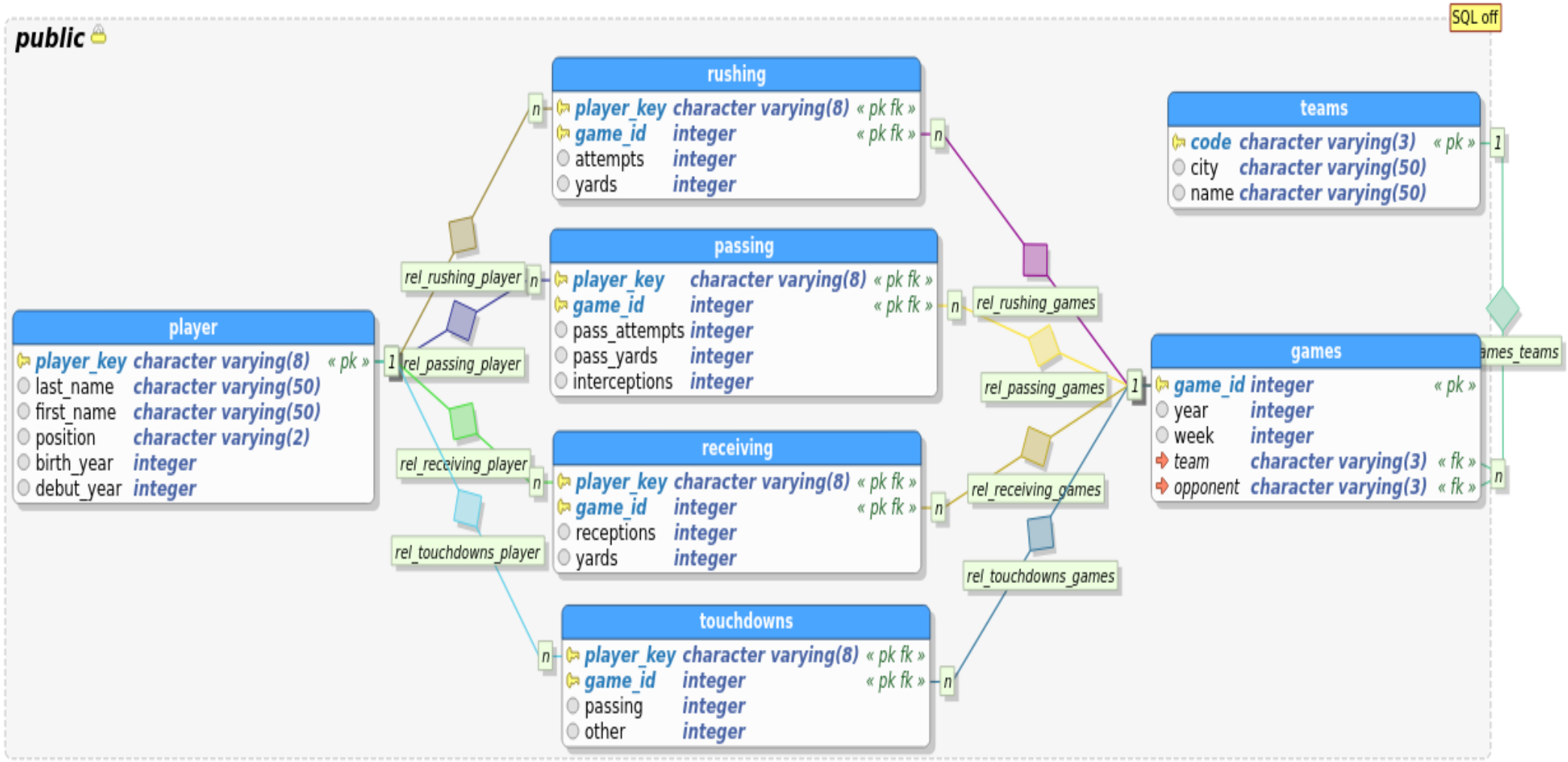
Business Rules

- 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



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;
```

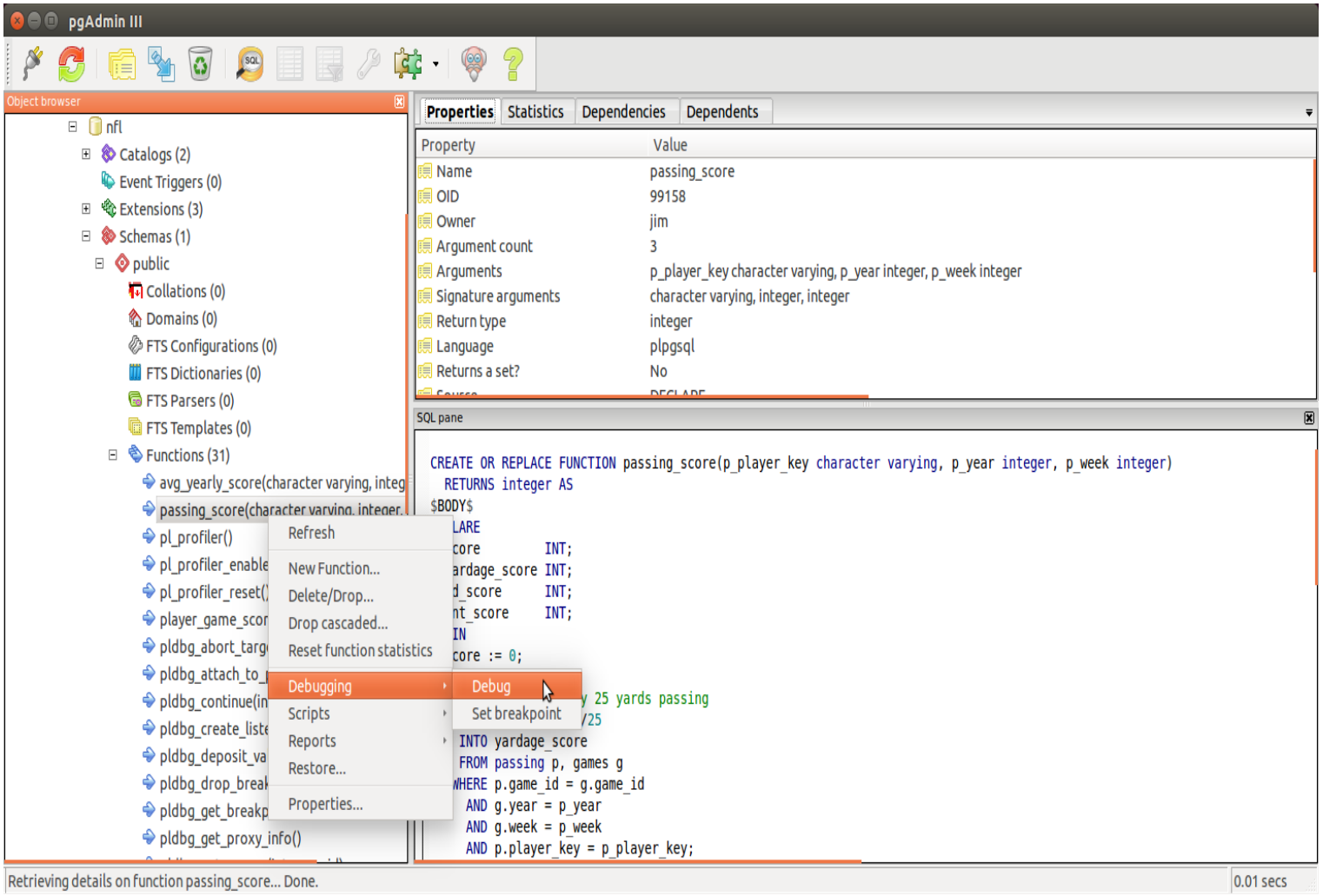
Debugging: RAISE

```
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

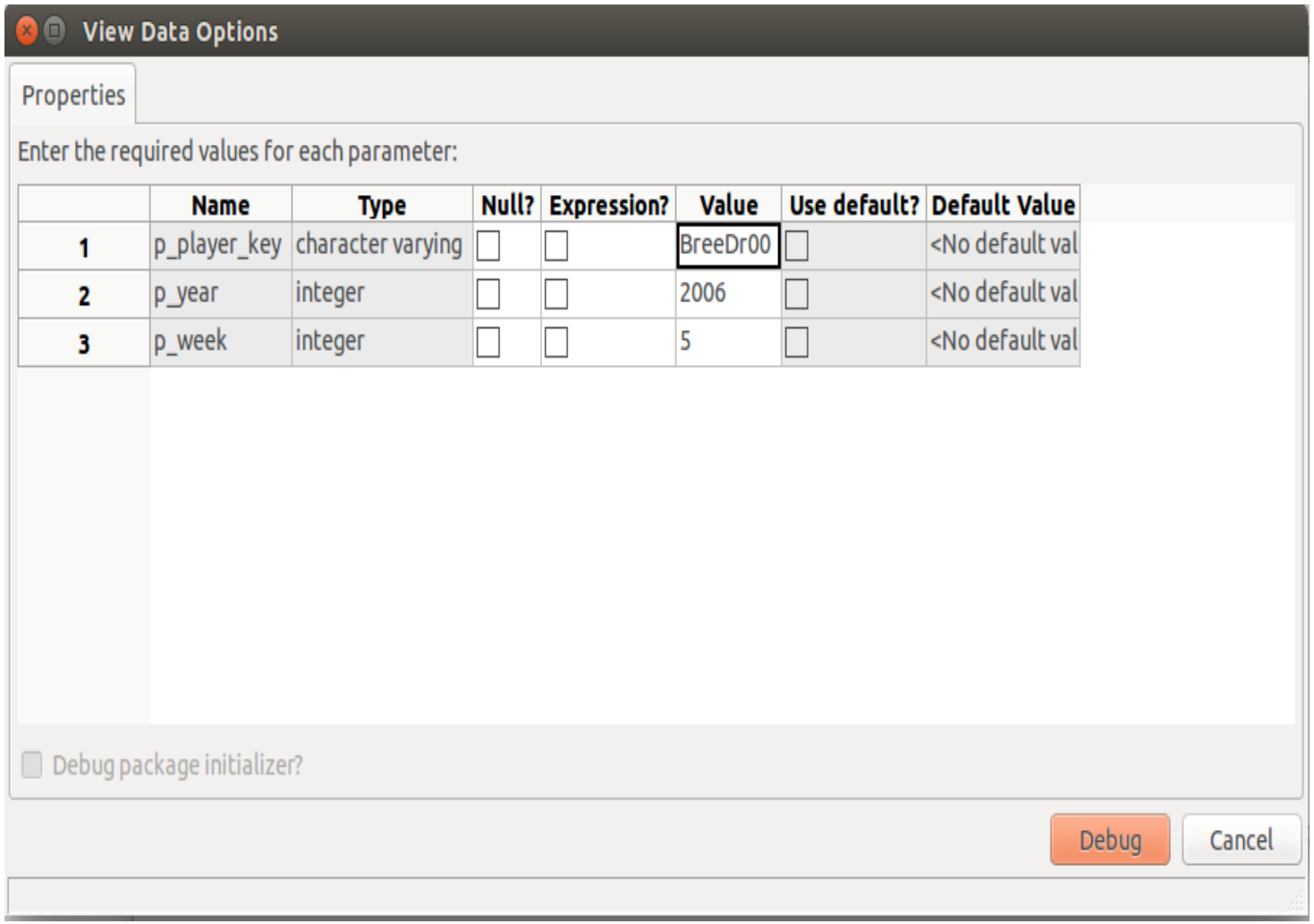
```
git://git.postgresql.org/git/pldebugger.git
make USE_PGXS=1
make install USE_PGXS=1

- OR -

www.bigsql.org/postgresql/installers.jsp

shared_preload_libraries =
'$libdir/plugin_debugger'

CREATE EXTENSION pldbgapi;
```

Debugger

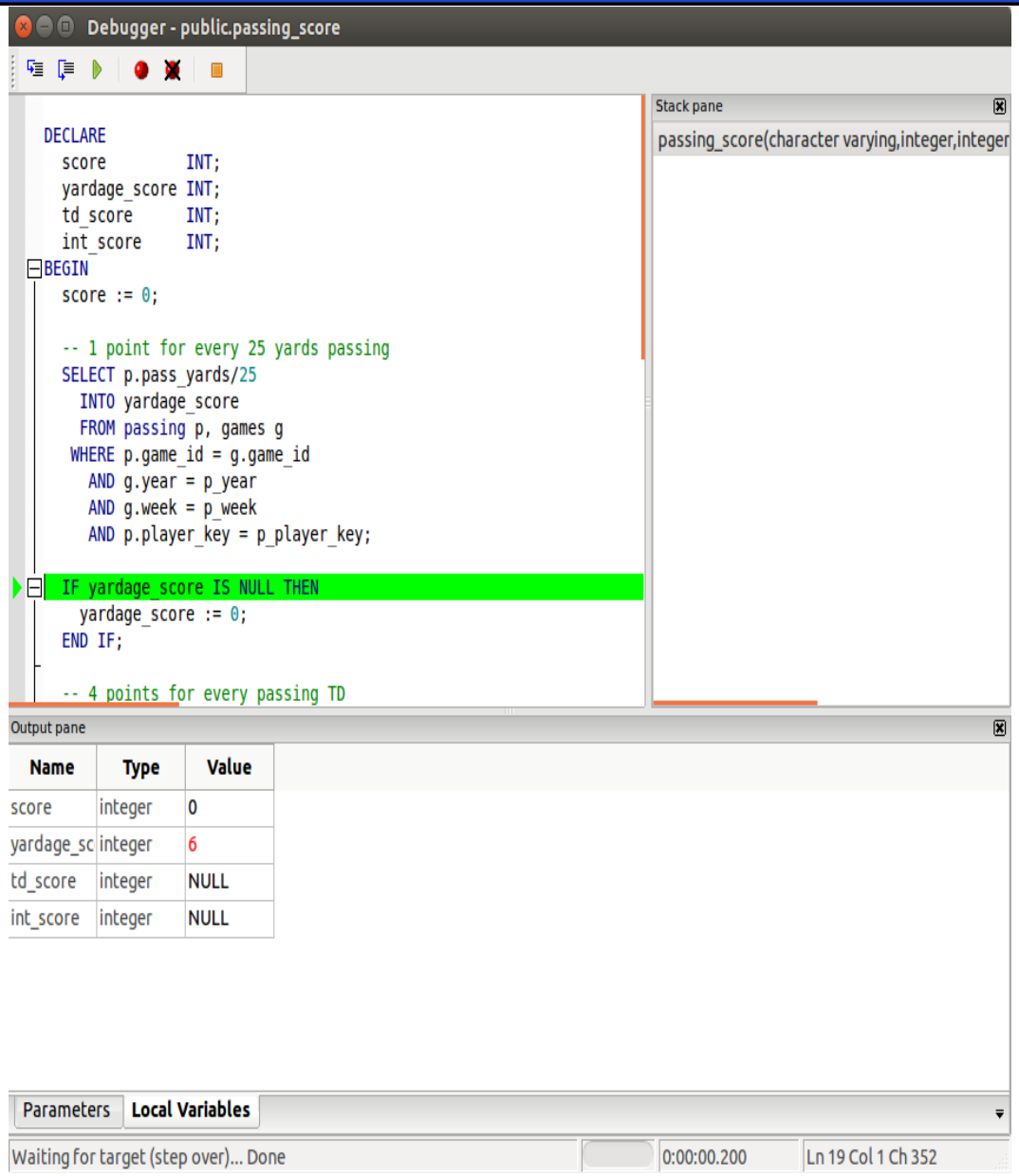


OpenSCG™

Debugger

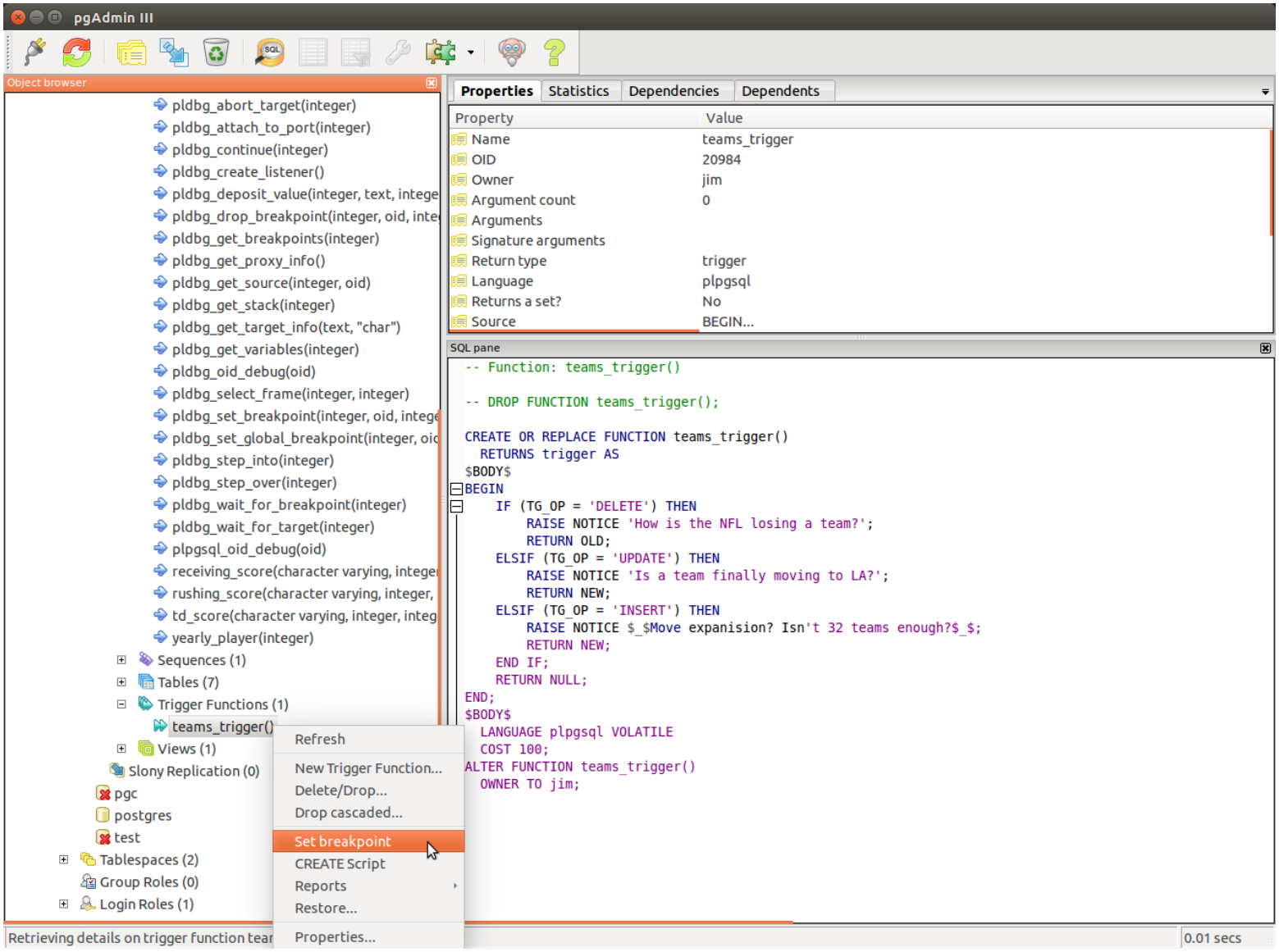

View Data Options
Properties
Enter the required values for each parameter:
Name
Type
Null?
Expression?
Value
Use default?
Default Value
1
p_player_key
character varying
BreeDr00
<No default val
2
p_year
integer
2006
<No default val
3
p_week
integer
5
<No default val
Debug package initializer?
Debug
Cancel

Debugger



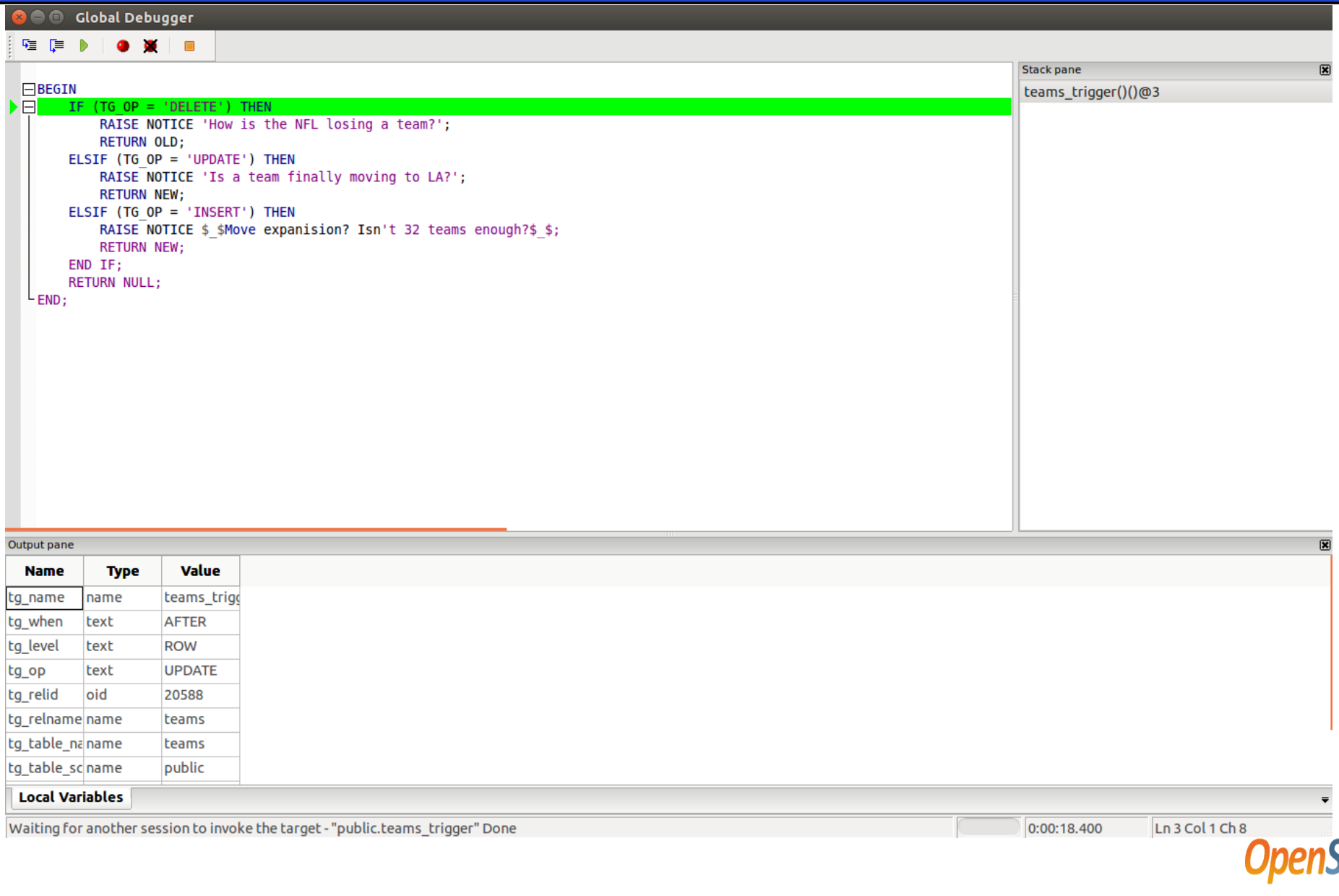
OpenSCG™

Debugging Triggers



OpenSCG™

Debugging Triggers



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.
```

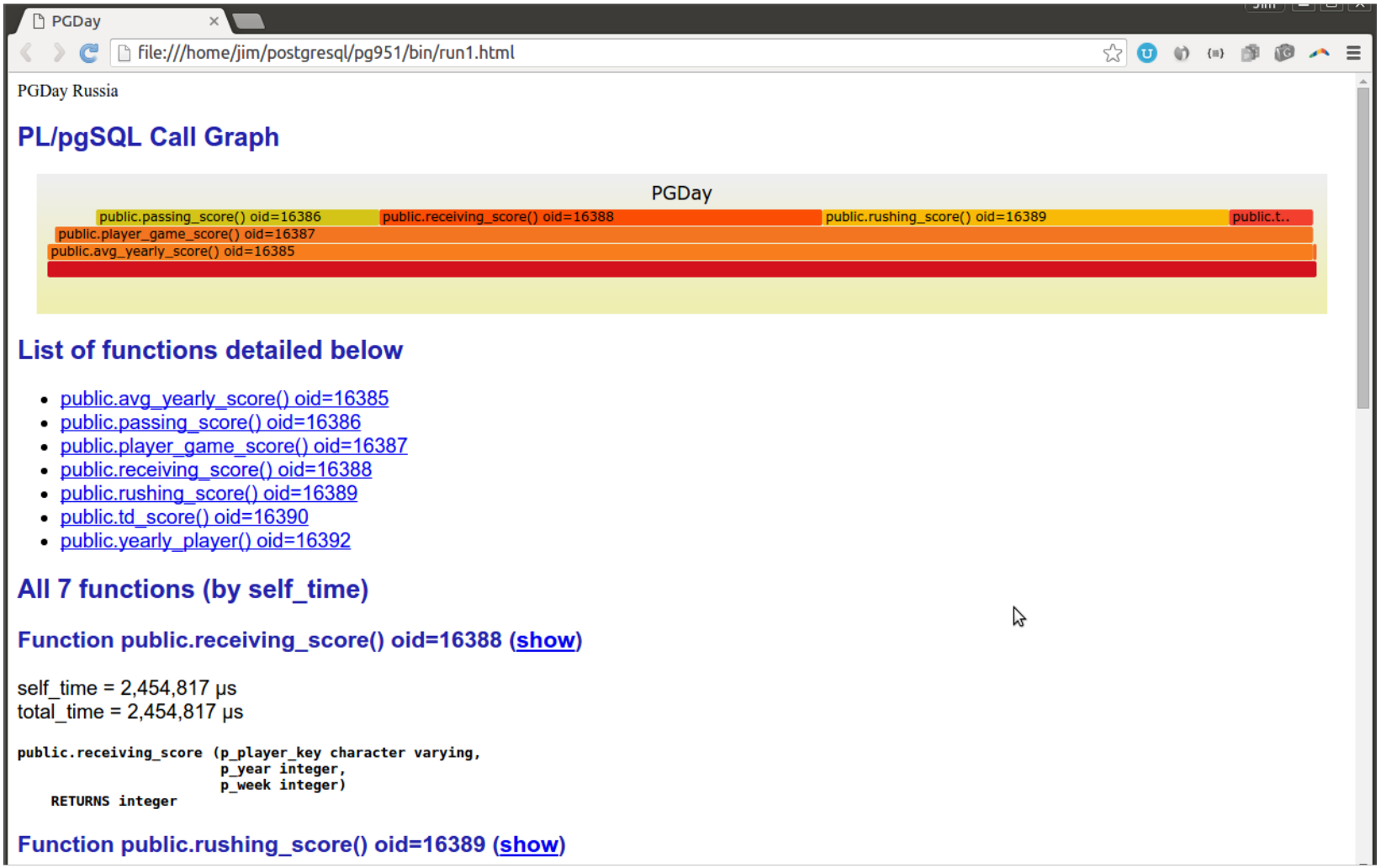
Profiler

```
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



OpenSCG™

Profiler

PGDay

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
```

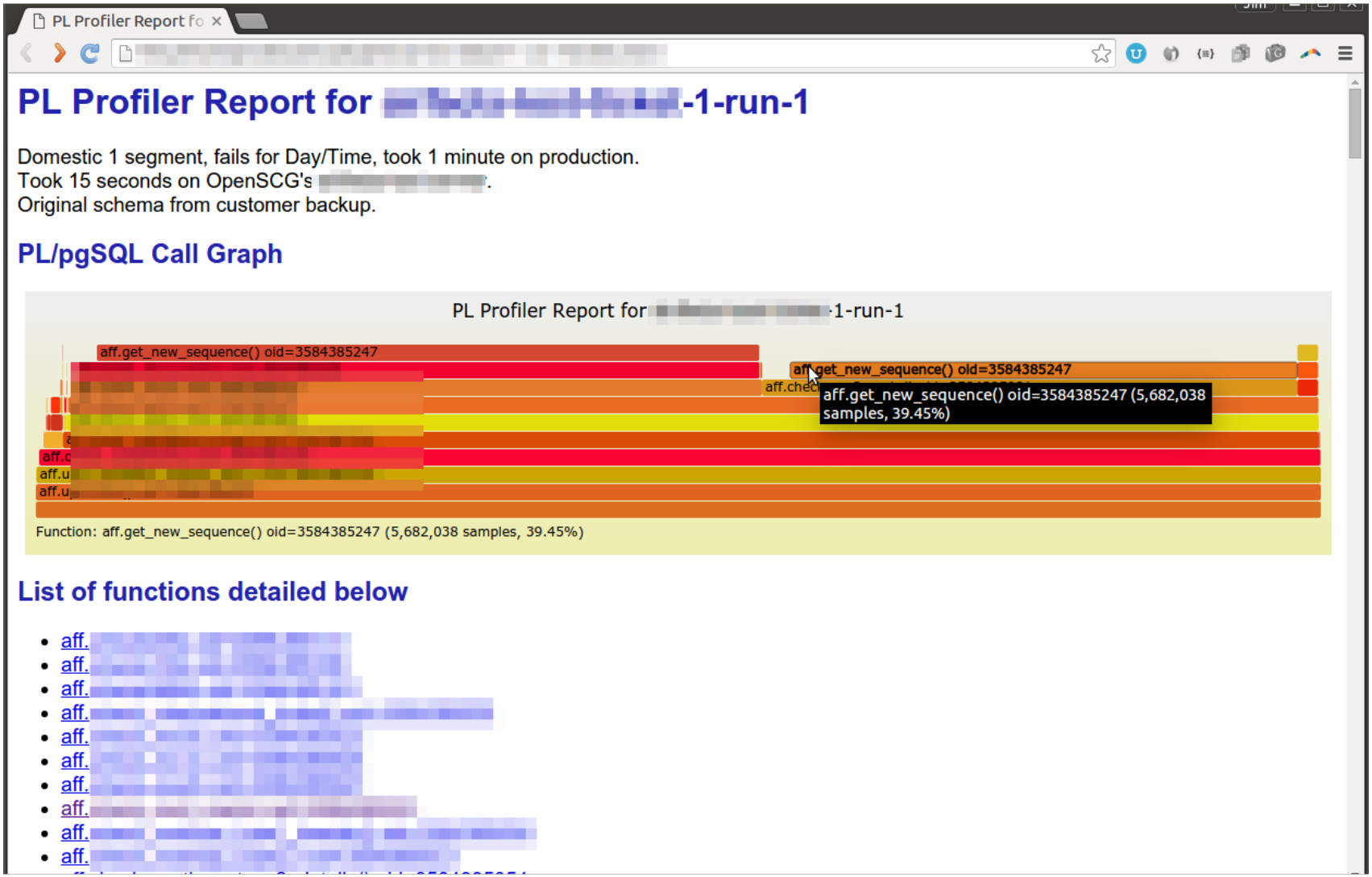
Line	exec_count	total_time	longest_time	Source Code
0	4,811	2,454,817 µs (100.00%)	1,296 µs	-- Function Totals
1	0	0 µs (0.00%)	0 µs	
2	0	0 µs (0.00%)	0 µs	DECLARE
3	0	0 µs (0.00%)	0 µs	score INT;
4	0	0 µs (0.00%)	0 µs	BEGIN
5	0	0 µs (0.00%)	0 µs	-- 1 point for every 10 yards receiving
6	4,811	2,442,218 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)

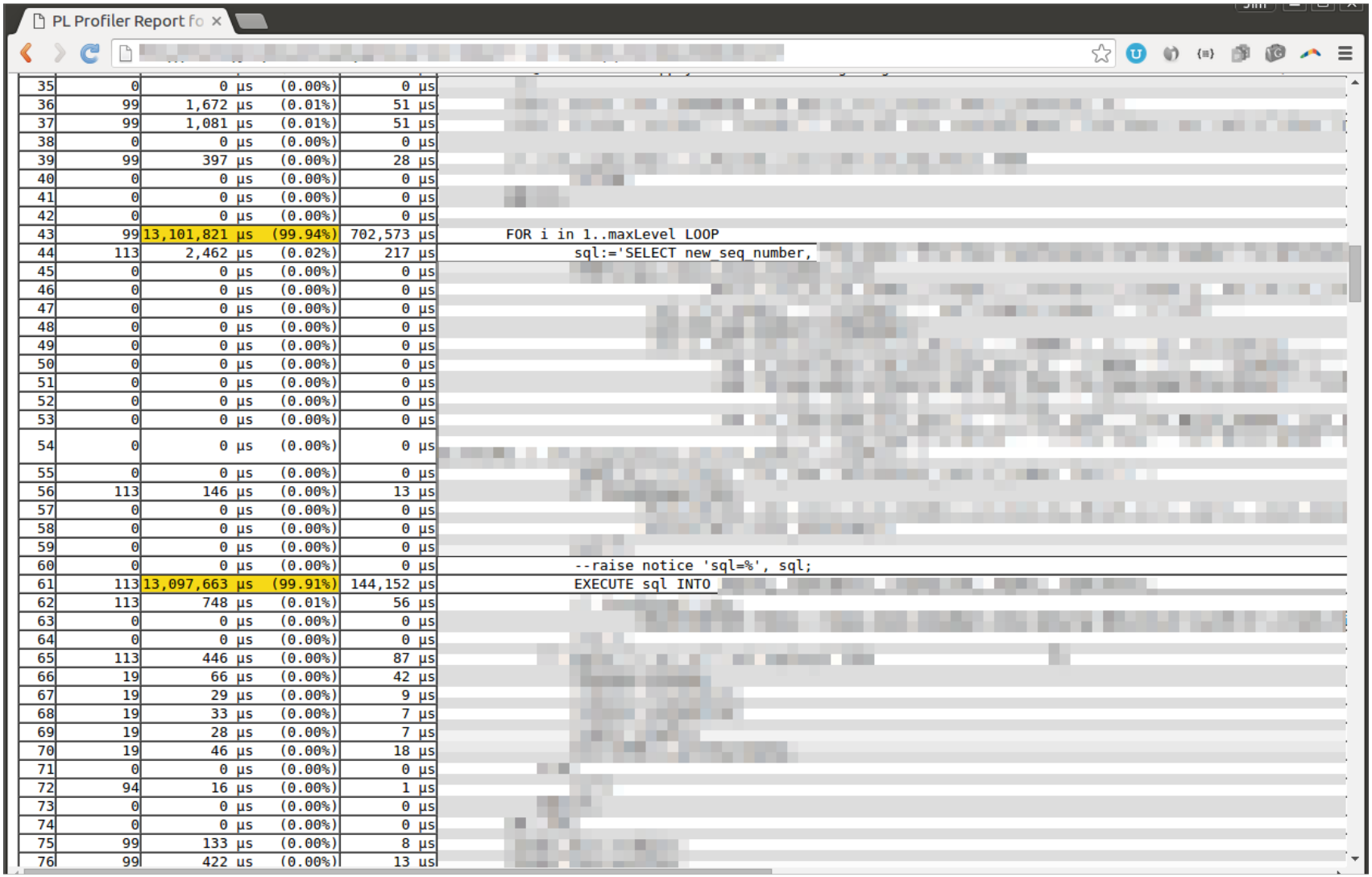
self_time = 2,254,588 µs
total_time = 2,254,588 µs

OpenSCG™

Real Use Case



Real Use Case



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

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