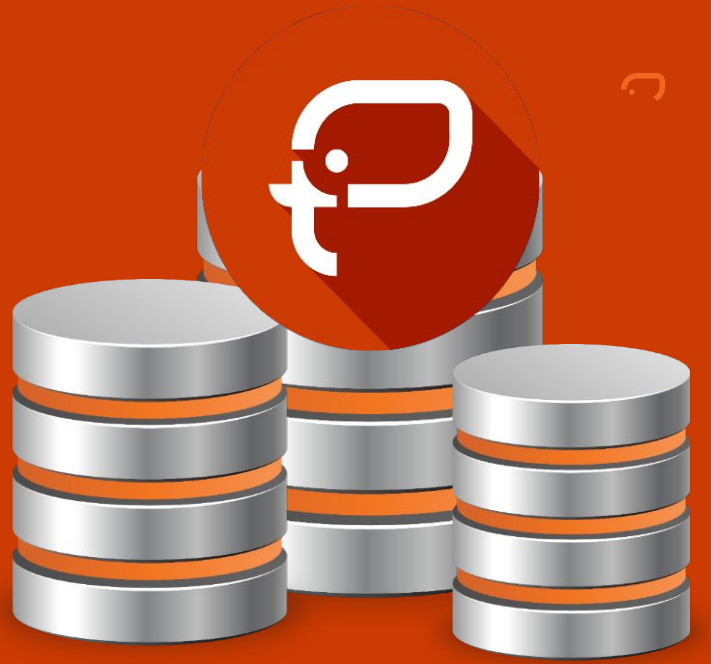
tantor



pg_stat_advisor - PostgreSQL advisor to create statistics

Ilia Evdokimov, Tantor Labs LLC

Contents

- autovacuum specific
- statistics
- extended statistics
- **pg_stat_advisor**

autovacuum features

Command	Free Space Map (FSM)	Visibility Map (VM)	Heap	Indexes	Tables in schema pg_catalog: pg_class	Tables in schema pg_catalog: pg_statistic
VACUUM	Updates	Updates	Removes dead tuples	Removes links to dead tuples	Update attributes: reltuples, relpages, relallvisible	Not directly affected
ANALYZE	Not affected	Not affected	Collects usage statistics	Collects usage statistics	Update attributes: reltuples, relpages. But not relallvisible.	Updates usage statistics for query plan optimization

GUC-parameters of autovacuum

```
SELECT name, short_desc FROM pg_settings
WHERE name = 'autovacuum_analyze_threshold'
   OR name = 'autovacuum_analyze_scale_factor';
>>>
              name                |              short_desc
----------------------------------+-------------------------------------
 autovacuum_analyze_scale_factor | Number of tuple inserts, updates, or
                 deletes prior to analyze as a fraction of reltuples.

 autovacuum_analyze_threshold    | Minimum number of tuple inserts,
                                    updates, or deletes prior to analyze.
(2 rows)
```

Prepare test table

Create table with disabled autovacuum to update statistics manually:

```
CREATE TABLE my_tbl(fld_1 INTEGER, fld_2 BIGINT)
WITH (autovacuum_enabled = false);
>>>
    CREATE TABLE
```

Evaluate data cardinality

Evaluate test data before INSERT:

```
SELECT
     COUNT(DISTINCT T.fld_1)::float/10000000,
     COUNT(DISTINCT T.fld_2)::float/10000000
FROM (
     SELECT i/100 AS fld_1, i/500 AS fld_2
     FROM generate_series(1, 10000000) s(i)
) T;
>>>
      0.01      0.002       <-- low cardinality
```

Insert values

Fill the table with diverse data:

```
INSERT INTO my_tbl (fld_1, fld_2)
SELECT
     i/100 as fld_1,
     i/500 as fld_2
FROM generate_series(1, 10000000) s(i);
>>>
     INSERT 0 1000000
```

Check basic statistics

Show attributes of my_tbl:

```
SELECT reltuples, relpages, relallvisible
FROM pg_class
WHERE relname = 'my_tbl';
>>>
    reltuples | relpages | relallvisible
    ----------+----------+---------------
           -1 |        0 |             0
    (1 row)
```

Refresh statistics

Show again attributes of my_tbl after ANALYZE:

```
ANALYZE VERBOSE my_tbl;
>>>
    INFO:  analyzing "public.my_tbl"

SELECT reltuples, relpages, relallvisible
FROM pg_class
WHERE relname = 'my_tbl';
>>>
    reltuples | relpages | relallvisible
    ----------+----------+--------------
        1e+06 |     5406 |             0
    (1 row)
```

VACUUM

Show attributes of my_tbl after VACUUM:

```
VACUUM VERBOSE my_tbl;
>>>
     INFO:  vacuuming "public.my_tbl"

SELECT reltuples, relpages, relallvisible
FROM pg_class
WHERE relname = 'my_tbl';
>>>
     reltuples | relpages | relallvisible
     ----------+----------+--------------
         1e+06 |     5406 |          5406
     (1 row)
```

default_statistics_target

Show current value:

```
SHOW default_statistics_target;
>>>
    default_statistics_target
   ---------------------------
    100                  <-- this is default value
   (1 row)
```

default_statistics_target - sets the default statistics target for table columns without a column-specific target set via ALTER TABLE SET STATISTICS. Larger values increase the time needed to do ANALYZE, but might improve the quality of the planner's estimates.

Statistics

```
\x
SELECT *, array_length(histogram_bounds, 1) as histogram_bounds_len
FROM pg_stats
WHERE tablename = 'my_tbl' AND attname = 'fld_1';
>>>
     -[ RECORD 1 ]----------+----------------------------------
     schemaname             | public
     tablename              | tbl
     attname                | fld_1
     null_frac              | 0
     avg_width              | 4
     n_distinct             | 1000
     most_common_vals       | {318,564,596,...}
     most_common_freqs      | {0.00173333,0.0017,0.00166667,...}
     histogram_bounds       | {0,8,20,30,39,...}
     ...
     histogram_bounds_len   | 101   <-- default_statistics_target
```

Extended statistics

dependencies	Evaluates functional dependencies between columns
ndistinct	Counts unique combinations of values in columns
mcv	Analyzes the most frequent value combinations in columns

```
CREATE STATISTICS stat_tbl (dependencies)
ON fld_1, fld_2
FROM my_tbl;
>>>
      CREATE STATISTICS
```

pg_stats_ext

There are no statistics before executing ANALYZE:

```
SELECT * FROM pg_stats_ext
WHERE statistics_name = 'stat_tbl';
>>>
     no rows
```

pg_stats_ext after ANALYZE

```
ANALYZE VERBOSE my_tbl;
>>>
       INFO:  analyzing "public.my_tbl"

SELECT * FROM pg_stats_ext
WHERE statistics_name = 'stat_tbl';
>>>
       "my_tbl"    "{fld_1,fld_2}"    "{""1 => 2"": 1.000000}"
```

Remove statistics for the upcoming examples:

```
DROP STATISTICS stat_tbl;
>>>
       DROP STATISTICS
```

Compare estimated and actual rows

For the best plan estimated rows should be similar with actual rows:

```
EXPLAIN ANALYZE
SELECT * FROM my_tbl
WHERE fld_1 = 500 AND fld_2 = 100;
>>>
     Gather  (cost=1000.00..12656.10 rows=1 width=12) (actual
time=5.148..57.646 rows=100 loops=1)"
       Workers Planned: 2"
       Workers Launched: 2"
       ->  Parallel Seq Scan on my_tbl  (cost=0.00..11656.00 rows=1
width=12) (actual time=21.646..36.275 rows=33 loops=3)"
             Filter: ((fld_1 = 500) AND (fld_2 = 100))"
             Rows Removed by Filter: 333300"
     Planning Time: 0.128 ms"
     Execution Time: 57.699 ms"
```

CREATE STATISTICS

Create and collect (refresh) statistics:

```
CREATE STATISTICS stat_tbl (dependencies)
ON fld_1, fld_2
FROM my_tbl;
>>>
     CREATE STATISTICS

ANALYZE VERBOSE my_tbl;
>>>
     INFO:  analyzing "public.my_tbl"
```

Estimated rows after creating extended statistics

Compare estimated and actual rows after CREATE STATISTICS:

```
EXPLAIN ANALYZE
SELECT * FROM my_tbl
WHERE fld_1 = 500 AND fld_2 = 100;
>>>
     Gather  (cost=1000.00..12666.00 rows=100 width=12) (actual
time=5.176..59.361 rows=100 loops=1)"
       Workers Planned: 2"
       Workers Launched: 2"
       ->  Parallel Seq Scan on my_tbl  (cost=0.00..11656.00 rows=42
width=12) (actual time=23.620..39.169 rows=33 loops=3)"
             Filter: ((fld_1 = 500) AND (fld_2 = 100))"
             Rows Removed by Filter: 333300"
     Planning Time: 0.165 ms"
     Execution Time: 59.404 ms"
```

About pg_stat_advisor

What is the Purpose of pg_stat_advisor?

- Advises on executing CREATE STATISTICS based on specific criteria.
- Suggests performing ANALYZE operations in particular scenarios.

Motivation Behind Creating pg_stat_advisor

- Faced challenges with complex SQL queries where precise row count estimations significantly impacted query performance.
- Belief in the extension's potential utility for others in the community.

How use pg_stat_advisor

Manual by command for current session:

```
LOAD 'pg_stat_advisor';
>>>
     LOAD
```

Or add library to postgresql.conf:

```
shared_preload_libraries = 'pg_stat_advisor'
```

Set extension parameters in postgresql.conf or current session:

- pg_stat_advisor.analyze_scale_factor
- pg_stat_advisor.suggest_statistics_threshold

Continue using of test table

Drop previously created extended statistics and add more data:

```
DROP STATISTICS stat_tbl;
>>>
     DROP STATISTICS

INSERT INTO my_tbl (fld_1, fld_2)
SELECT i/100 AS fld_1, i/500 AS fld_2
FROM generate_series(1000000, 1500000) s(i);
>>>
     INSERT 0 500001
```

Configure analyze_scale_factor

```
-- n_mod_since_analyze / n_live_tup > pg_stat_advisor.analyze_scale_factor
SET pg_stat_advisor.analyze_scale_factor = 0.01;
>>>
     SET

EXPLAIN ANALYZE SELECT * FROM my_tbl WHERE fld_1 = 500 AND fld_2 = 100;
>>>
     NOTICE:  pg_stat_advisor suggestion: 'ANALYZE my_tbl'
     Gather  (cost=1000.00..24309.94 rows=1 width=12) (actual
                time=101.366..108.701 rows=100 loops=1)"
           Workers Planned: 2"
           Workers Launched: 2"
           ->  Parallel Seq Scan on my_tbl  (cost=0.00..23309.84 rows=1
     width=12) (actual time=64.040..95.332 rows=33 loops=3)"
                 Filter: ((fld_1 = 500) AND (fld_2 = 100))"
                 Rows Removed by Filter: 666634"
          Planning Time: 0.272 ms"
          Execution Time: 114.312 ms"
```

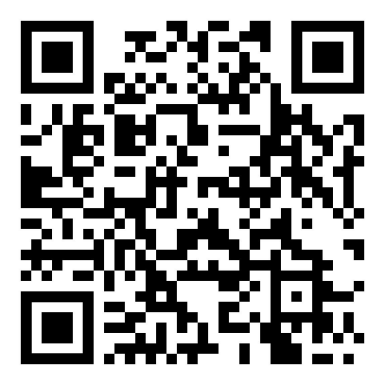
Configure suggest_statistics_threshold

```
-- estimated_row / actual rows < pg_stat_advisor.suggest_statistics_threshold
SET pg_stat_advisor.suggest_statistics_threshold = 0.01;
>>>
     SET

EXPLAIN ANALYZE SELECT * FROM my_tbl WHERE fld_1 = 500 AND fld_2 = 100;
>>>
     NOTICE:  pg_stat_advisor suggestion: CREATE STATISTICS
                  my_tbl_fld_1_fld_2_ON fld_1, fld_2 FROM my_tbl
    Gather  (cost=1000.00..24311.11 rows=1 width=12) (actual
    time=104.973..113.246 rows=100 loops=1)"
      Workers Planned: 2"
      Workers Launched: 2"
      ->  Parallel Seq Scan on my_tbl  (cost=0.00..23311.01 rows=1 width=12)
    (actual time=68.527..101.757 rows=33 loops=3)"
            Filter: ((fld_1 = 500) AND (fld_2 = 100))"
            Rows Removed by Filter: 666634"
    Planning Time: 0.109 ms"
    Execution Time: 116.551 ms"
```

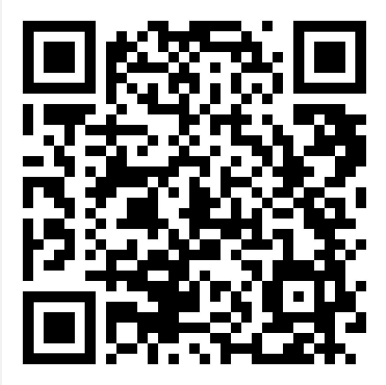
Results of applying recomendations

```
CREATE STATISTICS stat_tbl (dependencies)
ON fld_1, fld_2
FROM my_tbl;
>>>
     CREATE STATISTICS

ANALYZE VERBOSE my_tbl;
>>>
     INFO:  analyzing "public.my_tbl"

EXPLAIN ANALYZE
SELECT * FROM my_tbl
WHERE fld_1 = 500 AND fld_2 = 100;
>>>
     Gather  (cost=1000.00..24324.71 rows=100 width=12) (actual
          time=102.944..114.427 rows=100 loops=1)"
      ...
```

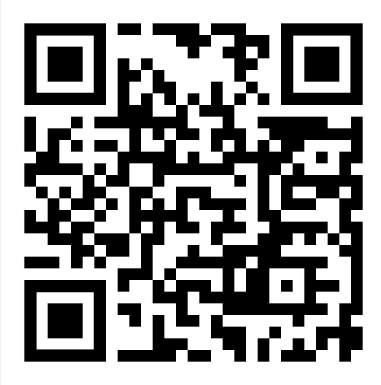
Thank You for Your Attention



ilia-evdokimov



EvdokimovIlia



ilidock95