

2ndQuadrant[®] 
PostgreSQL

CREATE STATISTICS

What is it for?

Tomas Vondra <tomas.vondra@2ndquadrant.com>



Agenda

- Quick intro into planning and estimates.
- Estimates with correlated columns.
- CREATE STATISTICS to the rescue!
 - functional dependencies
 - ndistinct
- Future improvements.



ZIP_CODES

```
CREATE TABLE zip_codes (  
    zip_code    INT PRIMARY KEY,  
    city        TEXT,  
    state       TEXT,  
    county      TEXT,  
    latitude    REAL,  
    longitude   REAL  
);
```

```
cat no_postal_codes_utf.csv | \  
psql test -c 'copy zip_codes from stdin \  
              with (format csv, header true)'
```

```
-- https://www.aggdata.com/free/norway-postal-codes
```



EXPLAIN

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE city = 'Oslo';
```

QUERY PLAN

```
-----
Seq Scan on zip_codes (cost=... rows=642 width=36)
      (actual rows=642 loops=1)
   Filter: (city = 'Oslo'::text)
  Rows Removed by Filter: 3932
Planning time: 0.158 ms
Execution time: 1.206 ms
(5 rows)
```



reltuples , relpages

```
SELECT reltuples, relpages
   FROM pg_class
  WHERE relname = 'zip_codes';
```

```
reltuples | relpages
-----+-----
      4574 |        40
(1 row)
```



```
SELECT * FROM pg_stats
  WHERE tablename = 'zip_codes' AND attname = 'city';
```

```
-----+-----
schemaname      | public
tablename       | zip_codes
attname         | city
...            | ...
most_common_vals | {Oslo, Trondheim, Bergen, ...}
most_common_freqs | {0.140359, 0.0301705, 0.0255794, ...}
...            | ...
```



```
EXPLAIN (ANALYZE, TIMING OFF)
SELECT * FROM zip_codes WHERE city = 'Oslo';
```

QUERY PLAN

```
-----
Seq Scan on zip_codes (cost=... rows=642 width=36)
      (actual rows=642 loops=1)
```

```
reltuples          | 4574
most_common_vals   | {Oslo, ...}
most_common_freqs  | {0.140359, ...}
```

$$4574 * 0.140359 = 642.002066$$



Underestimate

```
EXPLAIN (ANALYZE, TIMING off)
```

```
SELECT * FROM zip_codes WHERE city = 'Oslo' AND county = 'Oslo';
```

QUERY PLAN

```
-----  
Seq Scan on zip_codes (cost=0.00..108.61 rows=90 width=36)  
      (actual rows=642 loops=1)  
   Filter: ((city = 'Oslo'::text) AND (county = 'Oslo'::text))  
  Rows Removed by Filter: 3932  
Planning time: 0.276 ms  
Execution time: 1.962 ms  
(5 rows)
```




$$P(A \& B) = P(A) * P(B)$$



```
SELECT * FROM zip_codes  
    WHERE city = 'Oslo' AND county = 'Oslo';
```

```
P(city = 'Oslo' & county = 'Oslo')  
= P(city = 'Oslo') * P(county = 'Oslo')  
= 0.14 * 0.14  
= 0.0196
```

```
4574 * 0.0196 = 89.65
```



Overestimate

```
EXPLAIN (ANALYZE, TIMING off)
```

```
SELECT * FROM zip_codes WHERE city = 'Oslo' AND county != 'Oslo';
```

QUERY PLAN

```
-----  
Seq Scan on zip_codes (cost=0.00..108.61 rows=552 width=36)  
      (actual rows=0 loops=1)  
    Filter: ((city = 'Oslo'::text) AND (county != 'Oslo'::text))  
    Rows Removed by Filter: 4574  
Planning time: 0.180 ms  
Execution time: 1.470 ms  
(5 rows)
```



Correlated columns

- Attribute Value Independence Assumption (AVIA)
 - may result in wildly inaccurate estimates
 - both underestimates and overestimates
- consequences
 - poor scan choices (Seq Scan vs. Index Scan)
 - poor join choices (Nested Loop)



Poor scan choices

Index Scan using orders_city_idx on orders

(cost=0.28..185.10 **rows=90** width=36)

(actual **rows=12248237** loops=1)

Seq Scan using on orders

(cost=0.13..129385.10 **rows=12248237** width=36)

(actual **rows=90** loops=1)



Poor join choices

- > Nested Loop (... rows=90 ...) (... rows=12248237 ...)
- > Nested Loop (... rows=90 ...) (... rows=12248237 ...)
 - > Index Scan using orders_city_idx on orders
(cost=0.28..185.10 rows=90 width=36)
(actual rows=12248237 loops=1)
 - ...
 - > Index Scan ... (... loops=12248237)
- > Index Scan ... (... loops=12248237)
- > Index Scan ... (... loops=12248237)



functional dependencies (WHERE)



Functional Dependencies

- value in column A determines value in column B
- trivial example: primary key determines everything
 - zip code \rightarrow {city, county, state}
 - 1792 \rightarrow Tistedal \rightarrow Halden \rightarrow Ostfold
- other dependencies:
 - city \rightarrow county
 - county \rightarrow state



CREATE STATISTICS

```
CREATE STATISTICS s (dependencies)
  ON city, state, county FROM zip_codes;
      2       3       4
```

```
ANALYZE zip_codes;
```

```
SELECT stxdependencies FROM pg_statistic_ext WHERE stxname = 's';
```

stxdependencies

```
-----
{"2 => 3": 1.000000, "2 => 4": 0.985789,
 "3 => 2": 0.140359, "3 => 4": 0.140359,
 "4 => 2": 0.207040, "4 => 3": 0.995846,
 "2, 3 => 4": 0.985789,
 "2, 4 => 3": 1.000000,
 "3, 4 => 2": 0.207477}
(1 row)
```



`city` → `county`: 0.985789 = `d`

$$P(\text{city}='Oslo' \ \& \ \text{county}='Oslo')$$
$$= P(\text{city}='Oslo') * [d + (1-d) * P(\text{county}='Oslo')]$$
$$4574 * 0.14 * (0.986 + (1-0.986) * 0.14) = 633$$



Underestimate : fixed

```
EXPLAIN (ANALYZE, TIMING off)
```

```
SELECT * FROM zip_codes WHERE city = 'Oslo' AND county = 'Oslo';
```

QUERY PLAN

```
-----  
Seq Scan on zip_codes (cost=0.00..108.61 rows=634 width=36)  
      (actual rows=642 loops=1)
```

```
  Filter: ((city = 'Oslo'::text) AND (county = 'Oslo'::text))
```

```
  Rows Removed by Filter: 3932
```

```
  Planning time: 0.235 ms
```

```
  Execution time: 1.721 ms
```

```
(5 rows)
```



Overestimate #1: not fixed

```
SELECT * FROM zip_codes WHERE city = 'Oslo' AND county != 'Oslo';
```

QUERY PLAN

```
-----  
Seq Scan on zip_codes (cost=0.00..108.61 rows=552 width=36)  
    (actual rows=0 loops=1)  
    Filter: ((city = 'Oslo'::text) AND (county != 'Oslo'::text))  
    Rows Removed by Filter: 4574  
Planning time: 0.239 ms  
Execution time: 1.422 ms  
(5 rows)
```

Functional dependencies only work with equalities.



Overestimate #2: not fixed :-)

```
SELECT * FROM zip_codes WHERE city = 'Oslo' AND county = 'Halden';
```

QUERY PLAN

```
-----  
Seq Scan on zip_codes (cost=0.00..108.61 rows=633 width=36)  
    (actual rows=0 loops=1)  
    Filter: ((city = 'Oslo'::text) AND (county = 'Halden'::text))  
    Rows Removed by Filter: 4574  
Planning time: 0.253 ms  
Execution time: 1.279 ms  
(5 rows)
```

The queries need to respect the functional dependencies.



ndistinct (GROUP BY)



```
EXPLAIN (ANALYZE, TIMING off) SELECT 1 FROM zip_codes GROUP BY county;
```

QUERY PLAN

```
-----  
HashAggregate (cost=3086.60..3090.87 rows=427 width=11)  
  (actual rows=427 loops=1)
```

Group Key: county

```
-> Seq Scan on zip_codes (cost=0.00..2720.68 rows=146368 width=7)  
    (actual rows=146368 loops=1)
```

```
EXPLAIN (ANALYZE, TIMING off) SELECT 1 FROM zip_codes GROUP BY state;
```

QUERY PLAN

```
-----  
HashAggregate (cost=3086.60..3086.79 rows=19 width=13)  
  (actual rows=19 loops=1)
```

Group Key: state

```
-> Seq Scan on zip_codes (cost=0.00..2720.68 rows=146368 width=9)  
    (actual rows=146368 loops=1)
```



```
SELECT attname, n_distinct
       FROM pg_stats WHERE tablename = 'zip_codes';
```

attname	n_distinct
zip_code	-1
city	1826
county	427
state	19
longitude	2393
latitude	2341

(6 rows)



```
EXPLAIN (ANALYZE, TIMING off)
SELECT 1 FROM zip_codes GROUP BY state, county;
```

QUERY PLAN

```
-----
HashAggregate  (cost=3452.52..3533.65 rows=8113 width=20)
    (actual rows=429 loops=1)
    Group Key: state, county
    -> Seq Scan on zip_codes  (cost=0.00..2720.68 rows=146368 width=16)
        (actual rows=146368 loops=1)
```

Planning time: 0.162 ms

Execution time: 60.277 ms

(5 rows)



$\text{ndistinct}(\text{county, state}) = \text{ndistinct}(\text{county}) * \text{ndistinct}(\text{state})$

$427 * 19 = 8113$



```
CREATE STATISTICS s (ndistinct)
  ON county, state, city
  FROM zip_codes;

ANALYZE zip_codes;

SELECT stxndistinct FROM pg_statistic_ext;
```

stxndistinct

```
-----
{"2, 3": 1825, "2, 4": 1828,
 "3, 4": 429, "2, 3, 4": 1828}
(1 row)
```



```
EXPLAIN (ANALYZE, TIMING off)
SELECT 1 FROM zip_codes GROUP BY state, county;
```

QUERY PLAN

```
HashAggregate  (cost=3452.52..3456.81 rows=429 width=20)
  (actual rows=429 loops=1)
  Group Key: state, county
    -> Seq Scan on zip_codes  (cost=0.00..2720.68 rows=146368 width=16)
      (actual rows=146368 loops=1)
```

Planning time: 0.227 ms

Execution time: 58.386 ms

(5 rows)



ndistinct

- the “old behavior” was defensive
 - unreliable estimates with multiple columns
 - HashAggregate can’t spill to disk (OOM)
 - rather than crash do Sort+GroupAggregate (slow)
- ndistincts coefficients
 - make multi-column ndistinct estimates more reliable
 - reduced danger of OOM
 - large tables + GROUP BY multiple columns



Future Improvements

- additional types of statistics
 - MCV lists, histograms, ...
- statistics on expressions
 - currently only simple column references
 - alternative to functional indexes
- improving join estimates
 - using MCV lists
 - special multi-table statistics (syntax already supports it)



<https://2018.nordicpgday.org/feedback>

Tomas Vondra

tomas.vondra@2ndquadrant.com
tomas@pgaddict.com



[@fuzzycz](https://twitter.com/fuzzycz)



Questions?

Tomas Vondra

tomas.vondra@2ndquadrant.com
tomas@pgaddict.com

 [@fuzzycz](https://twitter.com/fuzzycz)