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
  - MCV lists
- Future improvements.

ZIP_CODES

```
CREATE TABLE zip_codes (
  postal_code       INT,
  place_name        VARCHAR(180),
  county_name       VARCHAR(100),
  county_code       INT,
  latitude          REAL,
  longitude         REAL
);

cat create-table.sql | psql test
cat zip-codes-czech.csv | psql test -c "copy zip_codes from stdin"

-- http://download.geonames.org/export/zip/
```

EXPLAIN

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

                                QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..41836.20 rows=169 width=37)
                        (actual rows=384 loops=1)
   Filter: ((place_name)::text = 'Michle'::text)
   Rows Removed by Filter: 1984512
 Planning Time: 0.552 ms
 Execution Time: 395.270 ms
(5 rows)
```

reltuples , relpages

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

  reltuples   | relpages
--------------+----------
 1.984896e+06 |    17025
```

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

------------------+---------------------------------
schemaname        | public
tablename         | zip_codes
attname           | place_name
...               | ...
most_common_vals  | {Lhota, ..., Michle, ...}
most_common_freqs | {0.0043, ..., 0.000086, ...}
...               | ...
```

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

                              QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..41836.20 rows=169 width=37)
                        (actual rows=384 loops=1)
   Filter: ((place_name)::text = 'Michle'::text)
   Rows Removed by Filter: 1984512

reltuples          | 1.984896e+06
most_common_vals   | {..., Michle, ...}
most_common_freqs  | {..., 0.0000851429, ...}

                  1.984896e+06 * 0.0005335 = 168.9998016384
```

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE county_name = 'Hlavní město Praha';

                                QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..41836.20 rows=39698 width=37)
                        (actual time=0.040..420.438 rows=40960 loops=1)
   Filter: ((county_name)::text = 'Hlavní město Praha'::text)
   Rows Removed by Filter: 1943936
```

```
reltuples          | 1.984896e+06
most_common_vals   | {..., Hlavní město Praha, ...}
most_common_freqs  | {..., 0.02, ...}
```

```
                    1.984896e+06 * 0.02 =  39697.92
```

Underestimate

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Michle'
                          AND county_name = 'Hlavní město Praha';

                                QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..46798.44 rows=3 width=37)
                        (actual rows=384 loops=1)
   Filter: (((place_name)::text = 'Michle'::text) AND
            ((county_name)::text = 'Hlavní město Praha'::text))
   Rows Removed by Filter: 1984512
```

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

```
SELECT * FROM zip_codes
        WHERE place_name = 'Michle'
          AND county_name = 'Hlavní město Praha';
```

```
P(place_name = 'Michle' & county_name = 'Hlavní město Praha')
  = P(place_name = 'Michle') * P(county_name = 'Hlavní město Praha')
  = 0.0000851429 * 0.02
  = 0.000001702858
```

```
0.000001702858 * 1.984896e+06 = 3.38
```

Underestimate

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Vinohrady'
                          AND province_name = 'Hlavní město Praha';

                           QUERY PLAN
-----------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..46798.44 rows=21 width=37)
                        (actual rows=768 loops=1)
   Filter: (((place_name)::text = 'Vinohrady'::text) AND
            ((county_name)::text = 'Hlavní město Praha'::text))
   Rows Removed by Filter: 1984128
```

Overestimate

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Michle'
                          AND county_name != 'Hlavní město Praha';

                                QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..46798.44 rows=165 width=37)
                        (actual rows=0 loops=1)
   Filter: (((county_name)::text <> 'Hlavní město Praha'::text) AND
            ((place_name)::text = 'Michle'::text))
   Rows Removed by Filter: 1984896
```

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 …)
    ->  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)
```

Poor join choices

```
->  Nested Loop (… rows=90 …) (… rows=12248237 …)
    ->  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)
->  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 → {place, county}
  - 14000 → {Michle, Hlavní město Praha}
- other dependencies:
  - place → county

CREATE STATISTICS

```
CREATE STATISTICS s (dependencies)
    ON place_name, county_name FROM zip_codes;
           2           3
ANALYZE zip_codes;

SELECT dependencies FROM pg_stats_ext WHERE statistics_name = 's';

     dependencies
----------------------
 {"2 => 3": 0.617530}
```

```
place → county: 0.617530 = d
```

```
P(place = 'Michle' & county = 'Hlavní město Praha') =
P(place = 'Michle') * [d + (1-d) * P(county = 'Hlavní město Praha')]
```

```
1.984896e+06 * 0.0000851429 * (0.617530 + (1.0 - 0.617530) * 0.02)
                                = 105.655
```

Underestimate : fixed

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Michle'
                          AND county_name = 'Hlavní město Praha';

                                QUERY PLAN
-------------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..46798.44 rows=105 width=37)
                        (actual rows=384 loops=1)
   Filter: (((place_name)::text = 'Michle'::text) AND
            ((county_name)::text = 'Hlavní město Praha'::text))
   Rows Removed by Filter: 1984512
```

Overestimate #1: not fixed :-(

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Michle'
                          AND county_name != 'Hlavní město Praha';

                                QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..46798.44 rows=165 width=37)
                        (actual rows=0 loops=1)
   Filter: (((county_name)::text <> 'Hlavní město Praha'::text) AND
            ((place_name)::text = 'Michle'::text))
   Rows Removed by Filter: 1984896
```

Functional dependencies only work with equalities.

Overestimate #2: not fixed :-(

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Michle'
                          AND county_name = 'Brno';

                           QUERY PLAN
-----------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..46798.44 rows=212 width=37)
                        (actual rows=0 loops=1)
   Filter: (((place_name)::text = 'Michle'::text) AND
            ((county_name)::text = 'Brno'::text))
   Rows Removed by Filter: 1984896
```

The queries need to respect the functional dependencies.

ndistinct (GROUP BY)

```
EXPLAIN (ANALYZE, TIMING off)
SELECT count(*) FROM zip_codes GROUP BY county_name;

                                   QUERY PLAN
-------------------------------------------------------------------------------
 HashAggregate  (cost=46798.44..46799.21 rows=77 width=19)
                (actual rows=77 loops=1)
   Group Key: county_name
   ->  Seq Scan on zip_codes  (cost=0.00..36873.96 rows=1984896 width=11)
                              (actual rows=1984896 loops=1)
 Planning Time: 0.152 ms
 Execution Time: 999.121 ms
(5 rows)
```

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

  attname    | n_distinct
-------------+------------
 county_code |         77
 postal_code |       2694
 place_name  |      11283
 county_name |         77
 longitude   |       1020
 latitude    |        748
(7 rows)
```

```
EXPLAIN (ANALYZE, TIMING off)
SELECT count(*) FROM zip_codes GROUP BY county_name, postal_code;

                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 GroupAggregate  (cost=312344.35..334178.21 rows=198490 width=23)
                 (actual rows=2770 loops=1)
   Group Key: county_name, postal_code
   ->  Sort  (cost=312344.35..317306.59 rows=1984896 width=15)
             (actual rows=1984896 loops=1)
         Sort Key: county_name, postal_code
         Sort Method: external merge  Disk: 51768kB
         ->  Seq Scan on zip_codes  (cost=0.00..36873.96 rows=1984896 width=15)
                                    (actual rows=1984896 loops=1)
 Planning Time: 26.155 ms
 Execution Time: 3711.060 ms
```

```
EXPLAIN (ANALYZE, TIMING off)
SELECT count(*) FROM zip_codes GROUP BY county_name, postal_code;

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=312344.35..334178.21 rows=198490 width=23)
                 (actual rows=2770 loops=1)
   Group Key: county_name, postal_code
   ->  Sort  (cost=312344.35..317306.59 rows=1984896 width=15)
             (actual rows=1984896 loops=1)
         Sort Key: county_name, postal_code
         Sort Method: external merge  Disk: 51768kB
         ->  Seq Scan on zip_codes  (cost=0.00..36873.96 rows=1984896 width=15)
                                    (actual rows=1984896 loops=1)
 Planning Time: 26.155 ms
 Execution Time: 3711.060 ms
```

```
ndistinct(county, zip)

=

ndistinct(county) * ndistinct(zip)
```

```
77 * 2694 = 207438
```

```
ndistinct(county, zip)
=
ndistinct(county) * ndistinct(zip)
```

```
77 * 2694 = 207438
```

(capped to 10% of the table)

```
CREATE STATISTICS s (ndistinct)
    ON place_name, province_name, state_name
  FROM zip_codes;

ANALYZE zip_codes;

SELECT n_distinct FROM pg_stats_ext WHERE statistics_name = 's';

   n_distinct
---------------
 {"1, 3": 2770}
```

```
EXPLAIN (ANALYZE, TIMING off)
SELECT count(*) FROM zip_codes GROUP BY county_name, postal_code;

                                  QUERY PLAN
-------------------------------------------------------------------------------
 HashAggregate  (cost=51760.68..51788.38 rows=2770 width=23)
                (actual rows=2770 loops=1)
   Group Key: county_name, postal_code
   ->  Seq Scan on zip_codes  (cost=0.00..36873.96 rows=1984896 width=15)
                              (actual rows=1984896 loops=1)
 Planning Time: 0.642 ms
 Execution Time: 1149.229 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)
- ndistinct 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 (PG12), 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)

2ndQuadrant PostgreSQL

Questions?

Tomas Vondra

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

@fuzzycz