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 (
  country_code      VARCHAR(2),
  postal_code       VARCHAR(20),
  place_name        VARCHAR(180),
  state_name        VARCHAR(100),
  state_code        VARCHAR(20),
  ...
);
cat create.sql | psql test
cat zip-codes-germany.csv | \
    psql test -c "copy zip_codes from stdin \
                  with (format csv, header true, \
                        delimiter E'\t')"
-- http://download.geonames.org/export/zip/
```

EXPLAIN

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

                             QUERY PLAN
------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..57509.07 rows=22858 width=95)
                        (actual rows=23296 loops=1)
   Filter: ((place_name)::text = 'Berlin'::text)
   Rows Removed by Filter: 2086144
 Planning Time: 0.289 ms
 Execution Time: 203.923 ms
```

reltuples , relpages

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

  reltuples  | relpages
-------------+----------
 2.10993e+06 |    31135
(1 row)
```

```
SELECT * FROM pg_stats
 WHERE tablename = 'zip_codes'
   AND attname = 'place_name';
------------------+---------------------------------
schemaname        | public
tablename         | zip_codes
attname           | place_name
...               | ...
most_common_vals  | {Berlin, Hamburg, München, Köln, ...}
most_common_freqs | {0.0110, 0.0061, 0.0045, 0.0027, ...}
...               | ...
```

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

                              QUERY PLAN
------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..57509.07 rows=22858 width=95)
                        (actual rows=23296 loops=1)

reltuples         | 16480
most_common_vals  | {Berlin,…}
most_common_freqs | {0.0.0108333,…}

                  2.10993e+06 * 0.0108333 = 22857.5046690
```

Underestimate

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

                           QUERY PLAN
----------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..62783.89 rows=263 width=95)
                        (actual rows=23296 loops=1)
   Filter: (((place_name)::text = 'Berlin'::text)
        AND ((state_name)::text = 'Berlin'::text))
   Rows Removed by Filter: 2086144
 Planning Time: 0.157 ms
 Execution Time: 220.953 ms
```

```
P (A & B) = P(A) * P(B)
```

```
SELECT * FROM zip_codes
        WHERE place_name = 'Berlin'
          AND state_name = 'Berlin';
```

```
  P(place_name = 'Berlin' & state_name = 'Berlin')
    = P(city = 'Berlin') * P(state_name = 'Berlin')
    = 0.0108333 * 0.0115
    = 0.00012458295
```

```
              2.10993e+06 * 0.00012458295 = 262.86
```

Underestimate

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

                         QUERY PLAN
------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..62783.89 rows=263 width=95)
                        (actual rows=23296 loops=1)
   Filter: (((place_name)::text = 'Berlin'::text)
        AND ((state_name)::text = 'Berlin'::text))
   Rows Removed by Filter: 2086144
 Planning Time: 0.174 ms
 Execution Time: 217.859 ms
```

Overestimate

```
EXPLAIN (ANALYZE, TIMING off)
SELECT * FROM zip_codes WHERE place_name = 'Berlin'
                          AND state_name != 'Berlin';

                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..62783.89 rows=22595 width=95)
                        (actual rows=0 loops=1)
   Filter: (((state_name)::text <> 'Berlin'::text)
        AND ((place_name)::text = 'Berlin'::text))
   Rows Removed by Filter: 2109440
 Planning Time: 0.143 ms
 Execution Time: 232.766 ms
```

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, community, county, state}
  - 89346 → {Bibertal, Landkreis Günzburg, Swabia, Bayern}
- other dependencies:
  - place → community
  - community → county
  - county → state

CREATE STATISTICS

```
CREATE STATISTICS s (dependencies)
    ON place_name, state_name, county_name FROM zip_codes;
            3           4            6
ANALYZE zip_codes;
SELECT stxdependencies FROM pg_statistic_ext WHERE stxname = 's';

               stxdependencies
------------------------------------------
 {"3 => 4": 0.918083, "3 => 6": 0.954369,
  "4 => 6": 0.689745, "6 => 4": 0.310255,
  "3, 4 => 6": 0.994842,
  "3, 6 => 4": 0.954672}
(1 row)
```

```
place → state: 0.918083 = d

P(place = 'Berlin' & state = 'Berlin') =
P(place = 'Berlin') * [d + (1-d) * P(state = 'Berlin')]

16480 * 0.011 * (0.918 + (1-0.918) * 0.012) = 166.6
```

Underestimate : fixed

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

                           QUERY PLAN
----------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..62783.89 rows=263 width=95)
                        (actual rows=23296 loops=1)
   Filter: (((place_name)::text = 'Berlin'::text) AND
((state_name)::text = 'Berlin'::text))
   Rows Removed by Filter: 2086144
 Planning Time: 0.159 ms
 Execution Time: 213.455 ms
```

Overestimate #1: not fixed :-(

```
SELECT * FROM zip_codes WHERE place_name = 'Berlin'
                          AND state_name != 'Berlin';

                              QUERY PLAN
------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..62783.89 rows=22595 width=95)
                        (actual rows=0 loops=1)
   Filter: (((state_name)::text <> 'Berlin'::text)
        AND ((place_name)::text = 'Berlin'::text))
   Rows Removed by Filter: 2109440
 Planning Time: 0.152 ms
 Execution Time: 240.476 ms
```

Functional dependencies only work with equalities.

Overestimate #2: not fixed :-(

```
SELECT * FROM zip_codes WHERE place_name = 'Berlin'
                          AND state_name = 'Bayern';

                           QUERY PLAN
-----------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..62783.89 rows=3103 width=95)
                        (actual rows=0 loops=1)
   Filter: (((place_name)::text = 'Berlin'::text)
        AND ((state_name)::text = 'Bayern'::text))
   Rows Removed by Filter: 2109440
 Planning Time: 0.000 ms
 Execution Time: 206.455 ms
```

The queries need to respect the functional dependencies.

ndistinct
(GROUP BY)

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

                              QUERY PLAN
--------------------------------------------------------------------------
 HashAggregate  (cost=57509.08..57513.07 rows=399 width=23)
                (actual rows=400 loops=1)
   Group Key: community_name
   ->  Seq Scan on zip_codes  (cost=0.00..52234.26 rows=2109926 width=19)
                              (actual rows=2109440 loops=1)
 Planning Time: 0.087 ms
 Execution Time: 445.727 ms
```

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

    attname     | n_distinct
----------------+------------
 country_code   |          1
 postal_code    |       7798
 place_name     |      13326
 state_name     |         16
 stat_code      |         16
 county_name    |         19
 county_code    |         20
 community_name |        399
 community_code |        401
 ...            |        ...
(12 rows)
```

```
EXPLAIN (ANALYZE, TIMING off)
SELECT 1 FROM zip_codes GROUP BY state_name, county_name, community_name;


                                    QUERY PLAN
-------------------------------------------------------------------------------
 Group  (cost=418019.55..439114.20 rows=121296 width=55)
        (actual rows=400 loops=1)
   Group Key: state_name, county_name, community_name
   ->  Sort  (cost=418019.55..423293.22 rows=2109465 width=51)
             (actual rows=2109440 loops=1)
         Sort Key: state_name, county_name, community_name
         Sort Method: external merge  Disk: 102160kB
         ->  Seq Scan on zip_codes (cost=0.00..52229.65 rows=2109465 width=51)
                                   (actual rows=2109440 loops=1)
 Planning Time: 0.276 ms
 Execution Time: 3100.593 ms
(8 rows)
```

```
ndistinct(state, county, community)

=

ndistinct(state) * ndistinct(county) * ndistinct(community)


16 * 19 * 399 = 121296
```

```
CREATE STATISTICS s (ndistinct)
    ON state_name, county_name, community_name
  FROM zip_codes;
ANALYZE zip_codes;

SELECT stxndistinct FROM pg_statistic_ext;

          stxndistinct
--------------------------------
 {"4, 6": 31, "4, 8": 399,
  "6, 8": 399, "4, 6, 8": 397}
(1 row)
```

```
EXPLAIN (ANALYZE, TIMING off)
 SELECT 1 FROM zip_codes GROUP BY state_name, county_name, community_name;

                               QUERY PLAN
--------------------------------------------------------------------------
 HashAggregate  (cost=68051.18..68055.15 rows=397 width=55)
                (actual rows=400 loops=1)
   Group Key: community_name, state_name, county_name
   ->  Seq Scan on zip_codes  (cost=0.00..52229.96 rows=2109496 width=51)
                              (actual rows=2109440 loops=1)
 Planning Time: 0.243 ms
 Execution Time: 656.465 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)
- ndistincs 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)

Questions?

Tomas Vondra

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

@fuzzycz

Questions?

Tomas Vondra

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

🐦 **@fuzzycz**