



# CREATE STATISTICS

## What is it for?

Tomas Vondra, 2ndQuadrant  
[tomas.vondra@2ndquadrant.com](mailto:tomas.vondra@2ndquadrant.com)  
PGCon 2020, May 26-29

# Agenda

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

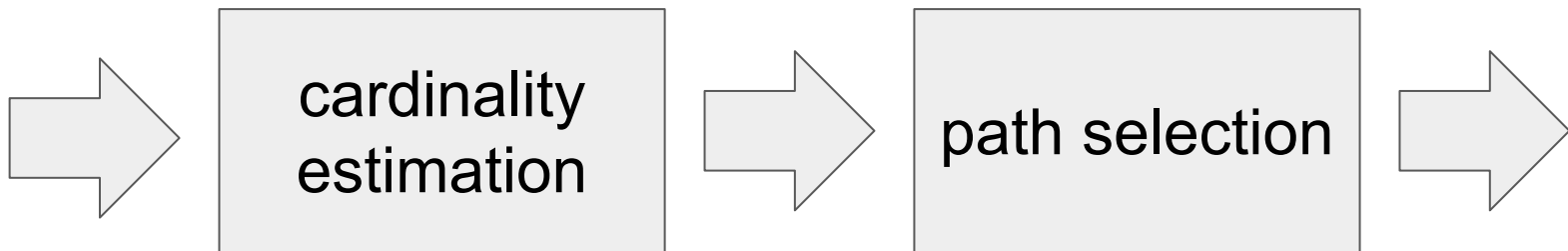
# ZIP\_CODES

```
CREATE TABLE zip_codes (  
    postal_code      VARCHAR(20),  
    place_name       VARCHAR(180),  
    state_name       VARCHAR(100),  
    county_name      VARCHAR(100),  
    community_name   VARCHAR(100),  
    latitude         REAL,  
    longitude        REAL  
);
```

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

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

# Why should you care?



# EXPLAIN

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

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..42175.91 rows=14028 width=67)
    (actual rows=13889 loops=1)
    Filter: ((place_name)::text = 'Manchester'::text)
    Rows Removed by Filter: 1683064
Planning Time: 0.113 ms
Execution Time: 151.340 ms
(5 rows)
```

# relpages, reltuples

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

reltuples		relpages
-----+-----		
1.696953e+06		20964

# pg\_stats

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

```
-----+-----
schemaname      | public
tablename       | zip_codes
attname         | place_name
...             | ...
most_common_vals | {London, Birmingham, Glasgow, Manchester, ...}
most_common_freqs | {0.1012, 0.012433333, 0.009966667, 0.0082665813, ...}
...             | ...
```

```
SELECT * FROM zip_codes WHERE place_name = 'Manchester';
```

#### QUERY PLAN

```
-----  
Seq Scan on zip_codes  (cost=0.00..42175.91 rows=14028 width=67)  
    (actual rows=13889 loops=1)  
    Filter: ((place_name)::text = 'Manchester'::text)  
    Rows Removed by Filter: 1683064
```

```
reltuples           | 1.696953e+06  
most_common_vals    | {..., Manchester, ...}  
most_common_freqs   | {..., 0.0082665813, ...}
```

$1.696953e+06 * 0.0082665813 = 14027.9999$



```
SELECT * FROM zip_codes WHERE community_name = 'Manchester';
```

### QUERY PLAN

```
-----  
Seq Scan on zip_codes (cost=0.00..42175.91 rows=13858 width=67)  
    (actual rows=13912 loops=1)  
    Filter: ((community_name)::text = 'Manchester'::text)  
    Rows Removed by Filter: 1683041
```

```
reltuples           | 1.696953e+06  
most_common_vals    | {..., Manchester, ...}  
most_common_freqs   | {..., 0.0081664017, ...}
```

$1.696953e+06 * 0.0081664017 = 13857.99987$

# Underestimate

```
SELECT * FROM zip_codes WHERE place_name = 'Manchester'  
                                AND community_name = 'Manchester';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes (cost=0.00..46418.29 rows=115 width=67)  
    (actual rows=11744 loops=1)  
  Filter: (((place_name)::text = 'Manchester'::text) AND  
           ((community_name)::text = 'Manchester'::text))  
Rows Removed by Filter: 1685209
```

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

```
SELECT * FROM zip_codes
WHERE place_name = 'Manchester'
AND community_name = 'Manchester';
```

```
P(place_name = 'Manchester' & community_name = 'Manchester')
= P(place_name = 'Manchester') * P(community_name = 'Manchester')
= 0.0082665813 * 0.0081664017
= 0.00006750822358150821
```

$0.00006750822358150821 * 1.696953e+06 = 114.558282531$

# Underestimate

```
SELECT * FROM zip_codes WHERE place_name = 'Manchester'  
        AND community_name = 'Manchester';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes (cost=0.00..46418.29 rows=115 width=67)  
    (actual rows=11744 loops=1)  
  Filter: (((place_name)::text = 'Manchester'::text) AND  
           ((community_name)::text = 'Manchester'::text))  
Rows Removed by Filter: 1685209
```

# Overestimate

```
SELECT * FROM zip_codes WHERE place_name != 'London'  
                                AND community_name = 'Westminster';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=10896 width=67)  
    (actual rows=4 loops=1)  
  Filter: (((place_name)::text <> 'London'::text) AND  
           ((community_name)::text = 'Westminster'::text))  
Rows Removed by Filter: 1696949
```

# 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  $\rightarrow$  {place, state, county, community}
  - M11 0AT  $\rightarrow$  {Manchester, England, Greater Manchester, Manchester District (B)}
- other dependencies:
  - place  $\rightarrow$  community
  - community  $\rightarrow$  county
  - county  $\rightarrow$  state

# CREATE STATISTICS

```
CREATE STATISTICS s (dependencies)
  ON place_name, community_name FROM zip_codes;
           2           5
```

```
ANALYZE zip_codes;
```

```
SELECT dependencies FROM pg_stats_ext WHERE statistics_name = 's';
```

dependencies

-----  
{ "2 => 5": 0.697633, "5 => 2": 0.095800 }

place → community: 0.697633 = d

$$\begin{aligned} &P(\text{place} = \text{'Manchester'} \ \& \ \text{community} = \text{'Manchester'}) \\ &= \\ &P(\text{place} = \text{'Manchester'}) * [d + (1-d) * P(\text{community} = \text{'Manchester'})] \\ \\ &1.697\text{e}+06 * 0.0083 * (0.698 + (1.0 - 0.698) * 0.0082) \\ &= \\ &9281.03 \end{aligned}$$

# Underestimate - fixed

```
SELECT * FROM zip_codes WHERE place_name = 'Manchester'  
        AND county_name = 'Manchester';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes (cost=0.00..46418.29 rows=9307 width=67)  
    (actual rows=11744 loops=1)
```

```
Filter: (((place_name)::text = 'Manchester'::text) AND  
        ((community_name)::text = 'Manchester'::text))
```

```
Rows Removed by Filter: 1685209
```

(was 115 before)

# Overestimate #1: not fixed :-)

```
SELECT * FROM zip_codes WHERE place_name != 'London'
                                AND community_name = 'Westminster';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=10896 width=67)
    (actual rows=4 loops=1)
  Filter: (((place_name)::text <> 'London'::text) AND
    ((community_name)::text = 'Westminster'::text))
Rows Removed by Filter: 1696949
```

Functional dependencies only work with equalities.



## Overestimate #2: not fixed :-)

```
SELECT * FROM zip_codes WHERE place_name = 'Manchester'
                                AND county_name = 'Westminster';
```

### QUERY PLAN

```
-----
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=9305 width=67)
    (actual rows=0 loops=1)
    Filter: (((place_name)::text = 'Manchester'::text) AND
              ((community_name)::text = 'Westminster'::text))
    Rows Removed by Filter: 1696953
```

The queries need to “respect” the functional dependencies.

ndistinct  
(GROUP BY)

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

#### QUERY PLAN

```
-----
HashAggregate  (cost=46418.29..46421.86 rows=358 width=29)
    (actual rows=359 loops=1)
    Group Key: community_name
    -> Seq Scan on zip_codes  (cost=0.00..37933.53 rows=1696953 width=21)
        (actual rows=1696953 loops=1)

Planning Time: 0.087 ms
Execution Time: 337.718 ms
(5 rows)
```

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

attname	n_distinct
community_name	358
county_name	91
latitude	59925
longitude	64559
place_name	12281
postal_code	-1
state_name	3

(7 rows)

```
SELECT count(*) FROM zip_codes GROUP BY community_name, place_name;
```

#### QUERY PLAN

---

```
GroupAggregate (cost=294728.63..313395.11 rows=169695 width=40)
    (actual rows=15194 loops=1)
    Group Key: community_name, place_name
    -> Sort (cost=294728.63..298971.01 rows=1696953 width=32)
        (actual rows=1696953 loops=1)
        Sort Key: community_name, place_name
        Sort Method: external merge  Disk: 69648kB
        -> Seq Scan on zip_codes (cost=0.00..37933.53 rows=1696953 width=32)
            (actual rows=1696953 loops=1)
```

Planning Time: 0.374 ms

Execution Time: 1554.933 ms

```
SELECT count(*) FROM zip_codes GROUP BY community_name, place_name;
```

#### QUERY PLAN

---

```
GroupAggregate (cost=294728.63..313395.11 rows=169695 width=40)
    (actual rows=15194 loops=1)
    Group Key: community_name, place_name
    -> Sort (cost=294728.63..298971.01 rows=1696953 width=32)
        (actual rows=1696953 loops=1)
        Sort Key: community_name, place_name
        Sort Method: external merge  Disk: 69648kB
        -> Seq Scan on zip_codes (cost=0.00..37933.53 rows=1696953 width=32)
            (actual rows=1696953 loops=1)
```

Planning Time: 0.374 ms

Execution Time: 1554.933 ms

```
ndistinct(community, place)  
  
=  
  
ndistinct(community) * ndistinct(place)
```

```
358 * 12281 = 4396598 (1.7M rows?)
```

```
ndistinct(community, place)  
  
=  
  
ndistinct(community) * ndistinct(place)
```

```
358 * 12281 = 169695
```

```
(capped to 10% of the table)
```



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

```
ANALYZE zip_codes;
```

```
SELECT stxndistinct FROM pg_stats_ext WHERE stxname = 's';
```

n\_distinct

-----  
{ "2, 4": 12996, "2, 5": 13221, "4, 5": 399, "2, 4, 5": 13252 }

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

#### QUERY PLAN

---

```
HashAggregate  (cost=50660.68..50792.89 rows=13221 width=40)
    (actual rows=15194 loops=1)
    Group Key: community_name, place_name
    -> Seq Scan on zip_codes  (cost=0.00..37933.53 rows=1696953 width=32)
        (actual rows=1696953 loops=1)
```

Planning Time: 0.056 ms

Execution Time: 436.828 ms

(5 rows)

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

#### QUERY PLAN

---

```
HashAggregate  (cost=50660.68..50792.89 rows=13221 width=40)
    (actual rows=15194 loops=1)
    Group Key: community_name, place_name
    -> Seq Scan on zip_codes  (cost=0.00..37933.53 rows=1696953 width=32)
        (actual rows=1696953 loops=1)
```

Planning Time: 0.056 ms

Execution Time: 436.828 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

# MCV lists (PG12)

# Estimation issues

1) underestimate (fixed)

```
SELECT * FROM zip_codes WHERE place_name = 'London'
                               AND county_name = 'Greater London';
```

2) Overestimate #1 (not fixed)

```
SELECT * FROM zip_codes WHERE place_name != 'London'
                               AND county_name = 'Greater London';
```

3) Overestimate #2 (not fixed)

```
SELECT * FROM zip_codes WHERE place_name = 'London'
                               AND county_name = 'Greater Manchester';
```

# MCV stats

```
CREATE STATISTICS s (mcv) ON place_name, county_name FROM zip_codes;
```

```
SET default_statistics_target = 10000;
```

```
ANALYZE zip_codes;
```

```
SELECT most_common_vals, most_common_freqs  
FROM pg_stats_ext WHERE statistics_name = 's';
```

```
most_common_vals | {{London,"Greater London"},{Birmingham,"West Midlands"}, ...  
most_common_freqs | {0.1028343153876389, 0.012347425061271585, ...
```

# Underestimate (no stats)

```
SELECT * FROM zip_codes WHERE place_name = 'London'
                                AND county_name = 'Greater London';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=18306 width=67)
    (actual time=18.444..224.413 rows=174505 loops=1)
  Filter: (((place_name)::text = 'London'::text)
    AND ((county_name)::text = 'Greater London'::text))
 Rows Removed by Filter: 1522448
```



# Underestimate (with dependencies)

```
SELECT * FROM zip_codes WHERE place_name = 'London'
                                AND county_name = 'Greater London';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=133249 width=67)
    (actual time=17.677..224.120 rows=174505 loops=1)
  Filter: (((place_name)::text = 'London'::text)
    AND ((county_name)::text = 'Greater London'::text))
 Rows Removed by Filter: 1522448
```

no stats: 18306

# Underestimate (with MCV)

```
SELECT * FROM zip_codes WHERE place_name = 'London'
                                AND county_name = 'Greater London';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=174505 width=67)
    (actual time=18.467..221.760 rows=174505 loops=1)
  Filter: (((place_name)::text = 'London'::text)
    AND ((county_name)::text = 'Greater London'::text))
 Rows Removed by Filter: 1522448
```

no stats: 18306

dependencies: 133249

# Overestimate #1 (no stats)

```
SELECT * FROM zip_codes WHERE place_name != 'London'  
        AND county_name = 'Greater London';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes (cost=0.00..46418.29 rows=157930 width=67)  
    (actual time=125.545..166.035 rows=1731 loops=1)  
  Filter: (((place_name)::text <> 'London'::text)  
    AND ((county_name)::text = 'Greater London'::text))  
Rows Removed by Filter: 1695222
```

# Overestimate #1 (with MCV)

```
SELECT * FROM zip_codes WHERE place_name != 'London'
                                AND county_name = 'Greater London';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=33623 width=67)
    (actual time=124.716..160.768 rows=1731 loops=1)
  Filter: (((place_name)::text <> 'London'::text)
    AND ((county_name)::text = 'Greater London'::text))
 Rows Removed by Filter: 1695222
```

no stats: 157930      dependencies: 157930

## Overestimate #2 (no stats)

```
SELECT * FROM zip_codes WHERE place_name = 'London'  
                                AND county_name = 'Greater Manchester';
```

### QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=7345 width=67)  
    (actual time=144.571..144.572 rows=0 loops=1)  
    Filter: (((place_name)::text = 'London'::text)  
            AND ((county_name)::text = 'Greater Manchester'::text))  
    Rows Removed by Filter: 1696953
```

# Overestimate #2 (with dependencies)

```
SELECT * FROM zip_codes WHERE place_name = 'London'
                                AND county_name = 'Greater Manchester';
```

## QUERY PLAN

---

```
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=130264 width=67)
    (actual time=144.693..144.694 rows=0 loops=1)
  Filter: (((place_name)::text = 'London'::text)
    AND ((county_name)::text = 'Greater Manchester'::text))
 Rows Removed by Filter: 1696953
```

no stats: 7345

## Overestimate #2 (with MCV)

```
SELECT * FROM zip_codes WHERE place_name = 'London'  
                                AND county_name = 'Greater Manchester';
```

### QUERY PLAN

```
-----  
Seq Scan on zip_codes  (cost=0.00..46418.29 rows=7345 width=67)  
    (actual time=144.020..144.021 rows=0 loops=1)  
    Filter: (((place_name)::text = 'London'::text)  
        AND ((county_name)::text = 'Greater Manchester'::text))  
    Rows Removed by Filter: 1696953
```

no stats: 7345

dependencies: 130264

# Summary



# Future Improvements

- additional types of statistics
  - 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)

# Q & A