CREATE STATISTICS

Tomas Vondra, EDB
tomas.vondra@enterprisedb.com
pgconf.de 2022, May 13, Leipzig

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

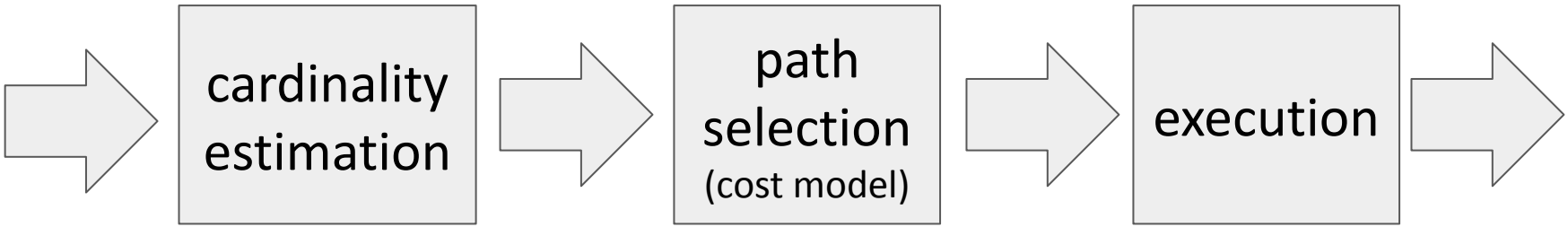
```
CREATE TABLE zip_codes (
  postal_code     VARCHAR(20),
  place_name      VARCHAR(180),
  state_name      VARCHAR(100),
  county_name     VARCHAR(100),
  community_name  VARCHAR(100),
  latitude        REAL,
  longitude       REAL
);

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

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

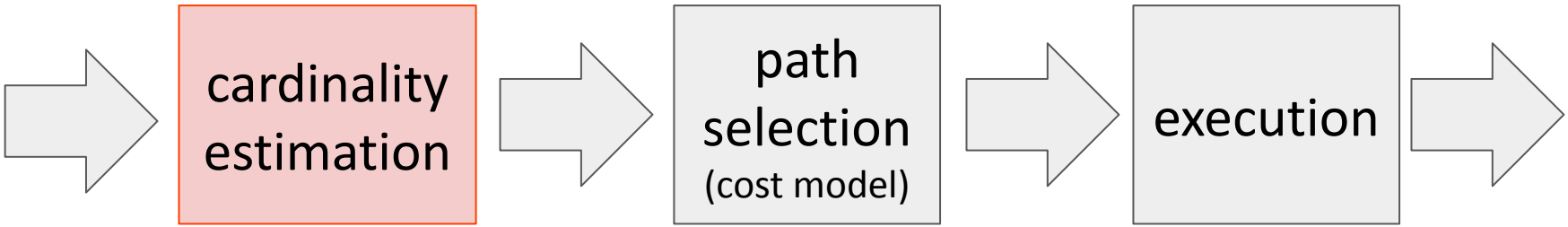

EDB

Why should you care?



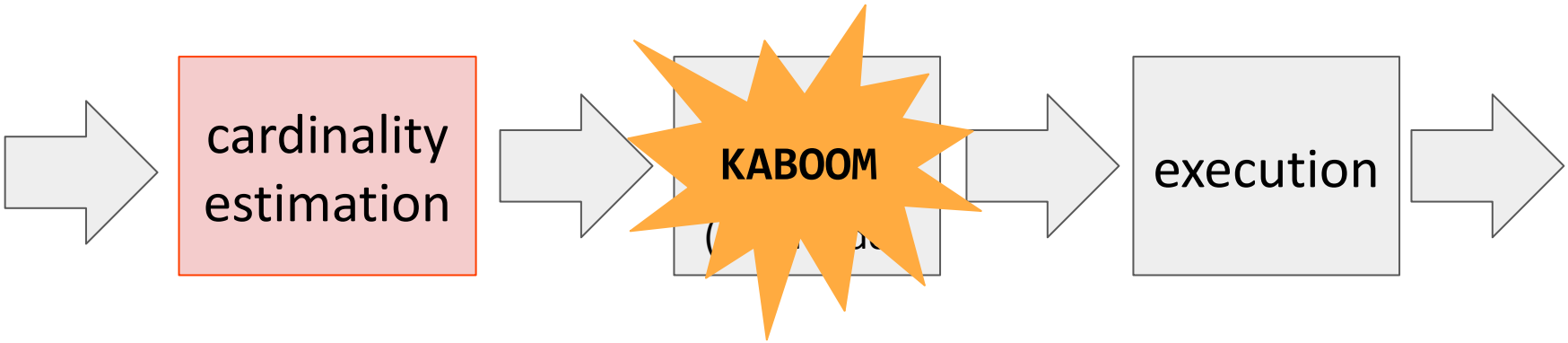
EDB

Why should you care?



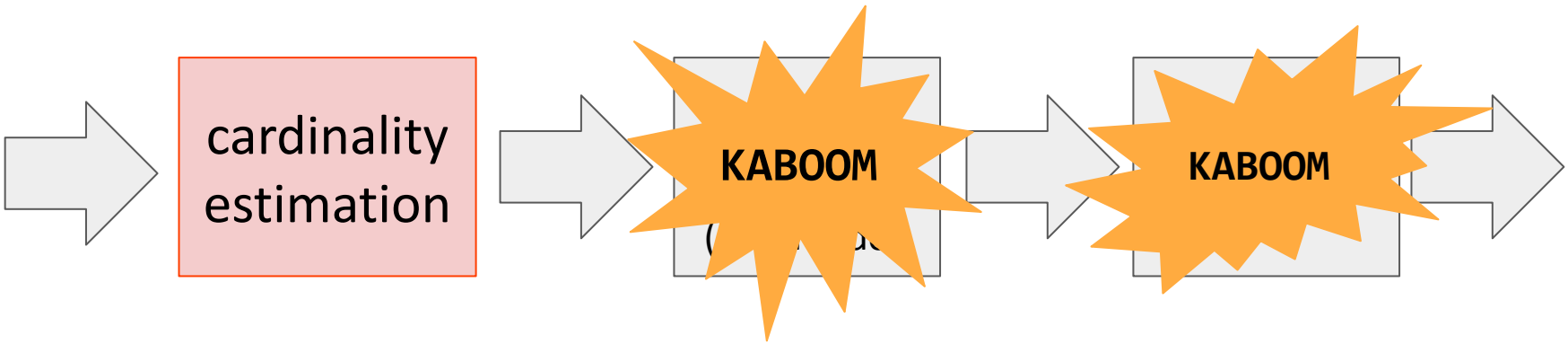
EDB

Why should you care?



EDB

Why should you care?



EDB

EXPLAIN

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

                            QUERY PLAN
-------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..25844.55 rows=2144 width=63)
                        (actual rows=1984 loops=1)
   Filter: ((place_name)::text = 'Leipzig'::text)
   Rows Removed by Filter: 1052608
 Planning Time: 0.074 ms
 Execution Time: 176.556 ms
(5 rows)
```


EDB

relpages, reltuples

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

  reltuples   | relpages
--------------+----------
 1.054604e+06 | 12662
```

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  | {Berlin, Hamburg, München, ..., Leipzig, Dortmund, ...}
most_common_freqs | {0.0116, 0.0062, 0.00457, ..., 0.002033, 0.00183, ...}
...               | ...
```


EDB

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

                          QUERY PLAN
---------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..25844.55 rows=2144 width=63)
                        (actual rows=1984 loops=1)
   Filter: ((place_name)::text = 'Leipzig'::text)

reltuples         | 1.054604e+06
most_common_vals  | {..., Leipzig, ...}
most_common_freqs | {..., 0.002033, ...}

                          1.054604e+06 * 0.002033 = 2144
```


EDB

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

                              QUERY PLAN
-----------------------------------------------------------------------
Seq Scan on zip_codes  (cost=0.00..25844.40 rows=145604 width=63)
                       (actual rows=144576 loops=1)
   Filter: ((state_name)::text = 'Bayern'::text)
   Rows Removed by Filter: 910016

reltuples         | 1.054604e+06
most_common_vals  | {..., Bayern, ...}
most_common_freqs | {..., 0.13806666, ...}

                    1.054604e+06 * 0.13806666 = 145605.65190264
```


EDB

Underestimate

```
EXPLAIN (ANALYZE, TIMING OFF)
SELECT * FROM zip_codes WHERE place_name = 'München'
                          AND state_name = 'Bayern';

                             QUERY PLAN
--------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..28480.88 rows=733 width=63)
                        (actual rows=4736 loops=1)
   Filter: (((place_name)::text = 'München'::text) AND
            ((state_name)::text = 'Bayern'::text))
   Rows Removed by Filter: 1049856
```

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

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

```
  P(place_name = 'München' & state_name = 'Bayern')
     = P(place_name = 'München') * P(state_name = 'Bayern')
     = 0.0050333333                * 0.13806666
     = 0.000694935517397778
```

```
0.000694935517397778 * 1.054604e+06 = 732.88
```


EDB

Underestimate

```
EXPLAIN (ANALYZE, TIMING OFF)
SELECT * FROM zip_codes WHERE place_name = 'München'
                          AND state_name = 'Bayern';

                             QUERY PLAN
--------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..28480.88 rows=733 width=63)
                        (actual rows=4736 loops=1)
   Filter: (((place_name)::text = 'München'::text) AND
            ((state_name)::text = 'Bayern'::text))
   Rows Removed by Filter: 1049856
```

0.000694935517397778 * 1.054604e+06 = **732.88**

EDB

Overestimate

```
EXPLAIN (ANALYZE, TIMING OFF)
SELECT * FROM zip_codes WHERE place_name = 'München'
                          AND state_name != 'Bayern';

                             QUERY PLAN
-------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..28480.88 rows=4575 width=63)
                        (actual rows=0 loops=1)
   Filter: (((state_name)::text <> 'Bayern'::text) AND
            ((place_name)::text = 'München'::text))
   Rows Removed by Filter: 1054592
```

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)
```


EDB

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)
```


EDB

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)
```


EDB

functional dependencies (WHERE)

Functional Dependencies

- value in column A determines value in column B
- trivial example: primary key determines everything
  - zip code ⟶ {place, state, county, community}
  - 04103 ⟶ {Leipzig, Sachsen, Kreisfreie Stadt Leipzig}
- other dependencies:
  - place ⟶ community
  - community ⟶ county
  - county ⟶ state

CREATE STATISTICS

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

ANALYZE zip_codes;

SELECT dependencies FROM pg_stats_ext WHERE statistics_name = 's';

                dependencies
-------------------------------------------
 {"2 => 3": 0.925000, "3 => 2": 0.000100}
```


EDB

```
place => state: 0.925000 = d
```

```
P(place = 'München' & state = 'Bayern')
=
P(place = 'München') * [d + (1-d) * P(state = 'Bayern')]
```

```
0.00503 * (0.925000 + (1.0 - 0.925000) *  0.138067) * 1.054604e+06
=
4961.74
```


EDB

Underestimate - fixed

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

                          QUERY PLAN
-------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..28480.88 rows=4962 width=63)
                        (actual rows=4736 loops=1)
   Filter: (((place_name)::text = 'München'::text) AND
            ((state_name)::text = 'Bayern'::text))
   Rows Removed by Filter: 1049856
```

(was **733** before)

EDB

Overestimate #1: not fixed :-(

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

                          QUERY PLAN
-----------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..28480.88 rows=4575 width=63)
                        (actual rows=0 loops=1)
   Filter: (((state_name)::text <> 'Bayern'::text) AND
            ((place_name)::text = 'München'::text))
   Rows Removed by Filter: 1054592
```

Functional dependencies only work with equalities.

EDB

Overestimate #2: not fixed :-(

```
SELECT * FROM zip_codes WHERE place_name = 'München'
                          AND state_name = 'Rheinland-Pfalz';

                         QUERY PLAN
-------------------------------------------------------------------
 Seq Scan on zip_codes  (cost=0.00..28480.88 rows=4446 width=64)
                        (actual rows=0 loops=1)
   Filter: (((place_name)::text = 'München'::text) AND
            ((state_name)::text = 'Rheinland-Pfalz'::text))
   Rows Removed by Filter: 1054592
```

The queries need to “respect” the functional dependencies.

(1402 without dependencies)

EDB

ndistinct
(GROUP BY)

EDB

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

                                QUERY PLAN
-------------------------------------------------------------------------
 HashAggregate  (cost=47432.68..47435.63 rows=397 width=16)
                (actual rows=400 loops=1)
   Group Key: community_name
   Batches: 1  Memory Usage: 109kB
   ->  Seq Scan on zip_codes  (cost=0.00..23207.92 rows=1054592 width=19)
                              (actual rows=1054592 loops=1)
```


EDB

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

     attname     | n_distinct
-----------------+------------
 community_name |        397
 county_name    |         20
 latitude       |      10361
 longitude      |      10538
 place_name     |      13340
 postal_code    |       7831
 state_name     |         17


(7 rows)
```


EDB

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

                                QUERY PLAN
-------------------------------------------------------------------------
 HashAggregate  (cost=109882.20..125355.04 rows=105459 width=38)
                (actual rows=14518 loops=1)
   Group Key: community_name, place_name
   Planned Partitions: 4  Batches: 1  Memory Usage: 3361kB
   ->  Seq Scan on zip_codes  (cost=0.00..23207.92 rows=1054592 width=30)
                              (actual rows=1054592 loops=1)
```


EDB

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

                                QUERY PLAN
-------------------------------------------------------------------------
 HashAggregate  (cost=109882.20..125355.04 rows=105459 width=38)
                (actual rows=14518 loops=1)
   Group Key: community_name, place_name
   Planned Partitions: 4  Batches: 1  Memory Usage: 3361kB
   ->  Seq Scan on zip_codes  (cost=0.00..23207.92 rows=1054592 width=30)
                              (actual rows=1054592 loops=1)
```


EDB

```
ndistinct(community, place)

=

ndistinct(community) * ndistinct(place)
```

```
397 * 13340 = 5295980
```

```
ndistinct(community, place)

=

ndistinct(community) * ndistinct(place)
```

```
397 * 13340 = 5295980 => 105459
```

(capped to 10% of the table)

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

ANALYZE zip_codes;

SELECT n_distinct FROM pg_stats_ext WHERE statistics_name = 's';

                           n_distinct
-----------------------------------------------------------------
 {"2, 4": 13555, "2, 5": 14157, "4, 5": 397, "2, 4, 5": 14157}
```


EDB

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

                                    QUERY PLAN
-------------------------------------------------------------------------------
 HashAggregate  (cost=31117.36..31258.93 rows=14157 width=38)
                (actual rows=14518 loops=1)
   Group Key: community_name, place_name
   Batches: 1  Memory Usage: 2593kB
   ->  Seq Scan on zip_codes  (cost=0.00..23207.92 rows=1054592 width=30)
                              (actual rows=1054592 loops=1)
```

ndistinct

- the “old behavior” was defensive
  - unreliable estimates with multiple columns
  - HashAggregate can’t spill to disk (OOM, PG13)
  - 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)

MCV stats

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

ANALYZE zip_codes;

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

```
most_common_vals  | {{Berlin,Berlin}, {Hamburg,Hamburg}, {München,Bayern}, …
most_common_freqs | {0.0097, 0.006, 0.004833333333333334, …
```


EDB

Expressions

Stats on Expressions

- cheaper alternative to functional indexes
- allows multi-column statistics on expressions

```
CREATE STATISTICS s ON (a+b) FROM t;

CREATE STATISTICS s ON (a+b), extract('hour' from c) FROM t;
```


EDB

Summary

Future Improvements

- additional types of statistics
  - histograms (??), …
- improving join estimates
  - using MCV lists
  - special multi-table statistics (syntax already supports it)

Q & A