Adaptive Query Optimization

Approaches and Challenges

What is Adaptive Query Optimization?

How does DBMS execute a query?

SQL Query → … → Result

SQL query

```sql
SELECT *
FROM students, departments
WHERE students.GPA > 4
AND departments.name = 'Data Science'
AND students.department_id = departments.id;
```

How does DBMS execute a query?

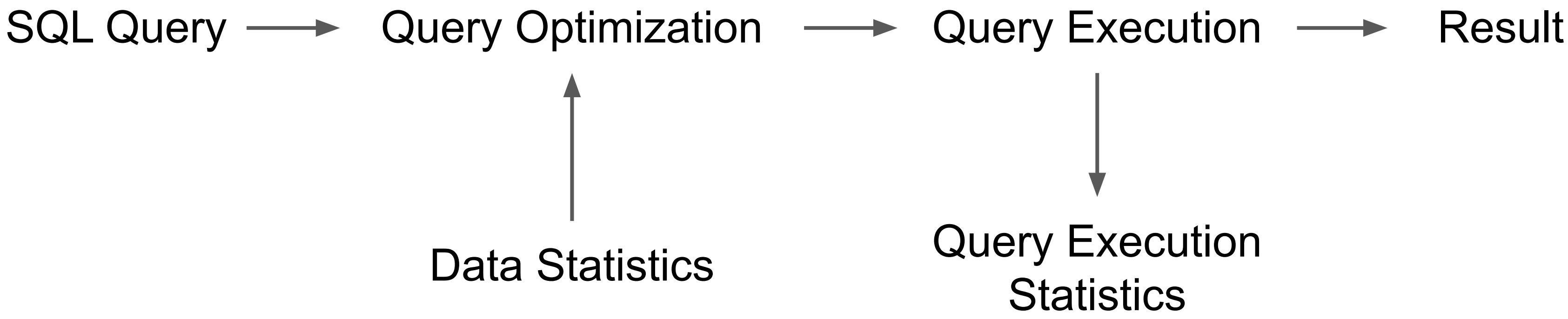
SQL Query → Query Optimization → … → Result

Data Statistics ↑ (into Query Optimization)

Query plan

```
Hash Join  (cost=25.95..28.42 rows=1 width=59)
  Hash Cond: (students.department_id = departments.id)
  ->  Seq Scan on students  (cost=0.00..2.24 rows=89 width=23)
        Filter: (gpa > '4'::double precision)
  ->  Hash  (cost=25.88..25.88 rows=6 width=36)
        ->  Seq Scan on departments  (cost=0.00..25.88 rows=6 width=36)
              Filter: (name = 'Data Science'::text)
```

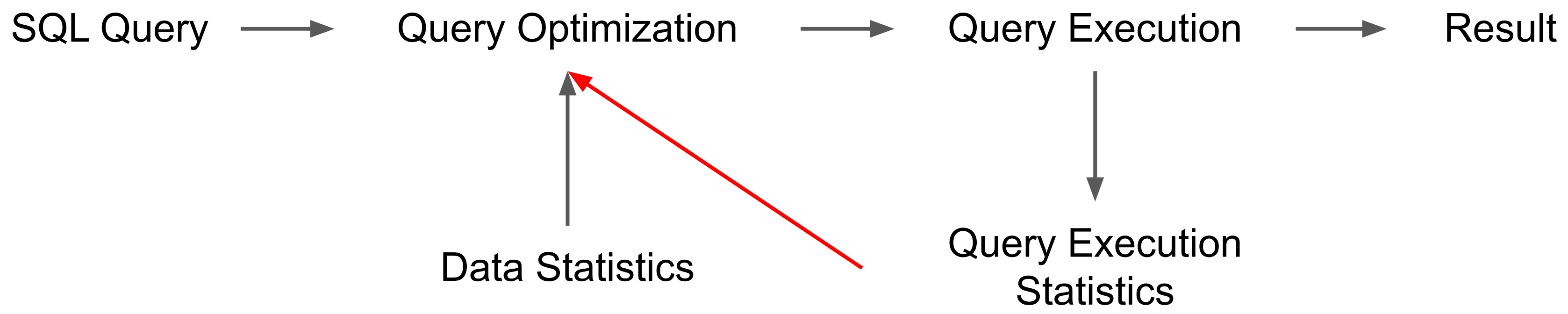
How does DBMS execute a query?



Query Execution Statistics

```
Hash Join  (cost=25.95..28.42 rows=1 width=59) (actual time=0.061..0.147 rows=41 loops=1)
  Hash Cond: (students.department_id = departments.id)
  ->  Seq Scan on students  (cost=0.00..2.24 rows=89 width=23) (actual time=0.018..0.058 rows=92 loops=1)
        Filter: (gpa > '4'::double precision)
        Rows Removed by Filter: 9
  ->  Hash  (cost=25.88..25.88 rows=6 width=36) (actual time=0.014..0.014 rows=1 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
        ->  Seq Scan on departments  (cost=0.00..25.88 rows=6 width=36) (actual time=0.007..0.009 rows=1 loops=1)
              Filter: (name = 'Data Science'::text)
              Rows Removed by Filter: 1
```

Adaptive Query Optimization is the red line



How does Query Optimization work?

Cost-based Query Optimization

Plan

Cost

CostFunction (

Hash Join (cost=25.95..28.42 rows=1 width=59)
Hash Cond: (students.department_id = departments.id)
-> **Seq Scan** on students (cost=0.00..2.24 rows=89 width=23)
Filter: (gpa > '4'::double precision)
-> **Hash** (cost=25.88..25.88 rows=6 width=36)
-> **Seq Scan** on departments (cost=0.00..25.88 rows=6 width=36)
Filter: (name = 'Data Science'::text)

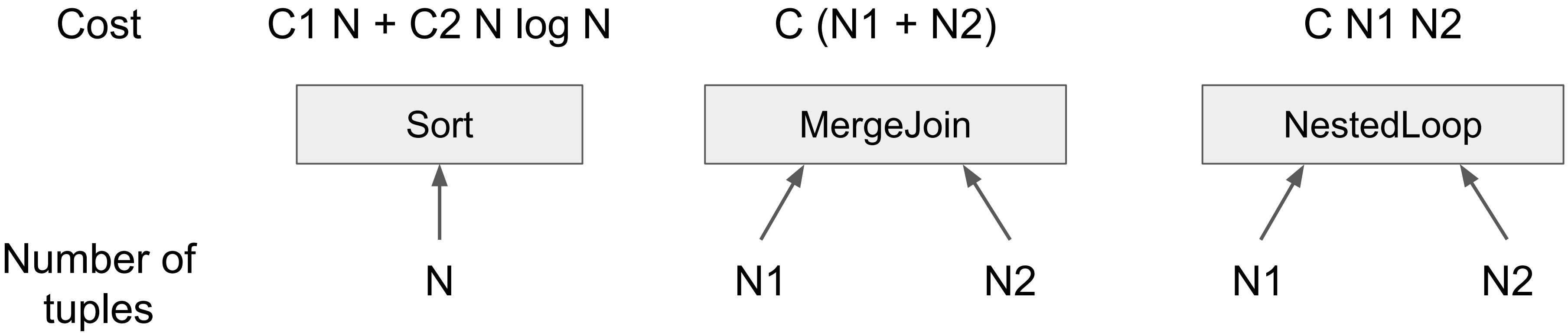
) = 28.42

BestPlan = ArgMin (CostFunction (Plan))

CostFunction is additive: $\square + \square + \square = \square$

```
Hash Join  (cost=25.95..28.42 rows=1 width=59)
  Hash Cond: (students.department_id = departments.id)
  ->  Seq Scan on students  (cost=0.00..2.24 rows=89 width=23)
        Filter: (gpa > '4'::double precision)
  ->  Hash  (cost=25.88..25.88 rows=6 width=36)
        ->  Seq Scan on departments  (cost=0.00..25.88 rows=6 width=36)
              Filter: (name = 'Data Science'::text)
```

CostFunction (Plan)



The main problem is in Cardinality Estimation

How good are query optimizers, really?
V. Leis, A. Gubichev, A. Mirchev, P. Boncz,
A. Kemper, and T. Neumann,
Proc. VLDB, Nov. 2015

Adaptive Cardinality Estimation
O. Ivanov, S. Bartunov,
Arxiv, Nov. 2017

Cardinality Estimation: no clauses

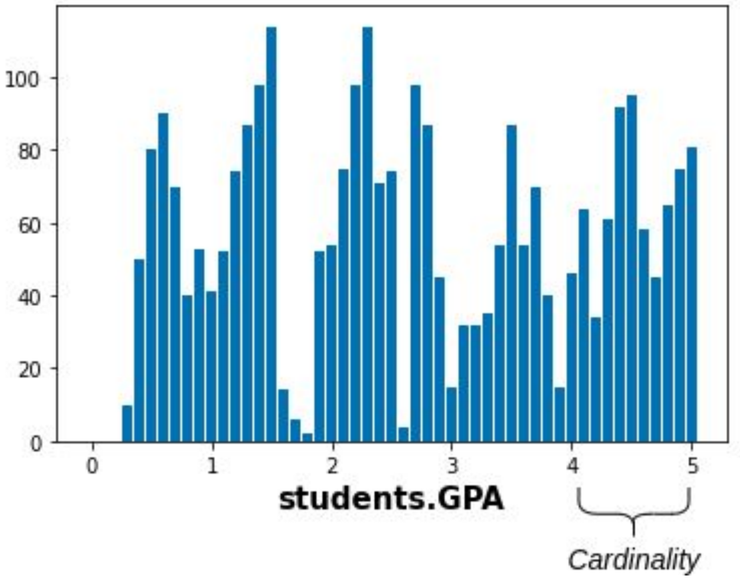
Table

Cardinality is known

Cardinality Estimation: one clause

Table + Clause1

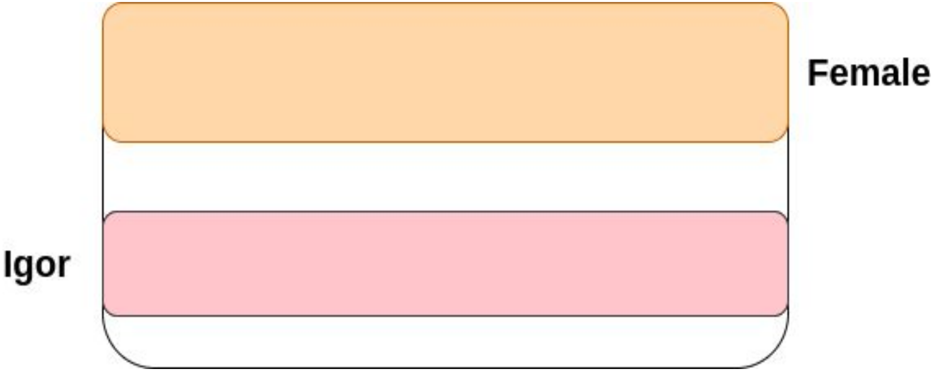
SELECT *
FROM students
WHERE students.GPA > 4;



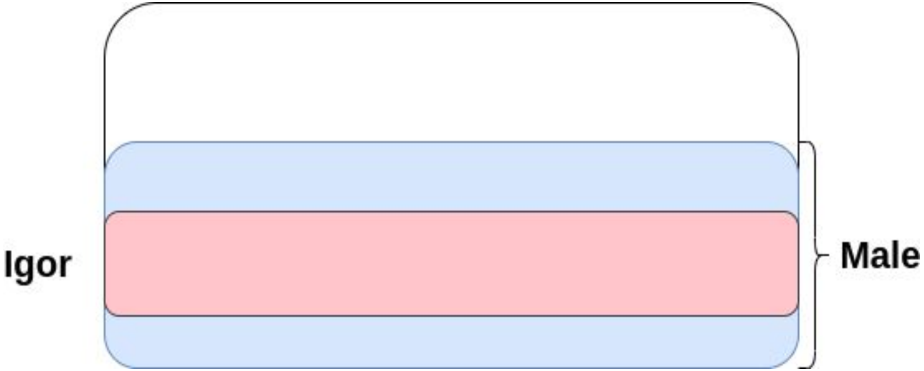
Cardinality Estimation: two or more clauses

Table + Clause1
+ Clause2

SELECT *
FROM students
WHERE students.name = 'Igor'
AND students.gender = 'Female';



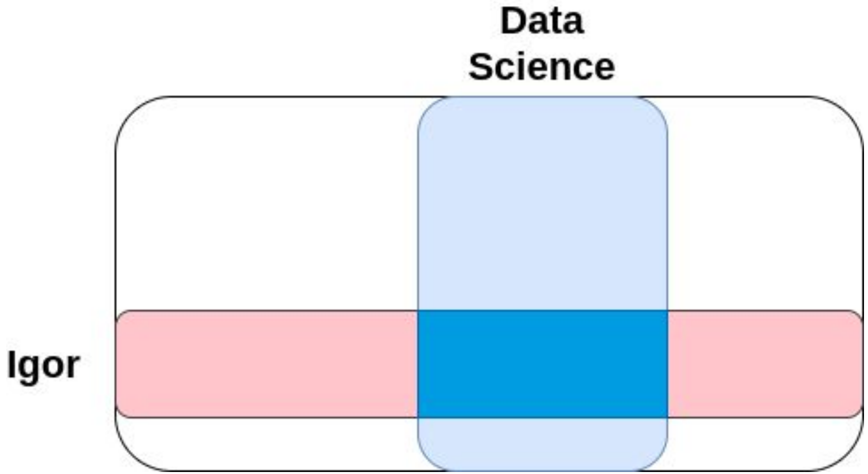
SELECT *
FROM students
WHERE students.name = 'Igor'
AND students.gender = 'Male';



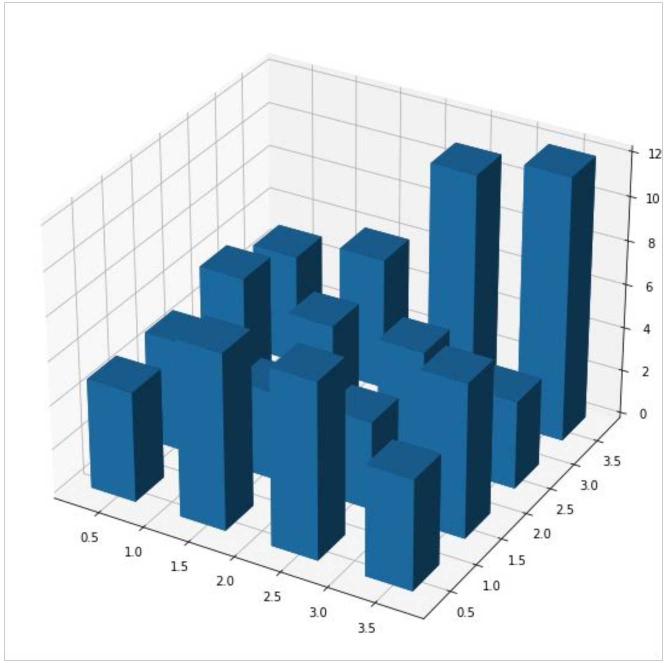
PostgreSQL default assumption: clauses independence

Table + Clause1 + Clause2

```
SELECT *
FROM students
WHERE students.name = 'Igor'
AND students.department_name = 'Data Science';
```



Multidimensional Histograms (aka Cross-Column Statistics)



Query Optimization: conclusions

- \+ OLTP
- − OLAP
- − ORM (sometimes)

Adaptive Cardinality Estimation

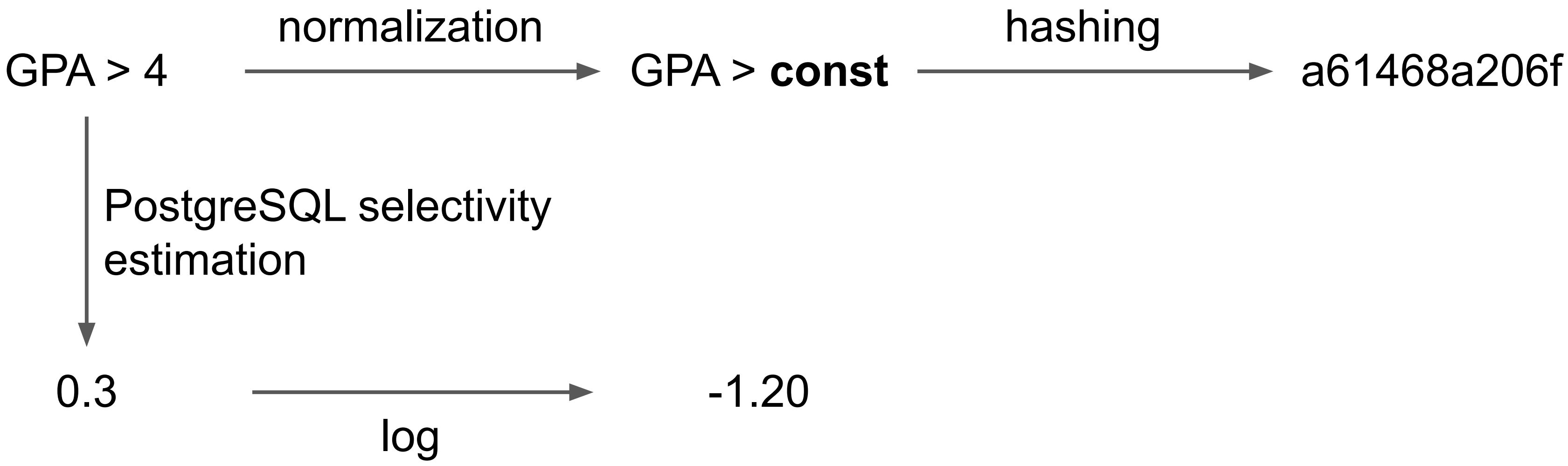
What information do we have?

```
Hash Join  (actual time=0.061..0.147 rows=41 loops=1)
  Hash Cond: (students.department_id = departments.id)
  ->  Seq Scan on students  (actual time=0.018..0.058 rows=92 loops=1)
        Filter: (gpa > '4'::double precision)
        Rows Removed by Filter: 9
  ->  Hash  (actual time=0.014..0.014 rows=1 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
        ->  Seq Scan on departments  (actual time=0.007..0.009 rows=1 loops=1)
              Filter: (name = 'Data Science'::text)
              Rows Removed by Filter: 1
```

Information in one relation

$$\text{students} \xrightarrow{\text{hashing}} \text{fd528b449f20}$$

Information in one clause



One plan node = Object

Clauses info	Relations hashes	Log Cardinality
(a61468a206ff, -1.20) (654101251a11, -9.21) (9141e9f5e5f5, -0.11)	fd528b449f20 27cb8c47b899	7.08

K Nearest Neighbours

Similarity between objects

65410f251a11	fd528b449f20	5.32
-9.21		

9141e9f5e5f5	fd528b449f20	2.75
-0.11		

Incomparable

K Nearest Neighbours

Similarity between objects

9141e9f5e5f5 -9.21	a61468a206ff -1.20	fd528b449f20	5.32

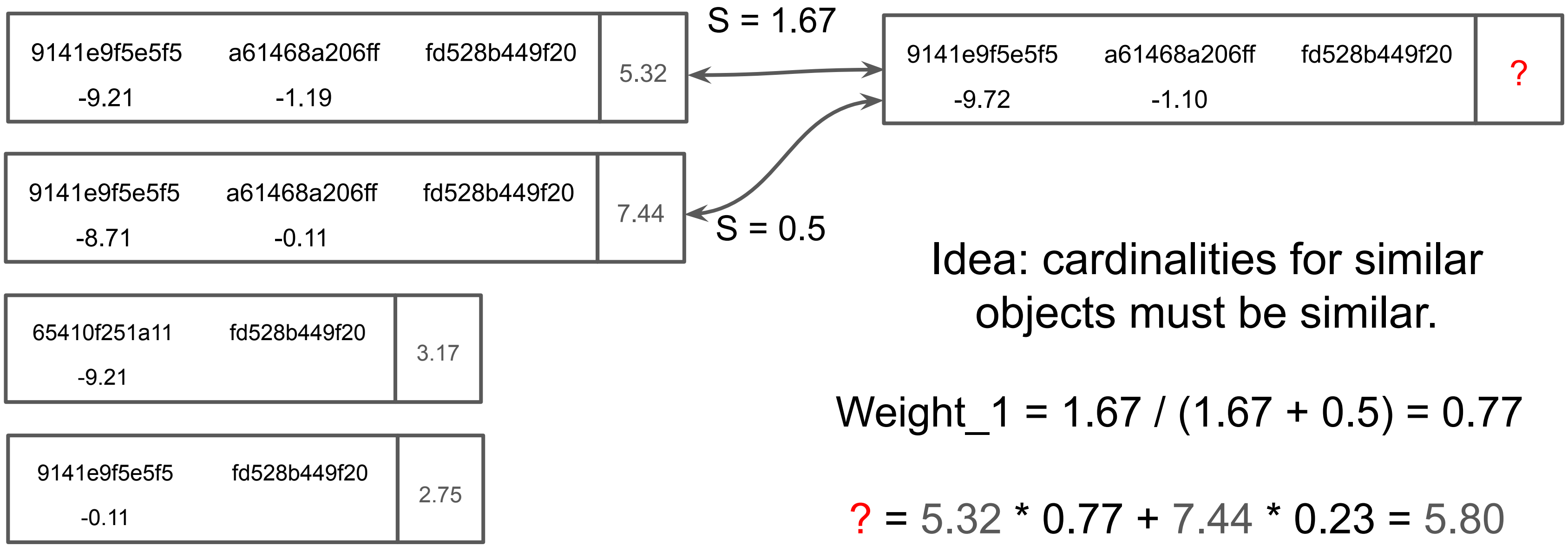
9141e9f5e5f5 -8.67	a61468a206ff -0.11	fd528b449f20	7.44

$$\text{Distance} = \|(-9.21, -1.20) - (-8.67, -0.11)\|$$

$$\text{Similarity} = 1 / (\text{Distance} + \text{EPS})$$

K Nearest Neighbours

Stored objects



Idea: cardinalities for similar objects must be similar.

Weight_1 = 1.67 / (1.67 + 0.5) = 0.77

? = 5.32 * 0.77 + 7.44 * 0.23 = 5.80

K Nearest Neighbours

Stored objects

9141e9f5e5f5 -9.21	a61468a206ff -1.19	fd528b449f20	5.32

9141e9f5e5f5 -8.71	a61468a206ff -0.11	fd528b449f20	7.44

65410f251a11 -2.43	fd528b449f20	3.17

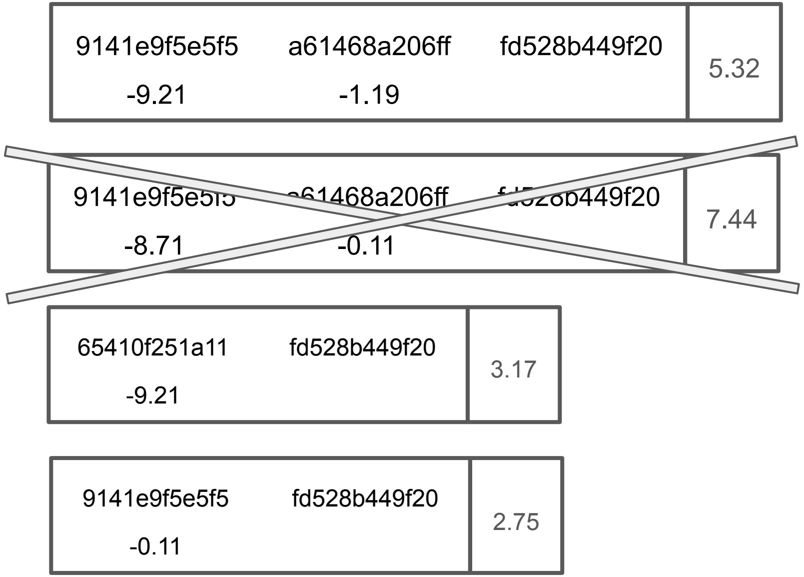
9141e9f5e5f5 -0.72	fd528b449f20	2.75

6f410f251a11 -9.72	a61468a206ff -1.10	fd528b449f20	?

No similar objects -> use PostgreSQL estimator

K Nearest Neighbours

Number of stored objects *of each type* is limited



K Nearest Neighbours: Results

- TPC-DS*:
  - Maximum performance boost is 95x
  - Mean execution time decreased by 50%
- Join Order Benchmark:
  - Maximum performance boost is 6.2x
  - Mean execution time decreased by 30%

- Rarely AQO slows down query (errors in optimizator)
- Theoretical guarantees for convergence

QUERY PLAN

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=20193.92..20193.93 rows=1 width=96) (actual time=909.499..912.051 rows=1 loops=1)
   ->  Gather  (cost=20193.69..20193.90 rows=2 width=96) (actual time=908.966..912.041 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=19193.69..19193.70 rows=1 width=96) (actual time=906.441..906.445 rows=1 loops=3)
         ->  Hash Join  (cost=8.73..19193.61 rows=11 width=65) (actual time=164.241..898.222 rows=37034 loops=3)
         Hash Cond: (mi.info_type_id = it2.id)
         ->  Nested Loop  (cost=6.30..19187.85 rows=1224 width=69) (actual time=164.150..885.947 rows=172109 loops=3)
               ->  Nested Loop  (cost=5.87..19113.34 rows=39 width=35) (actual time=164.130..794.055 rows=5490 loops=3)
               ->  Hash Join  (cost=5.45..18681.66 rows=968 width=39) (actual time=163.526..592.504 rows=100870 loops=3)
               Hash Cond: (mc.company_type_id = ct.id)
               ->  Nested Loop  (cost=4.39..18659.67 rows=3870 width=43) (actual time=163.508..568.962 rows=257695 loops=3)
               ->  Hash Join  (cost=3.96..18217.25 rows=727 width=31) (actual time=163.479..398.759 rows=69960 loops=3)
                     Hash Cond: (t.kind_id = kt.id)
                     ->  Nested Loop  (cost=2.86..18194.71 rows=5089 width=35) (actual time=0.062..382.378 rows=153308 loops=3)
                     ->  Hash Join  (cost=2.43..15253.60 rows=5089 width=10) (actual time=0.029..80.692 rows=153308 loops=3)
                     Hash Cond: (miidx.info_type_id = it.id)
                     ->  Parallel Seq Scan on movie_info_idx miidx  (cost=0.00..13685.15 rows=575015 width=14) (actual time=0.010..33.836 rows=460012 loops=3)
                     ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.010..0.011 rows=1 loops=3)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.008..0.009 rows=1 loops=3)
                           Filter: ((info)::text = 'rating'::text)
                           Rows Removed by Filter: 112
                     ->  Index Scan using title_pkey on title t  (cost=0.43..0.58 rows=1 width=25) (actual time=0.002..0.002 rows=1 loops=459925)
                     Index Cond: (id = miidx.movie_id)
                     ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.012..0.012 rows=1 loops=3)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.009..0.010 rows=1 loops=3)
                     Filter: ((kind)::text = 'movie'::text)
                     Rows Removed by Filter: 6
               ->  Index Scan using movie_id_movie_companies on movie_companies mc  (cost=0.43..0.56 rows=5 width=12) (actual time=0.001..0.002 rows=4 loops=209880)
                     Index Cond: (movie_id = t.id)
               ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.009..0.009 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.006..0.006 rows=1 loops=3)
                     Filter: ((kind)::text = 'production companies'::text)
                     Rows Removed by Filter: 3
               ->  Index Scan using company_name_pkey on company_name cn  (cost=0.42..0.45 rows=1 width=4) (actual time=0.002..0.002 rows=0 loops=302610)
               Index Cond: (id = mc.company_id)
               Filter: ((country_code)::text = '[de]'::text)
               Rows Removed by Filter: 1
               ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.49 rows=42 width=50) (actual time=0.003..0.014 rows=31 loops=16470)
               Index Cond: (movie_id = t.id)
         ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.021..0.021 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.012..0.018 rows=1 loops=3)
               Filter: ((info)::text = 'release dates'::text)
               Rows Removed by Filter: 112
```

Planning Time: 51.227 ms
Execution Time: 913.359 ms

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=45364.09..45364.10 rows=1 width=96) (actual time=441.826..444.935 rows=1 loops=1)
   ->  Gather  (cost=45363.86..45364.07 rows=2 width=96) (actual time=441.719..444.929 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=44363.86..44363.87 rows=1 width=96) (actual time=437.915..437.920 rows=1 loops=3)
         ->  Hash Join  (cost=4800.09..44016.67 rows=46292 width=65) (actual time=138.523..428.773 rows=37034 loops=3)
         Hash Cond: (t.kind_id = kt.id)
         ->  Nested Loop  (cost=4798.99..44015.25 rows=76 width=69) (actual time=71.000..422.457 rows=44730 loops=3)
               ->  Hash Join  (cost=4798.56..43980.32 rows=76 width=60) (actual time=70.982..347.084 rows=44730 loops=3)
               Hash Cond: (mi.info_type_id = it2.id)
               ->  Nested Loop  (cost=4796.13..43954.56 rows=8566 width=64) (actual time=70.925..330.473 rows=220915 loops=3)
               ->  Hash Join  (cost=4795.70..43433.79 rows=274 width=14) (actual time=70.902..196.418 rows=10503 loops=3)
               Hash Cond: (miidx.info_type_id = it.id)
               ->  Nested Loop  (cost=4793.27..43347.09 rows=30943 width=18) (actual time=70.880..193.050 rows=31523 loops=3)
                     ->  Hash Join  (cost=4792.84..37544.66 rows=10990 width=4) (actual time=70.814..149.115 rows=23765 loops=3)
                     Hash Cond: (mc.company_type_id = ct.id)
                     ->  Parallel Hash Join  (cost=4791.78..37305.94 rows=43958 width=8) (actual time=8.571..144.378 rows=49377 loops=3)
                     Hash Cond: (mc.company_id = cn.id)
                     ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..29660.37 rows=1087137 width=12) (actual time=0.012..56.202 rows=869710 loops=3)
                     ->  Parallel Hash  (cost=4721.92..4721.92 rows=5589 width=4) (actual time=8.383..8.384 rows=3258 loops=3)
                           Buckets: 16384  Batches: 1  Memory Usage: 544kB
                           ->  Parallel Seq Scan on company_name cn  (cost=0.00..4721.92 rows=5589 width=4) (actual time=0.009..7.826 rows=3258 loops=3)
                           Filter: ((country_code)::text = '[de]'::text)
                           Rows Removed by Filter: 75074
                     ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.006..0.007 rows=1 loops=3)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.005..0.005 rows=1 loops=3)
                           Filter: ((kind)::text = 'production companies'::text)
                           Rows Removed by Filter: 3
                     ->  Index Scan using movie_id_movie_info_idx on movie_info_idx miidx  (cost=0.43..0.50 rows=3 width=14) (actual time=0.001..0.002 rows=1 loops=71294)
                     Index Cond: (movie_id = mc.movie_id)
               ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.011..0.011 rows=1 loops=3)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.008..0.009 rows=1 loops=3)
                     Filter: ((info)::text = 'rating'::text)
                     Rows Removed by Filter: 112
               ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.48 rows=42 width=50) (actual time=0.003..0.011 rows=21 loops=31510)
               Index Cond: (movie_id = mc.movie_id)
               ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.015..0.016 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.008..0.013 rows=1 loops=3)
               Filter: ((info)::text = 'release dates'::text)
               Rows Removed by Filter: 112
               ->  Index Scan using title_pkey on title t  (cost=0.43..0.46 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=134189)
               Index Cond: (id = mi.movie_id)
         ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.013..0.013 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.009..0.010 rows=1 loops=3)
               Filter: ((kind)::text = 'movie'::text)
               Rows Removed by Filter: 6
```

Planning Time: 54.386 ms

Execution Time: 446.338 ms

QUERY PLAN

```
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=56848.43..56848.44 rows=1 width=96) (actual time=796.287..799.621 rows=1 loops=1)
   ->  Gather  (cost=56848.20..56848.41 rows=2 width=96) (actual time=796.279..799.615 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=55848.20..55848.21 rows=1 width=96) (actual time=793.114..793.121 rows=1 loops=3)
         ->  Hash Join  (cost=38087.36..55501.01 rows=46292 width=65) (actual time=209.505..784.926 rows=37034 loops=3)
         Hash Cond: (mc.company_type_id = ct.id)
         ->  Hash Join  (cost=38086.30..55499.62 rows=61 width=69) (actual time=209.215..772.258 rows=185437 loops=3)
               Hash Cond: (mi.info_type_id = it2.id)
               ->  Nested Loop  (cost=38083.87..55478.48 rows=6873 width=73) (actual time=209.177..705.145 rows=1042784 loops=3)
               Join Filter: (t.id = mi.movie_id)
               ->  Hash Join  (cost=38083.44..55094.99 rows=220 width=39) (actual time=209.144..329.985 rows=16213 loops=3)
               Hash Cond: (t.kind_id = kt.id)
               ->  Nested Loop  (cost=38082.34..55087.40 rows=1538 width=43) (actual time=144.179..325.886 rows=27966 loops=3)
               ->  Parallel Hash Join  (cost=38081.91..54293.04 rows=1538 width=18) (actual time=144.152..250.758 rows=27966 loops=3)
                     Hash Cond: (miidx.movie_id = mc.movie_id)
                     ->  Hash Join  (cost=2.43..15253.60 rows=191635 width=10) (actual time=0.045..73.090 rows=153308 loops=3)
                     Hash Cond: (miidx.info_type_id = it.id)
                     ->  Parallel Seq Scan on movie_info_idx miidx  (cost=0.00..13685.15 rows=575015 width=14) (actual time=0.015..30.857 rows=460012 loops=3)
                     ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.015..0.016 rows=1 loops=3)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.011..0.012 rows=1 loops=3)
                           Filter: ((info)::text = 'rating'::text)
                           Rows Removed by Filter: 112
                     ->  Parallel Hash  (cost=37307.96..37307.96 rows=61722 width=8) (actual time=143.594..143.597 rows=49377 loops=3)
                     Buckets: 262144  Batches: 1  Memory Usage: 7904kB
                     ->  Parallel Hash Join  (cost=4793.79..37307.96 rows=61722 width=8) (actual time=9.859..134.904 rows=49377 loops=3)
                     Hash Cond: (mc.company_id = cn.id)
                     ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..29660.37 rows=1087137 width=12) (actual time=0.014..55.117 rows=869710 loops=3)
                     ->  Parallel Hash  (cost=4721.92..4721.92 rows=5750 width=4) (actual time=9.748..9.749 rows=3258 loops=3)
                           Buckets: 16384  Batches: 1  Memory Usage: 576kB
                           ->  Parallel Seq Scan on company_name cn  (cost=0.00..4721.92 rows=5750 width=4) (actual time=0.013..9.158 rows=3258 loops=3)
                           Filter: ((country_code)::text = '[de]'::text)
                           Rows Removed by Filter: 75074
               ->  Index Scan using title_pkey on title t  (cost=0.43..0.52 rows=1 width=25) (actual time=0.002..0.002 rows=1 loops=83899)
                     Index Cond: (id = mc.movie_id)
               ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.008..0.008 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.006..0.006 rows=1 loops=3)
                     Filter: ((kind)::text = 'movie'::text)
                     Rows Removed by Filter: 6
               ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.48 rows=21 width=50) (actual time=0.003..0.016 rows=64 loops=48640)
               Index Cond: (movie_id = mc.movie_id)
               ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.014..0.015 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.007..0.012 rows=1 loops=3)
               Filter: ((info)::text = 'release dates'::text)
               Rows Removed by Filter: 112
         ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.015..0.016 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.012..0.013 rows=1 loops=3)
               Filter: ((kind)::text = 'production companies'::text)
               Rows Removed by Filter: 3
```

Planning Time: 52.883 ms
Execution Time: 800.859 ms

QUERY PLAN

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Finalize Aggregate  (cost=62909.11..62909.12 rows=1 width=96) (actual time=471.929..475.366 rows=1 loops=1)
  ->  Gather  (cost=62908.89..62909.10 rows=2 width=96) (actual time=471.542..475.358 rows=3 loops=1)
        Workers Planned: 2
        Workers Launched: 2
        ->  Partial Aggregate  (cost=61908.89..61908.90 rows=1 width=96) (actual time=469.143..469.148 rows=1 loops=3)
        ->  Hash Join  (cost=4802.10..61561.70 rows=46292 width=65) (actual time=120.574..459.759 rows=37034 loops=3)
        Hash Cond: (miidx.info_type_id = it.id)
        ->  Nested Loop  (cost=4799.67..61550.27 rows=3308 width=69) (actual time=120.503..450.410 rows=111685 loops=3)
              Join Filter: (mi.movie_id = miidx.movie_id)
              ->  Hash Join  (cost=4799.25..60968.84 rows=1139 width=71) (actual time=120.440..370.111 rows=42998 loops=3)
              Hash Cond: (mi.info_type_id = it2.id)
              ->  Nested Loop  (cost=4796.82..60615.80 rows=128740 width=75) (actual time=120.376..353.587 rows=220499 loops=3)
              Join Filter: (t.id = mi.movie_id)
              ->  Hash Join  (cost=4796.39..53111.84 rows=4244 width=25) (actual time=120.347..219.486 rows=10951 loops=3)
              Hash Cond: (t.kind_id = kt.id)
              ->  Nested Loop  (cost=4795.29..52985.55 rows=29706 width=29) (actual time=73.198..216.449 rows=23765 loops=3)
                    ->  Hash Join  (cost=4794.86..37642.70 rows=29706 width=4) (actual time=73.159..151.142 rows=23765 loops=3)
                    Hash Cond: (mc.company_type_id = ct.id)
                    ->  Parallel Hash Join  (cost=4793.79..37307.96 rows=61722 width=8) (actual time=9.937..146.294 rows=49377 loops=3)
                    Hash Cond: (mc.company_id = cn.id)
                    ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..29660.37 rows=1087137 width=12) (actual time=0.013..57.188 rows=869710 loops=3)
                    ->  Parallel Hash  (cost=4721.92..4721.92 rows=5750 width=4) (actual time=9.764..9.764 rows=3258 loops=3)
                          Buckets: 16384  Batches: 1  Memory Usage: 576kB
                          ->  Parallel Seq Scan on company_name cn  (cost=0.00..4721.92 rows=5750 width=4) (actual time=0.012..9.166 rows=3258 loops=3)
                          Filter: ((country_code)::text = '[de]'::text)
                          Rows Removed by Filter: 75074
                    ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.005..0.006 rows=1 loops=3)
                          Filter: ((kind)::text = 'production companies'::text)
                          Rows Removed by Filter: 3
                    ->  Index Scan using title_pkey on title t  (cost=0.43..0.52 rows=1 width=25) (actual time=0.002..0.002 rows=1 loops=71294)
                    Index Cond: (id = mc.movie_id)
              ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.010..0.010 rows=1 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.008..0.008 rows=1 loops=3)
                    Filter: ((kind)::text = 'movie'::text)
                    Rows Removed by Filter: 6
              ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.48 rows=23 width=50) (actual time=0.003..0.010 rows=20 loops=32853)
              Index Cond: (movie_id = mc.movie_id)
              ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.011..0.011 rows=1 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.004..0.009 rows=1 loops=3)
              Filter: ((info)::text = 'release dates'::text)
              Rows Removed by Filter: 112
              ->  Index Scan using movie_id_movie_info_idx on movie_info_idx miidx  (cost=0.43..0.50 rows=1 width=14) (actual time=0.001..0.001 rows=3 loops=128993)
              Index Cond: (movie_id = mc.movie_id)
        ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.021..0.021 rows=1 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.018..0.019 rows=1 loops=3)
              Filter: ((info)::text = 'rating'::text)
              Rows Removed by Filter: 112
```

Planning Time: 52.142 ms
Execution Time: 476.792 ms

```
                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------
Finalize Aggregate  (cost=72802.97..72802.98 rows=1 width=96) (actual time=497.952..501.376 rows=1 loops=1)
  ->  Gather  (cost=72802.75..72802.96 rows=2 width=96) (actual time=497.607..501.366 rows=3 loops=1)
        Workers Planned: 2
        Workers Launched: 2
        ->  Partial Aggregate  (cost=71802.75..71802.76 rows=1 width=96) (actual time=494.947..494.952 rows=1 loops=3)
        ->  Hash Join  (cost=4802.10..71455.56 rows=46292 width=65) (actual time=120.047..486.851 rows=37034 loops=3)
        Hash Cond: (miidx.info_type_id = it.id)
        ->  Hash Join  (cost=4799.67..71072.93 rows=139606 width=69) (actual time=119.909..477.976 rows=111685 loops=3)
              Hash Cond: (mi.info_type_id = it2.id)
              ->  Nested Loop  (cost=4797.25..70052.54 rows=373777 width=73) (actual time=119.876..442.275 rows=520885 loops=3)
              Join Filter: (t.id = mi.movie_id)
              ->  Nested Loop  (cost=4796.81..60099.72 rows=5629 width=39) (actual time=119.855..240.656 rows=16483 loops=3)
              Join Filter: (t.id = miidx.movie_id)
              ->  Hash Join  (cost=4796.39..53111.84 rows=13689 width=25) (actual time=119.772..216.804 rows=10951 loops=3)
              Hash Cond: (t.kind_id = kt.id)
              ->  Nested Loop  (cost=4795.29..52985.55 rows=29706 width=29) (actual time=72.763..213.854 rows=23765 loops=3)
                    ->  Hash Join  (cost=4794.86..37642.70 rows=29706 width=4) (actual time=72.725..149.540 rows=23765 loops=3)
                    Hash Cond: (mc.company_type_id = ct.id)
                    ->  Parallel Hash Join  (cost=4793.79..37307.96 rows=61722 width=8) (actual time=10.158..144.766 rows=49377 loops=3)
                    Hash Cond: (mc.company_id = cn.id)
                    ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..29660.37 rows=1087137 width=12) (actual time=0.012..56.315 rows=869710 loops=3)
                    ->  Parallel Hash  (cost=4721.92..4721.92 rows=5750 width=4) (actual time=9.984..9.985 rows=3258 loops=3)
                          Buckets: 16384  Batches: 1  Memory Usage: 576kB
                          ->  Parallel Seq Scan on company_name cn  (cost=0.00..4721.92 rows=5750 width=4) (actual time=0.012..9.344 rows=3258 loops=3)
                          Filter: ((country_code)::text = '[de]'::text)
                          Rows Removed by Filter: 75074
                    ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.005..0.005 rows=1 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.003..0.004 rows=1 loops=3)
                          Filter: ((kind)::text = 'production companies'::text)
                          Rows Removed by Filter: 3
                    ->  Index Scan using title_pkey on title t  (cost=0.43..0.52 rows=1 width=25) (actual time=0.002..0.002 rows=1 loops=71294)
                    Index Cond: (id = mc.movie_id)
              ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.008..0.009 rows=1 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.006..0.007 rows=1 loops=3)
                    Filter: ((kind)::text = 'movie'::text)
                    Rows Removed by Filter: 6
              ->  Index Scan using movie_id_movie_info_idx on movie_info_idx miidx  (cost=0.43..0.50 rows=1 width=14) (actual time=0.002..0.002 rows=2 loops=32853)
              Index Cond: (movie_id = mc.movie_id)
              ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.48 rows=23 width=50) (actual time=0.002..0.009 rows=32 loops=49448)
              Index Cond: (movie_id = mc.movie_id)
              ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.011..0.011 rows=1 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.004..0.009 rows=1 loops=3)
              Filter: ((info)::text = 'release dates'::text)
              Rows Removed by Filter: 112
        ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.021..0.021 rows=1 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.017..0.018 rows=1 loops=3)
              Filter: ((info)::text = 'rating'::text)
              Rows Removed by Filter: 112
```

Planning Time: 54.055 ms
Execution Time: 502.734 ms

```
QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------
Finalize Aggregate  (cost=73834.19..73834.20 rows=1 width=96) (actual time=358.340..361.647 rows=1 loops=1)
  ->  Gather  (cost=73833.97..73834.18 rows=2 width=96) (actual time=358.037..361.640 rows=3 loops=1)
        Workers Planned: 2
        Workers Launched: 2
        ->  Partial Aggregate  (cost=72833.97..72833.98 rows=1 width=96) (actual time=355.503..355.508 rows=1 loops=3)
        ->  Hash Join  (cost=4802.10..72486.78 rows=46292 width=65) (actual time=119.616..347.321 rows=37034 loops=3)
        Hash Cond: (mi.info_type_id = it2.id)
        ->  Nested Loop  (cost=4799.67..71898.44 rows=215136 width=69) (actual time=119.486..335.296 rows=172109 loops=3)
              Join Filter: (t.id = mi.movie_id)
              ->  Hash Join  (cost=4799.24..59679.74 rows=6862 width=35) (actual time=119.457..233.297 rows=5490 loops=3)
              Hash Cond: (t.kind_id = kt.id)
              ->  Nested Loop  (cost=4798.14..59623.31 rows=13129 width=39) (actual time=73.689..231.693 rows=10503 loops=3)
              ->  Hash Join  (cost=4797.71..52842.31 rows=13129 width=14) (actual time=73.665..197.705 rows=10503 loops=3)
              Hash Cond: (miidx.info_type_id = it.id)
              ->  Nested Loop  (cost=4795.28..52732.57 rows=39403 width=18) (actual time=73.644..194.510 rows=31523 loops=3)
                    ->  Hash Join  (cost=4794.86..37642.70 rows=29706 width=4) (actual time=73.564..149.878 rows=23765 loops=3)
                    Hash Cond: (mc.company_type_id = ct.id)
                    ->  Parallel Hash Join  (cost=4793.79..37307.96 rows=61722 width=8) (actual time=9.862..145.146 rows=49377 loops=3)
                    Hash Cond: (mc.company_id = cn.id)
                    ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..29660.37 rows=1087137 width=12) (actual time=0.011..57.195 rows=869710 loops=3)
                    ->  Parallel Hash  (cost=4721.92..4721.92 rows=5750 width=4) (actual time=9.737..9.737 rows=3258 loops=3)
                          Buckets: 16384  Batches: 1  Memory Usage: 576kB
                          ->  Parallel Seq Scan on company_name cn  (cost=0.00..4721.92 rows=5750 width=4) (actual time=0.012..9.129 rows=3258 loops=3)
                          Filter: ((country_code)::text = '[de]'::text)
                          Rows Removed by Filter: 75074
                    ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.008..0.008 rows=1 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=3)
                          Filter: ((kind)::text = 'production companies'::text)
                          Rows Removed by Filter: 3
                    ->  Index Scan using movie_id_movie_info_idx on movie_info_idx miidx  (cost=0.43..0.50 rows=1 width=14) (actual time=0.001..0.002 rows=1 loops=71294)
                    Index Cond: (movie_id = mc.movie_id)
              ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.011..0.011 rows=1 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.009..0.009 rows=1 loops=3)
                    Filter: ((info)::text = 'rating'::text)
                    Rows Removed by Filter: 112
              ->  Index Scan using title_pkey on title t  (cost=0.43..0.52 rows=1 width=25) (actual time=0.003..0.003 rows=1 loops=31510)
              Index Cond: (id = mc.movie_id)
              ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.008..0.009 rows=1 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=3)
              Filter: ((kind)::text = 'movie'::text)
              Rows Removed by Filter: 6
              ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.48 rows=24 width=50) (actual time=0.003..0.015 rows=31 loops=16470)
              Index Cond: (movie_id = mc.movie_id)
        ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.021..0.021 rows=1 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.014..0.019 rows=1 loops=3)
              Filter: ((info)::text = 'release dates'::text)
              Rows Removed by Filter: 112
```

Planning Time: 51.492 ms

Execution Time: 362.949 ms

```
                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=73834.19..73834.20 rows=1 width=96) (actual time=356.744..359.896 rows=1 loops=1)
   ->  Gather  (cost=73833.97..73834.18 rows=2 width=96) (actual time=356.724..359.890 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=72833.97..72833.98 rows=1 width=96) (actual time=353.917..353.922 rows=1 loops=3)
         ->  Hash Join  (cost=4802.10..72486.78 rows=46292 width=65) (actual time=119.148..345.717 rows=37034 loops=3)
         Hash Cond: (mi.info_type_id = it2.id)
         ->  Nested Loop  (cost=4799.67..71898.44 rows=215136 width=69) (actual time=119.012..333.552 rows=172109 loops=3)
               Join Filter: (t.id = mi.movie_id)
               ->  Hash Join  (cost=4799.24..59679.74 rows=6862 width=35) (actual time=118.983..231.986 rows=5490 loops=3)
               Hash Cond: (t.kind_id = kt.id)
               ->  Nested Loop  (cost=4798.14..59623.31 rows=13129 width=39) (actual time=73.336..230.384 rows=10503 loops=3)
               ->  Hash Join  (cost=4797.71..52842.31 rows=13129 width=14) (actual time=73.310..196.399 rows=10503 loops=3)
               Hash Cond: (miidx.info_type_id = it.id)
               ->  Nested Loop  (cost=4795.28..52732.57 rows=39403 width=18) (actual time=73.289..193.206 rows=31523 loops=3)
                     ->  Hash Join  (cost=4794.86..37642.70 rows=29706 width=4) (actual time=73.189..148.577 rows=23765 loops=3)
                     Hash Cond: (mc.company_type_id = ct.id)
                     ->  Parallel Hash Join  (cost=4793.79..37307.96 rows=61722 width=8) (actual time=10.177..143.898 rows=49377 loops=3)
                     Hash Cond: (mc.company_id = cn.id)
                     ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..29660.37 rows=1087137 width=12) (actual time=0.014..56.806 rows=869710 loops=3)
                     ->  Parallel Hash  (cost=4721.92..4721.92 rows=5750 width=4) (actual time=9.993..9.993 rows=3258 loops=3)
                           Buckets: 16384  Batches: 1  Memory Usage: 576kB
                           ->  Parallel Seq Scan on company_name cn  (cost=0.00..4721.92 rows=5750 width=4) (actual time=0.011..9.366 rows=3258 loops=3)
                           Filter: ((country_code)::text = '[de]'::text)
                           Rows Removed by Filter: 75074
                     ->  Hash  (cost=1.05..1.05 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=3)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on company_type ct  (cost=0.00..1.05 rows=1 width=4) (actual time=0.005..0.006 rows=1 loops=3)
                           Filter: ((kind)::text = 'production companies'::text)
                           Rows Removed by Filter: 3
                     ->  Index Scan using movie_id_movie_info_idx on movie_info_idx miidx  (cost=0.43..0.50 rows=1 width=14) (actual time=0.001..0.002 rows=1 loops=71294)
                     Index Cond: (movie_id = mc.movie_id)
               ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.011..0.011 rows=1 loops=3)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on info_type it  (cost=0.00..2.41 rows=1 width=4) (actual time=0.008..0.009 rows=1 loops=3)
                     Filter: ((info)::text = 'rating'::text)
                     Rows Removed by Filter: 112
               ->  Index Scan using title_pkey on title t  (cost=0.43..0.52 rows=1 width=25) (actual time=0.003..0.003 rows=1 loops=31510)
               Index Cond: (id = mc.movie_id)
               ->  Hash  (cost=1.09..1.09 rows=1 width=4) (actual time=0.008..0.009 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on kind_type kt  (cost=0.00..1.09 rows=1 width=4) (actual time=0.006..0.007 rows=1 loops=3)
               Filter: ((kind)::text = 'movie'::text)
               Rows Removed by Filter: 6
               ->  Index Scan using movie_id_movie_info on movie_info mi  (cost=0.43..1.48 rows=24 width=50) (actual time=0.003..0.015 rows=31 loops=16470)
               Index Cond: (movie_id = mc.movie_id)
         ->  Hash  (cost=2.41..2.41 rows=1 width=4) (actual time=0.021..0.021 rows=1 loops=3)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on info_type it2  (cost=0.00..2.41 rows=1 width=4) (actual time=0.013..0.019 rows=1 loops=3)
               Filter: ((info)::text = 'release dates'::text)
               Rows Removed by Filter: 112
```

Planning Time: 54.904 ms
Execution Time: 361.197 ms

AQO now

- ➢ Have a production code: https://github.com/postgrespro/aqo
- ➢ Is a part of PostgresPro Enterprise
- ➢ Helps DBA to find a better plan for the query

Adaptive Cardinality Estimation: ongoing research

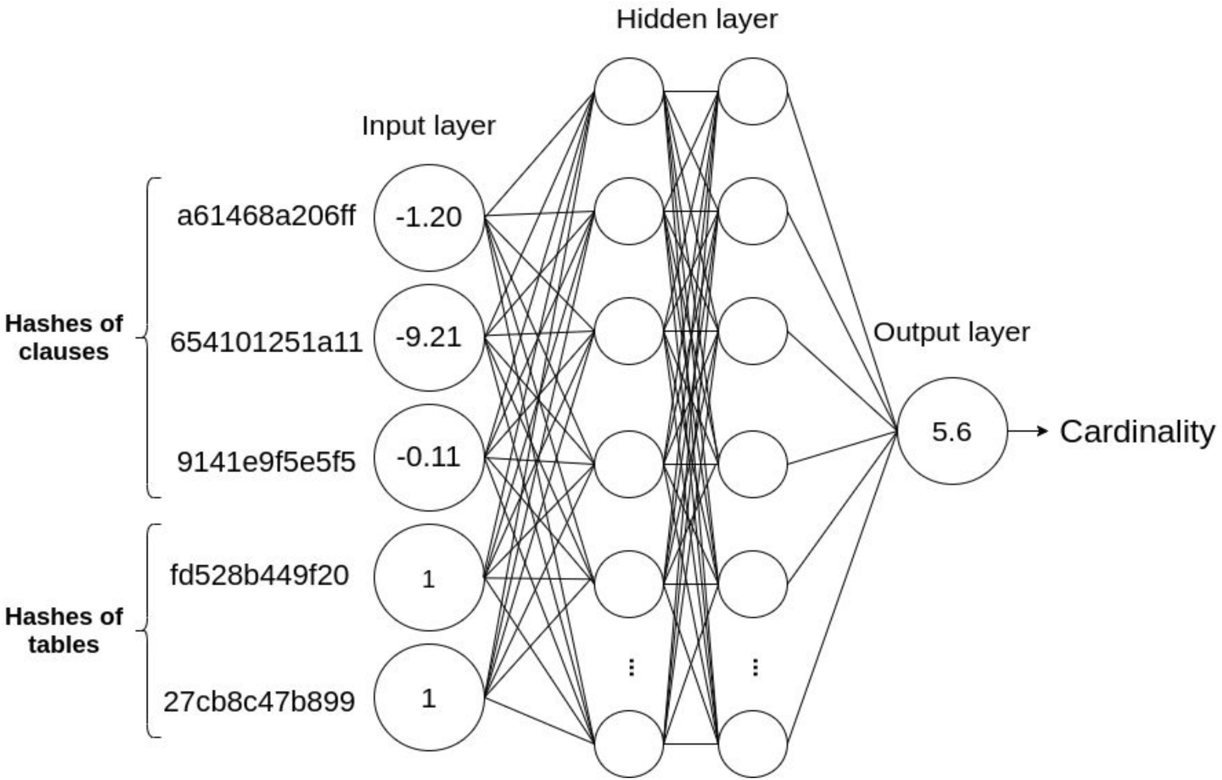
K Nearest Neighbours: challenges

- ➢ Memory consumption
- ➢ Stability = low generalization ability = local optima

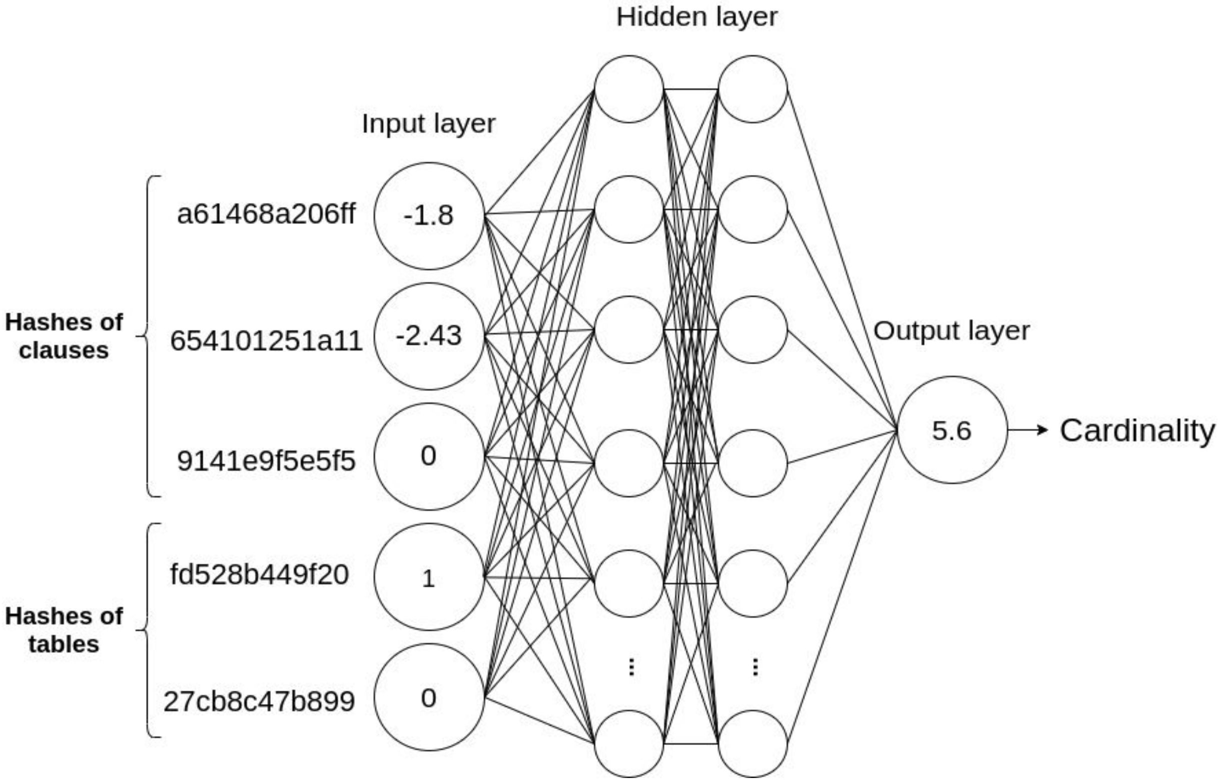
Neural Networks

Clauses info	Relations hashes	Log Cardinality
(a61468a206ff, -1.20) (654101251a11, -9.21) (9141e9f5e5f5, -0.11)	fd528b449f20 27cb8c47b899	7.08

Neural Networks



Neural Networks

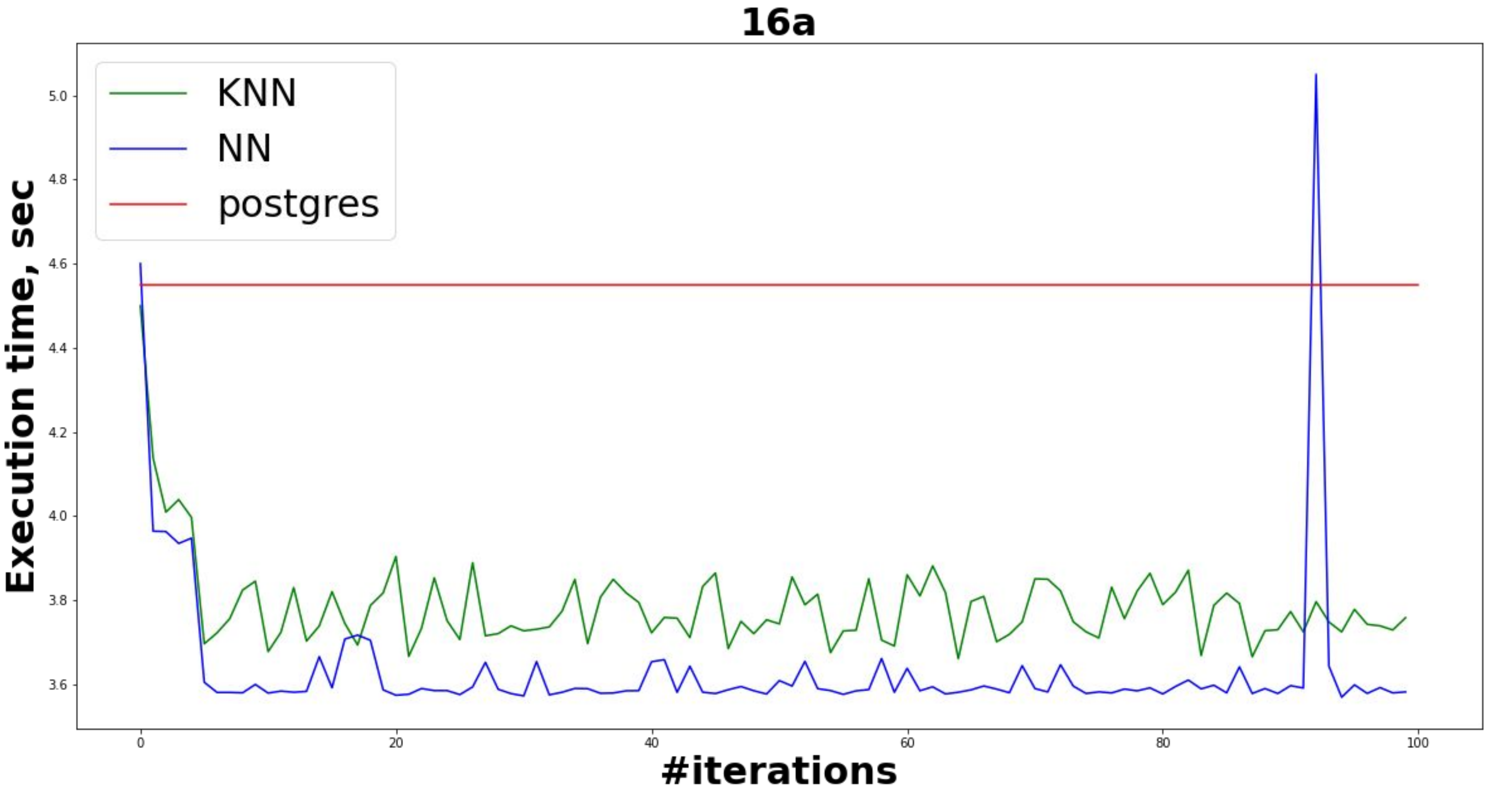


Neural Networks: Results

- Join Order Benchmark*:
  - Maximum performance boost is 2x
  - Mean execution time decreased by 70%

- Join Order Benchmark* - joint:
  - Maximum performance boost is 2x
  - Mean execution time decreased by 40%

Neural Networks: Results



Neural Networks:

- ➢ are less stable
- ➢ generalize better and find better plans
- ➢ can use information from other queries

Challenges

Challenges: low level

- Better regressor
- Better support for complex clauses
- “Learnable histograms” for uncommon operators and data types
- Utilization of rolled back queries statistics

Challenges: high level

Goal: to select the best plan *automatically*

Means:

- ➢ Meta-optimization
- ➢ Plans space exploration
- ➢ Interruption and re-execution of bad plans
- ➢ ...

Major Contributors

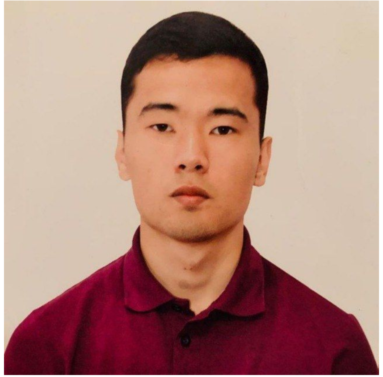
PostgresPro

Company where AQO was created and evolved

Oleg Bartunov
Vision, Idea

Oleg Ivanov
Idea, Research, First version of the code

Andrey Lepikhov
Production code refactoring and support



Yerzhaisang Taskali
AQO-NN research

Takeaways

- ➢ Cardinality estimation errors are common in OLAP workloads
- ➢ AQO uses query execution statistics for query optimization
- ➢ AQO implementation is available for everyone
  - ➢ It usually provides significant speed up for complex OLAP queries
- ➢ AQO is a framework
  - ➢ There is still a lot of work to do
    - ➢ In progress...