PostgresPro

# Постгрессовый оптимизатор с памятью

ЛЕПИХОВ Андрей
Postgres Professional
a.lepikhov@postgrespro.ru

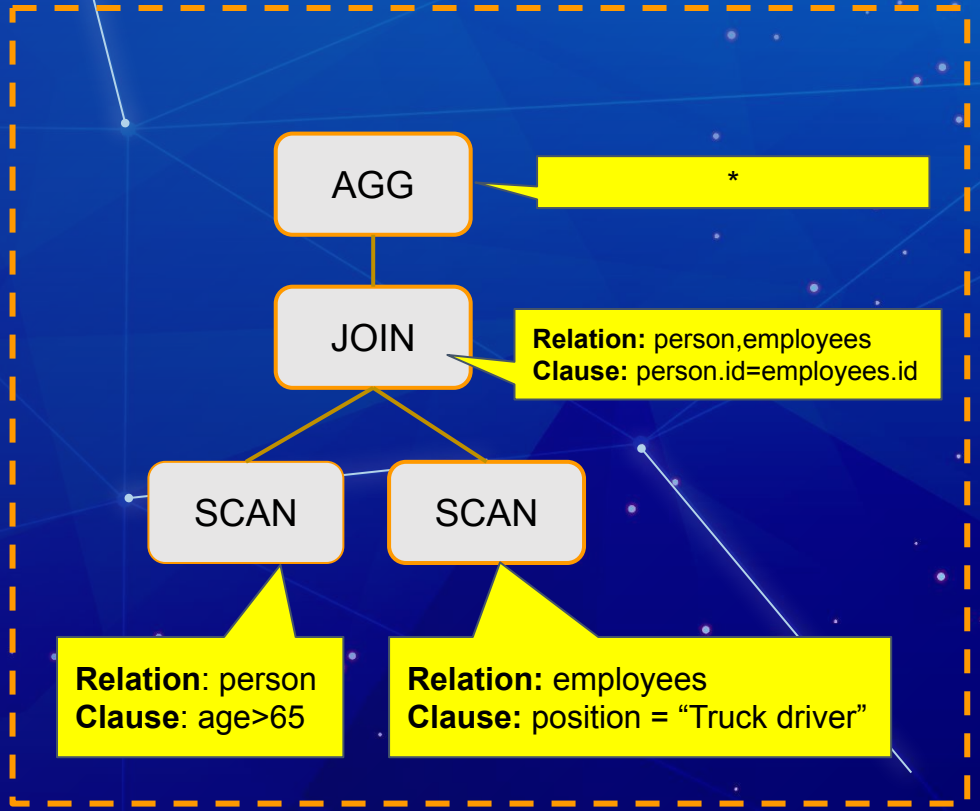
# Адаптивная оптимизация

- **A**daptive **Q**uery **O**ptimization (AQO), 2016, tigvarts
- Использует знания о ранее выполнявшихся запросах для оптимизации новых
- Хранит сведения о результатах выполнения запросов в компактной форме

Репозиторий AQO на github:

# Система сигнатур

```
SELECT count(*) FROM person
JOIN employees USING (id)
WHERE
  position = “Truck driver”
  AND age>65
```




SQL-схема данных

# Система сигнатур

**SCAN:** {person} {age, <} → $FSS_1$

**SCAN:** {employees} {age, <} → $FSS_2$

**JOIN:** {employees,person} {age, <, position, =} {p.id, e.id, =} → $FSS_3$

**Query:** nodeToString(<Parsed query>) → FS

**Disk storage:**

| FS | FSS | Feature selectivity 1 … Feature selectivity N | Cardinality |
|---|---|---|---|
| 1234567 | 7654321 | 0.321 | 300 |
| 1234567 | -2436564 | 0.5, 0.321 | 545 |

# Взаимодействие с оптимизатором

- При выборе плана запроса - подстановка кардинальности через перехват хуков оптимизатора
- После завершения выполнения запроса - анализ плана выполненного запроса через хук executor’a

# Функциональные зависимости

```
EXPLAIN (ANALYZE, SUMMARY OFF)
     SELECT id FROM person WHERE age<18;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Seq Scan on person  (rows=286 width=4) (actual time=0.022..0.355 rows=286 loops=1)
   Filter: (age < 18)
   Rows Removed by Filter: 714
(3 rows)
```

# Функциональные зависимости

```
EXPLAIN (ANALYZE, SUMMARY OFF)
     SELECT id FROM person WHERE age<18;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Seq Scan on person  (rows=286 width=4) (actual time=0.022..0.355 rows=286 loops=1)
   Filter: (age < 18)
   Rows Removed by Filter: 714
(3 rows)


EXPLAIN (ANALYZE, SUMMARY OFF)
     SELECT id FROM person WHERE age<18 AND passport IS NOT NULL;
                                               QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Seq Scan on person  (rows=181 width=4) (actual time=0.204..0.204 rows=2 loops=1)
   Filter: ((passport IS NOT NULL) AND (age < 18))
   Rows Removed by Filter: 1000
(3 rows)
```

# Функциональные зависимости

Данные и SQL-запросы:

```
SET aqo.mode='learn';
SET aqo.show_details = 'on';
TRUNCATE aqo_data;
EXPLAIN (ANALYZE, SUMMARY OFF)
     SELECT id FROM person WHERE age<18 AND passport IS NOT NULL;

                                                                                 QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Seq Scan on person  (cost=0.00..19.50 rows=2 width=4) (actual time=0.219..0.219 rows=2 loops=1)
   AQO: rows=2, error=0%, fss = -1866815964
   Filter: ((passport IS NOT NULL) AND (age < 18))
   Rows Removed by Filter: 1000
 Using aqo: true
 AQO mode: LEARN
```

# Расширенная статистика

```
CREATE STATISTICS corr (dependencies) ON age, passport FROM person;
EXPLAIN (ANALYZE, SUMMARY OFF)
    SELECT id FROM person WHERE age<18 AND passport IS NOT NULL;

CREATE STATISTICS

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on person  (cost=0.00..19.50 rows=2 width=4) (actual time=0.276..0.277 rows=2 loops=1)
   Filter: ((passport IS NOT NULL) AND (age < 18))
   Rows Removed by Filter: 1000
```

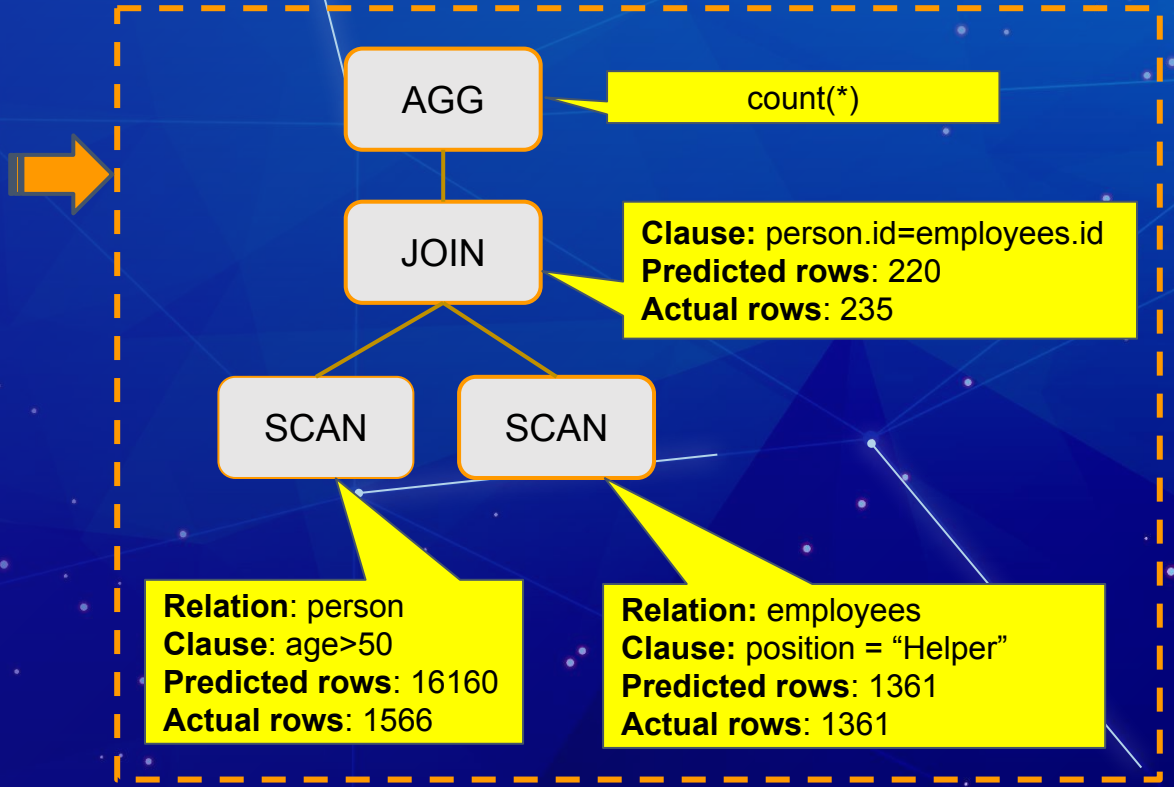
# Кардинальность JOIN’а

SQL-скрипт

- Оптимизатор не может отслеживать смысловые зависимости между таблицами
  Пример: поиск грузчиков моложе 18 лет.
- Актуализация статистики не работает
- Расширенная статистика не работает

# Кардинальность JOIN’а (без корреляций)

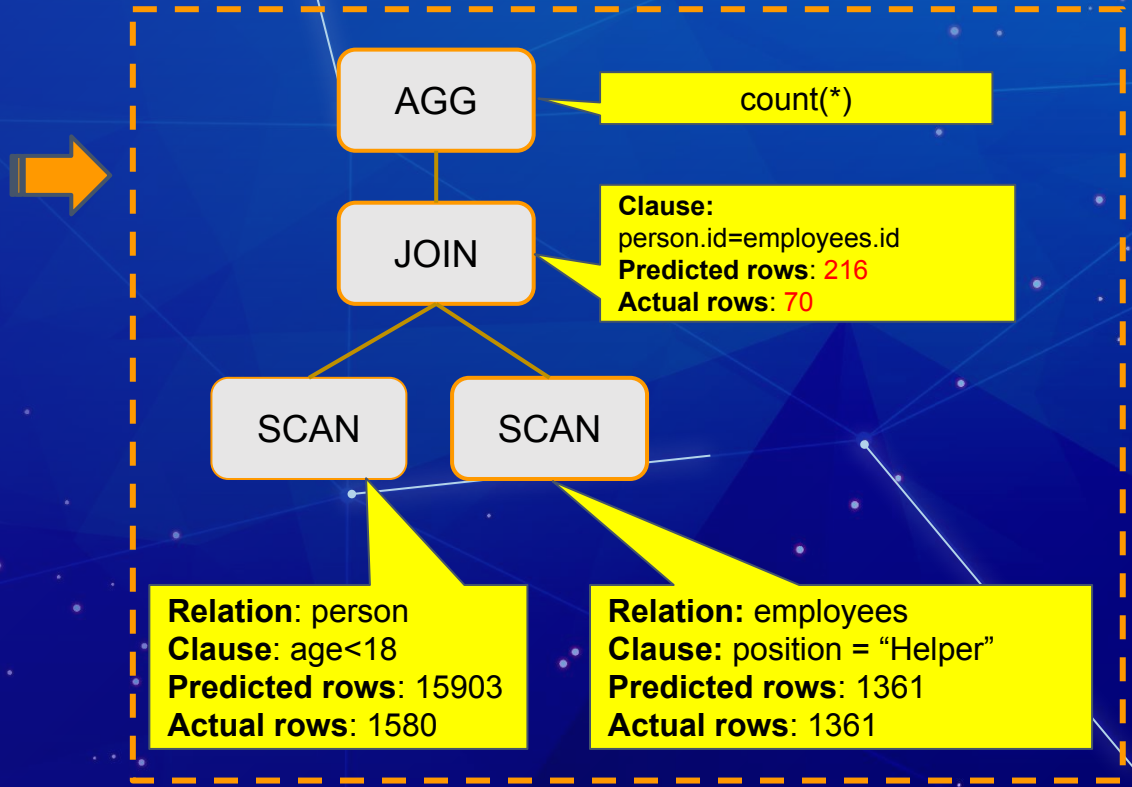
```
SELECT count(*)
      FROM person
JOIN employees USING (id)
WHERE age>50 AND
position='Helper';
```




**Запрос и план:**

# Кардинальность JOIN’а (с корреляцией)

```
SELECT count(*)
      FROM person
JOIN employees USING (id)
WHERE age<18 AND
position='Helper';
```

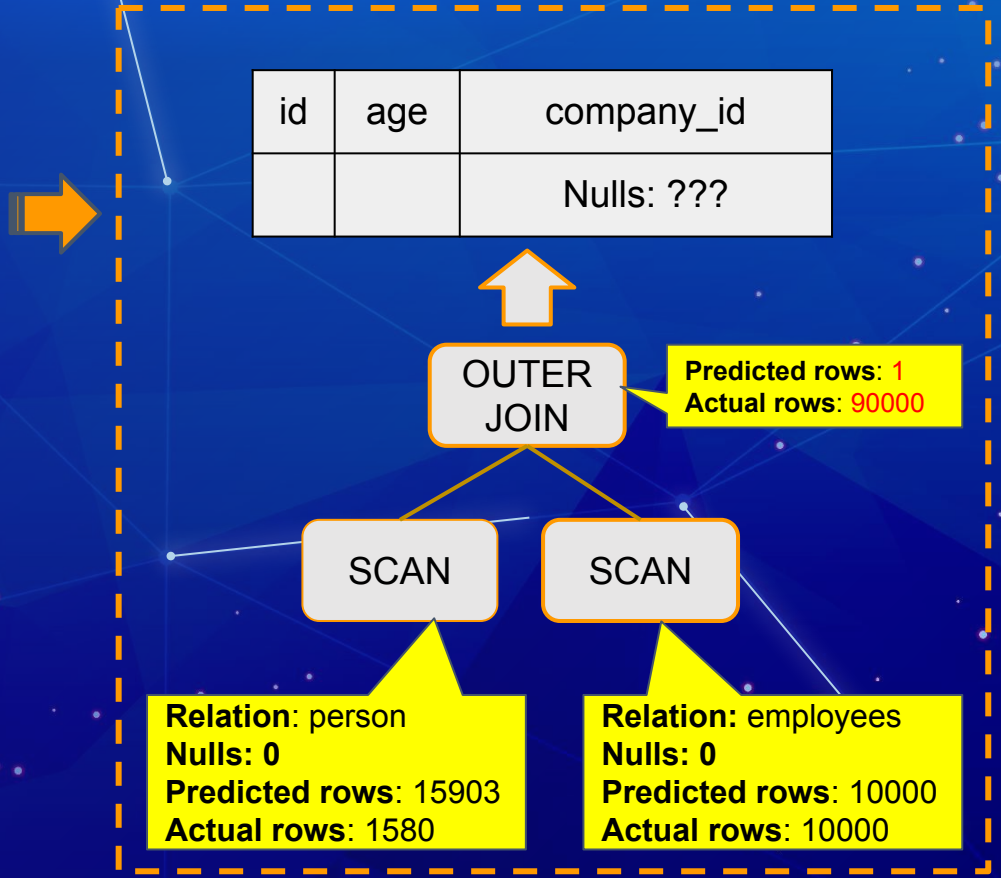



**Запрос и план:**

# Проблема OUTER JOIN

**Запрос:**
**“определить количество неработающих граждан”**

```
SELECT count(*) FROM (
     SELECT id,age,cid
     FROM person
     LEFT JOIN employees USING (id)
) AS q1 WHERE cid IS NULL;
```




**Данные и запросы:**

# Проблема OUTER JOIN
**(к чему приводит)**

**Запрос: “количество людей с инвалидностью среди неработающих”.**

```
SELECT count(*) FROM disabled JOIN (
     SELECT id,age FROM (
          SELECT id,age,cid
          FROM person
          LEFT JOIN employees USING (id)
     ) AS q1 WHERE cid IS NULL
) AS q2 ON person_id=id;
```

Время выполнения: 70 ms.

После уточнения кардинальности нод: 16 ms.

Заполнение таблиц и запросы:

# Проблема OUTER JOIN
(результат)

**Стандартная оптимизация:**

```
Aggregate (rows=1 loops=1)
 -> Hash Join (rows=4498)
     Hash Cond: (disabled.person_id =
person.id)
     -> Seq Scan on disabled (rows=5000)
     -> Hash (rows=90000 loops=1)
      -> Hash Left Join (rows=90000)
         Hash Cond: (person.id = employees.id)
         Filter: (employees.cid IS NULL)
         -> Seq Scan on person (rows=100000)
         -> Hash (rows=10000)
          -> Seq Scan on employees (rows=10000)
```

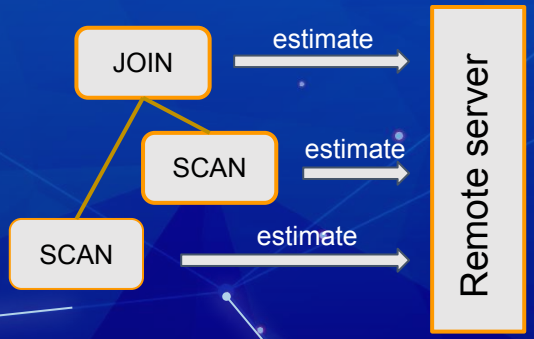
**Оптимизация с учетом знания о кардинальности:**

```
Finalize Aggregate (rows=1)
 -> Gather (rows=3)
    Workers Planned: 2
    -> Partial Aggregate (rows=1)
    -> Parallel Hash Left Join (rows=1499)
       Hash Cond: (person.id = employees.id)
       Filter: (employees.cid IS NULL)
       -> Nested Loop (rows=1667)
        -> Parallel Seq Scan on disabled (rows=1667)
         -> Index Only Scan on person (rows=1)
            Index Cond: (id = disabled.person_id)
        -> Parallel Hash (rows=3333)
          -> Seq Scan on employees (rows=3333)
```

Планы запросов

# Помогаем Foreign Data Wrappers

- “Тяжёлое” обновление статистики Foreign-таблиц
- Партиционированные таблицы
- Планирование push-down JOIN
- Не эффективный remote_estimate



# Исследование подозрительного плана запроса

```
SELECT id,age FROM employees JOIN person USING (id) WHERE position='Manager' AND cid=1;

 Merge Join  (rows=122) (actual rows=10 loops=1)
   Merge Cond: (person.id = employees.id)
   ->  Index Scan using person_pkey on person  (rows=100000) (actual rows=9738 loops=1)
   ->  Sort  (rows=122) (actual rows=10 loops=1)
         Sort Key: employees.id
         ->  Seq Scan on employees  (rows=122) (actual rows=10 loops=1)
               Filter: (("position" = 'Manager'::text) AND (cid = 1))
               Rows Removed by Filter: 9990
 Planning Time: 0.232 ms
 Execution Time: 5.389 ms
```

Заполнение таблиц
и запросы:

# Исследование подозрительного плана запроса

```
Merge Join  (rows=10) (actual rows=10 loops=1)
  AQO: rows=10, error=0%, fss = -2033561247
  Merge Cond: (person.id = employees.id)
  ->  Index Scan using person_pkey on person  (rows=9738) (actual rows=9738 loops=1)
      AQO: rows=9738, error=0%, fss = -1150030445
  ->  Sort  (rows=10) (actual rows=10 loops=1)
        (AQO not used, fss hash = 0)
        Sort Key: employees.id
        ->  Seq Scan on employees  (rows=10) (actual rows=10 loops=1)
            AQO: rows=10, error=0%, fss = 1887321922
              Filter: (("position" = 'Manager'::text) AND (cid = 1))
              Rows Removed by Filter: 9990
Planning Time: 1.363 ms
Execution Time: 7.246 ms
```

**Попробуем изменить кардинальность подозрительной ноды (`fss = -1150030445`)**

# Пробуем альтернативы

**Смотрим системное описание ноды:**

```
SELECT * FROM aqo_data WHERE fsspace_hash=-1150030445;

 fspace_hash | fsspace_hash | nfeatures | features | targets
-------------+--------------+-----------+----------+---------
  1916361181 |  -1150030445 |         0 |          | {4.25}
(1 row)
```

**Дёргаем кардинальность:**

```
UPDATE aqo_data SET targets[1]=0.01
    WHERE fsspace_hash=-1150030445;
```

**Смотрим, что получилось ...**

# Пробуем альтернативы

```
Nested Loop  (rows=10) (actual rows=10 loops=1)
     AQO: rows=10, error=0%, fss hash = -2033561247
  ->  Seq Scan on employees  (rows=10) (actual rows=10 loops=1)
       AQO: rows=10, error=0%, fss hash = 1887321922
        Filter: (("position" = 'Manager'::text) AND (cid = 1))
        Rows Removed by Filter: 9990
  ->  Index Only Scan on person  (rows=1) (actual rows=1 loops=10)
       AQO not used, fss hash = 1972255378
        Index Cond: (id = employees.id)
Planning Time: 1.384 ms
Execution Time: 3.012 ms
```

PostgresPro

# Спасибо за внимание!

ЛЕПИХОВ Андрей
Postgres Professional
a.lepikhov@postgrespro.ru

# Несвежая статистика

```
INSERT INTO person (age, passport) (SELECT 16, -1 FROM generate_series(1,1000));
ANALYZE person;
DELETE FROM person WHERE passport=-1;
```

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Seq Scan on person  (cost=0.00..37.00 rows=1291 width=4) (actual time=0.012..0.423 rows=291 loops=1)
   Filter: (age < 18)
   Rows Removed by Filter: 709
(3 rows)
```

**Если включить AQO, то при повторном запуске результат будет скорректирован ...**

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Seq Scan on person  (cost=0.00..24.50 rows=291 width=4) (actual time=0.013..0.219 rows=291 loops=1)
   Filter: (age < 18)
   Rows Removed by Filter: 709
(3 rows)
```

# Проблема OUTER JOIN

```
Aggregate  (rows=1) (actual rows=1 loops=1)
  ->  Hash Left Join  (rows=1) (actual rows=90000 loops=1)
        Hash Cond: (person.id = employees.id)
        Filter: (employees.cid IS NULL)
        Rows Removed by Filter: 10000
        ->  Seq Scan on person  (rows=100000) (actual rows=100000 loops=1)
        ->  Hash  (rows=10000) (actual rows=10000 loops=1)
              ->  Seq Scan on employees  (rows=10000) (actual rows=10000 loops=1)
Planning Time: 0.523 ms
Execution Time: 40.154 ms
```

Планнер не смог определить количество NULL-значений, созданных OUTER JOIN !