2ndQuadrant®
PostgreSQL

PostgreSQL Training
from 2ndQuadrant

EXPLAIN erklärt

Explain

Bei der Analyse von Performance-Problemen (Warum ist diese Abfrage langsam) kann eine Auswertung des Abfrageplanes via EXPLAIN helfen. Doch wie liest man dies?

Die OpenSource-Datenbank kommt mit einem leistungsfähigen Planner & Optimizer daher. Dieser berechnet mögliche Ausführungspläne einer an die Datenbank gestellten Abfrage und bewertet diese mit Kosten, um den günstigsten Abfrageplan auszuwählen.
Doch wie arbeitet dieses Kostenmodell, und wie interpretiert man einen solchen Kostenplan?
Im Vortrag sollen die grundlegende Arbeitsweise von EXPLAIN und die Auswertung von Ausführungsplänen vorgestellt werden.

About

```
test=# create table about_me (id serial primary key, info text, value text);
CREATE TABLE
test=# insert into about_me (info, value) values (
  'Name',
  'Andreas Kretschmer');
INSERT 0 1
test=# insert into about_me (info, value) values (
  'arbeitet für',
  '2ndQuadrant Deutschland GmbH');
INSERT 0 1
test=# insert into about_me (info, value) values (
  'was macht 2ndQ',
  E'operiert weltweit\n24*7*365 Support\n
    Weiterentwicklung von PostgreSQL, z.B. BDR');
INSERT 0 1
test=# select * from about_me;
 id |      info      |                   value
----+----------------+-------------------------------------------
  1 | Name           | Andreas Kretschmer
  2 | arbeitet für   | 2ndQuadrant Deutschland GmbH
  3 | was macht 2ndQ | operiert weltweit                        +
    |                | 24*7*365 Support                         +
    |                | Weiterentwicklung von PostgreSQL, z.B. BDR
(3 Zeilen)

test=#
```

Ausführung von Abfragen

- komplexe Abfragen werden in Teilschritte zerlegt
  - welche Tabellen sind abzufragen?
  - sind Aggregationen nötig?
  - ist das Resultat zu sortieren?
- für diese wird geschätzt, wie hoch der Aufwand ist
  - Anzahl der zu lesenden Blöcke
  - CPU-Aufwand
- oft gibt es mehrere Alternativen
  - Seq.-Scan oder Index, welche Indexe
  - welche Reihenfolge von Joins

Kostenmodell

- Bewertung von CPU-Aktivitäten und IO-Operationen
- wesentlicher Kostenfaktor sind Disk-Zugriffe
- Default-Modell rotiernde Festplatten
- `random_page_cost = 4` und `seq_page_cost = 1`
- SSD: `random_page_cost` geringer setzen

Voraussetzungen

- akurate Statistiken, um Ergebnissmengen zu schätzen
- diese Schätzungen können falsch sein -> schlechte Pläne
  - Statistiken zu alt
  - Stichprobe zu gering (Default: 100)
- Statistikmodell nicht perfekt -> Multi Column Stats seit 10
- wichtige Tabellen: pg_class, pg_stats

Aufruf

- EXPLAIN (schnell, nur Plan)
- EXPLAIN ANALYSE (führt Abfrage aus, estimates und actuals)
- EXPLAIN (Analyse, Buffers) (wie Explain, Ausgabe Bufferzugriffe)

Ein Blick hinter die Kulissen

```
demo=# create table explain_demo (id serial primary key, a int, b int, c int);
CREATE TABLE
demo=*# insert into explain_demo (a,b,c) select random()*10000, random()*10000, random()*10000 from generate_series(1,5) s;
INSERT 0 5
demo=*# analyse explain_demo;
ANALYZE
demo=*# explain analyse select * from explain_demo where a < 4;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on explain_demo  (cost=0.00..1.06 rows=1 width=16) (actual time=0.013..0.013 rows=0 loops=1)
   Filter: (a < 4)
   Rows Removed by Filter: 5
 Planning Time: 0.209 ms
 Execution Time: 0.042 ms
(5 rows)

demo=*# select histogram_bounds from pg_stats where tablename = 'explain_demo' and attname = 'a';
     histogram_bounds
---------------------------
 {30,4438,6792,8805,9385}
(1 row)

demo=*# select a from explain_demo order by a;
  a
------
   30
 4438
 6792
 8805
 9385
(5 rows)


demo=*# select reltuples, relpages from pg_class where relname = 'explain_demo';
 reltuples | relpages
-----------+----------
         5 |        1
(1 row)
```

Teil 2

```
demo=*# insert into explain_demo (a,b,c) select random()*10000, random()*10000, random()*10000 from generate_series(1,50000000) s;
INSERT 0 50000000
demo=*# analyse explain_demo;
ANALYZE
```

```
demo=*# select reltuples, relpages from pg_class where relname = 'explain_demo';
  reltuples  | relpages
-------------+----------
 5.00001e+07 |   270271
(1 row)

demo=*# select substring(array_to_string(histogram_bounds,','),1,50) from pg_stats where tablename = 'explain_demo' and attname = 'a';
                    substring
-----------------------------------------------------
 1,101,204,298,410,501,607,711,822,919,1031,1141,12
(1 row)
```

```
demo=*# explain analyse select * from explain_demo where a < 4;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=1000.00..533168.88 rows=14805 width=16) (actual time=6.592..5647.376 rows=17536 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Seq Scan on explain_demo  (cost=0.00..530688.38 rows=6169 width=16) (actual time=2.386..5636.224 rows=5845 loops=3)
         Filter: (a < 4)
         Rows Removed by Filter: 16660823
 Planning Time: 0.344 ms
 Execution Time: 5649.515 ms
(8 rows)
```

Beispiel: sequential Scan

```
demo=*# explain analyse select * from explain_demo where a = 4711;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=1000.00..531688.47 rows=1 width=16) (actual time=3.854..2364.220 rows=4880 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Seq Scan on explain_demo  (cost=0.00..530688.38 rows=1 width=16) (actual time=2.793..2356.790 rows=1627 loops=3)
         Filter: (a = 4711)
         Rows Removed by Filter: 16665042
 Planning Time: 0.154 ms
 Execution Time: 2364.703 ms
(8 rows)
```

Beispiel: Index-Scan

```
demo=*# explain analyse select * from explain_demo where id = 4711;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using explain_demo_pkey on explain_demo  (cost=0.56..8.58 rows=1 width=16) (actual time=4.214..4.218 rows=1 loops=1)
   Index Cond: (id = 4711)
 Planning Time: 0.117 ms
 Execution Time: 4.268 ms
(4 rows)

demo=*#
```

Beispiel: Bitmap Index Scan

```
test=# create index explain_a on explain_demo (a);
CREATE INDEX
test=# create index explain_b on explain_demo (b);
CREATE INDEX
test=# create index explain_c on explain_demo (c);
CREATE INDEX
test=# explain analyse select * from explain_demo where a = 4711 and b = 4711 and c = 4711;
                                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on explain_demo  (cost=188.17..192.19 rows=1 width=16) (actual time=6.293..6.293 rows=0 loops=1)
   Recheck Cond: ((b = 4711) AND (c = 4711))
   Filter: (a = 4711)
   Rows Removed by Filter: 1
   Heap Blocks: exact=1
   ->  BitmapAnd  (cost=188.17..188.17 rows=1 width=0) (actual time=6.208..6.208 rows=0 loops=1)
         ->  Bitmap Index Scan on explain_b  (cost=0.00..93.82 rows=4967 width=0) (actual time=2.779..2.779 rows=5074 loops=1)
               Index Cond: (b = 4711)
         ->  Bitmap Index Scan on explain_c  (cost=0.00..94.10 rows=5005 width=0) (actual time=2.711..2.711 rows=4997 loops=1)
               Index Cond: (c = 4711)
 Planning Time: 4.017 ms
 Execution Time: 6.338 ms
(12 Zeilen)
```

Beispiel: komb. Indexe

```
test=# create index explain_abc on explain_demo (a,b,c);
CREATE INDEX
test=# explain analyse select * from explain_demo where a = 4711 and b = 4711 and c = 4711;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using explain_abc on explain_demo  (cost=0.56..8.59 rows=1 width=16) (actual time=0.065..0.065 rows=0 loops=1)
   Index Cond: ((a = 4711) AND (b = 4711) AND (c = 4711))
 Planning Time: 0.470 ms
 Execution Time: 0.099 ms
(4 Zeilen)

test=#
```

Beispiel: Index (Only) Scan

```
test=# explain analyse select * from explain_demo where a between 4711 and 5120 and b between 4711 and 5120;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using explain_abc on explain_demo  (cost=0.56..351359.90 rows=83713 width=16) (actual time=0.053..2532.441 rows=84337 loops=1)
   Index Cond: ((a >= 4711) AND (a <= 5120) AND (b >= 4711) AND (b <= 5120))
 Planning Time: 0.300 ms
 Execution Time: 2542.144 ms
(4 Zeilen)

test=# explain analyse select a,b,c from explain_demo where a between 4711 and 5120 and b between 4711 and 5120;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using explain_abc on explain_demo  (cost=0.56..61412.42 rows=83713 width=12) (actual time=0.079..161.366 rows=84337 loops=1)
   Index Cond: ((a >= 4711) AND (a <= 5120) AND (b >= 4711) AND (b <= 5120))
   Heap Fetches: 0
 Planning Time: 0.109 ms
 Execution Time: 165.920 ms
(5 Zeilen)

test=#
```

Beispiel: Aggregation

```
demo=*# set max_parallel_workers_per_gather=0;
SET
demo=*# explain analyse select avg(a), sum(a) from explain_demo;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=1020273.04..1020273.05 rows=1 width=40) (actual time=11204.195..11204.195 rows=1 loops=1)
   ->  Seq Scan on explain_demo  (cost=0.00..770272.36 rows=50000136 width=4) (actual time=0.095..5712.087 rows=50000005 loops=1)
 Planning Time: 0.102 ms
 Execution Time: 11204.250 ms
(4 rows)

demo=*# set max_parallel_workers_per_gather=2;
SET
demo=*# show parallel_leader_participation;
 parallel_leader_participation
-------------------------------
 on
(1 row)

demo=*# explain analyse select avg(a), sum(a) from explain_demo;
                                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=583772.07..583772.08 rows=1 width=40) (actual time=4796.334..4796.334 rows=1 loops=1)
   ->  Gather  (cost=583771.85..583772.06 rows=2 width=40) (actual time=4796.237..4798.734 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=582771.85..582771.86 rows=1 width=40) (actual time=4791.158..4791.158 rows=1 loops=3)
               ->  Parallel Seq Scan on explain_demo  (cost=0.00..478604.90 rows=20833390 width=4) (actual time=0.054..2461.475 rows=16666668 loop
 Planning Time: 0.161 ms
 Execution Time: 4798.821 ms
(8 rows)

demo=*#
```

Beispiel: ORDER BY

```
test=# set work_mem = '4MB';
SET
test=# explain analyse select id,a,b,c from explain_demo where a between 4711 and 5120 and b between 4711 and 5120 order by c,b,a;
                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=358204.76..358414.04 rows=83713 width=16) (actual time=664.370..675.264 rows=84337 loops=1)
   Sort Key: c, b, a
   Sort Method: external merge  Disk: 2152kB
   ->  Index Scan using explain_abc on explain_demo  (cost=0.56..351359.90 rows=83713 width=16) (actual time=0.143..614.765 rows=84337 loops=1)
         Index Cond: ((a >= 4711) AND (a <= 5120) AND (b >= 4711) AND (b <= 5120))
 Planning Time: 0.304 ms
 Execution Time: 679.798 ms
(7 Zeilen)

test=# set work_mem = '8MB';
SET
test=# explain analyse select id,a,b,c from explain_demo where a between 4711 and 5120 and b between 4711 and 5120 order by c,b,a;
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=246445.99..246655.27 rows=83713 width=16) (actual time=670.067..677.483 rows=84337 loops=1)
   Sort Key: c, b, a
   Sort Method: quicksort  Memory: 6684kB
   ->  Bitmap Heap Scan on explain_demo  (cost=60596.21..239601.13 rows=83713 width=16) (actual time=164.560..629.150 rows=84337 loops=1)
         Recheck Cond: ((a >= 4711) AND (a <= 5120) AND (b >= 4711) AND (b <= 5120))
         Heap Blocks: exact=72469
         ->  Bitmap Index Scan on explain_abc  (cost=0.00..60575.29 rows=83713 width=0) (actual time=150.902..150.902 rows=84337 loops=1)
               Index Cond: ((a >= 4711) AND (a <= 5120) AND (b >= 4711) AND (b <= 5120))
 Planning Time: 0.289 ms
 Execution Time: 681.434 ms
(10 Zeilen)

test=#
```

wie langsamme Abfragen finden

- `log_min_duration_statement`
- auto_explain
- pg_stat_statements

Questions?