Datenbanktuning und Query-Optimierung am Beispiel von PostgreSQL – Best Practices

Anja Baasch, MIM13

Inhalt

Ausführungspläne erstellen

```
EXPLAIN [ ( option [,...] ) ] statement

    ANALYZE [ boolean ]

    VERBOSE [ boolean ]

    COSTS [ boolean ]

    BUFFERS [ boolean ]

    TIMING [ boolean ]

    FORMAT { TEXT | XML | JSON | YAML }
```

Ausführungspläne erstellen

AUTO_EXPLAIN

- LOAD 'auto_explain'
- shared_preload_libraries = ‘auto_explain'
- auto_explain.log_min_duration
- auto_explain.log_nested_statements

Ausführungspläne auswerten

```
dbi=# EXPLAIN SELECT * FROM gpsp_one.gpsp;

QUERY PLAN
-----------------------------------------------------------------
Seq Scan on gpsp  (cost=0.00..2163395.12 rows=59362712 width=188)


dbi=# EXPLAIN ANALYZE SELECT * FROM gpsp_one.gpsp;

QUERY PLAN
-----------------------------------------------------------------
Seq Scan on gpsp  (cost=0.00..2163395.12 rows=59362712 width=188)
(actual time=0.025..296632.647 rows=59327165 loops=1)
Total runtime: 307480.076 ms
```

Ausführungspläne auswerten

Pro Node

- costs (startup..finished)
- rows
- width
- actual time (startup..finished)

Ausführungspläne auswerten

Nodes

- seq scan, index scan (Zugriffs- und Filterprädikate), index only scan (seit pg 9.2),
- nested loops, hash join, merge join
- sort, groupaggregate, hashaggregate
- limit, windowagg

Ausführungspläne Beispiel

```
QUERY PLAN
-----------------------------------------------------------------------
Index Scan using gpsp_box_id_date_vbat_idx on gpsp
(cost=0.56..8.58 rows=1 width=188) (actual time=45.473..45.473
rows=0 loops=1)
   Index Cond: ((box_id = 302) AND (date = '2011-05-05'::date))
Total runtime: 80.408 ms

QUERY PLAN
-----------------------------------------------------------------------
Index Scan using gpsp_box_id_vbat_date_idx on gpsp
(cost=0.56..2739.80 rows=1 width=189) (actual time=68.681..68.681
rows=0 loops=1)
   Index Cond: ((box_id = 302) AND (date = '2011-05-05'::date))
Total runtime: 68.762 ms
```

Hints

- kein Hintssystem in pg
- join_collapse_limit, from_collapse_limit
- cursor_tuple_fraction
- enable_seqscan, enable_indexscan, enable_bitmapscan
- enable_nestloop, enable_hashjoin, enable_mergejoin
- enable_hashagg, enable_material

Statistiken

- ANALYZE

```
ANALYZE [ VERBOSE ] [ table [ ( column [, ...] ) ] ]
```

```
dbi=# ANALYZE VERBOSE gpsp_two.gpsp;
INFO:  analyzing "gpsp_two.gpsp"
INFO:  "gpsp": scanned 30000 of 1569768 pages, containing
1133833 live rows and 0 dead rows; 30000 rows in sample,
59327189 estimated total rows
ANALYZE
```

- AUTO_VACUUM

Monitoring und Engpassanalyze

- log_min_duration_statement
- log_temp_files
- log_checkpoints
- log_connections/log_disconnections
- log_lock_waits
- log_autovacuum_min_duration
- log_parser_stats, log_planner_stats, log_executor_stats, log_statement_stats

Ausführungspläne auswerten

- pgAdmin3 - visual explain
- http://explain.depesz.com/

Konfigurationsdatei optimieren

- pgtune

Logs analysieren

- pgFouine

Quellen

- Winand, Marcus „SQL Performance Explained“, Marcus Winand, 2012

- Smith, Gregory „PostgreSQL 9.0 High Performance“, Packt Publishing, 2010

- http://www.postgresql.org/docs/

- http://wiki.postgresql.org/wiki/OptimizerHintsDiscussion

- http://blog.2ndquadrant.com/hinting_at_postgresql/