INDEXY JSOU GRUNT

Pavel Stěhule

GoodData

Indexy

- bez indexu čteme vše a zahazujeme nechtěné
- s indexem čteme pouze to co nás zajímá
- POZOR - indexy vedou k random IO, navíc se čtou dvě databázové relace (index a heap)
- POZOR - z disku se čtou vždy kompletní datové stránky - nikoliv řádky
- bez indexu musíme relaci seřadit (pokud je to vyžadováno), což může vést k pomalému řazené s využitím IO (external sort)

Definice

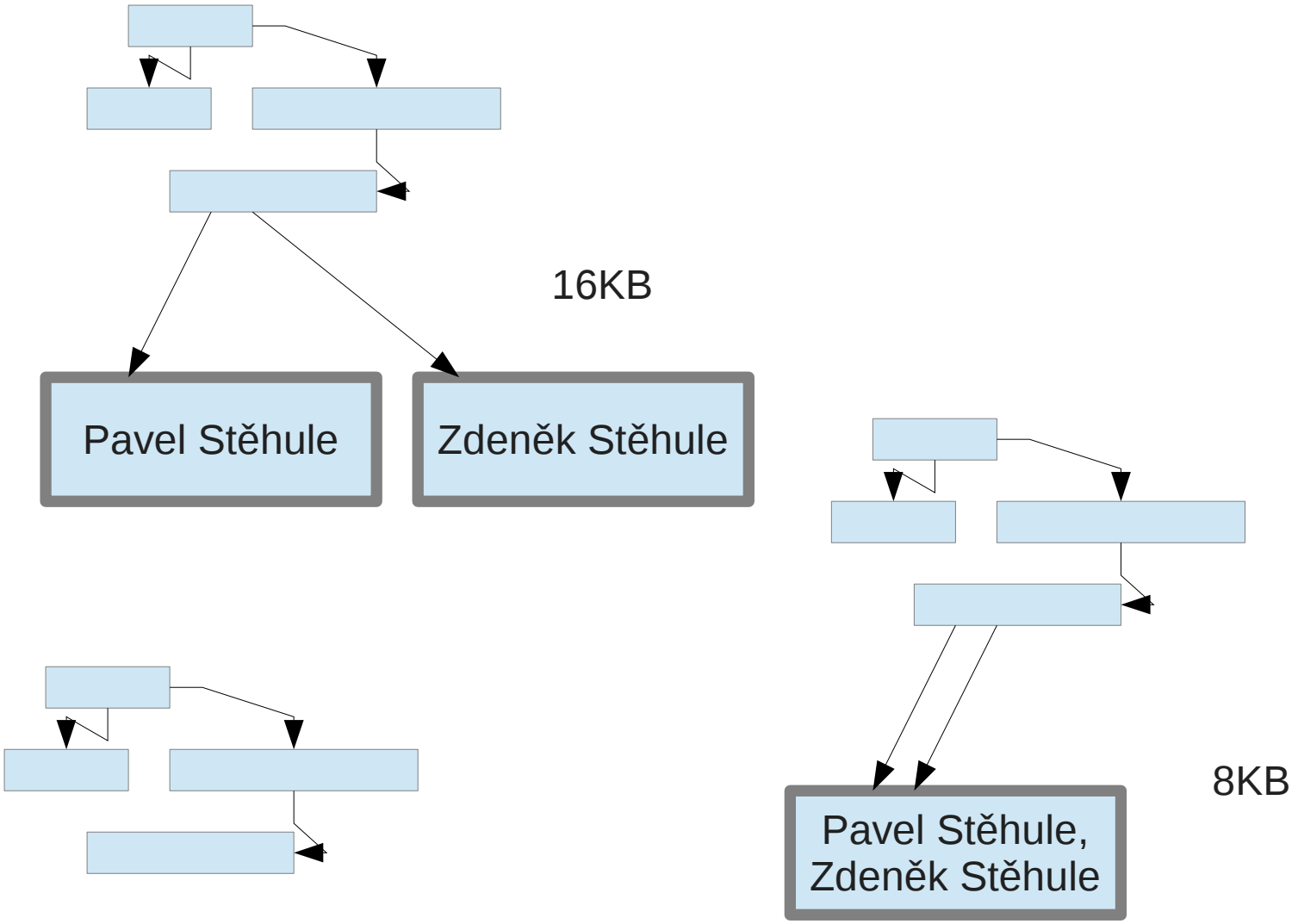
- databázová relace obsahující uspořádanou množinu dvojic (klíč, pozice)
- pozitivní (v 99%) vliv na čtení z db, negativní vliv na zápis
- ***v PostgreSQL index neobsahuje informace o viditelnosti záznamu (o jeho existenci) - DELETE, VACUUM nemodifikují indexy***

Optimizer
Filter

- RANDOM_PAGE_COST = 4
- SEQ_PAGE_COST = 1
- EFFECTIVE_CACHE_SIZE

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a = 10;
                                 QUERY PLAN
-----------------------------------------------------------------------------
 Index Only Scan using foo_a_idx on foo  (cost=0.00..19.24 rows=1 width=4)
   Index Cond: (a = 10)
(2 rows)
```

Vztah haldy a indexu



INDEX ONLY SCAN

Vztah haldy a indexu

- žádný
- halda je organizovaná podle indexu
  - jednorázově - PostgreSQL příkaz CLUSTER)
  - průběžne - cluster table (tabulka je organizovaná podle primárního klíče - MSSQL, InnoDB)

Optimizer
Order by

- WORK_MEM

```
postgres=# EXPLAIN SELECT * FROM foo ORDER BY a;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Index Only Scan using foo_a_idx on foo  (cost=0.00..303949.99 rows=9999985 width=4)
(1 row)

postgres=# SET work_mem = '500MB';
SET

postgres=# EXPLAIN SELECT * FROM foo ORDER BY a;
                            QUERY PLAN
------------------------------------------------------------------
 Sort  (cost=1306920.83..1331920.79 rows=9999985 width=4)
   Sort Key: a
   ->  Seq Scan on foo  (cost=0.00..144247.85 rows=9999985 width=4)
(3 rows)
```

Využití

- WHERE (index scan, index only scan)
- JOIN (merge join)
- ORDER BY
- GROUP BY
- DISTINCT

Konkurence

- Clustered table/index (částečně příkaz CLUSER)
- Coverage indexes (téměř index only scan)

Možné variace indexu

- Podmíněný index
- Funkcionální index
- Jednoduchý index
- Složený index
- Multidimenzionální index
- Fulltextový index

Podporované formáty

- Btree - Balanced Tree
- GiST - Generalized Search Tree
- GIN - Generalized Inverted Indexes
  - sdílené klíče, Btree

Podmíněný index

- omezuje velikost relace

```
CREATE TABLE message(
    id serial PRIMATY KEY,
    "to" text,
    inserted timestamp,
    subject text,
    read boolean);

CREATE INDEX ON message("to", id DESC)
  WHERE read;

SELECT *
   FROM message
  WHERE read
  ORDER BY id DESC;
```

Funkcionální index

- zabraňuje opakovanému volání funkce

```
CREATE TABLE message(
    id serial PRIMATY KEY,
    "to" text,
    inserted timestamp,
    subject text,
    read boolean);

CREATE INDEX ON message((lower("to")))

SELECT *
   FROM message
  WHERE lower("to") = lower('Pavel.stehule@gmail.com');
```

Složený index

- zvyšuje selektivitu indexu

```
CREATE TABLE message(
    id serial PRIMATY KEY,
    "to" text,
    inserted timestamp,
    subject text,
    read boolean);

CREATE INDEX ON message("to", inserted)

SELECT *
   FROM message
  WHERE "to" = 'Pavel.stehule@gmail.com'
    AND inserted BETWEEN current_date - 30 AND current_date;
```

Fulltextový index

- umožňuje hledat slova

```
CREATE TABLE message(
    id serial PRIMATY KEY,
    "to" text,
    inserted timestamp,
    subject text,
    read boolean);

CREATE INDEX ON message (to_tsvector('simple', subject));

postgres=# SELECT *
              FROM message
             WHERE to_tsvector('simple', subject) @@ to_tsquery('žluté');
              subject
-----------------------------------
 žluťoučký kůň se napil žluté vody
(1 row)
```

Speciální indexy

- Indexy typu GiST, GIN, SP-GiST
- V PostgreSQL ve spojení se zákaznickými typy

```
postgres=# explain analyze select * from obce where nazposty like '%Benešov%';
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on obce  (cost=0.00..335.05 rows=61 width=29) (actual time=0.023..2.978 rows=96 loops=1)
   Filter: (nazposty ~~ '%Benešov%'::text)
   Rows Removed by Filter: 16548
 Total runtime: 3.004 ms
(4 rows)

postgres=# explain analyze select * from obce where nazposty ilike '%Benešov%';
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on obce  (cost=0.00..335.05 rows=61 width=29) (actual time=0.091..21.541 rows=96 loops=1)
   Filter: (nazposty ~~* '%Benešov%'::text)
   Rows Removed by Filter: 16548
 Total runtime: 21.577 ms
(4 rows)
```

Extenze trigramy

```
postgres=# CREATE EXTENSION pg_trgm;
CREATE EXTENSION

postgres=# CREATE INDEX ON obce USING gin (nazposty gin_trgm_ops);
CREATE INDEX

postgres=# EXPLAIN ANALYZE SELECT * FROM obce WHERE nazposty ILIKE '%Benešov%';
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on obce  (cost=44.47..151.12 rows=61 width=29)
   Recheck Cond: (nazposty ~~* '%Benešov%'::text)
   ->  Bitmap Index Scan on obce_nazposty_idx  (cost=0.00..44.46 rows=61 width=0)
         Index Cond: (nazposty ~~* '%Benešov%'::text)
 Total runtime: 0.396 ms
```

KNN search

- dohledání nejbližšího bodu
- Operátor <-> pro výpočet vzdálenosti

```
postgres=# SELECT DISTINCT nazposty, nazposty <-> 'Benešov'
             FROM obce ORDER BY nazposty <-> 'Benešov'
             LIMITt 5;
      nazposty      | ?column?
--------------------+----------
 Dolní Benešov      | 0.428571
 Horní Benešov      | 0.428571
 Benešov u Semil    |      0.5
 Benešov u Prahy    |      0.5
 Benešov u Boskovic | 0.555556
(5 rows)

postgres=# explain SELECT nazobce, nazobce <-> 'Benešov' dif
                     FROM obce
                     ORDER BY nazobce <-> 'Benešov'
                     LIMIT 100;
                                    QUERY PLAN
-------------------------------------------------------------------------------------------
 Limit  (cost=0.00..8.61 rows=100 width=10)
   ->  Index Scan using obce_nazobce_idx on obce  (cost=0.00..1433.14 rows=16644 width=10)
         Order By: (nazobce <-> 'Benešov'::text)
(3 rows)
```