

Indexen

wim.bertels@ucll.be

Naamsvermelding-NietCommercieel-GelijkDelen 4.0
Unported Licentie

INDEX

- Doel: beïnvloeden van de verwerkingsstijd!



INDEX

- Doel: beïnvloeden van de verwerkingstijd!

OS:

- Rijen worden in bestanden opgeslagen
- Een bestand bestaat uit pagina's
- Als een rij opgehaald wordt :
 - de betreffende pagina wordt opgehaald
 - de betreffende rij wordt opgehaald

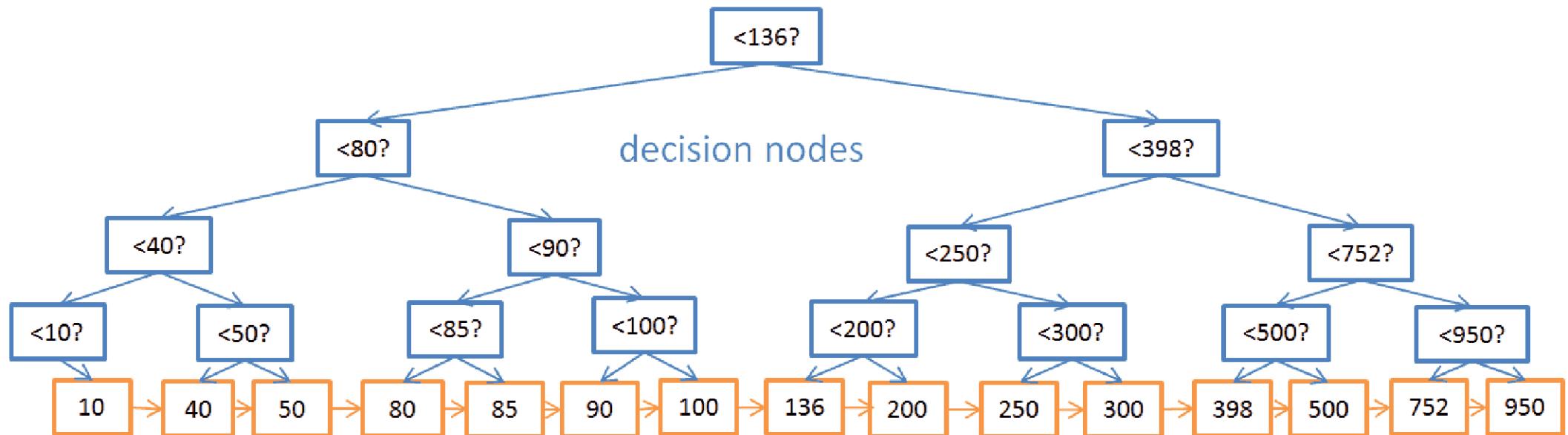
Werking van een INDEX

- 2 methodes voor het opzoeken:
 - Sequentiële zoekmethode: rij voor rij
 - Tijdrovend en inefficiënt
- Geïndexeerde zoekmethode: index (B-tree)
 - Boom
 - Knooppunten
 - Leafpage (bevat referentie naar pagina+rij)
 - 2 methodes:
 - Zoeken van rijen met een bepaalde waarde
 - Doorlopen van de hele tabel via een gesorteerde kolom (geclusterde index)

B-tree (default)

- Binaire zoek boom (node kan 2+leaves hebben)
- Best voor $>$ $<$ $=$ operatoren

B+ Tree / Database index

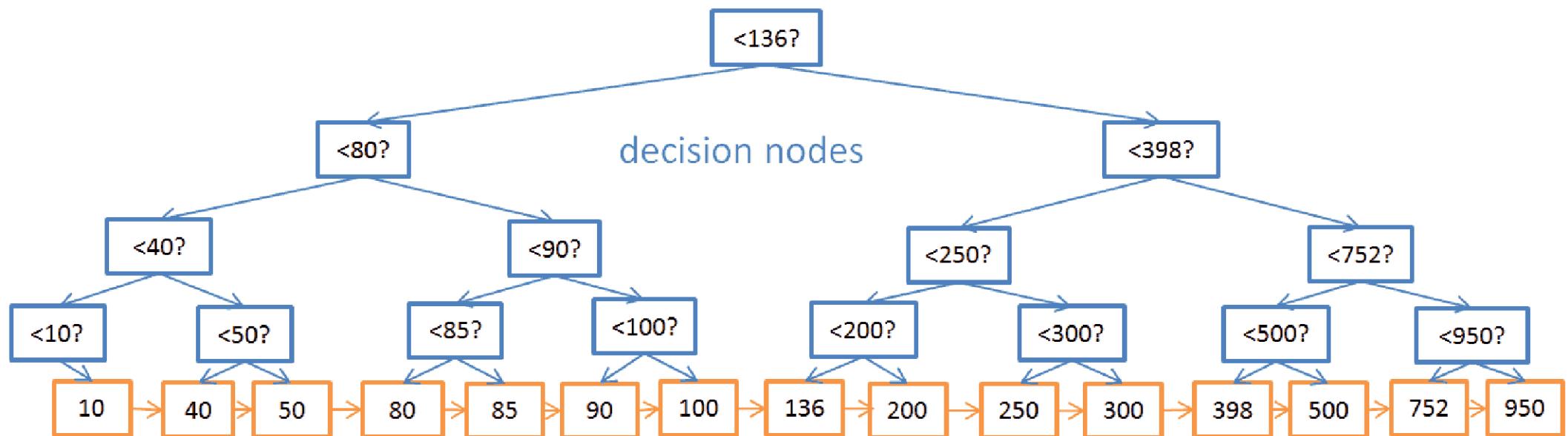


nodes with a pointer to a row in the associated table
they also have a link to their successor in the B+ Tree

B-tree (voorbeeld)

- Sequentiel: max 15 stappen
- Via boom: max 4 stappen, namelijk $\lceil \log_2 15 \rceil$

B+ Tree / Database index



Werking van een INDEX

- Opmerkingen:
 - Wanneer tabel word aangepast: word index aangepast
 - Index: ook op niet-unieke kolom
 - Op één tabel:
 - meerdere indexen
 - één geclusterde index
 - Samengestelde index
- Opgelet:
 - Index neemt opslagruimte in beslag
 - Als index vol is: reorganisatie van de index
- Meerdere indexvormen zijn mogelijk

Planner / Optimiser

- Conceptueel: bv denken in lussen of geneste lessen bij joins, de volgorde van verwerking van een SELECT instructie
- Concreet: intern kan dit helemaal anders verwerkt worden
- SQL is een declaratieve (programmeer)taal, geen imperatieve (programmeer)taal.
 - Idee: Je zegt wat er (logisch) moet gebeuren, niet (expliciet) hoe het moet gebeuren.
 - 1 van de sterke punten van SQL
- Redeneer conceptueel en groei

Optimiser

- Zoekt de beste strategie
 - Verwachte verwerkingstijd
 - Aantal rijen
 - Indexen
 - Interne statistieken
 - ..

CREATE INDEX

- Geen ANSI of ISO specificatie
- Vendor specificatie :
<https://www.postgresql.org/docs/current/sql-createindex.html>

CREATE INDEX

- Postgresql:

```
CREATE INDEX spelers_postcode_idx  
    ON spelers (postcode asc);  
CREATE UNIQUE INDEX spelers_naam_vl_idx  
    ON spelers (naam, voorletters);  
-- UNIQUE !  
CREATE INDEX spelers_naam_vl_partial_idx  
    ON spelers (naam, voorletters)  
    WHERE spelersnr < 100;  
CLUSTER spelers USING speler_naam_vl_idx;
```

DELETE/UPDATE

Wat is het effect van een geclusterde index?

REINDEX

- Vendor specificatie:

<https://www.postgresql.org/docs/current/sql-reindex.html>

- Postgresql:

- REINDEX INDEX een_index;
- REINDEX TABLE een_tabel;
- REINDEX DATABASE een_database;

INDEX management

- CREATE
- ALTER
 - `ALTER INDEX groot_idx
SET TABLESPACE ergens_anders;`
- DROP
- Meeste SQL-producten :
 - automatische creatie van index op primaire en secudaire sleutels bij het maken van de tabel
 - naam wordt afgeleid uit de naam van de tabel en de betreffende kolommen

Wanneer INDEXeren?

- Index:
 - Voordeel: index versnelt verwerking
 - Nadeel:
 - index neemt opslagruimte
 - elke mutatie vraagt aanpassing van index
=> verwerking vertraagt

Welke kolommen?

- Richtlijnen voor keuze van kolommen :
 - Unieke index op kandidaatsleutels
 - Index op refererende sleutels
 - Index op kolommen waarop (veel) geselecteerd wordt
 - Grootte van de tabel
 - Kardinaliteit (verschillende waarden) van de tabel
 - Distributie (verdeling) van de waarden
 - Index op een combinatie van kolommen
 - Index op kolommen waarop gesorteerd wordt
 - ..

Speciale indexvormen

- Multi-tabelindex :
= index op kolommen in meerdere tabellen
- Virtuele-kolomindex :
= index op een expressie
- Selectieve index
= index op een selectie van de rijen
- Hash-index
= index op basis van het adres van de pagina
- Bitmapindex
= interessant als er veel dubbele waardes zijn

Nieuwere indexvormen: GiST en SP-GIST

- GiST (Generalized Search Tree)
 - gebalanceerd
 - template voor verschillende index schema's
 - Voor "clusters" volgens een afstandsmaat
 - Bv vergelijken van intervallen, GIS, bevat, dichtste buren, tekst
- SP-GiST (space-partitioned GiST)
 - Hoeft niet gebalanceerd te zijn
 - Geen overlap tussen de clusters (GiST)

Nieuwere indexvormen: Gin en Brin

- GIN (Generalized Inverted Index)
 - Interessant bij veel dubbele waarden
 - Er worden meerdere opzoekwaarden tegelijk aangemaakt (handig voor rij, tekst, json,...)
- BRIN : voor grote geclusterde tabellen
 - Klein, minder performant tenzij:
 - min/max waarde per blok

Conclusie

Een index op elke tabel en iedere kolom?

Samengevat

- B-tree:
 - goeie standaard, < \leq = \geq >
- Hash:
 - alternatief voor 1 rij, =
- GIN:
 - efficiënt bij dubbels, meerdere opzoekwaarden per veld, <@ @> = &&
- GiST:
 - veel mogelijkheden, minder performant, << &< .. <<| &<| <@ ~= &&
- Sp-GiST:
 - niet overlappende GiST, << >> ~= <@ <<| |>>
- BRIN:
 - grote geclusterde data, < \leq = \geq >

INDEX catalog_table

- Postgresql: pg_index
- <http://www.postgresql.org/docs/current/static/catalog-pg-index.html>
- <http://www.postgresql.org/docs/current/static/internals.html>
- Referentie: Index Internals, Heikki Linnakangas (Pivotal)
- <https://www.pexels.com/photo/blurred-book-book-pages-literature-46274/>

Algemene Richtlijnen Optimalisaties

wim.bertels@ucll.be

Naamsvermelding-NietCommercieel-GelijkDelen 4.0
Unported Licentie

1. Inleiding

- Optimiser – Beste uitvoeringsplan?
 - Indexen – Query snelheid verhogen?
 - Inzicht – Wat gebeurt er intern?
 - Maar soms niet optimaal
- => Herformulering om tot een efficiënte verwerkingsstrategie te komen !

2. Vermijd de OR-operator

- OR : index wordt meestal niet gebruikt
- Alternatief (indien mogelijk) :
 - Vervangen door een conditie met IN
 - Vervangen door 2 selects met UNION
- Vb.

...
where spelersnr = 15
or spelersnr = 29
or spelersnr = 55

=> where spelersnr in (15, 29, 55)

3. Onnodig gebruik van UNION

- UNION : dezelfde tabel meerdere malen doorlopen
- Alternatief (indien mogelijk) :
 - Herformuleren waarbij alle voorwaarden in één select instructie geplaatst worden

4. Vermijd de NOT-operator

- NOT : index wordt niet gebruikt
- Alternatief (indien mogelijk) :
 - Vervang NOT door vergelijkingsoperatoren
- Vb.

...

where not (jaartoe > 1980)

4. Vermijd de NOT-operator

- Oplossing:

...

where not (jaartoe > 1980)

=> where jaartoe <= 1980

5. Isoleer kolommen in condities

- Kolom in een berekening of in een scalaire functie : index wordt niet gebruikt
- Alternatief (indien mogelijk) :
 - Isoleer de kolom
- Vb.

...

where jaartoe + 10 = 1990

=> where jaartoe = 1980

6. Gebruik de BETWEEN-operator

- AND : gebruikt de index meestal niet
- Vb.

...

where jaartoe >= 1985

and jaartoe <= 1990

6. Gebruik de BETWEEN-operator

- AND : gebruikt de index meestal niet
- Oplossing :
 - Gebruik van BETWEEN
- Vb.

...

```
where    jaartoe >= 1985  
and      jaartoe <= 1990
```

=> where jaartoe between 1985 and 1990

7. Bepaalde vormen van LIKE-operator

- LIKE : index wordt niet gebruikt als patroon begint met % of _
- Alternatief :
- Geen, tenzij..
- Vb.

...

wherenaam like ‘%sen’

=> ???

8. Redundante condities bij joins

- Redundante condities : om SQL te verplichten om een bepaald pad te kiezen
- Vb.

...

```
where boetes.spelersnr = spelers.spelersnr  
and   boetes.spelersnr = 44
```

```
=>where boetes.spelersnr = spelers.spelersnr  
      and   boetes.spelersnr = 44  
      and   spelers.spelersnr = 44
```

9. Vermijd de HAVING-component

- Condities in HAVING : index wordt niet gebruikt
- Alternatief (indien mogelijk)
 - Zoveel mogelijk condities in WHERE
- Vb. --hoort dit thuis in de having?

...
group by spelersnr
having spelersnr >= 40

=> where spelersnr >= 40
 group by spelersnr

10. SELECT-component : compact

- SELECT-component zo compact mogelijk
 - Onnodiige kolommen weglaten uit SELECT
 - Bij gecorreleerde subquery met exists : één expressie bestaande uit één constante
- Vb.

```
select spelersnr, naam
from spelers
where exists (select '1'
               from boetes
               where boetes.spelersnr =
                     spelers.spelersnr)
```

11. Vermijd DISTINCT

- DISTINCT : verwerkingstijd verlengd
- Alternatief (indien mogelijk)
 - Vermijden als het overbodig is
- Vb.

```
select distinct wedstrijdnr, naam  
from wedstrijden, spelers  
where wedstrijden.spelersnr =  
      spelers.spelersnr
```

=> select wedstrijdnr, naam

12. ALL-optie bij set operatoren

- Zonder ALL : verwerkingstijd verlengd
- Data moeten gesorteerd worden om dubbels eruit te halen
- Vb.

```
Select    naam, voorletters  
from     spelers  
where    spelersnr = 10  
union all  
select    naam, voorletters  
from     spelers  
where    spelersnr = 18
```

13. Kies outer-joins boven UNION

- UNION : verwerkingstijd verlengd
- Alternatief (indien mogelijk)
 - Outer-join is beter

- Vb.

```
select spelers.spelersnr, naam, bedrag
from spelers, boetes
where spelers.spelersnr = boetes.spelersnr
union
select spelersnr, naam, null
from spelers
where spelersnr not in (select spelersnr
                        from boetes)
order by 1
```

- Vb. Oplossing

```
select spelers.spelersnr, naam, bedrag
from spelers, boetes
where spelers.spelersnr = boetes.spelersnr
union
select spelersnr, naam, null
from spelers
where spelersnr not in (select spelersnr
                           from boetes)
order by 1
```

=> select spelersnr, naam, bedrag
 from spelers left outer join boetes
 using (spelersnr)
 order by 1

14. Vermijd datatype-conversies

- Converteren van datatypes : verwerkingstijd verlengd
- Alternatief (indien mogelijk) :
 - Datatype-conversie vermijden
- Vb.

```
select      *
from        spelers
where       spelersnr = '15'
```

=> where spelersnr = 15

15. Volgorde tabellen

- Volgorde van tabellen kan belangrijk zijn
- Afhankelijk van de juistheid van de interne statistieken
- Vb.

```
select spelers.spelersnr, naam, teamnr  
from spelers, teams  
where spelers.spelersnr = teams.spelersnr  
  
<=> select spelers.spelersnr, naam, teamnr  
from teams, spelers  
where spelers.spelersnr =  
      teams.spelersnr
```

16. Vermijd ANY- en ALL-operatoren

- ANY en ALL : index wordt niet gebruikt
- Alternatief (indien mogelijk)
 - ?
- Vb.

```
Select      spelersnr, naam, geb_datum  
from       spelers  
where      geb_datum <= all (select geb_datum  
                           from spelers)
```

```
=> select spelersnr, naam, geb_datum  
      from spelers  
     where geb_datum = (select min(geb_datum)  
                          from spelers)
```

Vervang door min of max

Hoe nakijken

- Explain
- Explain analyze

EXPLAIN

inleidende voorbeelden query plan

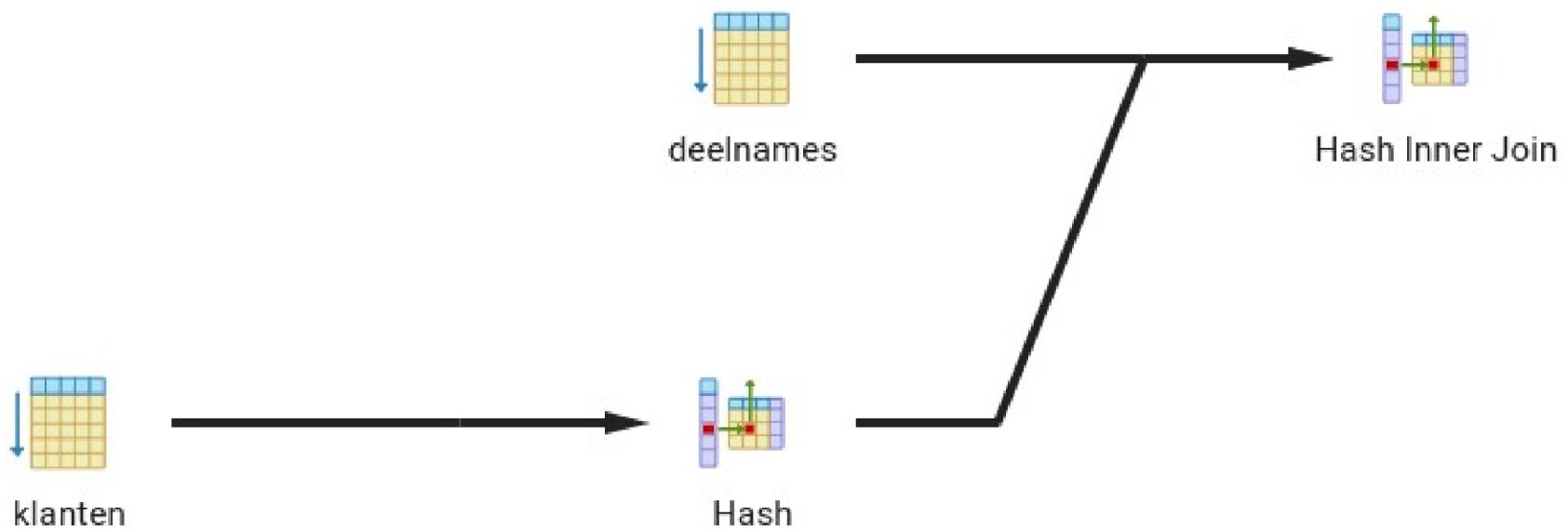
wim.bertels@ucll.be

Naamsvermelding-NietCommercieel-GelijkDelen 4.0
Unported Licentie

```
3 select *
4 from klanten natural inner join deelnames;
```

Data Output Messages Explain X Notifications

Graphical Analysis Statistics



```

3 select *
4 from klanten natural inner join deelnames;

```

Data Output Messages Explain Notifications

Graphical Analysis Statistics

#	Node
1.	→ Hash Inner Join Hash Cond: (deelnames.klantnr = klanten.klantnr)
2.	→ Seq Scan on deelnames as deelnames
3.	→ Hash
4.	→ Seq Scan on klanten as klanten

Graphical Analysis Statistics

Statistics per Node Type

Node type	Count
Hash	1
Hash Inner Join	1
Seq Scan	2

Statistics per Relation

Relation name	Scan count
Node type	Count
deelnames	1
klanten	1

1 tabel

```
CREATE TABLE een_miljoen  
  (teller      integer,  
   random_tekst  text);
```

```
INSERT INTO een_miljoen  
  SELECT i, md5(random()::text)  
    FROM generate_series(1, 1000000) AS i;
```

```
EXPLAIN
SELECT *
FROM een_miljoen;
```

QUERY PLAN

```
Seq Scan on een_miljoen
(cost=0.00..18918.18
rows=1058418 width=36)
```

```
ANALYZE een_miljoen;
```

```
EXPLAIN
```

```
SELECT *
```

```
FROM een_miljoen;
```

QUERY PLAN

```
Seq Scan on een_miljoen  
(cost=0.00..18334.00  
rows=1000000 width=37)
```

EXPLAIN ANALYZE

```
SELECT *
```

```
FROM een_miljoen;
```

QUERY PLAN

```
Seq Scan on een_miljoen
(cost=0.00..18334.00
rows=1000000 width=37)
(actual time=0.008..78.234
rows=1000000 loops=1)
```

```
Planning Time: 0.012 ms
```

```
Execution Time: 106.864 ms
```

```
EXPLAIN
```

```
SELECT *
```

```
FROM een_miljoen
```

```
WHERE teller > 500;
```

QUERY PLAN

```
Seq Scan on een_miljoen
(cost=0.00..20834.00
rows=999507 width=37)
```

```
Filter: (teller > 500)
```

```
CREATE INDEX ON een_miljoen(teller);
EXPLAIN
SELECT *
FROM   een_miljoen
WHERE  teller > 500;
```

QUERY PLAN

Seq Scan on een_miljoen (cost=0.00..20834.00 rows=999491 width=37)

Filter: (teller > 500)

```
EXPLAIN  
SELECT *  
FROM een_miljoen  
WHERE teller < 500;
```

QUERY PLAN

Index Scan using een_miljoen_teller_idx on een_miljoen
(cost=0.42..25.32 rows=508 width=37)

Index Cond: (teller < 500)

???

filter (> 500) Seq Scan: geen index

filter (< 500) Index Scan

- Misschien..

```
SET enable_seqscan TO off;  
EXPLAIN ANALYZE  
SELECT *  
FROM een_miljoen  
WHERE teller > 500;
```

QUERY PLAN

Index Scan using
een_miljoen_teller_idx on
een_miljoen
(cost=0.42..36800.52
rows=999491 width=37)
(actual time=0.023..255.981
rows=999500 loops=1)

Index Cond: (teller >
500)

Planning Time: 0.051 ms

Execution Time: 296.085 ms

QUERY PLAN

Seq Scan on een_miljoen
(cost=0.00..20834.00
rows=999491 width=37)

Filter: (teller > 500)

SET enable_seqscan TO on;

meerdere tabellen

```
CREATE TABLE half_miljoen
  (teller          integer ,
   random_munt    boolean);

INSERT INTO half_miljoen
  SELECT i, i%2=1
    FROM generate_series(1, 500000) AS i;
ANALYZE half_miljoen;
```

```
CREATE TABLE een_miljoen
  (teller      integer ,
   random_tekst text);

INSERT INTO een_miljoen
  SELECT i, md5(random()::text)
    FROM generate_series(1, 1000000) AS i;
```

```
EXPLAIN ANALYZE
SELECT *
FROM een_miljoen JOIN half_miljoen
ON (een_miljoen.teller=half_miljoen.teller);
```

QUERY PLAN

Hash Join (cost=15417.00..60081.00 rows=500000 width=42)
(actual time=102.561..679.936 rows=500000 loops=1)

Hash Cond: (een_miljoen.teller = half_miljoen.teller)

-> Seq Scan on een_miljoen
(cost=0.00..18334.00 rows=1000000 width=37)
(actual time=0.006..76.141 rows=1000000 loops=1)

-> Hash (cost=7213.00..7213.00 rows=500000 width=5)
(actual time=102.465..102.466 rows=500000 loops=1)

Buckets: 262144 Batches: 4 Memory Usage: 6562kB

-> Seq Scan on half_miljoen
(cost=0.00..7213.00 rows=500000 width=5)
(actual time=0.005..33.529 rows=500000 loops=1)

Planning Time: 0.193 ms

Execution Time: 693.629 ms

```
EXPLAIN ANALYZE
SELECT *
FROM een_miljoen JOIN half_miljoen
ON (een_miljoen.teller=half_miljoen.teller);
```

```
CREATE INDEX ON half_miljoen(teller);
EXPLAIN ANALYZE
SELECT *
FROM een_miljoen JOIN half_miljoen
ON (een_miljoen.teller=half_miljoen.teller);
```

QUERY PLAN

Merge Join (cost=1.31..40111.03 rows=500000 width=42)
(actual time=0.014..298.152 rows=500000 loops=1)

Merge Cond: (een_miljoen.teller = half_miljoen.teller)

-> Index Scan using een_miljoen_teller_idx on een_miljoen
(cost=0.42..34317.43 rows=1000000 width=37)
(actual time=0.006..77.998 rows=500001 loops=1)

-> Index Scan using half_miljoen_teller_idx on half_miljoen
(cost=0.42..15212.42 rows=500000 width=5)
(actual time=0.004..83.697 rows=500000 loops=1)

Planning Time: 0.264 ms

Execution Time: 311.930 ms

Hash Join (cost=15417.00..60081.00

Aggregaties

```
EXPLAIN  
SELECT count(*)  
FROM een_miljoen;
```

QUERY PLAN

```
Finalize Aggregate  (cost=14542.55..14542.56 rows=1 width=8)  
-> Gather  (cost=14542.33..14542.54 rows=2 width=8)  
    Workers Planned: 2  
    -> Partial Aggregate  (cost=13542.33..13542.34 rows=1 width=8)  
        -> Parallel Seq Scan on een_miljoen  (cost=0.00..12500.67 rows=416667 width=0)
```

```
EXPLAIN ANALYZE
```

```
SELECT max(random_tekst)  
FROM een_miljoen;
```

QUERY PLAN

```
Finalize Aggregate  (cost=14542.55..14542.56 rows=1 width=32)  
                    (actual time=75.277..79.133 rows=1 loops=1)
```

```
-> Gather     (cost=14542.33..14542.54 rows=2 width=32)  
                  (actual time=75.198..79.124 rows=3 loops=1)
```

Workers Planned: 2

Workers Launched: 2

```
-> Partial Aggregate  (cost=13542.33..13542.34 rows=1 width=32)  
                      (actual time=72.948..72.948 rows=1 loops=3)
```

```
    -> Parallel Seq Scan on een_miljoen  
        (cost=0.00..12500.67 rows=416667 width=33)  
        (actual time=0.008..24.153 rows=333333 loops=3)
```

Planning Time: 0.073 ms

Execution Time: 79.155 ms

```
CREATE INDEX ON een_miljoen(random_tekst);
EXPLAIN ANALYZE
SELECT max(random_tekst)
FROM een_miljoen;
```

QUERY PLAN

Result (cost=0.47..0.48 rows=1 width=32)
(actual time=0.035..0.035 rows=1 loops=1)

InitPlan 1 (returns \$0)

-> Limit (cost=0.42..0.47 rows=1 width=33)
(actual time=0.031..0.032 rows=1 loops=1)

-> Index Only Scan Backward using een_miljoen_random_tekst_idx on een_miljoen
(cost=0.42..46340.43 rows=1000000 width=33)
(actual time=0.030..0.031 rows=1 loops=1)

Index Cond: (random_tekst IS NOT NULL)

Heap Fetches: 0

Planning Time: 0.156 ms

Execution Time: 0.049 ms

Groeperingen

```
DROP INDEX een_miljoen_random_tekst_idx;  
EXPLAIN ANALYZE  
SELECT random_tekst, count(*)  
FROM een_miljoen  
GROUP BY random_tekst;
```

QUERY PLAN

```
HashAggregate  (cost=105834.00..131459.00 rows=1000000 width=41)  
              (actual time=473.085..1013.928 rows=1000000 loops=1)  
  
Group Key: random_tekst  
  
Planned Partitions: 16  Batches: 81  Memory Usage: 8337kB  Disk Usage: 63536kB  
  
-> Seq Scan on een_miljoen  
              (cost=0.00..18334.00 rows=1000000 width=33)  
              (actual time=0.007..63.300 rows=1000000 loops=1)
```

```
CREATE INDEX ON een_miljoen(random_tekst);
EXPLAIN ANALYZE
SELECT random_tekst, count(*)
FROM een_miljoen GROUP BY random_tekst;
```

QUERY PLAN

GroupAggregate (cost=0.42..58840.43 rows=1000000 width=41)
(actual time=0.030..440.101 rows=1000000 loops=1)

Group Key: random_tekst

-> Index Only Scan using een_miljoen_random_tekst_idx on een_miljoen
(cost=0.42..43840.43 rows=1000000 width=33)
(actual time=0.024..133.736 rows=1000000 loops=1)

Conclusie

- EXPLAIN (afhankelijk van de statistieken)
Oplossing: ANALYZE voordien
- EXPLAIN ANALYZE
Actuele Uitvoertijd, maar voert dus ook effectief uit!
- EXPLAIN
Werkt ook voor andere DML (insert, ..)
Kijk in eerste instantie naar de totale kost
Kijk eventueel naar de scan methoden

Referenties

- Understanding Explain, guillaume.lelarge@dalibo.com

Dedectie en referentiepunten



Wim.bertels@ucll.be

Trage queries?

- casus: databank beheerder
- middel: pg_stat_statements
- Bewerk postgresql.conf om deze module beschikbaar te maken
 - produkt specifiek

```
# postgresql.conf  
shared_preload_libraries = 'pg_stat_statements'
```

pg_stat_statements.max = 10000

pg_stat_statements.track = all

Create Extension

- In elke databank waar je dit wil gebruiken:
 - CREATE EXTENSION pg_stat_statements;
- `SELECT *
FROM pg_stat_statements;`

Gebruik?

userid	37919	shared_blk_hit	0
dbid	12413	shared_blk_read	0
queryid	2401821812	shared_blk_dirtied	0
query	SELECT version()	shared_blk_written	0
calls	1	local_blk_hit	0
total_time	0.021	local_blk_read	0
min_time	0.021	local_blk_dirtied	0
max_time	0.021	local_blk_written	0
mean_time	0.021	temp_blk_read	0
stddev_time	0	temp_blk_written	0
rows	1	blk_read_time	0
		blk_write_time	0

Referentiepunten

- Middel: pgbench
- Voorbeeld localhost:
- \$ pgbench -i probeer
- \$ pgbench -T 120 probeer
- ..heeft meerdere opties

Resultaten

starting vacuum...end.

transaction type: TPC-B (sort of)

scaling factor: 1

query mode: simple

number of clients: 1

number of threads: 1

duration: 120 s

number of transactions actually processed: 42594

latency average: 2.817 ms

tps = 354.944850 (including connections establishing)

tps = 354.961873 (excluding connections establishing)

Bruikbaarheid

- Simulaties vs Realiteit
- Opvolging!
- Een redelijk vertrekpunt
- Notas:
 - De *initialization scale factor* (-s) zou minstens zo groot moeten zijn als het hoogste aantal klanten (*clients*) dat je wil testen (-c)
 - Autovacuum en belasting(load)

Referenties

- <https://www.postgresql.org/docs/current/static/pgstatstatements.html>
- <https://www.postgresql.org/docs/current/static/pgbench.html>

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License