Uso degli indici nell'ottimizzazione delle query SQL

Gianni Ciolli,
2ndQuadrant Italia

7° PGDay italiano, Prato, 25 ottobre 2013

Outline

Cosa è un indice?

- Una struttura *persistente*
  - i.e. su disco
- Usata nelle query SQL per ottimizzare l'accesso
  - a una certa tabella
  - per certi tipi di domande
- Dipende:
  - dal tipo di dato
  - dall'operatore utilizzato

Esempio 0
Qui potrebbe convenire un indice

- La query SQL:

```
SELECT *
FROM t
WHERE n > 20;
```

- La tabella: `t`
- La domanda: “le righe dove *n* è maggiore di 20"
  - Il tipo di dato:
    `integer`
  - L’operatore:
    `> (integer)`
    ossia “più grande di un altro numero intero”

Tabelle, Blocchi, Selettività

Semplificato ma non troppo

- Tabelle
  - Ogni riga è memorizzata in un blocco di 8K
  - Una tabella si compone di (tanti) blocchi
- Selettività
  - L'inverso di *densità*
  - Bassa selettività = alta densità
    - Sono selezionate molte righe
  - High selectivity = low density
    - Sono selezionate poche righe

Con e senza
Differenze

- Senza un indice
  - Tutte le righe nella tabella sono caricate in memoria
  - La condizione viene calcolata su tutte le righe
  - Vengo restituite solo le righe che soddisfano la condizione
- Con un indice
  - L'indice viene caricato in memoria
  - L'indice fornisce un elenco di righe
  - Le righe dell'elenco sono caricate in memoria
  - La condizione viene calcolata solo su queste righe
    - Ci possono essere dei "falsi positivi" (e.g. indici "lossy", visibilità)
  - Vengono restituite solo le righe che soddisfano la condizione

Con e senza
Differenze, in tabella

	Senza	Con
Caricato indice	No	Sì
Caricata tabella	Tutta	Parte = (*)
Condizione calcolata su	Tutte le righe	(*)

Costo di un indice
Ahimè

- Gli indici hanno un *costo* aggiuntivo
- Si applica a ogni scrittura
- Come un'assicurazione:
  - Un piccolo costo regolare
  - Un grosso beneficio
- Forse

Quali indici?
Tipi di indice supportati da PostgreSQL

- B-tree
  - Tipi di dato totalmente ordinati
- Hash
  - Per ogni tipo di dato
- GIN
  - Per valori compositi
- GiST
  - Algoritmi basati su alberi
  - L'albero è bilanciato
- SP-GiST
  - Alberi di ricerca partizionati
  - L'albero non deve necessariamente essere bilanciato

Operatori scalari
Numeri, testo, giorni, orari

- Ordinamento

  `<` `<=` `=` `>=` `>`

- Esempi:
  - `val >= 15`
  - `orario < timestamp '2013-10-25 14:20'`
  - `database = 'PostgreSQL'`

Operatori su array

`A` e `B` array dello stesso tipo

- inclusione (tra insiemi)
  i.e. ogni elemento di `A` sta anche in `B`

```
A <@ B
B @> A
```

- incidenza
  i.e. `A` e `B` hanno degli elementi in comune

```
A && B
```

- uguaglianza (tra liste)

```
A = B
```

Operatori `hstore`

Per ogni `h` hstore, `k` testo ed `a` array di testo

- `h` contiene la chiave `k`?

  ```
  h ? k
  ```

- `H` contiene tutte le chiavi in `a`?

  ```
  h ?& a
  ```

- `H` contiene almeno una delle chiavi in `a`?

  ```
  h ?| a
  ```

Operatori spaziali
Operatori booleani tra poligoni

- sinistra, destra, sotto, sopra “di tipo forte”

  `<<` `>>` `<<|` `|>>`

- sinistra, destra, sotto, sopra “di tipo debole”

  `&<` `&>` `&<|` `|*>`

- uguaglianza estensionale e di area

  `~=` `=`

- sovrapposizione, sottoinsieme, soprainsieme

  `&&` `<@` `@>`

Esempio 1

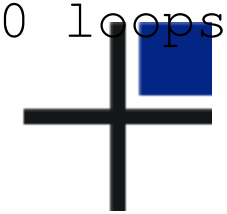
- Dati
  - una collezione di (parecchie) forme 2D
  - un punto
- Trovare tutte le forme che contengano un punto dato

Soluzione

```
WITH p AS (
   SELECT *
   FROM point
   WHERE id = :id
)
SELECT *
FROM shape s, p
WHERE ST_Intersects(p.geom,s.geom);
```

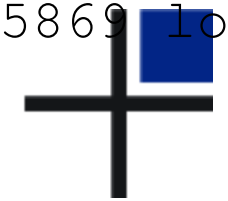
Profilazione, senza GiST

```
Nested Loop
    (actual time=0.854..1072.591 rows=2788 loops=1)
  Join Filter: ((p.geom && s.geom) AND _st_intersects(p.
  Rows Removed by Join Filter: 997212
  CTE p
    ->  Index Scan using id2 on point
          (actual time=0.029..0.032 rows=1 loops=1)
        Index Cond: (id = 12345)
  ->  CTE Scan on p
        (actual time=0.032..0.036 rows=1 loops=1)
  ->  Seq Scan on shape s
        (actual time=0.009..276.710 rows=1000000 loops
Total runtime: 1073.214 ms
```



Profilazione, con GiST

```
Nested Loop
    (actual time=11.786..53.273 rows=2788 loops=1)
  CTE p
    ->  Index Scan using id2 on point
          Index Cond: (id = 12345)
  ->  CTE Scan on p
  ->  Bitmap Heap Scan on shape s
        (actual time=11.770..51.844 rows=2788 loops=1)
        Recheck Cond: (p.geom && geom)
        Filter: _st_intersects(p.geom, geom)
        Rows Removed by Filter: 3081
        ->  Bitmap Index Scan on i1
              (actual time=10.667..10.667 rows=5869 lo
              Index Cond: (p.geom && geom)
Total runtime: 53.823 ms
```



Esempio 2

- Dati
  - una collezione di (parecchi) punti
  - una forma 2D
- Trovare tutti i punti contenuti nella forma data

Soluzione

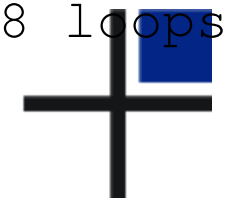
```
WITH s AS (
   SELECT *
   FROM shape
   WHERE id = :id
)
SELECT *
FROM point p, s
WHERE ST_Intersects(p.geom,s.geom);
```

Profiling, without GiST

```
Nested Loop
    (actual time=1.384..1072.624 rows=90 loops=1)
  Join Filter: ((p.geom && s.geom) AND _st_intersects(p.
  Rows Removed by Join Filter: 999910
  CTE s
    ->  Index Scan using id1 on shape
          Index Cond: (id = 12345)
  ->  CTE Scan on s
  ->  Seq Scan on point p
        (actual time=0.007..233.351 rows=1000000 loops
Total runtime: 1072.684 ms
```

Profilatura, con GiST

```
Nested Loop
    (actual time=0.915..6.790 rows=90 loops=1)
  CTE s
    ->  Index Scan using id1 on shape
          Index Cond: (id = 12345)
  ->  CTE Scan on s
  ->  Bitmap Heap Scan on point p
        (actual time=0.898..6.705 rows=90 loops=1)
      Recheck Cond: (geom && s.geom)
      Filter: _st_intersects(geom, s.geom)
      Rows Removed by Filter: 788
      ->  Bitmap Index Scan on i2
            (actual time=0.667..0.667 rows=878 loops
          Index Cond: (geom && s.geom)
Total runtime: 6.858 ms
```



Esempio 3

- Dati
  - una collezione di (parecchi) punti
  - una forma 2D
- Trovare tutti i punti contenuti nella forma data
- Come l'esempio 2, ma la forma è "sfortunata"
  - l'area è solo il 5% dell'area della *bounding box*

Profilatura, con GiST

```
Nested Loop
    (actual time=0.253..16.209 rows=57 loops=1)
  ->  Seq Scan on thin t
  ->  Index Scan using i2 on point p
        (actual time=0.246..16.170 rows=57 loops=1)
      Index Cond: (geom && t.geom)
      Filter: _st_intersects(geom, t.geom)
      Rows Removed by Filter: 5384
Total runtime: 16.261 ms
```

Profilatura, con GiST, usando la *bounding box*

```
Nested Loop
    (actual time=0.075..7.582 rows=5441 loops=1)
  ->  Seq Scan on native_thin t
  ->  Index Scan using i4 on native_point p
        (actual time=0.055..5.411 rows=5441 loops=1)
        Index Cond: (p <@ box(t.p))
Total runtime: 8.534 ms
```

Profilatura, con SP-GiST, usando la *bounding box*

```
Nested Loop
    (actual time=0.031..7.990 rows=5441 loops=1)
  ->  Seq Scan on native_thin t
  ->  Index Scan using i5 on native_point p
        (actual time=0.024..5.814 rows=5441 loops=1)
      Index Cond: (p <@ box(t.p))
Total runtime: 8.914 ms
```

E adesso...

Domande?

E infine...

Grazie!

Licenza d'uso