Indexes of PostgreSQL

Emre Hasegeli
CREATE INDEX

CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ name ]
ON table_name [ USING method ]
( { column_name | ( expression ) } )
[ COLLATE collation ] [ opclass ]
[ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ... ]
[ WITH ( storage_parameter = value [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ WHERE predicate ]
DROP INDEX

DROP INDEX [ CONCURRENTLY ] [ IF EXISTS ] name [ , ... ]
[ CASCADE | RESTRICT ]
REINDEX

REINDEX [ ( { VERBOSE } [, ...] ) ]

{ INDEX | TABLE | SCHEMA | DATABASE | SYSTEM } name
CREATE UNIQUE INDEX ON product 
  ((listened_at::date));

CREATE UNIQUE INDEX ON product 
  (lower(name));

CREATE EXTENSION unaccent;

CREATE INDEX ON product 
  (lower(unaccent(name)));
WHERE Clause Example

CREATE UNIQUE INDEX ON product (stock_id) WHERE status = 'available';

SELECT * FROM product WHERE status = 'available' AND stock_id = 3;
Index Access Nodes

- Index Scan
  - Ordered
- Bitmap Index Scan
- Index Only Scan
  - Ordered
Index Access Methods in PostgreSQL 9.5

- B-tree
- GiST
- SP-GiST
- GIN
- BRIN

• (Hash)
B-tree
# B-tree operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>=&gt;</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td><del>&lt;</del></td>
<td>Variations on text_pattern_ops</td>
</tr>
<tr>
<td><del>&gt;</del></td>
<td></td>
</tr>
<tr>
<td><del>&lt;&lt;=</del></td>
<td></td>
</tr>
<tr>
<td><del>=&gt;</del></td>
<td></td>
</tr>
</tbody>
</table>
# CREATE INDEX ON product (name text_pattern_ops);

# EXPLAIN SELECT * FROM product
     WHERE name LIKE 'aa%';

QUERY PLAN
-----------------------------------------------------------------------
Bitmap Heap Scan on product
Filter: (name ~ 'aa%':text)
     -> Bitmap Index Scan on product_name_idx1
        Index Cond: ((name ~=>'aa':text) AND (name ~<'ab':text))
Sort by min
Sort by max
Group into clusters
## GiST operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Left of</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>Overlaps</td>
</tr>
<tr>
<td><code>&gt;&gt;</code></td>
<td>Right of</td>
</tr>
<tr>
<td><code>@&gt;</code></td>
<td>Contains</td>
</tr>
<tr>
<td><code>&lt;@</code></td>
<td>Contained by</td>
</tr>
<tr>
<td><code>&lt;-&gt;</code></td>
<td>Distance</td>
</tr>
</tbody>
</table>
Nearest Neighbor Search

SELECT * FROM people
ORDER BY point <-> '(3,5)' LIMIT 5;
Exclusion Constraints

# ALTER TABLE allowed_network
  ADD CONSTRAINT allowed_network_address_excl
  EXCLUDE USING gist (address WITH &&);

# ALTER TABLE meeting
  ADD CONSTRAINT meeting_room_excl
  EXCLUDE USING gist (period WITH &&,
                      room_id WITH =
                       );
Space-Partitioned GiST
Prefix Tree

- A
  - NKARA
  - MSTERDAM
- B
  - HELSINKI
    - U
      - CHAREST
      - DAPEST
  - NACO
    - SCOW
  - IN
    - L
      - RN
        - GRADE
GIN

```
       1
      / \  
     3   4
    / \   |
   6   4   |
    |     |  
   7   6   
     |     |
    9     
```

Nodes connected by edges form sets as follows:
- 3 connects to 1 and 4
- 6 connects to 3, 4, and 7
- 7 connects to 6
- 4 connects to 1
- 4 and 9 form a set
- 1, 5, and 9 form a set
## pg_trgm

# CREATE EXTENSION pg_trgm;

# CREATE INDEX ON people USING gin (name gin_trgm_ops);

# EXPLAIN SELECT * FROM people
   WHERE name ~ '.*(a|b)cd.*';

---

**QUERY PLAN**

-----------------------------------------------
Bitmap Heap Scan on people
    Recheck Cond: (name ~ '.*(a|b)cd.*':::text)
    -> Bitmap Index Scan on people_name_idx
       Index Cond: (name ~ '.*(a|b)cd.*':::text)
# CREATE EXTENSION hstore;

# CREATE INDEX ON server USING gin (attributes);

# EXPLAIN SELECT * FROM server
  WHERE attributes @> 'market => en';

QUERY PLAN

-----------------------------------------------------------------------
Bitmap Heap Scan on server
  Recheck Cond: (attributes @> ""market"=>"en"::hstore)
  ->  Bitmap Index Scan on server_attributes_idx
      Index Cond: (attributes @> ""market"=>"en"::hstore)
{ "firstName": "John", "lastName": "Smith", "isAlive": true, "age": 25, "address": { "streetAddress": "21 2nd Street", "city": "New York", "state": "NY", "postalCode": "10021-3100" }, "phoneNumbers": [ { "type": "home", "number": "212 555-1234" }, { "type": "office", "number": "646 555-4567" } ], "children": [], "spouse": null }
# SELECT DISTINCT category FROM product;

<table>
<thead>
<tr>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>book.fiction.fantasy</td>
</tr>
<tr>
<td>book.fiction.horror</td>
</tr>
<tr>
<td>book.nonfiction.art</td>
</tr>
<tr>
<td>book.nonfiction.history</td>
</tr>
<tr>
<td>electronic</td>
</tr>
</tbody>
</table>

# CREATE INDEX ON product USING gist (category);

# SELECT * FROM product WHERE category <@ 'book '::ltree;
Block Range Index

0: Amsterdam – Astana
1: Athens – Berlin
2: Bern – Bucharest
3: Budapest – Dublin
4: Helsinki – Ljubljana
### Ordering

UPDATE cities SET name='Zagreb' WHERE ...

<table>
<thead>
<tr>
<th>Order</th>
<th>City 1</th>
<th>City 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Amsterdam</td>
<td>Zagreb</td>
</tr>
<tr>
<td>1:</td>
<td>Athens</td>
<td>Zagreb</td>
</tr>
<tr>
<td>2:</td>
<td>Bern</td>
<td>Zagreb</td>
</tr>
<tr>
<td>3:</td>
<td>Budapest</td>
<td>Zagreb</td>
</tr>
<tr>
<td>4:</td>
<td>Helsinki</td>
<td>Zagreb</td>
</tr>
</tbody>
</table>

**Cities:**
- Amsterdam
- Ankara
- Astana
- Athens
- Baku
- Berlin
- Bern
- Brussels
- Bucharest
- Budapest
- Copenhagen
- Dublin
- Helsinki
- Kiev
- Ljubljana
- Zagreb
SELECT * FROM people
WHERE point @ '(0,0),(3,5)'::box;
Summary

- **B-tree**
  - Default
  - Unique indexes

- **GiST**
  - Containment
  - KNN search

- **SP-GIST**
  - Non-overlapping

- **GIN**
  - Multiple values per row
  - Stores duplicates efficiently

- **BRIN**
  - Containment
  - For ordered data
  - Tiny index