Efficient K-nearest neighbour search in PostgreSQL

Oleg Bartunov, Teodor Sigaev
Knn-search: The Problem

```
  knn=# select id, date, event from events
      order by date <-> '1957-10-04'::date asc limit 10;
```

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td>58137</td>
<td>1957-10-04</td>
<td>U.S.S.R. launches Sputnik I, 1st artificial Earth satellite</td>
</tr>
<tr>
<td>58136</td>
<td>1957-10-04</td>
<td>&quot;Leave It to Beaver,&quot; debuts on CBS</td>
</tr>
<tr>
<td>117062</td>
<td>1957-10-04</td>
<td>Gregory T Linteris, Demarest, New Jersey, astronaut, sk: STS 83</td>
</tr>
<tr>
<td>117061</td>
<td>1957-10-04</td>
<td>Christina Smith, born in Miami, Florida, playmate, Mar, 1978</td>
</tr>
<tr>
<td>102670</td>
<td>1957-10-05</td>
<td>Larry Saumell, jockey</td>
</tr>
<tr>
<td>31456</td>
<td>1957-10-03</td>
<td>Willy Brandt elected mayor of West Berlin</td>
</tr>
<tr>
<td>58291</td>
<td>1957-10-05</td>
<td>12th Ryder Cup: Britain-Ireland, 7 -4 at Lindrick GC, England</td>
</tr>
<tr>
<td>58290</td>
<td>1957-10-05</td>
<td>11th NHL All-Star Game: All-Stars beat Montreal 5-3 at Montreal</td>
</tr>
<tr>
<td>58292</td>
<td>1957-10-05</td>
<td>Yugoslav dissident Milovan Djilos sentenced to 7 years</td>
</tr>
<tr>
<td>102669</td>
<td>1957-10-05</td>
<td>Jeanne Evert, tennis player, Chris' sister</td>
</tr>
</tbody>
</table>

(10 rows)

Time: 115.548 ms

- Very inefficient:
  - Full table scan, btree index on date won't help.
  - Sort full table
Knn-search: Existing solutions

- Traditional way to speedup query
  - Use indexes - very inefficient (no search query!)
    - Scan full index
    - Full table scan, but in random order!
    - Sort full table
    - Better not to use index at all!
  - Constrain data space (range search)
    - Incremental search → too many queries
    - Need to know in advance the size of neighbourhood, how? 1 Km is ok for Paris, but too small for Siberia
    - Maintain 'density map'?
Knn-search: What do we want!

- We want to avoid full table scan – read only right tuples
  - So, we need index
- We want to avoid sorting – read right tuples in right order
  - So, we need special strategy to traverse index
- We want to support tuples visibility
  - So, we should be able to resume index traverse
R-tree index

- Visualization of R-tree index using Gevel

- Greece
  (data from rtreeportal.org)
Knn-search: Index traverse

- Depth First Search (stack, LIFO)
  - R-tree search

- Breadth First Search (queue, FIFO)

- Both strategies are not good for us – full index scan
Knn-search: Index traverse

- Best First Search (PQ, priority queue). Maintain order of items in PQ according their distance from given point
  - Distance to MBR (rectangle for Rtree) for internal pages – minimum distance of all items in that MBR
  - Distance = 0 for MBR with given point
  - Distance to point for leaf pages
- Each time we extract point from PQ we output it – it is next closest point! If we extract rectangle, we expand it by pushing their children (rectangles and points) into the queue.
- We traverse index by visiting only interesting nodes!
Knn-search: Performance

- SEQ (no index) – base performance
  - Sequentially read full table + Sort full table (can be very bad, sort_mem !)
- DFS – very bad !
  - Full index scan + Random read full table + Sort full table
- BFS – the best for small k !
  - Partial index scan + Random read k-records
    - $T(\text{index scan}) \sim \text{Height of Search tree} \sim \log(n)$
    - Performance win BFS/SEQ $\sim \text{Nreldpages/k, for small k.}$
      - The more rows, the more benefit !
    - Can still win even for $k=n$ (for large tables) - no sort !
Knn-search: What do we want!

- + We want to avoid full table scan – read only right tuples
  - So, we need index
- + We want to avoid sorting – read right tuples in right order
  - So, we need special strategy to traverse index
- + We want to support tuples visibility
  - So, we should be able to resume index traverse
- We want to support many data types
  - So, we need to modify GiST
Knn-search: modify GiST

• GiST – Generalized Search Tree, provides
  – API to build custom disk-based search trees (any tree, where key of internal page is a Union of keys on children pages)
  – Recovery and Concurrency
  – Data type and query extendability
• GiST is widely used in GIS (PostGIS), text search, astro, bio, ...
• Current strategy of search tree traverse is DFS
  – Change to BFS (Best First Search) strategy
  – Retain API compatibility
**Depth First Search**

push Stack, Root;
While Stack {
  If p is heap {
    output p;
  } else {
    children = get_children(p);
    push Stack, children;
  }
}

**Best First Search**

push PQ, Root;
While PQ {
  If p is heap {
    output p;
  } else {
    Children = get_children(p);
    push PQ, children;
  }
}

- For non-knn search all distances are zero, so PQ => Stack and BFS => DFS
- We can use only one strategy (BFS) for both – normal search and knn-search!
Knn-search: What do we want!

- + We want to avoid full table scan – read only <right> tuples
  - So, we need index
- + We want to avoid sorting – read <right> tuples in <right> order
  - So, we need special strategy to traverse index
- + We want to support tuples visibility
  - So, we should be able to resume index traversal
- + We want to support many data types
  - So, we need to modify GiST
Knn-search: syntax

- Knn-query uses ORDER BY clause

```sql
SELECT ... FROM ... WHERE ...
ORDER BY p <-> '(0.,0.)'::point
LIMIT k;
```

-\(\leftrightarrow\) - distance operator, should be provided for data type
Knn-search: Examples

- Synthetic data – 1,000,000 randomly distributed points

```sql
create table qq ( id serial, p point, s int4);
insert into qq (p, s) select point( p.lat, p.long),
(random()*1000)::int
from ( select (0.5-random())*180 as lat, random()*360 as long
      from ( select generate_series(1,1000000) ) as t ) as p;
create index qq_p_s_idx on qq using gist(p);
analyze qq;
```

- Query – find k-closest points to (0,0)

```sql
set enable_indexscan=on|off;
explain (analyze on, buffers on)
select * from qq order by (p <-> '(0,0)') asc limit 10;
```
Knn-search: Examples

- postgresql.conf:
  
  - shared_buffers = 512MB #32MB
  - work_mem = 32MB #1MB
  - maintenance_work_mem = 256MB #16MB
  - checkpoint_segments = 16
  - effective_cache_size = 1GB #128MB

- Index statistics (n=1000,000)

  - Number of levels: 3
  - Number of pages: 8787
  - Number of leaf pages: 8704
  - Number of tuples: 1008786
  - Number of invalid tuples: 0
  - Number of leaf tuples: 1000000
  - Total size of tuples: 44492028 bytes
  - Total size of leaf tuples: 44104448 bytes
  - Total size of index: 71983104 bytes
Knn-search: Examples

\[ k=1, \ n=1,000,000 \]

Limit (cost=0.00..0.08 rows=1 width=24) (actual time=0.104..0.104 rows=1 loops=1)
  Buffers: shared hit=4
  ->  Index Scan using qq_p_idx on qq (cost=0.00..82060.60 rows=1000000 width=24) (actual time=0.104..0.104 rows=1 loops=1)
      Sort Cond: (p <-> '(0,0)'::point)
      Buffers: shared hit=4

Total runtime: 0.117 ms  4000 times faster!

Limit (cost=24853.00..24853.00 rows=1 width=24) (actual time=469.129..469.130 rows=1 loops=1)
  Buffers: shared hit=7353
  ->  Sort (cost=24853.00..27353.00 rows=1000000 width=24) (actual time=469.128..469.128 rows=1 loops=1)
      Sort Key: ((p <-> '(0,0)'::point))
      Sort Method: top-N heapsort  Memory: 25kB
      Buffers: shared hit=7353
  ->  Seq Scan on qq (cost=0.00..19853.00 rows=1000000 width=24) (actual time=0.007..241.539 rows=1000000 loops=1)
      Buffers: shared hit=7353

Total runtime: 469.150 ms
# Knn-search: Examples

## N=1,000,000

<table>
<thead>
<tr>
<th>k</th>
<th>hit</th>
<th>knn</th>
<th>seq</th>
<th>sortmem (seq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.117</td>
<td>469.150</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>0.289</td>
<td>471.735</td>
<td>25</td>
</tr>
<tr>
<td>100</td>
<td>118</td>
<td>0.872</td>
<td>468.244</td>
<td>32</td>
</tr>
<tr>
<td>1000</td>
<td>1099</td>
<td>7.107</td>
<td>473.840</td>
<td>127</td>
</tr>
<tr>
<td>10000</td>
<td>10234</td>
<td>31.629</td>
<td>525.557</td>
<td>1550</td>
</tr>
<tr>
<td>100000</td>
<td>101159</td>
<td>321.182</td>
<td>994.925</td>
<td>13957</td>
</tr>
</tbody>
</table>
# Knn-search: Examples

\[ n = 10,000 \]

<table>
<thead>
<tr>
<th>K</th>
<th>hit</th>
<th>knn</th>
<th>seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.117</td>
<td>6.072</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>0.247</td>
<td>5.014</td>
</tr>
<tr>
<td>100</td>
<td>103</td>
<td>0.295</td>
<td>6.381</td>
</tr>
<tr>
<td>1000</td>
<td>996</td>
<td>1.605</td>
<td>8.670</td>
</tr>
</tbody>
</table>
| 10000| 9916 | 16.487| 14.706 | -> knn lose if k=n, n is small
Knn-search: Examples

- Real data
  - 2 mln points
  - US, geonames
Knn-search: Examples

- Query: find 10 closest points in US to the point (5,5) with 'mars' in names - create composite index:

```sql
create index pt_fts_idx on geo
    using gist(point, to_tsvector('english', asciiname));
```

```sql
=# explain (analyze on, buffers on)
select asciiname, point, (point <-> '5.0,5.0'::point) as dist from geo
where to_tsvector('english', asciiname) @@ to_tsquery('english', 'mars')
order by dist asc limit 10;
```

```
QUERY PLAN

Limit  (cost=0.00..33.55 rows=10 width=35) (actual time=0.452..0.597 rows=10 loops=1)
Buffers: shared hit=56
  ->  Index Scan using pt_fts_idx on geo  (cost=0.00..34313.91 rows=10227 width=35)
      (actual time=0.452..0.592 rows=10 loops=1)
        Index Cond: (to_tsvector('english'::regconfig, (asciiname)::text) @@ 'mar'::tsquery)
        Sort Cond: (point <-> '5.0,5.0'::point)
        Buffers: shared hit=56
Total runtime: 0.629 ms
```

Oleg Bartunov     PGDay-2010, Roma, Dec 10, 2010
## Knn-search: The Problem

### Query

```sql
knn=# select id, date, event from events
    order by date <-> '1957-10-04'::date asc limit 10;
```

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td>58137</td>
<td>1957-10-04</td>
<td>U.S.S.R. launches Sputnik I, 1st artificial Earth satellite</td>
</tr>
<tr>
<td>58136</td>
<td>1957-10-04</td>
<td>&quot;Leave It to Beaver,&quot; debuts on CBS</td>
</tr>
<tr>
<td>117062</td>
<td>1957-10-04</td>
<td>Gregory T Linteris, Demarest, New Jersey, astronaut, sk: STS 83</td>
</tr>
<tr>
<td>117061</td>
<td>1957-10-04</td>
<td>Christina Smith, born in Miami, Florida, playmate, Mar, 1978</td>
</tr>
<tr>
<td>102670</td>
<td>1957-10-05</td>
<td>Larry Saumell, jockey</td>
</tr>
<tr>
<td>31456</td>
<td>1957-10-03</td>
<td>Willy Brandt elected mayor of West Berlin</td>
</tr>
<tr>
<td>58291</td>
<td>1957-10-05</td>
<td>12th Ryder Cup: Britain-Ireland, 7 -4 at Lindrick GC, England</td>
</tr>
<tr>
<td>58290</td>
<td>1957-10-05</td>
<td>11th NHL All-Star Game: All-Stars beat Montreal 5-3 at Montreal</td>
</tr>
<tr>
<td>58292</td>
<td>1957-10-05</td>
<td>Yugoslav dissident Milovan Djilos sentenced to 7 years</td>
</tr>
<tr>
<td>102669</td>
<td>1957-10-05</td>
<td>Jeanne Evert, tennis player, Chris' sister</td>
</tr>
</tbody>
</table>

**Time:** 115.548 ms

- Very inefficient:
  - Full table scan, btree index on date won't help.
  - Sort full table
Knn-search: The Problem

contrib/btree_gist

\[
\text{knn=} \# \text{ select id, date, event from events} \\
\text{order by date <-> '1957-10-04'::date asc limit 10;}
\]

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td>58137</td>
<td>1957-10-04</td>
<td>U.S.S.R. launches Sputnik I, 1st artificial Earth satellite</td>
</tr>
<tr>
<td>58136</td>
<td>1957-10-04</td>
<td>&quot;Leave It to Beaver,&quot; debuts on CBS</td>
</tr>
<tr>
<td>117062</td>
<td>1957-10-04</td>
<td>Gregory T Linteris, Demarest, New Jersey, astronaut, sk: STS 83</td>
</tr>
<tr>
<td>117061</td>
<td>1957-10-04</td>
<td>Christina Smith, born in Miami, Florida, playmate, Mar, 1978</td>
</tr>
<tr>
<td>102670</td>
<td>1957-10-05</td>
<td>Larry Saumell, jockey</td>
</tr>
<tr>
<td>31456</td>
<td>1957-10-03</td>
<td>Willy Brandt elected mayor of West Berlin</td>
</tr>
<tr>
<td>58291</td>
<td>1957-10-05</td>
<td>12th Ryder Cup: Britain-Ireland, 7 -4 at Lindrick GC, England</td>
</tr>
<tr>
<td>58290</td>
<td>1957-10-05</td>
<td>11th NHL All-Star Game: All-Stars beat Montreal 5-3 at Montreal</td>
</tr>
<tr>
<td>58292</td>
<td>1957-10-05</td>
<td>Yugoslav dissident Milovan Djilos sentenced to 7 years</td>
</tr>
<tr>
<td>102669</td>
<td>1957-10-05</td>
<td>Jeanne Evert, tennis player, Chris' sister</td>
</tr>
</tbody>
</table>

Time: 0.590 ms

- Very inefficient:
  - 8 index pages read + 10 tuples read, no sorting
  - No sorting
  - 200 times faster!
Committed to 9.1