Generalized Inverted Index

• An inverted index is an index structure storing a set of (key, posting list) pairs, where 'posting list' is a set of documents in which the key occurs.

• Generalized means that the index does not know which operation it accelerates. It works with custom strategies, defined for specific data types. GIN is similar to GiST and differs from B-Tree indices, which have predefined, comparison-based operations.
GIN Structure

Entry page, level 0 (leaf)

aaa
- Pointer to posting tree: B-Tree over ItemPointer to heap

abc
- Posting list: sorted array of ItemPointer to heap

Entry page, level 0

baa
- bar

Posting page, level N: ItemPointer

14:17 218:1 1021:6

Posting page, level 0 (leaf)

1:33 2:7 14:17
- Right bound 14:17

Posting page, level 0 (leaf)

123:1 158:18
- Right bound 218:1
GIN features

• Concurrency
  – Lehman and Yao's high-concurrency B-tree management algorithm

• WAL
  – Recovery

• User-defined opclasses
  – The scheme is similar to GiST
GIN Interface

Four interface functions (pseudocode):

• Datum* extractValue(Datum inputValue, uint32* nentries)

• int compareEntry(Datum a, Datum b)

• Datum* extractQuery(Datum query, uint32* nentries, StrategyNumber n)

• bool consistent(bool check[], StrategyNumber n, Datum query)
GIN Interface: extractValue

Datum* extractValue(Datum inputValue, uint32* nentries)

Returns an array of Datum of entries of the value to be indexed. nentries should contain the number of returned entries.

Tsearch2 example: inputValue is tsvector, output is array of text type, containing lexemes.
int compareEntry(Datum a, Datum b)

Compares two entries (not the indexing values), returns <0, 0, >0

Tsearch2 example: built-in bttextcmp(), used for built-in B-Tree index over texts.
GIN Interface: extractQuery

Datum* extractQuery(Datum query, uint32* nentries, StrategyNumber n)

Returns an array of Datum of entries of the query to be executed. n is the strategy number of the operation. Depending on n, query can be different type.

Tsearch2 example: query is tsquery, output is array of text type, containing lexemes.
bool consistent(bool check[],
StrategyNumber n, Datum query)

Each element of the check array is true if
the indexed value has a corresponding
entry in the query: if (check[i] = TRUE)
then the i-th entry of the query is
present in the indexed value. The
function should return true if the
indexed value matches by
StrategyNumber and the query.
GIN Interface: consistent

Tsquery: fat & ( cat | dog )

Posting lists (logically) of ItemPointers

<table>
<thead>
<tr>
<th>Posting</th>
<th>Posting</th>
<th>Posting</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:7</td>
<td>1:15</td>
<td>1:15</td>
</tr>
<tr>
<td>1:15</td>
<td>15:7</td>
<td>1:15</td>
</tr>
<tr>
<td>33:11</td>
<td>33:11</td>
<td>33:11</td>
</tr>
<tr>
<td>33:11</td>
<td>34:1</td>
<td>34:1</td>
</tr>
</tbody>
</table>

bool check[]

<table>
<thead>
<tr>
<th>bool check[]</th>
<th>Consistent function</th>
</tr>
</thead>
<tbody>
<tr>
<td>T,F,F</td>
<td>T&amp;(F</td>
</tr>
<tr>
<td>T,F,T</td>
<td>T&amp;(F</td>
</tr>
<tr>
<td>F,T,F</td>
<td>F&amp;(T</td>
</tr>
<tr>
<td>T,T,T</td>
<td>T&amp;(T</td>
</tr>
<tr>
<td>F,T,T</td>
<td>F&amp;(T</td>
</tr>
</tbody>
</table>
GIN: create index flow

extractValue()

Tuple to index
Value (tsvector):
cat:2 fat:1
ItemPointer: 17:9

maintenance_work_mem cache:
Sorted array of entries
Arrays of ItemPointers

cat
12:3, 14:5, 15:1, 17:9
foo
1:1, 2:5, 15:3
fat
2:3, 17:9

HDD
Gin opclasses

- Built-in support for any one-dimensional array
  - && - overlap
  - @ - contains
  - ~ - contained
- Tsearch2
- Intarray – enhanced support for int4[]
GIN tips

• GUC variable:
  `gin_fuzzy_search_limit` - soft upper limit on the returned results for very frequent words

• Create is much faster than inserts
GIN limitations

- No support for multicolumn indices
- GIN doesn't use scan->kill_prior_tuple & scan->ignore_killed_tuples
- GIN searches entries only by equality matching
- GIN doesn't support full scans of index
- GIN doesn't index NULL values
• Two kinds of NULL
  • (NULL = NULL) is NULL
  • ('{NULL}'::int[]='{NULL}') is TRUE
• Multidimensional arrays: &&, @, ~ ?
  • '{1,2},{3,4}' @ '{2,3}' - ?
• Recent fillfactor patch – nested B-Tree