University of Zurich[UZH]

Language Technology and Web Applications

Databases II

Johannes Graën

Wednesday 1st November, 2023

Department of Computational Linguistics & Linguistic Research Infrastructure, University of Zurich

What is still missing?

- indices
- (full text search)

Overview

Indices

Full Text Search

Learning Goals for this Week

- You can explain what indices are good for and know how to use them

Indices

Indices

- searching data in a table linearly: $O(n)$
- ... feasible for considerably small numbers of record
- searching data in cross-joined tables (self join): $O(n^2)$
- ... feasible for considerably small numbers of records of the cartesian product

$\Rightarrow$ linear search is not a good strategy for most queries

- index search considerably faster: $O(\log n)$

Index Types

- B-Tree (balanced tree)
- Hash
- GIN (Generalized Inverted Index)
- GiST (Generalized Search Tree)

https://www.postgresql.org/docs/current/indexes-types.html

B-Tree Index

Operations supported by index type:

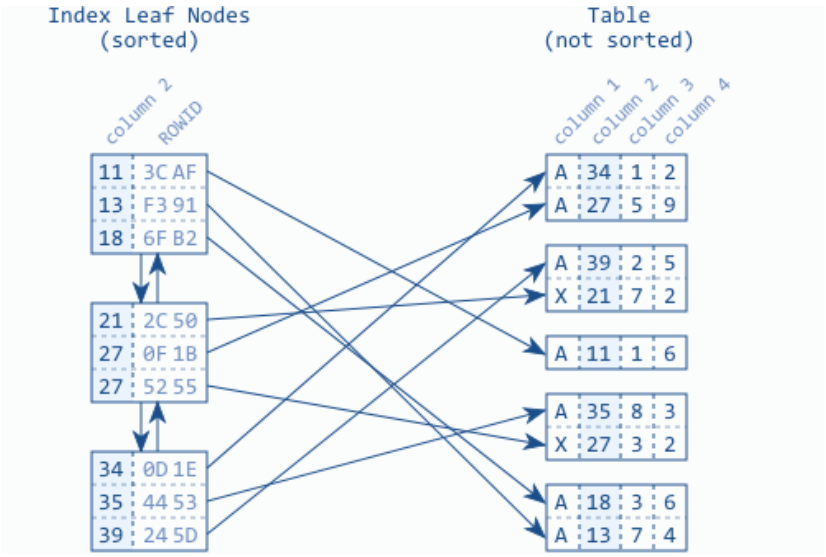
- equality (=)
- inequality (<, <=, >=, >)

Examples:

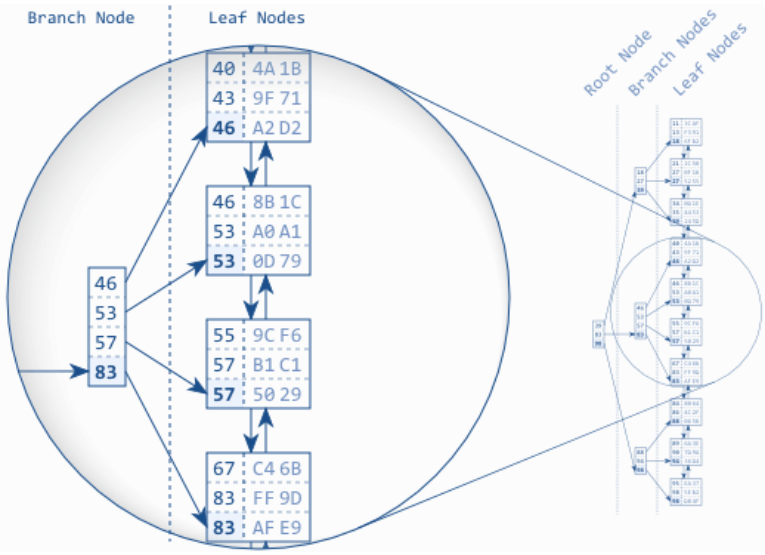
```
SELECT *
FROM large_corpus
WHERE word = 'Калашников'
```

```
SELECT *
FROM event
WHERE event_date BETWEEN '2022-01-01' AND '2022-12-31'
```

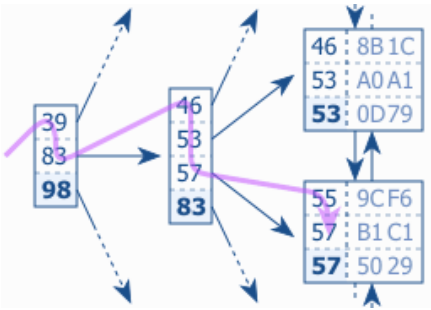
B-Tree (1)


Index Leaf Nodes
(sorted)
column 2
ROWID
11 3C AF
13 F3 91
18 6F B2
21 2C 50
27 0F 1B
27 52 55
34 0D 1E
35 44 53
39 24 5D
Table
(not sorted)
column 1
column 2
column 3
column 4
A 34 1 2
A 27 5 9
A 39 2 5
X 21 7 2
A 11 1 6
A 35 8 3
X 27 3 2
A 18 3 6
A 13 7 4

B-Tree (2)


Branch Node
Leaf Nodes
40 4A 1B
43 9F 71
46 A2 D2
46 8B 1C
53 A0 A1
53 0D 79
55 9C F6
57 B1 C1
57 50 29
67 C4 6B
83 FF 9D
83 AF E9
46
53
57
83
Root Node
Branch Nodes
Leaf Nodes

B-Tree (3)



©Images: *https://use-the-index-luke.com/*

Hash Index

Only operation supported by index type:

- equality (=)

Advantages:

- smaller on attributes with larger values (“UUIDs, URLs”)
- “SELECT and UPDATE-heavy workloads”
- “single-column indexes” (vs. multi-column indices)

Examples:

```
SELECT *
FROM large_corpus
WHERE word = 'Калашников'
```

https://www.postgresql.org/docs/current/hash-intro.html

GIN Indices

Operations supported by the standard GIN index:

- equality (=)
- containment (<@, @>)
- overlap (&&)

Examples:

```
SELECT ARRAY[1,2,3] =  ARRAY[1,3]; -- false
SELECT ARRAY[1,2,3] <@ ARRAY[1,3]; -- false
SELECT ARRAY[1,2,3] @> ARRAY[1,3]; -- true
SELECT ARRAY[1,2]   && ARRAY[1,3]; -- true
```

https://www.postgresql.org/docs/current/gin-builtin-opclasses.html

GiST Indices

Operations supported by the standard GiST index:

- geometric relations
- GIN operators for different data types

Examples:

```
SELECT '<(0,0),1>'::circle <@ '<(2,0),3>'::circle; -- true
SELECT '<(0,0),1>'::circle <@ '<(2,1),3>'::circle; -- false

SELECT *
FROM places
ORDER BY location <-> point '(101,456)'
LIMIT 1;
```

https://www.postgresql.org/docs/current/gist-builtin-opclasses.html

Advanced Indexing

- functional indices (e.g. **lower(word)**
- multicolumn indices (another index instead of leave node)

Full Text Search

Full Text Search (Intro)

How to index text?

- with a B-Tree index ⇒ prefix search
- with a Hash index ⇒ exact matches
- with a GIN index ⇒ something **in** text

Example

- file ‘dewac.tsv’ contains 44m sentences from ‘deWaC’ (German web as a corpus)[1]
- tokens are separated by whitespace

[1] *https://wacky.sslmit.unibo.it*

Trigram Extension

```
CREATE EXTENSION pg_trgm;
SELECT show_trgm('Hundesteuer');
```

- * h*
- * hu*
- *des*
- *er *
- *est*
- *eue*
- *hun*
- *nde*
- *ste*
- *teu*
- *uer*
- *und*

Inverted Index

Word in document:

	doc1	doc2	doc3	doc4
Hund		x		x
Katze		x	x	x
Tierarzt			x	x
Vermögenssteuer	x	x		

Trigram in sentence:

	sent1	sent2	sent3	sent4
eue		x		x
ste	x	x	x	
ock	x		x	x
ühn		x	x	

Query on Textual Data

```
SELECT *
FROM dewac
WHERE sentence LIKE '%Katze%' AND sentence LIKE '%Hund%';
```

Trigram Index

```
CREATE INDEX dewac_sentence_gin_idx ON dewac
    USING GIN (sentence gin_trgm_ops);
```

Query on Textual Data (again)

```
SELECT *
FROM dewac
WHERE sentence LIKE '%Katze%' AND sentence LIKE '%Hund%';
```

LIKE Operator for String Matches

We also get the following hits:

- Hunde und Katzen
- das Lied des Katzenpatriarchen
- eine “Katze-frißt-Hund”-Szene
- Straßen-Hunden und -Katzen
- Hunde- oder Katzengegner
- Katzen- und Hundeausstellungen
- Der Katzenfloh ist unternehmungslustiger als der Hundefloh
- ...

Full Text Search (FTS)

- represent tokens as points in a vector space
- i.e. map them to numbers
- use stems instead of full word forms (reduces number of ‘lexemes’)
- add positional information as list

Example

```
SELECT ts_lexize('german_stem', 'Sobald'); -- {sobald}
SELECT ts_lexize('german_stem', 'die'); -- {}
SELECT ts_lexize('german_stem', 'Polnische'); -- {polnisch}
SELECT ts_lexize('german_stem', 'Provisorische'); -- {provisor}
SELECT ts_lexize('german_stem', 'Regierung'); -- {regier}
```

Example

```
SELECT to_tsvector('german',sentence) FROM dewac WHERE id = 1234;
```

Debug mode:

```
SELECT (ts_debug('german', sentence)).*
FROM dewac
WHERE id = 1234;
```