
I am inevitable.
imgflip.com


TA
Did you install PostgreSQL?
STUDENT
I dont know
what is PostgreSQL
imgflip.com

Database Systems
Indexes in Databases(PostgreSQL)

Week 5 – Lab


INDEXES EVERYWHERE
LIGHTYEAR
PostgreSQL
imgflip.com

Todays lab

- In this lab we will populate our PostgreSQL database using a Python script & we will create multiple indexes to compare their performance
- For your Python code you can use any library that you like (for example pg8000)
- Use this as a reference for PostgreSQL

Warm up Exercise

- Using a library of choice connect to your PostgreSQL database
- Create a table of customers
- Each customer should have:
  - A unique ID
  - Name
  - Address
  - Age
  - A review that he/she has left for some product (for simplicity you can insert randomly generated text here)
- Generate and insert 100k customers into the table

- Hint https://github.com/enghamzasalem/Create_db_faker


B
Gin
#
GIST develop in the exterior
areas of the stomach wall
imgflip.com

Generalized Inverted Indexes

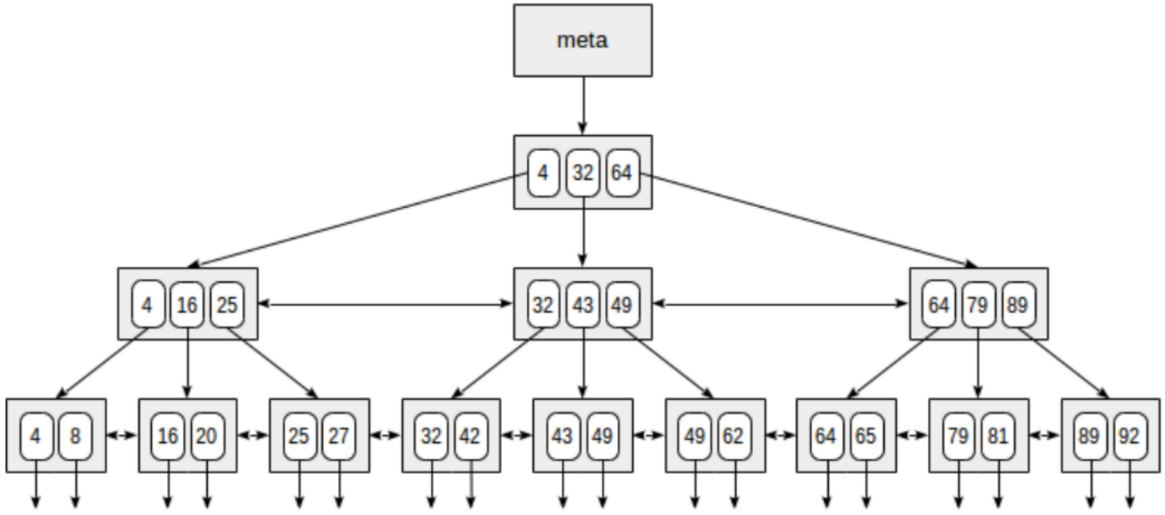
Generalized Inverted Indexes

Generalized Search Tree indexes

imgflip.com

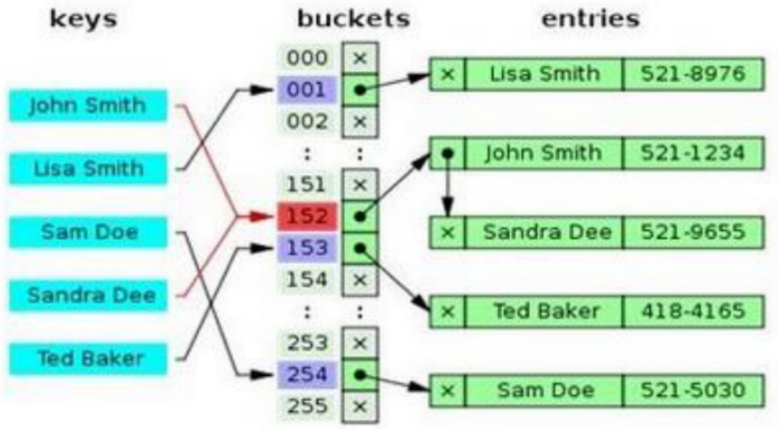
B tree Index

- B-tree index type, implemented as «btree» access method, is suitable for data that can be sorted. In other words, «greater», «greater or equal», «less», «less or equal», and «equal» operators must be defined for the data type. Note that the same data can sometimes be sorted differently.



Hash Index

- The idea of hashing is to associate a small number (from 0 to *N*–1, *N* values in total) with a value of any data type. Association like this is called *a hash function*. The number obtained can be used as an index of a regular array where references to table rows (TIDs) will be stored. Elements of this array are called *hash table buckets* — one bucket can store several TIDs if the same indexed value appears in different rows.



GIN index

- GIN (or Generalized Inverted Indexes) are useful when an index must map many values to one row, whereas B-Tree indexes are optimized for when a row has a single key value. GINs are good for indexing array values as well as for implementing full-text search.
- More about GIN index here.

GIST Index

- GiST (or Generalized Search Tree) indexes allow you to build general balanced tree structures, and can be used for operations beyond equality and range comparisons. They are used to index the geometric data types, as well as full-text search.
- More on GiST here.

Demo pgAdmin

- Use DVD DB to execute these queries:-
  A. Explain ANALYZE SELECT * FROM public.payment
  B. Explain ANALYZE SELECT * FROM public.payment where amount>2 and amount<4
  C. Explain SELECT * FROM public.payment where amount=2
- What is the different?
- Seq Scan on payment (**cost=0.00..253.96** rows=14596 width=26)
- Seq Scan on payment (**cost=0.00..326.94** rows=4228 width=26)
- ???
- Create Index on Payment and execute B

  (cost=**0.00..110.56** rows=4228 width=0)

Exercise 1

- Explore the generated data and try to query it using Python or pgAdmin.. For Example, use SELECT statement.
- Create single-column b-tree and hash indexes on the previously created table using any fields you like (but different fields for each!)
- Create a Python script that gets the data from the same queries and shows the elapsed time using EXPLAIN
- Is there any difference? Which queries are faster? (If you can’t see the difference try to increase the generated data to 1M)

- Hint : to check time for a query add EXPLAIN ex:

EXPLAIN analyze SELECT * FROM foo WHERE foo.bar = 'infrastructure as a service' OR foo.bar = 'iaas';

Exercise 2

- Try to query the review column using full-text search queries, measure the elapsed time again
- Create GIN and GiST indexes separately on the review column.
- Make the same query again
- Does it affect performance?

Useful resources

- https://github.com/tlocke/pg8000
- https://github.com/enghamzasalem/Create_db_faker
- https://www.postgresql.org/docs/10/index.html
- https://www.postgresql.org/docs/current/btree.html
- https://www.postgresql.org/docs/current/gist.html
- https://www.postgresql.org/docs/current/gin.html

See you next week ☺