NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

CASE STUDY

on

B-TREE DATABASE INDEXING TECHNIQUE

Submitted in partial fulfilment of the requirement for the award of Degree of

Bachelor of Engineering

in

Information Science and Engineering

Submitted by:

Jayam Naga tulasi	1NT20IS066
A N Kruthika Chowdary	1NT20IS001
Jaya shiva Darshini	1NT20IS065
Deepak. H	1NT20IS046

Under the Guidance of

Dr. Tejaswini R Murgod,Associate Professor, Dept. of ISE, NMIT

Department of Information Science and Engineering

(Accredited by NBA Tier-1)

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAU

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

Department of Information Science and Engineering

(Accredited by NBA Tier-1)

KNOWLEDGE * CHARACTER * UNITY

CERTIFICATE

This is to certify that the Project Report on "**B-Tree Database Indexing Technique"** is an authentic work carried out by **JayamNagatulasi (1NT20IS001),A N Kruthika Chowdary(1NT20IS001), Jaya shiva Darshini (1NT19IS067),Deepak.H(1NT20IS046)** Bonafede students of Nitte Meenakshi Institute of Technology, Bangalore in partial fulfilment for the award of the degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2021-2022**.** It is certified that all corrections and suggestions indicated during the internal assessment has been incorporated in the report.

Internal Guide	**Signature of the HOD**
Dr. Tejaswini R Murgod	Dr. Mohan SG
Associate Professor	Professor, Head, Dept. ISE,
Dept. ISE	NMIT Bangalore

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. I express my sincere gratitude to our Principal **Dr. H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing facilities.

We wish to thank our HOD**, Dr. Mohan S. G.** for the excellent environment created to further educational growth in our college. We also thank him for the invaluable guidance provided which has helped in the creation of a better project.

I hereby like to thank our ***Dr.Tejaswini R Murgod, Associate Professor*** Department of Information Science & Engineering on their periodic inspection, time to time evaluation of the project and help to bring the project to the present form.

Thanks to our Departmental Project coordinators. We also thank all our friends, teaching and non-teaching staff at NMIT, Bangalore, for all the direct and indirect help provided in the completion of the project.

Table of Contents

Chapter-1

INTRODUCTION

A Good Database Has a Unique Identifier, Known as The Primary Key. As The Data Size Grows Bigger, The Time Taken to Retrieve and Search Query Are Also Increasing. There Are Several Techniques in Shortening the Time Taken to Retrieve a Query Such as Indexing Techniques.

To Increase the Performance of The Model, It Needs Indexing Techniques to Increase the Performance of Queries and To Retrieve the Results Fast. In A Nutshell, Indexing Is a Significant Technique for Managing Larger Databases. If The Right Index Structures Are Built on Columns, The Performance of Queries, Especially Ad Hoc Queries Will Be Greatly Enhanced. The Most Well-Known Indexing Technique Is the Tree Index.

Trees Are Most Commonly Used to Manage and Organize Large Databases Which Support Record Insertion, Deletion, And Key Range Searches. B-Trees Are the Most Widely Used Indexing Method for Large Disk-Based Databases and For Implementing File Systems.

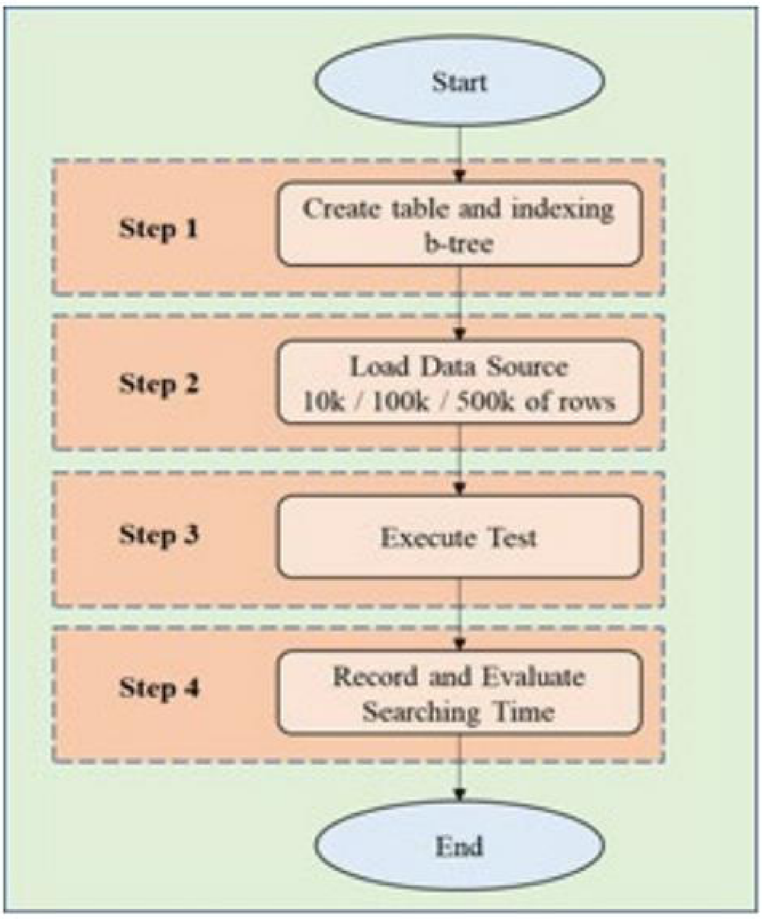
Instead Of Only Fetching and Comparing a Single Index Key Per Search Step Like in Binary Search, B-Tree Search Compares and Fetches Blocks of Multiple Index Keys Per Search Step with The Motivation of Reducing Hard Disk Seeks.

The Idea of Grouping Multiple Items into Nodes Was Originally Used to Reduce the Number of Hard Disk Seeks Per Index Lookup in Traditional Database Systems. It Can, However, Be Used in Main Memory Databases to Reduce the Number of Page Misses for An Index Lookup as Well.

Chapter-2

METHODOLDOGY

Instead Of Only Fetching and Comparing a Single Index Key Per Search Step Like in Binary Search, B-Tree Search Compares and Fetches Blocks of Multiple Index Keys Per Search Step with The Motivation Of Reducing Hard Disk Seeks. The Idea of Grouping Multiple Items into Nodes Was Originally Used to Reduce the Number of Hard Disk Seeks Per Index Lookup in Traditional Database Systems. It Can, However, Be Used in Main Memory Databases to Reduce the Number of Page Misses for An Index Lookup as Well.


Start
Step 1
Create table and indexing b-tree
Step 2
Load Data Source 10k / 100k / 500k of rows
Step 3
Execute Test
Step 4
Record and Evaluate Searching Time
End

FLOWCHART OF B-TREE INDEXING EVALUATION

This section shows the implementation of the B-Tree indexing technique. The technique then is implemented on PostgreSQL. Moreover, the process for the implementation will be explained and showed in detail.

Starting from how to create a database, create an index, load data into the database, query and get the result, and the end of the process that is making a comparative analysis based on the result.

The example of research process where all the process is shown in Figure 1, where an overall overview of the research is presented and described. The result of test based on speeds that are estimated based upon a mathematical model of computation, or extrapolated from other experiments, will be clearly identified. The metric that will be using is in millisecond of each the process.

After that, the result was made between both techniques based on query execution time. Both these techniques then will be implemented on different type of RDMS which is on PostgreSQL and MySQL. Next section will discussion

A. Step 1: Create table & indexing.

B. Step 2: Load data.

C. Step 3: Execute test and Finally

D. Step 4: Evaluating the result of the outcomes.

Chapter-3

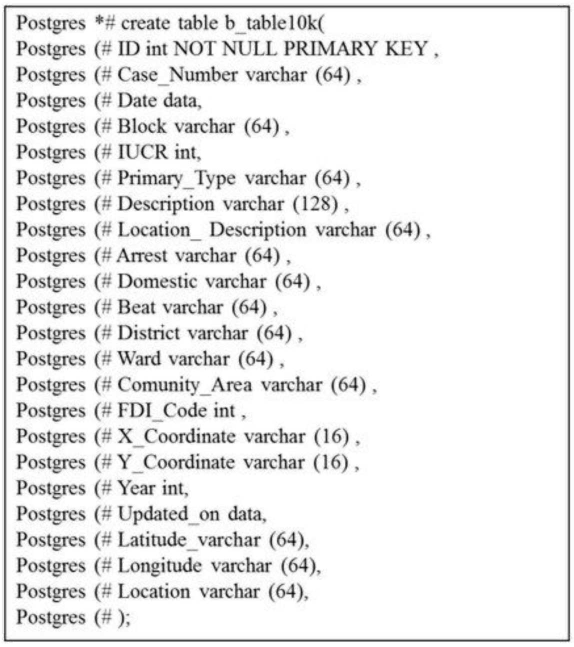
CREATE TABLE AND INDEXING DATABASE

This Section Explains How to Create a Table And B-Tree Index. A Table with Twenty-Two (22) Columns Are Created Using SQL Statement. If There Is No Indexing Created, PostgreSQL Will Automatically Assign a Binary Indexing Technique Once the Table Is Created. The Primary Key Is Identified as ID And Set to NOT NULL. Once A Table Is Created, Indexing Can Be Implemented by Using SQL

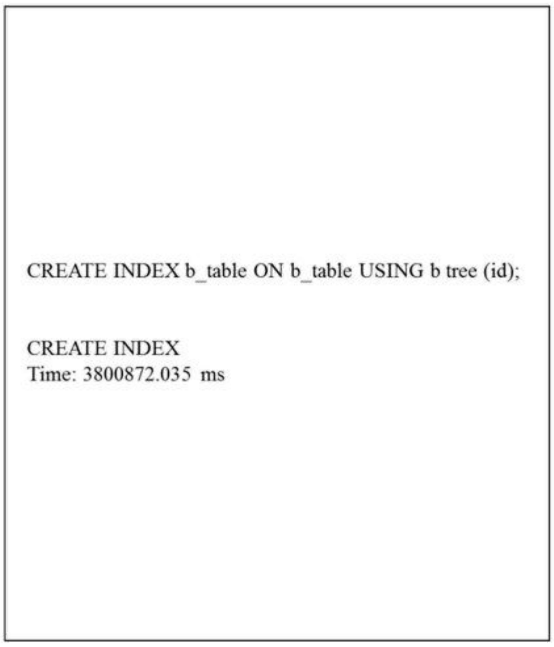
Statement Syntax

CREATE-INDEX-INDEX_NAME-ON?
TABLE_NAME-USING-INDEX_TYPES.

It Shows an Example of How to Create A Simple Index For PostgreSQL Platform Using SQL Syntax. Then, Each Table on What the Indexing Techniques Are Used Using \D TABLE_NAME Is Checked. Next Section Will Discuss on How to Load The Data into The PostgreSQL Database.



```
Postgres *# create table b_table10k(
Postgres (# ID int NOT NULL PRIMARY KEY ,
Postgres (# Case_Number varchar (64) ,
Postgres (# Date data,
Postgres (# Block varchar (64) ,
Postgres (# IUCR int,
Postgres (# Primary_Type varchar (64) ,
Postgres (# Description varchar (128) ,
Postgres (# Location_ Description varchar (64) ,
Postgres (# Arrest varchar (64) ,
Postgres (# Domestic varchar (64) ,
Postgres (# Beat varchar (64) ,
Postgres (# District varchar (64) ,
Postgres (# Ward varchar (64) ,
Postgres (# Comunity_Area varchar (64) ,
Postgres (# FDI_Code int ,
Postgres (# X_Coordinate varchar (16) ,
Postgres (# Y_Coordinate varchar (16) ,
Postgres (# Year int,
Postgres (# Updated_on data,
Postgres (# Latitude_varchar (64),
Postgres (# Longitude varchar (64),
Postgres (# Location varchar (64),
Postgres (# );
```



```
CREATE INDEX b_table ON b_table USING b tree (id);


CREATE INDEX
Time: 3800872.035 ms
```

Fig. 4 - Creation Table Using SQL Statement

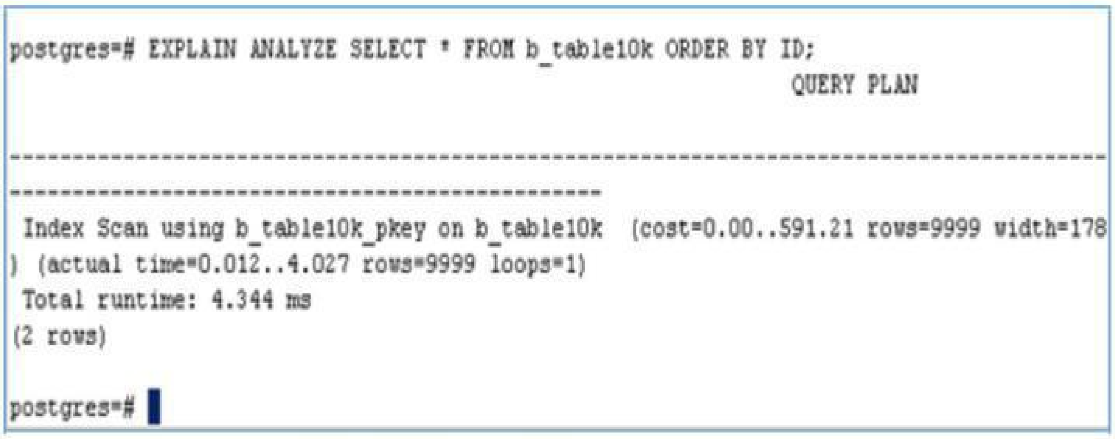
Fig. 5 - Indexing Creation SQL Syntax

Chapter-4

TESTING

For execution, each of the tests must be implemented using B-Tree.

Every query on each table must be scanned. By using simple SELECT statement of SQL it can identify every parameter including the cost, speed, and row on each record. Figure 7 shows the example of PostgreSQL query scan and all its details.



```
postgres=# EXPLAIN ANALYZE SELECT * FROM b_table10k ORDER BY ID;
                                                              QUERY PLAN

-------------------------------------------------------------------------------------
----------------------------------------------
 Index Scan using b_table10k_pkey on b_table10k  (cost=0.00..591.21 rows=9999 width=178
) (actual time=0.012..4.027 rows=9999 loops=1)
 Total runtime: 4.344 ms
(2 rows)

postgres=#
```

Fig. 7- PostgreSQL query scan using select

Chapter-5

EVALUATING RESULT AND DISCUSSION

According to the analysis and result from Step 3, the query execution time was evaluated and projected in Table 1, where the results are compiled into tabular form. From the table, clearly shown that B-Tree time retrieval of query takes 1.851 milliseconds (ms) for 10K of rows. Then, for 100K of rows the time is 346.082 ms. Next, for 500K of rows the time taken is 1268.114 ms. This justifies from that the larger the data is the slower the retrieval of the query time. Moreover, the result of this research shows that indexing techniques are significant in term of performance

Table 2 - Summary of B-Tree analysis result

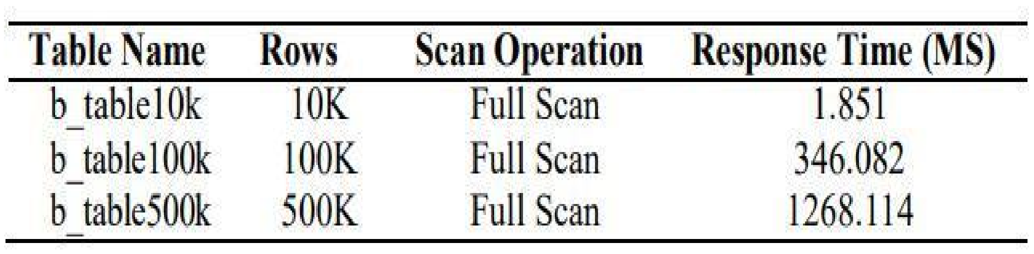


Table Name	Rows	Scan Operation	Response Time (MS)
b_table10k	10K	Full Scan	1.851
b_table100k	100K	Full Scan	346.082
b_table500k	500K	Full Scan	1268.114

CONCLUSION

For conclusion, this research had discussed the works of the indexing techniques with their algorithm. Through the experiment, this research implemented B-Tree index for PostgreSQL database.

By analyzing the result of the outcomes, shows that B-Tree index is significant in database system in term of performance. This research recommends further investigation such as an improved algorithm from existing B-Tree where B-Tree prone to have slower time query as data grows larger.

Hopefully, this case study may be useful for future research in developing new indexing techniques which will add new techniques together with existing techniques to increase the performance of the data query

References

1. https://www.researchgate.net/publication/342832268_A_Case_Study_on_B-Tree_Database_Indexing_Technique
2. https://dzone.com/articles/database-btree-indexing-in-sqlite
3. https://www.geeksforgeeks.org/introduction-of-b-tree-2/