

REPORT ON

“Players’ Information Systems”

Prepared by

SHASHADHAR DAS (2111CS14)

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY
PATNA**

ABSTRACT

We often use various database engines but fail to know how they actually store the data inside tables or any other form.

This assignment acknowledges us how the data is actually stored and how it is retrieved.

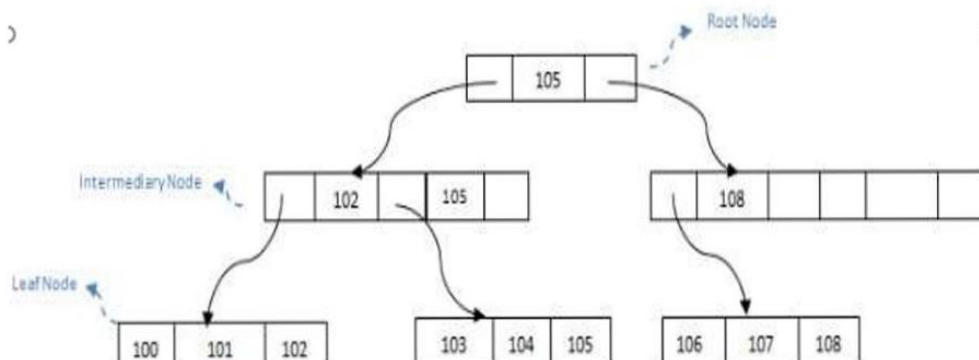
The search in a Database is done through B+ Tree Indexing or Hash based Indexing. Our system is based on B+ Tree Indexing for data storage and retrieval. B+ tree Indexing makes the search efficient.

Another feature of our system is that it read data from and writes data on disk in blocks, which enhances the security of system and also make the access of data easier.

Introduction

Consider the Player Information below. This can be stored in B+ tree structure as shown below. We can observe here that it divides the records into two and splits into left node and right node. Left node will have all the values less than or equal to root node and the right node will have values greater than root node. The intermediary nodes at level 2 will have only the pointers to the leaf nodes. The values shown in the intermediary nodes are only the pointers to next level. All the leaf nodes will have the actual records in a sorted order.

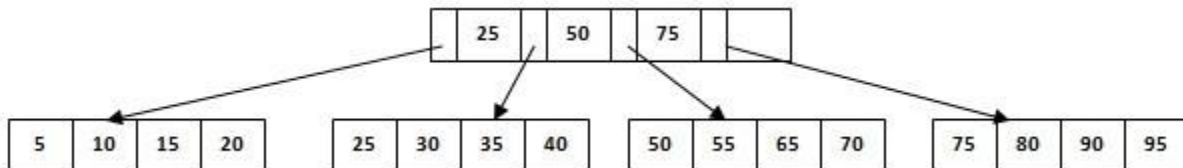
1	Id	Name	BirthYear	MatchesPlayed	RunsScored	WicketsTaken	CatchesTaken
2							
3	100	Das	1999	56	200	67	87
4	101	Shashadhar	2016	23	100	89	45
5	102	Virat	1987	32	345	90	34
6	103	Rohit	1987	43	50	67	32
7	104	Pant	2000	44	66	34	23
8	105	Rahul	2016	12	34	56	24
9							



We have stored all the index tress in B+ tree as it will make search and retrieval of the information efficient.

Searching of Record

Suppose we want to search player id 65 in the below B+ tree index structure. First we will fetch for the intermediary node which will direct to the leaf node that can contain record for 65. So we find branch between 50 and 75 nodes in the intermediary node. Then we will be redirected to the third leaf node at the end. Here DBMS will perform sequential search to find 65.



We have created the index file for each of the search field and stored in the file. The B+ tree is loaded when the program is run. When we search by the player Id, the index tree gives the id and by which we load the data file. The search time complexity is $\log n$.

```
===== PLAYER Information System=====
1: Add a Player Record
2: Update A Player Record
3: Delete Player by playerId
4: Search Player using Player Id
5: Search Player using Player Name
6: List of player having runs scored more than and equal
7: List of player having wickets taken more than and equal
8: List of player having catches taken more than and equal
9: Exit from Program

Enter your choice...!
4
Enter the player id: 2

2      SD      456      34      23      34      21

===== PLAYER Information System=====
1: Add a Player Record
```

Insertion of Record

To insert a new record, the index tree is searched if we already have the player id added. If the player id is not added, the new data file is created and new player id index is added in the index tree. We are creating file for each record and the file pointer is stored in the index tree for retrieval. The time complexity for insertion is $O(\log n)$.

```
===== PLAYER Information System=====
1: Add a Player Record
2: Update A Player Record
3: Delete Player by playerId
4: Search Player using Player Id
5: Search Player using Player Name
6: List of player having runs scored more than and equal
7: List of player having wickets taken more than and equal
8: List of player having catches taken more than and equal
9: Exit from Program

Enter your choice...!
1
Enter player id :
23
Enter the name of player:
Das

Enter player birth year:
1987

Enter matches played:
456

Enter runs scored:
234

Enter wickets taken:
23
```

Deletion of Record

For deletion of record by player id , the player id index tree is searched to get the id. If id is available in the index tree , the file pointer is retrieved and deleted. The index tree also updated.

```
1: Add a Player Record
2: Update A Player Record
3: Delete Player by PlayerId
4: Search Player using Player Id
5: Search Player using Player Name
6: List of player having runs scored more than and equal
7: List of player having wickets taken more than and equal
8: List of player having catches taken more than and equal
9: Exit from Program

Enter your choice...!
3
Enter the palyer id: 12
Deleted record:










12      Das      2345      23      12      234      234

===== PLAYER Information System=====
1: Add a Player Record
```

Our system

We are creating the database folder name “PlayerDatabase” and the records are stored as a data file inside the folder. The file pointer is stored in the index file for retrieval. Keeping the files separately makes it easy to delete, insert and search a record.

The database folder and files are created when the program is run for the first time.

 2.dat	2/13/2022 2:05 AM	DAT File	1 KB
 7.dat	2/13/2022 2:22 AM	DAT File	1 KB
 23.dat	2/13/2022 4:35 AM	DAT File	1 KB
 34.dat	2/13/2022 2:14 AM	DAT File	1 KB
 45.dat	2/12/2022 10:48 PM	DAT File	1 KB
 playerCatchesIndex.dat	2/13/2022 2:14 AM	DAT File	1 KB
 playerIdIndex.dat	2/13/2022 2:14 AM	DAT File	1 KB
 playerRunsScoredIndex.dat	2/13/2022 2:14 AM	DAT File	1 KB
 playerWicketsIndex.dat	2/13/2022 2:14 AM	DAT File	1 KB

Analysis and Conclusion

From the assignment, we get acknowledged about how data is actually stored inside the table, how Indexing is done, how file pointers are created.

Also, B+ Tree Indexing makes search efficient for huge amount of data which cannot be loaded into primary memory at the time of execution of program so simultaneously we require writing data back to secondary storage.

There is another approach for Indexing called Hash based Indexing in which search is faster but much time delay is generated for handling collisions.