09/12/22. ↳ Index Files are typically much smaller than the original File.

Index Evaluation Metrics :-

time complexity
Hash Map — O(1)
B+ (or) B-Tree — O(log n)

1) Execution time of Data Retrieval.
2) Creation Time of Index.
3) Size of the Index should be less than Data File.
4) Time taken to insert, delete and updating time of Index.

It doesn't have to be unique

Also called clustering index

Primary Index :- ↳ When we create a Primary key, Primary Index is automatically created.
→ Primary Index decides the order the data is stored (sequential) in the disk.
↳ If there are two Primary keys, then only one Primary Index will be created; because we can only have one global sorted order.

It is of two types;
1) Dense &
2) Sparse.

Secondary Indices :- → It is not unique, it needs to be pointed to the primary Index.
→ It is used as a Index Scan.
→ It is of only one type.
i.e., Dense.

09/14/22.

→ Secondary Index doesn't have to be unique.
→ The Entries of the Secondary Index is sorted.
→ Why not create secondary Index on all the Attributes?
Ans) It depends on the application and the database.

→

Multilevel Index:-

→ If the primary Index is large for the main memory then we need to create a Sparse Index.

and even if the Sparse Index is large for the main memory, we need to create another Sparse Index on top of the sparse (previously built)

B^+ - Tree :-

→ It can be both Primary Index (or) a Secondary Index.

→ No deep is good (or) no bushy level tree is good.

→ All the Trade offs can be answered by doing Experiments.

09/19/22.

→ what is the main problem in a B^+ - Tree, when we are inserting data?

Ans) Insertion overhead.

→ why did they create B^+ - Tree?

Ans) To solve the degradation problem and to speed up queries while making sure the insertion & deletion are constantly taking place.

09/21/22 → How did they make insertion & deletion faster in B^+ - Tree?

Ans) They are good in handling insertion & updation, because of the free space they have, they can insert into the free space and hence it increases the fastness, same as for deletion.

Pro's and Con's of B-Tree's and B^+-Tree's:-

↳ B-Tree is faster because it can access the data faster since it has no duplicate records (according to the pic on slides).

↳ The Time and Space complexity is better for B-Tree.

↳ when we do sequential scan or search and no record is found in the B-Tree then we need to go back to the Root node and start again. Hence, it needs extra operation.

↳ Non-leaf nodes are larger, so fan-out is reduced.

↳ Insertion & Deletion is more complicated than B^+-Tree's.

↳ B-Tree's have deeper Trees than B^+-Tree's.
It is a bad case for the B-Trees because the time complexity increases, i.e., ($> O(\log n)$).

09/26/22

Bit map Indices:

→ we'll manually create a range and then assign the range's order this is called Binning.

→ Bitmap is useful when the no. of values for that row are very low.

→ Insertion & updating a Bit map is annoying.

→ Bitmap's are highly compressable.

Existance Bitmap:- →It is used to let the Bitmap indices know that a row has been deleted.

(used for Deleting rows in Bitmap.)

Ex:- original Bitmap : 1 0 0 1 0

Existance Bitmap: 1 1 1 1 0

Do an AND operation: 1 0 0 0 0.

→ Bitmaps are beneficial for column storing.

Hashing:-

→ We'll have bucket overflow, if our Hash Function is very bad and/or by having same record numbers.

To overcome this overflow, we use overflow buckets also called closed hashing

→ In order to create a Good hash Function, we create our buckets based on the data.

Extendable Hashing:-

→ Minimal space overhead.

→ Even if data grows, performance doesn't degrade.

B-Trees (or) B+-Tree Vs Hash map:-

↳ Point queries (a particular query of what we want) then Hash index is better.

↳ A query for a Range query (where the instructors salary ranges) then B+-Trees are better.

↳ Hash maps are useless for Range queries. (very slow).

→ Dis-Advantages of Extendable Hashing:

→ Additional checking is required in order to know which bucket the data belongs to and if the data belongs to it (or) not, hence time to data retrieval increases.