UNIT V – Physical Storage and MongoDB

Data Storage and Indexes – RAID- File organization-Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing. Query Processing Overview-Algorithms for Selection and Sorting Basics of MongoDB, Procedural Language

Static Hashing and Dynamic Hashing

Hashing is an efficient technique to directly search the location of desired data on the disk without using an index structure. Data is stored at the data blocks whose address is generated by using a hash function. The memory location where these records are stored is called a data block or data bucket.

Hash File Organization

- Data bucket – Data buckets are the memory locations where the records are stored. These buckets are also considered Units of Storage.
- Hash Function – The hash function is a mapping function that maps all the sets of search keys to the actual record address. Generally, the hash function uses the primary key to generate the hash index – the address of the data block. The hash function can be a simple mathematical function to any complex mathematical function.
- Hash Index-The prefix of an entire hash value is taken as a hash index. Every hash index has a depth value to signify how many bits are used for computing a hash function. These bits can address 2n buckets. When all these bits are consumed? then the depth value is increased linearly and twice the buckets are allocated.

Static Hashing

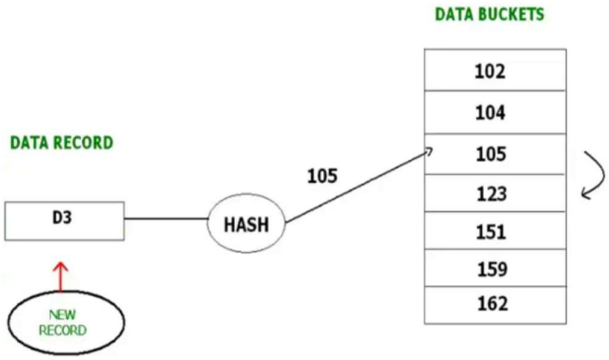
In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if we want to generate an address for STUDENT_ID = 104 using a mod (5) hash function, it always results in the same bucket address 4. There will not be any changes to the bucket address here. Hence a number of data buckets in the memory for this static hashing remain constant throughout.

Operations:

- Insertion – When a new record is inserted into the table, The hash function h generates a bucket address for the new record based on its hash key K. Bucket address = h(K)
- Searching – When a record needs to be searched, The same hash function is used to retrieve the bucket address for the record. For Example, if we want to retrieve the whole record for ID 104, and if the hash function is mod (5) on that ID, the bucket address generated would be 4. Then we will directly got to address 4 and retrieve the whole record for ID 104. Here ID acts as a hash key.
- Deletion – If we want to delete a record, Using the hash function we will first fetch the record which is supposed to be deleted. Then we will remove the records for that address in memory.
- Updation – The data record that needs to be updated is first searched using the hash function, and then the data record is updated.

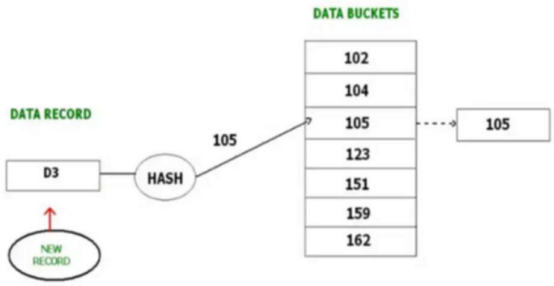
Some commonly used methods are discussed below:

- **Open Hashing** – In the Open hashing method, the next available data block is used to enter the new record, instead of overwriting the older one. This method is also called linear probing. For example, D3 is a new record that needs to be inserted, the hash function generates the address as 105. But it is already full. So the system searches the next available data bucket, 123, and assigns D3 to it.



- **Closed hashing** – In the Closed hashing method, a new data bucket is allocated with the same address and is linked to it after the full data bucket. This method is also known as overflow chaining. For example, we have to insert a new record D3 into the tables. The static hash function generates the data bucket address as 105. But this bucket is full to store the new data. In this case, a new

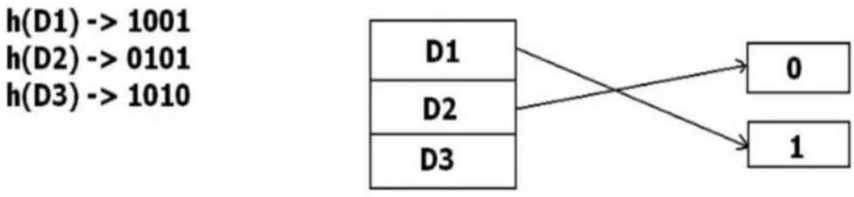
data bucket is added at the end of the 105 data bucket and is linked to it. The new record D3 is inserted into the new bucket.



- **Quadratic probing:** Quadratic probing is very much similar to open hashing or linear probing. Here, The only difference between old and new buckets is linear. The quadratic function is used to determine the new bucket address.
- **Double Hashing:** Double Hashing is another method similar to linear probing. Here the difference is fixed as in linear probing, but this fixed difference is calculated by using another hash function. That's why the name is double hashing.

Dynamic Hashing

The drawback of static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks. In Dynamic hashing, data buckets grow or shrink (added or removed dynamically) as the records increase or decrease. Dynamic hashing is also known as extended hashing. In dynamic hashing, the hash function is made to produce a large number of values. For Example, there are three data records D1, D2, and D3. The hash function generates three addresses 1001, 0101, and 1010 respectively. This method of storing considers only part of this address – especially only the first bit to store the data. So it tries to load three of them at addresses 0 and 1.



But the problem is that No bucket address is remaining for D3. The bucket has to grow dynamically to accommodate D3. So it changes the address to have 2 bits rather

than 1 bit, and then it updates the existing data to have a 2-bit address. Then it tries to accommodate D3.

h(D1) -> 1001
h(D2) -> 0101
h(D3) -> 1010