DBMS Hashing

For a huge database structure it is not sometime feasible to search index through all its level and then reach the destination data block to retrieve the desired data. Hashing is an effective technique to calculate direct location of data record on the disk without using index structure.

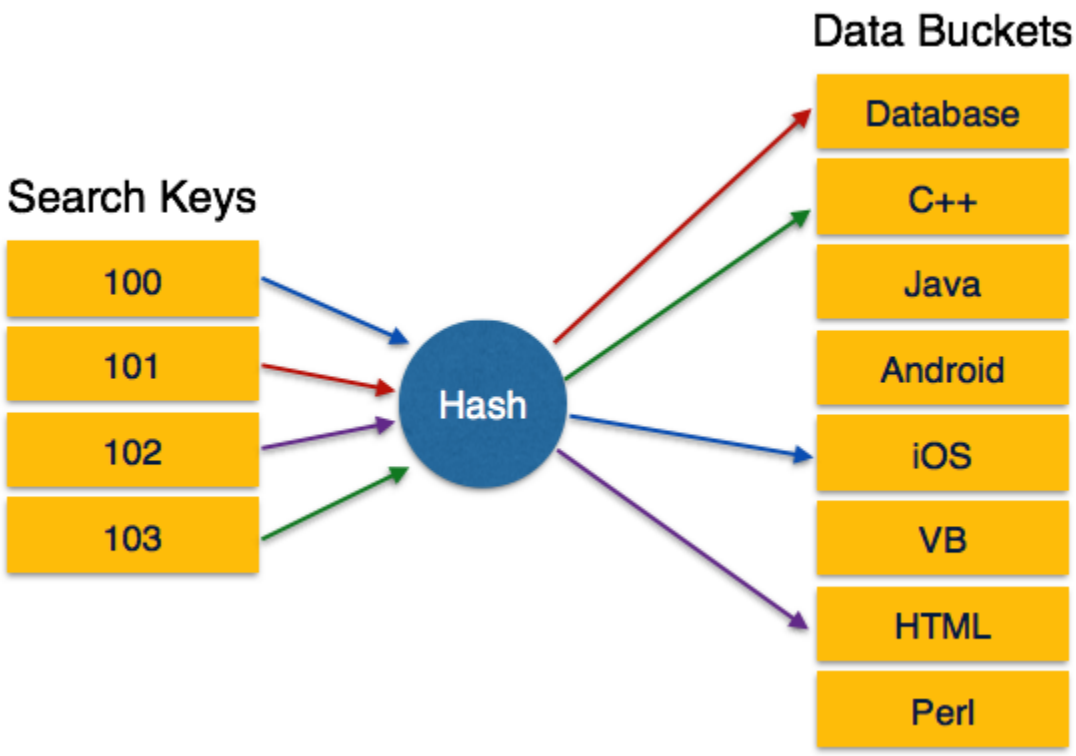
It uses a function, called hash function and generates address when called with search key as parameters. Hash function computes the location of desired data on the disk.

Hash Organization

- **Bucket:** Hash file stores data in bucket format. Bucket is considered a unit of storage. Bucket typically stores one complete disk block, which in turn can store one or more records.
- **Hash Function:** A hash function h, is a mapping function that maps all set of search-keys K to the address where actual records are placed. It is a function from search keys to bucket addresses.

Static Hashing

In static hashing, when a search-key value is provided the hash function always computes the same address. For example, if mod-4 hash function is used then it shall generate only 5 values. The output address shall always be same for that function. The numbers of buckets provided remain same at all times.



[*Image: Static Hashing*]

Operation:

- **Insertion:** When a record is required to be entered using static hash, the hash function h, computes the bucket address for search key K, where the record will be stored.

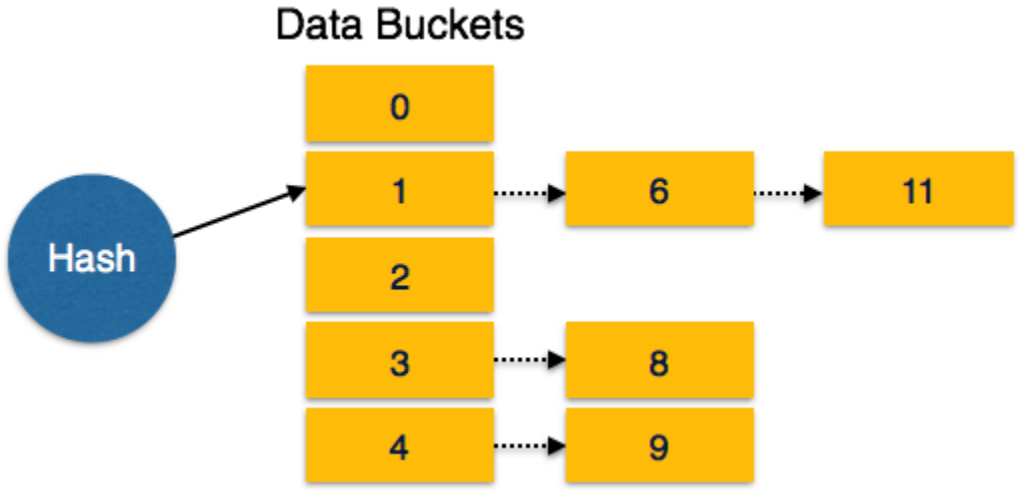
Bucket address = h(K)

- **Search:** When a record needs to be retrieved the same hash function can be used to retrieve the address of bucket where the data is stored.
- **Delete:** This is simply search followed by deletion operation.

BUCKET OVERFLOW:

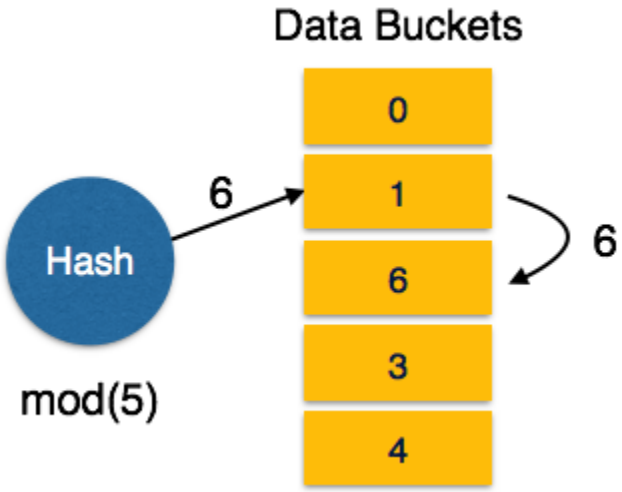
The condition of bucket-overflow is known as collision. This is a fatal state for any static hash function. In this case overflow chaining can be used.

- **Overflow Chaining:** When buckets are full, a new bucket is allocated for the same hash result and is linked after the previous one. This mechanism is called Closed Hashing.



[*Image: Overflow chaining*]

- **Linear Probing:** When hash function generates an address at which data is already stored, the next free bucket is allocated to it. This mechanism is called Open Hashing.



[*Image: Linear Probing*]

For a hash function to work efficiently and effectively the following must match:

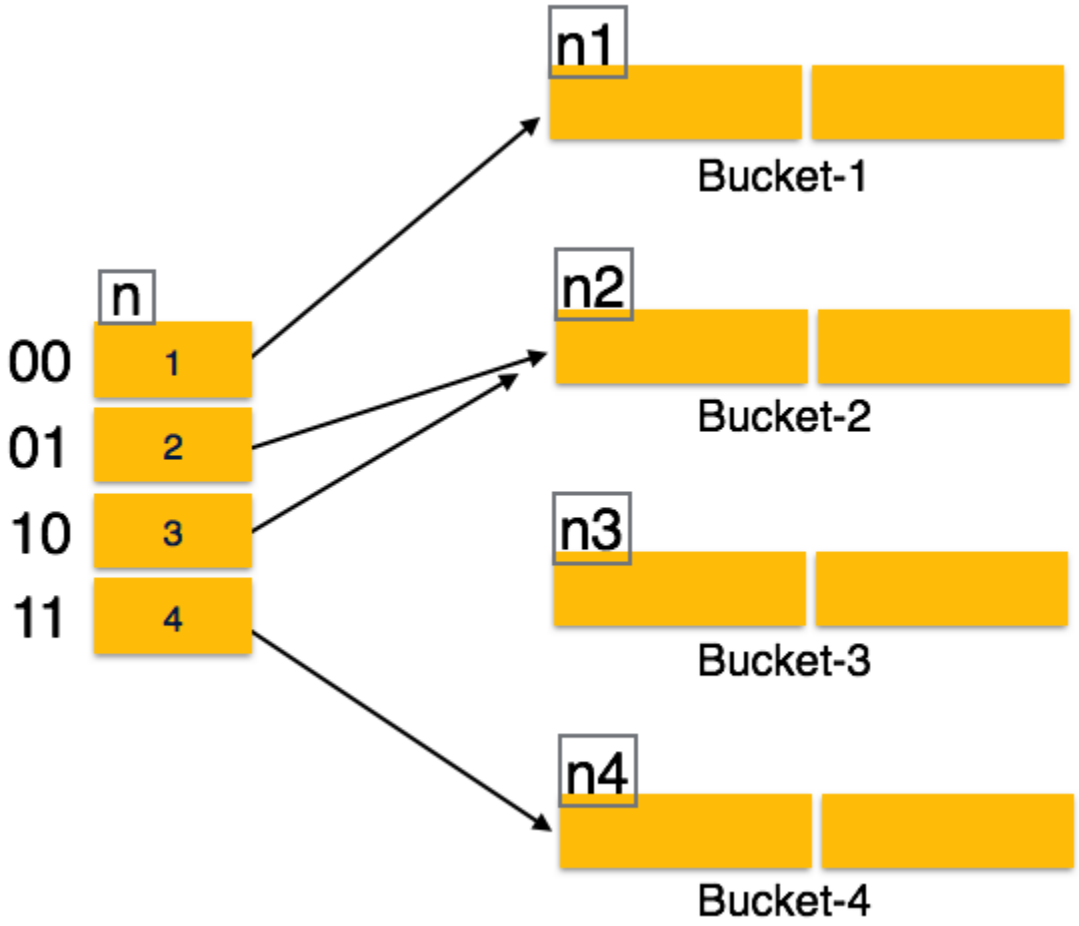
- Distribution of records should be uniform

- Distribution should be random instead of any ordering

Dynamic Hashing

Problem with static hashing is that it does not expand or shrink dynamically as the size of database grows or shrinks. Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand. Dynamic hashing is also known as extended hashing.

Hash function, in dynamic hashing, is made to produce large number of values and only a few are used initially.



[*Image: Dynamic Hashing*]

ORGANIZATION

The prefix of entire hash value is taken as hash index. Only a portion of hash value is used for computing bucket addresses. Every hash index has a depth value, which tells it how many bits are used for computing hash function. These bits are capable to address 2n buckets. When all these bits are consumed, that is, all buckets are full, then the depth value is increased linearly and twice the buckets are allocated.

OPERATION

- **Querying:** Look at the depth value of hash index and use those bits to compute the bucket address.
- **Update:** Perform a query as above and update data.

- **Deletion:** Perform a query to locate desired data and delete data.
- **Insertion:** compute the address of bucket

  - If the bucket is already full

    - Add more buckets

    - Add additional bit to hash value

    - Re-compute the hash function

  - Else

    - Add data to the bucket

  - If all buckets are full, perform the remedies of static hashing.

Hashing is not favorable when the data is organized in some ordering and queries require range of data. When data is discrete and random, hash performs the best.

Hashing algorithm and implementation have high complexity than indexing. All hash operations are done in constant time.

Source:

http://www.tutorialspoint.com/dbms/dbms_hashing.htm