SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

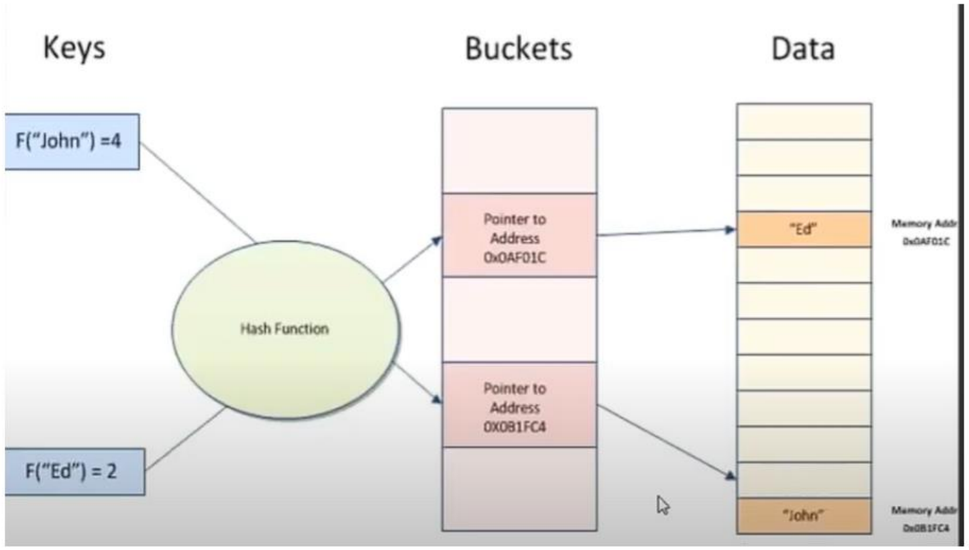
Department of Artificial Intelligence and Data Science

Course Name – 23ITB204-Modern Database Management Systems
II Year / III Semester

Topic - HASHING

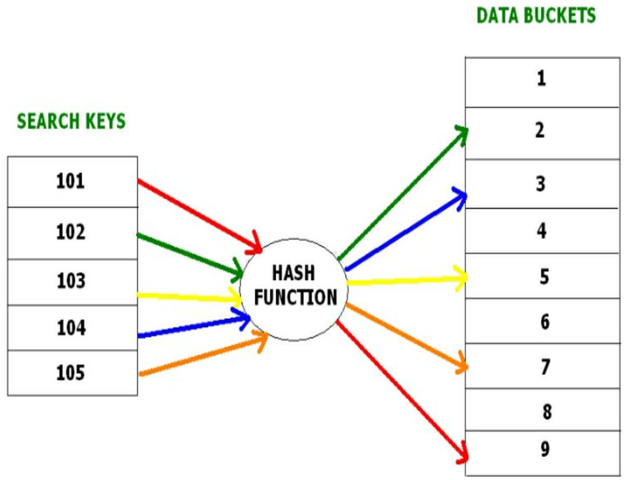
HASHING – WHY?

- Ordered Indexing
  - When we want to retrieve a particular data, It becomes very inefficient to search all the index values and reach the desired data.
  - For a huge database structure, it's tough to search all the index values through all its level and then you need to reach the destination data block to get the desired data.
  - Hashing technique comes into picture



HASHING - DEFINITION

- Hashing is an efficient technique to directly search the location of desired data on the disk without using index structure.
- Data is stored at the data blocks whose address is generated by using hash function.
- The memory location where these records are stored is called as data block or data bucket.
- used to index and retrieve items in a database as it is faster to search that specific item using the shorter hashed key instead of using its original value



HASHING – TERMINOLOGIES

Key

attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table)

Data Buckets

memory locations where the records are stored also known as Unit of Storage

Hash function

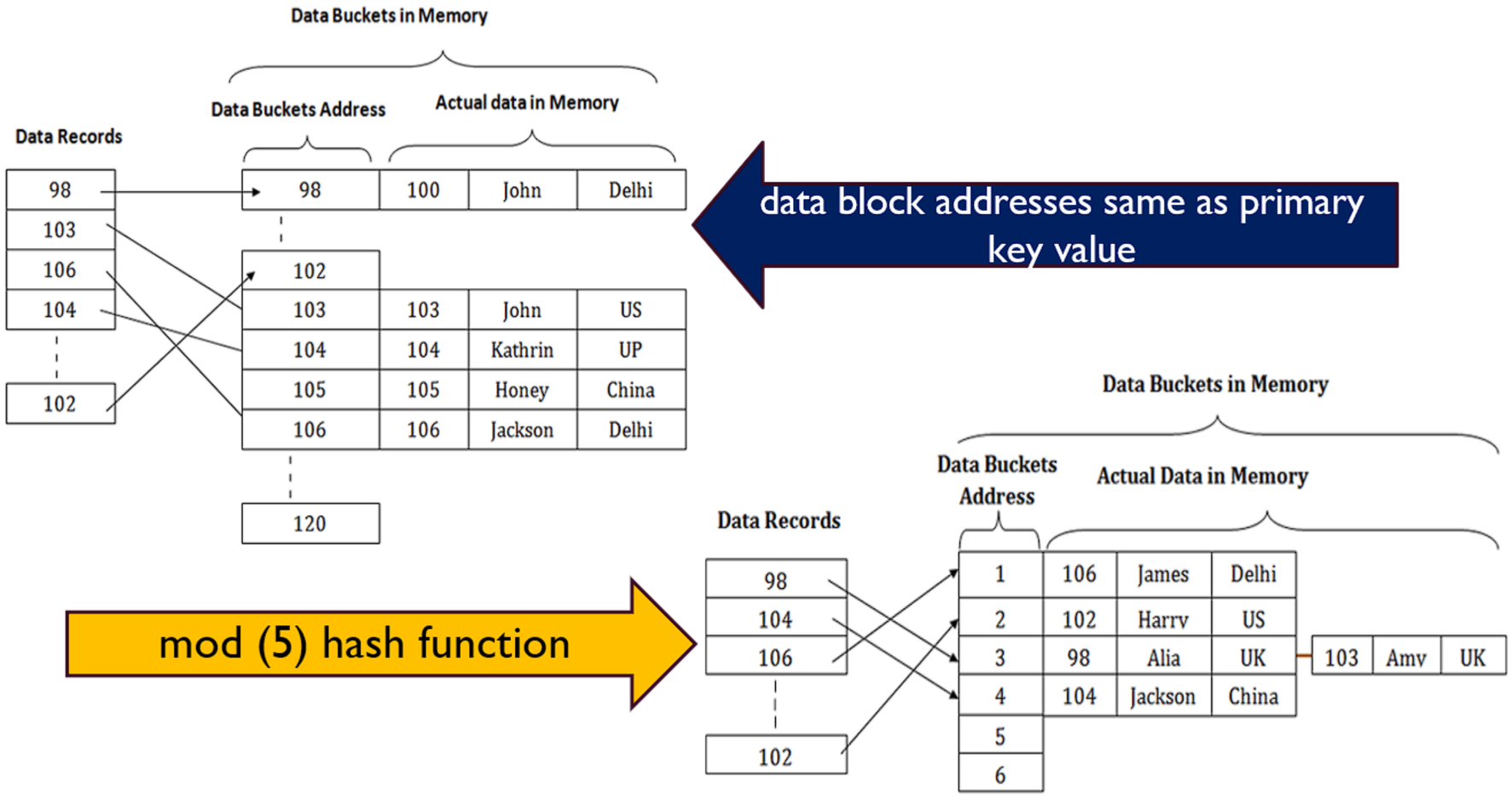
mapping function which maps all the set of search keys to the address where actual records are placed

Hash index

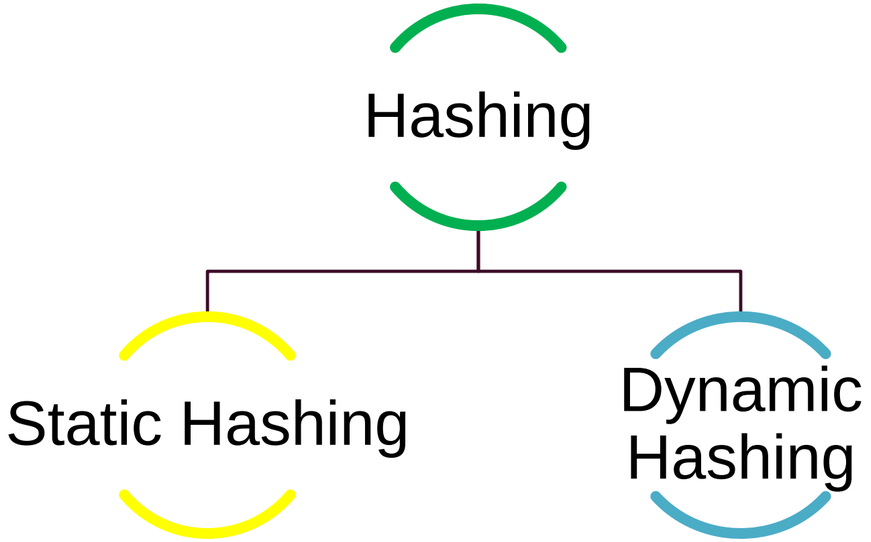
address of the data block

HASH FUNCTION

- Hash function is a mapping function that maps all the set of search keys to actual record address.
- Generally, hash function uses primary key to generate the hash index – address of the data block.
- Hash function can be simple mathematical function to any complex mathematical function.
- It is a function from search keys to bucket addresses.
- a hash function can choose any of the column value to generate the address(mostly primary key)
- This hash function can also be a simple mathematical function like exponential, mod, cos, sin

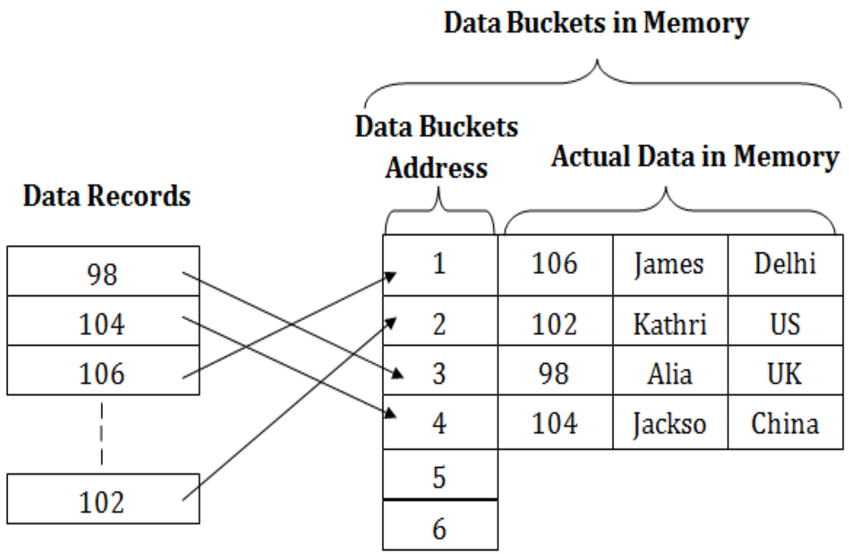

Data Buckets in Memory
Data Buckets Address
Actual data in Memory
Data Records
98
103
106
104
102
98
100
John
Delhi
102
103
103
John
US
104
104
Kathrin
UP
105
105
Honey
China
106
106
Jackson
Delhi
120
data block addresses same as primary key value
mod (5) hash function
Data Buckets in Memory
Data Buckets Address
Actual Data in Memory
Data Records
98
104
106
102
1
106
James
Delhi
2
102
Harry
US
3
98
Alia
UK
4
104
Jackson
China
5
6
103
Amy
UK

HASHING - TYPES



STATIC HASHING

- In the static hashing, the resultant data bucket address will always remain the same.
- the number of data buckets in memory always remains constant.



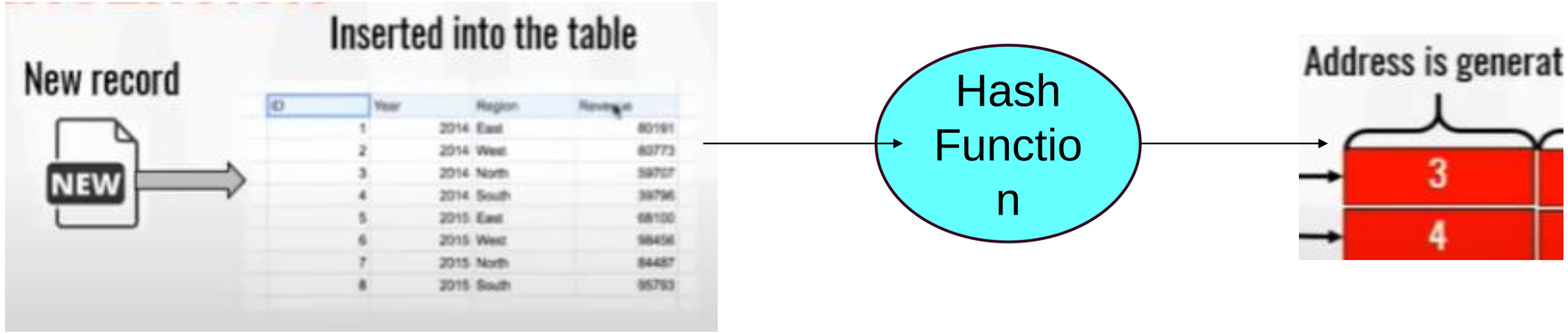
if we generate an address for EMP_ID =103 using the hash function mod (5) then it will always result in same bucket address 3
If I fix buckets as 5,you cannot go beyond 5

number of data buckets in memory remains constant throughout

STATIC OPERATIONS

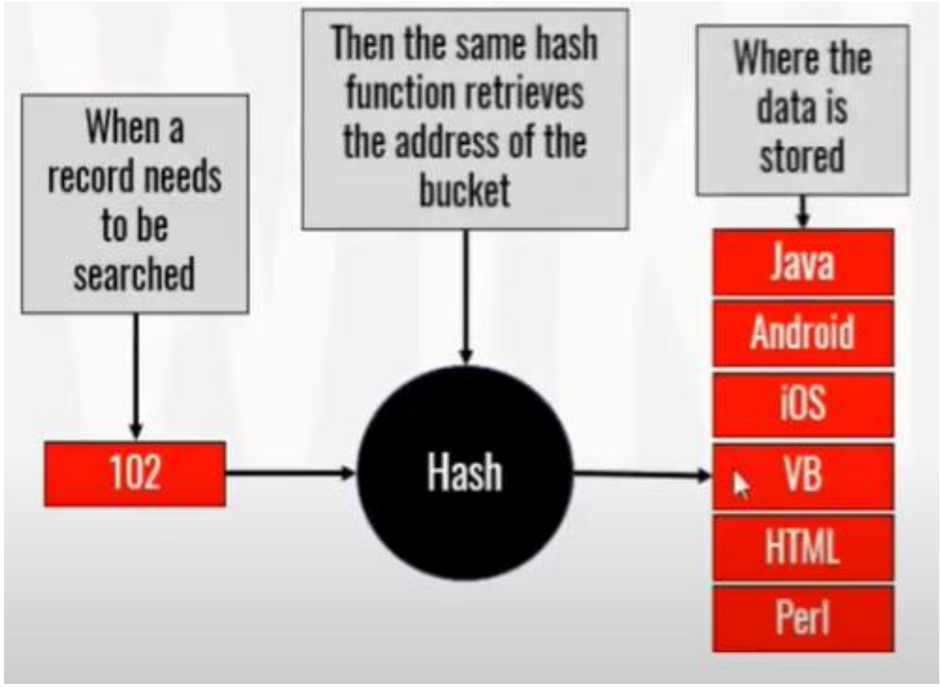
INSERTION

- When a new record requires to be inserted into the table, you can generate an address for the new record using its hash key. When the address is generated, the record is automatically stored in that location.



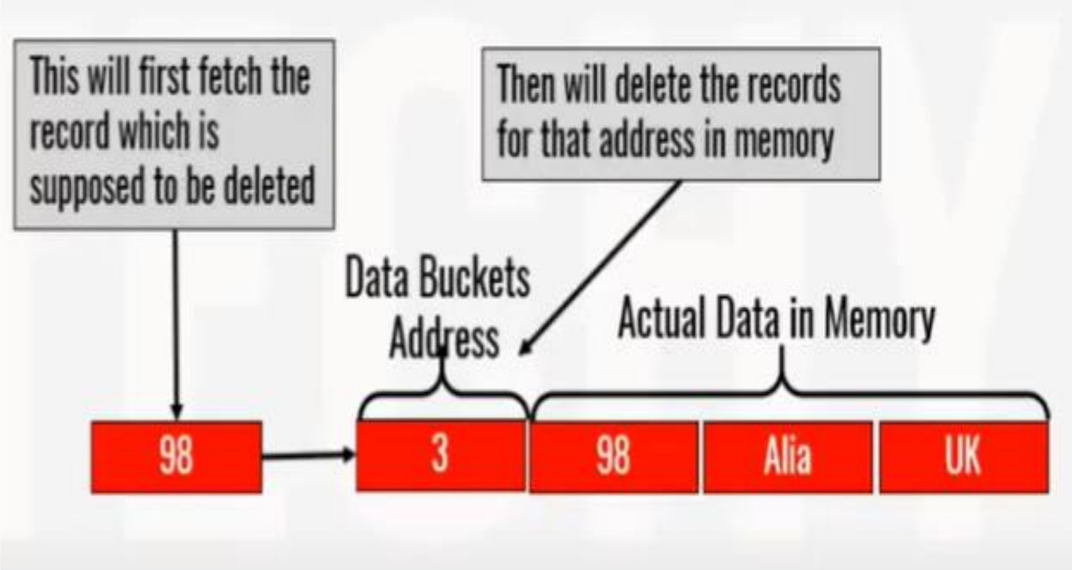
SEARCHING

- When you need to retrieve the record, the same hash function should be helpful to retrieve the address of the bucket where data should be stored.

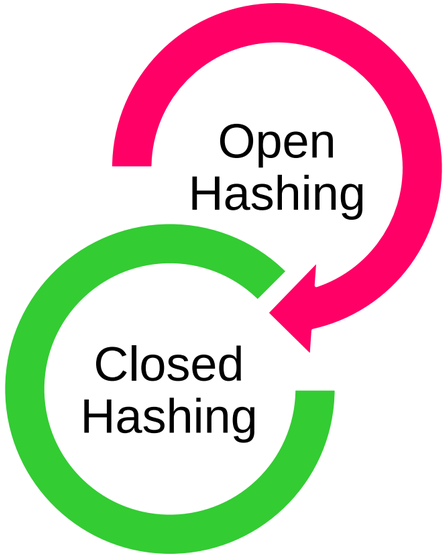


DELETION

- Using the hash function, you can first fetch the record which is you wants to delete. Then you can remove the records for that address in memory.

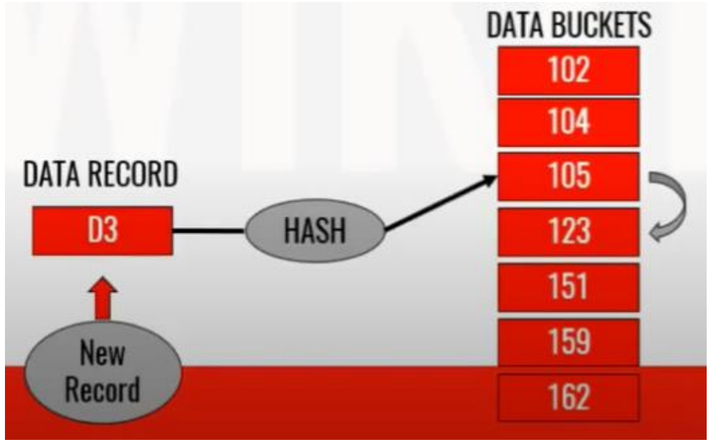


STATIC HASHING - TYPES



STATIC HASHING – OPEN HASHING

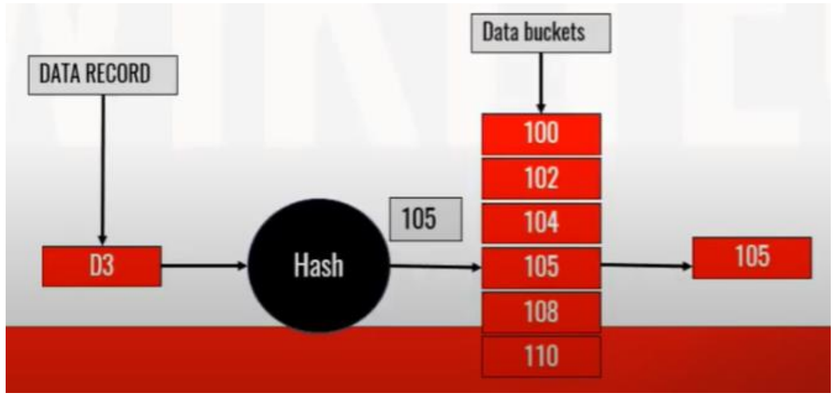
- In Open hashing method, Instead of overwriting older one the next available data block is used to enter the new record.
- This method is also known as linear probing.



For Example,
D3 is a new record which needs to be inserted, the hash function generates address as 105. But it is already full. So the system searches next available data bucket, 123 and assigns D3 to it

STATIC HASHING – CLOSE HASHING

- In the close hashing method, when buckets are full, a new bucket is allocated for the same hash and result are linked after the previous one.
- In Closed hashing method, a new data bucket is allocated with same address and is linked it after the full data bucket. This method is also known as overflow chaining.



For example,
D3 have to be inserted. Hash function generates address which is 105. but 105 has data already. So it will create a new bucket in same data bucket 105 and links this to previous 105 data bucket

STATIC HASHING – DRAWBACKS

- The drawback of static hashing is that that it does not expand or shrink dynamically as the size of the database grows or shrinks

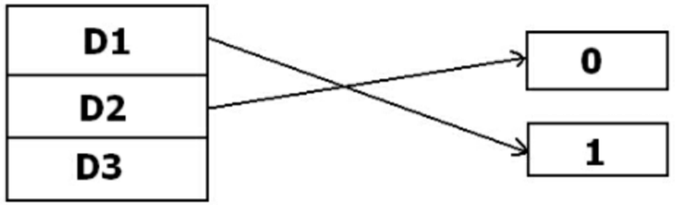
DYNAMIC HASHING

- In Dynamic hashing, data buckets grows or shrinks (added or removed dynamically) as the records increases or decreases. 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,

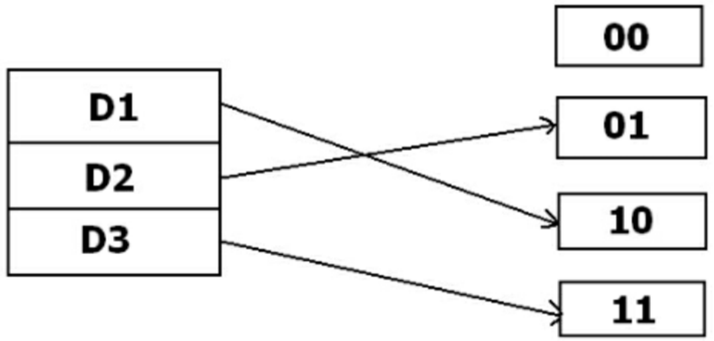
there are three data records D1, D2 and D3 .
The hash function generates three addresses 1001, 0101 and 1010 respectively.
This method of storing considers only part of this address –
especially only first one bit to store the data. So it tries to load three of them at address 0 and 1.

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

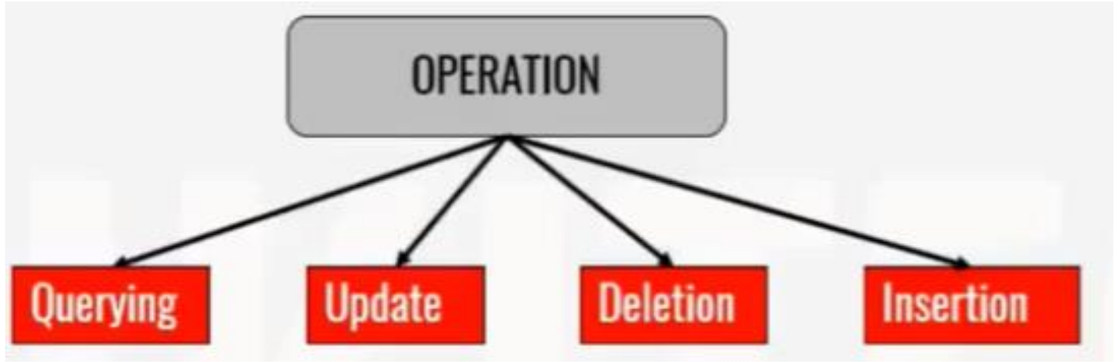


But the problem is that No bucket address is remaining for D3.
The bucket has to grow dynamically to accommodate D3.
So it changes the address have 2 bits rather than 1 bit, and then it updates the existing data to have 2 bit address.
Then it tries to accommodate D3.

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



DYNAMIC HASHING - OPERATIONS



QUERYING	Look at the depth value of the hash index and use those bits to compute the bucket address.
UPDATE	Perform a query and update the data.
DELETION	Perform a query to locate the desired data and delete the same.

INSERTION

Compute the address of the bucket

- If the bucket is already full.
  - Add more buckets.
  - Add additional bits to the hash value.
  - Re-compute the hash function.
- Else
  - Add data to the bucket,
- If all the buckets are full, perform the remedies of static hashing.

Thank you