Hashing

4.2

hashing

- Hashing technique is used to calculate the direct location of a data record on the disk without using index structure.
- In this technique, data is stored at the data blocks whose address is generated by using the hashing function.
- The memory location where these records are stored is known as data bucket or data blocks.
- In this, a hash function can choose any of the column value to generate the address.
- Most of the time, the hash function uses the primary key to generate the address of the data block.
- A hash function is a simple mathematical function to [redacted]hematical function

Why do we need Hashing?

- 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 method is 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 is an ideal method to calculate the direct location of a data record on the disk without using index structure.
- It is also a helpful technique for implementing dictionaries.

Terminologies in Hashing

- Data bucket –
- Data buckets are memory locations where the records are stored. It is also known as Unit Of Storage. It refers to a specific slot or location within a hash table where data elements are stored.
-
- Hash function: A hash function, is a mapping function which maps all the set of search keys to the address where actual records are placed.

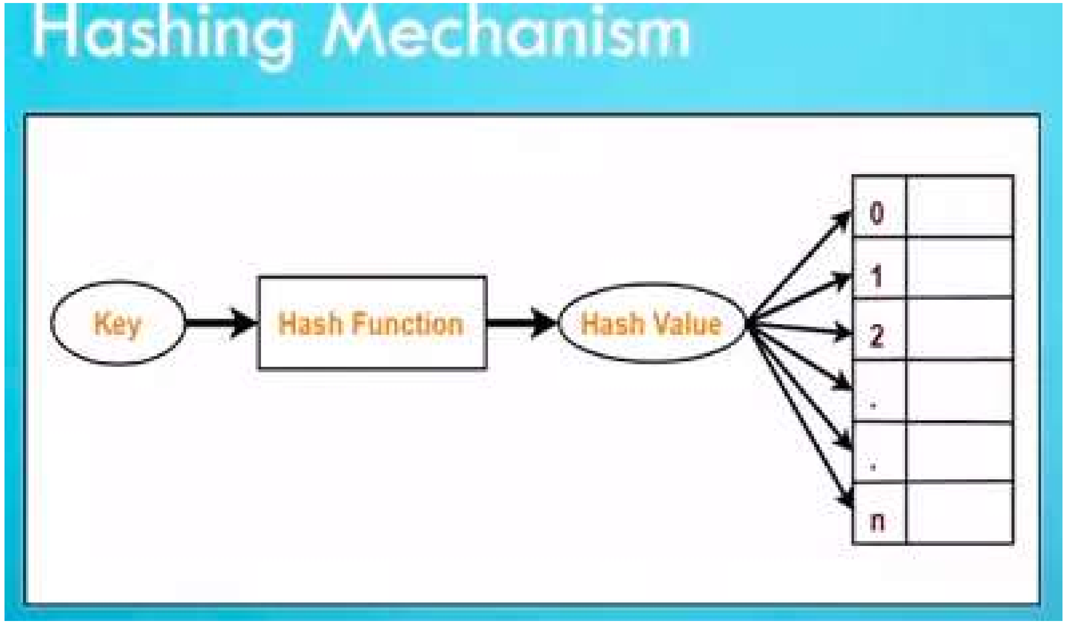
- Linear Probing – Linear probing is a fixed interval between probes.
- In this method, the next available data block is used to enter the new record, instead of overwriting on the older record.

- Hash index – It is an address of the data block. A hash function could be a simple mathematical function to even a complex mathematical function.

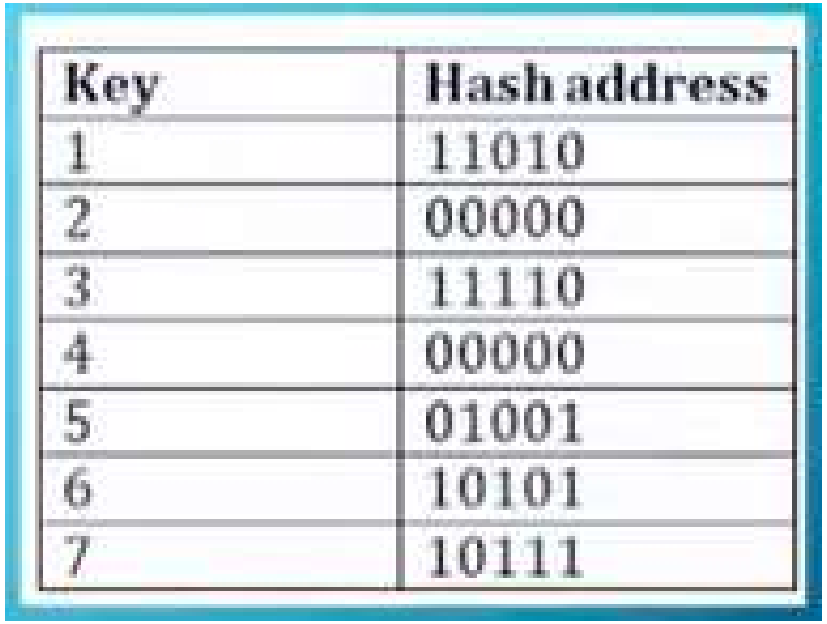
- Double Hashing –Double hashing is a computer programming method used in hash tables to resolve the issues of has a collision.

- Bucket Overflow: The condition of bucket-overflow is called collision. This is a fatal stage for any static has to function.

Hashing mechanism

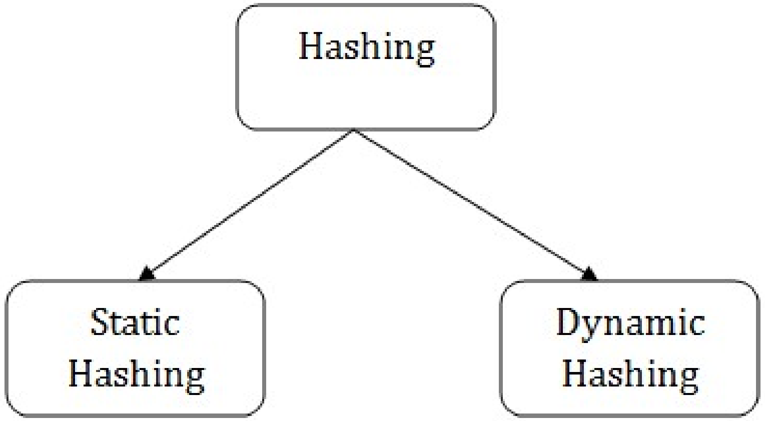


Hashing Mechanism



Key	Hash address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10101
7	10111

- Types of Hashing:



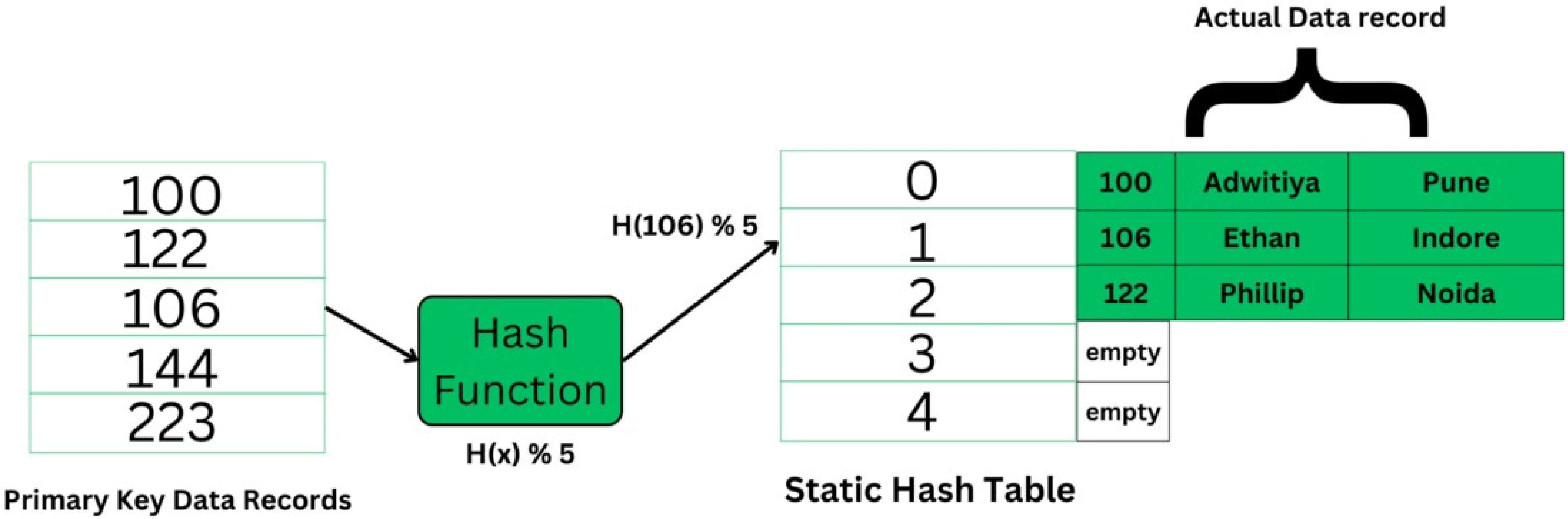
Hashing

Static Hashing

Dynamic Hashing

Static Hashing

- In the static hashing, the resultant data bucket address will always remain the same.
- In this, the number of data buckets in memory remains constant throughout.
- Example:
- if we have a data record for employee_id = 107, the hash function is mod-5
- which is – H(x) % 5, where x = id.
- Then the operation will take place like this:
- H(106) % 5 = 1.
  This indicates that the data record should be placed or searched in the 1st bucket (or 1st hash index) in the hash table.

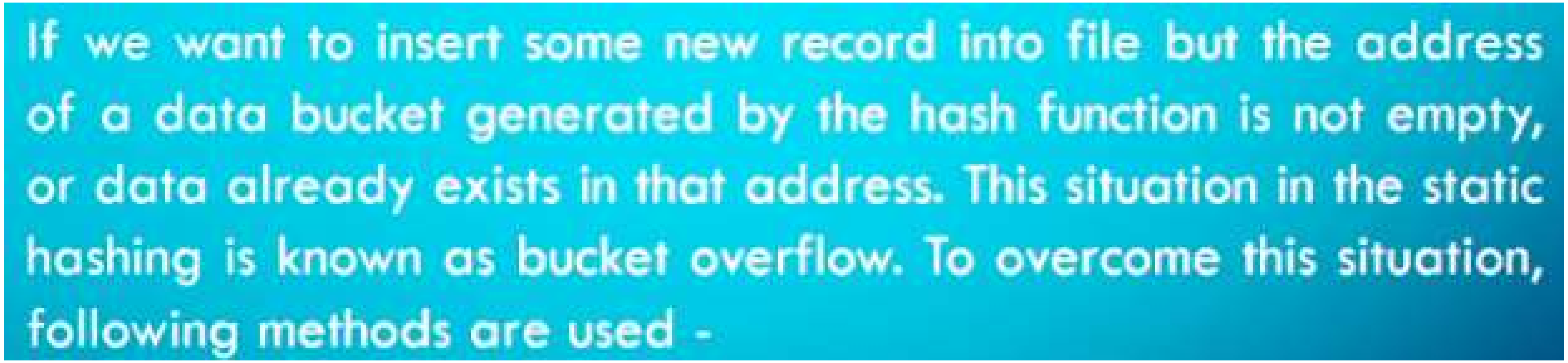


0	100	Adwitiya	Pune
1	106	Ethan	Indore
2	122	Phillip	Noida
3	empty		
4	empty		

Static Hash Functions

- Inserting a record:
- 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.
- Delete a record:
- 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.

Bucket overflow/Collision



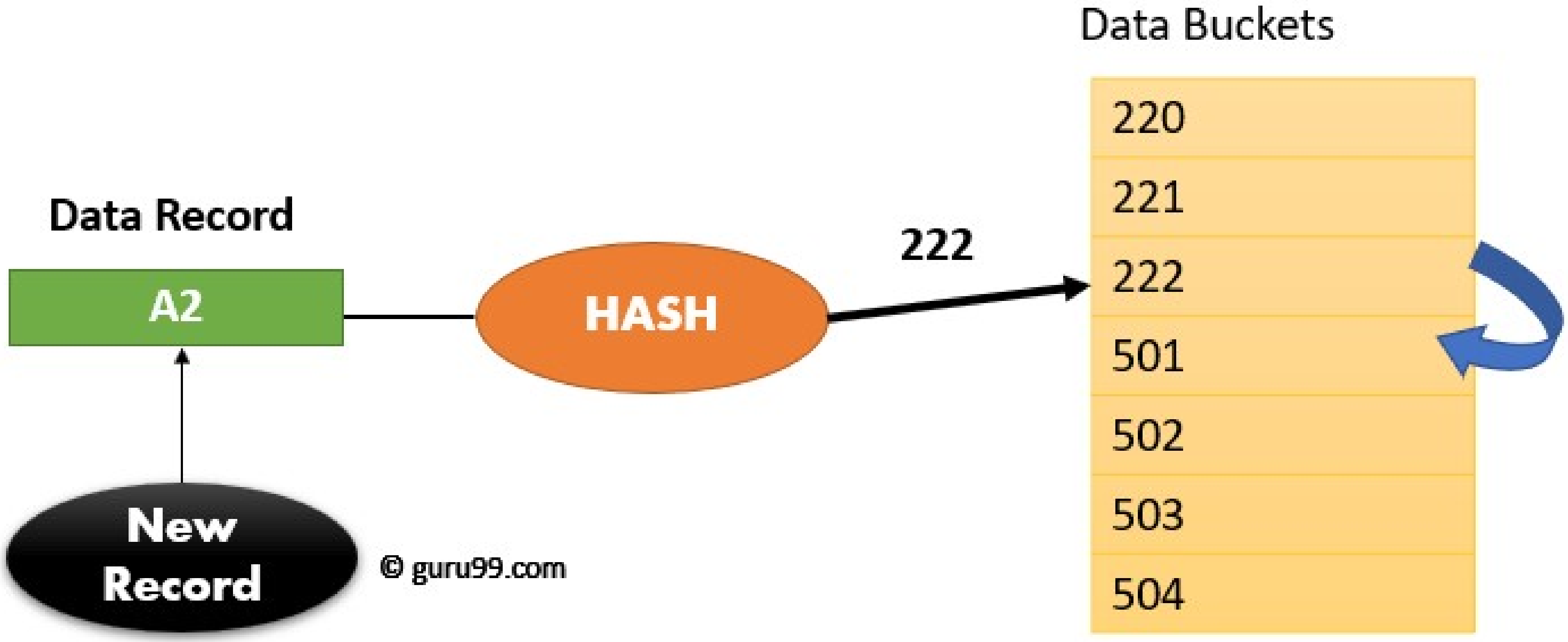
If we want to insert some new record into file but the address of a data bucket generated by the hash function is not empty, or data already exists in that address. This situation in the static hashing is known as bucket overflow. To overcome this situation, following methods are used -

- To resolve this problem of bucket overflow, techniques are used such as
- – chaining (closed hashing)
- -open addressing

Open Hashing

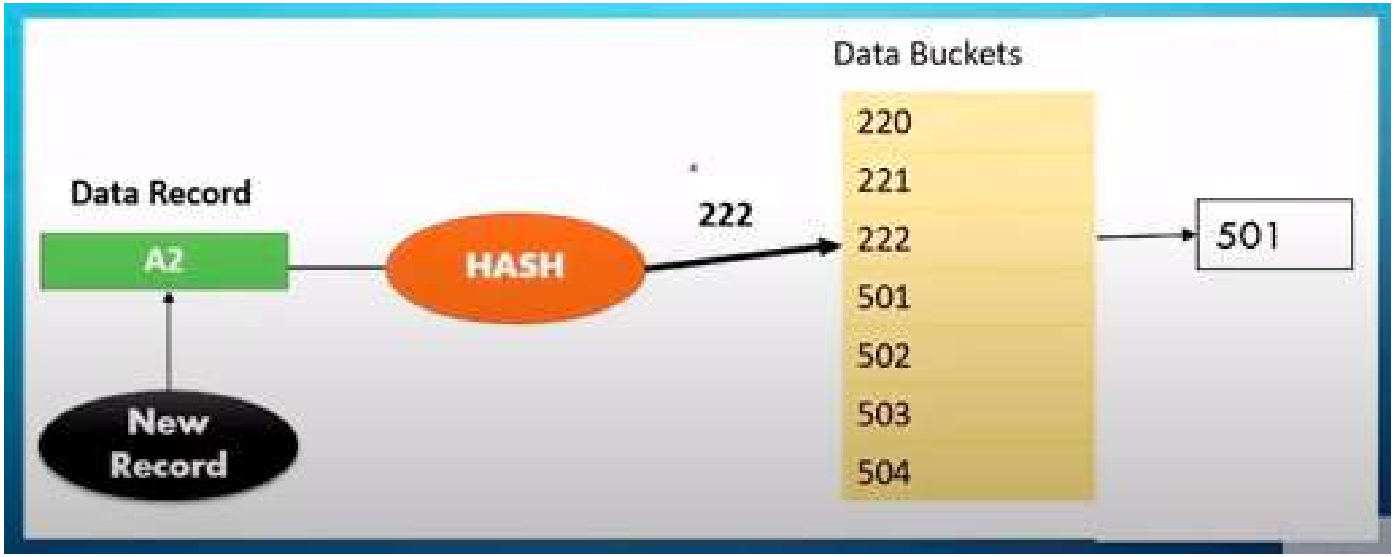
- In Open hashing method, Instead of overwriting older one the next available data block is used to enter the new record,
- This method is also known as linear probing.
- For example,
- A2 is a new record which you wants to insert. The hash function generates address as 222. But it is already occupied by some other value. That’s why the system looks for the next data bucket 501 and assigns A2 to it

Example:

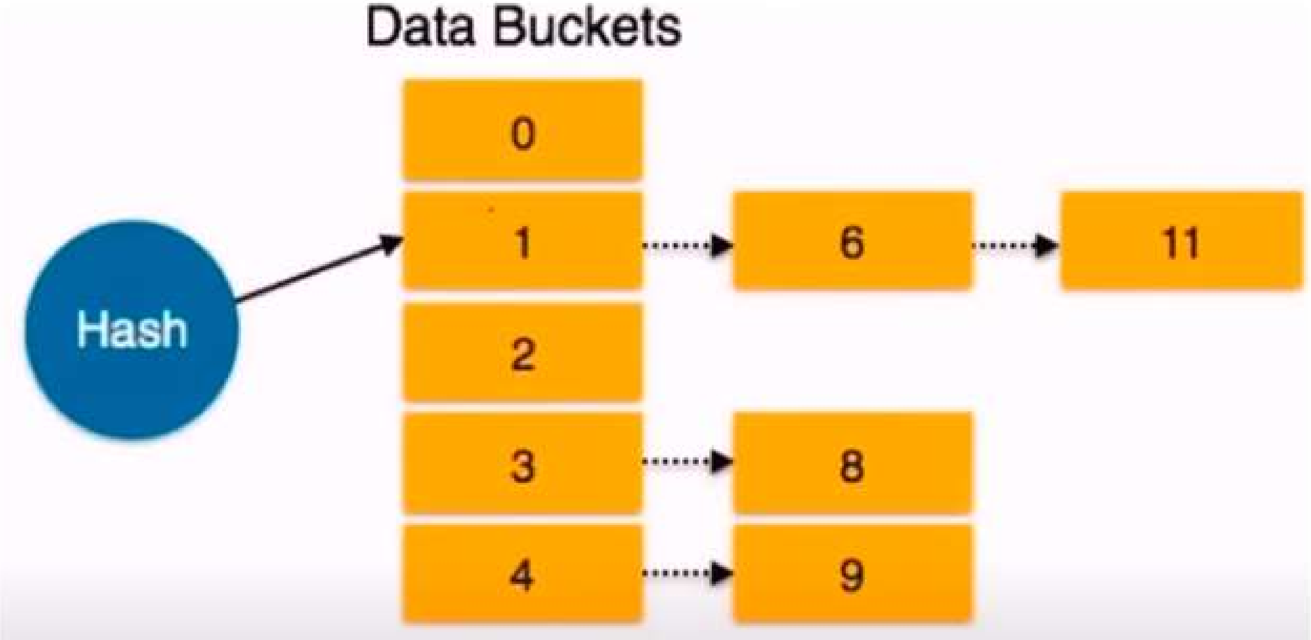

Data Buckets
220
221
222
501
502
503
504
Data Record
A2
HASH
222
New Record
© guru99.com

Chaining/closed 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.

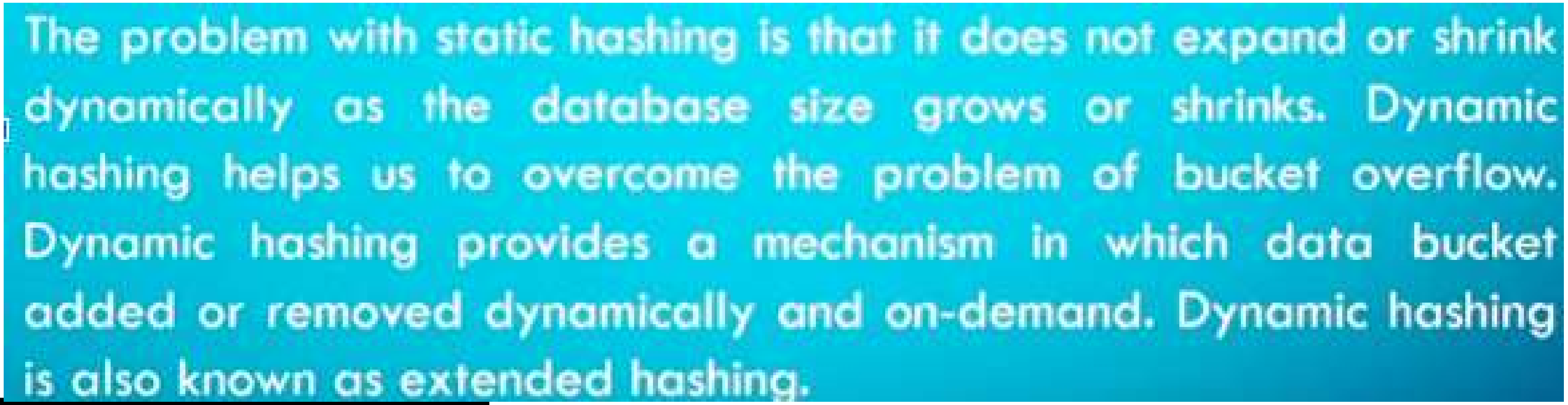


Example 2



Dynamic Hashing

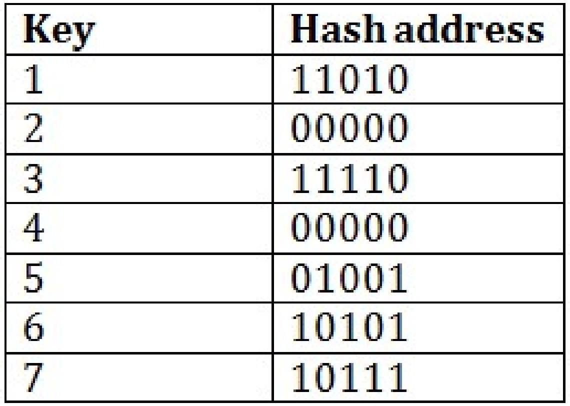
- Dynamic hashing offers a mechanism in which data buckets are added and removed dynamically and on demand.
- In this hashing, the hash function helps you to create a large number of values.



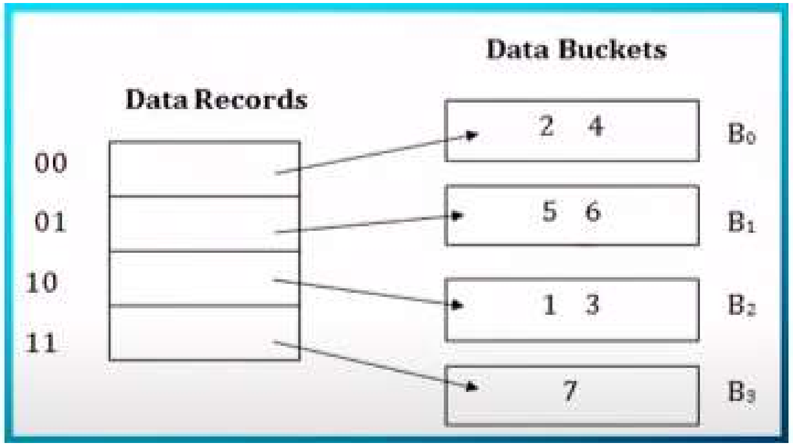
The problem with static hashing is that it does not expand or shrink dynamically as the database size grows or shrinks. Dynamic hashing helps us to overcome the problem of bucket overflow. Dynamic hashing provides a mechanism in which data bucket added or removed dynamically and on-demand. Dynamic hashing is also known as extended hashing.

Example

- Consider the following grouping of keys into bucket, depending on the prefix of their hash address.



Key	Hash address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10101
7	10111



- The last two bits of 2 and 4 are 00. So it will go into bucket B0. The last two bits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10, so it will go into bucket B2. The last two bits of 7 are 11, so it will go into B3.

Insert key 9 with hash address 10001 into the above structure:

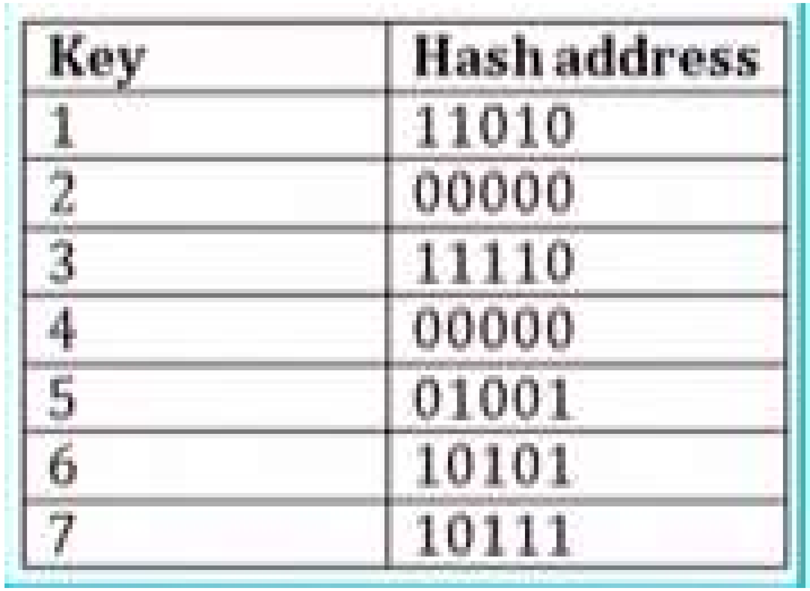
- Since key 9 has hash address 10001, it must go into the first bucket. But bucket B1 is full, so it will get split.

- The splitting will separate 5, 9 from 6 since last three bits of 5, 9 are 001, so it will go into bucket B1, and the last three bits of 6 are 101, so it will go into bucket B5.

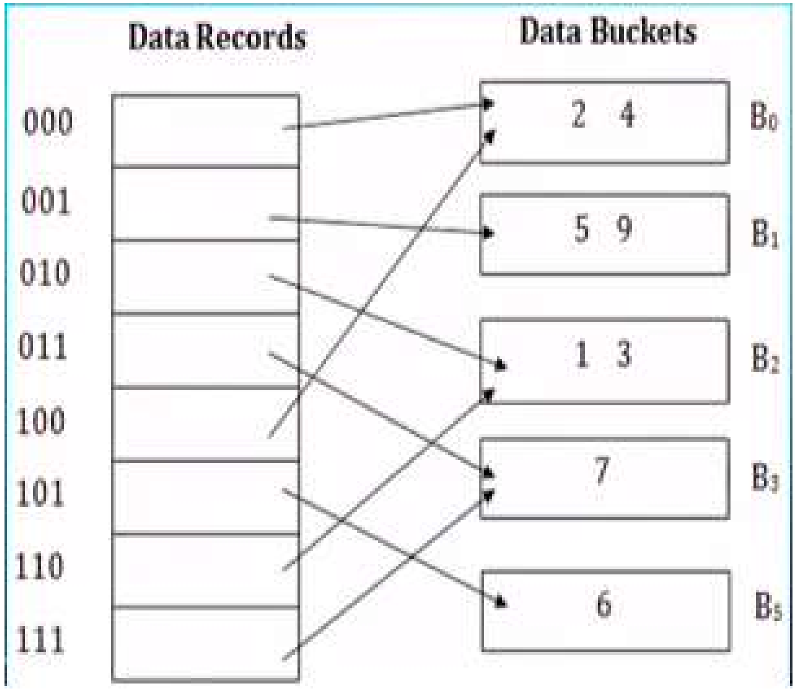
- Keys 2 and 4 are still in B0. The record in B0 pointed by the 000 and 100 entry because last two bits of both the entry are 00.

- Keys 1 and 3 are still in B2.
- The record in B2 pointed by the 010 and 110 entry because last two bits of both the entry are 10.

- Key 7 are still in B3. The record in B3 pointed by the 111 and 011 entry because last two bits of both the entry are 11.



Key	Hash address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10101
7	10111



Advantages of dynamic hashing

- In this method, the performance does not decrease as the data grows in the system. It simply increases the size of memory to accommodate the data.
- In this method, memory is well utilized as it grows and shrinks with the data. There will not be any unused memory lying.
- This method is good for the dynamic database where data grows and shrinks frequently.

Disadvantages of dynamic hashing

- In this method, if the data size increases then the bucket size is also increased.
- These addresses of data will be maintained in the bucket address table.
- This is because the data address will keep changing as buckets grow and shrink.
- If there is a huge increase in data, maintaining the bucket address table becomes tedious.
- In this case, the bucket overflow situation will also occur. But it might take little time to reach this situation than static hashing.