STATIC HASHING IN DBMS

PRESENTED BY:

BHAVANA THERUVATH

WHAT IS STATIC HASHING?

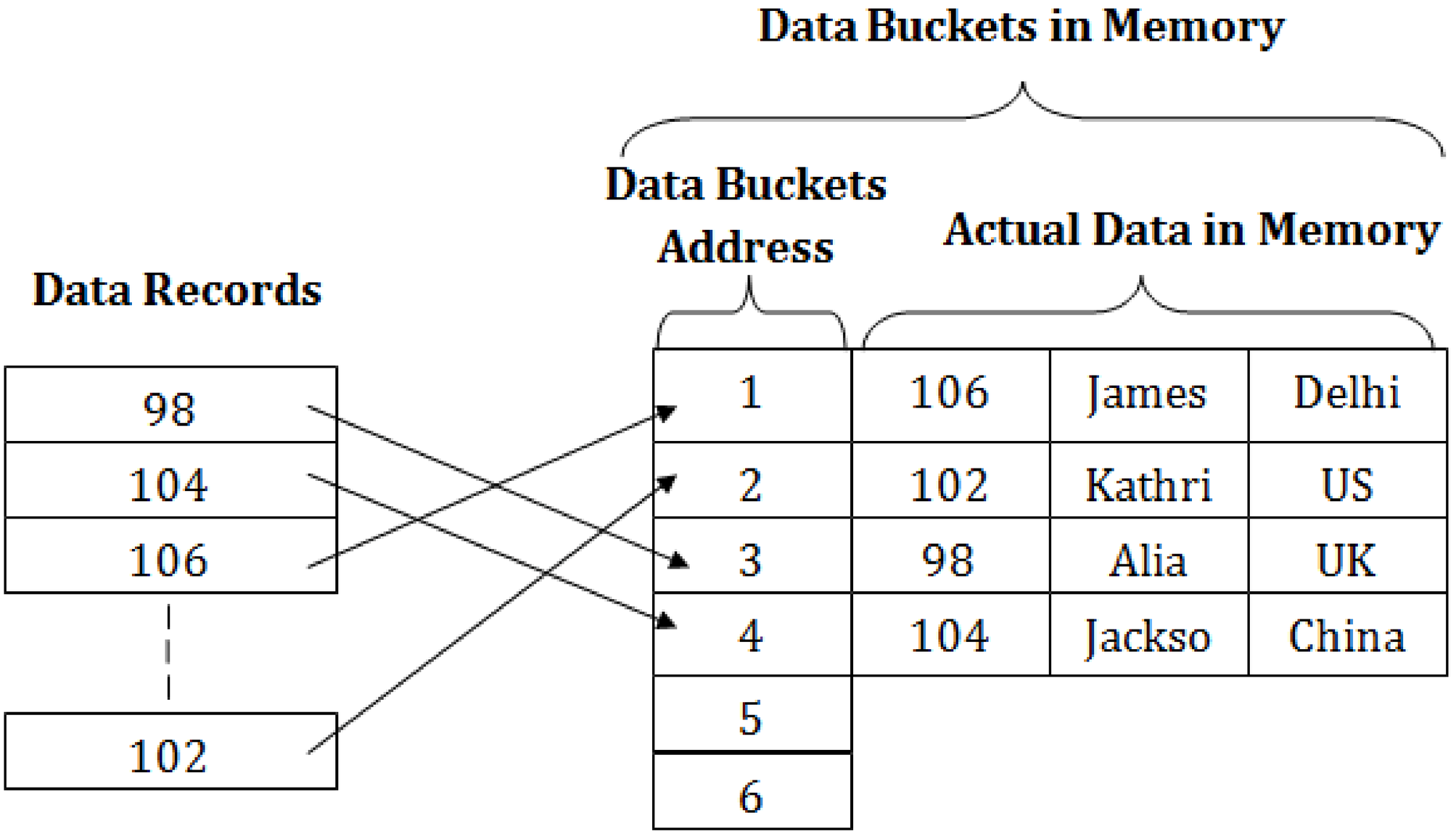
- Static Hashing is a technique used in databases to store and retrieve data using a fixed number of memory locations (buckets).
- A **bucket** is a fixed storage location in memory where data is stored in Static Hashing. Each bucket can hold one or more records.
- When you enter a data record, the **hash function** calculates an address and places the data there.
- When you search for that data, the **same hash function** will return the same address.
- The number of memory locations (buckets) **does not change** over time.

FOR EXAMPLE:

Imagine you have 5 buckets (memory locations) and you store Employee IDs using mod 5 as the hash function.

EMPLOYEE ID	HASH FUNCTION(ID%5)	BUCKET
98	98%5=3	3
104	104%5=4	4
106	106%5=1	1

So, if you search for **Employee ID 104**, the hash function returns **Bucket 4**, where the data is stored.


Data Buckets in Memory
Data Buckets Address
Actual Data in Memory
Data Records
98
104
106
102
1
2
3
4
5
6
106
James
Delhi
102
Kathri
US
98
Alia
UK
104
Jackso
China

OPERATIONS ON STATIC HASHING

1.SEARCHING DATA

- To retrieve data, the same hash function is applied to the search key, leading directly to the packet where the data is stored. This process offers efficient data retrieval.
- Example: Searching for EmpId 103, the function returns Bucket 3, where we find the data.

2.INSERTING DATA

- When new data is added to the table, the hash key is used to determine the address for the new data, and the data is placed at that location, assuming there's space.
- Example: Insert EmpId 104, and if 104 % 5 = 4, it goes to Bucket 4.

OPERATIONS ON STATIC HASHING

3.DELETING A RECORD

- For record deletion, the target record is identified, and its corresponding data is removed from memory. This ensures data integrity within the DBMS.

4.UPDATING A RECORD

- To update, we use the hash function to find the record and then change the profile info.
- If the address of the dataset created by the hash function is not empty or there is data in the address, we cannot add information. This is called bucket overflow.

BUCKET OVERFLOW PROBLEM

A bucket overflow occurs when we attempt to insert new data, but the memory location (or bucket) determined by the hash function is already full. This situation requires resolution to ensure efficient database performance.

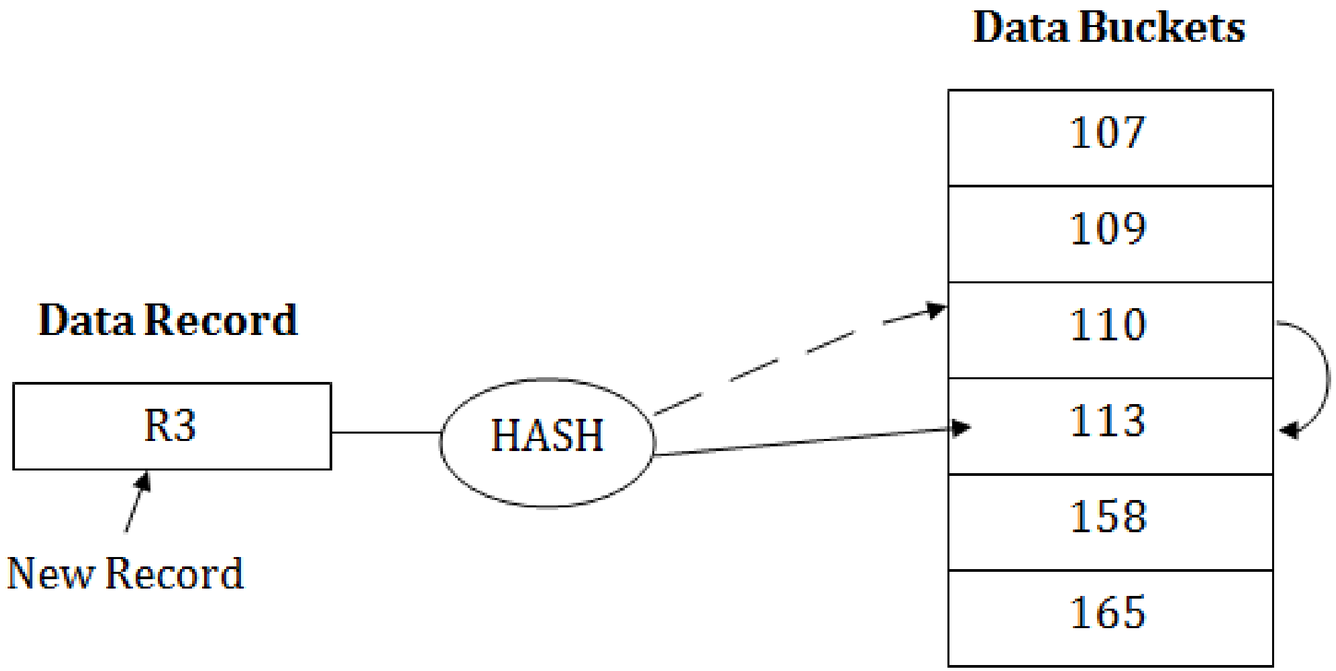
To overcome this situation, there are various methods. Some commonly used methods are as follows:

1.Open Hashing

2.Close Hashing

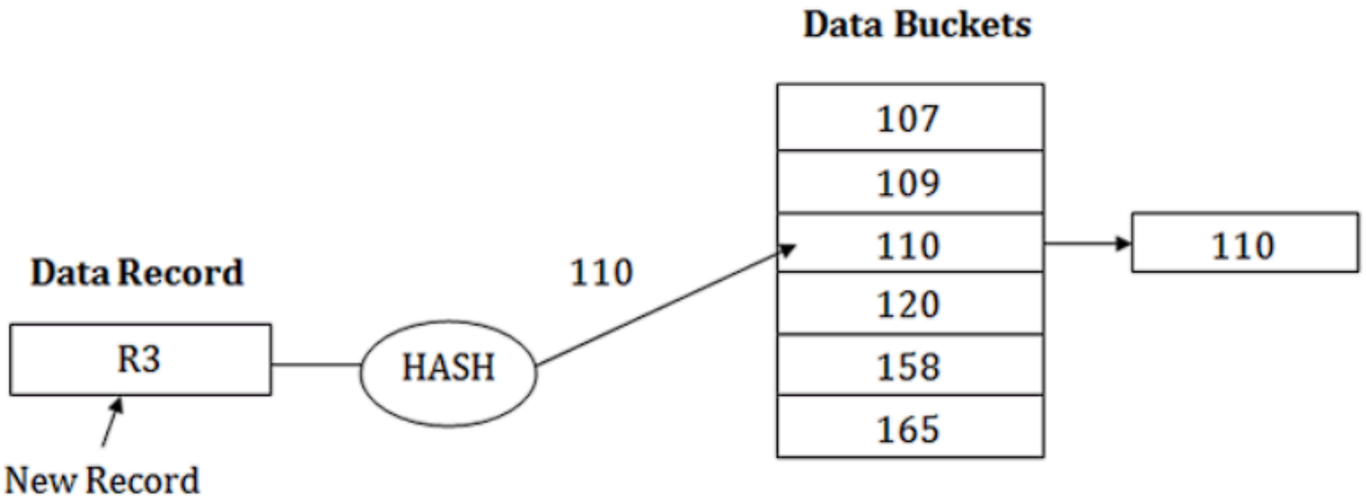
1.OPEN HASHING (LINEAR PROBING)

- When a hash function generates an address at which data is already stored, then the next bucket will be allocated to it.
- For example: suppose R3 is a new address which needs to be inserted, the hash function generates address as 112 for R3. But the generated address is already full. So the system searches next available data bucket, 113 and assigns R3 to it.



2.CLOSE HASHING (OVERFLOW CHAINING)

- When buckets are full, then a new data bucket is allocated for the same hash result and is linked after the previous one
- For example: Suppose R3 is a new address which needs to be inserted into the table, the hash function generates address as 110 for it. But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110 buckets and is linked to it.



ADVANTAGES OF STATIC HASHING

1.GOOD PERFORMANCE FOR SMALL DATA

Static hashing provides very efficient search and retrieval times when dealing with relatively small datasets, making it suitable for applications with limited data volumes.

2.EFFICIENT STORAGE MANAGEMENT

Static hashing simplifies storage management because the memory allocation is predetermined.

3.FACILITATES ACCESS

Direct access using hash keys.

DISADVANTAGES OF STATIC HASHING

1.NOT SUITABLE FOR BIG DATABASES (LIMITED BUCKETS).

Static hashing isn't well-suited for large datasets. Because the hash function must traverse all memory locations in the DBMS, this becomes inefficient.

2.NOT EXTENSIBLE

Static hashing doesn't work with extensible databases. The fixed bucket size and static nature make it difficult to adapt to growing or changing data needs.

3.SLOWER FOR LARGE DATASETS (DUE TO SEARCHING IN FIXED MEMORY LOCATIONS).

Sorting is not as efficient in static hashing, compared to other hashing techniques. The hash table order is unrelated to the key values

CONCLUSION

Static Hashing is a simple and effective way to store and retrieve data in databases **when the number of records is fixed**. However, for large and growing databases, **dynamic hashing** is a better choice.

THANK YOU...