4. HASHING

- Hashing is an effective technique to calculate the direct location of a data record on the disk without using index structure.
- Hashing uses hash functions with search keys as parameters to generate the address of a data record.

Hash Organization

Bucket

A hash file stores data in bucket format. Bucket is considered a unit of storage. A 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 the set of search-keys K to the address where actual records are placed. It is a function from search keys to bucket addresses.

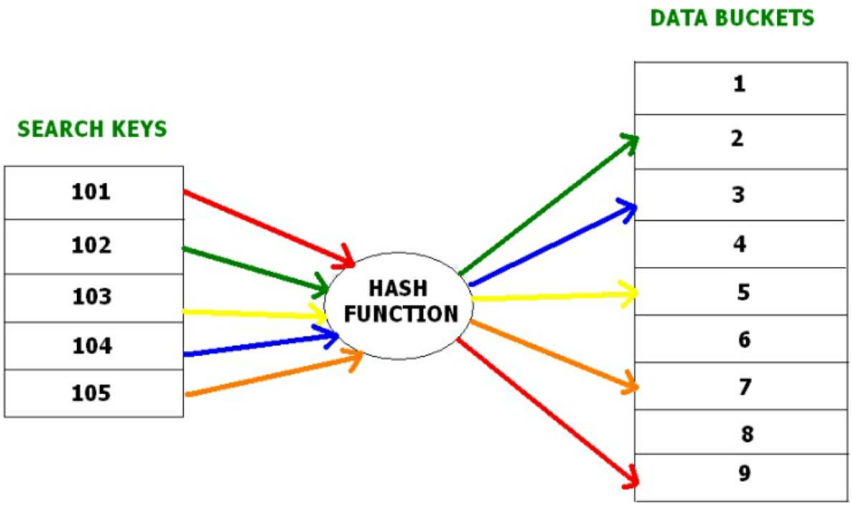
- Worst hash function maps all search-key values to the same bucket.
- An ideal hash function is uniform, i.e., each bucket is assigned the same number of search-key values from the set of all possible values.
- Ideal hash function is random, so each bucket will have the same number of records.

Types

- Static Hashing
- Dynamic Hashing

4.1 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 number of buckets provided remains unchanged at all times.
- There are 10 buckets,
- The hash function returns the sum of the binary representations of the characters modulo 10 – E.g. h(Perryridge) = 5 h(Round Hill) = 3 h(Brighton) = 3



Operations

(i) **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)

(ii) Search – When a record needs to be retrieved, the same hash function can be used to retrieve the address of the bucket where the data is stored.

(iii) Delete – This is simply a search followed by a deletion operation.

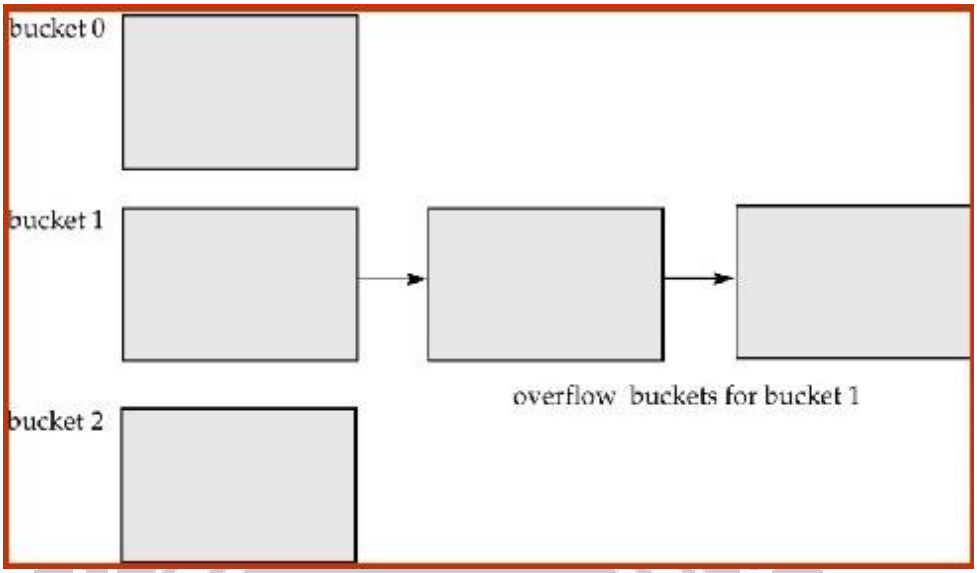
Bucket overflow can occur because of

- Insufficient buckets
- Skew in distribution of records. This can occur due to :
  - Multiple records have same search-key value

Although the probability of bucket overflow can be reduced, it cannot be eliminated; it is handled by using overflow buckets.

Overflow chaining

- The overflow buckets of a given bucket are chained together in a linked list.
- Above scheme is called closed hashing.
- An alternative, called open hashing, which does not use overflow buckets, is not suitable for database applications.

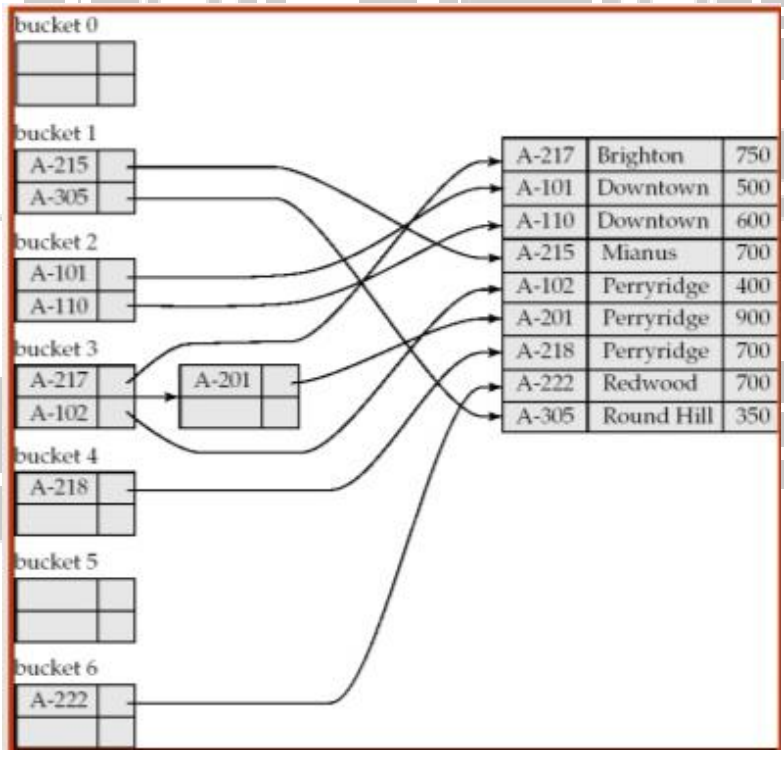


Hash Indices

• Hashing can be used not only for file organization, but also for index-structure creation.

• A hash index organizes the search keys, with their associated record pointers, into a hash file structure.

• Hash indices are always secondary indices



Deficiencies of Static Hashing

In static hashing, function h maps search-key values to a fixed set of B of bucket addresses.

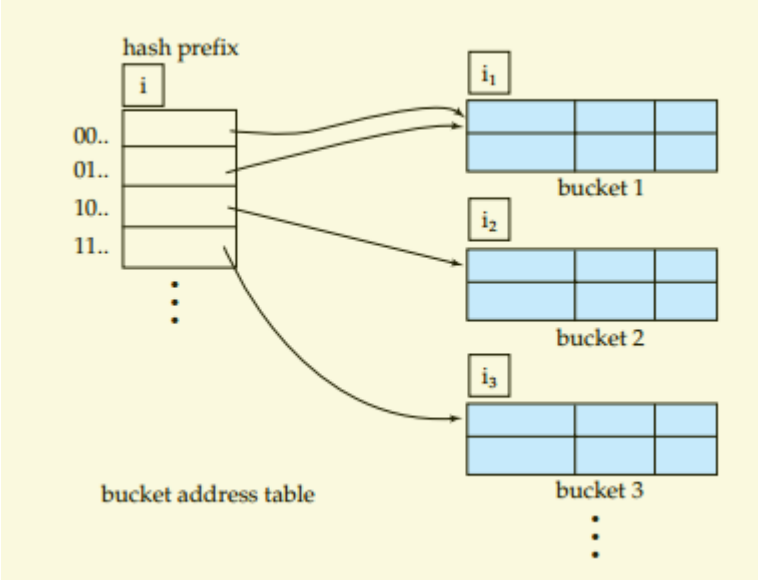
- Databases grow with time. If initial number of buckets is too small, performance will degrade due to too much overflows.
- If file size at some point in the future is anticipated and number of buckets allocated accordingly, significant amount of space will be wasted initially.
- If database shrinks, again space will be wasted.
- These problems can be avoided by using techniques that allow the number of buckets to be modified dynamically.

4.2 Dynamic Hashing

- Good for database that grows and shrinks in size
- Allows the hash function to be modified dynamically
- Extendable hashing – one form of dynamic hashing
  - Hash function generates values over a large range
  - Typically b-bit integers, with b = 32.
  - At any time use only a prefix of the hash function to index into a table of bucket addresses.
  - Let the length of the prefix be i bits, 0 <= i <=32.
  - Bucket address table size = 2i. Initially i = 0
  - Value of i grows and shrinks as the size of the database grows and shrinks.
  - Multiple entries in the bucket address table may point to a bucket.
  - Thus, actual number of buckets is < 2i
  - The number of buckets also changes dynamically due to coalescing and splitting of buckets.

4.3 General Extendable Hash

In this structure, i2 = i3 = i, whereas i1 = i – 1



Insertion in Extendable Hash Structure

To split a bucket j when inserting record with search-key value Kj:

- If i > ij (more than one pointer to bucket j)
  - Allocate a new bucket z, and set ij = iz = (ij + 1)
  - Update the second half of the bucket address table entries originally pointing to j, to point to z
  - Remove each record in bucket j and reinsert (in j or z)
  - Recompute new bucket for Kj and insert record in the bucket (further splitting is required if the bucket is still full)
- If i = ij (only one pointer to bucket j)
  - If i reaches some limit b, or too many splits have happened in this insertion, create an overflow bucket
- Else
  - Increment i and double the size of the bucket address table.
  - Replace each entry in the table by two entries that point to the same bucket.
  - Recompute new bucket address table entry for Kj
  - Now i > ij so use the first case above.

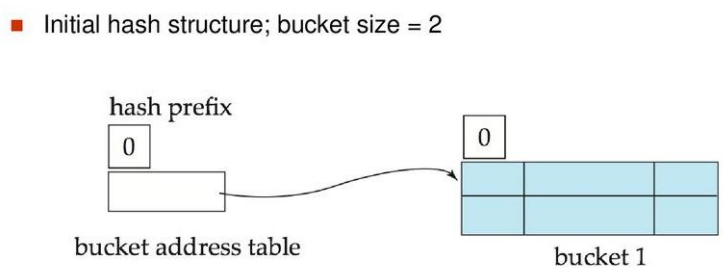
Deletion in Extendable Hash Structure

To delete a key value,

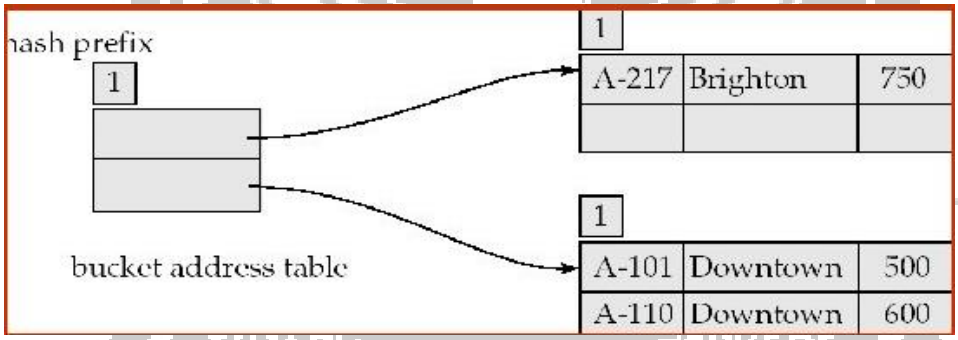
- Locate it in its bucket and remove it.
- The bucket itself can be removed if it becomes empty (with appropriate updates to the bucket address table).
- Coalescing of buckets can be done (can coalesce only with a “buddy” bucket having same value of ij and same ij –1 prefix, if it is present)
- Decreasing bucket address table size is also possible

• Note: decreasing bucket address table size is an expensive operation and should be done only if number of buckets becomes much smaller than the size of the table

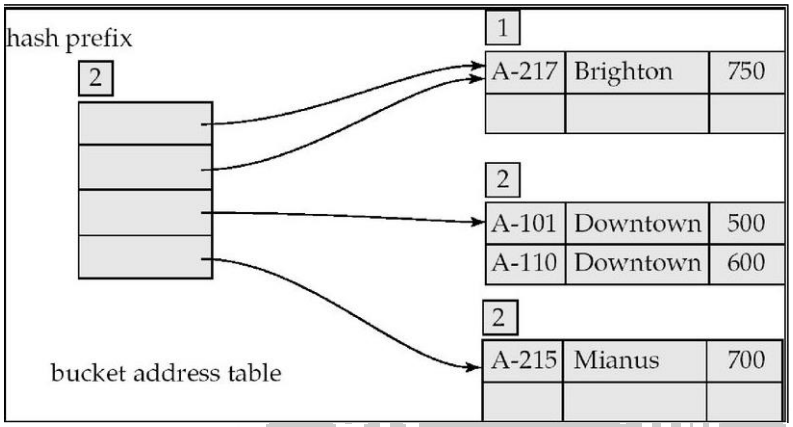
Example



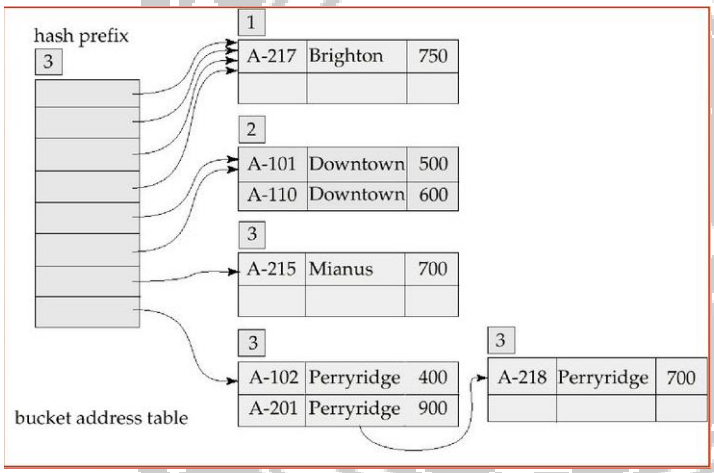
(i)Hash structure after insertion of one Brighton and two Downtown records



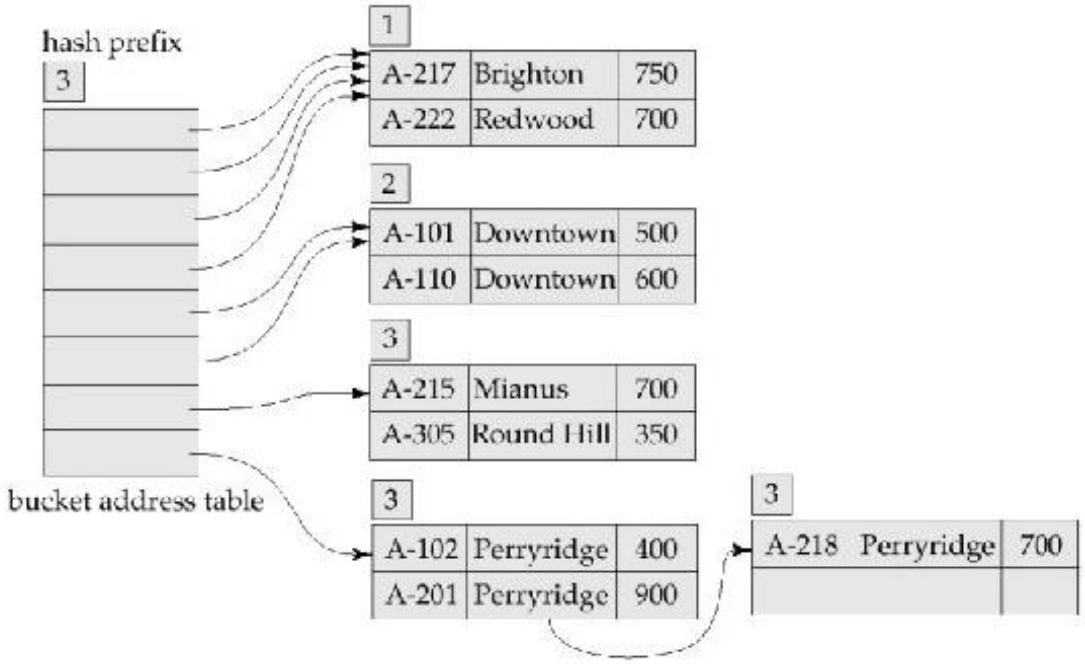
(ii) Hash structure after insertion of Mianus record



(iii) Hash structure after insertion of three Perryridge records



(iv)Hash structure after insertion of Redwood and Round Hill records



Use of Extendable Hash Structure

To locate the bucket containing search-key Kj:

1. Compute h(Kj) = X

2. Use the first i high order bits of X as a displacement into bucket address table, and follow the pointer to appropriate bucket

Updates in Extendable Hash Structure

- To insert a record with search-key value Kj
  - Follow same procedure as look-up and locate the bucket, say j.
  - If there is room in the bucket j insert record in the bucket.
  - Overflow buckets used instead in some cases.

- To delete a key value,
  - Locate it in its bucket and remove it.
  - The bucket itself can be removed if it becomes empty
  - Coalescing of buckets can be done
  - Decreasing bucket address table size is also possible
- Benefits of extendable hashing:
  - Hash performance does not degrade with growth of file
  - Minimal space overhead
- Disadvantages of extendable hashing
  - Extra level of indirection to find desired record

Bucket address table may itself become very big.