SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : 23CST207 - DATABASE MANAGEMENT SYSTEMS

II YEAR / IV SEMESTER

Unit 5- Physical Storage and MongoDB
Topic 6 : Static and Dynamic Hashing

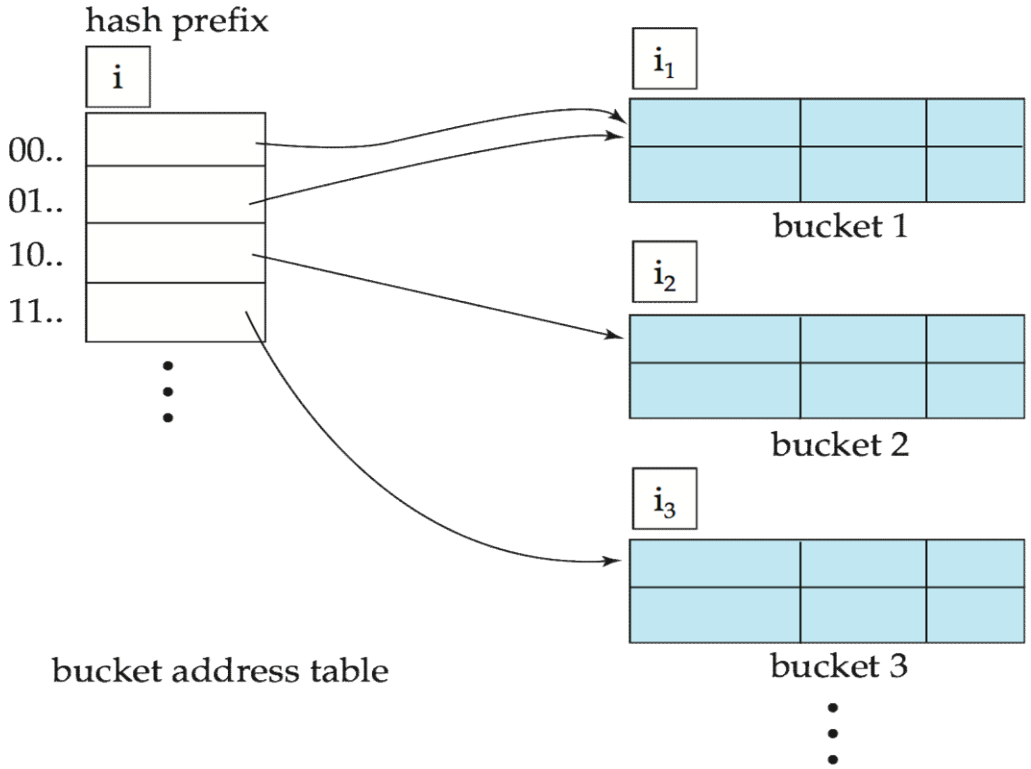
Static Hashing

- In static hashing, function *h* maps search-key values to a fixed set of *B* of bucket addresses. Databases grow or shrink with time.
  - If initial number of buckets is too small, and file grows, performance will degrade due to too much overflows.
  - If space is allocated for anticipated growth, a significant amount of space will be wasted initially (and buckets will be underfull).
  - If database shrinks, again space will be wasted.
- One solution: periodic re-organization of the file with a new hash function
  - Expensive, disrupts normal operations
- Better solution: allow the number of buckets to be modified dynamically

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, $0 \leq i \leq 32$.
    - Bucket address table size = 2^{i}. 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 (why?)
  - Thus, actual number of buckets is $< 2^{i}$
    - The number of buckets also changes dynamically due to coalescing and splitting of buckets.

General Extendable Hash Structure



In this structure, $i_2 = i_3 = i$, whereas $i_1 = i - 1$ (see next slide for details)

Use of Extendable Hash Structure

- Each bucket *j* stores a value i_j
  - *A*ll the entries that point to the same bucket have the same values on the first i_j bits.
- To locate the bucket containing search-key K_j:
  1. Compute $h(K_j) = X$
  2. Use the first *i* high order bits of *X* as a displacement into bucket address table, and follow the pointer to appropriate bucket
- To insert a record with search-key value K_j
  - 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.
  - Else the bucket must be split and insertion re-attempted (next slide.)
    - Overflow buckets used instead in some cases (will see shortly)

Insertion in Extendable Hash Structure (Cont)

- If $i > i_j$ (more than one pointer to bucket j)
  - allocate a new bucket z, and set $i_j = i_z = (i_j + 1)$
  - 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 K_j and insert record in the bucket (further splitting is required if the bucket is still full)
- If $i = i_j$ (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 K_j
      Now $i > i_j$ so use the first case above.

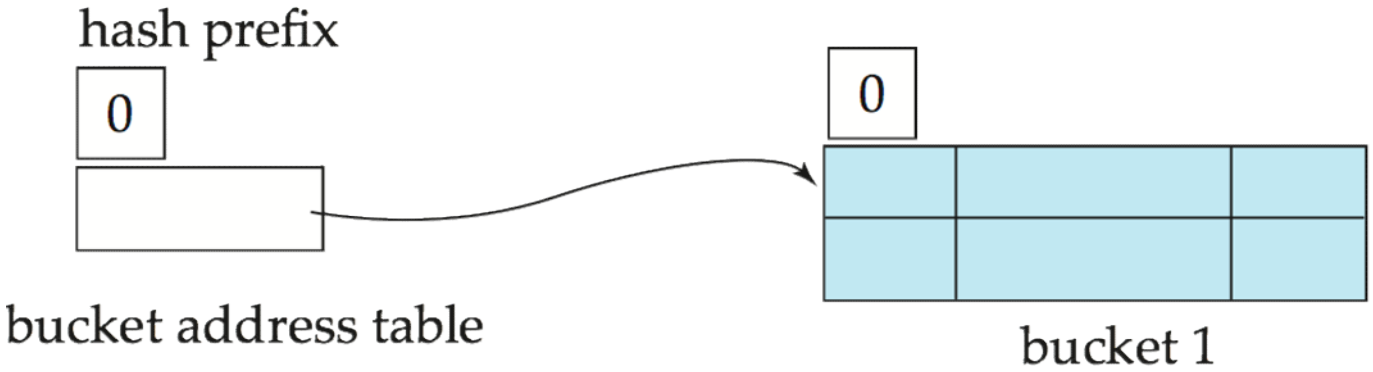
Deletion in Extendable Hash Structure

- 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 i_j and same i_j –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

Use of Extendable Hash Structure: Example

dept_name	h(*dept_name*)
Biology	0010 1101 1111 1011 0010 1100 0011 0000
Comp. Sci.	1111 0001 0010 0100 1001 0011 0110 1101
Elec. Eng.	0100 0011 1010 1100 1100 0110 1101 1111
Finance	1010 0011 1010 0000 1100 0110 1001 1111
History	1100 0111 1110 1101 1011 1111 0011 1010
Music	0011 0101 1010 0110 1100 1001 1110 1011
Physics	1001 1000 0011 1111 1001 1100 0000 0001

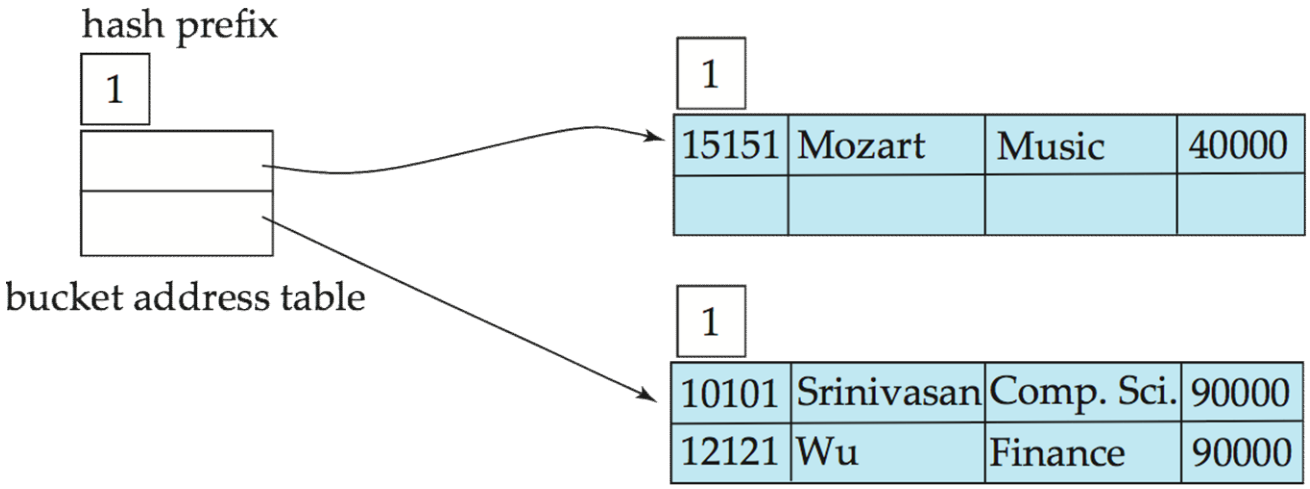
Example (Cont.)



- Initial Hash structure; bucket size = 2

Example (Cont.)

- Hash structure after insertion of “Mozart”, “Srinivasan”, and “Wu” records



Thank you