Dynamic Hashing

Database System Concepts, 6th Ed.

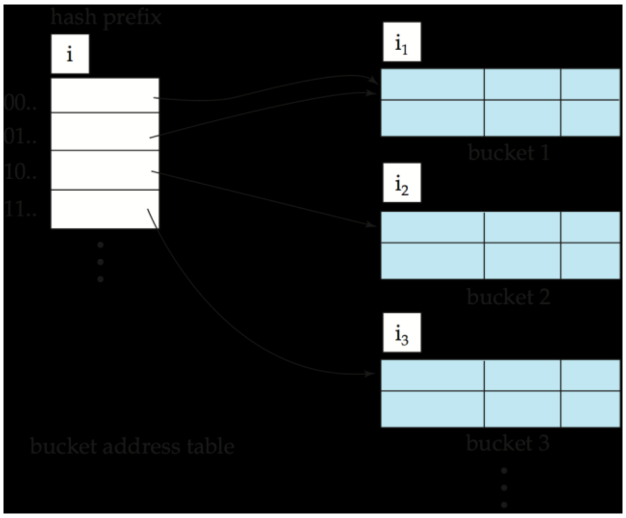
Deficiencies of Static Hashing

- 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 as the size of the database grows.
  - 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 depth *i*
  - *A*ll the entries that point to the same bucket have the same first *i* 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)

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

- Compare local depth to global depth
- If local depth == global depth,
  - Double directory size
  - Increase global depth by 1 bit

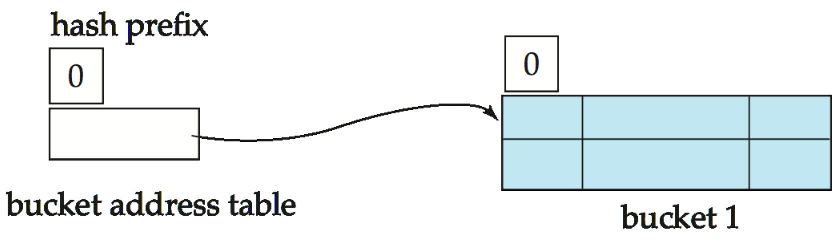
- Split bucket using 1 extra bit
- Adjust directory entry appropriately

Hash key is department code

dept_name	
Biology	0010
Comp. Sci.	1111
Elec. Eng.	0100
Finance	1010
History	1100
Music	0011
Physics	1001

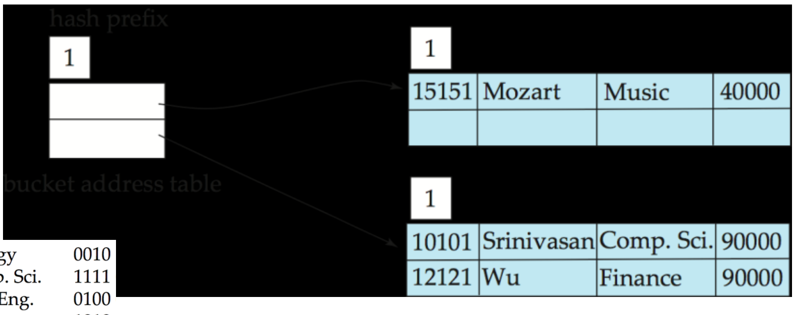
Example (Cont.)

- Initial Hash structure; bucket size = 2



Example (Cont.)

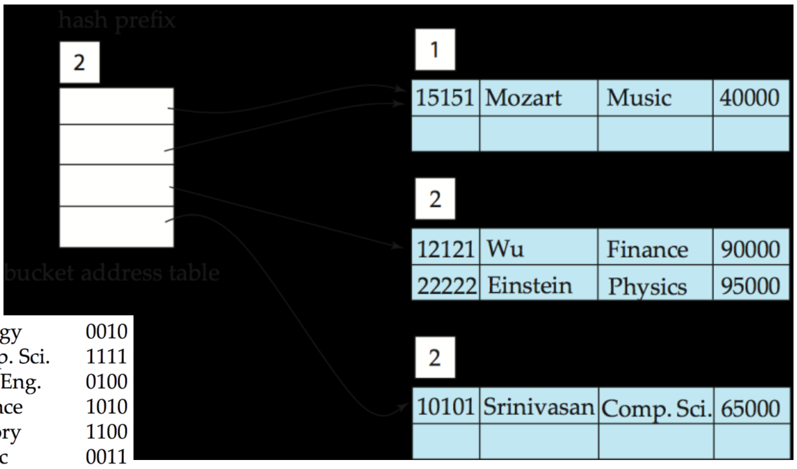
- Hash structure after insertion of “Mozart”, “Srinivasan”, and “Wu” records



Biology	0010
Comp. Sci.	1111
Elec. Eng.	0100
Finance	1010
History	1100
Music	0011
Physics	1001

Example (Cont.)

- Hash structure after insertion of Einstein record

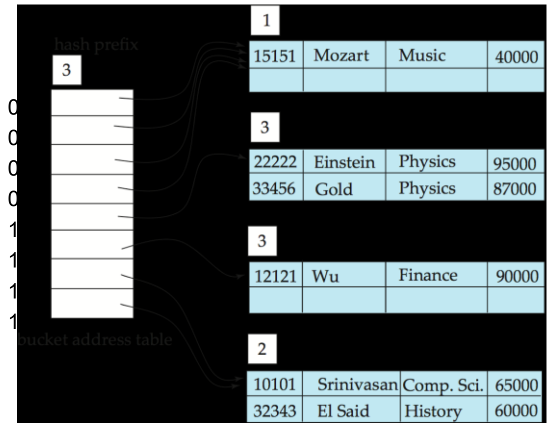


Biology	0010
Comp. Sci.	1111
Elec. Eng.	0100
Finance	1010
History	1100
Music	0011
Physics	1001

Example (Cont.)

- Hash structure after insertion of Gold and El Said records

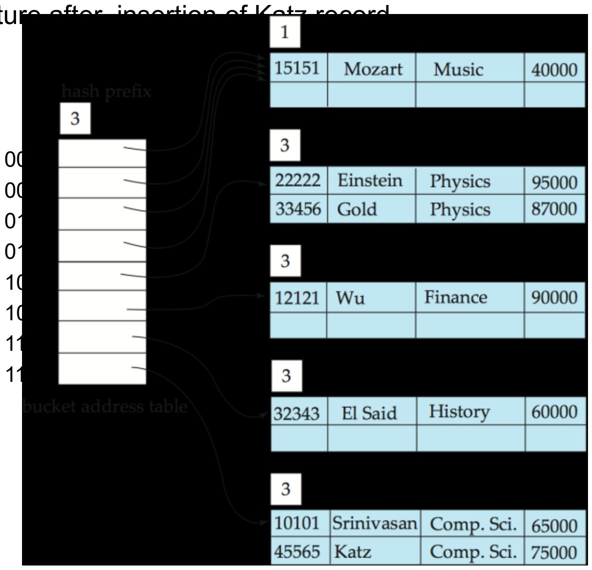
Biology	0010
Comp. Sci.	1111
Elec. Eng.	0100
Finance	1010
History	1100
Music	0011
Physics	1001



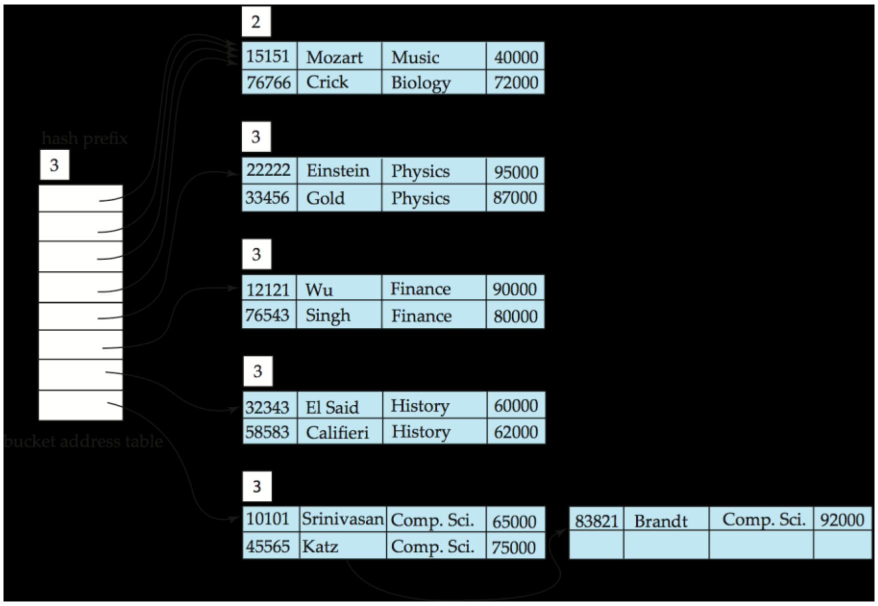
Example (Cont.)

- Hash structure after insertion of Katz record

Biology	0010
Comp. Sci.	1111
Elec. Eng.	0100
Finance	1010
History	1100
Music	0011
Physics	1001



Example (Cont.)



15151	Mozart	Music	40000
76766	Crick	Biology	72000

22222	Einstein	Physics	95000
33456	Gold	Physics	87000

12121	Wu	Finance	90000
76543	Singh	Finance	80000

32343	El Said	History	60000
58583	Califieri	History	62000

10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000

83821	Brandt	Comp. Sci.	92000

Extendable Hashing vs. Other Schemes

- 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 (larger than memory)
  - Changing size of bucket address table is an expensive operation