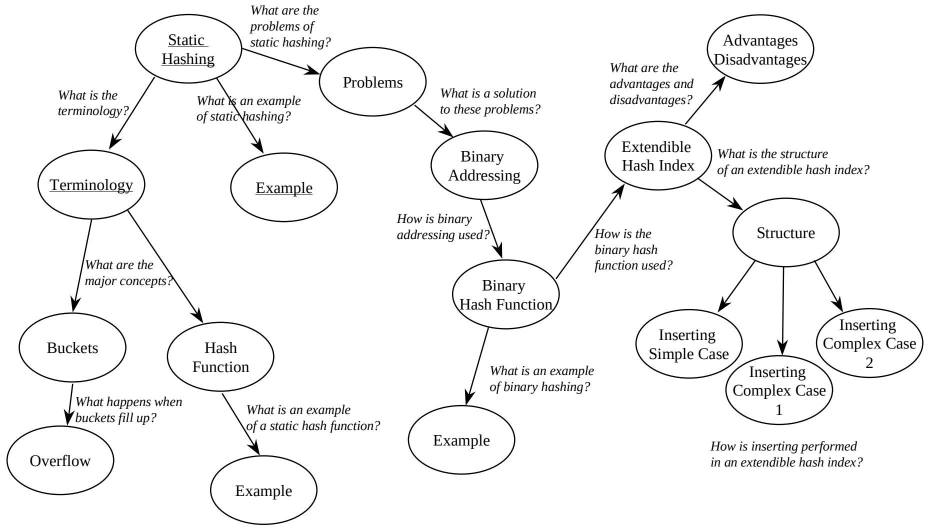
Extendible Hashing

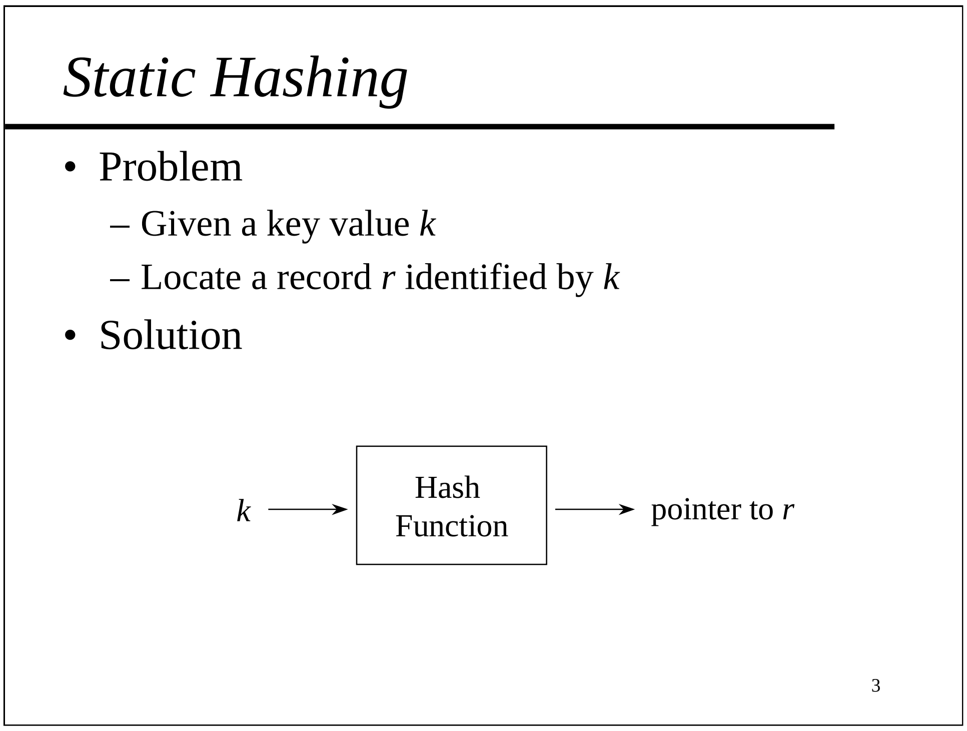
Database Systems Concepts
Silberschatz/ Korth
Sec. 11.5-11.7

Fundamentals of Database Systems
Elmasri/Navathe
Sec. 5.9

Overview


Static Hashing
What are the problems of static hashing?
Problems
What is the terminology?
What is an example of static hashing?
Terminology
Example
What are the major concepts?
Buckets
Hash Function
What happens when buckets fill up?
Overflow
What is an example of a static hash function?
Example
What is a solution to these problems?
Binary Addressing
How is binary addressing used?
Binary Hash Function
What is an example of binary hashing?
Example
How is the binary hash function used?
Extendible Hash Index
What are the advantages and disadvantages?
Advantages Disadvantages
What is the structure of an extendible hash index?
Structure
Inserting Simple Case
Inserting Complex Case 1
Inserting Complex Case 2
How is inserting performed in an extendible hash index?

2



Static Hashing

- Problem
  - – Given a key value *k*
  - – Locate a record *r* identified by *k*
- Solution

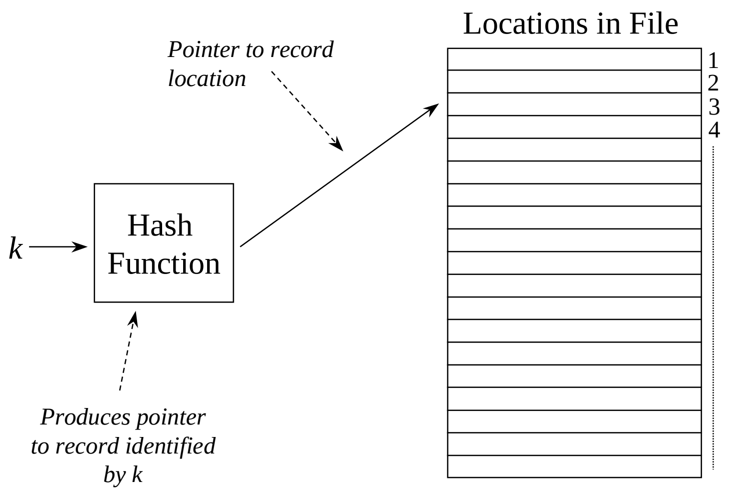
$$\begin{tikzcd} k \arrow[r] & \boxed{\text{Hash Function}} \arrow[r] & \text{pointer to } r \end{tikzcd}$$

3

- One problem with tree index structures, for example, the B$^+$-Tree, is that the index tree must be searched every time a record is sought.
- Hashing attempts to solve this problem by using a function, for example, a mathematical function, to *calculate* the address of a record from the value of its primary key.
- Static hashing uses a single function to calculate the position of a record in a fixed set of storage locations.

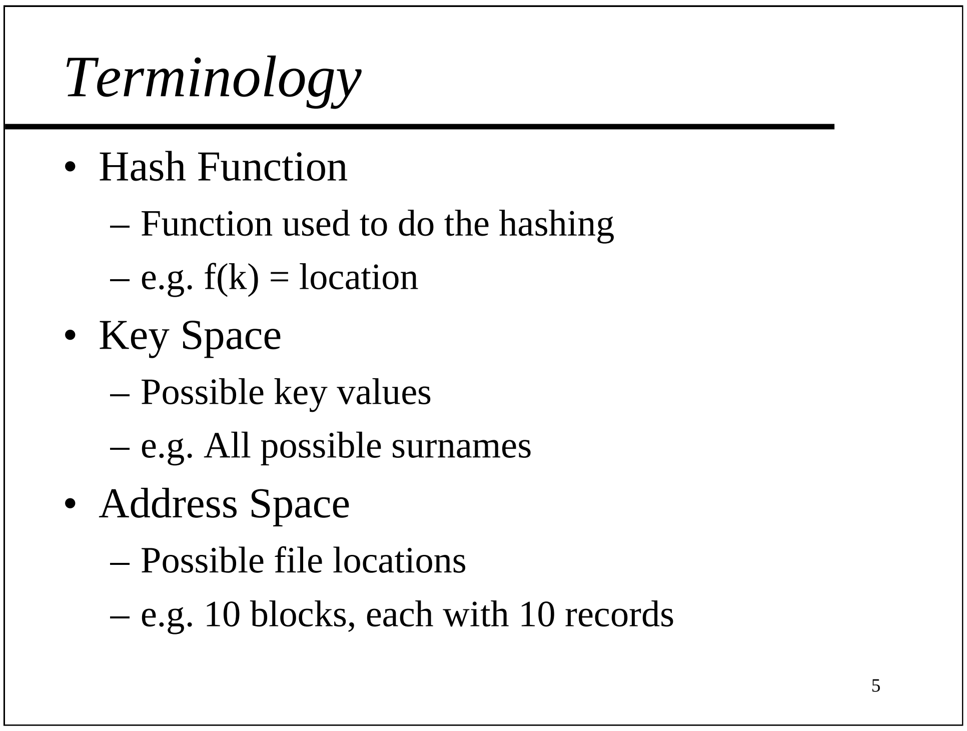
Ref: Silberschatz, sec 11.5; Elmasri, sec 5.9.

Example


Locations in File
Pointer to record location
k
Hash Function
Produces pointer to record identified by k
1
2
3
4

4

- Locating the position of a record identified by value *k* involves applying the hash function to *k*.
- The result of the hash function, called a *hash address*, is a pointer to the location in the file that should contain the record.
- When there are many possible records compared to the number of locations, it is possible for the hash function to point to the same location for two records, called a *collision*.
- A good hash function will limit the number of records with the same hashed address.



Terminology

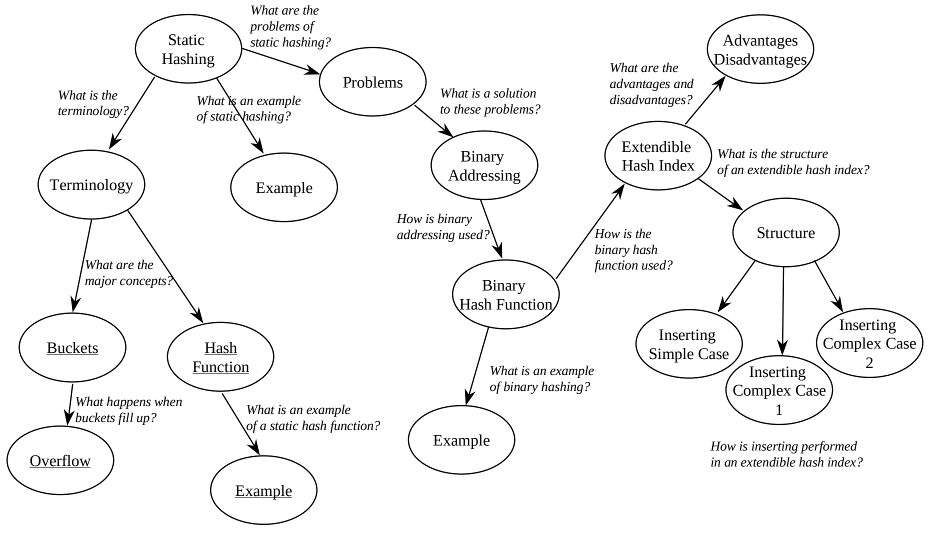
- Hash Function
  - – Function used to do the hashing
  - – e.g. f(k) = location
- Key Space
  - – Possible key values
  - – e.g. All possible surnames
- Address Space
  - – Possible file locations
  - – e.g. 10 blocks, each with 10 records

5

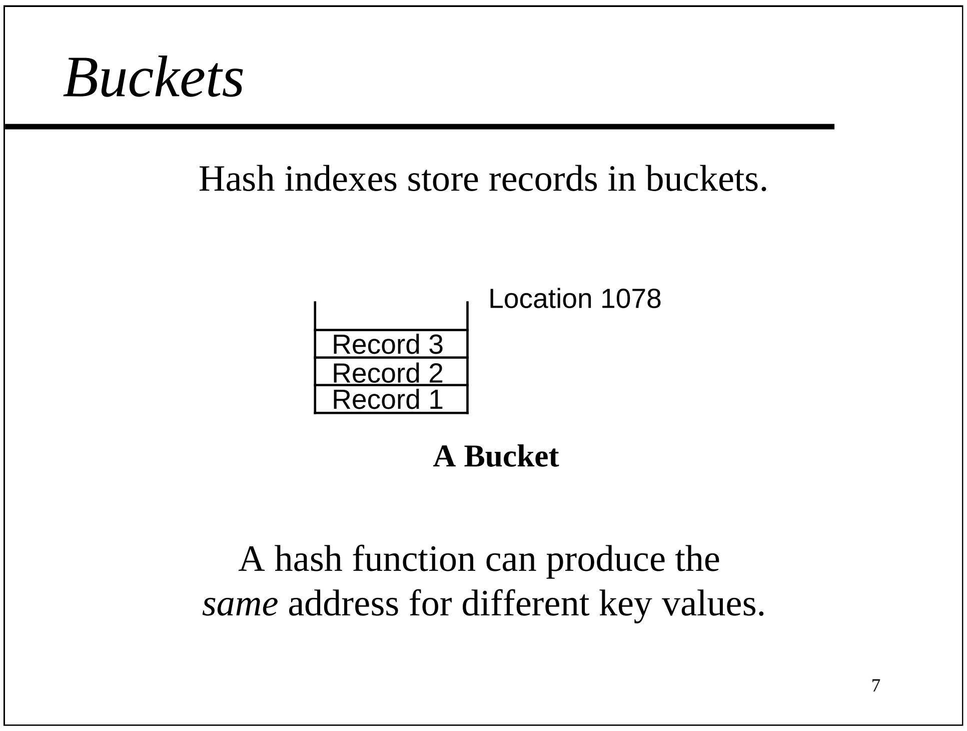
- A *hash function* is applied to a key value and returns the location in a file where the record should be stored.
  - For example, a function *f* when applied to a key value *k*, i.e. *f(k)* will return the address of the record identified by *k*.
- The *key space* is the set of all the key values that can appear in the database being indexed using the hash function. Elmasri et al calls the key space the *hash field space*.
  - For example, the key space for a student database will consist of the student numbers of all students to be stored in the database.
- The *address space* is the set of all locations in the file that will store the database.
  - For example, a file that consists of an address space of twenty has twenty locations in which to store records.
- The size of the key space will normally be larger than the size of the address space.
  - For example, although the address space of students may consist of 6000 students, the library may assume that only 4000 students will borrow books at any one time. Using this assumption the library will allocate an address space of 4000.
  - A hash function must be able to place any of the 6000 students into one of the 4000 addresses available.

Ref: Elmasri, sec 5.9; Silberschatz, sec 11.5.

Overview

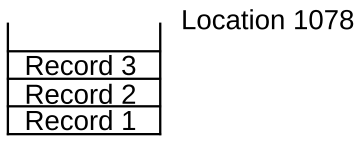

Static Hashing
What are the problems of static hashing?
Problems
What is the terminology?
What is an example of static hashing?
What is a solution to these problems?
Terminology
Example
Binary Addressing
What are the major concepts?
How is binary addressing used?
Buckets
Hash Function
Binary Hash Function
What happens when buckets fill up?
What is an example of a static hash function?
What is an example of binary hashing?
Overflow
Example
Example
What are the advantages and disadvantages?
Advantages Disadvantages
Extendible Hash Index
What is the structure of an extendible hash index?
How is the binary hash function used?
Structure
Inserting Simple Case
Inserting Complex Case 1
Inserting Complex Case 2
How is inserting performed in an extendible hash index?

6



Buckets

Hash indexes store records in buckets.



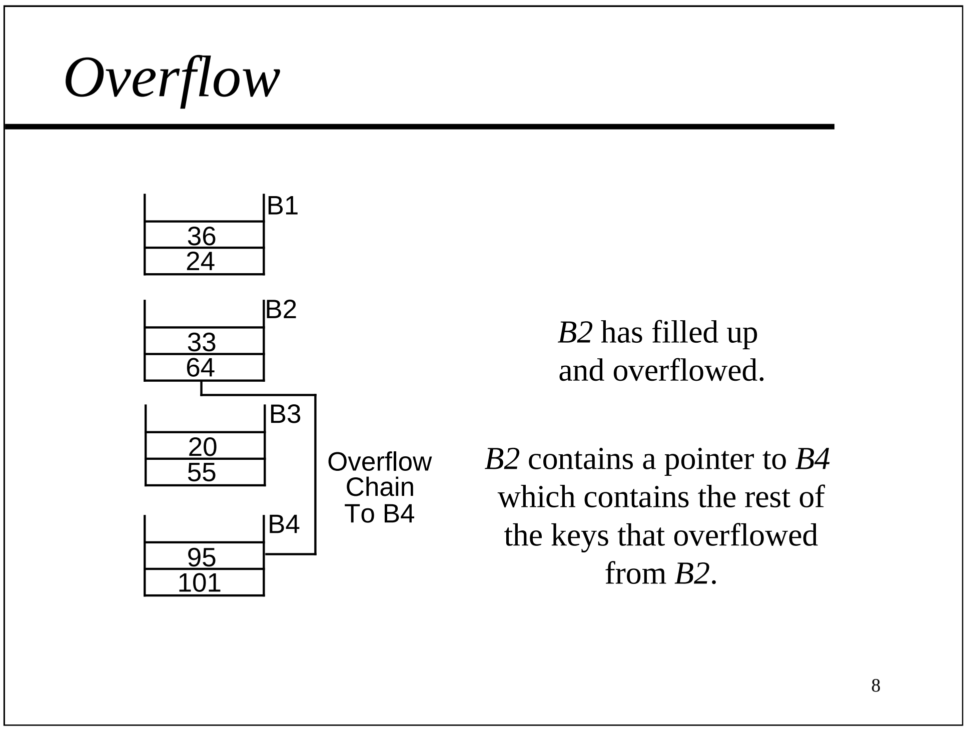
A Bucket

A hash function can produce the *same* address for different key values.

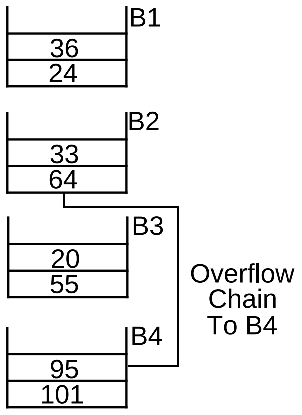
7

- Like a B^+-Tree, which stores records in blocks or pages on the disc, a hash index stores records in blocks called *buckets*.
- A bucket has a unique location address and may contain several records.
- A hash function must convert a key value into a bucket address. Two or more key values may map to the same bucket.
- In the above example, records 1, 2 and 3 returned the same hash address (1078) when the hash function has been applied to them.

Silberschatz, sec 11.5.



Overflow



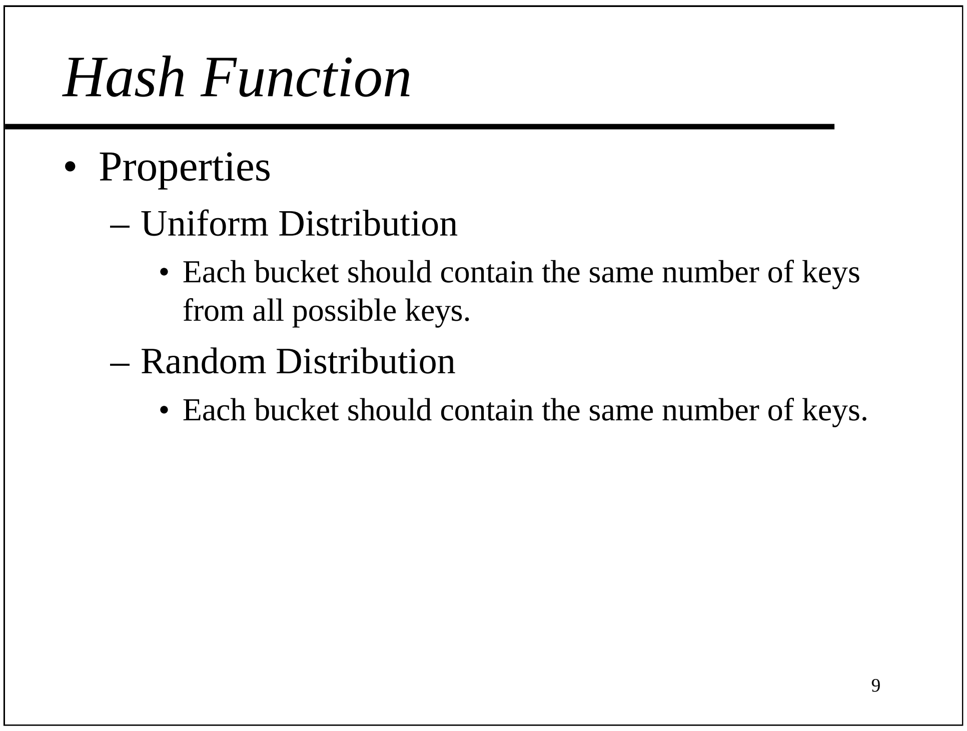
B2 has filled up and overflowed.

B2 contains a pointer to *B4* which contains the rest of the keys that overflowed from *B2*.

8

- It is possible for a hash function to try to put too many records into a bucket.
- In this case, it is necessary to use an *overflow bucket*.
- An overflow bucket contains records that will not fit into the bucket in which they have been placed by the hash function.
- Overflow buckets are undesirable because they make the length of a search unpredictable.
- Instead of the hash function producing the address of the bucket containing the record, the hash function gives the address of the first bucket in a chain of buckets. One bucket in the chain will contain the record.
- For instance, in the above example, two buckets must be read from the disc to find key 95, but only one bucket must be read from the disc to find key 36.

Ref: Elmasri, sec 5.9.



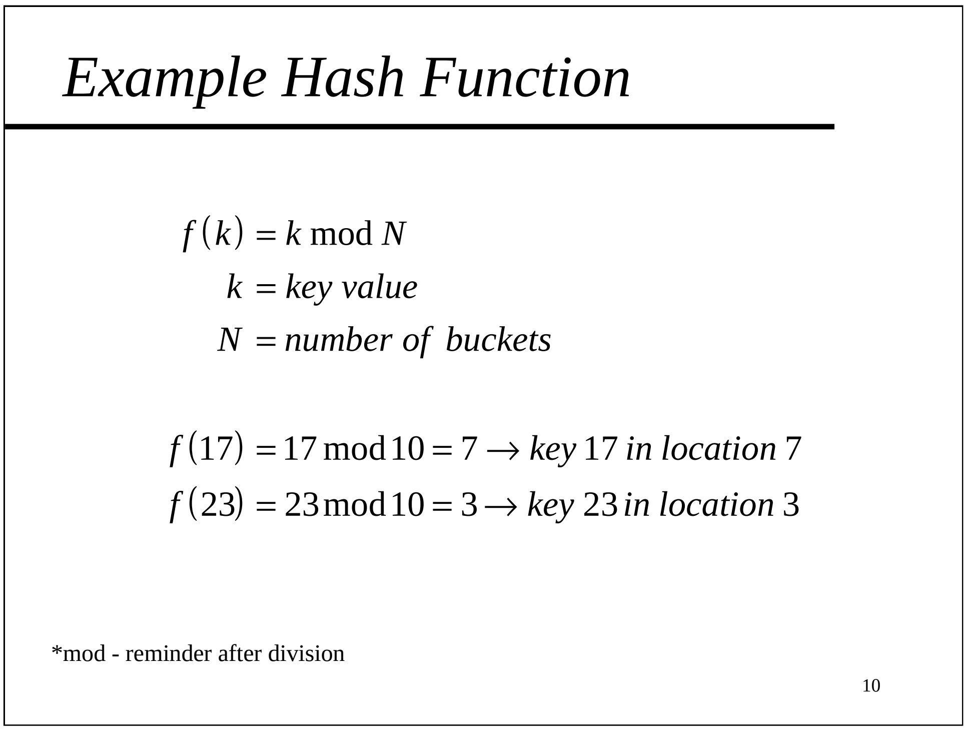
Hash Function

- Properties
  - Uniform Distribution
    - Each bucket should contain the same number of keys from all possible keys.
  - Random Distribution
    - Each bucket should contain the same number of keys.

9

- Korth et al states that a good hash function should have two properties:
  - *Uniform distribution* A hash function should ensure that each bucket contain keys from all parts of the key space. For example, a good hash function for names would ensure that each bucket had a set of names which began with letters from all parts of the alphabet.
  - *Random distribution* A hash function should distribute key values equally among the index locations. That is, each bucket should have approximately the same number of keys.
- These properties help to guarantee a good distribution of key values across all the buckets in the index.

Ref: Silberschatz, sec 11.5.



Example Hash Function

$$f(k) = k \bmod N$$
$$k = key\ value$$
$$N = number\ of\ buckets$$

$$f(17) = 17 \bmod 10 = 7 \rightarrow key\ 17\ in\ location\ 7$$
$$f(23) = 23 \bmod 10 = 3 \rightarrow key\ 23\ in\ location\ 3$$

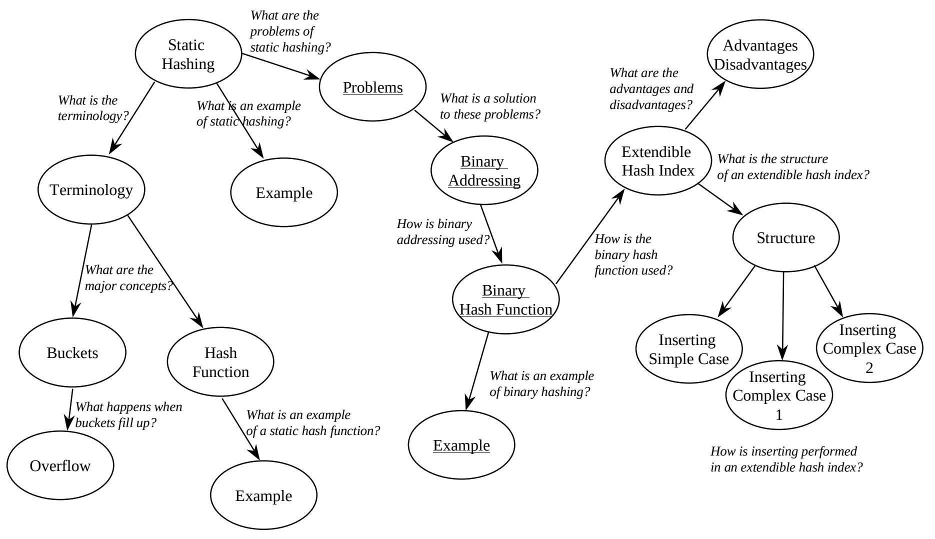
*mod - reminder after division

10

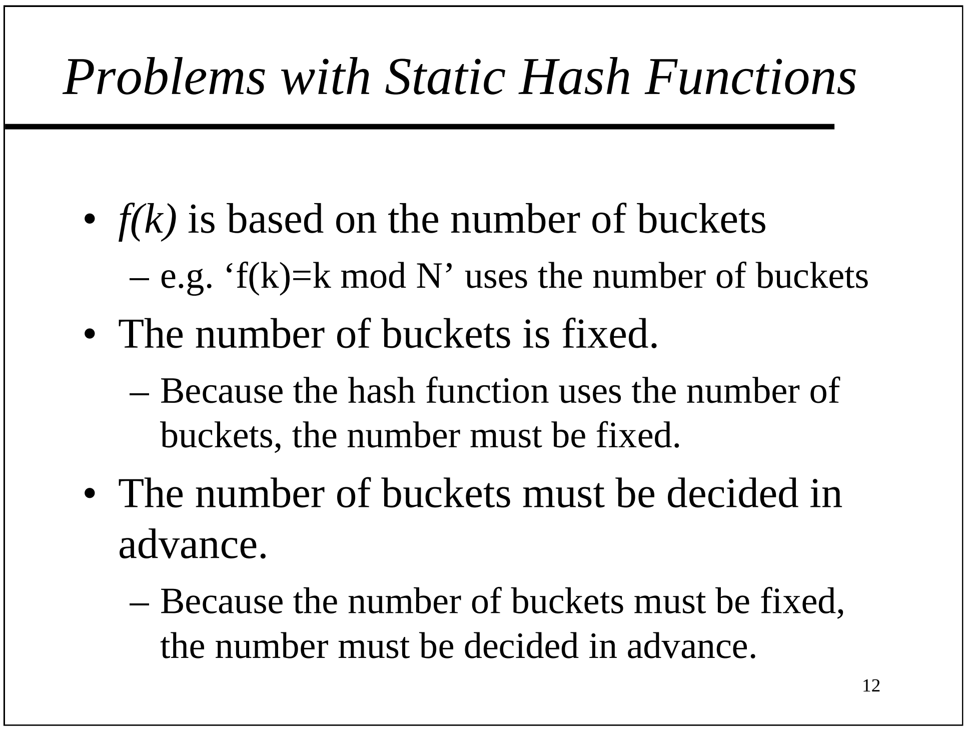
- A common hash function is the *f(k)=k mod N* function which calculates the location by using the remainder resulting from dividing the key by the number of buckets.
- If the key is not a number then it is converted to a number, for example, by using the ASCII code of the letters in the key.

Ref: Elmasri,sec 5.9.

Overview


Static Hashing
What are the problems of static hashing?
Problems
What is the terminology?
What is an example of static hashing?
What is a solution to these problems?
Terminology
Example
Binary Addressing
How is binary addressing used?
What are the major concepts?
Binary Hash Function
How is the binary hash function used?
Buckets
Hash Function
What happens when buckets fill up?
What is an example of a static hash function?
What is an example of binary hashing?
Overflow
Example
Example
Extendible Hash Index
What are the advantages and disadvantages?
Advantages Disadvantages
What is the structure of an extendible hash index?
Structure
Inserting Simple Case
Inserting Complex Case 1
Inserting Complex Case 2
How is inserting performed in an extendible hash index?

11



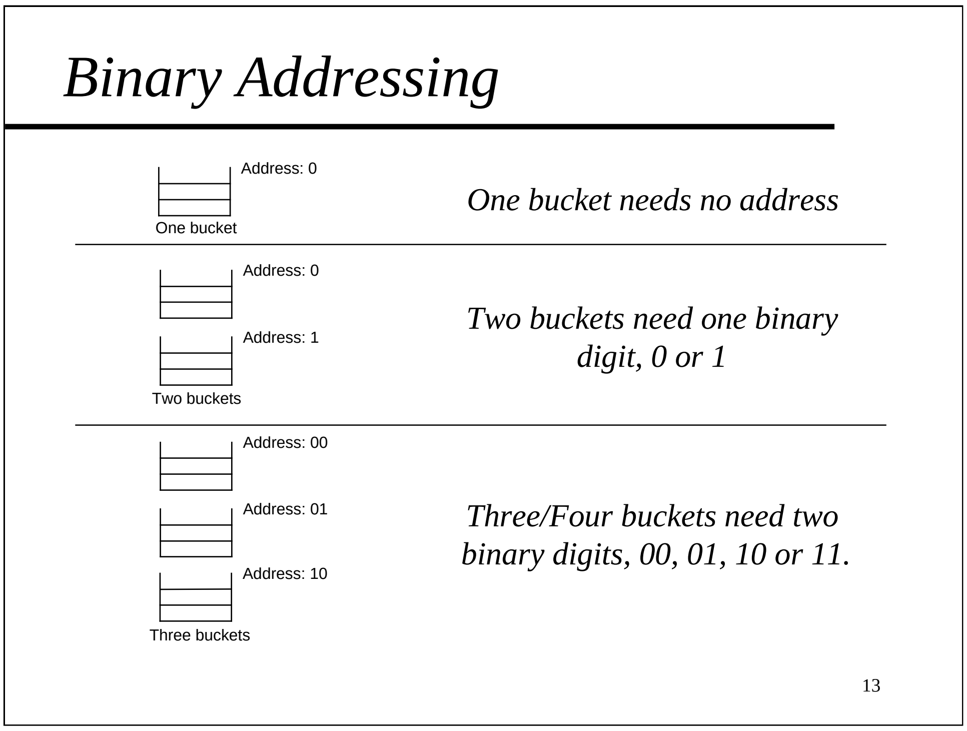
Problems with Static Hash Functions

- *f(k)* is based on the number of buckets
  – e.g. 'f(k)=k mod N' uses the number of buckets
- The number of buckets is fixed.
  – Because the hash function uses the number of buckets, the number must be fixed.
- The number of buckets must be decided in advance.
  – Because the number of buckets must be fixed, the number must be decided in advance.

12

- A static hash function such as 'f(k)=k mod N' uses the number of buckets in the file to calculate the hashed key.
- This means that the number of buckets in the file must be known in advance and must remain unchanged for the lifetime of the file.
- To use a static hash function there are three main options:
    - Base the hash function on the current number of records in the file. This will not be suitable if the number of records changes.
    - Base the hash function on the anticipated number of records in the file. This will not be suitable if estimates of the file size are incorrect.
    - Periodically re-organise the file and change the hash function. When a new hash function is created, all the record locations must be re-calculated.
- Alternatively, the hash function could be designed to change automatically as the file size grows and shrinks.

Ref: Silberschatz sec 11.6.



Binary Addressing

One bucket needs no address

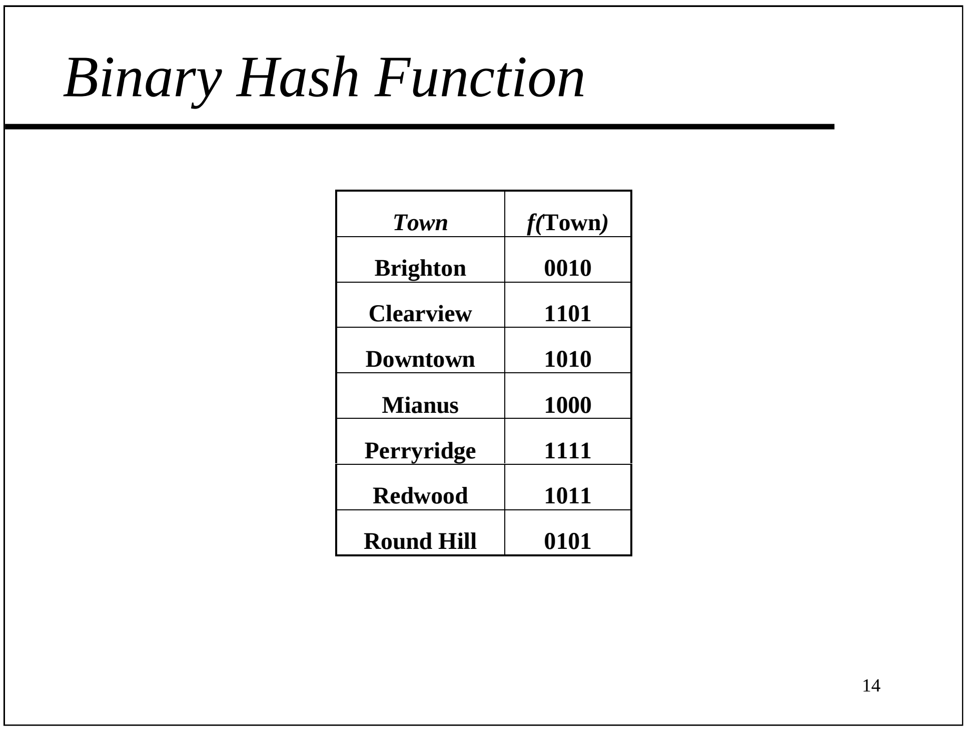
Two buckets need one binary digit, 0 or 1

Three/Four buckets need two binary digits, 00, 01, 10 or 11.

13

- Using binary addressing, the number of buckets that can be addressed may be doubled by adding one digit to the address.
- For instance, in the example above one binary digit can address two buckets, 0 and 1. Two binary digits can address four buckets, 00, 01, 10 and 11.
- Therefore, a hash function that grows and shrinks could be one that generates a binary code for each key value. The bucket address can be identified from the binary code.
- For example, if the *extendible hash* function generated a 32-bit code and the index currently has two buckets then the first binary digit should provide the bucket address. If the index currently has three or four buckets then the first two binary digits should provide the bucket address.

Ref: Silberschatz, sec 11.6; Elmasri, sec 5.9.3.

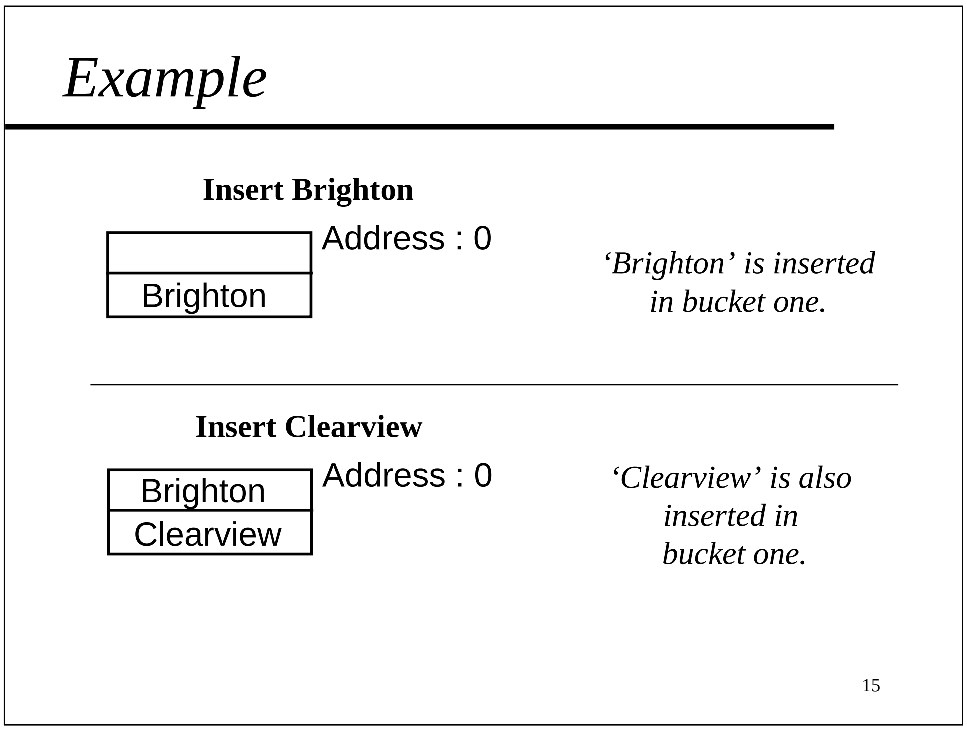


Binary Hash Function

Town	*f(*Town*)*
Brighton	**0010**
Clearview	**1101**
Downtown	**1010**
Mianus	**1000**
Perryridge	**1111**
Redwood	**1011**
Round Hill	**0101**

14

- Assume that it is possible to generate a binary value for any key value.
    - A hash function that generates a binary address can use the ASCII codes of the letters in the key value. For example, the ASCII code of ‘A’ is 65 or 1000001 (binary).
    - As with a static hash function, an ideal binary hash function must produce a uniform and random distribution of the keys.



Example

Insert Brighton

Brighton

Address : 0

'Brighton' is inserted in bucket one.

Insert Clearview

Brighton
Clearview

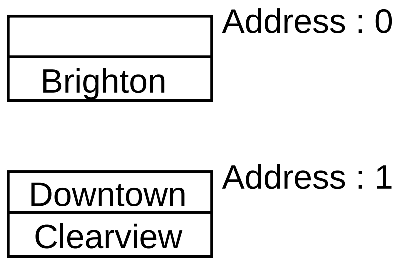
Address : 0

'Clearview' is also inserted in bucket one.

15

Example

Insert Downtown

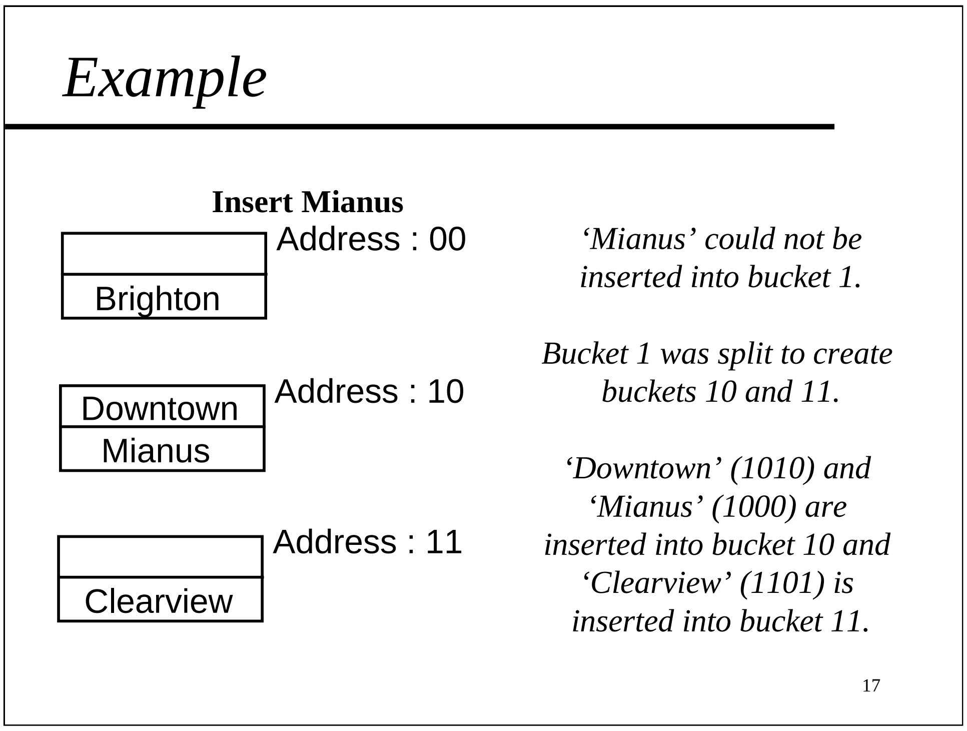


'Downtown' could not be inserted into bucket 0.

Bucket 0 was split to create buckets 0 and 1.

'Brighton' (0010) is inserted into bucket 0 and 'Downtown' (1010) and 'Clearview' (1101) are inserted into bucket 1.

16



Example

Insert Mianus

Address : 00

Brighton

Address : 10

Downtown

Mianus

Address : 11

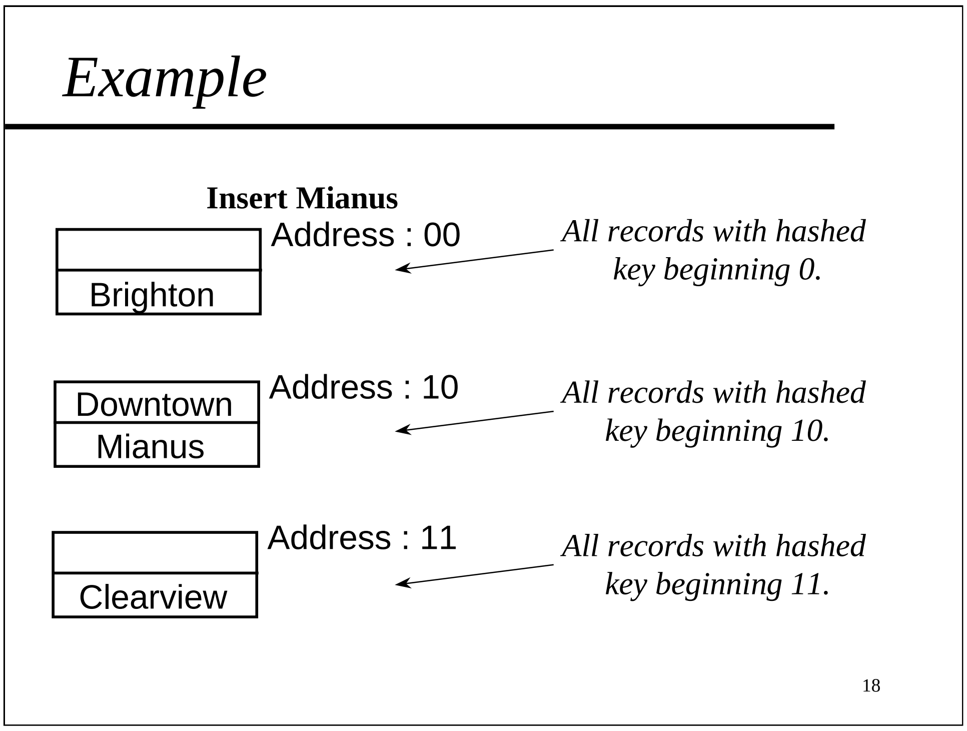
Clearview

'Mianus' could not be inserted into bucket 1.

Bucket 1 was split to create buckets 10 and 11.

'Downtown' (1010) and 'Mianus' (1000) are inserted into bucket 10 and 'Clearview' (1101) is inserted into bucket 11.

17

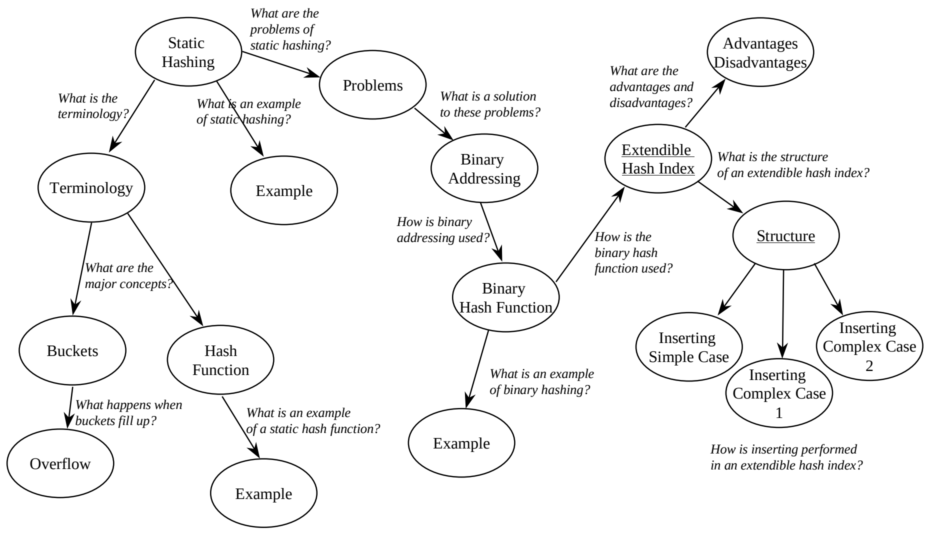


Example

Insert Mianus

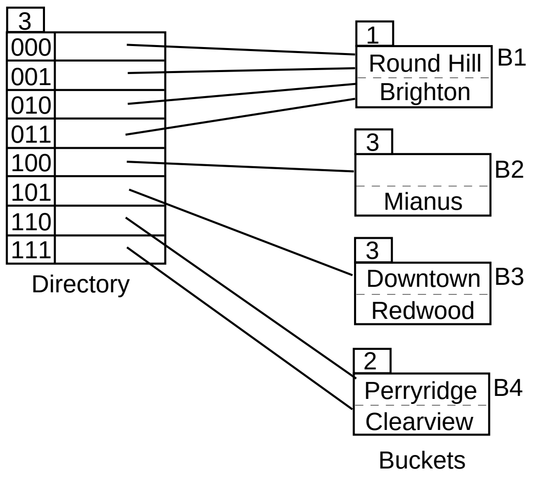
Address : 00
Brighton
All records with hashed key beginning 0.
Address : 10
Downtown
Mianus
All records with hashed key beginning 10.
Address : 11
Clearview
All records with hashed key beginning 11.
18

Overview


Static Hashing
What are the problems of static hashing?
Problems
What is the terminology?
What is an example of static hashing?
What is a solution to these problems?
Terminology
Example
Binary Addressing
How is binary addressing used?
What are the major concepts?
Binary Hash Function
How is the binary hash function used?
Buckets
Hash Function
What happens when buckets fill up?
What is an example of a static hash function?
What is an example of binary hashing?
Overflow
Example
Example
Advantages Disadvantages
What are the advantages and disadvantages?
Extendible Hash Index
What is the structure of an extendible hash index?
Structure
Inserting Simple Case
Inserting Complex Case 1
Inserting Complex Case 2
How is inserting performed in an extendible hash index?

19

Extendible Hash Index



20

- An extendible hash index consists of two parts:

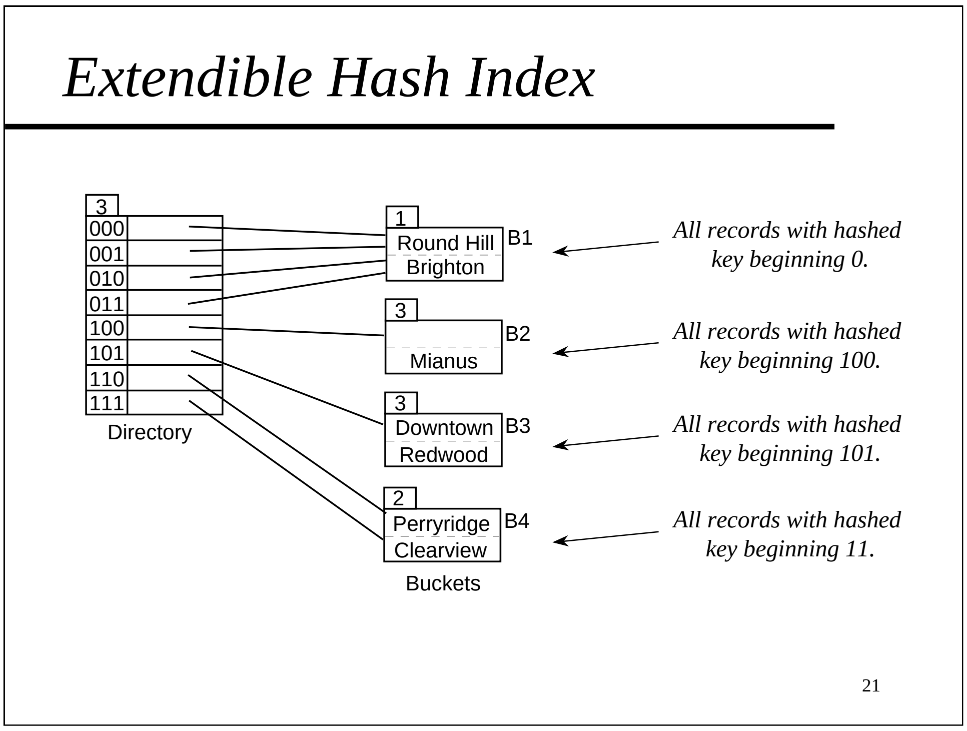
  Buckets Buckets are disc pages/blocks that are read and written by the system. The buckets have a physical address on the disc and contain a fixed number of records.

  Directory The directory indexes the buckets using a binary code. The directory consists of two parts:

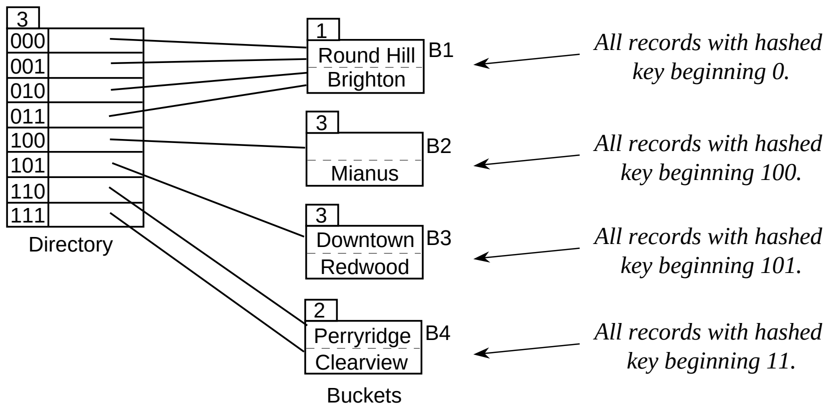
  1. A binary code which results from the hash function.
  2. A pointer to the bucket containing records matching the binary code.

  Two directory entries may point to the same record.
- To search for a record, for example, ‘Downtown’:
  1. Apply the hash function to ‘Downtown’, f(Downtown)=1010.
  2. Search the directory for 101.
  3. Read the bucket identified by the 101 pointer (B3).

Ref: Silberschatz sec 11.6.

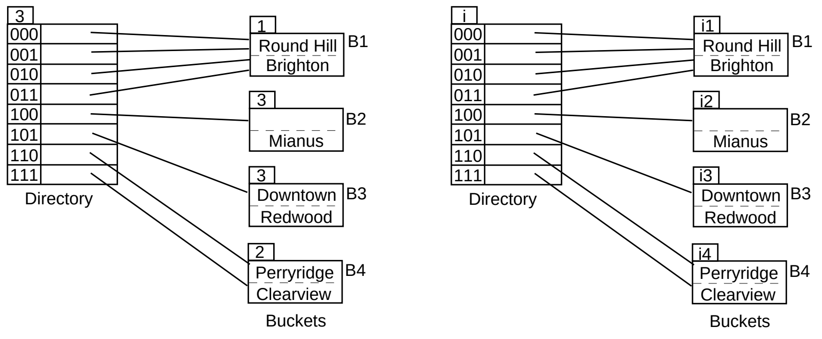


Extendible Hash Index


3
000
001
010
011
100
101
110
111
Directory
1
Round Hill
Brighton
B1
3
Mianus
B2
3
Downtown
Redwood
B3
2
Perryridge
Clearview
B4
Buckets
All records with hashed key beginning 0.
All records with hashed key beginning 100.
All records with hashed key beginning 101.
All records with hashed key beginning 11.

21

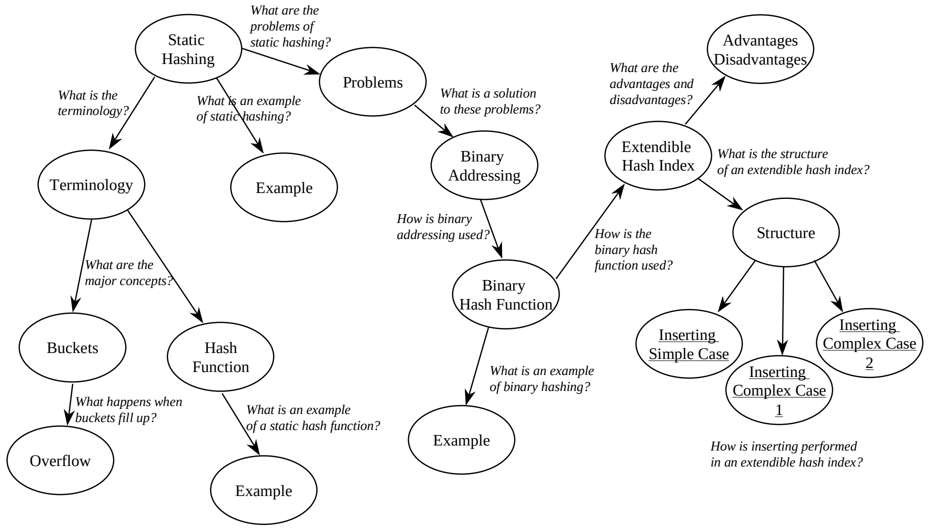
Structure



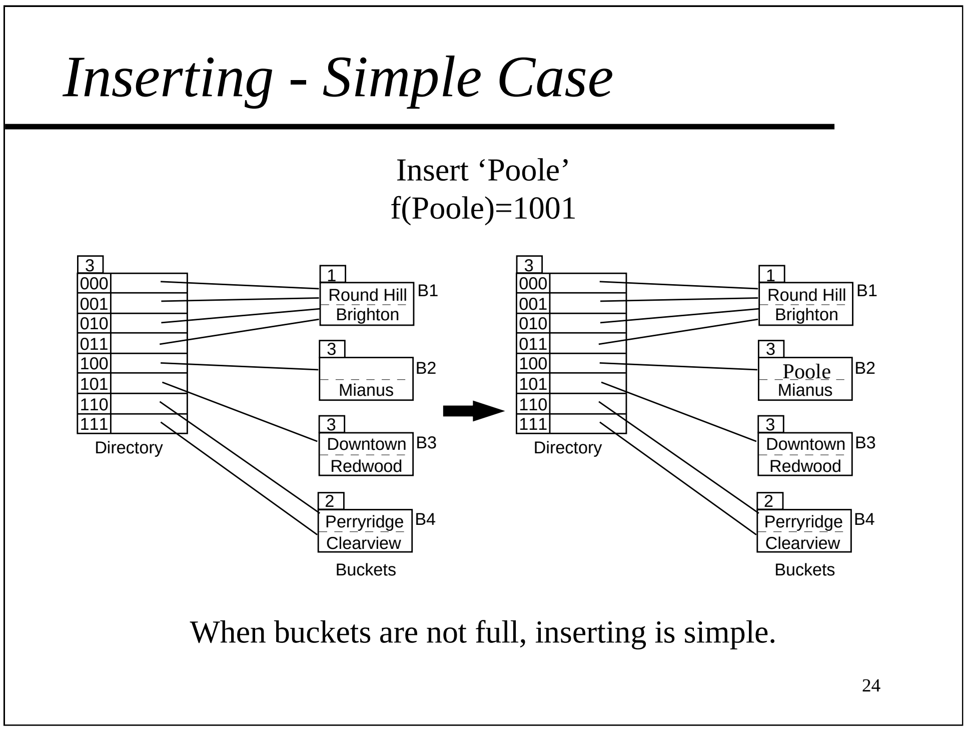
22

- Each entry in the directory contains a sequence of binary bits. The number of significant binary bits, that is, the number currently used in the index, is called *i*.
- Each bucket also has a significant number of bits called i_j. i_j represents the number of bits in the directory that are used to identify the bucket.
- The search algorithm uses the significant number of bits in the directory to determine which bucket to read. For example, to search for ‘Downtown’:
    1. Apply the hash function to ‘Downtown’, f(Downtown)=1010. The hash function may always return a fixed number of binary bits. (In this case, the hash function returns four bits.)
    2. Search the directory, which has three significant bits, for an entry matching 101 (the first three bits of ‘Downtown’).
    3. Read the bucket identified by the 101 pointer, that is, B3.

Overview

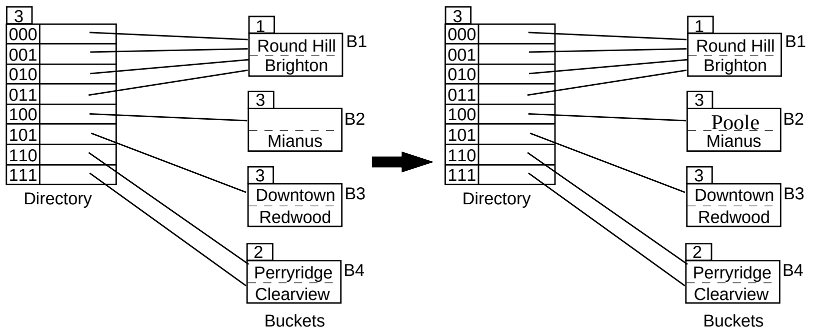

Static Hashing
What are the problems of static hashing?
Problems
What is the terminology?
What is an example of static hashing?
Terminology
Example
What are the major concepts?
Buckets
Hash Function
What happens when buckets fill up?
Overflow
What is an example of a static hash function?
Example
What is a solution to these problems?
Binary Addressing
How is binary addressing used?
Binary Hash Function
What is an example of binary hashing?
Example
How is the binary hash function used?
Extendible Hash Index
What are the advantages and disadvantages?
Advantages Disadvantages
What is the structure of an extendible hash index?
Structure
Inserting Simple Case
Inserting Complex Case 1
Inserting Complex Case 2
How is inserting performed in an extendible hash index?

23



Inserting - Simple Case

Insert ‘Poole’
f(Poole)=1001



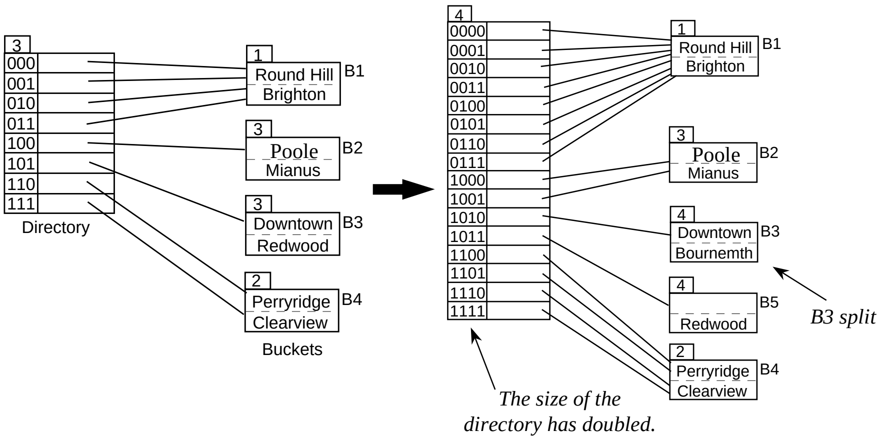
When buckets are not full, inserting is simple.

24

- When inserting a new record, a search is performed to locate the position for the record.
- If the bucket that should contain the record is less than full, then the record can be inserted into the bucket.
- The structure of the index does not change.
- In the example above, the key ‘Poole’ could be inserted into bucket B2 because B2 had a free space.

Inserting - Complex Case 1

Insert ‘Bournemouth’
f(Bournemouth)=1010



25

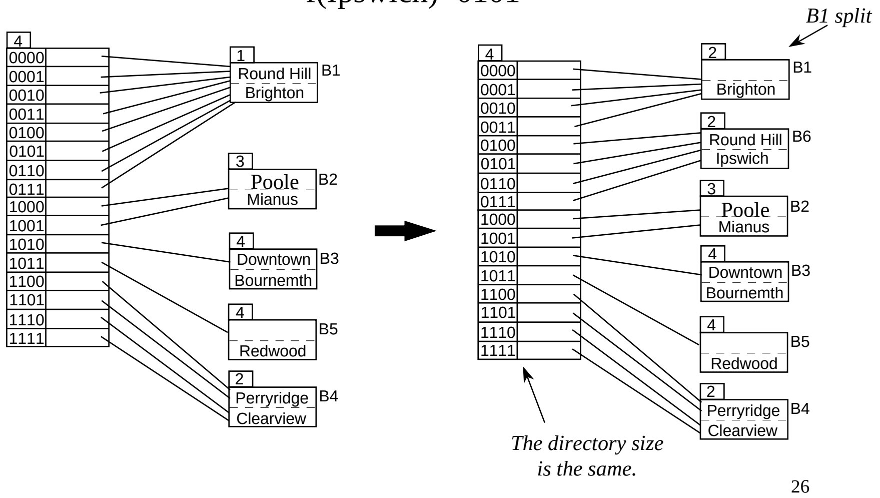
- In the example above, ‘Bournemouth’, which should be inserted into B3, could not be inserted because B3 was full.
- B3 has been split to created a new bucket B5.
- In the old index, only one pointer pointed to B3, that is, $i=i_j$ (3=3). *The number of significant bits required to identify the bucket was the same as the number of significant bits in the directory*.
- To increase the number of pointers in the directory, a new bit is added to the directory. This has the effect of doubling the size of the directory.
- The result of inserting ‘Bournemouth’ is that the number of significant bits in the directory is four. This means that there are twice the number of pointers.
- The contents of B3 have been redistributed between B3 and B5 according to their hashed values.
- The number of significant bits in B3 and B5 (i_1=3, i_5=3) is increased by one digit (i_1=4, i_5=4).

Ref: Silberschatz sec 11.6.

Inserting - Complex Case 2

Insert ‘Ipswich’

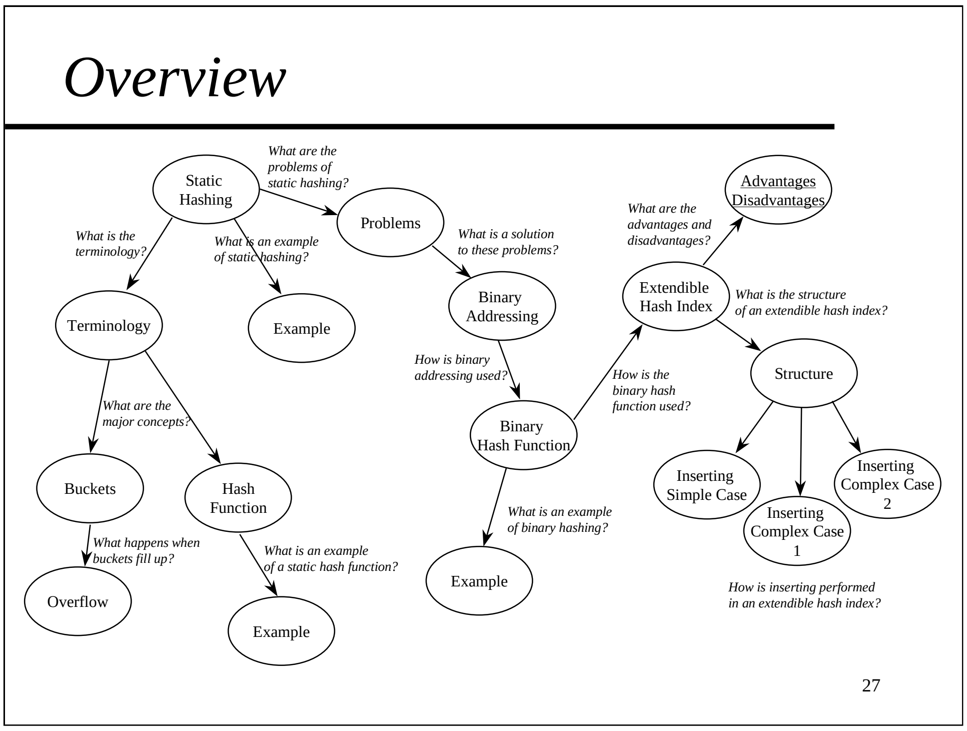
f(Ipswich)=0101



26

- The position for ‘Ipswich’ is in bucket B1.
- When ‘Ipswich’ is inserted into B1, B1 must be split because it is full. Splitting B1 creates B6.
- *However, the number of significant bits in B1, (i_1=1), is less than the number of significant bits in the directory, (i=4).* This means that there is more than one pointer pointing at B1.
- Therefore, instead of doubling the size of the directory, the pointers pointing at B1 can be redistributed between B1 and B6.
- The contents of B1 are also redistributed according to their hashed code.
- The number of significant bits in B1 and B6 (i_1=2, i_6=2) is increased by one digit (i_1=3, i_6=3).

Ref: Silberschatz sec 11.6.



Overview

Static Hashing
What are the problems of static hashing?
Problems
What is the terminology?
What is an example of static hashing?
Terminology
Example
What are the major concepts?
Buckets
Hash Function
What happens when buckets fill up?
Overflow
What is an example of a static hash function?
Example
What is a solution to these problems?
Binary Addressing
How is binary addressing used?
Binary Hash Function
What is an example of binary hashing?
Example
How is the binary hash function used?
Extendible Hash Index
What are the advantages and disadvantages?
Advantages Disadvantages
What is the structure of an extendible hash index?
Structure
Inserting Simple Case
Inserting Complex Case 1
Inserting Complex Case 2
How is inserting performed in an extendible hash index?
27

Advantages

- Performance does not degrade as file size increases
- Stores the minimum number of buckets
- Number of buckets grows/shrinks dynamically

Disadvantages

- The directory must be searched.
- The directory must be stored.