

# CSE 562 Database Systems

## Hashing and More

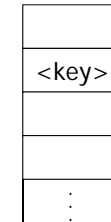
Some slides are based or modified from originals by  
*Database Systems: The Complete Book*,  
Pearson Prentice Hall 2<sup>nd</sup> Edition  
© 2008 Garcia-Molina, Ullman, and Widom

*cse@buffalo*

UB CSE 562 Spring 2009

## Hashing

key  $\rightarrow$  h(key)



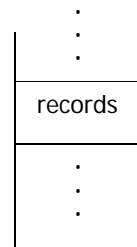
- hash function h(key) returns address of bucket or record

UB CSE 562 Spring 2009

2

## Two alternatives

(1) key  $\rightarrow$  h(key)

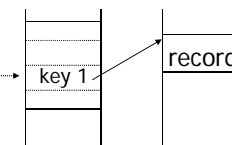


UB CSE 562 Spring 2009

3

## Two alternatives

(2) key  $\rightarrow$  h(key)



- Alt (2) for "secondary" search key

UB CSE 562 Spring 2009

4

### Example hash function

- Key = 'x<sub>1</sub> x<sub>2</sub> ... x<sub>n</sub>'  $n$  byte character string
- Have  $b$  buckets
- $h$ : add  $x_1 + x_2 + \dots + x_n$ 
  - compute sum modulo  $b$

UB CSE 562 Spring 2009

5

- ☒ This may not be best function ...
- ☒ Read Knuth Vol. 3 if you really need to select a good function

Good hash function: ☞ Expected number of keys/bucket is the same for all buckets

UB CSE 562 Spring 2009

6

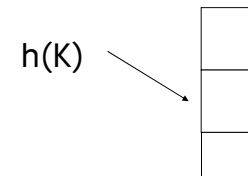
### Within a bucket:

- Do we keep keys sorted?
- Yes, if CPU time critical & Inserts/Deletes not too frequent

UB CSE 562 Spring 2009

7

Next: example to illustrate inserts, overflows, deletes



UB CSE 562 Spring 2009

8

### EXAMPLE 2 records/bucket

INSERT:

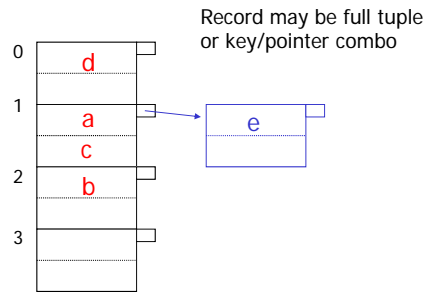
$h(a) = 1$

$h(b) = 2$

$h(c) = 1$

$h(d) = 0$

$h(e) = 1$



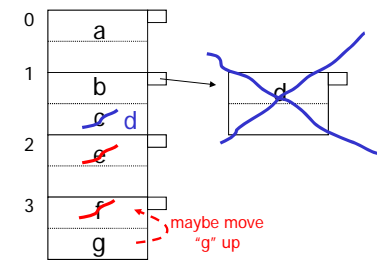
### EXAMPLE: deletion

Delete:

e

f

c



### Rule of thumb:

- Try to keep space utilization between 50% and 80%

$$\text{Utilization} = \frac{\# \text{ keys used}}{\text{total } \# \text{ keys that fit}}$$

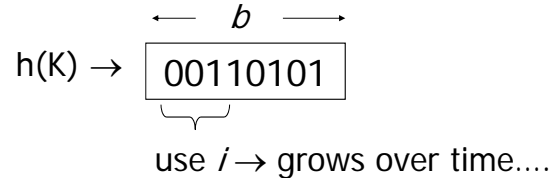
- If  $< 50\%$ , wasting space
- If  $> 80\%$ , overflows significant
  - ↳ depends on how good hash function is & on # keys/bucket

### How do we cope with growth?

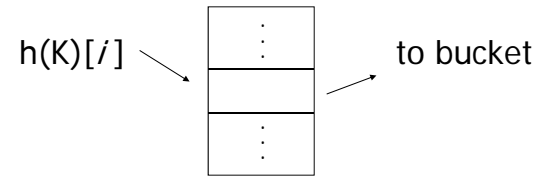
- Overflows and reorganizations
- Dynamic hashing
  - ↳ Extensible
  - ↳ Linear

Extensible hashing: two ideas

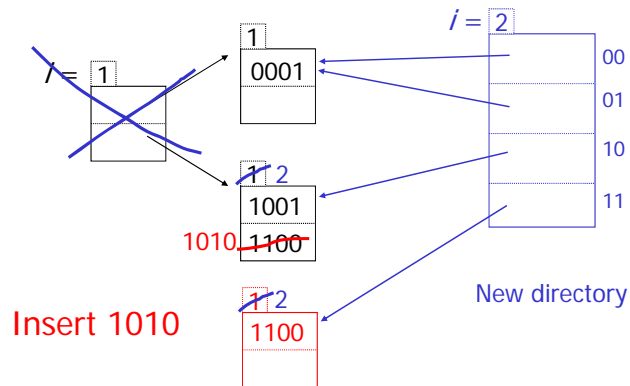
(a) Use  $i$  of  $b$  bits output by hash function



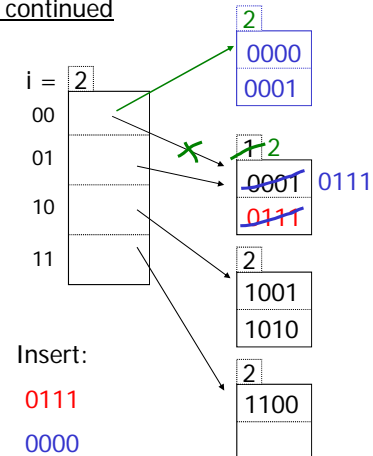
(b) Use directory

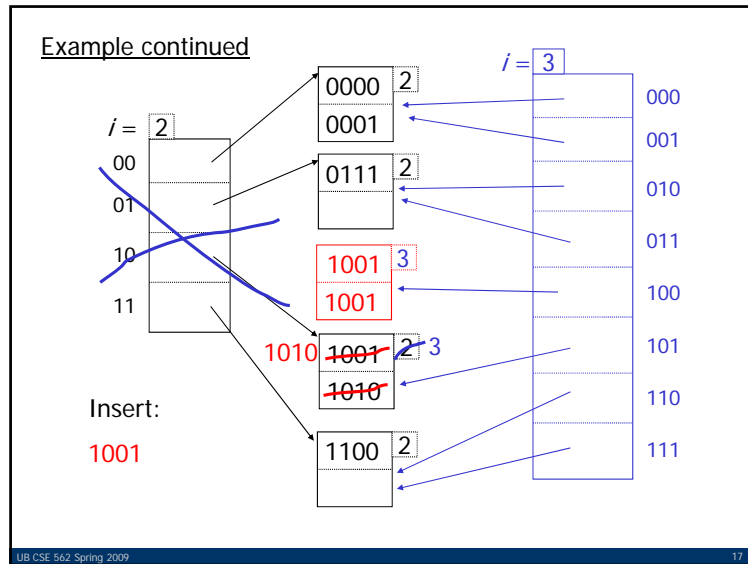


Example:  $h(k)$  is 4 bits; 2 keys/bucket



Example continued



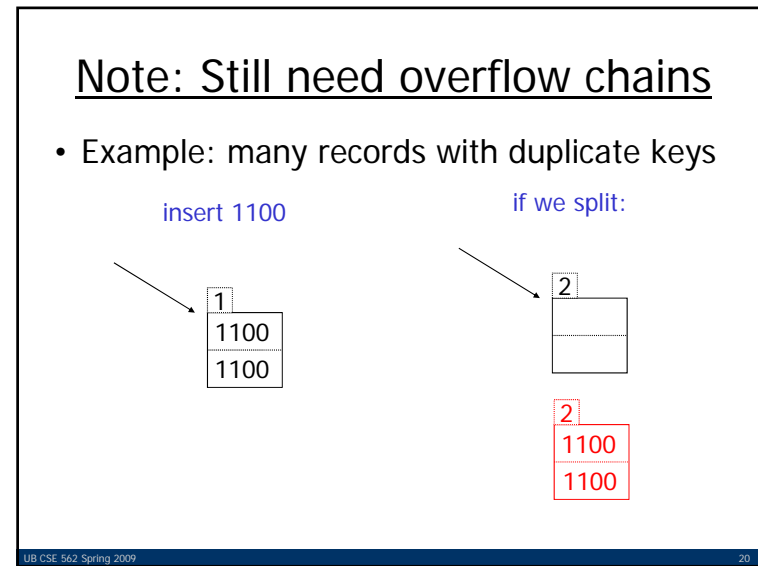


- ### Extensible hashing: deletion
- No merging of blocks
  - Merge blocks and cut directory if possible (Reverse insert procedure)
- UB CSE 562 Spring 2009 18

Deletion example:

- Run thru insert example in reverse!

UB CSE 562 Spring 2009 19



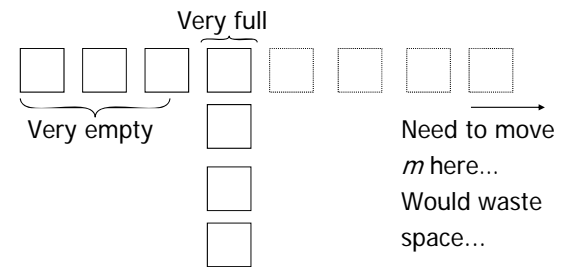




## Summary Linear Hashing

- ⊕ Can handle growing files
  - with less wasted space
  - with no full reorganizations
- ⊕ No indirection like extensible hashing
- ⊖ Can still have overflow chains

## Example: BAD CASE



## Summary

### Hashing

- How it works
- Dynamic hashing
  - Extensible
  - Linear

## Next:

- Indexing vs Hashing
- Index definition in SQL
- Multiple key access



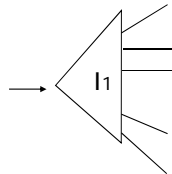
Note ATTRIBUTE LIST  $\Rightarrow$  MULTIKEY INDEX  
(next)  
e.g., CREATE INDEX foo ON R(A,B,C)

### Multi-key Index

Motivation: Find records where  
DEPT = "Toy" AND SAL > 50k

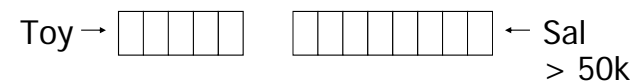
### Strategy I:

- Use one index, say Dept.
- Get all Dept = "Toy" records and check their salary



### Strategy II:

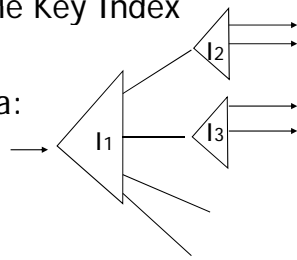
- Use 2 Indexes; Manipulate Pointers



### Strategy III:

- Multiple Key Index

One idea:



### Example

Art	
Sales	
Toy	

Dept  
Index

10k	
15k	
17k	
21k	

12k	
15k	
15k	
19k	

Salary  
Index

Example  
Record

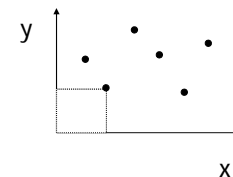
Name=Joe  
DEPT=Sales  
SAL=15k

For which queries is this index good?

- Find RECs Dept = "Sales"  $\wedge$  SAL=20k
- Find RECs Dept = "Sales"  $\wedge$  SAL  $\geq$  20k
- Find RECs Dept = "Sales"
- Find RECs SAL = 20k

Interesting application:

- Geographic Data



DATA:

<X1,Y1, Attributes>

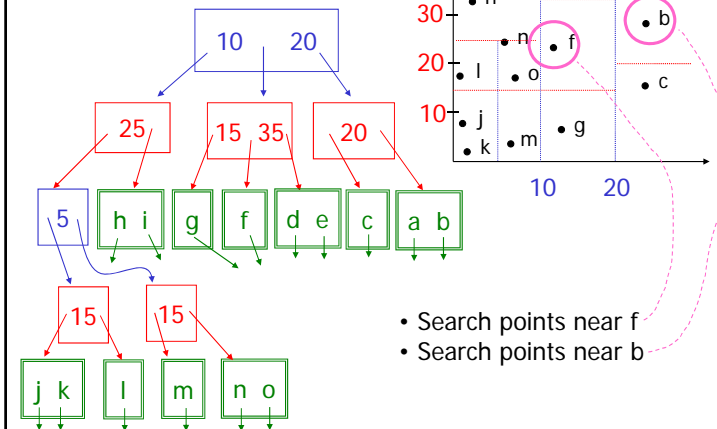
<X2,Y2, Attributes>

⋮

### Queries:

- What city is at  $\langle X_i, Y_i \rangle$ ?
- What is within 5 miles from  $\langle X_i, Y_i \rangle$ ?
- Which is closest point to  $\langle X_i, Y_i \rangle$ ?

### Example



- Search points near f
- Search points near b

### Queries

- Find points with  $Y_i > 20$
- Find points with  $X_i < 5$
- Find points "close" to  $i = \langle 12, 38 \rangle$
- Find points "close" to  $b = \langle 7, 24 \rangle$

- Many types of geographic index structures have been suggested
  - kd-Trees (very similar to what we described here)
  - Quad Trees
  - R Trees
  - ...

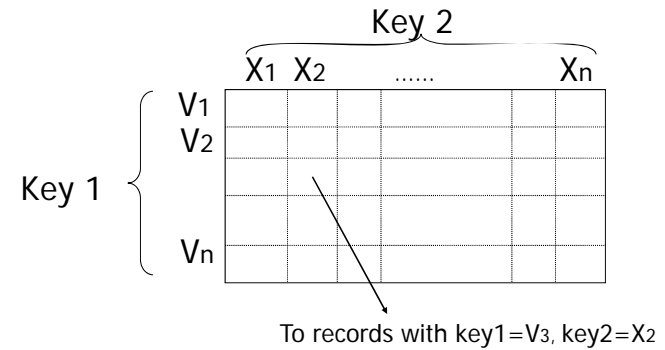
## Two more types of multi key indexes

- Grid
- Partitioned hash

UB CSE 562 Spring 2009

49

## Grid Index



UB CSE 562 Spring 2009

50

## CLAIM

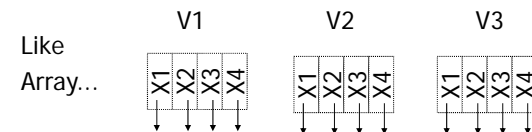
- Can quickly find records with
  - key 1 =  $V_i \wedge$  Key 2 =  $X_j$
  - key 1 =  $V_i$
  - key 2 =  $X_j$
- And also ranges....
  - E.g., key 1  $\geq V_i \wedge$  key 2  $< X_j$

UB CSE 562 Spring 2009

51

✉ But there is a catch with Grid Indexes!

- How is Grid Index stored on disk?



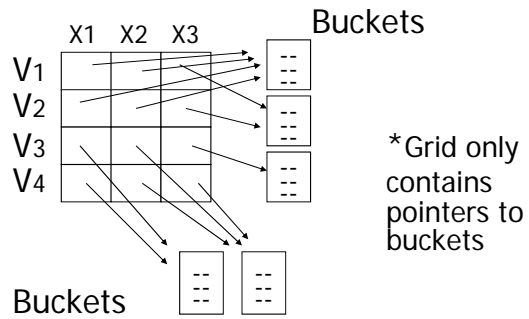
### Problem:

- Need regularity so we can compute position of  $\langle V_i, X_j \rangle$  entry

UB CSE 562 Spring 2009

52

## Solution: Use Indirection



UB CSE 562 Spring 2009

53

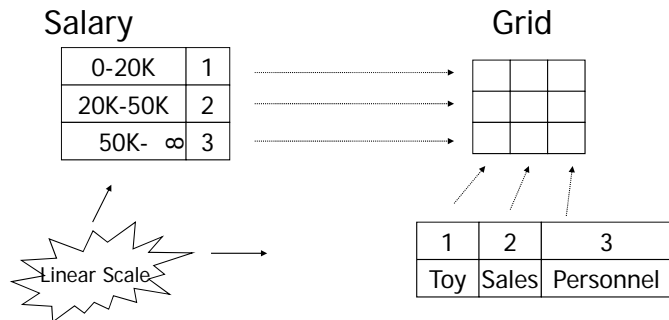
## With indirection:

- Grid can be regular without wasting space
- We do have price of indirection

UB CSE 562 Spring 2009

54

## Can also index grid on value ranges



UB CSE 562 Spring 2009

55

## Grid files

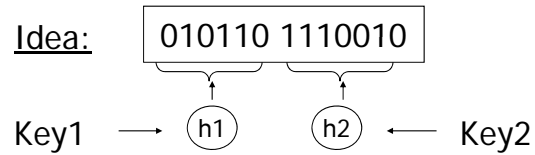
- + Good for multiple-key search
- Space, management overhead  
(nothing is free)
- Need partitioning ranges that evenly split keys

UB CSE 562 Spring 2009

56

## Partitioned hash function

Idea:



UB CSE 562 Spring 2009

57

EX:

h1(toy)	=0	000	
h1(sales)	=1	001	<Fred>
h1(art)	=1	010	
⋮		011	
h2(10k)	=01	100	
h2(20k)	=11	101	<Joe><Sally>
h2(30k)	=01	110	
h2(40k)	=00	111	
⋮			

Insert → <Fred,toy,10k>, <Joe,sales,10k>  
<Sally,art,30k>

UB CSE 562 Spring 2009

58

h1(toy)	=0	000	<Fred>
h1(sales)	=1	001	<Joe><Jan>
h1(art)	=1	010	<Mary>
⋮		011	
h2(10k)	=01	100	<Sally>
h2(20k)	=11	101	
h2(30k)	=01	110	<Tom><Bill>
h2(40k)	=00	111	<Andy>
⋮			

- Find Emp. with Dept. = Sales  $\wedge$  Sal=40k

UB CSE 562 Spring 2009

59

h1(toy)	=0	000	<Fred>
h1(sales)	=1	001	<Joe><Jan>
h1(art)	=1	010	<Mary>
⋮		011	
h2(10k)	=01	100	<Sally>
h2(20k)	=11	101	
h2(30k)	=01	110	<Tom><Bill>
h2(40k)	=00	111	<Andy>
⋮			

- Find Emp. with Sal=30k

look here

UB CSE 562 Spring 2009

60

h1(toy)	=0	000	<Fred>
h1(sales)	=1	001	<Joe><Jan>
h1(art)	=1	010	<Mary>
.		011	
h2(10k)	=01	100	<Sally>
h2(20k)	=11	101	
h2(30k)	=01	110	<Tom><Bill>
h2(40k)	=00	111	<Andy>
.			

- Find Emp. with Dept. = Sales look here

## Summary

### Post hashing discussion:

- Indexing vs. Hashing
- SQL Index Definition
- Multiple Key Access
  - Multi Key Index
  - Variations: Grid, Geo Data
- Partitioned Hash

## Reading Chapter 14

- Skim the following sections:
  - Sections 14.6.6, 14.6.7, 14.6.8
  - Sections 14.7.2, 14.7.3, 14.7.4
- Read the rest

## The BIG picture....

- Chapter 13: Storage, records, blocks...
- Chapter 14: Access Mechanisms
  - Indexes
  - B-Trees
  - Hashing
  - Multi key
- Chapter 15 & 16: Query Processing **NEXT**