Linear Hashing

Key Notes: I = MSB, N = #Buckets

Linear Hashing: ALGORITHM

Split if : (1) $\frac{\text{\# of full Buckets}}{\text{\# of Buckets}} > \text{load}$

(2) Overflow Block is full

when do you use an overflow block?

if Bucket is full but Not $\frac{\text{\# Buckets full}}{\text{\# Buckets}} > \text{load}$

$\implies$ Use overflow

using linear hashing, with an overflow of 0.9 add these values

- 0000
- 1010
- 1111
- 1101
- 0001
- 0000
- 1111

extra

Extendible hashing

Algorithm : take mod of value then
Convert Binary fit into bucket
If # in Bucket > records ⇒ MSB + 1

Using an extendible hashing thingy, write the structure:

$h(x) = x \bmod \_\_$ each bucket can hold $\underline{4}$
Indexes

2, 7, 3, 4, 5, 11, 17, 20, 22, 14, 9

me

B+ trees

Tips: n= value

n-1 = # values in box

So You can have n children

Minimum element : 2 for n=5

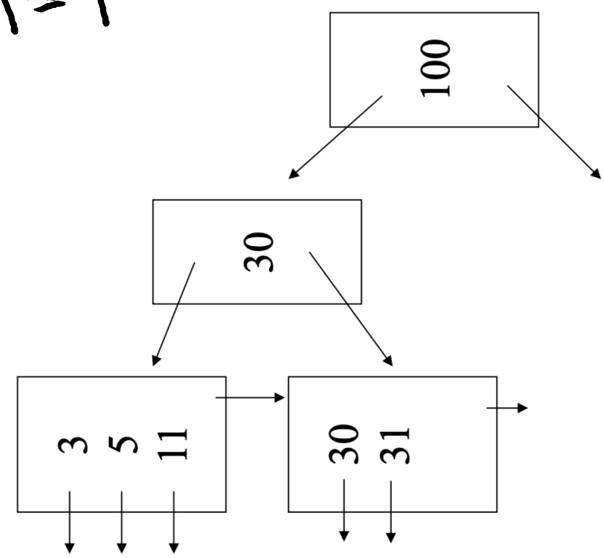
1 for n=4

How do we Know this?

$$\text{ceil} \left\lceil \frac{n}{2} \right\rceil - 1$$

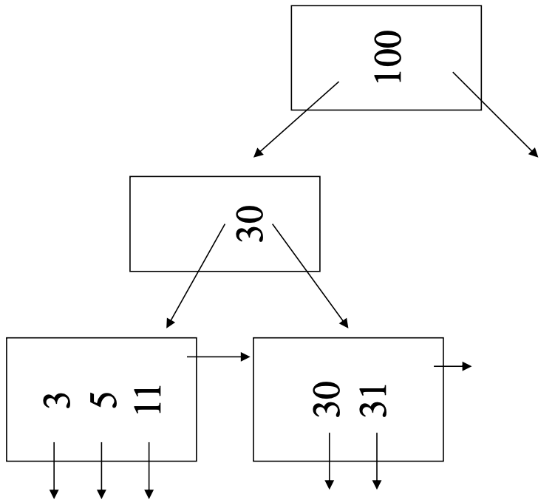
value in Branch found in leaf to the > side

n=4



Insert 31

Insert 7

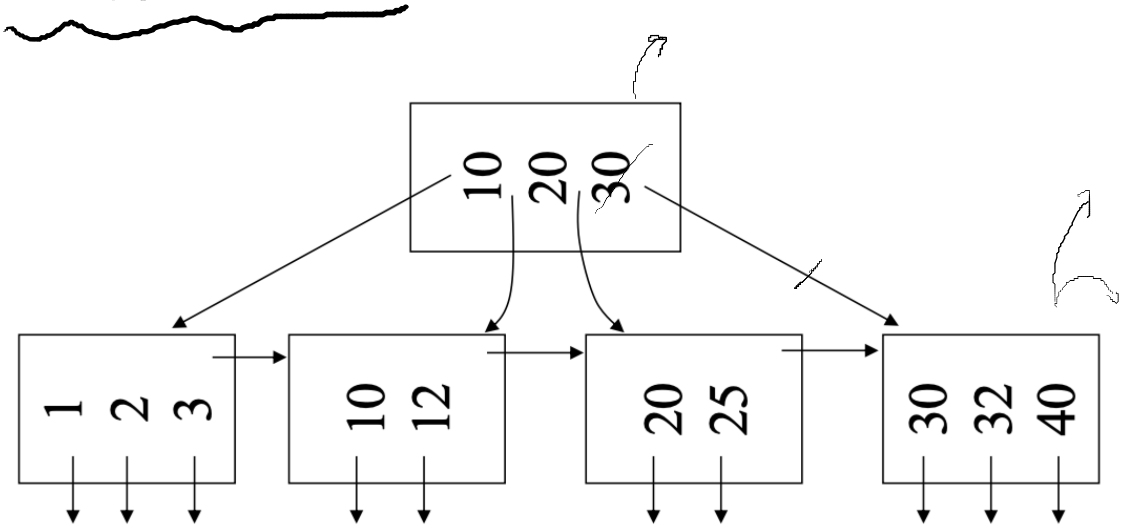


more nuanced examples in slides

english?

1 more important addition (new root)

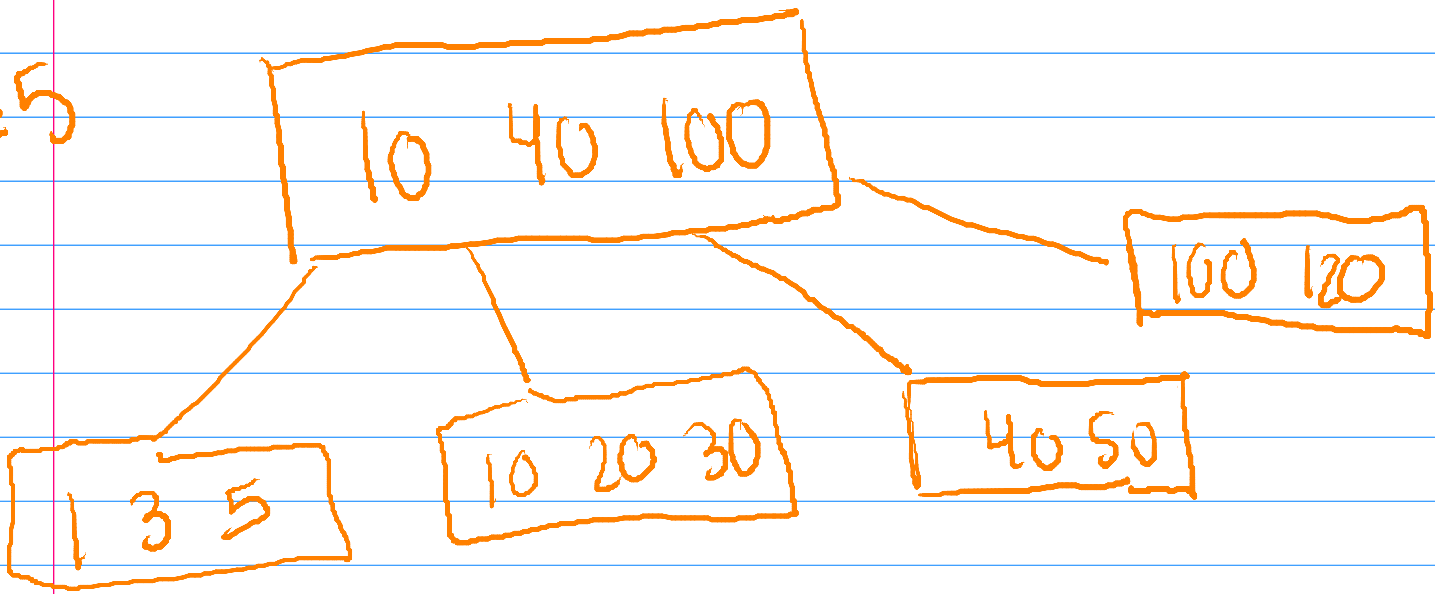
Insert 50



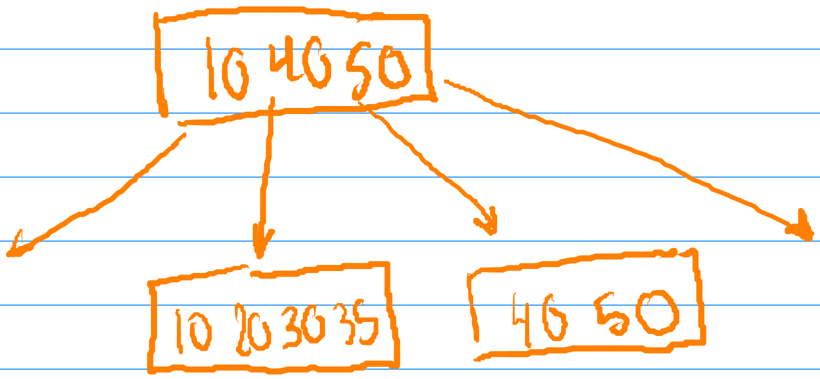
Deletion B+ tree

Delete 50

n=5



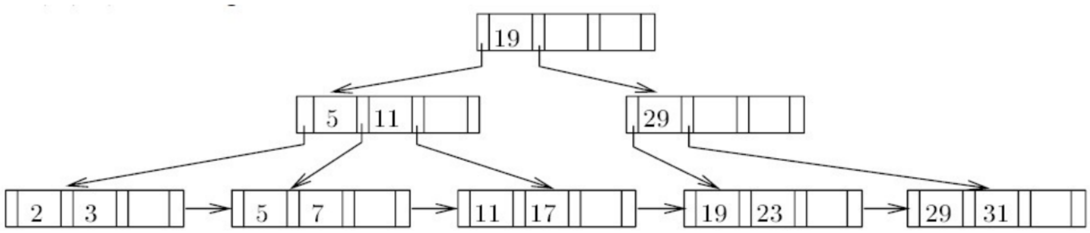
Delete 50



Find record with Key = 29

Junk drawer:

Assume that the following B+ tree is given:



find values from 5 to 17 inclusive

(A lot more examples in notes)

Some formulas: $\# \text{ leaf} \approx \left\lceil \frac{\text{records}}{\text{Keys}} \right\rceil$

$\# \text{ Blocks in level} = \left\lceil \frac{\text{leaves}}{\text{keys} + 1} \right\rceil$

untill 2

then +1

- Assume a relation R(A,B,C,D) with 1000 records and a B+tree index on C. There are 14 keys per index block. How many leaf blocks does the index have:

- Assume a B+tree index with 83 leaves and 6 keys per index block. How many levels does the tree have?

File & Storage Mgmt

⇒ read Slides

1. Consider a disk with a sector size of 512 bytes, 100 sectors per track. Given a rotational speed of 7200 revolutions per minute, what is the maximum rotational delay to the start of a sector? Assuming that one track of data can be transferred per revolution, what is the transfer rate?

?? max rotational delay??

$= 1/(\text{rev}/\text{min})$

?? transfer Rate??

→ (bytes/min)

$\Rightarrow \frac{\text{bytes}}{\text{sector}} \times \frac{\text{sectors}}{\text{track}} \times \frac{\text{revolutions}}{\text{min}}$

Normalization

Consider the following functional dependencies for the relation scheme R(A,B,C,D,E):

A → BC

CD → E

B →D

E → A.

* Rules in slides make this ez

Show that all the attributes of R are functionally dependent on each of the following sets of attributes:

i) A

ii) BC

Which FD do not hold in table?

Key things: mk = multi attribute primary Key

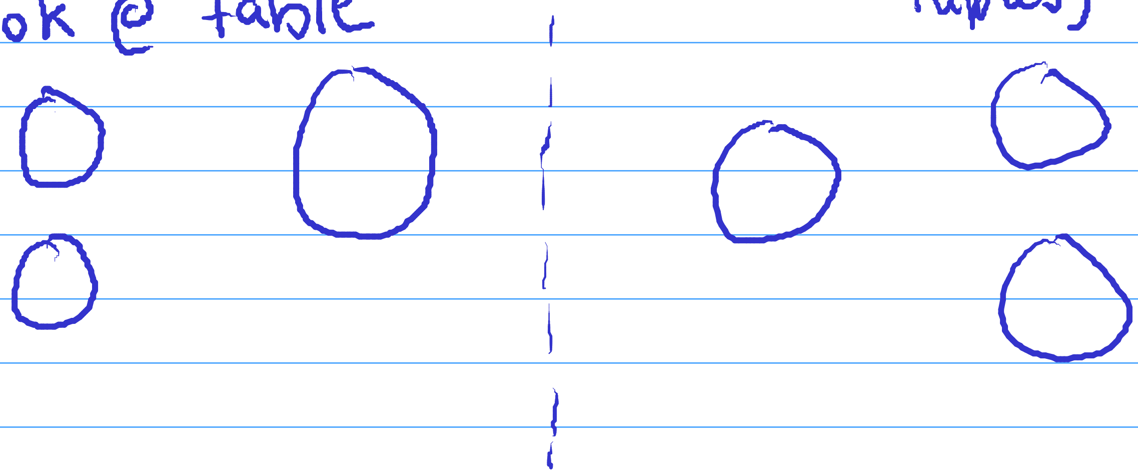
case 1:

MK → MK ✗ Bad

If MK, you dont have unique primary key (Not unique for all tuples)

case 2:

Look @ table



Tips Define SuperKey - A set where ...

Canadiate Key - A superkey that ...

Primary Key - A canidate key ...

Consider the following table called CSP

C#	CName	P#	PName	Type	Colour	S#	Sname	Date	Qty
100	dupont	10	wheel	a32	black	30	doe	10.10	25
100	dupont	20	tyre	b12	black	30	doe	10.10	30
200	martin	50	door	x21	white	10	minty	20.9	50
200	martin	20	tyre	b12	black	10	minty	20.9	50
300	dupont	70	bumper	a10	grey	30	doe	20.9	20

Of the following dependencies which definitely do not hold in CSP?

a. C# →P#

b. C# → CName

c. P# →Type

d. Colour → PName

e. S# → Qty

Consider again the CSP relation:

C#	CName	P#	PName	Type	Colour	S#	Sname	Date	Qty
100	dupont	10	wheel	a32	black	30	doe	10.10	25
100	dupont	20	tyre	b12	black	30	doe	10.10	30
200	martin	50	door	x21	white	10	minty	20.9	50
200	martin	20	tyre	b12	black	10	minty	20.9	50
300	dupont	70	bumper	a10	grey	30	doe	20.9	20

Suppose that (C#, P#, Date) is the primary key of CSP and that the following FDs hold in CSP:

C# → CName
P# → PName
S# → Sname
PName → Type
PName → Colour
C#, P# → S#

2NF Non key element fully dependant on primary key (not just a subset)

Show that CSP is not 2NF

Give lossless decomp :

* Algorithm

-

A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes that are not part of any candidate key) is dependent on any proper subset of any candidate key of the table. In other words,

If the proper subset of the candidate key determines a non-prime attribute, it is called partial dependency. The normalization of 1NF relations to 2NF involves the removal of partial dependencies. If a partial dependency exists, we remove the partially dependent attribute(s) from the relation by placing them in a new relation along with a copy of their determinant.

Consider again the CSP relation:

C#	CName	P#	PName	Type	Colour	S#	Sname	Date	Qty
100	dupont	10	wheel	a32	black	30	doe	10.10	25
100	dupont	20	tyre	b12	black	30	doe	10.10	30
200	martin	50	door	x21	white	10	minty	20.9	50
200	martin	20	tyre	b12	black	10	minty	20.9	50
300	dupont	70	bumper	a10	grey	30	doe	20.9	20

Suppose that (C#, P#, Date) is the primary key of CSP and that the following FDs hold in CSP:

C# → CName
P# → PName
S# → Sname
PName → Type
PName → Colour
C#, P# → S#

Give a lossless decomposition of CSP into 2NF relation schemes.

Definitions

1NF

2NF

3NF

check lossless decomp:

if intersection holds R_1 or R_2
⇒ then lossless

Not Covered 3NF, BCNF

- **SQL** ⇒ Do Practice in Slides

given database:

Consider the Sailors-Boats-Reserves database.

s (sid, sname, rating, age)
b (bid, bname, color)
r (sid, bid, date)

Write each of the following queries in SQL.

1. Find the colors of boats reserved by Albert.

2. Find all sailor id's of sailors who have a rating of at least 8 or reserved boat 103.

Relational Algebra

* Symbols :

  Π