Database Review 12

2 min

Joey-Issa-2024

Linear Hashing

Key Notes: I = MSB, N = #Buckets

Linear Hashing: ALGORITHM

0.9

i = MSB.

N = # Buckets

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

(2) Overflow Block is full

0 → 00
1 → 01
10
11

when do you use an overflow block?

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

Joey-Issa-2024

⇒ Use overflow

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

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

MSB = 1

0 → 0000, 1010
1 → 1111, 1101

full Buckets = 2

n = 2

2/2 = 1 > 0.9

0001 extra

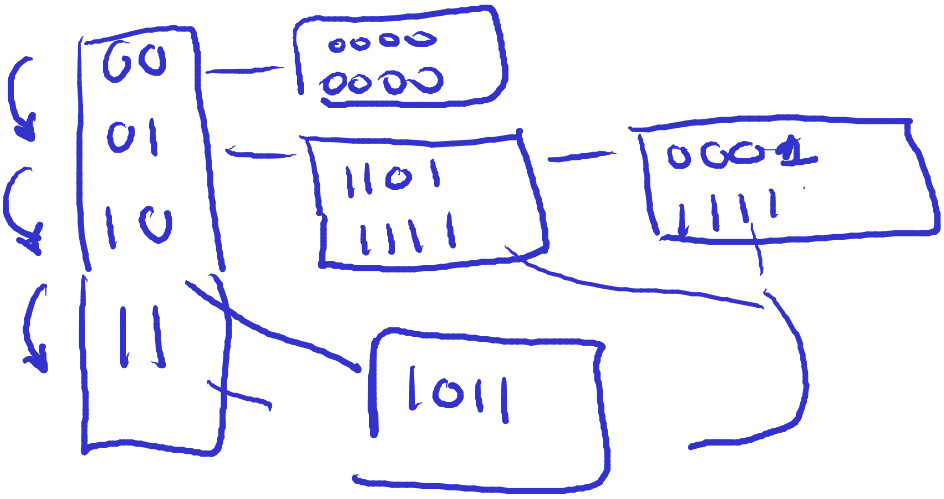
MSB + 1 = 2

00 → 0000, 0000
01 → 1101, 1111
10 → 1010
11 → 0001, 1111

n = 3

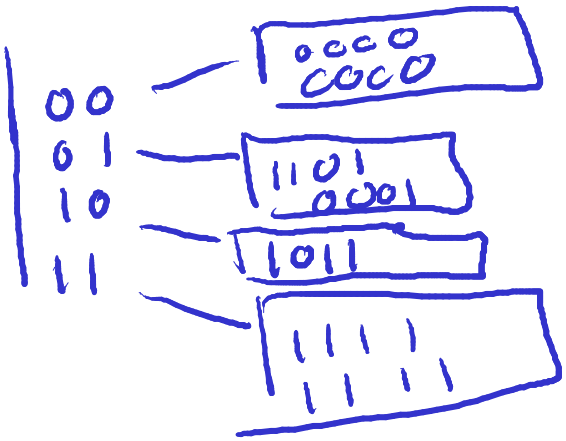
1/3 = 0.33

0.9 > 0.3


00
01
10
11
0000
0000
1101
1111
0001
1111
1011

MSB+1

=3


00
01
10
11
0000
0000
1101
0001
1011

mod X = Binary

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:

$h(x) = x \bmod 5$ (5 Indexes) each bucket can hold 4

Joey-Issa-2024

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

2 mod 5 = 2 010
7 mod 5 = 2 010
3 mod 5 = 3 011
11 mod 5 = 1 001

17 mod 5 = 2 = 010
20 mod 5 = 0 = 000
22 mod 5 = 2 = 010

MSB+1

0 → 2, 17, 22
→ 7, 20

1 → 3
→ 1

14 mod 5 = 4
= 100

9 mod 5 = 100
= 4

00 → 20, 9, 14
10 → 2, 7, 17, 22
01 → 11, 1
11 → 3

me

B+ trees

Tips: n = value

n-1 = # values in box

so You can have n children

n=4

Joey-Issa-2024

Minimum element : 2 for n=5
1 for n=4

How do we know this?

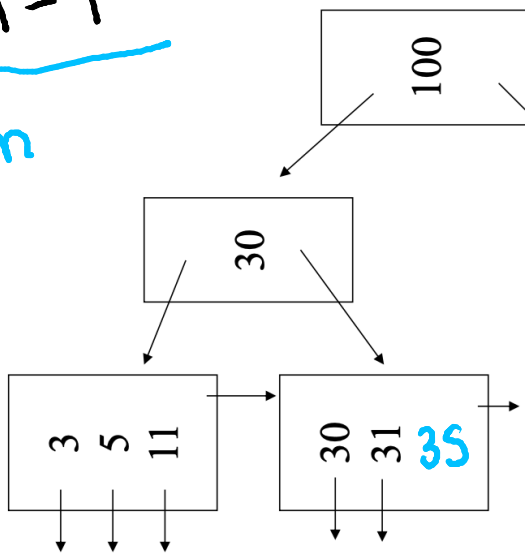
$$\text{ceil}\left\lceil \frac{n}{2} \right\rceil - 1$$

value in Branch found in leaf to the > side.

$$\text{ceil}\left\lceil \frac{5}{2} \right\rceil = 2$$

n=4

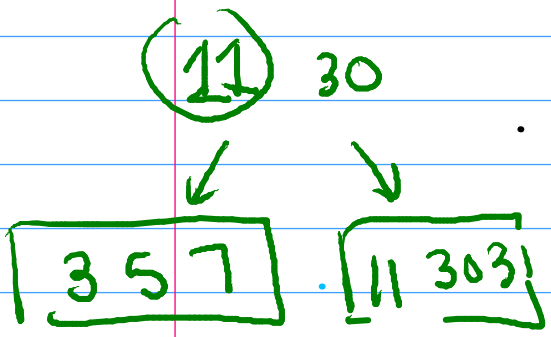
3 elem

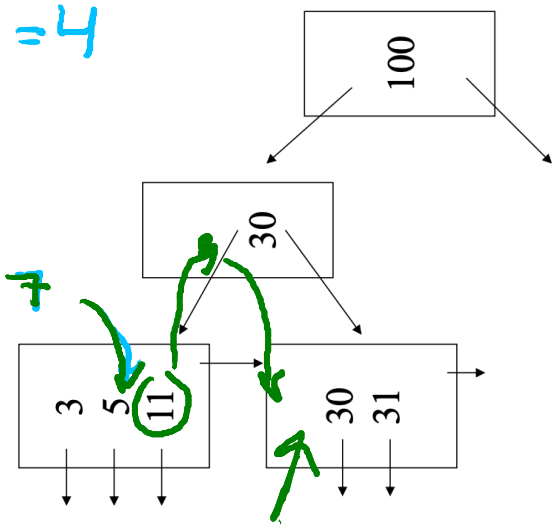


Insert 35

Insert 7

n = 4



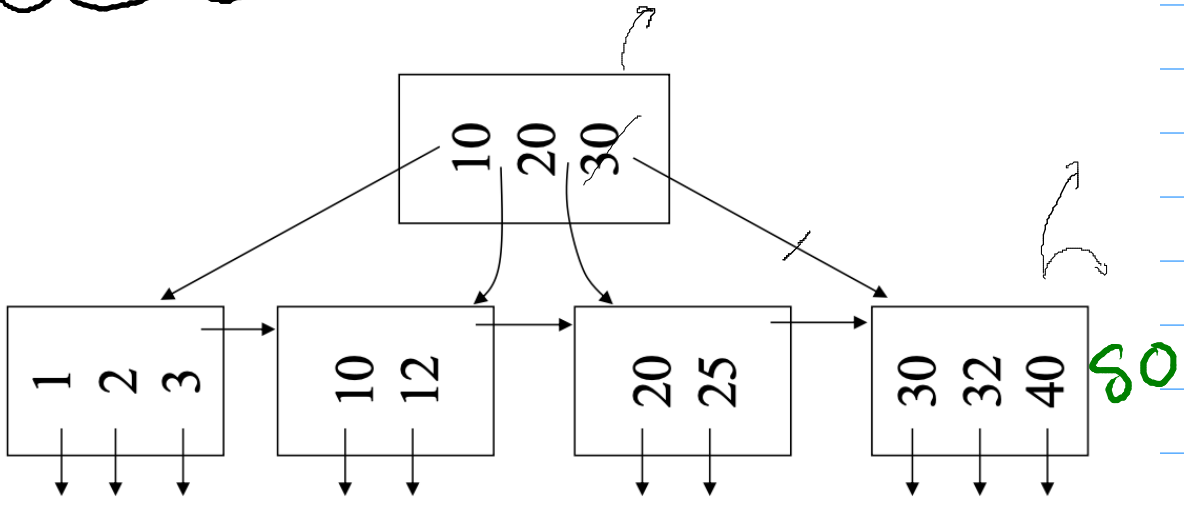


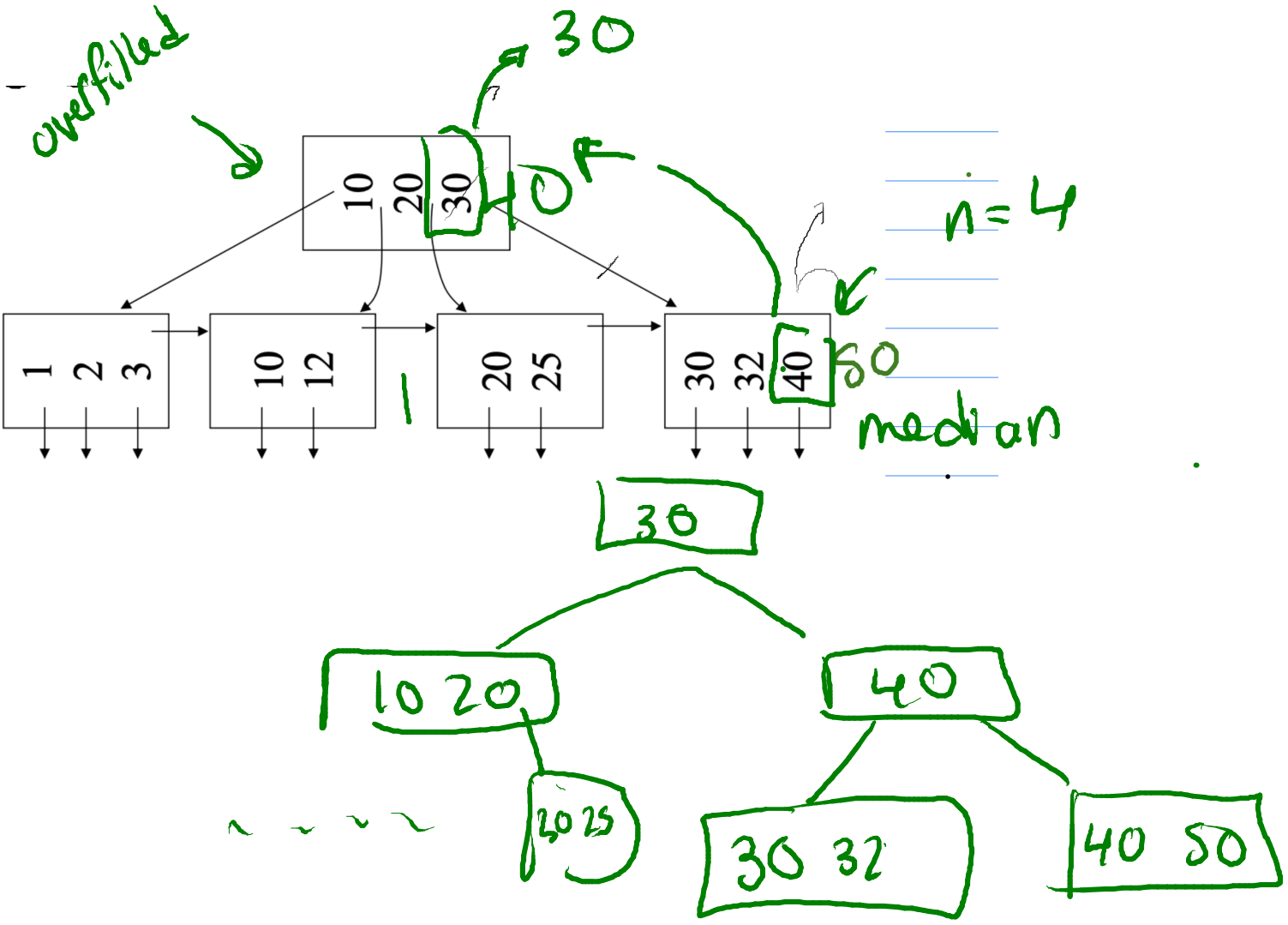
More examples in slides

1 more important addition (new root)

n = 4

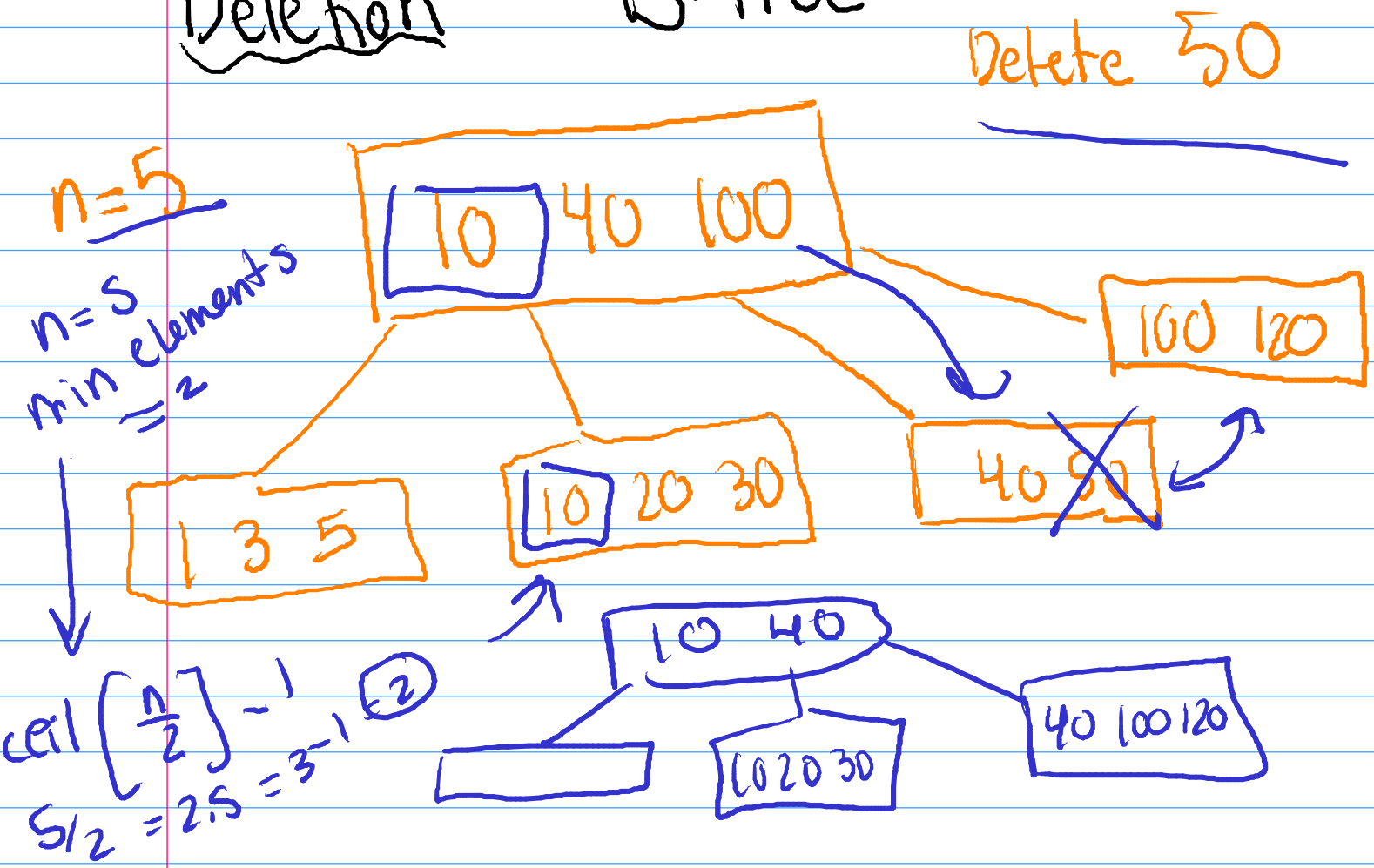
Insert 50



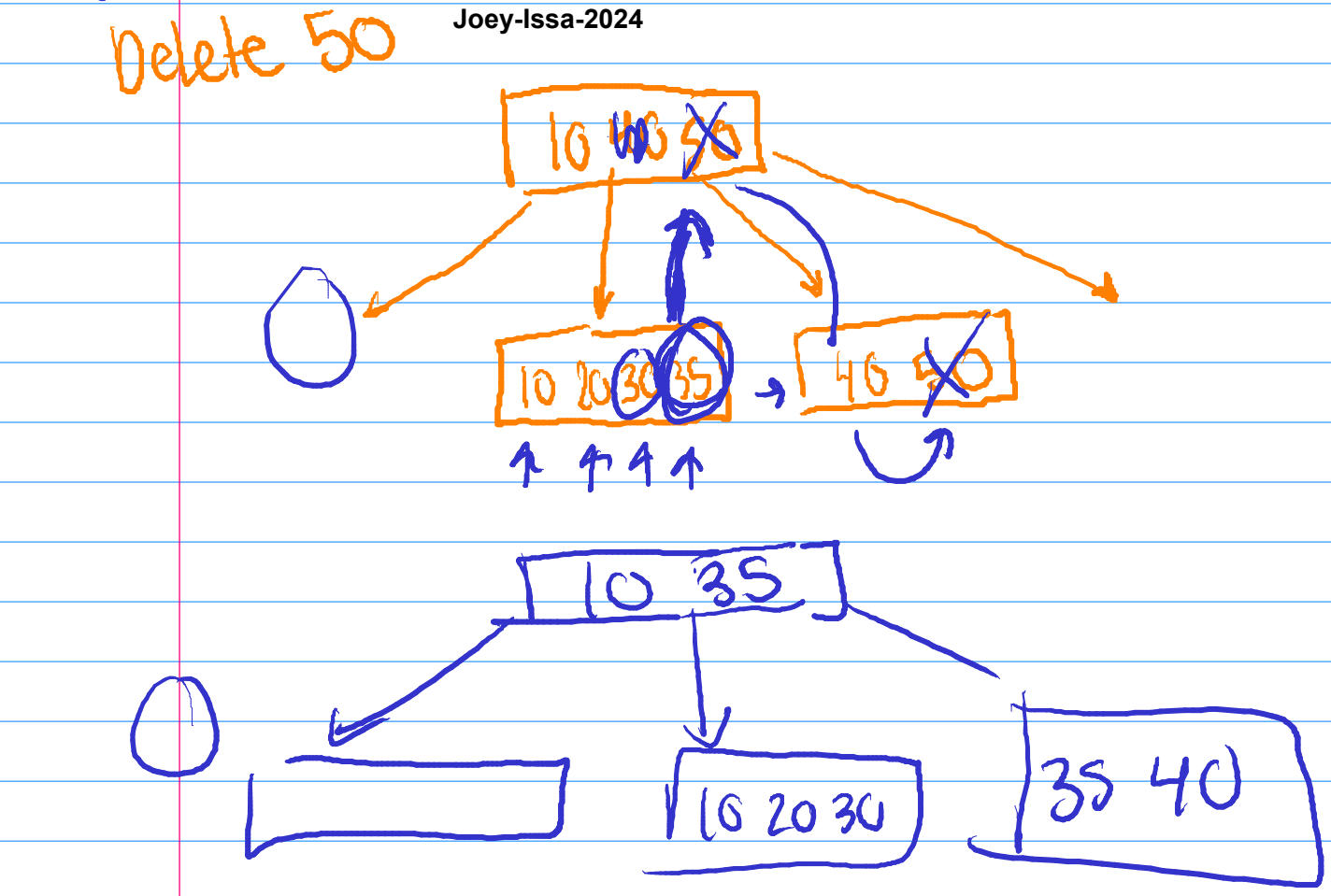

overfilled
30
10
20
30
40
n = 4
1
2
3
10
12
20
25
30
32
40
50
median

30
10 20
40
20 25
30 32
40 50

Deletion B+tree



Joey-Issa-2024

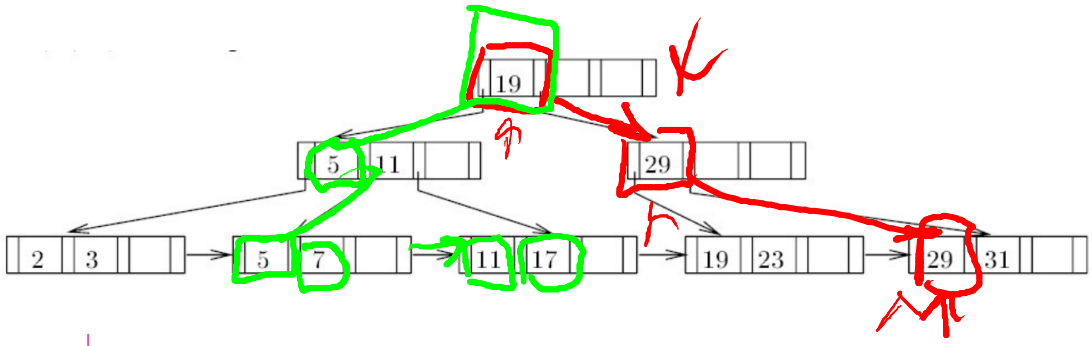


Find record with key = 29

Junk drawer :

Joey-Issa-2024

Assume that the following B+ tree is given:



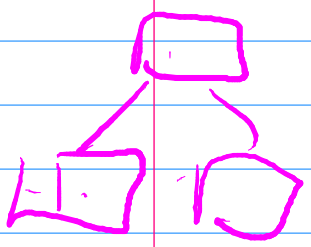
Find values from 5 to 17 inclusive

5 7 11 17

(A lot more examples in notes)

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

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



→ ↻ until 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:

leaf = $\lceil \frac{1000}{14} \rceil$ ← ceil round up

Joey-Issa-2024

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

83 leaves 6 key

#Blocks = $\lceil \frac{\text{leaves}}{\text{keys}+1} \rceil$

= $\lceil \frac{12}{7} \rceil$

= 2

= $\lceil \frac{83}{7} \rceil$

= 12

4 levels

1+2+12+83

File & Storage Mgmt

Slides Chapter

⇒ read slides

?? max rotational delay ??

Joey-Issa-2024

= 1/(rev/min)

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?

?? transfer Rate ??

00 → (bytes/min)

Rotational delay = 1 track / rev

= 1 / rev/min

= 1/7200

⇒ bytes/sector × sectors/track × revolutions/min × track/rev

bytes/min

tR = 512 bytes/sector × 100 sectors/track × 7200 × 1 track/rev

Normalization

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

A → BC

CD → E

B →D

E → A.

FD

* Rules in slides make this e2

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

i) A

ii) BC

A = {B, C, D, E, A}

A → E

E → A

A → A

AFD

A → BC

A → B, A → C

B → D, A → B

⇒ A → D

CD → # A → C, A → D

A → C, A → D

CD → E = A → E

Rules

- $A \to B$ & $B \to C$ then $A \to C$
- $B \subseteq A$, $A \to B$
- $A \to BC$ then $A \to B$ & $A \to C$
- $A \to B$ then $XA \to XB$
- $A \to B$ & $B \to C$ then $A \to C$
- $A \to B$ & $CD \to D$, $AC \to D$

* $CD \to A$, $A \to C$, $A \to D$
  - $A \to A$

Joey-Issa-2024

BC
$B \to D$

$CD \to E$
$CB \to E$
$E \to A$
$\to CB \to A$

Also FD

Which FD do not hold in table?

Key things: mk = multi attribute primary key

case 1:

$Mk \to MK$ [X] Bad

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

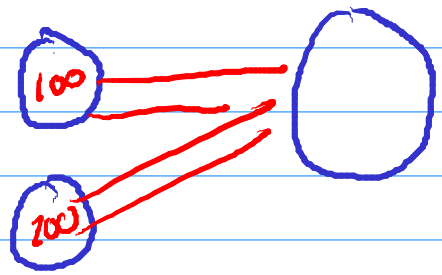
subset primary key

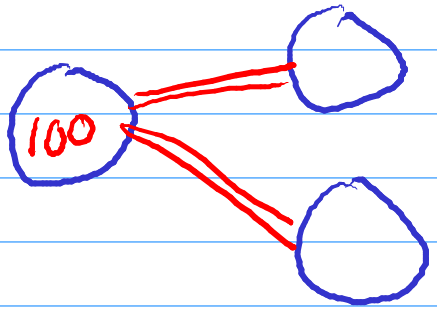
subset prim. key

case 2:

Look @ table

LS





Tips Define Superkey - A set where you can unique ID every element

Superkey = $\{A, B, C\}$

$\times \{A, B\}$ ← Canadiate Key - A superkey that ... does not $\subseteq$ inside that is also a superkey

Primary Key - A canidate key ... User defined

Joey-Issa-2024

Duplicated 100 → value

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# ✗ hold
b. C# → CName ✓
c. P# →Type ✓
d. Colour → PName ✗
e. S# → Qty

LS → RS
↑
LS = RS

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#

Not just a subset

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

Joey-Issa-2024

Show that CSP is not 2NF

CName only depends on a subset of a primary key

Give lossless decomp :

* Algorithm

- LS → RS R(LS, RS)
  LS, LS → RS R(LS, LS, RS)

element not found RS

→ whole primary key R(P.key, RS)

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#

R(C#, CName)

LS primary key

Duplicated

R(S, Sname)

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

R(P, PName, Type, Color)

R(C, P, S)

R(C, P, Date, Qty)

Definitions

Key

Phone # < 613

1NF — No atomic values
↳ value composed of ≥ two values (unique)

2NF — Non key element fully dependant on primary key ← any
↳ P.K → PS
unique element P.key
2NF

3NF — Key attribute depends on transitivity
A → B, B → C
A → C

BCNF
LS
Must key

Joey-Issa-2024

• check lossless decomp :

if intersection holds R_1, R_2
⇒ then lossless

Not Covered 3NF, BCNF

- SQL ⇒ Do Practice in Slides

given database : easy – intermediate → ~~hard~~

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.

SQL
RC
domain C

1. Find the colors of boats reserved by Albert.

$$\Pi_{color}(\sigma_{sname=Albert}(S \bowtie b \bowtie r))$$

```
Select color from s,b,r
where s.sid = r.sid and
      b.bid = r.bid
      and
s.sname = "Albert"
```

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

```
(Select sid from S where rating >8)
        U
(Select sid from r wher bid =103)
```

$$\Pi_{sid}((\sigma_{rating>8} S) \cup \Pi_{sid}(\sigma_{bid=103}))$$

Relational Algebra

* Symbols :

Π project σ select

ex. "σ dept_name = "Physics" (Instructor)"

Natural join r ⋈ s

Ex: Π cus_name, branch_name (deposit ⋈ account)

- result removes duplicates (if both tables have them)
- extending table (combining them)

Relations r, s:

r

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

s

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

if no match between tables it gets ignored (only include it if it matches row from another table)

r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

* Key is: column in R & S must have same name.

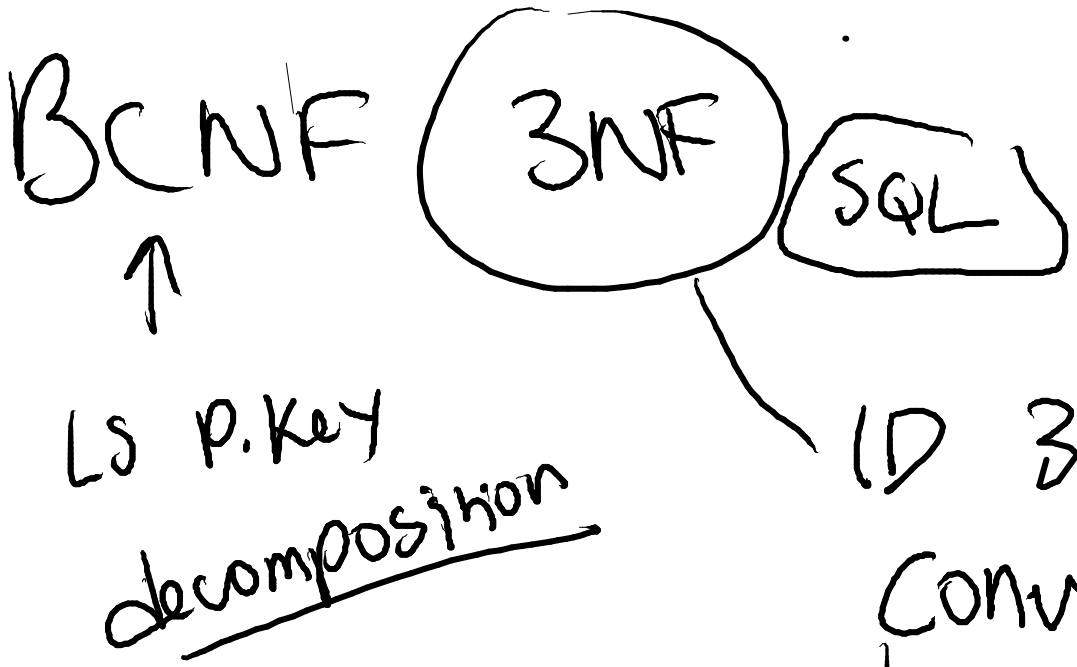
Joey-Issa-2024

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
δ	2	β	b

⋈

B	C	E
1	α	α
3	α	β
2	b	δ
3	b	ε

A	B	C	D	E
a	2	a	a	a
B	2	Y	a	c
B	2	Y	b	6



FD → what is FD?
normal form?

$2F3^{+}$ → Decomposed relations

B+ tree → inserting + Delete

Liar extendible hashing
• ording indexes, insert, overflow