72+312+152+35

Page No.
Date

DBMS :-

→ Issues with file system :-

1. Data maping & access
2. Redundancy of data
3. Inconsistency of data
4. Unauthorised access of security
5. No concurrent access
6. No backup & recovery.

→ Why DBMS is required ?

→ Overcome drawbacks of file system.

optimizes :-

- → storage of large data
- → Faster retrieval of data.

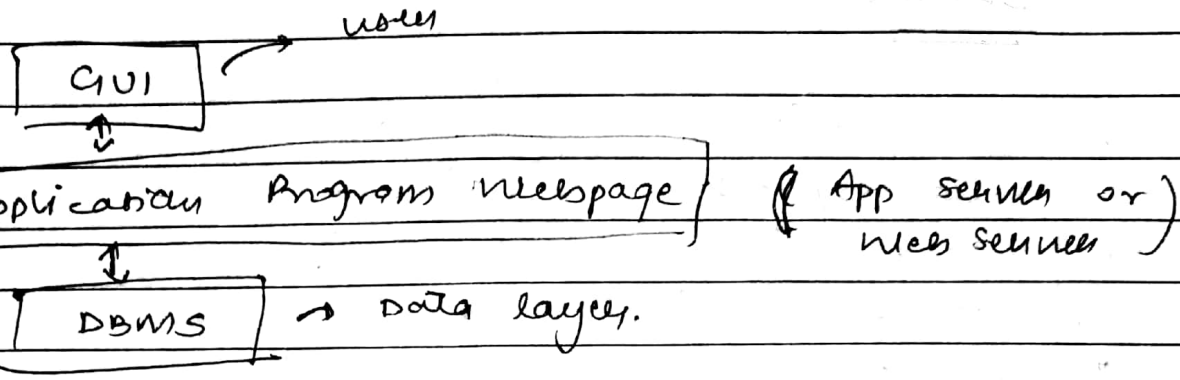
Characteristics :

→ Real world entity
→ Relation based tables
→ Less Redundancy
→ Query Language
→ consistency
→ Concurrent Access
→ Security

Architecture of DBMS :-

→ main advantage → we can access different modules independently.

- 3 tier Arch :-

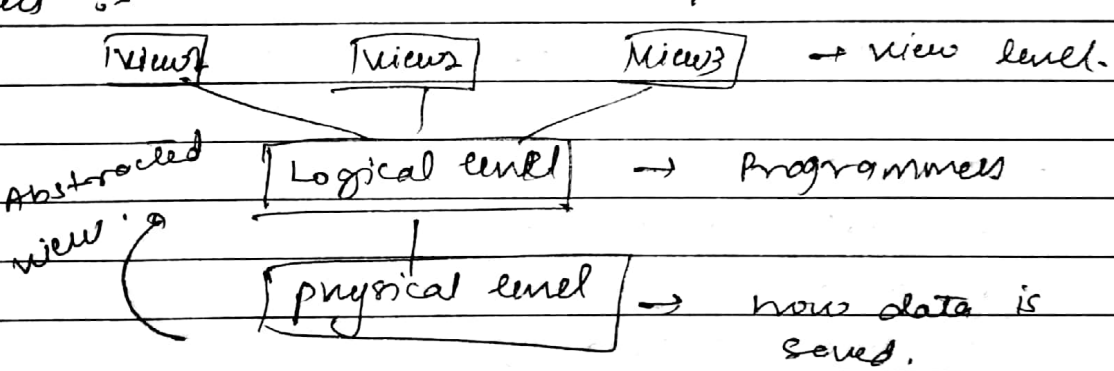


Adv :- 1. Easy to maintain & modify
2. Security
3. Performance.

Disadv :- Complex

- Data Abstraction :-

Levels :-



- Instance : Saving data in database at any point of time is called instance.

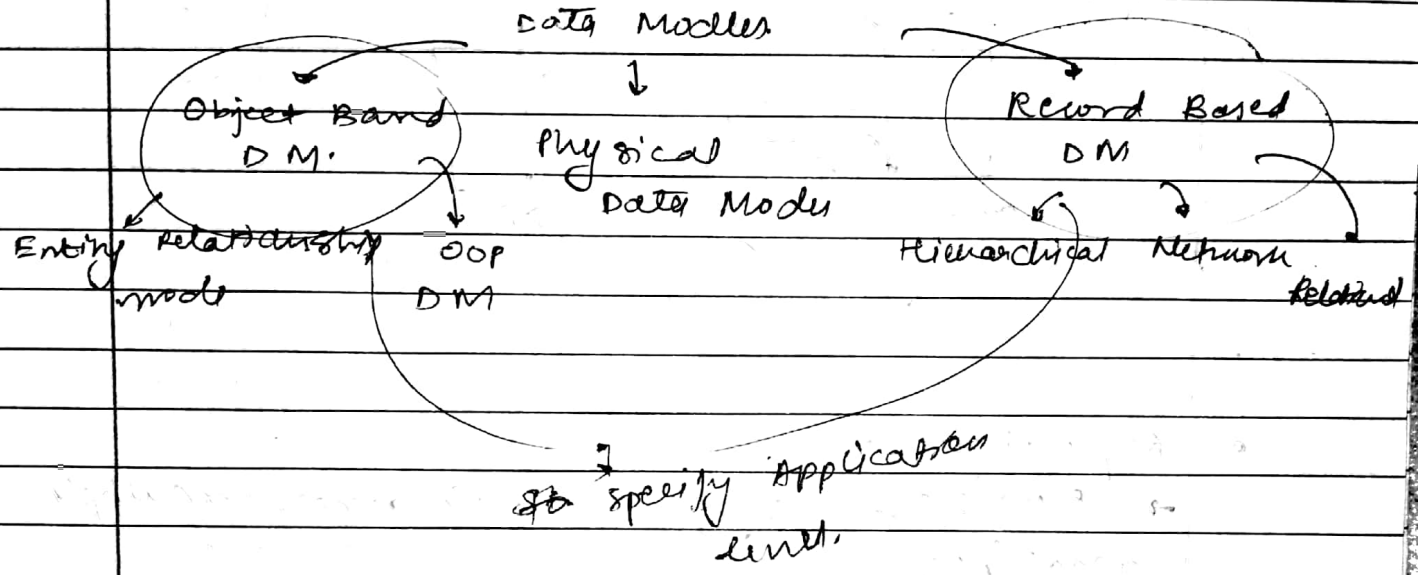
- Schema :- → Data structures.
→ Physical schema : Design of physical level.
→ logical schema : Defines Logical level.
→ View schema : How the data is showed to the end user.

Page No.
Date

→ Physical Data Independence :- changing the physical level will not effect the logical level..

Data Models :-

Describes real database.

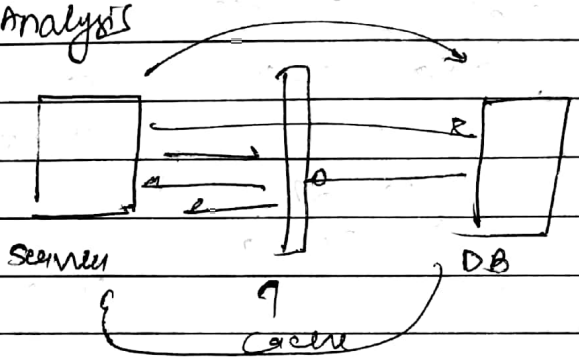


Cron Job

Page No.
Date

System Design :-

1. Requirements collection.
2. Architecture Design → what all components.
3. Final Analysis



- Replication :-
  → Frequently copying the data across multiple machines.
  → Post replication, multiple copies of the data exists across machines.

- Vertical scaling → Adding more servers. (Horizontal scaling)
  ↓
  Increasing Resources, (RAM, CPU, storage).

- Sharding : Sharding refers to splitting the very large database into smaller, faster & more manageable parts called data shards.

Availability
↑
CAP Theorem. Not possible to simultaneously guarantee all of the following
Consistency
Partition Tolerance.

Page No.
Date

Steps to approach :-

1. Feature expectations
2. Estimations
3. Design Goals.
4. Skeleton of the design
5. Deep dive.

Constraints :-

→ Primary key → ~~unique + not null.~~
→ Composite key
→ Unique key → can be Null. (but only 1).
→ Foreign key
→ check constraint
→ Not Null.

NORMALIZATION :-

→ Insertion Anomaly
→ Deletion Anomaly
→ Updation Anomaly.

Functional Dependency:

→ It is a relationship that exists between multiple attributes of a relationship.
→ Dependent → when one attribute uniquely identifies another attribute of the same table..

Axioms :

• ABC → AB → ~~W→Z~~ W → Z X → Y ⇒ WX → YZ
Reflexive.

4104-8351484-2793947

Page No.
Date

Candidate key should be minimal.

1NF :-

→ There cannot be multivalued or composite attributes

→ Each record needs to be unique.

2NF :-

→ Table should be in 1NF (P → N.P)

→ There should be no Partial dependency.

Partial dependency

X → a

1) X is a proper subset of some candidate key & a is non-(prime) or non key attribute.

↓ Part of any candidate key.

3NF :- (N.P → N.P)

→ Table should be in 2NF

→ No transitive dependency

X → a

↑ both are non prime

BCNF :- X → Y

↑

super key.

Page No.
Date

Search engine :-

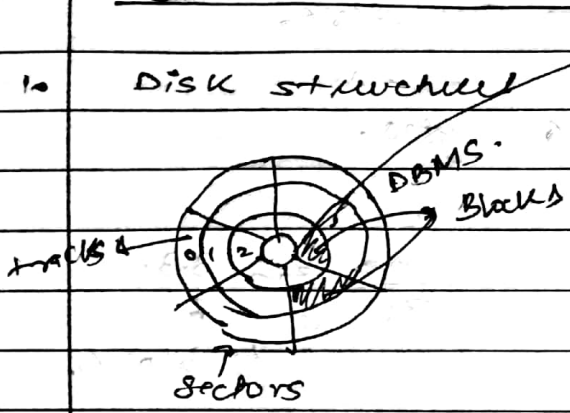
→ Indexes of other websites / webpages.

→ crawling / spiders → web scraping.

Page No.
Date

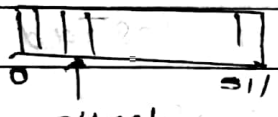
B-Trees & B+-Trees :-

1. Disk structure



Block address → track number, sector number
(512 bytes)

→ I/O read & write in terms of blocks



spinning → sectors are changed
moving → tracks are changed.

2. How data is organised.

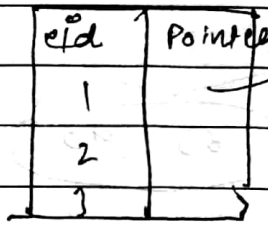
Employee
eid - 10
name - 50
dep - 10
section - 6
add - 50
128

512B = 4
4 records per block.

100/4 = 25 blocks in total.

- So the data is stored in 25 blocks

Index



Page No.
Date

* Index is also stored on the disk.

let pointer is 6 bytes.

10 + 6 = 16

$\frac{512}{16} = 32$ $\frac{100}{32} \approx 3.3$.

4 blocks.

So to access some data we need to search only 4 blocks instead of 25.
+
1 block of table.

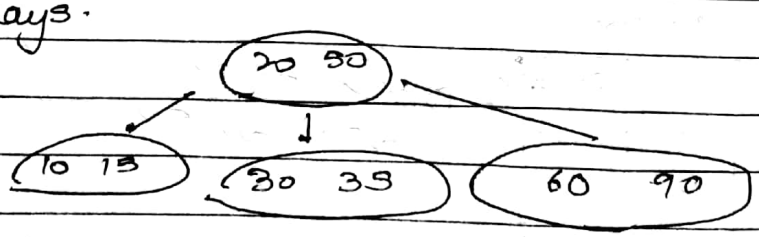
Now we can have another index - pointer for those 4 blocks.
~~Sparse index~~

3. Multilevel index
→ It looks like a tree if we turn it upside down.

4. M - way search Tree.

Binary Trees → 1 key
2 children.

M - ways.



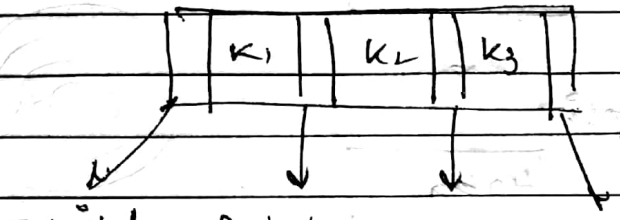
2 - key
3 - child
3 ways ST

M-way search tree.

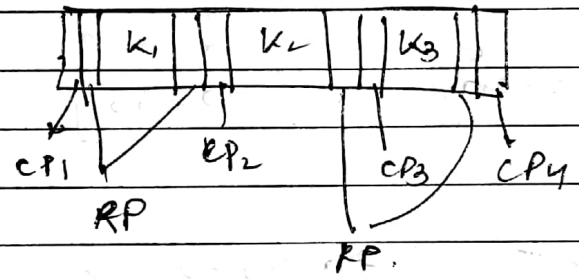
Page No.
Date

* m-way → m-1 keys.

4-way tree.



→ M-wayST For data base Indexing



Problem with m-way search tree :-

→ No control on insertion, we can insert as we like.

→ height of m way search tree can be n.

→ It will take a lot of time in searching.

B-Trees :-

→ M-way search trees with some rules.

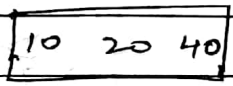
1. Every node must have $\lceil m/2 \rceil$ (ceil) value.
2. Root can have minimum 2 children.
3. All leaf nodes must be at same level.
4. Process is bottom up.

Page No.
Date

B-Tree :-

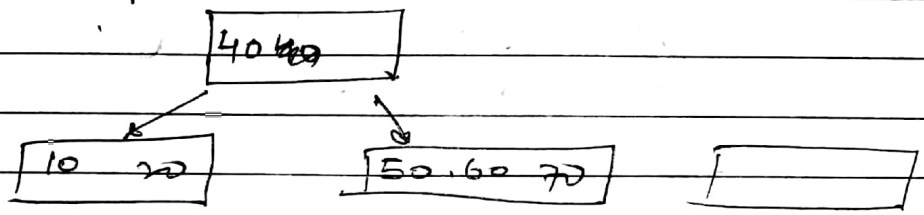
$m = 4$ key : 10, 20, 40, 50, 60, 70, 80, 30, 35

(1)



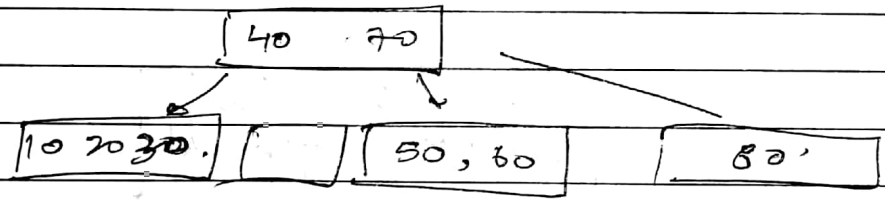
split

(2)

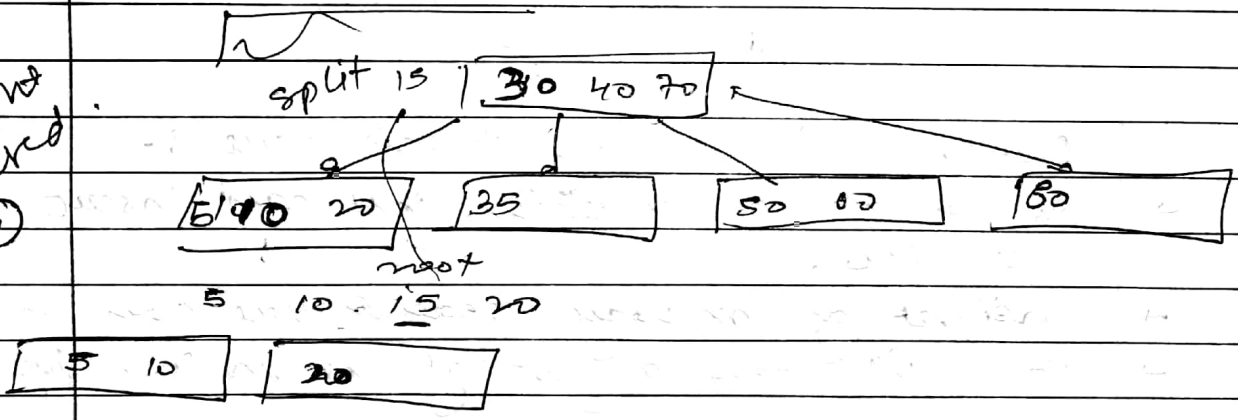


split

(3)

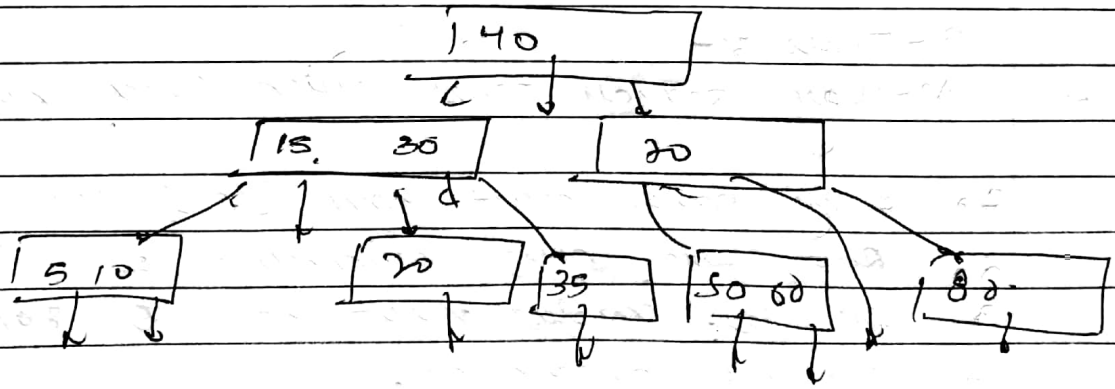


(4)



right biased

(5)



Page No.
Date

→ As we are taking the right middle element it is right biased b-tree.

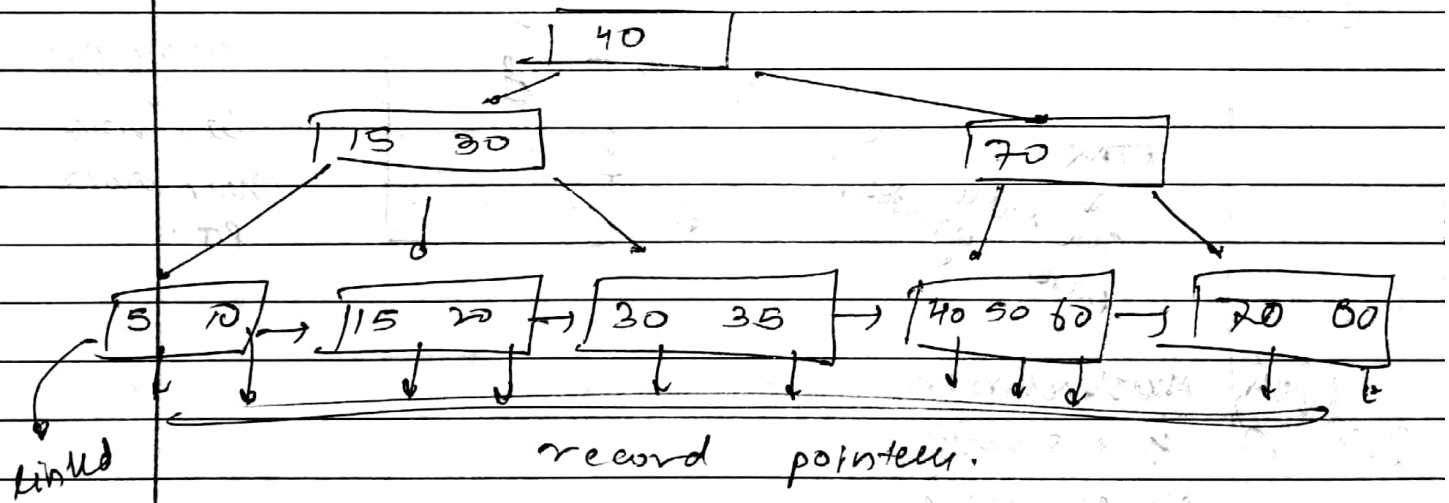
→ we can also take left middle element.

→ B+ → Tree :-

→ We don't have record pointers from every node.

→ We only have record pointers from leaf nodes.

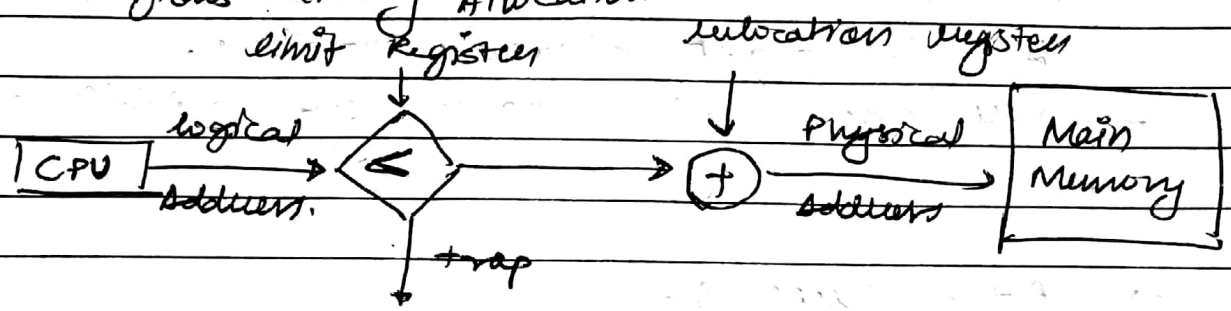
→ every ~~~~ parent will have a copy of its node in the child.



contigious memory allocation → External Fragmentation
↳ Fixed size ↳ Variable size ↳ Internal Fragmentation

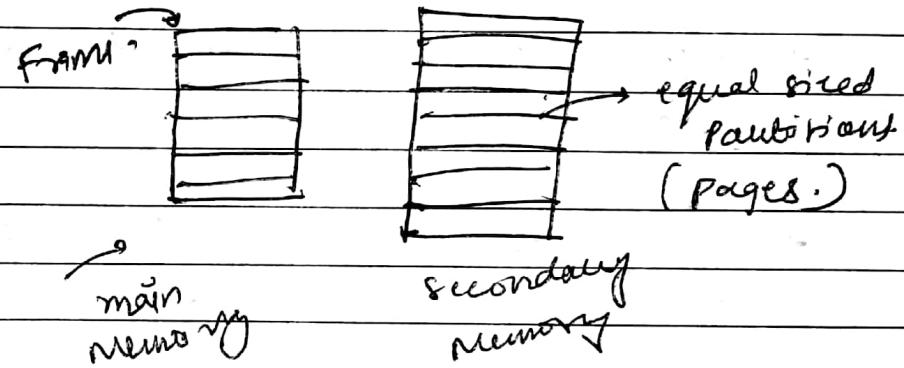
Page No.
Date

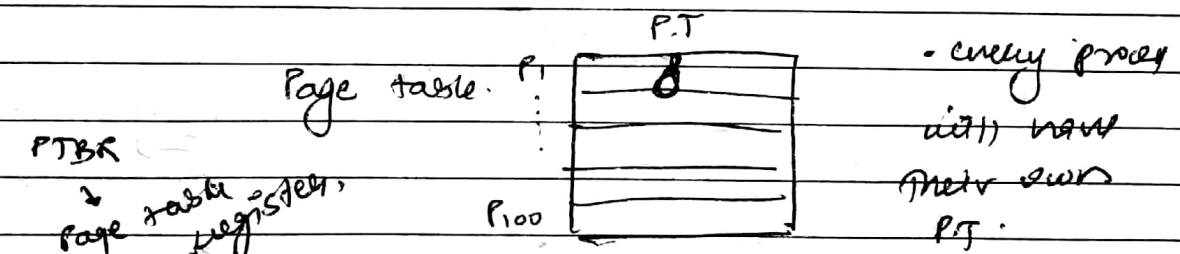
→ Contigious Memory Allocation :-



Non contigious. (Fixed size Partitioning)

→ Paging :-





Deadlock
Mutual exclusion.
Hold & wait
No preemption
Circular wait

Page No.
Date

REACT :-

→ Redux a layer top on React.
→ helps with the state management of the app.
→ data in the app
→ UI states of the app.

→ **Component & templates :-**
- components look like HTML templates.
- They can contain state
- They also can contain Javascript for functionality

→ **component state :-** → Data
→ UI-state.
- Javascript object.
- describes the current state of the component i.e data, UI-state
- The state can be updated over time
  - Modal could close, the data we output could change.

eg:- state = {
  name : 'Ryu',
  age : 30
}

- **DOM Events :-**

Event handler eg: onClick = {function}
this.function.

change state using
~~setching~~ setState()