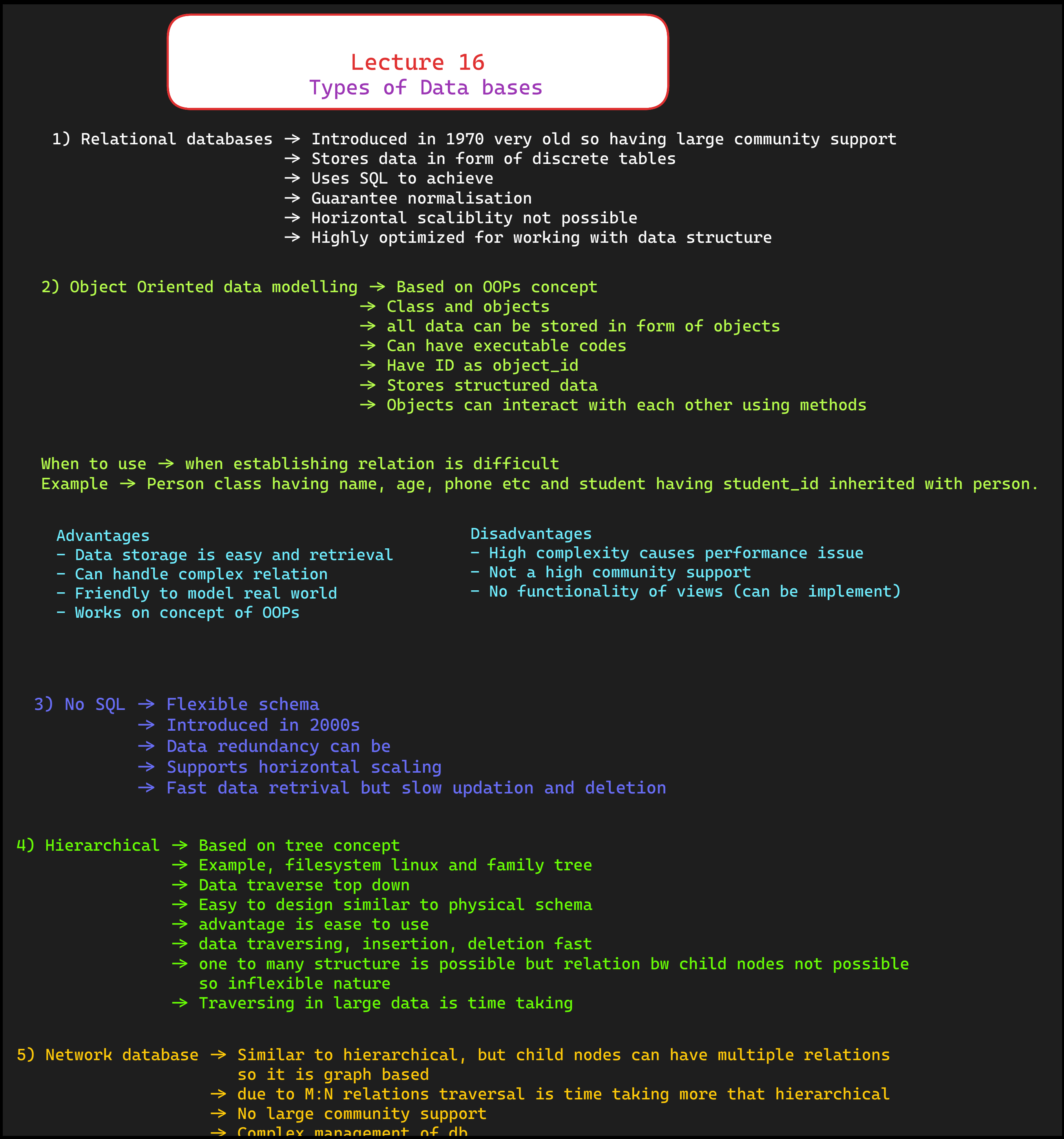


Lecture 16
Types of Data bases

1) Relational databases → Introduced in 1970 very old so having large community support
→ Stores data in form of discrete tables
→ Uses SQL to achieve
→ Guarantee normalisation
→ Horizontal scaliblity not possible
→ Highly optimized for working with data structure

2) Object Oriented data modelling → Based on OOPs concept
→ Class and objects
→ all data can be stored in form of objects
→ Can have executable codes
→ Have ID as object_id
→ Stores structured data
→ Objects can interact with each other using methods

When to use → when establishing relation is difficult
Example → Person class having name, age, phone etc and student having student_id inherited with person.

Advantages
- Data storage is easy and retrieval
- Can handle complex relation
- Friendly to model real world
- Works on concept of OOPs

Disadvantages
- High complexity causes performance issue
- Not a high community support
- No functionality of views (can be implement)

3) No SQL → Flexible schema
→ Introduced in 2000s
→ Data redundancy can be
→ Supports horizontal scaling
→ Fast data retrival but slow updation and deletion

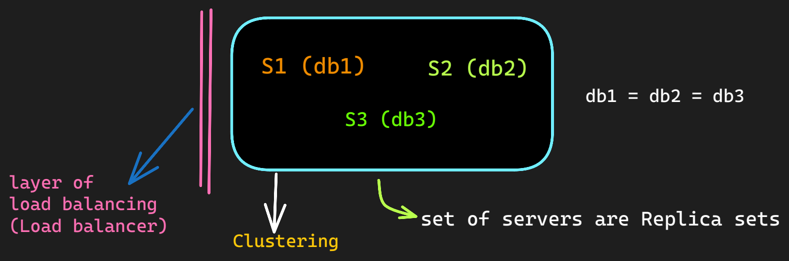
4) Hierarchical → Based on tree concept
→ Example, filesystem linux and family tree
→ Data traverse top down
→ Easy to design similar to physical schema
→ advantage is ease to use
→ data traversing, insertion, deletion fast
→ one to many structure is possible but relation bw child nodes not possible so inflexible nature
→ Traversing in large data is time taking

5) Network database → Similar to hierarchical, but child nodes can have multiple relations so it is graph based
→ due to M:N relations traversal is time taking more that hierarchical
→ No large community support
→ Complex management of db

→ Complex management of db

Lecture 17
What is clustering and replication

Cluster and replica sets →



→ Increase redundancy of db which makes high availability
→ Data abstraction (user don't know from which server data is fetched)
→ Load balancing

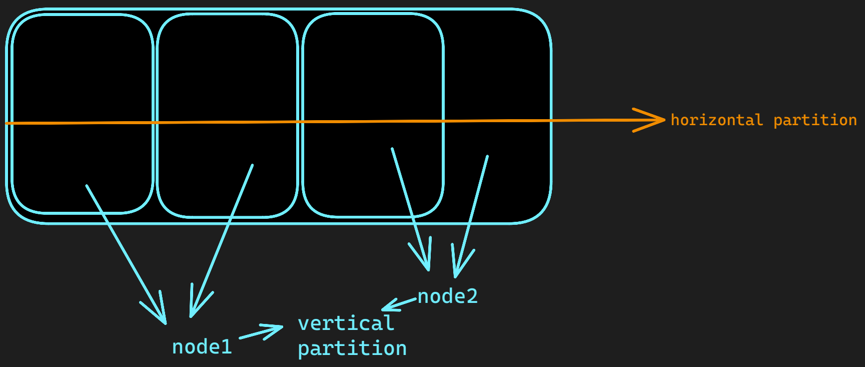
(Content delivering Network) HW → db is distributed on various servers on the basis of geographical region like videos on youtube uploaded by Indian youtubers is mostly watched by Indians so on Indian server these videos will be stored so fast access to the user.

How clustering work → Load balancer check which server has availability to access the data send the request to that db of that server.

Lecture 18
*Partition and sharding in DBMS

If we store data in one system then it will be complex and slow
to get rid from this we can do,

1) Scale up (hardware upgrade) → costly and still takes time for requests
2) Replica sets (clustering) → effective but updation takes time and have propogation delay
3) Partitioning → way of scale out (horizontal scaling), adding different new nodes (data is divided in nodes either horizontally or vertically)



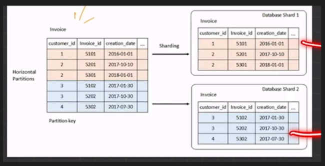
Advantages of Partitioning →

1) Parallelism
2) Availability
3) Performance increase (less load)
4) Ease manageability
5) Less costly than scale up

response time less, prevent vertical scaling which is not suitable and costly

Distributed database → is single logical database which is distributed at various locations (servers) and logically interconnected by servers.

Sharding → Technique to apply horizontal partitioning.
It introduce routing layer (look up) which independent db part has to send the request (sharding and partitionig kind of similar terms)



Invoice

customer_id	Invoice_id	creation_date	...
1	5101	2016-01-01	
2	5201	2017-10-10	
2	5301	2018-01-01	
3	5102	2017-01-30	
3	5202	2017-10-30	
4	5302	2017-07-30	

Database Shard 1

Invoice

customer_id	Invoice_id	creation_date	...
1	5101	2016-01-01	
2	5201	2017-10-10	
2	5301	2018-01-01	

Database Shard 2

Invoice

customer_id	Invoice_id	creation_date	...
3	5102	2017-01-30	
3	5202	2017-10-30	
4	5302	2017-07-30	

Pros → (above advantages)
Cons → i) Routing layer to be implemented, increase complexity
ii) Non uniformity and creates requirement of re - sharding
iii) Not suitable for analytical query.

// Sql has queries for sharding.

Lecture 19
DataBase scaling patterns

Let's understand by an example,

Cab booking app -

. Tiny startup
. 5 users (say)
. 1 trip in every 5mints
. single small db machine stores data like trip history, amount, distance....

- app becomes famous
problem begins,
. getting 30 booking requests per minute
. tiny db starts performing poorly
. API latency increases
. transaction failure
. slugish app
. customer satisfaction decreases

solution,
.need to implement performation optimization
.scaling the db

Pattern 1 → Query optimization and connection pool implementation

.cache store frequently non dynamic data like booking history, payment
.introduce database redundancy so joins time will be saved (or using no sql fast)
.connection pool libraries implement cache db connection
. now efficient

now getting 100 bookings per minute,

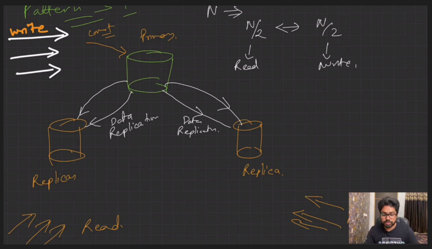
Pattern 2,
scaling up (vertical scaling till pocket friendly)

.2x RAM, 3x SSD
.high processor
. cost increases but now ok

now getting 300 bookings per minute,

Pattern 3,
command query responsibility segregation (CQRS)

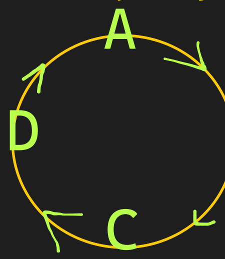
.separate read/write requests
.read requests on replicas and write on primary db,
requests will fast
.now db is optimized and users increases



now due to increase of users write requests will increases and slow

Pattern 4,
Multi primary replication

.distribute replicas (primary replicas) in multiple (replication)
. all works as primary and replicas



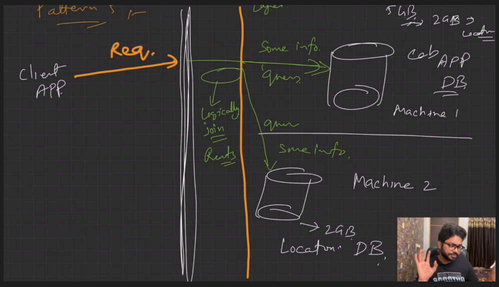
db ⇒ A=B=C=D (replicas)

→ write goes to any one random node
→ read request will broadcast bw replicas

then, 50 requests per second then,

Pattern 5,
Partitioning data by functionality

.different collection of tables in different dbs (multiple db schema of one
db on basis of functionality)
.different dbs with primary and multiple replicas(pattern3) or
multi primary replicas (pattern4)
. need to implement one more layer of look up



now, business expanded on country level then,

Pattern 6,
Horizontal scaling or scale out

.Sharding- multiple shards
. let say 50 machines, having same database schema, having some part of db
. locality of data must be there
. each machine could have replicas
. sharding is complex but no pain no gain

now, business expanded on continent level then,

Pattern 7,
data center wise partition

.data centers across continents having high latency
.maintains availability of system
.enable cross data center replication saves from disaster (center having data of other center as well)