SPECIALIZED STORAGE PARADIGMS

1] BLOB STORAGE (BLOB = Binary large Object)

i) Blob → arbitrary piece of unstructured data
eg - video file, img file, large txt file, large binary (compiled code) file

ii) Blob store → Storage soln for blobs.

iii) Info abt Blob stores

(1) They don't count as a dB since they only only allow storage and retrieval of data based on the name of the blob.

(2) They're usually used for storin large size blobs for small and large scale sys.s

(3) Blob-store accesses blobs usually thru a key or blob name
It specializes in storin massive amts. of unstructured data i.e. blobs.

(4)

Blob	k-v store
Blob accessed thru key	Value accessed thru key
Optimized to store massive amts of unstructured data → high availability and durability of data	Optimized for latency

Date:
M T W T F S S

(5) Really complex prob → accessin blobs from blobstore, so generally relyin on popular blobstore soln. (eg:- Google Cloud Storage, Amazon S3) is suggested as these are big companies that have that kinda infra. to support blobstore soln.
Money is charged based on
- how much storage is used
- how often are blobs stored, retrieved

2] Time Series Database (TSDB)

i) TSDM - special kinda dB optimized for storin and analyzin time-indexed data

ii) Time Indexed Data - Data pts. that specifically occur at a given moment in time
Data ≡ events that happen at a given time say every millisecond

To perform time-series like computa"s on this data, eg
↓
TSDB is used
- computin avg
- computin local maxima, minima
- aggregatin all data b/w 2 specific pts in time

iii) Use cases:

① Monitorin: Lookin, analyzin, computin a bunch of events occurin in our sys. at a given timestamp.

② IoT : Lots of devices which are constantly sending or capturing telemetry or some other data in their environments.

③ Stock Prices, Cryptocurrency prices → changing all the time

(iv) Eg :- InfluxDB, Prometheus

3] Graph dB

(i) Used when within the data that you store, there are a lotta relationships b/w the data (i.e. indiv data pts. in dataset) / multiple lvls of relationship b/w data

eg :- Facebook accounts

dataset
- data pt 1 → acc 1 : Harry, age : 16, eye-color : green
- data pt 2 → acc 2 : Ron, age : 17, eye-color : brown

} RELATIONSHIP : friends

(ii) In SQL table format, querying certain pieces of data that rely on a lotta diff relationships b/w data stored in tables, becomes really complicated

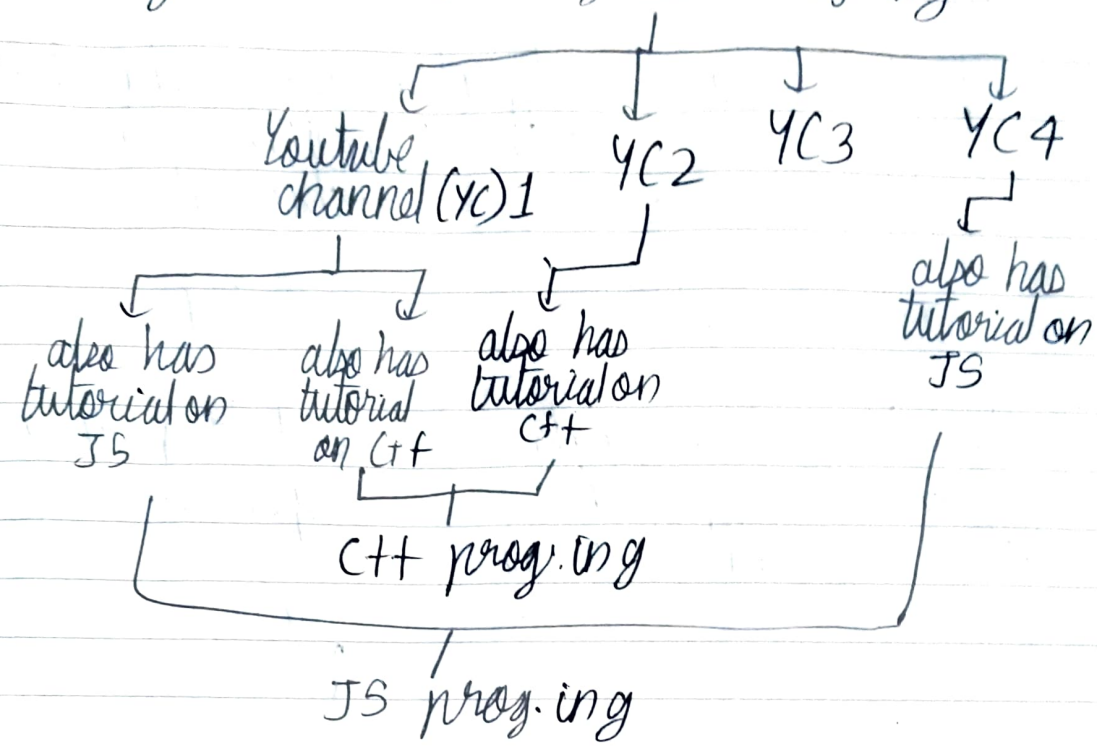
(iii) Graph dB → ① dB built on top of the graph data model.
② ∴ the concept of a relationship b/w indiv data pts is high priority
③ Complex queries on deeply connected data done very fast

(iv) Use cases :

① Social Networks

(2) Websites with relq^n ships b/w people and pages or content

eg:- Content → Python Prog. ing



(v) Most popular graph dB - Neo4j

(vi) Cypher (.cql) - A 'graph query lang' originally developed for the Neo4j graph dB but is now used in a lotta other dBs (aka "SQL for graphs")

Cypher queries are lot simpler than SQL queries when dealin with deeply connected data

eg:- to find cypher query in Neo4j

MATCH (some_node : Some Label) -[: SOME_RELATIONSHIP]-> (some_other_node : Some Label { some prop : val })

4] Spatial dB

i) Optimized for storing spatial data

ii) Spatial Data : literally any data that deals with geometric space.

eg:- loca"s on a map, restaurants in a locality

iii)

Database idx	Spatial idx
i) Used to perform complex queries on 1 col (i.e 1 attr say latitude) really fast	Used to perform complex queries on 2 cols (i.e 2 attrs → latitude, longitude) Also, these queries are referred as spatial related queries

eg.s of spatial related queries :→ finding all loca" in vicinity of a specific loca"

→ dist b/w 2 loca"s

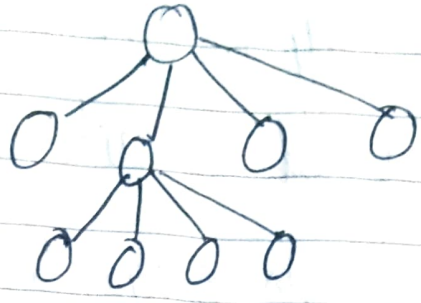
eg.s of a spatial idx :→ Quad tree
→ R tree
→ K-D tree
→ M tree

iv) Quad tree

① A tree data struc used to index 2-D spatial data

② Rule : Each tree node has no. of child nodes → == 0 (leaf node)
→ == 4

eg: of quad tree



#4 or nothin

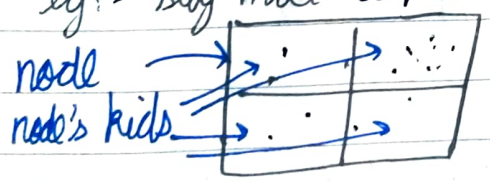
(3) Quad tree nodes → contains some form of spatial data that has a max capa.

(a) for node value
- < max capa. node is undivided i.e. leaf node condi"
- > max capa. node is given 4 kids nodes and its data entries is split across the 4 kid nodes

(4) Graphical visualisa" of quad tree

(a) quad tree node = rect.
(b) quad tree node value = dot inside rect.
(c) max capa. of quad tree node value = max no. of dots a rect. can accomodate before we split it into 4 kids each hjin some of these dots.

eg:- say max capa. = 10



(d) q entire quad tree = grid filled with rect.s that are recursively subdivided into sub-rect.s

Date:
M T W T F S S

eg on opp page: storing loca^{n}s in the world,
let max-node-capa. = n

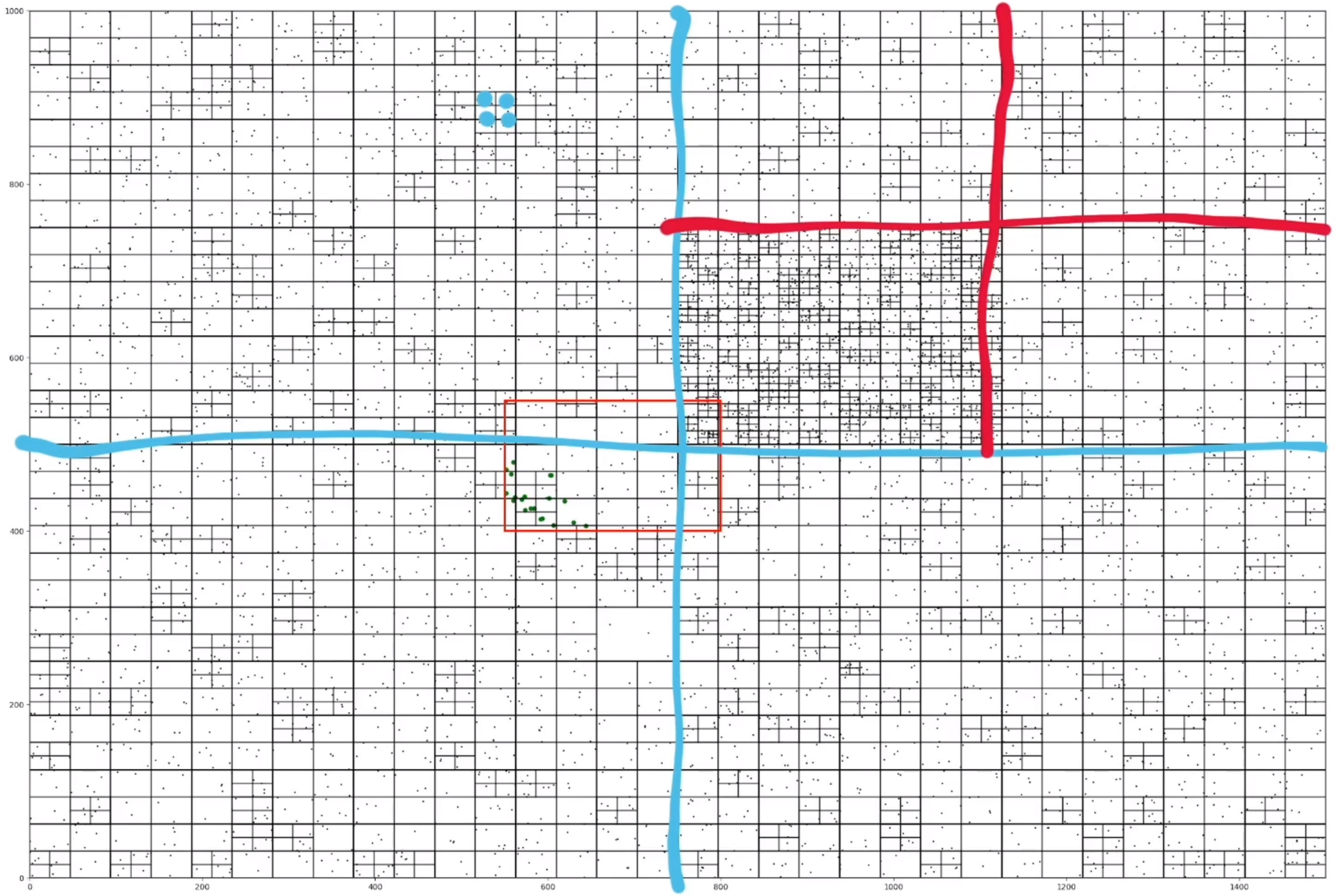
i) Root Node ≡ world ≡ outermost rect.

ii) If entire world has more than n-loca^{n}s, outermost rect divided into 4 quadrants (each representing a ¼ region in the world)

iii) Regions that have
- > n loca^{n}s → further subdivided into 4 small sub-regions (sub-rects.) i.e. corres. quad tree node is given 4 kids nodes
- < n loca^{n}s → undivided rects ≡ leaf nodes in quad tree

iv) Parts of grid with
- many subdivided rects → represent densely populated areas eg: cities
- few subdivided rects → represent sparsely populated areas. eg: villages

v) Finding a given loca^{n} in quad tree: $O(\log_4 x)$T
where x = total no. of loca^{n}s
Analogy: kinda like Quadrary search ≈ version of Binary Search


1000
800
600
400
200
0
0
200
400
600
800
1000
1200
1400