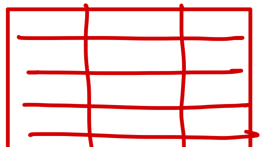
all RDBMS → SQL

↓

Relational

↓

tabular [T1 ⟷ T2]



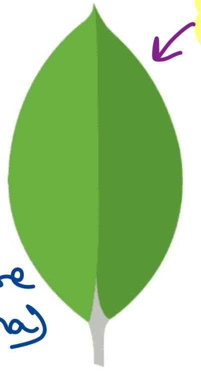
Rows & cols

↑

✓ fixed structure (schema)

✓ 100s of GB

✓ high consistency / transactions.



mongo

cassandra

neo4j

redis

✓ flexible schema

✓ 100s of TB/PB (scaling)

data clients

✓ economical

✓ eventual consistency

NoSQL Databases

→ no standard query language ✗

Trainer: Mr. Nilesh Ghule

→ Not Only SQL

Database

- A database is an organized collection of data, generally stored and accessed electronically from a computer system
- A database refers to a set of related data and the way it is organized
- Database Management System
  - Software that allows users to interact with one or more databases and provides access to all of the data contained in the database
- Types
  - RDBMS
  - NoSQL
  - NewSQL

RDBMS

- The idea of RDBMS was borne in 1970 by E. F. Codd. mathetician.
- Structured and organized data
- Structured query language (SQL)
- DML, DQL, DDL, DTL, DCL. ✓ system tables
- Data and its relationships are stored in separate tables.
- Tight Consistency (tx).
- Based on Codd's rules
- ACID transactions.
  - ✓ Atomic
  - ✓ Consistent
  - ✓ Isolated
  - ✓ Durable

NoSQL

- Refer to non-relational databases
- Stands for Not Only SQL
- Term NoSQL was first used by Carlo Strozzi in 1998.
- No declarative query language → each nosql db has its own lang.
- No predefined schema, Unstructured and unpredictable data → images, audio, video, ...
- Eventual consistency rather ACID property
- Based on CAP Theorem (Brewer's).
- Prioritizes high performance, high availability and scalability
- BASE Transaction
  - Basically Available → basic service is run 24×7.
  - Soft state → data is auto adjusted as per cluster size. → flexible schema.
  - Eventual consistency → changes will be visible all clients eventually (not immediately).

1970 - RDBMS concepts
⋮
1980 - RDBMS - oracle
⋮
1983 - internet
- static pages.

1995 - Java applets
- dynamic pages

internet - business commn
— " —

data burst

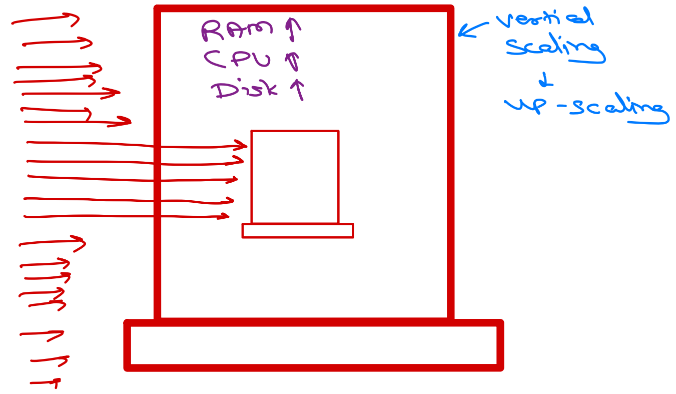
1998 - Carl
↳ No SQL
↳ made his own database.

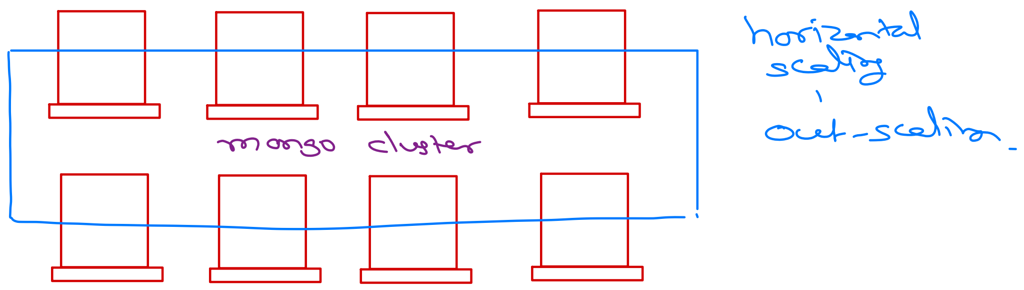
high volume of data
variety data (flexi)
scalability

2004: global conference
- twitter: #nosql

Scaling

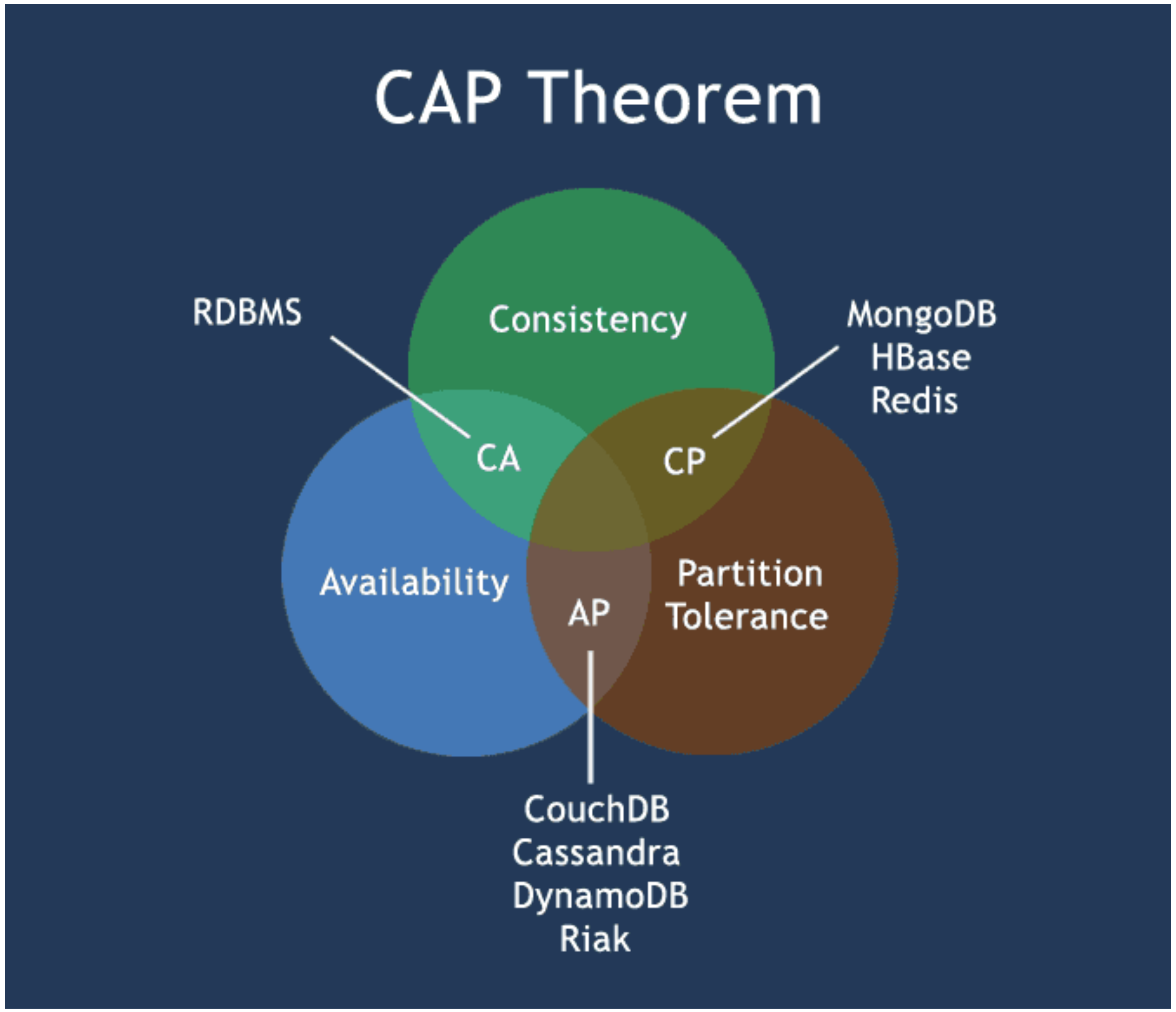
- Scalability is the ability of a system to expand to meet your business needs.
- E.g. scaling a web app is to allow more people to use your application.
- Types of scaling
  - Vertical scaling: Add resources within the same logical unit to increase capacity. E.g. add CPUs to an existing server, increase memory in the system or expanding storage by adding hard drives.
  - Horizontal scaling: Add more nodes to a system. E.g. adding a new computer to a distributed software application. Based on principle of distributed computing.
- NoSQL databases are designed for Horizontal scaling. So they are reliable, fault tolerant, better performance (at lower cost), speed.

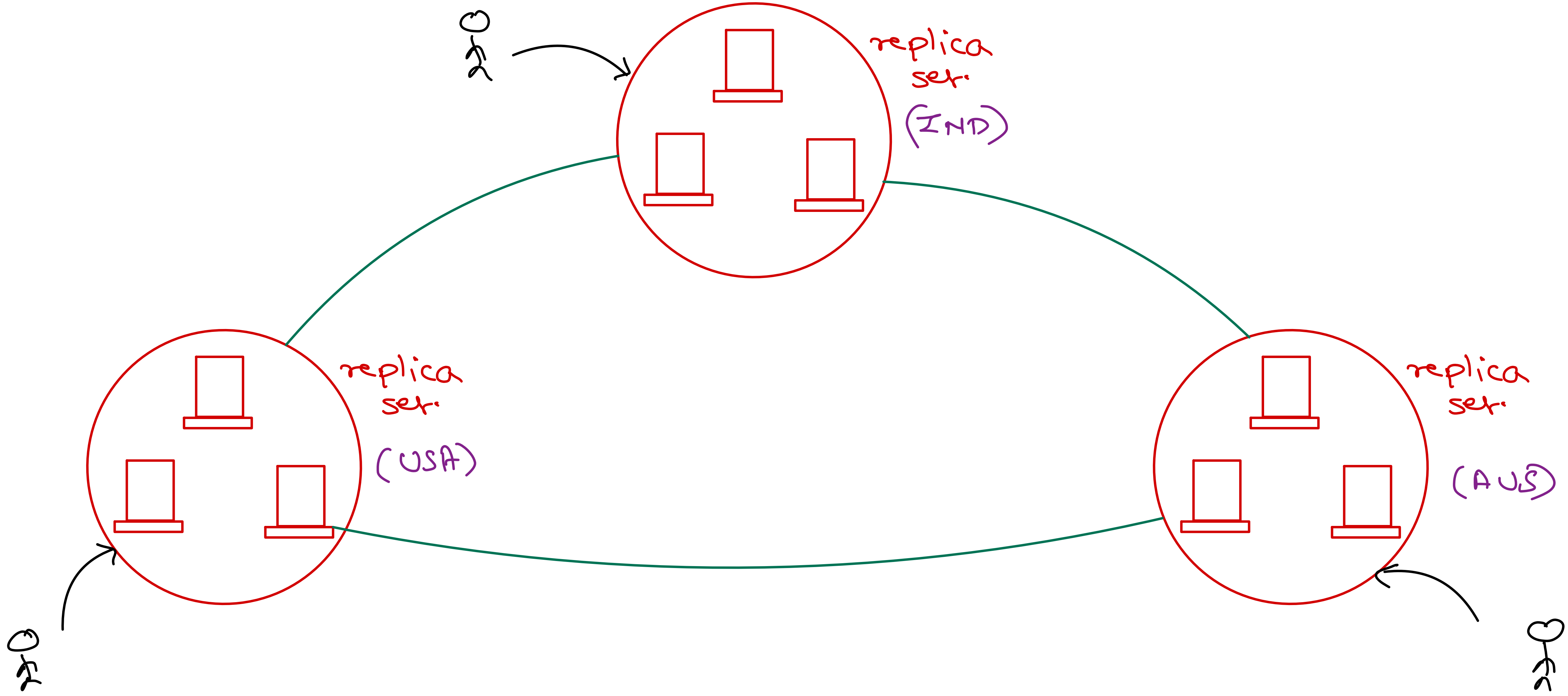




CAP (Brewer's) Theorem

- **Consistency** - Data is consistent after operation. After an update operation, all clients see the same data.
- **Availability** - System is always on (i.e. service guarantee), no downtime.
- **Partition Tolerance** - System continues to function even the communication among the servers is unreliable.
- Brewer's Theorem
  - It is impossible for a distributed data store to simultaneously provide more than two out of the above three guarantees.




replica set.
(IND)
replica set.
(USA)
replica set.
(AUS)

Advantages of NoSQL

- High scalability
  - This scaling up approach fails when the transaction rates and fast response requirements increase. In contrast to this, the new generation of NoSQL databases is designed to scale out (i.e. to expand horizontally using low-end commodity servers).
- Manageability and administration
  - NoSQL databases are designed to mostly work with automated repairs, distributed data, and simpler data models, leading to low manageability and administration.
- Low cost
  - NoSQL databases are typically designed to work with a cluster of cheap commodity servers, enabling the users to store and process more data at a low cost.
- Flexible data models
  - NoSQL databases have a very flexible data model, enabling them to work with any type of data; they don't comply with the rigid RDBMS data models. As a result, any application changes that involve updating the database schema can be easily implemented.

Disadvantages of NoSQL

- Maturity
  - Most NoSQL databases are pre-production versions with key features that are still to be implemented. Thus, when deciding on a NoSQL database, you should analyse the product properly to ensure the features are fully implemented and not still on the To-do list.
- Support
  - Support is one limitation that you need to consider. Most NoSQL databases are from start-ups which were open sourced. As a result, support is very minimal as compared to the enterprise software companies and may not have global reach or support resources.
- Limited Query Capabilities
  - Since NoSQL databases are generally developed to meet the scaling requirement of the web-scale applications, they provide limited querying capabilities. A simple querying requirement may involve significant programming expertise.
- Administration
  - Although NoSQL is designed to provide a no-admin solution, it still requires skill and effort for installing and maintaining the solution.
- Expertise
  - Since NoSQL is an evolving area, expertise on the technology is limited in the developer and administrator community.

Applications

- When to use NoSQL?
  - Large amount of data (TBs)
  - Many Read/Write ops
  - Economical Scaling → horizontal scaling
  - Flexible schema
- Examples:
  - ✓ Social media
  - ✓ Recordings
  - ✓ Geospatial analysis
  - ✓ Information processing

- When Not to use NoSQL?
  - ✓ Need ACID transactions
  - ✓ Fixed multiple relations
  - ✓ Need joins
  - ✓ Need high consistency
- Examples
  - ✓ Financial transactions
  - ✓ Business operations

RDBMS vs NoSQL

	RDBMS	NoSQL
Types	All types support SQL standard	Multiple types exists, such as document stores, key value stores, column databases, etc
History	Developed in 1970	Developed in 2000s
Examples	SQL Server, Oracle, MySQL	MongoDB, HBase, Cassandra, Redis, Neo4J
Data Storage Model	Data is stored in rows and columns in a table, where each column is of a specific type	The data model depends on the database type. It could be Key-value pairs, documents etc
Schemas	Fixed structure and schema	Dynamic schema. Structures can be accommodated
Scalability	Scale up approach is used	Scale out approach is used
Transactions	Supports ACID and transactions	Supports partitioning and availability
Consistency	Strong consistency	Dependent on the product [Eventual Consistency]
Support	High level of enterprise support	Open source model
Maturity	Have been around for a long time	Some of them are mature; others are evolving

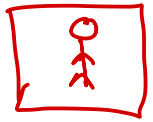
NoSQL database

- NoSQL databases are non-relational. → non-tabular
- There is no standardization/rules of how NoSQL database to be designed.
- All available NoSQL databases can be broadly categorized as follows:
  - ✓ Key-value databases
  - ✓ Column-oriented databases
  - ✓ Graph databases
  - ✓ Document oriented databases → mongo

Key-value database

- Based on Amazon's Dynamo database.
- For handling huge data of any type. → GB , TB
- Keys are unique and values can be of any type i.e. JSON, BLOB, etc.
- Implemented as big distributed hash-table for fast searching.
- Example: redis, dynamodb, riak, ...

across multiple machines

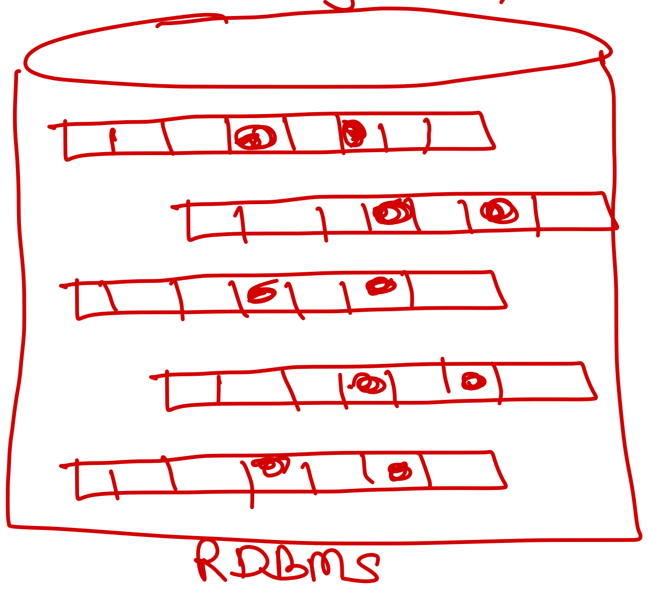
key	value
/sunbeam/dac/1/name	Abc
/sunbeam/dac/1/addr	Pune
/sunbeam/dac/1/hobbies	[Prog, Cooking, Read]
/sunbeam/dac/1/photo	 (binary).

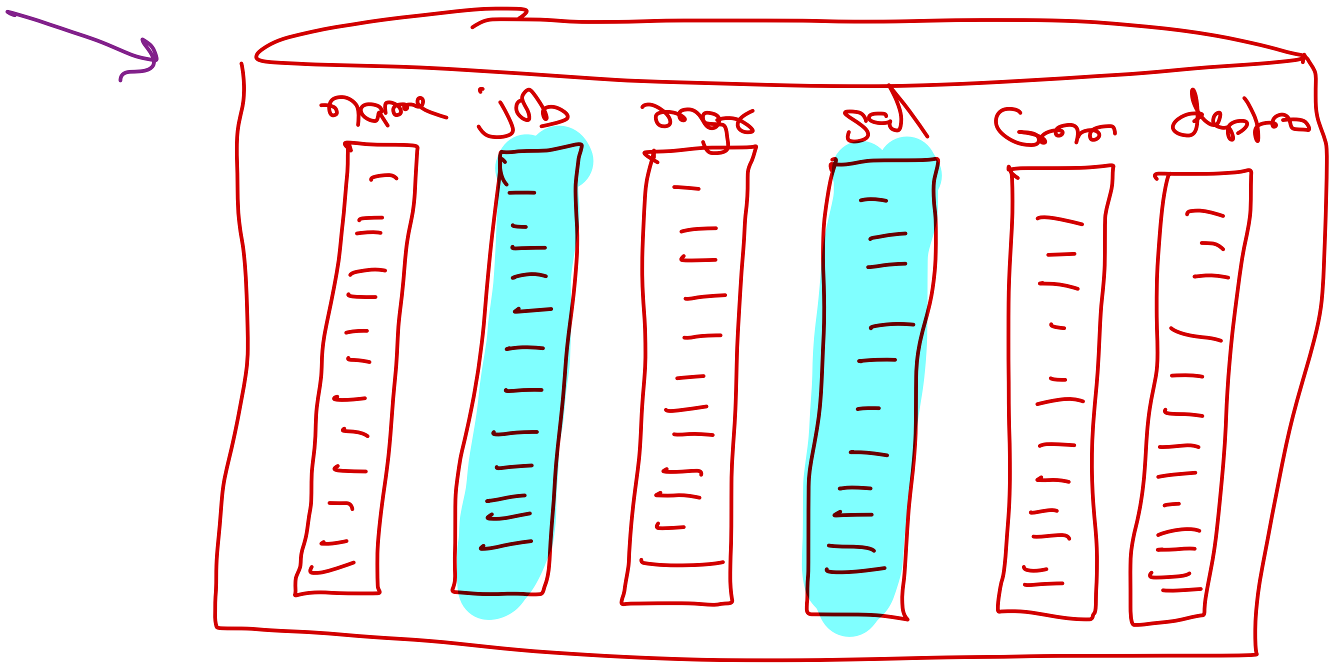
Column-oriented databases

- Values of columns are stored contiguously.
- Better performance while accessing few columns and aggregations.
- Good for data-warehousing, business intelligence, CRM, ...
- Examples: hbase, cassandra, bigtable, …

→ columnar db

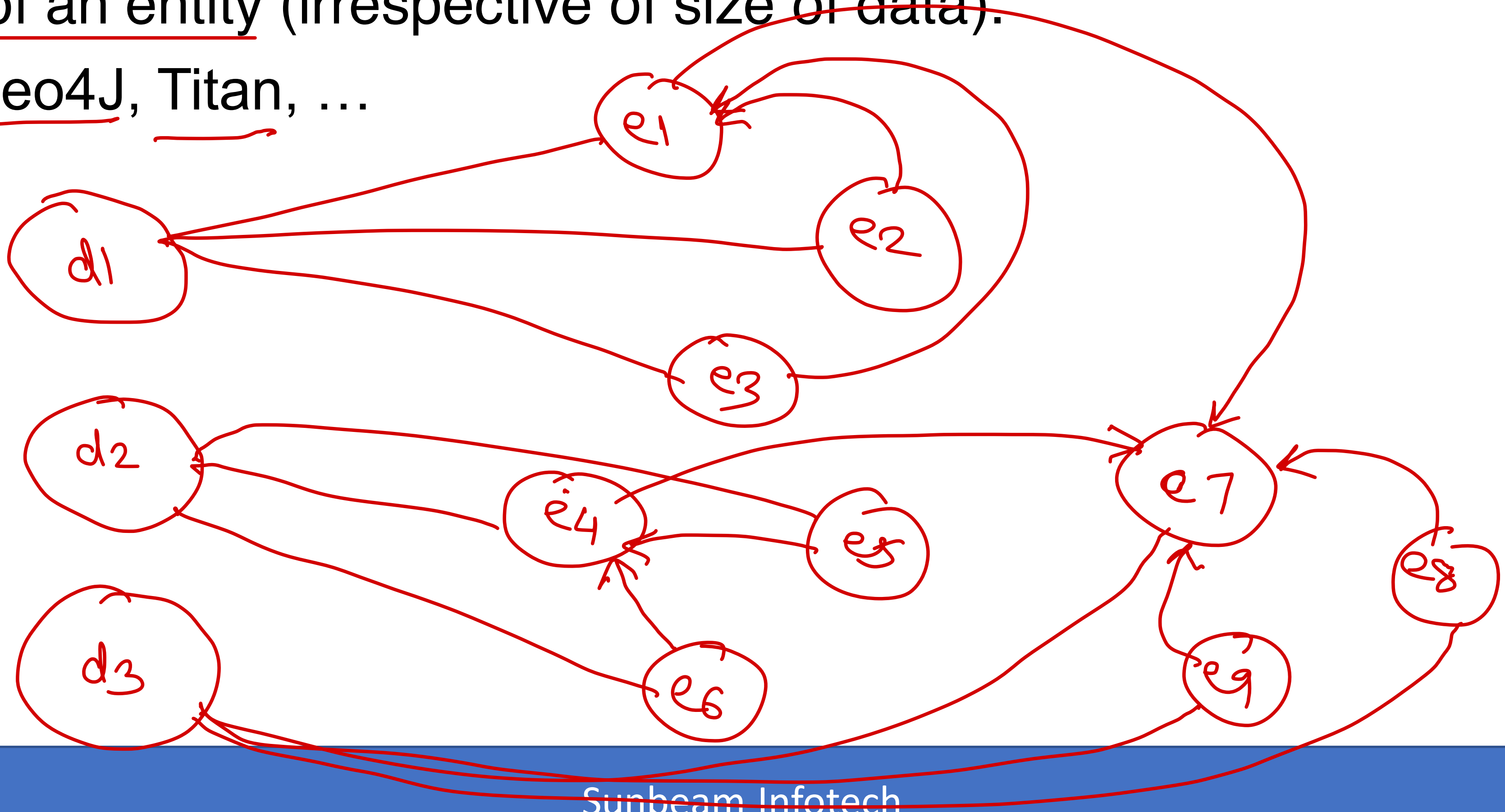
select job, sum(sal) from emp
group by job;





Graph databases

- Graph is collection of vertices and edges (lines connecting vertices).
- Vertices keep data, while edges represent relationships.
- Each node knows its adjacent nodes. Very good performance, when want to access all relations of an entity (irrespective of size of data).
- Examples: Neo4J, Titan, …



Document oriented databases

Java Script Object Notation

- Document contains data as key-value pair as JSON or XML.
- Document schema is flexible & are added in collection for processing.
- RDBMS tables → Collections
- RDBMS rows → Documents
- RDBMS columns → Key-value pairs in document
- Examples: MongoDb, CouchDb, …

b1 JSON → Java Script Object Notation.

```
{
  "id" : 1,            → int
  "title" : "Let us C",  → string
  "author" : "Kanetkar",
  "price" : 240.4      → double
}
```

r1 → JSON document

```
{
  id : 1,
  name : "Nilesh",
  age : 38,
  hobbies : ["Programm", "Reading", ...],
  addr : { area : "Katraj", city : "Pune", pin : 411046 },
  Political : false,
  height : 5.9,
  bloodgroup : null
}
```

MongoDb Databases

Trainer: Mr. Nilesh Ghule

Mongo – QUERY

- db.contacts.find(); → returns cursor on which following ops allowed:
  - hasNext(), next(), skip(n), limit(n), count(), toArray(), forEach(fn), pretty()
- Shell restrict to fetch 20 records at once. Press "it" for more records.
- db.contacts.find({ name: "nilesh" });
- db.contacts.find({ name: "nilesh" }, { _id:0, name:1 });
- Relational operators: $eq, $ne, $gt, $lt, $gte, $lte, $in, $nin
- Logical operators: $and, $or, $nor, $not
- Element operators: $exists, $type
- Evaluation operators: $regex, $where, $mod
- Array operators: $size, $elemMatch, $all, $slice

Mongo – DELETE

- db.contacts.remove(criteria);
- db.contacts.deleteOne(criteria);
- db.contacts.deleteMany(criteria);
- db.contacts.deleteMany({}); → delete all docs, but not collection
- db.contacts.drop(); → delete all docs & collection as well : efficient

Mongo – UPDATE

- db.contacts.update(criteria, newObj);
- Update operators: $set, $inc, $dec, $push, $each, $slice, $pull
- In place updates are faster (e.g. $inc, $dec, …) than setting new object. If new object size mismatch with older object, data files are fragmented.
- Update operators: $addToSet
- example: db.contacts.update({ name: "peter" },
- { $push : { mobile: { $each : ["111", "222"], $slice : -3 } } });
- db.contacts.update({ name: "t" }, { $set : { "phone" : "123" } }, true);
    - If doc with given criteria is absent, new one is created before update.

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>