Data models - NoSQL
Normalization and De-normalization of SQL --> this is the entrypoint to NoSQL

Normalization:
Breaking data into smaller chunks
User SQL

ID	Name	City
1	ABC	NewYork

Order

ID	User.ID	Amount
1	1	1000
2	1	3000

If I de-normalize them

ID	User_ID	Name	City	Order_ID	Amount
1	1	ABC	NewYork	1	1000
2	1	ABC	NewYork	2	3000

We see that the User data is duplcated and redundant. Normalization removes redundancy

Advantages of Normalization: removes redundancy, ensures integrity of data is safe, easier to update

Disadvantages of Normalization: requries a join between tables, makes query complex, expensive

There are many reasons for choosing de-normalization over normalization:
Databases such as Cassandra, MongoDB where we require to perform high-level of analytics, or when we have to create cert-systems, high-rated systems, high-read systems. In these cases, de-normalized databases will work faster than normalized ones. In Normalized tables, we got to 'Join' first before 'Select' query. However, in De-normalized tables, we have one 'Select' query and it works much faster than working with the 'Join'.

NoSQL databases:

Key-value database:
Say in cache and session, we save data in the form of key:value where key is unique (some ID), value can be anything String, JSON, Number, Blob, List, Object

Column database:

Student

ID	Name	Marks	Grade
1	ABC	90	X
2	XYZ	91	XI

We write values column-wise (left to right). We have different column fields like ID, Name they accumulate to create an entity. In SQL, read is left-to-right and in Columnar db, read is top-to-bottom.

Say we want to calculate Average marks in Grade X --> in SQL, it is scanned row-by-row, which is slow. In Columnar database, the values are stored as columns with IDs. So it is fast to retrieve in terms of column. In Columnar database, we give importance to columns rather than rows.

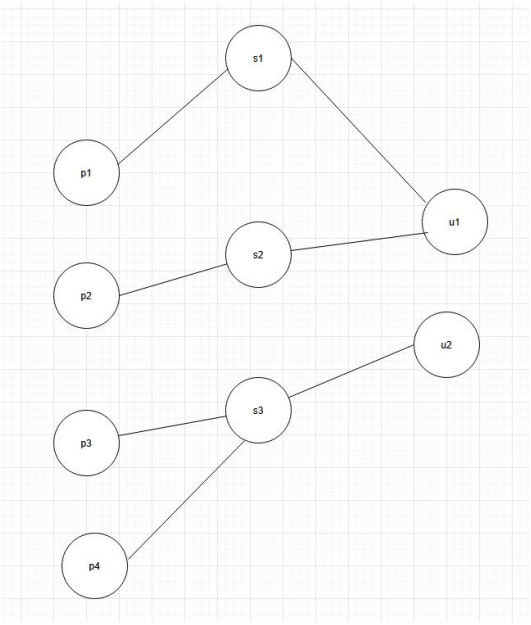
Say to extract only Marks, they can be retrieved based on column Marks along with IDs

ID	Marks
1	90
2	91

In terms of reading data, it is great but in terms of writing or managing it could be a tedious job
Because it is NoSQL, we can write a JSON ({'X':10,…}) as well instead of say '10' in Marks

Graph DB

Student s1, s2, s3, University u1, u2, u3, Professor p1, p2, p3, p4



Say Professor p4 teaches Student s3 at University u2. Say from p3 we can go to p4 because they are teaching at the same university. No connection between p2 and p3.
Say a situation, all students like s1 are dropping out of University u1. So could we blame on Professor p1? answer is no, we need more information like recent incidents at u1 etc
Disadvantage of Graph is, it gets complex when we have large amount of data
It is used highly in data science to analyze the patterns of the current workflow and to predict the future workflows.
For graphs, we have other query languages: Gremlin, SparQL, Cypher
Con: Complexity, Slow if root is too far, Risky

Document DB:
This is the most popular NoSQL DB
Example: MongoDB, Cassandra, CouchDB
Advantages:

- We can save any kind of value that means flexible in nature,
- Dynamic schema,
- Stores it value in JSON format,
- Fast retrieval,
- Easy to understand,
- Easy to scale vertically also because we don't have to maintain any relations like 1:M,
- We need to migrate only one single system

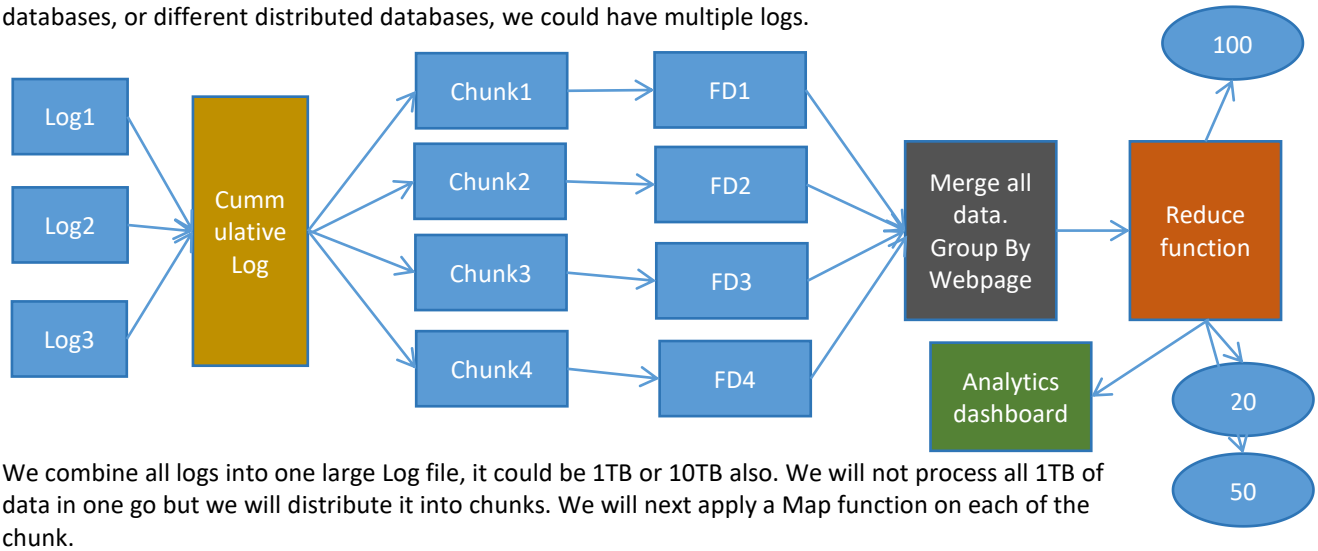
Disdavantages: Again complexity,

JSON object
{ '_id': '234',
 'name': 'Abc',
 'address': {
 'city': 'NewYork',
 'state': 'NY',
 }
 'skills':["Python", "Java", "SQL"]
}
{ '_id': '323',
 'name': 'xyz',
}

In the next _id, I have entirely different keys and data.
Say we are a new startup and growing then we decide we don't need to store Employee fax number or personal email, we can remove on the go. It is very flexible
Use-Cases: UserProfile, Analytics

Map Reduce

We have already setup Analytics in our system and we want to know how many users are spending time on different pages of application. Say how many users we get in Home page, About page, ContactUs page? Lets say we have run some sale on Black Friday and which product(s) got the most traffic? We can find all these information from our Log files. If we have different systems, different databases, or different distributed databases, we could have multiple logs.



We combine all logs into one large Log file, it could be 1TB or 10TB also. We will not process all 1TB of data in one go but we will distribute it into chunks. We will next apply a Map function on each of the chunk.
Map Function (FD): it can contain a Key and Value
Key: homepage
Value: 1 (frequency we will count once we see a click)
Reduce function will increment the count, if value is 10, will increment to 11 etc
Lets say, Homepage has 100 hits, Page2 has 20 hits, Product page has 50 hits
Map is basically seggregating data and creating data in key:value format
Reduce is reducing data again and creating data in the format required by Analytics

ETL --> OLTP and OLAP
Apple sells two things: one is their products and two is their insurance
So we have two DBs, ProductDB and InsuranceDB

ProductDB schema

StoreName	CustomerName	Device	Price	SaleDate
Abc	A	A1	1000	
Bcd	B	A2	7000	

InsuranceDB schema

StoreName	CustomerName	Period of Insurance	Plan	Price
Abc	X	2	Screen protection	500
Bcd	Y	3		200

As an Apple manager, I want to know which storelocation is generating more sales to Apple

Say, we have a database for all customer system and it should be separate from internal database that's handling all internal analytics. We will replicate the data from Customer database into Internal database. We have to aggregate this data and create our own DB. This Aggregation is Extraction

StoreName	DeviceSell	InsuranceSell
Abc	$1000	$500
Bcd	$7000	$200

From the huge chunk of above data, we only need the small summary data
Say if we have to convert data into Euros or some other currencies. Say some stores were in a different location. For converting the amount into equivalent units, we need to perform Transformation.

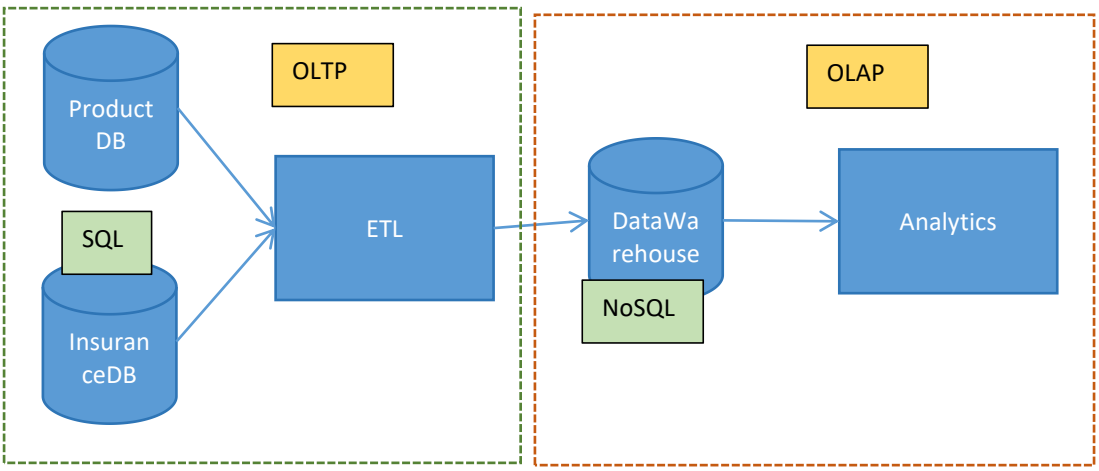
Now this data goes into a DataWarehouse to make a decision

From DataWarehouse, Analytics team can do their work like showing graphs to managers

Aggregation to convert raw data into summary is Extraction
Converting summary into equivalent units (currency) is Transformation
Loading data into DataWarehouse

ETL stands for Extraction Transformation Load

Basically raw data will go through ETL before being stored in DataWarehouse



OLTP: Online Transactional Processing System
OLAP: Online Analytical Processing System
OLTP Databases are ProductDB and InsuranceDB. They should be always available and functional
So OLTP is more required than OLAP. If OLAP fails then only our Analytics will be down and it is Internal.
OLAP requires faster retrieval of data (read-intensive), so NoSQL database is recommended.
Consider OLTP DBs like ProductDBs or InsuranceDBs that are client-facing must be SQL because they are write-intensive.
As Developers we tend to work more on OLTP systems (SQL) that's why we generally have less knowledge of NoSQL
OLAP could have SQL but relatively easier and better to have a NoSQL database for OLAP