Distributed Databases for:

- reliability during blackouts
- availability (closer to end user) -> Facebook
- Computing (collective computation)
- Sometimes it’s forced.
    - Data is naturally on multiple nodes. Ex: Kayak that searches through different databases.

Bottleneck in slow query is disk access. Now with multiple sources, we have another bottleneck:

+ **Synchronization across data in multiple places**
+ **Communication lag.**
+ **Concurrency of transactions updating data all the while**

Data warehouse: Big database that collects data from different places and stores them.
Data mining: Sorting through data to identify patterns, establish relationship.
Aggregate queries: COUNT SUM AVG MIN MAX

Integration and Quality - same data, different formats. Ex: different data types, different date structure, incorrect values, missing values.

Big data (Storage and Analytics (executing fast) important): not the size of the data, but the **amount of computing and processing it takes**. They **challenge computational resources** (phones, tablets). This is different from cloud.

3V’s of big data: Challenges

1. *Velocity*: Moves @ high rates (think sensor driven systems) Valuable in its temporal state.
    (eg weather and traffic sensors)
2. *Volume:* Fast moving data creates massive historical archives. (eg stock market - must update, demonstrate, and analyze quick). This is valuable for mining patterns, trends, and relationships.
3. *Variety:* From structured to unstructured. Valuable for data enrichment and fusion.

Different storage:

1. NEW SQL

2. NO SQL (not only sql) - mongoDB (No relational schema, create small program to access data, making data distributed) solution to storing massive quantities of big data at rest more cheap. Also handles high velocity of incoming data.

Application of big data example: google saw an outbreak of influenza by seeing people google the symptoms more.

Bigger the data, the simpler the model (Algorithm makes no assumption)

Distributed Computing Systems: interconnected by computer network to perform tasks. It is autonomous and NOT homogenous.
DDB - a collection of multiple logically interrelated databases over a computer network.
Distributed DBMS - manages DDB (software system)
DBBS = DBB + D-DBMS (data stored in multiples sites, which each logically consists of a single processor, and connected by computer network.

Why Distribute?

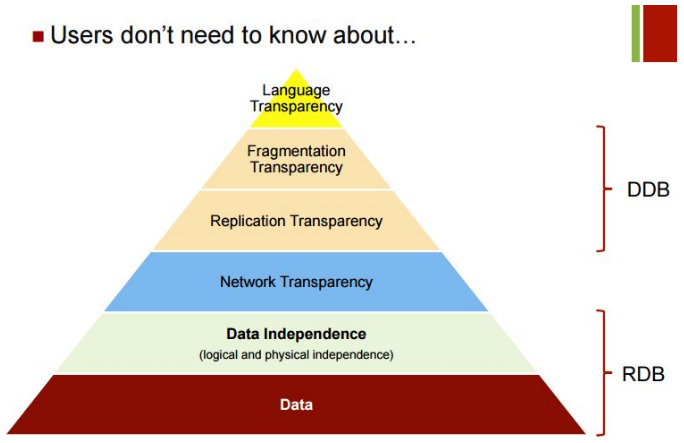
1. **Organisational Structure** - different geographical locations of enterprise or different departments.
2. **Economic Benefits** - more processing power. (Ultimate goal: higher processing efficiency and lower data transmission costs).

Conditions for DDB:

1. Sites connected by communication network that **transmits data and command.**
2. Information in the databases to be logically related.
3. Heterogenous = all sites have different data, hardware, software.

Benefits of DDBS **(TRAPE):**

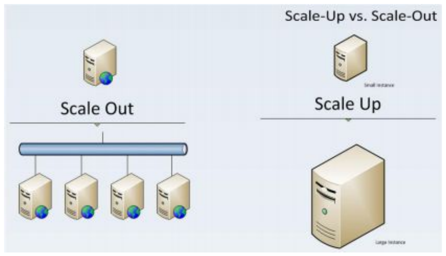
1. **Transparency** - Users don't need to know the design of the database and where a transaction executes or where a data is coming from.


Users don't need to know about…
Language Transparency
Fragmentation Transparency
Replication Transparency
Network Transparency
Data Independence (logical and physical independence)
Data
DDB
RDB

2. **Reliability** - One site can fail without affecting the rest.
3. **Availability** - Other data can be accessed (Replication)
4. **Performance** - "Data Localisation" - reduced contention of I/O and CPU and reduces access delays. Multiple queries at different sites. Closer Proximity of data to it's points of use. Reduces communication overhead.
5. **Expansion** - Expansion of the system is easier. Ex: adding more data, increasing database size, adding more processors.

Types:

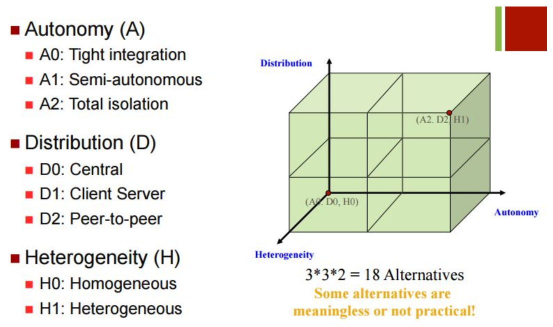
Scale-Out: adding new nodes (Ex: Black Friday, Fluctuating loads of users, HADOOP - cheaper)
Scale-UP: adding more storage, CPU to a node (more expensive)



Parallel Processing:

1. Inter-Query: Several different queries execute concurrently. Executes different operations at the same time in different processors.
2. Intra-Query: Single query decomposed to smaller tasks. Execute each individual operations in a query at the same time. Ex: joining 4 tables -> 2 processors join 2 each and do final join.

3 DImensions of Distribution:



- Autonomy (A)
  - A0: Tight integration
  - A1: Semi-autonomous
  - A2: Total isolation
- Distribution (D)
  - D0: Central
  - D1: Client Server
  - D2: Peer-to-peer
- Heterogeneity (H)
  - H0: Homogeneous
  - H1: Heterogeneous

3*3*2 = 18 Alternatives
Some alternatives are meaningless or not practical!

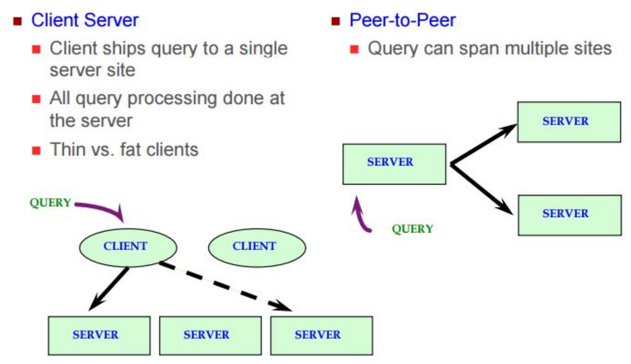
Autonomy: Degree to which components of the database can operate independently.

- Tight Integration: all data shared. Available across database.
- Semi-autonomous: each site can operate independently. Each site can modify settings to share data.
- Total Isolation: sites don't know of each other's existence or how to communicate with them.

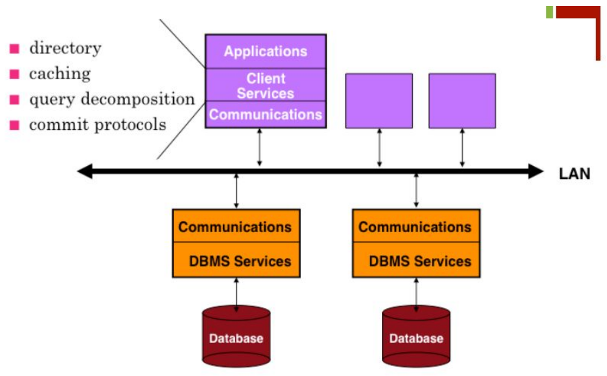
Distribution: Data Language and Transaction management Algorithm etc.

- Central (non-distribution)
- Client Server (Like ATM or most mobile apps) data management duties at server, which sends results to user interface.
- Peer to peer: each site can communicate with other machines to execute queries and transactions. No distinctions b/w client machines versus servers.

Heterogeneity: Difference of data model, language and hardware.



- Client Server
  - Client ships query to a single server site
  - All query processing done at the server
  - Thin vs. fat clients
- Peer-to-Peer
  - Query can span multiple sites



- directory
- caching
- query decomposition
- commit protocols