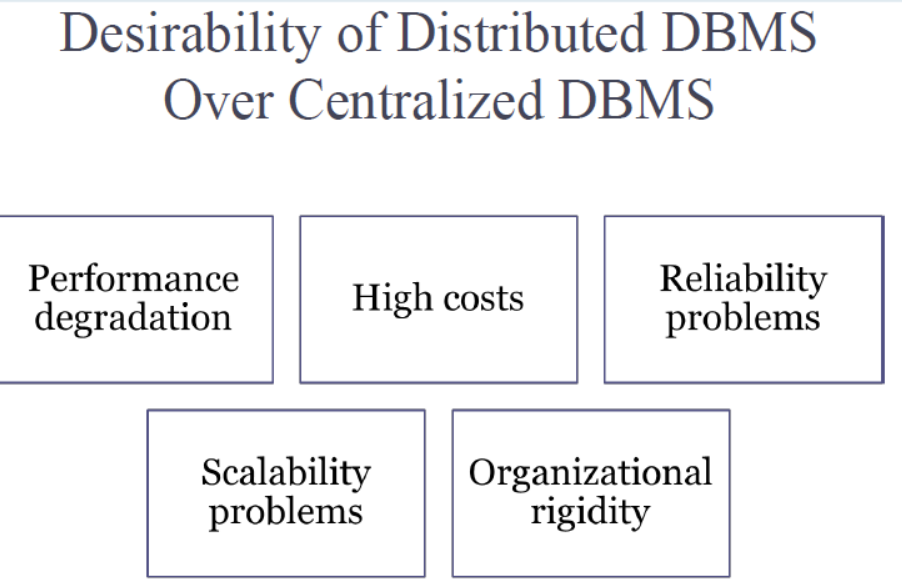
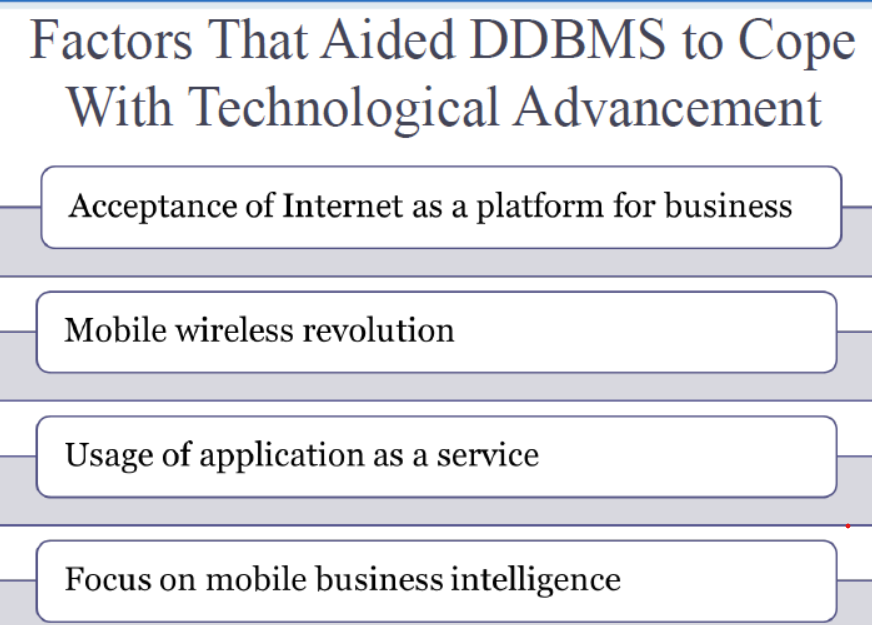
**LECTURE 8**

Need of DDBMS:

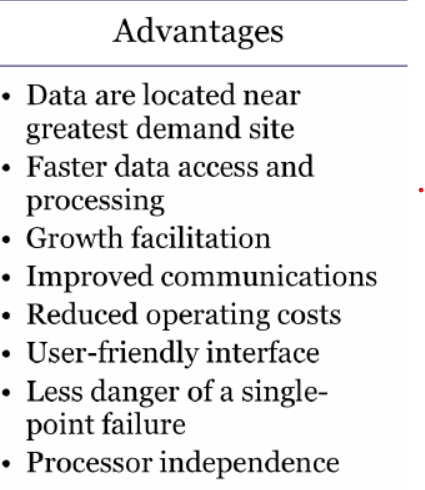
rapid, ad-hoc access to data was needed ['Internet speed' decision-making]

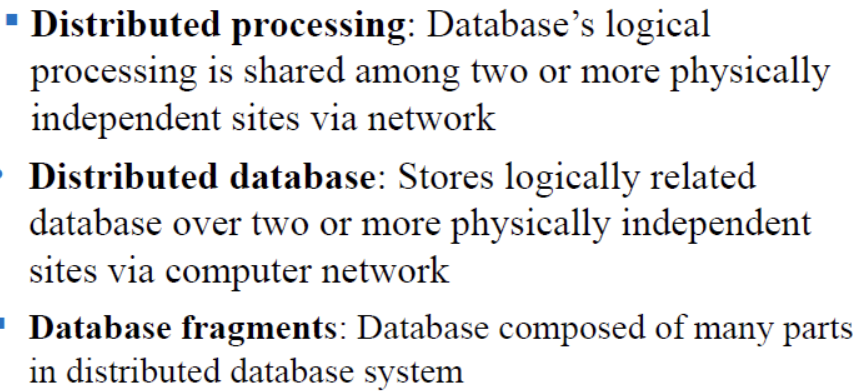
distributed data access was needed, to serve dispersed business units [globalization]





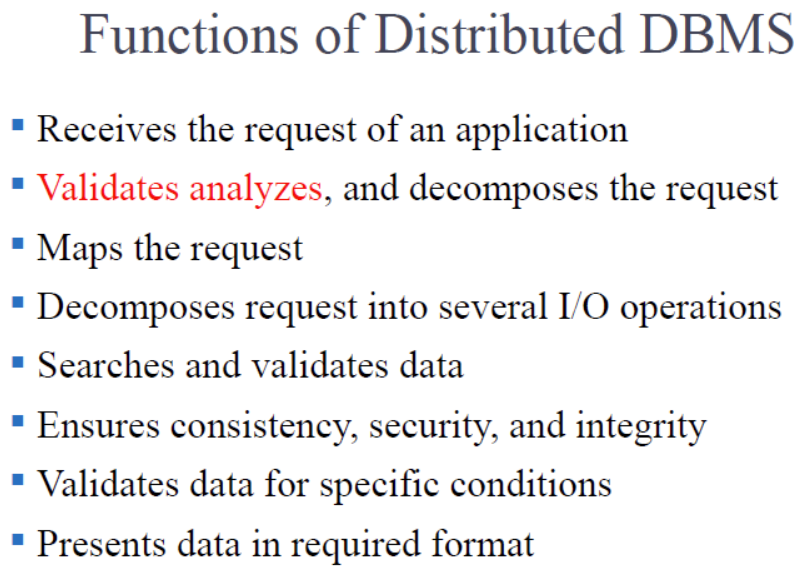
Adv of DDBMS:

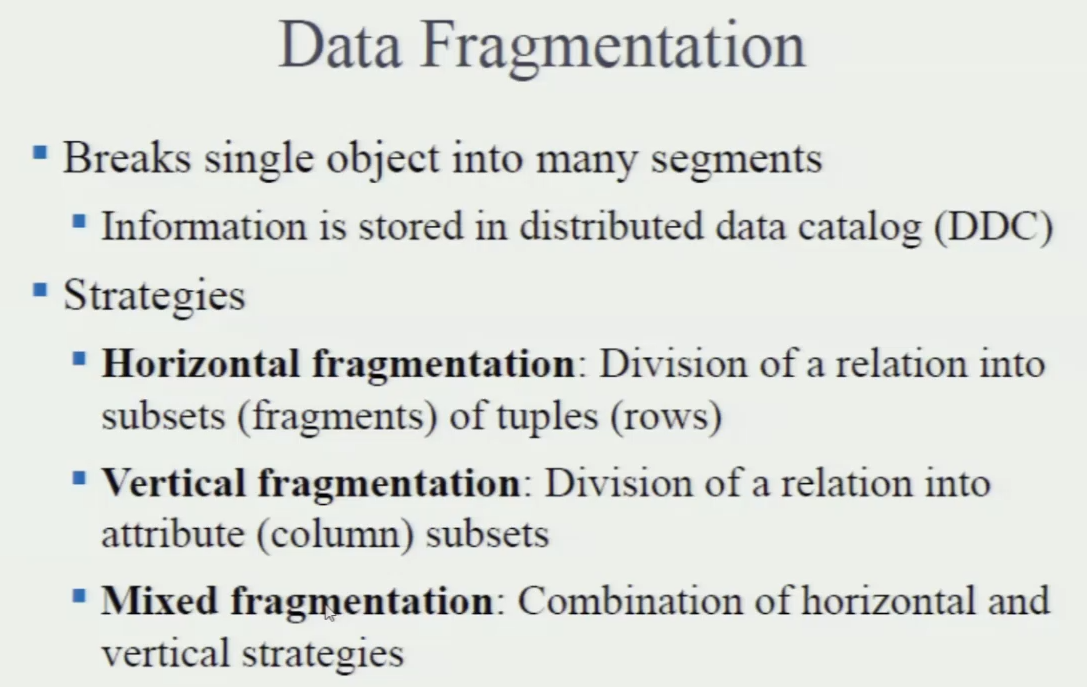


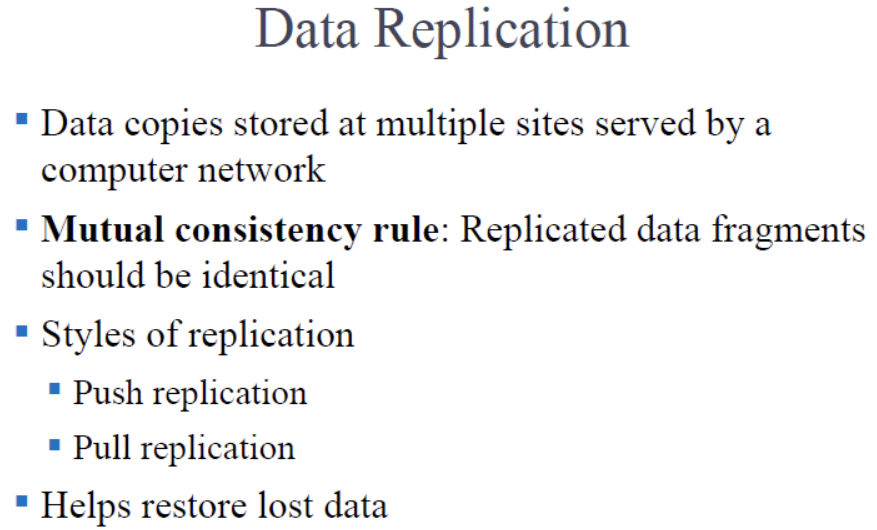


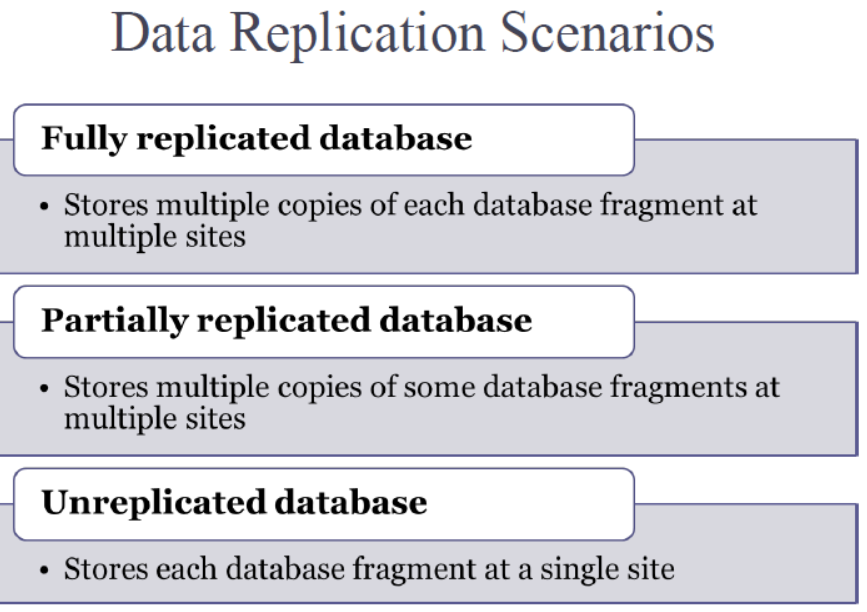
Suppose we have to check whatsapp chats done in one day. There may be billions of them. We use horizontal fragmentation of data.

Create Read Write Delete. Create a new ride, read the drive and customer, write as in match lyft driver to customer, delete the data after the ride









CAP Theorem:

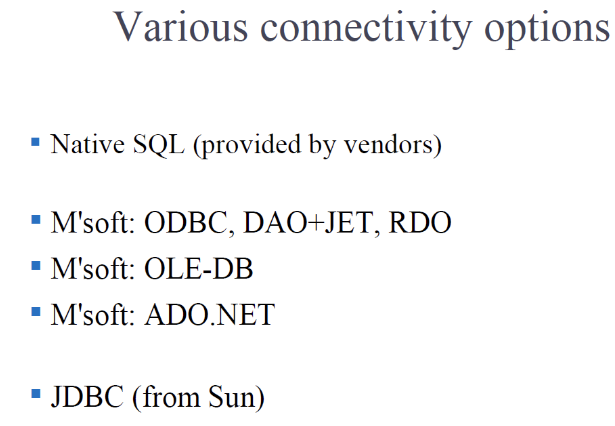
Consistency: always correct data.

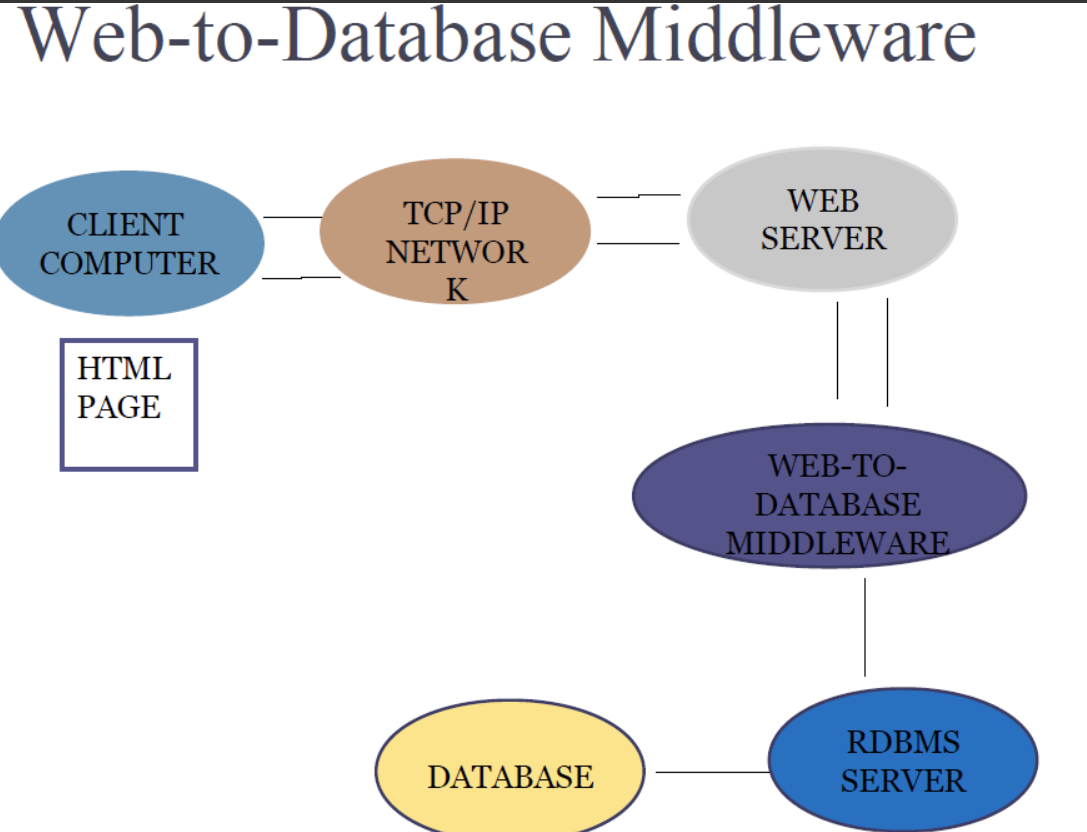
Availability: requests are always filled.

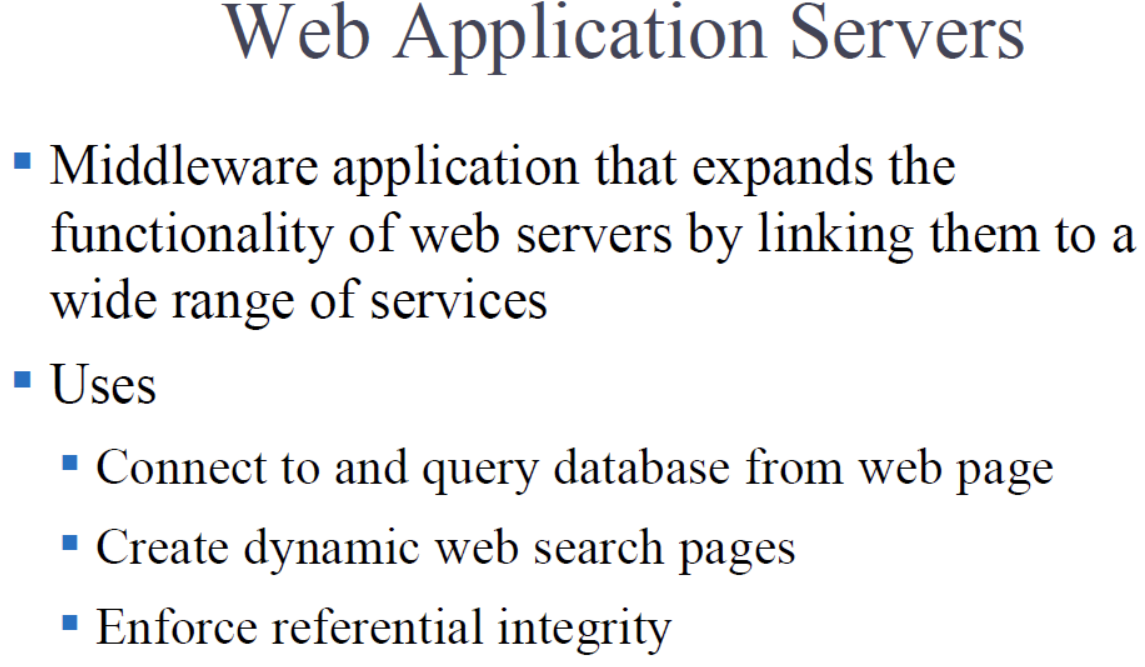
Partition ("outage") tolerance: continue to operate even if (some/most) nodes fail.

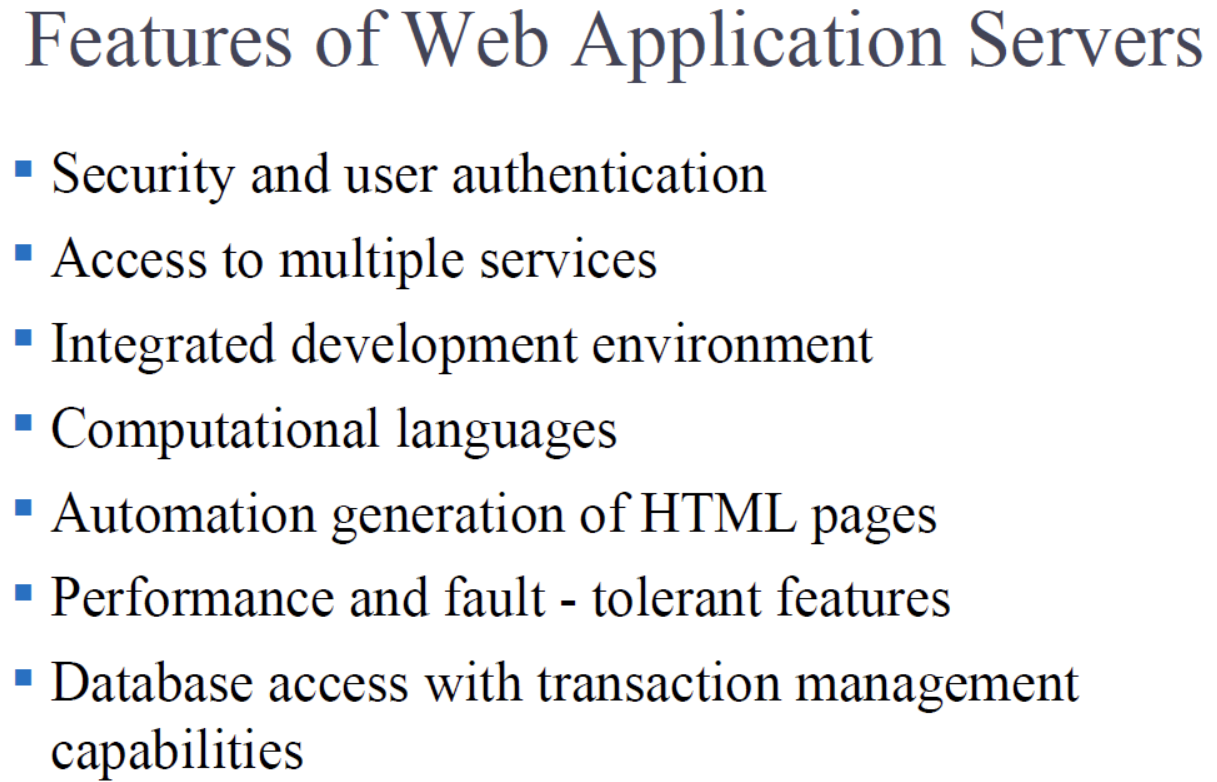
The CAP theorem 'used to say' that in a networked (distributed) DB system, at most 2 out of 3 of the above are achievable [PC, CA, or PA]. But CA means low P, that means that we can't even operate, which makes CA a moot point!

**LECTURE 9**









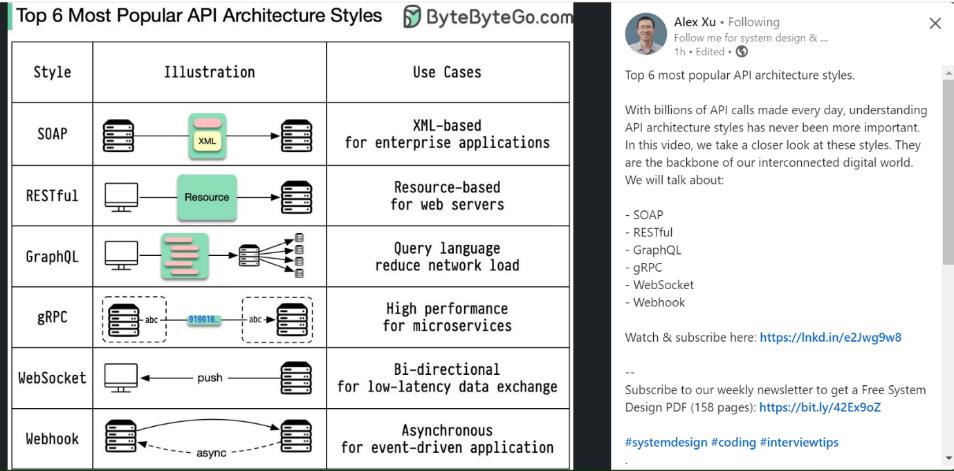
MCC: - \*\*\*IMP\*\*\*

Microservices: Microservices is an architectural style where an application is composed of small, independent services that are developed, deployed, and scaled independently. Each service focuses on a specific business capability and communicates with other services through well-defined APIs.

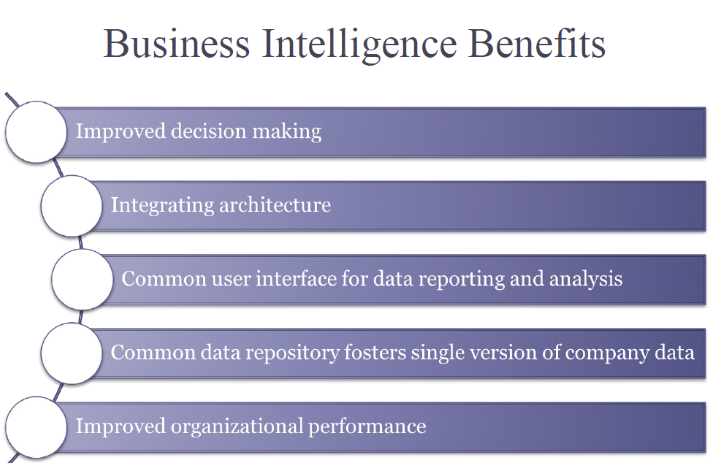
Cloud Computing: Cloud computing involves the delivery of computing services over the internet. These services typically include computing power, storage, databases, networking, and more. Cloud computing allows businesses to access resources on-demand and scale them up or down as needed, without the need to invest in and maintain physical infrastructure.

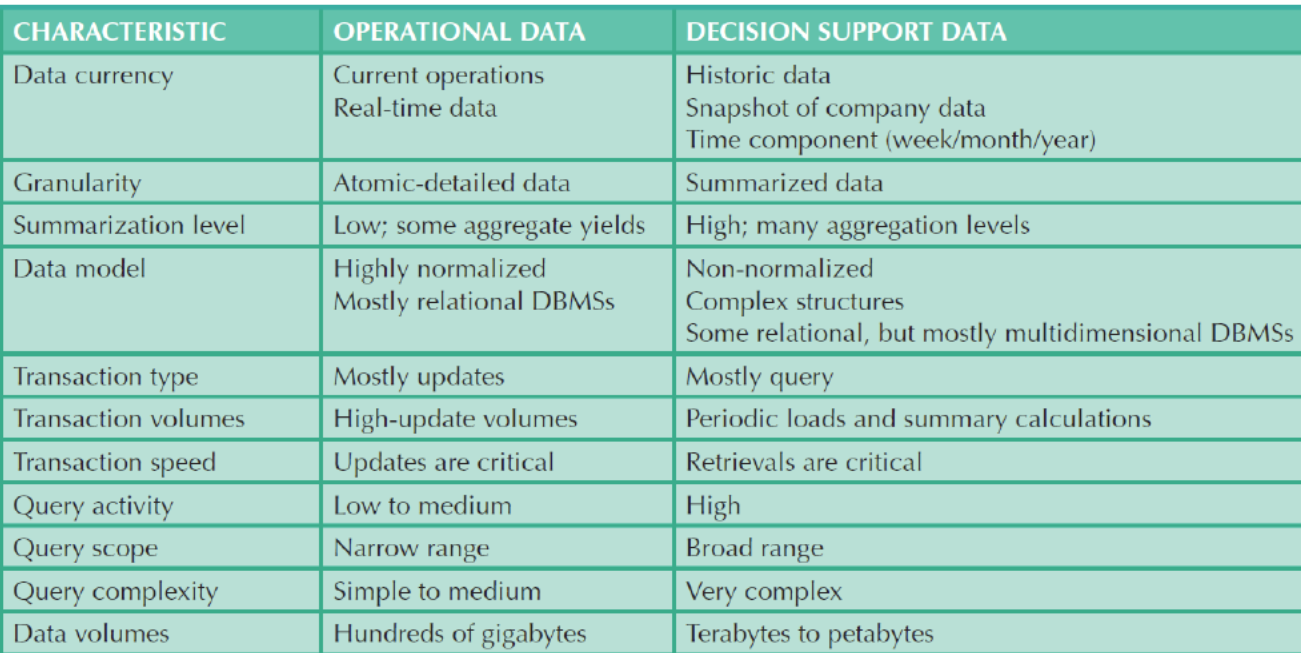
Containers: Containers are lightweight, portable, and self-sufficient units that package software and its dependencies together. They provide a consistent environment for applications to run across different computing environments, such as development, testing, and production. Popular containerization technologies include Docker and Kubernetes.

Combining these concepts, a "microservices cloud container" likely refers to deploying microservices-based applications within containers on cloud infrastructure. This approach offers benefits such as scalability, resource efficiency, ease of deployment, and consistent runtime environments. Tools like Kubernetes, Docker Swarm, and cloud platforms like AWS, Azure, and Google Cloud Platform (GCP) provide the necessary capabilities to deploy and manage microservices within containers in the cloud.



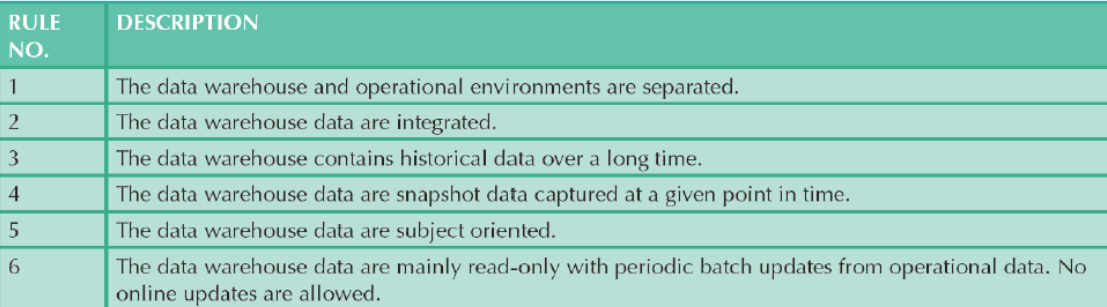
**LECTURE 10 and 11**

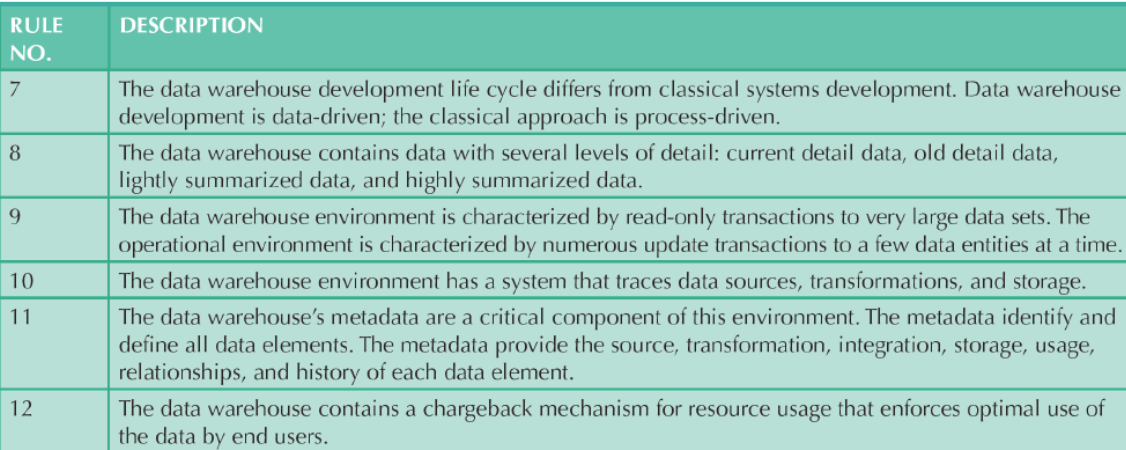


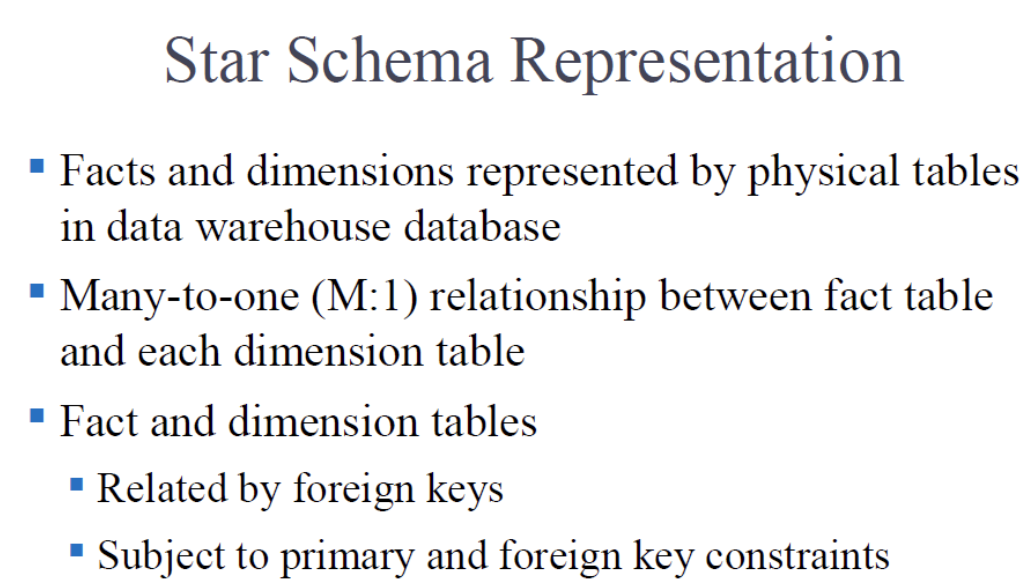


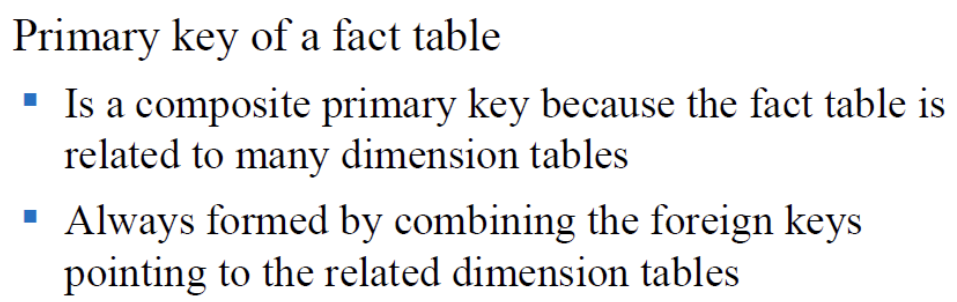
Data lake: - A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale. It enables you to analyze large volumes of data in its native format, without having to first structure it. This flexibility makes data lakes valuable for big data analytics and machine learning. YOU DUMP RAW DATA IN IT, later someone can read data and make schema from it. Data lake erases the first rule below.

12 rules:



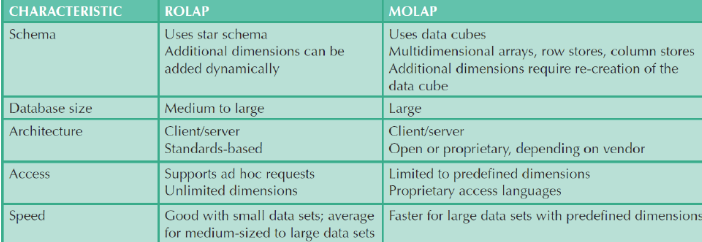






Fact and one dimension level: - Star schema

Dimension table split further: - Snowflake schema





SPATIAL DATA:

A spatial database is a database that is optimized to store and query data related to objects in space, including points, lines and polygons.

Types of Spatial Data

* Points, vertices, nodes
* Polylines, arcs, linestrings
* Polygons, regions
* Pixels, raster

GIS - Geographic Information System

Specific application architecture built on top of a general SDBMS.

Used for: search, location analysis, terrain analysis, flow analysis, distribution, spatial analysis, measurements

Characteristics of geographic data:

* has location
* has size
* is auto-correlated
* scale dependent
* might be temporally dependent too

Entity View vs Field View Two conceptions of space:

Entity View: space as an area filled with a set of discrete objects

Field View: space as an area covered with essentially continuous surfaces

All spatial data can be described via the following entities/types:

* points/vertices/nodes
* polylines/arcs/linestrings
* polygons/regions
* pixels/raster

Q. why would any three points be enough to describe a perfect circle?

A. Find midpoints of the lines after joining it. Draw perpendicular bisector. The intersection of perpendicular bisectors of any two chords of a circle uniquely determines the centre of the circle.

In 1D (and higher), spatial relationships can be expressed with the

following terms:

* Touch
* Inside
* Contains
* Disjoint
* Equals
* On
* Covers
* Overlap

Spatial relationships can be:

* topology-based [using defns of boundary, interior, exterior]
* metric-based [distance/Euclidian, angle measures]
* direction-based
* network-based [eg. shortest path]

Topological relationships could be further grouped like so:

* proximity
* overlap
* containment

Returns a Geometry

* Union
* Difference
* Intersect
* XOR
* Buffer
* CenterPoint
* ConvexHull

Returns a number

* Length
* Area
* Distance

DBs with Spatial Extenstions

* Oracle: Locator, Spatial, SDO
* Postgres: PostGIS
* DB2: Spatial Datablade
* SQL Server: Geometric and Geodetic types
* MySQL: Built-in Spatial Library
* SQLite: SpatiaLite

Spatial Indexing

Vastly speeds up processing of spatial data, by allowing us to search a

subset of the data

B-Trees (self-balancing generalized binary search tree (nodes can have

more than 2 leaves))

Issues

Sorting is not naturally defined on spatial data

Many efficient search method are based on sorting data sets

Solution

Space filling curves impose an ordering on the locations in a

multi-dimensional space

R-Trees:

Use MBRs to create a hierarchy of bounds

Contains NN R-Trees built up to big root R Trees

Smallest R-Nodes point to parents which contain them

KD-Trees: Space-partitioning data structure for organizing points in a kdimensional space

Quad Trees: Each node is a leaf node with indexed points or null, an

internal node has exactly 4 children

Minimum Bounding Rectangles (MBRs)

Bounding boxes used to estimate and compute spatial relationships

between objects

Used in the Filter and Refine step of Query Processing

Oracle Spatial

Oracle offers a 'Spatial' library for spatial queries

Handles Spatial Data Types, Spatial Analysis and Spatial Indexing in Oracle

DB, accessible through SQL

Oracle Spatial Data Types:

Networks (lines)

Locations (points)

Parcels (polygons)

Imagery (raster, grids)

Topological Relations (persistent topology)

Addresses (geocoded points)

Oracle Spatial Operators:

Filter (find objects by primary filter)

Relate (find all objects by primary and secondary (relational i.e. touch)

filter)

NN (find n nearest neighbors)

WithinDistance (find all objects within a distance of target)

Spatially Indexed with R-Trees

Postgres PostGIS Application

Spatial DB functionality add-on for Postgres

Supports Queries: distance, equals, disjoint, intersect, touches, crosses,

overlaps, contains, length, area, centroid

Google KML

Google's format for encoding spatial data

Spatial Data Visualizations

Dot Map: Map with points shown for events

Proportional Symbol Map: Map with different sized symbols for higher

probability regions

Diagram Map: Map with diagrams displaying information about a region

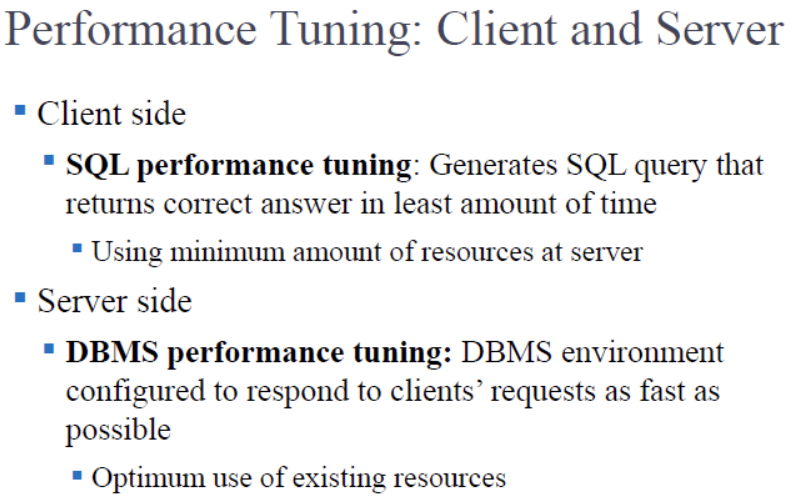
Choropleth Map: Map with different regions colored differently based on a

Metric

Q. What role do minimum bounding rectangles (MBRs) play, in spatial query processing (how are they used/helpful)?

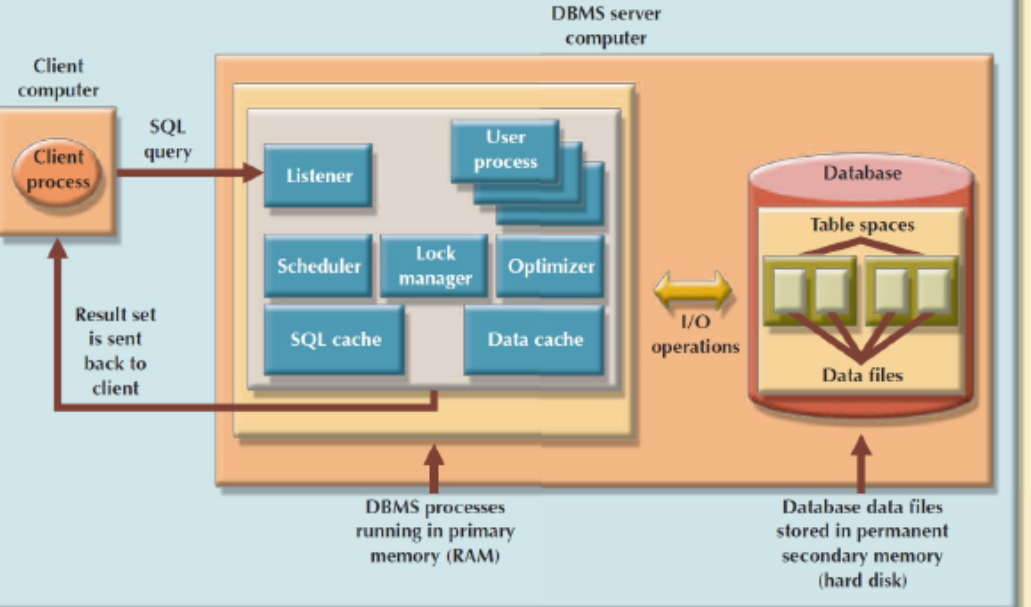
A. MBRs play a pivotal role in spatial query processing, primarily serving three key functions. Firstly, they facilitate **efficient indexing** by organizing spatial data structures like R-trees, which partition the spatial domain based on MBRs, thereby enabling quick retrieval. Secondly, MBR-based indexes expedite **retrieval and querying processes** by swiftly narrowing down the search space to relevant spatial objects, consequently reducing query response times. Finally, MBR comparisons **optimize various spatial operations** such as joins and range queries by efficiently identifying intersecting or relevant spatial objects, thereby streamlining overall query processing. Thus, MBRs contribute significantly to enhancing the efficiency and effectiveness of spatial query processing systems.

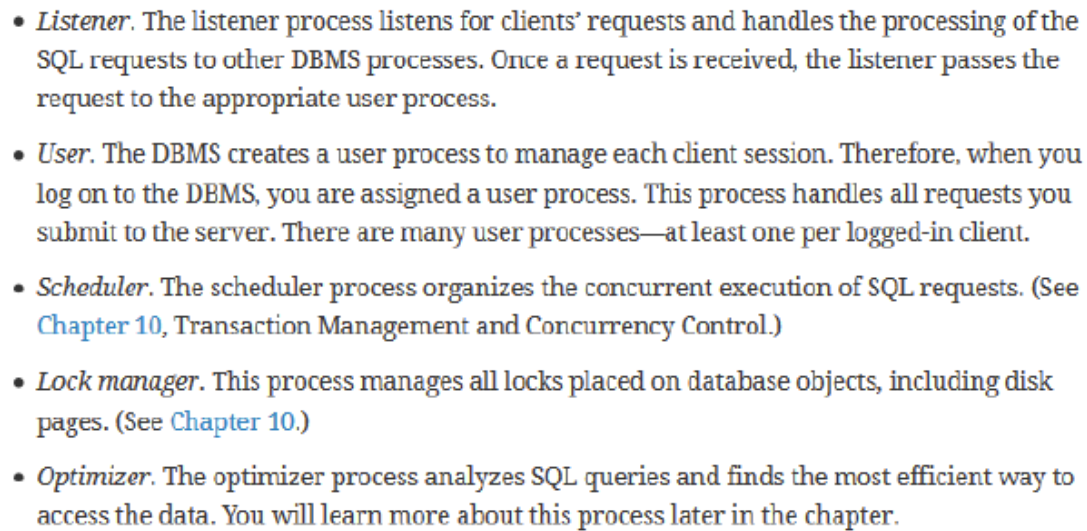
**LECTURE 12**

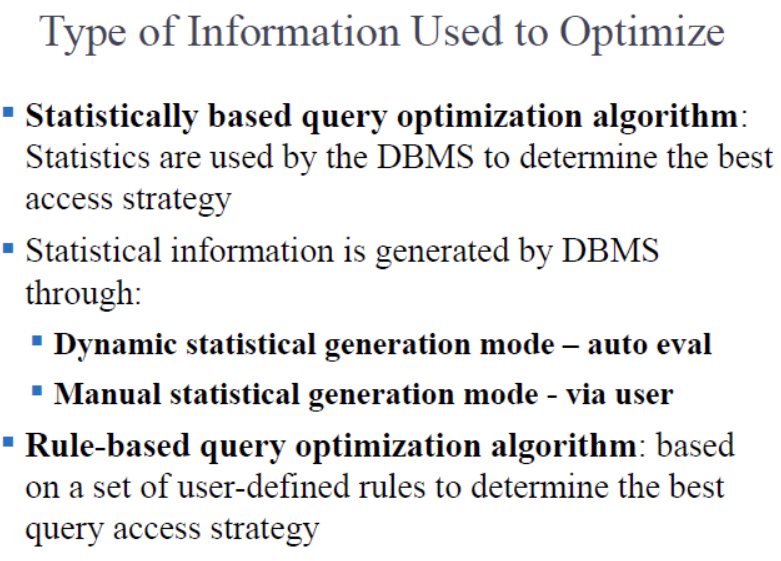


Client means the app that is getting the data.

Server side: - DBA can set some parameters like a requirements file. DB can obey those parameters to increase speed.

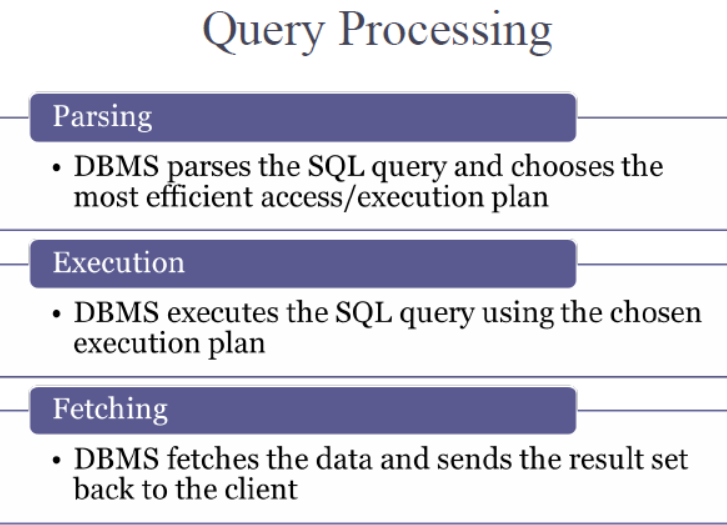


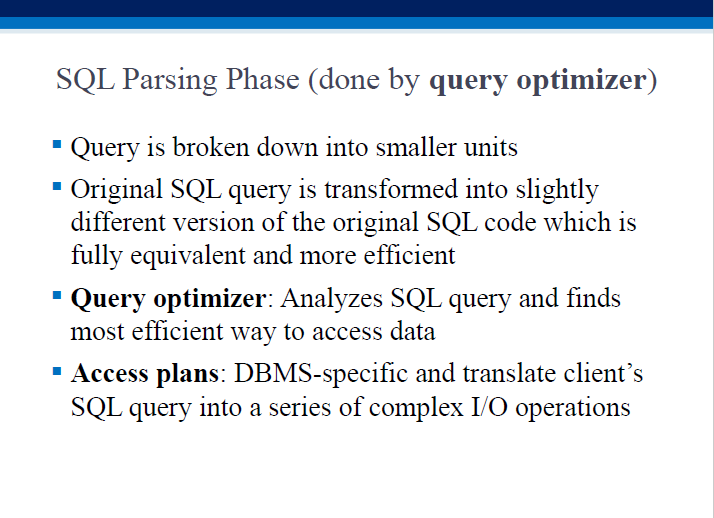


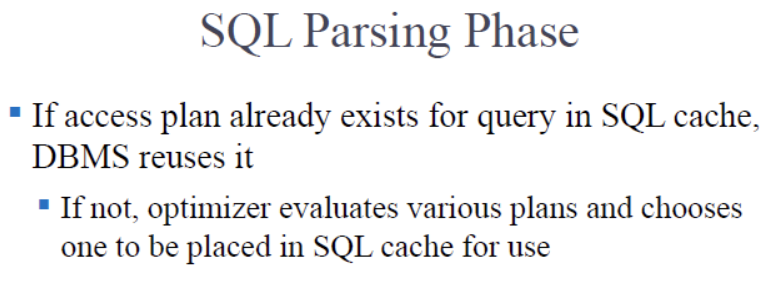


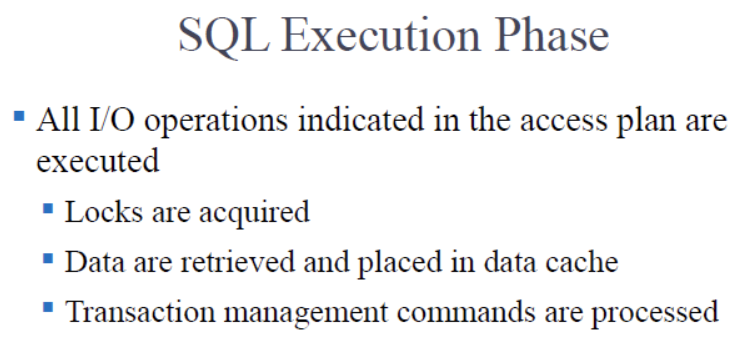
Q. What does this remind you of? What is the distinction between statistical and ruled based query optimization remind you of?

A. AI vs Machine Learning. AI has a notion of rules. AI is rule based. ML collects lots of data (eg: - brain images for tumor with normal brain images). This is statistics based. It learns from past data and then applies it on new images.

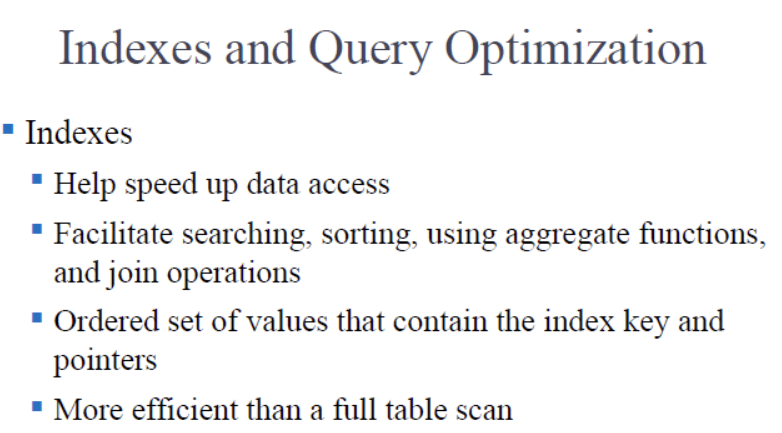












The reason for using an index is simple - provided we incur an upfront cost of creating (computing) one, runtime lookup costs using the index are vastly cheaper than doing full searches through non-indexed rows.

Eg. given this book, find all the places that discuss table updating..

Q. How would you fill 0…1000 numbers in 10 slots(0,1….9), collisions don’t matter?

A. 3 methods: -

1. Padding: - 1 becomes 001, 2 becomes 002….10 becomes 010. For each number, you can pick the rightmost digit. Suppose rightmost digit is 0, then 0, 010, 020… all go to slot 0.
2. Modulus: - Mod every number by 10.
3. Sin/cosine: - use sin cosine values of the numbers.

Index selectivity: a measure of how likely an index will be used in a query. So we strive to create indexes that have high selectivity..

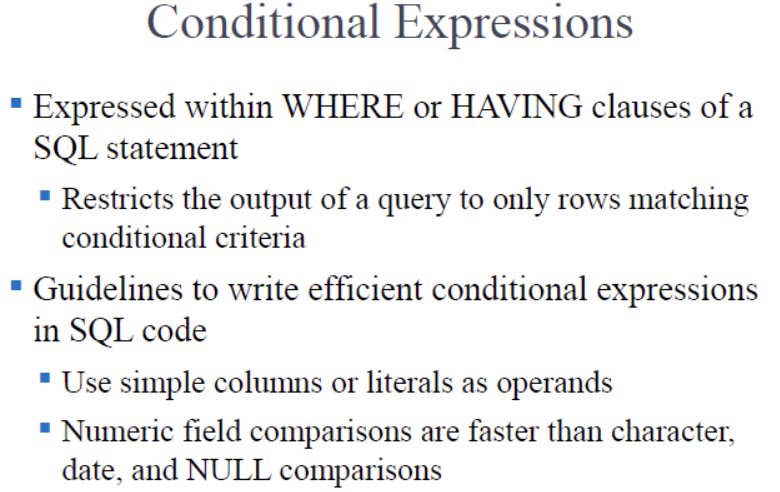
Indexes are useful when:

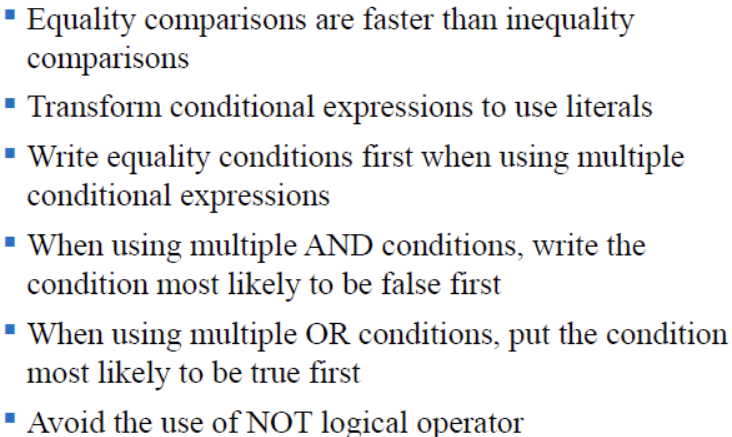
* an indexable column occurs in a WHERE or HAVING search expression
* an indexable column appears in a GROUP BY or ORDER BY clause
* MAX or MIN is applied to an indexable column
* there is high data sparsity on an indexable column

Index on LHS should be evaluated with all values shifted on RHS.

Eg: -

Price\*1.1>10 should be price>10/1.1 to yield better output.





Big Data

Data that is too big to be stored and/or processed by a single machine

Data that has variety, velocity, volume.

Uses of Big Data

Industry/Business Optimization

Data Source Combination

Exploitation of Unstructured Data (sound, video, images)

Provide insights

NOSQL: -

Need a flexible, efficient, available, scalable solution/DB design! THAT is what NoSQL provides - high performance, high availability at a large scale.

NoSQL Databases

Used to handle BigUsers, BigData, BigVariety, and BigFlux, where RDBMSs

are unsuited

Flexible, efficient, avaiable and scalable db design

Schema-less: no tables, no relations; NoSQL doesnt care what its storing;

schema is implicit - it lies in application code that reads and writes data

Flexible: easy to add new types of data

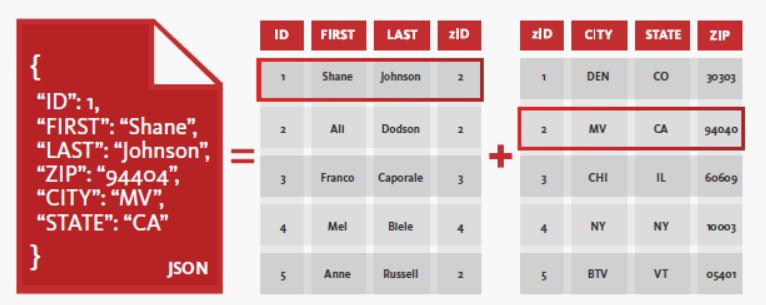
Scalable: ability to horizontally scale (adding more nodes)

Fast: easy to processe large volumes of data

JSON: JavaScript Object Notation

In JSON, table can have blank values.

Eg:



Keys always have to be string. Values can be number, string, Boolean, array, object.

Array can have mixed elements.

Eg: -

[

{

"name": "Alice",

"age": 25,

"city": "New York"

},

{

"name": "Bob",

"age": 30,

"city": "Los Angeles"

},

{

"name": "Charlie",

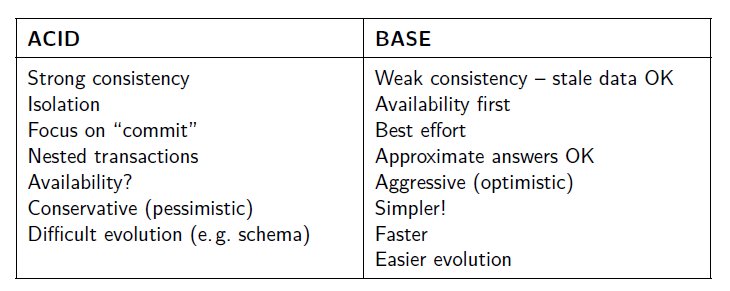
"age": 35,

"city": "Chicago"

}

]





Key-Value DBs

Database is constructed of keys which represent a value in the db

Querying occurs on keys only

Entire value for matching key is returned when queried

Entire value for matching key must be updated when its necessary to make

changes

Memcached, Redis, DynamoDB

Column Family DBs

Rather than dealing with rows of data, we deal with columns of them

Good for aggregate queries and queries involving a subset of all columns

Components:

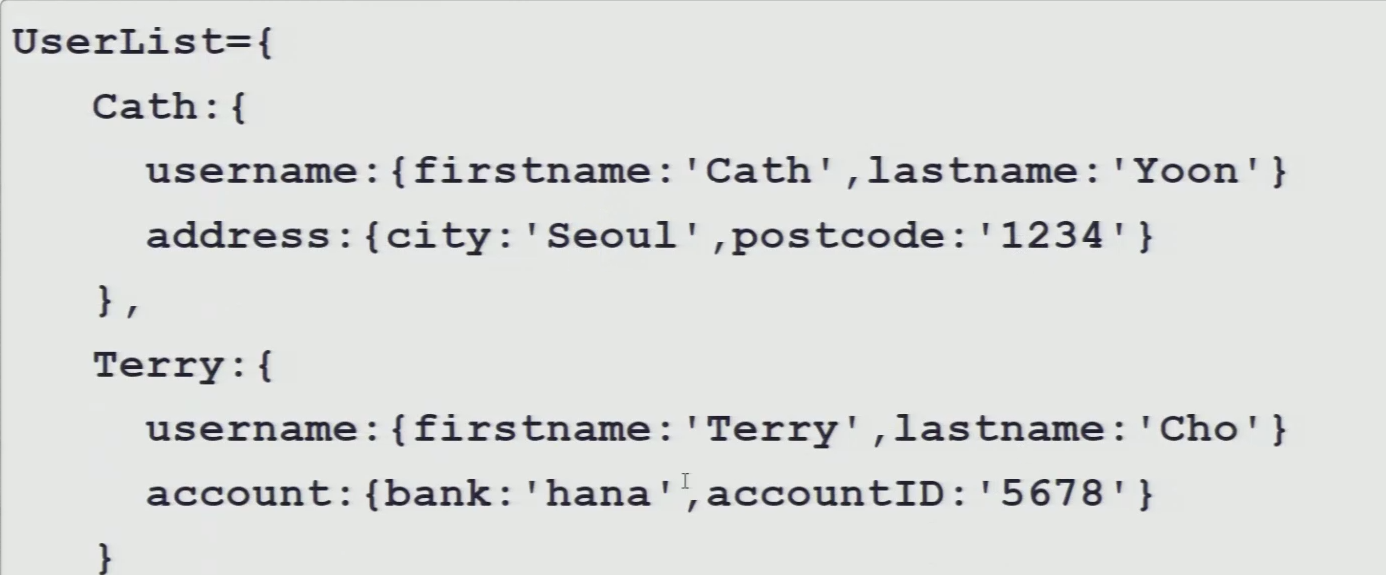
Columns - consist of a name (key) and value

Supercolumn - group of columns

Column Family - contains columns of related data

Supercolumn Family - collection of supercolumns

BigTable, HBase, Cassandra



Document DBs

Collection of documents with an arbitrary number of fields in each

document

Documents can be JSON, XML, etc

CouchBase, MongoDB

GraphDBs

Data structure comprised of verticies and edges

Neo4j, Giraph, FlockDB, Tinkerpop

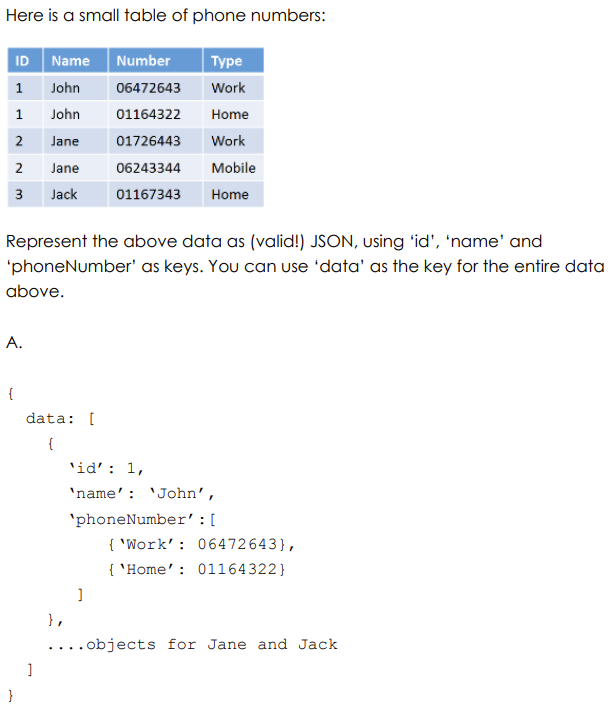
Most flexible way to model any data

Polyglot Persistence

Use of different storage technologies to store different parts of a single

application.

Individual data stores are tied together by the application using DB APIs



Q. Usually in Big Data processing, we would employ horizontal fragmentation

(split up a relation into groups of rows) to speed up processing. But we also

use a strategy where we do vertical fragmentation (where we split

columns).

a. What is this type of NoSQL database called?

A. Column family database.

b. How does this strategy help process queries faster (what is the

advantage of doing this)?

A. Column families (groups of columns) that are accessed more frequently

can be kept in a separate file, and non-essential columns in another file.

The essential columns file can then be stored in high speed memory (eg.

SSD) and accessed faster.

Q. Recall ‘polyglot persistence’, when it comes to NoSQL DBs. What two

[somewhat inter-related, but distinct] reasons can you think of, for this to

be a ‘bad’ thing (eg. why a company wouldn’t choose to encourage this

in their data infrastructure)?

A. By possibly needing to ‘port’ code from one language to another (eg.

when a decision is made to switch a part of the application from Python to

R), bugs and inefficiencies could creep in; also, the lack of a standard

modeling+query language such as SQL, creates lowered productivity

overall (solutions can’t be reused across other in-house applications,

algorithms need to be coded up from scratch...).

Q. Before NoSQL, almost all databases were based on the relational model:

tables (entities), and PK/FK relationships between them. The NoSQL paradigm offers us

alternatives. There is one specific ‘freedom’ that NoSQL offers, from an architectural

standpoint (ie. in the design, and redesign, of a large, complex application). What is it?

Be specific, and describe it in a paragraph or two, with an illustration.

A. We have the freedom to implement a mixed-model architecture for our data

storage, ie. we can have ‘polyglot persistence’. With polyglot persistence, we don’t

need to decide on a single type of NoSQL database for the entire application; instead,

based on the data to store, we can pick the appropriate type (k-v, column, document

or graph).

Q. What is the most flexible way to model (‘any’) data? And, what structure

can be used to do so?

A. A graph DB offers the most flexible way, via nodes and edges to model data and

relationships.

Specifically, a ‘triple store’ ie. (subject,predicate,object), can be used.

DATA MINING: -

Q. How is ML different from DM?

A. Machine learning (ML) and data mining (DM) are related fields within AI and data science, but differ in focus and methods:

* Focus: ML develops algorithms for learning from data and making predictions, while DM focuses on extracting patterns and insights from large datasets.
* Applications: ML is used in predictive analytics, NLP, computer vision, etc., while DM is common in market analysis, fraud detection, and customer segmentation.
* Methods: ML includes supervised, unsupervised, and reinforcement learning, while DM uses techniques like clustering, association rule mining, and anomaly detection.
* Scope: ML is broader, covering various tasks like prediction and classification, while DM focuses specifically on exploratory data analysis and knowledge discovery.

Practically all data mining ('DM') algorithms neatly fit into one of these 4 categories:

Classification: involves LABELING data

Clustering: involves GROUPING data, based on similarity

Association: involves RELATING data

Regression: involves COUPLING data [incl finding 'outliers']

Data Mining vs Statistics

Statistics is about summarization of data.

In statistics, we collect and analyze numerical data of a smaller

representative sample for the purpose of inferring proportions in a whole.

In Data Mining, we don't summarize or make inferences about a larger

population, we analyze all available data and look for patterns in it.

Data Mining vs Machine Learning

Data Mining is a subset of Machine Learning.

Data Mining stops with the discovery of patterns in data.

In Machine Learning, we publish the model that we mine, and continue

processing new incoming data, using that model.

DECISION TREE:

Classification and regression trees (aka decision trees) are machine-learning methods for constructing prediction models from data. The models are obtained by recursively partitioning the data space and fitting a simple prediction model within each partition.

The decision tree algorithm works like this:

user provides a set of input (training) data, which consists of features (independent parameters) for each piece of data, AND an outcome (a 'label', ie. a class name)

the algorithm uses the data to build a 'decision tree' [with feature-based conditionals (eqvt to 'if' or 'case' statements) at each non-leaf node], leading to the outcomes (known labels) at the terminals

the user makes use of the tree by providing it new data (just the feature values) - the algorithm uses the tree to 'classify' the new item into one of the known outcomes (classes)

SVM: -

There is a data: blue or red. SVM gives a strip that maximizes the separation between blue data and red data.

New point will inherit properties of the clusters.

KNN: -

kNN (k Nearest Neighbors) algorithm picks 'k' neartest neighbors, closest to our unclassified (new) point, considers the 'k' neighbors' types (classes), and attempts to classify the unlabeled point using the neighbors' type data - majority wins (the new point's type will be the type of the majority of its 'k' neighbors).

kNN is a 'lazy learner' - just stores input data, uses it only when classifying an unlabeled (new) input. VERY easy to understand, and implement!

Checks odd number of neighbors.

NAÏVE BEYES: -

The 'naïve' Bayes algorithm is a probability-based, supervised, classifier algorithm

K-Means Clustering

Creates k clusters from input data using a measure of closeness

Unsupervised algorithm

Hierarchical Clustering

Seperates data set into hierarchical clusters by merging smaller clusters

into bigger superclusters or dividing large clusters into smaller ones

A Priori

Looks for hidden relationships in large data sets

Generates association rules between data

Collaborative filtering:

Collaborative filtering is a method used in recommendation systems to suggest items or content to users based on their past preferences and behaviors, as well as the preferences and behaviors of similar users. Rather than relying on explicit information about items or content, collaborative filtering algorithms analyze patterns of interactions among users and items. This approach assumes that users who have similar preferences in the past are likely to have similar preferences in the future. It's widely used in platforms like Netflix, Amazon, and Spotify to personalize recommendations for users, enhancing their overall experience by suggesting relevant movies, products, or songs they might enjoy based on the collective behavior of similar users.

Linear Regression:

This (mining) technique is straight out of statistics - given a set of training pairs for a feature x and outcome y, fit the best line describing the relationship between x,y. The line describes the relationship/pattern.

Logistic Regression

Classification algorithm where we compute regression coefficients

corresponding to a decision boundary, then use that to compute an

outcome for new data, and finally transform the outcome to a logistic

regression value between 0 and 1 to predict a binary outcome

Expectation Maximization

EM is an algorithm that can solve for hidden variables in a model by using

collected data

Process

start with random values for the model parameters

use these compute probabilites for all possible values for each hidden

var, and then do a weighted average to compute the best value for

each latent var

use the hidden vars values to improve the model parameters

iterate the above two steps till values converge