Homework 2.1

Memcached:

Memcached is a using of cache and cache the data into the memory instead of disk or SSDs. The time performance is better because it is eliminating the need to access the disk, so Memcached can access the data in microseconds. Memcached is also distributed, meaning that is easy to scale out by adding new nodes. Memcached logistically put all the node together, so less wasting on the memory usage. Memcached is also multithreaded, so we can easily scale up the compute capacity by use multiple cores on a given node. It is also open-source that mean it supported by a lot of languages. Its most popular use cases are caching and session store. It is good for Caching because it implement higher performance in-memory cache and decrease the data access latency, increase throughput and ease the load off back-off system. Since Memcached provide small latency and scale, it is a good choice for internet applications in cases where persistence is not critical, it usually used to manage session data like user profile, credentials, and session state.

Different between Memcached and Redis:

Redis and Memcached are the two most popular int-memory key-value data stores. Memcached is designed for simplicity while Redis offers a rich set of features that make it effective for a wide range of use cases. Redis offers Advanced data structures like Strings, lists, sets…, but Memcached not. Memcached is multithreaded, it can make use of multiple process cores. Redis also have good features that Memcached doesn't have like snapshots (used for archiving or recovery), Replication (can create replicas of a Redis primary, scale the databases read and highly available clusters), Transaction (execute a group of commands atomically), Pub/Sub (messaging with pattern matching, chatrooms and streams), Lua scripting (boost performance and simplify the application) Geospatial support (get the distance, and find elements in a given distance.)

Sources:

https://aws.amazon.com/elasticache/redis-vs-memcached/

https://aws.amazon.com/memcached/

https://memcached.org/about

Homework 2.2

Vertical scaling vs Horizontal scaling:

Horizontal scaling (HS) means scaling by adding more machines to your pool (scaling out), and vertical scaling (VS) refers to scaling by adding more power (CPU, RAM) to the existing machines (scaling up). The horizontal scaling required to break a logically data into smaller pieces so that can be executed in parallel across multiple servers. Vertical scaling is just increase the size only, so it is easier because logic doesn't need to change. From Databases wise: HS scaling is bases on the partitioning of the data, and VS is just using the CPU and RAM resources of the machine to scaling the data. For Downtime wise:

HS means not limit to the single unit's capacity and making it possible to scale with less downtime. VS is limited by the machine, and it could have upper limit. For Concurrency: HS is distributed jobs across the machines, but VS is performed with multi-core and with multi-threading and in-process message passing. For Message passing: HS have a lot of machines with make data sharing more complex and costly. VS can do that work in by multi-thread can be done by passing a reference. Example Using HS are Cassandra, MongoDB, Google Cloud Spanner. For VS, MySQL, Amazon RDS…

Hierarchical data store:

Hierarchical database (HD)is a data model in which data is stored and organized into a tree-like structure, or parent-child structure, parent can have multiple child nodes. HD is a tree-like, and Relational databases is stored in tables with unique identifier for each record. Relational databases have easy identification and access related data in the database. When in HD, a parent record can have several child records, but each child record can only have one parent record. Data stored in the form od fields, and each field can only contain one value.

BASE:

NoSQL relies upon a softer model called BASE model (Basically Available, Soft state, Eventual consistency.) Basically Available: Guarantees the availability of the data, there will be a response for any request. Soft state: the state of the system can change over time. Eventual consistency: the system will become consistent once it stops receiving input.

Sources:

https://www.section.io/blog/scaling-horizontally-vs-vertically/

https://drib.tech/programming/hierarchical-data-in-relational-databases

https://www.freecodecamp.org/news/nosql-databases-5f6639ed9574/

Homework 2.3

View and stored procedure:

A view represents a virtual table. Giving a view and use the view to present data as if the data is from the data or not. Does not accept parameters can by used as building block in a larger query, can contain only SELET query, can not perform modification, can use a the target of an INSERT, UPDATE or DELETE statement.

Stored Procedure using parameters to do a function, it is updating and inserting data, or returning single values or data sets. Accepts parameters, can Not be used as building block in a larger query. Can contain several statement like if else, and loops, can preform modifications to one or several tables, and can not be used as an INSERT, UPDATE or DELETE statement.

Sources:

https://stackoverflow.com/questions/5194995/what-is-the-difference-between-a-stored-procedure-and-a-view