A screen shot of a computer

Description automatically generated<https://www.youtube.com/watch?v=W2Z7fbCLSTw>

7 Database Paradigms

1. Key-value (Redis)
   1. Javascript Object
   2. Stored in RAM not Disk
      1. A screenshot of a computer

         Description automatically generatedPro: Fast
      2. Cons: Limited memory, no queries
   3. Used for caching
2. Wide Column
   1. Like a relational but doesn’t have a schema
   2. Can scale horizontally
   3. Good for time series, historical records (Netflix storying history)
   4. A diagram of a document

      Description automatically generatedNo joins (?)
3. Document (MongoDB, Firestorm, DynamoDB?)
   1. Document is a container for key, value pairs
   2. Unstructured and doesn’t require a schema
   3. Grouped together inside a collection
   4. Cons: doesn’t support joins, so instead of normalizing your data, encourage to embed the data into a single document. Pro: reads from a front end app are much faster but writing/updating is more complex
   5. Not ideals for graphs (social networking apps)
4. Relational (MySQL, postgres)
   1. Inspired the development of SQL (Structured Query Language)
   2. Foreign keys, because they reference data in a different table
   3. A screenshot of a computer

      Description automatically generatedOrganizes data in its smallest normal form
   4. Requires a schema
   5. ACID (Atomicity, …)

SQL vs NoSQL

<https://www.youtube.com/watch?v=_Ss42Vb1SU4>

SQL

-Structured Query Language

-Table Based

-Vertical Scaling (increase size of instance)

-Relational

* allows easy querying on relationships between data amount multiple tables

-Data is structured

* therefore potential error is reduced
* because SQL schemas require the data model and format of the data to be known before storying anything

-ACID compliant

* Atomicity (all or nothing transactions)
* Consistency (Data is valid before or after)
* Isolation (multiple transaction at the same time)
* Durability (committed data is never lost)

-SQL transaction are groups of statements that are executed atomically (meaning all or nothing if any statement in the group fails)

Cons

-Structure (columns/tables) must be created in advance

-Not effective for storing and querying unstructured data

-Difficult to scale horizontally because of their relational nature. For read heavy systems its straightforward to provision multiple read only replicas but write heavy systems usually the option is to vertically scale the database up which is generally more expensive than provisioning additional servers

noSQL

-document, key value, graph, or wide column

-Horizontal Scaling (adding more instances)

-More flexible and easier to setup

-Data sharding (technique where a large dataset is divided into smaller), makes it more easier to horizontal scaling

Cons

-Loss of consistency

-Eventual Consistency

Comments:

-SQL can scale horizontally well

Horizontally scaling: can add more servers or nodes to DB cluster to handle increased data volume, traffic

Vertical Scaling: adding more CPU, RAM or storage capacity to a single server to handle increased demands

-MongoDB is ACID after version 4?

A black screen with white text

Description automatically generated