Cloud Applications Architecture

Course 8 - Databases

Introduction

What is a database?

What is a DBMS?

Types of Workloads

Online Transactional Processing (OLTP)

The most common workload
Examples: banking, ERP, booking, billing

Characteristics:

- Transactional behavior
  - If 2 people try to book the same place, only one succeeds.
- Known access patterns
  - The uses cases of the system are well-defined
- Write-heavy
- Known data schema
- Short operations (~milliseconds)
- Back-up is highly important

Online Analytical Processing (OLAP)

Used to be relevant only for large businesses.
Based on data warehouses (central data repository).

Examples: analyze sales in a certain period/location.

Characteristics:

- Access patterns/use cases not known entirely (Ad-hoc)
- Read heavy
- **Large** data volumes
- Possibly long operation (minutes, hours)
- Data consistency & back-up not crucial

Decision Support System (DSS)

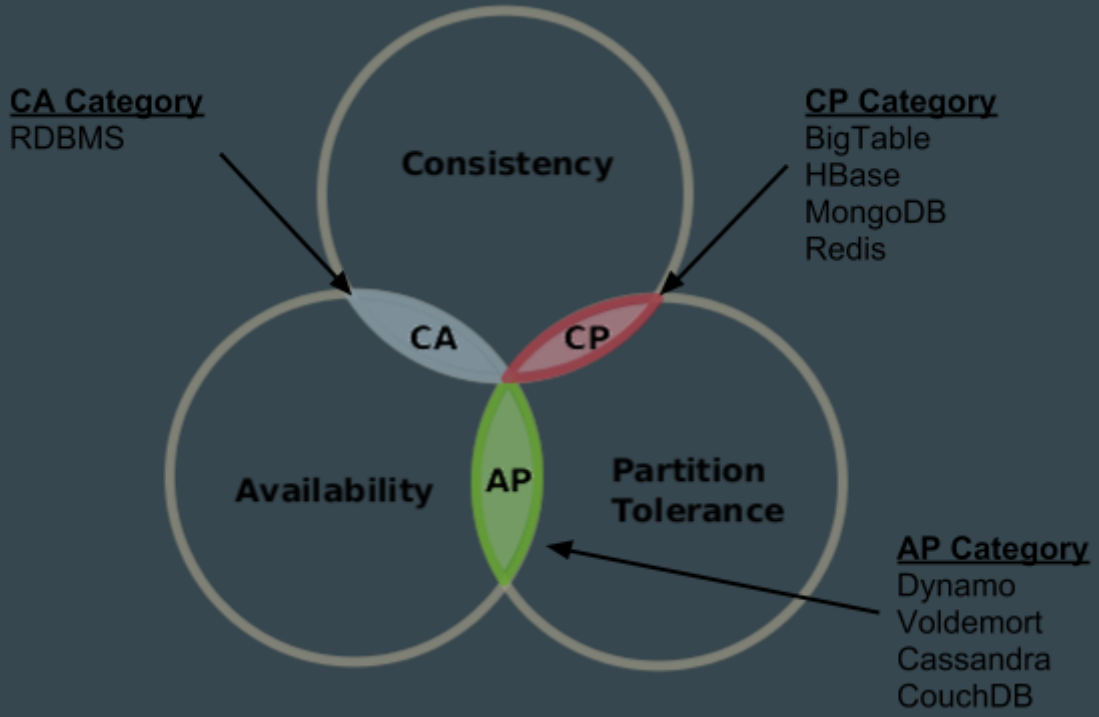
Similar characteristics to OLAP

- Queries might be known since the beginning
- Data spans longer timeframes (more historical data)

Types of Databases

Recall

CAP Theorem



ACID vs BASE

Relational

Based on **SQL** (Structured Query Language)

- Declarative

RDBMS

Optimized for storage (reduced data duplication - normalization).

Examples:

- Open source: PostgreSQL (postgres), MySQL, MariaDB, SQLite, H2
- Commercial: SQL Server, Oracle, DB2, Cloud Spanner, HANA, Aurora
- Commercial extensions: Percona (for MySQL), Citus (for Postgres clusters)

NoSQL

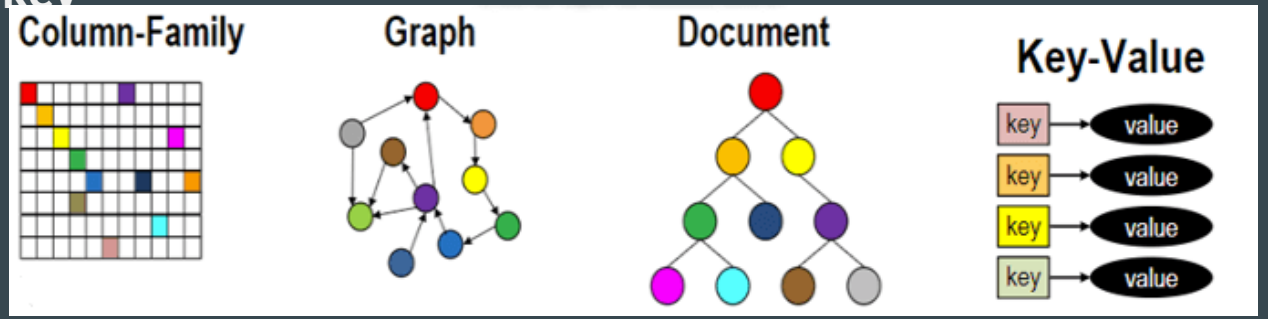
Started as a twitter tag

Stands for **N**ot **O**nly **SQL**

Covers a wide range of database types

Optimized for **compute** & **scalability**

All are based on **partition key**



Key-Value

Data is always accessed based on a key.

DBs can be scaled infinitely (since there is no relation between items).

Ultra low and **consistent** latency. (sub-millisecond)

- Especially for some in-memory databases

High and **consistent** throughput.

Examples:

- Caching systems: Redis, Memcached
- Amazon Dynamo

Column

There are 2 (somehow similar) types:

- Column stores

Stores data tables by column rather than by row
Serializes all of the values of a column together, similar to an index organization for row oriented dbs
Reduces the amount of data read from disk by compressing the similar columnar data and by reading only the necessary data
Examples

- HBase, Cassandra
- MariaDB ColumnStore

Column

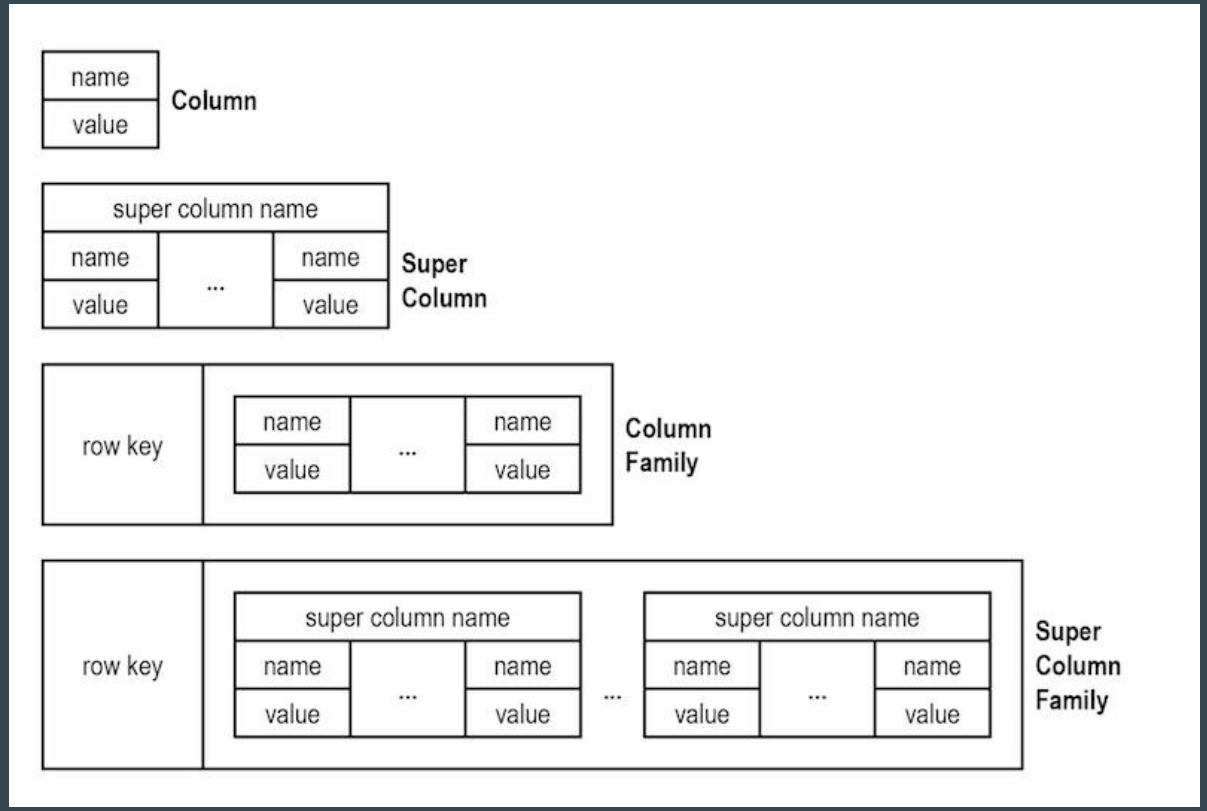
- Wide column databases

Has an architecture that uses persistent, sparse matrix, multi-dimensional mapping (row-value, column-value, and timestamp) in a tabular format meant for massive scalability (over and above the petabyte scale)

Examples:

- BigTable, Cassandra, Accumulo

Column


name
value
Column
super column name
name
value
...
name
value
Super Column
row key
name
value
...
name
value
Column Family
row key
super column name
name
value
...
name
value
...
super column name
name
value
...
name
value
Super Column Family

Document

Focus on storage and access methods optimized for documents

Data modeled as collections of documents containing key-value pairs.

Values can be scalar values but also nested documents or lists as well as

Attribute names are dynamically defined for each document (no schema enforcement)

Examples

- MongoDB
- CouchDB
- Elasticsearch ！

Graph

Use topographical data models to store data

Connect specific data points called nodes

Create relationships called edges in the form of graphs

- contains nodes and relationships
- nodes contain properties (key-value pairs)
- relationships are named and directed, and always have a start and end node
- relationships can also contain properties

Examples
- Neo4J
- OrientDB

Time-Series

Optimized for time-stamped data

- The data that arrives is almost always recorded as a new entry
- The data typically arrives in time order
- Time is a primary axis (time-intervals can be either regular or irregular)

Time-series data workloads are generally “append-only.”

Data could be measurements or events tracked, monitored and aggregated over time (server metrics, network data, sensor data, events, clicks, etc)

Examples
- InfluxDB
- TimescaleDB

Search

Dedicated to the search of data content

Use indexes to categorize the similar characteristics among data and facilitate search capability

Optimized for dealing with data that may be long, semistructured, or unstructured

Offer specialized methods such as full-text search, complex search expressions, and ranking of search results

Examples

- Elasticsearch
- Solr

Ledger

Key characteristics of a ledger database:

- Immutable: the past doesn’t change. Each write to the database is appended to the past.
- Transparent: there is access to the past. The log information can be retrieved.
- Verifiable: offers a way to validate the complete history of changes.

The database contains the log and tables which are a view into the log’s data

It’s not enough to have access to historical data, you must be able to verify the authenticity of that history (cryptographically verifiable transaction log owned by a central authority).

Understand Security

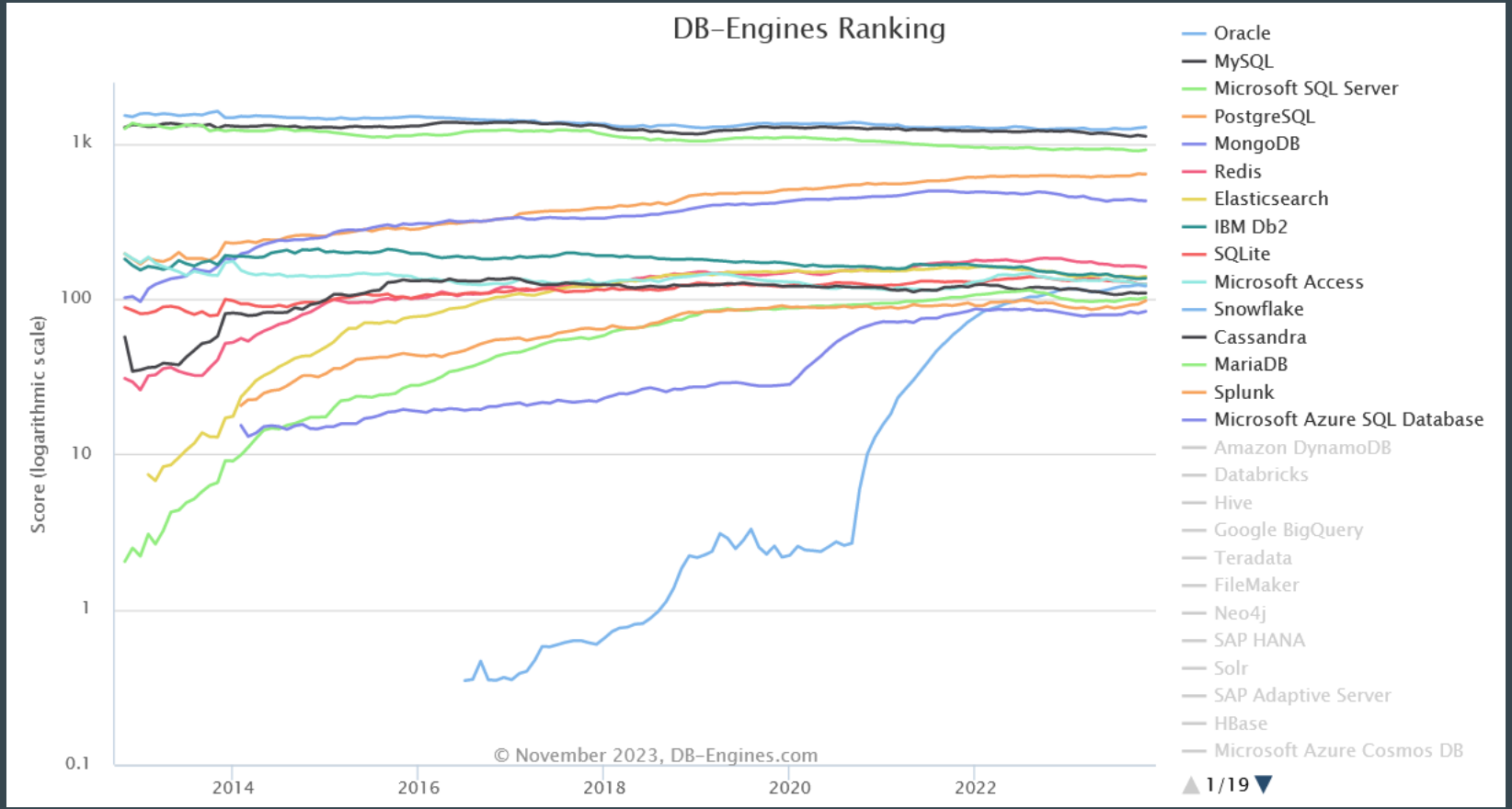
- Administrative user or authentication is not enabled by default.
- It has a very weak password storage
- Client communicates with server via plaintext(MongoDB)
- Cannot use external encryption tools like LDAP, Kerberos etc
- Lack of encryption support for the data files
- Weak authentication both between client and the servers
- Vulnerability to SQL injection
- Denial of service attacks

Examples
Data breach1
Data breach 2

416 systems in ranking, November 2023

Rank Nov 2023	Rank Oct 2023	Rank Nov 2022	DBMS	Database Model	Score Nov 2023	Score Oct 2023	Score Nov 2022
1.	1.	1.	Oracle	Relational, Multi-model	1277.03	+15.61	+35.34
2.	2.	2.	MySQL	Relational, Multi-model	1115.24	-18.07	-90.30
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	911.42	+14.54	-1.09
4.	4.	4.	PostgreSQL	Relational, Multi-model	636.86	-1.96	+13.70
5.	5.	5.	MongoDB	Document, Multi-model	428.55	-2.87	-49.35
6.	6.	6.	Redis	Key-value, Multi-model	160.02	-2.95	-22.03
7.	7.	7.	Elasticsearch	Search engine, Multi-model	139.62	+2.48	-10.70
8.	8.	8.	IBM Db2	Relational, Multi-model	136.00	+1.13	-13.56
9.	9.	↑ 10.	SQLite	Relational	124.58	-0.56	-10.05
10.	10.	↓ 9.	Microsoft Access	Relational	124.49	+0.18	-10.53
11.	11.	↑ 12.	Snowflake	Relational	121.00	-2.24	+10.84
12.	12.	↓ 11.	Cassandra	Wide column, Multi-model	109.17	+0.34	-8.96
13.	13.	13.	MariaDB	Relational, Multi-model	102.09	+2.43	-2.82
14.	14.	14.	Splunk	Search engine	97.32	+4.95	+3.10
15.	15.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model	83.17	+2.24	-0.49
16.	16.	↓ 15.	Amazon DynamoDB	Multi-model	82.24	+1.32	-3.16

Source: https://db-engines.com/en/ranking



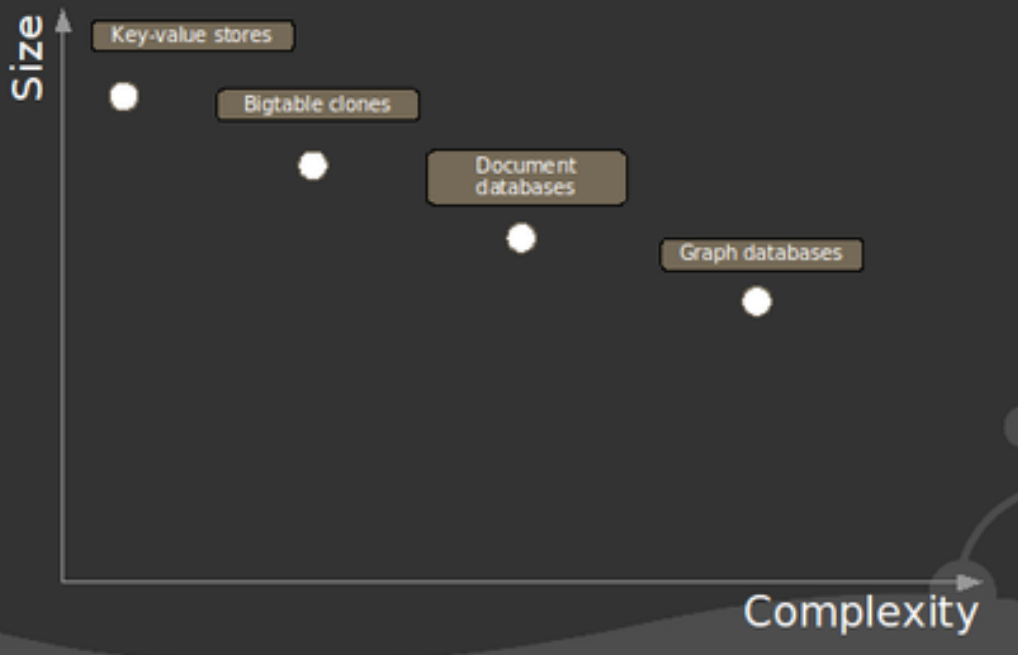
Source: https://db-engines.com/en/ranking

Choosing the Right Database(**s**)

Considerations

1. The purpose of the system (e.g. is it a social network?)
2. Are most access patterns known?
3. Performance
4. Volume
5. Does a user work mostly with his/her data? (as opposed to accessing the same data as everyone else - e.g. a leaderboard)
6. Scale/intended reach (local, regional, global)
7. Does one model fit all cases?

NOSQL data models



Managed Databases

Migrating to Cloud

Lift & Shift

Each cloud provider has its own tools to facilitate migration. Examples:

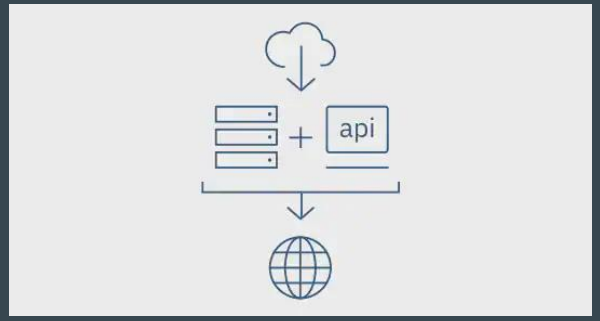
- AWS Database Migration Service
- Azure Database Migration Service

Usually support both homogeneous and heterogeneous source/target DBs

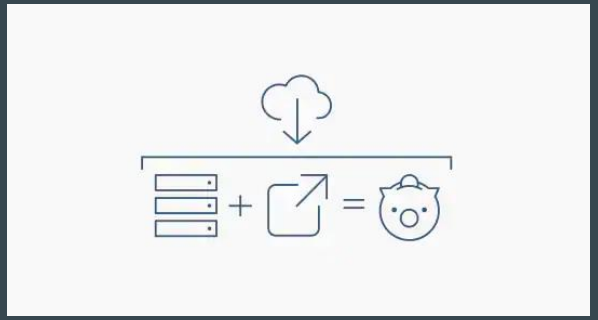
Disaster recovery strategies/services can also be used

Advantages

Easy access



Scalable



Disaster safety

Must Watch

Martin Fowler’s Intro to NoSQL

AWS re:Invent talks:

- DAT301
- DAT403-R1
- DAT205-R1