Cloud Computing for Beginners

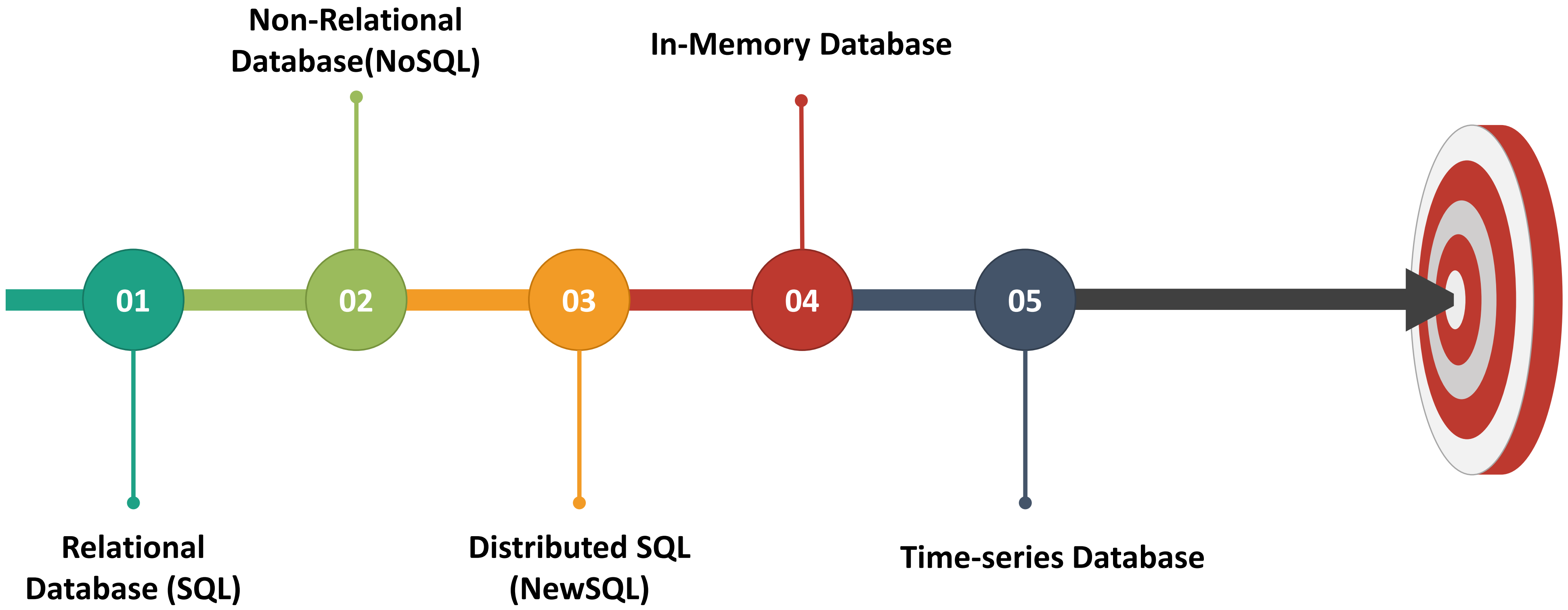
Database Technologies

By Idan Gabrieli

Database Technologies

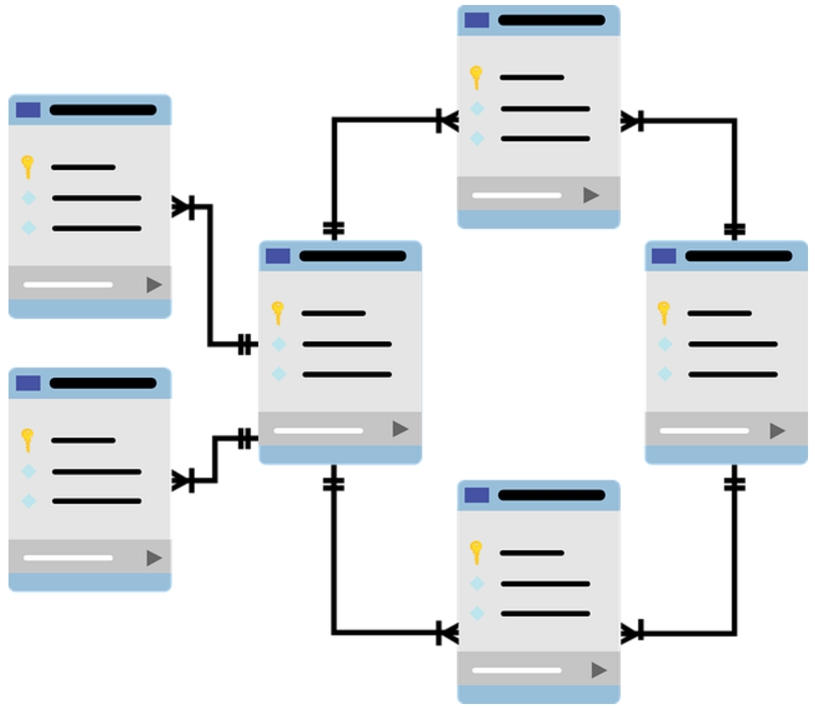
What, Where, and Why?

Database Technologies



Traditional Relational Database (SQL)

- A **relational** database stores data **in tables**
  - Each table is looking like an excel sheet with rows and columns
  - Each database will have a **predefined schema**
  - One or more columns in each table are used as the **primary key**
  - Rows in a table can be linked to rows in other tables using **foreign keys**



Traditional Relational Database (SQL)

A Database Online Shop

Products

primary key

Product Num, **Description, Price, Inventory…..**

Customers

Customer ID, Name, Phone, Address…

primary key

Orders

primary key

Foreign key

Foreign key

Order Num, Price, Date, Customer ID, Product Num

Traditional Relational Database (SQL)

- SQL - Structured Query Language
- Relational databases are a very popular option, specifically while handling operational data (**transactions)**
- The predefined database schema model is helping to avoid errors and keep things organized
- Popular SQL databases:

MySQL MariaDB PostgreSQL Microsoft SQL Server

Traditional Relational Database (SQL)

Who is using relational databases?

- **Enterprise Companies**
  - Many types of applications installed on-premise/private cloud
  - The dominate database technology is **relational database**
- **Internet Web Players**
  - Serving **millions of customers** worldwide
  - Having more challenges with scalability issues
  - Websites are expected to be **on-line and available all the time**
  - Traditional relational databases are NOT a great fit those such requirements

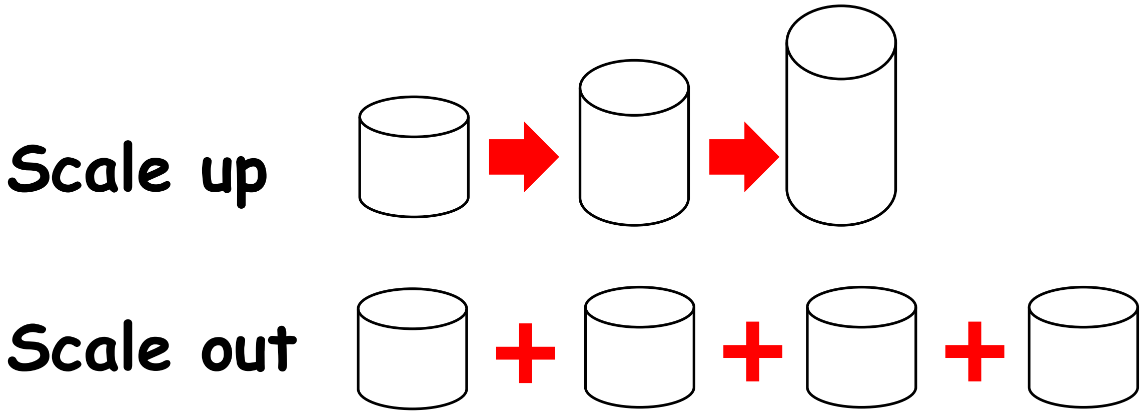
Traditional Relational Database (SQL)

What's the Problem?

- **Scalability!**
  - One of the biggest challenges with traditional relational databases is **scalability**
  - Only **vertical scaling** (scale up/down)
    - Bigger and bigger machines
    - Unacceptable downtime when scaling up
- For web applications availability and performance are critical requirements
  - Looking for alternatives
  - Moving to non-relational databases…

Non-relational Database (NoSQL)

- **Amazon, Facebook, Google….**
  - **BIG** internet players with **BIG** data challenges
  - Scaling up a database was an **inefficient and expensive**
- **Non-relational Databases**
  - Supporting web-based applications
  - Let's focus on two main things: **scale-out** and **Availability**



Non-relational Database (NoSQL)

- **Commercial and open-source** non-relational databases

Google Cloud Bigtable

cassandra

APACHE HBASE

mongoDB®

Amazon DynamoDB

Azure Cosmos DB

Non-relational Database (NoSQL)

What is the meaning of a **non-relational** database?

- **Simple Data Models**
  - No predefined strong schema model
  - Using more simple data models when storing data in the database
- **NoSQL Database**
  - "No support for SQL" and then evolved to "**Not only SQL**”
- **Types of NoSQL Databases**
  - Key-value
  - Document
  - Wide column
  - Graph

Non-relational Database (NoSQL)

What is the meaning of a non-relational database?

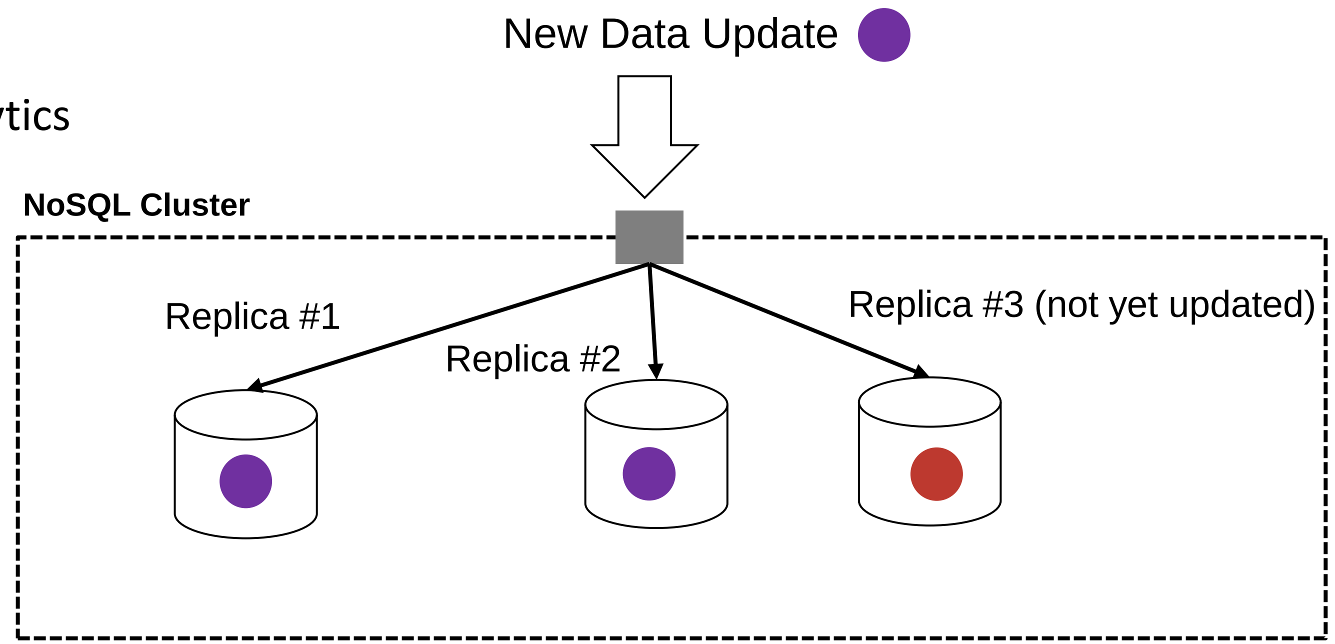
- **Advantages**

Schema-on-Read	Speed	Scaling Out	High Availability
SQL – Schema-on-Write NoSQL – Schema-on-Read	Fast Read/Write	SQL – Scaling Up NoSQL – Scaling Up & Out	No single point of failure Data replication

Non-relational Database (NoSQL)

What is the meaning of a non-relational database?

- **Disadvantages**
  - **Not Transactional !!!**
    - Not comply with ACID model
  - **Eventual Consistency Model**
    - “Weaker” consistency model
    - All DB cluster will be synced with the SAME replica **AFTER** some **convergence time**
    - Reading data may not be consistence
- **NoSQL Use Cases**
  - Mainly used for data analytics



SQL and NoSQL Database

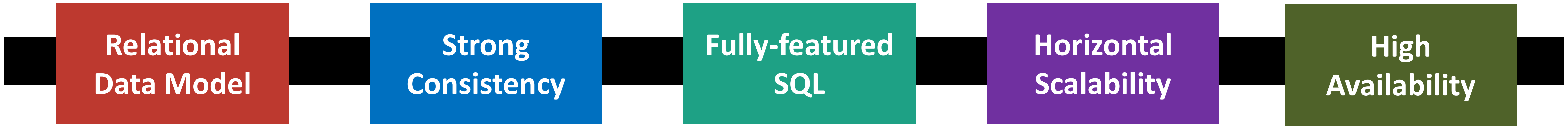
- Traditional SQL **Database**
  - **Limited scalability** and a **low resilience** to failures
  - Can handle **transactions!**
- NoSQL **Databases**
  - Designed to **scale out** using a distributed architecture
  - **Availability** is more important than **consistency**
  - Limited support for handling transactions
  - Not optimized for SQL queries
- **You win something, you lose something…**

We believe it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise, rather than always coding around the lack of transactions. (Google)

Distributed SQL Database (NewSQL)

- NewSQL databases emerged to **make SQL scalable**

__NewSQL__ is a class of __relational__ database management systems that seek to provide the scalability of NoSQL systems for online transaction processing (OLTP) workloads while maintaining the ACID guarantees of a traditional database system. (Wikipedia)



Distributed SQL Database (NewSQL)

- A **single** logical relational database instance that is distributed on a global scale over **multiple geographic regions**
- Data will be **replicated** into multiple nodes inside the cluster
- **Scale out** the cluster and add more nodes

Amazon Aurora

Cloud Spanner

SingleStore

VOLTDB

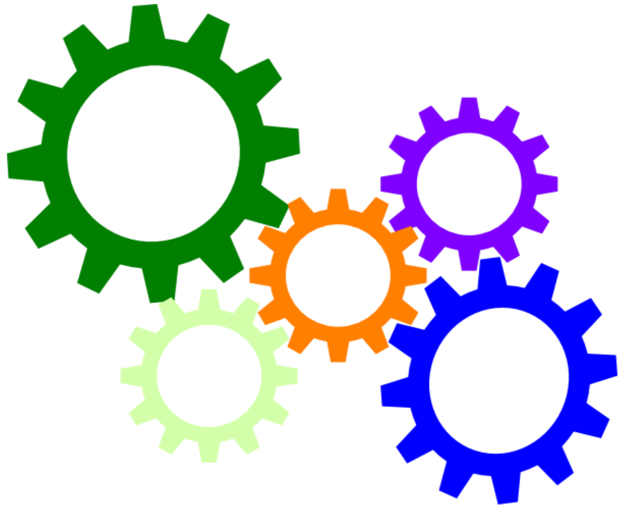
NUODB

Cockroach DB

Apache Trafodion

Distributed SQL Database (NewSQL)

- **NewSQL Core Functionalities**
  - Supporting SQL (relational model)
  - Geographically distributed
  - Scale on demand (up and out!)
  - ACID Compliant (transactions….)
  - Deployment options
    - On premises
    - Self-hosted cloud
    - Managed service (DBaaS)



In-memory Database (IMDB)

How the database store the data?

- **Disk-based Database**
  - The most popular and traditional option
  - SSD/HDD technologies
  - Data is **persistence** (will not be erased after a server reboot)
  - The **disk access** time is the main **bottleneck** point for databases
- **In-Memory Database (IMDB)**
  - Store the data completely **in-memory**
  - Can be SQL/NoSQL
  - Eliminating the time needed to query data from a disk
  - A memory in a computer is much more expensive than disk storage

In-memory Database (IMDB)

How the database store the data?

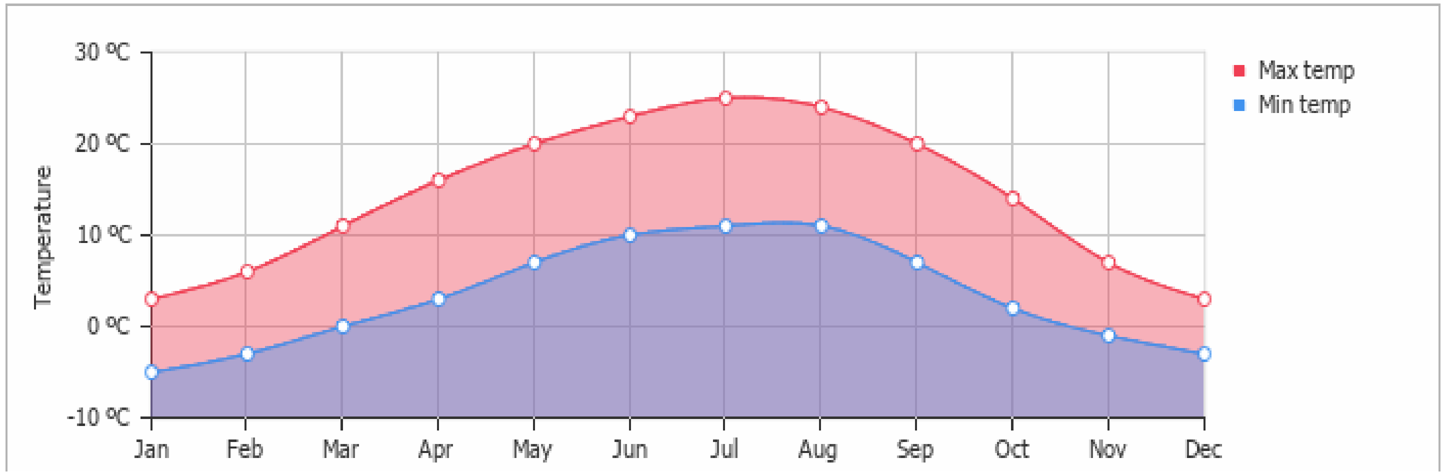
- In-Memory **Database (IMDB)**
  - A great solution for applications that require **microsecond response times**
  - One of the top use cases of an in-memory database is for **data caching**
  - Popular in-memory solutions:

redis MEMCACHED apache Ignite® VOLTDB

Time-Series Database (TSDB)

It's about TIME!

- One of the **fastest-growing** database categories
  - Connected objects → data, data and data
  - E.g. IoT, Monitoring IT…
- **The IoT Wave**
  - Using sensors to measure something…
  - Monitor, Analyze, and Explore
  - A typical dimension that is very important → **TIME**
    - Timestamp



Time-Series Database (TSDB)

It’s about TIME!

- **Time-Series Data**
  - A series of values where the **X-axis is time**
  - Modeling Components
    - **Timestamp**
    - **Subject**
    - **Measurement/s**
  - Regular intervals, and also fluctuating or random events
  - Typically high data volumes
- **Required Database Technology**
    - High-frequency data write
    - Time functions for developers
    - Using traditional relational database???

Time-Series Database (TSDB)

- **Time Series Database (TSDB)**
  - **Specialized** database technology for time series data
  - More **optimized** from a performance perspective
  - Automatically store recent data in-memory for fast access and move historical data to disk storage
  - Handle **high-volume data ingestion** (writing to the database)
  - Built-in specialized **time-related functions** while using SQL

influxdb

Prometheus

kx

Amazon Timestream

graphite

OPENTSDB