alterra academy

System Design

OUTLINE

- Diagram Design
- Characteristics of Distributed Systems
- Horizontal Scaling vs Vertical Scaling
- Job/Work Queue
- Load Balancing
- Monolithic vs Microservices
- SQL vs NoSQL
- Caching
- Database Indexing
- Database Replication

Diagram

A diagram is a symbolic representation of information using visualization techniques.

Diagram - Wikipedia

alterra academy

Diagram Design Tools

smartdraw

Lucidchart

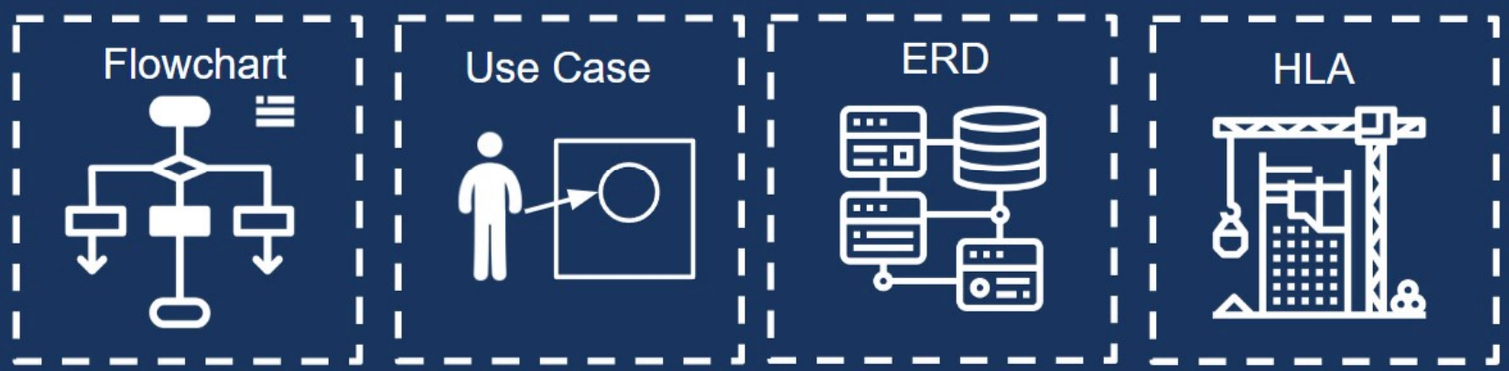
Whimsical

draw.io

Visio

alterra academy

A common used software design



Flowchart

- The theory - represent 1 process.
- Process, Decision, Terminator
- Example 1: Lucid's Flowchart Template
- Example 2: Final Project Example

alterra academy

Use Case Diagram

A use case diagram summarize the details of your **system's users** (also known as actors) and their **interactions with the system**.

- Example: Lucid's "Banking System Use Case Diagram"
- Example: Final Project Use Case Diagram

Entity Relationship Diagram

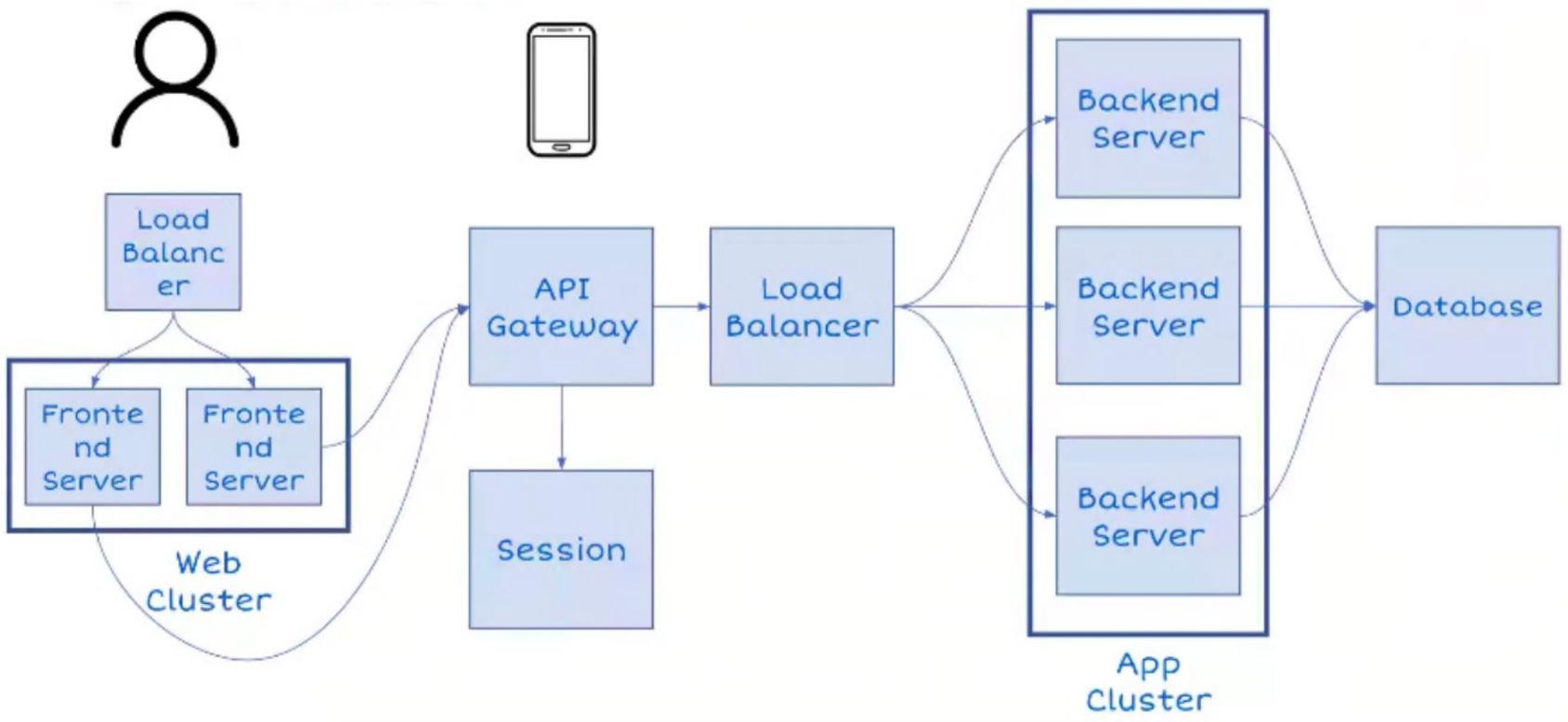
The theory: An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

✔ Example: Lucid’s “Entity Relationship Diagram”

alterra academy

High Level Architecture

- **Overall system design.**
- **Check your target audience.**
- Example: **“Final Project HLA”**
- Example: **“Final Project HLA”**


Load Balanc er
Fronte nd Server
Fronte nd Server
Web Cluster
API Gateway
Session
Load Balancer
Backend Server
Backend Server
Backend Server
App Cluster
Database

alterra
academy

Horizontal Scaling
And
Vertical Scaling

alterra
academy

System Design Basics

Whenever we are designing a large system, we need to consider a few things:

1. What are the different architectural pieces that can be used?
2. How do these pieces work with each other?
3. How can we best utilize these pieces: what are the right tradeoffs?

Familiarizing these concepts would greatly benefit in understanding distributed system concepts.

alterra academy

Key Characteristics of Distributed Systems

Scalability

Reliability

Availability

Efficiency

Serviceability
or Manageability

alterra
academy

Scalability

Scalability is the capability of a system, process, or a network to grow and manage increased demand. Any distributed system that can continuously evolve in order to support the growing amount of work is considered to be scalable.

A system may have to scale because of many reasons like increased data volume or increased amount of work, e.g., number of transactions. A scalable system would like to achieve this scaling without performance loss.

Scaling

Study Case

Our core business is selling goods online. Therefore we need hire an employee that responsible to pack the items.

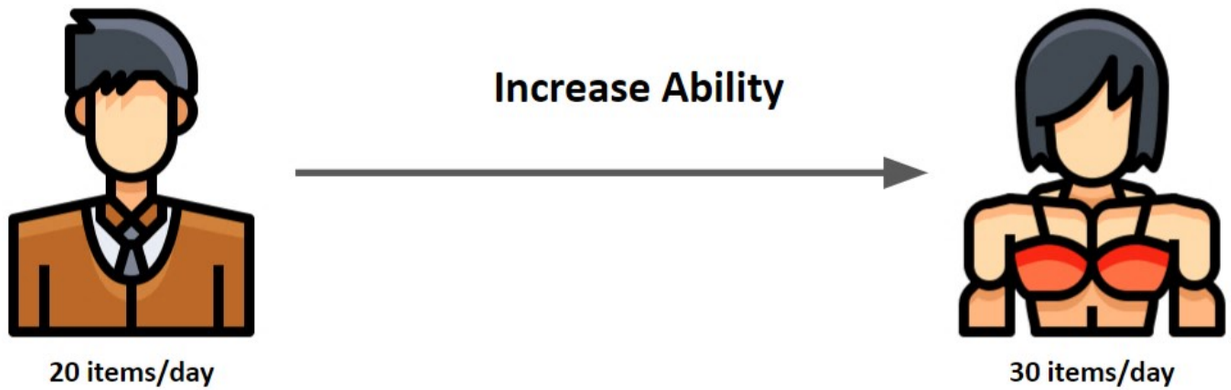
We have a candidate and based on interview he will able to pack maximum 20 items per day.

Our transaction have 10 items per day now, so he is able to solve our problem → HIRED him!

20 items/day

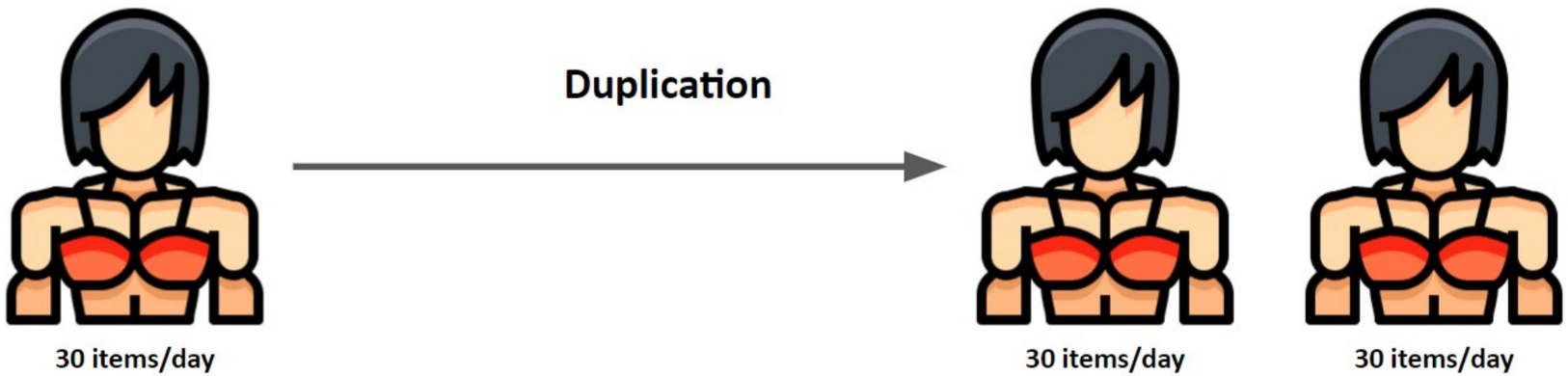
Scaling

Thanks to god, our transaction was increase at second month. We now have 25 items per day. But our employee only able to pack 20 items per day at max.

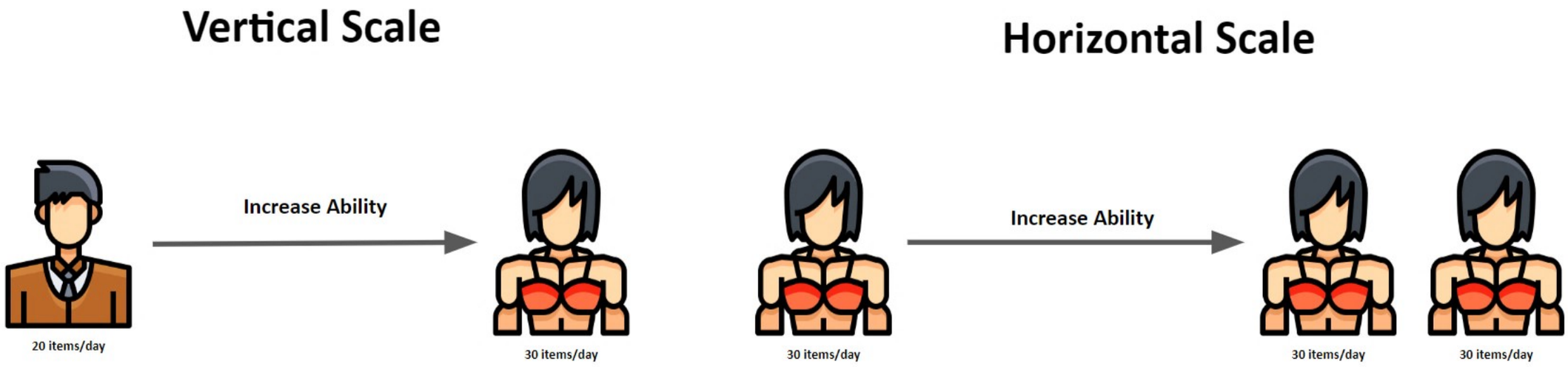


Scaling

At third month our transaction is increase become to 50 items per day. Currently we not increase the ability but we trying another approach, hiring another man to help us.



Scaling


Vertical Scale
Increase Ability
20 items/day
30 items/day
Horizontal Scale
Increase Ability
30 items/day
30 items/day
30 items/day

alterra academy

Which one you prefer?

Why?

Scaling

If we still have possibility to increase ability and can guarantee it will be stand for long term then **vertical scale** is a good candidate

→ no changes on our code.

If we are not able to predict and there are possibility that our increase will be happened fast, it will be save to follow **horizontal scale** approach

→ there are possibilities have changes on our code to adapt.

alterra
academy

Reliability

By definition, reliability is the probability a system will fail in a given period. In simple terms, a distributed system is considered reliable if it keeps delivering its services even when one or several of its software or hardware components fail. Reliability represents one of the main characteristics of any distributed system, since in such systems any failing machine can always be replaced by another healthy one, ensuring the completion of the requested task.

Availability

By definition, availability is the time a system remains operational to perform its required function in a specific period. It is a simple measure of the percentage of time that a system, service, or a machine remains operational under normal conditions.

Example: an aircraft that can be flown for many hours a month without much downtime can be said to have a high availability.

alterra academy

Efficiency

To understand how to measure the efficiency of a distributed system, let's assume we have an operation that runs in a distributed manner and delivers a set of items as result. Two standard measures of its efficiency are the response time (or latency) that denotes the delay to obtain the first item and the throughput (or bandwidth) which denotes the number of items delivered in a given time unit (e.g., a second).

alterra
academy

Serviceability or Manageability

Serviceability or manageability is the simplicity and speed with which a system can be repaired or maintained; if the time to fix a failed system increases, then availability will decrease. Things to consider for manageability are the ease of diagnosing and understanding problems when they occur, ease of making updates or modifications, and how simple the system is to operate (i.e., does it routinely operate without failure or exceptions?).

alterra
academy

Job/Work Queue

Job/Work Queue

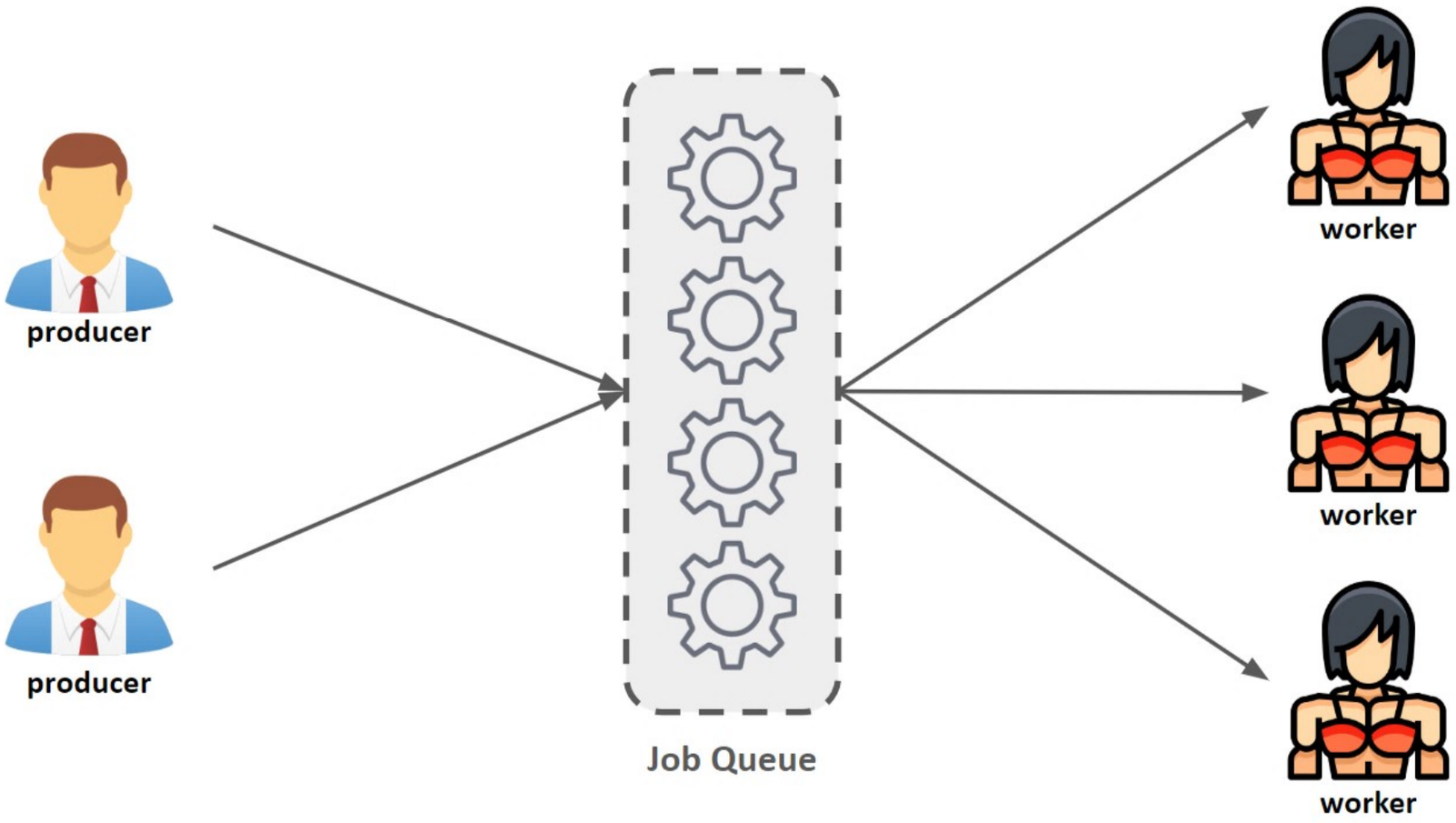
“In system software, a job queue (sometimes batch queue), is a data structure maintained by job scheduler software containing jobs to run.”

Wikipedia - Job Queue

“Work Queue is a framework for building large master-worker applications that span thousands of machines drawn from clusters, clouds, and grids.”

CCL University of Notre Dame - Work Queue: A Scalable Master/Worker Framework

Job/Work Queue



alterra
academy

Load Balancing

alterra
academy

Load Balancing

Load Balancer (LB) is another critical component of any distributed system. It helps to spread the traffic across a cluster of servers to improve responsiveness and availability of applications, websites or databases. LB also keeps track of the status of all the resources while distributing requests. If a server is not available to take new requests or is not responding or has elevated error rate, LB will stop sending traffic to such a server.

alterra academy

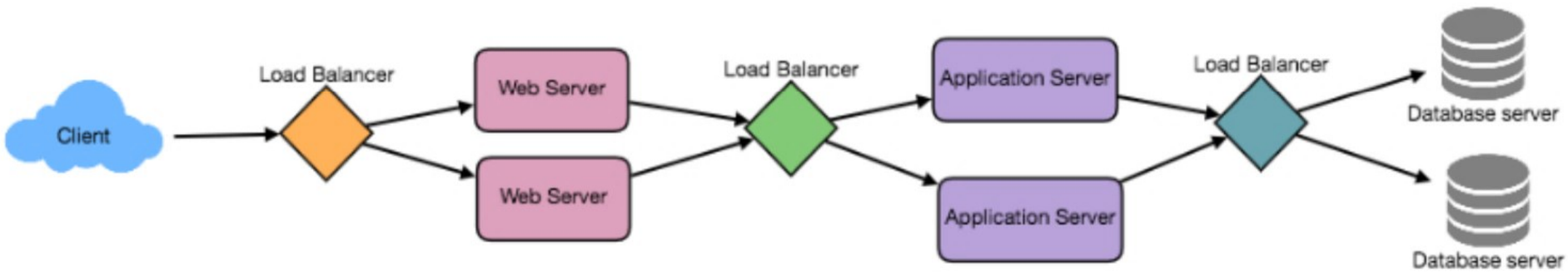
Load Balancing

To utilize full scalability and redundancy, we can try to balance the load at each layer of the system. We can add LBs at three places:

- Between the user and the web server
- Between web servers and an internal platform layer, like application servers or cache servers
- Between internal platform layer and database.

alterra academy

Load Balancing



alterra
academy

Monolithic and Microservices

alterra academy

Monolithic

Monolithic application has single code base with multiple modules. Modules are divided as either for business features or technical features. It has single build system which build entire application and/or dependency. It also has single executable or deployable binary.

alterra
academy

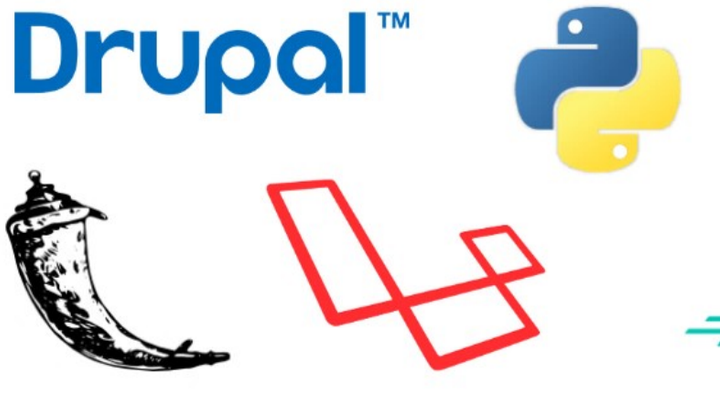
Microservices

Microservices are independently deployable services modeled around a business domain. They communicate with each other via networks, and as an architecture choice offer many options for solving the problems you may face. It follows that a microservice architecture is based on multiple collaborating microservices.

Simple Tech Stack

2018 - Tech Stack: monolith -> Microservices
















alterra
academy

SQL and NoSQL

alterra
academy

SQL vs. NoSQL

SQL and NoSQL (or relational databases and non-relational databases).

- Relational databases are structured and have predefined schemas like phone books that store phone numbers and addresses.
- Non-relational databases are unstructured, and have a dynamic schema like file folders that hold everything from a person's address and phone number to their Facebook 'likes' and online shopping preferences.

alterra academy

BENEFIT RELATIONAL DB

Dirancang untuk segala keperluan

SQL - Memiliki standar yang jelas

Memiliki banyak tool (administrasi, reporting, framework)

alterra academy

A **Atomicity**; transaksi terjadi semua atau tidak sama sekali

C **Consistency**; data tertulis merupakan data valid yang ditentukan berdasarkan aturan tertentu

I **Isolation**; pada saat terjadi request yang bersamaan (concurrent), memastikan bahwa transaksi dieksekusi seperti dijalankan secara sekuensial

D **Durability**; jaminan bahwa transaksi yang telah tersimpan, tetap tersimpan.

alterra
academy

Not only SQL

is a whole ***new way of thinking*** *about a database.*
NoSQL is **not** *a relational database.*

Not only SQL

DBMS yang menyediakan mekanisme yang lebih fleksibel dibandingkan dengan model RDBMS (Sifat ACID).

Menghindari:

- Effort pada sifat transaksi ACID
- Kompleksitas SQL
- Design schema di depan
- Transactions (ditangani oleh aplikasi)

alterra academy

Kelebihan: schema less, fast development, etc.

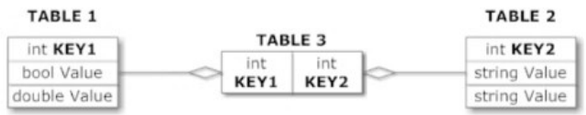
Kapan digunakan: membutuhkan skema fleksibel, ACID tidak diperlukan, data logging (json), data sementara (chace), etc

Kapan tidak direkomendasikan : data finansial, data transaksi, business critical, ACID - compliant.

Schema-less

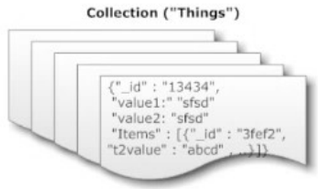
- Traditional RDBMS
  - Tidak bisa menambah data yang tidak sesuai skema
  - Perlu menambahkan data NULL pada item yang tidak memiliki data
  - Memiliki tipe data yang strict
  - Tidak dapat menambah beberapa item data pada sebuah field

Relational Model



- NoSQL DBMS
  - Tidak memiliki skema ketika menambahkan data
  - Aplikasi menangani proses validasi tipe data
  - Mendukung proses aggregasi dokumen pada item.

Document Model



alterra

TIPE / KATEGORI

Key / Value

Column - family

Graph

Document - based

Others
Geospatial
File-system
Object

alterra
academy

SQL vs. NoSQL

When it comes to database technology, there's no one-size-fits-all solution. That's why many businesses rely on both relational and non-relational databases for different needs. Even as NoSQL databases are gaining popularity for their speed and scalability, there are still situations where a highly structured SQL database may perform better; choosing the right technology hinges on the use case.

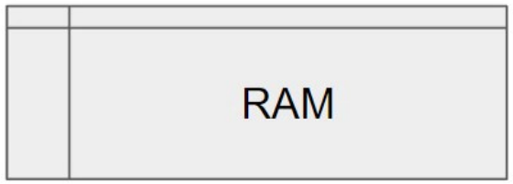
alterra
academy

Caching

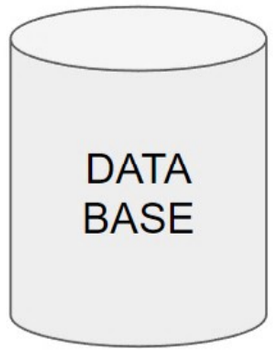
alterra
academy

Caching

Caches used in recently requested **data is likely to be requested again**. They are used in almost every layer of computing: hardware, operating systems, web browsers, web applications, and more.
A cache is like short-term memory: it has a limited amount of space, but is typically faster than the original data source and contains the most recently accessed items. Caches can exist at all levels in architecture, but are often found at the level nearest to the front end where they are implemented to return data quickly without taxing downstream levels.



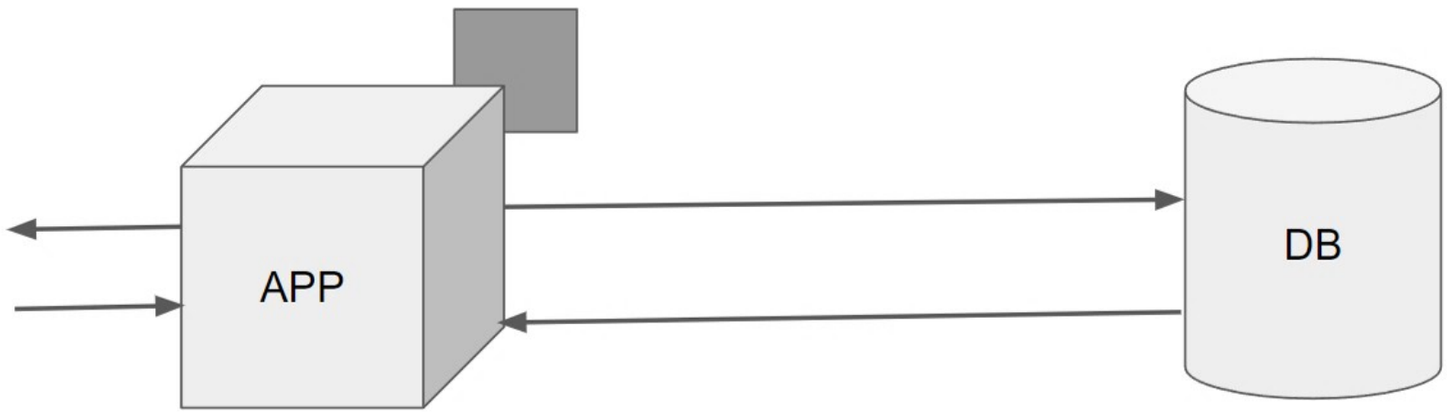
Penyimpanan Data Sementara

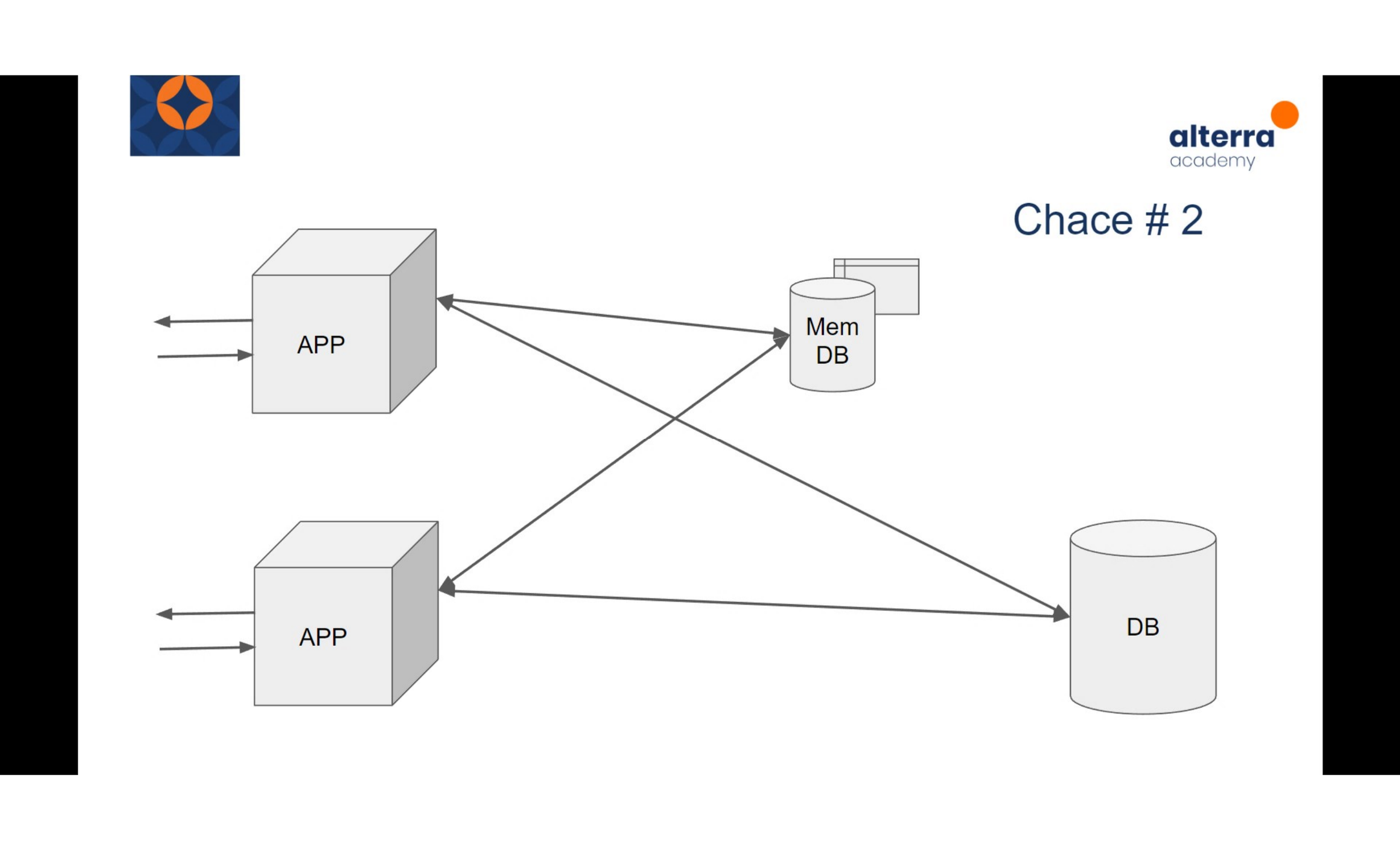


Penyimpanan
Data Permanen

alterra academy

Chace # 1




alterra
academy
Chace # 2
APP
APP
Mem
DB
DB

alterra
academy

Database Replication

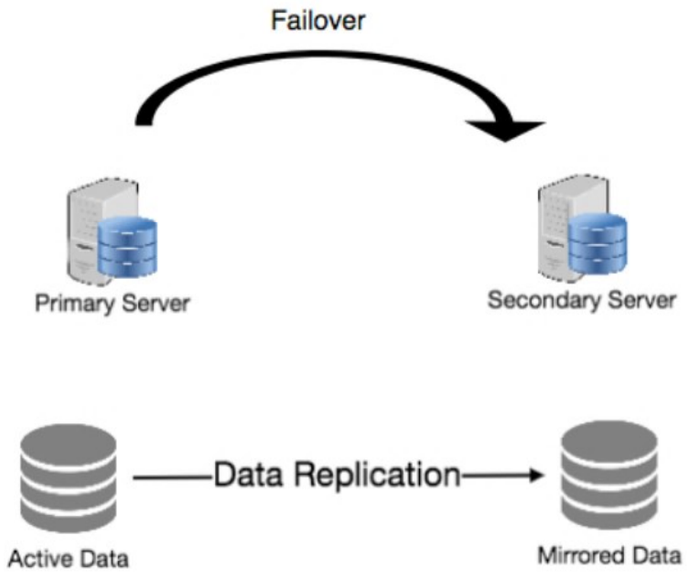
alterra
academy

Redundancy and Replication

Redundancy is the duplication of critical components or functions of a system with the intention of increasing the reliability of the system, usually in the form of a backup or fail-safe, or to improve actual system performance. For example, if there is only one copy of a file stored on a single server, then losing that server means losing the file. Since losing data is seldom a good thing, we can create duplicate or redundant copies of the file to solve this problem.

alterra academy

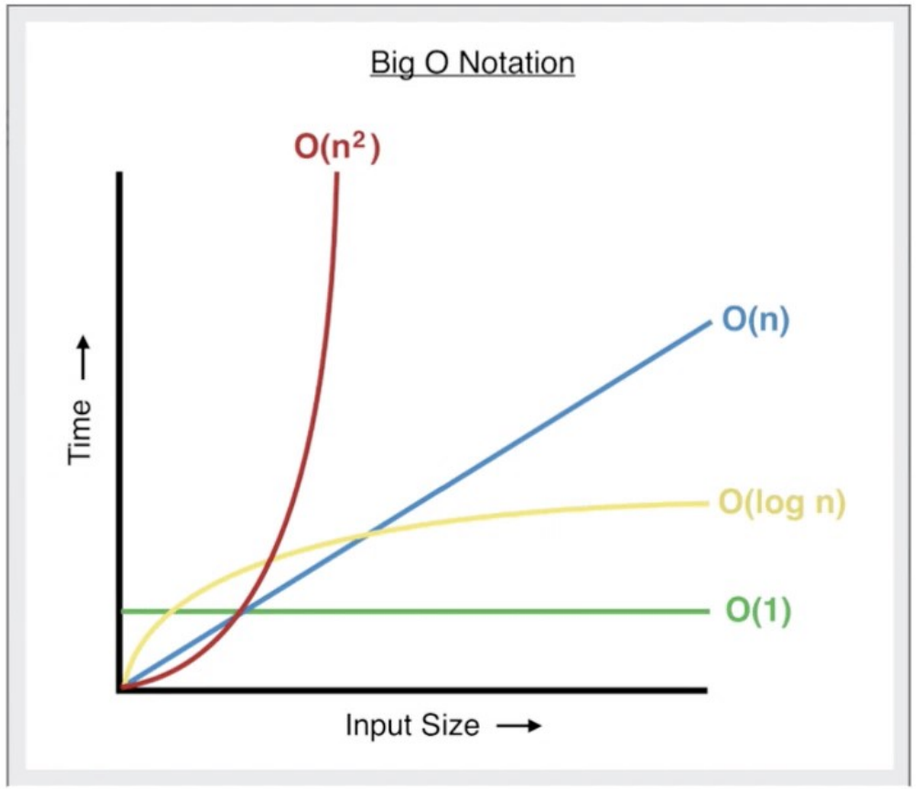
Redundancy and Replication



alterra
academy

Database Indexing

Complexity



Indexing

“Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.”

Geeksforgeeks - Indexing Database

Most of the database using B-Tree as their data structure for indexing. And B-Tree have a complexity of **O(log n)** for **search, delete** and **insert** operation.

Database Indexing

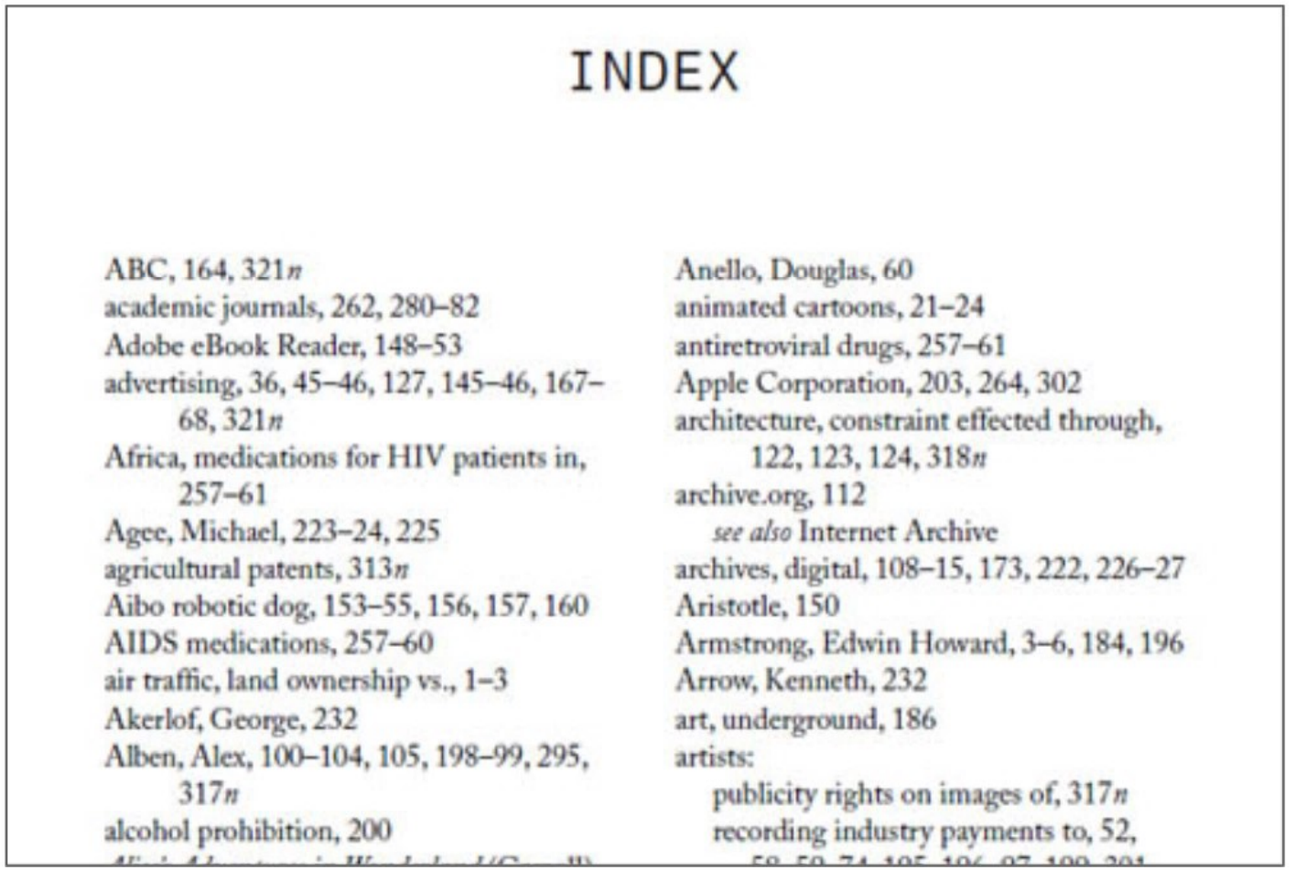
Consider a "Book" of 1000 pages, divided by 10 Chapters, each section with 100 pages.

Simple, huh?

Now, imagine you want to find a particular Chapter that contains a word "Alchemist". Without an index page, you have no other option than scanning through the entire book/Chapters. i.e: 1000 pages.

This analogy is known as "Full Table Scan" in database world.

Database Indexing



INDEX

Database Indexing

With an index page, you know where to go! And more, to lookup any particular Chapter that matters, you just need to look over the index page, again and again, every time. After finding the matching index you can efficiently jump to that chapter by skipping the rest.

But then, in addition to actual 1000 pages, you will need another ~10 pages to show the indices, so totally 1010 pages.

Database Indexing

The index is a separate section that stores values of indexed column + pointer to the indexed row in a sorted order for efficient lookups.

W3schools - Indexing

- https://www.w3schools.com/sql/sql_create_index.asp