BENR2423

Database and Cloud System

Chapter 1:
Introduction

Contact

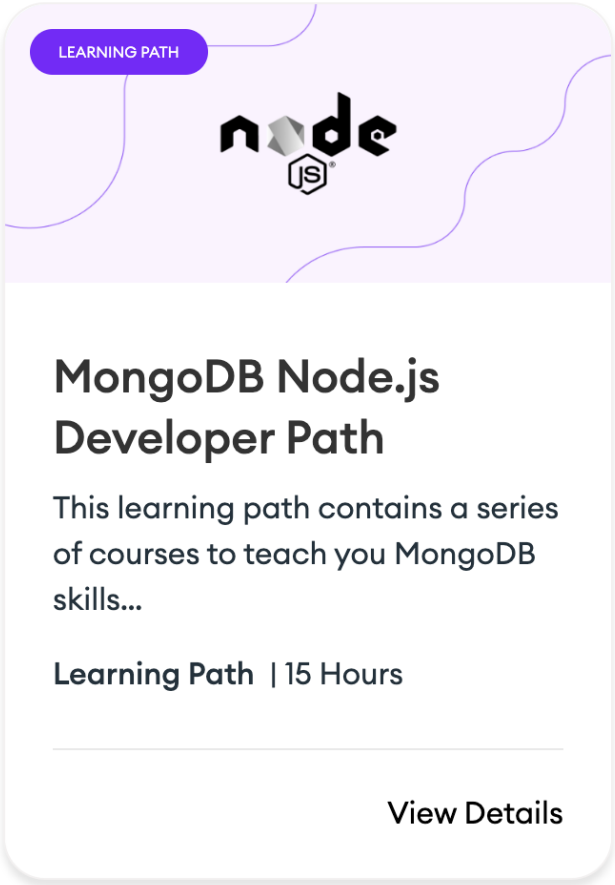
Soo Yew Guan

soo@utem.edu.my

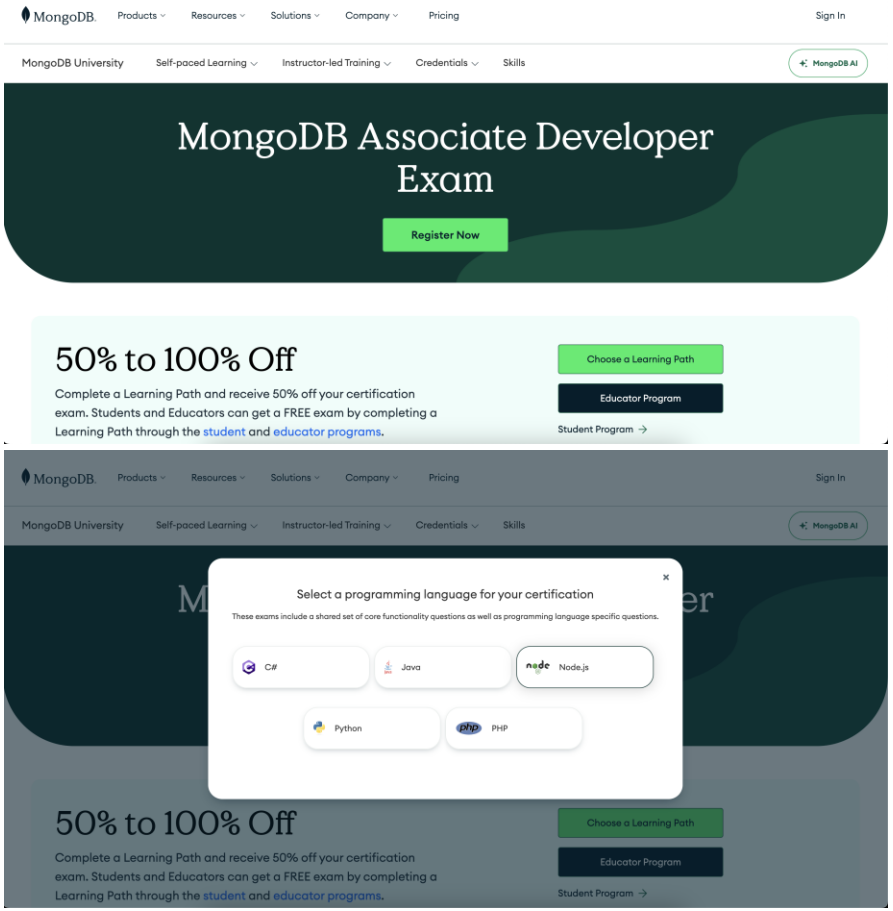
Research Lab III

- Course Work: 40%
  - Quiz: 5%
  - Test: 15%
  - Assignment: 20%
- Final Exam: 60%
- MongoDB Associate Developer Exam: 5%

Grading



https://learn.mongodb.com/learning-paths/mongodb-nodejs-developer-path



https://learn.mongodb.com/pages/mongodb-associate-developer-exam

MongoDB Associate Developer Exam

- Database Design and Queries
- Fundamental Backend Development
- Edge to Cloud Communication Protocol

Course Topics

Platform

- Mongo Atlas
  - https://account.mongodb.com/account/login
- Microsoft Azure
  - https://azure.microsoft.com/en-us
- Github
  - https://github.com/

Framework

- MongoDB
  - https://www.mongodb.com/
- NodeJs
  - https://nodejs.org/en/

Programming Language

- JavaScript

Programming and Tools

Tools

- MongoDB Compass
  - https://www.mongodb.com/try/download/compass
- Microsoft Visual Studio Code
  - https://code.visualstudio.com/
- Postman
  - https://www.postman.com/
- Git
  - https://git-scm.com/downloads/win
- Wireshark
  - https://www.wireshark.org/

Week 1

- **Lab 1 – Environment Setup & Introduction to MongoDB**
  - **Objectives:**
    - Install and configure essential development tools (VSCode, NodeJS, Git, and MongoDB)
    - Familiarize with the MongoDB shell and Compass for basic database interactions
  - **Activities:**
    - Set up the development environment and install required software
    - Create a simple “Hello World” NodeJS project that connects to MongoDB
    - Explore sample databases using both the command line and MongoDB Compass

Week 2

- **Lab 2 – Basic MongoDB CRUD Operations & Query Language**
  - **Objectives:**
    - Master MongoDB's CRUD operations using its native query language
    - Utilize MongoDB Compass and CLI to manipulate data
  - **Activities:**
    - Insert, query, update, and delete sample documents in a collection
    - Develop basic queries and filters using the MongoDB query language

Week 3

- **Lab 3 – Building a Basic REST API with NodeJS and MongoDB**
  - **Objectives:**
    - Introduce Express.js for building RESTful services
    - Integrate MongoDB operations within a NodeJS server environment
    - Validate endpoints using Postman
  - **Activities:**
    - Scaffold a NodeJS/Express project and establish API routes for a “users” collection
    - Implement endpoints that perform CRUD operations integrated with MongoDB
    - Test endpoints using Postman and capture the results

Week 4

- **Lab 4 – HTTP Protocols, API Design Fundamentals & Wireshark TCP Stream Analysis**
  - **Objectives:**
    - Deepen understanding of HTTP methods (GET, POST, PUT, DELETE) and status codes
    - Introduce API design best practices, including the creation of API flow diagrams
    - Learn to use Wireshark to capture and analyze TCP streams corresponding to HTTP transactions
  - **Activities:**
    - Capture HTTP request/response sequences using Wireshark with a focus on the TCP stream
    - Generate a Wireshark sequence diagram to visualize the TCP handshake, HTTP request/response flow, and correlate these activities with the designed API flow
    - Refactor existing endpoints to align with best practices based on the analysis

Week 5

- **Lab 5 – Implementing JWT Authentication**
  - **Objectives:**
    - Secure the REST API with JSON Web Tokens (JWT)
    - Implement role-based access control (RBAC) in a multi-role system
  - **Activities:**
    - Integrate JWT into the NodeJS application for user authentication
    - Develop middleware to protect routes and manage roles (e.g., customer, driver, admin)
    - Test secure endpoints using Postman with both valid and invalid tokens

Week 6:

- **Lab 6 – Deep Dive into the MongoDB Aggregation Framework**
  - **Objectives:**
    - Master the aggregation pipeline for complex data queries in MongoDB
    - Understand key stages such as $match, $group, $sort, and $project
  - **Activities:**
    - Create aggregation queries to produce summary reports (e.g., user statistics)
    - Compare aggregation query results with standard query methods
    - Visualize the aggregation pipeline using MongoDB Compass

Week 7

- **Lab 7 – Database Indexing and Performance Optimization**
  - **Objectives:**
    - Learn to implement indexing in MongoDB to enhance query performance
    - Utilize performance analysis tools and explain plans to refine queries
  - **Activities:**
    - Create indexes on collections and evaluate their impact using MongoDB’s explain plan
    - Conduct performance tests and adjust queries/indexes accordingly

Week 8

- **Lab 8 – Database Design: ERD & Crow's Foot Notation**
- **Objectives:**
  - Understand fundamental database design principles using ER diagrams and crow's foot notation
  - Develop a robust schema for a ride-sharing system scenario
- **Activities:**
  - Design an ERD that models multi-role interactions in a ride-sharing system
  - Discuss normalization and relationships between entities
  - Translate the ERD into a MongoDB schema design, identifying collections and document relationships

Week 9

- **Lab 9 – Advanced NodeJS: Multi-role Architecture and API Enhancements**
- **Objectives:**
  - Expand the API to support a multi-role system (customer, driver, admin)
  - Refine API design with enhanced route structuring and detailed flow diagrams
- **Activities:**
  - Implement additional role-specific endpoints and middleware for role validation
  - Update API flow diagrams to clearly outline the request handling process for each role
  - Refactor the NodeJS codebase for improved scalability and maintainability

Week 10

- **Lab 10 – Introduction to CI/CD with GitHub Actions**
  - **Objectives:**
    - Grasp continuous integration/continuous deployment (CI/CD) principles
    - Configure GitHub Actions to automate testing, linting, and builds
  - **Activities:**
    - Set up a GitHub repository for the NodeJS project
    - Create GitHub Actions workflows that trigger on pushes and pull requests
    - Analyze build logs and refine the CI pipeline configuration

Week 11

- **Lab 11 – Cloud Deployment on Microsoft Azure**
  - **Objectives:**
    - Introduce Microsoft Azure services for web app hosting
    - Deploy the NodeJS application to Azure using automated pipelines
  - **Activities:**
    - Create an Azure Web App and configure the necessary deployment settings
    - Integrate GitHub Actions with Azure deployment pipelines
    - Monitor the deployed application and resolve common deployment issues

Week 12

- **Lab 12 – Debugging, Logging, and Enhanced Wireshark TCP Stream Analysis**
- **Objectives:**
  - Improve debugging and logging skills using VSCode's debugging tools
  - Use Wireshark to capture and analyze the TCP stream in detail, focusing on debugging and network issues
  - Correlate the captured TCP stream with activities shown in a Wireshark sequence diagram
- **Activities:**
  - Set breakpoints and step through the NodeJS application using VSCode
  - Capture live HTTP/TCP traffic using Wireshark, applying TCP stream filtering
  - Generate a detailed Wireshark sequence diagram and analyze it based on the specific activities (e.g., TCP handshake, data exchange, error handling) observed during the lab

Week 13

- **Lab 13 – Mini Project: Building a Ride-Hailing API Subsystem**
- **Objectives:**
  - Integrate the skills acquired in previous labs to construct a core ride-hailing API subsystem
  - Emphasize API design, database integration, security, and performance optimization
- **Activities:**
  - Develop endpoints for booking rides, managing users, and assigning drivers
  - Implement JWT authentication and role validation within the subsystem
  - Produce supporting documentation including ERDs, API flow diagrams, and a Wireshark sequence diagram (if applicable) to validate API behavior

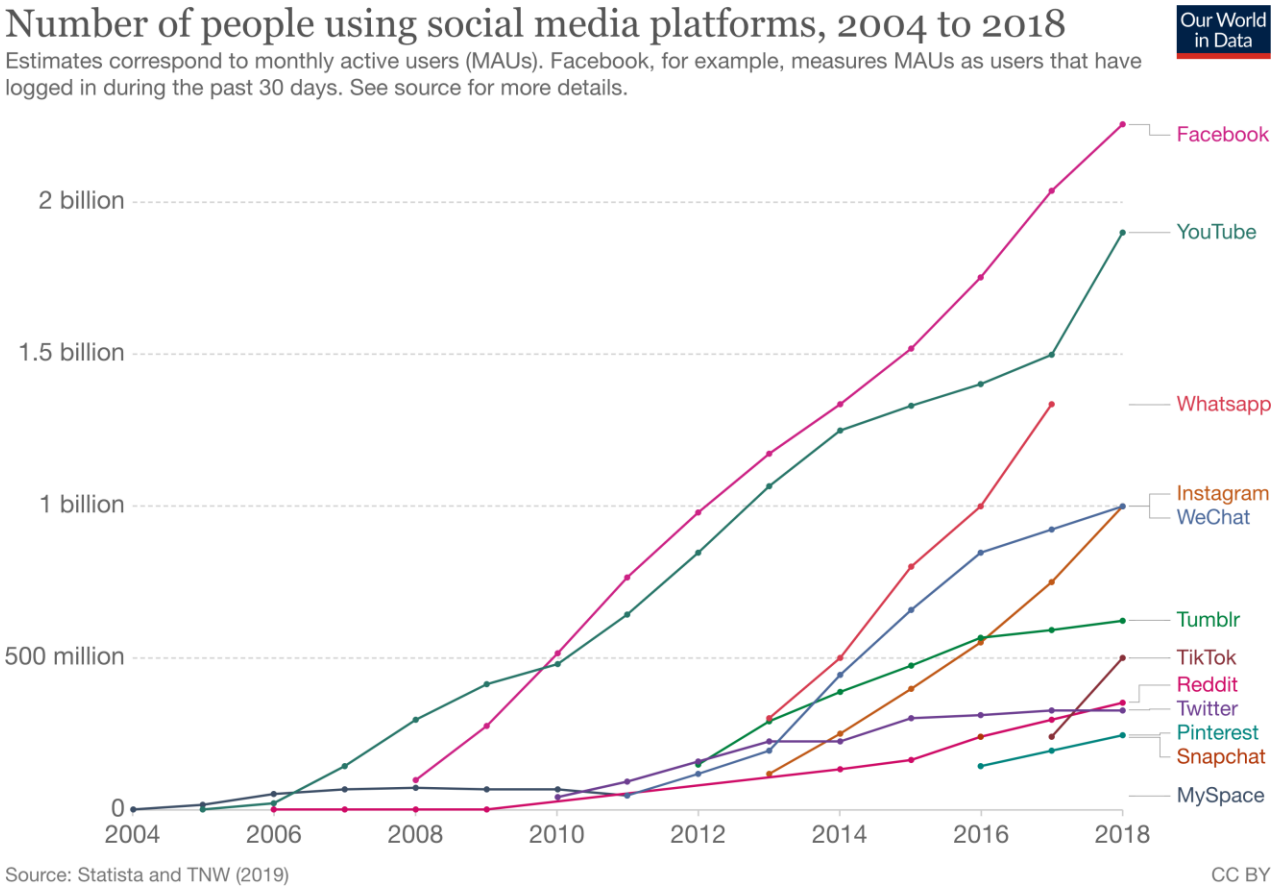
Week 14

- **Lab 14 – Capstone Project: Develop a Grab-like Solution**
  - **Objectives:**
    - Consolidate all learned concepts into a comprehensive, full-stack ride-hailing application
    - Demonstrate professional-level software development, including security, scalability, and continuous deployment practices
  - **Activities:**
    - Design and implement a full-fledged ride-hailing application with multi-role support (customers, drivers, admins)
    - Develop secure REST APIs using JWT, design robust MongoDB schemas with proper indexing and aggregation, and apply CI/CD pipelines using GitHub Actions for automated deployment to Azure
    - Prepare detailed design documents including ERDs, API flow diagrams, and Wireshark sequence diagrams to illustrate TCP stream analyses during critical operations
    - Present and demo the complete solution in a final presentation

Introduction to Database

Chapter 1

- The world is drowning in data

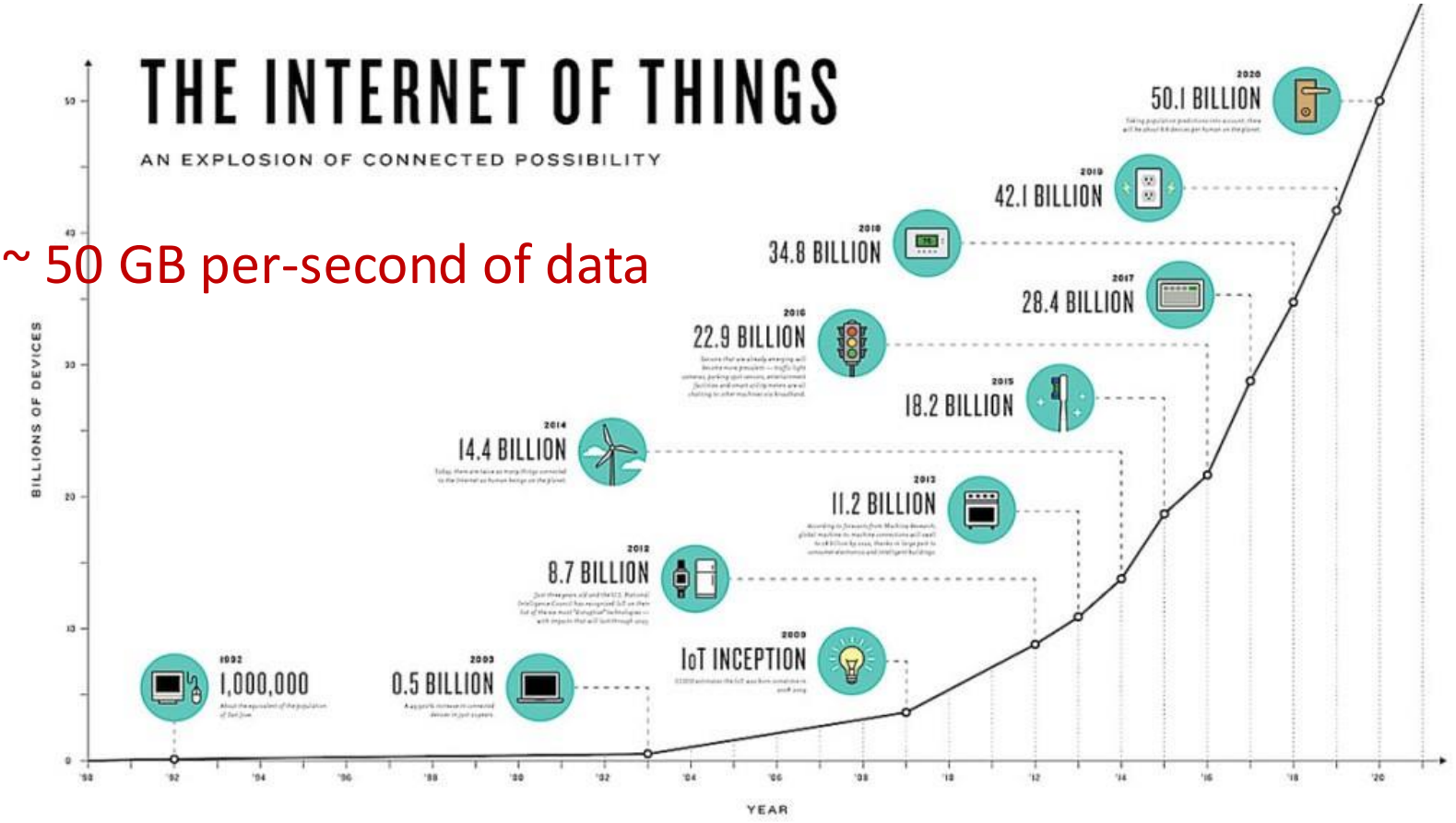


Motivation

Network of physical objects that are connected to the internet allowing them to send, receive and exchange data

Internet of Things

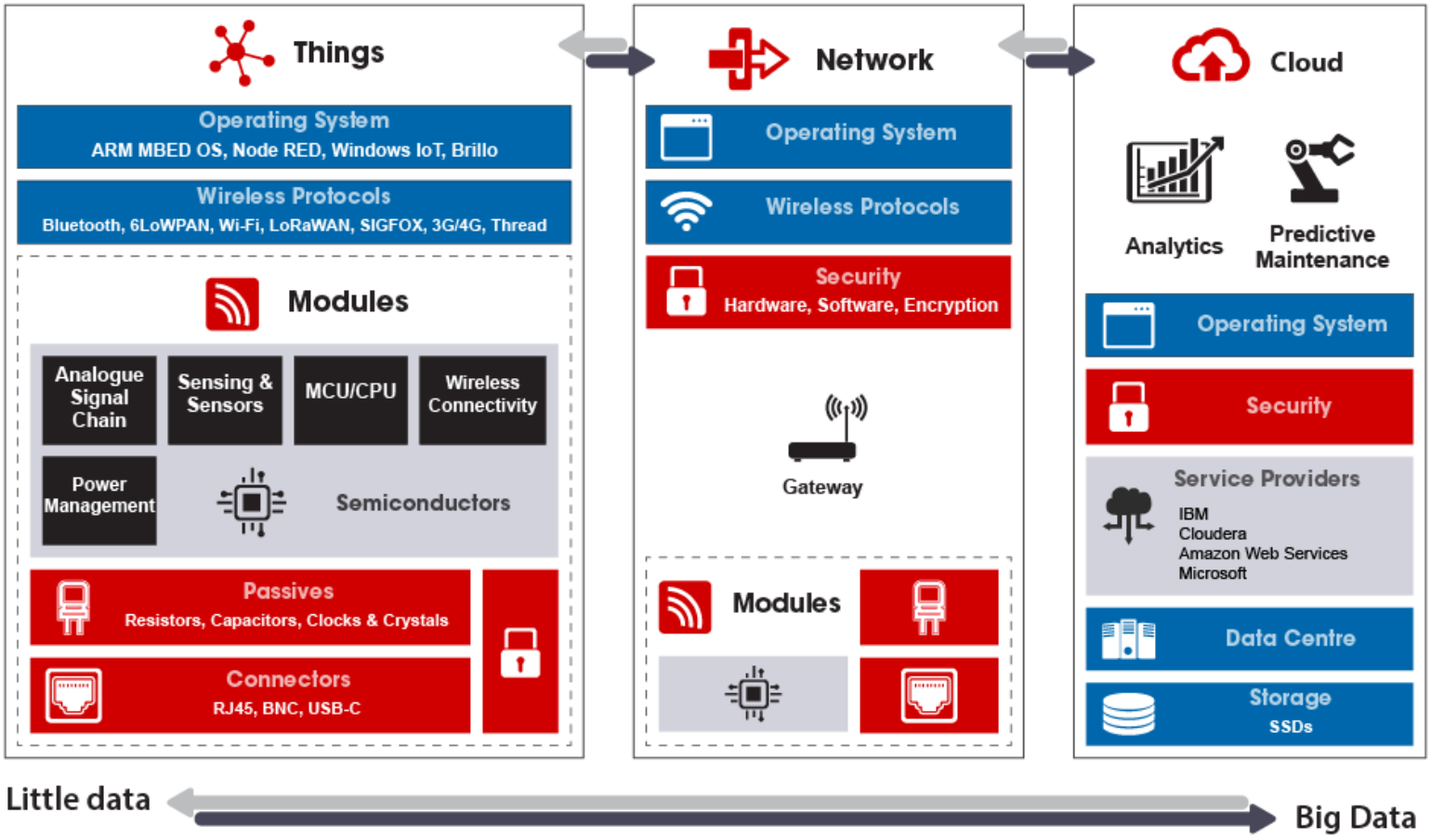
- Imagine a device push a byte per-second to the cloud



~ 50 GB per-second of data

Internet of Things

IoT Architecture



- A collection of files storing related data

- For example:
  - Facebook Post
  - Bank Account
  - UTeM Student's Records
  - Contact List on Cell Phone

Database

Database Management Systems (DBMS)

- A software that allows client to manage large database that stores, access, and manipulates data saved on disk, or even in RAM
- Organize data according to a specific pattern, called database model
- The DBMS has three core components:
  - Data storage engine
  - Query / Update engine
  - Schema management system

- Data storage can be persistence or ephemeral
- The format of that data can vary widely: small or large files, organized by row, by column, or by content.

A/c number	First name	Last name	A/c Type	Branch
12345756453	Michael	Calder	Saving	Manhattan
12345978675	Nick	Brown	Current	Brooklyn

```
{
  A/c number: 12345756453,
  First name: "Michael",
  Last name: "Calder",
  A/c Type: "Saving",
  Branch: "Manhattan"
}
```

Data Storage Engine

- Query Language are used for retrieving and storing data of different types.
  - SQL (Structured Query Language)
  - MQL (MongoDB Query Language)
- In order to expose those functions, the DBMS exposes APIs for client applications

Query / Update engine

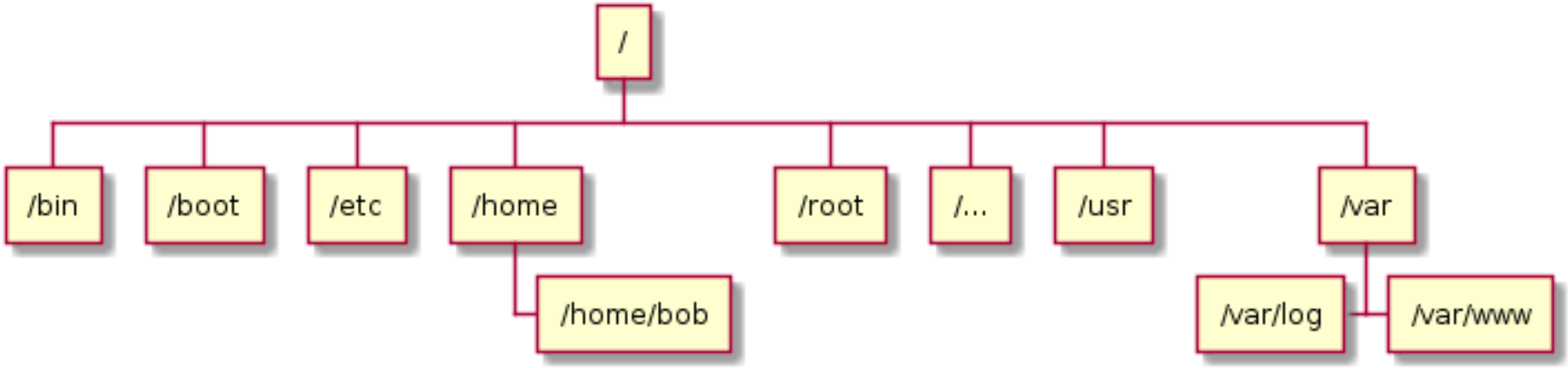
- Define how you want your data to be structured
- Rigid schemas
  - useful when you know the exact specifications of each record in advance, and don’t need to change often
- Flexible schemas
  - useful when you're working with unstructured data, or have constantly changing, variability in your data set, or need to respond to more rapidly changing requirements

Schema Management system

- Hierarchical databases
- Network databases
- Relational databases
- Non-relational databases
- Time series databases

Types of Databases

- Hierarchical databases
  - Encode a relationship between items where every record has a single parent.
  - Tree-like structure that can be used to categorize records according to their parent record.



Types of Databases

- Relational databases
  - Organize data using tables
  - Column has a name and a data type
  - Row represents a data item

A/c number	First name	Last name	A/c Type	Branch
12345756453	Michael	Calder	Saving	Manhattan
12345978675	Nick	Brown	Current	Brooklyn

Types of Databases

Types of Databases

- Non-relational databases
  - Also known as document-oriented databases
  - Use a key to uniquely identify data within the database
  - Do not prescribe any specific format or schema.

```
{
  A/c number: 12345756453,
  First name: "Michael",
  Last name: "Calder",
  A/c Type: "Saving",
  Branch: "Manhattan"
}
```

ID: breakfast

```
{
        "type": "toast",
        "bread": "whole wheat",
        "spread": [
                "butter",
                "jam"
        ]
}
```

ID: lunch

```
{
        "type": "salad",
        "vegetarian": false,
        "ingredients": [
                "spinach",
                "tomato",
                "cucumber",
                "carrot",
                "dressing": [
                        "olive oil",
                        "vinegar",
                        "honey",
                        "lemon",
                        "salt",
                        "pepper",
                ],
                "tuna",
                "walnuts"
        ],
        "rating": "5 stars",
        "restaurant": "Skylight Diner"
}
```

ID: dinner

```
{
        "type": "pizza",
        "size": "large",
        "toppings": [
                "pepperoni",
                "tomato",
                "sausage"
        ],
        "price": 9.00,
        "presliced": true
}
```

- Time series databases
  - Data stores that focus on collecting and managing values that change over time
  - Organized into structures that record the values for a single item over time or using timestamps as keys to store values for multiple metrics or columns at once.

Time	CPU Temp
2019-10-31T03:48:05+00:00	37
2019-10-31T03:48:10+00:00	42
2019-10-31T03:48:15+00:00	33
2019-10-31T03:48:20+00:00	34
2019-10-31T03:48:25+00:00	40
2019-10-31T03:48:30+00:00	42
2019-10-31T03:48:35+00:00	41

Time	CPU Temp	System Load	Memory Usage %
2019-10-31T03:48:05+00:00	37	0.85	92
2019-10-31T03:48:10+00:00	42	0.87	90
2019-10-31T03:48:15+00:00	33	0.74	87
2019-10-31T03:48:20+00:00	34	0.72	77
2019-10-31T03:48:25+00:00	40	0.88	81
2019-10-31T03:48:30+00:00	42	0.89	82
2019-10-31T03:48:35+00:00	41	0.88	82

Types of Databases

- Also known as NoSQL Database
- Schema-less

Non-Relational Database

- Also known as NoSQL Database
- Schema-less

string	number	string	string
Name	**Phone**	**Address**	**E-Mail**
Soo	1234567890	FKEKK, UTeM	soo@utem.edu.my

Non-Relational Database

- Also known as NoSQL Database
- Schema-less

string	number	string	string
Name	**Phone**	**Address**	**E-Mail**
Soo	1234567890	FKEKK, UTeM	soo@utem.edu.my
Ali	**Mobile: 2344935809**	FKEKK, UTeM	ali@utem.edu.my

Non-Relational Database

- Also known as NoSQL Database
- **Schema-less**

```
{
    name: Soo,
    phone: 1234567890,
    address: FKEKK, UTeM
    email: soo@utem.edu.my
}
```

```
{
    name: Ali,
    phone: {
        mobile: 2344935809
        home: 234439559
    },
    address: FKEKK, UTeM
    email: ali@utem.edu.my
}
```

Non-Relational Database

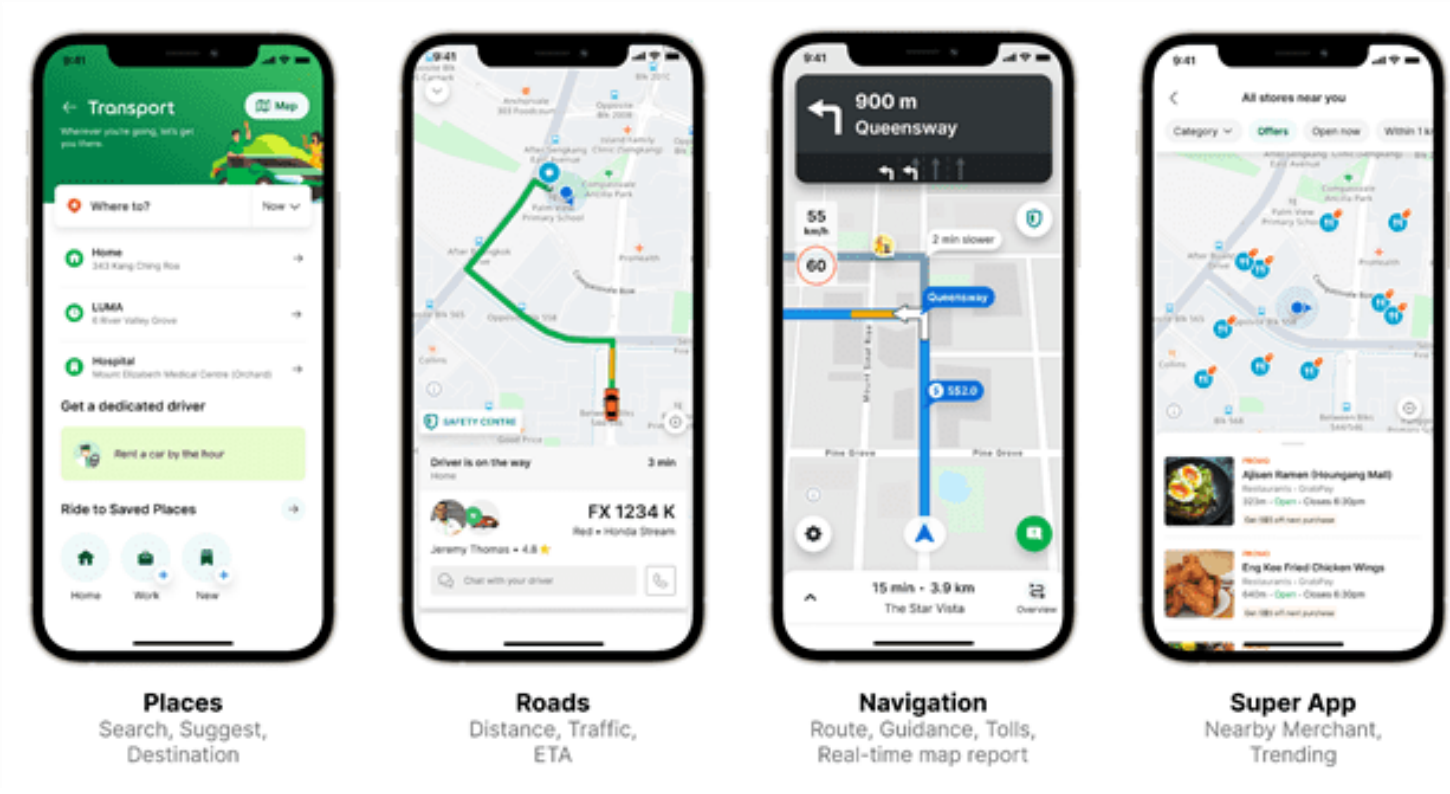
- Using key-value stores

Key	Value	
1234567890	KFC	**string**
3953833459	Pizzahut	**string**

2345938930	{
	name: McDonalds
	country: MY
	}
		JSON

Non-Relational Database

- Description of an information
- Example:



Non-Relational Database