BITS Pilani
Pilani | Dubai | Goa | Hyderabad

FrontEnds

Chandan Ravandur N

WebApps

Working

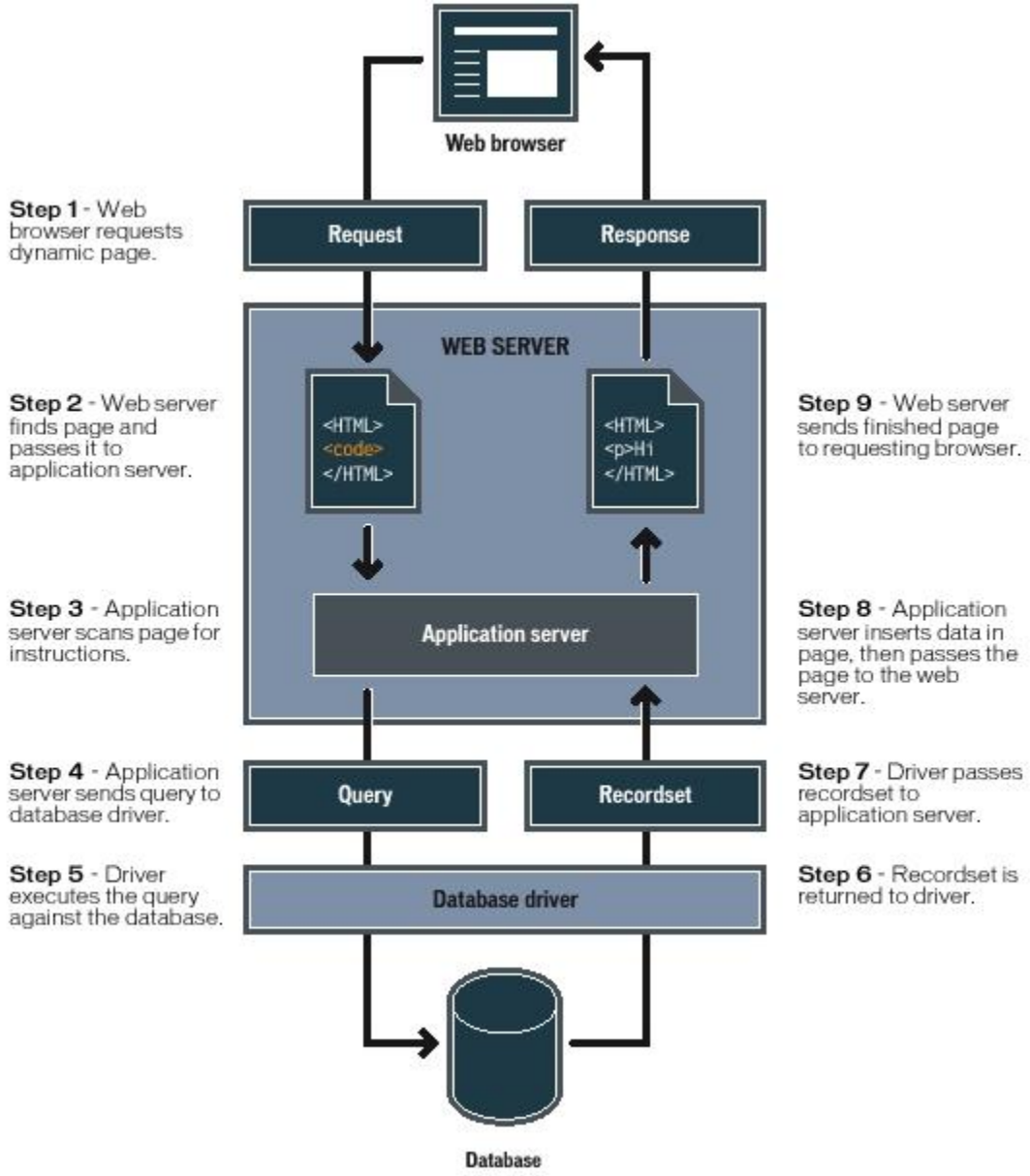


Image source

Front end and back end

the separation of concerns

- In software engineering,
  - Front end refer to the presentation layer (front end),
  - Backend refer to the data access layer (back end) of a piece of software, or the physical infrastructure or hardware

- In the client–server model,
  - the client is usually considered the front end
  - the server is usually considered the back end, even when some presentation work is actually done on the server itself.

- A rule of thumb is that
  - the client-side (or "front end") is any component manipulated by the user
  - the server-side (or "back end") code usually resides on the server, often far remote from user

Front-end

Focus is on

- User Interface elements
- Mark-up and web languages such as HTML, CSS, JavaScript and supporting libraries
- Asynchronous request handling and AJAX
- Single-page applications (with frameworks like React, AngularJS or Vue.js)
- Web performance
- Responsive web design
- Cross-browser compatibility issues and workarounds
- End-to-end testing with a headless browser
- Build automation to transform and bundle JavaScript files, reduce images size
- Search engine optimization
- Accessibility concerns

Back-end

Focus is on

- Software Architecture
- Application Business Logic
- Application Data Access
- Database management
- Scripting languages like JavaScript, Node.js, PHP, Python, Ruby, Java etc.
- Automated testing frameworks for the language being used
- Scalability
- High availability
- Security concerns, authentication and authorization

Front end – Back end

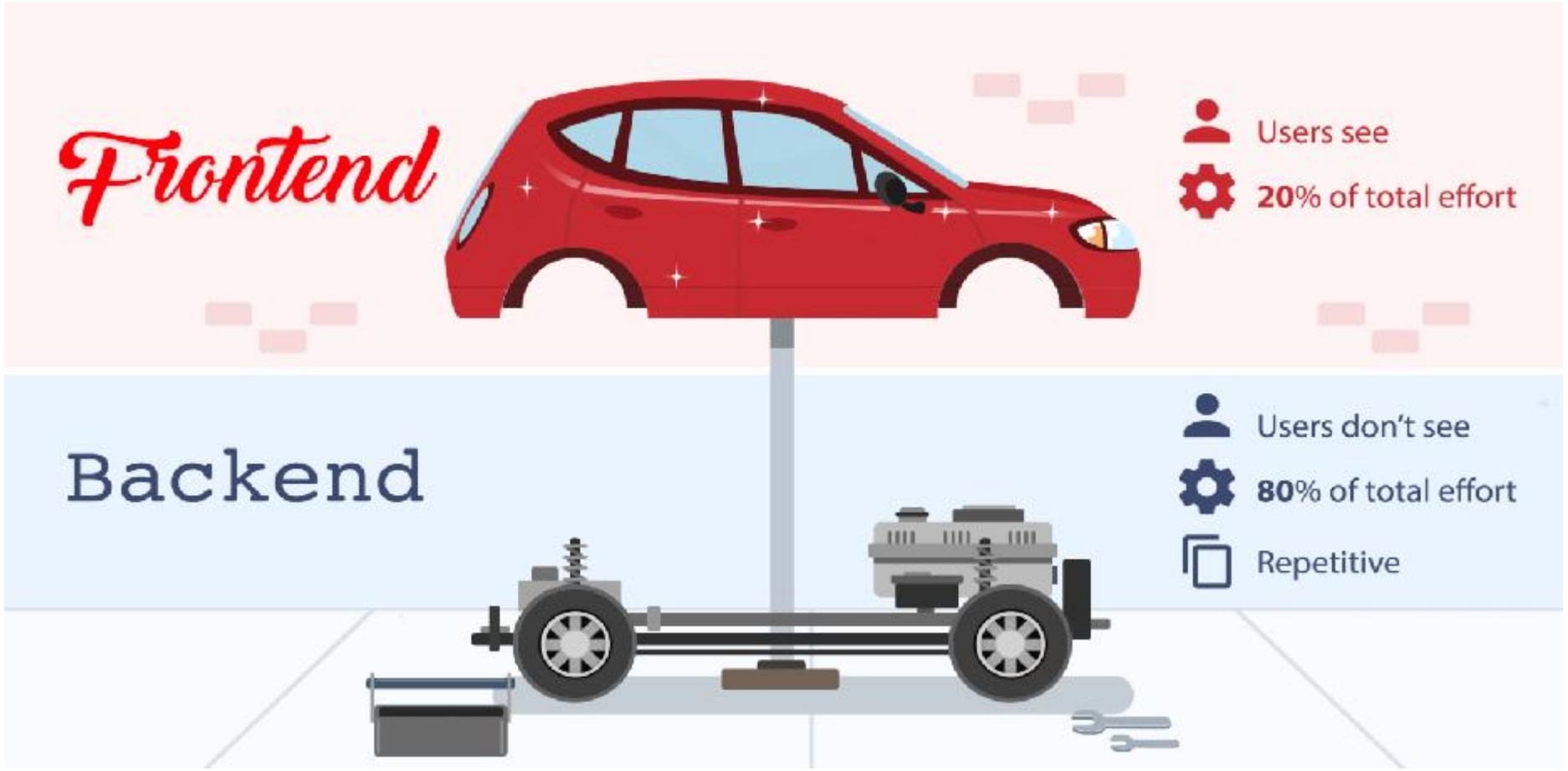


Image Source : Back4App

Frontend

Elements

- Not just about the beautification of the web / mobile interfaces

- Involves elements for
  - Content Structure
  - Styling
  - Interaction

- Deals with
  - Rendering of the content on the browsers / within apps
  - Beautification of content rendered
  - Interaction carried out by user on web pages / mobile apps

- Building blocks – HTML + CSS + JS!

Content Structure

Markup

- Structure of the page is
  - the foundation of websites
  - essential for search engine optimization
  - vital to provide the style and the interaction that the reader will ultimately use

- Hyper Text Meta Language (HTML)
  - will be at the very center of it all regardless of how complex the site is
  - uses tags, as opposed to a programming language, in order to identify all the various types of content out there on every single webpage

  - For example, in a newspaper article, a header, sub header, text body and other things are present
  - HTML works in the exact same way to label all the stuff on the webpage, except it uses HTML tags
  - even a JavaScript webpage, is comprised of HTML tags corresponding to each element on the webpage and every single content type is bundled into HTML tags as well.

Styling

Cascading Style Sheets

- A core functionality of front-end development
- lays out the page and give it both its unique visual flair and a clear, user-friendly view to allow readers

- An important aspect of styling is checking across several browsers and to write concise, terse code that is
  - specific yet generic at the same time
  - displays well in as many renderers as possible

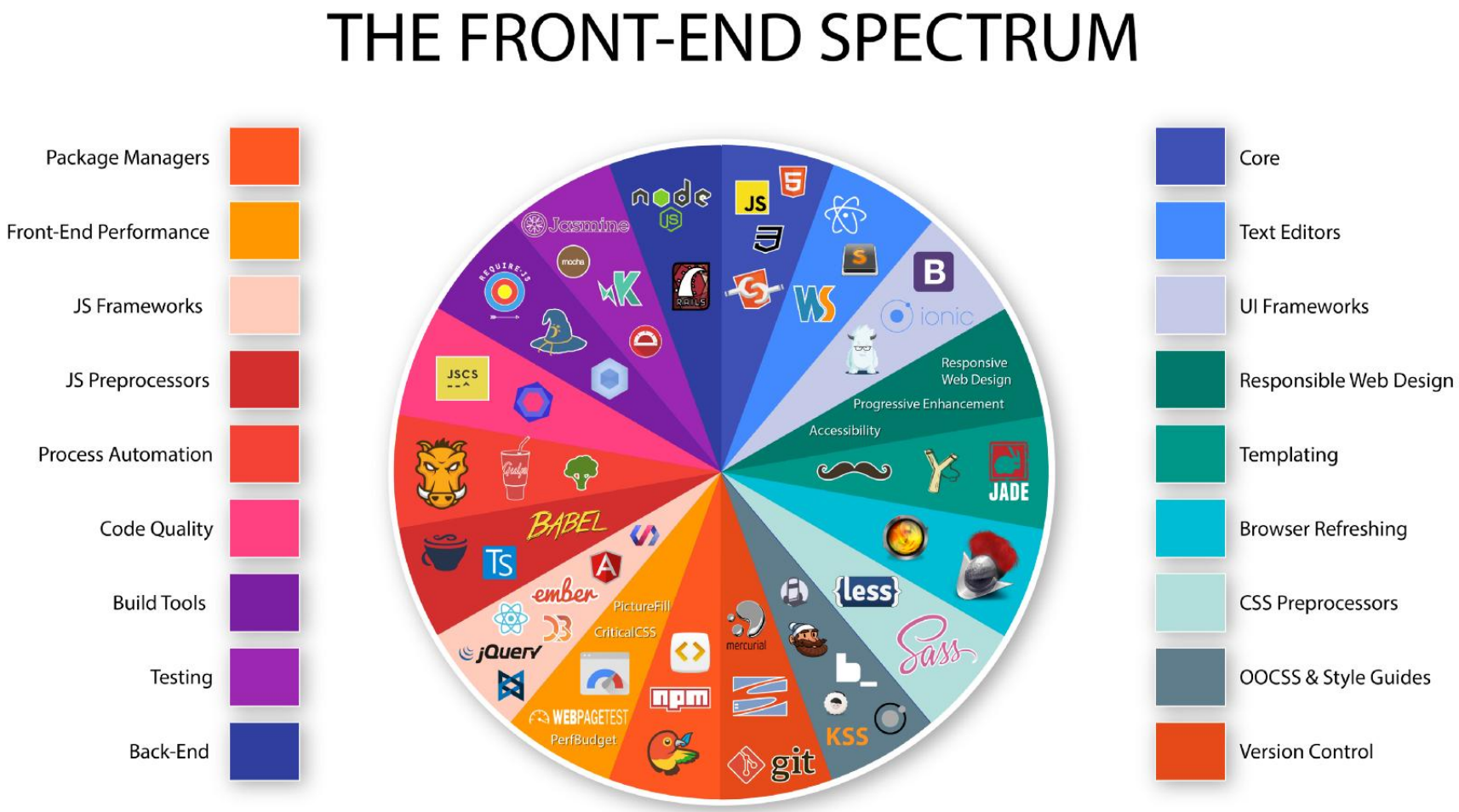
- CSS determines how all the HTML elements must appear on the frontend
  - HTML gives all of the raw tools necessary to structure webpage
  - CSS allows to style everything so it will appear to the user exactly the way you would like it to

Interaction

User Interaction and interaction with backend

- JavaScript
- User Interaction on page
  - Supported by all modern browsers
  - employed by pretty much every website in order to gain increased functionality and power
  - Used mainly for website content adjustment and to make the website itself act certain ways depending on the user's actions
  - Used for creating call-to-action buttons, confirmation boxes and adding new details to current information
- Dynamic Behavior
  - drastically improves a browser's default actions and controls
  - helps in placing asynchronous requests to server side and render the response received

Frontend Tools Spectrum



Source : Medium

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

FrontEnd - Development

Chandan Ravandur N

Typical Development scenario

- A front-end developer
  - architects and develops websites and applications using web technologies (i.e., HTML, CSS, DOM, and JavaScript)
  - which run on the web platform or act as compilation input for non-web platform environments (i.e., NativeScript)

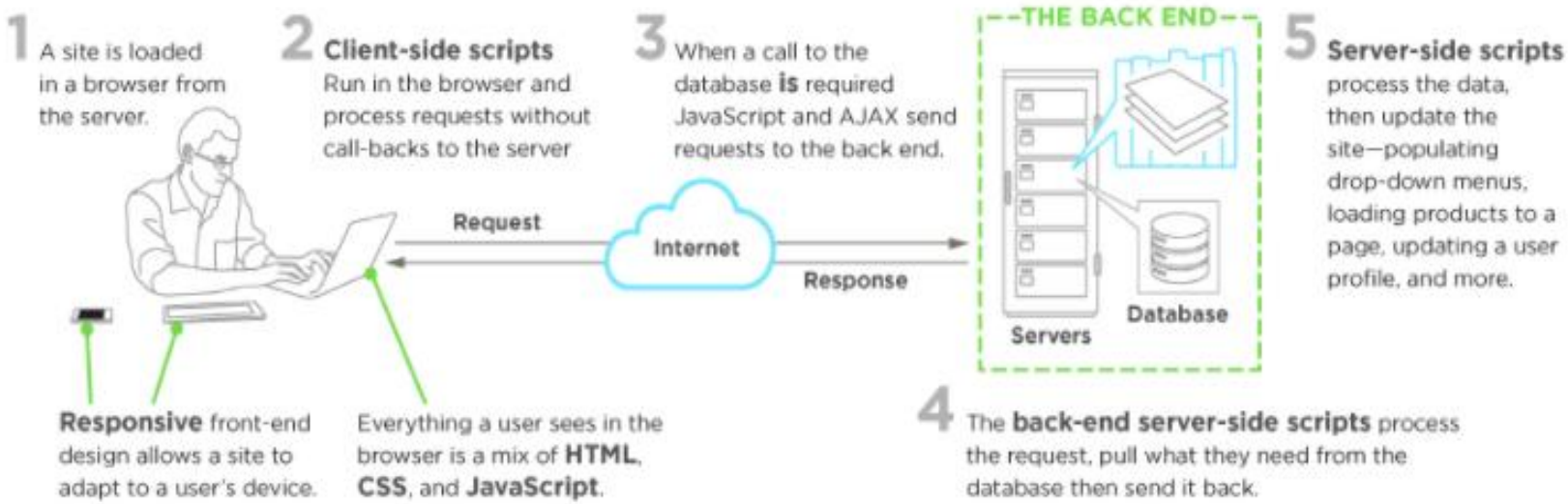


Image Source : upwork

Role of a front end developer

- A front-end developer crafts HTML, CSS, and JS that typically runs on the web platform (e.g. a web browser) delivered from one of the following operating systems (aka OSs):
  - Android
  - Chromium
  - iOS
  - Ubuntu (or some flavor of Linux)
  - Windows

- These operating systems typically run on one or more of the devices:
  - Desktop computer
  - Laptop / netbook computer
  - Mobile phone
  - Tablet
  - TV
  - Watch
  - Things (i.e., anything you can imagine, car, refrigerator, lights, thermostat, etc.)

Web Platform

- Front-end technologies can run on the aforementioned operating systems and devices using the following run time web platform scenarios:
  - A web browser (examples: Chrome, IE, Safari, Firefox).
  - A headless browser (examples: phantomJS).
  - A WebView/browser tab
  - A native application

Web Platform

Web Browsers

- Is software used to retrieve, present, and traverse information on the WWW
- Typically run on a desktop or laptop computer, tablet, or phone,
  - but as of late a browser can be found on just about anything (i.e., on a fridge, in cars, etc.).

- The most common web browsers are (shown in order of most used first):
  - Chrome
  - Internet Explorer
  - Firefox
  - Safari

Web Platform

Headless Browsers

- Are a web browser without a graphical user interface t
- that can be controlled from a command line interface programmatically
- Used for the purpose of web page automation
  - e.g., functional testing, scraping, unit testing, etc.

- Think of headless browsers as a browser that can be run from the command line that can retrieve and traverse web pages

- The most common headless browsers are:
  - PhantomJS
  - slimerjs
  - trifleJS

Web Platform

Webviews

- Are used by a native OS, in a native application, to run web pages
- Think of a webview like an iframe or a single tab from a web browser that is embedded in a native application running on a device
  - e.g., webviews iOS, android, windows

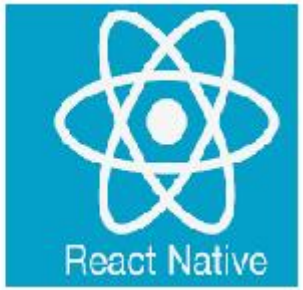
- The most common solutions for webview development are:
  - Cordova (typically for native phone/tablet apps)
  - NW.js (typically used for desktop apps)
  - Electron (typically used for desktop apps)

Web Platform

Native from Web Tech

- Web browser development can be used by front-end developers to craft code for environments that are not fueled by a browser engine
- Development environments are being dreamed up that use web technologies (e.g., CSS and JavaScript), without web engines, to create native applications

- Some examples of these environments are:
  - NativeScript
  - React Native



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

FrondEnd - Frameworks

Chandan Ravandur N

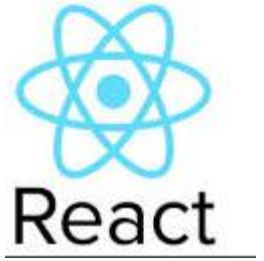
Why frameworks?

- For frontend developers, it’s increasingly challenging to make up their minds about which JavaScript application framework to choose
  - especially when need to build a single-page application

- Requirements
- First, modern frontend JavaScript frameworks must respect the Web Components specification.
  - should have support for custom HTML elements building
- Second, a solid JavaScript framework should have its own ecosystem
  - Ready solutions aim at solving various problems of client-side development such as
    - Routing
    - managing app state
    - communicating with the backend

React

- Stormed the JS world several years ago to become its definite leader
- Encourages to use a reactive approach and a functional programming paradigm
- Introduced many of its own concepts to define its unique approach to frontend web development
- Need to master a component-based architecture, JSX, and unidirectional data flow



- Ecosystem:
  - The React library itself plus React-DOM for DOM manipulation
  - React-router for implementing routes
  - JSX, a special markup language that mixes HTML into JavaScript
  - React Create App, a command line interface that allows you to rapidly set up a React project
  - Axios and Redux-based libraries, JS libraries that let you organize communication with the backend.
  - React Developer Tools for Chrome and Firefox browsers.
  - React Native for development of native mobile applications for iOS and Android.

Angular 2

- Marks a turning point in the history of the Angular framework
- Has substantially changed its architecture to come to terms with React
- Is from a Model-View-Whatever architecture to a component-based architecture

- Ecosystem:
  - A series of modules that can be selectively installed for Angular projects: @angular/common, @angular/compiler, @angular/core, @angular/forms, @angular/http, @angular/platform-browser, @angular/platform-browser-dynamic, @angular/router, and @angular/upgrade
  - TypeScript and CoffeeScript, supersets of JavaScript that can be used with Angular
  - Angular command line interface for quick project setup
  - Zone.js, a JS library used to implement zones, otherwise called execution context, in Angular apps
  - RxJS and the Observable pattern for asynchronous communication with server-side apps
  - Angular Augury, a special Chrome extension used for debugging Angular apps
  - Angular Universal, a project aimed at creating server-side apps with Angular
  - NativeScript, a mobile framework for developing native mobile applications for iOS and Android platforms

Vue

- At first sight, it may look like the Vue library is just a mix of Angular and React
- Borrowed concepts from Angular and React
- For example,
  - Vue wants you to store component logic and layouts along with stylesheets in one file. That's how React works without stylesheets
  - To let Vue components talk to each other, Vue uses the props and state objects - existed in React before Vue adopted it
  - Similarly to Angular, Vue wants you to mix HTML layouts with JavaScript

- The VueJS ecosystem consists of:
  - Vue as a view library.
  - Vue-loader for components.
  - Vuex, a dedicated library for managing application state with Vue; Vuex is close in concept to Flux.
  - Vue.js devtools for Chrome and Firefox.
  - Axios and vue-resource for communication between Vue and the backend.
  - Nuxt.js, a project tailored to creating server-side applications with Vue; Nuxt.js is basically a competitor to Angular Universal.
  - Weex, a JS library that supports Vue syntax and is used for mobile development.

Ember

- Like Backbone and AngularJS, is an “ancient” JavaScript framework
- But the fact that Ember is comparatively old doesn’t mean that it’s out of date
- Allows implement component-based applications just like Angular, React, and Vue do
- One of the most difficult JavaScript frameworks for frontend web development
- Realizes a typical MVC JavaScript framework, and Ember’s architecture comprises the following parts:
  - adapters, components, controllers, helpers, models, routes, services, templates, utils, and addons.

- The EmberJS ecosystem includes:
  - The actual Ember.js library and Ember Data, a data management library.
  - Ember server for development environments, built into the framework.
  - Handlebars, a template engine designed specifically for Ember applications.
  - QUnit, a testing JavaScript framework used with Ember.
  - Ember CLI, a highly advanced command line interface tool for quick prototyping and managing Ember dependencies.
  - Ember Inspector, a development tool for Chrome and Firefox.
  - Ember Observer, public storage where various addons are stored. Ember addons are used for Ember apps to implement generic functionalities.

Backbone/Marionette

- Is an MV* framework. Backbone partly implements an MVC architecture, as Backbone's View part carries out the responsibilities of the Controller
- Has only one strong dependency – the Underscore library that gives us many helper functions for convenient cross-browser work with JavaScript
- Attempts to reduce complexity to avoid performance issues

- The BackboneJS ecosystem contains:
  - The Backbone library, which consists of events, models, collections, views, and router
  - Underscore.js, a JavaScript library with several dozen helper functions that you can use to write cross-browser JavaScript
  - Underscore's microtemplating engine for Backbone templates
  - BackPlug, an online repository with a lot of ready solutions (Backbone plugins) tailored for Backbone-based apps
  - Backbone generator, a CLI for creating Backbone apps
  - Marionette, Thorax, and Chaplin – JS libraries that allow you to develop a more advanced architecture for Backbone apps

Compared

Name	Type	Shadow DOM EcmasScript 6+	Relative Popularity	Difficulty ofLearning
React	Library	Supported	*****	*****
Angular	Framework	Supported	***	*****
Ember	Framework	Supported	*	*****
Vue	Library	Supported	**	***
Backbone	Framework	Supported	*	***

Source : RubyGarage

Reference : RubyGarage Blog

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend

Chandndan Ravandu N

Server side

Interaction between front-end and back-end

- To understand server side, one need to know the front end and how the two interact

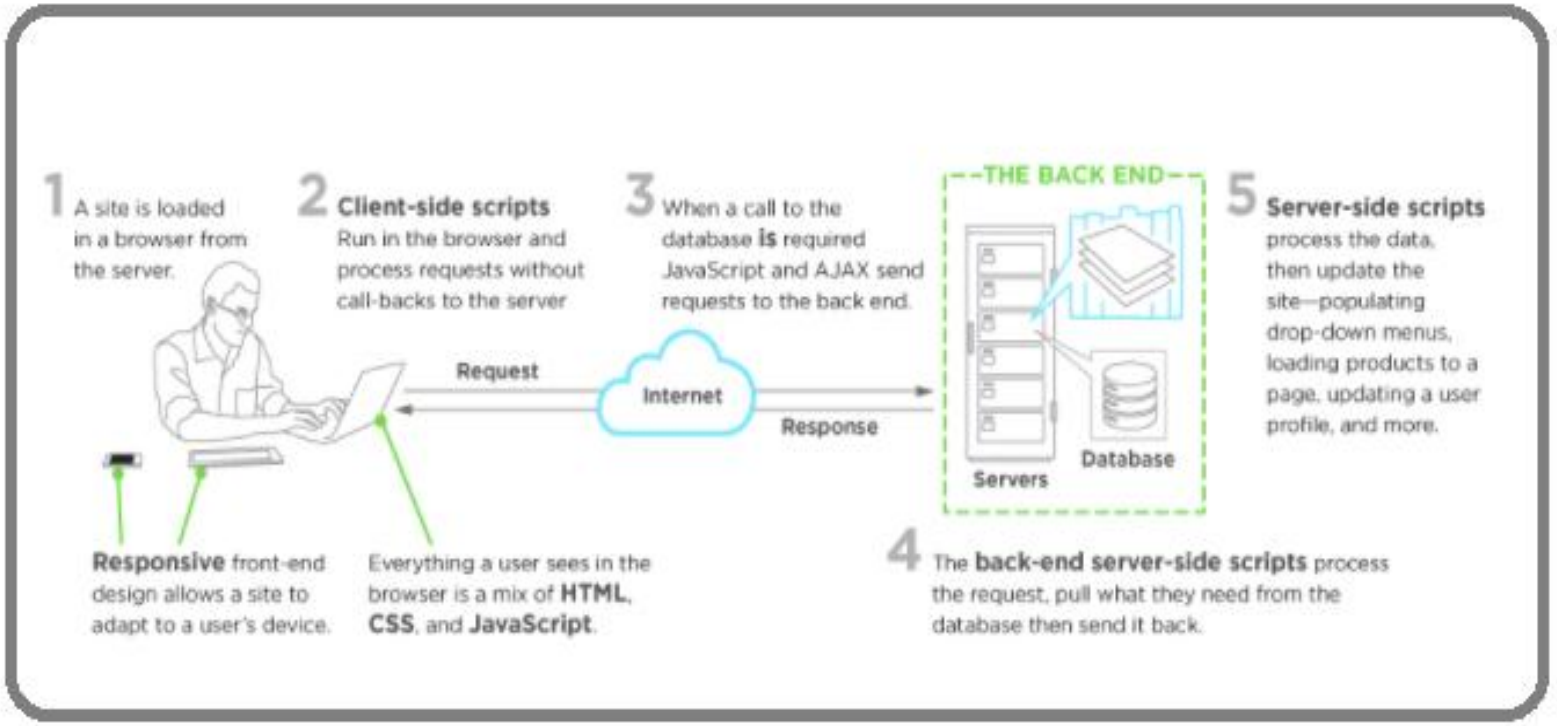


Image : Upwork

Server side

Interaction Explained

- The front end is what happens in the browser—everything the end users see and interact with
- The back end, on the other hand, happens on the server
  - on site, or in the cloud
  - databases

- Machinery that works behind the scenes—everything the end user doesn't see or directly interact with, but that powers what's happening!

- Server-side manages all those requests that come from users' clicks
  - Front-end scripts sends those requests over to the server side to be processed, returning the appropriate data to the front end
  - Often happens in a constant loop of requests and responses to the server

Server side architecture

Components

- Server (web / application server)
- Database
- Operating System
- API

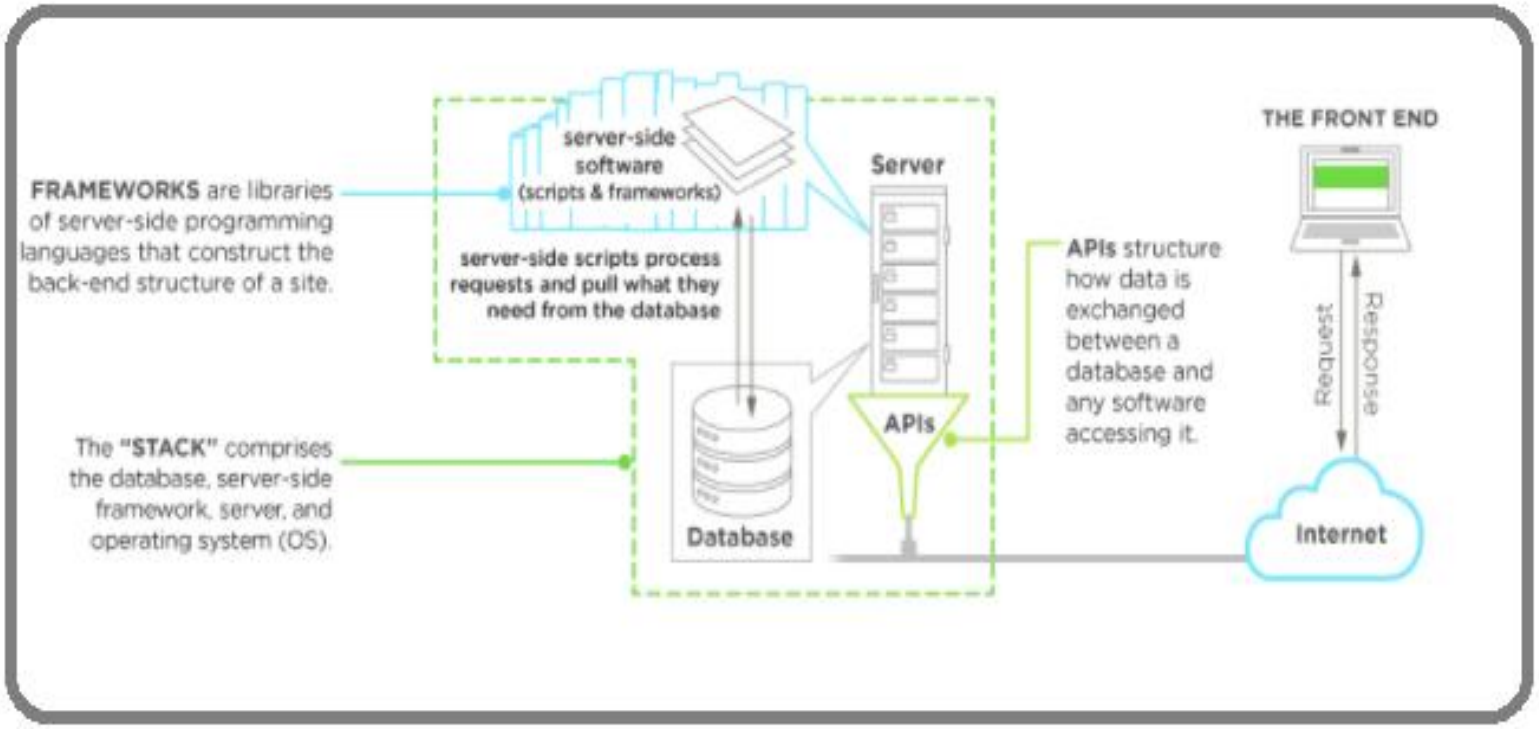


Image : Upwork

Server side architecture

Components

- The “traditional” back end was simple
  - Mix of the server, databases, APIs, and operating systems that power an app’s front end

- The back end of applications can look very different from application to application based upon
  - frameworks
  - cloud-based servers
  - containerization with a service like Docker
  - Backend-as-a-Service (BaaS) providers
  - APIs to replace more complex processing

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend - Components

Chandan Ravandur N

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend - WebServers

Chandan Ravandur N

Web Server

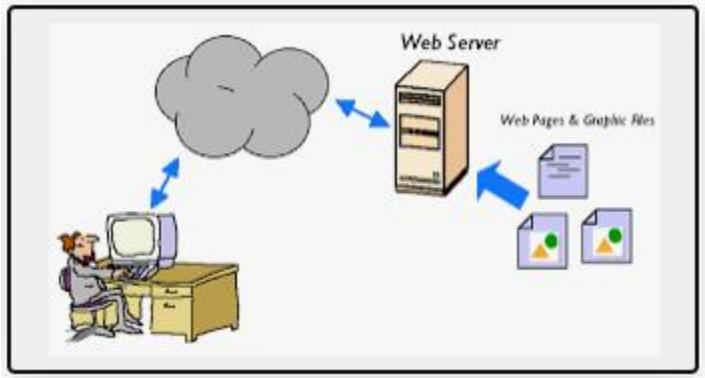
Defined

- All computers that host websites must have web server software

- Software and hardware that uses HTTP (Hypertext Transfer Protocol)to respond to client requests made over the World Wide Web
  - Main job of a web server is to display website content through storing, processing and delivering webpages to users
  - Also support SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol), used for email, file transfer and storage

  - Web server hardware is connected to the internet and allows data to be exchanged with other connected devices
  - software controls how a user accesses hosted files

- Used for
  - web hosting
  - hosting of data for websites
  - web applications



Web Server

Working

- Accessed through the domain names of websites and ensures the delivery of the site's content to the requesting user
- The software side at least an HTTP server - understand HTTP and URLs
- The hardware side, a computer that stores web server software and other files related to a website, such as HTML documents, images and JavaScript files

- A user needs a file hosted on web server
  - A person will specify a URL in a web browser's address bar
  - The web browser will then obtain the IP address of the domain name -- translating the URL through DNS (Domain Name System)
  - The browser will then request the specific file from the web server by an HTTP request
  - When the request is received by the web server,
    - the HTTP server will accept the request
    - find the content
    - send it back to the browser through HTTP
  - If the requested page does not exist or if something goes wrong, the web server will respond with an error message. The browser will then be able to display the webpage.

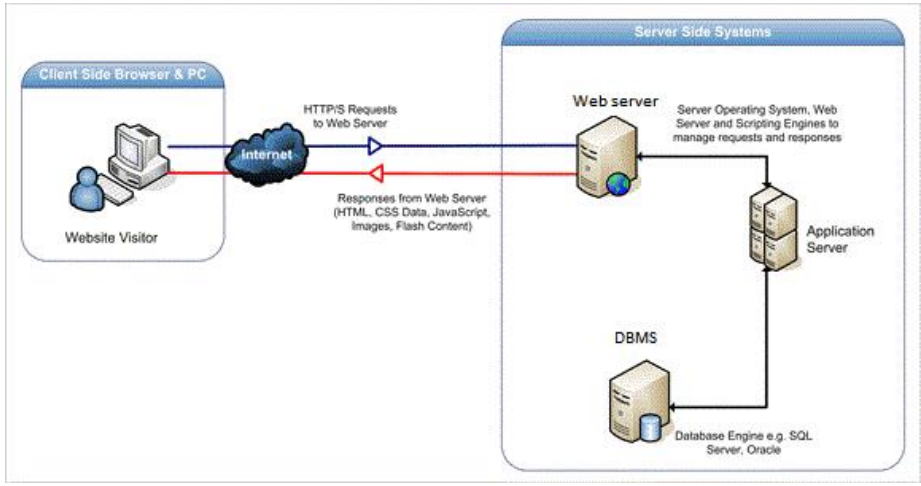


Image source : stackoverflow

Web Server - Types

Dynamic vs. static web servers

- A web server can be used to serve either static or dynamic content
  - Static refers to the content being shown as is
  - Dynamic content can be updated and changed
- A static web server
  - Consist of a computer and HTTP software
  - Considered static because the sever will send hosted files as is to a browser
- Dynamic web server
  - Consist of a web server and other software such as an application server and database
  - Considered dynamic because the application server can be used to update any hosted files before they are sent to a browser
  - Can generate content when it is requested from the database
  - Process is more flexible, it is also more complicated

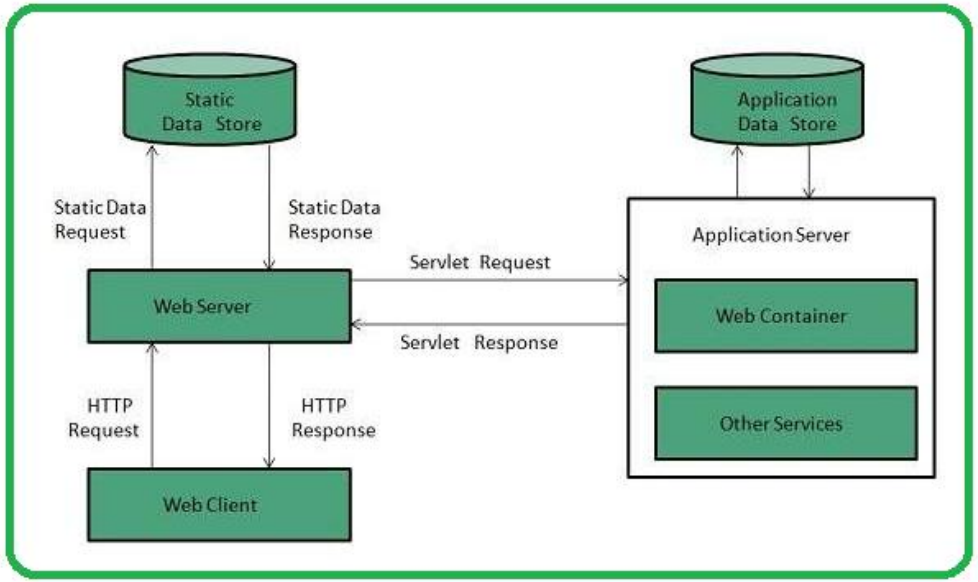


Image Source : TutorialsPoint

Common Web servers

- Apache HTTP Server
  - Developed by Apache Software Foundation
  - Free and open source web server
  - For Windows, Mac OS X, Unix, Linux, Solaris and other operating systems

- Microsoft Internet Information Services (IIS)
  - Developed by Microsoft for Microsoft platforms
  - Not open sourced, but widely used

- Nginx
  - A popular open source web server for administrators
  - Has very light resource utilization and scalability
  - Can handle many concurrent sessions due to its event-driven architecture
  - Can be used as a proxy server and load balancer

- Sun Java System Web Server
  - A free web server from Sun Microsystems that can run on Windows, Linux and Unix
  - Well-equipped to handle medium to large websites

Thank You!

In our next session:

Server side Components

Servers: The machinery

- Server acts as the lifeblood of the network
  - Can be on-site or in the cloud

- High-powered computers provide shared resources that need to run for application
  - file storage
  - security and encryption
  - databases
  - email
  - web services

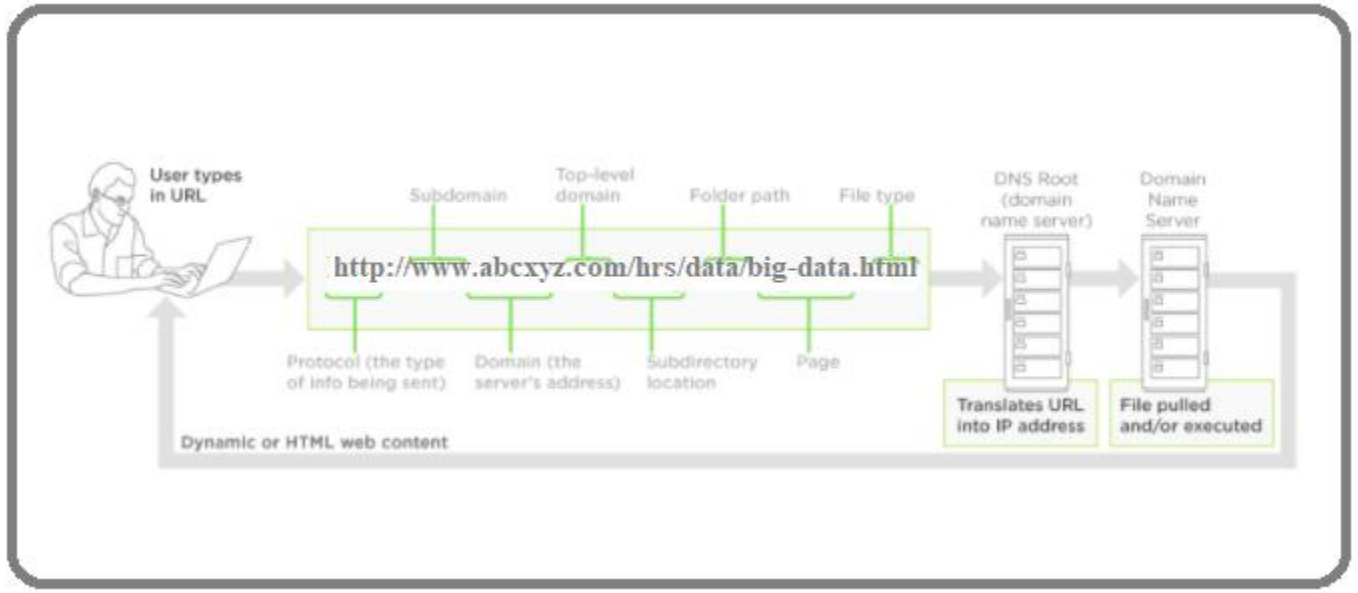


Image : Adapted from Upwork

Server side Components

Databases: The brains

- Are the brains that make websites dynamic
- Is responsible for accepting that query, fetching the data, and returning it to the website
  - Searching for a product in on online store
  - Searching for hotel locations within a specific state

- Accepts new and edited data when users of a website or application interact with them
- The client can change information in a database from the browser such as
  - posting articles to a CMS
  - uploading photos to a social media profile
  - updating their customer information

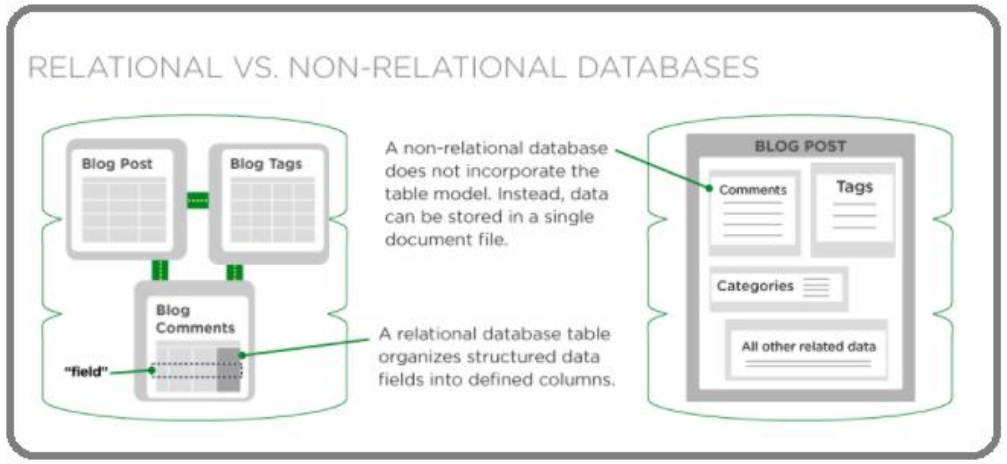
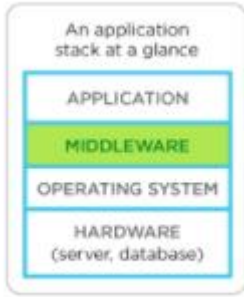


Image : Upwork

Server side Components

Middleware: The plumbing

- Any software on the server that connects an application's front end to its back end
- Think of it as plumbing for your site
  - pipes any communication, like requests and responses, back and forth between your application and server/database
  - you don't see middleware, but it's there and it has to be reliable and always do what's expected of it



- Middleware (server-side software) facilitates client-server connectivity, forming a middle layer between the app(s) and the network
- Can be multi-layered, organized into different layers of a site, whether it's the presentation layer or the business layer
- Web APIs can play into the stack, providing a bridge between the business layer and presentation layer

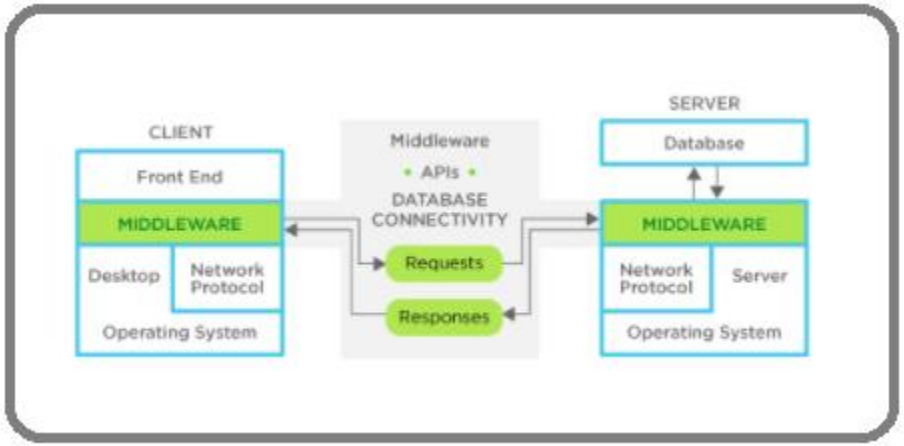


Image : Upwork

Server side Components

Programming languages & frameworks: The nuts & bolts

- Can choose from a variety of languages and frameworks depending on
  - the type of application being built
  - the specific processing requirements
  - other components already exist on the back end
- Languages will differ in file size, performance, compatibility, how many lines of code required, and the style of programming
  - Some back-end scripting languages are object-oriented programming languages
  - Other languages may be compiled rather than interpreted
  - affects load time, readability, and processing power required to run the application
- Big hitters in back-end programming, like:
  - Java
  - C# and C++
  - .NET
  - Perl
  - Scala
  - Node.js
- Frameworks that developers use as scaffolding to build server-side applications
  - Django (for Python)
  - Spring framework (for Java)
  - Node.js including MeteorJS and ExpressJS (for JavaScript with Node.js)
  - Ruby on Rails
  - Symfony (for PHP)

Server side Components

APIs: Crucial tech in Back-End programming

- Can’t talk about the back-end portion of an application these days without touching on APIs (application programming interfaces)
  - Connect software, applications, databases, and services together seamlessly
- Plays an integral role in how most server-side software architectures are built
  - Replacing more complicated programming to allow software to communicate and data to be transferred

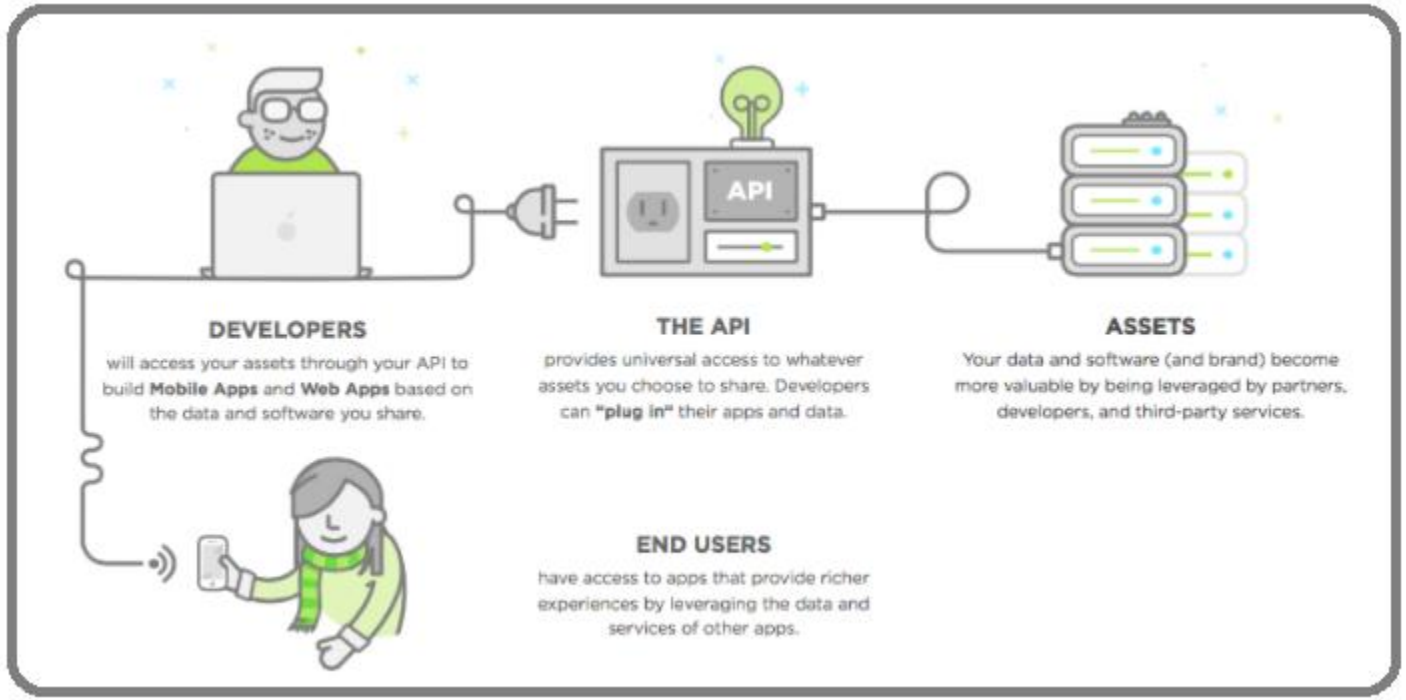
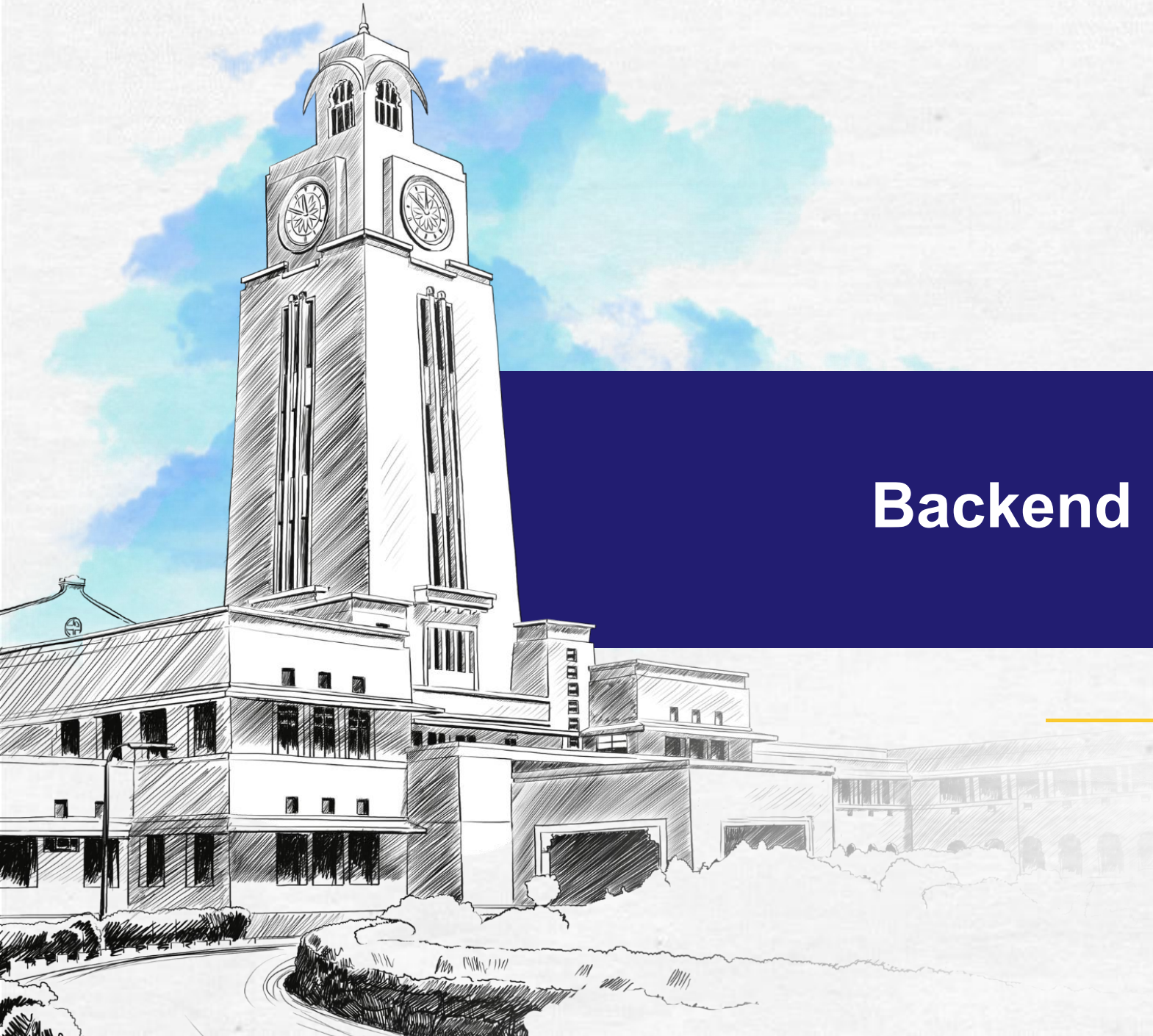


Image : Upwork

Thank You!

In our next session:



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend - Server Side Scripting

Chandan Ravandur N

Server Side Scripting

- A technique used in web development which involves employing scripts on a web server
  - which produce a response customized for each user's (client's) request to the website
- Scripts can be written in server-side scripting languages
  - PhP
  - Python
  - JSP
  - ASP .net
  - Java

- Is often used to provide a customized interface for the user
- Scripts may assemble client characteristics for use in customizing the response based on those characteristics, the user's requirements, access rights, etc.
- Enables the website owner to hide the source code that generates the interface, whereas with client-side scripting, the user has access to all the code received by the client
- A down-side to the use of server-side scripting is that the client needs to make further requests over the network to the server in order to show new information to the user via the web browser

Server-side script basics

- Runs on a server, embedded in the site's code
- Facilitates the transfer of data from server to browser, bringing pages to life in the browser
- Designed to interact with back-end permanent storage, like databases,
- Runs on-call. When a webpage is "called up," or when parts of pages are "posted back" to the server with AJAX, server-side scripts process and return data
- Powers functions in dynamic web applications
  - user validation
  - saving and retrieving data
  - navigating between other pages
- Build application programming interfaces (APIs)
  - which control what data and software a site shares with other apps

Popular server-side languages

- PHP
  - The most popular server-side language on the web, PHP is designed to pull and edit information in the database
  - Most commonly bundled with databases written in the SQL language
  - PHP-powered sites: WordPress, Wikipedia, Facebook

- Python
  - With fewer lines of code, is fast, making it ideal for getting things to market quickly
  - Emphasis is on readability and simplicity, which makes it great for beginners
  - The oldest of the scripting languages, is powerful, and works well in object-oriented designs
  - Python-powered sites: YouTube, Google, The Washington Post

- Ruby
  - Handles complicated logic on the database side of site vrey well
  - Great for startups, easy maintenance, and high-traffic demands
  - Requires developers to use the Ruby on Rails framework, which has vast libraries of code to streamline back-end development
  - Ruby-powered sites: Hulu, Twitter (originally), Living Social, Basecamp

Popular server-side languages

- C#
  - Language of Microsoft's .NET Framework—the most popular framework on the web
  - Combines productivity and versatility by blending the best aspects of the C and C++ languages
  - Excellent for developing Windows applications, and can be used to build iOS, Android mobile apps with the help of a cross-platform technology like Xamarin

- Java
  - Comes with a huge ecosystem of add-on software components
  - "Compile once, run anywhere" is its motto
  - Excellent for enterprise-level applications, high-traffic sites, and Android apps
  - Java sites: Twitter, Verizon, AT&T, Salesforce

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend - TechStacks

Chandan Ravandur N

Tech stack

- Is the combination of programming languages, tools and frameworks that the developers use to create web and mobile applications
  - Two main components to any application, known as client side and server side, also popular as front end and back end
- A stack is created when one layer of application is built atop the other
  - with the help of codes and hardware modules ranging from generic to specific
- A stack contains different layers of components/servers that developers use to build software applications and solutions

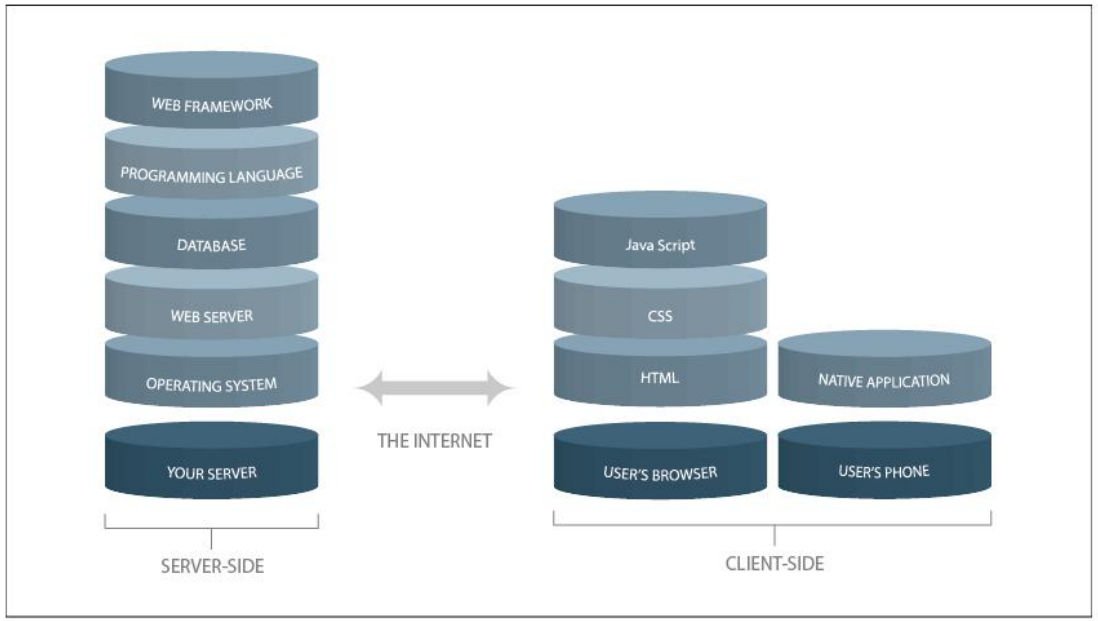


Image source : hackernoon

Tech stack

Deciphered

- Client-side
  - Is where the real interaction with the user happens
  - The user will interact with the website, the web app or a mobile app depending on what he is using
  - Three main technologies in front end: HTML, CSS and JavaScript

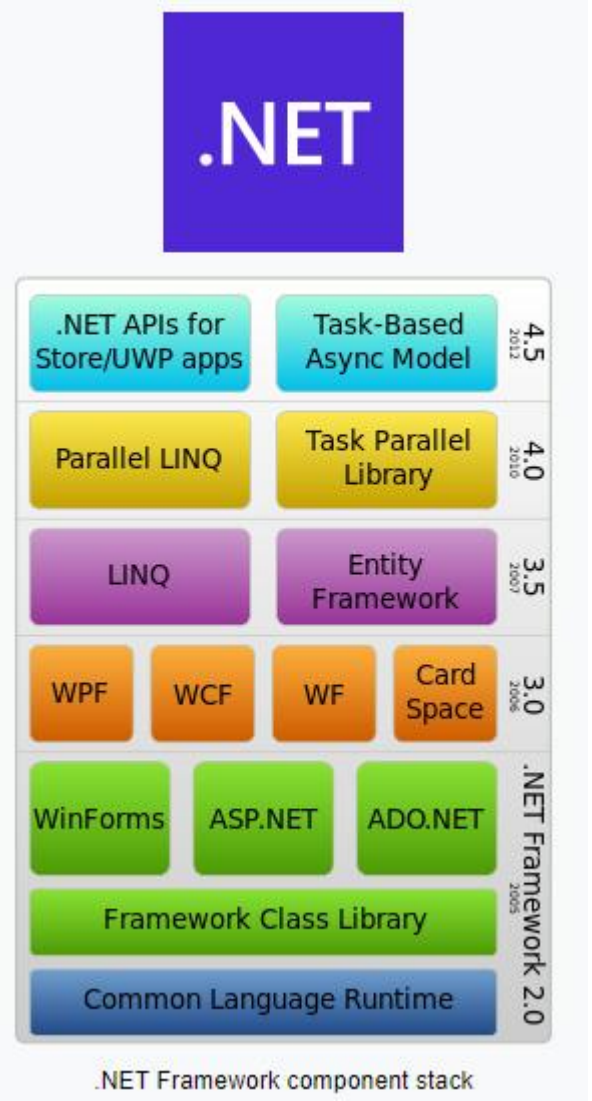
- Server-side
  - The back end consists of a server, an application (OS, Web server, Programming language, Web framework), and a database
  - Umbrella term used for websites where developers perform the following functions like
    - programming the business logic
    - server side hosting
    - app deployments
    - integrating with databases

- Common Stacks
  - .net
  - LAMP
  - MEAN / MERN / MEVN / MEEN
  - RoR

.net stack

- Developed by Microsoft
- Is a feature rich, thoroughly battle tested framework that lets to build dynamic and interactive web apps
- Subset of Overflow Stack, a comprehensive tech stack that fulfils all the requirements of web front-end, database and .NET developers
- C# language and .NET framework form the centerpiece
- Capable of supporting a wide variety of applications
  - right from small scale server side web applications
  - to huge enterprise-level, transaction processing systems

- .NET Stack has about 60 frameworks, platforms, SDKs, IDEs, SOA, libraries, etc. spread over 13 layers



.NET Framework component stack

Image source : wikipedia

LAMP

Linux, Apache, MySQL, Python / Perl/ Php

- Made of all open source software components is the one of the earliest stacks to get popular
- The most traditional stack model around, and very solid

- The main components are
  - Linux operating system,
  - Apache HTTP Server
  - MySQL RDBMS
  - PHP programming language
  - PHP is interchangeable with the languages Perl and Python.

- Variations of LAMP include:
  - WAMP (Windows/Apache/MySQL/PHP): uses IIS server.
  - LAPP (Linux/Apache/PostgreSQL/PHP):uses PostgreSQL database variation that's configured to work with enterprise-level projects
  - MAMP (Mac OS X/Apache/MySQL/PHP): Mac OS X operating system variation
  - XAMP (Linux, Mac OS X, Windows/Apache/MySQL/PHP, Perl): More complete bundle, and runs on Linux, Windows, and Mac operating systems

ME(A/R/V)N Stack

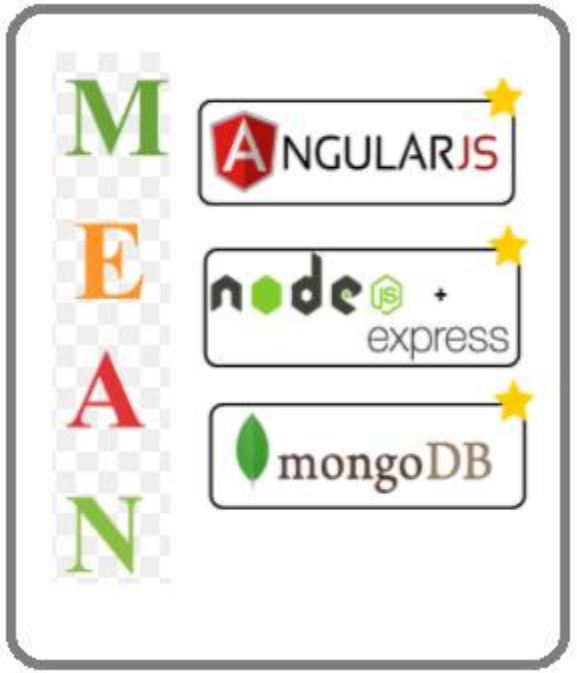
MongoDB, Express, Angular, Node

- Open source, free Javascript software stack developers use for building dynamic web apps and websites
- Supports MVC pattern

- Components are
  - MongoDB—a NoSQL database
  - Express.js—a web app framework that runs on Node.js
  - Angular JS (or Angular) Javascript MVC framework that run in browser JavaScript engines
  - Node.js—an execution domain for event-driven server-side

- Programs are coded in JavaScript - helps language uniformity
- Makes it easier for developer working on the client side, understand the codes of the server side
- Uses JSON for data transfer

- The variation of MEAN include:
  - MERN: MongoDB, Express.js, React.js and Node.js
  - MEVN: MongoDB, Express.js, Vue.js and Node.js.
  - MEEN: MongoDB, Express.js, Ember.js and Node.js



Ruby on Rails

- Facilitates quick app development thanks to its rich repository of gems—library integrations
- Highly scalable and it follows the ActiveRecord pattern
- Compatible with MySQL on the back end.

- Has its own built in database
  - Can abstract away the lower data base commands
  - Can write the codes in Active Record syntax and then swap it with lower level database specific queries
  - Useful when you don't use too much database specific queries

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend - API design styles

Chandan Ravandur N

REST

Representation State Transfer

- Most commonly known item in API space
- has become very common amongst web APIs
- First defined by Roy Fielding in his doctoral dissertation in the year 2000
- Architectural system defined by a set of constraints for web services based on
  - stateless design ethos
  - standardized approach to building web APIs

- Operations are usually defined using GET, POST, PUT, and other HTTP methodologies
- One of the chief properties of REST is the fact that it is hypermedia rich
- Supports a layered architecture, efficient caching, and high scalability

- All told, REST is a very efficient, effective, and powerful solution for the modern micro service API industry

gRPC

Backed by Google

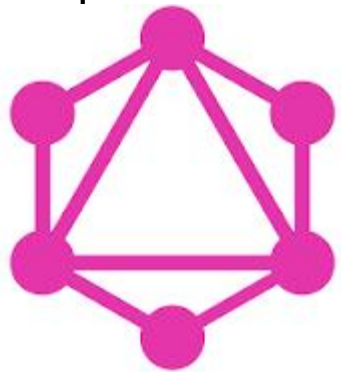
- Actually a new take on an old approach known as RPC, or Remote Procedure Call
- RPC is a method for executing a procedure on a remote server
- RPC functions upon an idea of contracts, in which the negotiation is defined and constricted by the client-server relationship rather than the architecture itself
  - RPC gives much of the power (and responsibility) to the client for execution
  - offloading much of the handling and computation to the remote server hosting the resource
- RPC is very popular for IoT devices and other solutions requiring custom contracted communications for low-power devices
  - gRPC is a further evolution on the RPC concept, and adds a wide range of features
- The biggest feature added by gRPC is the concept of protobufs
  - Protobufs are language and platform neutral systems used to serialize data, meaning that these communications can be efficiently serialized and communicated in an effective manner
- gRPC has a very effective and powerful authentication system that utilizes SSL/TLS through Google's token-based system
- Open source, meaning that the system can be audited, iterated, forked, and more

GraphQL

Backed by Facebook

- Approach to the idea of client-server relationships is unique amongst all other options
- GraphQL is a query language for APIs and a runtime for fulfilling those queries with existing data
- Client determines what data they want, how they want it, and in what format they want it in
  - Reversal of the classic dictation from the server to the client
  - Allows for a lot of extended functionality

- A huge benefit of GraphQL is - it typically delivers the smallest possible request
- More useful in specific use cases where
  - a needed data type is well-defined
  - a low data package is preferred



- Defines a “new relationship between client and data”

REST vs gRPC vs GraphQL

When to use?

- REST
  - A stateless architecture for data transfer that is dependent on hypermedia
  - Tie together a wide range of resources that might be requested in a variety of formats for different purposes
  - Systems requiring rapid iteration and standardized HTTP verbiage will find REST best suited for their purposes

- gRPC
  - A nimble and lightweight system for requesting data
  - Best used when a system requires a set amount of data or processing routinely and requester is either low power or resource-jealous
  - IoT is a great example

- GraphQL
  - An approach wherein the user defines the expected data and format of that data
  - Usefull in situations in which the requester needs the data in a specific format for a specific use

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backend as a Service (BaaS)

Chandan Ravandur N

BaaS

- is a platform that
  - automates backend side development
  - takes care of the cloud infrastructure

- App teams
  - outsource the responsibilities of running and maintaining servers to a third party
  - focus on the frontend or client-side development

- Provides a set of tools to help developers to create a backend code speedily
- with help of ready to use features such as
  - scalable databases
  - APIs
  - cloud code functions
  - social media integrations
  - file storage
  - push notifications

Apps suitable for BaaS

- Social media apps
  - alike Facebook, Instagram
- Real-time chat applications
  - alike WhatsApp
- Taxi apps
  - alike Uber, Ola
- Video and music streaming apps
  - similar to Netflix
- Mobile games
- Ecommerce apps

Why Backend as a service?

- A BaaS platform solves two problems:
  - Manage and scale cloud infrastructure
  - Speed up backend development

- Business reasons to use BaaS:
  - Reduce time to market
  - Save money and decrease the cost of development
  - Assign fewer backend developers to a project
  - Outsource cloud infrastructure management

Technical reasons to use BaaS:

- Focus on frontend development
- Excludes redundant stack setup
- No need to program boilerplate code
- Standardize the coding environment
- Let backend developers program high-value lines of code
- Provides ready to use features like authentication, data storage, and search

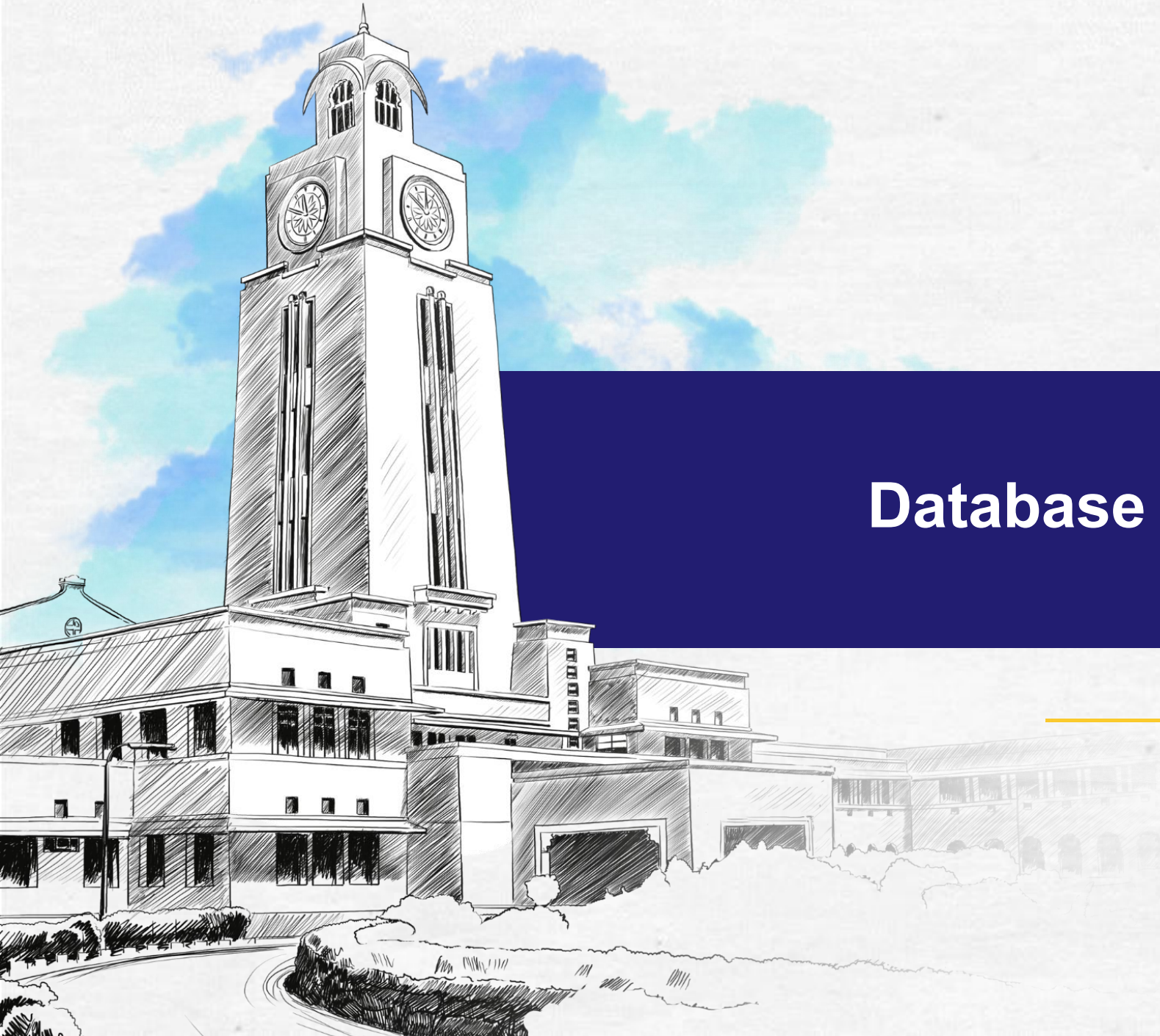
Pros-Cons

- Advantages
  - Speedy Development
  - Reduced Development price
  - Serverless, and no need to manage infrastructure

- Disadvantages
  - Less flexible as compared to custom coding / deployments
  - Less customization in comparison to a custom backend
  - Vendor lock-in possible

Thank You!

In our next session:



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Database Management Systems

Chandan Ravandur N

Traditional file systems for storing the data

Issues

- In any business application, information about customers and savings accounts etc. need to be stored
- One way to keep the information on computers is to store in files provided by operating systems (OS).

- Disadvantages of the above System
  - Difficulty in accessing data (possible operations need to be hardcoded in programs)
  - Redundancy leading to inconsistency
  - Inconsistent changes made by concurrent users
  - No recovery on crash
  - The security provided by OS in the form of password is not sufficient
  - Data Integrity is not maintained

DBMS

Solution to Data Storage and Retrieval issues

- Databases and Systems to manage them have become significant components of any present day business of any nature

- These databases help businesses to perform their day-to-day activities in an efficient and effective manner
  - Banking
  - Travel ticket reservation
  - Library catalog search

- Advances in technology have given raise to new concepts-
  - Multimedia databases
  - GIS
  - Web data
  - Data warehousing and mining

Data and Databases

Related

- Data
  - Known fact that can be recorded and that has implicit meaning
  - Example – Students data
    - ID
    - Name
    - Telephone number
    - Email id
    - Programme
  - The data can be stored in a file on a computer

- Database
  - A collection of logically related data
  - Example – University database
    - collection of all students data
    - courses data
    - faculties data
  - A database is designed, built and populated with data for a specific purpose

DBMS

Defined

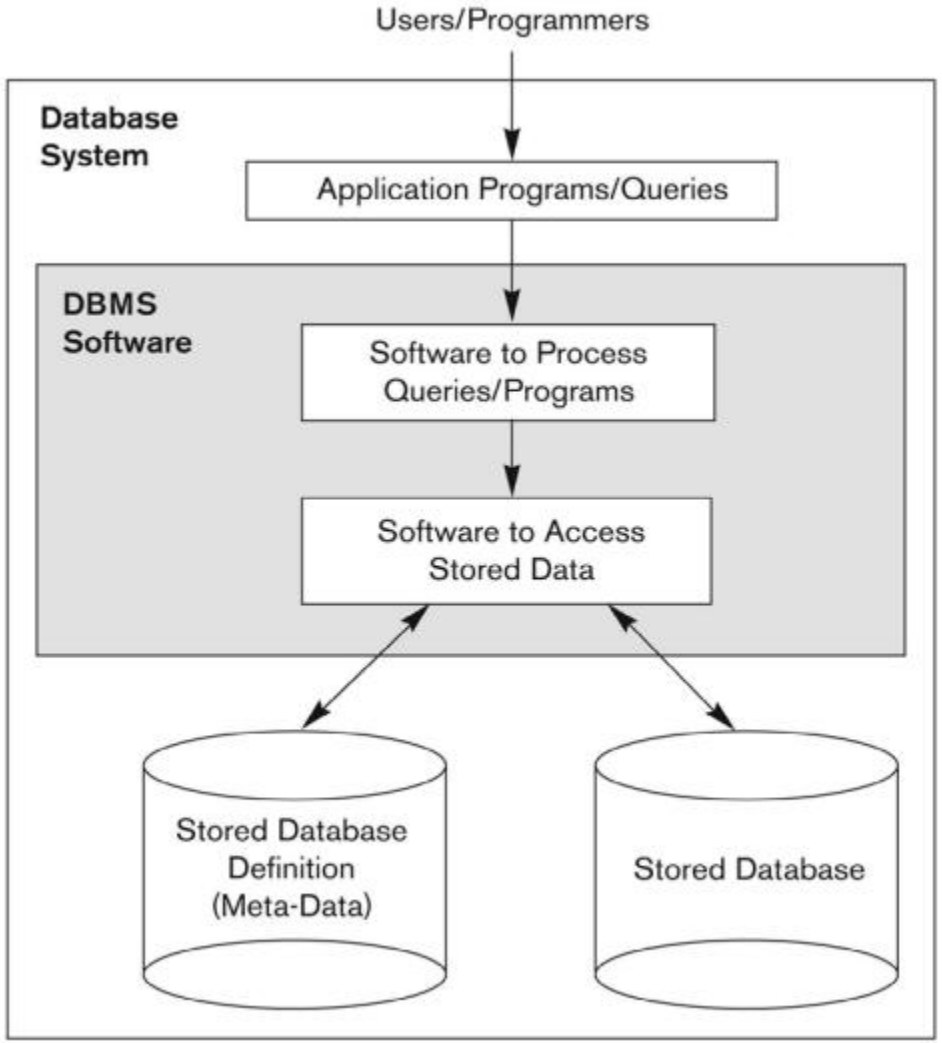
- A collection of programs that enables users to create and maintain databases in a convenient and effective manner

- DBMS is a software system that facilitates the following:

1. Defining the database
   - includes defining the structures, data types, constraints, indexes etc.
   - Like Database catalog/Data dictionary/ called as Meta-data
2. Constructing the database
   - means storing data into the database structures and storing on some storage medium
3. Manipulating database for various applications
   - encompasses activities like –
     - querying the database
     - inserting new records into the database
     - updating some data items
     - deleting certain items from the database

DBMS

A simplified database system environment



Adapted from Fundamentals of Database Systems by Elmasri, Navathe

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DBMS - Actors

Chandan Ravandur N

Actors

Users of DBMS

- Day Today using Databases (on the scene)
  - Database Admins
  - Database Designers
  - System Analysts and Application Programmers
  - End users

- Maintains the databases (behind the scene)
  - System designers and implementers
  - Tool developers
  - Operators and maintenance personnel

On the scene actors

Deals directly with data

- DBA
  - Oversees and manages the database resources
  - Responsible for access management, usage monitoring
  - Needs to look into security aspects, and performance issues
  - In large organizations, pool of DBAs can be deployed for these purpose

- Database Designers
  - Defines the schema and actual data which needs to be stored in database
  - Identifies the data to be stored in the database and select proper structures for the same
  - Needs to interact with all stakeholders to understand their data requirements and then design the database matching to those requirements
  - Needs to design views of database based on users requirements

On the scene actors

- System Analysts
  - Determine the requirements of end users and prepares the standard canned transactions
- Application Programmers
  - Implements the specifications as programs, test, debug and maintain it
  - Needs to be familiar with full range of capabilities of DBMS
- End Users
- Casual
  - occasionally access database through standard query interface
  - Middle or high level managers
- Naïve (Parametric)
  - sizeable portion of users
  - uses standard types of queries (canned transactions) to deal with database
- Sophisticated
  - knows thoroughly about capabilities of DBMS and implements query / programs to fulfil their data requirements
- Standalone
  - uses menu based user interface to interact with database

Behind the Scene

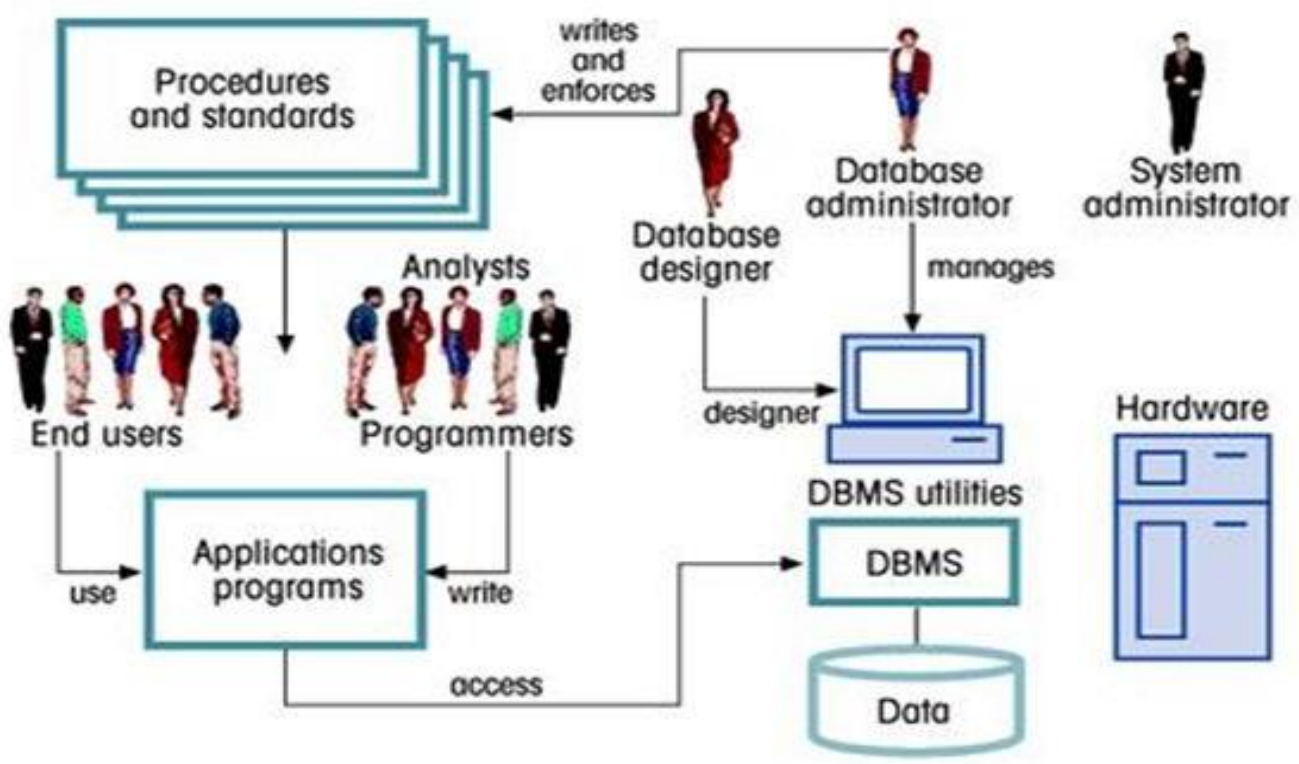
Not interested into the content of database

- System designers and implementers
  - DBMS is complex software – backup , security, catalog, query processing , interface etc.
  - Design and implement these modules and interfaces

- Tool developers
  - Implements tools that facilitates database modeling and design, database system design etc.
  - Many times optional, needs to be purchased / obtained separately need basis
  - Performance monitoring tools, graphical interfaces etc.

- Operators and Maintenance personnel
  - Responsible for actual running and maintenance of hardware and software environment for DBMS

In Summary



The Database System Environment

Source : MyReadingRoom

Reference :
Fundamentals of Database Systems, by Elmasri, Navathe

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Advantages Disadvantages of DBMS

Chandan Ravandur N

DBMS

Advantages

- Data independence
- Efficient data access
- Data integrity and security
- Easier Data Administration
- Concurrent access and Crash recovery
- Reduced application development time

DBMS

Disadvantages

- Increased cost
  - Hardware / software cost
  - Training cost
  - People cost
- Increased complexities
  - Lot of components involved – backup , recovery , replication utilities
  - Integration among sub systems is important
  - Specialized skillsets required
- Frequent upgrades / maintenance cycles
  - Patches needs to be applied
  - Versions need to be upgraded
  - Needs to insure that nothing is broken

When not to use DBMS?

- Overhead costs associated with DBMS
  - High initial investment in hardware, software and training
  - The generality that DBMS provides for defining and processing data
  - Overhead for providing security, concurrency, recovery and integrity function

- Develop customized database applications for
  - Simple, well defined feature sets that are not expected to change at all
  - Stringent, real time requirements
  - Embedded systems with limited storage
  - No multiple users access to data is required

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Classification of DBMS

Chandan Ravandur N

Classification of DBMS

Criteria's

- Data Model
  - On which DBMS is based
- Number of Users supported
  - Accessing the database
- Number of sites
  - Over which database is distributed
- Cost
- Usage Purpose

Classification of DBMS

Data Model

- Relational Data model
  - Quite common in commercial DBMS products
  - Aka SQL systems
  - Oracle , SQLServer, MySQL
- Object Data model
  - Not much usage - Tornado
- NoSQL Data model
  - Aka Big Data systems
  - Further classified as
    - Document based – MongoDB
    - Graph based – Neo4J
    - Column based – HBase, Cassandra
    - Key-value based – Redis, DynamoDB
- Tree structured model
  - Conventional XML based systems

Classification of DBMS

Number of Users supported

- Based upon concurrent number of users supported

- Single User Systems
  - Supports only one user at a time and used in PCs

- Multi User Systems
  - Includes majority of DBMS
  - Support concurrent multiple users

Classification of DBMS

Number of sites

- Number of places where database is distributed

- Centralized
  - Data is stored at a single computer site
  - Can support multiple users
  - Prone to failure

- Distributed
  - Actual database and DBMS software distributed over many sites connected by network
  - Big Data systems are common with massive distributed architecture
  - Data often replicated on multiple sites – making it failure proof

Classification of DBMS

Cost

- Open Source Products
  - Source code available for anybody
  - Driven by community of developers
  - MySQL, PostgreSQL
- Open Source with Third party vendors
  - Third party vendors provides support with additional services
- Licenced versions
  - Most commercial DBMS products offers various licences
  - Personnel usage – less cost, may not all features supported / required
  - Based on sites count – allows unlimited use of DBMS with any number of copies running on customer site
  - Based on users count – number of concurrent users supported

Classification of DBMS

Usage Purpose

- Special Purpose
  - When performance is primary concern , such special purpose systems are used
  - Optimized
  - Can’t be used for other types of data
  - DBMS for embedded devices

- General Purpose
  - Flexible enough to be used various purposes

Reference :
Fundamentals of Database Systems, by Elmasri, Navathe

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

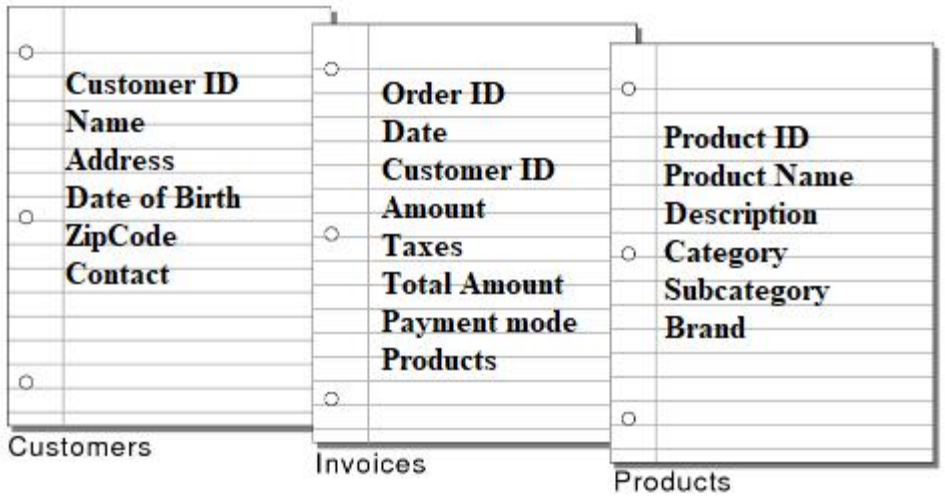
Relational DBMS

Chandan Ravandur N

Table

- Table
  - An arrangement of words, numbers or signs , or combinations of them
  - as in parallel columns,
  - To exhibit a set of facts or relations in a definite, compact and comprehensive form

- Webster’s Dictionary of English Language



Relational DBMS

Table based

- Relational model proposed by Codd in 1970
- Revolutionized the database field and replaced earlier models
  - Hierarchical and network data models
- Several vendors offering relational DBMS software
  - Oracle, IBM DB2, Informix, Sybase, MS Access, MS SQL Server etc

- Ubiquitous in marketplace and multibillion dollar industry!

- Bases on very simple and elegant data model
- Provides easy to understand use query language

Relational model

Relation

- Based on concept of relation or table

- Database is represented as collection of one or more relations
- Each relation is table with rows and columns

- Simple to understand for novice users as well
- Structured Query Language (SQL) allows use of simple, high level language to query data

- Advantage
  - Simple data representation
  - Ease with which complex queries can be written

ID	Name	Age	Marks
1	Suresh	21	34
2	Dinesh	22	21
3	Mahesh	21	45

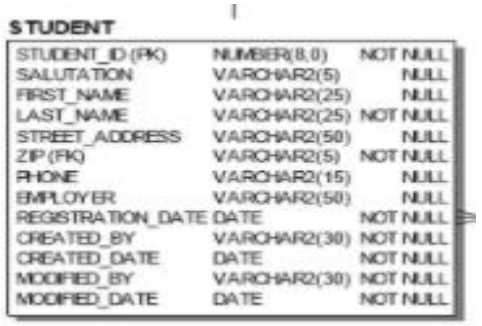
Relation

Explained

- Relation is main construct of RDBMS
- Consists of relation schema and relation instance

- Relation schema
  - describes the columns of the table
  - Specifies the relations name, name of each field, type of each field

- Relation instance
  - table containing the records
  - Set of tuples
  - Each tuple has same number of fields as the schema



STUDENT

STUDENT_ID (PK)	NUMBER(8,0)	NOT NULL
SALUTATION	VARCHAR2(5)	NULL
FIRST_NAME	VARCHAR2(25)	NULL
LAST_NAME	VARCHAR2(25)	NOT NULL
STREET_ADDRESS	VARCHAR2(50)	NULL
ZIP (FK)	VARCHAR2(5)	NOT NULL
PHONE	VARCHAR2(15)	NULL
EMPLOYER	VARCHAR2(50)	NULL
REGISTRATION_DATE	DATE	NOT NULL
CREATED_BY	VARCHAR2(30)	NOT NULL
CREATED_DATE	DATE	NOT NULL
MODIFIED_BY	VARCHAR2(30)	NOT NULL
MODIFIED_DATE	DATE	NOT NULL

ID	Name	Age	Marks
1	Suresh	21	34
2	Dinesh	22	21
3	Mahesh	21	45

SQL

Language of RDBMS

- SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS)
- The scope of SQL includes
  - data query
  - data manipulation (insert, update and delete)
  - data definition (schema creation and modification)
  - data access control

- SQL Consists of sublanguages
- Data definition or data description language (DDL)
  - A syntax for creating and modifying database objects such as tables, indexes, and users
  - Statements are similar to a computer programming language for defining data structures, especially database schemas
  - Common examples of DDL statements include CREATE, ALTER, and DROP

- Data manipulation language (DML)
  - A syntax for adding (inserting), deleting, and modifying (updating) data in a database
  - Statements consists of different clauses like where , group , having etc.

Reference :
Database Management Systems, by Ramakrishna

Thank You!

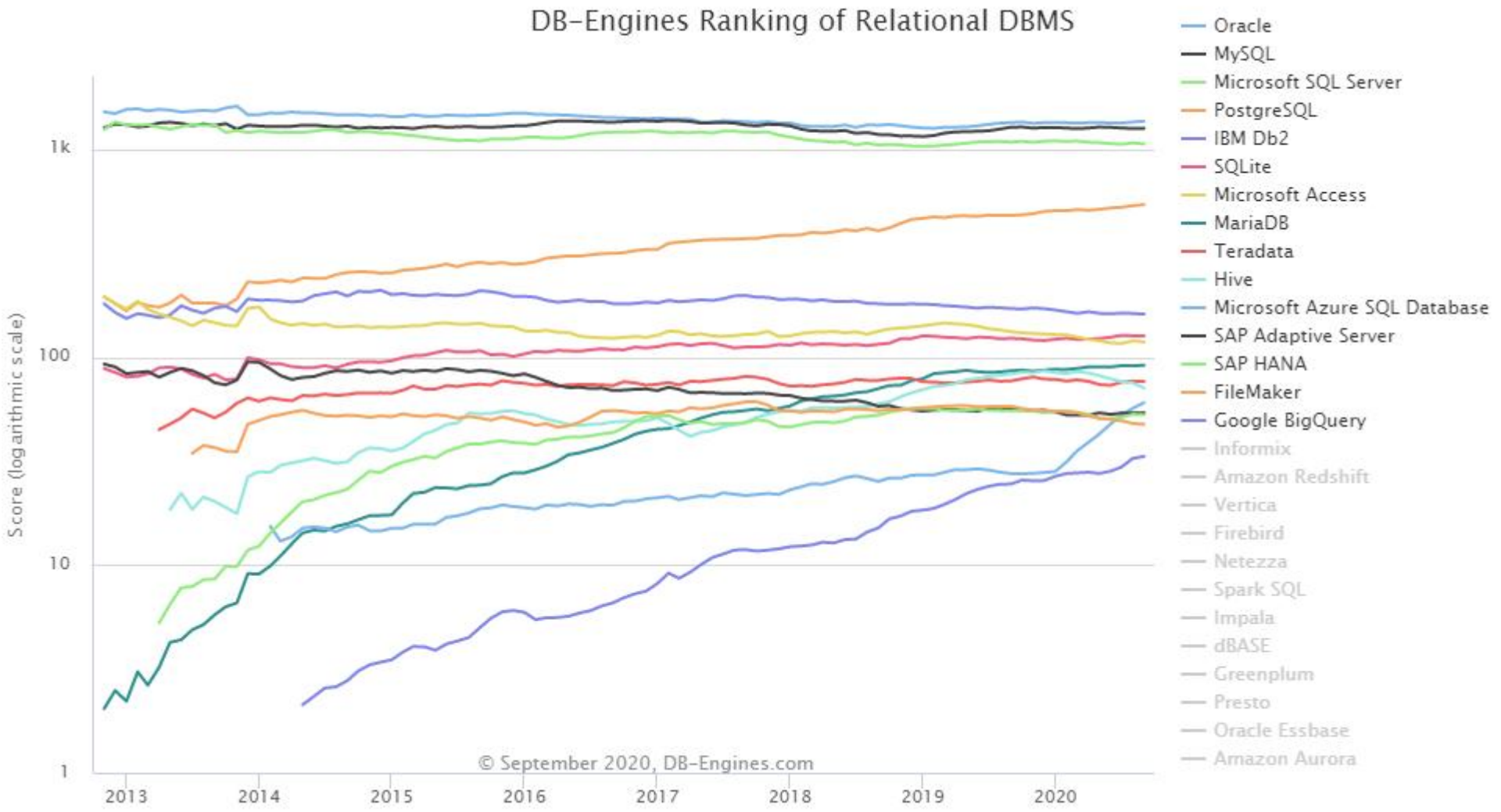
In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

RDBMS - Landscape

Chandan Ravandur N

RDBMS Landscape



Source : db-engines

Dominant Players

- Commercial Vendors

- Open Source Challengers

RDBMS

Advantages

- Well-documented and mature technologies, and RDBMSs are sold and maintained by a number of established corporations
- SQL standards are well-defined and commonly accepted
- A large pool of qualified developers have experience with SQL and RDBMS
- All RDBMS are ACID-compliant, meaning they satisfy the requirements of
  - Atomicity
  - Consistency
  - Isolation
  - Durability

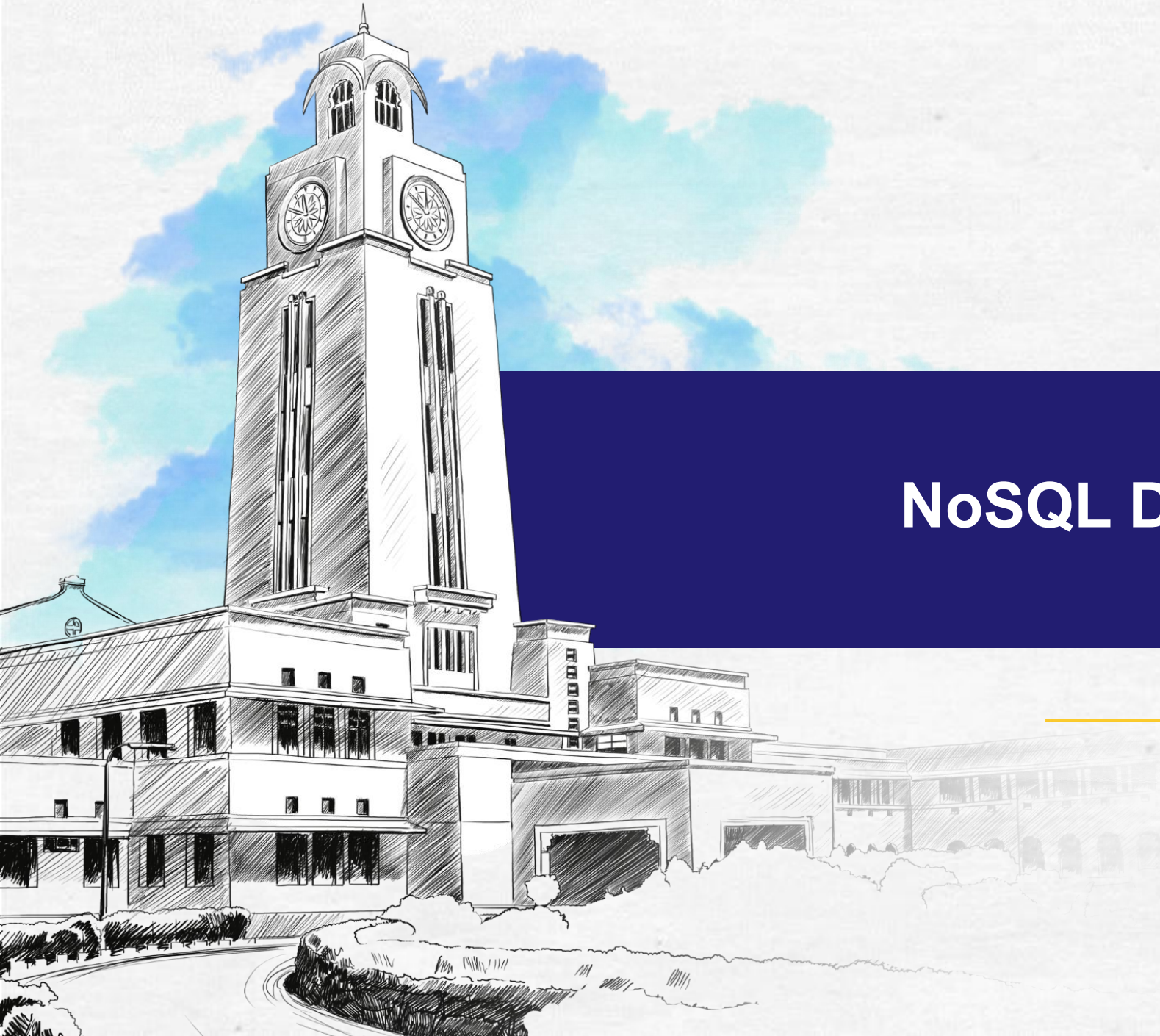
RDBMS

Disadvantages

- RDBMSs don't work well — or at all — with unstructured or semi-structured data due to schema and type constraints
  - Makes them ill-suited for big data analytics
- The tables in relational database will not necessarily map one-to-one with an object or class representing the same data
  - Resulting into mismatch between tables and data structures used in programme
- When migrating one RDBMS to another, schemas and types must generally be identical between source and destination tables for migration to work (schema constraint)
- Extremely complex datasets or those containing variable-length records are generally difficult to handle with an RDBMS schema

Thank You!

In our next session:



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

NoSQL Databases - Categories

Chandan Ravandur N

NoSQL

- Not Only SQL
- Meant to convey that many applications need systems other than traditional relational SQL systems to manage their data management needs

- Most of them are distributed databases or distributed storage systems with focus on
  - ✓ Semi structured data storage
  - ✓ High performance
  - ✓ Availability
  - ✓ Data replication
  - ✓ Scalability

Image Source : DataVarsity

Emergence of NoSQL systems

Use cases

- Free email application like Gmail
  - Have millions of users, each user with thousands of emails – need for storage
  - SQL may not be appropriate because
    - ✓ It has too many services which are not required here
    - ✓ Structured data model is restrictive

- Facebook – social media platforms
  - Have millions of users, each user submits posts – images, videos, text , displayed on other users pages
  - User profiles, relationships , posts requires – storage
  - Needs to show the posts to other users – processing
  - SQL many not be appropriate because
    - ✓ Can not handle some of this data well
    - ✓ Can not handle when storage is huge and distributed across nodes

Emergence of NoSQL systems (2)

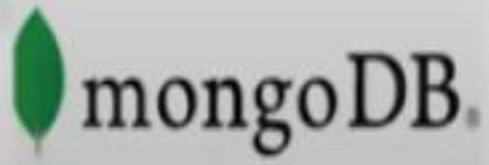
Soultions

- BigTable
  - ✓ Googles proprietary NoSQL systems used in many of Googles applications
  - ✓ dealing with vast amounts of data storage like Gmail, Maps, Web site indexing etc.
  - ✓ Apache Hbase is open source version of BigTable
  - ✓ Column-based or wide column stores

- DynamoDB
  - ✓ Amazons NoSQL system available on AWS
  - ✓ Innovation in key-value type data stores

- Cassandra
  - ✓ Facebooks NoSQL system
  - ✓ Uses concepts from both key-value stores and column-based systems

- MongoDB
  - ✓ Document based NoSQL system



Characteristics of NoSQL systems

- Related to Distributed databases and distributed systems
  - ✓ Scalability
  - ✓ Availability
  - ✓ Replication models
  - ✓ Sharding
  - ✓ High performance

- Related to data models and query languages
  - ✓ No fixed schema
  - ✓ Less powerful query languages
  - ✓ Versioning

Characteristics of NoSQL systems (2)

Related to Distributed databases and distributed systems

- Scalability
  - ✓ Horizontal or Vertical
  - ✓ In NoSQL systems, horizontal scalability is used by adding more nodes for data storage and processing as the volume of data grows
  - ✓ Used when system is operational, so techniques for distributing the existing data among new nodes without interrupting the system operation is of upmost importance

- Availability
  - ✓ NoSQL systems requires continuous system availability
  - ✓ Data is replicated over two or more nodes in transparent manner so that if one node fails, data is still available on the other nodes
  - ✓ Replication improves the data availability and performance as the reads can be answered from replicas as well
  - ✓ Write performance become cumbersome as updated must be applied to every copy

Characteristics of NoSQL systems (3)

Related to Distributed databases and distributed systems

- Replication models
- Master-slave or Master – master

- Master – slave configuration
  - ✓ One copy as master, others as slaves
  - ✓ Changes are first applied to master and then propagated to slaves, using eventual consistency
  - ✓ Reads can be done in two ways
  - ✓ Read from master – always latest data returned
  - ✓ Read from slaves – no guarantee of latest data as its based on eventual consistency

- Master-Master configuration
  - ✓ Allows read and write at any of replicas but may not guarantee that reads at nodes that store different copies see the same values

Characteristics of NoSQL systems (4)

Related to Distributed databases and distributed systems

- Sharding of files
  - ✓ Files (collections of data objects) have millions of records which are accessed simultaneously
  - ✓ Not practical to store the file at one node
  - ✓ Sharding – horizontal partitioning is used to distribute loads of accessing the files records to multiple nodes
  - ✓ Combination of sharding and replication improve load balancing and data availability

- High Performance Data Access
  - ✓ Finding a data record is important in many NoSQL systems
  - ✓ Either uses hashing or range partitioning of keys
  - ✓ In hashing, hash function h(K) is applied to key K and location of object with key K is determined by value of h(K)
  - ✓ In range partitioning, location is determined by range of key values, location i would hold objects whose key values K are in range Kimin <= K<= Kimax.

Characteristics of NoSQL systems

Related to Data models and query language

- Not requiring Schema
  - ✓ Allowed by semi structured, self describing data
  - ✓ Not required to have a schema in most of NoSQL systems
  - ✓ There may not be a schema to specify constraints, any constraints on the data would have to be programmed in applications
  - ✓ Languages like JSON, XML are used while defining models
- Low powerful query languages
  - ✓ Many applications not require powerful query languages as SQL as read queries in these systems often locate single objects in a single file based on their object keys
  - ✓ NoSQL systems provide a set of functions and APIs
  - ✓ Supports SCRUD operations – Search , CRUD
  - ✓ Many systems do not provide join operations as part of language, hence needs to be implemented in application
- Versioning

NoSQL

- Not Only SQL
- Coined by Carlo Strozzi in 1998 to name lightweight, open source, relational database that did not expose the standard SQL interface
- Johan Oskarsson , in 2009 reintroduced the term to discuss open-source distributed network

- Features
  - ✓ Open source
  - ✓ Non-relational
  - ✓ Distributed
  - ✓ Schema-less
  - ✓ Cluster friendly
  - ✓ Born out of 21st century web applications

Categories (1)

- Document based
- Key-Value stores
- Column based or wide column
- Graph based

Key-value stores	Column-Oriented	Document based	Graph based
Riak	Cassandra	MongoDB	Neo4J
Redis	Hbase	CouchDB	InfiniteGraph
Membase	HyperTable	RavenDB	AllegroGraph

Categories (2)

Document based

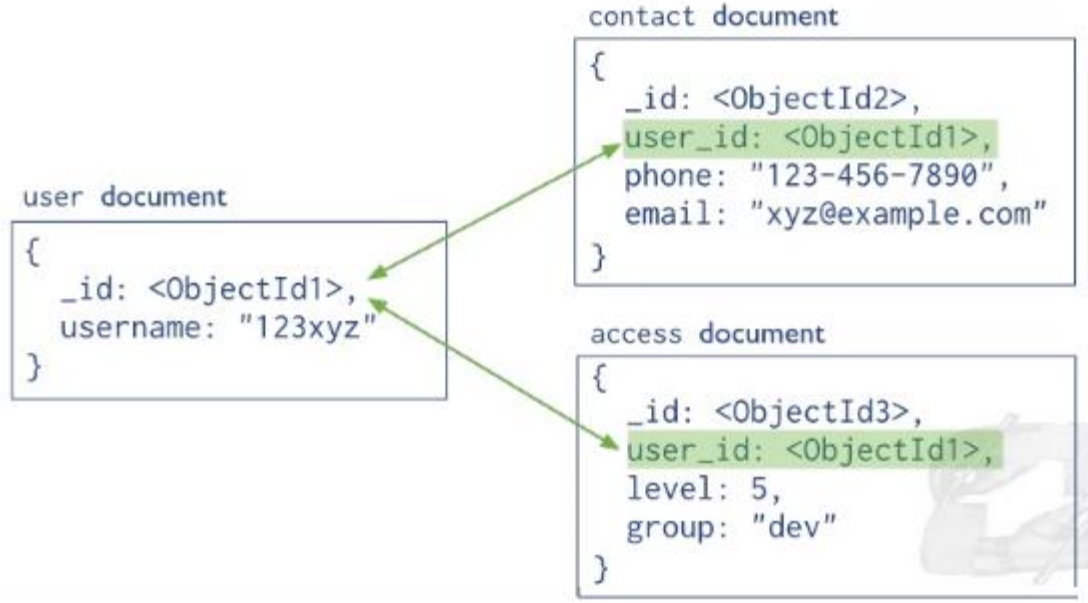
- Store data in form of documents using well known formats like JSON
- Documents accessible via their id, but can be accessed through other index as well
- Maintains data in collections of documents
- Example,
  - MongoDB, CouchDB, CouchBase

- Book document :

{

“Book Title” : “Database Fundamentals”,
“Publisher” : “My Publisher”,
“Year of Publication” : “2020”

}

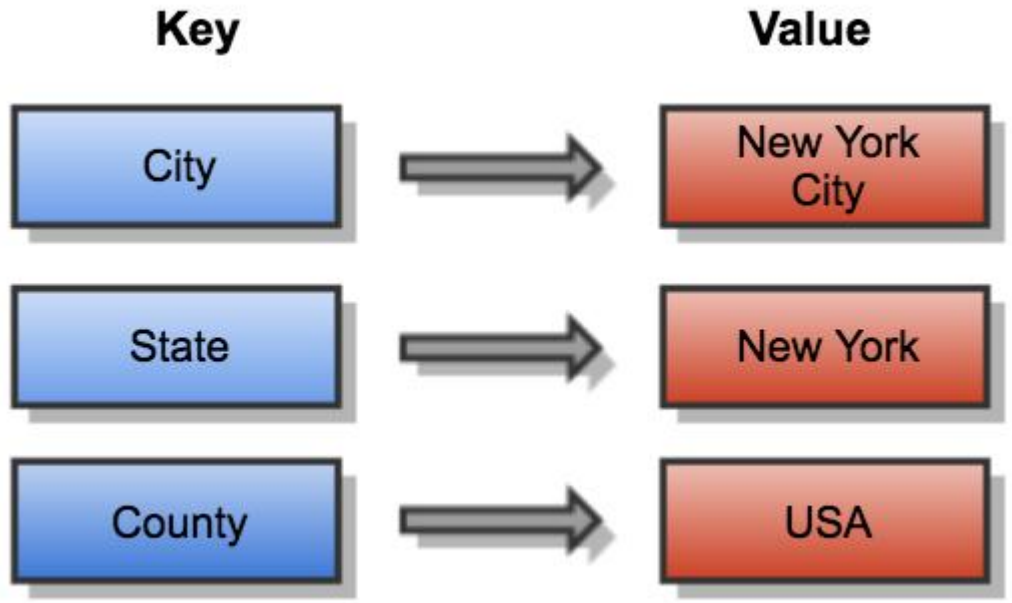


Categories (3)

Key-value stores

- Simple data model based on fast access by the key to the value associated with the key
- Value can be a record or object or document or even complex data structure
- Maintains a big hash table of keys and values
- For example,
  - ✓ Dynamo, Redis, Riak

Key	Value
2014HW112220	{ Santosh,Sharma,Pilani}
2018HW123123	{Eshwar,Pillai,Hyd}

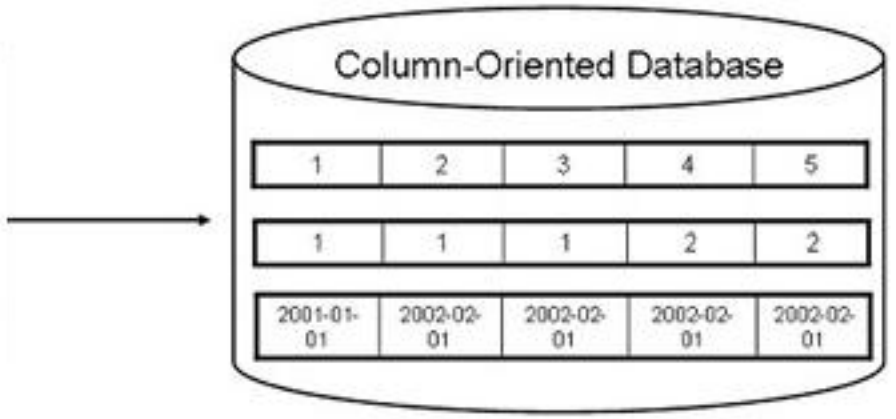


Categories (4)

Column based

- Partition a table by column into column families
- A part of vertical partitioning where each column family is stored in its own files
- Allows versioning of data values
- Each storage block has data from only one column
- Example,
  - ✓ Cassandra, Hbase

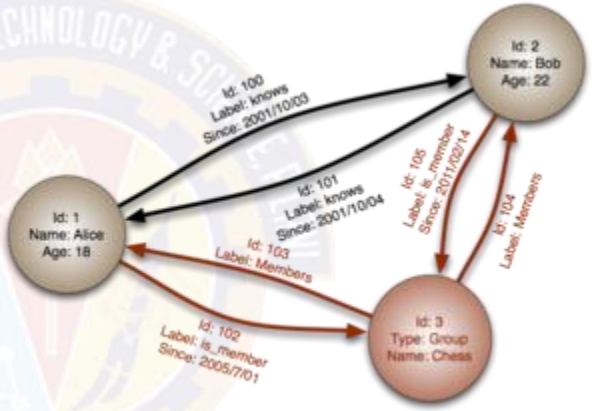
Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura



Categories (5)

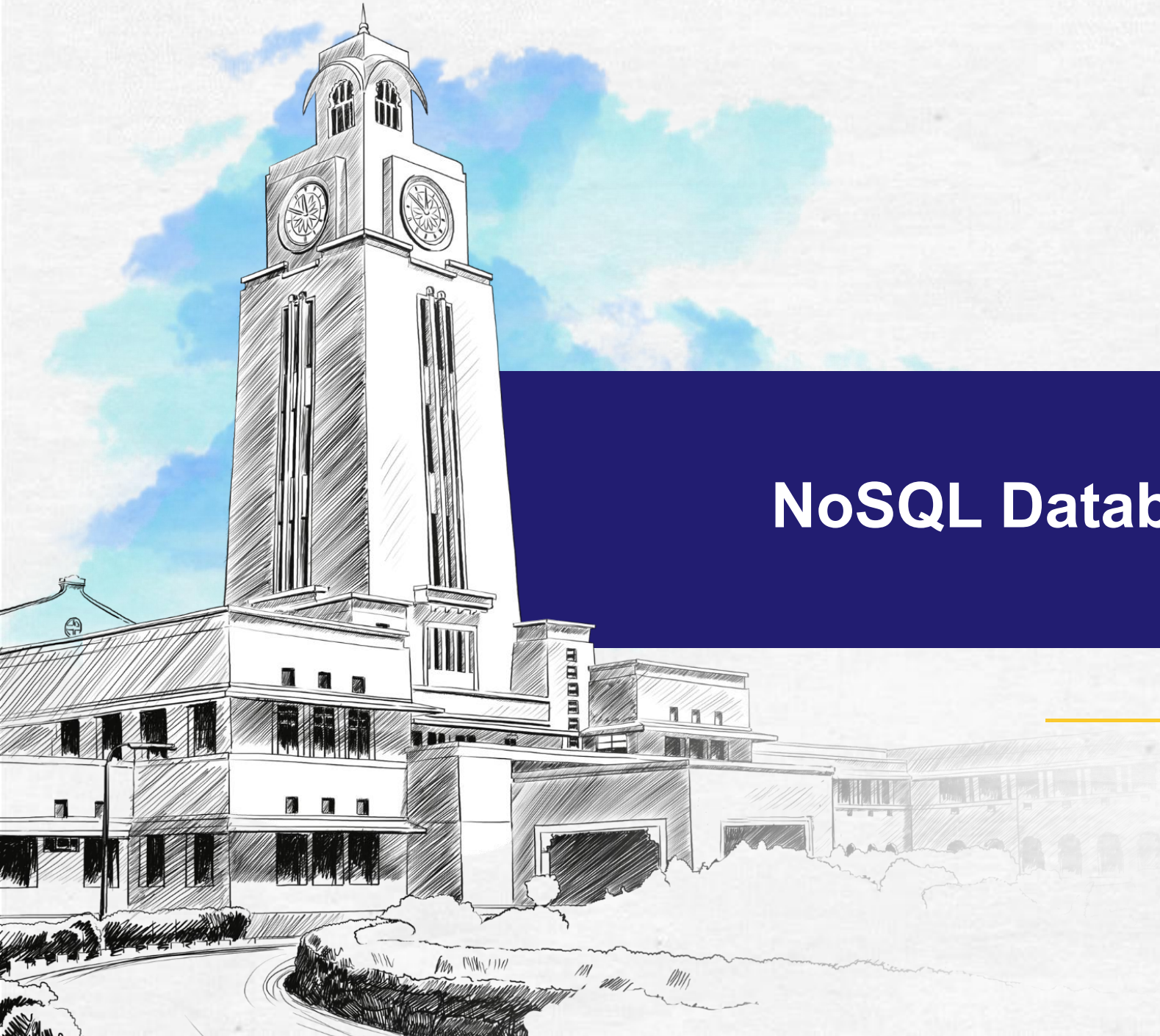
Graph Based

- Data is represented as graphs and related nodes can be found by traversing the edges using the path expression
- aka network database
- Example
  - ✓ Neo4J,HyperGraphDB



Thank You!

In our next session: Document based NoSQL Databases



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

NoSQL Databases - Characteristics

Chandan Ravandur N

NoSQL

- Not Only SQL
- Meant to convey that many applications need systems other than traditional relational SQL systems to manage their data management needs

- Most of them are distributed databases or distributed storage systems with focus on
  - ✓ Semi structured data storage
  - ✓ High performance
  - ✓ Availability
  - ✓ Data replication
  - ✓ Scalability

Image Source : DataVarsity

Emergence of NoSQL systems

Use cases

- Free email application like Gmail
  - Have millions of users, each user with thousands of emails – need for storage
  - SQL may not be appropriate because
    - ✓ It has too many services which are not required here
    - ✓ Structured data model is restrictive

- Facebook – social media platforms
  - Have millions of users, each user submits posts – images, videos, text , displayed on other users pages
  - User profiles, relationships , posts requires – storage
  - Needs to show the posts to other users – processing
  - SQL many not be appropriate because
    - ✓ Can not handle some of this data well
    - ✓ Can not handle when storage is huge and distributed across nodes

Emergence of NoSQL systems (2)

Soultions

- BigTable
  - ✓ Googles proprietary NoSQL systems used in many of Googles applications
  - ✓ dealing with vast amounts of data storage like Gmail, Maps, Web site indexing etc.
  - ✓ Apache Hbase is open source version of BigTable
  - ✓ Column-based or wide column stores

- DynamoDB
  - ✓ Amazons NoSQL system available on AWS
  - ✓ Innovation in key-value type data stores

- Cassandra
  - ✓ Facebooks NoSQL system
  - ✓ Uses concepts from both key-value stores and column-based systems

- MongoDB
  - ✓ Document based NoSQL system

- Neo4J
  - ✓ Graph based NoSQL systems

Characteristics of NoSQL systems

- Related to Distributed databases and distributed systems
  - ✓ Scalability
  - ✓ Availability
  - ✓ Replication models
  - ✓ Sharding
  - ✓ High performance

- Related to data models and query languages
  - ✓ No fixed schema
  - ✓ Less powerful query languages
  - ✓ Versioning

Characteristics of NoSQL systems (2)

Related to Distributed databases and distributed systems

- Scalability
  - ✓ Horizontal or Vertical
  - ✓ In NoSQL systems, horizontal scalability is used by adding more nodes for data storage and processing as the volume of data grows
  - ✓ Used when system is operational, so techniques for distributing the existing data among new nodes without interrupting the system operation is of upmost importance

- Availability
  - ✓ NoSQL systems requires continuous system availability
  - ✓ Data is replicated over two or more nodes in transparent manner so that if one node fails, data is still available on the other nodes
  - ✓ Replication improves the data availability and performance as the reads can be answered from replicas as well
  - ✓ Write performance become cumbersome as updated must be applied to every copy

Characteristics of NoSQL systems (3)

Related to Distributed databases and distributed systems

- Replication models
- Master-slave or Master – master

- Master – slave configuration
  - ✓ One copy as master, others as slaves
  - ✓ Changes are first applied to master and then propagated to slaves, using eventual consistency
  - ✓ Reads can be done in two ways
  - ✓ Read from master – always latest data returned
  - ✓ Read from slaves – no guarantee of latest data as its based on eventual consistency

- Master-Master configuration
  - ✓ Allows read and write at any of replicas but may not guarantee that reads at nodes that store different copies see the same values
  - ✓ Different users may write same data item concurrently at different nodes of system, so values of item will be temporarily inconsistent

Characteristics of NoSQL systems (4)

Related to Distributed databases and distributed systems

- Sharding of files
  - ✓ Files (collections of data objects) have millions of records which are accessed simultaneously
  - ✓ Not practical to store the file at one node
  - ✓ Sharding – horizontal partitioning is used to distribute loads of accessing the files records to multiple nodes
  - ✓ Combination of sharding and replication improve load balancing and data availability

- High Performance Data Access
  - ✓ Finding a data record is important in many NoSQL systems
  - ✓ Either uses hashing or range partitioning of keys
  - ✓ In hashing, hash function h(K) is applied to key K and location of object with key K is determined by value of h(K)
  - ✓ In range partitioning, location is determined by range of key values, location i would hold objects whose key values K are in range Kimin <= K<= Kimax.

Characteristics of NoSQL systems

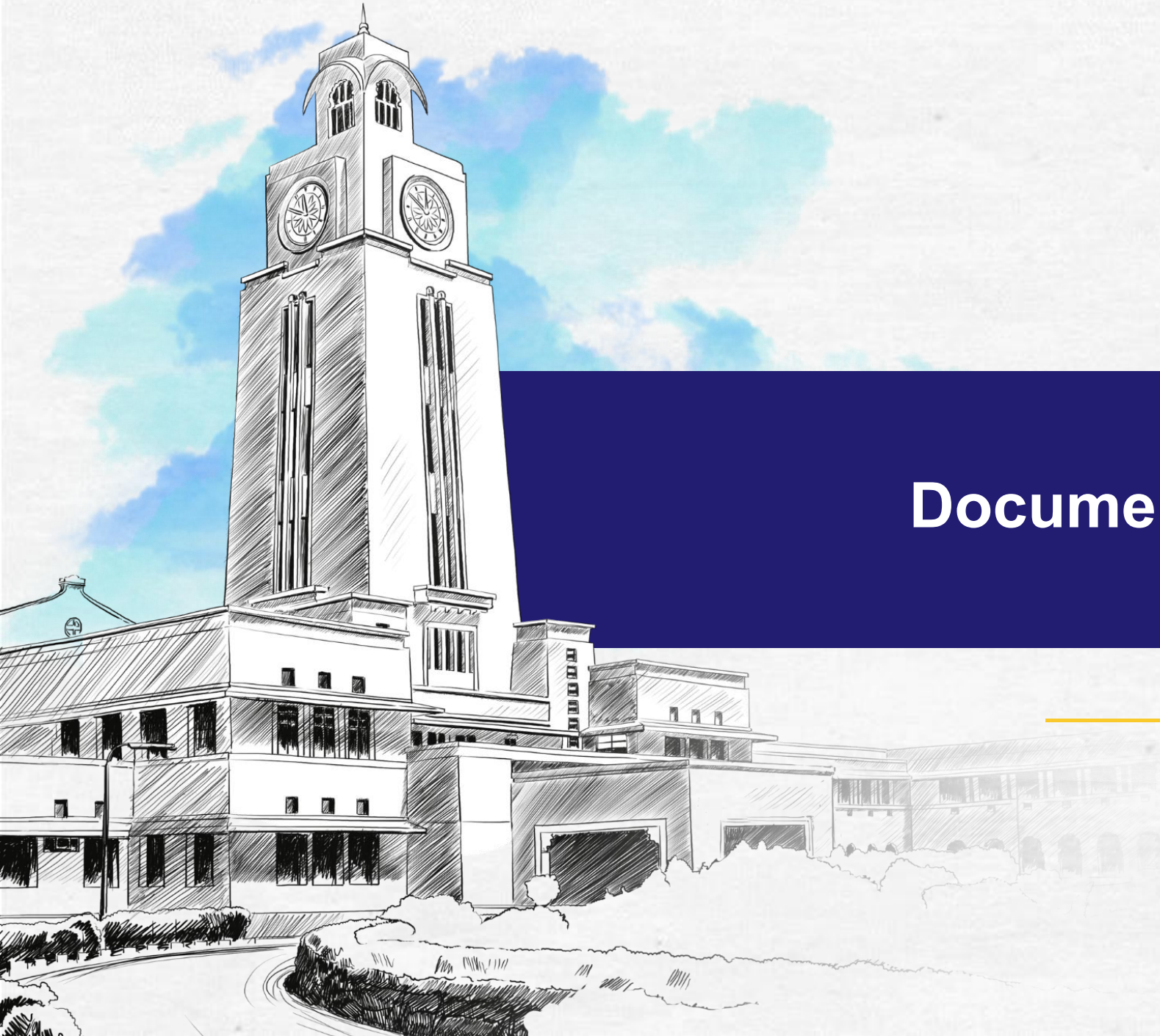
Related to Data models and query language

- Not requiring Schema
  - ✓ Allowed by semi structured, self describing data
  - ✓ Not required to have a schema in most of NoSQL systems
  - ✓ There may not be a schema to specify constraints, any constraints on the data would have to be programmed in applications
  - ✓ Languages like JSON, XML are used while defining models
- Low powerful query languages
  - ✓ May applications not require powerful query languages as SQL as read queries in these systems often locate single objects in a single file based on their object keys
  - ✓ NoSQL systems provide a set of functions and APIs
  - ✓ Supports SCRUD operations – Search , CRUD
  - ✓ Many systems do not provide join operations as part of language, hence needs to be implemented in application
- Versioning
  - ✓ Some NoSQL systems provides storage of multiple versions of data items with timestamps

Reference :
Fundamentals of Database Systems, by Elmasri, Navathe

Thank You!

In our next session: NoSQL Databases - Categories



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Document Oriented Databases

Chandan Ravandur N

Document based Databases

Document based

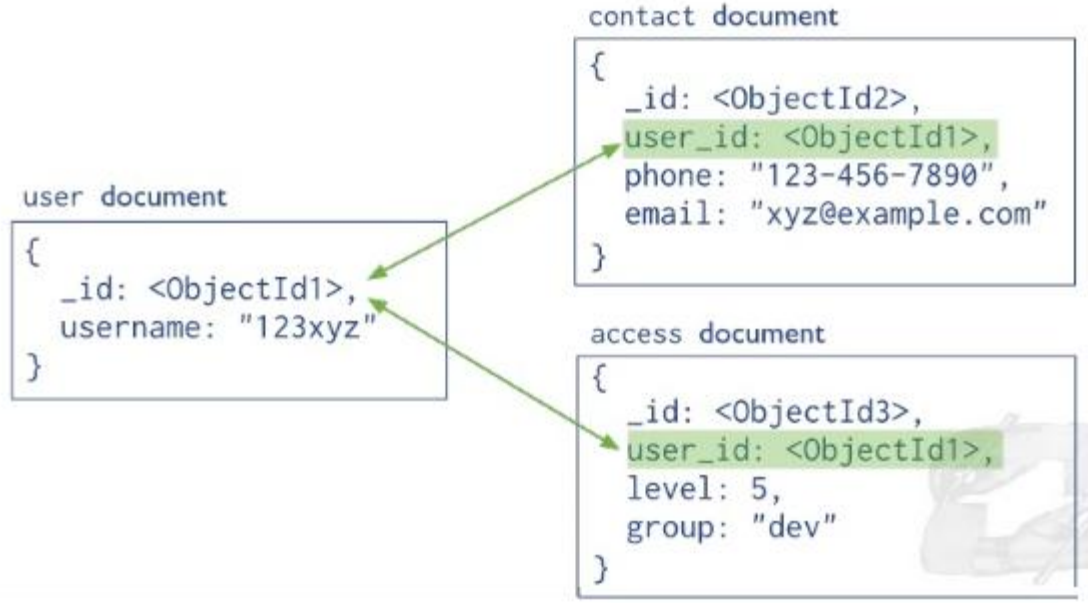
- Store data in form of documents using well known formats like JSON
- Documents accessible via their id, but can be accessed through other index as well
- Maintains data in collections of documents
- Example,
  - MongoDB, CouchDB, CouchBase

- Book document :

{

“Book Title” : “Database Fundamentals”,
“Publisher” : “My Publisher”,
“Year of Publication” : “2020”

}



What?

MongoDB is

- NoSQL
- Cross-platform
- Distributed
- Document-oriented
- Open source

Datastore

Image Source : MongoDB

Why?

Challenges with Relational Databases

- Not able to cope with high data volume
- Limited capabilities to deal with unstructured data
- Restrictions on the scalable needs of enterprise data

- **Require a Data store that supports**
  - ✓ Horizontal scaling
  - ✓ Flexible schema
  - ✓ Partitioning

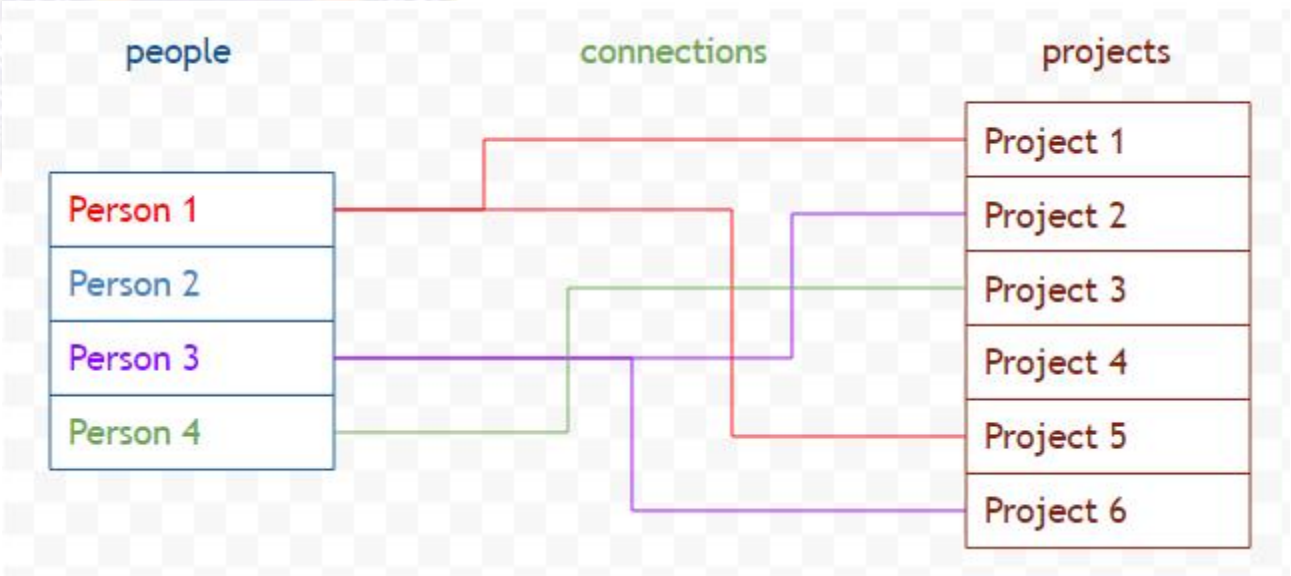
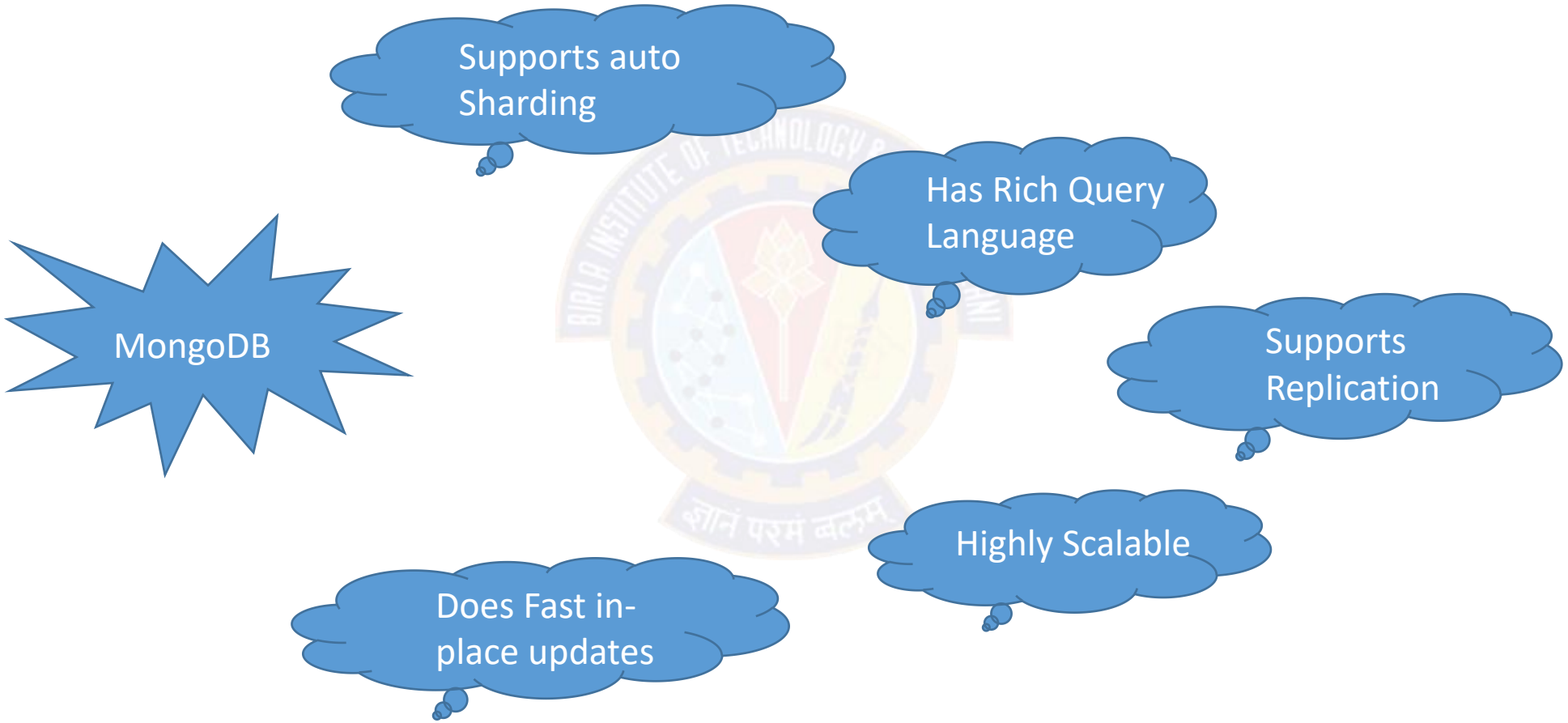


Image source : GoogleSite

Why? (2)


MongoDB
Supports auto Sharding
Has Rich Query Language
Supports Replication
Highly Scalable
Does Fast in-place updates

MongoDB and JSON

- MongoDB uses BSON (Binary JSON) – Open standard to store complex data structures

- Example
- Assume following Employee Record is stored in a file
  - ❑ 123, Rohit Kumar , 8123561290
  - ❑ 124, Naresh Pande , 8904561239

- Corresponding JSON representation

```
{
        EmpId : 123,
        EmpName : Rohit Kumar,
        ContactNo : 8123561290
}

{
        EmpId : 124,
        EmpName : Naresh Pande ,
        ContactNo : 8904561239
}
```

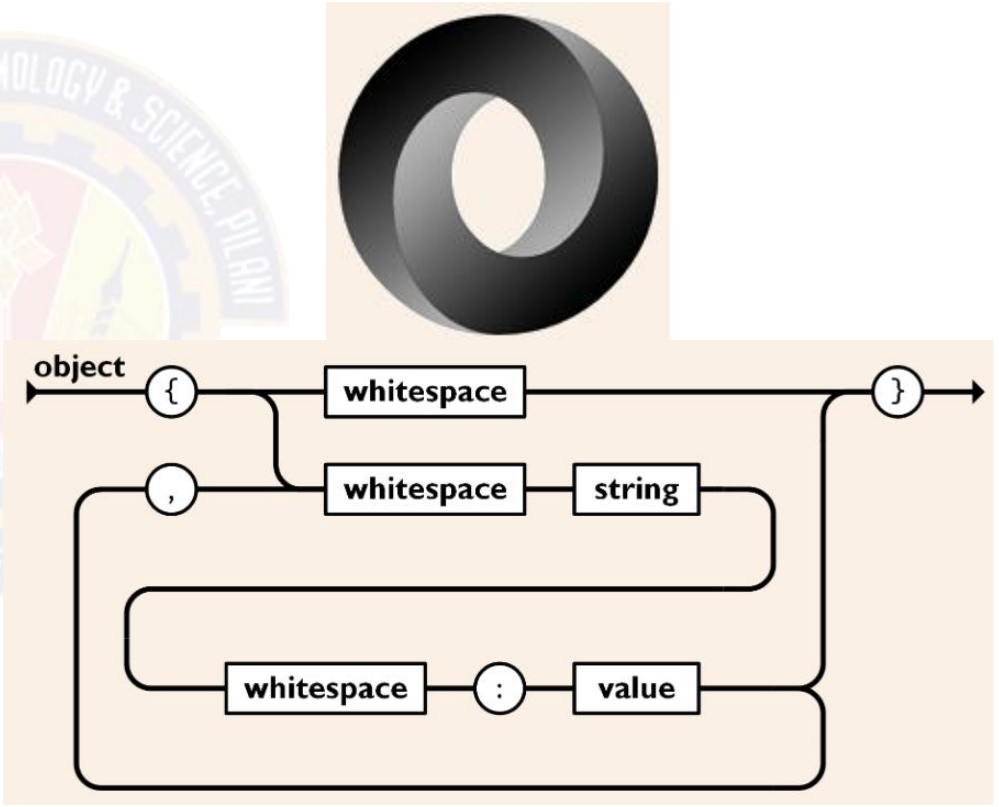


Image Source : Json.org

MongoDB concepts

Database, Collection and Document

- Database
  - ✓ Collection of collections
  - ✓ Container for collections
  - ✓ Gets created when its referenced or can be manually created on demand

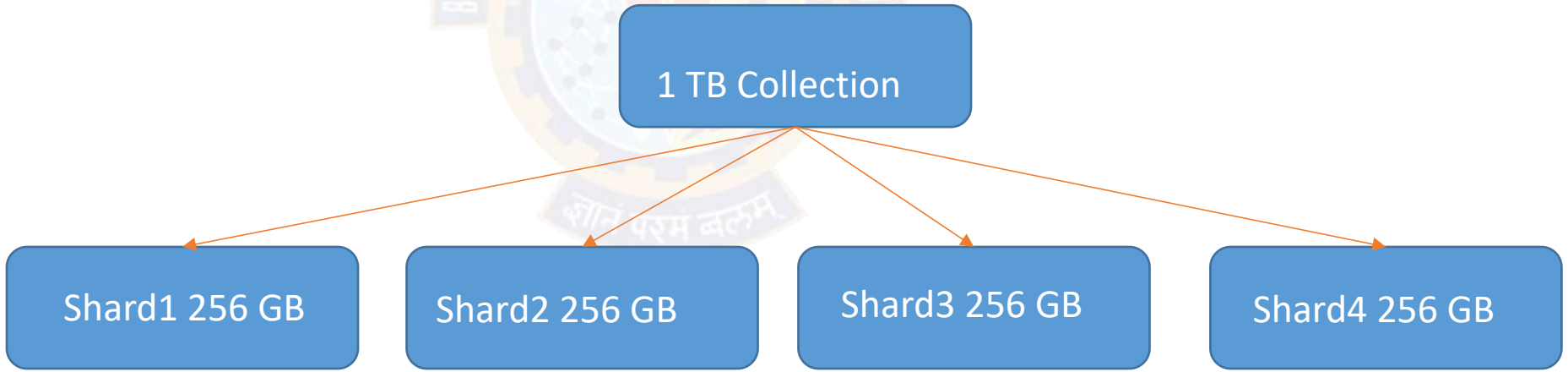
- Collection
  - ✓ Similar to table in RDBMS
  - ✓ Holds multiple documents within it
  - ✓ No enforcement of Schema
  - ✓ Gets created when its referenced or can be manually created on demand

- Document
  - ✓ Similar to row in RDBMS table
  - ✓ Has a unique identifier
  - ✓ Supports dynamic schema

MongoDB Features

Auto Sharding

- Similar to Horizontal scaling
- Dataset is divided and distributed over multiple nodes or shards
- Each shard is independent database
- Collectively all databases constitutes the logical database
- Reduces the number of operations that each shard handles



MongoDB Features(2)

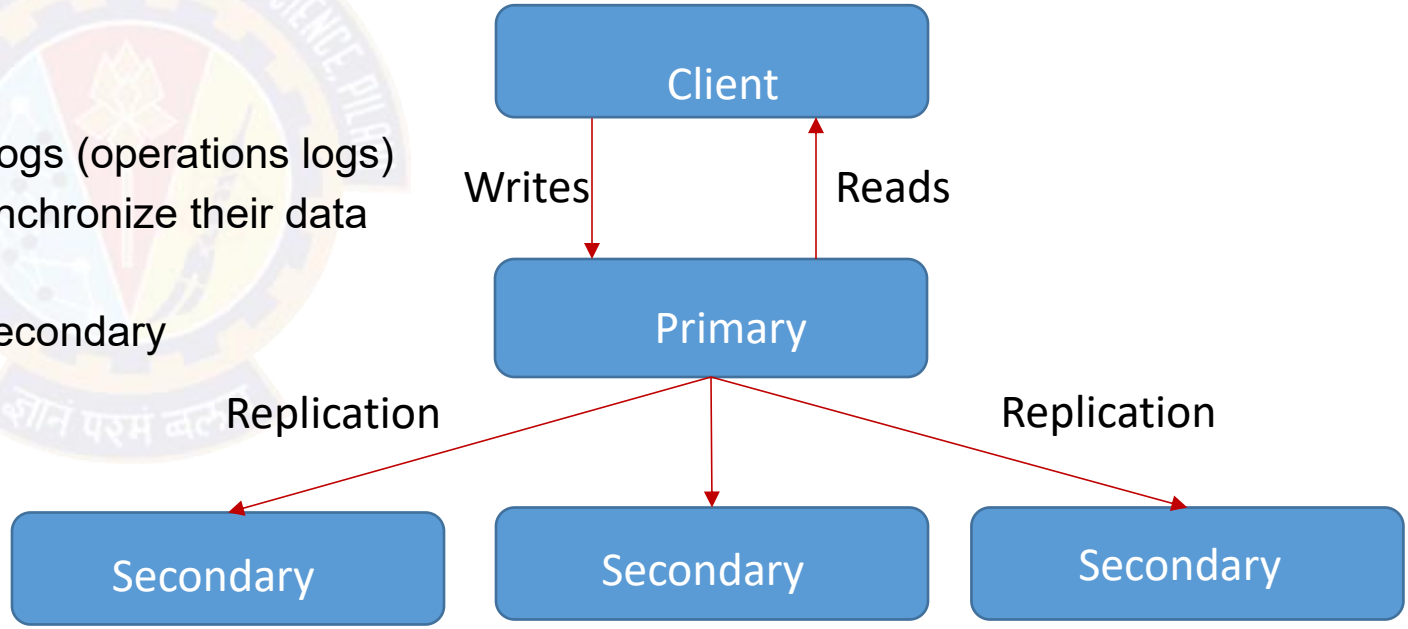
Rich Query Language

- For read and write operations (CRUD)
- Data Aggregation
- Text Search and Geospatial Queries

MongoDB Features(3)

Replication

- Supports data redundancy and high availability
- Helps to recover from hardware failure and service interruptions
- Replica has single primary and several secondary's
- Working
  - ✓ Write is directed to primary
  - ✓ Primary then logs all write requests to OpLogs (operations logs)
  - ✓ OpLog is used by secondary replicas to synchronize their data
  - ✓ Clients read from primary
  - ✓ Client can specify a read preference to a secondary



MongoDB Features(4)

Scalability

- Stores data in binary format (BSON)
- Provides GridFS to support storage of data
- Can easily store photographs, small audio clips
- Stores metadata in a collection called “file”
- Breaks the data into small pieces called “chunks”
- “Chunks” are stored in “chunks” collection
- This process takes care about need for easy scalability

MongoDB Features(5)

In-place data updates

- Updates data in-place
- Means updates the data wherever it is available
- Does it by lazy writes
- Writes to disk once every second
- Fewer reads and writes to disk hence performance is better
- No guarantee that data will be stored safely on the disk

Thank You!

In our next session: Columnar Databases

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Today's application requirements

Chandan Ravandur N

Today’s application requirements

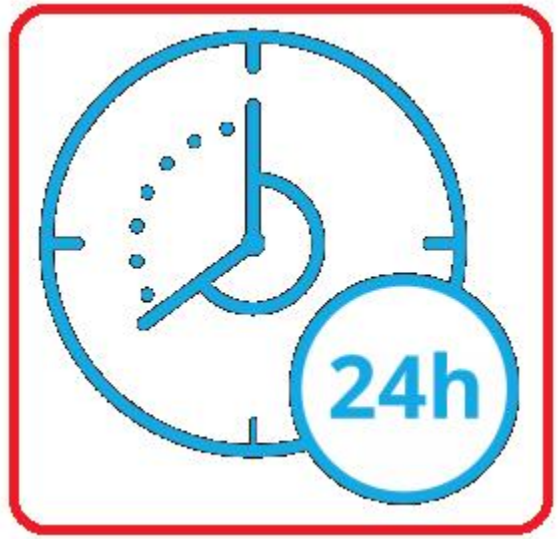
- Digital experiences play a major part in many or most of the activities that we engage in on a daily basis
  - ✓ pushed the boundaries of expectations from the software :
  - ✓ want applications to be always available
  - ✓ be perpetually upgraded with new whizzbang features
  - ✓ provide personalized experiences

- Fulfilling these expectations is something that must be addressed right from the beginning of the idea-to-production lifecycle

- Key requirements:
  - ✓ Zero downtime
  - ✓ Shortened feedback cycles
  - ✓ Mobile and multi-device support
  - ✓ Connected devices—also known as the Internet of Things
  - ✓ Data-driven

Zero downtime

- One of the key requirements of the modern application: it must always be available
  - ✓ The world is working in "now mode"
  - ✓ Forgotten are the days when even short maintenance windows during which applications are unavailable are tolerated



- Unplanned downtime has never been desirable, its impact has reached astounding levels
  - ✓ For example, long back Forbes estimated that Amazon lost almost $2 million during a 13-minute unplanned outage

- Downtime, planned or not, results in significant revenue loss and customer dissatisfaction
  - ✓ no more just a problem only for the operations team
  - ✓ Software developers or architects are responsible for developing zero downtime software's!

Shortened feedback cycles

- Another critical ability is to release code frequently
  - ✓ Required by significant competition and ever-increasing consumer expectations

- Application releases are being made available to customers
  - ✓ several times a month
  - ✓ numerous times a week
  - ✓ even several times a day

Source : facebook

- The biggest driver for these continuous releases is the reduction of risk!
  - ✓ The best way to get mitigate the risk is to release an early version of a feature and get feedback
  - ✓ Using feedback, can make adjustments or even change direction entirely
  - ✓ Frequent software releases shorten feedback loops and reduce risk!

Mobile and multi-device support

- Internet usage via mobile devices has already eclipsed that of desktop / laptop computers

- Today's applications need to support at least
  - ✓ two mobile device platforms, iOS and Android
  - ✓ as well as the desktop

- Users increasingly expect to seamlessly move from one device to another as they navigate through the day with same app experience

Source : savoirfairelinux

- For example,
  - ✓ Users may be watching a cricket match on an Android TV and then transition to viewing the program on a mobile device when they're travelling

- Designing applications the right way is essential to meeting these needs!

Connected devices

Internet of Things

Source : ThreatPost

- The internet is no longer only for connecting humans to systems
- Today, billions of devices are connected to the internet
  - ✓ Allowing devices to be monitored and even controlled by other connected entities
- The home-automation market alone is estimated to be a $53 billion market by 2022
- The connected home has sensors and remotely controlled devices such as
  - ✓ motion detectors
  - ✓ cameras
  - ✓ smart thermostats
  - ✓ lighting systems etc.
- Other connected devices are automobiles, home appliances, farming equipment, jet engines, and the smartphone
- Internet-connected devices change the nature of the software in two fundamental ways
  - ✓ the volume of data flowing over the internet is dramatically increased
  - ✓ the computing infra must be significantly different from those of the past, to capture and process this data
- Difference in data volume and infrastructure architecture necessitates new software designs and practices.

Data-driven

- Volumes of data are increasing
- Sources of data are becoming more widely distributed
  - ✓ These factors are affecting the large, centralized, shared database making them unusable
- Instead of the single, shared database, application requirements
  - ✓ call for a network of smaller, localized databases,
  - ✓ software that manages data relationships across that collection of data management systems
- New approaches drive the need for software development and management agility all the way through to the data tier.
- All of the newly available data is wasted if it goes unused
  - ✓ Today's applications must increasingly use data to provide greater value to the customer through smarter applications

Source : Dataversity

Reference:
Cloud Native Patterns by Cornelia Davis

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Modern Architectures

Chandan Ravandur N

Approach

- The cloud is changing how applications are designed and secured
  - Instead of monoliths, applications are decomposed into smaller, decentralized services
  - These services communicate through APIs or by using asynchronous messaging or eventing
  - Applications scale horizontally, adding new instances as demand requires

- Requires a structured approach for designing applications that are
  - Scalable
  - Secure
  - Resilient
  - and highly available

- These trends bring new challenges!

source : softeng

New challenges

- Application state is distributed
- Operations are done in parallel and asynchronously
- Applications must be resilient when failures occur
- Malicious actors continuously target applications
- Deployments must be automated and predictable
- Monitoring and telemetry are critical for gaining insight into the system

Source : prismetric

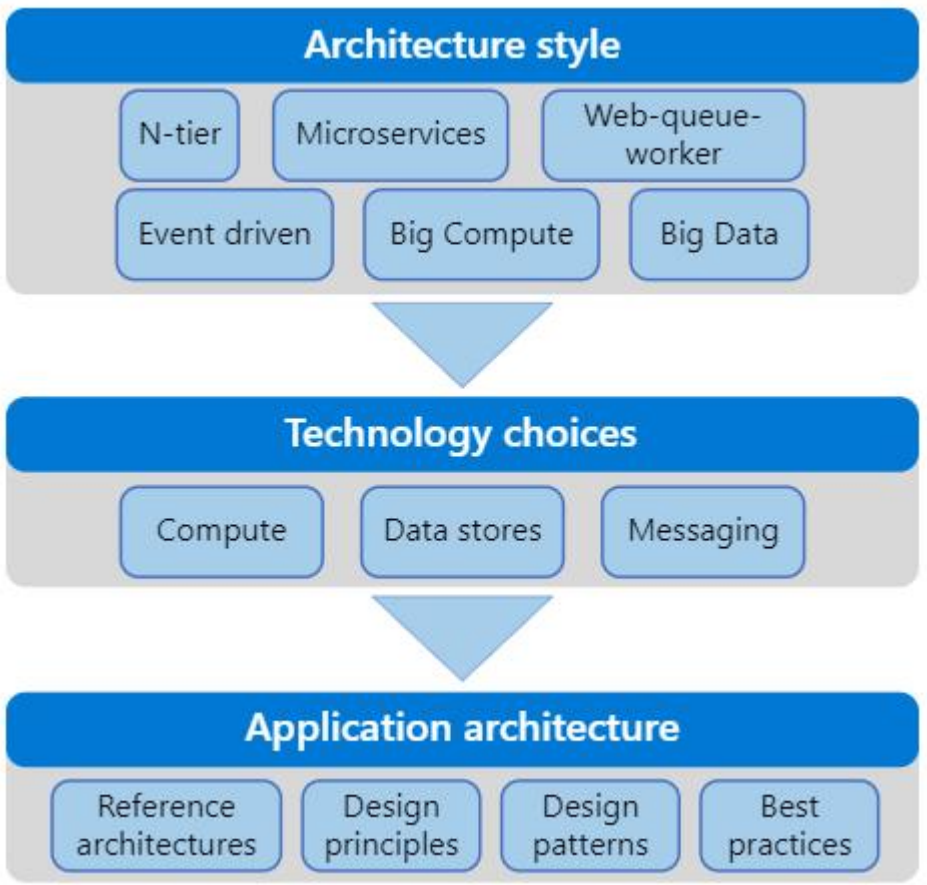
Compared

Traditional on-premises	Modern cloud
Monolithic	Decomposed
Designed for predictable scalability	Designed for elastic scale
Relational database	Polyglot persistence (mix of storage technologies)
Synchronized processing	Asynchronous processing
Design to avoid failures (MTBF)	Design for failure (MTTR)
Occasional large updates	Frequent small updates
Manual management	Automated self-management
Snowflake servers	Immutable infrastructure

source : Microsoft

Structured Approach Needed

- Series of steps, from the architecture and design to implementation are involved
- For each step, there is supporting guidance needed to design application architecture

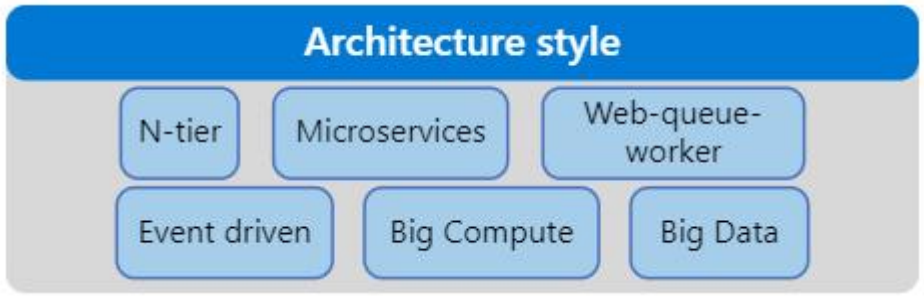


source : Microsoft

Architecture styles

- The first decision point is the most fundamental

- What kind of architecture are you building?
- It might be
  - ✓ a microservices architecture,
  - ✓ a more traditional N-tier application,
  - ✓ or a big data solution



- There are benefits and challenges to each.

Technology choices

- Now you can start to choose the main technology pieces for the architecture

- Critical Technology choices are:
  - ✓ Compute
  - ✓ Data stores
  - ✓ Messaging

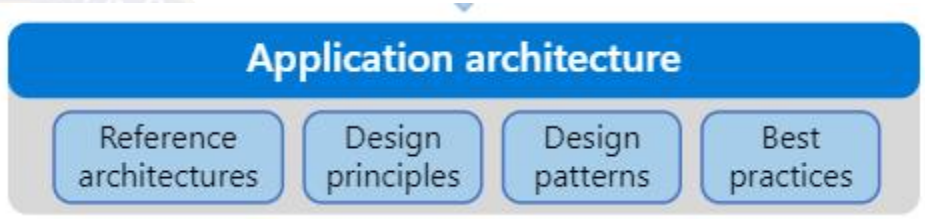
- Compute refers to the hosting model for the computing resources that your applications run on
- Data stores include databases but also storage for message queues, caches, logs, and anything else that an application might persist to storage
- Messaging technologies enable asynchronous messages between components of the system

- You will probably have to make additional technology choices along the way, but these three elements (compute, data, and messaging) are central to most cloud applications and will determine many aspects of your design.

Application Architecture

- Ready to tackle the specific design of your application

- Every application is different, but the following resources can help you along the way:
  - ✓ Reference architectures
  - ✓ Design principles
  - ✓ Design patterns
  - ✓ Best practices



- Reference architectures
  - ✓ Depending on your scenario, one of our reference architectures may be a good starting point
  - ✓ Each reference architecture includes recommended practices, along with considerations for scalability, availability, security, resilience, and other aspects of the design
  - ✓ Most also include a deployable solution or reference implementation.

- Design patterns
  - ✓ Software design patterns are repeatable patterns that are proven to solve specific problems
  - ✓ They address aspects such as availability, high availability, operational excellence, resiliency, performance, and security

Reference:
Azure Application Architecture Guide

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Architectural Styles Overview

Chandan Ravandur N

An architecture style

- A family of architectures that share certain characteristics

- For example,
  - ✓ N-tier is a common architecture style.
  - ✓ More recently, microservice architectures have started to gain favor

- Architecture styles don't require the use of particular technologies, but some technologies are well-suited for certain architectures
  - ✓ For example, containers are a natural fit for microservices

source : pintrest

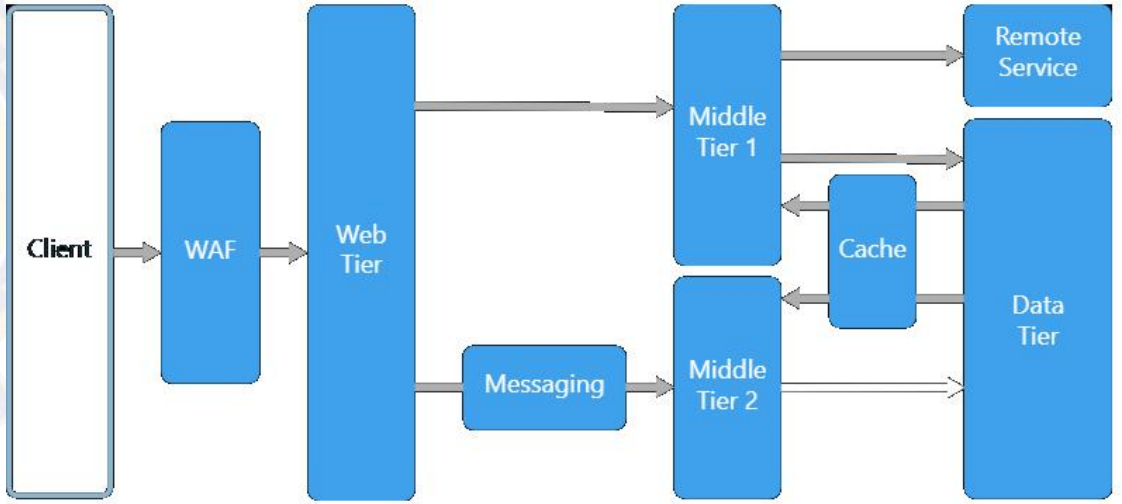
A quick tour of the styles

- A set of architecture styles that are commonly found in cloud applications

- N-tier
- Web-Queue-Worker
- Microservices
- Event-driven architecture
- Big Data, Big Compute

N-tier

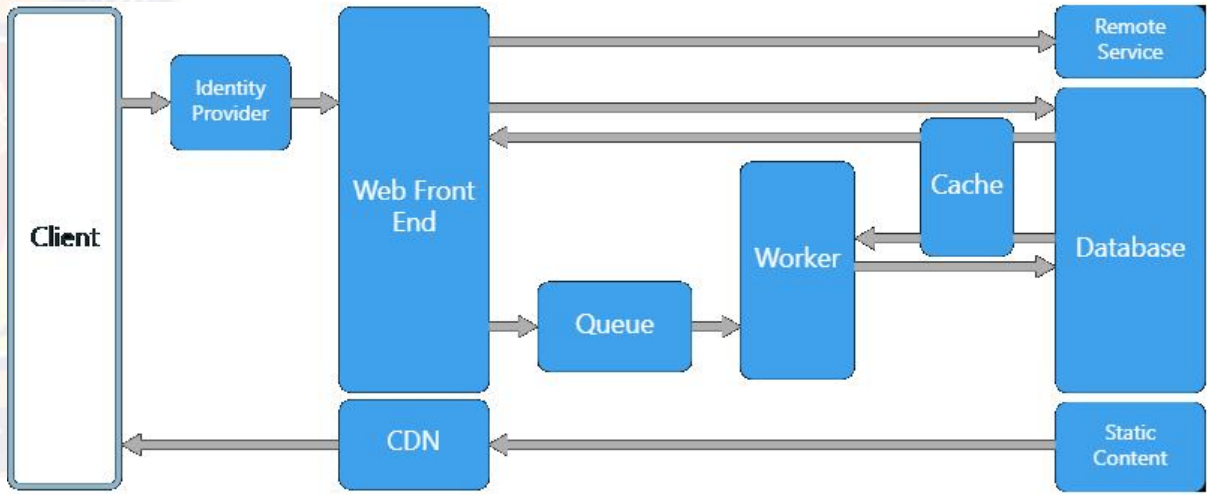
- A traditional architecture for enterprise applications
- Dependencies are managed by dividing the application into layers that perform logical functions, such as
  - ✓ presentation,
  - ✓ business logic,
  - ✓ and data access
- A layer can only call into layers that sit below it
  - ✓ this horizontal layering can be a liability
  - ✓ can be hard to introduce changes in one part of the application without touching the rest of the application
  - ✓ makes frequent updates a challenge, limiting how quickly new features can be added
- Is a natural fit for migrating existing applications that already use a layered architecture
  - ✓ For that reason, often seen in
  - ✓ infrastructure as a service (IaaS) solutions,
  - ✓ or application that use a mix of IaaS and managed services



source : Microsoft

Web-Queue-Worker

- For a purely PaaS solution, consider a Web-Queue-Worker architecture
- Application has
  - ✓ a web front end that handles HTTP requests
  - ✓ a back-end worker that performs CPU-intensive tasks or long-running operations
  - ✓ front end communicates to the worker through an asynchronous message queue
- Suitable for relatively simple domains with some resource-intensive tasks
  - ✓ easy to understand
  - ✓ use of managed services simplifies deployment and operations
- But with complex domains, it can be hard to manage dependencies
  - ✓ front end and the worker can easily become large, monolithic components that are hard to maintain and update



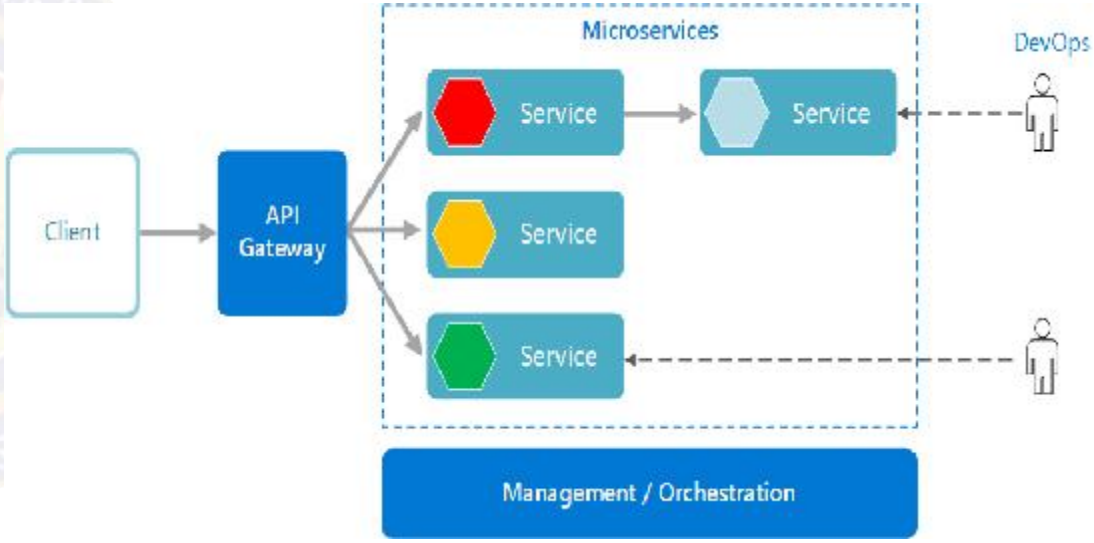
source : Microsoft

Microservices

- If your application has a more complex domain, consider moving to a Microservices architecture

- Microservice Application
  - ✓ is composed of many small, independent services
  - ✓ each service implements a single business capability
  - ✓ are loosely coupled, communicating through API contracts
  - ✓ can be built by a small, focused development team

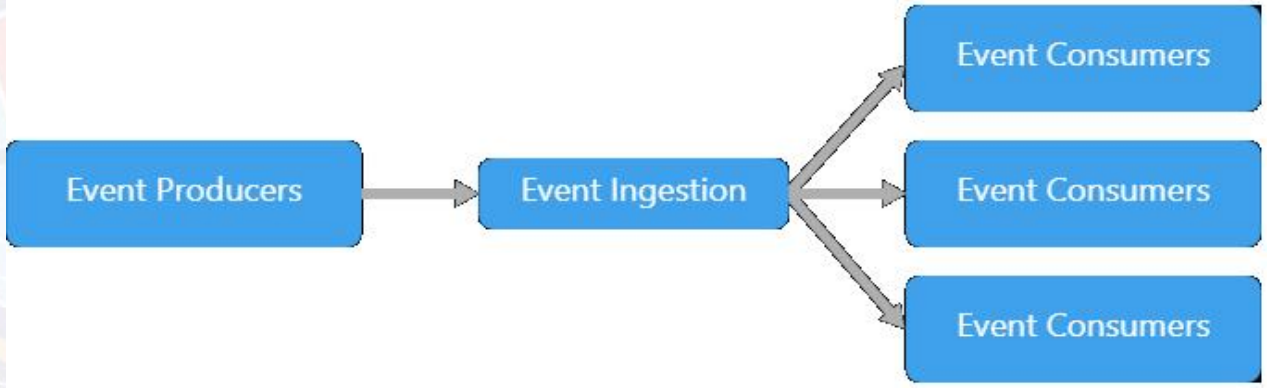
- More complex to build and manage than either N-tier or web-queue-worker
  - ✓ requires a mature development and DevOps culture
  - ✓ can lead to higher release velocity, faster innovation, and a more resilient architecture



source : Microsoft

Event-driven architecture

- Use a publish-subscribe (pub-sub) model, where producers publish events, and consumers subscribe to them
- Producers are independent from the consumers, and consumers are independent from each other

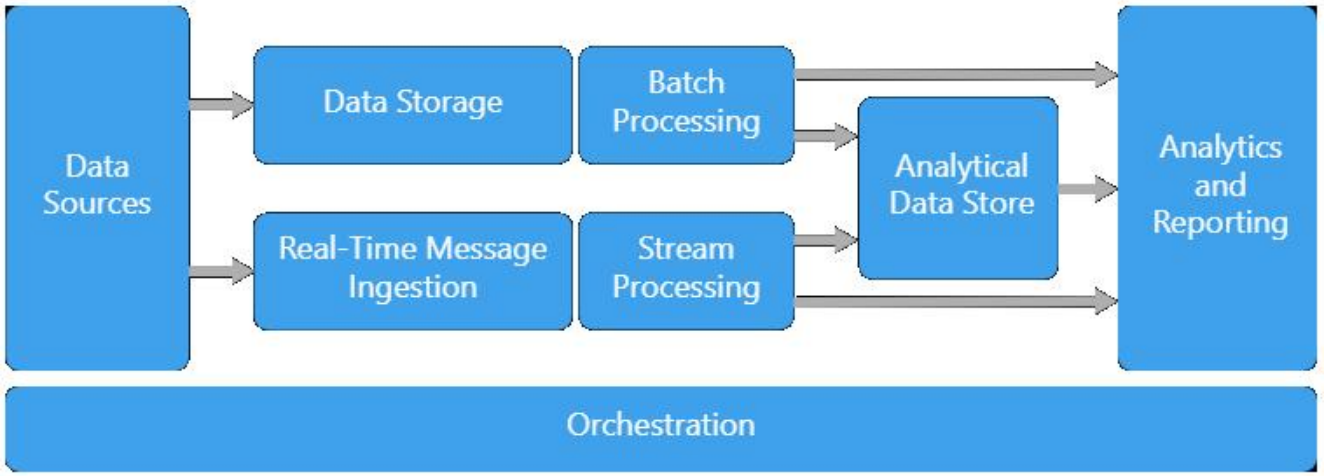


source : Microsoft

- Useful for applications that
  - ✓ ingest and process a large volume of data with very low latency, such as IoT solutions
  - ✓ has different subsystems must perform different types of processing on the same event data

Big Data, Big Compute

- Specialized architecture styles for workloads that fit certain specific profiles
- Big data divides a very large dataset into chunks, performing parallel processing across the entire set, for analysis and reporting
- Big compute, also called high-performance computing (HPC), makes parallel computations across a large number (thousands) of cores
- Domains include simulations, modeling, and 3-D rendering.



source : Microsoft

Reference:
Azure Application Architecture Guide

Thank You!

In our next session:

Slides contribution from prof Pravin Y Pawar