CM2040: Databases, Networks and the Web

Summary

Arjun Muralidharan

21st April 2020

Contents

List Of Figures

List Of Tables

List Of Algorithms

1 Three-Tier Web Applications

Key Concepts

✓ Intro to module and lab

✓ Static vs. dynamic web applications

✓ 3-tier web applications architecture

Learning Outcomes

✓ Recognize the tools available for this module to edit a Node file and run it

✓ Describe what static and dynamic web applications are

✓ Describe what a 3-tier web application architecture is

1.1 Static Vs. Dynamic Web Applications

A **web application** is a client-server software application in which the user interface runs in a web browser. It could be a computer program which allows a user to submit and retrieve data to or from a database over the Internet using their preferred browser. It is an application in which all or some parts of the software are downloaded from the web each time it is run.

Static web applications are static internet resources with little to no interaction with the user. **Dynamic web applications** have more user interactions, and users can input, change and manipulate data.

Differences between web applications and desktop applications are shown in Table 1.

Desktop	**Web**
Accessed through OS	Accessed through browser
Different appearance/experience in each OS	Consistent appearance/experience across platforms
Access to system resources; fast	Less access to system resources; slow
Lower risk of data loss	Higher risk of data loss
Different version for each OS	Same version across all platforms
Multiple updates required	Single update for all users

Table 1. *Desktop vs. Web Applications*

When a (static) web application is accessed, the following steps occur in order.

1. A **URL** is typed into the browser

2. Browser cache is checked; if the application is already available, the procedure skips to the last step
3. **DNS** lookup finds the **IP** address of the server
4. Browser initiates a **TCP** connection with the server
5. Browser sends an **HTTP** request to the server
6. Server handles the incoming request
7. Browser receives the **HTTP** response
8. Browser displays the **HTML** content

HTTP is a *client-server application protocol.* The client sends a request and a server provides a response.

HTTP requests have a fixed format, which consists of three parts: a **request line** , some optional message **headers** , and the **request body** (also optional). The request line is a formatted string, which consists of three parts; the HTTP method (e.g. `GET` or `POST`), the URL of the requested resource, and the protocol version. For example, the following request line:

```
GET /unbound/flashbks/computer/bushf.htm HTTP/1.1
```

A server responds with an HTTP response message, which is structured in three parts: a **status line** , a set of optional **headers** , and a **message body**.

The status line consists of the protocol version followed by a numeric status code and its associated message, e.g.

```
HTTP/1.1 404 Not found
```

There are four kinds of **headers**:

1. **General headers**: These apply to both the request and response, e.g. Date
2. **Request headers**: Specific to requests, e.g. Accept-Language, which sends which response languages are acceptable
3. **Response headers**: Specific to responses, e.g. WWW-Authenticate which returns if a user is authorized or not
4. **Entity headers**: Apply to the content of the request or response body, e.g. Content-Length specifies the length (in bytes) of the body

HTTP is **stateless**. Each call is treated independently and there is no information maintained between two successive requests. HTTP cannot maintain a user session.

HTTP is **pull-based**. Only the client can initiate a request-response action, the server cannot contact the client.

However, these notions are valid as of HTTP1.1 and may not be valid in HTTP/2 and newer versions.

1.2 Three-Tier Application Architecture

The core principle of **three-tier architectures** is to provide an intermediate layer between the client and the data tier, which centralizes middleware services and the business logic of the application. Three-tier architectures offer a higher degree of scalability than two-tier con- figurations, thanks to better network utilization and to the virtually unlimited replication and load distribution capabilities of the middle tier. A comparison is shown in Figure 1.

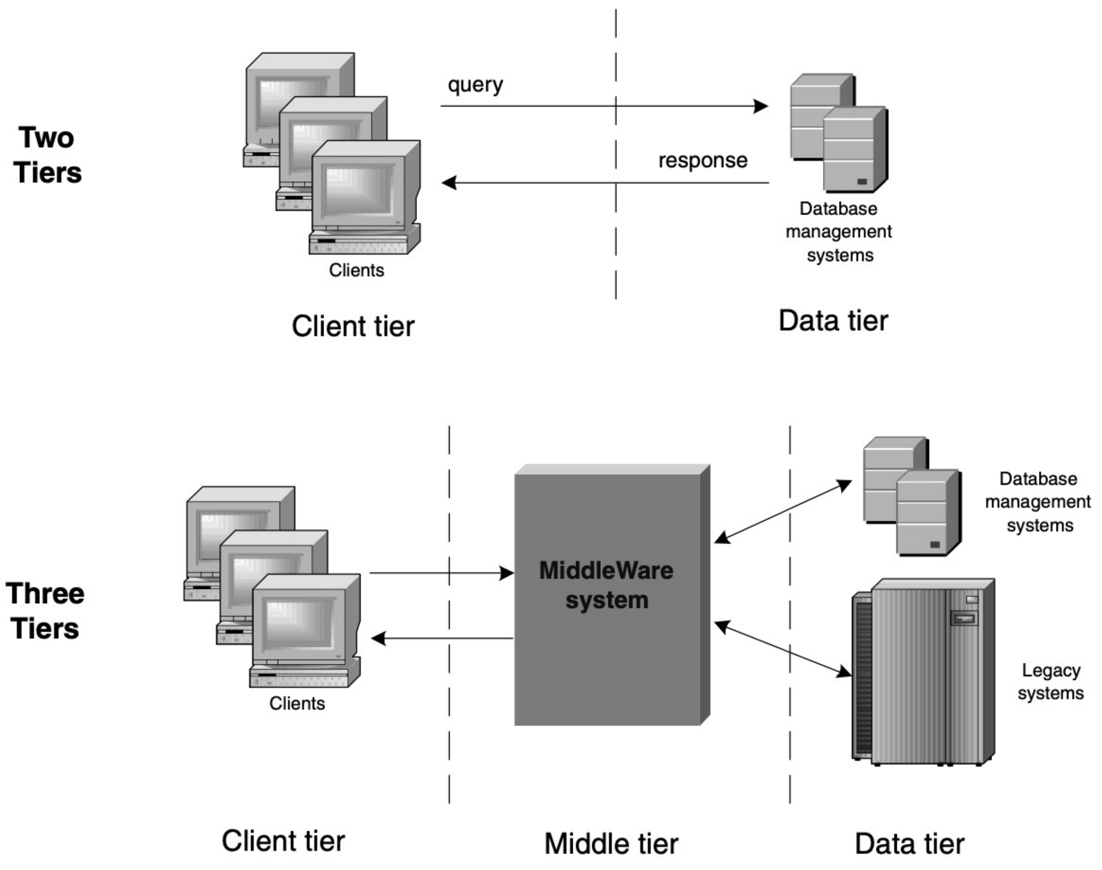

Figure 1. *Two-tier vs. three-tier architecture*

2 Building Simple Web Servers

Key Concepts

✓ Web server and web hosting

✓ How to build a simple web server with Node

✓ Building an Express web server

Learning Outcomes

- ✓ Explain what a web server is
- ✓ How to build a simple web server with Node.js
- ✓ Building an Express web server

3 Generating Web Pages From Data Using Templates

Key Concepts

- ✓ Routing in dynamic web applications
- ✓ Separation of Concern (SoC)
- ✓ Rendering html files and Templating

Learning Outcomes

- ✓ Describe and apply routing in web server development
- ✓ Describe and apply Separation of Concern principle of programming in your web application development
- ✓ Describe and apply templating in web application

4 Handling Forms To Input Data

Key Concepts

- ✓ Form handling and HTTP GET and POST methods
- ✓ HTTP GET method, form handling and collecting form-data
- ✓ HTTP POST method, form handling and collecting form-data

Learning Outcomes

- ✓ Describe and apply form handling in their web application
- ✓ Describe and apply GET and POST request methods in dynamic web application
- ✓ Describe and run the code to retrieve form data in middleware

5 Representing Data In Databases, Relational Databases

Key Concepts

- ✓ Introduction to databases
- ✓ Relational databases
- ✓ MySQL shell

Learning Outcomes

- ✓ Describe what a list of data is
- ✓ Describe modification problems related to lists of data
- ✓ Describe what a relational database is

6 Basic Database Operations, Providing Access To Databases From Middleware

Key Concepts

- ✓ Basic SQL, review create, select and insert into and introduce update and delete
- ✓ Primary keys
- ✓ Access to database from middleware

Learning Outcomes

- ✓ Describe and apply statements in SQL to do basic database operations.
- ✓ Describe and apply primary key and foreign keys in relational databases
- ✓ Recognize and apply access to databases from middleware code in Node to build dynamic web applications

7 Building A Dynamic Web Application

Key Concepts

- ✓ Passing variables from middleware to front- end (templates)
- ✓ Passing variables from middleware to backend (database)

- ✓ Review what you have learned so far for development of a dynamic web application

Learning Outcomes

- ✓ Create a complete dynamic web application performing basic database operations
- ✓ Describe and apply passing variables from middleware to front-end (templates)
- ✓ Describe and apply passing variables from middleware to backend (database)

8 Database Schema, ERD

Key Concepts

- ✓ Database schema
- ✓ Junction (bridge) tables
- ✓ SQL Join

Learning Outcomes

- ✓ Explain the context behind the relational model and describe the process of schema development
- ✓ Explain the context behind the junction (bridge) tables in relational model and describe the process of schema development
- ✓ Explain and apply foreign keys in relational databases
- ✓ Perform queries on databases using multiple related tables by JOIN queries

9 Querying A Database (Advanced)

Key Concepts

- ✓ Aggregate functions in SQL
- ✓ Left and right joins in SQL
- ✓ Nested select in SQL

Learning Outcomes

- ✓ Perform advanced queries on databases using aggregate functions

- ✓ Perform advanced queries on databases using multiple related tables by LEFT and RIGHT JOIN queries
- ✓ Perform advanced queries on databases using multiple related tables by nested SELECT queries

10 Networking Concepts

Key Concepts

- ✓ Basic concepts of computer networking
- ✓ TCP/IP model and network protocol

Learning Outcomes

- ✓ Explain and apply the basic concepts of computer networking
- ✓ Describe TCP/IP model and layers in the model
- ✓ Identify network protocols in each layer