THE UNIVERSITY OF MELBOURNE

Applications. Web Applications and Databases

Database Systems & Information Modelling
INFO90002

Week 11 – Web Applications
Dr Tanya Linden
Dr Greg Wadley
David Eccles

This lecture discusses

How end-users access databases

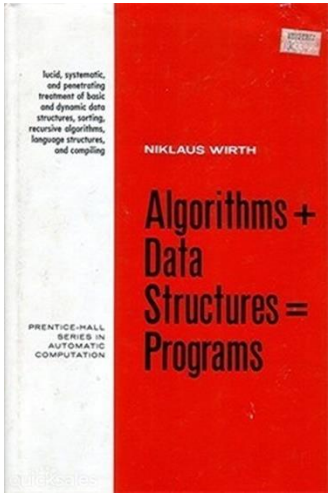
Business logic

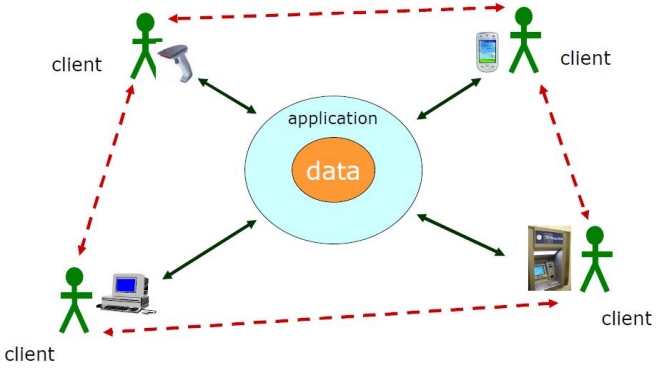
Embedding databases inside applications

Application architectures

Web applications

- How web apps work
- Making an HTML document
- Connecting to the DB
- Web services

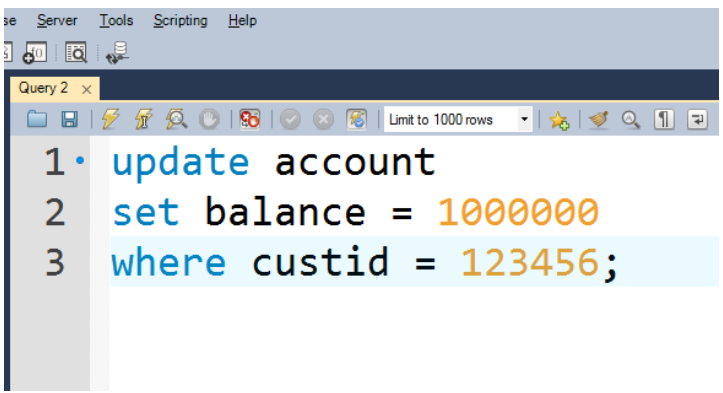


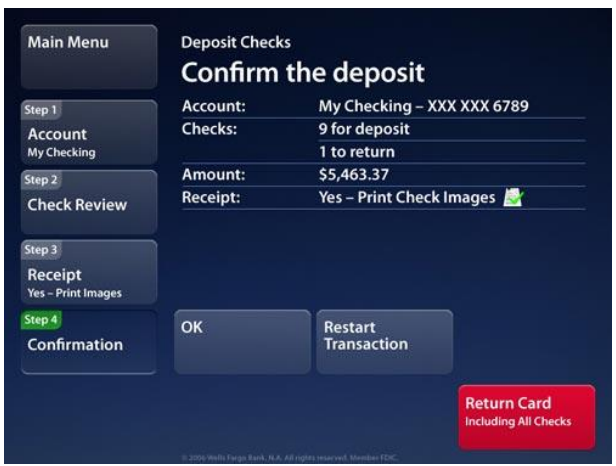


Limitations of SQL (they're not user friendly)

SQL is declarative, intuitive, versatile, but …

- cannot express all possible queries in SQL
- need to enforce business rules beyond domain/referential integrity
- need procedural constructs such as loops and decisions
- would you give end-users a query browser? Why not?
- need a user interface that is both friendly and constraining





How to handle business logic?

Examples of business logic:

- Check name and password. If good, login, if bad, error message
- Insert one row in *Order* table, then several rows in *OrderItem* table
- Check amount < balance. If so, subtract amount from one row in bank account table, then add amount to another row in another table
- For all rows in Customer table, send out monthly statements

Procedural programming languages can do:

- Sequence (several steps performed in order)
- Iteration (loops)
- Control flow (conditions, decisions)
- User interface (accept input and present output for users)

SQL is specialised for low-level data access

→ how process should work for a user who needs to do something. Need to go through the business process. the examples of business logic, which are better done by the application sitting on top of the database.

three main: sequence, selection, iteration. Iteration doesn't exist in the SQL. And also user interface, you can't make it user friendly by using SQL only, you really need some programming language on top of it to show some statement.

Example business logic

Customer places an order

- Accept inputs from user (e.g. via web form)
- Insert row into Order table
- Repeat for each product ordered:
  - Check Product table shows sufficient quantity in stock. If so:

    Insert one row into OrderItem table

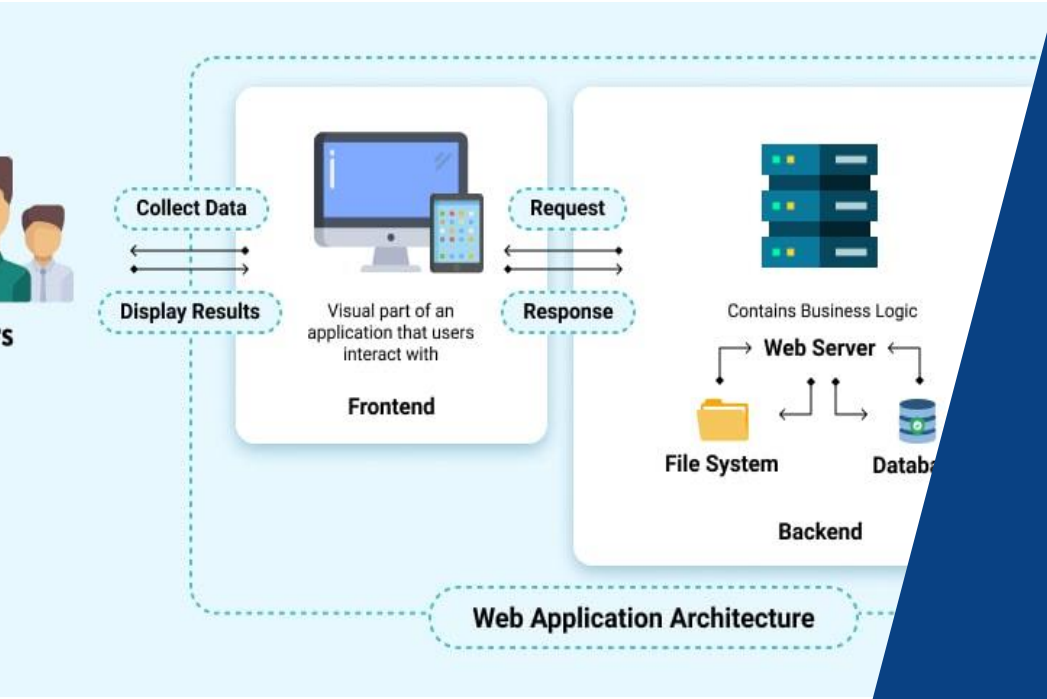
    Change Product table in-stock, Customer table amount-owing
- If no errors encountered, end successfully

Customer moves money from savings to credit card account

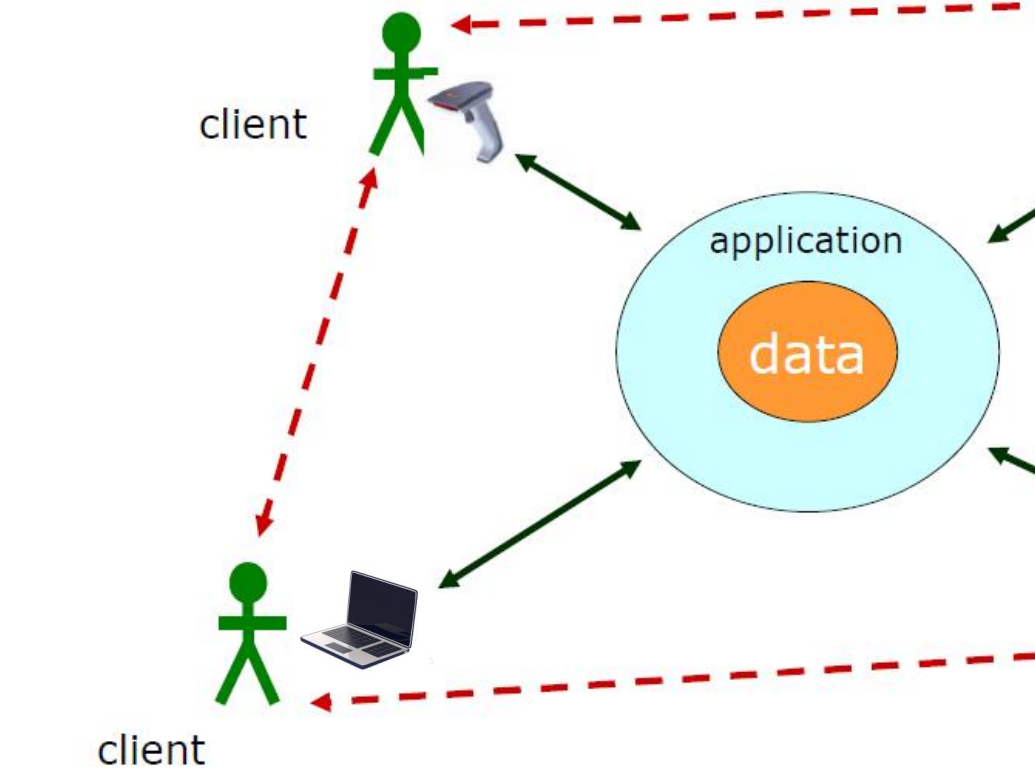
- Accept inputs from user (via ATM, internet banking or mobile app)
- Select balance from savings account
- Is there enough money to withdraw? If so:
  - Update savings account balance = balance – withdrawal
  - Update credit card balance = balance + withdrawal
- If no errors encountered, end successfully

Three user interface are different, but logic behind the scenes will be exactly the same reusable.
→ This tell us that we should keep user interface seperate from business logic.

Application Architectures



System Architecture



Regardless of what device we are using, we are getting to the application with the business logic which is going and accessing data and returning results to us.

So, in the middle data is the same. regardless of the device application with the business logic.

But based on the device, what is doing the device specific app will be different.

System Architecture

So we have basically different tiers. Different logic split into tiers.

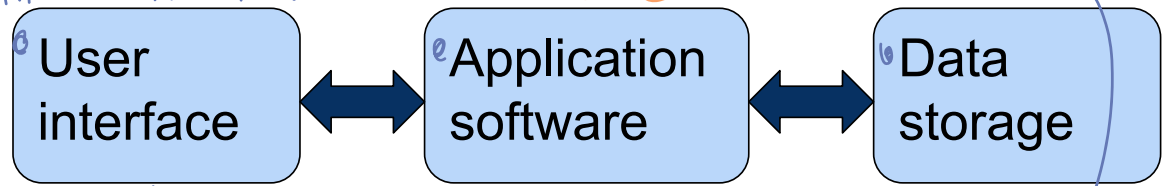
An information system must provide

- Presentation logic
  - input (keyboard, touchscreen, voice, sensor, barcode reader, etc.)
  - output (large screen, printer, phone, ATM, etc.)
- Business logic
  - input and command handling
  - enforcement of business rules
- Storage logic
  - persistent storage of data
  - enforcement of data integrity

⇒ We have presentation logic: output: is kind of screen.; input: touch screen, normal keyboard....

Business logic: is processing that input enforcing business rules. and generate the output based on the type of device

Three different tiers: These are logical tiers, because often



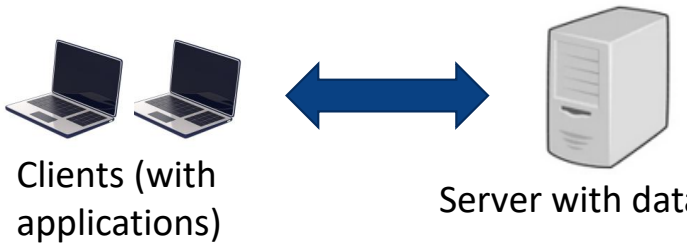
storage logic: It's a persistent storage of data. This is what database are. It looks after data integrity

user interface and application are sitting on

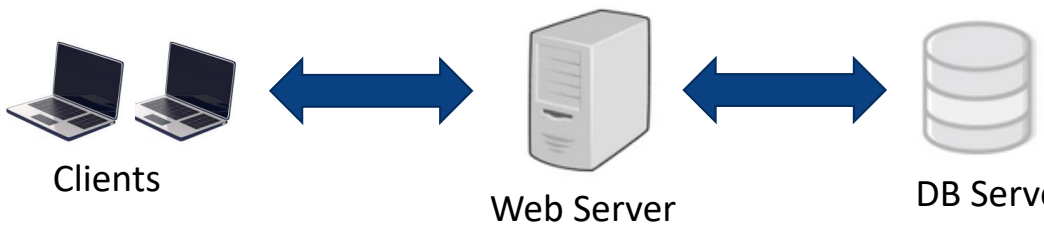
Multi-tiered Architectures

the same physical tiers (on the same computer) or opposite. user interface is sitting on your device, and application and data storage are sitting together on the same server.

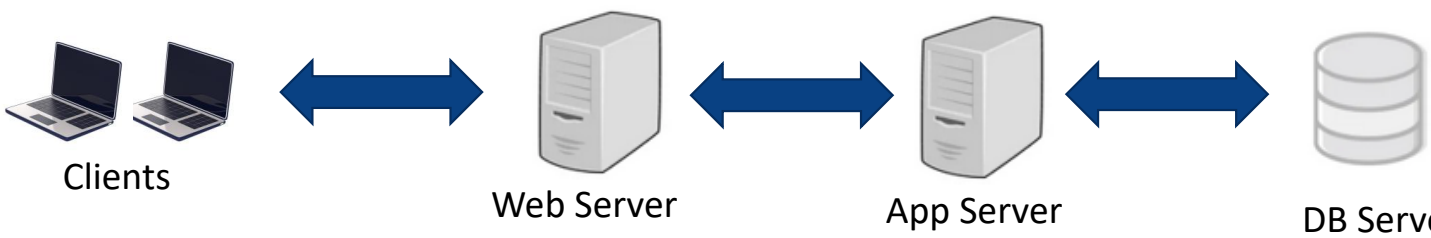
- 2 tiers



- 3 tiers



- 4 tiers



Evolution of application architectures

Mainframe / dumb terminal

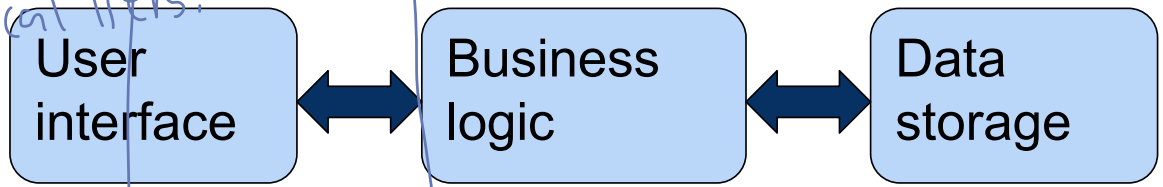
- One large computer handles all logic
- Problem: doesn't scale with number of users

Client-Server architecture

- 2-tier: e.g. file server, database, web server
- 3-tier: separation of Presentation, Processing and Storage logic

Web architecture

- a particular form of 3 or 4 tier architecture



database appeared similar shareable, database accessed by different application installed on individual personal computer.

other users and then put over. They could overwrite my file with their own version, so we had special file server)

As above mentioned, presentation and processing were sitting together, and database were sitting on the different(another) computer.

one more tier could appear, which is like seperating Web server from the application server. So that web server handles the requests to enter into the network. and then it would talk to the application server that could you process this request.

It's possible to have dedicated application servers. And in this case, depending on your request. Web server will know which application server to direct it to.

Mainframe ("1Tier")

All tiers are on the same computers.

ex: NASA are still using this.

these computers are really powerful

Mainframes and mini-computers

Dumb terminals (no processing at client end)

Entire application ran on the same computer (mainframe) for all clients.

- Database
- Business logic
- User interface

Enabling technologies included:

- Embedded SQL
- Report generators

Client Server - 2 Tier

The application was on every individual computer. The problem is that user interface and business logic are together.

Server is a relational DBMS

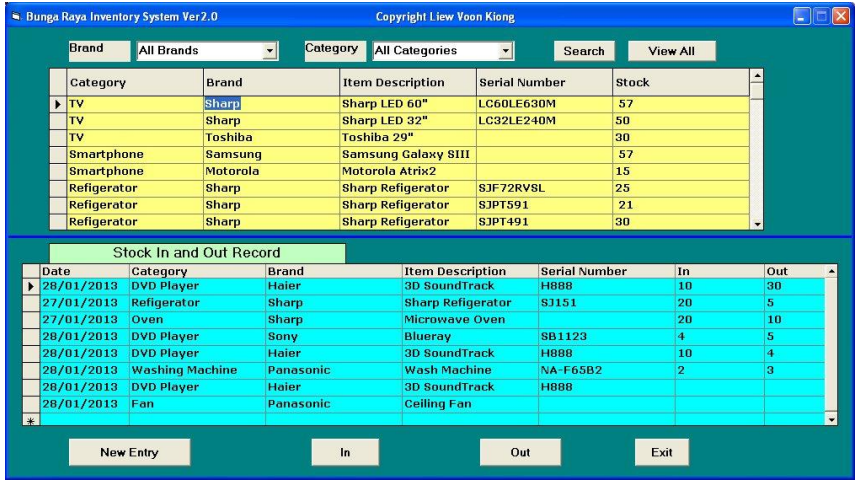
- data storage and access is done at the DBMS

But database were seperate, so these application could talk to database. And often the database was on the server on the network.

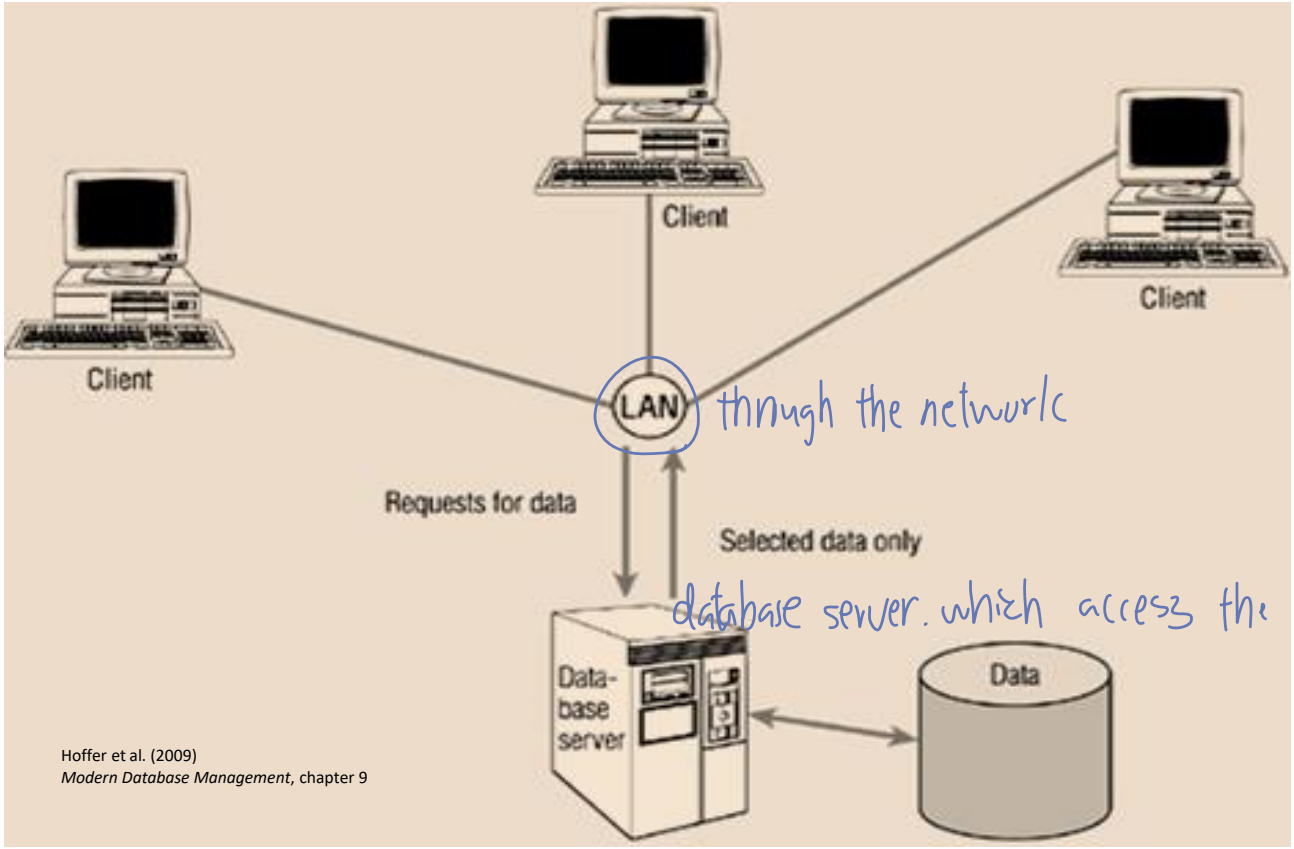
SQL queries sent to DB server, which returns raw data

Presentation, business logic is handled in client application

Platforms like Visual Basic (1990s into 2000s)



2 Tier Example

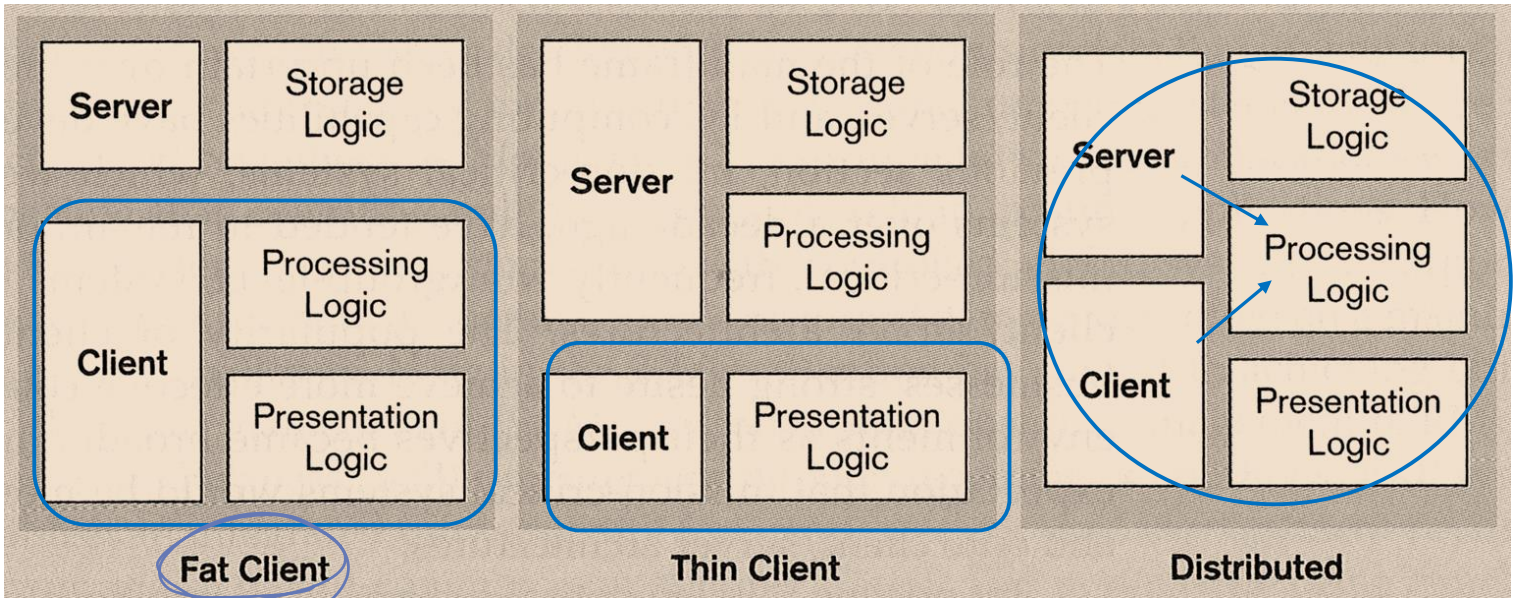


Hoffer et al. (2009)
Modern Database Management, chapter 9

Distribution of Processing Logic

2-tier distributions

- Processing logic could be at client, server, or both



Hoffer et al. (2009)
Modern Database Management, chapter 9

There are two approaches:

The server is not very powerful. So quite often the server had only storage logic. And the client has both processing and presentation logic. This is called fat client.

However, that would have a problem with security. If at least one of client is not paged on time, the virus could propagate through the whole networks. And obviously, these clients need to be powerful to do all the processing.

So later on, it was decided that maybe a client should have only presentation logic and all processing and storage logic should be on the server. So, if we want to improve business logic. there is only one place for update. (server)

This is call thin client. Your browser only a presentation take your input and show your output.

Distributed, because in thin client server is overloaded.

For example: validation of input on the client side and other processing on the server side,

But it is better to double check the validate input on the server side.

For example: with browers javasript often used to validate your input.

2-Tier Advantages and Disadvantages

Advantages

1. Clients and server share processing load
2. Good data integrity since data is processed centrally
3. Stored procedures allow some business rules to be implemented on the database server

Disadvantages

- Presentation, business logic, data model are intertwined at client
- If DB schema changes, all clients break ⇒ If you change data model (like add column to a table, actually you may need to rewrite the whole application to ensure that it doesn't crash). To update the client, you have to go with the disk from one computer to another.
- Updates need to be deployed to all clients
- DB connection for every client, thus difficult to scale
  - ⇒ to make sure that database server can actually manage the workload.
- Difficult to implement beyond the organisation (to customers, to suppliers)
- Interoperability issues 互操作性
  - And it's difficult to work with external parties with external

The issue with customer and supplier how to improve communication with them. improve order processing and all the arrivals.

stakeholders.

3-Tier architecture

the client are thiner and thiner.

Client program <-> Application server <-> Database server

presentation logic

deal with business logic

deal with storage logic

You have three logical tiers, but have 2 or 3 physical tiers

Presentation logic

- Client handles interface
  - Thinner clients

Limited or no data storage (possibly no hard disk)

Business logic

- Application Server deals with business logic

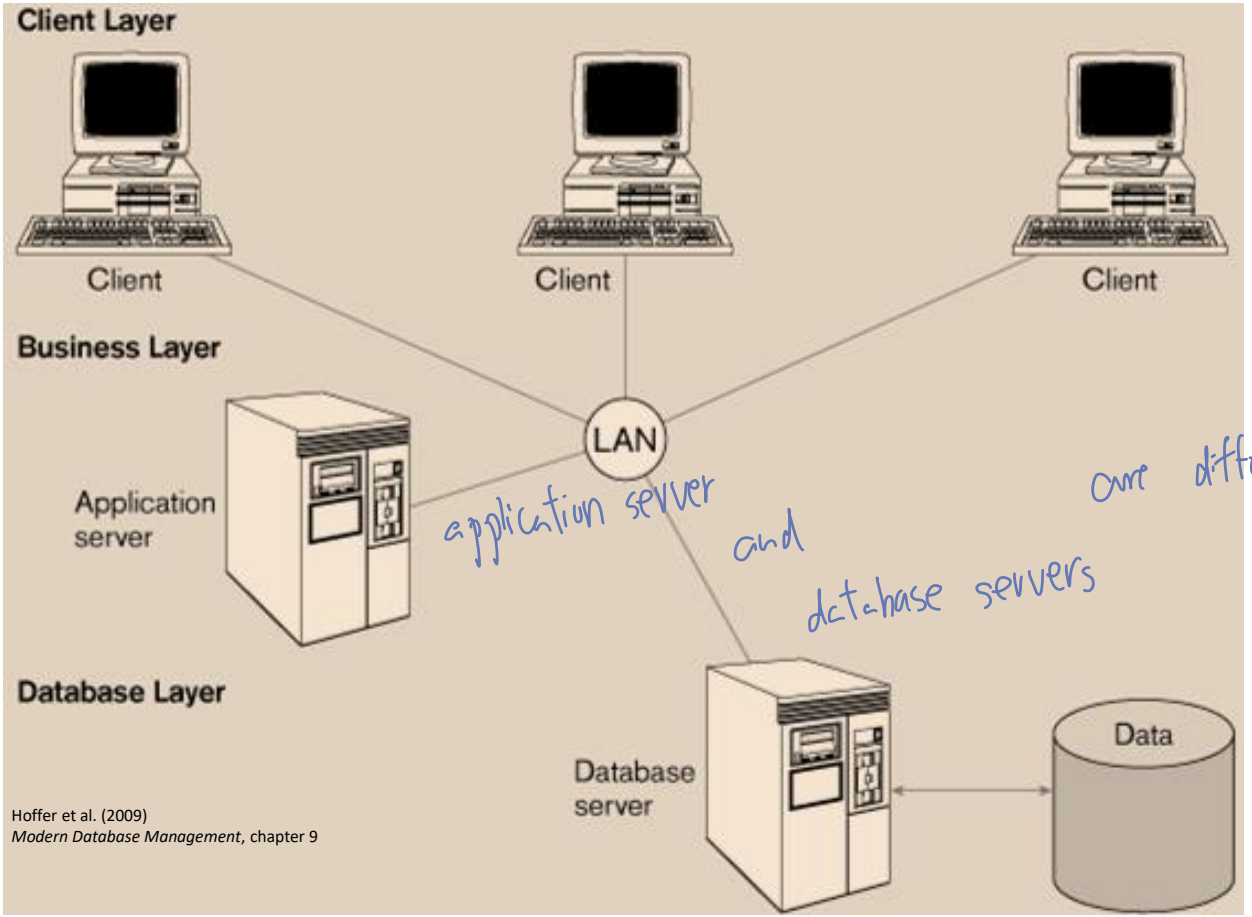
Storage logic

- Database server deals with data persistence and access

Logical tiers – 3
Physical tiers – 2 or 3
(application server and database server may physically sit on the same computer)

A Three-tier architecture - Example

3 tier architecture



Hoffer et al. (2009)
Modern Database Management, chapter 9

3-Tier advantages and disadvantages

Advantages

1. Scalability
2. Technological flexibility (can change business logic easily) when change you only need to update one place.
3. Can swap out any single component fairly easily (easily replaced single component)
4. Long-term cost reduction (for long term is cheaper)
5. Improved security – customer machine does presentation only (because no application on the customer server, so nothing for hacker to interference)

Disadvantages

- High short-term costs
- Tools and training
- Complex to design
- Variable standards

different server won't communicate (like LINUX and Microsoft Server)

3-Tier (web based)

Browser handles presentation logic

Browser talks to web server via simple, standard protocol

Business logic and data storage handled on server(s)

Pros

- Everyone has a browser
- No need for install and maintain client software
- HTML and HTTP are simple standards, widely supported
- Opens up the possibility of global access to database

(maybe in Google Chrome or Firefox have slightly different. But overall it will be the same experience)

Cons

- Even more complexity in the middle-tier

what is middle-tier in 3Tier

- Simple standards = hard to make complex application
- Global access = potential security nightmare (next slide)

security should has to be enforced at every tier.

Security in Multi-tier Applications

Network environment creates complex security issues

Security can be enforced at different tiers:

- application password security
  - for allowing access to the application software
- database-level security
  - for determining access privileges to tables
- secure client/server communication (when our data is travelling)
  - via encryption

Exercise

Consider MySQL Workbench and MySQL Server

presentation is workbench, compilation processing is happening by MySQL server database management system.

1. How many logical tiers? 2, because MySQL Server has processing power plus user interface
2. If MySQL server is on the engineering server, how many physical tiers are there? 2 physical tier, server and your computer
3. If MySQL server is a local instance, how many physical tiers are there? Only one. Because everything happen on one computer

Logically, it's more than one because workbench is user interface and database server is seperate thing

THE UNIVERSITY OF MELBOURNE

Web Applications

Overview of Web Apps

business logic with data access is sitting on the webserver or Webserver could be a gatekeeper, sending it to dedicated application server.

Why web apps?

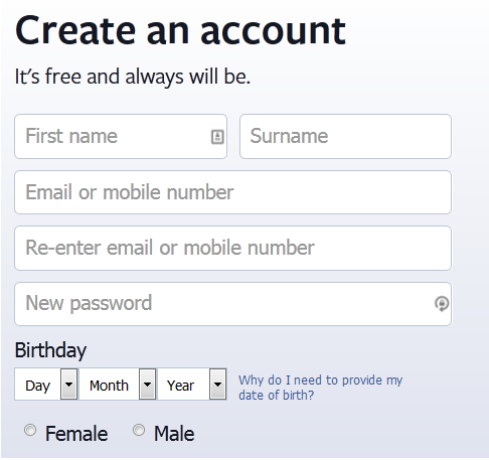
How web apps work

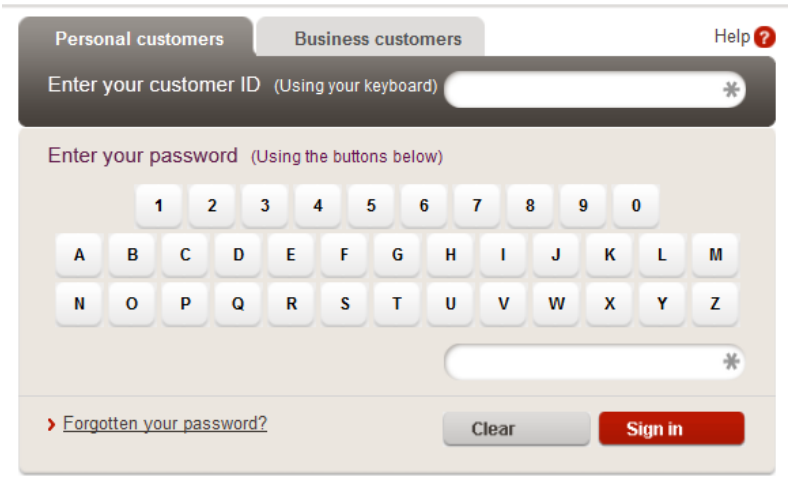
Making an HTML document (browsers work in HTML)

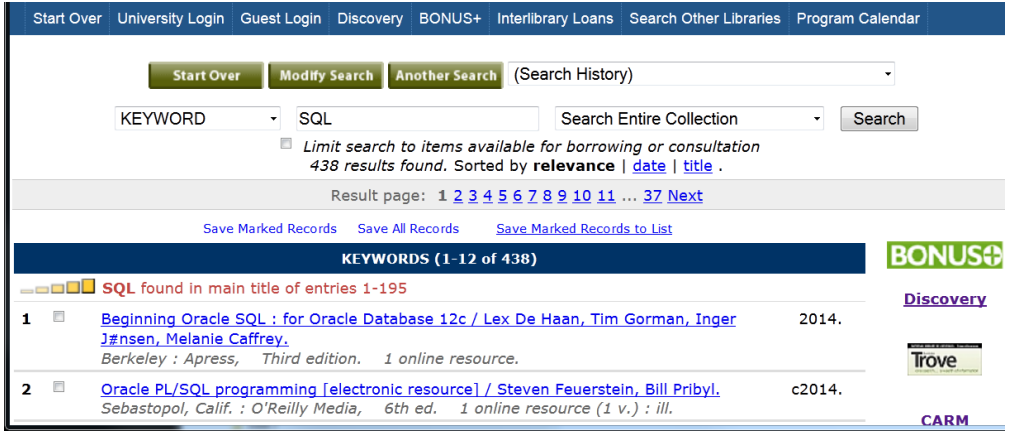
Connecting to the DB

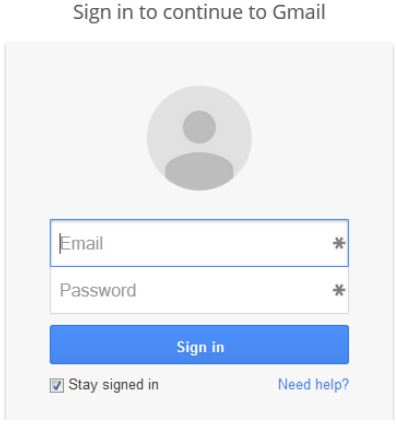
Web services

Example web applications

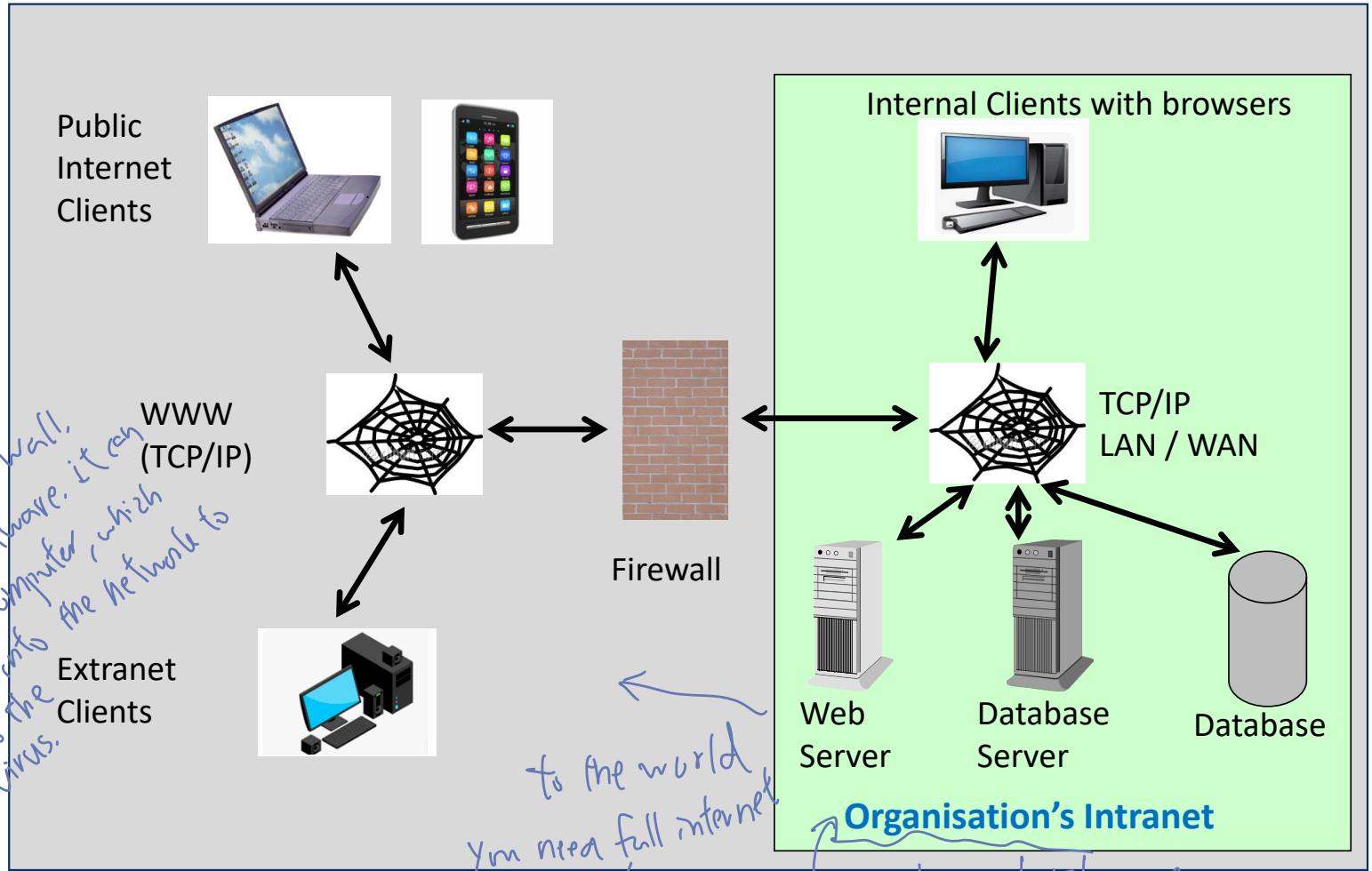








Architecture of a web app



Why create web applications?

Web browsers are ubiquitous

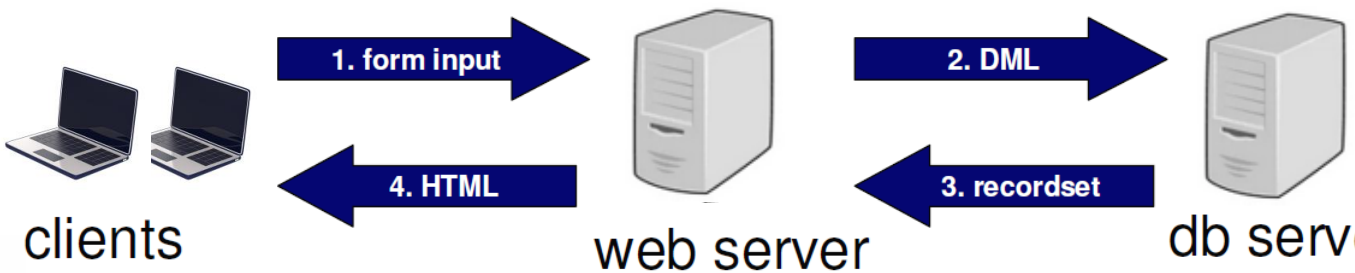
No need to install client software for external customers

Simple communication protocols

Platform and Operating System independent ("interoperable")

Reduction in development time and cost

Has enabled eGov, eBusiness, eCommerce, B2B, B2C



Exercise: re-draw the diagram with an application server being a separate physical instance.

Web Infrastructure

Browser

- Software that retrieves and displays HTML documents

Web Server

- Software that responds to requests from browsers by transmitting HTML and other documents to browsers

Web pages (HTML documents)

- Static web pages
  - content established at development time
- Dynamic web pages
  - content dynamically generated using data from database

World Wide Web (WWW)

- The total set of interlinked hypertext documents residing on Web servers worldwide

Internet

- Global network infrastructure that hosts the WWW (and other applications, e.g. email, chat rooms, etc.)

Web-related Languages

Hypertext Markup Language (HTML)

- Markup language used to define a web page (content and structure)

Cascading Style Sheets (CSS)

- Control appearance of an HTML document

JavaScript (JS)

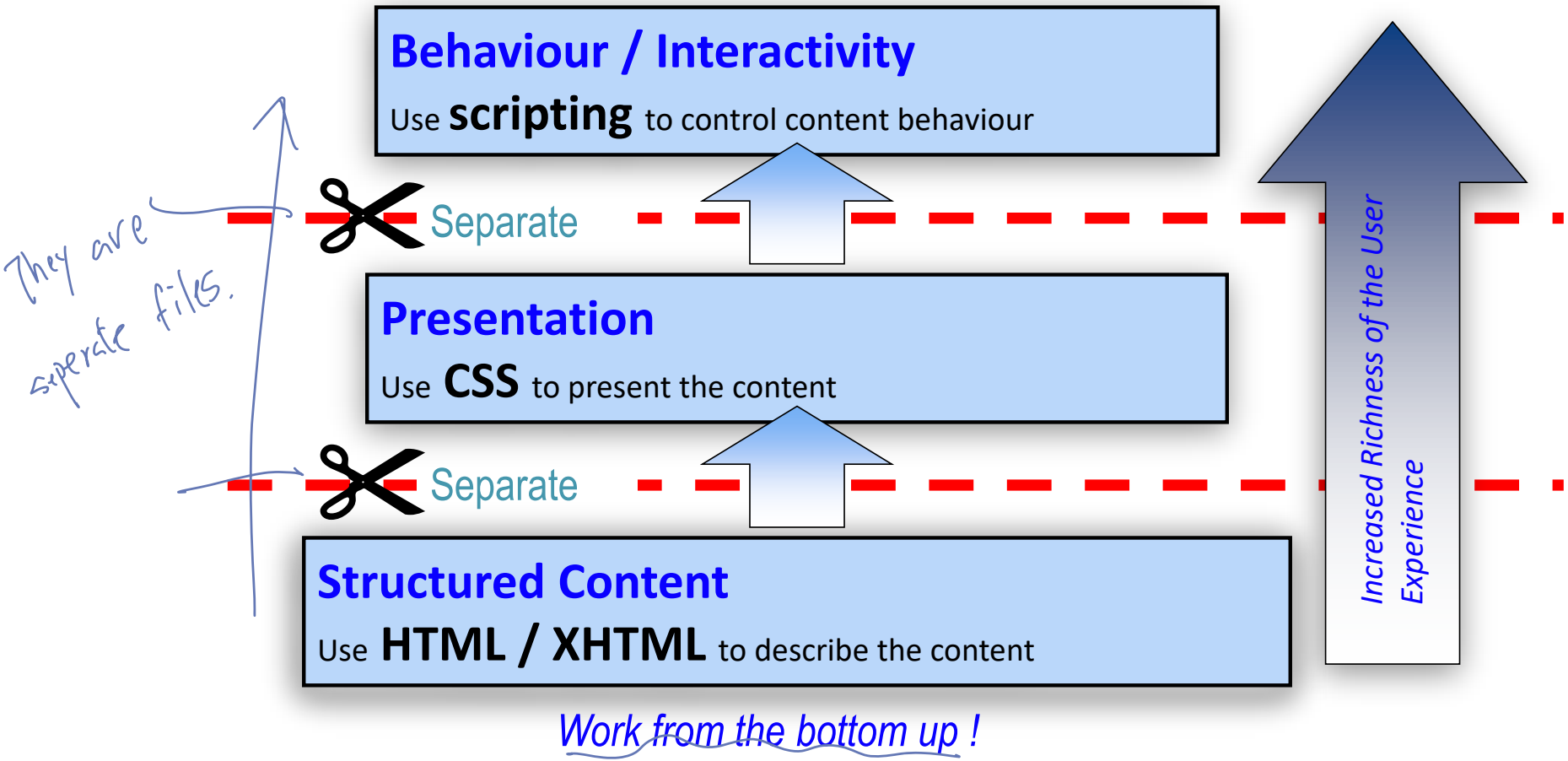
- Scripting language that enable interactivity in HTML documents

Extensible Markup Language (XML)

- Markup language used to transport data between web services

For more info www.w3schools.com

Build your webpages using the correct tools



Work from the bottom up !

Web page = HTML document

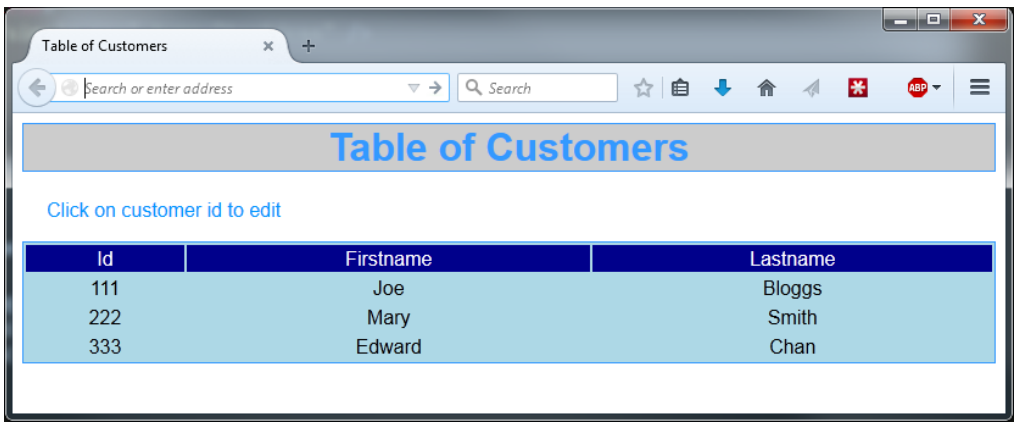
A structured file of elements defined by HTML tags

Interpreted by web browser for display

```
<head>
    <title>Table of Customers</title>
    <link rel="stylesheet" href="simple.css" type="text/css" />
</head>

<body>
    <h1>Table of Customers</h1>
    <p>Click on customer id to edit</p>
    <table>
        <thead>
            <tr><td>Id<td>Firstname<td>Lastname</tr>
        </thead>
        <tr><td>111<td>Joe<td>Bloggs</tr>
        <tr><td>222<td>Mary<td>Smith</tr>
        <tr><td>333<td>Edward<td>Chan</tr>
    </table>
</body>

```

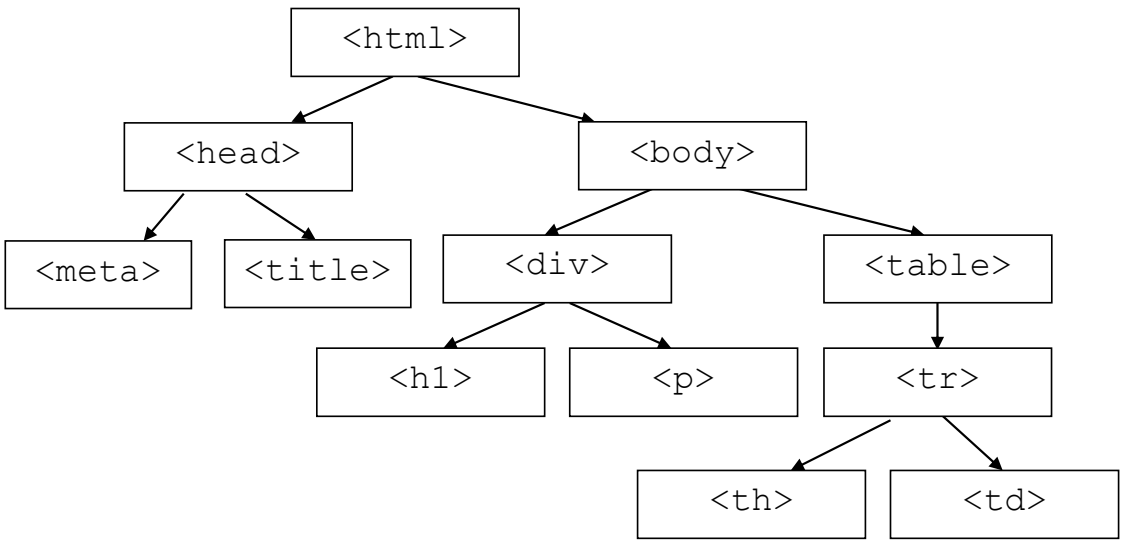


HTML Document Structure – Tree View

The "**root**" element of any html document, is the **`html`** element, which usually contains only two children **`head`** and **`body`**

- The **`head`** then contains the **`title`**, and other 'head' elements.
- The **`body`** can contain many other elements

```
<html lang="en">
<head>
    <meta ....  />
    <title>...</title>
</head>
<body>
  <div>
    <h1>...</h1>
    <p>...</p>
  </div>
  <table>
    <tr>
        <th>...</th>
        <td>...</td>
    </tr>
  </table>
</body>
</html>
```



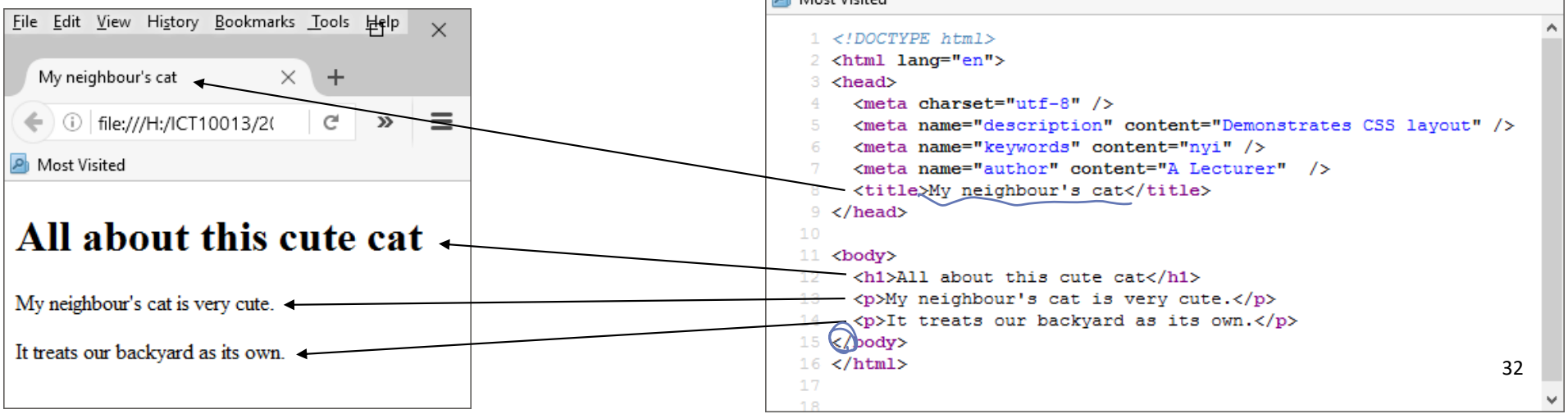
Web Browsers display basic web page

The <h1> is a heading element

- there are six heading sizes ranging from <h1> to <h6>
- <h6> is the smallest

The <p> is a paragraph element

- A browser inserts empty lines before each paragraph



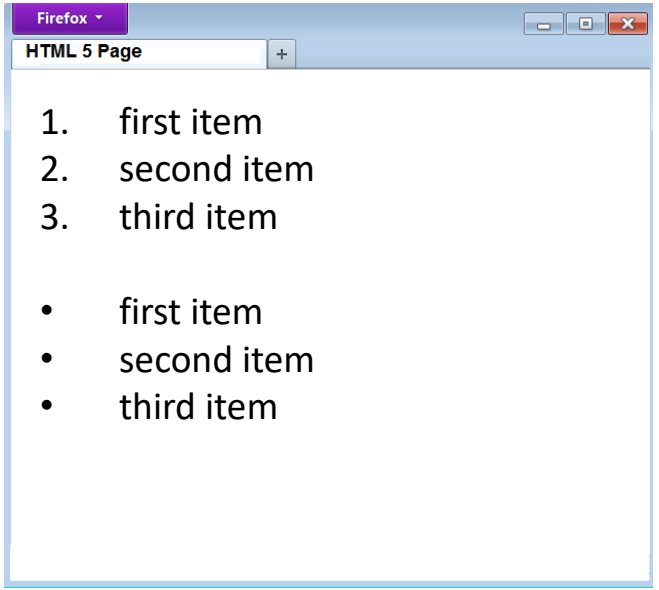
Lists

Ordered list example

```
<ol>
  <li>first item</li>
  <li>second item</li>
  <li>third item</li>
</ol>
```

Unordered list example with bullet point

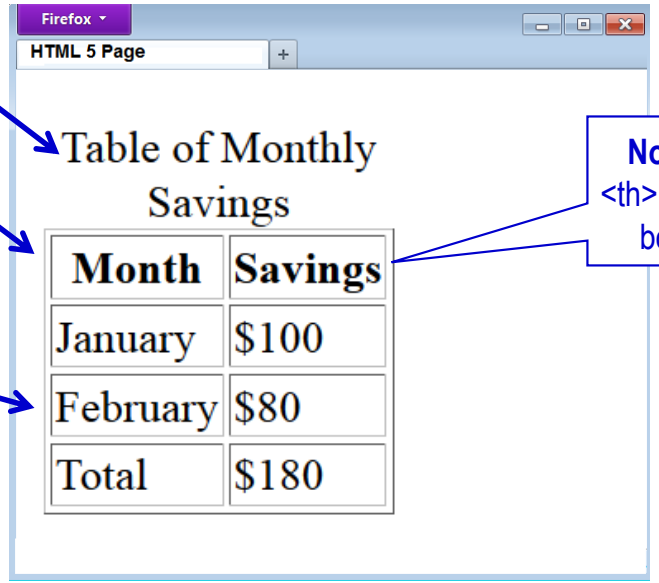
```
<ul>
  <li>first item</li>
  <li>second item</li>
  <li>third item</li>
</ul>
```



Table

Deprecated attribute border = can now only be "1" (show a border) or "0" (do not show a border). Better way to style is CSS.

```
<table>
  <caption>Table of Monthly Savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
  <tr>
    <td>Total</td>
    <td>$180</td>
  </tr>
</table>
```



Note: by default the <th> cells are presented bold and centred !

Form element

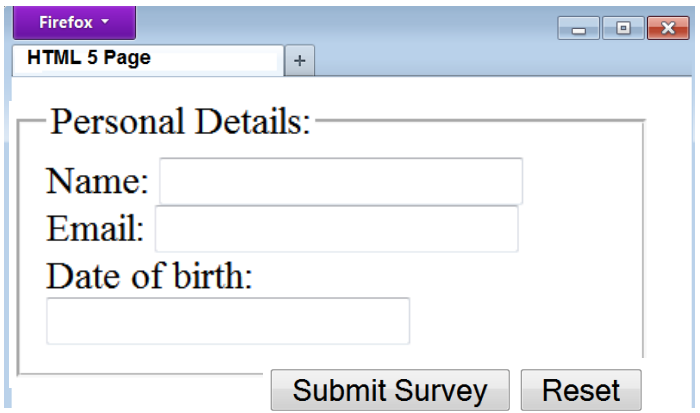
if you want to get from user you need forms.

<form> … </form> provides a mechanism to allow a user to enter data into a web page.

Entered data can be submitted to a server, which it turn can receive the data, process the data and generate a response.

Possible responses may include: (for submit)

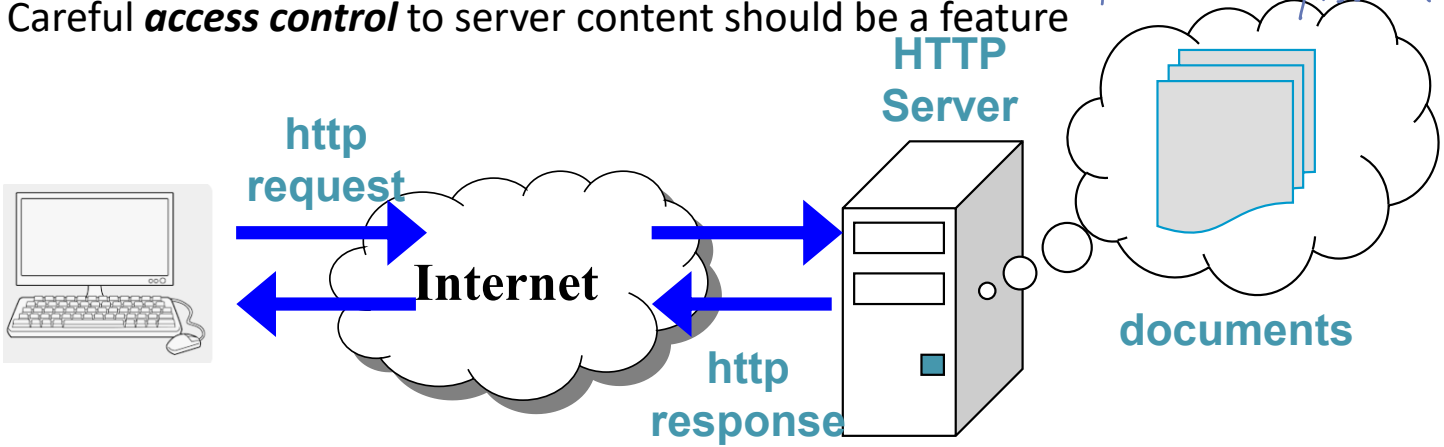
- display information on a web page;
- adding data to a database; or
- sending an email message.



Web Server Features

A Web server is made up of several components:

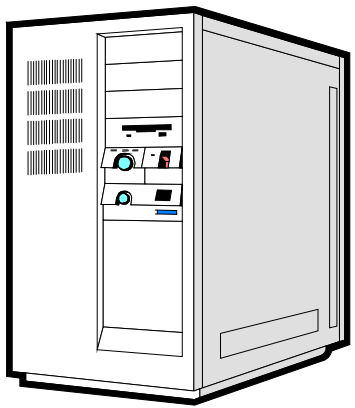
- A **computer** with an **Internet connection** and **operating system**.
  - The server program usually ***runs continuously***.
- **Web server software** to receive and respond to HTTP requests.
  - Handles ***multiple requests***
- **Information**: a collection of documents to be served.
  - Careful ***access control*** to server content should be a feature



The client, using standard WWW technology, sends a request for a document to the Web Server using HTTP. The request takes the form of a URL specification.

Client

Web Server



browser send the request with IP address.

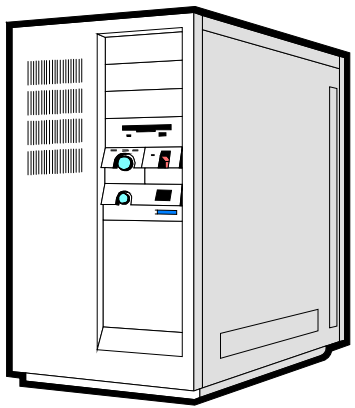
GET http://www.server.com/home.html

The web server locates the file by mapping the URL specified by the client to a file in the local resource base (e.g. hard disk).

Client

GET http://www.server.com/home.html

Web Server

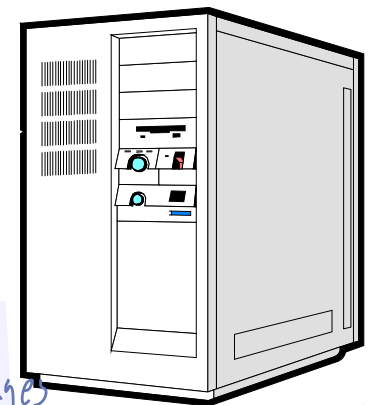


C:\htmldocs\home.html

The Web Server sends the file back to the client 'as is' along with specific server generated headers (containing control information for the browser)

Web Server

Client



GET http://www.server.com/home.html

server generated headers +
home.html

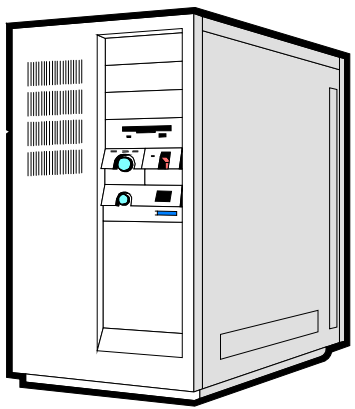
generate HTML page and send back to the browser and shown to the user.

```
<html>
<body>
<h1> WELCOME TO SERVER.COM </h1>
</body>
</html>
```

Client web browser interprets and displays (renders) the HTML document.

Client

Web Server



GET http://www.server.com/home.html

server generated headers +
home.html

WELCOME TO SERVER.COM

Static vs Dynamic web pages

STATIC web page

- the URL identifies a file on the server's file system
- server fetches the file and sends it to the browser
- the file contains HTML
- browser interprets the HTML for display on screen

DYNAMIC web page

- URL identifies a program to be run
- web app runs the program
- program typically retrieves data from database
- elements such as TABLE, LIST are populated with data
  - web app uses LOOPS to fill the contents of TABLEs and LISTs.
  - e.g. SELECT * FROM Product; (returns a set of product entities)
  - FOR p IN ProductList, print a row in HTML table

dynamic web pages, instead of searching for HTML page, the server will search for the actual program that can process that request and that web app will run the program

program typically retrieve data from database, element will be positioned into web page / HTML page and send to the browser.

Problems with old-style web apps

```
30  $sql = "insert into EVENT values (null, null, 'L', '" . $_SESSION["thisClient"] . "', 'logged in')";
31  mysql_query($sql);
```

Placing “raw” SQL inside PHP/HTML files

- Mixes presentation, business, database logic
- Hard to maintain when things change
- Best practice: separation of logical elements, e.g. MVC (model view controller)

Lots of reinvention of wheels

- each developer writes their own solution to common features
- e.g. login security, presentation templates, database access

Increasing variety of clients, e.g. phones and tablets

- Manual programming for different platforms

=> web application frameworks

- examples: Ruby on Rails, .Net, Symfony, AngularJS, Django

SQL within PHP

```
<html><body><?php
$host="localhost"; $user="root"; $password="";
$database="cars";
$connection=mysqli_connect($host, $user, $password)
 or die("could not connect to server");
$db=mysql_select_db($database, $connection)
 or die("could not select the database");
$query="select regNo, colour, price from car
        where make='Toyota' AND model ='Yaris'";
$result=mysqli_query($query) or die("could not run
the query");
//Display result
echo "<h1>Toyota Yaris</h1>";
echo"<table cellspacing='15'>";
while ($row=mysqli_fetch_array($result)) {
   //format price to 2 dec. places
   $f_price=number_format($row[2],2);
   echo "<tr>\n
        <td>$row[0]</td>\n
        <td>$row[1]</td>\n
        <td align='right'>\$$f_price</td> </tr>\n";
   //Note \$ is to display $
}
echo"</table>";
?></body></html>
```

Web Services (like API for interacting with remote database, object, programs)

The WWW allows humans to access remote databases

Web Services allow *computers* to access remote databases

What is a web service?

- A web-based interface for interacting with a remote component/object, regardless of the location, nature or environment of the component.
- Software listening for requests at a particular port over a network

Why are they such a big deal? (business to business)

- Because they provide a simple, reliable method of connectivity for applications both within and across enterprises, with a business logic focus
- Web services are platform independent, discoverable and self-describing.

Interoperability?

- Platform independence
  - Components can be activated by function/characteristics independent of implementation details
- Can dynamically invoke across different architectures

Web Services

OPTIONAL

2 major approaches: SOAP and REST

Both define how to build application programming interfaces (APIs), which allow data to be communicated between web applications.

- Simple Object Access Protocol (SOAP) is a text-based messaging protocol that runs on HTTP
  - sends messages between web service clients and web service servers in XML
- Representational State Transfer – another client-server based architecture (not a protocol); it defines four interface constraints:
  - Identification of resources
  - Manipulation of resources
  - Self-descriptive messages
  - Hypermedia as the engine of application state
  - Structured data usually returned in XML or JSON format
  - REST nouns are resources, addressed via URIs
  - REST verbs correspond to DML statements

    GET (select),
    POST (insert),
    PUT (update),
    DELETE (delete)

More details
https://www.cleo.com/blog/knowledge-base-web-services

XML and JSON data formats

used by web services for data exchange

- JSON
  JavaScript Object Notation
- XML (take more space) because it has tags
  eXensible Markup Language

Source: www.w3school.org

The following JSON example defines an employees object, with an array of 3 employee records:

JSON Example

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

The following XML example also defines an employees object with 3 employee records:

XML Example

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

What's examinable

Identify the limitations of SQL

Distribution of Processing Logic

Database Architectures

Web languages

Web architecture

HTML elements

How static and dynamic web pages work (high level)

THE UNIVERSITY OF MELBOURNE

Thank you