THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

1b. Course Overview

CSCI 2541 Database Systems & Team Projects

Parmer

Slides adapted from ones by Prof. Bhagi Narahari & Wood

A DBMS should provide…

Structure that is **independent** of the underlying file formats

Queries to flexibly read, update, and delete information

Transactions that provide guarantees about **multi-user consistency**

DBMS: How to structure the data?

Structure that is **independent** of the underlying file formats

What **data** is needed?

- Eg: What do we need to store to uniquely identify a restaurant customer?

How to **store & organize** the data?

- How many attributes are really needed about a student/course/faculty
- What is an efficient way to organize the data?
  - This is why we will need to study schema design and Normal forms

Data Models and Representation

Structure that is **independent** of the underlying file formats

A **data model** is a formal framework for describing data.

- Data objects, relationships, constraints (business rules)
- Provides primitives for data manipulation and data definition
- Provides us with the mathematical basis to prove/assert properties and show correctness of algorithms

The **relational model** is one of the main models of data that is **independent** of its data structures and implementation

- Data organized as **relations** (“tables”)

Not the only way!

- NoSQL databases model unstructured and big data without requiring strict relations
- Other data models: network, hierarchical, Object Oriented…
- Relational model can be inefficient for many such applications

DBMS: How to provide abstraction?

Structure that is **independent** of the underlying file formats

The major problem with developing applications based on files is that the application is **dependent** on the file structure.

There is no **program-data independence** separating the application from the data it is manipulating.

- If the data file changes, the code that accesses the file may require changes to the application.

A major advantages of DBMS is they provide **data abstraction**.

Data abstraction allows the *internal definition of an object* to change without *affecting programs that use the object* through an external definition.

Database Schema

Structure that is **independent** of the underlying file formats

Similar to **types** and **variables** in programming languages

Schema – **structure** of the database

- Ex: database contains information about Students and Courses and the relationships between them
- Defines columns and the type of data that can be stored in them

Occurs at multiple levels:

- **Logical Level**: Database design, definition of structure and relations
- **Physical Level**: Implementation of how data is stored on disk

Customers Relation (Table)

first_name	last_name	email	phone	reservation	birthday
Kelli	Perris	kperris0@nifty.com	963-930-8531	1/6/2020	9/12/1958
Goddart	Braams	gbraams1@ted.com	534-300-7372	1/26/2020	1/18/1979
Merrel	Clere	mclere2@blogger.com	194-430-7153	1/25/2020	2/12/1957
Towney	Bratcher	tbratcher3@narod.ru	304-227-0235	1/5/2020	7/10/1977
Latia	Peete	lpeete4@w3.org	448-368-1546	1/28/2020	3/6/1964
Hadria	Rann	hrann5@cbsnews.com	206-421-4913	1/24/2020	1/5/1976
Bastian	Clother	bclother6@microsoft.com	104-598-7586	1/25/2020	9/15/1965
Corene	Attoe	cattoe7@soup.io	819-616-3261	1/20/2020	3/7/1946
Sara-ann	Creeboe	screeboe8@theatlantic.com	831-348-1941	1/13/2020	4/15/1998

Levels of Data Modeling

Structure that is **independent** of the underlying file formats

Logical Level: describes data stored in the database and the relationship between them

```
ex:  type customer {
         name: string
         email: string
         birthday: date
     }
```

Physical Level: describes how a record is stored (i.e., how is data organized on the disk)

- Ex: sorting, page alignment, index

Big Idea: Logical and Physical level independence

- Can change one without changing the other!!

Data Independence

Structure that is **independent** of the underlying file formats

Logical data independence

- Protects the user from changes in the logical structure of the data
- Lets us reorganize the customer “schema” without changing how we query/store it

Physical data independence

- Protects the user from changes in the physical structure of data
- Lets us change how student data is stored in memory/disk without changing how the user would write the query

Additional Views:

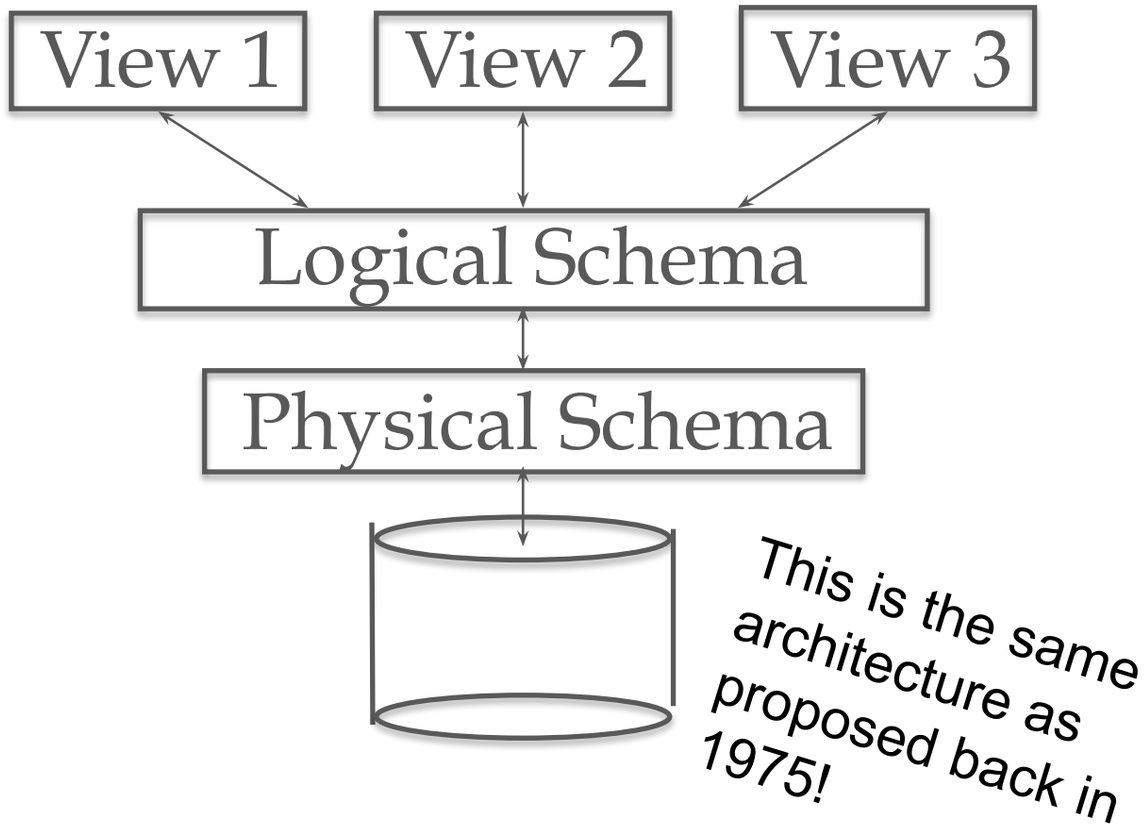
- DB applications hide details of data types and can also hide some information (salary?) for security & privacy purposes

Summary- Levels of Abstraction

Structure that is **independent** of the underlying file formats

Many views, single conceptual (logical) schema and physical schema

- Views describe how users see the data
- Conceptual/Logical schema defines logical structure
- Physical schema describes the files and indexes used



Schemas are defined using DDL; data is modified/queried using DML

Confusing? Curious?

ust covered…

Structure that is **independent** of the underlying file formats

Queries to flexibly read, update, and delete information

Transactions that provide guarantees about **multi-user consistency**

…marching on!

Queries & Data Independence

Queries to flexibly read, update, and delete information

What this means….

A user of a relational database system should be able to use the database without knowing precisely how data is stored, e.g.

```
SELECT Name, Reservation
FROM Customers
WHERE Name= 'Alan Turing'
```

The above "query" does not need to know how the data in Customers is stored. Why should you need to worry about that?!

How to define and use a database

Queries to flexibly read, update, and delete information

Data **Definition** Language (**DDL**) to specify database schema

- What data, and how it is organized (**logical level**)

Data **Manipulation** Language (**DML**) allows users to access or manipulate data as organized by data model

- procedural DMLs: require user to specify **what data and how** to get it
- *declarative* DMLs: require user to specify what data is needed **without** specifying how to get it.

Often, one language provides both features (e.g., SQL)

Relational DB Query Languages

Queries to flexibly read, update, and delete information

Formal query languages:

- **Relational algebra,**
- Relational Calculus,
- Why study formal languages?

Commercial query language: **SQL**

SQL: “descendent” of SEQUEL; mostly relational algebra and some aspects of relational calculus

- has procedural and non-procedural aspects
- Has DDL and DML components

What is SQL?

Queries to flexibly read, update, and delete information

SQL is **not** a specific database software

It is a standardized query language

- Defines how to create a database schema and issue read/write queries

DBMS software must implement the SQL **standard**

- Plus some of their own extensions
- None actually follow the official ANSI SQL standard precisely…

Good news: SQL queries are “cross platform” and will work on many different database systems

Confusing? Curious? Similar example?

Just covered…

Structure that is **independent** of the underlying file formats

Queries to flexibly read, update, and delete information

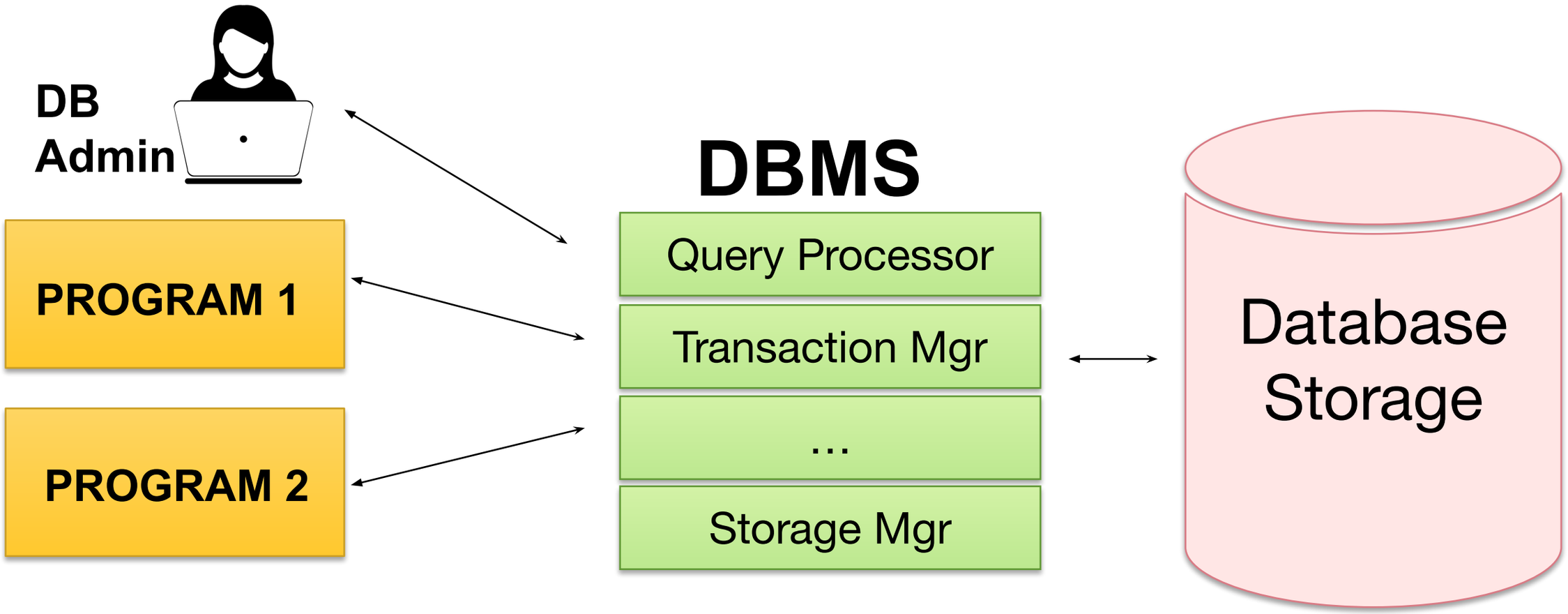
Transactions that provide guarantees about **multi-user consistency**

…marching on!

Connecting to a DB

Transactions that provide **multi-user consistency**

A typical DB Application design



DBMS

Transactions that provide **multi-user consistency**

The data abstraction is provided by the DBMS

- Separation b/w Logical and Physical, Query language parsing, multi-user, etc.

A database management system provides efficient, convenient, and safe multi-user storage and access to massive amounts of persistent data

- **Efficient & Convenient** - Able to handle large data sets, complex queries without searching all files and data items, easy to write queries
- **Scalability** – Large/huge data
- **Persistence & Safety** - Data exists after program execution completes, handles loss of power
- **Multi-user** - More than one user can access and update data at the same time while preserving consistency….concept of transactions

Components of a DBMS

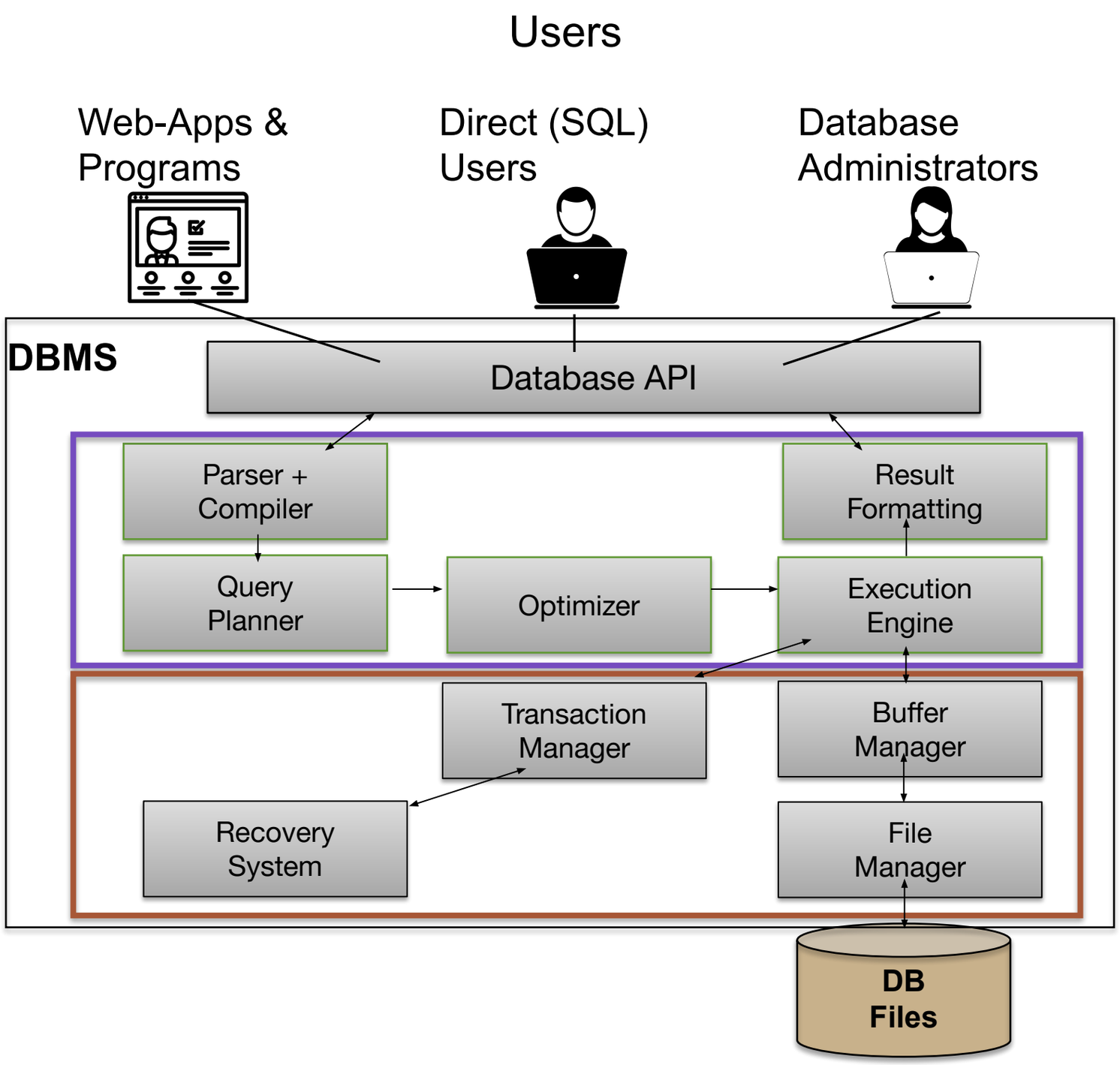
Transactions that provide **multi-user consistency**

A database management system provides efficient, convenient, and safe multi-user storage and access to massive amounts of persistent data.

A DBMS is a complicated software system containing many components:

- **Query processor** - translates user/application queries into low-level data manipulations
  - Sub-components: query parser, query optimizer
- **Storage manager** - maintains storage information including memory allocation, buffer management, and file storage
  - Sub-components: buffer manager, file manager
- **Transaction manager** - performs scheduling of operations and implements concurrency control algorithms
  - You will learn more about storage management and concurrency in the Operating Systems course… enjoy!

DBMS Architecture: Complete Picture



1. Structure that is **independent** of the underlying file formats

2. Queries to flexibly read, update, and delete information

3. Transactions that provide **multi-user consistency**

Which components provide these properties?

This course is about Database Design…

Focus is on design of databases

- Working at the **logical level**

Internals of DBMS is not the focus in this course

- BUT we will touch upon a few key concepts that make DBMS' work

- DBMS design brings together several key concepts from Computer Science
  - Languages, Compilers/translation, Algorithms, Data structures, Operating systems…

Database System Architectures

There are several different database architectures:

- **Embedded architecture** - DB files and DBMS processing occurs at the client's process (e.g. Microsoft Access or SQLite)

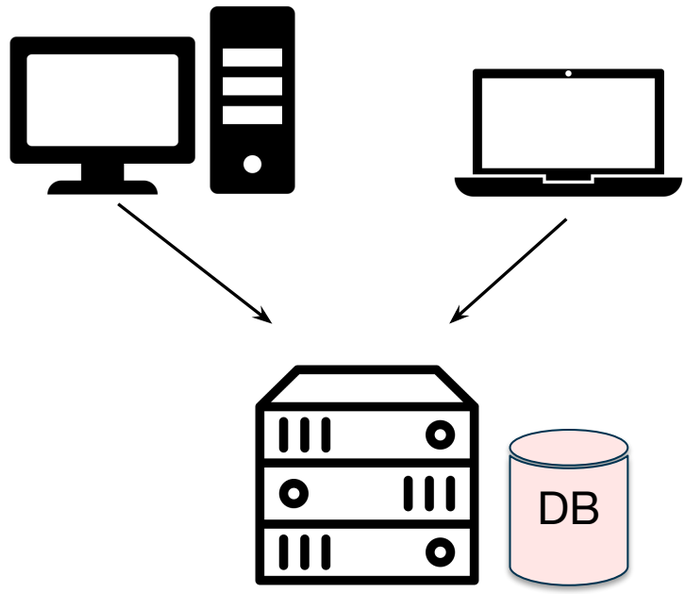
- **DB client-server architecture** - dedicated machine running DBMS accessed by remote clients (e.g. MS SQL Server, MySQL, Postgres)

- **Multi-Tier client-server architecture** - DBMS is bottom tier, second tier is an application server containing business logic, top tier is clients
  Web browser → App Server (Web Server + Business Logic) → Database

DB Client-Server Architecture

Clients

Server



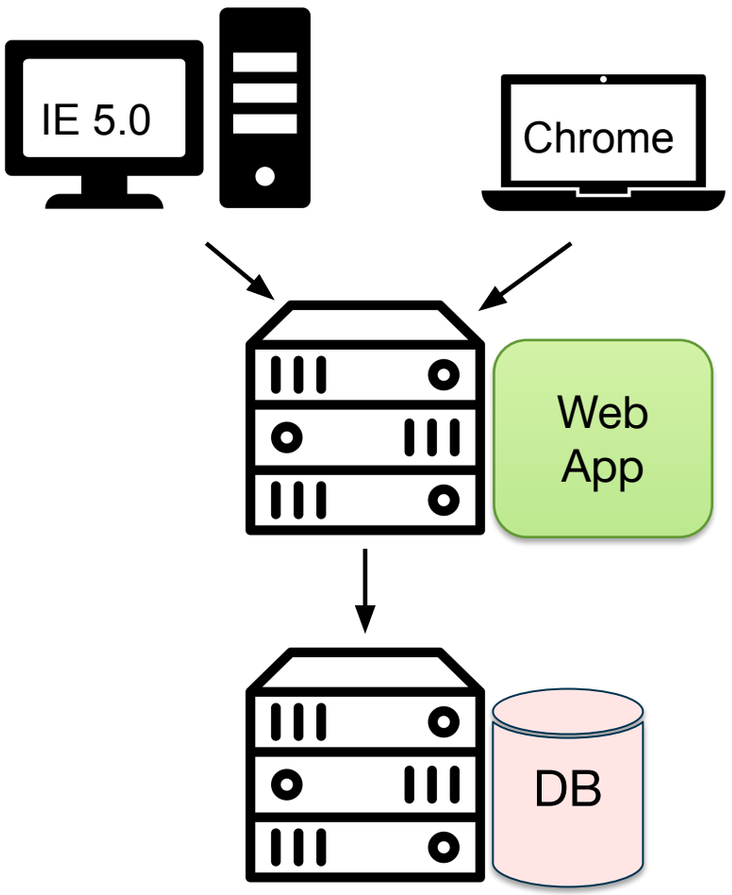
Tier 1: Client Application
- User Interface
- Business and Data Logic

Tier 2: Database Server
- Server-side validation
- Data storage/management

Advantages:

- Only one copy of DBMS software on dedicated machine
- Increased performance
- Easier to maintain consistency and manage concurrency

Three-Tier Client-Server Architecture – our approach



Tier 1: Client (Web/mobile)

- User Interface

HTML/CSS/Javascript

Tier 2: Application Server

- Business logic & data processing

Python+Flask

Tier 3: Database Server

- DBMS and data storage

MySQL

Advantages:

- Reduced client administration and cost using thin web clients.
- Easy to scale architecture and perform load balancing.

Attributions

These slides are adapted from materials made by Prof. Bhagi Narahari

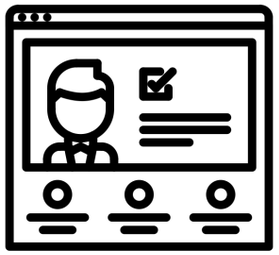
Image attribution:

Created by Wilson Joseph
from Noun Project

Created by Ditta
from Noun Project

Created by Gregor Cresnar
from Noun Project

Created by Wilson Joseph
from Noun Project

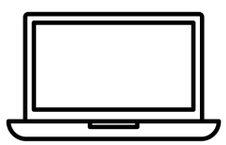


Created by lastspark
from Noun Project

Created by Wilson Joseph
from Noun Project

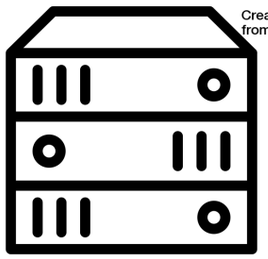
Created by Saifurrijal
from Noun Project

Created by ainul muttaqin
from Noun Project



Created by ainul muttaqin
from Noun Project

Created by ainul muttaqin
from Noun Project



Created by Srinivas Agra
from Noun Project

Created by Dawid Sobolewski
from Noun Project

Created by Yazmin Alanis
from Noun Project