GAZİ UNIVERSITY

FACULTY OF ENGINEERING
DEPARTMENT OF INDUSTRIAL ENGINEERING

Lecturer : Dr Ercan Ezin

IE326-DATABASE MANAGEMENT SYSTEMS

WEEK 1: INTRO TO DATABASE SYSTEMS

INTRODUCTION

AGENDA

- Course Introduction
- Classroom Introduction
- Week 1: Intro to Database Systems

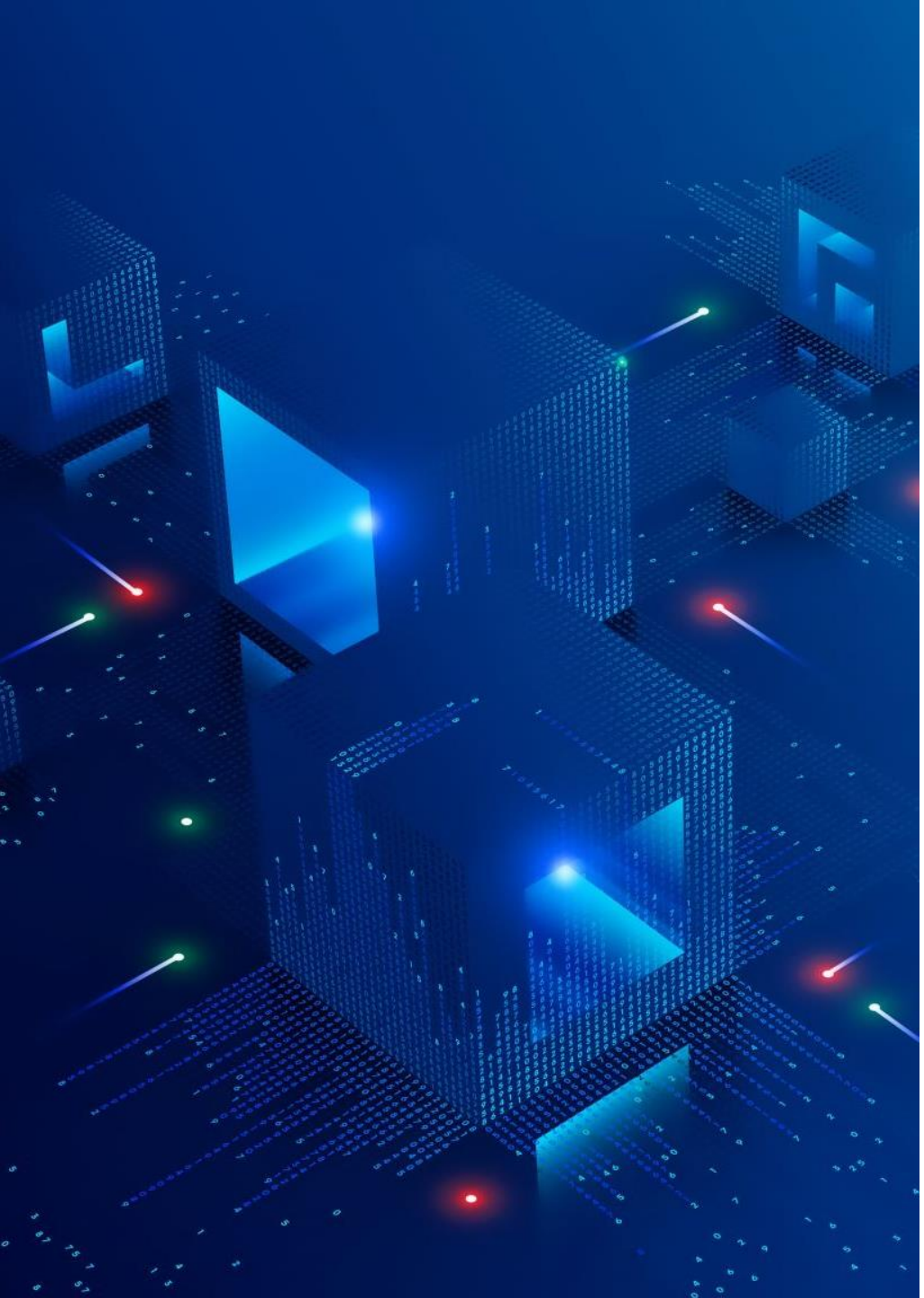
ABOUT THE COURSE

- **Course location:** Gazi University, Engineering Faculty, Classroom 401
- **Course Date and Time:** Mondays, between 13:30-16:30 (3 hours)
- **Course Length:** 15 Weeks
- **Evaluation:**1 project, 1 Midterm, 1 Final Exam
- **Contact:** dr.ercan.ezin@gmail.com (Always start with who you are and what course it is)
- **Attendance:** Compulsory for at least %70 percent. Every week with a slide is counted for the percentage.
- **Office Hours:** See me after the class due to being a part-time lecturer.

COURSE STRUCTURE

Check out https://gazi-end-muh.github.io/dbms/index.html for full course weekly structure.

Intro to Database Systems



Outline

- Database-System Applications
- Purpose of Database Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Database Management System

Database Management System (DBMS) contains information about a particular enterprise

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both *convenient* and *efficient* to use

Database systems are used to manage collections of data that are:

- Highly valuable
- Very large
  - What is large?
- Accessed by multiple users and applications, often at the same time.
  - What is a possible problem?

A modern database system is a complex software system whose task is to manage a large, complex collection of data.

Databases touch all aspects of our lives

Database Applications Examples

Enterprise Systems

- **Sales**: customers, products, purchases
- **Accounting**: payments, receipts, assets
- **Human Resources**: Information about employees, salaries, payroll taxes.

Banking

- Customer information, accounts, loans, and banking transactions.
- Credit card transactions

Finance

- Sales and purchases of financial instruments (e.g., stocks and bonds;
- Storing real-time market data

Database Applications Examples (Cont.)

Universities: registration, grades

Airlines: reservations, schedules

Manufacturing: management of production, inventory, orders, supply chain.

Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards

Web-based services

- Online retailers: order tracking, customized recommendations
- Online advertisements

Document databases

Navigation systems: for maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Why do we need a specialized DB

Why not build the DB directly on top of file systems.

Use the OS file system

Write the programs accessing the data using a regular PL

C,

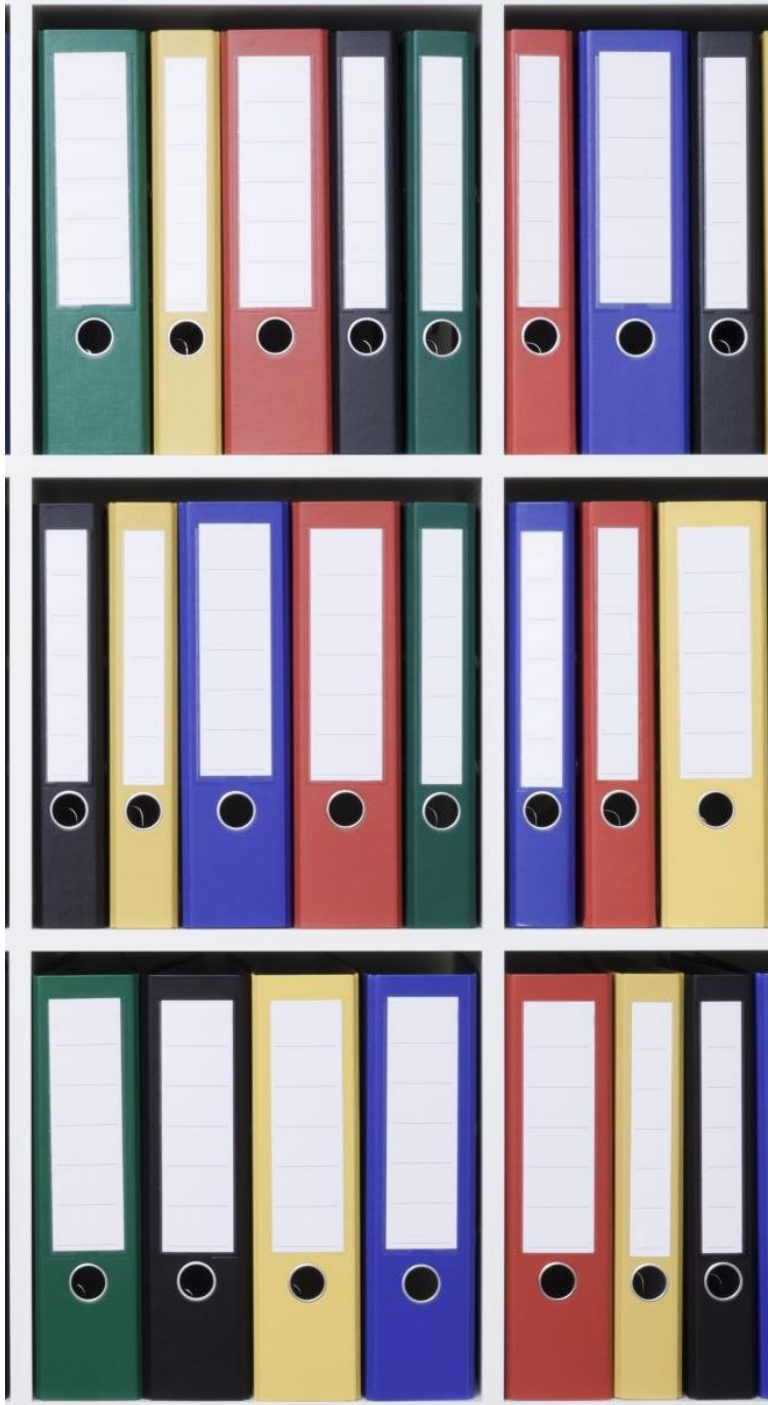
Java,

Python

Early Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- Data redundancy and inconsistency: data is stored in multiple file formats resulting induplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones



Early Database Systems (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Protection problems
  - Hard to provide user access to some, but not all, data
- Modern Database systems offer solutions to all the problems mentioned here.

History of Database Systems

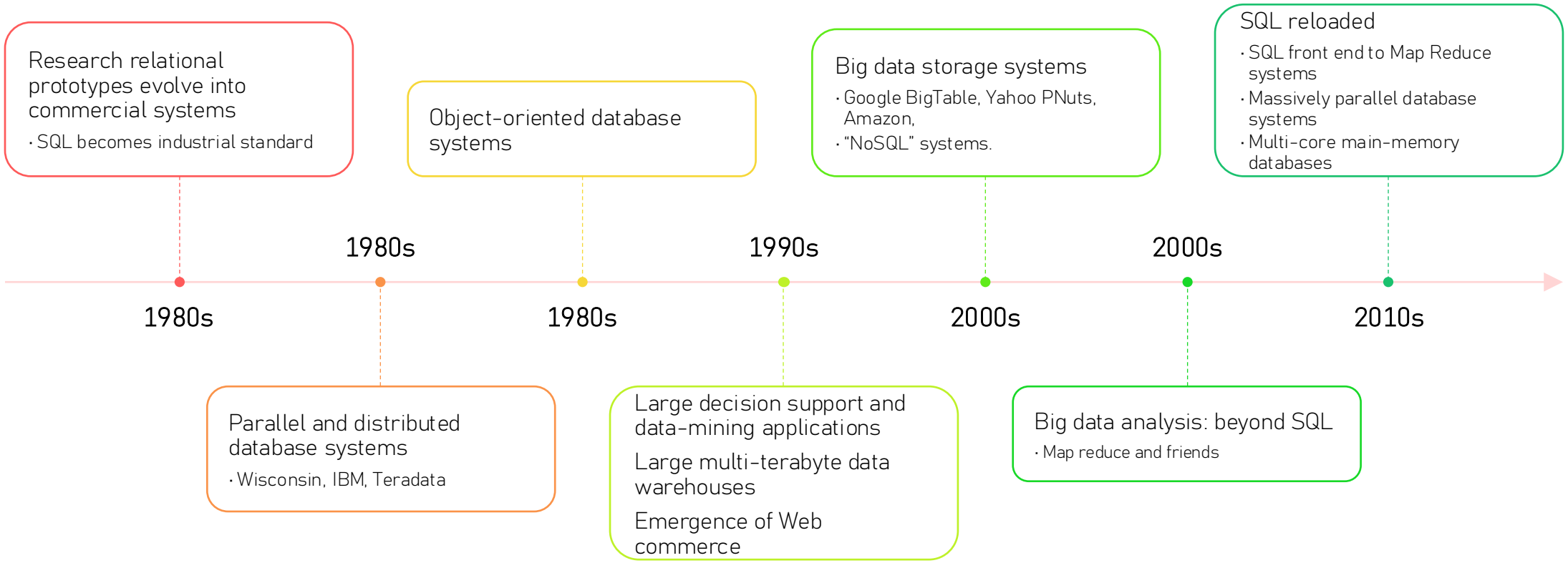
1950s and early 1960s:

- Data processing using magnetic tapes for storage
  - Tapes provided only sequential access
- Punched cards for input

Late 1960s and 1970s:

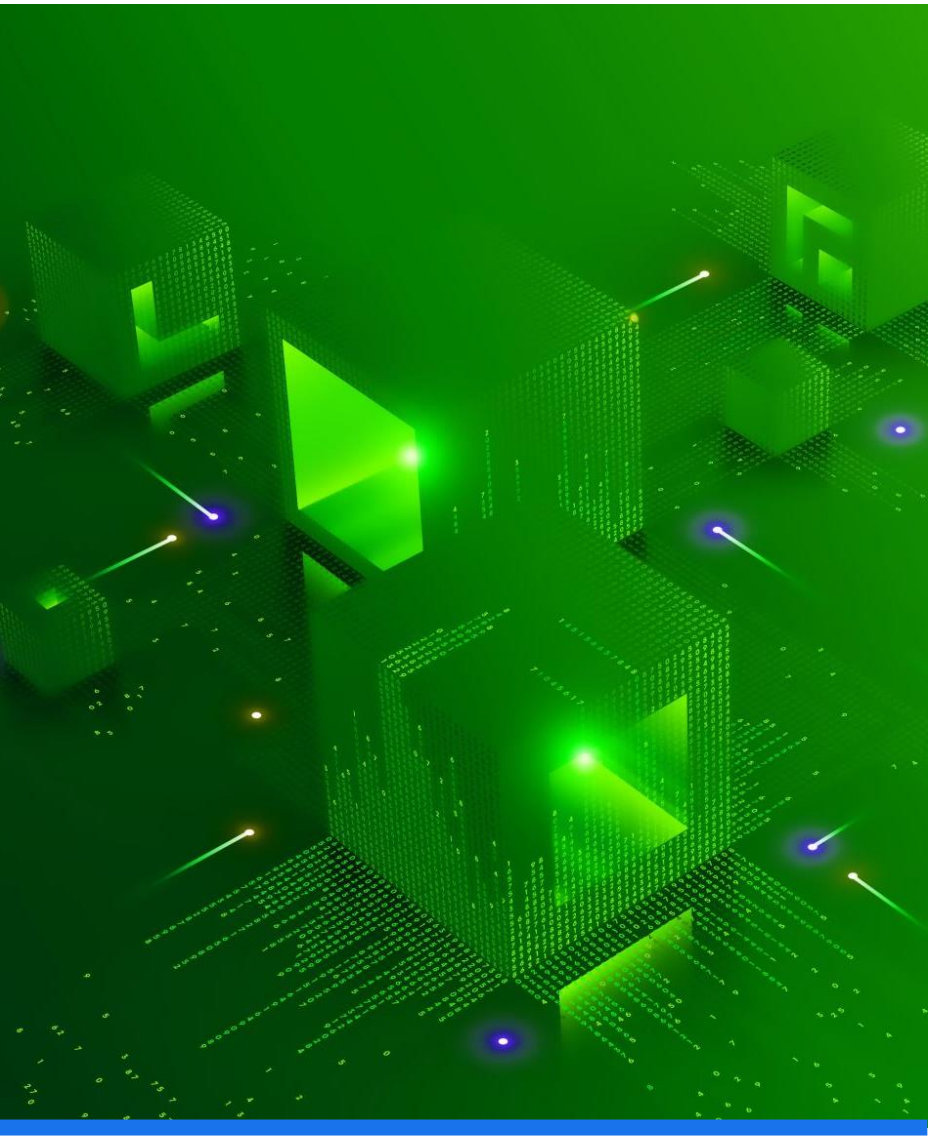
- Hard disks allowed direct access to data
- Network and hierarchical data models in widespread use
- **Ted Codd defines the relational data model**
  - Would win the ACM Turing Award for this work
  - IBM Research begins System R prototype
  - UC Berkeley (Michael Stonebreaker) begins Ingres prototype
  - Oracle releases first commercial relational database
- High-performance (for the era) transaction processing

History of Database Systems (Cont.)



View of Data

- A major purpose of a database system is to provide users with an abstract view of the data.
  - Data models
    - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - Data abstraction
    - Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

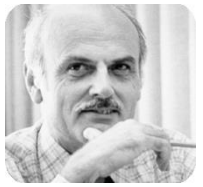


Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Many different data models
  - Relational model
  - Entity-Relationship data model (mainly for database design)
  - Object-based data models (Object-oriented and Object-relational)
  - Semi-structured data model (XML)
  - Other older models:
    - Network model
    - Hierarchical model

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



Ted Codd
Turing Award 1981

Columns

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Rows

(a) The *instructor* table

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

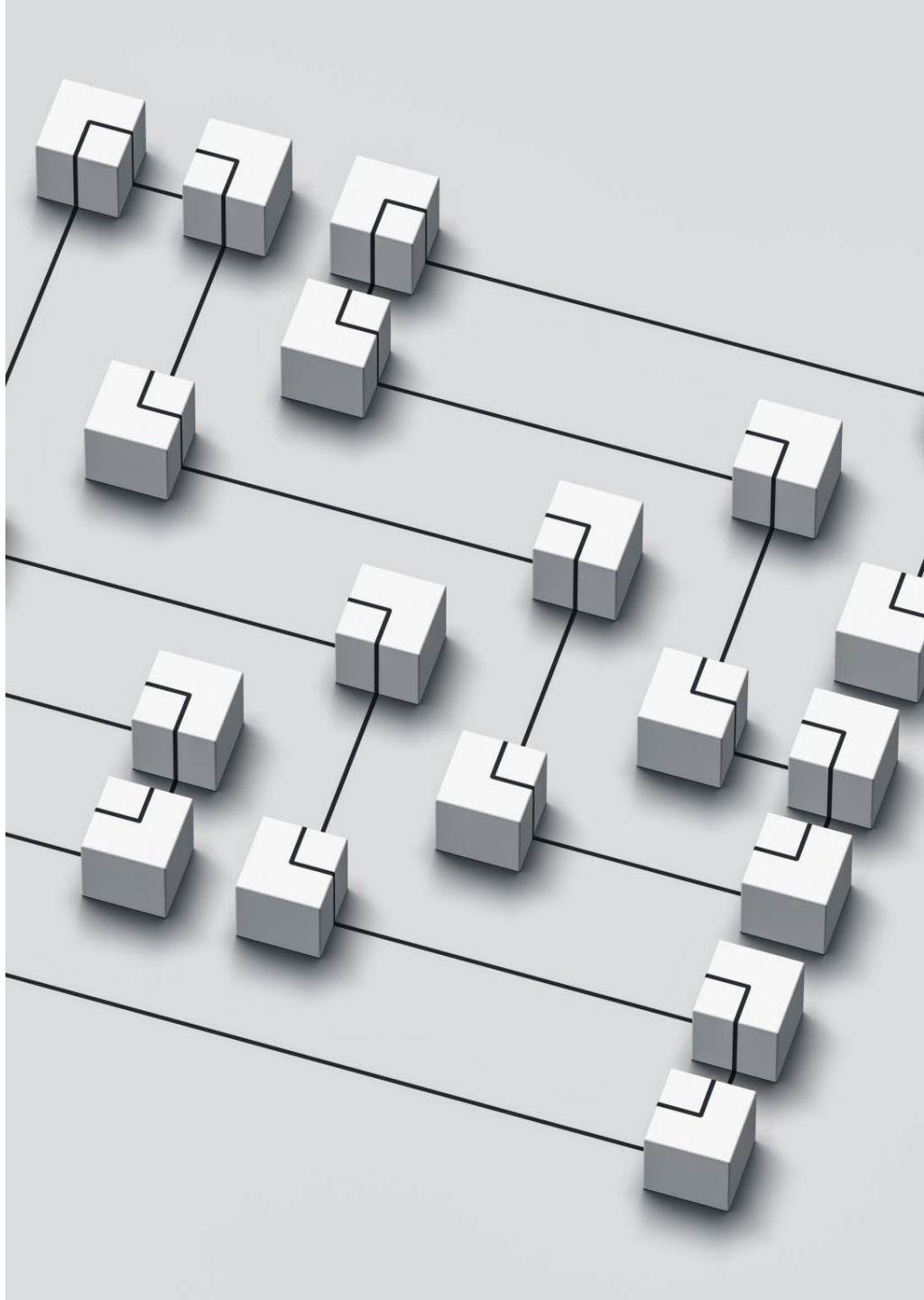
(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Schemas

- Logical Schema – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- Physical schema – the overall physical structure of the database e.g. how the data is stored physically.



Data Definition Language (DDL)

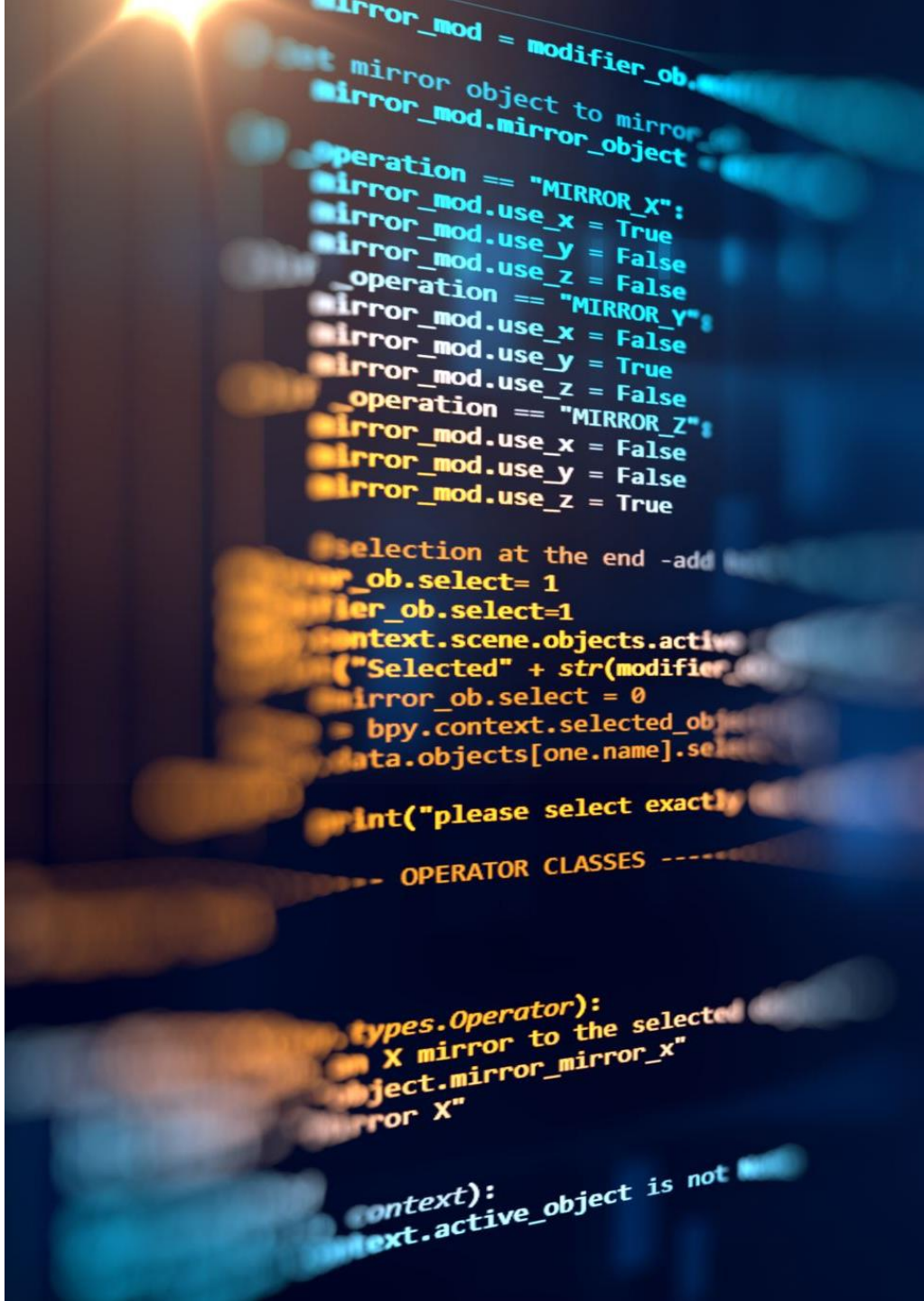
- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates that are stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (example -- ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as **query language**
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language



Data Manipulation Language (Cont.)

- There are basically two types of data-manipulation language
  - Procedural DML -- require a user to specify what data are needed and how to get those data.
  - Declarative DML -- require a user to specify what data are needed without specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a **query** language.

SQL Query Language

SQL is a query language which is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table.

Example to find the name of all instructors in Comp. Sci. dept

```
select name
        from instructor
        where
dept_name = 'Comp. Sci.'
```

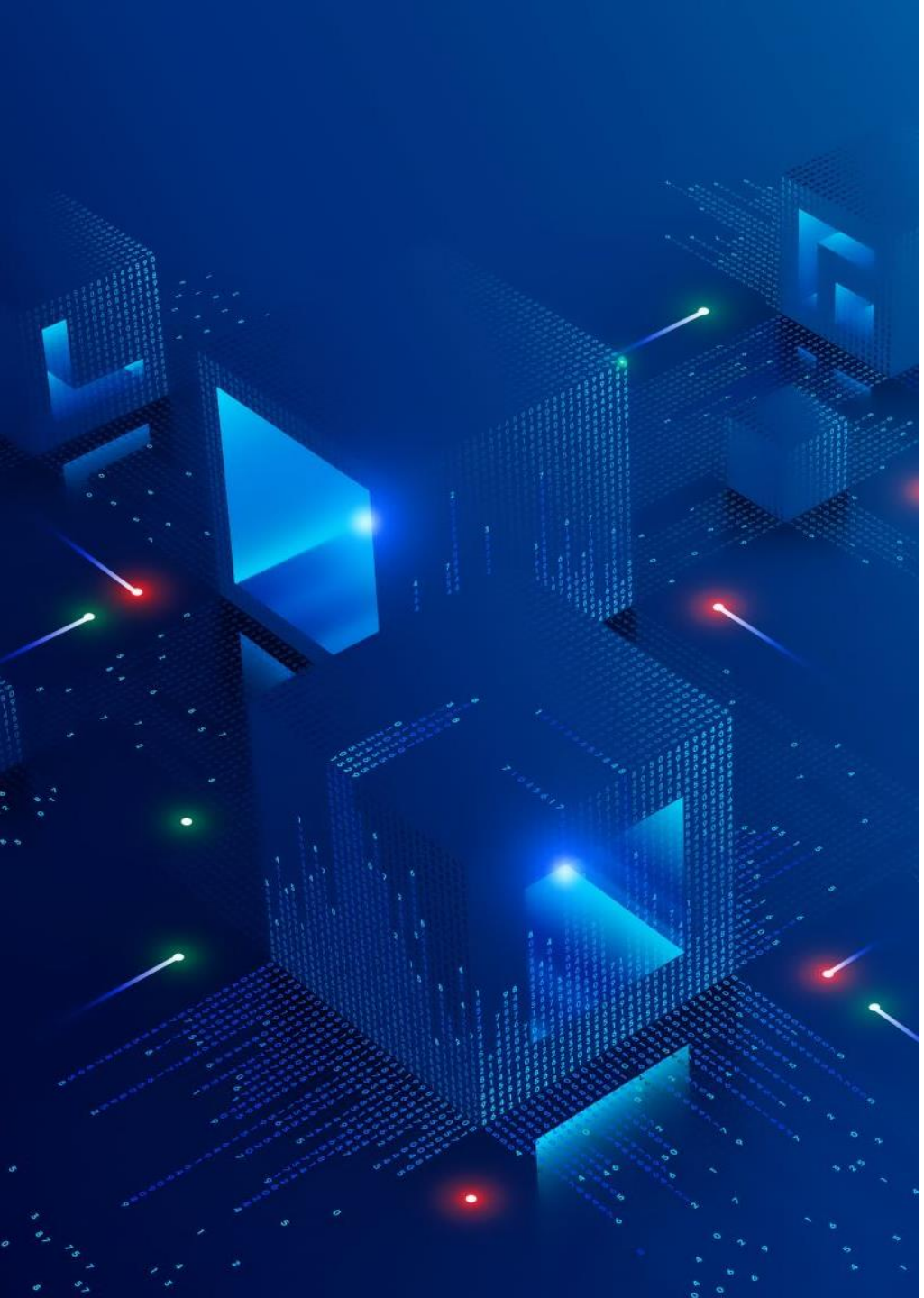
SQL is **NOT** a Turing machine equivalent language

Database Access from Application Program

SQL does not support actions such as input from users, output to displays, or communication over the network.

Such computations and actions must be written in a **host language**, such as C, C++, Java, or Python, with embedded SQL queries that access the data in the database.

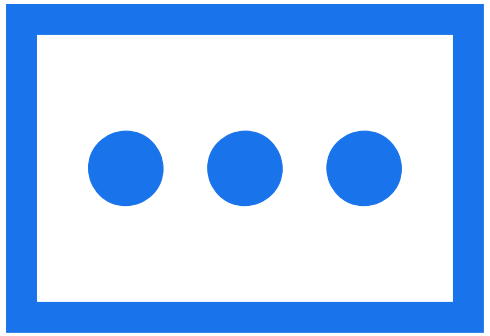
Application programs -- are programs that are used to interact with the database in this fashion.



Database Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be divided into
  - The storage manager,
  - The query processor component,
  - The transaction management component.

Storage Manager



- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager (data on disk)
  - Buffer manager (data in main memory)

Storage Manager (Cont.)

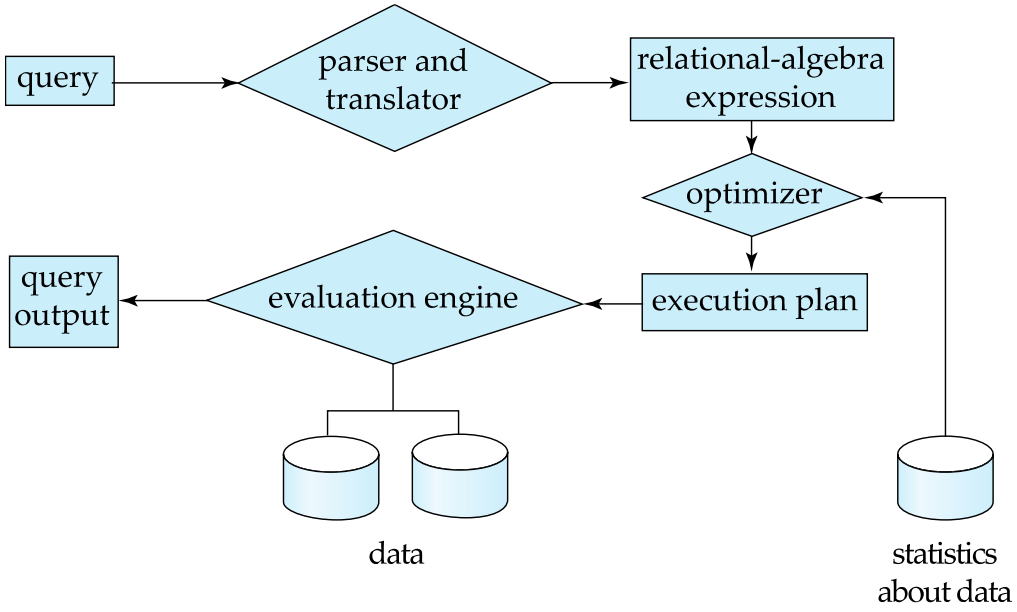
- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.

Query Processor Components

- DDL interpreter -- interprets DDL statements and records the definitions in the data dictionary.
- DML compiler -- translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
  - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
- Query evaluation engine -- executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Transaction Management

A **transaction** is a collection of operations that performs a single logical function in a database application

Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Architecture

Centralized databases

- One to a few cores, shared memory

Client-server

- One server machine executes work on behalf of multiple client machines.

Parallel databases

- Many core shared memory
- Shared disk
- Shared nothing

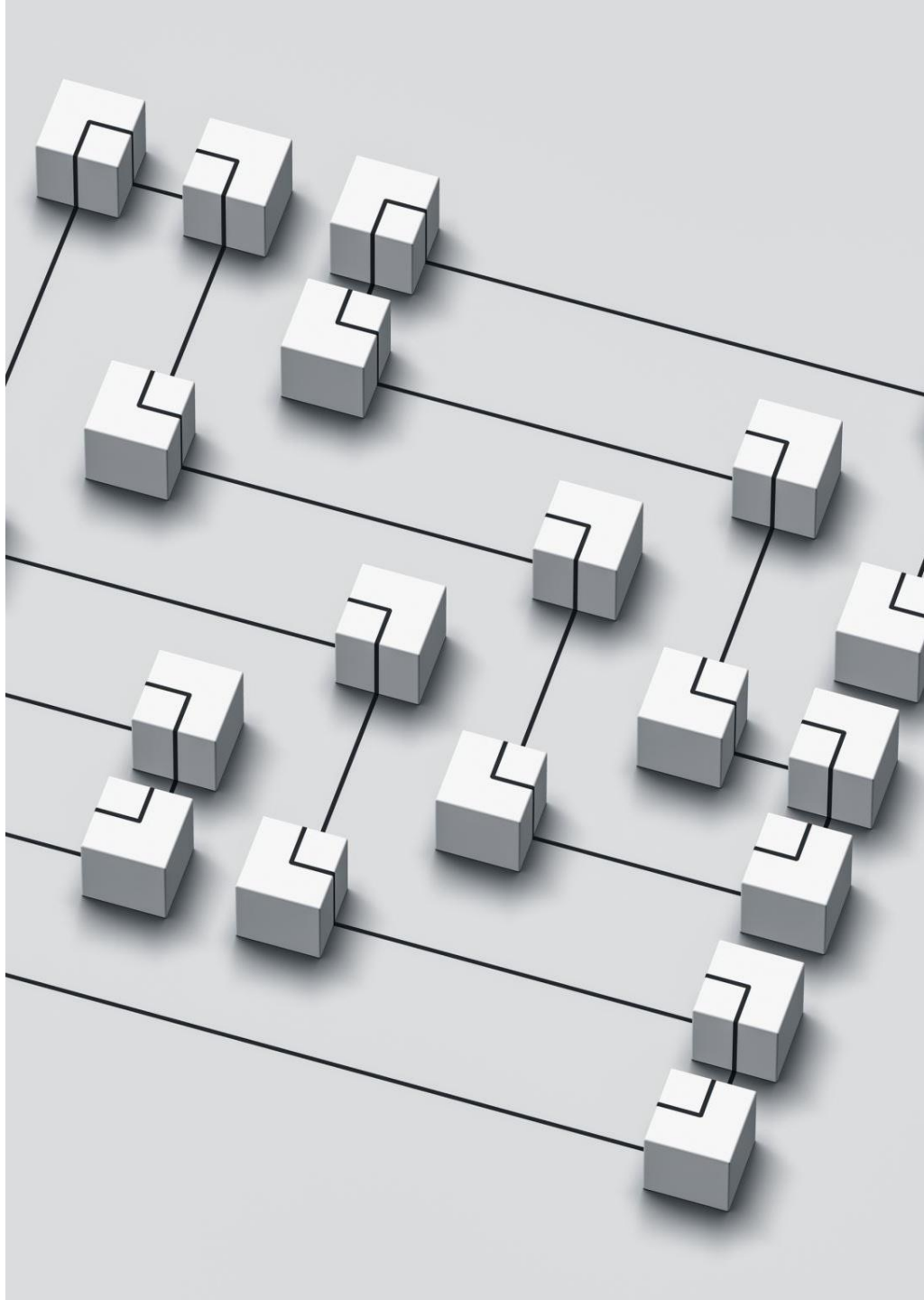
Distributed databases

- Geographical distribution
- Schema/data heterogeneity

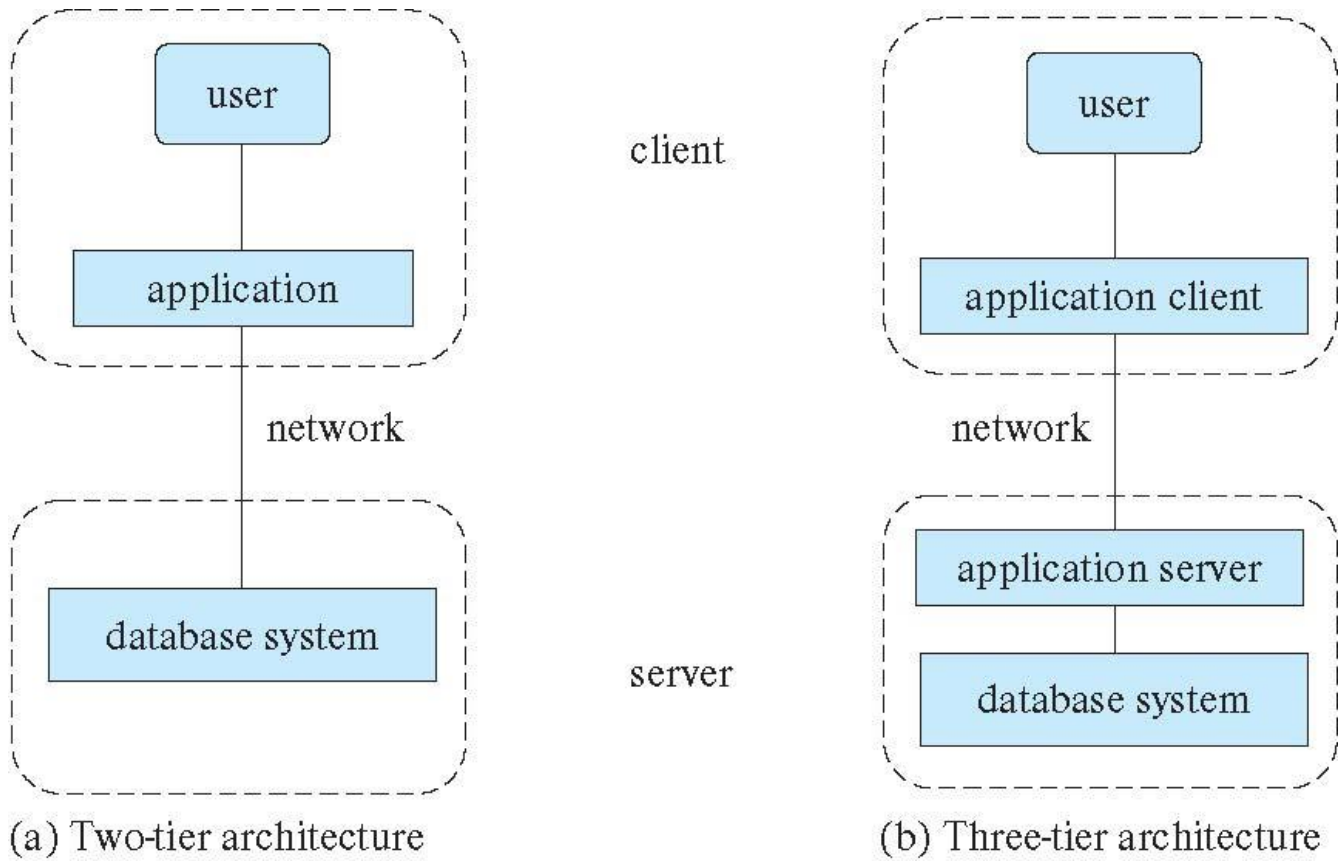
Database Applications

Database applications are usually partitioned into two or three parts

- Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine
- Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.
  - The client end communicates with an application server, usually through a forms interface.
  - The application server in turn communicates with a database system to access data.



Two-tier and three-tier architectures



(a) Two-tier architecture

(b) Three-tier architecture

Database Users

There are four different types of database-system users

Naive users -- unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

Application programmers -- computer professionals who write application programs.

Sophisticated users -- interact with the system without writing programs

- Using a database query language or by
- Using tools such as data analysis software.

Specialized users --write specialized database applications that do not fit into the traditional data-processing framework. For example, CAD, graphic data, audio, video.

Database Administrator

A person who has central control over the system is called a **database administrator** (**DBA**), whose functions are:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

Conclusion

- THANK YOU and See you next week!
- Got any questions later?

  Send me an email: dr.ercan.ezin@gmail.com