# CS4.301 Data & Applications

Ponnurangam Kumaraguru (“PK”)
#ProfGiri @ IIIT Hyderabad

 pk.profgiri  /in/ponguru  @ponguru  Ponnurangam.kumaraguru

# Protocol

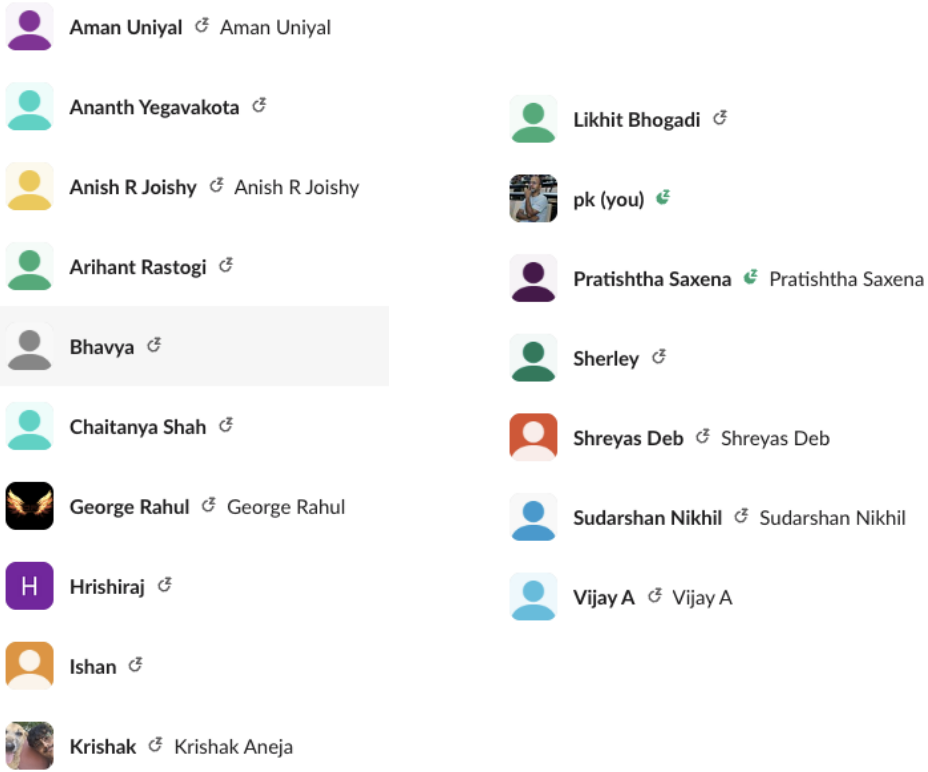

# Grading, Relative

| Type of Evaluation | Weightage (in %) |
|---|---|
| Class Quizzes (3) | 45 |
| Assignments / Project | 25 |
| End Sem Exam | 30 |

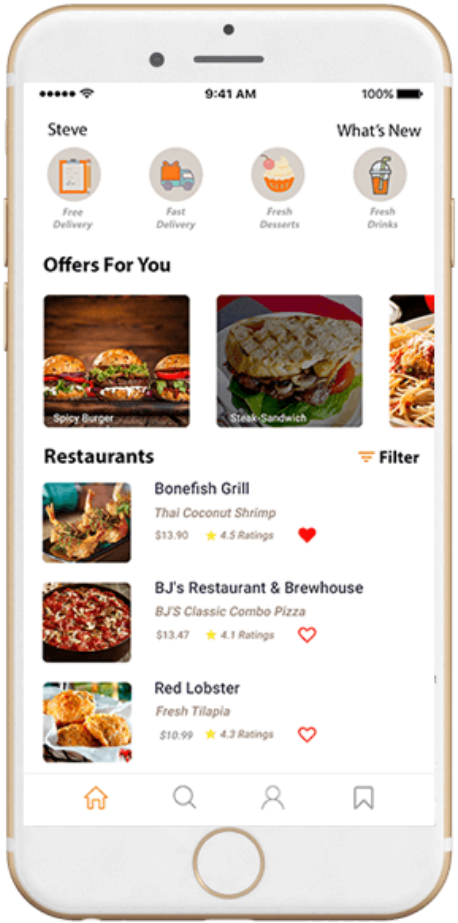
# TAs

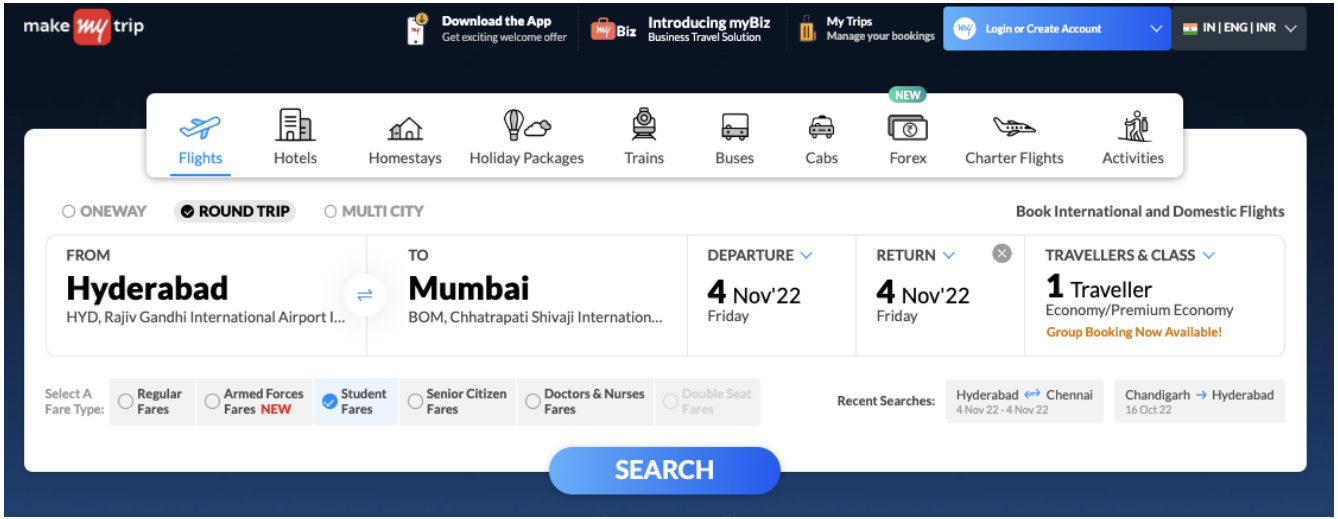
16 TAs

Students will be assigned among TAs for all evaluations



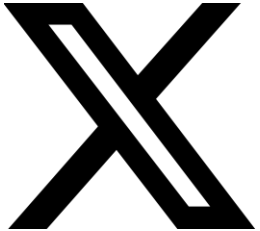

Memory challenge for me ☺




What kind of data are we generating here?






What kind of data are we generating here?

# Types of Databases and Database Applications

Traditional Applications:

- Numeric and Textual Databases

More Recent Applications:

- Multimedia Databases
- Geographic Information Systems (GIS)
- Biological and Genome Databases
- Data Warehouses
- Mobile databases
- Real-time and Active Databases
- Anything more?

# Developments

Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:

- Facebook

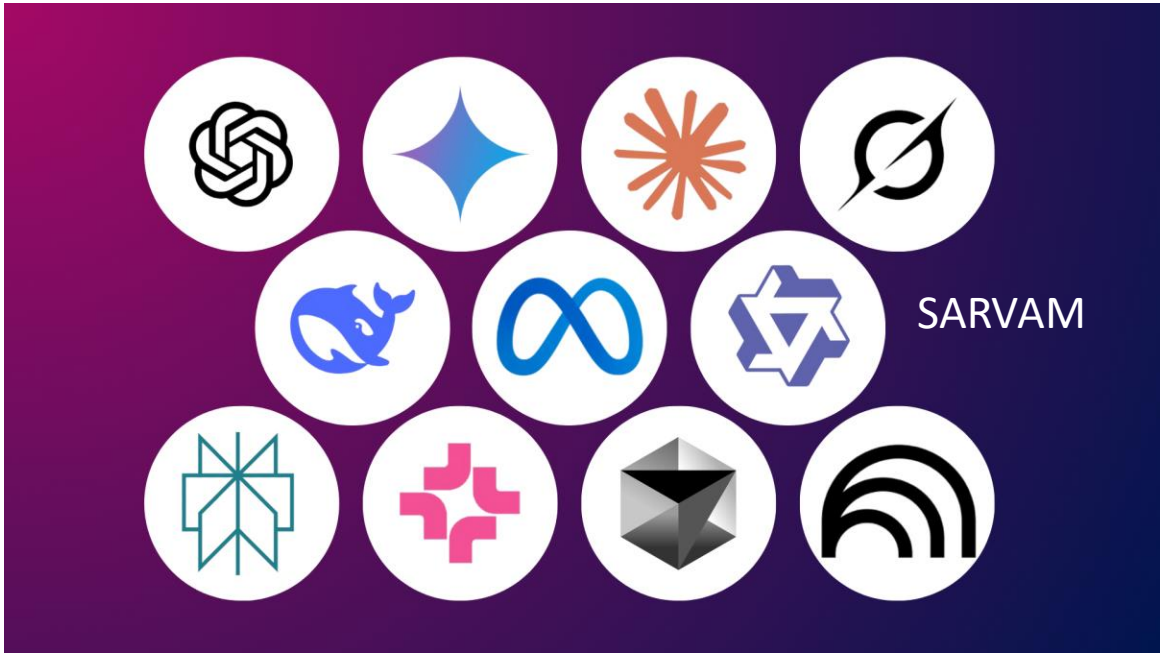
- Twitter

- LinkedIn

All of the above constitutes data

Search Engines, Google, Bing, Yahoo: collect their own repository of web pages for searching purposes

# Developments



# Basic Definitions

**Database:**

A collection of related data.

**Data:**

Known facts that can be recorded and have an implicit meaning.

**Mini-world:**

Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.

**Database Management System (DBMS):**

A software package/ system to facilitate the creation and maintenance of a computerized database.

**Database System:**

The DBMS software together with the data itself. Sometimes, the applications are also included.

# Impact of Databases and Database Technology

Businesses: Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing

Service Industries: Financial, Real-estate, Legal, Electronic Commerce, Small businesses

Education : Resources for content and Delivery

More recently: Social Networks, Environmental and Scientific Applications, Medicine and Genetics, LLMs, VLMs

Personalized Applications: based on smart mobile devices

# Typical DBMS Functionality

*Define* a particular database in terms of its data types, structures, and constraints

*Construct* or Load the initial database contents on a secondary storage medium

*Manipulating* the database:

- Retrieval: Querying, generating reports
- Modification: Insertions, deletions and updates to its content
- Accessing the database through Web applications

*Processing* and *Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

# Application Activities Against a Database

Applications interact with a database by generating

- Queries: that access different parts of data and formulate the result of a request
- Transactions: that may read some data and “update” certain values or generate new data and store that in the database

Applications must not allow unauthorized users to access data

Applications must keep up with changing user requirements against the database

# Additional DBMS Functionality

DBMS may additionally provide:

- Protection or Security measures to prevent unauthorized access
- “Active” processing to take internal actions on data
- Presentation and Visualization of data
- Maintenance of the database and associated programs over the lifetime of the database application
  - Called database, software, and system maintenance

# What is a Database?

Data: factual (undoubted) information that can be recorded and have implicit meaning

A database is a collection of related data

# What is a Database?

A database has the following implicit properties:

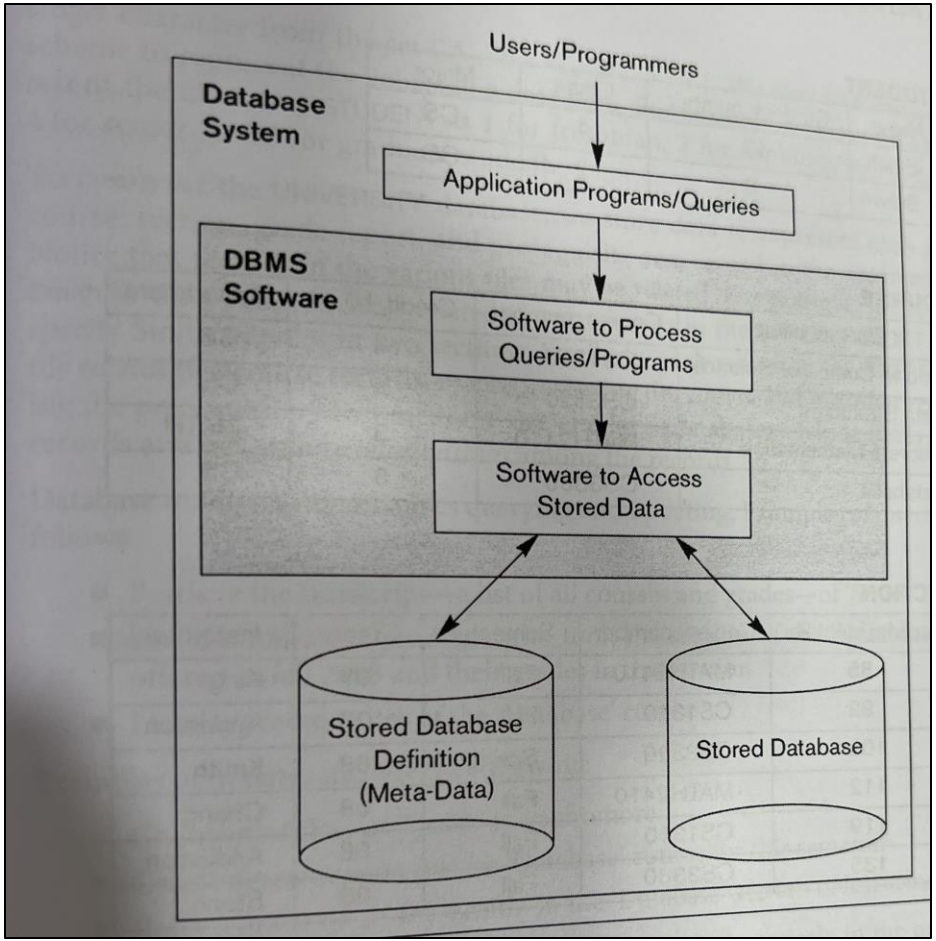
A database represents some aspect of the real world (mini-world or Universe of Discourse (UoD))

A database is a logically coherent (associated, related) collection of data with some inherent meaning

A database is designed, built and populated with data for a specific purpose

It has an intended group of users and some preconceived (already thought of) applications in which these users are interested

# Simplified Database




Book

# Example of Database: University

Data:

STUDENTs

COURSEs

SECTIONs (of COURSEs)

(academic) DEPARTMENTs

INSTRUCTORs

Relation:

SECTIONs are of specific COURSEs

STUDENTs take SECTIONs

COURSEs have prerequisite COURSEs

INSTRUCTORs teach SECTIONs

COURSEs are offered by DEPARTMENTs

STUDENTs major in DEPARTMENTs

# This Lecture


Any Questions?

# Example of a simple database

**COURSE**

| Course_name | Course_number | Credit_hours | Department |
|---|---|---|---|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Discrete Mathematics | MATH2410 | 3 | MATH |
| Database | CS3380 | 3 | CS |

**SECTION**

| Section_identifier | Course_number | Semester | Year | Instructor |
|---|---|---|---|---|
| 85 | MATH2410 | Fall | 04 | King |
| 92 | CS1310 | Fall | 04 | Anderson |
| 102 | CS3320 | Spring | 05 | Knuth |
| 112 | MATH2410 | Fall | 05 | Chang |
| 119 | CS1310 | Fall | 05 | Anderson |
| 135 | CS3380 | Fall | 05 | Stone |

**GRADE_REPORT**

| Student_number | Section_identifier | Grade |
|---|---|---|
| 17 | 112 | B |
| 17 | 119 | C |
| 8 | 85 | A |
| 8 | 92 | A |
| 8 | 102 | B |
| 8 | 135 | A |

**PREREQUISITE**

| Course_number | Prerequisite_number |
|---|---|
| CS3380 | CS3320 |
| CS3380 | MATH2410 |
| CS3320 | CS1310 |

**Figure 1.2**
A database that stores student and course information.

# Example of a simplified database catalog

**RELATIONS**

| Relation_name | No_of_columns |
|---|---|
| STUDENT | 4 |
| COURSE | 4 |
| SECTION | 5 |
| GRADE_REPORT | 3 |
| PREREQUISITE | 2 |

**COLUMNS**

| Column_name | Data_type | Belongs_to_relation |
|---|---|---|
| Name | Character (30) | STUDENT |
| Student_number | Character (4) | STUDENT |
| Class | Integer (1) | STUDENT |
| Major | Major_type | STUDENT |
| Course_name | Character (10) | COURSE |
| Course_number | XXXXNNNN | COURSE |
| …. | …. | ….. |
| …. | …. | ….. |
| …. | …. | ….. |
| Prerequisite_number | XXXXNNNN | PREREQUISITE |

*Note*: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

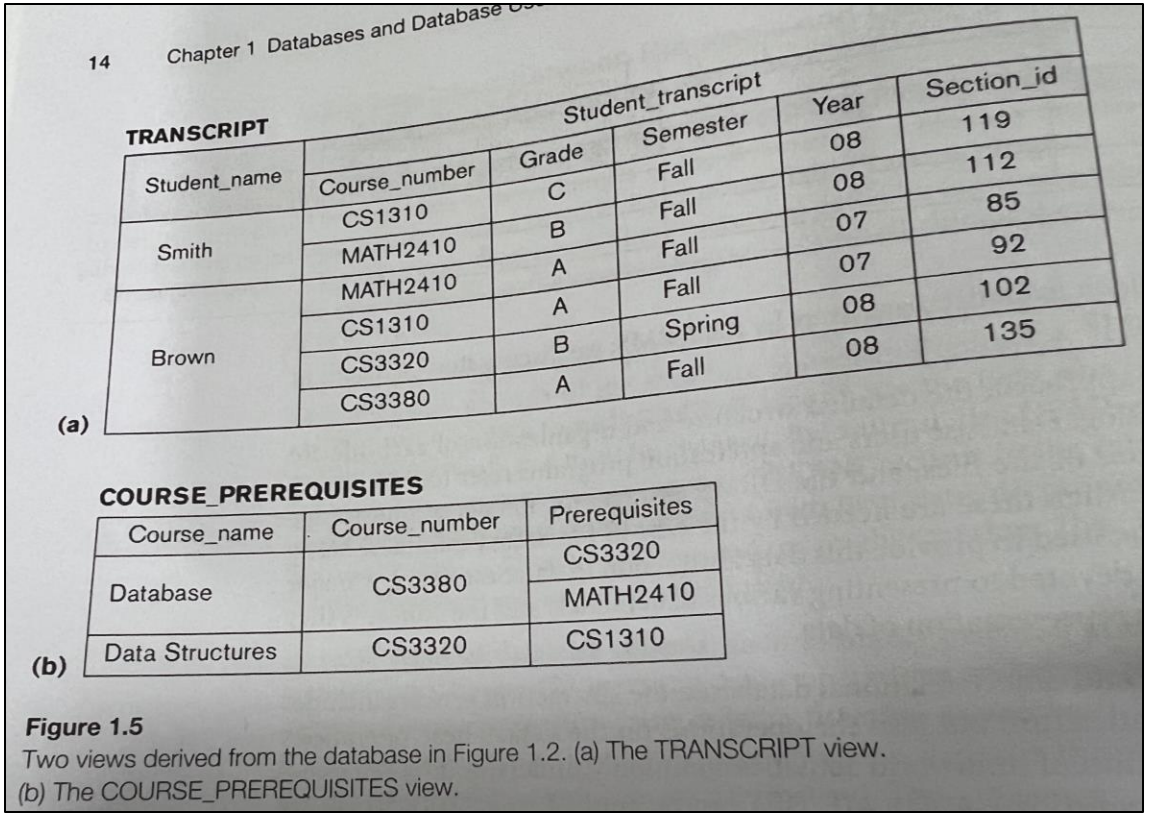
**Figure 1.3**
An example of a database catalog for the database in Figure 1.2.

# Views

Many users to DB

Each users may require a different view

View may be a subset or virtual data derived





**TRANSCRIPT**

| Student_name | Student_transcript | | | | |
|---|---|---|---|---|---|
| | Course_number | Grade | Semester | Year | Section_id |
| Smith | CS1310 | C | Fall | 08 | 119 |
| | MATH2410 | B | Fall | 08 | 112 |
| Brown | MATH2410 | A | Fall | 07 | 85 |
| | CS1310 | A | Fall | 07 | 92 |
| | CS3320 | B | Spring | 08 | 102 |
| | CS3380 | A | Fall | 08 | 135 |

(a)

**COURSE_PREREQUISITES**

| Course_name | Course_number | Prerequisites |
|---|---|---|
| Database | CS3380 | CS3320 |
| | | MATH2410 |
| Data Structures | CS3320 | CS1310 |

(b)

***Figure 1.5***

*Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.*

# Online Transaction Processing (OLTP)

Multiuser DB

Concurrency control

Flight ticket booking, seats available

# Transaction

Executing program or process that includes one or more database accesses, reading or updating of database records

Properties [ACID]

- Atomicity: either all are executed or none are executed [A/c A → A/c B]
- Consistency: any data written to a DB must be valid according to the defined rules [telephone number]
- Isolation: each transaction appears to execute in isolation, even though 100s may be executing at the same time [updating the seat preference]
- Durability: guarantees that once a transaction has been committed, it will remain committed even in the case of a system failure

# Actors on the Scene: Day-to-Day use of DB

Database administrators

- authorizing access to DB, coordinating & monitoring its use, accountable for security breaches & response time

Database designers

- responsible for identifying the data to be stored in the DB, interact with potential group of users and develop *views* of the DB

End Users: Casual, naïve / parametric, sophisticated, stand-alone users

- Casual: occasional users, typically middle or high-level managers
- Naïve / parametric: constantly updating the db using *canned transaction,* done using mobile apps
  - bank tellers checking balances post withdrawals & deposits
  - reservation agents checking for availability
  - social media users post and read items on platforms

# Actors on the Scene: Day-to-Day use of DB

End Users: Casual, naïve, sophisticated, stand-alone users

- sophisticated: thoroughly familiarize themselves with all facilities of DBMS, implement their own, complex requirements
- stand-alone: maintain personal DB using ready-made programs; TALLY

System analysts & application programmers

- determine the requirements of end-users, including naïve, develop specifications for canned transactions
- map implement above specifications as programs, they test – debug maintain these canned transactions
- software developers / engineers play these roles sometimes

# Actors Behind the Scene: Maintain the DB

DBMS designers & implementers

design and implement the DBMS modules; complex modules like query language processing, interface processing, controlling concurrency, handling data recovery & security

Tool developers

design & implement tools; optional packages that are often purchased separately; facilitate DB modeling & design, system design, and improved performance

Operators & maintenance personnel

responsible for running & maintenance of the hardware & software environment for DB

# Advantages of using DBMS approach

Controlling redundancy

redundancy in storing the same data multiple times e.g. student details in university maintained by acad & finance office separately

duplication of efforts, storage space, inconsistent data

ideally student details in only one place, *data normalization*

keeping all needed data together, *denormalization*

# Advantages of using DBMS approach

Restricting unauthorized access

> your grades accessible to only some; my salary and personal details only to some [hopefully ☺]

Providing storage structures and search techniques for efficient query processing

> efficiently executing queries & updates; creating *indexes* and maintaining it; *buffering* & *caching* modules

# Advantages of using DBMS approach

Providing backup & recovery

- provide facilities for recovering from hardware & software failures
- complex updates, should not crash; if crash what state to recover

Providing multiple user interfaces

- apps for mobile users; query language for causal; programming language for application programmers; forms / command codes for parametric; menu & natural language interfaces for standalone

# Advantages of using DBMS approach

Representing Complex relationship among data

DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve / update related data easily & efficiently

Enforcing integrity constraints

student name: 30 alphabetic characters; record in one file must be related to records in other files [e.g. every SECTION record must be related to a COURSE record] *referential integrity*

uniqueness on data item values [e.g. every COURSE record must have a unique value for COURSE_NUMBER] *key or uniqueness constraint*

# Advantages of using DBMS approach

Permitting inferencing and actions using rules and triggers

- triggers associated with tables; trigger is a rule activated by updates to the table results in performing some addition operations to other tables, sending messages, etc.
- stored procedures are invoked appropriately when some conditions are met

# Historical Development of Database Technology

Early Database Applications:

- The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
- A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.

Relational Model based Systems:

- Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
- Relational DBMS Products emerged in the early 1980s.

Object-oriented and emerging applications:

- Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
  - Their use has not taken off much.
- Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relationa*l DBMSs (ORDBMSs)
- *Extended relational* systems add further capabilities (e.g. for multimedia data, text, XML, and other data types)

# Historical Development of Database Technology (continued)

Web contains data in HTML (Hypertext markup language) with links among pages.

This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language). (see Ch. 13).

Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database (see Ch. 11).

- Also allow database updates through Web pages

Social Media platforms such as Facebook and Twitter are generating millions of transactions a day and businesses are interested to tap into this data to “understand” the users

Cloud Storage and Backup is making unlimited amount of storage available to users and applications

“Big Data”, Hadoop, Mapreduce, Spark based technology.

NOSQL -- systems have been designed for rapid search and retrieval from documents, processing of huge graphs occurring on social networks, and other forms of unstructured data with flexible models of transaction processing (Chapter 24).

# When not to use a DBMS

Main inhibitors (costs) of using a DBMS:

- High initial investment and possible need for additional hardware.
- Overhead for providing generality, security, concurrency control, recovery, and integrity functions.

When a DBMS may be unnecessary:

- If the database and applications are simple, well defined, and not expected to change.
- If access to data by multiple users is not required.

When a DBMS may be infeasible:

- In embedded systems where a general purpose DBMS may not fit in available storage

# Data Models

**Data Model:**

A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.

**Data Model Structure and Constraints:**

Constructs are used to define the database structure

Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups

Constraints specify some restrictions on valid data; these constraints must be enforced at all times

# Data Models (continued)

**Data Model Operations:**

These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.

Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

# Categories of Data Models

**Conceptual (high-level, semantic) data models:**

Provide concepts that are close to the way many users perceive data.

(Also called ***entity-based*** or ***object-based*** data models.)

**Physical (low-level, internal) data models:**

Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals

**Implementation (representational) data models:**

Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

**Self-Describing Data Models:**

Combine the description of data with the data values. Examples include XML, key-value stores and some NOSQL systems.

# Schemas

Database Schema:

The ***description*** of a database.

Includes descriptions of the database structure, data types, and the constraints on the database.

Schema Diagram:

An ***illustrative*** display of (most aspects of) a database schema.

Schema Construct:

A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

# Schemas

Database State:

- The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
- Also called database instance (or occurrence or snapshot).
  - The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

# Database Schema vs. Database State

Database State:

Refers to the ***content*** of a database at a moment in time.

Initial Database State:

Refers to the database state when it is initially loaded into the system.

Valid State:

A state that satisfies the structure and constraints of the database.

Distinction

The ***database schema*** changes very infrequently.

The ***database state*** changes every time the database is updated.

# Example of a Database Schema

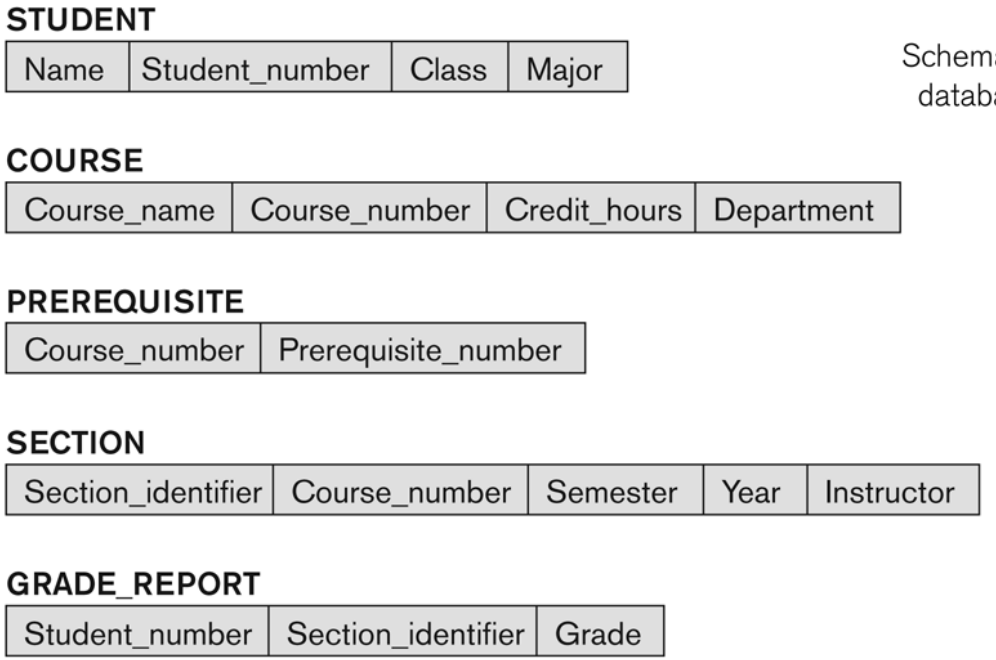



**Figure 2.1**
Schema diagram for the database in Figure 1.2.

**COURSE**

| Course_name | Course_number | Credit_hours | Department |
|---|---|---|---|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Discrete Mathematics | MATH2410 | 3 | MATH |
| Database | CS3380 | 3 | CS |

**SECTION**

| Section_identifier | Course_number | Semester | Year | Instructor |
|---|---|---|---|---|
| 85 | MATH2410 | Fall | 04 | King |
| 92 | CS1310 | Fall | 04 | Anderson |
| 102 | CS3320 | Spring | 05 | Knuth |
| 112 | MATH2410 | Fall | 05 | Chang |
| 119 | CS1310 | Fall | 05 | Anderson |
| 135 | CS3380 | Fall | 05 | Stone |

**GRADE_REPORT**

| Student_number | Section_identifier | Grade |
|---|---|---|
| 17 | 112 | B |
| 17 | 119 | C |
| 8 | 85 | A |
| 8 | 92 | A |
| 8 | 102 | B |
| 8 | 135 | A |

**PREREQUISITE**

| Course_number | Prerequisite_number |
|---|---|
| CS3380 | CS3320 |
| CS3380 | MATH2410 |
| CS3320 | CS1310 |

**Figure 1.2**
A database that stores student and course information.

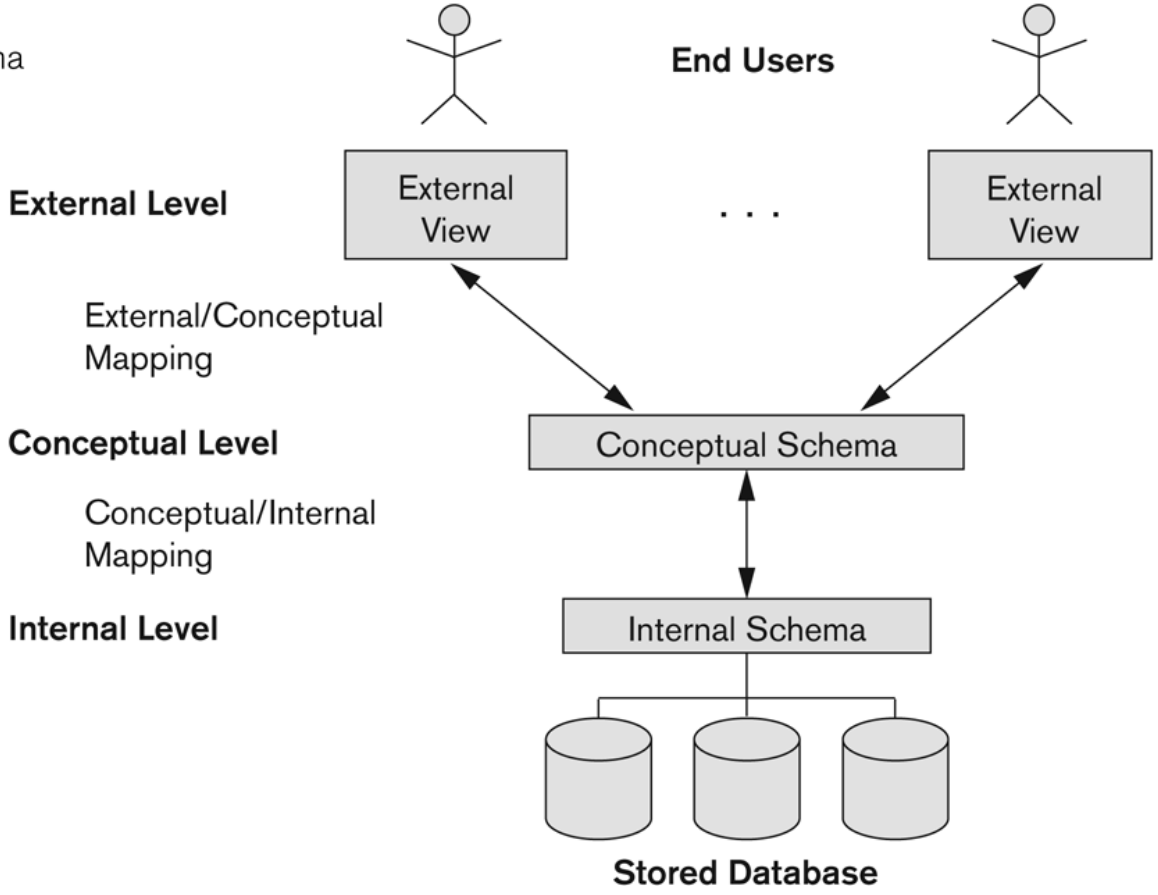
# Example of a database state

# Three-Schema Architecture

Defines DBMS schemas at ***three*** levels:

- **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
  - Typically uses a **physical** data model.
- **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
  - Uses a **conceptual** or an **implementation** data model.
- **External schemas** at the external level to describe the various user views.
  - Usually uses the same data model as the conceptual schema.

**Figure 2.2**
The three-schema architecture.




# The three-schema architecture

# Typical DBMS Component Modules

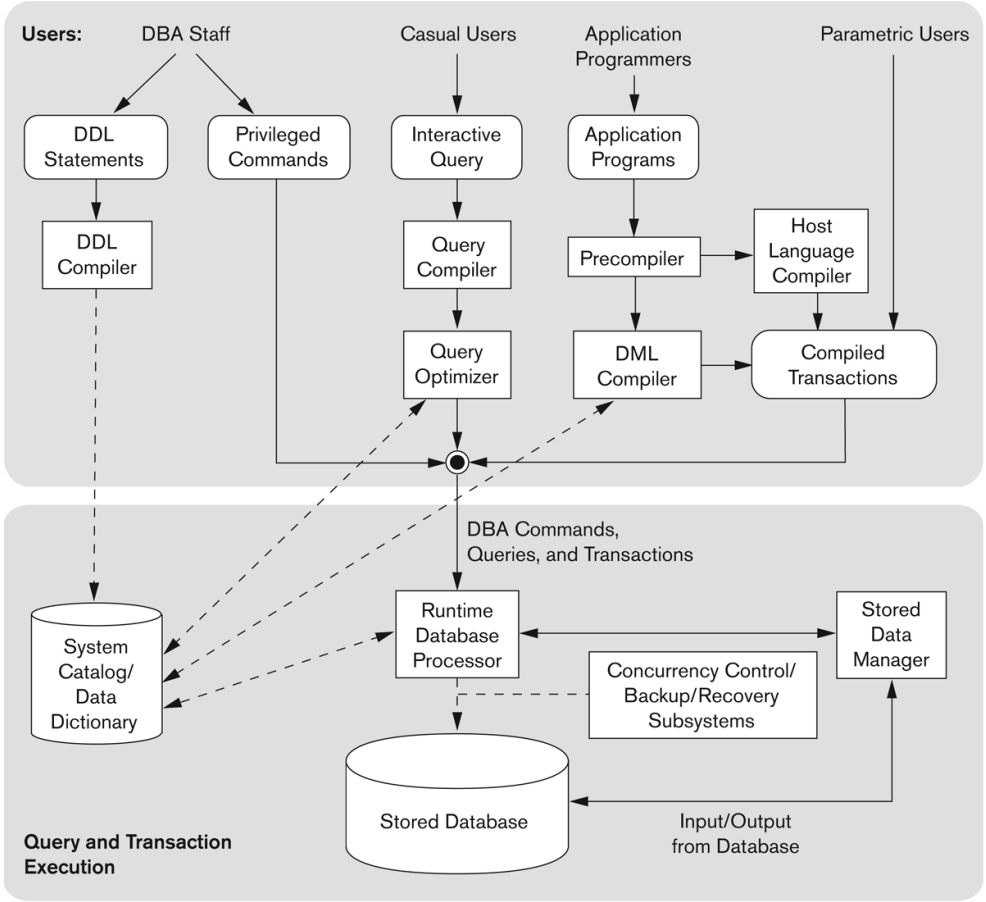



**Figure 2.3**
Component modules of a DBMS and their interactions.

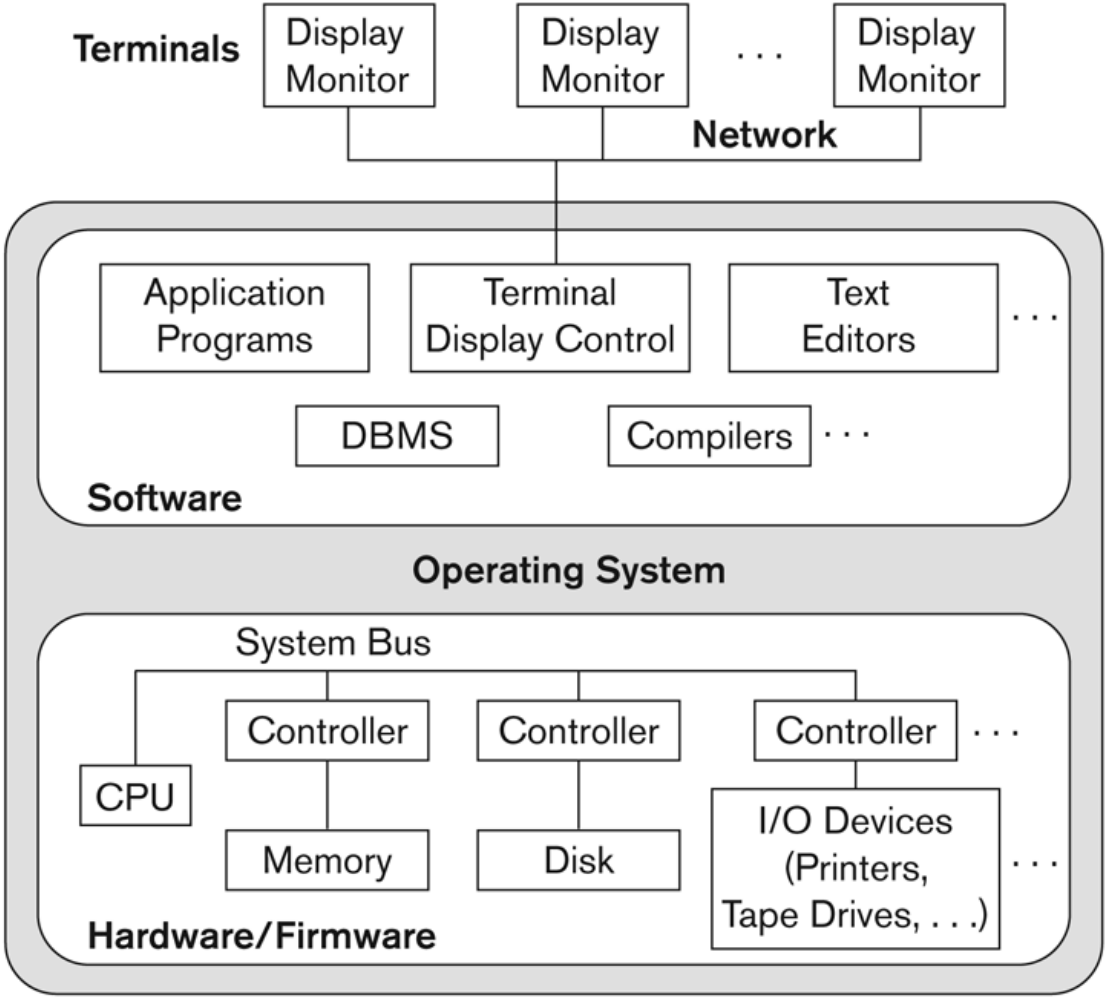



**Figure 2.4**
A physical centralized architecture.

# Logical two-tier client server architecture

**Figure 2.5**
Logical two-tier client/server architecture.

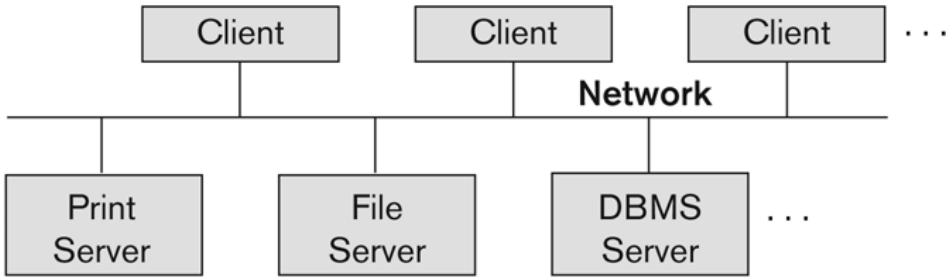
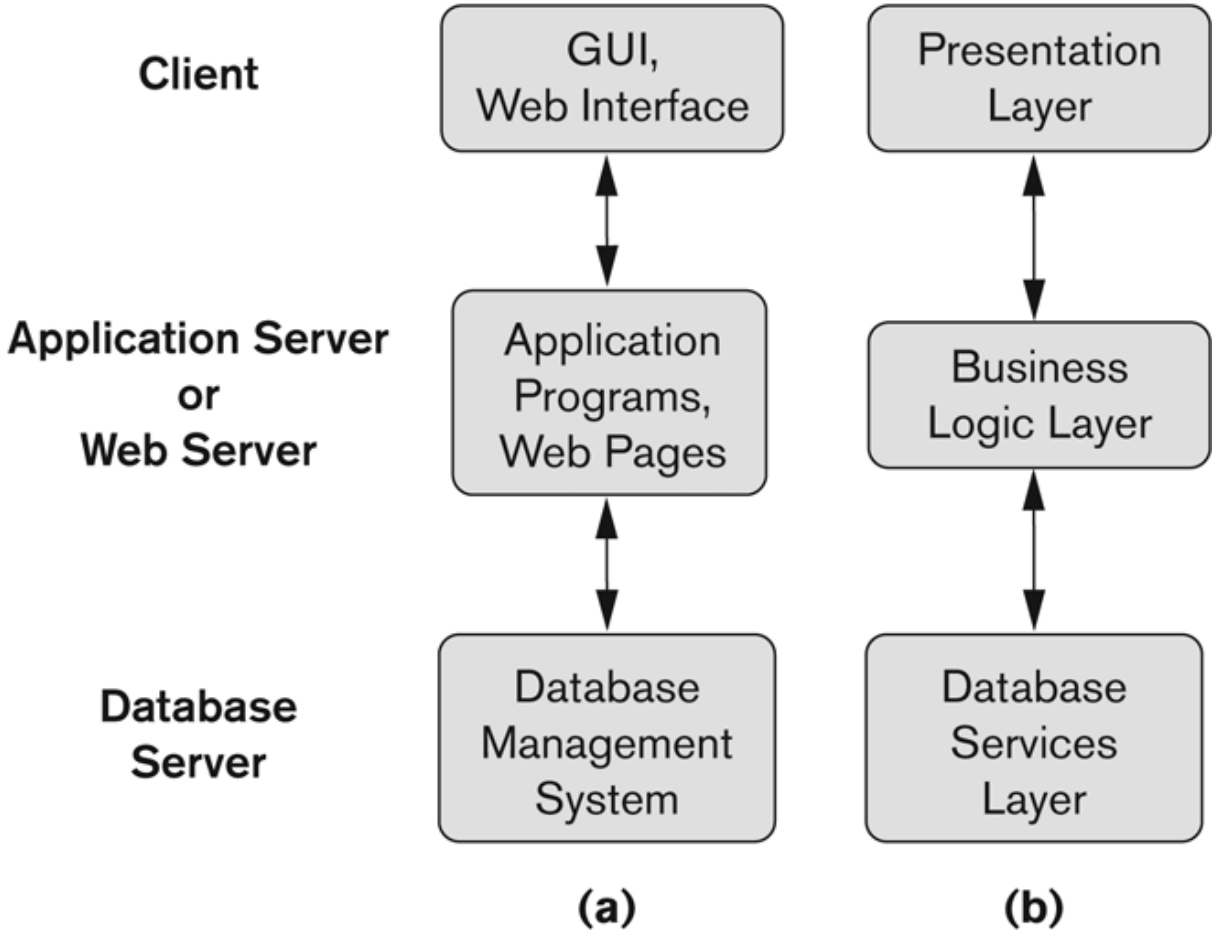


**Figure 2.7**
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



# DBMS Languages

Data Definition Language (DDL)

Data Manipulation Language (DML)

# DBMS Languages

**Data Definition Language (DDL):**

Used by the DBA and database designers to specify the conceptual schema of a database.

In many DBMSs, the DDL is also used to define internal and external schemas (views).

In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.

SDL is typically realized via DBMS commands provided to the DBA and database designers

# DBMS Languages

**Data Manipulation Language (DML):**

- Used to specify database retrievals and updates
- DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
  - A library of functions can also be provided to access the DBMS from a programming language
- Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

# Types of DML

**High Level or Non-procedural Language:**

- For example, the SQL relational language
- Are “set”-oriented and specify what data to retrieve rather than how to retrieve it.
- Also called **declarative** languages.
- QBE – Query By Example

**Low Level or Procedural Language:**

- Retrieve data one record-at-a-time;
- Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.
- COBOL / FORTRAN

# Any questions?

# Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in