

CS4.301 Data & Applications

Ponnurangam Kumaraguru ("PK")
#ProfGiri @ IIIT Hyderabad

Lecture 2



pk.profgiri



/in/ponguru



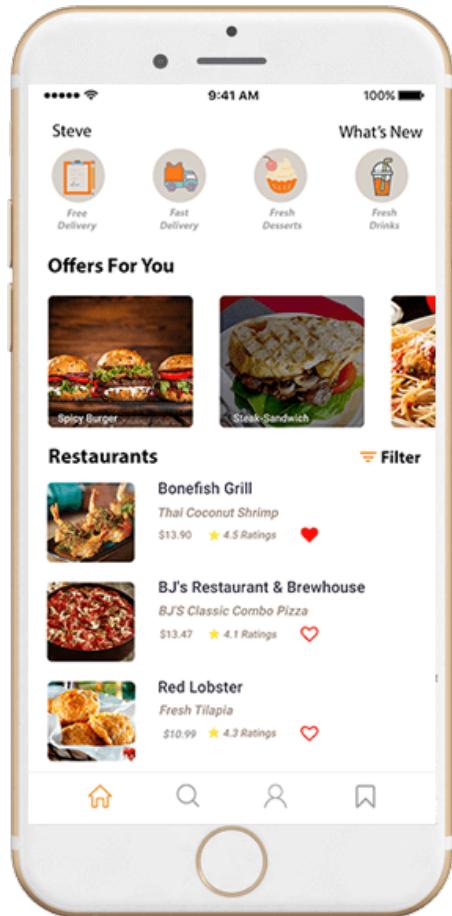
@ponguru



Ponnurangam.kumaraguru

Protocol





make my trip

Download the App
Get exciting welcome offer

myBiz Introducing myBiz Business Travel Solution

My Trips Manage your bookings

Login or Create Account

IN | ENG | INR

Flights Hotels Homestays Holiday Packages Trains Buses Cabs Forex Charter Flights Activities

ONEWAY ROUND TRIP MULTICITY

FROM TO DEPARTURE RETURN TRAVELLERS & CLASS

Hyderabad Mumbai 4 Nov'22 4 Nov'22 1 Traveller

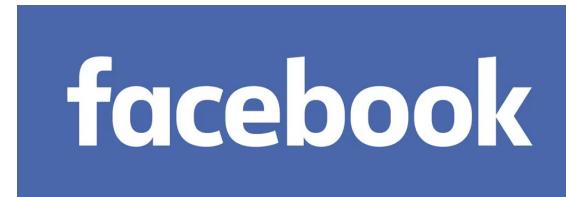
HYD, Rajiv Gandhi International Airport I... BOM, Chhatrapati Shivaji Internation... Friday Friday Economy/Premium Economy Group Booking Now Available!

Select A Fare Type: Regular Fares Armed Forces Fares NEW Student Fares Senior Citizen Fares Doctors & Nurses Fares Double Seat Fares

Recent Searches: Hyderabad → Chennai 4 Nov 22 - 4 Nov 22 Chandigarh → Hyderabad 16 Oct 22

SEARCH

What kind of data are we generating here?



What kind of data are we generating here?

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.

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

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

Database catalogue

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 enumerated type with all known majors.
XXXXNNNN is used to define a type with four alphabetic characters followed by four digits.

Book

Views

Many users to DB

Each user may require a different view

View may be a subset or virtual data derived

14 Chapter 1 Databases and Database Views

(a) TRANSCRIPT

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

(b) COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

Figure 1.5
Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view.
(b) The COURSE_PREREQUISITES view.

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

ap implement above specifications as programs, they test – debug – maintain these canned transactions

software developers / engineers play these roles sometimes

Workers 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 [Jan-19- 1998 vs Jan-29-1998]

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

keeping all ~~needed~~ data together, *denormalization*

Restricting unauthorized access

This Lecture

Late submission

8 late days you can use

20% reduction beyond this 8 days

Maximum of 2 late days in one submission

Administrativia

Tutorial: 1st Oct 1130 – 1300hrs, H205

1st Quiz: 10th Oct

Email ID for any questions: dna-m22-tas@googlegroups.com

PK Office Hours: 1000 – 1030hrs, 5th floor - THub

Monday

TA hours

#	TA Name	Day	Time	Location
1	Venika Sruthi Annam	Monday	1600 – 1630	THub 5 th Floor
2	Devesh Marwah	Monday	1710 – 1740	H-303
3	Priyanshul Govil	Tuesday	1145 – 1215	THub 5 th Floor
4	Swetha Vipparla	Tuesday	1600 – 1630	Workspace
5	Abhishek Sharma	Wednesday	1615 – 1645	Workspace
6	Arvindh A	Wednesday	1700 – 1730	THub 5 th Floor
7	Soveet Kumar Nayak	Thursday	1700 – 1730	Workspace
8	Ayush Agrawal	Friday	1100 – 1130	Workspace
9	Sri Loukya Pappula	Friday	1600 – 1630	Workspace
10	Abhijeeth Singam	Saturday	1100 – 1130	THub 5 th Floor
11	Aashwin Vaish	Saturday	1600 – 1630	MT-NLP lab, LTRC

Your queries / SLA

1st to TAs, if needed you can CC me; you should get a response in 12hrs, if not you are welcome to remind or escalate

Our response could be, looking into it, confirming it with PK, etc.

You did not get a response in 12 hrs, you can escalate it to me and cc the TAs; again response time is 12 hrs or lesser

You got a response and not satisfied, please ask again to TAs

If still not satisfied, you are welcome to escalate, but keep in mind, you should have all the interactions with TA kept in the email to me, otherwise, will ignore

Advantages of using DBMS approach

Restricting unauthorized access

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

Providing persistent storage for program objects

complex object in C++ can be stored permanently in an object- oriented DBMS; this ability to be persistent and to be retrieved by programs later

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.

Historical Development of Database Technology (continued)

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-relational* 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)

Data on the Web and E-commerce Applications:

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

Extending Database Capabilities (1)

New functionality is being added to DBMSs in the following areas:

Scientific Applications – Physics, Chemistry, Biology - Genetics

Earth and Atmospheric Sciences and Astronomy

XML (eXtensible Markup Language)

Image Storage and Management

Audio and Video Data Management

Data Warehousing and Data Mining – a very major area for future development using new technologies (see Chapters 28-29)

Spatial Data Management and Location Based Services

Time Series and Historical Data Management

The above gives rise to *new research and development* in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

Extending Database Capabilities (2)

Background since the advent of the 21st Century:

First decade of the 21st century has seen tremendous growth in user generated data and automatically collected data from applications and search engines.

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

Extending Database Capabilities (3)

Emergence of Big Data Technologies and NOSQL databases

New data storage, management and analysis technology was necessary to deal with the onslaught of data in petabytes a day (10^{15} bytes or 1000 terabytes) in some applications – this started being commonly called as “Big Data”.

Hadoop (which originated from Yahoo) and Mapreduce Programming approach to distributed data processing (which originated from Google) as well as the Google file system have given rise to Big Data technologies (Chapter 25). Further enhancements are taking place in the form of Spark based technology.

NOSQL (Not Only SQL- where SQL is the de facto standard language for relational DBMSs) 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

When not to use a DBMS

When no DBMS may suffice:

If there are stringent real-time requirements that may not be met because of DBMS overhead (e.g., telephone switching systems)

If the database system is not able to handle the complexity of data because of modeling limitations (e.g., in complex genome and protein databases)

If the database users need special operations not supported by the DBMS (e.g., GIS and location based services).

Any questions?

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 versus Instances

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 versus Instances

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

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Figure 2.1

Schema diagram for the database in Figure 1.2.

Example of a database state

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.

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.

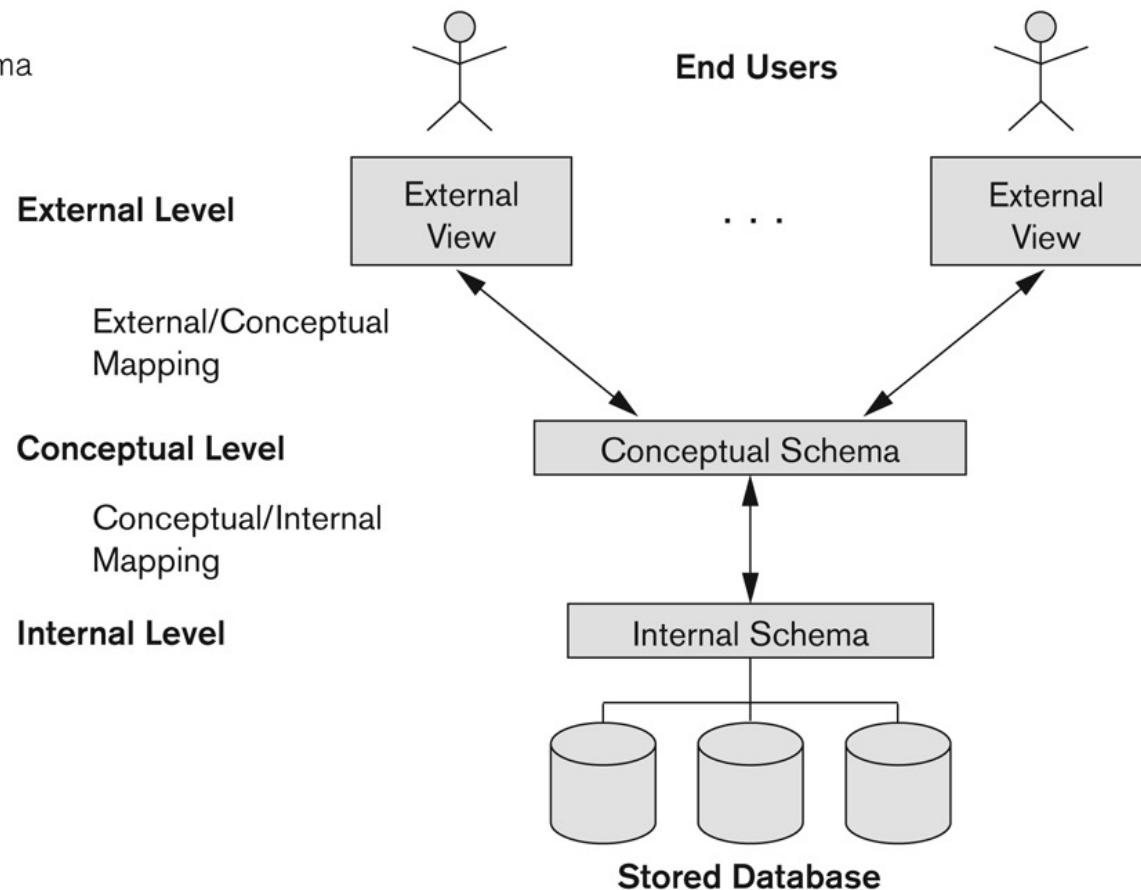
Uses a **conceptual** or an **implementation** data model.

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

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.

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.

DBMS Interfaces

Stand-alone query language interfaces

Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)

Programmer interfaces for embedding DML in programming languages

User-friendly interfaces

Menu-based, forms-based, graphics-based, etc.

Mobile Interfaces:interfaces allowing users to perform transactions using mobile apps

DBMS Programming Language Interfaces

Programmer interfaces for embedding DML in a programming languages:

Embedded Approach: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)

Procedure Call Approach: e.g. JDBC for Java, ODBC (Open Database Connectivity) for other programming languages as API's (application programming interfaces)

Database Programming Language Approach: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components

Scripting Languages: PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

User-Friendly DBMS Interfaces

Menu-based (Web-based), popular for browsing on the web

Forms-based, designed for naïve users used to filling in entries on a form

Graphics-based

- Point and Click, Drag and Drop, etc.

- Specifying a query on a schema diagram

Natural language: requests in written English

Combinations of the above:

- For example, both menus and forms used extensively in Web database interfaces

galaxy Zoo

Other DBMS Interfaces

Natural language: free text as a query

~~Speech~~ : Input query and Output response

Web Browser with keyword search

Parametric interfaces, e.g., bank tellers using function keys.

Interfaces for the DBA:

Creating user accounts, granting authorizations

Setting system parameters

Changing schemas or access paths

Database System Utilities

To perform certain functions such as:

Loading data stored in files into a database. Includes data conversion tools.

Backing up the database periodically on tape.

Reorganizing database file structures.

Performance monitoring utilities.

Report generation utilities.

Other functions, such as sorting, user monitoring, data compression, etc.

Typical DBMS Component Modules

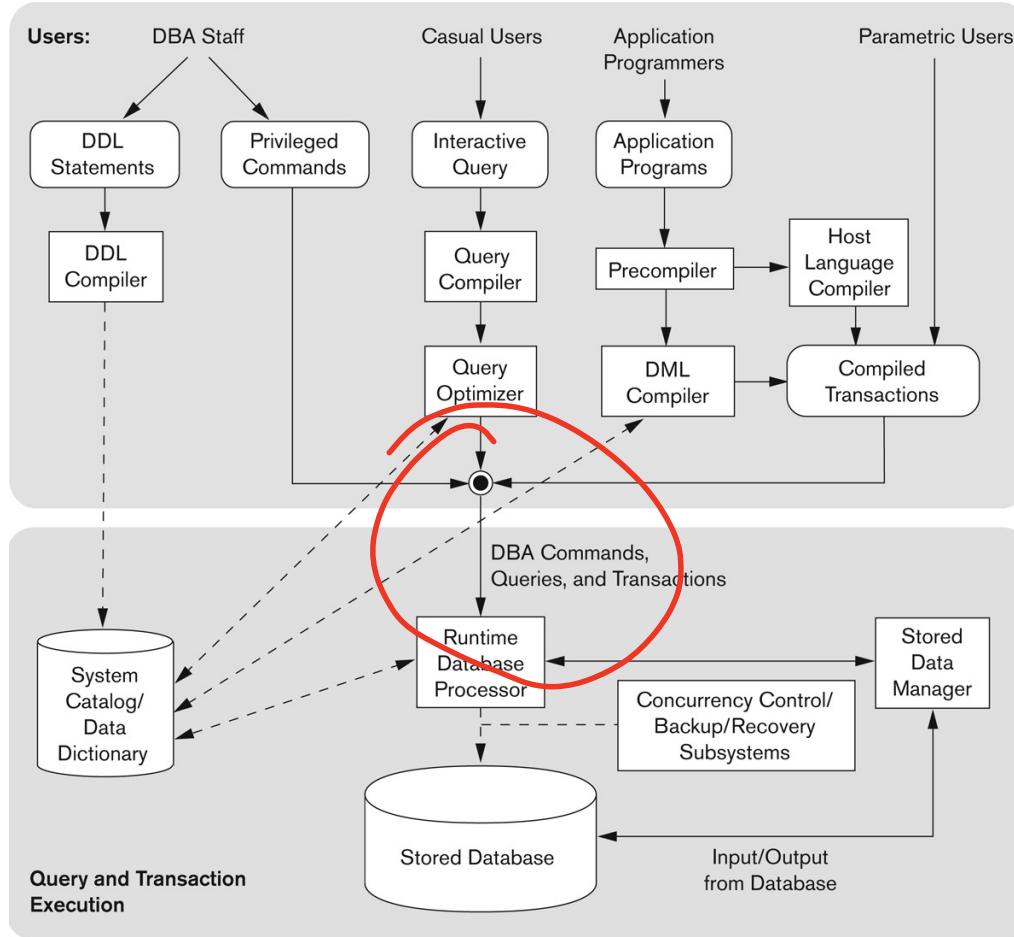


Figure 2.3

Component modules of a DBMS and their interactions.

*Ankit
trace*

Centralized and Client-Server DBMS Architectures

Centralized DBMS:

Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.

User can still connect through a remote terminal – however, all processing is done at centralized site.

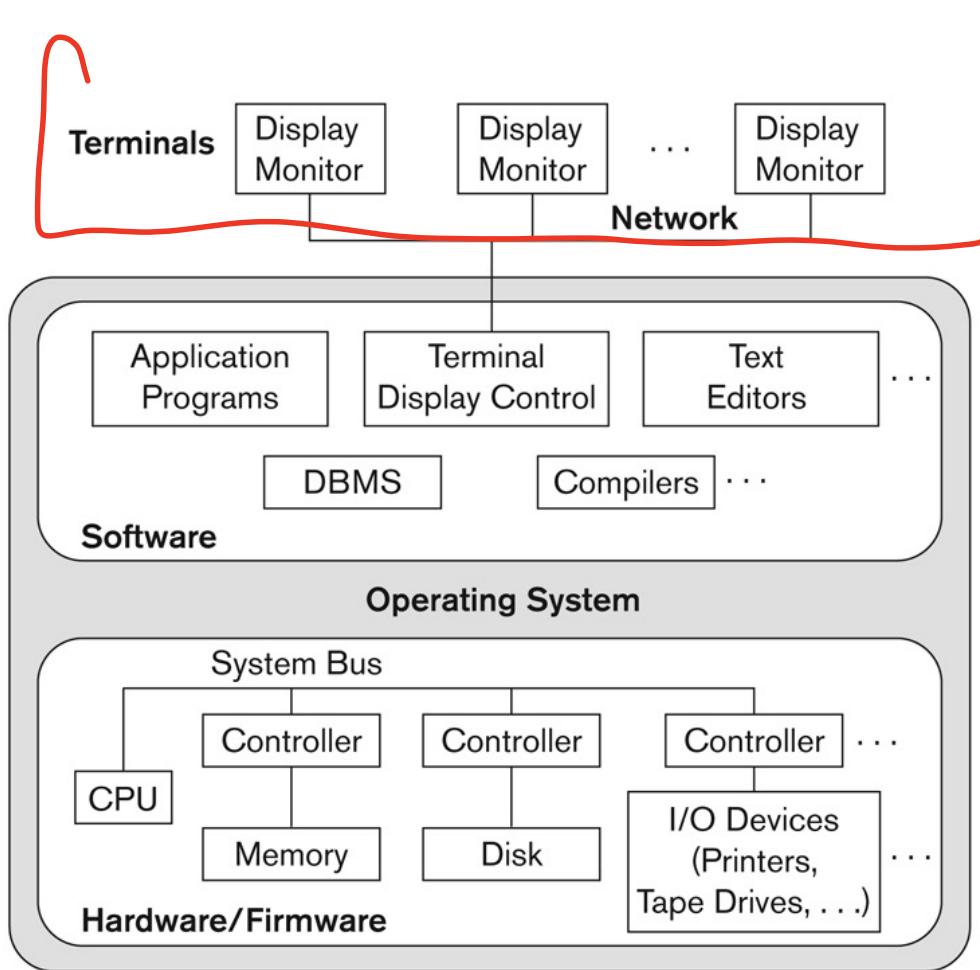
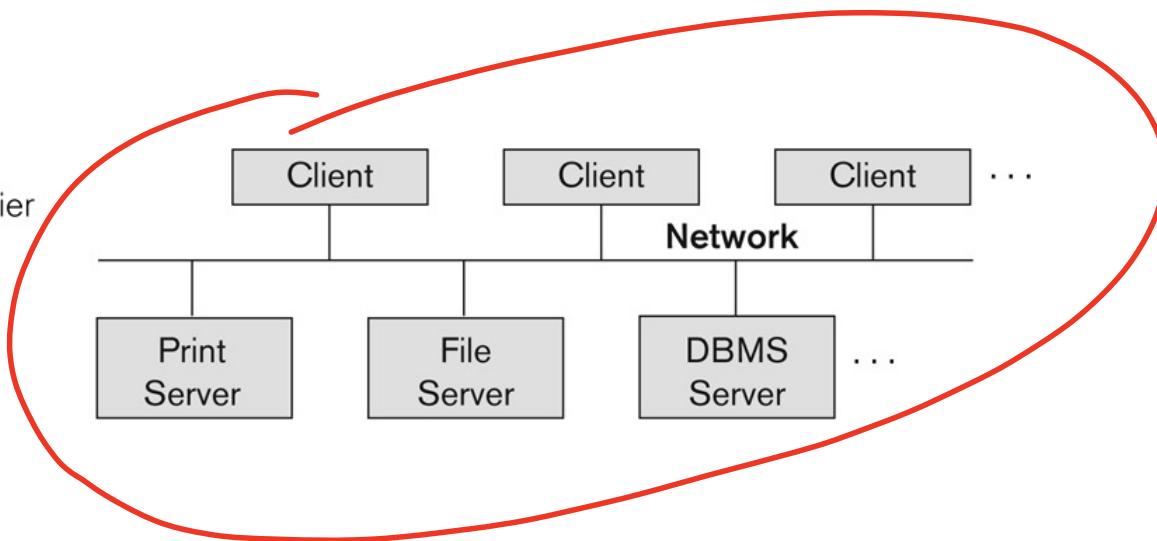


Figure 2.4
A physical centralized architecture.

A Physical Centralized Architecture

Logical two-tier client server architecture

Figure 2.5
Logical two-tier
client/server
architecture.



Clients

Provide appropriate interfaces through a client software module to access and utilize the various server resources.

Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.

Connected to the servers via some form of a network.

(LAN: local area network, wireless network, etc.)

DBMS Server

Provides database query and transaction services to the clients

Relational DBMS servers are often called SQL servers, query servers, or transaction servers

Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:

ODBC: Open Database Connectivity standard

JDBC: for Java programming access

Two Tier Client-Server Architecture

Client and server must install appropriate client module and server module software for ODBC or JDBC

A client program may connect to several DBMSs, sometimes called the data sources.

In general, data sources can be files or other non-DBMS software that manages data.

See Chapter 10 for details on Database Programming

Three Tier Client-Server Architecture

Common for Web applications

Intermediate Layer called Application Server or Web Server:

Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server

Acts like a conduit for sending partially processed data between the database server and the client.

Three-tier Architecture Can Enhance Security:

Database server only accessible via middle tier

Clients cannot directly access database server

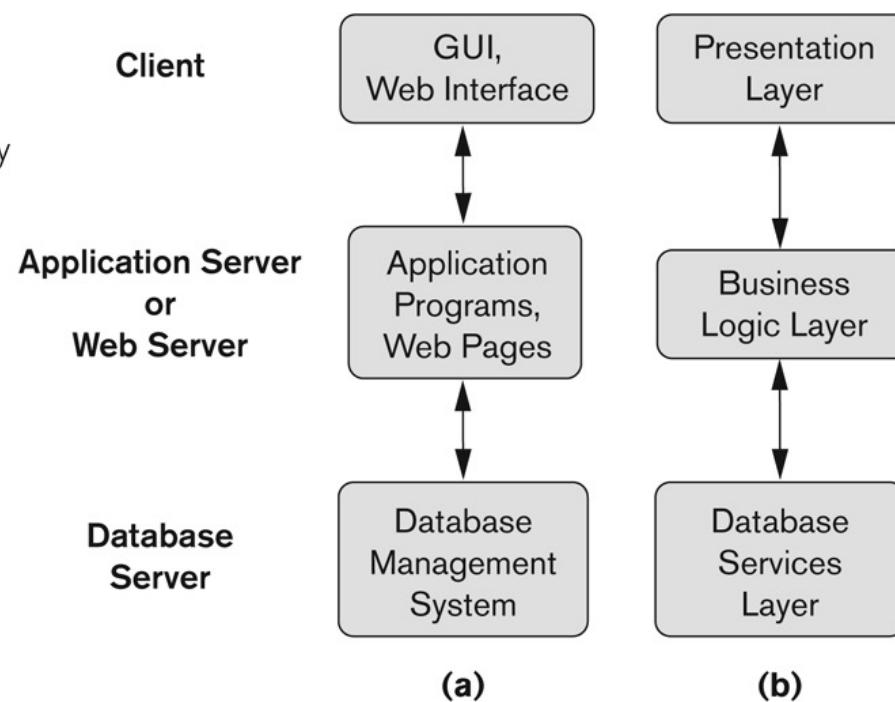
Clients contain user interfaces and Web browsers

The client is typically a PC or a mobile device connected to the Web

Three-tier client-server architecture

Figure 2.7

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



Classification of DBMSs

Based on the data model used

Legacy: Network, Hierarchical.

Currently Used: Relational, Object-oriented, Object-relational

Recent Technologies: Key-value storage systems, NOSQL systems: document based, column-based, graph-based and key-value based. Native XML DBMSs.

Other classifications

Single-user (typically used with personal computers)
vs. multi-user (most DBMSs).

Centralized (uses a single computer with one database) vs. distributed (multiple computers, multiple DBs)

Variations of Distributed DBMSs (DDBMSs)

Homogeneous DDBMS

Heterogeneous DDBMS

Federated or Multidatabase Systems

Participating Databases are loosely coupled with high degree of autonomy.

Distributed Database Systems have now come to be known as client-server based database systems because:

They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Cost considerations for DBMSs

Cost Range: from free open-source systems to configurations costing millions of dollars

Examples of free relational DBMSs: MySQL, PostgreSQL, others

Commercial DBMS offer additional specialized modules, e.g. time-series module, spatial data module, document module, XML module

These offer additional specialized functionality when purchased separately

Sometimes called cartridges (e.g., in Oracle) or blades

Different licensing options: site license, maximum number of concurrent users (seat license), single user, etc.

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e



pk.profgiri



Ponnurangam.kumaraguru



/in/ponguru



ponguru



pk.guru@iiit.ac.in

Thank you
for attending
the class!!!