

Database System Concepts and Architecture

CS 301

“Goal of the chapter”

“Learn the terminology and basic concepts of DBMS.”

1. Discuss data models and define the concepts of schemas and instances.
2. Three-schema architecture and data independence.
3. Database language and interfaces.
4. The database system environment.
5. Centralized and client/server architecture for DBMS.
6. Classification of DBMS.

Data models, schemas and instances: some terminologies

- DBMS approach provides some level of data abstraction *.
- Data abstraction – suppression of details of data organization and storage, and highlighting of the essential features for an improved understanding of the data. It allows different users to perceive data at their preferred levels of detail.
- Data model – A “collection of concepts” that describe structure (such as data types, relationships, constraints) of a DB.
 - They include a set of basic operations for specifying retrievals and updates on the DB.
 - They also take account of “dynamic aspect or behavior” of the DB application that allows to specify user-defined operations on the DB objects. Example, COMPUTE_GPA that can be applied to a STUDENT object.

* The characteristics that allow program-data independence and program-operation independence is called *data abstraction*.

- Conceptual data model – They are high level concepts that are close to the way many users perceive data. They use concepts such as entities, attributes, and relationships.
- Physical data model – They are low level concepts that describe the details of how data is/are stored on the storage media. They are mostly used by specialists and they contain information such as formats, record orderings, access paths, etc.
- Representational (or implementation or record based) data model – They provide concepts that may be easily understood by end users but are not too far from the way data is/are organized in storages. They hide details of data storage but can be implemented on a computer system directly.
 - Traditional commercial DBMS have relational data model, network data model or hierarchical data model. New ones include “self-describing data models” that combine the description of data with the data values.

- An entity – “represents a real-world object or concept” (example, employee) or “project from the real world” described in the DB.
- An attribute – “represents some property of interest” that further describes an entity (example, employee’s name).
- A relationship – It “exist between two or more entities representing an association” among the entities (example, a “works-on” relationship between an employee and a project).
- Object data model – It is a higher level implementation data model that are closer to conceptual data model.
- Access path – It “is a search structure for searching a DB record” efficiently *such as indexing or hashing*.
- An index – It is an “Access path that allows direct access to data using an index term or a keyword” (similar to an index in a book) organized in a *linear or hierarchical or tree structure format*.

Schemas, instances and database state

- Database schema (or *scheme* or *intension*) – description of the DB which is specified during DB design and it not expected to change frequently.
- Schema changes (schema evolution) are usually required when there is a change in the DB application.
- A displayed schema is called a “*schema diagram*”.

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
----------------	--------------------	-------

Schema diagram for a DB.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

Database

Schema
construct

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
----------------	--------------------	-------

Schema diagram for the DB

Displays the structure of each record type (such as name of record type, data items, constraints and relationships) but not the actual instances of records.

- Database state (or snapshot or extension) – The data in the database at a particular moment in time. It is also called the *current* set of **occurrences** or **instances** in the DB.
- In a given DB state, each schema construct has its own *current* set of instances.
- A new DB definition requires specification of a DB schema, while the corresponding DB is in the empty state (without data).
- When the DB is *initiated* or *populated* or *loaded* with data, its initial state is obtained.
- Every time a record is *inserted* or *deleted* or *changed*, its state changes to another.
- At any point in time, the DB has a *current state* or *snapshot* or *instance*.

Three-schema architecture and data independence

- Three characteristics of the DB approach:
 1. Use of catalog to store schema.
 2. Insulation of program and data (program-data and program-operation independence), and
 3. Support of multiple user views.
- A Three-Schema Architecture helps achieve and visualise all three above characteristics.

The Three-Schema Architecture

The goal of the three schema architecture is to separate the user application from the physical DB. Schema can be defined at the following three levels:

1. The internal level has an **internal schema** – it describes the physical storage structure of the DB. It uses a physical data model and describes the complete details of data storage and access path for the DB.
2. The conceptual level has a **conceptual schema** – it describes the structure of the whole DB. It also hides the details of physical storage and focuses on describing entities, data types, relationships, user operations and constraints. Usually, a representative data model is used to describe the conceptual schema.
3. The external or view level – it has a number of **external schemas** or user views. Each external schema describes part of the DB in which a particular user group is interested and hides the rest of the DB. This is also implemented using a representational data model.

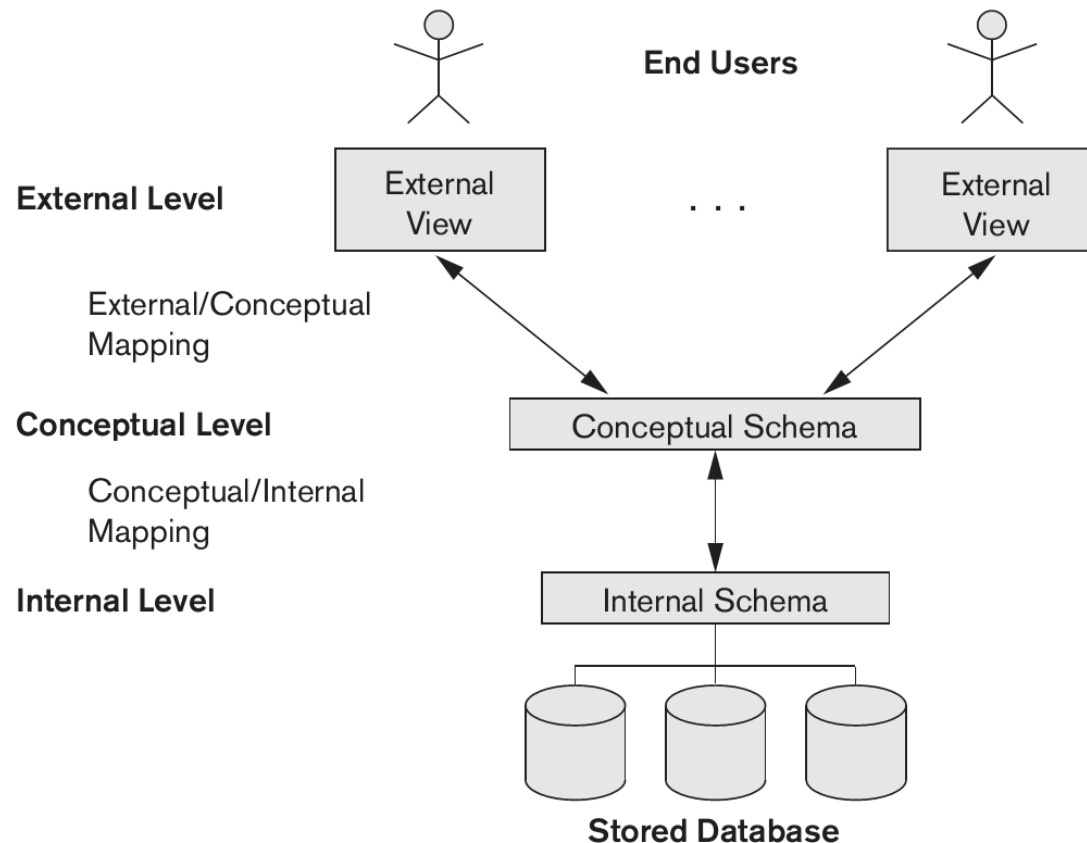


Figure: The three-schema architecture.

In most DBMS such as Oracle, SQL Server, etc. that support user views, external schemas are specified in the same data model that describes the conceptual-level information.

- Important – the three schemas are only “descriptions” of data and the actual data are stored at the physical level.
 - A DBMS transforms a request specified on an external schema into a request on the internal schema for processing over the stored DB.
- If the request is a DB retrieval, the data extracted from the stored DB is reformatted to match the user’s requirement – this process of request and result transformation between levels is called mapping.

Data independence

- Data independence allows to change the DB schema at one level without having to change the schema at the next higher level.
 1. Logical data independence – It is the capacity to change the conceptual schema without changing the external schema (application programs). For example, expanding the DB by adding a record type or data item, changing constraints, reducing DB by removing a record, etc. Only the view and mapping needs to be changed in a DBMS that supports logical data independence.
 - Important – After the conceptual schema undergoes a logical reorganisation, application programs that refer the external schema construct must work like before. Here, changes to constraints can be applied to the conceptual schema without affecting the external schemas / application programs.

Data independence

2. Physical data* independence – It is the capacity to change the internal schema without changing the conceptual schema. Hence, the external schema need not be changed.
 - The changes in internal schema happen when there is a reorganisation of files – for example, by creating additional access structures that is meant to help in gaining improvements in performance of update/retrieval.
- * Physical data details include location of data, storage encoding, placement, compression, splitting/merging of records, etc. that are hidden from the user.
- Important – Data independence occurs because when the schema is changed at some level, the schema at the next higher level (for example, application programs) remains unchanged; only the mapping between the levels is changed.

DB languages and interfaces

- DBMS supports appropriate languages and interfaces for each category of users.
- Data definition language (DDL) – Used by DBA and DB designers to define conceptual and internal schemas and mapping between the two.
 - DBMS have a DDL compiler which processes DDL statements to identify descriptions of the schema constructs and store them in DBMS catalog.
- Storage definition language (SDL) – In some DBs that have separate conceptual and internal schema levels, DDL along with a separate SDL exist; SDL is used to specify the internal schema.


- In modern RDBMS, there is no specific language that performs the role of SDL.
- In RDBMS, the internal schema is instead specified by a combination of functions, parameters, and specifications related to storage of files.
 - Hence, DBA can control indexing choices and mapping of data to storage.
- View definition language (VDL) – In a three schema architecture, VDL is used to specify “user views” and “their mappings to the conceptual schema”.
 - However in most DBMSs, DDL defines both conceptual and external schemas. Example, SQL is used as a VDL to define user or application views for predefined queries.

- Data manipulation language (DML) – DML are used to manipulate DBMS with operations like retrieval, insertion, deletion, and modification of data.
- Types of DMLs: **high-level and low-level**
 1. A high-level or nonprocedural DML – Used to specify DB “**operations as statements**” either interactively from a terminal or through a *programming language*. In the *program*, the DML statements are identified and extracted by a “precompiler”. Example, “SQL that can specify and retrieve **many records** in a single DML statement”.
 - They are also called set-at-a-time or set-oriented DMLs.
 2. A low-level or procedural DML – It is embedded in a programming language. The DML retrieves **individual records or objects** from the DB and “**processes each separately through loops**”.
 - They are also called record-at-a-time DMLs.

- Structured query language (SQL) – In current DBMSs, a comprehensive integrated language is used that includes constructs for conceptual schema definition (DDL), view definition (VDL), and data manipulation (DML) - this is “**SQL**”.
 - SQL also allows constraint specification, schema evolution, etc.
 - SDL is not part of SQL.
- Whenever DML commands (high or low) are embedded in a programming language, the language is called host language and the DML is called data sublanguage.
- A high level DML used in a standalone interactive manner is called a query language.
- Casual end users use a high-level query language whereas programmers use the DML in its embedded form.

- Naïve / parametric / casual users have “user-friendly interfaces” to interact with the DB. They are
 - Menu-based interfaces for web clients or browsing – They have a list of options (menus) that lead the user through the formulation of a request. “Menus substitute the commands and syntax of a query language”. Example, Web-based user interfaces, browsing interfaces, etc.
 - Apps for mobile devices – Mobile users get access (through login) to data and limited menu options as in banking, reservations, etc.
 - Forms-based interfaces – It displays a form to be filled by naïve users for canned transactions. Example, Oracle Forms.
 - GUIs – “It consists of a schema shown to the user in a diagrammatic form”. The user can specify a query by manipulating the diagram through menus and forms.

- Natural language interfaces – These interfaces accept requests in English or other language with their *own schema* and *dictionary of important words* which are used to interpret the request. The interface generates a query and submits it to the DBMS for processing.
- Keyword-based DB search – They are similar to Web search engines accepting *strings* and matching them with documents at specific sites / search engines and present the matched words in decreasing order of relevance / match.
- Speech input and output – These are applications with limited vocabularies such as “*telephone directory, flight arrival/departure, credit card account information, etc.*”. The speech input is detected using a library of predefined words and used to set up queries that output text or numbers into speech.

- 
- Interface for parametric users – Example, bank tellers who perform small set of operations (routine) repeatedly (such as deposits, withdrawals, etc.), where interfaces are designed by programmers to minimise number of keystrokes with abbreviated commands for each request.
 - Interfaces for the DBA – They are privileged commands “to create accounts, setting system parameters, granting account authorisation, changing a schema, reorganising the storage structure, etc.”. of a DB by the DBA.

The DB system environment

- The DB system environment deals with the **software components** in a DBMS and **system software** with which the DBMS interacts.
- DBMS component modules:
 - DBMS components are divided into two parts:
 - (i) Top part refers to the various users and their interfaces while
 - (ii) Bottom part shows the internal modules responsible for storage of data and processing of transactions.

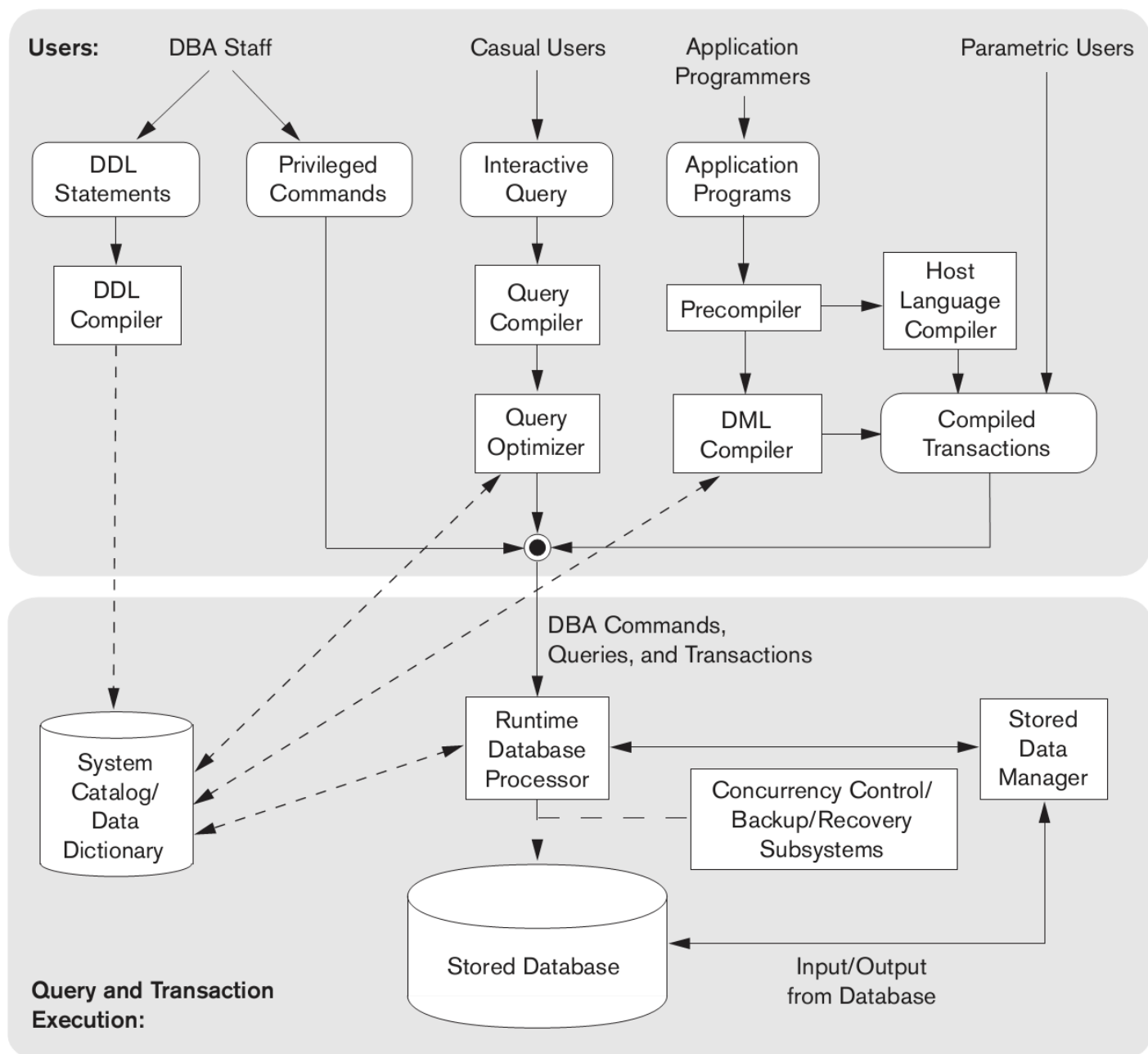


Figure: Component modules of a DBMS and their interactions.


- In the top part, interfaces for the DBA, casual users, application programmers and parametric users are present.
- DBA staff define the DB and tune it using DDL. DDL compiler processes schema definitions specified in DDL and stores descriptions (meta-data) in the DBMS catalog containing names, data types, storage details, mapping information, schemas, constraints, etc.
- Interactive query – casual users who need information from the DB interact with interactive query.
- Query compiler – The interactive queries are parsed and validated for correctness (such as for query syntax, file names, etc.) by query compiler.
- Query optimizer – It is concerned with the rearrangement and possible reordering of operations, eliminations of redundancies, and use of efficient search algorithms during execution.

- Application programs – Program that are written in host languages (C, C++ or Java) are send to precompiler to extract DML commands where DML compiler converts them into object code for DB access. Rest of the program is sent to host language compiler. The object code and rest of the program are linked for canned transactions (executed via PCs or mobile apps), example bank payment transaction. Now, PHP and Python are also becoming popular to write DB programs.
- Lower part has runtime database processor that executes (1) the privileged commands, (2) the executable query plan, (3) the canned transactions with runtime parameters. It also works with system catalog and stored data manager.
- Stored data manager – It uses OS services for input/output (read/write) operations between disk and main memory.
- There are often “client programs” also known as “client computer” that has DBMS client software and accesses data from a DB server or sometimes through an application server which in turn access DB server.

- DB system utilities:

- DBMS have DB utilities that help DBA manage the DB system. These include

1. Loading – Used to load existing data files such as text files, sequential files, etc. into the DB. From the current (source) format of the data file and the desired (target) DB file structure specification, the utility automatically reformats the data and stores in the DB.
2. Backup – This utility creates a backup copy on tape or other storage for restoring in case of disk failure. They also support incremental backups to record “only changes” since it saves storage space.
3. DB storage reorganisation – It reorganises DB files into different file organisation and creates new access paths to improve performance.

- 
4. Performance monitoring – This utility monitors DB usage and provides statistics to the DBA to make decision on reorganising files, adding/dropping indexes to improve performance.
 - Other utilities such as for *sorting files, handling data compression, monitoring access by users, interfacing with the network*, etc. are also available.

- Tools, application environments and communication facilities:
 - Other tools are also available to DB designers and users such as CASE (computer aided software engineering), data dictionary, etc. Data dictionary **stores catalog information** about schemas and constraints, design decisions, usage standards, application program description, user information, etc. that can be accessed directly by users or the DBA when needed.
 - Application development environments – They provide an environment for developing DB applications, DB design, GUI development, querying, updating, application program development, etc.
 - Communication software – It allows users at remote locations to access the DB connected through routers, phone lines, local networks, satellite communication devices, etc. **Some DBs are physically distributed over multiple machines connected through LANs.**

Client/Server Architectures

- Centralised DBMSs Architecture
 - Most users access DBMS through computer terminals, PCs, workstations, and mobile devices that have only display capabilities with little processing power.
 - All processing are performed on a remote system housing DBMS, and only display information and controls are sent from the central computer to the display terminals through communication channels.
 - Centralised DBMS servers support DBMS functionality, application program, user interface, etc.

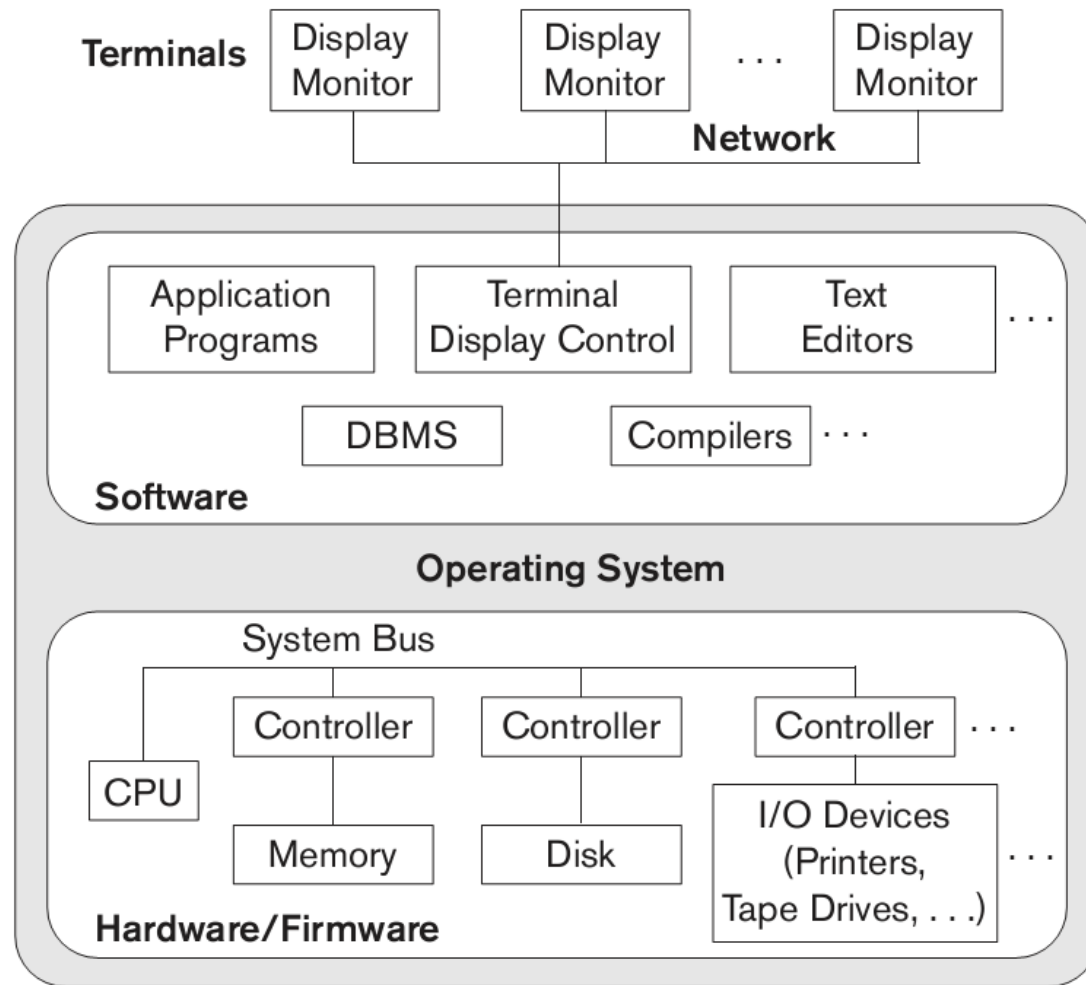


Figure: Physical components in a centralised architecture.

- DBMS have recently started exploiting processing power at user side leading to client/server DBMS architecture.

Client/Server Architectures

- Client/server architecture was developed to deal with environments where large number of PCs, workstations, file servers, printers, DB servers, Web servers, e-mail servers and other machines are connected through a network.
- This made possible to connect specialised servers (such as file server, printer server - that connects various printers and all print requests by the clients are forwarded to this machine, Web servers, e-mail servers, etc.) to a number of PCs, workstations as clients.
- The client machines provide the user with interfaces to utilise these servers as well as with local processing power to run local applications.
 - Example, CAD, ANSYS, Matlab – stored on specific server and being made accessible to multiple clients.

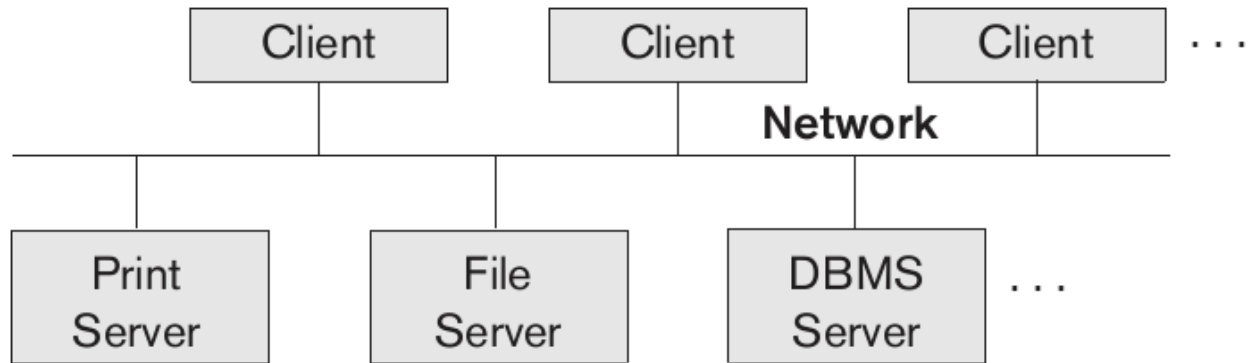


Figure: Logical two tier client/server architecture.

- Some machines are client sites only (mobile device, workstations/PCs) that have only client software installed.
- Other machines are dedicated servers and a third category have both client and server functionality.

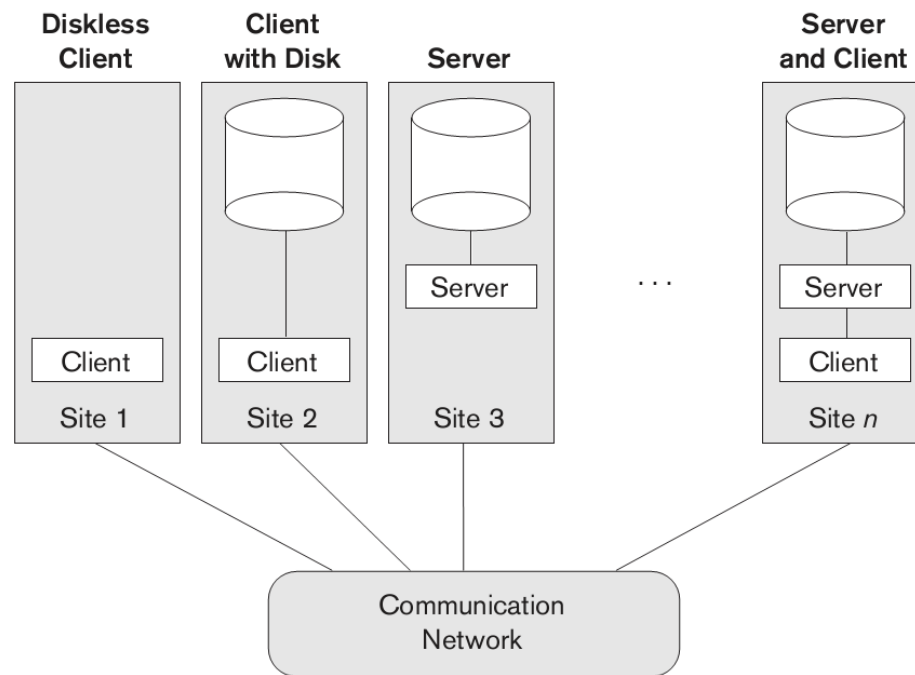


Figure: Physical two tier client/server architecture.

- **Client** – A user machine that provides user interface capabilities and local processing. DB access is done by connecting the client to the server.
- **Server** – A system containing both hardware and software that provide services to the client machines – example, *file access, printing, archiving, etc.*

Two-Tier Client/Server Architectures

- In RDBMS such as SQL, query and transaction functionality is on the server side. So, a server is called a “query server” or “transaction server” or a “SQL server”.
- When DBMS access is required from the client side, the program establishes a connection to the server through ODBC (open DB connectivity) through an API (application programming interface).
- A client program can connect to several RDBMSs and send query and transaction request using ODBC API which are then processed at the server sites. The query results are then communicated to the client program which processes and displays the results as needed. Example, JDBC.
- This is a **two-tier architecture** because the software components are distributed over two systems: client and server.

Three-Tier Client/Server Architectures

- Web applications use three-tier architecture, which adds an intermediate layer between the client and the DB server.

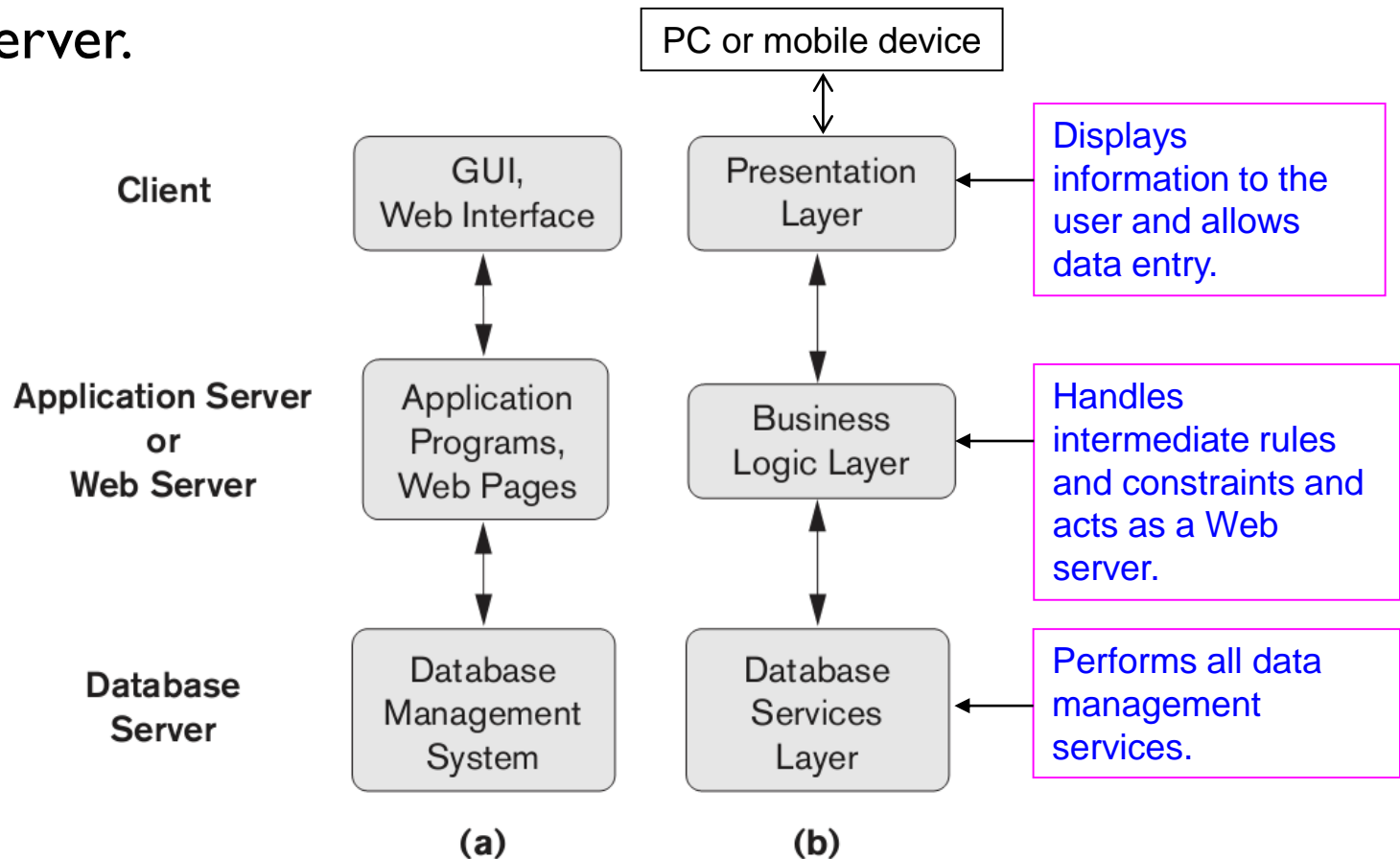
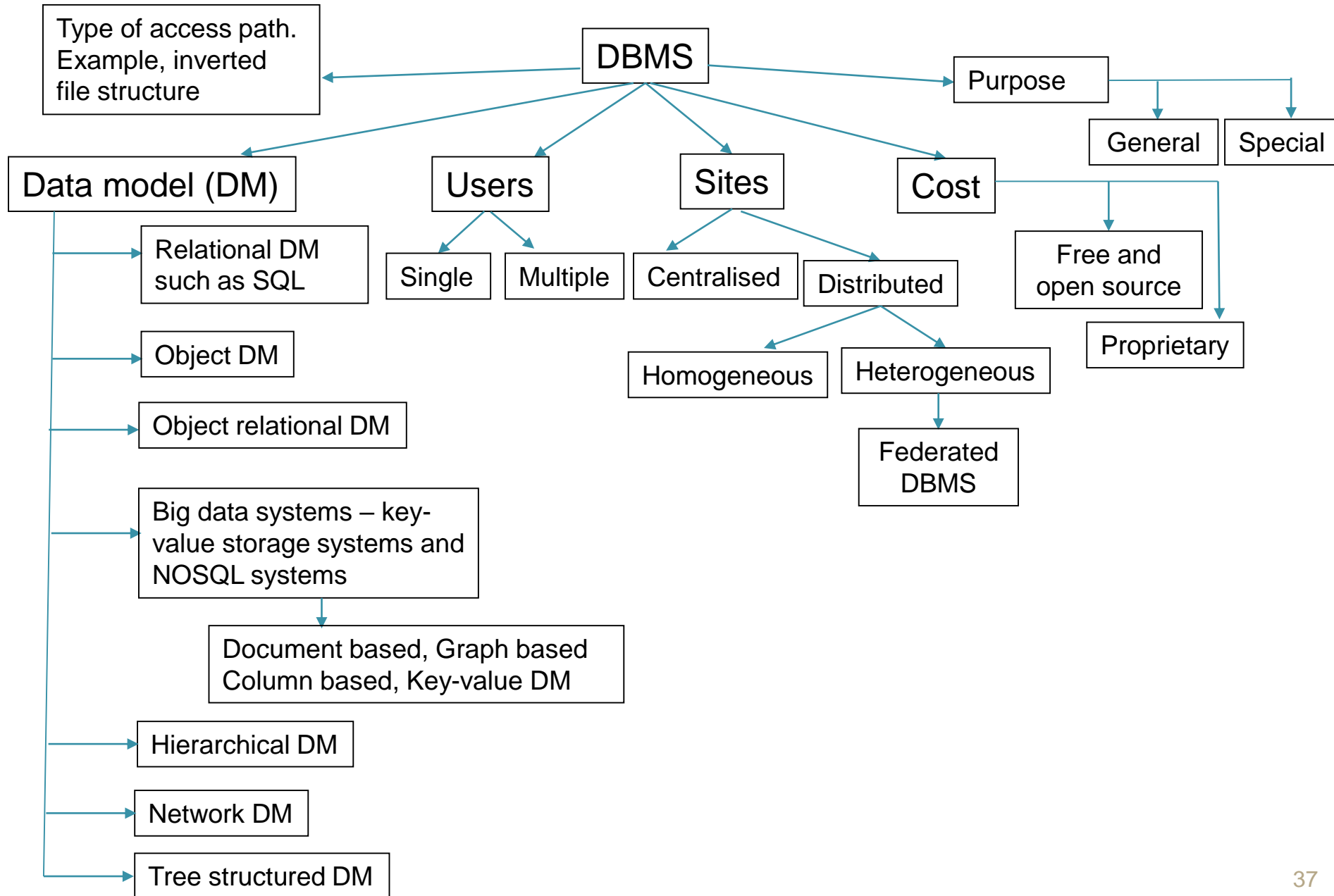



Figure: Logical three-tier client/server architecture.

- The intermediate or middle layer is called the application server or the Web server serving an intermediary role by running application programs and storing rules (procedures or constraints) and by checking a client's credentials before forwarding a request to the DB server to access the data.
 - It passes partially processed data from the DB server to the clients where it may be processed further and filtered to be presented to the users.
- Thus, the *user interface*, *application rules*, and *data access* act as three tiers.
- Sometimes, the layers between the user and the stored data are divided into finer components resulting in n -tier architectures, where $n = 4$ or 5 .
 - Various technologies for data compression are used to transfer large amounts of encrypted data from servers to clients (where data are decrypted) over wired or wireless networks.

Classification of DBMS



- Relational data model (DM) – It represents a DB as a collection of tables, where each table is a separate file. Most RDBMS use high-level query language such as SQL and support limited form of user views.
- Object DM – It defines a DB in terms of objects, their properties and operations. Objects with the same structure and behaviour belong to a class and are organised into hierarchies. The operations of each class are specified in terms of “procedures” called “methods”. Relational DBMSs are extended to incorporate object DB concepts in object relational systems.
- Big data systems are based on the following DM –
 - Key-value DM where a unique key is associated with each record or object providing fast access to a value given its key.
 - Document DM based on JSON (Java Script Object Notation) that stores the data as documents.
 - Graph DM – stores objects as graph nodes and relationship among objects as directed graph edges.
 - Column based DM – stores the columns clustered on disk pages for fast access.

- 
- Hierarchical DM – represents data as hierarchical tree structures where each hierarchy represents a number of related records.
 - Network DM – represents data as record type and also represents a limited type of 1:N relationship.
 - XML model is a standard for “exchanging data over the Web” using hierarchical “tree structures”. This model resembles the object model.

Summary

- Data model –
 - (i) High level or conceptual data models.
 - (ii) Low level or physical data models.
 - (iii) Representational or implementational data model.
- Schema, state of a DB.
- Three schema DB architecture – an internal schema that describes physical storage structure, a conceptual schema that is a high-level description of the DB, external schema that describes the views of different user groups.
- Mapping among the schemas to transform requests and query results from one level to the next.

- Languages and interfaces – DDL, SDL, DML (high level or set oriented, nonprocedural, OR low level or record oriented, procedural). A high level DML can be embedded in a host programming language or as a stand alone language – called a query language.
- Different types of interface and users, database system environment, modules and utilities.
- Two and three-tier architectures.
- Classification of DBMS.



Practice questions (check yourself)

1. Define the following: data model, conceptual, physical and representational data models.
2. Define entity, attribute, relationship, schema and query language.
3. What is a three-schema architecture? What is meant by mapping among the schemas?
4. Explain logical and physical data independence.
5. What are the different types of user-friendly interfaces to interact with the DB?
6. Discuss DB system utilities and client/server architecture.