1. What is database? Describe the advantage & disadvantages of using DBMS. (15)

Sol: A database is a collection of related data. By data we mean known facts that can be recorded and that have implicit meaning. For eg: consider the student name, mark of a class. We may have to record this data in an indexed address book or may have to stored it on harddisk. This collection of data with an implicit meaning is a database. The preceding definition of database is quite general.

A database has following implicit properties

* A database represents some aspect of the real world, sometimes called the miniworld. Changes to the miniworld are reflected in the database.

* A database is a logically coherant collection of data with some inherent

Date: / /

meaning: A random assortment of data cannot correctly be ~~refered~~ referred to as a database.

* A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which users are interested.

Advantages of DBMS.

(i) Controlling Data Redundancy

In traditional computer file processing each application program has its own files. In this case, the duplicated copies of the same data are created at many places. In DBMS, all the data of an organization is integrated into a single database. The data is recorded at only one place in the database & it is not duplicated. When they are converted

into database, the data is integrated into a single database so that multiple copies of the same data are reduced to single copy. In DBMS, Data ~~redud~~ redundancy can be controlled or reduced but it is not completely removed.

Data Consistency

By controlling the data redundancy the data consistency is obtained. If a data item appears only once, any update to it's value has to be performed only once & the updated value is immediately available to all users.

Data Security

Data security is the protection of DBMS from unauthorized users. Only authorized persons are allowed to access the database. Some users

maybe allowed to access only a part of database. ie; the data that is related to them or related to their department.

Data Atomicity

A transaction in commercial database is preferred to as atomic unit of work. All these transaction must be completed in all; otherwise partially completed tasks are rolled back. Thus through DBMS it is ensured that only consistent data exists within the database.

Data Independence

Seperation of data structure of database from the application program that is used to access data from database is called data independence.

In DBMS, database & application programs are separated from each other. The DBMS sits in between

them. So we can change the structure of database without modifying application program

Backup & Recovery Procedures

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damaging due to failures to the computer system or application program. It is a time consuming method, if the volume of data is large.

Most of the DBMS provide the 'Backup & recovery' sub system that automatically create backup of data & restore data if required.

Disadvantages of DBMS

Cost of Hardware & software

A processor with high speed of data processing & memory of large size is required to run the DBMS software. That means you have to upgrade the hardware used for file-based system. Similiarly DBMS software is also very costly.

Damage of DBMS

In most of the organizations all data is integrated into a single database. If the database is corrupted due to the power failure or it is corrupted on the storage media our valuable data may be lost or the whole system stops.

Cost of Data Conversion

When the computer filesystem is replaced with a database system the data stored into data file must be converted to database files. It is difficult & time consuming method to convert data of data files into database.

Appointing Technical Staff

Training and handling of DBMS are performed by trained technical persons such as database administrator & application programmers are required to handle the DBMS.

2. What are the functions of DBA?

Sol: Functions of DBA are

1. Schema definition: The DBA creates the original database schema by executing a set of data definition statements in the DDL.

2. Storage structure & access-method definition.

3. Schema & physical-organization modification:

DBA carries out changes to the schema & physical organization to reflect the changing needs of the organization.

4. Granting of authorization for data access: By granting different types of authorization, the database administrator can regulate which

parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

Routine maintainance: Some of the DBA's routine maintainance activity are:

1. Periodically backing up the database.
2. Ensuring that enough free disk space.
3. Monitoring jobs running on the database & ensuring that performance is not degraded by very expensive task submitted by some users.

3. Define Data Independence? What are the different types?

Sol: The ability to modify a schema definition in one level without affecting a scheme definition in the next higher level is called data independence.

There are two levels of data independence:

1. Physical data independence.

It is the ability to modify ~~affecting~~ the physical schema without causing application programs to be rewritten Modifications at the physical level are occasionally necessary in order to improve performance.

2. Logical Data Independence

Logical Data Independence is the ability to modify the conceptual scheme without causing application

programs to be rewritten. Modifications at the conceptual level are necessary whenever the logical structure of the database is altered.

4. Who are database designer?

Ans: Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent & store this data. These task are mostly undertaken before the database is actually implemented & populated with data. Database designers typically interact with each potential group of users & develop views of the database that meet the data & processing requirements of these groups.

5. What are the three levels of data abstraction?

Sol: The three levels of data abstraction are:

(i) Physical level
(ii) Logical level
(iii) View level

Physical Level

It is the lowest level of abstraction that describes how the data are actually stored. The physical level describes complex

Logical Level

It is the next higher level of abstraction that describes what data are stored in the database & what relationships exist among those data.

View level

It is the highest level of abstraction that describes only part of the entire database.

6. What are the different types of database users?

Types of Database users are:

Native Users

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

- Native users use a form interface, where the users can fill in appropriate fields of the form.

2. Application Programmers

Application Programmers are computer professionals who write application programs. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms & reports without writing a program.

3. Sophisticated Users

Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each query to a query processor that the storage manager understands.

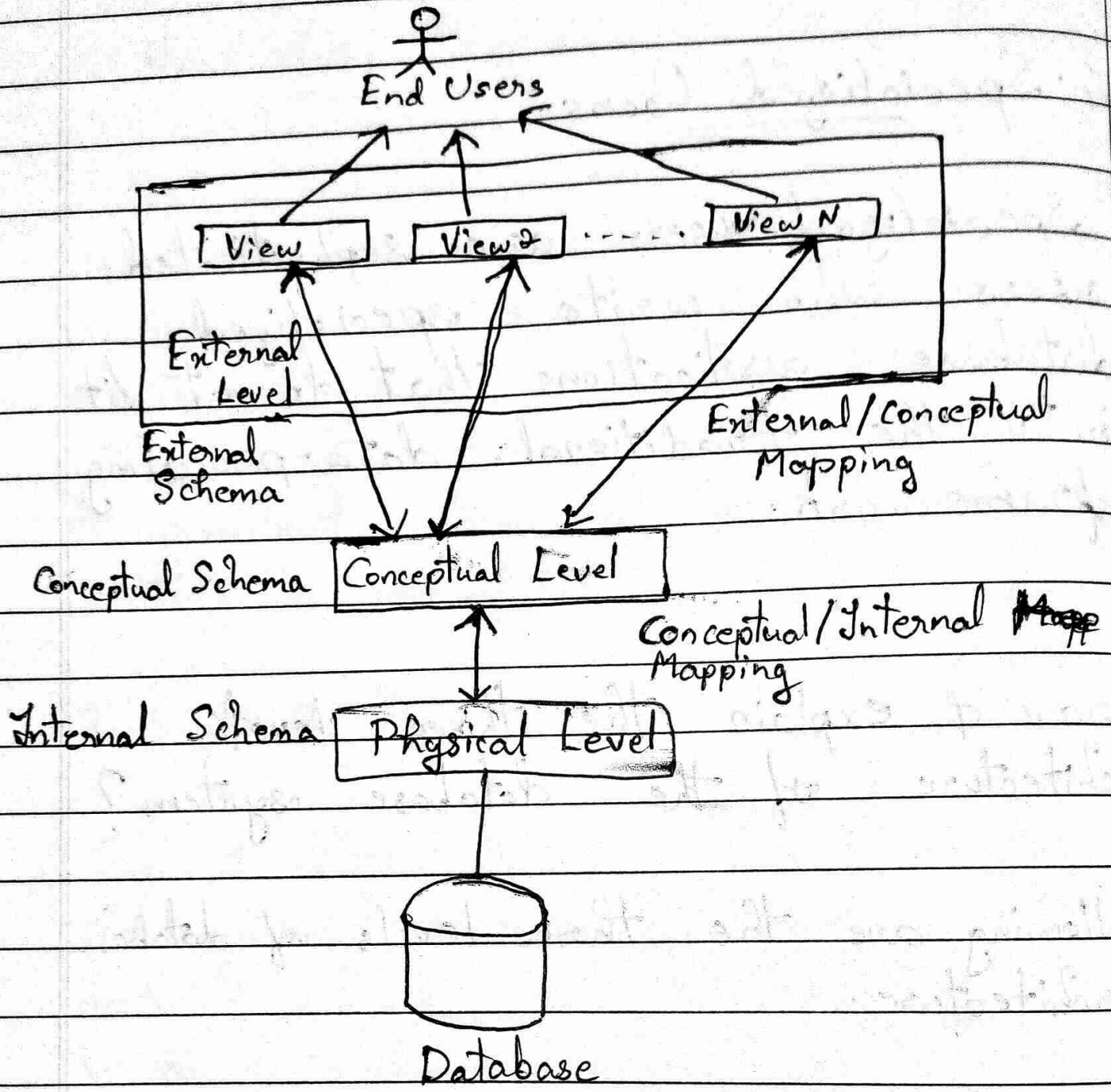
4. Specialized Users

Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

7) Draw & explain the three level architecture of the database system?

Ans: Following are the three levels of database architecture.

1. Physical ~~Level~~ Level
2. Conceptual Level
3. External Level



Physical Level

Physical level describes the physical storage structure of data in database. It is also known as Internal Level. This level is very close to physical storage of data. At lowest level, it is

stored in the form of bits with the physical addresses on the secondary storage device. and at highest level, it can be viewed in the form of files.

Conceptual Level

Conceptual Level describes the structure of the whose database for a group of users. It is also called as the data model. Conceptual schema is a representation of the entire content of the database. These schema contains all information to build relevant external ~~record~~ records. It hides internal details of physical storage.

External Level

External Level is related to the data which is viewed by individual end users. This level includes a no. of user views or external schemas.

Page No.
Date : / /

and this level is closest to the user. External view describes the segments of the database that is required for a particular user group & hides the rest of the database from that user group.

8. What is a data model? Discuss different categories.

Sol: Underlying structure of the database is called as data model. It is a collection of conceptual tools for describing data, data relationships, data semantics & consistency constraints.

Different types of Data Model

- Hierarchical model
- Network model
- Object Oriented model
- Object relational model
- Relational model
- Entity relationship model

Hierarchical Model

The hierarchical data model organizes data in a tree structure. In this model, each entity has only one parent but can have several children. Only one entity at the top of the hierarchy is called as Root. The structure is based on the rule that one parent can have many children but children are allowed only one parent - Linkages are only possible vertically but not horizontally or diagonally, ie; there is no relation between different trees at the same level unless they share the same parent.

Advantages

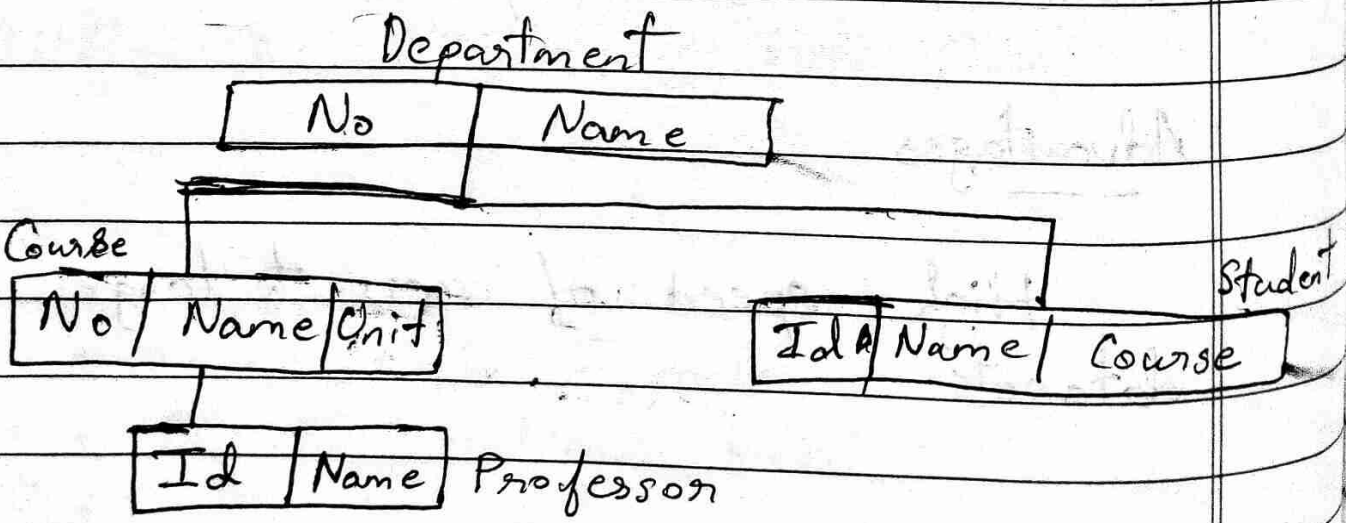
High speed of access to large datasets.

Data security: Hierarchical model was the first database model that offered the data security that is provided & enforced by DBMS

Efficiency: The Hierarchical model is very efficient when the database contains a large number of transactions using data whose relationships are fixed.

Disadvantage

- Implementation complexity
- Database management problems
- Lack of structural independence.



Network Model

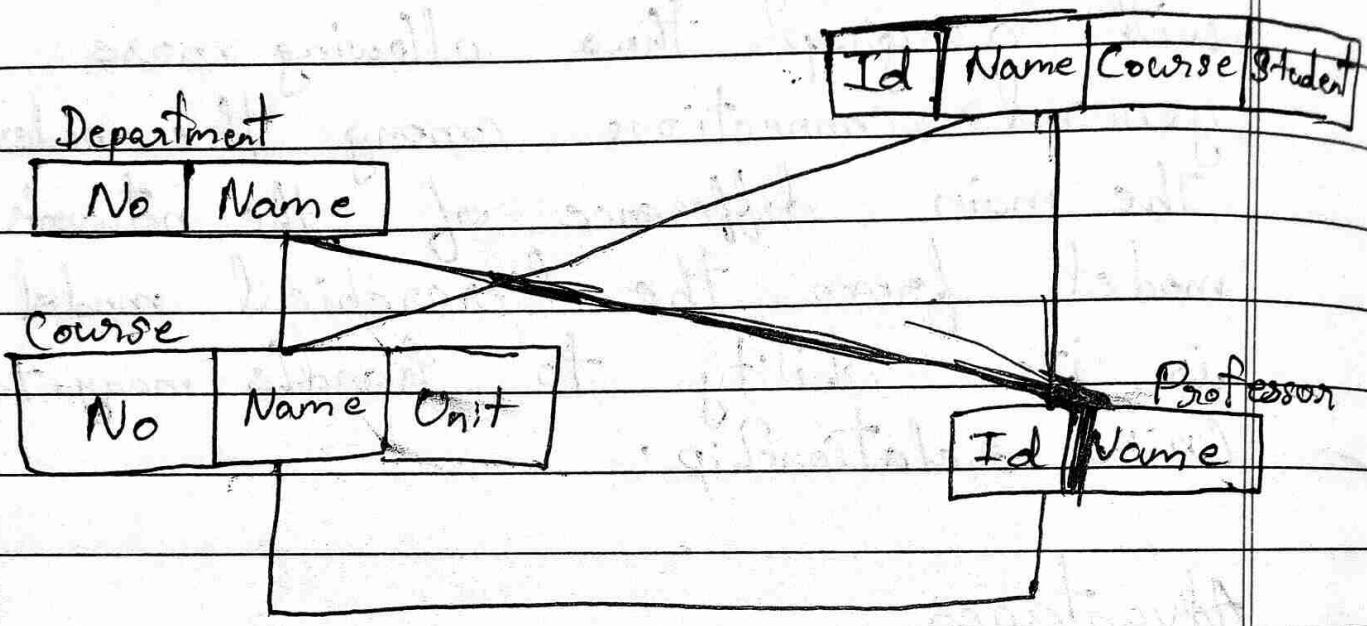
Network data model is based on directed graph theory. The network model replaces the hierarchical tree with a graph thus allowing more general connections among the nodes. The main difference of the network model from the hierarchical model is its ability to handle many-to-many (n:n) relationship.

Advantages

- Conceptual simplicity
- Capability to handle more relationship types
- Data Independence

Disadvantage

- Lack of structural independence
- Detailed structural knowledge is required.



Object-Oriented Model

The object-oriented model is based on a collection of objects. An object contains values stored in instance variable within the object. An object also contains bodies of code that operate in the object, these bodies of codes are called methods.

Advantages

- Applications require less code
- Applications use more natural data model
- Code is easier to maintain
- Object-oriented features improve productivity.

Object Relational Model

Object Relational Model adds new object storage capabilities to the relational systems at the core of modern information systems. These new facilities integrate management of traditional fielded data, complex objects such as time-series & geospatial data & diverse binary media. By encapsulation methods with data structures, this model can execute complex analytical & data manipulation operations to search and transform other complex objects.

Date :

Relational Model

The relational model represents data in the form of two dimensional tables. The organization of data into relational tables is known as the logical views of the database.

Characteristics of Relational Model

- Relational model eliminated all parent child relationships & instead represented all data in the database as simple row/column tables of data values.

- A relation is similar to a table with rows/columns of data values

- Relational model of data management is based on set theory.

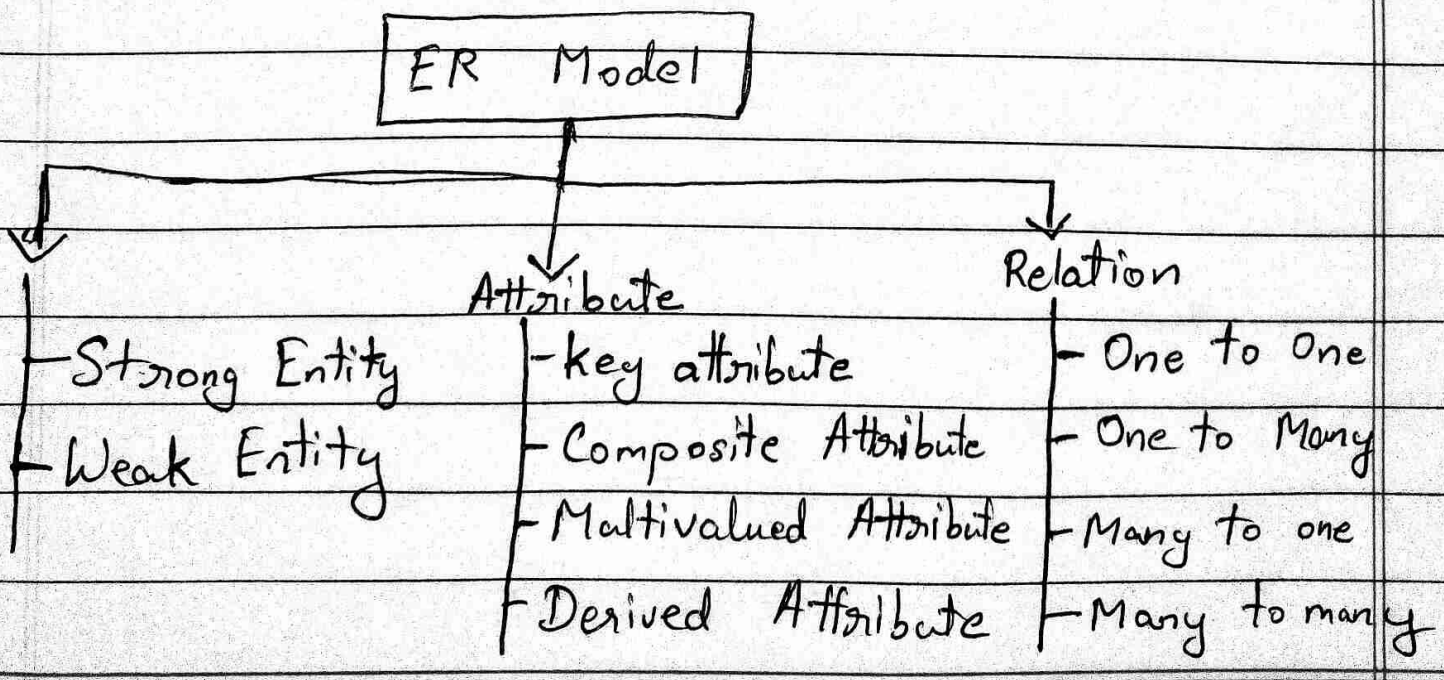
- Each table is an independent entity & there is no physical relationship

between tables.

Entity Relationship Model

Entity Relationship Model (E-R Model) is a high-level data model. This model is used to define the data elements & relationship for a specific system. It develops a conceptual design for the database & also develops a very simple & easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram

Component of ER Diagram



function keys in a terminal can be programmed to initiate various commands. This allows the parametric user to proceed with a minimal number of keystrokes.

Interfaces for the DBA. Most database systems contain privileged commands that can be used only by the DBA staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

2.4 The Database System Environment

A DBMS is a complex software system. In this section we discuss the types of software components that constitute a DBMS and the types of computer system software with which the DBMS interacts.

2.4.1 DBMS Component Modules

Figure 2.3 illustrates, in a simplified form, the typical DBMS components. The figure is divided into two parts. The top part of the figure refers to the various users of the database environment and their interfaces. The lower part shows the internals of the DBMS responsible for storage of data and processing of transactions.

The database and the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the **operating system** (**OS**), which schedules disk read/write. Many DBMSs have their own **buffer management** module to schedule disk read/write, because this has a considerable effect on performance. Reducing disk read/write improves performance considerably. A higher-level **stored data manager** module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

Let us consider the top part of Figure 2.3 first. It shows interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries, application programmers who create programs using some host programming languages, and parametric users who do data entry work by supplying parameters to predefined transactions. The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.

The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints. In addition, the catalog stores many other types of information that are needed by the DBMS modules, which can then look up the catalog information as needed.

Casual users and persons with occasional need for information from the database interact using some form of interface, which we call the **interactive query** interface in Figure 2.3. We have not explicitly shown any menu-based or form-based interaction that may be used to generate the interactive query automatically. These queries are parsed and validated for correctness of the query syntax, the names of files and

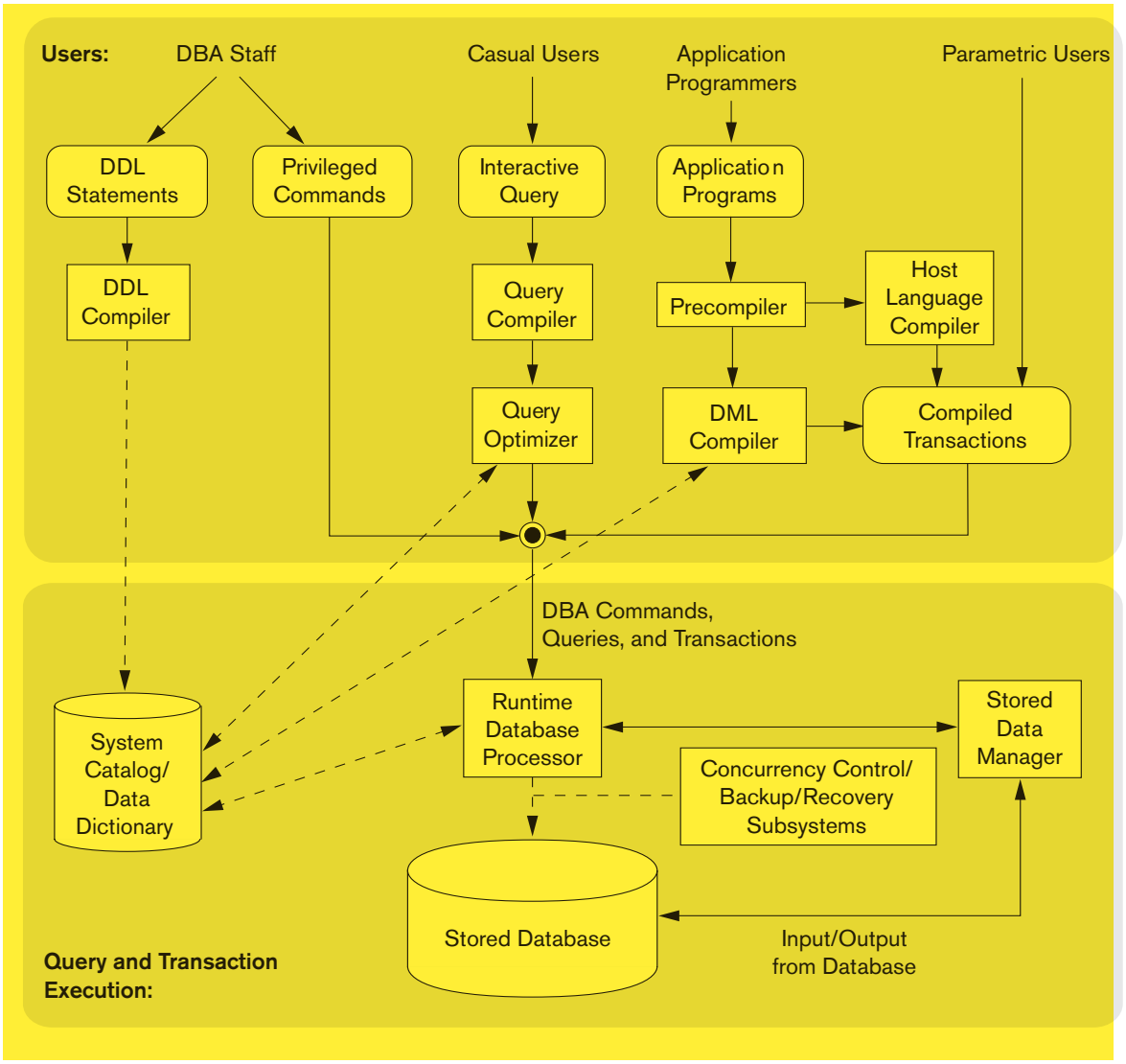


Figure 2.3
Component modules of a DBMS and their interactions.

data elements, and so on by a **query compiler** that compiles them into an internal form. This internal query is subjected to query optimization (discussed in Chapters 19 and 20). Among other things, the **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution. It consults the system catalog for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor.

Application programmers write programs in host languages such as Java, C, or C++ that are submitted to a precompiler. The **precompiler** extracts DML commands from an application program written in a host programming language. These commands are sent to the DML compiler for compilation into object code for database access. The rest of the program is sent to the host language compiler. The object codes for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor. Canned transactions are executed repeatedly by parametric users, who simply supply the parameters to the transactions. Each execution is considered to be a separate transaction. An example is a bank withdrawal transaction where the account number and the amount may be supplied as parameters.

In the lower part of Figure 2.3, the **runtime database processor** executes (1) the privileged commands, (2) the executable query plans, and (3) the canned transactions with runtime parameters. It works with the **system catalog** and may update it with statistics. It also works with the **stored data manager**, which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory. The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory. Some DBMSs have their own buffer management module while others depend on the OS for buffer management. We have shown **concurrency control** and **backup and recovery systems** separately as a module in this figure. They are integrated into the working of the runtime database processor for purposes of transaction management.

It is now common to have the **client program** that accesses the DBMS running on a separate computer from the computer on which the database resides. The former is called the **client computer** running a DBMS client software and the latter is called the **database server**. In some cases, the client accesses a middle computer, called the **application server**, which in turn accesses the database server. We elaborate on this topic in Section 2.5.

Figure 2.3 is not meant to describe a specific DBMS; rather, it illustrates typical DBMS modules. The DBMS interacts with the operating system when disk accesses—to the database or to the catalog—are needed. If the computer system is shared by many users, the OS will schedule DBMS disk access requests and DBMS processing along with other processes. On the other hand, if the computer system is mainly dedicated to running the database server, the DBMS will control main memory buffering of disk pages. The DBMS also interfaces with compilers for general-purpose host programming languages, and with application servers and client programs running on separate machines through the system network interface.

2.4.2 Database System Utilities

In addition to possessing the software modules just described, most DBMSs have **database utilities** that help the DBA manage the database system. Common utilities have the following types of functions:

- **Loading.** A loading utility is used to load existing data files—such as text files or sequential files—into the database. Usually, the current (source) for-

mat of the data file and the desired (target) database file structure are specified to the utility, which then automatically reformats the data and stores it in the database. With the proliferation of DBMSs, transferring data from one DBMS to another is becoming common in many organizations. Some vendors are offering products that generate the appropriate loading programs, given the existing source and target database storage descriptions (internal schemas). Such tools are also called **conversion tools**. For the hierarchical DBMS called IMS (IBM) and for many network DBMSs including IDMS (Computer Associates), SUPRA (Cincom), and IMAGE (HP), the vendors or third-party companies are making a variety of conversion tools available (e.g., Cincom's SUPRA Server SQL) to transform data into the relational model.

- **Backup.** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. The backup copy can be used to restore the database in case of catastrophic disk failure. Incremental backups are also often used, where only changes since the previous backup are recorded. Incremental backup is more complex, but saves storage space.
- **Database storage reorganization.** This utility can be used to reorganize a set of database files into different file organizations, and create new access paths to improve performance.
- **Performance monitoring.** Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files or whether to add or drop indexes to improve performance.

Other utilities may be available for sorting files, handling data compression, monitoring access by users, interfacing with the network, and performing other functions.

2.4.3 Tools, Application Environments, and Communications Facilities

Other tools are often available to database designers, users, and the DBMS. CASE tools[12] are used in the design phase of database systems. Another tool that can be quite useful in large organizations is an expanded **data dictionary** (or **data repository**) **system**. In addition to storing catalog information about schemas and constraints, the data dictionary stores other information, such as design decisions, usage standards, application program descriptions, and user information. Such a system is also called an **information repository**. This information can be accessed *directly* by users or the DBA when needed. A data dictionary utility is similar to the DBMS catalog, but it includes a wider variety of information and is accessed mainly by users rather than by the DBMS software.

[12]Although CASE stands for computer-aided software engineering, many CASE tools are used primarily for database design.

Application development environments, such as PowerBuilder (Sybase) or JBuilder (Borland), have been quite popular. These systems provide an environment for developing database applications and include facilities that help in many facets of database systems, including database design, GUI development, querying and updating, and application program development.

The DBMS also needs to interface with **communications software**, whose function is to allow users at locations remote from the database system site to access the database through computer terminals, workstations, or personal computers. These are connected to the database site through data communications hardware such as Internet routers, phone lines, long-haul networks, local networks, or satellite communication devices. Many commercial database systems have communication packages that work with the DBMS. The integrated DBMS and data communications system is called a **DB/DC** system. In addition, some distributed DBMSs are physically distributed over multiple machines. In this case, communications networks are needed to connect the machines. These are often **local area networks (LANs)**, but they can also be other types of networks.

2.5 Centralized and Client/Server Architectures for DBMSs

2.5.1 Centralized DBMSs Architecture

Architectures for DBMSs have followed trends similar to those for general computer system architectures. Earlier architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality. The reason was that most users accessed such systems via computer terminals that did not have processing power and only provided display capabilities. Therefore, all processing was performed remotely on the computer system, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.

As prices of hardware declined, most users replaced their terminals with PCs and workstations. At first, database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a **centralized** DBMS in which all the DBMS functionality, application program execution, and user interface processing were carried out on one machine. Figure 2.4 illustrates the physical components in a centralized architecture. Gradually, DBMS systems started to exploit the available processing power at the user side, which led to client/server DBMS architectures.

2.5.2 Basic Client/Server Architectures

First, we discuss client/server architecture in general, then we see how it is applied to DBMSs. The **client/server architecture** was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, data-