Course Contents

Unit-01: Database Concepts and Architecture (4 Hrs.)

- Database, Database Management System, Database Users, and Benefits of Databases;
- Data Models, Schemas, and Instances;
- Three-Schema Architecture and Data Independence;
- Database Languages and Interfaces;
- Database System Environment;
- Centralized and Client/Server Architectures for DBMSs;
- Classification of Database Management Systems

Course Contents

Unit-01: Database Concepts and Architecture (4 Hrs.)

- Database, Database Management System, Database Users, and Benefits of Databases;
- Data Models, Schemas, and Instances;
- Three-Schema Architecture and Data Independence;
- Database Languages and Interfaces;
- Database System Environment;
- Centralized and Client/Server Architectures for DBMSs;
- Classification of Database Management Systems

Database Concepts and Architecture

Database, Database Management System, Database Users, and Benefits of Databases:

After reading, answer these questions.:

1. Differentiate between Data, Database, Database Management System and Relational Database Management System?
2. Traditional Database Application
3. What are different types of Database Users?
4. What are the Advantages of using DBMS?

Database Concepts and Architecture

Unit-1: Database Concepts and Architecture

- Functions and Job of any organization is the “ Data Management” upon which the Efficiency is measured.
- Data were or are still managing in paper and saved all related documents in the hardcopy file folder called “ Paper based Data Management System”
- After computer evolution, Data are managing by two ways
    - Every department have their own master and transaction files maintained and processed separately in their own department.
        - OR,
    - Data manage centrally and accessed by all staffs.

Database Concepts and Architecture

- Data Management System Key problems with managing separately by each department: Data redundancy
- Data redundancy leads to lack of data integrity and consistency.

Database Concepts and Architecture

Database concept (Data, Database, DBMS and RDBMS)

- Data is a known facts that can be recorded and that have implicit meaning. For example, names, telephone numbers, and addresses of the people you know. Nowadays, this data is typically stored in mobile phones, which have their own simple database software. Each and every organisations collect huge range of data and use in their decision making as and when required.
- Database is a collection of related data (data & associated objects) which help to organize the related information in a logical fashion for easy access and retrieval. A database, then, is a repository for stored data. This collection of related data with an implicit meaning is a database. A database can be of any size and complexity.
- The overall purpose of database is to record and maintain information required by organization in decision making.

Database Concepts and Architecture

- A Database Management System (DBMS) is a computerized system that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data.
- A Database Management System (DBMS) is a software package designed to store and manage databases.
- Data that is stored more or less permanently in a computer, we term a Database. The software that allows one or many persons to use and/or modify this database is a Database Management System (DBMS).

Database Concepts and Architecture

- A Relational Database Management System (RDBMS) stores data in many related Tables. The user can ask complex questions from one or more of these related tables and answers come back to the user as Forms and Reports.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Database Concepts and Architecture

- Traditional Database Application: in which most of the information that is stored and accessed is either textual or numeric.
- Big data storage systems, or NOSQL systems : The creation of social media Web sites, such as Facebook, Twitter, and Flickr, among many others, has required the creation of huge databases that store nontraditional data, such as posts, tweets, images, and video clips. This New types of database systems, often referred to as big data storage systems, or NOSQL systems, have been created to manage data for social media applications. These types of systems are also used by Google, Amazon, and Yahoo, to manage the data required in their Web search engines, as well as to provide cloud storage, whereby users are provided with storage capabilities on the Web for managing all types of data including documents, programs, images, videos and emails.

Database Concepts and Architecture

File-Oriented Approach and Database Approach:

- Data was stored in files and some organization is also storing now.
- For an application, multiple files are required to be created. Each file stores and maintains its own related data.

For example, a student information system would include files like student profile, student course, student result, student fees etc. The application is built on top of the file system.

However, there are many drawbacks of using the file system.

Database Concepts and Architecture

File-Oriented Approach and Database Approach:

- Data redundancy means storing the same data at multiple locations. In an application, a file may have fields that are common to more than one file. The data for these common fields is thus replicated in all the files having these fields. This results in data redundancy, as more than one file has the same data values stored in them.

For example, student_name and student_course may be stored in two files: “student profile” and “student fees”.

- Data inconsistency: having different data values for the common fields in different files. During the updating process, the common fields may not get updated in all the files. This may result in different data values for the common fields in different files.

For example, a student having different home address in two different files. Data redundancy provides opportunity for data inconsistency.

Database Concepts and Architecture

File-Oriented Approach and Database Approach:

- Different file format for different files: The files in which the data is stored can have different file formats. This results in difficulty in accessing the data from the files since different methods are required for accessing the data from the files having different formats.
- Constraints difficult to apply: In a file system, the constraints of the system (for example, student age >17) become part of the program code. Adding new constraints or changing an existing one becomes difficult.
- Concurrency lead to inconsistency and security problems: The files can be accessed concurrently by multiple users. Uncontrolled concurrent access may lead to inconsistency and security problems.

For example, two users may try to update the data in a file at the same time.

“Database approach provides solutions for handling the problems of the file system approach”

Database Concepts and Architecture

Characteristics of Database Approach:

Main characteristics of the database approach:

- Data Redundancy is Minimized: Database system keeps data at one place in the database. The data is integrated into a single, logical structure. Different applications refer to the data from the centrally controlled location. The storage of the data, centrally, minimizes data redundancy.
- Data Inconsistency is Reduced: Minimizing data redundancy using database system reduces data inconsistency too. Updating of data values becomes simple and there is no disagreement in the stored values.
- Data is Shared: Data sharing means sharing the same data among more than one user. Each user has access to the same data, though they may use it for different purposes. Authorized users are permitted to use the data from the database. Users are provided with views of the data to facilitate its use.

Database Concepts and Architecture

Characteristics of Database Approach:

- Data Independence: It is the separation of data description (metadata) from the application programs that use the data. Data descriptions are stored in a central location called the data dictionary. This property allows an organization’s data to change and evolve (within limits) without changing the application programs that process the data.
- Data Integrity maintain: Stored data is changed frequently for variety of reasons such as adding new data item types, and changing the data formats. The integrity and consistency of the database are protected using constraints on values that data items can have.
- Data Security improve: The database is kept secure by limiting access to the database by authorized personnel. Authorized users are generally restricted to the particular data they can access, and whether they can update it or not. Access is often controlled by passwords.

Database Concepts and Architecture

Characteristics of Database Approach:

- Backup and Recovery Support : Backup and recovery are supported by the software that logs changes to the database. This support helps in recovering the current state of the database in case of system failure.
- Standards are Enforced: Since the data is stored centrally, it is easy to enforce standards on the database. Standards could include the naming conventions, and standard for updating, accessing and protecting data. Tools are available for developing and enforcing standards.
- Application Development Time is Reduced: The database approach greatly reduces the cost and time for developing new business applications. Programmer can focus on specific functions required for the new application, without having to worry about design, or low-level implementation details; as related data have already been designed and implemented. Tools for the generation of forms and reports are also available.

Database Concepts and Architecture

Database Users:

For a small personal database, one person typically defines, constructs, and manipulates the database, and there is no sharing. However, in large organizations, many people are involved in the design, use, and maintenance of a large database with hundreds or thousands of users.

So, users may be divided into

- **Actors on the Scene:** The people whose jobs involve the day-to-day use of a large database; we call them the actors on the scene. Those who actually use and control the database content, and those who design, develop and maintain database applications (called "**Actors on the Scene**").
- **Workers Behind the Scene:** Those who design and develop the DBMS software and related tools, and the computer systems operators (called "**Workers Behind the Scene**").

Database Concepts and Architecture

Actors on the Scene:

1. Database Administrators: In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. The DBA is accountable for problems such as security breaches and poor system response time.

2. Database Designers: Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

Database Concepts and Architecture

3. End Users: End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:

- Casual end users: occasionally access the database, but they may need different information each time. They use a sophisticated database query interface to specify their requests and are typically middle- or high-level managers.
- Naive or parametric end users: Main job function revolves around constantly querying and updating the database, using standard types of queries and updates.
- Sophisticated end users: include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirement.
- Standalone users: maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a financial software package that stores a variety of personal financial data.

Database Concepts and Architecture

Actors on the Scene:

A typical DBMS provides multiple facilities to access a database.

- Naive end users need to learn very little about the facilities provided by the DBMS; they simply have to understand the user interfaces of the apps or standard transactions designed and implemented for their use.
- Casual users learn only a few facilities that they may use repeatedly.
- Sophisticated users try to learn most of the DBMS facilities in order to achieve their complex requirements.
- Standalone users typically become very proficient in using a specific software package.

Database Concepts and Architecture

4. System Analysts and Application Programmers (Software Engineers):

System analysts determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements.

Application programmers implement these specifications as programs; then they test, debug, document, and maintain these transactions. Such analysts and programmers - commonly referred to as software developers or Software engineers - should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

Database Concepts and Architecture

Workers behind the Scene:

In addition to those who design, use, and administer a database, others are associated with the design, development, and operation of the DBMS software and system environment. These persons are typically not interested in the database content itself. We call them the workers behind the scene, and they include the following categories:

1. DBMS system designers and implementers design and implement the DBMS modules and interfaces as a software package. A DBMS is a very complex software system that consists of many components, or modules, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security. The DBMS must interface with other system software, such as the operating system and compilers for various programming languages.

Database Concepts and Architecture

Workers behind the Scene:

2. Tool developers design and implement tools—the software packages that facilitate database modeling and design, database system design, and improved performance. Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation. In many cases, independent software vendors develop and market these tools.

3. Operators and maintenance personnel (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system.

Although these categories of workers behind the scene are instrumental in making the database system available to end users, they typically do not use the database contents for their own purposes.

Database Concepts and Architecture

Advantages of using DBMS:

1. Controlling Redundancy
2. Restricting Unauthorized Access- A DBMS provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions and other previlages.
3. Providing Persistent Storage for Program Objects - a complex object in C++ can be stored permanently in an object-oriented DBMS. Such an object is said to be persistent, since it survives the termination of program execution and can later be directly retrieved by another program.
4. Providing Storage Structures and Search Techniques for Efficient Query Processing -Database systems must provide capabilities for efficiently executing queries and updates. Because the database is typically stored on disk, the DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records. Techniques like Indexing, Buffering, Caching etc
5. Providing Backup and Recovery - provide facilities for recovering from hardware or software failures.
6. Providing Multiple User Interfaces – Different types of users with varying levels of technical knowledge use a database and DBMS should provide a variety of user interfaces. These include apps for mobile users, query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for standalone users.
7. Representing Complex Relationships among Data - A 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 and update related data easily and efficiently.

Database Concepts and Architecture

Advantages of using DBMS:

8. Enforcing Integrity Constraints – Key constraints, Referential Integrity constraints, Business rule, Semantic
9. Permitting Inferencing and Actions Using Rules and Triggers -A trigger is a form of a rule activated by updates to the table, which results in performing some additional operations to some other tables, sending messages, and so on.
10. Additional Implications of Using the Database Approach
    - Potential for Enforcing Standards.
    - Reduced Application Development Time.
    - Flexibility - change the structure of a database as requirements change.
    - Availability of Up-to-Date Information - As soon as one user’s update is applied to the database, all other users can immediately see this update.
    - Economies of Scale- invest in more powerful processors, storage devices, or networking gear, rather than having each department purchase its own (lower performance) equipment.

Course Contents

Unit-01: Database Concepts and Architecture (4 Hrs.)

- Database, Database Management System, Database Users, and Benefits of Databases;
- Data Models, Schemas, and Instances;
- Three-Schema Architecture and Data Independence;
- Database Languages and Interfaces;
- the Database System Environment;
- Centralized and Client/Server Architectures for DBMSs;
- Classification of Database Management Systems

Course Contents

Database System Concepts and Architecture:

- The evolution computing trends replaced large centralized mainframe computers by hundreds of distributed workstations and personal computers connected via communications networks to various types of server machines—Web servers, database servers, file servers, application servers, and so on.
- The current cloud computing environments consist of thousands of large servers managing so-called big data for users on the Web.
- In a basic client/server DBMS architecture, the system functionality is distributed between two types of modules- Client Module and Server Module.

Course Contents

Database System Concepts and Architecture:

- A client module is designed to run on a mobile device, user workstation, or personal computer (PC).
- Application programs and user interfaces that access the database run in the client module. Hence, the client module handles user interaction and provides the user-friendly interfaces such as apps for mobile devices, or forms- or menu based GUIs (graphical user interfaces) for PCs. Where as a server module, handles data storage, access, search, and other functions.

Course Contents

Database System Concepts and Architecture:

- We will learn about Centralized and Client/Server Architectures for DBMSs.
- But before to learn about Centralized and Client/Server Architectures,
- We learn the terminology and basic concepts about Data Model and concept of schemas and instances that will be used throughout the course, which are fundamental to the study of database systems.
- We further discuss the three-schema DBMS architecture and data independence and provides a user's perspective on what a DBMS is supposed to do. We learn the types of interfaces and languages that are provided by a DBMS and about the database system software environment.
- Finally, a classification of the types of DBMS packages.

Course Contents

Data Models, Schemas, and Instances:

Data models are a collection of conceptual tools for describing **data**, **data relationships**, **data semantics** and **data constraints**. It is a graphical representation of data.

Data models are groups are as:

1. Entity-Relationship Model
2. Hierarchical
3. Network Models
4. Relational Model

Course Contents

Database Models?

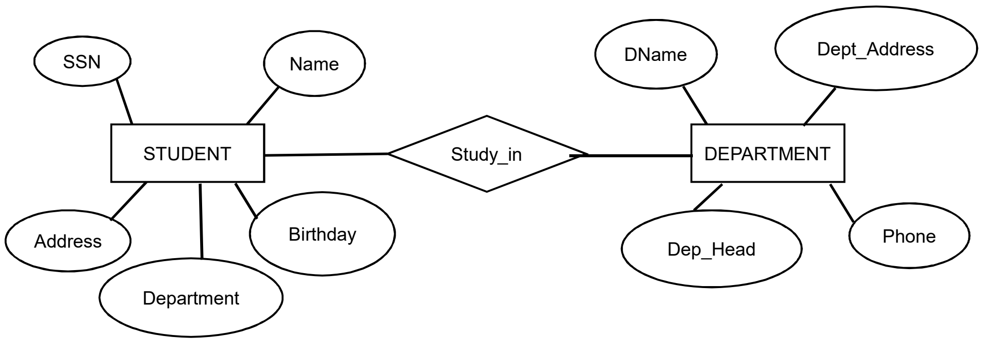
E-R Diagram:

- The entity-relationship model is based on a perception of the world as consisting of a collection of basic objects (entities) and relationships among these objects.
- An entity is a distinguishable objects that exists and we want to keep the information about. Each entity has associated with it a set of attributes describing it.

Course Contents

Database Models?

E-R Diagram:



Course Contents

Database Models?

Relational Model:

- Data and relationships are represented by a collection of tables.
- Each table has a number of columns with unique names, e.g. STUDENT, DEPARTMENT

Course Contents

Database Models?

Relational Model:

STUDENT

SSN	Name	Address	Department	Birthday
001	Shankar	Kathmandu	Computer Sc.	04/01/2031
002	Hari	Lalitpur	Computer Sc.	04/02/2032
003	Shyam	Lalitpur	Physics	04/03/2033
004	Ram	Kathmandu	Physics	05/12/2034
005	Deepak	Kirtipur	Chemistry	06/11/2035

DEPARTMENT

DName	Dept_Address	Dep_Head	Phone
Computer Sc.	Lalitpur	AAA	4444444
Physics	Lalitpur	BBB	5555555
Chemistry	Lalitpur	CCC	6666666

Course Contents

Database Models?

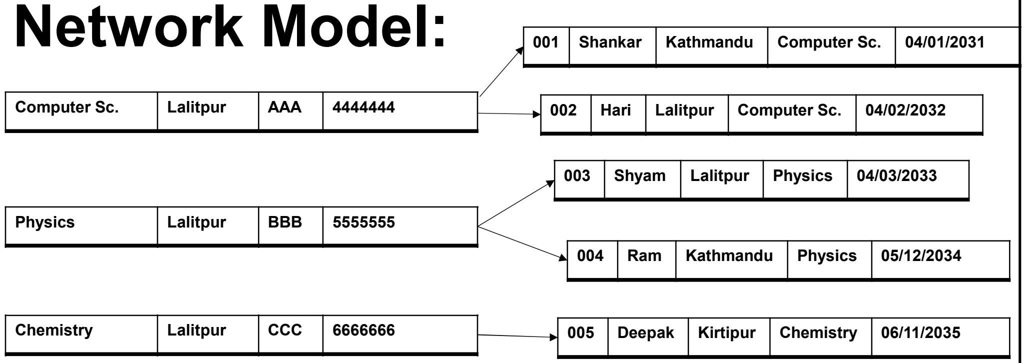
Network Model:

- Data are represented by collections of records.
- Relationships among data are represented by links.
- Organization is that of an **arbitrary graph**

Course Contents

Database Models?

Network Model:



Course Contents

Database Models?

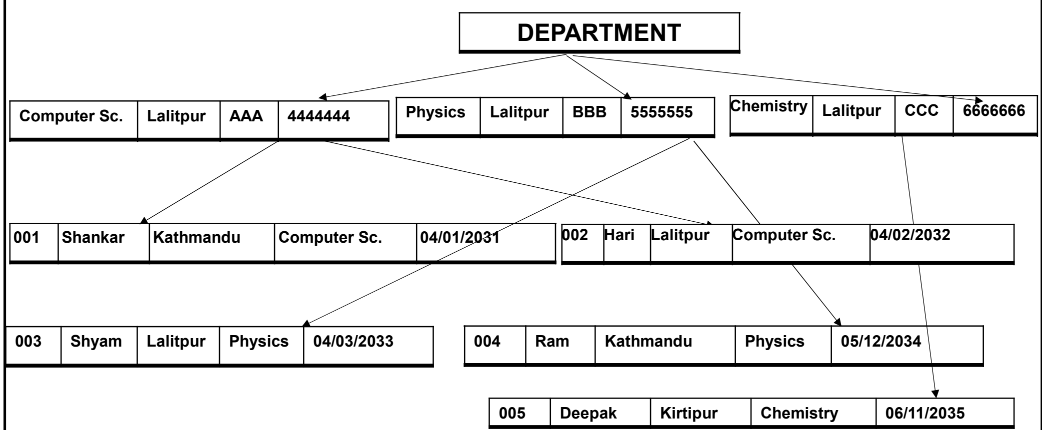
Hierarchical Model:

- Similar to the network model.
- Organization of the records is as a collection of trees, rather than arbitrary graphs.

Course Contents

Database Models?

Hierarchical Model:



Course Contents

Data Models:

- Database approach provides some level of data abstraction.
- Data abstraction generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data. One of the main characteristics of the database approach is to support data abstraction so that different users can perceive data at their preferred level of detail.
- A data model - a collection of concepts that can be used to describe the structure of a database - provides the necessary means to achieve this abstraction. By structure of a database mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database.

Course Contents

Data Models:

In addition to the basic operations provided by the data model, it is becoming more common to include concepts in the data model to specify the dynamic aspect or behavior of a database application.

This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects.

An example of a user-defined operation could be COMPUTE_GPA, which can be applied to a STUDENT object. On the other hand, generic operations to insert, delete, modify, or retrieve any kind of object are often included in the basic data model operations.

Concepts to specify behavior are fundamental to object-oriented data models but are also being incorporated in more traditional data models.

Course Contents

Categories of Data Models:

Many data models have been proposed, which we can categorize according to the database structure they use.

1. High-level or conceptual data models - Entity–Relationship model
2. Representational (or implementation) data models - relational data model, network and hierarchical models.
3. Object data model – UML Diagrams
4. Physical data models – How data stored in hard-disk (Sequential, Hashing, Clustering)
5. Self-describing data models - Meta data, XML (eXtensible Markup Language)

Course Contents

Categories of Data Models:

Many data models have been proposed, which we can categorize according to the types of concepts they use to describe the database structure.

High-level or conceptual data models provide concepts that are close to the way many users perceive data, whereas low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks.

Concepts provided by physical data models are generally meant for computer specialists, not for end users. Between these two extremes is a class of representational (or implementation) data models, which provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.

Course Contents

Categories of Data Models:

1. High-level or conceptual data models: Entity–Relationship model- a popular high-level conceptual data model.

2. Representational (or implementation) data models: most frequently use in traditional commercial DBMSs. These include the widely used relational data model, as well as the so-called legacy data models-the network and hierarchical models - that have been widely used in the past. Representational data models represent data by using record structures and hence are sometimes called record-based data models.

3. Object data model: an example of a new family of higher-level implementation data models that are closer to conceptual data models. A standard for object databases called the ODMG object model has been proposed by the Object Data Management Group (ODMG).

Course Contents

Categories of Data Models:

4. Physical data models: describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths. An access path is a search structure that makes the search for particular database records efficient, such as indexing or hashing.

5. Self-describing data models: The data storage in systems based on these models combines the description of the data with the data values themselves. In traditional DBMSs, the description (schema) is separated from the data. These models include XML (eXtensible Markup Language) as well as many of the key-value stores and NOSQL systems that were recently created for managing big data.

Schemas and Instances

- The current contents of a database, we term an instance of the database.
- When the database is designed, we are interested in plans(Schema) for the database. But when it is used, we are concerned with the actual data present in the database. The data in a database changes frequently, while the plans remain the same over long period of time.
- So Schema is a plan where data(instances) are changing.

Schemas and Instances

Example:

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.

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.

Course Contents

Schemas, Instances, and Database State:

In a data model, it is important to distinguish between the description of the database and the database itself.

The description of a database is called the database schema, which is specified during database design and is not expected to change frequently. Most data models have certain conventions for displaying schemas as diagrams. A displayed schema is called a schema diagram. The diagram displays the structure of each record type but not the actual instances of records.

Schema Diagram for database:

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

We call each object in the schema- such as STUDENT or COURSE - a schema construct.

Course Contents

Schemas, Instances, and Database State:

A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints. Other aspects like data type of each data item, the relationships among the various files are not specified in the schema diagram. Many types of constraints are not represented in schema diagrams.

Schema Diagram for database:

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

Course Contents

Schemas, Instances, and Database State:

The data in the database at a particular moment in time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database.

The actual data in a database may change quite frequently. For example, the database shown in Figure changes every time we add a new student or enter a new grade.

Although, the schema is not supposed to change frequently, but changes occasionally need to be applied to the schema as the application requirements change. For example, we may decide that another data item needs to be stored for each record in a file, such as adding the Date_of_birth to the STUDENT schema. This is known as schema evolution. Most modern DBMSs include some operations for schema evolution that can be applied while the database is operational.

Course Contents

Schemas, Instances, and Database State:

The distinction between database schema and database state is very important. When we define a new database, we specify its database schema only to the DBMS. At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database is first inserted with the initial data. From then on, every time an update operation is applied to the database, we get another database state. At any point in time, the database has a current state. The DBMS is partly responsible for ensuring that every state of the database is a valid state—that is, a state that satisfies the structure and constraints specified in the schema. Hence, specifying a correct schema to the DBMS is extremely important and the schema must be designed with utmost care. The DBMS stores the descriptions of the schema constructs and constraints—also called the meta-data - in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to. The schema is sometimes called the intension, and a database state is called an extension of the schema.

Course Contents

Three-Schema Architecture and Data Independence:

Data Abstraction

Three-Schema Architecture

- Abstraction, in general, is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.
- The major purpose of a database system is to provide users with an abstract view of the system. The system hides certain details of how data is stored and created and maintained.
- Complexity should be hidden from database users.

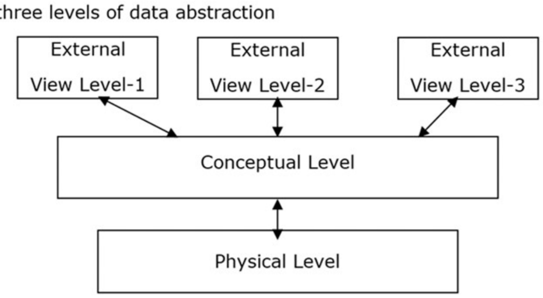
So, abstraction refers to the act of representing essential features without including the background details or explanations.

Data Abstraction

There are 3 levels of abstraction.

The levels form a three-level architecture that includes an external, a conceptual, and an internal(physical) level. The way users recognize the data is called the external level. The way the DBMS and the operating system distinguish the data is the internal level, where the data is stored using the data structures and file. The conceptual level offers both the mapping and the desired independence between the external and internal levels.

The three levels of data abstraction



Data Abstraction

There are 3 levels of abstraction.

1. **Physical Level:**
   - How the data are stored (e.g. index, B-tree, hashing.
   - Lowest level of abstraction.
   - Complex low-level structures described in detail.
2. **Conceptual Level:**
   - Next highest level of abstraction.
   - Describes what data are stored.
   - Describes the relationships among data.
   - Database administrator level.

Data Abstraction

There are 3 levels of abstraction.

3. **View Level:**
   - Highest level.
   - Describes part of the database for a particular group of users.
   - Can be many different views of a database.

Course Contents

Three-Schema Architecture and Data Independence:

Three of the four important characteristics of the database approach,

1. Self-describing nature of a database system - use of a catalog to store the database description (schema) so as to make it self-describing
2. Insulation between programs and data, and data abstraction - program-data and program-operation independence
3. Support of multiple views of the data
4. Sharing of data and multiuser transaction processing

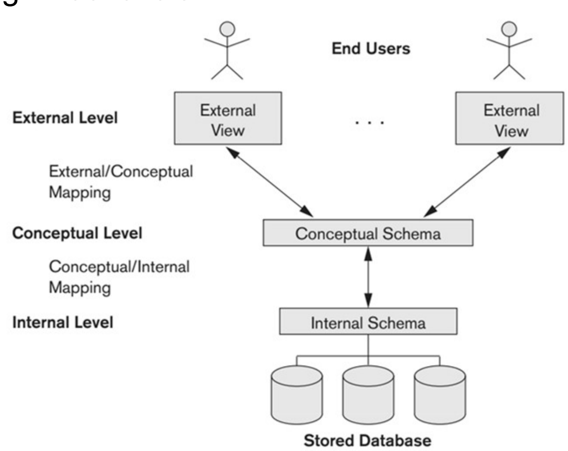
An architecture for database systems, called the three-schema architecture, to help achieve and visualize these characteristics.

We discuss further the concept of data independence.

Course Contents

The Three-Schema Architecture:

The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:



Course Contents

The Three-Schema Architecture:

The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

Course Contents

The Three-Schema Architecture:

3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level conceptual data model.

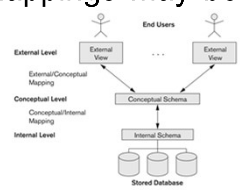
The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system.

Course Contents

The Three-Schema Architecture:

Notice that the three schemas are only descriptions of data while the actual data is stored at the physical level only.

In the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings. These mappings may be time-consuming.



Data Independence

The ability to modify a scheme definition in one level without affecting a schema definition in the next higher level is called data independence. From the three level of data abstraction, view to conceptual to physical database, provides two levels of Data Independence.

1. Physical data independence

- The ability to modify the physical scheme without causing application programs to be rewritten
- Modifications at this level are usually to improve performance

2. Logical data independence

- The ability to modify the conceptual scheme without causing application programs to be rewritten
- Usually done when logical structure of database is altered

Course Contents

Data Independence:

The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

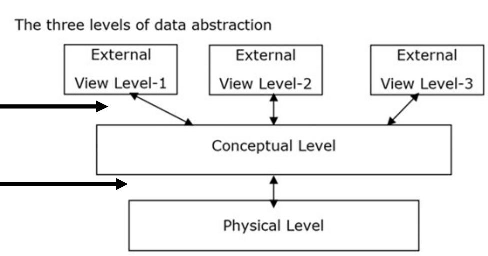
We can define two types of data independence:

1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).

Course Contents

Data Independence:

External schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before. Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.



Logical Data Independence

Physical Data Independence

Course Contents

Data Independence:

2. Physical data independence: is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized. For example, by creating additional access structures to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

Course Contents

Data Independence:

Generally, physical data independence exists in most databases and file environments where physical details, such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user. Applications remain unaware of these details.

On the other hand, logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs—a much stricter requirement.

Course Contents

Database Languages and Interfaces:

- We discussed the variety of users supported by a DBMS.
- The DBMS must provide appropriate languages and interfaces for each category of users.
- We learn the types of languages and interfaces provided by a DBMS and the user categories targeted by each interface.

Database Languages

Database Languages are mainly of two types:

- DDL (Data Definition Language)
- DML (Data Manipulation Language)

DDL can also be defined as

- SDL (Storage Definition Language)
- VDL (View Definition Language)

Database Languages

DDL (Data Definition Language):

- In ordinary high level Language, both Declaration and Executable statements are all part of one language.
- In DBMS, Declaration and Executables(Computation) into two different languages i.e. DDL and DML
- DDL Language is used to specify a database scheme as a set of definitions.
- DDL is not a procedural language. It is for describing the types of entities and relationships among entities.
- It is not used for obtaining or modify the data itself.

Database Languages

DML (Data Manipulation Language):

- A DML is a language which enables users to access and manipulate data.
- DML is for retrieval, insertion, deletion and modification of information from or in the database

There are two types of DML:

Procedural: the user specifies what data is needed and how to get it. E.g. SQL
Nonprocedural: the user only specifies what data is needed (E.g. QBE)

Course Contents

DBMS Languages:

Once the design of a database is completed and a DBMS is chosen to implement the database, the first step is to specify conceptual and internal schemas for the database and any mappings between the two.

- One language, called the data definition language (DDL), is used by the DBA and by database designers to define both schemas. The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. Another language, the storage definition language (SDL), is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.
- In most relational DBMSs today, there is no specific language that performs the role of SDL.

Course Contents

DBMS Languages:

We would need a third language, the view definition language (VDL), to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas. In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries.

Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the data manipulation language (DML) for these purposes.

Course Contents

DBMS Interfaces:

User-friendly interfaces provided by a DBMS may include the following:

1. Menu-based Interfaces for Web Clients or Browsing.
2. Apps for Mobile Devices.
3. Forms-based Interfaces.
4. Graphical User Interfaces.
5. Natural Language Interfaces.
6. Keyword-based Database Search.
7. Speech Input and Output.
8. Interfaces for Parametric Users.
9. Interfaces for the DBA.

Database Concepts and Architecture

The Database System Environment:

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

1. Users of the database environment and their interfaces
2. Internal modules of the DBMS
    - System Catalogue /Data Dictionary
    - Stored Database
    - Runtime database processor
    - Stored Data Manager
    - Concurrency Control / Backup/Recovery Subsystem

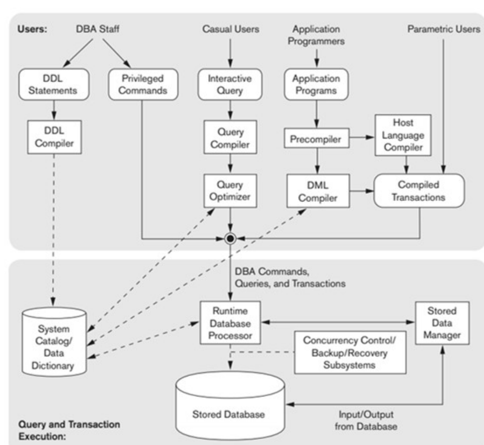


Figure 2.3
Component modules of a DBMS and their interactions.

Database Concepts and Architecture

DBMS Component Modules:

A simplified form of 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 internal modules of the DBMS responsible for storage of data and processing of transactions.

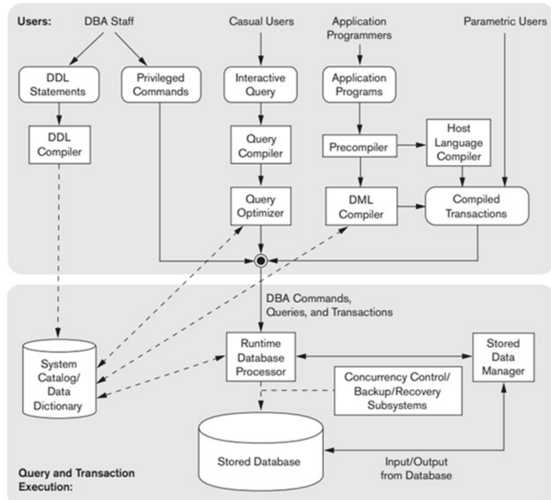


Figure 2.3
Component modules of a DBMS and their interactions.

Database Concepts and Architecture

DBMS Component Modules:

Stored Data Manager: The database and the DBMS catalog are usually stored on disk. 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. 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 management of buffer storage has a considerable effect on performance. Reducing disk read/write improves performance considerably.

- Various types of database users are shown on top of figure. Each users have their own jobs and functions and interfaces with different components of Database system.
- 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 (System Catlogue /Data Dictionary). 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.

Database Concepts and Architecture

DBMS Component Modules:

Casual users and persons: who occasionally need for information from the database interact using the interactive query interface. Interactive queries are in the form of menu-based or form-based or mobile interactions that are typically used to generate the interactive query automatically or to access transactions. These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a query compiler that compiles them into an internal form.

This internal query is passed through the query optimizer which rearrangement and possible reordering of operations, elimination of redundancies, and use of efficient search algorithms 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.

Database Concepts and Architecture

DBMS Component Modules:

Application programmers write programs in host languages such as Java, C, or C++ and 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 via PCs or mobile apps; these users simply supply the parameters to the transactions.

Database Concepts and Architecture

DBMS Component Modules:

Runtime Database Processor : The lower part of Figure, 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 whereas others depend on the OS for buffer management. The concurrency control and backup and recovery systems separately as a module in the figure. They are integrated into the working of the runtime database processor for purposes of transaction management.

Course Contents

Database System Utilities:

DBMSs have many software modules for possessing as described in Database System Environment. There are database utilities also in DBMS that help the DBA to manage the database system. Common utilities have the following types of functions:

1. Loading: A loading utility is used to load existing data files (text files or others) into the database. Usually, the current (source) format 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. Transferring data from one DBMS to another is becoming common in many organizations. Some vendors offer conversion tools that generate the appropriate loading programs, given the existing source and target database storage descriptions (internal schemas).

2. 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.

Course Contents

Database System Utilities:

3. 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.

4. Performance monitoring: This 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.

Course Contents

Tools, Application Environments, and Communications Facilities:
Other tools are often available to database designers, users, and the DBMS.

1. CASE tools are used in the design phase of database systems.

2. 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.

3. Application development environments: 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 i.e. PowerBuilder (Sybase) or Jbuilder.

Course Contents

Tools, Application Environments, and Communications Facilities:

4. Database Communication Interfaces: Different users may access the database through a network environment. So the DBMS provide communication functions to access the database through computer network environment. 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.

Centralized and Client/Server Architectures for DBMSs;

Two generic database architectures or types of database are:

1. Centralized DBMSs Architecture

2. Client/Server architecture Databases
   - Basic Client/Server Architectures
   - Two-Tier Client/Server Architectures for DBMSs
   - Three-Tier and n-Tier Architectures for Web Applications

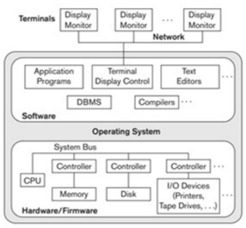
3. Distributed Database
   - Homogeneous Databases
   - Heterogeneous Databases

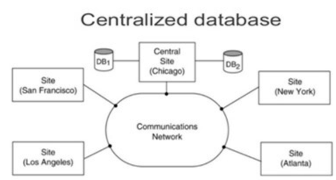
Centralized and Client/Server Architectures for DBMSs;

Centralized DBMSs Architecture

A centralized database is stored at a single location such as a mainframe computer. It is maintained and modified from that location only and usually accessed using an internet connection such as a LAN or WAN.

Older architectures used mainframe computers to provide the mainprocessing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality. The reason was that in older systems, most users accessed the DBMS via computer terminals that did not have processing power and only provided display capabilities.





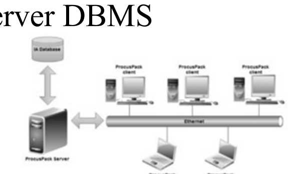
Centralized and Client/Server Architectures for DBMSs;

Client/Server architecture Databases:

The client/server databases are used in small to medium organization or businesses to share data among multiple users in local area network. The microcomputers are often used in a local area network.

The client/server architecture is designed for the distribution of work on a computer network in which many clients may share the data (or services). Here is an example of client/server database.

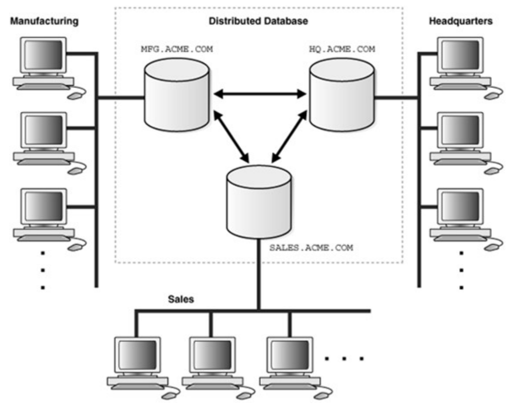
DBMS itself is still a centralized DBMS in which all the DBMS functionality, application program execution, and user interface processing is carried out on client machine. Gradually, DBMS systems started to exploit the available processing power at the user side, which led to client/server DBMS architectures.



Centralized and Client/Server Architectures for DBMSs;

Distributed Processing Database System:

A distributed database is a database that consists of two or more files located in different sites either on the same network or on entirely different networks. Portions of the database are stored in multiple physical locations and processing is distributed among multiple database nodes.



Course Contents

Basic Client/Server Architectures:

Course Contents

Two-Tier Client/Server Architectures for DBMSs:

Two main types of basic DBMS architectures were created on the Basic/Client Server Architecture underlying client/server framework: two-tier and three-tier.

Many of relational database management systems (RDBMSs) started as centralized systems. The system components that were first moved to the client side were the user interface and application programs. Because SQL provided a standard language for RDBMSs and this created a logical dividing point between client and server. Hence, the query and transaction functionality related to SQL processing remained on the server side. In such an architecture, the server is often called a query server or transaction server because it provides these two functionalities. In an RDBMS, the server is also often called an SQL server.

Course Contents

Two-Tier Client/Server Architectures for DBMSs:

The user interface programs and application programs can run on the client side. When DBMS access is required, the program establishes a connection to the DBMS (which is on the server side); once the connection is created, the client program can communicate with the DBMS.

A standard called Open Database Connectivity (ODBC) provides an application programming interface (API), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed.

Most DBMS vendors provide ODBC drivers for their systems. A client program can actually connect to several RDBMSs and send query and transaction requests using the ODBC API, which are then processed at the server sites. Any query results are sent back to the client program, which can process and display the results as needed. A related standard for the Java programming language, called JDBC, has also been defined. This allows Java client programs to access one or more DBMSs through a standard interface.

Course Contents

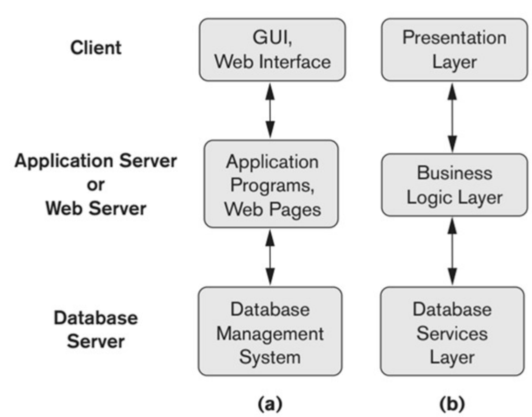
Two-Tier Client/Server Architectures for DBMSs:

The architectures are called two-tier architectures because the software components are distributed over two systems: client and server.

Course Contents

Three-Tier and n-Tier Architectures for Web Applications:

Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server, as shown in Figure.



Course Contents

Three-Tier and n-Tier Architectures for Web Applications:

This intermediate layer or middle tier is called the application server or the Web server, depending on the application. This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client's credentials before forwarding a request to the database server.

Clients contain user interfaces and Web browsers. The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database 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 the three tiers.

Course Contents

Three-Tier and n-Tier Architectures for Web Applications:

The presentation layer displays information to the user and allows data entry.

The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS.

The bottom layer includes all data management services.

The middle layer can also act as a Web server, which retrieves query results from the database server and formats them into dynamic Web pages that are viewed by the Web browser at the client side.

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

Course Contents

Classification of Database Management Systems:

Several criteria can be used to classify DBMSs.

1. Based on Data Model: relational, object, object-relational, NOSQL, key-value, hierarchical, network, and other.

2. Based on Number of users supported by the system: Single-user systems support only one user at a time and are mostly used with PCs. Multiuser systems, which include the majority of DBMSs, support concurrent multiple users.

3. Based on Number of sites over which the database is distributed: A DBMS is centralized if the data is stored at a single computer site. A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site. A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites connected by a computer network. Big data systems are often massively distributed, with hundreds of sites. The data is often replicated on multiple sites so that failure of a site will not make some data unavailable.

Course Contents

Classification of Database Management Systems:

Homogeneous DDBMSs use the same DBMS software at all the sites, whereas heterogeneous DDBMSs can use different DBMS software at each site. It is also possible to develop middleware software to access several autonomous preexisting databases stored under heterogeneous DBMSs.

4. Based on cost: Open source (free) DBMS products like MySQL and PostgreSQL that are supported by third-party vendors with additional services. Licenses - site licenses allow unlimited use of the database system with any number of copies running at the customer site. Another type of license limits the number of concurrent users or the number of user seats at a location. Standalone single-user versions of some systems like Microsoft Access are sold per copy or included in the overall configuration of a desktop or laptop. In addition, data warehousing and mining features, as well as support for additional data types, are made available at extra cost. It is possible to pay millions of dollars for the installation and maintenance of large database systems annually.

Course Contents

Classification of Database Management Systems:

5. General purpose or Special purpose. When performance is a primary consideration, a special-purpose DBMS can be designed and built for a specific application; such a system cannot be used for other applications without major changes. Many airline reservations and telephone directory systems developed in the past are special-purpose DBMSs. These fall into the category of online transaction processing (OLTP) systems, which must support a large number of concurrent transactions without imposing excessive delays.

Database Concepts and Architecture

Database system environment:

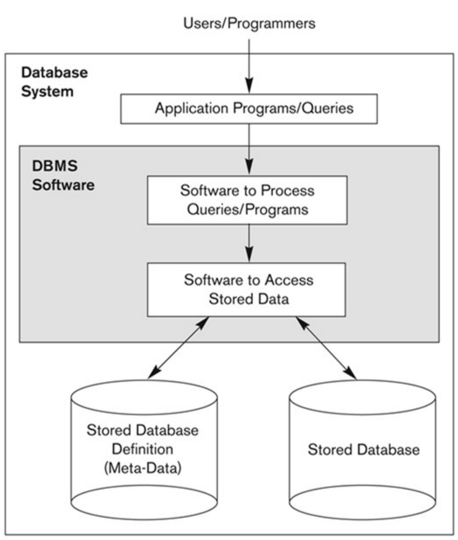


Figure 1.1
A simplified database system environment