INTRODUCTION OF DATABASE

Muhammad Amar Primus Firdaus
11230940000067
PTIK-Matematika

CONTENTS

CHAPTER I

Databases and Database Users

CHAPTER II

Database System Concepts And Architecture

CHAPTER 1

INTRODUCTION

what is a database ?

A database is a collection of interconnected data and the data contains known facts that can be recorded and that have implicit meaning.

For example, consider the names, phone numbers, and addresses of people you know. Nowadays, this data is usually stored in cell phones, which have simple database software. This data can also be recorded in an indexed address book or stored on a hard drive, using a personal computer and software such as Microsoft Access or Excel.

A database has the following implicit properties:

1. The database represents some aspect of the real world, sometimes called a mini-world or universe of discourse (UoD). Changes to the mini-world are reflected in the database.
2. A database is a logically coherent collection of data with some inherent meaning. A random collection of data cannot be properly called a database.
3. A database is designed, built and filled with data for a specific purpose. The database has an intended user group and some predefined applications that the users are interested in. In other words, the database has some sources from which the data comes, some level of interaction with events in the real world, and an audience that is actively interested in its contents.

AN EXAMPLE

An example a UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment.

To define this database, we must specify the structure of the records of each file by specifying the different types of data elements to be stored in each record.

The STUDENT file stores data on each student, the COURSE file stores data on each course, the SECTION file stores data on each section of a course, the GRADE_REPORT file stores the grades that students receive in the various sections they have completed, and the PREREQUISITE file stores the prerequisites of each course.

To define this database, we must specify the structure of the records of each file by specifying the different types of data elements to be stored in each record.

STUDENT

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

COURSE

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

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

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

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Source : Page 8 of fundamentals of database system

ACTORS ON THE SCENE

In this section we identify the people whose jobs involve the day-to-day use of a large database; we call them the actors on the scene.

A. Database Administrators

A chief administrator oversees and manages resources in an organization, including databases. The database administrator (DBA) manages access, monitors usage, and acquires necessary resources. They are responsible for security breaches and system response time, often assisted by a staff in large organizations.

B. Database Designers

Database designers identify and select data storage structures before database implementation. They communicate with potential users to understand their requirements and create a design that meets them. Designers interact with each user group, developing views that meet their data and processing requirements. The final design must support all user groups' requirements.

C. End Users

End users access databases for querying, updating, and generating reports. There are casual, sophisticated, and naive end users. Sophisticated end users use DBMS for applications, while standalone users use pre-made packages. A DBMS offers multiple facilities.

D. System Analysts and Application Programmers (Software Engineers)

System analysts determine end user requirements and create specifications for canned transactions. Application programmers implement these specifications, test, debug, document, and maintain these transactions, using DBMS capabilities.

WORKERS BEHIND THE SCENE

Workers behind the scene, such as those involved in database design, development, and operation, are not directly involved in the content of the database.We call them the workers behind the scene, and they include the following categories:

A. DBMS system designers and implementers

DBMS system designers and implementers create and execute complex software packages, including modules for catalog, query language processing, interface processing, data access, concurrency control, and data recovery and security.

B. Tool developers design and implement tools

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

C. Operators and maintenance personnel

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

ADVANTAGES OF USING THE DBMS APPROACH

In this section we discuss some additional advantages of using a DBMS and the capabilities that a good DBMS should possess.

A. Controlling Redundancy

Traditional software development often leads to redundancy and issues like duplication of effort and wasted storage space. Database approaches integrate views of different user groups, ensuring consistency and saving space. Data normalization stores logical data items in one place, but controlled redundancy may be necessary for improved query performance.

B. Restricting Unauthorized Access

A database management system (DBMS) should provide security and authorization subsystems for creating accounts and enforcing restrictions, ensuring only authorized staff can access privileged software or parametric users accessing the database through predefined apps or canned transactions.

C. Providing Persistent Storage for Program Objects

Object-oriented database systems offer persistent storage for program objects and data structures, compatible with programming languages like C++ and Java, enabling easy retrieval of persistent objects, overcoming impedance mismatch issues.

D. Providing Storage Structures and Search Techniques for Efficient Query Processing

Database systems (DBMS) need to efficiently execute queries and updates, using specialized data structures and search techniques. Indexes, auxiliary files, are used for disk search. DBMSs often have buffering or caching modules to maintain database parts in main memory. The query processing and optimization module selects an efficient query execution plan based on existing storage structures. Physical database design and tuning are responsibilities of the DBA staff.

ADVANTAGES OF USING THE DBMS APPROACH

E. Providing Backup and Recovery

A DBMS must offer recovery capabilities for hardware or software failures, with the backup and recovery subsystem responsible for restoring the database to its original state during complex updates or catastrophic disk failures.

F. Providing Multiple User Interfaces

A database management system (DBMS) should offer various user interfaces, including apps for mobile users, query languages for casual users, programming language interfaces for programmers, forms and command codes for parametric users, and menu-driven interfaces for standalone users.

G. Representing Complex Relationships among Data

A database can hold various types of interrelated data, such as student and section records. It must represent complex relationships, define new ones, and efficiently retrieve and update related data.

H. Enforcing Integrity Constraints

Database applications require integrity constraints for data security, which can be simple or complex. Database designers identify these constraints during design, which can be enforced automatically or manually. Errors in data entry must be discovered and corrected later.

I. Permitting Inferencing and Actions Using Rules and Triggers

Deductive database systems infer new information from stored data, useful for probation determination. Traditional DBMS require explicit code, while relational systems offer more flexibility and powerful functionality.

I. Permitting Inferencing and Actions Using Rules and Triggers

The database approach benefits organizations by defining standards, promoting communication, reducing application development time, allowing flexibility, providing up-to-date information.

A Brief History of Database Applications

A. Early Database Applications Using Hierarchical and Network Systems

Early database applications in large organizations struggled with insufficient data abstraction, program-data independence, and programming language interfaces. Implementing these systems on expensive mainframe computers from the mid-1960s to the 1970s and 1980s was time-consuming and expensive.

B. Providing Data Abstraction and Application Flexibility with Relational Databases

Relational databases, initially designed for storage and representation, introduced high-level query languages, improved data abstraction, and program-data independence, despite slow performance, enhancing their dominance in traditional applications.

C. Object-Oriented Applications and the Need for More Complex Databases

Object-oriented databases (OODBs) emerged in the 1980s, initially competing with relational databases but now mainly used in specialized applications like engineering design and manufacturing systems.

D. Interchanging Data on the Web for E-Commerce Using XML

The World Wide Web connects computers, enabling static HTML pages and e-commerce in the 1990s. Techniques like XML facilitate data interchange between databases and web pages.

E. Extending Database Capabilities for New Applications

Database systems are essential for various applications, requiring complex data structures, new data types, operations, and query language constructs, and are developed by DBMS developers for general or special-purpose purposes.

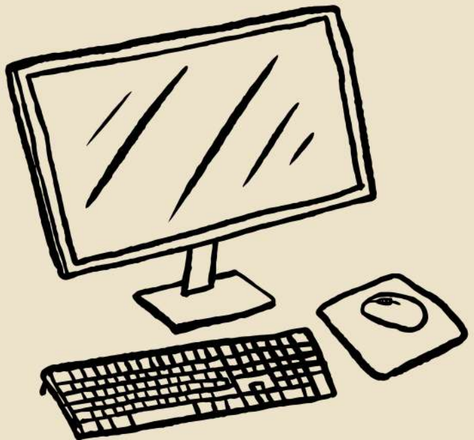
When Not to Use a DBMS

DBMS offers advantages but may incur unnecessary overhead costs due to the following factors:

- High initial investment in hardware, software, and training
- The generality that a DBMS provides for defining and processing data
- Overhead for providing security, concurrency control, recovery, and integrity functions

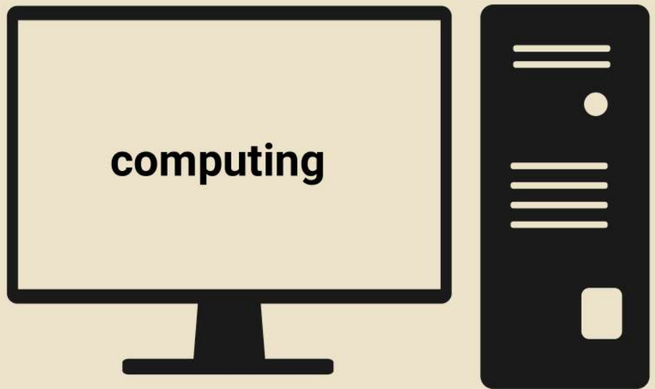
Therefore, it may be more desirable to develop customized database applications under the following circumstances:

- Simple, well-defined database applications that are not expected to change at all.
- Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead
- Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit
- No multiple-user access to data.



Summary

This chapter defines a database as a collection of related data used for specific purposes by users. A database management system (DBMS) is a software package for implementing and maintaining a computerized database. The chapter distinguishes the database approach from traditional file-processing applications and discusses the main categories of database users and support personnel. It also provides a list of capabilities for database designers and end users, discusses the evolution of database applications, and discusses overhead costs.



CHAPTER 2

Data Models, Schemas, and Instances

The database approach emphasizes data abstraction, highlighting essential features and defining a data model. It includes basic operations for retrievals and updates, as well as dynamic aspects for valid user-defined operations, fundamental to object-oriented and traditional models.

A. Categories of Data Models

Data models are categorized based on the concepts they use to describe database structure. High-level or conceptual data models provide concepts similar to users' perceptions, while low-level or physical data models describe details of data storage on computer storage media. Representational data models, on the other hand, provide concepts that are easily understood by end users but not too far removed from data organization in computer storage. Conceptual data models use concepts like entities, attributes, and relationships, representing real-world objects or concepts. Representational data models are commonly used in traditional commercial DBMSs, such as relational and legacy data models. Object data models are a new family of higher-level implementation data models that are closer to conceptual data models.

B. Schemas, Instances, and Database State

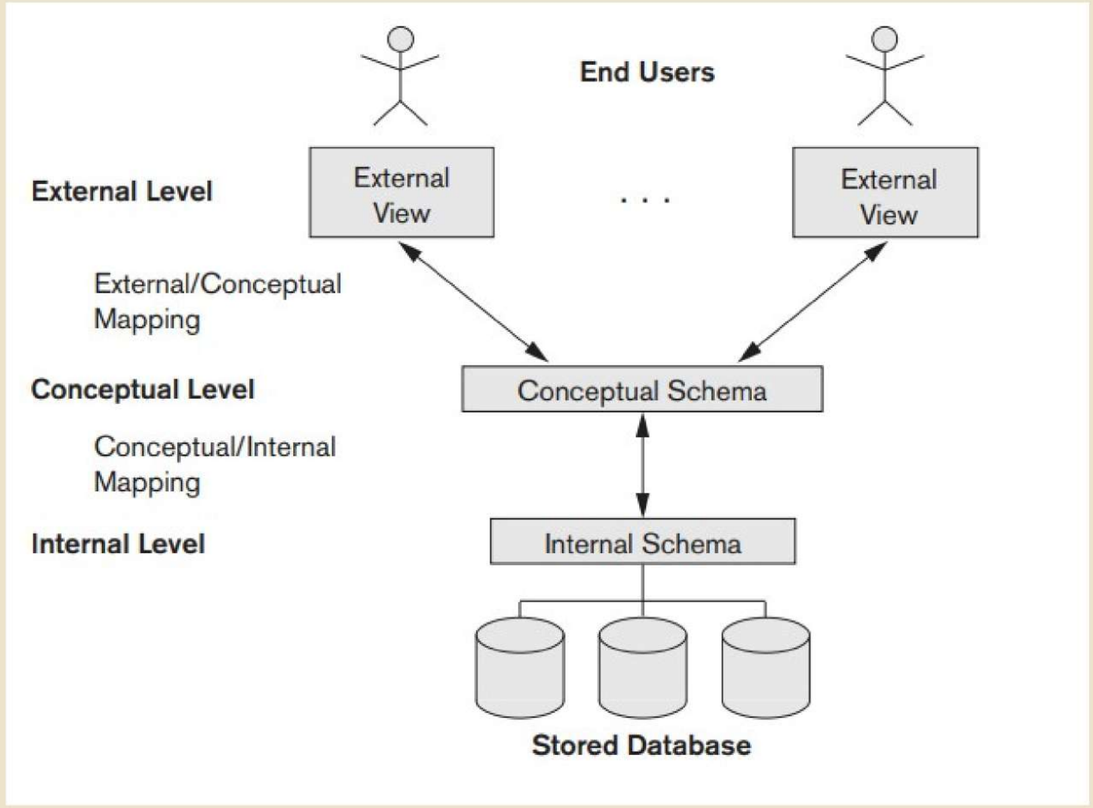
In a data model, it's essential to distinguish between the database schema and the database itself. The database schema is specified during design and is not expected to change frequently. Schema diagrams display aspects of a schema, but constraints are not represented. The actual data in a database may change frequently, and the database state is the empty state with no data. The DBMS stores schema constructs and constraints in the catalog, and schema evolution can be applied while the database is operational.

Three-Schema Architecture and Data Independence

A. The Three-Schema Architecture

The three-schema architecture aims to separate user applications from the physical database by defining schemas at three levels:

1. ***The internal schema*** at the internal level outlines the physical storage structure and access paths of the database using a physical data model.
2. ***The conceptual schema*** describes the database structure for a community of users, hiding physical storage details and focusing on entities, data types, relationships, user operations, and constraints. It is typically described using a representational data model, often based on a conceptual schema design in a high-level data model.
3. ***The external level*** of a database consists of external schemas or user views, each describing a specific group's interest in a specific part of the database and hiding the rest from them. These schemas are typically implemented using a representational data model.



Source : Page 37

Three-Schema Architecture and Data Independence

B. Data Independence

The three-schema architecture explains data independence, allowing schema changes at one level without needing to change at another, with two types of data independence defined.

1. Logical data independence allows for changes to the conceptual schema without affecting external schemas or application programs. This can be done to expand the database, change constraints, or reduce the database. However, external schemas referring only to the remaining data should not be affected. Only the view definition and mappings need to be changed in a DBMS that supports logical data independence. After a logical reorganization, application programs that reference external schema constructs should function as before. Changes to constraints can be applied to the conceptual schema without affecting external schemas or application programs.
2. Physical data independence allows for changes to internal schema without altering the conceptual schema, ensuring that external schemas do not need to be altered. Changes may be necessary for reorganizing physical files or improving retrieval or update performance. If the same data remains in the database, conceptual schema changes should not be necessary. For example, improving retrieval speed of SECTION records by semester and year does not require a change in the conceptual schema.

Database Languages and Interfaces

A. DBMS Languages

The design of a database involves specifying conceptual and internal schemas and mappings between them. In DBMSs, the data definition language (DDL) is used to define both schemas, while the storage definition language (SDL) specifies the internal schema. In relational DBMSs, the DDL is used to define both conceptual and external schemas, while SQL is used as the view definition language (VDL) to define user views.

Once database schemas are compiled and populated with data, users need to have means to manipulate the database. The data manipulation language (DML) is provided by the DBMS for retrieval, insertion, deletion, and modification. In current DBMSs, these languages are not distinct but rather a comprehensive integrated language that includes constructs for conceptual schema definition, view definition, and data manipulation.

B. DBMS Interfaces

A database management system (DBMS) offers various user-friendly interfaces, including menu-based interfaces for web clients or browsing, apps for mobile devices, forms-based interfaces, and graphic user interfaces. Menus allow users to simulate queries step-by-step, eliminating the need to memorize specific commands and syntax. Mobile devices provide access to user data, such as banking, reservations, and insurance companies. Forms-based interfaces display a form for users to fill out, allowing them to insert new data or retrieve matching data. Graphical user interfaces display a schema in diagrammatic form, allowing users to specify queries by manipulating the diagram.

The Database System Environment

A. DBMS Component Modules

The database and catalog are typically stored on disk, with access controlled by the operating system. Many DBMSs have buffer management modules to manage buffer storage, which improves performance. A higher-level stored data manager module controls access to disk-based information. Interfaces for DBA staff, casual users, application programmers, and parametric users.

B. Database System Utilities

Database Management Systems (DBMSs) often have database utilities that assist Database Administrators (DBAs) in managing the database system. These utilities include loading, which loads existing data files into the database, and backup, which creates a backup copy for disaster recovery. Database storage reorganization utilities can reorganize files and create new access paths for improved performance.

Performance monitoring utilities provide statistics to DBAs, helping them make decisions on file reorganization and index addition. Other utilities may include sorting files, handling data compression, user access monitoring, and network interfacing.

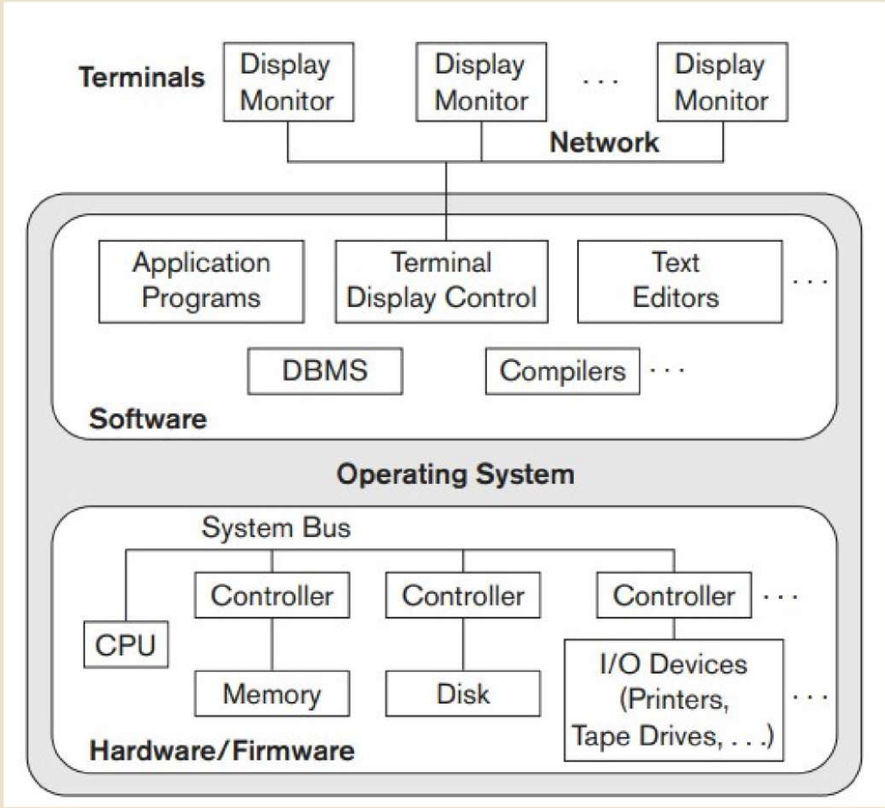
C. Tools, Application Environments,and Communications Facilitie

Database designers, users, and DBMS use tools like CASE and expanded data dictionary for design. Application development environments like PowerBuilder and JBuilder aid in GUI development, querying, updating, and program development. DB/DC systems integrate DBMS and data communications.

Centralized and Client/Server Architectures for DBMSs

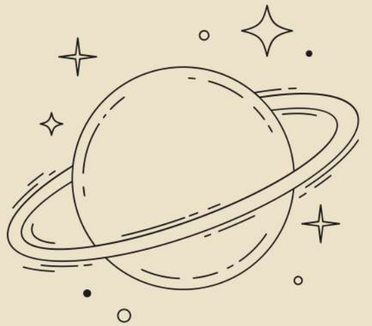
A. Centralized DBMSs Architecture

DBMS architectures have evolved over time, with older systems using mainframe computers for processing all system functions and user application programs. This was because users accessed the DBMS via display terminals, which had limited processing power. As hardware prices declined, users replaced terminals with PCs, workstations, and mobile devices. Initially, database systems used these computers similarly to display terminals, maintaining a centralized DBMS with all DBMS functionality.



Source : Page 47

Centralized and Client/Server Architectures for DBMSs



B. Basic Client/Server Architectures

The client/server architecture is a system designed for computing environments involving numerous devices connected via a network. It consists of specialized servers with specific functions, such as file servers, printer servers, and web servers. These servers can be accessed by multiple client machines, providing appropriate interfaces and local processing power for local applications. This concept can also be applied to other software packages, such as CAD packages, which are stored on specific server machines and accessible to multiple clients.

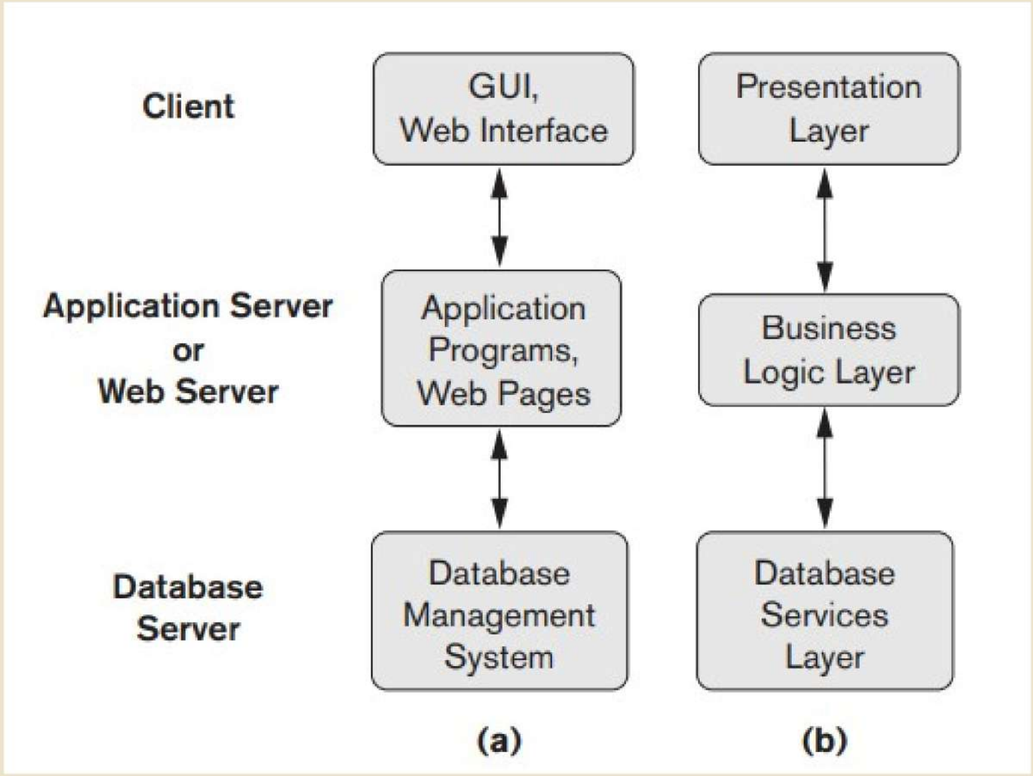
C. Two-Tier Client/Server Architectures for DBMSs

RDBMSs initially started as centralized systems with user interface and application programs on the client side. SQL provided a standard language for RDBMSs, creating a logical divide between client and server. The server is often called a query or transaction server. Client programs can run on the client side, establishing a connection to the DBMS. Open Database Connectivity (ODBC) provides an API for client-side programs to call the DBMS. Java programming language JDBC allows Java client programs to access DBMSs through a standard interface. Two-tier architectures are simple and seamless, while the three-tier architecture evolved with the Web's emergence.

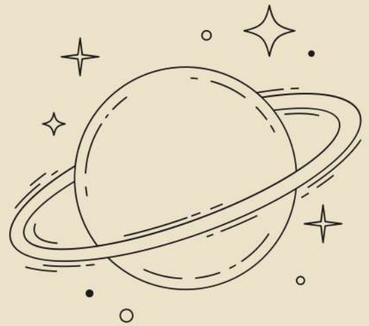
Centralized and Client/Server Architectures for DBMSs

D. Three-Tier and n-Tier Architectures for Web Applications

The three-tier architecture is a common design for web applications, which adds an intermediate layer between the client and the database server. This middle layer, called the application server or Web server, runs application programs and stores business rules for accessing data from the database server. It improves database security by checking a client's credentials before forwarding a request to the server. The presentation layer displays information to the user, while the business logic layer handles intermediate rules and constraints. The bottom layer includes all data management services. The middle layer can also act as a Web server, retrieving query results and formatting them into dynamic Web pages. Other architectures, such as n-tier architectures, can divide the layers between the user and stored data further into finer components. Advances in encryption and decryption technology make it safer to transfer sensitive data from server to client, but network security issues remain a major concern.



Classification of Database Management Systems



Database Management Systems (DBMSs) are categorized based on various factors, including data model, user support, site distribution, cost, and purpose. Relational databases use SQL and have limited user views. Object databases define databases in terms of objects, properties, and operations. Big data systems use key-value, document, graph, and column-based models. The XML model is a standard for exchanging data over the Web, using hierarchical tree structures and combining database concepts with document representation models. Legacy data models include network and hierarchical models.

Summary

In this chapter we introduced the main concepts used in database systems. We defined a data model and we distinguished three main categories:

- High-level or conceptual data models (based on entities and relationships)
- Low-level or physical data models
- Representational or implementation data models (record-based, objectoriented)

We distinguished the schema, or description of a database, from the database itself. The schema does not change very often, whereas the database state changes every time data is inserted, deleted, or modified. Then we described the three-schema DBMS architecture, which allows three schema levels:

- An internal schema describes the physical storage structure of the database.
- A conceptual schema is a high-level description of the whole database.
- External schemas describe the views of different user groups.

THANK YOU