RHEMA UNIVERSITY, ABA

COLLEGE OF BASIC & APPLIED SCIENCES
DEPARTMENT OF PHYSICAL SCIENCES
(COMPUTER SCIENCE)

COURSE TITLE
DATABASE DESIGN AND MANAGEMENT I

COURSE CODE
CSC 411
(2 CREDIT UNITS)

LECTURERS
DR. C. V. ENYINNAYA
DR. MRS. P. C. NWOSU

Course Contents

- Information storage and retrieval
- Information management applications
- Information capture and representation
- Analysis and indexing, search, retrieval
- Information privacy, integrity, security, scalability, efficiency and effectiveness

- Introductions to database systems
- Components of database systems
- DBMS functions
- Database architecture
- Data independence
- Use of database query language

Introduction

A database is an integrated collection of logically related data or records consolidated into a common pool for one or more users well organized for easy information retrieval and update. It focuses on storing, retrieving, and managing large volumes of structured information.

Databases come in all sizes, from simple collections of a few records to huge systems holding millions of records. The database technology eliminates many of the problems associated with the traditional file organization; it centralizes data, controls redundant data, and serves many applications and different groups at the same time, especially with huge amount of data.

A database consists of both data and metadata. Metadata is the data that describes the data structure within a database. If you know how your data is arranged, then you can retrieve it. Database is integrated because it includes not only data items but also the relationships among data items. The database stores metadata in an area called the data dictionary, which describes the tables, columns, indexes, constraints, and other items that make up the database.

Record keeping is an important aspect of every business and organization. Information management has become imperative in our information-based society and much of the world's computing and information management system is dedicated to maintaining and using databases. Databases of all kinds pervade almost every business or organization. All kinds of data, from emails and contact information to financial data and records of sales, are stored in some form of a database.

The primary goal of a database management system is to provide a way to store and retrieve database information that is both convenient and efficient and they are designed to manage large amount of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

Before database management systems (DBMSs) came along, organizations usually stored information using traditional file system. Keeping organizational information in a file-processing system has a number of major disadvantages; thus, DBMS is a major improvement over the traditional file system.

Databases come in all sizes, from simple collections of a few records to huge systems holding millions of records. A personal database is designed for use by a single person on a single computer. Such a database usually has a rather simple structure and a relatively small size. A departmental or workgroup database is used by the members of a single department or workgroup within an organization. This type of database is generally larger than a personal database and is necessarily more complex; such a database must handle multiple users trying to access the same data at the same time. An enterprise database can be huge. Enterprise databases may model the critical information flow of entire large organizations.

History of Database

Database management has undergone more than four decades of evolution producing vast range of research and extensive array of technology solutions. The database research community and software industry have responded to numerous challenges resulting from changes in user requirements and opportunities presented by hardware advances.

The origin of the database dated back to libraries, governmental, business and medical records before the computers were invented. Early data management was done by manual file system, such file system was traditionally composed of a collection of file folders each properly tagged and kept in filing cabinets. Once people realized they needed to have a means to store data and maintain the data files for later retrieval, they were trying to find ways to store, index, and retrieve data. With the emergence of computers, the world of the database changed rapidly, making it an easy, cost effective, and less space-consuming task to collect and maintain the database.

The introduction of the term database coincided with the availability of direct-access storage (disks and drums) from the mid-1960s onward. The term represented a contrast with the tape-based systems of the past, allowing shared interactive use rather than daily batch processing.

The first generation of database systems was navigational. Applications typically accessed data by following pointers from one record to another. Storage details depended on the type of data to be stored. Thus, adding an extra field to database required rewriting the underlying access/modification scheme. Emphasis was on records to be processed, not overall structure of the system. A user would need to know the physical structure of the database in order to query for information.

Until the 1970's, databases store large amounts of data in structures that were inflexible and difficult to manage and navigate. Programmers needed to know what clients wanted to do with the data before the database was designed; adding or changing the way data was stored or analyzed was expensive and time consuming.

The relational database model was conceived by E.F. Codd in 1970. This model was a departure from tradition by insisting that applications should search for data by content rather than by following links. His system can be defined using two terminologies. By the 1980's, three models for organizing the data were already developed - the network model, the hierarchic model, and the relational model. Once the relational model became prevalent, the history of databases took a sharp turn and started heading toward a better future for data management - data storage and data integrity.

Commercialization of relational systems begins as a boom in computer purchasing fuels the DB (database) market for business. Network and hierarchical models lost their charm with no

further development of these systems, but some legacy systems are still in use today. During this period, the concept of the object-oriented database was developed. An object database, also object-oriented database management system, is one in which information is represented in the form of objects as used in object-oriented programming. Some object-oriented databases are designed to work well with object-oriented programming languages, such as Delphi, Ruby, Python, Perl, Java, C#, and Visual Basic .NET, C++, Objective-C, and Smalltalk.

A major issue with object-oriented DBMS was switching an existing database to object-oriented DBMS, as the transition requires an entire change from scratch and it is typically tied to a specific programming language and an API (Application Programming Interface) which reduces the flexibility of the database. In the early 1990's, the Object Relational Database Model was developed in order to overcome the problems of object-oriented DBMS and fully explore the benefits of the relational model and object-oriented model.

Another major development which took place during 1997 was the introduction of the Extensible Markup Language (XML) . XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The XML applied to database processing which solved longstanding database problems. Major vendors begin to integrate XML into DBMS products.

By the year 2000, the Y2K fear was the main reason for the fast pace technology changes which took place in the late 1990's. Once the 2000's arrived, there was a steep decline in the Internet industry as a whole, but solid growth of database applications continued. More interactive applications appeared with the use of PDAs (Personal Data Assistants), Point-of-sales transactions, and consolidation of vendors. Three main companies dominated the larger market share of databases - IBM, Microsoft, and Oracle.

In 1998, Carlo Strozzi introduced a lightweight, open-source relational database that did not expose the standard SQL interface. NoSQL database systems rose alongside major internet companies, such as Google, Amazon, Twitter, and Facebook, which had significantly different challenges in dealing with data that the traditional RDBMS solutions could not cope. They are often highly optimized for retrieve and append operations and often offer little functionality beyond record storage.

Currently databases are beginning to take on even more complex logic. Another feature that is being expanded is the concept of separation of location from the abstract concept of the database itself; which Codd defined long ago. This feature enables a database to reside in more than one location and to be queried as a continuous unit. Such instances are called distributed or federated databases. Distributed transaction processing is becoming the norm for business planning in many areas.

Most recent database challenges include internet scale databases—databases that manage hundreds of millions of users and cloud databases that use novel technique for managing massive amounts of data.

Types of Database

There are two major types of database systems, namely:

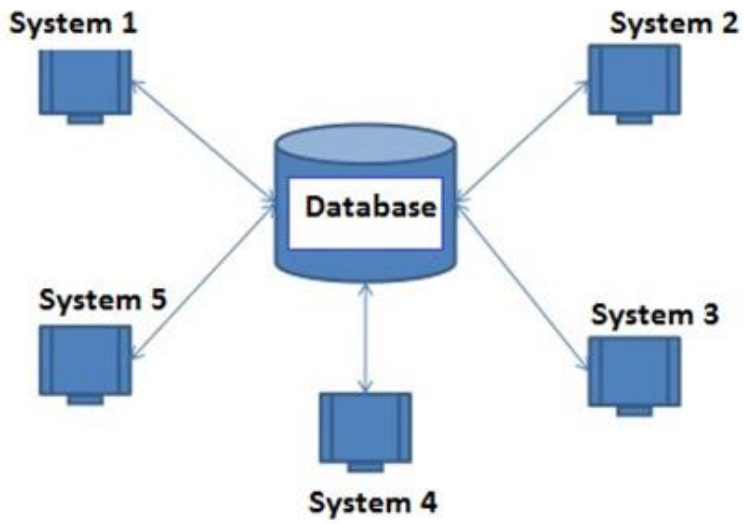
i. Centralized Database
ii. Distributed Database

i. Centralized Database

A centralized database system is a system that keeps the data in one single database at one single location. A single machine called a database server hosts the DBMS and the database. Multiple users or client workstations can work simultaneously on a centralized database system using the Client/Server configuration, or the Intranet configuration using Local Area Network (LAN) or Wide Area Network (WAN).

A Centralized database usually stores data or information in a particular location within a network. It allows data from existing database to be collected and stored in a single database for sharing, analysis, or updating in an organization. In centralized databases all data are managed by a single DBMS and placed on a single node, only users being distributed in the network. For centralized databases, major benefits are determined by a good data integration that ensures data consistency and easy management of transactions in strict compliance with the ACID properties.

The main disadvantage of centralized database systems is that of single point of failure. When the database fails, all the users are interrupted. Also, in the case where WAN is used, failure of part of the network means the interruption of work at the remote location. Centralized databases are easier to manage, maintain and control for security purposes; therefore, it should be the option to be selected if there is no need for a more complex architecture.



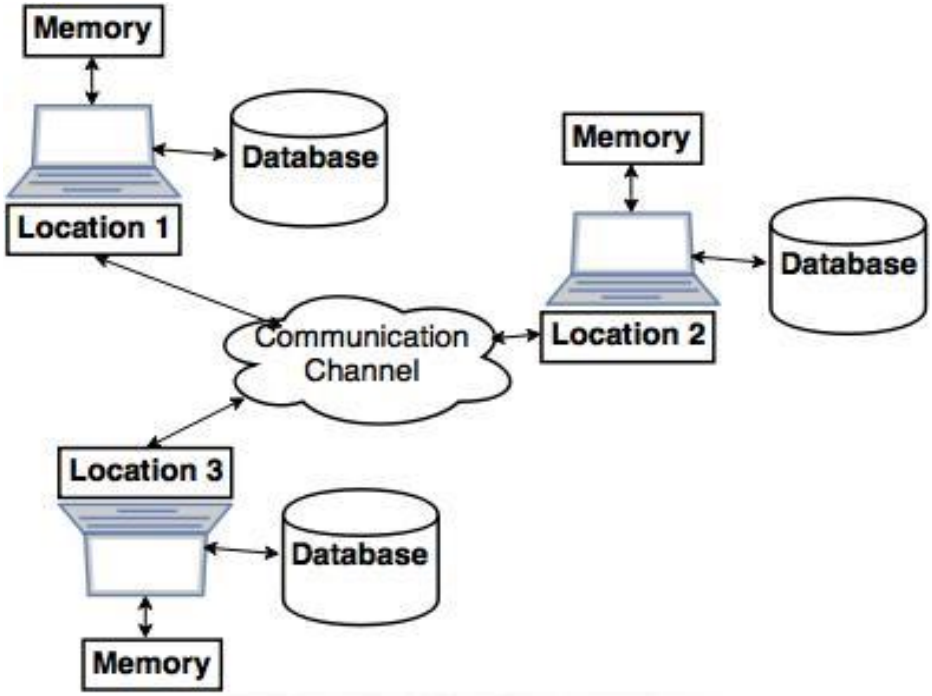
A Centralized Database System

ii. Distributed Database

A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. Here, the actual database and the DBMS software are distributed at various sites that are connected by a computer network and allow access to the database from different computers.

In distributed database environment, new problems that are not applicable in centralized environments could be faces, such as fragmentation and data replication. A data fragment constitutes some subset of the original database. A data replica constitutes some copy of the

whole or part of the original database. Distributed database is also referred to as decentralized database.



A Distributed Database System

DATABASE MANAGEMENT SYSTEM (DBMS)

DBMS is a software system that interacts with the user, applications, and the database itself to capture and analyze data. Its primary function is to provide a way to store, retrieve, update, and manage data. It provides users with a systematic way to create, retrieve, update and manage data. This is done by ensuring that the data is consistent, secure, and easily accessible to users.

It provides a systematic and organized way of storing large amounts of data. This data can be stored in various formats such as tables, reports, and spreadsheets. The DBMS also provides a way to manage these data formats. This includes defining the data types, structures, and constraints for the stored data.

Database Management System (DBMS) is a collection of programs that enables users to create, manage and maintain databases and as well control all access to them. The database approach to data management utilizes the database management system and the primary goal of a DBMS is to provide an environment that is both convenient and efficient for users to store and retrieve information. It allows organizations to place control of organization-wide database development in the hands of Database Administrators (DBAs) and other specialists. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way.

Database management systems are usually categorized according to the database model that they support, such as the network, relational or object model. The model tends to determine the query languages that are available to access the database. One commonly used query language for the relational database is SQL, although SQL syntax and function can vary from

one DBMS to another. A great deal of the internal engineering of a DBMS is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products.

Basically, a DBMS acts as an intermediary between the user and the database, ensuring that the data is easily accessible, consistently organized, and securely maintained.

There are several popular Database Management Systems (DBMS), which include:

- **Microsoft SQL Server:** This is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

- **Oracle Database:** This is commonly referred to as Oracle RDBMS or simply as Oracle. It is a multi-model database management system produced and marketed by Oracle Corporation. It is a database commonly used for running online transaction processing (OLTP), data warehousing (DW) and mixed (OLTP&DW) database workloads.

- **dBase:** This was one of the first database management systems for microcomputers, and the most successful in its day. The dBase system includes the core database engine, a query system, a forms engine, and a programming language that ties all of these components together. dBase was originally published by Ashton-Tate for microcomputer operating system CP/M in 1980, and later ported to Apple II and presently owned by dBase LLC.

- **Microsoft Access:** Microsoft Access is a database management system (DBMS) from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately.

  Microsoft Access is a file-based database. Unlike client-server relational database management systems (RDBMS), e.g., Microsoft SQL Server, Microsoft Access does not implement database triggers, stored procedures, or transaction logging. All database tables, queries, forms, reports, macros, and modules are stored in the application's database engine as a single file. This makes Microsoft Access useful in small applications, teaching, etc. because it is easy to move from one computer to another.

- **MySQL:** This is an open-source relational database management system (RDBMS) originally designed by MySQL AB, a Swedish software company which was later acquired by Sun Microsystems in 2008. MySQL is currently owned by Oracle Corporation after it acquired Sun Microsystems in 2010. MySQL is a central component of the LAMP open-source web application software stack. LAMP is an acronym for Linux, Apache, MySQL, Perl/PHP/Python.

- **DB2:** This is a database-server developed by IBM that supports the relational database model, but in recent years, some products have been extended to support

object-relational features and non-relational structures. DB2 traces its roots back to the beginning of the 1970s when Edgar F. Codd, a researcher working for IBM, described the theory of relational databases.

- **PostgreSQL**: This is also known as Postgres, it is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It is supported on all major operating systems, including Windows, Linux, macOS, etc and handles a range of workloads from single machines to data warehouses, data lakes, or web services with many concurrent users. It evolved from the Ingres project at the University of California (PostgreSQL Global Development Group). In 1982, the leader of the Ingres team, Michael Stonebraker, left Berkeley to make a proprietary version of Ingres. He returned to Berkeley in 1985, and began a post-Ingres project to address the problems with contemporary database systems that had become increasingly clear during the early 1980s. It went through several stages of evolution and in 1996; the project was renamed to PostgreSQL to reflect its support for SQL.

DBMS FUNCTIONS

Database management systems perform several important functions that guarantee the integrity and consistency of the data in the database. The most important functions of Database Management System include:

1. Data Storage: One of the main functions of a DBMS is data storage. It provides a systematic and organized way of storing large amounts of data. This data can be stored in various formats such as tables, reports, and spreadsheets. The DBMS also provides a way to manage these data formats. This includes defining the data types, structures, and constraints for the stored data. it helps in creating and managing the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics.

2. Security Management: Security Management is another important function of Database Management System (DBMS). The DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database, which data items each user can access, and which data operations (read, add, delete, or modify) the user can perform. This is especially important in multi-user database systems.

3. Multi-user Access Control: Multi-user access control is another important DBMS Function. To provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of the database.

4. Backup and Recovery Management: The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures. Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database's integrity.

5. Data Integrity Management: Data integrity management is another important DBMS function. The DBMS promotes and enforces integrity rules, thus minimizing data

redundancy and maximizing data consistency. The data relationships stored in the data dictionary are used to enforce data integrity. Ensuring data integrity is important DBMS functionality in transaction-oriented database systems.

6. Database Access Languages and Application Programming Interfaces: The DBMS provides data access through a query language. A query language is a non-procedural language—one that lets the user specify what must be done without having to specify how it is to be done.

DATABASE ARCHITECTURE

Database architecture is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. It allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

This architecture determines how data is stored, organized, and retrieved, playing a crucial role in the efficiency and effectiveness of data management. It also describes how a database management system is integrated with applications. The architecture of a database is not a one-size-fits-all solution. It varies significantly based on the needs of the organization, the type of data being managed, and the specific applications that interact with the database. From simple structures that manage daily transactions in a small business to complex architectures that handle massive amounts of data in large enterprises, the spectrum of database architecture is broad and diverse.

Types of DBMS Architecture

There are mainly three types of DBMS architecture:

1. One-Tier Architecture
2. Two-Tier Architecture
3. Three-Tier Architecture

1. One-Tier Architecture in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.

2. Two-Tier Architecture: Two-tier architecture in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two-tier architecture consists of multiple clients connecting directly to a database, it is also known as client-server architecture. It provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.

3. Three-Tier Architecture: This is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface are done independently as separate modules. Three-tier architecture contains a presentation layer, an application layer, and a database server. It is an extension of the 2-tier client-server architecture, it consists of the following;

    - Presentation layer— PC, tablet, Mobile devices, etc.
    - Application layer (server)
    - Database Server

DATABASE MODELS

A database model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. To use a common analogy, the data model is equivalent to an architect's building plans.

The goal of the database model is to make sure that all data objects required by the database are completely and accurately represented. It simply represents a plan for building a database. To be effective, it must be simple enough to communicate to the end user the data structure required by the database yet detailed enough for the database design to use to create the physical structure. The main function of data models is to help in understanding the complexity of the real-world environment. It facilitates interaction among the designer, application programs and the end user.

Types of Database Models

1. Relational Model

This is the most popular and the most extensively used model for data storage and processing. The data is stored in tables called a relation having rows and columns, where rows represent records and columns represent the fields. Each row in a relation represents a single entity and it is known as tuple, each column represents values from same domain, these values are characteristics that belong to the entities modeled by the table and are known as attributes.

S/N	First_name	Last_name	RegNumber	Gender
1	Chinazo	Nwosu	G2017/PhD/COMP/FT/007	Female
2	Helen	Onungwe	G2017/PhD/COMP/FT/005	Female
3	Emma	Oviebor	G2017/PhD/COMP/FT/012	Male
4	Greg	Johnson	G2017/PhD/COMP/FT/010	Male
5	Dan	Thompson	G2017/PhD/COMP/FT/002	Male

Representation of a Relational Model

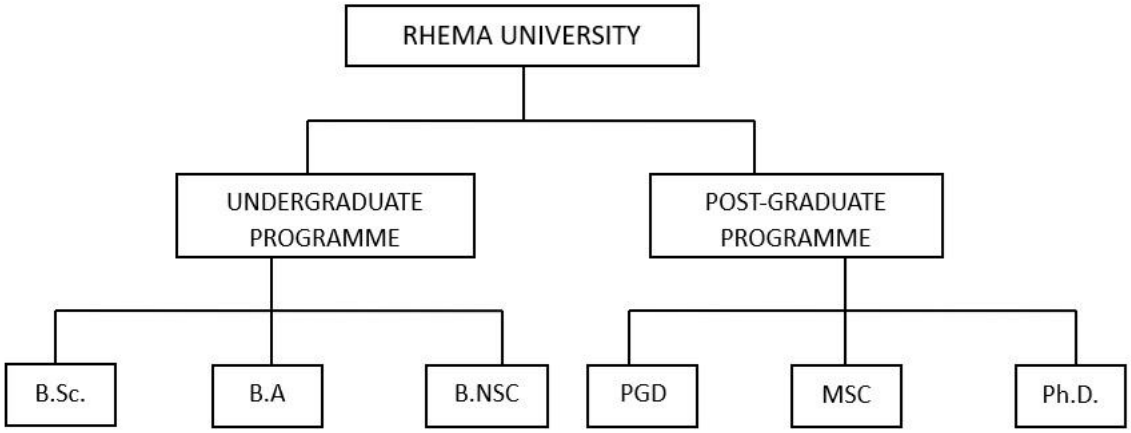
Relational model was introduced by an IBM staff E. F. Codd in 1970; it was a major breakthrough for both users and software designers. The most important advantage of the relational model is its ability to hide the complexities of the model from the user. It has dominance because of its powerful and flexible query language. Relational models use the Structured Query Language (SQL), which is a fourth generation programming language that allows users to specify what must be done without how it must be done. The SQL is used to translate user's query into instructions for retrieving data from a database with less effort.

Properties of a Relation

i. No Duplicate Tuples: This implies that a relation cannot contain two or more rows which have the same values for all the attributes. i.e. every row is unique.
ii. Tuples are unordered: The order of rows in a relation is unimportant.
iii. Attributes are unordered: The order of columns in a relation is immaterial.
iv. Attribute Values are Atomic: Each tuple contains exactly one value for each attribute.

2. Hierarchical Model

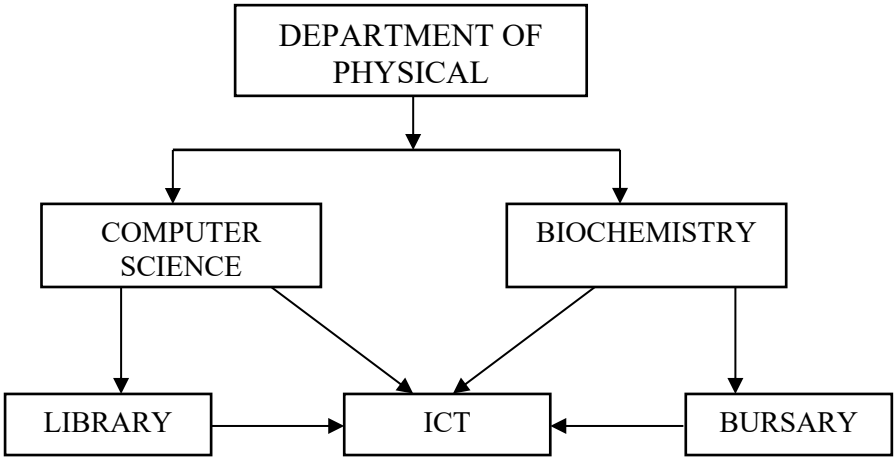
This model was developed in the 1960's to manage large amount of data for complex manufacturing projects. The hierarchical structure contains levels or segments. A segment is equivalent of a file system record type. Within the hierarchy, the top level is perceived as the parent of the segment directly under it. This model yields a lot of advantages over the file system and it formed the bases for the development of subsequent models.This type of model has one parent entity with several children entity but at the top there is only one entity called root. For example, an organization has different department with several sub-departments. Example is shown in Figure 1.



Representation of a Hierarchical Model

3. Network Model

This model was developed in the 1970's to represent complex data relationships more effectively than the hierarchical model in order to improve database performance. The network model looks like the hierarchical model; however, a network model allows a record to have more than one parent. In this model a relationship is called a set with each set composing of at least two record types. It has flexible way of representing objects and their relationships. Its distinguishing feature is that the schema, viewed as a graph in which object types are nodes and relationship types are arcs, is not restricted to being a hierarchy or lattice. It replaces the hierarchical model with a graph thus allowing more general connections among the nodes. This type of model has the entities which are organized in a graphical representation and some entities in the graph can be accessed through several paths.



Representation of a Network Model

4. Entity-Relationship Model (E-R Model)

The rapidly increasing transaction and information requirement created the need for more complex database implementation structure, thus creating more effective database design tools. Though the relational model was a major improvement over the hierarchical and network model, it lacked the features that would make it an effective tool in database design tool.

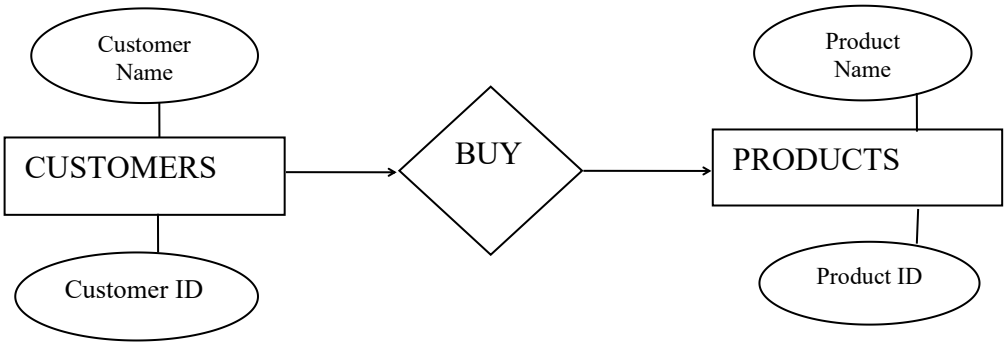
The Entity-Relationship (ER) model was originally proposed by Peter Chen in 1976 as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. The overall logical structure (schema) of a database can be expressed graphically by an E-R diagram.

Component of E-R Model:

- **Entities:** These are real-world objects or concepts within the system, such as customers, products, or orders. An entity is represented by a rectangle in an E-R diagram with its name usually a noun written in the center of the rectangle.
- **Attributes:** Describe the properties or characteristics of entities, such as customer name, product price, or order date.
- **Relationship:** This describes the association among the different entities. It is represented by a diamond with the name of the relationship usually a passive or active verb written in the centre of the diamond.
- **Cardinality:** Specifies the maximum and minimum number of occurrences of one entity that may be related to another entity. Is is represented by oval shape.

Examples 1:

Examples 2:



Object-Oriented Model

Complex real-world problems demonstrated the need for a data model that would more closely represent the real world data. In object-oriented model, both the data and the relationships are represented in a single structure known as the object. The object is seen as a box and the attributed of the object and the relationship to the other objects are within the same object box. It reflects a very different way to define and use the entities.

In early 90s a number of database management systems were as pure object-oriented database systems with the goal to address the limitations of relational databases by adopting a completely new database model with support for objects with unique identifiers, methods, inheritance, encapsulation, polymorphism and other features commonly associated with object-oriented programming languages. The basic idea was to build on top of object-oriented programming languages and provide persistence for application objects achieving homogeneous programming environment with close correspondence between application objects and objects stored in the database. Modern database applications are characterized by four categories of requirements:

i. The need to store and manage large multimedia data objects – images, sound clips, videos, maps, etc.
ii. The requirement for database data types to mirror application-level data types, including the ability for users to define their own data types as needed by specific applications.
iii. The representation of complex relationships, including composition and aggregation, e.g. multi-level component assemblies used in Computer-Aided Design (CAD) and similar applications.
iv. The need for seamless integration with object-oriented programming languages.

In addition to the need to store large and complex objects in the database, there is another important requirement that motivated the introduction of object support at the database level. Most modern applications are developed using object-oriented programming languages (i.e. Java, Python, C++, C#) and close integration of the database language SQL with object-oriented programming languages reduces impedance mismatch (i.e. differences between the type systems, error handling, etc.) with corresponding improvements in programmer productivity.

Many regarded object-oriented database as the next generation of database technology that will certainly supersede relational databases; however, this has as been largely unsuccessful as object-oriented database has not been able to match relational database technology in a number of important aspects, including reliability, scalability and level of standardization. Even more importantly, while popular in some niche application areas (e.g. CAD/CAM), object-oriented databases have not been able to address the wider requirements of mainstream corporate applications.

5. Object-Relational Model

This model was developed as a result of the increasing complexity of applications. It is also referred to as the Extended Relational Database Model (ERDM). It is a hybrid model that combines the functionality of the relational model and the object-oriented model. Its primary goal is geared towards business applications while the object-oriented model focuses on very specialized engineering and scientific applications. The most likely scenario is the integration of relational and object-oriented model concepts and procedures with emphasis of database model that facilitates the internet-age technology.

A number of influential database researches formed a committee for Advanced DBMS function with the objective to define the requirements for the next generation database systems, and published the Third-Generation Database Systems Manifesto as a blueprint for future database development to preserve the benefits of relational database and at the same time to take advantage of object features. The evolutionary approach was what gave birth to a new hybrid model, the object-relational database technology. However, bringing object features into SQL did not turn out to be an easy task, and the approach has struck numerous challenges and produced a number of changes in direction.

DATA INDEPENDENCE

Data Independence is a property of database management system that helps to change the database schema at one level of a database system without requiring changing the schema at the next higher level. It helps to keep data separated from all programs that make use of it. Basically, it allows a user or database administrator to change the schema at one level without affecting the data or schema at another level.The purpose of data independence is to enhance the security of the system, save time and reduce costs needed once the information is changed or altered.

A database system normally contains a lot of data; it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. As a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job. So, data independence is the separation of data from the programs that use the data. Most modern applications are based on the principle of data independence. The whole concept of a database management system (DBMS) supports the notion of data independence since it represents a system for managing data separately from the programs that use the data.

Achieving data independence involves data abstraction. Data Abstraction allows extracting the necessary data by ignoring the remaining irrelevant details. The main purpose of data abstraction is to achieve data independence. There are three levels of abstraction.

1. Physical or Internal Level - Physical level is the lowest level of data abstraction and it indicates how the data will be stored and describes the complex data structures and access methods to be used by the database. The internal level is used to describe the entire database architecture.

2. Conceptual or Logical Level - The Conceptual database schema is additionally called the logical structure because it defines the logical relations between the data. The separation of the conceptual view from the internal view enables us to provide a logical description of the database concepts without the need to specify physical structures. The conceptual level comes between the physical level and the view level. It provides the link between the external schema and the internal schema of the database.

3. External or View Level - It is the highest level of data abstraction. The external level describes the user interaction with the centralized database management system. This level is used to provide a Graphical User Interface to the user, and the user does not know about the file structure, access method, and other internal details of the database.

Types of Data Independence

There are two levels of data independence in DBMS:

1. Physical Level Data Independence

Physical Data Independence can be defined as the ability to change the physical level without affecting the logical or Conceptual level. Physical data independence gives us the freedom to modify the - Storage device, File structure, location of the database, etc. without changing the definition of conceptual or view level. Example: For example, if we take the database of the banking system and we want to scale up the database by changing the storage size and also want to change the file structure, we can do it without affecting any functionality of logical schema. These changes can be done at the physical layer without affecting the conceptual layer:

- Changing the storage devices like SSD, HDD and magnetic tapes, etc.
- Changing the access technique and modifying indexes.
- Changing an algorithm.

2. Logical Level Data Independence

Logical Data Independence is a property of a database that can be used to change the logic behind the logical level without affecting the other layers of the database. Logical data independence is usually required for changing the conceptual schema without having to change the external schema or application programs. It allows us to make changes in a conceptual structure like adding, modifying, or deleting an attribute in the database. Example: If there is a database of a banking system and we want to add the details of a new customer or we want to update or delete the data of a customer at the logical level data will be changed but it will not affect the Physical level or structure of the database.

These changes can be done at a logical level without affecting the application program or external layer.

- Adding, deleting, or modifying the entity or relationship.
- Merging or splitting the record.

DATABASE QUERY LANGUAGE

Database Query Language (DQL)is a computer programming language used to make queries and retrieve information from databases. It allows users to communicate with the database management system (DBMS) in order to perform operations such as inserting, updating, deleting, and retrieving data.

Query

A query is an SQL statement used for information retrieval. It is the most common operation in SQL which is performed with the declarative SELECT statement. Query allows the user to describe desired data, leaving the database management system (DBMS) responsible for planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk (*) can also be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses that specify what exactly should be returned in the result.

Structured Query Language (SQL)

Structured Query Language (SQL) is the most popular and commonly query language designed for managing data in a relational database management system (RDBMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables. It was developed by IBM and has been endorsed by the ANSI SQL-92 standard. SQL is a major component of DBMS and it comprises of other sub-languages:

- **Data Definition Language (DDL):** This is used to specify the database schema, which is an overall design of the database. A database system consults the data dictionary before reading or modifying actual data. A data dictionary contains metadata—that is, data about data. The SQL commands used for this purpose include CREATE, ALTER and DROP.

- **Data Manipulation Language (DML):** This is a language that enables users to access or manipulate data as organized by the appropriate data model. It is used to add, delete and update data in a database. The DML component of the structured query language is non-procedural. The portion of a DML that involves information retrieval is called a query language. The SQL commands used for this purpose include INSERT, UPDATE and DELETE.

- **Data Control Language (DCL)**: This is a syntax similar to a computer programming language used to control access to data stored in a database. It is one of the logical group in SQL commands, DCL commands are used for access control and permission management for users in the database, it easily allow or deny some actions for users on the tables or records, examples include GRANT, REVOKE and DENY.

- **Data query language (DQL):** DQL statements are used for performing queries on the data within schema objects. The purpose of DQL commands is to get the schema relation based on the query passed to it, example SELECT.

Properties of Database Management Systems

A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations. In order to maintain consistency in a database, before and after transaction, certain properties are followed. These are known as the ACID properties.

These properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

a) **Atomicity**: This mean that either the entire transaction takes place at once or does not happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves two operations:

- **Abort**: If a transaction aborts, changes made to database are not visible.
- **Commit**: If a transaction commits, changes made are visible.

b) **Consistency**: This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to correctness of a database. The total amount before and after the transaction must be maintained.

c) **Isolation**: This property ensures that multiple transactions can occur concurrently without leading to inconsistency of database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved if these were executed serially.

d) **Durability**: This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored and remain consistent even if system failure occurs. These updates now become permanent and are stored in a non-volatile memory. The results of the transaction; thus, are never lost.

Advantages of Database

i. It improves data sharing among users.
ii. It reduces data redundancy.
iii. It reduces updating errors and increases consistency.
iv. It provides better data integrity and independence from applications programs.
v. It improves data security. DBMS provides a framework for better enforcement of data privacy and security policies.

Disadvantages of Database

i. Database systems are complex, difficult, and time-consuming to design.
ii. It involves substantial hardware and software start-up costs.
iii. Damage to database affects virtually all applications programs.
iv. Technical knowledge is required for all database users.