B.Com II Year

Syllabus

Subject: Relational Database management System

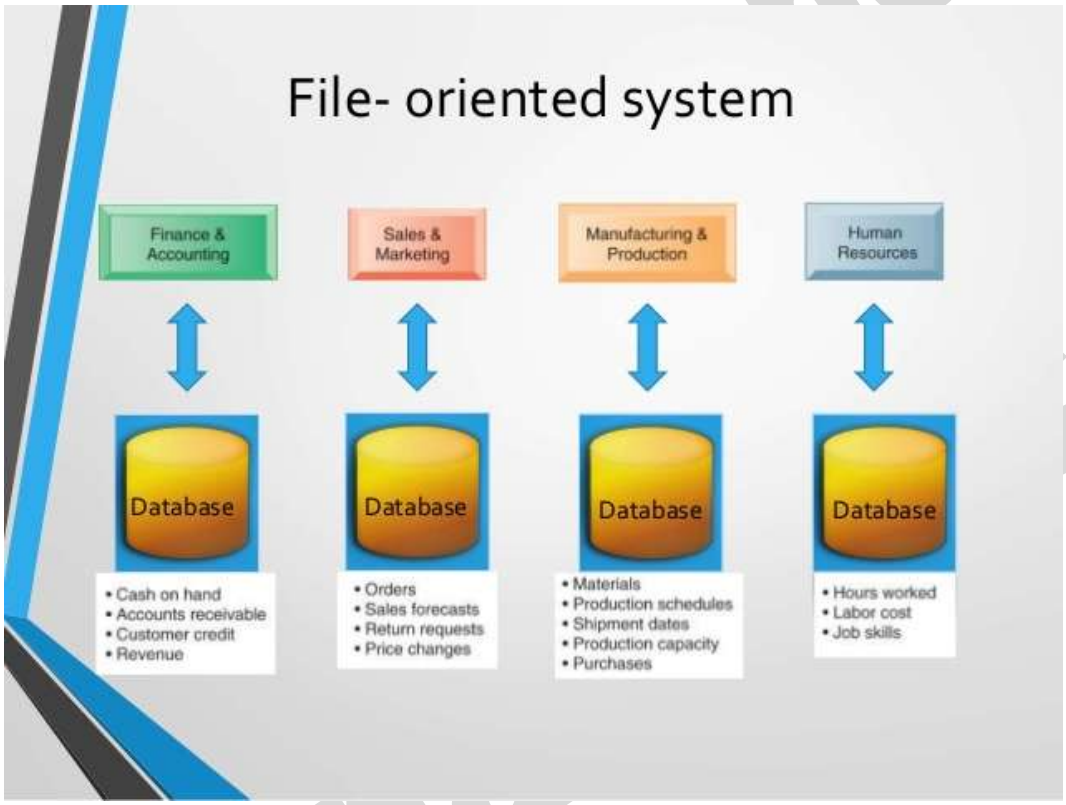
Unit – I	Evolution of Database Technology, File-Oriented system, Database system, Client Server Platform:, Database system in the organization: Database and Data sharing, Startegic database planning, Management control, risks and cost of database: logical and physical data representation.
Unit – II	Database Development Life Cycle (DDLC). Principles of conceptual Database Design, Objects, Specialization Generalization, Relationship, Cardinality, Attributes, Relational data Model: Fundamental Concepts, Normalization process (1NF, 2NF, 3NF, BCNF, 4NF) Transforming Conceptual Model to a Relational Model.
Unit – III	Relational Algebra, Relational implementation with SQL, introduction, Data Definition Language (DDL), Data Manipulation Language (DML), Data control Language (DCL), Transaction control Language (TCL), Schema and Table Definition, SQL functions: Mathematical functions, Group functions, view definition: Introduction command to create VIEW.
Unit – IV	Physical, storage media, Disk performance factors data storage format file organization and addressing method implementing, Managing the data base environment- Database administration and control, DBA functions, goals, integrity, security and recovery.
Unit – V	Introduction to SQL- Components of SQL, DDL,DML, Query Language DCI, TCI, SCL etc invoking sql plus The oracle data types two dimensional matrix creation, Insertion updation, deletion operations, the many faces of SELECT component creating tables using query, inserting data using query, modifying the structure of tables, renaming tables, dropping tables, dropping columns, logical operators, range searching, pattern matching, use of Alias, Oracle Function. Accessing data from multiple tables. Set operations. Union, Intersect, Minus. Data constraints: 1/0 constricting Business Rule constraints. Groping Data save point, ROLLBACK & COMMIT constructing creating user accounts, granting permission, revoking permission

Unit 1

Evolution of Database technology:-

File System

A File Management system is a DBMS that allows acces to single files or tables at a time. In a File System, data is directly stored in set of files. It contains flat files that have no relation to other files (when only one table is stored in single file, then this file is known as flat file).



Disadvantages of File processing system

1) Data redundancy

In computer system many files are likely in the different formats and the programs are written in different programming languages. Moreover, the same information may be duplicated in several files, this duplication of data is known as data redundancy.

Example: The address and telephone number of a particular customer may appear in a file that consist of saving account records and in a file that consist of checking account record.

2) Data inconsistency

Various copies contain the same type of data which may no longer which means that various copies of same data may contain different kind of information.

Example: A changed customer address may be reflected in savings account records but not elsewhere in the system.

3) Difficulty in accessing data

In file processing system it is very difficult to access the data in a specific way and it also require a special application program which carry out new task.

4) Data isolation

Because data are scattered in various files and files may be in different formats, writing new applications program to retrieve the appropriate data is difficult.

5) Integrity problem

Database must satisfy a particular consistency constraint. These constraints are added in application program.

Example: The balance of a bank account may never fall below a prescribed amount.

6) Atomicity problem

A computer system, like any other mechanical or electrical devices, is subject to failure. In many applications, it is crucial that if failure occurs, the data be restored to the consistent state that existed prior to the failure.

7) Concurrent access anomalies

If two programs run concurrently it is important to has supervision. But supervision is difficult to provide because data is decentralized in file processing system. In such an environment, interaction updates may result in inconsistent data.

8) Security problems

In this not every user of the database system should be able to access all the data.

Advantage of File-oriented system:

1. Backup:

- It is possible to take faster and automatic back-up of database stored in files of computer-based systems.

- computer systems provide functionalities to serve this purpose.it is also possible to develop specific application program for this purpose.

2. Compactness:

- It is possible to store data compactly.

3. Data Retrieval:

- Computer-based systems provide enhanced data retrieval techniques to retrieve data stored in files in easy and efficient way.

4. Editing:

- It is easy to edit any information stored in computers in form of files.
- Specific application programs or editing software can be used for this purpose.

5. Remote Access:

- In computer-based systems, it is possible to access data remotely.
- So ,to access data it is not necessary for a user to remain present at location where these data are kept.

6. Sharing:

- Data stored in files of computer-based systems ca be shared among multiple users at a same time.

DBMS

A Database Management System (DBMS) is a system software that allows users to efficiently define, create, maintain and share databases. Defining a database involves specifying the data types, structures and constraints of the data to be stored in the database. Creating a database involves storing the data on some storage medium that is controlled by DBMS. Maintaining a database involves updating the database whenever required to evolve and reflect changes in the miniworld and also generating reports for each change. Sharing a database involves allowing multiple users to access the database. DBMS also serves as an interface between the database and end users or application programs. It provides control access to the data and ensures that data is consistent and correct by defining rules on them.
An application program accesses the database by sending queries or requests for data to the DBMS. A query causes some data to be retrieved from database.

Advantages of DBMS over File system –

- **Data redundancy and inconsistency –** Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control redundancy of data as

each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining same files data for different applications. Hence changes made by one user does not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.

- **Data sharing –** File system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to centralized system.
- **Data concurrency –** Concurrent access to data means more than one user is accessing the same data at the same time. Anomalies occur when changes made by one user gets lost because of changes made by other user. File system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.
- **Data searching –** For every search operation performed on file system, a different application program has to be written. While DBMS provides inbuilt searching operations. User only have to write a small query to retrieve data from database.
- **Data integrity –** There may be cases when some constraints need to be applied on the data before inserting it in database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user defined constraints on data by itself.

From pre-stage flat-file system, to relational and object-relational systems, database technology has gone through several generations and its history that is spread over more than 40 years now.

Difference between File Management System v/s Database Management System

File Management System	Database Management System
File System is a general, easy-to-use system to store general files which require less security and constraints.	Database management system is used when security constraints are high.
Data Redundancy is more in file management system.	Data Redundancy is less in database management system.
Data Inconsistency is more in file system.	Data Inconsistency is less in database management system.

Centralisation is hard to get when it comes to File Management System.	Centralisation is achieved in Database Management System.
User locates the physical address of the files to access data in File Management System.	In Database Management System, user is unaware of physical address where data is stored.
Security is low in File Management System.	Security is high in Database Management System.
File Management System stores unstructured data as isolated data files/entities.	Database Management System stores structured data which have well defined constraints and interrelation.

Database Applications:

★ Banking: all transactions

★ Airlines: reservations, schedules

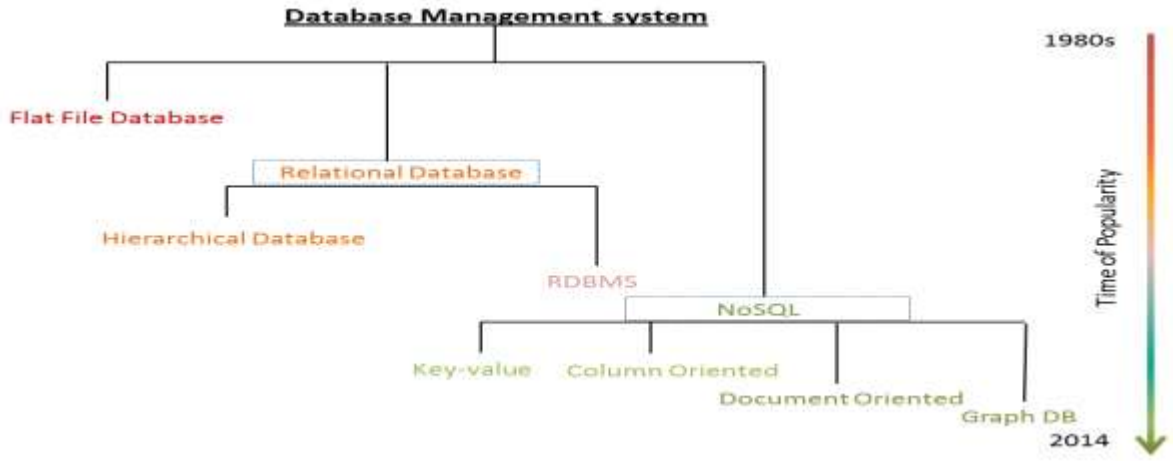
★ Universities: registration, grades

★ Sales: customers, products, purchases

★ Manufacturing: production, inventory, orders, supply chain

★ Human resources: employee records, salaries, tax deductions

The Evolution :



1968 File-Based: predecessor of database**,** Data was maintained in a flat file.

Flat Files:

Earlier, punched cards technology was used to store data – later, files. But the files have no as such advantage, rather have several limitations.

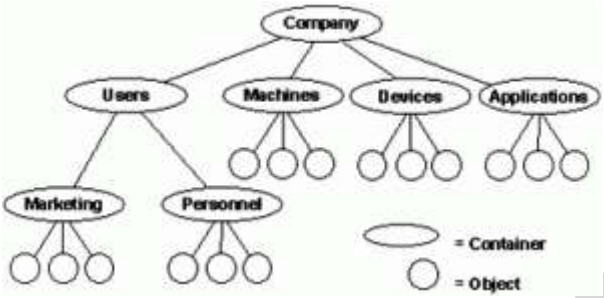
Advantages	***Limitations***
Various access methods , e.g., sequential, indexed, random	Requires extensive programming in third-generation language such as COBOL, BASIC.
	Separation and isolation: Each program maintains its own set of data, users of one program may not be aware of holding or blocking by other programs that are being used somewhere else, by another user.
	Duplication of data – same data is held by different programs, thus, wastes space and resources.
	High maintenance costs such as ensuing data consistency and controlling access
	Sharing granularity is very coarse.
	Weak security.

[1968-1980] Era of Hierarchical Database: Prominent hierarchical database model was IBM's first DBMS called **IMS** (**Information Management System**).

Hierarchical Data Model:

Mid 1960s Rockwell collaborates with IBM to create the Information Management System (IMS), IMS lead the mainframe database market in 70's and early 80's.

In this model, files are related in a parent/child manner, with each child file having at most one parent file.



Advantages	*Limitations*
Efficient searching.	Complex implementation
Less redundant data.	Difficult to manage and lack of standards, can't easily handle many-many relationships.
Data independence.	Lacks structural independence.
Database security and integrity.	

Network Data Model:

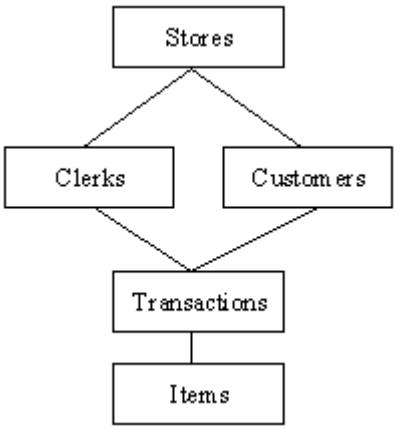
Early 1960s, Charles Bachmann developed first DBMS at Honeywell, **Integrated Data Store** (**IDS**)

It standardized in 1971 by the **CODASYL** group (**Conference on Data Systems Languages**).

In Network data model, files are related as owners and members, similar to the common network model except that each member file can have more than one owner.

Network data model identified the following three database components:

1. Network schema—database organization[structure]
2. Sub-schema—view s of database per user
3. Data management language — at low level , procedural



Advantages	***Limitations***
Ability to handle more relationship types	System complexity and difficult to design and maintain
Ease of data access	Lack of structural independence as data access method is navigational.
Data Integrity	
Data Independence	

Prominent network database model was **CODASYL DBTG** model where as **IDMS** was the most popular network DBMS.

Here, I am clearly mentioning one thing that the Hierarchical Model and the Network Model were in use in almost the same era.

[1970-present] Era of Relational Database and Database Management: The relational database model was conceived by E. F. Codd in 1970. It can be defined using the following two terminologies:

1. Instance – a table with rows or columns.
2. Schema – specifies the structure (name of relation, name and type of each column)

The model is based on branches of mathematics called set theory and predicate logic.

Relational DBMS at a glance:

Fundamental Relational Database Characteristics	***Database Schema(The description of the user data in the database)***	***DBMS Functions***	***Database Approach***	***Advantages and disadvantages of DBMSs***
The internal structure of an operating database is basically fixed in the 'row' direction The user will interact with a logical view of the data, and need not know anything about the actual internal structure.	§ **Conceptual schema**: logically describes all data in the database § **Internal schema** (Physical schema): describes how data are actually stored. § External **schema** (User view): describes the data that is of use to the user.	§ Data dictionary management § Data storage management § Data transformation and presentation § Security management § Multi-user access control § Backup and recovery management § Data integrity management § Database language and application programming interfaces § Database communication interfaces	§ Data definition language (**DDL**): define database structure [e.g. tables] § Data manipulation language (**DML**): to retrieve, insert, delete and update data in the database. Query language are part of DML § Data control language (**DCL**): control the access of data.	**Advantages:** § Control of data redundancy, consistency, abstraction, sharing § Improved data integrity, security, enforcement of standards and economy of scale. § Balanced conflicting requirements § Improved data accessibility, responsiveness, maintenance § Increase productivity, concurrency, backup and recovery services. **Disadvantages:** § Complexity, size, cost of DBMSs § Higher impact of a failure

General Comparison:

Here is a glimpse of all those database models we have discussed till now.

Concept	*Relational*	*Network*	*Hierarchic*
Item	role name/domain	data item type	item/field
Item value	component	data item occurrence	value
Group	not allowed	group	group
Entity (type)	relation	record type	entry/segment type
Entity instance	tuple	record occurrence	entry/segment occurrence
Relationship	foreign key comparable underlying domains	set type	hierarchic (implied)
Relationship instance		set occurrence	assembly
Data administrator view	data model	logical structure	logical structure
Definition of data administrator view	data model definition	schema	schema
User view	data submodel		
Definition of user view	data submodel definition	subschema	subschema
Data-base subdivision		area	
Entry points	primary key	singular sets CALC records	root group root segment
Single unique/identifier	candidate key	key	sequencer (unique)

Object Oriented Database Model:

It supports the modeling and creation of the data as objects.

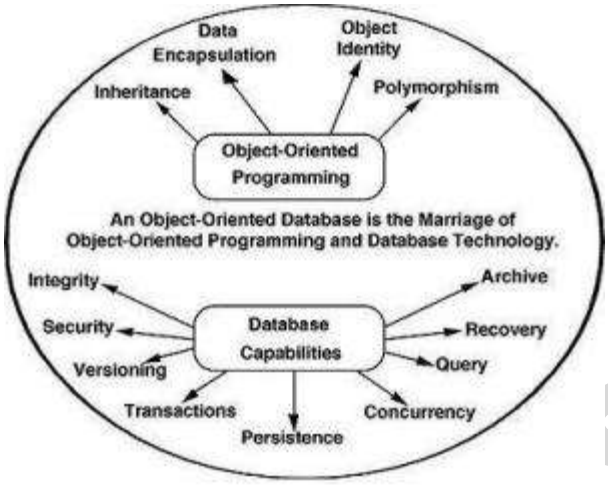


Figure 1. Makeup of an Object-Oriented Database

Advantages	***Limitations***
Can efficiently manage a large number of different data types.	Switching an existing database to OODBMS requires an entire change from scratch.
Objects with complex behaviors are easy to handle using inheritance and polymorphism etc.	An OODBMS is typically tied to a specific programming language and an API; this reduces its flexibility.
Reduces the large number of relations by creating objects.	Ad-hoc queries are difficult to implement as one cannot join two classes as one can join two tables in RDBMS. Therefore, queries depend upon the design of the system.
	Creates problems when deleting data in bulk.

Object Relational Database Model:

Object relational databases span the object and relational concepts.

Advantages	***Limitations***
Large storage capacity	The architecture of the object relational model is not appropriate for web applications.
High access speed	

Let's take a fleeting look at the history.

1970: Ted Codd at IBM's San Jose Lab proposed relational models.

Two major projects start and both were operational in late 1970s

INGRES at University of California, Berkeley became commercial and followed up POSTGRES which was incorporated into Informix.

System R at IBM san Jose Lab, later evolved into DB2, which became one of the first DBMS product based on the relational model. (Oracle produced a similar product just prior to DB2.)

1976: Peter Chen defined the Entity-relationship(ER) model

1980s: Maturation of the relational database technology, more relational based DBMS were developed and SQL standard adopted by ISO and ANSI.

1985: Object-oriented DBMS (OODBMS) develops.

1990s: Incorporation of object-orientation in relational DBMSs, new application areas, such as data warehousing and OLAP, web and Internet, Interest in text and multimedia, enterprise resource planning (ERP) and management resource planning (MRP)

1991: Microsoft ships access, a personal DBMS created as element of Windows gradually supplanted all other personal DBMS products.

1995: First Internet database applications

1997: XML applied to database processing, which solves long-standing database problems. Major vendors begin to integrate XML into DBMS products.

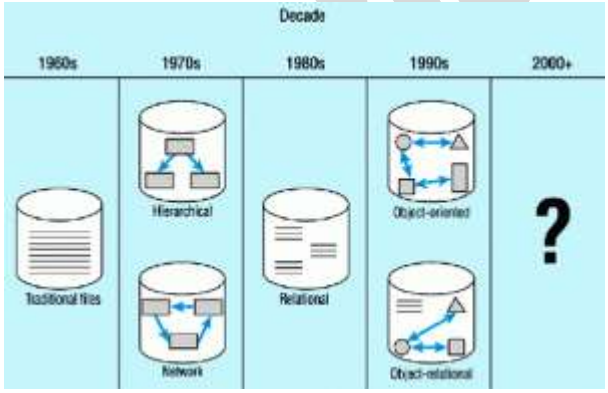


Figure 1 . Pictorial representation of Database Evolution over decades

Obviously, we cannot discuss all of the history material here, so if anyone wants to study more, here are the names of the models proposed up till now:

1. Semantic [SIM: Semantic Information Manager]
2. Multidimensional [MDDBMS]
3. Associative
4. Concept-Oriented [CODM: Concept-Oriented Data Model] etc.

People who deal with databases

Many persons are involved in the design, use and maintenance of any database. These persons can be classified into 2 types as below.

Actors on the scene:

The people, whose jobs involve the day-to-day use of a database are called as 'Actors on the scene', listed as below.

1. Database Administrators (DBA):

The DBA is responsible for authorizing access to the database, for Coordinating and monitoring its use and for acquiring software and hardware resources as needed. These are the people, who maintain and design the database daily.

DBA is responsible for the following issues.

a. Design of the conceptual and physical schemas:

The DBA is responsible for interacting with the users of the system to understand what data is to be stored in the DBMS and how it is likely to be used.

The DBA creates the original schema by writing a set of definitions and is Permanently stored in the 'Data Dictionary'.

b. Security and Authorization:

The DBA is responsible for ensuring the unauthorized data access is not permitted.

The granting of different types of authorization allows the DBA to regulate which parts of the database various users can access.

c. Storage structure and Access method definition:

The DBA creates appropriate storage structures and access methods by writing a set of definitions, which are translated by the DDL compiler.

d. Data Availability and Recovery from Failures:

The DBA must take steps to ensure that if the system fails, users can continue to access as much of the uncorrupted data as possible.

The DBA also work to restore the data to consistent state.

e. Database Tuning:

The DBA is responsible for modifying the database to ensure adequate Performance as requirements change.

f. Integrity Constraint Specification:

The integrity constraints are kept in a special system structure that is consulted by the

DBA whenever an update takes place in the system.

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.

3. End Users:

People who wish to store and use data in a database.

End users are the people whose jobs require access to the database for querying, updating and generating reports, listed as below.

a. Casual End users:

These people occasionally access the database, but they may need different information each time.

b. Naive or Parametric End Users:

Their job function revolves around constantly querying and updating the database using standard types of queries and updates.

c. Sophisticated End Users:

These include Engineers, Scientists, Business analyst and others familiarize to implement their applications to meet their complex requirements.

d. Stand alone End users:

These people maintain personal databases by using ready-made program packages that

provide easy to use menu based interfaces.

4. System Analyst:

These people determine the requirements of end users and develop specifications for transactions.

5. Application Programmers (Software Engineers):

These people can test, debug, document and maintain the specified transactions.

b. Workers behind the scene:

1. Database Designers and Implementers:

These people who design and implement the DBMS modules and interfaces as a software package.

2. Tool Developers:

Include persons who design and implement tools consisting the packages for design, performance monitoring, and prototyping and test data generation.

3. Operators and maintenance personnel:

These are the system administration personnel who are responsible for the actual running and maintenance of the hardware and software environment for the database system.

Client Server Platform:

Client/server is a term used to describe a computing model for the development of computerized systems. This model is based on the distribution of functions between two types of independent and autonomous processes; servers and clients.

Basic Components

A client is any process that requests specific services from server processes.
A server is a process that provides requested services for clients.
Both clients and servers can reside in the same computer or in different computers connected by a network.

Variations on Client Server

The key to client/server power is where the requested processing takes place.

In mainframe systems and Application Server based systems all processing takes place on the server, and the client is used to display the data screens.

With PC and File servers all processing takes place on the PC and the server is used only for storage

There are many variations of these models today (see below)

The client/server environment provides a clear separation of server and client processes.

Basic Client Server Model

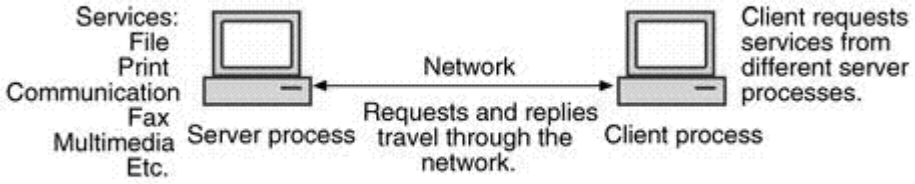


FIGURE 12.1 A BASIC CLIENT/SERVER COMPUTING MODEL

Three Components of Client/Server Architecture

The *client* is any computer process that requests services from the server. It is also known as the front-end application.

The *server* is any computer process providing services to the clients. The server is also known as the *back-end applicatio*n.

The *communication middlewar*e is any computer process(es) through which clients and servers communicate. It is also known as middleware or communications layer.

Interaction of Client and Server Components

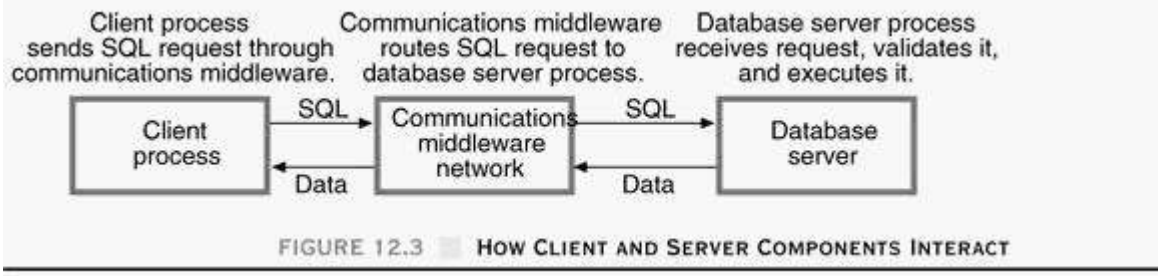


FIGURE 12.3 HOW CLIENT AND SERVER COMPONENTS INTERACT

Purpose of Database Management Systems

Organizations use large amounts of data. A **database management system** (DBMS) is a software tool that makes it possible to organize data in a database.

The standard acronym for database management system is **DBMS**, so you will often see this instead of the full name. The ultimate purpose of a database management system is to store and transform data into information to support making decisions.

A DBMS consists of the following three elements:

1. The **physical database**: the collection of files that contain the data
2. The **database engine**: the software that makes it possible to access and modify the contents of the database
3. The **database scheme**: the specification of the logical structure of the data stored in the database

While it sounds logical to have a DBMS in place, it is worth thinking for a moment about the alternative. What would the data in an organization look like without a DBMS? Consider yourself as the organization for a moment, and the data are all the files on your computer. How is your data organized? If you are like most typical computer users, you have a large number of files, organized in folders.

You may have word processor documents, presentation files, spreadsheets, photographs, etc. You find the information you need based on the folder structure you have created and the names you have given to your files. This is called a **file system** and is typical for individual computer users.

Now consider the challenges you are faced with. Have you ever lost a file? Have you had difficulty finding a file? Probably. Perhaps you are using multiple computers and your files are located in different physical locations. And, when was the list time you created a backup of all your files? You do back up, right?

You probably get the picture. A file system is relatively simple, but it only works if you keep yourself very organized and disciplined. Now consider an organization with 1,000 employees, each with their own computer. Can you see some of the challenges when using a file system? Do you really want critical financial data floating around the offices as simple files on individual computers?

Functions of a DBMS

So, what does a DBMS really do? It organizes your files to give you more control over your data.

A DBMS makes it possible for users to create, edit and update data in database files. Once created, the DBMS makes it possible to store and retrieve data from those database files.

More specifically, a DBMS provides the following functions:

- Concurrency: concurrent access (meaning 'at the same time') to the same database by multiple users

- Security: security rules to determine access rights of users
- Backup and recovery: processes to back-up the data regularly and recover data if a problem occurs
- Integrity: database structure and rules improve the integrity of the data
- Data descriptions: a data dictionary provides a description of the data

Within an organization, the development of the database is typically controlled by **database administrators (DBAs)** and other specialists. This ensures the database structure is efficient and reliable.

Database administrators also control access and security aspects. For example, different people within an organization use databases in different ways. Some employees may simply want to view the data and perform basic analysis. Other employees are actively involved in adding data to the database or updating existing data. This means that the database administrator needs to set the user permissions. You don't want someone who only needs to view the database to accidentally delete parts of the database.

Characteristics and Benefits of a Database

There are a number of characteristics that distinguish the database approach from the file-based system or approach. This chapter describes the benefits (and features) of the database system.

Self-describing nature of a database system

A database system is referred to as *self-describing* because it not only contains the database itself, but also *metadata* which defines and describes the data and relationships between tables in the database. This information is used by the DBMS software or database users if needed. This separation of data and information about the data makes a database system totally different from the traditional file-based system in which the data definition is part of the application programs.

Insulation between program and data

In the file-based system, the structure of the data files is defined in the application programs so if a user wants to change the structure of a file, all the programs that access that file might need to be changed as well.

On the other hand, in the database approach, the data structure is stored in the system catalogue and not in the programs. Therefore, one change is all that is needed to change the structure of a file. This insulation between the programs and data is also called program-data independence.

Support for multiple views of data

A database supports multiple views of data. A *view* is a subset of the database, which is defined and dedicated for particular users of the system. Multiple users in the system might have different views of the system. Each view might contain only the data of interest to a user or group of users.

Sharing of data and multiuser system

Current database systems are designed for multiple users. That is, they allow many users to access the same database at the same time. This access is achieved through features called *concurrency control strategies*. These strategies ensure that the data accessed are always correct and that data integrity is maintained.

The design of modern multiuser database systems is a great improvement from those in the past which restricted usage to one person at a time.

Control of data redundancy

In the database approach, ideally, each data item is stored in only one place in the database. In some cases, data redundancy still exists to improve system performance, but such redundancy is controlled by application programming and kept to minimum by introducing as little redudancy as possible when designing the database.

Data sharing

The integration of all the data, for an organization, within a database system has many advantages. First, it allows for data sharing among employees and others who have access to the system. Second, it gives users the ability to generate more information from a given amount of data than would be possible without the integration.

Enforcement of integrity constraints

Database management systems must provide the ability to define and enforce certain constraints to ensure that users enter valid information and maintain data integrity. A *database constraint* is a restriction or rule that dictates what can be entered or edited in a table such as a postal code using a certain format or adding a valid city in the City field.

There are many types of database constraints. *Data type*, for example, determines the sort of data permitted in a field, for example numbers only. *Data uniqueness* such as the primary key ensures that no duplicates are entered. Constraints can be simple (field based) or complex (programming).

Restriction of unauthorized access

Not all users of a database system will have the same accessing privileges. For example, one user might have *read-only access* (i.e., the ability to read a file but not make changes), while another might have *read and write privileges*, which is the ability to both read and modify a file. For this reason, a database management system should provide a security subsystem to create and control different types of user accounts and restrict unauthorized access.

Data independence

Another advantage of a database management system is how it allows for data independence. In other words, the system data descriptions or data describing data (metadata) are separated from the application programs. This is possible because changes to the data structure are handled by the database management system and are not embedded in the program itself.

Transaction processing

A database management system must include concurrency control subsystems. This feature ensures that data remains consistent and valid during transaction processing even if several users update the same information.

Provision for multiple views of data

By its very nature, a DBMS permits many users to have access to its database either individually or simultaneously. It is not important for users to be aware of how and where the data they access is stored

Backup and recovery facilities

Backup and recovery are methods that allow you to protect your data from loss. The database system provides a separate process, from that of a network backup, for backing up and recovering data. If a hard drive fails and the database stored on the hard drive is not accessible, the only way to recover the database is from a backup.
If a computer system fails in the middle of a complex update process, the recovery subsystem is responsible for making sure that the database is restored to its original state. These are two more benefits of a database management system.

DATA SHARING:

DATA SHARING:

The most significant difference between a file based systems and database systems is ***Data sharing***. Data sharing also requires a major change in the way of data are handled and managed with in the organization. Data sharing are of 3 (three) types. They are

1. Sharing Data between functional units.
2. Sharing data between management units.
3. Sharing data between geographically dispersed location.

1. *Sharing data between functional units:* The term data sharing suggests that people in different functional areas are use a common pool of data. Each of these are own applications without data sharing the marketing group may have their data files. The purchasing group like accounts group their own data files and marketing group have their own data files and so on… each group benefits from its own data. **Eg diagram.**

In contrasts the effect of combining data into database is **synergistic** that is the combined data are more valuable then the some of the data in separated files .Not only does each group continue to have access to its own data but with in a reasonable limits and control they have access to other data as well. In this environment the marketing department for eg: Is better of because it has access to data from purchasing, especially product evolution which provide valuable input for marketing campaigns.

2. *Sharing data between Management units:* **Diagram**

Different levels of users also need to share data . The three different levels of users are 1. Operation level, 2. Middle Management Level, 3. Execute level.

These three levels are corresponded to the three different types of system these are *Electronic data processing, Management information system, and Decision support system.*

EXECUTIVE à Decision Support System à Strategic Reports, Queries, Analysis.

MIDDLE MANAGEMENT à Management Information System à Management reports of functional areas.

Operations à Electronic Data Processing System à Transaction file, Maintenance Processing and control reports.

The relationships of the above three systems have to different management levels is shown in **diagram .**These levels of users and system naturally requires three different types of data. The user at the operational level needs data for transaction processing that is include data for new accounts are changes to an existing accounts management. The management **information system** level utilize summaries to indicate which sales representatives were most or least productive and executives at highest level used decision support system to discover long term trends that apply to their own corporation as well as to identify the economic social and political environment in which they operate. The **DSS** means Decision Support System help them makes the decisions such as building a new factory starting or dropping a product line and DSS uses summery data from with in the company as well as market and author data form outside sources.

3.*Sharing data between geographically dispersed location:* A company with several

locations has important **data distributed** over a valid geographically area sharing . these data is a significant problems. A centralized database is physically contained to a single location controlled by a single computer that is Personal computer most function for which databases are created and accomplished more easily . If the database is centralized and it is easily to update and back up , recovery and control access to a database . If we know database exactly where it is and what's software control it and identify the remote place where it is located.

Data modeling

Data modeling is the process of creating a data model for the data to be stored in a Database. This data model is a conceptual representation of

- **Data objects**
- **The associations between different data objects**
- **The rules.**

Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. Data Models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data.

Data model emphasizes on what data is needed and how it should be organized instead of what operations need to be performed on the data. Data Model is like architect's building plan which helps to build a conceptual model and set the relationship between data items.

The two types of Data Models techniques are

1. Entity Relationship (E-R) Model
2. UML (Unified Modelling Language)

Data Model

The primary goal of using data model are:

- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

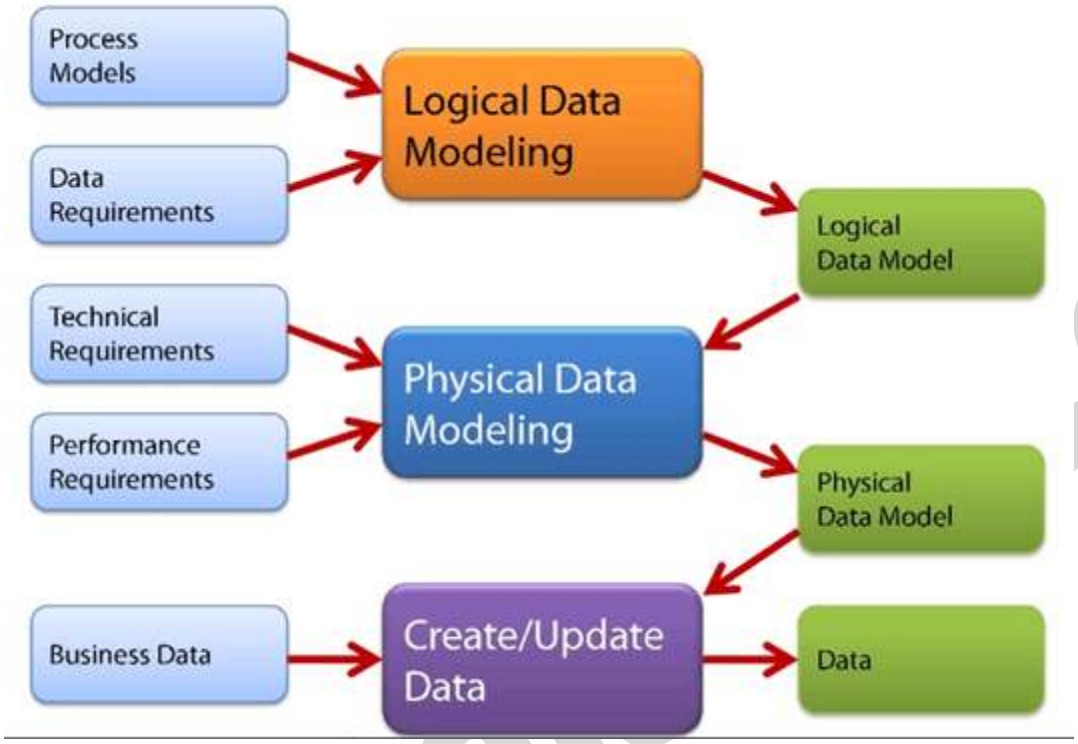
Types of Data Models

There are mainly three different types of data models:

1. **Conceptual:** This Data Model defines WHAT the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.

2. **Logical:** Defines HOW the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.

3. **Physical**: This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.



Conceptual Model

The main aim of this model is to establish the entities, their attributes, and their relationships. In this Data modeling level, there is hardly any detail available of the actual Database structure.

The 3 basic tenants of Data Model are

Entity: A real-world thing

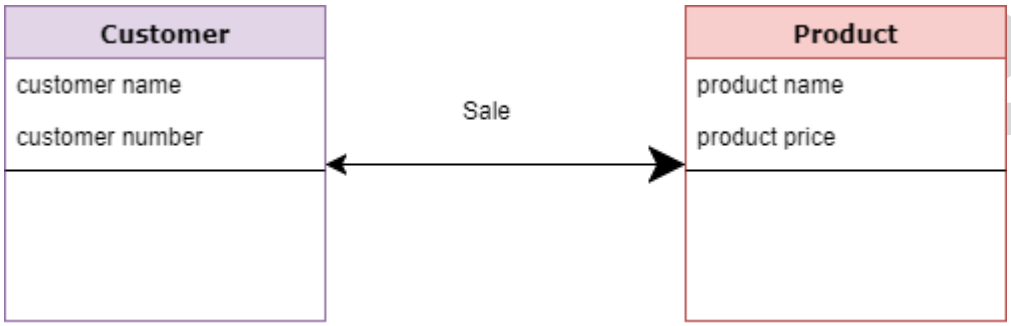
Attribute: Characteristics or properties of an entity

Relationship: Dependency or association between two entities

For example:

- Customer and Product are two entities. Customer number and name are attributes of the Customer entity

- Product name and price are attributes of product entity
- Sale is the relationship between the customer and product



Characteristics of a conceptual data model

- Offers Organisation-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world."

Conceptual data models known as Domain models create a common vocabulary for all stakeholders by establishing basic concepts and scope.

Logical Data Model

Logical data models add further information to the conceptual model elements. It defines the structure of the data elements and set the relationships between them.

The advantage of the Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic.

At this Data Modeling level, no primary or secondary key is defined. At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships.

Characteristics of a Logical data model

- Describes data needs for a single project but could integrate with other logical data models based on the scope of the project.
- Designed and developed independently from the DBMS.
- Data attributes will have datatypes with exact precisions and length.
- Normalization processes to the model is applied typically till 3NF.

Physical Data Model

A Physical Data Model describes the database specific implementation of the data model. It offers an abstraction of the database and helps generate schema. This is because of the richness of meta-data offered by a Physical Data Model.

This type of Data model also helps to visualize database structure. It helps to model database columns keys, constraints, indexes, triggers, and other RDBMS features.

Characteristics of a physical data model:

- The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact datatypes, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined.

Advantages and Disadvantages of Data Model:

Advantages of Data model:

- The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately.
- The data model should be detailed enough to be used for building the physical database.
- The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.
- Data Model helps business to communicate the within and across organizations.
- Data model helps to documents data mappings in ETL process
- Help to recognize correct sources of data to populate the model

Disadvantages of Data model:

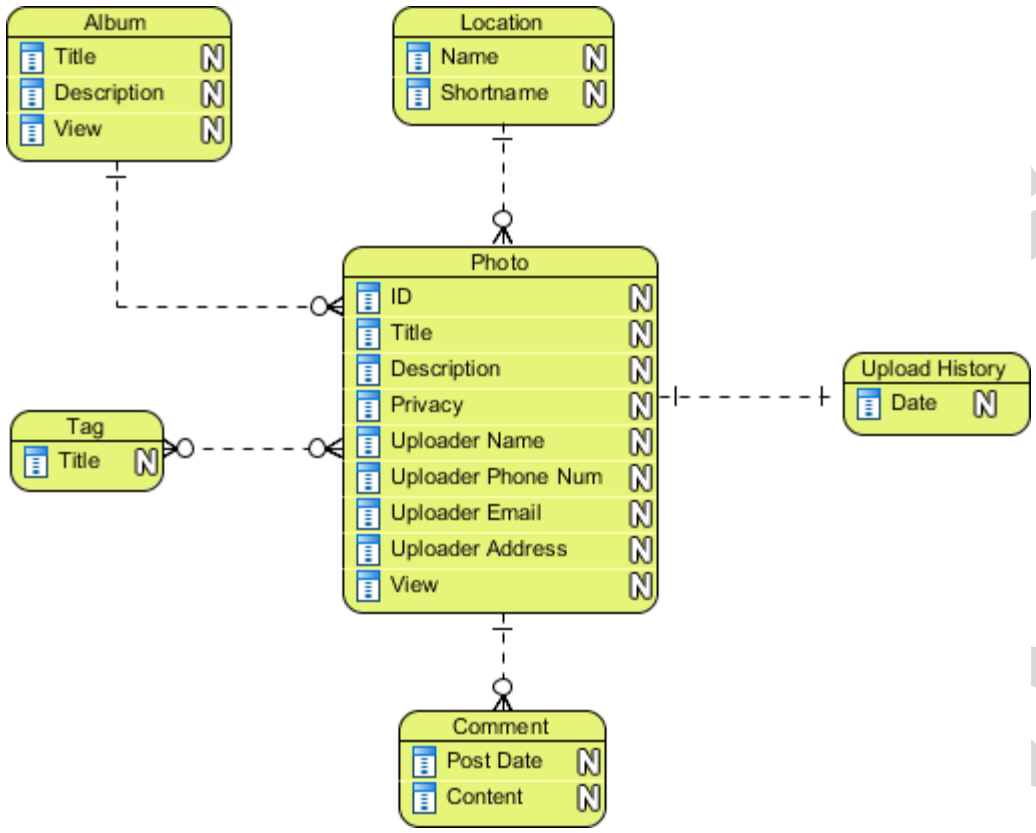
- To developer Data model one should know physical data stored characteristics.
- This is a navigational system produces complex application development, management. Thus, it requires a knowledge of the biographical truth.
- Even smaller change made in structure require modification in the entire application.
- There is no set data manipulation language in DBMS.

Conceptual, Logical and Physical Data Model

Conceptual, logical and physical model or ERD are three different ways of modeling data in a domain. While they all contain entities and relationships, they differ in the purposes they are created for and audiences they are meant to target. A general understanding to the three models is that, business analyst uses conceptual and logical model for modeling the data required and produced by system from a business angle, while database designer refines the early design to produce the physical model for presenting physical database structure ready for database construction.
With Visual Paradigm, you can draw the three types of model, plus to progress through models through the use of Model Transitor.

Conceptual Model

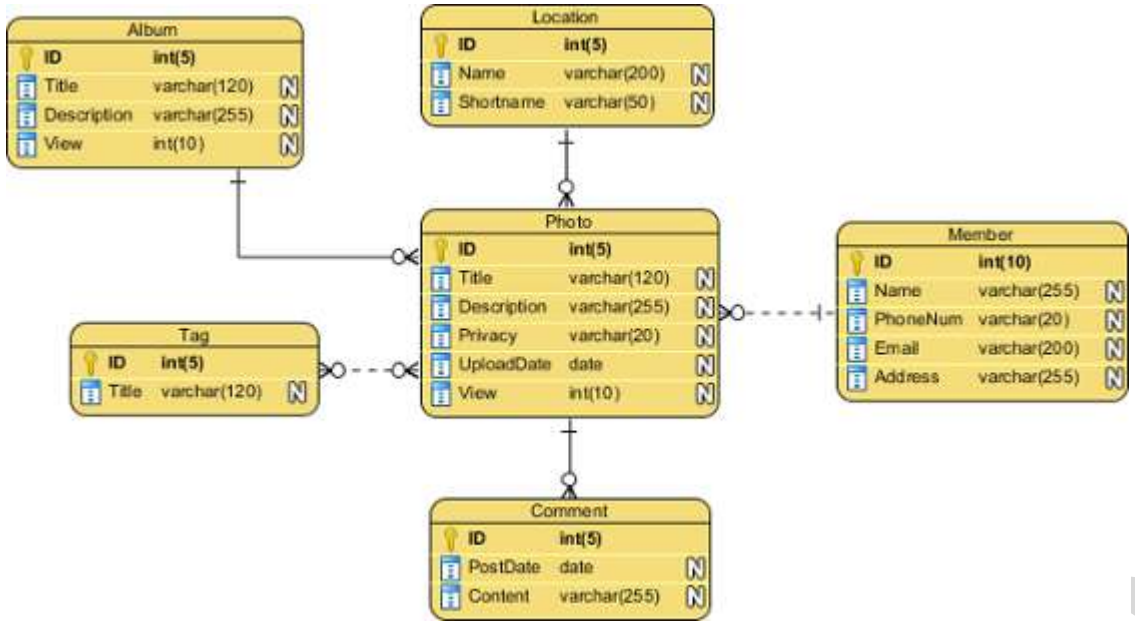
Conceptual ERD models information gathered from business requirements. Entities and relationships modeled in such ERD are defined around the business's need. The need of satisfying the database design is not considered yet. Conceptual ERD is the simplest model among all.



Conceptual ERD example

Logical Model

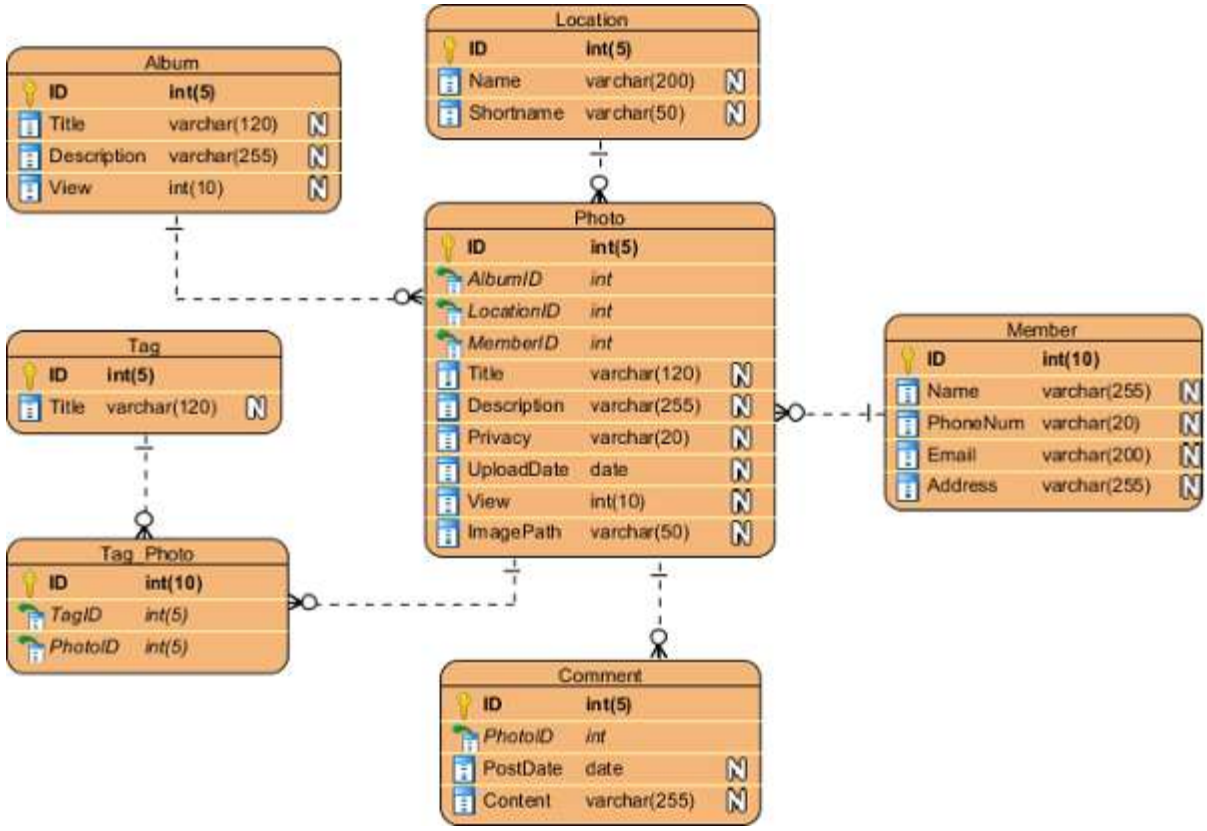
Logical ERD also models information gathered from business requirements. It is more complex than conceptual model in that column types are set. Note that the setting of column types is optional and if you do that, you should be doing that to aid business analysis. It has nothing to do with database creation yet.



Logical ERD example

Physical Model

Physical ERD represents the actual design blueprint of a relational database. It represents how data should be structured and related in a specific DBMS so it is important to consider the convention and restriction of the DBMS you use when you are designing a physical ERD. This means that an accurate use of data type is needed for entity columns and the use of reserved words has to be avoided in naming entities and columns. Besides, database designers may also add primary keys, foreign keys and constraints to the design.



Physical ERD example

Transiting from Conceptual/Logical to Physical ERD

Model Transitor enables you to transit a logical ERD to a physical ERD and with the transition relationship maintained. To transit, right click on the background of your conceptual/logical ERD, and then select **Utilities > Transit to Logical/Physical ERD...** from the popup menu. This will form a new ERD with new entities there. You can make changes like to rename the entities and columns, or to add extra entities in the new ERD.

STRATEGY OF DATABASE PLANNING:

Data must be viewed as a corporate resource and other corporate resources must be devoted to the development, implement and use of one or more databases. Database planning is strategy corporate effort to determine the information needs of the organization form extended period into the future. A successful database planning project will precede operational project to design and implement new databases to satisfy the organizational information needs.

The need for database planning: Database planning is directed by the information needs of the organization which in turn or determined by the company's **business plan** . the process is shown in the below diagram.

Business Plan à Information Needs à Database Plan à Database Development Projects.

The corporation formulates has its strategy business plan for the next 5 (five) years . Accomplishing the objectives of this plans depends on the availability of certain identify types of information . The information can we obtained only if the data sources are identified in the database planning, or in place . This indicates the needs of the database development projects which create new database or enhance or integrate existing databases. *Database planning has significant advantages* James cites several advantages of formal information resource plan.

1. It expresses management current understanding of the information resources.
2. It identifies and justifies resources requirements, helping ensure that the resources will be available.
3. It identifies and justifies for effective resources management including the collaboration among departments or diligence with in the organization.
4. It Specifies action plans for achieving objectives.
5. It can provide a powerful stimulus and sense of direction to employees at all levels.

COSTS & RISKS involved in database development:

In database approach in order to maintain or develop database we should take a risk and we should invest money , time and environment. Database approach when we develop a new database or when we maintain an existing database we should consider the following points.

- New specialized persons.
- Installation and manage cost and complexity.
- Conversion costs.

- Need for explicit back up and recovery.
- Organizational conflict.

New specialized persons: Organization adopted the database approach to maintain individuals to design and implement database for that an organization provides DBA and manage a staff of new people because there is a rapid changes in the technology .New people will have to be retrained or upgraded their knowledge in database approach in an organization required specialized skill persons to maintain the database.

Installation and manage cost and complexity: A multi user database management system is a large and complex . It has a high level install cost requires a staff to trained persons to install and operate . It also require professionals to maintain annual maintenance and cost .These systems are require to install new software to upgrade the database for that we should take a risk to modify Hard ware and data communications in an organization.

Unit II

Database Development Process

A core aspect of software engineering is the subdivision of the development process into a series of phases, or steps, each of which focuses on one aspect of the development. The collection of these steps is sometimes referred to as the *software development life cycle (SDLC)*. The software product moves through this life cycle (sometimes repeatedly as it is refined or redeveloped) until it is finally retired from use. Ideally, each phase in the life cycle can be checked for correctness before moving on to the next phase.

Software Development Life Cycle – Waterfall

Let us start with an overview of the *waterfall model* such as you will find in most software engineering textbooks. This waterfall figure, seen in Figure 13.1, illustrates a general waterfall model that could apply to any computer system development. It shows the process as a strict sequence of steps where the output of one step is the input to the next and all of one step has to be completed before moving onto the next.

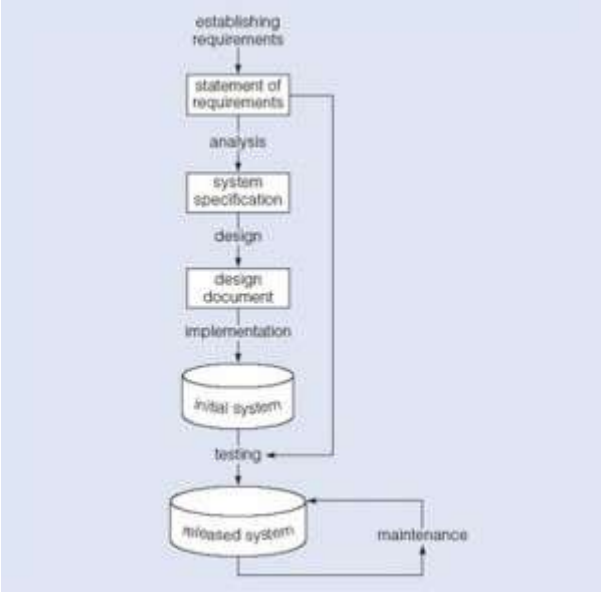


Figure 13.1. Waterfall model.

We can use the *waterfall process* as a means of identifying the tasks that are required, together with the input and output for each activity. What is important is the scope of the activities, which can be summarized as follows:

- *Establishing requirements* involves consultation with, and agreement among, stakeholders about what they want from a system, expressed as a statement of requirements.
- *Analysis* starts by considering the statement of requirements and finishes by producing a system specification. The specification is a formal representation of what a system should do, expressed in terms that are independent of how it may be realized.
- *Design* begins with a system specification, produces design documents and provides a detailed description of how a system should be constructed.
- *Implementation* is the construction of a computer system according to a given design document and taking into account the environment in which the system will be operating (e.g., specific hardware or software available for the development). Implementation may be staged, usually with an initial system that can be validated and tested before a final system is released for use.
- *Testing* compares the implemented system against the design documents and requirements specification and produces an acceptance report or, more usually, a list of errors and bugs that require a review of the analysis, design and implementation processes to correct (testing is usually the task that leads to the waterfall model iterating through the life cycle).
- *Maintenance* involves dealing with changes in the requirements or the implementation environment, bug fixing or porting of the system to new environments (e.g., migrating a system

from a standalone PC to a UNIX workstation or a networked environment). Since maintenance involves the analysis of the changes required, design of a solution, implementation and testing of that solution over the lifetime of a maintained software system, the waterfall life cycle will be repeatedly revisited.

Database Life Cycle

We can use the waterfall cycle as the basis for a model of database development that incorporates three assumptions:

1. We can separate the development of a database – that is, specification and creation of a schema to define data in a database – from the user processes that make use of the database.
2. We can use the three-schema architecture as a basis for distinguishing the activities associated with a schema.
3. We can represent the constraints to enforce the semantics of the data once within a database, rather than within every user process that uses the data.

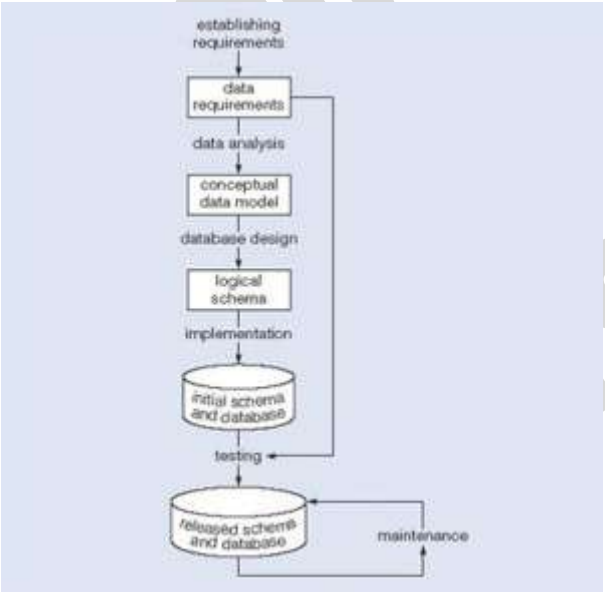


Figure 13.2. A waterfall model of the activities and their outputs for database development.

Using these assumptions and Figure 13.2, we can see that this diagram represents a model of the activities and their outputs for database development. It is applicable to any class of DBMS, not just a relational approach.

Database application development is the process of obtaining real-world requirements, analyzing requirements, designing the data and functions of the system, and then implementing the operations in the system.

Requirements Gathering

The first step is *requirements gathering*. During this step, the database designers have to interview the customers (database users) to understand the proposed system and obtain and document the data and functional requirements. The result of this step is a document that includes the detailed requirements provided by the users.

Establishing requirements involves consultation with, and agreement among, all the users as to what persistent data they want to store along with an agreement as to the meaning and interpretation of the data elements. The data administrator plays a key role in this process as they overview the business, legal and ethical issues within the organization that impact on the data requirements.

The *data requirements document* is used to confirm the understanding of requirements with users. To make sure that it is easily understood, it should not be overly formal or highly encoded. The document should give a concise summary of all users' requirements – not just a collection of individuals' requirements – as the intention is to develop a single shared database.

The requirements should not describe how the data is to be processed, but rather what the data items are, what attributes they have, what constraints apply and the relationships that hold between the data items.

Analysis

Data analysis begins with the statement of data requirements and then produces a conceptual data model. The aim of analysis is to obtain a detailed description of the data that will suit user requirements so that both high and low level properties of data and their use are dealt with. These include properties such as the possible range of values that can be permitted for attributes (e.g., in the school database example, the student course code, course title and credit points).

The conceptual data model provides a shared, formal representation of what is being communicated between clients and developers during database development – it is focused on the data in a database, irrespective of the eventual use of that data in user processes or implementation of the data in specific computer environments. Therefore, a conceptual data model is concerned with the meaning and structure of data, but not with the details affecting how they are implemented.

The conceptual data model then is a formal representation of what data a database should contain and the constraints the data must satisfy. This should be expressed in terms that are independent of how the model may be implemented. As a result, analysis focuses on the questions, "What is required?" not "How is it achieved?"

Logical Design

Database design starts with a conceptual data model and produces a specification of a logical schema; this will determine the specific type of database system (network, relational, object-oriented) that is

required. The relational representation is still independent of any specific DBMS; it is another conceptual data model.

We can use a relational representation of the conceptual data model as input to the logical design process. The output of this stage is a detailed relational specification, the logical schema, of all the tables and constraints needed to satisfy the description of the data in the conceptual data model. It is during this design activity that choices are made as to which tables are most appropriate for representing the data in a database. These choices must take into account various design criteria including, for example, flexibility for change, control of duplication and how best to represent the constraints. It is the tables defined by the logical schema that determine what data are stored and how they may be manipulated in the database.

Database designers familiar with relational databases and SQL might be tempted to go directly to implementation after they have produced a conceptual data model. However, such a direct transformation of the relational representation to SQL tables does not necessarily result in a database that has all the desirable properties: completeness, integrity, flexibility, efficiency and usability. A good conceptual data model is an essential first step towards a database with these properties, but that does not mean that the direct transformation to SQL tables automatically produces a good database. This first step will accurately represent the tables and constraints needed to satisfy the conceptual data model description, and so will satisfy the completeness and integrity requirements, but it may be inflexible or offer poor usability. The first design is then flexed to improve the quality of the database design. *Flexing* is a term that is intended to capture the simultaneous ideas of bending something for a different purpose and weakening aspects of it as it is bent.

Figure 13.3 summarizes the iterative (repeated) steps involved in database design, based on the overview given. Its main purpose is to distinguish the general issue of what tables should be used from the detailed definition of the constituent parts of each table – these tables are considered one at a time, although they are not independent of each other. Each iteration that involves a revision of the tables would lead to a new design; collectively they are usually referred to as *second-cut designs*, even if the process iterates for more than a single loop.

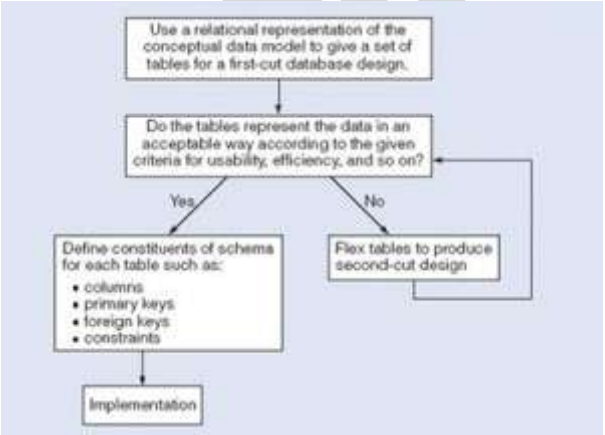


Figure 13.3. A summary of the iterative steps involved in database design.

First, for a given conceptual data model, it is not necessary that all the user requirements it represents be satisfied by a single database. There can be various reasons for the development of more than one database, such as the need for independent operation in different locations or departmental control over "their" data. However, if the collection of databases contains duplicated data and users need to access data in more than one database, then there are possible reasons that one database can satisfy multiple requirements, or issues related to data replication and distribution need to be examined.

Second, one of the assumptions about database development is that we can separate the development of a database from the development of user processes that make use of it. This is based on the expectation that, once a database has been implemented, all data required by currently identified user processes have been defined and can be accessed; but we also require flexibility to allow us to meet future requirements changes. In developing a database for some applications, it may be possible to predict the common requests that will be presented to the database and so we can optimize our design for the most common requests.

Third, at a detailed level, many aspects of database design and implementation depend on the particular DBMS being used. If the choice of DBMS is fixed or made prior to the design task, that choice can be used to determine design criteria rather than waiting until implementation. That is, it is possible to incorporate design decisions for a specific DBMS rather than produce a generic design and then tailor it to the DBMS during implementation.

It is not uncommon to find that a single design cannot simultaneously satisfy all the properties of a good database. So it is important that the designer has prioritized these properties (usually using information from the requirements specification); for example, to decide if integrity is more important than efficiency and whether usability is more important than flexibility in a given development.

At the end of our design stage, the logical schema will be specified by SQL data definition language (DDL) statements, which describe the database that needs to be implemented to meet the user requirements.

Implementation

Implementation involves the construction of a database according to the specification of a logical schema. This will include the specification of an appropriate storage schema, security enforcement, external schema and so on. Implementation is heavily influenced by the choice of available DBMSs, database tools and operating environment. There are additional tasks beyond simply creating a database schema and implementing the constraints – data must be entered into the tables, issues relating to the users and user processes need to be addressed, and the management activities associated with wider aspects of corporate data management need to be supported. In keeping with the DBMS approach, we want as many of these concerns as possible to be addressed within the DBMS. We look at some of these concerns briefly now.

In practice, implementation of the logical schema in a given DBMS requires a very detailed knowledge of the specific features and facilities that the DBMS has to offer. In an ideal world, and in keeping with good

software engineering practice, the first stage of implementation would involve matching the design requirements with the best available implementing tools and then using those tools for the implementation. In database terms, this might involve choosing vendor products with DBMS and SQL variants most suited to the database we need to implement. However, we don't live in an ideal world and more often than not, hardware choice and decisions regarding the DBMS will have been made well in advance of consideration of the database design. Consequently, implementation can involve additional flexing of the design to overcome any software or hardware limitations.

Realizing the Design

After the logical design has been created, we need our database to be created according to the definitions we have produced. For an implementation with a relational DBMS, this will probably involve the use of SQL to create tables and constraints that satisfy the logical schema description and the choice of appropriate storage schema (if the DBMS permits that level of control).

One way to achieve this is to write the appropriate SQL DDL statements into a file that can be executed by a DBMS so that there is an independent record, a text file, of the SQL statements defining the database. Another method is to work interactively using a database tool like SQL Server Management Studio or Microsoft Access. Whatever mechanism is used to implement the logical schema, the result is that a database, with tables and constraints, is defined but will contain no data for the user processes.

Populating the Database

After a database has been created, there are two ways of populating the tables – either from existing data or through the use of the user applications developed for the database.

For some tables, there may be existing data from another database or data files. For example, in establishing a database for a hospital, you would expect that there are already some records of all the staff that have to be included in the database. Data might also be brought in from an outside agency (address lists are frequently brought in from external companies) or produced during a large data entry task (converting hard-copy manual records into computer files can be done by a data entry agency). In such situations, the simplest approach to populate the database is to use the import and export facilities found in the DBMS.

Facilities to import and export data in various standard formats are usually available (these functions are also known in some systems as loading and unloading data). Importing enables a file of data to be copied directly into a table. When data are held in a file format that is not appropriate for using the import function, then it is necessary to prepare an application program that reads in the old data, transforms them as necessary and then inserts them into the database using SQL code specifically produced for that purpose. The transfer of large quantities of existing data into a database is referred to as a *bulk load*. Bulk loading of data may involve very large quantities of data being loaded, one table at a time so you may find that there are DBMS facilities to postpone constraint checking until the end of the bulk loading.

Guidelines for Developing an ER Diagram

Note: These are general guidelines that will assist in developing a strong basis for the actual database design (the logical model).

1. Document all entities discovered during the information-gathering stage.
2. Document all attributes that belong to each entity. Select candidate and primary keys. Ensure that all non-key attributes for each entity are full-functionally dependent on the primary key.
3. Develop an initial ER diagram and review it with appropriate personnel. (Remember that this is an iterative process.)
4. Create new entities (tables) for multivalued attributes and repeating groups. Incorporate these new entities (tables) in the ER diagram. Review with appropriate personnel.
5. Verify ER modeling by normalizing tables.

The Relational Data Model

The relational data model was introduced by C. F. Codd in 1970. Currently, it is the most widely used data model.

The relational model has provided the basis for:

- Research on the theory of data/relationship/constraint
- Numerous database design methodologies
- The standard database access language called s*tructured query language (SQL)*
- Almost all modern commercial database management systems

The relational data model describes the world as "a collection of inter-related relations (or tables)."

Fundamental Concepts in the Relational Data Model

Relation

A *relation,* also known as a *table* or *file*, is a subset of the Cartesian product of a list of domains characterized by a name. And within a table, each row represents a group of related data values. A *row*, or record, is also known as a *tuple*. The columns in a table is a field and is also referred to as an attribute. You can also think of it this way: an attribute is used to define the record and a record contains a set of attributes.

The steps below outline the logic between a relation and its domains.

1. Given *n* domains are denoted by D1, D2, … Dn
2. And *r* is a relation defined on these domains

3. Then $r \subseteq D1 \times D2 \times \ldots \times Dn$

Table

A database is composed of multiple tables and each table holds the data. Figure 7.1 shows a database that contains three tables**.**

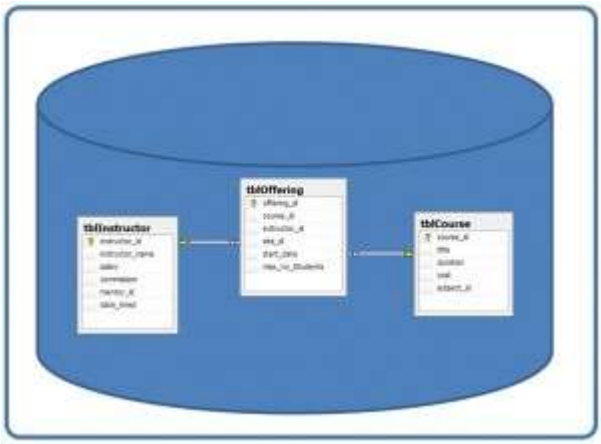


Figure 7.1. Database with three tables.

Column

A database stores pieces of information or facts in an organized way. Understanding how to use and get the most out of databases requires us to understand that method of organization.

The principal storage units are called *columns* or *fields* or *attributes*. These house the basic components of data into which your content can be broken down. When deciding which fields to create, you need to think generically about your information, for example, drawing out the common components of the information that you will store in the database and avoiding the specifics that distinguish one item from another.

Look at the example of an ID card in Figure 7.2 to see the relationship between fields and their data.

Field Name	Data
First Name	Isabelle
Family Name	Whelan
Nationality	British
Salary	109,900
Date of Birth	15 September 1983
Marital Status	Single
Shift	Mon, Wed
Place of issue	Addis Ababa
Valid until	17 December 2003

Figure 7.2. Example of an ID card by A. Watt.

Domain

A *domain* is the original sets of atomic values used to model data. By *atomic value*, we mean that each value in the domain is indivisible as far as the relational model is concerned. For example:

- The domain of Marital Status has a set of possibilities: Married, Single, Divorced.
- The domain of Shift has the set of all possible days: {Mon, Tue, Wed…}.
- The domain of Salary is the set of all floating-point numbers greater than 0 and less than 200,000.
- The domain of First Name is the set of character strings that represents names of people.

In summary, a domain is a set of acceptable values that a column is allowed to contain. This is based on various properties and the data type for the column.

Records

Just as the content of any one document or item needs to be broken down into its constituent bits of data for storage in the fields, the link between them also needs to be available so that they can be reconstituted into their whole form. Records allow us to do this. *Records* contain fields that are related, such as a customer or an employee. As noted earlier, a tuple is another term used for record.

Records and fields form the basis of all databases. A simple table gives us the clearest picture of how records and fields work together in a database storage project.

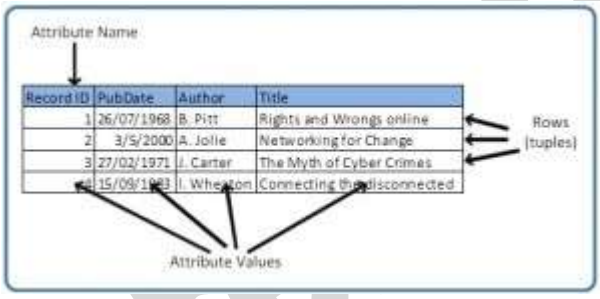


Figure 7.3. Example of a simple table by A. Watt.

The simple table example in Figure 7.3 shows us how fields can hold a range of different sorts of data. This one has:

- A Record ID field: this is an ordinal number; its data type is an integer.
- A PubDate field: this is displayed as day/month/year; its data type is date.
- An Author field: this is displayed as Initial. Surname; its data type is text.
- A Title field text: free text can be entered here.

You can command the database to sift through its data and organize it in a particular way. For example, you can request that a selection of records be limited by date: 1. all before a given date, 2. all after a given date or 3. all between two given dates. Similarly, you can choose to have records sorted by date. Because the field, or record, containing the data is set up as a Date field, the database reads the information in the Date field not just as numbers separated by slashes, but rather, as dates that must be ordered according to a calendar system.

Degree

The *degree* is the number of attributes in a table. In our example in Figure 7.3, the degree is 4.

Properties of a Table

- A table has a name that is distinct from all other tables in the database.
- There are no duplicate rows; each row is distinct.
- Entries in columns are atomic. The table does not contain repeating groups or multivalued attributes.
- Entries from columns are from the same domain based on their data type including:
    - number (numeric, integer, float, smallint,…)
    - character (string)
    - date
    - logical (true or false)
- Operations combining different data types are disallowed.
- Each attribute has a distinct name.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.

The Entity Relationship Data Model

The *entity relationship (ER) data model* has existed for over 35 years. It is well suited to data modelling for use with databases because it is fairly abstract and is easy to discuss and explain. ER models are readily translated to relations. ER models, also called an ER schema, are represented by ER diagrams.

ER modelling is based on two concepts:

- Entities, defined as tables that hold specific information (data)

- *Relationships,* defined as the associations or interactions between entities

Here is an example of how these two concepts might be combined in an ER data model: Prof. Ba (entity) teaches (relationship) the Database Systems course (entity).

For the rest of this chapter, we will use a sample database called the COMPANY database to illustrate the concepts of the ER model. This database contains information about employees, departments and projects. Important points to note include:

- There are several departments in the company. Each department has a unique identification, a name, location of the office and a particular employee who manages the department.
- A department controls a number of projects, each of which has a unique name, a unique number and a budget.
- Each employee has a name, identification number, address, salary and birthdate. An employee is assigned to one department but can join in several projects. We need to record the start date of the employee in each project. We also need to know the direct supervisor of each employee.
- We want to keep track of the dependents for each employee. Each dependent has a name, birthdate and relationship with the employee.

Entity, Entity Set and Entity Type

An *entity* is an object in the real world with an independent existence that can be differentiated from other objects. An entity might be

- An object with physical existence (e.g., a lecturer, a student, a car)
- An object with conceptual existence (e.g., a course, a job, a position)

Entities can be classified based on their strength. An entity is considered weak if its tables are existence dependent.

- That is, it cannot exist without a relationship with another entity
- Its primary key is derived from the primary key of the parent entity
    - The Spouse table, in the COMPANY database, is a weak entity because its primary key is dependent on the Employee table. Without a corresponding employee record, the spouse record would not exist.

An entity is considered strong if it can exist apart from all of its related entities.

- Kernels are strong entities.
- A table without a foreign key or a table that contains a foreign key that can contain nulls is a strong entity

Another term to know is *entity type* which defines a collection of similar entities.

An *entity set* is a collection of entities of an entity type at a particular point of time. In an entity relationship diagram (ERD), an entity type is represented by a name in a box. For example, in Figure 8.1, the entity type is EMPLOYEE.

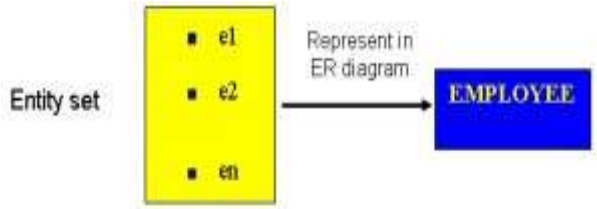


Figure 8.1. ERD with entity type EMPLOYEE.

Existence dependency

An entity's existence is dependent on the existence of the related entity. It is existence-dependent if it has a mandatory foreign key (i.e., a foreign key attribute that cannot be null). For example, in the COMPANY database, a Spouse entity is existence -dependent on the Employee entity.

Kinds of Entities

You should also be familiar with different kinds of entities including independent entities, dependent entities and characteristic entities. These are described below.

Independent entities

Independent entities, also referred to as kernels, are the backbone of the database. They are what other tables are based on. *Kernels* have the following characteristics:

- They are the building blocks of a database.
- The primary key may be simple or composite.
- The primary key is not a foreign key.
- They do not depend on another entity for their existence.

If we refer back to our COMPANY database, examples of an independent entity include the Customer table, Employee table or Product table.

Dependent entities

Dependent entities, also referred to as *derived entities*, depend on other tables for their meaning. These entities have the following characteristics:

- Dependent entities are used to connect two kernels together.

- They are said to be existence dependent on two or more tables.
- Many to many relationships become associative tables with at least two foreign keys.
- They may contain other attributes.
- The foreign key identifies each associated table.
- There are three options for the primary key:
    1. Use a composite of foreign keys of associated tables if unique
    2. Use a composite of foreign keys and a qualifying column
    3. Create a new simple primary key

Characteristic entities

Characteristic entities provide more information about another table. These entities have the following characteristics:

- They represent multivalued attributes.
- They describe other entities.
- They typically have a one to many relationship.
- The foreign key is used to further identify the characterized table.
- Options for primary key are as follows:
    1. Use a composite of foreign key plus a qualifying column
    2. Create a new simple primary key. In the COMPANY database, these might include:
        - Employee (EID, Name, Address, Age, Salary) – EID is the simple primary key.
        - EmployeePhone (EID, Phone) – EID is part of a composite primary key. Here, EID is also a foreign key.

Attributes

Each entity is described by a set of attributes (e.g., Employee = (Name, Address, Birthdate (Age), Salary).

Each attribute has a name, and is associated with an entity and a domain of legal values. However, the information about attribute domain is not presented on the ERD.

In the entity relationship diagram, shown in Figure 8.2, each attribute is represented by an oval with a name inside.

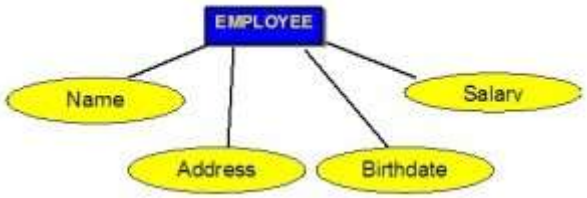


Figure 8.2. How attributes are represented in an ERD.

Types of Attributes

There are a few types of attributes you need to be familiar with. Some of these are to be left as is, but some need to be adjusted to facilitate representation in the relational model. This first section will discuss the types of attributes. Later on we will discuss fixing the attributes to fit correctly into the relational model.

Simple attributes

Simple attributes are those drawn from the atomic value domains; they are also called *single-valued attributes*. In the COMPANY database, an example of this would be: Name = {John} ; Age = {23}

Composite attributes

Composite attributes are those that consist of a hierarchy of attributes. Using our database example, and shown in Figure 8.3, Address may consist of Number, Street and Suburb. So this would be written as → Address = {59 + 'Meek Street' + 'Kingsford'}

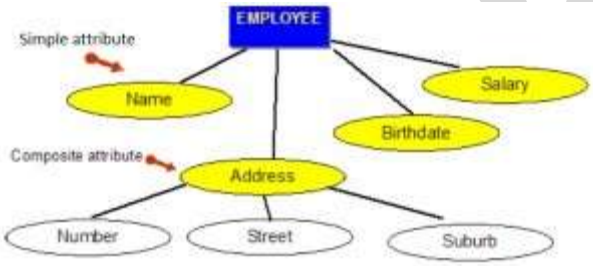


Figure 8.3. An example of composite attributes.

Multivalued attributes

Multivalued attributes are attributes that have a set of values for each entity. An example of a multivalued attribute from the COMPANY database, as seen in Figure 8.4, are the degrees of an employee: BSc, MIT, PhD.

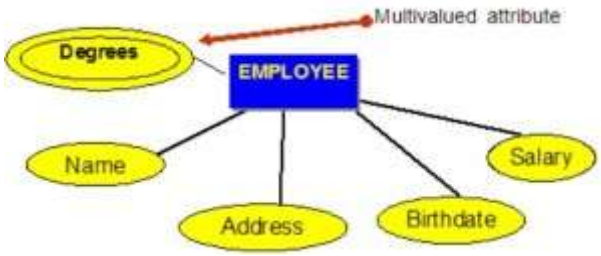


Figure 8.4. Example of a multivalued attribute.

Derived attributes

Derived attributes are attributes that contain values calculated from other attributes. An example of this can be seen in Figure 8.5. Age can be derived from the attribute Birthdate. In this situation, Birthdate is called a *stored attribute,* which is physically saved to the database.

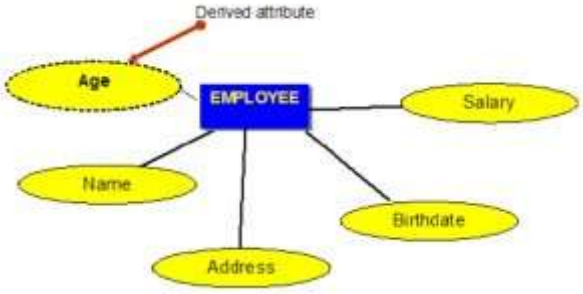


Figure 8.5. Example of a derived attribute.

Keys

An important constraint on an entity is the key. The *key* is an attribute or a group of attributes whose values can be used to uniquely identify an individual entity in an entity set.

Types of Keys

There are several types of keys. These are described below.

Candidate key

A *candidate key* is a simple or composite key that is unique and minimal. It is unique because no two rows in a table may have the same value at any time. It is minimal because every column is necessary in order to attain uniqueness.

From our COMPANY database example, if the entity is Employee(EID, First Name, Last Name, *SIN*, Address, Phone, BirthDate, Salary, DepartmentID), possible candidate keys are:

- EID, SIN

- First Name and Last Name – assuming there is no one else in the company with the same name
- Last Name and DepartmentID – assuming two people with the same last name don't work in the same department

Composite key

A *composite key* is composed of two or more attributes, but it must be minimal.

Using the example from the candidate key section, possible composite keys are:

- First Name and Last Name – assuming there is no one else in the company with the same name
- Last Name and Department ID – assuming two people with the same last name don't work in the same department

Primary key

The primary key is a candidate key that is selected by the database designer to be used as an identifying mechanism for the whole entity set. It must uniquely identify tuples in a table and not be null. The primary key is indicated in the ER model by underlining the attribute.

- A candidate key is selected by the designer to uniquely identify tuples in a table. It must not be null.
- A key is chosen by the database designer to be used as an identifying mechanism for the whole entity set. This is referred to as the primary key. This key is indicated by underlining the attribute in the ER model.

In the following example, EID is the primary key:

Employee(<u>EID</u>, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID)

Secondary key

A *secondary key* is an attribute used strictly for retrieval purposes (can be composite), for example: Phone and Last Name.

Alternate key

Alternate keys are all candidate keys not chosen as the primary key.

Foreign key

A *foreign key (FK)* is an attribute in a table that references the primary key in another table OR it can be null. Both foreign and primary keys must be of the same data type.

In the COMPANY database example below, DepartmentID is the foreign key:

Employee(EID, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID)

Nulls

A *null* is a special symbol, independent of data type, which means either unknown or inapplicable. It does not mean zero or blank. Features of null include:

- No data entry
- Not permitted in the primary key
- Should be avoided in other attributes
- Can represent
    - An unknown attribute value
    - A known, but missing, attribute value
    - A "not applicable" condition
- Can create problems when functions such as COUNT, AVERAGE and SUM are used
- Can create logical problems when relational tables are linked

Normalization

Normalization should be part of the database design process. However, it is difficult to separate the normalization process from the ER modelling process so the two techniques should be used concurrently.

Use an entity relation diagram (ERD) to provide the big picture, or macro view, of an organization's data requirements and operations. This is created through an iterative process that involves identifying relevant entities, their attributes and their relationships.

Normalization procedure focuses on characteristics of specific entities and represents the micro view of entities within the ERD.

What Is Normalization?

Normalization is the branch of relational theory that provides design insights. It is the process of determining how much redundancy exists in a table. The goals of normalization are to:

- Be able to characterize the level of redundancy in a relational schema
- Provide mechanisms for transforming schemas in order to remove redundancy

Normalization theory draws heavily on the theory of functional dependencies. Normalization theory defines six normal forms (NF). Each normal form involves a set of dependency properties that a schema must satisfy and each normal form gives guarantees about the presence and/or absence of update anomalies. This means that higher normal forms have less redundancy, and as a result, fewer update problems.

Normal Forms

All the tables in any database can be in one of the normal forms we will discuss next. Ideally we only want minimal redundancy for PK to FK. Everything else should be derived from other tables. There are six normal forms, but we will only look at the first four, which are:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)

BCNF is rarely used.

First Normal Form (1NF)

In the *first normal form*, only single values are permitted at the intersection of each row and column; hence, there are no repeating groups.

To normalize a relation that contains a repeating group, remove the repeating group and form two new relations.

The PK of the new relation is a combination of the PK of the original relation plus an attribute from the newly created relation for unique identification.

Process for 1NF

We will use the Student_Grade_Report table below, from a School database, as our example to explain the process for 1NF.

Student_Grade_Report (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

- In the Student Grade Report table, the repeating group is the course information. A student can take many courses.
- Remove the repeating group. In this case, it's the course information for each student.
- Identify the PK for your new table.

- The PK must uniquely identify the attribute value (StudentNo and CourseNo).
- After removing all the attributes related to the course and student, you are left with the student course table (StudentCourse).
- The Student table (Student) is now in first normal form with the repeating group removed.
- The two new tables are shown below.

Student (StudentNo, StudentName, Major)

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

How to update 1NF anomalies

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

- To add a new course, we need a student.
- When course information needs to be updated, we may have inconsistencies.
- To delete a *student,* we might also delete critical information about a course.

Second Normal Form (2NF)

For the *second normal form*, the relation must first be in 1NF. The relation is automatically in 2NF if, and only if, the PK comprises a single attribute.

If the relation has a composite PK, then each non-key attribute must be fully dependent on the entire PK and not on a subset of the PK (i.e., there must be no partial dependency or augmentation).

Process for 2NF

To move to 2NF, a table must first be in 1NF.

- The Student table is already in 2NF because it has a single-column PK.
- When examining the Student Course table, we see that not all the attributes are fully dependent on the PK; specifically, all course information. The only attribute that is fully dependent is grade.
- Identify the new table that contains the course information.
- Identify the PK for the new table.
- The three new tables are shown below.

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

CourseInstructor (CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)

How to update 2NF anomalies

- When adding a new instructor, we need a course.
- Updating course information could lead to inconsistencies for instructor information.
- Deleting a course may also delete instructor information.

Third Normal Form (3NF)

To be in *third normal form*, the relation must be in second normal form. Also all transitive dependencies must be removed; a non-key attribute may not be functionally dependent on another non-key attribute.

Process for 3NF

- Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have a transitive relationship.
- Create new table(s) with removed dependency.
- Check new table(s) as well as table(s) modified to make sure that each table has a determinant and that no table contains inappropriate dependencies.
- See the four new tables below.

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

Course (CourseNo, CourseName, InstructorNo)

Instructor (InstructorNo, InstructorName, InstructorLocation)

At this stage, there should be no anomalies in third normal form. Let's look at the dependency diagram (Figure 12.1) for this example. The first step is to remove repeating groups, as discussed above.

Student (StudentNo, StudentName, Major)

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

To recap the normalization process for the School database, review the dependencies shown in Figure 12.1.

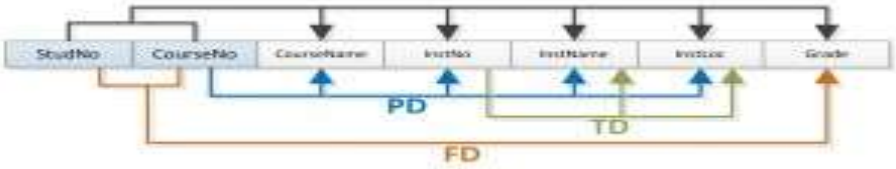


Figure 12.1 Dependency diagram, by A. Watt.

The abbreviations used in Figure 12.1 are as follows:

- PD: partial dependency
- TD: transitive dependency
- FD: full dependency (Note: FD typically stands for functional dependency. Using FD as an abbreviation for full dependency is only used in Figure 12.1.)

Boyce-Codd Normal Form (BCNF)

When a table has more than one candidate key, anomalies may result even though the relation is in 3NF. *Boyce-Codd normal form* is a special case of 3NF. A relation is in BCNF if, and only if, every determinant is a candidate key.

BCNF Example 1

Consider the following table (St_Maj_Adv).

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

The *semantic rules* (business rules applied to the database) for this table are:

1. Each Student may major in several subjects.

2. For each Major, a given Student has only one Advisor.
3. Each Major has several Advisors.
4. Each Advisor advises only one Major.
5. Each Advisor advises several Students in one Major.

The functional dependencies for this table are listed below. The first one is a candidate key; the second is not.

1. Student_id, Major ——> Advisor
2. Advisor ——> Major

Anomalies for this table include:

1. Delete – student deletes advisor info
2. Insert – a new advisor needs a student
3. Update – inconsistencies

Note: No single attribute is a candidate key.

PK can be Student_id, Major or Student_id, Advisor.

To reduce the St_Maj_Adv relation to BCNF, you create two new tables:

1. St_Adv (Student_id, Advisor)
2. Adv_Maj (Advisor, Major)

St_Adv table

Student_id	Advisor
111	Smith
111	Chan
320	Dobbs
671	White

803	Smith

Adv_Maj table

Advisor	Major
Smith	Physics
Chan	Music
Dobbs	Math
White	Physics

BCNF Example 2

Consider the following table (Client_Interview).

ClientNo	InterviewDate	InterviewTime	StaffNo	RoomNo
CR76	13-May-02	10.30	SG5	G101
CR56	13-May-02	12.00	SG5	G101
CR74	13-May-02	12.00	SG37	G102
CR56	1-July-02	10.30	SG5	G102

FD1 – ClientNo, InterviewDate –> InterviewTime, StaffNo, RoomNo (PK)

FD2 – staffNo, interviewDate, interviewTime –> clientNO (candidate key: CK)

FD3 – roomNo, interviewDate, interviewTime –> staffNo, clientNo (CK)

FD4 – staffNo, interviewDate –> roomNo

A relation is in BCNF if, and only if, every determinant is a candidate key. We need to create a table that incorporates the first three FDs (Client_Interview2 table) and another table (StaffRoom table) for the fourth FD.

Client_Interview2 table

ClientNo	InterviewDate	InterViewTime	StaffNo
CR76	13-May-02	10.30	SG5
CR56	13-May-02	12.00	SG5
CR74	13-May-02	12.00	SG37
CR56	1-July-02	10.30	SG5

StaffRoom table

StaffNo	InterviewDate	RoomNo
SG5	13-May-02	G101
SG37	13-May-02	G102
SG5	1-July-02	G102

Normalization and Database Design

During the normalization process of database design, make sure that proposed entities meet required normal form before table structures are created. Many real-world databases have been improperly designed or burdened with anomalies if improperly modified during the course of time. You may be asked to redesign and modify existing databases. This can be a large undertaking if the tables are not properly normalized.

Functional dependency

In a given table, an attribute *Y* is said to have a functional dependency on a set of attributes *X* (written $X \rightarrow Y$) if and only if each *X* value is associated with precisely one *Y* value.

For example, in an "Employee" table that includes the attributes "Employee ID" and "Employee Date of Birth", the functional dependency {Employee ID} →

{Employee Date of Birth} would hold. It follows from the previous two sentences that each {Employee ID} is associated with precisely one {Employee Date of Birth}.

Full functional dependency

An attribute is fully functionally dependent on a set of attributes X if it is:

- functionally dependent on X, and
- not functionally dependent on any proper subset of X. {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a *full* functional dependency, because it is also dependent on {Employee ID}.Even by the removal of {Skill} functional dependency still holds between {Employee Address} and {Employee ID}.

Transitive dependency

A transitive dependency is an indirect functional dependency, one in which $X \rightarrow Z$ only by virtue of $X \rightarrow Y$ and $Y \rightarrow Z$.

Trivial functional dependency

A trivial functional dependency is a functional dependency of an attribute on a superset of itself. {Employee ID, Employee Address} → {Employee Address} is trivial, as is {Employee Address} → {Employee Address}.

Multivalve dependency

A multivalued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows.

Join dependency

A table *T* is subject to a join dependency if *T* can always be recreated by joining multiple tables each having a subset of the attributes of *T*.

Normal form	Defined by	In	Brief definition	Description
1NF	First normal form	Two versions: E.F. Codd (1970), C.J.	1970and 2003	The domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

Normal form	Defined by	In	Brief definition	Description
		Date (2003)		
2NF	Second normal form	E.F. Codd	1971	No non-prime attribute in the table is functionally dependent on a proper subset of any candidate
3NF	Third normal form	Two versions: E.F. Codd (1971), C. Zaniolo (1982)	1971and 1982	Every non-prime attribute is non-transitively dependent on every candidate key in the table. The attributes that do not contribute to the description of the primary key are removed from the table. In other words, no transitive dependency is allowed.
EKNF	Elementary Key Normal Form	C. Zaniolo	1982	Every non-trivial functional dependency in the table is either the dependency of an elementary key attribute or a dependency on a superkey
BCNF	Boyce–Codd normal form	Raymond F. Boyceand E.F. Codd	1974	Every non-trivial functional dependency in the table is a dependency on a superkey
4NF	Fourth normal form	Ronald Fagin	1977	Every non-trivial multivalued dependency in the table is a dependency on a superkey
5NF	Fifth normal form	Ronald Fagin	1979	Every non-trivial join dependency in the table is implied by the superkeys of the table
DKNF	Domain/key normal form	Ronald Fagin	1981	Every constraint on the table is a logical consequence of the table's domain constraints and key constraints
6NF	Sixth	C.J.	2002	Table features no non-trivial join

Normal form	Defined by	In	Brief definition	Description
	normal form	Date, Hugh Darwen, and Nikos Lorentzos		dependencies at all (with reference to generalized join operator)

UNF: A table that contains 1/more repeating gps.

1NF: Each cell must have one value & no repeating gps.

2NF: Every non-primary key Attribute is fully functionaly dependent on primary key. i.e. Remove partially dependency.

3NF: Dependent on primary key. i.e. Remove transitive dependency.

BCNF: Boyce Codd Normal Form: Every determenent is a candidate key.

4NF---->5NF: Higher Normal Form

Modification Anomalies

Once our E-R model has been converted into relations, we may find that some relations are not properly specified. There can be a number of problems:

Deletion Anomaly: Deleting one fact or data point from a relation results in other information being lost.

Insertion Anomaly: Inserting a new fact or tuple into a relation requires we have information from two or more entities – this situation might not be feasible.

Update Anomaly: Updating one fact in a relation requires us to update multiple tuples.

Here is a quick example to illustrate these anomalies: A company has a Purchase Order form:

Normalization Process

- Relations can fall into one or more categories (or classes) called Normal Forms
- Normal Form: A class of relations free from a certain set of modification anomalies.
- Normal forms are given names such as:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)
- Fourth normal form (4NF)
- Fifth normal form (5NF)
- Domain-Key normal form (DK/NF)

- These forms are cumulative. A relation in Third normal form is also in 2NF and 1NF.
- The Normalization Process for a given relation consists of:
  a. Specify the Key of the relation
  b. Specify the functional dependencies of the relation.
     Sample data (tuples) for the relation can assist with this step.
  c. Apply the definition of each normal form (starting with 1NF).
  d. If a relation fails to meet the definition of a normal form, change the relation (most often by splitting the relation into two new relations) until it meets the definition.
  e. Re-test the modified/new relations to ensure they meet the definitions of each normal form.

In the next set of notes, each of the normal forms will be defined along with an example of the normalization steps.

First Normal Form (1NF)

A relation is in first normal form if it meets the definition of a relation:

- Each attribute (column) value must be a single value only.
- All values for a given attribute (column) must be of the same type.
- Each attribute (column) name must be unique.
- The order of attributes (columns) is insignificant
- No two tuples (rows) in a relation can be identical.
- The order of the tuples (rows) is insignificant.
  - If you have a key defined for the relation, then you can meet the unique row requirement.
  - Example relation in 1NF (note that key attributes are underlined):
  - STOCKS (Company, Symbol, Headquarters, Date, Close_Price)

Company	Symbol	Headquarters	Date	Close Price
Microsoft	MSFT	Redmond, WA	09/07/2013	23.96

Microsoft	MSFT	Redmond, WA	09/08/2013	23.93
Microsoft	MSFT	Redmond, WA	09/09/2013	24.01
Oracle	ORCL	Redwood Shores, CA	09/07/2013	24.27
Oracle	ORCL	Redwood Shores, CA	09/08/2013	24.14
Oracle	ORCL	Redwood Shores, CA	09/09/2013	24.33

Second Normal Form (2NF)

A relation is in second normal form (2NF) if all of its non-key attributes are dependent on all of the *key*.

Another way to say this: A relation is in second normal form if it is free from partial-key dependencies

Relations that have a single attribute for a key are automatically in 2NF.

This is one reason why we often use artificial identifiers (non-composite keys) as keys.

In the example below, Close Price is dependent on Company, Date

The following example relation *is not* in 2NF:

STOCKS (Company, Symbol, Headquarters, Date, Close_Price)

Company	Symbol	Headquarters	Date	Close Price
Microsoft	MSFT	Redmond, WA	09/07/2013	23.96
Microsoft	MSFT	Redmond, WA	09/08/2013	23.93
Microsoft	MSFT	Redmond, WA	09/09/2013	24.01
Oracle	ORCL	Redwood Shores, CA	09/07/2013	24.27
Oracle	ORCL	Redwood Shores, CA	09/08/2013	24.14
Oracle	ORCL	Redwood Shores, CA	09/09/2013	24.33

Company	Symbol	Headquarters
Microsoft	MSFT	Redmond, WA
Oracle	ORCL	Redwood Shores, CA

FD1: Symbol → Company, Headquarters

STOCK_PRICES relation:

Symbol	Date	Close Price
MSFT	09/07/2013	23.96
MSFT	09/08/2013	23.93
MSFT	09/09/2013	24.01
ORCL	09/07/2013	24.27
ORCL	09/08/2013	24.14
ORCL	09/09/2013	24.33

FD1: Symbol, Date → Close Price

In checking these new relations we can confirm that they meet the definition of 1NF (each one has well defined unique keys) and 2NF (no partial key dependencies).

Third Normal Form (3NF)

A relation is in third normal form (3NF) if it is in second normal form and it contains no *transitive dependencies*.

Consider relation R containing attributes A, B and C. R(A, B, C)

If A → B and B → C then A → C

Transitive Dependency: Three attributes with the above dependencies.

Example: At CUNY:

Course_Code → Course_Number, Section

Course_Number, Section → Classroom, Professor

Consider one of the new relations we created in the STOCKS example for 2nd normal form:

Company	Symbol	Headquarters
Microsoft	MSFT	Redmond, WA
Oracle	ORCL	Redwood Shores, CA

The functional dependencies we can see are:

FD1: Symbol → Company

FD2: Company → Headquarters

so therefore:
Symbol → Headquarters

This is a transitive dependency.
This gives us the following sample data and FD for the new relations

Company	Symbol
Microsoft	MSFT
Oracle	ORCL

FD1: Symbol → Company

Company	Headquarters
Microsoft	Redmond, WA
Oracle	Redwood Shores, CA

FD1: Company → Headquarters

Again, each of these new relations should be checked to ensure they meet the definition of 1NF, 2NF and now 3NF.

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF if every determinant is a candidate key.
- Recall that not all determinants are keys.
- Those determinants that are keys we initially call *candidate keys*.
- Eventually, we select a single candidate key to be *the key* for the relation.
- Consider the following example:
  - Funds consist of one or more Investment Types.
  - Funds are managed by one or more Managers
  - Investment Types can have one more Managers
  - Managers only manage one type of investment.
- Relation: FUNDS (FundID, InvestmentType, Manager)

FundID	InvestmentType	Manager
99	Common Stock	Smith

99	Municipal Bonds	Jones
33	Common Stock	Green
22	Growth Stocks	Brown
11	Common Stock	Smith

Fourth Normal Form (4NF)

A relation is in fourth normal form if it is in BCNF and it contains no *multivalued dependencies*.

Multivalued Dependency: A type of functional dependency where the determinant can determine more than one value.

More formally, there are 3 criteria:

1. There must be at least 3 attributes in the relation. call them A, B, and C, for example.
2. Given A, one can determine multiple values of B.
3. Given A, one can determine multiple values of C.
4. B and C are independent of one another.

Book example:

Student has one or more majors.

Student participates in one or more activities.

StudentID	Major	Activities
100	CIS	Baseball
100	CIS	Volleyball
100	Accounting	Baseball
100	Accounting	Volleyball
200	Marketing	Swimming

FD1: StudentID →→ Major

FD2: StudentID →→ Activities

Portfolio ID	Stock Fund	Bond Fund
999	Janus Fund	Municipal Bonds
999	Janus Fund	Dreyfus Short-Intermediate Municipal Bond

		Fund
999	Scudder Global Fund	Municipal Bonds
999	Scudder Global Fund	Dreyfus Short-Intermediate Municipal Bond Fund
888	Kaufmann Fund	T. Rowe Price Emerging Markets Bond Fund

A few characteristics:

- No regular functional dependencies
- All three attributes taken together form the key.
- Latter two attributes are independent of one another.
- Insertion anomaly: Cannot add a stock fund without adding a bond fund (NULL Value). Must always maintain the combinations to preserve the meaning.
  - Stock Fund and Bond Fund form a multivalued dependency on Portfolio ID.
  - PortfolioID →→ Stock Fund
  - PortfolioID →→ Bond Fund

Resolution: Split into two tables with the common key:

Portfolio ID	Stock Fund
999	Janus Fund
999	Scudder Global Fund
888	Kaufmann Fund

Portfolio ID	Bond Fund
999	Municipal Bonds
999	Dreyfus Short-Intermediate Municipal Bond Fund
888	T. Rowe Price Emerging Markets Bond Fund

Fifth Normal Form (5NF)

Also called "Projection Join" Normal form.
There are certain conditions under which after decomposing a relation, it cannot be reassembled back into its original form.
We don't consider these issues here.

Domain Key Normal Form (DK/NF)
A relation is in DK/NF if every *constraint* on the relation is a logical consequence of the definition of *keys*and *domains*.
Constraint: An rule governing static values of an attribute such that we can determine if this constraint is True or False. Examples:

- Functional Dependencies
- Multivalued Dependencies
- Inter-relation rules
- Intra-relation rules

However: Does *Not* include time dependent constraints.

Key: Unique identifier of a tuple.

Domain: The physical (data type, size, NULL values) and semantic (logical) description of what values an attribute can hold.
There is no known algorithm for converting a relation directly into DK/NF.

What is Normalization?
Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

Summary of the Normal Forms
The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). In practical applications, you'll often see 1NF, 2NF, and3NF along with the occasional 4NF. Fifth normal form is very rarely seen and won't be discussed in this article.

Before we begin our discussion of the normal forms, it's important to point out that they are guidelines and guidelines only. Occasionally, it becomes necessary to stray from them to meet practical business requirements. However, when variations take place, it's extremely important to evaluate any possible ramifications they could have on your system and account for possible inconsistencies. That said, let's explore the normal forms.

First Normal Form (1NF)

First normal form (1NF) sets the very basic rules for an organized database:

- Eliminate duplicative columns from the same table.
- Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).

Second Normal Form (2NF)

Second normal form (2NF) further addresses the concept of removing duplicative data:

- Meet all the requirements of the first normal form.
- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
- Create relationships between these new tables and their predecessors through the use of foreign keys.

Third Normal Form (3NF)

Third normal form (3NF) goes one large step further:

- Meet all the requirements of the second normal form.
- Remove columns that are not dependent upon the primary key.

Boyce-Codd Normal Form (BCNF or 3.5NF)

The Boyce-Codd Normal Form, also referred to as the "third and half (3.5) normal form", adds one more requirement:

- Meet all the requirements of the third normal form.
- Every determinant must be a candidate key.

Fourth Normal Form (4NF)

Finally, fourth normal form (4NF) has one additional requirement:

- Meet all the requirements of the third normal form.
- A relation is in 4NF if it has no multi-valued dependencies.

Remember, these normalization guidelines are cumulative. For a database to be in 2NF, it must first fulfill all the criteria of a 1NF database.

Unit III

Introduction to Database Languages

The main objective of a database management system is to allow its users to perform a number of operations on the database such as insert, delete, and retrieve data in abstract terms without knowing about the physical representations of data. To provide the various facilities to different types of users, a DBMS normally provides one or more specialized programming languages called Database (or DBMS) Languages.

There are many popular RDBMS available to work. They are as follows:-

- MySQL
- MS SQL Server
- ORACLE
- MS ACCESS

SQL:-

SQL (Structured Query Language) is a database sublanguage for querying and modifying relational databases. It was developed by IBM Research in the mid 70's and standardized by ANSI in 1986.

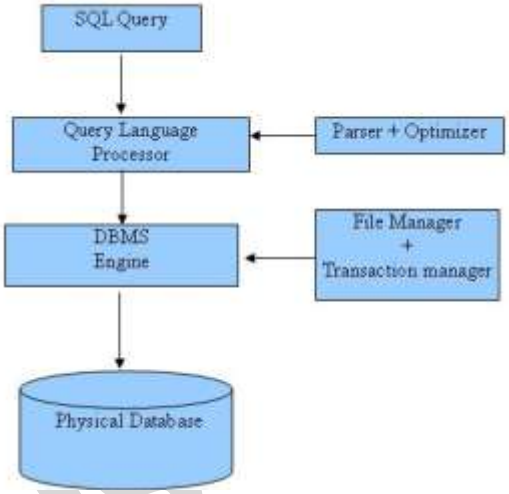
SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database.

SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.

Characteristics of SQL:-

- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows embedding within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures, and views

SQL Process:



SQL Functions:-
SQL has many built-in functions for performing calculations on data.

SQL Aggregate Functions
SQL aggregate functions return a single value, calculated from values in a column.
The Useful aggregate functions are as follows:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

SQL Scalar functions
SQL scalar functions return a single value, based on the input value.
The Useful scalar functions are as follows:

- UCASE() - Converts a field to upper case
- LCASE() - Converts a field to lower case
- MID() - Extract characters from a text field
- LEN() - Returns the length of a text field
- ROUND() - Rounds a numeric field to the number of decimals specified
- NOW() - Returns the current system date and time
- FORMAT() - Formats how a field is to be displayed

Components of SQL:-
SQL commands are instructions used to communicate with the database to perform specific task that work with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users. SQL commands are grouped into four major categories depending on their functionality:

- **Data Definition Language (DDL)** - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.
- **Data Manipulation Language (DML)** - These SQL commands are used for storing, retrieving, modifying, and deleting data. These commands are SELECT, INSERT, UPDATE, and DELETE.
- **Transaction Control Language (TCL)** - These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.
- **Data Control Language (DCL)** - These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

Some of the Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

Some of the Most Important SQL Commands with SQL statement

DML: Data Manipulation Language

SQL-Data Statements -- query and modify tables and columns

- SELECT Statement -- query tables and views in the database
- INSERT Statement -- add rows to tables
- UPDATE Statement -- modify columns in table rows
- DELETE Statement -- remove rows from tables

TCL:- Transaction Control Language

SQL-Transaction Statements -- control transactions

- COMMIT Statement -- commit the current transaction
- ROLLBACK Statement -- roll back the current transaction

DDL:- Data Definition Language

SQL-Schema Statements -- maintain schema (catalog)

- CREATE TABLE Statement -- create tables
- CREATE VIEW Statement -- create views
- DROP TABLE Statement -- drop tables
- DROP VIEW Statement -- drop views

- GRANT Statement -- grant privileges on tables and views to other users
- REVOKE Statement -- revoke privileges on tables and views from other users

The SQL SELECT Statement:-

The SELECT statement is used to select data from a database.
Syntax:
SELECT *column_name,column_name*
FROM *table_name*;
Or
SELECT * FROM *table_name*;

WHERE clause: - It is used to filter records.
SELECT *column_name,column_name*
FROM *table_name*
WHERE *column_name operator value*;

Operators in where clause:-

Operator	Description
=	Equal
<>	Not equal.
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL **AND & OR** Operators:-

- The AND & OR operators are used to filter records based on more than one condition.
- The AND operator displays a record if both the first condition AND the second condition are true.
- The OR operator displays a record if either the first condition OR the second condition is true.

IN

IN operator is used when you know the exact value you want to return for at least one of the columns

SQL ORDER BY Keyword:-

The ORDER BY keyword is used to sort the result-set. The ORDER BY keyword is used to sort the result-set by one or more columns.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

Syntax:-
SELECT *column_name,column_name*
FROM *table_name*
ORDER BY *column_name,column_name* ASC|DESC;

SQL INSERT INTO Statement
The INSERT INTO statement is used to insert new records in a table.
The first form does not specify the column names where the data will be inserted, only their values:

INSERT INTO *table_name*
VALUES (*value1*,*value2*,*value3*,...);
The second form specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*,*column2*,*column3*,...)
VALUES (*value1*,*value2*,*value3*,...);
SQL UPDATE Statement
The UPDATE statement is used to update records in a table.

UPDATE *table_name*
SET *column1=value1*,*column2=value2*,...
WHERE *some_column=some_value*;

SQL DELETE Statement
The DELETE statement is used to delete records in a table.
DELETE FROM *table_name*
WHERE *some_column=some_value*;

SQL CREATE TABLE Statement
The CREATE TABLE statement is used to create a table in a database.
Tables are organized into rows and columns; and each table must have a name.
CREATE TABLE *table_name*
(
column_name1 data_type(*size*),
column_name2 data_type(*size*),
column_name3 data_type(*size*),
....
);

The DROP TABLE Statement

The DROP TABLE statement is used to delete a table.

DROP TABLE table_name;

The ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

To add a column in a table, use the following syntax:

ALTER TABLE table_name

ADD column_name datatype

SQL GRANT Command

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

The Syntax for the GRANT command is:

GRANT privilege_name

ON object_name

TO {user_name |PUBLIC |role_name}

[WITH GRANT OPTION];

- *privilege_name* is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- *object_name* is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- *user_name* is the name of the user to whom an access right is being granted.
- *user_name* is the name of the user to whom an access right is being granted.
- *PUBLIC* is used to grant access rights to all users.
- *ROLES* are a set of privileges grouped together.
- *WITH GRANT OPTION* - allows a user to grant access rights to other users.

SQL REVOKE Command:

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

REVOKE privilege_name

ON object_name

FROM {user_name |PUBLIC |role_name}

The COMMIT Command:

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

Syntax:-

COMMIT;

The ROLLBACK Command:

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for ROLLBACK command is as follows:

ROLLBACK;

SQL Data Types:-

SQL data type is an attribute that specifies type of data of any object. Each column, variable and expression has related data type in SQL.

SQL offers six categories of data types for your use:

- **Exact Numeric Data Types:** int, numeric, bit etc
- **Approximate Numeric Data Types:** Float, real
- **Date and Time Data Types:** Datatime, date, time, smalldatetime
- **Character Strings Data Types:** Char, varchar,varchar(max), text
- **Unicode Character Strings Data Types:** Nchar, nvarchar,ntext
- **Binary Data Types:** Binary,varbinary
- **Misc Data Types**

SQL Operator

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations.

Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

Set operators:-

SQL support few of set operators on the SQL tables. They are as follows:-

- ✓ Union
- ✓ Intersect
- ✓ minus

ORDER BY :-
The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.
Syntax:-
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];

GROUP BY
The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups.
The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
Syntax:
SELECT column1, column2
FROM table_name
WHERE [conditions]
GROUP BY column1, column2
ORDER BY column1, column2

HAVING
The HAVING clause enables you to specify conditions that filter which group results appear in the final results.
The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.
Syntax
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

SELECT column1, column2
FROM table1, table2
WHERE [conditions]
GROUP BY column1, column2
HAVING [conditions]
ORDER BY column1, column2

Subquery

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow:

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery; however, the BETWEEN operator can be used within the subquery.

```
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
    (SELECT column_name [, column_name ]
    FROM table1 [, table2 ]
    [WHERE])
```

Join

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

SQL Join Types:

There are different types of joins available in SQL:

- INNER JOIN: returns rows when there is a match in both tables.
- LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN: returns rows when there is a match in one of the tables.

- SELF JOIN: is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- CARTESIAN JOIN: returns the Cartesian product of the sets of records from the two or more joined tables.

SQL Functions:-

There are two types of functions in SQL

1) **Single Row Functions:** Single row or Scalar functions return a value for every row that is processed in a query.
2) **Group Functions:** These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

There are four types of single row functions. They are:

- **Numeric Functions:** These are functions that accept numeric input and return numeric values.
- **Character or Text Functions:** These are functions that accept character input and can return both character and number values.
-
- **Date Functions:** These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.
- **Conversion Functions:** These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

Numeric Functions:

Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

ABS (x)
CEIL (x)
FLOOR (x)
TRUNC (x, y)
ROUND (x, y)

Character or Text Functions:

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

- LOWER (string_value)
- UPPER (string_value)
- INITCAP (string_value)
- LTRIM (string_value, trim_text)
- RTRIM (string_value, trim_text)
- TRIM (trim_text FROM string_value)
- SUBSTR (string_value, m, n)
- LENGTH (string_value)
- LPAD (string_value, n, pad_value)
- RPAD (string_value, n, pad_value)

Date Functions:

These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Few date functions are as given below.

- ADD_MONTHS (date, n)
- MONTHS_BETWEEN (x1, x2)
- ROUND (x, date_format)
- TRUNC (x, date_format)
- NEXT_DAY (x, week_day)
- LAST_DAY (x)
- SYSDATE
- NEW_TIME (x, zone1, zone2)

Conversion Functions:

These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.

Few of the conversion functions available in SQL are:

- TO_CHAR (x [,y])
- TO_DATE (x [, date_format])
- NVL (x, y)
- DECODE (a, b, c, d, e, default_value)

Unit IV

Extra Knowledge

Larger measures include:
kilobyte (KB) equal to 1,024 bytes
megabyte (MB) equal to 1,024 KB
gigabyte (GB) equal to 1,024 MB
terabyte (TB) equal to 1,024 GB
petabyte (PB) equal to 1,024 TB
exabyte (EB) equal to 1,024 PB

Definition

A data storage device is a piece of technology that is used either to temporarily or permanently hold information in a structured fashion. By structured, we mean that the information is organization in a way that makes it is easy to retrieve after-the-fact.

Types

There are basically two types of data storage devices available. Dynamic storage requires power to maintain the information it stores. An example of this type of store is Random Access Memory (RAM). This is the storage referred to when you talk about your computer's memory.

Serial Access Memory

Sequential access means the system must search the storage device from the beginning of the memory address until it finds the required piece of data. Memory device which supports such access is called a Sequential Access Memory or Serial Access Memory. Magnetic tape is an example of serial access memory.

Direct Access Memory

Direct access memory or Random Access Memory, refers to conditions in which a system can go directly to the information that the user wants. Memory device which supports such access is called a Direct Access Memory. Magnetic disks, optical disks are examples of direct access memory.

Static storage maintains its contents even with the power off. Examples of this type of storage fall into a number of categories. They include:

Magnetic Storage - this technology uses magnetism and small metal pieces embedded in a plastic film to encode information. Examples include cassette tapes, DAT tapes, floppy disks, and hard disks. Don't be surprised if you haven't heard of all of these. Some are likely before your time.

Optical Storage - this technology uses lasers and bumps on a plastic disk to encode information. Examples include CD's, DVD's, and Blu-rays.

Flash Storage - this technology uses electronics, particularly flash memory (memory that is readable and writable, and maintains its contents with the power off), to encode the information. Examples include memory sticks, and solid state (SSD) drives.

Cloud Storage - this technology is just a remote version of the types already mentioned. Examples include Amazon's AWS, and Microsoft's Azure.

Paper Storage - this technology encodes information by punching holes in paper cards. An example of this is punch cards, which have fallen out of use.

Memory

A memory is just like a human brain. It is used to store data and instruction. Computer memory is the storage space in computer where data is to be processed and instructions required for processing are stored.

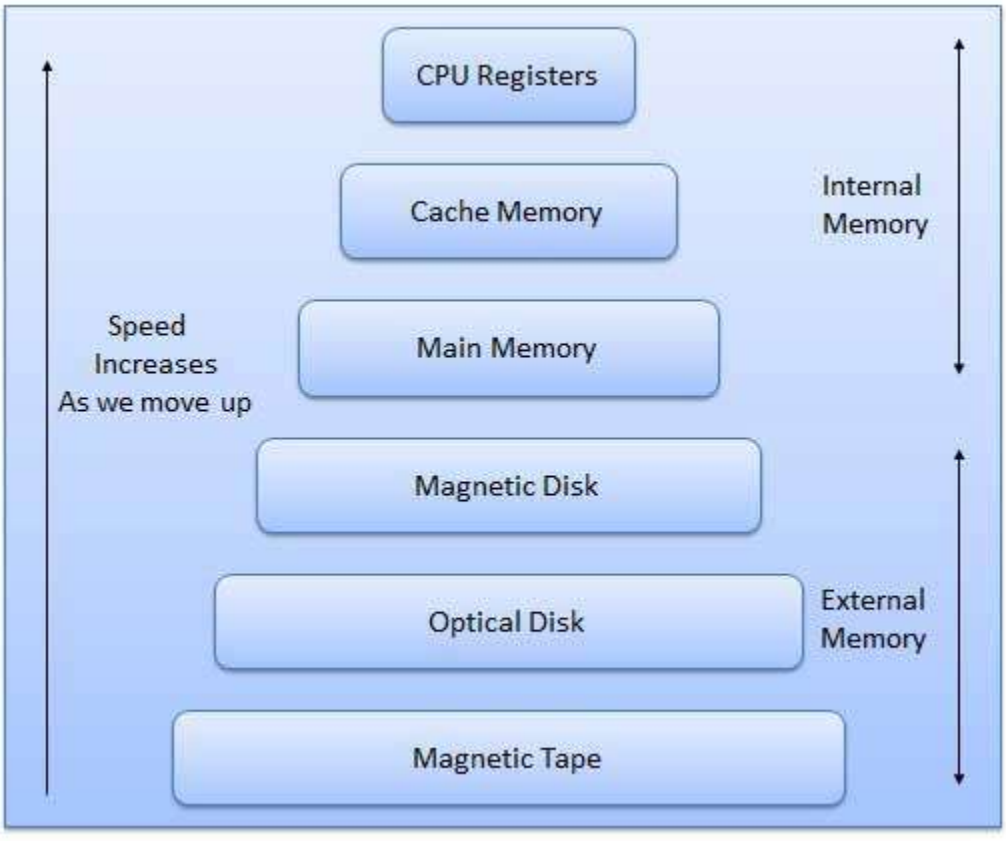
The memory is divided into large number of small parts. Each part is called a cell. Each location or cell has a unique address which varies from zero to memory size minus one.

For example if computer has 64k words, then this memory unit has 64 * 1024 = 65536 memory location. The address of these locations varies from 0 to 65535.

Memory is primarily of two types

Internal Memory – cache memory and primary/main memory

External Memory – magnetic disk / optical disk etc.



Characteristics of Memory Hierarchy are following when we go from top to bottom.

- Capacity in terms of storage increases.
- Cost per bit of storage decreases.
- Frequency of access of the memory by the CPU decreases.
- Access time by the CPU increases.

RAM

A RAM constitutes the internal memory of the CPU for storing data, program and program result. It is read/write memory. It is called random access memory (RAM).

Since access time in RAM is independent of the address to the word that is, each storage location inside the memory is as easy to reach as other location & takes the same amount of time. We can reach into the memory at random & extremely fast but can also be quite expensive.

RAM is volatile, i.e. data stored in it is lost when we switch off the computer or if there is a power failure. Hence, a backup uninterruptible power system (UPS) is often used with computers. RAM is small, both in terms of its physical size and in the amount of data it can hold.

RAM is of two types

Static RAM (SRAM)

Dynamic RAM (DRAM)

Static RAM (SRAM)

The word static indicates that the memory retains its contents as long as power remains applied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not have to be refreshed on a regular basis.

Because of the extra space in the matrix, SRAM uses more chips than DRAM for the same amount of storage space, thus making the manufacturing costs higher.

Static RAM is used as cache memory needs to be very fast and small.

Dynamic RAM (DRAM)

DRAM, unlike SRAM, must be continually refreshed in order for it to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory because it is cheap and small. All DRAMs are made up of memory cells. These cells are composed of one capacitor and one transistor.

ROM

ROM stands for Read Only Memory. The memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacture.

A ROM, stores such instruction as are required to start computer when electricity is first turned on, this operation is referred to as bootstrap. ROM chip are not only used in the computer but also in other electronic items like washing machine and microwave oven.

Following are the various types of ROM –

MROM (Masked ROM)

The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kind of ROMs are known as masked ROMs. It is inexpensive ROM.

PROM (Programmable Read Only Memory)

PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM programmer. Inside the PROM chip there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

EPROM (Erasable and Programmable Read Only Memory)

The EPROM can be erased by exposing it to ultra-violet light for a duration of upto 40 minutes. Usually, an EPROM eraser achieves this function. During programming an electrical charge is trapped in an insulated gate region. The charge is retained for more than ten years because the charge has no leakage path. For erasing this charge, ultra-violet light is passed through a quartz crystal window (lid). This exposure to ultra-violet light dissipates the charge. During normal use the quartz lid is sealed with a sticker.

EEPROM (Electrically Erasable and Programmable Read Only Memory)

The EEPROM is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times. Both erasing and programming take about 4 to 10 ms (millisecond). In EEPROM, any location can be selectively erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip. Hence, the process of re-programming is flexible but slow.

Cache Memory

Cache memory is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory. It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs, are transferred from disk to cache memory by operating system, from where CPU can access them.

Advantages

Cache memory is faster than main memory.
It consumes less access time as compared to main memory.
It stores the program that can be executed within a short period of time.
It stores data for temporary use.

Disadvantages

Cache memory has limited capacity.
It is very expensive.

Virtual memory

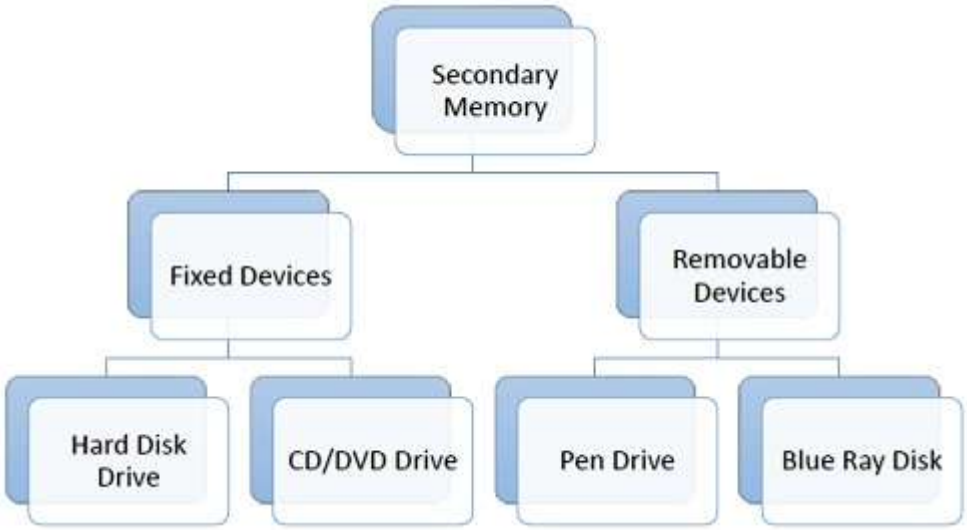
Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that

programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.
This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire program is not required to be loaded fully in main memory.
User written error handling routines are used only when an error occurred in the data or computation.
Certain options and features of a program may be used rarely.
Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
The ability to execute a program that is only partially in memory would counter many benefits.
Less number of I/O would be needed to load or swap each user program into memory.
A program would no longer be constrained by the amount of physical memory that is available.
Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Auxiliary Memory/ Secondary memory.
Auxiliary memory is much larger in size than main memory but is slower. It normally stores system programs, instruction and data files. It is also known as secondary memory. It can also be used as an overflow/virtual memory in case the main memory capacity has been exceeded. Secondary memories cannot be accessed directly by a processor. First the data/information of auxiliary memory is transferred to the main memory and then that information can be accessed by the CPU.
Characteristics of Auxiliary Memory are following –

- Non-volatile memory – Data is not lost when power is cut off.
- Reusable – The data stays in the secondary storage on permanent basis until it is not overwritten or deleted by the user.
- Reliable – Data in secondary storage is safe because of high physical stability of secondary storage device.
- Convenience – With the help of a computer software, authorised people can locate and access the data quickly.
- Capacity – Secondary storage can store large volumes of data in sets of multiple disks.
- Cost – It is much lesser expensive to store data on a tape or disk than primary memory.

Depending on whether secondary memory device is part of CPU or not, there are two types of secondary memory – fixed and removable.



Let us look at some of the secondary memory devices available.

Hard Disk Drive

Hard disk drive is made up of a series of circular disks called platters arranged one over the other almost ½ inches apart around a spindle. Disks are made of non-magnetic material like aluminum alloy and coated with 10-20 nm of magnetic material.

Floppy Disk

They are flexible plastic discs which can bend, coated with magnetic oxide and are covered with a plastic cover to provide protection. Floppy disks are also known as floppies and diskettes.

Memory Card

This has similar functionality to a flash drive but is in a card shape. It can easily plug into a port and removed after its work is done. A memory card is available in various sizes such as 8MB, 16MB, 64MB, 128MB, 256MB etc.

Flash Drive

This is also known as a pen drive. It helps in easy transportation of data from one system to another. A pen drive is quite compact and comes with various features and designs.

CD-ROM

This is short for compact disk - read only memory. A CD is a shiny metal disk of silver colour. It is already pre recorded and the data on it cannot be altered. It usually has a storage capacity of 700 MB.

Various types of Secondary Storage Devices are as Followings:

1) **Magnetic Tapes**: The Magnetic Tapes is the Type of Secondary Storage Device and this Device is used for taking back up of data and this Tape contains some magnetic fields and the Magnetic Tapes are used Accessing the data into the Sequential Form and the Tape Also Contains a Ribbon which is coated on the Single Side of the Tape and also contains a head which reads the data which is Recorded on to the Tape. And when we are reading the information from the disk then we can also read backward information means we can also back the Tape for Reading the Previous information. And For inserting the Tape into the System we also Requires Some Tape Drives Which Contains Tape and which is Responsible for Reading the contents from the Tapes.

They can Store huge Amount of data into the Tape Drive , But the Main Limitation of the Tape Drive is that we cant Access the Data from the Disks directly means if we wants to 100th Record from the Tape then we must have to move all the Previous i.e. 99th Records first. And the Tapes are also easily damaged due to the Human Errors.
2) **Magnetic Disks** : - This is also called as the hard disk and this is made from the thin metal platter which is coated on the both sides of the magnetic Disks. And the there are Many Plates or Platters into a single Hard Disk and all the Plates are Made from the Magnetic Materials and all the Disks are Rotate from the 700 to 3600 rpm means Rotation per Minute and the Hard Disk also Contains a head which is used for both Reading and Writing the Data from the Hard Disks.

The Plate of Disk is Divided into the Tracks and sectors and the collection of Tracks makes a Cylinder means all the Tracks of the Disk which a Consecutive Areas makes a Cylinder.

The Disk is first divided into the Number of Tracks and the Tracks are further divided into the sectors and the Number of Tracks Makes a Cylinder. All the data is Stored into the disk by using Some Sectors and each sectors belongs to a Tracks.
The Data is accessed from the Disk by using the heads, all the heads have Some Arm those are used for Reading the Data from the Particular Tracks and sector. When the Disk Rotates very high Speed then the Head also Moves, For Reading the data from the Disk the ARM touches with the Particular Track and read the data from that Location.
For Locating a Particular data from the Disk the head Moves Around the Disk very Fastly and data which a user wants to Access must have an Address So that Arm of the head just use that Address Means the Number of Cylinder, Number of Track and Number of Sectors from which user wants to read the data.
With the Help of these Read and Write heads we can also Read the Data from the Disk and we can also Stores some data onto the Disk. Some Time Considerations are also used when we are accessing or storing the data onto the hard disk.
Disk Performance factors:

1) **Seek Time:** - The Total Time which is Taken to Move on the Desired track is known as the seek Time. And time is always measured by using the Milliseconds.

2) **Latency Time**. : The time required to Bring the Particular Track to the Desired Location Means the Total Time to bring the Correct the Sector for Reading or for the read and Write head. This is also called as the Average Time.

3) **Data Transfer Time**: The Total Time which is required for Reading and Writing the data into the Disk is known as the Data transfer Time.

When we are Taking About the Magnetic Tapes then we can say that the Storage Capacity of the disk is Measure in the Form of Mega Bytes and when are talking about the Hard Disk then the Measurement will be in the Form of Giga Bytes. Means the Capacity of t the Hard Disk will be Read by using the Giga Bytes. The Magnetic Tapes are Sequential Access Device and the Hard Disk is the Direct Access Device means the data of this Disk will be Read from Any Location and the Data can be Read from the Disk by using the Read Write Heads. But hard Disks are Costlier than the Simple Magnetic Tapes. But the capacity of the Hard Disk is very high in compare to the Tapes.

3) **Floppy Diskette**: Floppy disk is a kind of storage device that can be used to carried around? The Floppy Disk is also a Secondary Storage device which is used for storing the data in a Permanent Manner. The floppy is made up of Rigid Mylar Plastic and also contains a Magnetic black disk inside the Plastic Cover.
The Floppy Disk also Stores all the Data into the Form of Tracks and Sectors and the floppy Disk provides both Reading and Writing the data into the Disk. The Floppy Disk is also called as Reusable Disk means the Floppy Disk Provides us the Facility to Read and Writes the Data into disk as and When Necessary and Also Many Times. We can Read and Write the data from the Disk.
The Main Advantage of the Floppy Disk is that the Data can be Stored many Times but the Main Limitation of the floppy Disk is that floppy Disk have a Small capacity and the Floppy Disk also doesn't have Reliability means the Data Stored into the Disk may not be used for Long Time because the floppy Disk is very Sensitive Thing when we Move the Head of the Disk Again and Again then the floppy disk gets Damaged.
So that we can say that Floppy Disk is not a Reliable thing. And I the Other side the Cost of floppy Disk is also high means with the Comparison of the Other Storage Media's Floppy Disk have some more cost.
But the Main Advantage of the Floppy Disk is that floppy Disk is used for Moving the data from one Computer to Another With the Advent of the Floppy Disk we can Store the Data Into the Floppy Disk and after that we can Easily Remove that Disk from the System and Also Put the Disk into the Another System for Taking the Data.

But we cannot Start or Run the System without the Hard Disk So that floppy Disk is used to Transfer the Files from one System into the. There are Two Types of floppy Disk Available first is the 3.5 and second is the 5.2. But for inserting the Floppy Disk into the System we must have to use the Floppy Disk Drive in the System.
For Reading the data from the Disk there are also Some Read and Write heads those are too used. And the Head will touch the Surface of the floppy Disk So that this will lead to the Damage of the Disk So Quickly because when the Head Directly Touch the Surface of the Disk, then this will lead to the Scratches on the Disk and also cause Damage of the Disk. And the Drive can take only one Disk Means we can insert only one Floppy Disk at a Time into the Floppy Drive. The capacity of the floppy Disk is 1.44 MB. So that we can Floppy Disks as rare as Possible.
Floppy Disk Contains a Notch which Specify Whether the data will be Read or Write Means to Say if we wants to Protect our data then we can set the Notch of the Floppy Disk as a Read Only.

4) **Optical Disks**: The Optical Disks are also called as the CD-ROM's means Compact Disk Read Only Memory which is also used for Storing the data into the Disk and this is called as the Optical Disk because the CD-ROM 's are made up of the Golden or Aluminum Material and the data is Stored on the Disk in the Form of the Tracks and Sectors.
The Whole Disk is Divided into the Number of Tracks and the Single Track is Divided into the Number of Sectors and the Data is Stored into the Sectors and the Disk is Divided into the Sectors as the first Track Contains the Sectors in the huge Size and the Other Tracks contains the Sectors in a Small Manner. So that as the Disk grows the Disk is Divided into the Small Number of Tracks and the Sectors.
CD-ROM Contains the data Which is truly Read able means we cant edit the contents of the CD-ROM Means once Data has been Written into the CD , we can be able to Change the Contents of the Disk and the Data which is Stored on the Disk can be Any Time Read by the user. The CD-ROM provides us the Large Capacity in compare to the Floppy Disks and the CDROM can Store the Data from 650 MB to 800 MB means the data can be Store up to this Space.
There are Many Disks that cant be Erased once Written So they are also called as the WORM Disks Means the Write Once and Read Many Mane a user can just Write the data only one Time and then after that he can use that Disk Many Times but a user cant Edit or Change those Contents after they are Written into the File. So that these Disks are not Reusable. So that these Types of Optical Disks are also called as the CD-ROM and also Some Times they are known as the CD-R Means the Read Only Disks because the data which is Written into these Types of Disks is never to be Erased.
Now these Days there are also Some CDs Available those are also called as the CD-RW or Read Writable Disks. As the Name Suggest these Disks Provides the feature to the user to Read and Write the Contents from the Disk as they feel Necessary So that the CD-RW are

now Most Popular because a user can any Time Remove the Contents from the Disk and also he can store the new Contents into the Disk.

CD Drive

CD stands for Compact Disk. CDs are circular disks that use optical rays, usually lasers, to read and write data. They are very cheap as you can get 700 MB of storage space for less than a dollar. CDs are inserted in CD drives built into CPU cabinet. They are portable as you can eject the drive, remove the CD and carry it with you. There are three types of CDs –

CD-ROM (Compact Disk – Read Only Memory) – The data on these CDs are recorded by the manufacturer. Proprietary Software, audio or video are released on CD-ROMs.

CD-R (Compact Disk – Recordable) – Data can be written by the user once on the CD-R. It cannot be deleted or modified later.

CD-RW (Compact Disk – Rewritable) – Data can be written and deleted on these optical disks again and again.

The CD-R and CD-RW both have Same Capacity and both these can be used for Transferring the Files from one System to another but the Main difference is that the cost. The CD-RW has Some More Cost in compare to the Simple CD or in Compare to the CD-R.

Winchester Disk: Another term for hard disk drive. The term Winchester comes from an early type of disk drive developed by IBM that had 30MB of fixed storage and 30MB of removable storage; so its inventors called it a Winchester in honor of its 30/30 rifle. Although modern disk drives are faster and hold more data, the basic technology is the same, so Winchester has become synonymous with hard.

Magnetic Drum: A direct-access, or *random-access*, storage device. A magnetic drum, also referred to as *drum*, is a metal cylinder coated with magnetic iron-oxide material on which data and programs can be stored. Magnetic drums were once used as a primary storage device but have since been implemented as auxiliary storage devices.

The tracks on a magnetic drum are assigned to channels located around the circumference of the drum, forming adjacent circular bands that wind around the drum. A single drum can have up to 200 tracks. As the drum rotates at a speed of up to 3,000 rpm, the device's read/write heads deposit magnetized spots on the drum during the write operation and sense these spots during a read operation. This action is similar to that of a magnetic tape or disk drive.

Unlike some disk packs, the magnetic drum cannot be physically removed. The drum is permanently mounted in the device. Magnetic drums are able to retrieve data at a quicker rate than tape or disk devices but are not able to store as much data as either of them.

Pen Drive

Pen drive is a portable memory device that uses solid state memory rather than magnetic fields or lasers to record data. It uses a technology similar to RAM, except that it is nonvolatile. It is also called USB drive, key drive or flash memory.

Blu Ray Disk
Blu Ray Disk (BD) is an optical storage media used to store high definition (HD) video and other multimedia filed. BD uses shorter wavelength laser as compared to CD/DVD. This enables writing arm to focus more tightly on the disk and hence pack in more data. BDs can store up to 128 GB data.

File organization
File organization refers to the way data is stored in a file. File organization is very important because it determines the methods of access, efficiency, flexibility and storage devices to use. There are four methods of organizing files on a storage media. This include:

- sequential,
- random,
- serial and
- indexed-sequential

Sequential file organization
Records are stored and accessed in a particular order sorted using a key field.
Retrieval requires searching sequentially through the entire file record by record to the end.
Because the record in a file are sorted in a particular order, better file searching methods like the binary search technique can be used to reduce the time used for searching a file .
Since the records are sorted, it is possible to know in which half of the file a particular record being searched is located, Hence this method repeatedly divides the set of records in the file into two halves and searches only the half on which the records is found.
For example, of the file has records with key fields 20, 30, 40, 50, 60 and the computer is searching for a record with key field 50, it starts at 40 upwards in its search, ignoring the first half of the set.
Advantages of sequential file organization
The sorting makes it easy to access records.
The binary chop technique can be used to reduce record search time by as much as half the time taken.
Disadvantages of sequential file organization

The sorting does not remove the need to access other records as the search looks for particular records.
Sequential records cannot support modern technologies that require fast access to stored records.
The requirement that all records be of the same size is sometimes difficult to enforce.

Random or direct file organization
Records are stored randomly but accessed directly.
To access a file stored randomly, a record key is used to determine where a record is stored on the storage media.
Magnetic and optical disks allow data to be stored and accessed randomly.
Advantages of random file access
Quick retrieval of records.
The records can be of different sizes.

Serial file organization
Records in a file are stored and accessed one after another.
The records are not stored in any way on the storage medium this type of organization is mainly used on magnetic tapes.
Advantages of serial file organization
It is simple
It is cheap
Disadvantages of serial file organization
It is cumbersome to access because you have to access all proceeding records before retrieving the one being searched.
Wastage of space on medium in form of inter-record gap.
It cannot support modern high speed requirements for quick record access.

Indexed-sequential file organization method
Almost similar to sequential method only that, an index is used to enable the computer to locate individual records on the storage media. For example, on a magnetic drum, records are stored sequential on the tracks. However, each record is assigned an index that can be used to access it directly.

Indexed sequential access file combines both sequential file and direct access file organization.
In indexed sequential access file, records are stored randomly on a direct access device such as magnetic disk by a primary key.
This file have multiple keys. These keys can be alphanumeric in which the records are ordered is called primary key.

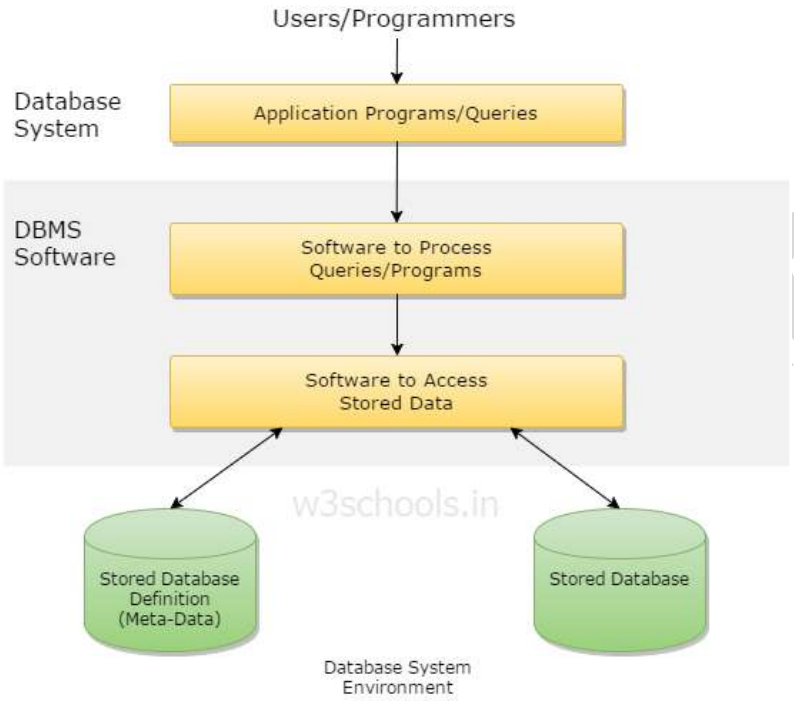
The data can be access either sequentially or randomly using the index. The index is stored in a file and read into memory when the file is opened.

Advantages of Indexed sequential access file organization

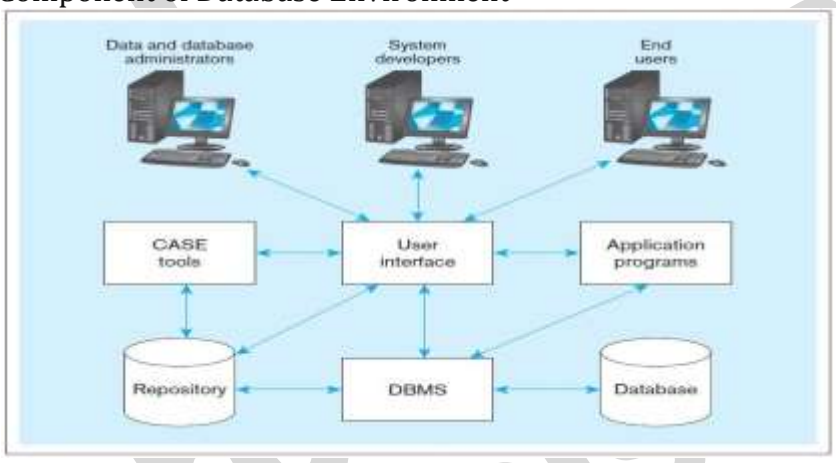
In indexed sequential access file, sequential file and random file access is possible.
It accesses the records very fast if the index table is properly organized.
The records can be inserted in the middle of the file.
It provides quick access for sequential and direct processing.
It reduces the degree of the sequential search.

Disadvantages of Indexed sequential access file organization
Indexed sequential access file takes longer time to search the index for the data access or retrieval.
It requires more storage space.
It is expensive because it requires special software.
It is less efficient in the use of storage space as compared to other file organizations.

Database environment
A database environment is a collective system of components that comprise and regulates the group of data, management, and use of data which consist of software, hardware, people, techniques of handling database and the data also.
Here, the hardware in a database environment means the computers and computer peripherals that are being used to manage a database and the software means the whole thing right from the operating system (OS) to the application programs that includes database management software like M.S. Access or SQL Server. Again the people in a database environment include those people who administrate and use the system. The techniques are the rules, concepts, and instructions given to both the people and the software along with the data with the group of facts and information positioned within the database environment.



Component of Database Environment



Database Administration and control

Before trying to understand the functions of the database administrator, it is necessary to first learn the three different functional levels needed to maintain a database. These levels are the data administration (DA), the database administration (DBA), and database steward.

A **data administration** (also known as a database administration manager, data architect, or information center manager) is a high level function responsible for the overall

management of data resources in an organization. In order to perform its duties, the DA must know a good deal of system analysis and programming.
These are the functions of a data administrator (not to be confused with database administrator functions):
1. Data policies, procedures, standards
2. Planning- development of organization's IT strategy, enterprise model, cost/benefit model, design of database environment, and administration plan.
3. Data conflict (ownership) resolution
4. Data analysis- Define and model data requirements, business rules, operational requirements, and maintain corporate data dictionary
5. Internal marketing of DA concepts
6. Managing the data repository

Database administration is more of an operational or technical level function responsible for physical database design, security enforcement, and database performance. Tasks include maintaining the data dictionary, monitoring performance, and enforcing organizational standards and security.

The functions of a database administrator
1. Selection of hardware and software
- Keep up with current technological trends
- Predict future changes
- Emphasis on established off the shelf products

2. Managing data security and privacy
- Protection of data against accidental or intentional loss, destruction, or misuse
- Firewalls
- Establishment of user privileges
- Complicated by use of distributed systems such as internet access and client/ server technology.

Responsibilities of DBA:

1. Schema Definition:
- The DBA definition the logical Schema of the database.A Schema refers to the overall logical structure of the database.
- According to this schema, database will be developed to store required data for an organization.

2. Storage Structure and Access Method Definition:
- The DBA decides how the data is to be represented in the stored database.

3. Assisting Application Programmers:
- The DBA provides assistance to application programmers to develop application programs.

4. Physical Organization Modification:

- The DBA modifies the physical organization of the database to reflext the changing needs of the organization or to improve performance.

5. Approving Data Access:

- The DBA determines which user needs access to which part of the database.
- According to this,various types of authorizations are granted to different users.

6. Monitoring Performance:

- The DBA monitors performance of the system.The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.

7. Backup and Recovery:

- Database should not be lost or damaged.
- The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.
- In case of failure, such as virus attack database is recovered from this backup.

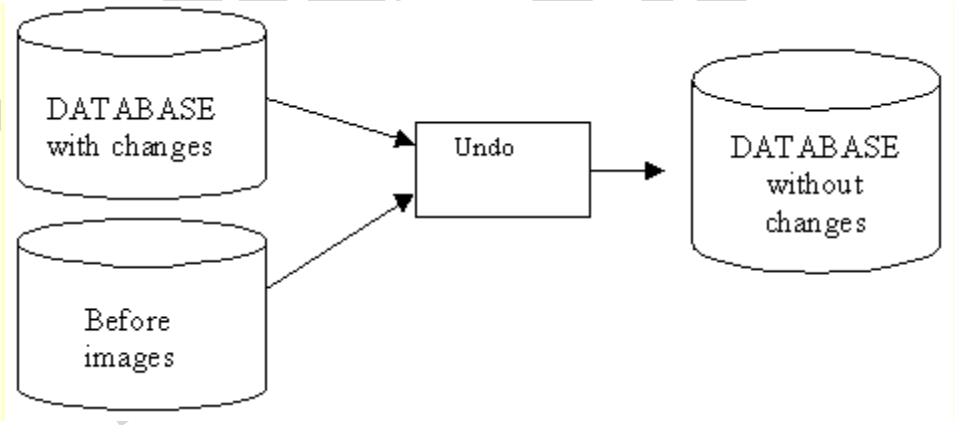
Database – Integrity, Security and Recovery

Recovery Techniques

Several recovery techniques have been proposed for database systems. As we have seen that two types of failures are there, so now we will discuss about how to recover from those two types of failures. Soft failure or Media failure recovery can be done using/restoring the last backup copy or by doing forward recovery if the system logs is intact. While Hard failure or system failure recovery using log include backward recovery as well as forward recovery. So there are two main strategies for performing recovery:

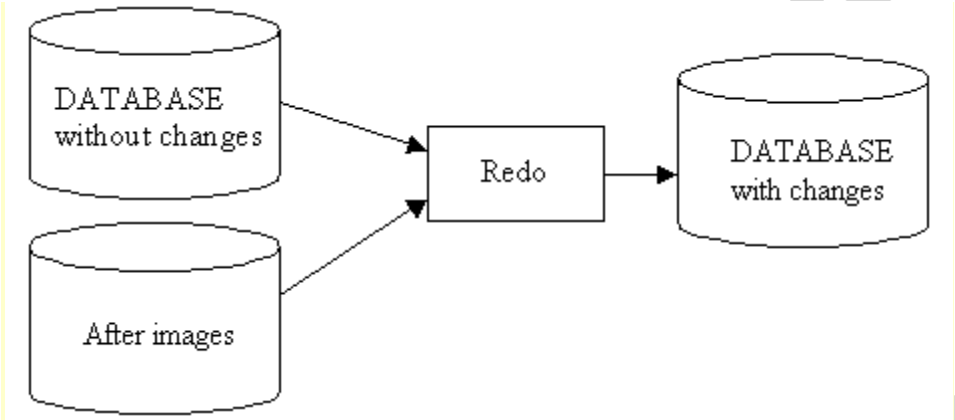
1) Backward Recovery (UNDO)

In this scheme the uncommitted changes made by a transaction to a database are undone. Instead the system is reset to some previous consistent state of database that is free from any errors.



2) Forward Recovery (Redo)

In this scheme the committed changes made by a transaction are reapplied to an earlier copy of the database.



In simpler words, when a particular error in system is detected, the recovery system makes an accurate assessment of the state of the system and then makes appropriate adjustment based on the anticipated results had the system been error free. One thing to be noted that the Redo operation must be idempotent i.e. executing it several times must be equivalent to executing it once. This characteristic is required to guarantee correct behavior of database even if a failure occurs during the recovery process.

Security & Integrity

Information security is the protection of information against unauthorized disclosure, alteration or destruction. Database security is the protection of information that is maintained in a database. It deals with ensuring only the "right people" get the rights access to the "right data". By right people are mean to those people who have the right to access or interact with the database. This ensured that the confidentiality of the data is maintained.

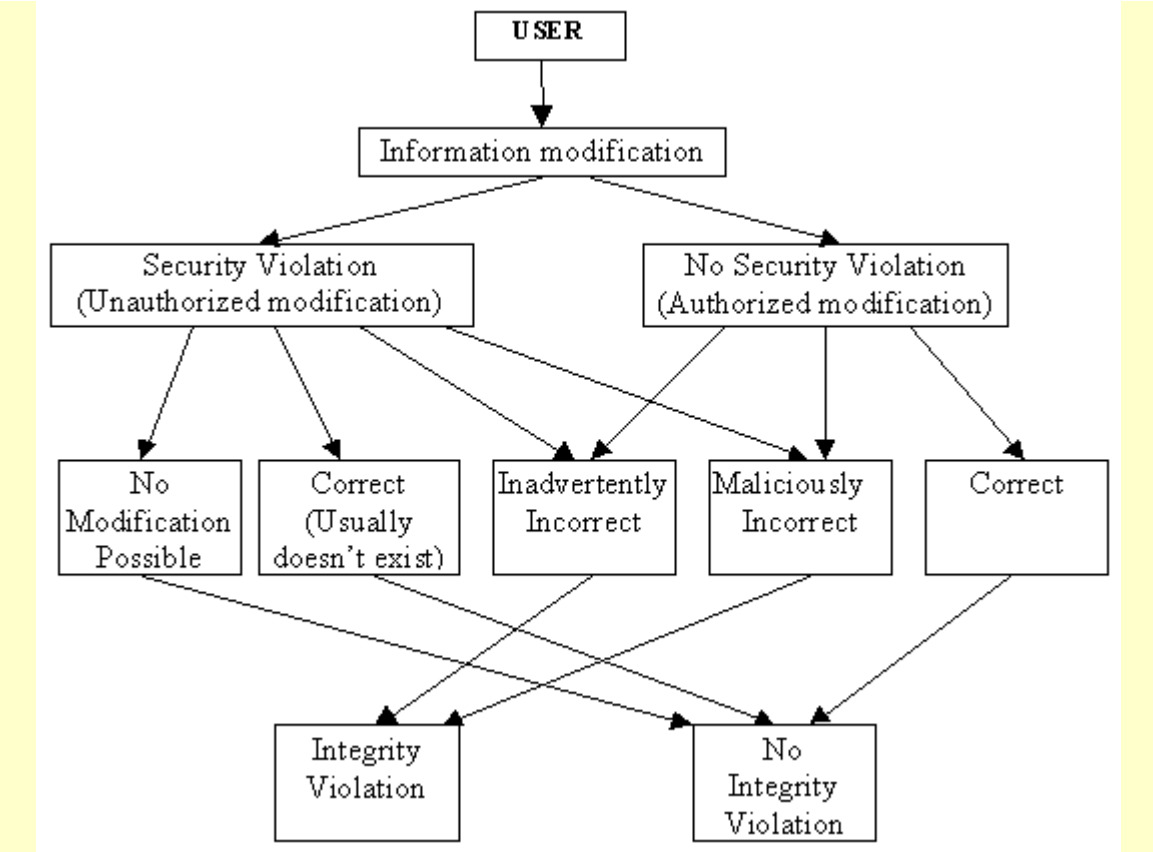
For e.g.: - In an educational instruction, information about student's grade, & university's personal information accessible only to authorities concern & not to everyone. Another example can be in case of medical records of patients in a hospital, these could accessible only to health care officials. In computer definition, specification of access rules about who has what type of access to what information is known as problem of authorization. These access rules are defined at the time database is defined. The person who writes access rules is called on authorizer. The process of ensuring that information & other protected object are accessed only in authorized ways is called access control. The term integrity is also applied to data & to the mechanism that help to ensure its correctness. Integrity refers to the avoidance of accidental loss of consistency. Protection of database contents from

unauthorized access includes legal & ethical issues, organization policies as well as database management policies. To protect database several levels of security measures are maintained: -

1. **Physical :** The site or sites containing the computer system must be physically secured against illegal entry of unauthorized person.
2. **Human :** A template authorization is given to user to reduce chance of any other user giving access to outsides in exchange of some favors.
3. **O.S. :** Even though a fool proof security measures are taken to secure database System, weakness in O.S. security may serve as a means of unauthorized access to the database.
4. **Network :** Since databases allow distributed or remote access through terminals or network, software level security within the network software is an important issue to be taken under consideration.
5. **Database system :** In database also according to user needs authorization is distributed or done. That is to say user may bee allowed to read data & issue queries but would not be allowed to deliberately modify the data. Only some upper level users may be allowed to do so giving them authorized access rights with database itself. It is the responsibility of database system to ensure that these authorization restrictions are not violated.

Relationship between security & integrity

Database security usually refers to access, where as database integrity refers to avoidance of accidental loss of consistency. But generally, the turning point or the dividing line between security & integrity is not always clear. Fig. Below shows relationship between data security & integrity.



Authorization

Authorization is the culmination of the administrative policies of the organization. As name specifies, authorization is a set of rules that can be used to determine which user has what type of access of which portion of the database. The person who writes access rules is called an authorizer.

An authorizer may set several forms of authorization on parts of the database. Among them are the following:

1. **Read Authorization:** allows reading, but not modification of data.
2. **Insert Authorization:** allows insertion of new data, but not the modification of existing data, e.g. insertion of tuple in a relation.
3. **Update authorization:** allows modification of data, but not its deletion. But data items like primary-key attributes may not be modified.
4. **Delete authorization:** allows deletion of data only.

Unit V

SQL:

As defined by Codd (see previous discussion) a relational database must have a common language that is used to create the database, store and manipulate the data within it and manage security.

SQL (often pronounced 'sequel') is the most commonly used language for relational databases. It is used in relational databases such as:

- Oracle (Oracle Corporation)
- DB2 (IBM)
- SQLServer and Access (Microsoft Corporation)
- MySQL (MySQL AB)

You will be using SQL to create tables and manipulate data within those tables. A worldwide standard is defined for SQL, overseen by the ANSI SQL Committee.

SQL Components

SQL consists of three components:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)

The Data Definition Language (DDL). This component of the SQL language is used to create and modify tables and other objects in the database. For tables there are three main commands:

CREATE TABLE tablename to create a table in the database
DROP TABLE tablename to remove a table from the database
ALTER TABLE tablename to add or remove columns from a table in the database

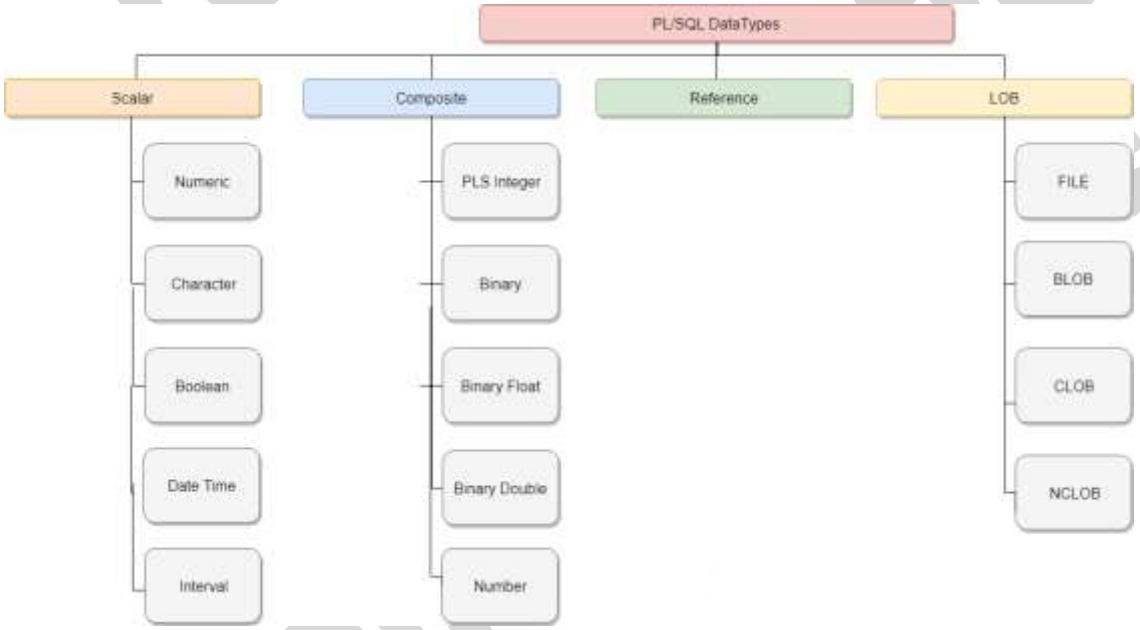
The Data Manipulation Language (DML) component of the SQL language is used to manipulate data within a table. There are four main commands:

SELECT to select rows of data from a table
INSERT to insert rows of data into a table
UPDATE to change rows of data in a table
DELETE to remove rows of data from a table

The Data Control Language (DCL) This component of the SQL language is used to create privileges to allow users access to, and manipulation of, the database. There are two main commands:

GRANT to grant a privilege to a user
REVOKE to revoke (remove) a privilege from a user

Data Types:



DATA TYPE	DATA TYPE	FROM	TO
INTEGER	bigint	-92,23,37,20,36,85,47,70,000	92,23,37,20,36,85,47,70,000
	int	-2,14,74,83,648	2,14,74,83,647
	smallint	-32,768	32,767
	tinyint	0	255
	bit	0	1
	decimal	1E+38	10^38 .1
REAL NUMBERS	float	-1.79E + 308	1.79E + 308
	real	-3.40E + 38	3.40E + 38

Datatypes cover three main areas - **Character, Numeric** and **Date**. The most common types, defined in the SQL ANSI standard are:

- **CHAR(length):** fixed-length character data
- **CHAR VARYING(length)** variable-length character data (length is the maximum number of characters that can be stored). Oracle allows the alternative word VARCHAR in place of CHAR VARYING.
- **DECIMAL (precision, scale):** floating point number. Precision is the number of digits in the number, scale is the number of digits to the right of the decimal point, eg: DECIMAL(7,2) could hold a number up to 99999.99
- **INTEGER:** a whole number. Oracle allows the alternative word NUMBER is place of DECIMAL and INTEGER.
- **DATE:** a valid date (in Oracle this datatype includes the time as well. There is no explicit time datatype).

Create command:

Creating a basic table involves naming the table and defining its columns and each column's data type.

The SQL **CREATE TABLE** statement is used to create a new table.

Syntax

The basic syntax of the CREATE TABLE statement is as follows –

```
CREATE TABLE table_name(
  column1 datatype,
  column2 datatype,
  column3 datatype,
  .....
  columnN datatype,
  PRIMARY KEY( one or more columns )
);
```

DROP TABLE

The syntax of this command is:

DROP TABLE tablename;

This command will remove a table from the database. No confirmation message is given, the table is simply removed. All data is lost and any indexes (see later) on the table are dropped as well. If the table does not exist a warning is given.

INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

Syntax

It is possible to write the INSERT INTO statement in **two ways.**

The **first way** specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

Second Way The INSERT INTO syntax would be as follows:

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

ALTER TABLE *table_name*
ADD *column_name datatype*;

ALTER TABLE - DROP COLUMN

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

ALTER TABLE - ALTER/MODIFY COLUMN

To change the data type of a column in a table, use the following syntax:

ALTER TABLE *table_name*
ALTER COLUMN *column_name datatype*;

Rename Table :

SQL RENAME TABLE syntax is used to change the name of a table. Sometimes, we choose non-meaningful name for the table. So it is required to be changed.

Syntax

ALTER TABLE table_name

RENAME TO new_table_name;

Optionally, you can write following command to rename the table.

RENAME old_table _name To new_table_name;

Drop Table:

A SQL DROP TABLE statement is used to delete a table definition and all data from a table.

Syntax

DROP TABLE "table_name";

Example

DROP TABLE STUDENTS;

Logical operator

- **AND**
- **OR**
- **NOT**

The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

Example

```
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Example

```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

Range Searching

In order to select data that is within a range of values, the BETWEEN operator is used. The BETWEEN operator allows the selection of rows that contain values within a specified lower and upper limit. The range coded after the word BETWEEN is inclusive.

The lower value must be coded first. The two values in between the range must be linked with the keyword AND. A BETWEEN operator can be used with both character and numeric data types. However, one cannot mix the data types that is the lower value of a range of values from a character column and the other from a numeric column.

Example:

Retrieve product_no,description,profit_percent,sell_price from the table product_master where the values contained within the field profit_percent is Between 10 and 20 both inclusive.

SELECT product_no,description,profit_percent,sell_price FROM Product_Master WHERE profit_percent BETWEEN 10 AND 20;

BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

BETWEEN Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* BETWEEN *value1* AND *value2;*

Example

SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;

NOT BETWEEN

Example

SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;

BETWEEN with IN Example

The following SQL statement selects all products with a price BETWEEN 10 and 20. In addition; do not show products with a CategoryID of 1,2, or 3:

Example

```
SELECT * FROM Products
WHERE (Price BETWEEN 10 AND 20)
AND NOT CategoryID IN (1,2,3);
```

IN

IN operator allows you to easily test if the expression matches any value in the list of values. It is used to remove the need of multiple OR condition in SELECT, INSERT, UPDATE or DELETE. You can also use NOT IN to exclude the rows in your list.

Syntax:

```
SELECT column_name(s)

FROM table_name

WHERE column_name IN (list_of_values);
```

Find the Fname, Lname of the Employees who have Salary equal to 30000, 40000 or 25000.

Example:

```
SELECT Fname, Lname

FROM Employee

WHERE Salary IN (30000, 40000, 25000);
```

Pattern Matching

LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

LIKE Syntax

SELECT *column1, column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*;

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of the query.

Alias Column Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Example

```
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;
```

Alias for Tables

```
SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName="Around the Horn" AND c.CustomerID=o.CustomerID;
```

Oracle Built in Functions

There are two types of functions in Oracle.
1) Single Row Functions: Single row or Scalar functions return a value for every row that is processed in a query.
2) Group Functions: These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

There are four types of single row functions. They are:
1) Numeric Functions: These are functions that accept numeric input and return numeric values.
2) Character or Text Functions: These are functions that accept character input and can return both character and number values.
3) Date Functions: These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.
4) Conversion Functions: These are functions that help us to convert a value in one form to

another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

SQL Joins

SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables.

A SQL Join condition is used in the SQL *WHERE Clause* of select, update, delete statements.

Joins in SQL

The **SQL Syntax** for joining two tables is:

```
SELECT col1, col2, col3...
FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1;
```

If a sql join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined. For example, if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10, or 200 rows. This query takes a long time to execute.

SQL Joins Example

Lets use the below two tables to explain the sql join conditions.

Database table "product";

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Vediocon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

Database table "order_items";

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam
5102	103	25	Wipro
5103	101	10	TCS

SQL Joins can be classified into **Equi join and Non Equi join.**

1**) SQL Equi joins**

It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join.

For example: You can get the information about a customer who purchased a product and the quantity of product.

2**) SQL Non equi joins**

It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=

1) SQL Equi Joins:

An equi-join is further classified into two categories:

a) SQL Inner Join

b) SQL Outer Join

a) SQL Inner Join:

All the rows returned by the sql query satisfy the sql join condition specified.

SQL Inner Join Example:

If you want to display the product information for each order the query will be as given below. Since you are retrieving the data from two tables, you need to identify the common column between these two tables, which is the product_id.

The query for this type of sql joins would be like,

```
SELECT order_id, product_name, unit_price, supplier_name,
total_units
FROM product, order_items
WHERE order_items.product_id = product.product_id;
```

The columns must be referenced by the table name in the join condition, because product_id is a column in both the tables and needs a way to be identified. This avoids ambiguity in using the columns in the SQL SELECT statement.

The number of join conditions is (n-1), if there are more than two tables joined in a query where 'n' is the number of tables involved. The rule must be true to avoid Cartesian product.

We can also use aliases to reference the column name, then the above query would be like,

```
SELECT o.order_id, p.product_name, p.unit_price,
p.supplier_name, o.total_units
FROM product p, order_items o
WHERE o.product_id = p.product_id;
```

b) SQL Outer Join:

This sql join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. The sql outer join operator in Oracle is (+) and is used on one side of the join condition only.

The syntax differs for different RDBMS implementation. Few of them represent the join conditions as "sql left outer join", "sql right outer join".

If you want to display all the product data along with order items data, with null values displayed for order items if a product has no order item, the sql query for outer join would be as shown below:

```
SELECT p.product_id, p.product_name, o.order_id,
o.total_units
FROM order_items o, product p
WHERE o.product_id (+) = p.product_id;
```

The output would be like,

product_id	product_name	order_id	total_units
100	Camera		
101	Television	5103	10
102	Refrigerator	5101	5
103	Ipod	5102	25
104	Mobile	5100	30

NOTE: If the (+) operator is used in the left side of the join condition it is equivalent to left outer join. If used on the right side of the join condition it is equivalent to right outer join.

SQL Self Join:

A Self Join is a type of sql join which is used to join a table to itself, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY. It is necessary to ensure that the join statement defines an alias for both copies of the table to avoid column ambiguity.

The below query is an example of a self join,

```
SELECT a.sales_person_id, a.name, a.manager_id,
b.sales_person_id, b.name
```

```
FROM sales_person a, sales_person b
WHERE a.manager_id = b.sales_person_id;
```

2) SQL Non Equi Join:

A Non Equi Join is a SQL Join whose condition is established using all comparison operators except the equal (=) operator. Like >=, <=, <, >

SQL Non Equi Join Example:

If you want to find the names of students who are not studying either Economics, the sql query would be like, (lets use student_details table defined earlier.)

```
SELECT first_name, last_name, subject
FROM student_details
WHERE subject != 'Economics'
```

The output would be something like,

first_name	last_name	subject
Anajali	Bhagwat	Maths
Shekar	Gowda	Maths
Rahul	Sharma	Science
Stephen	Fleming	Science

UNION Operator

In Oracle, UNION operator is used to combine the result sets of two or more Oracle SELECT statements. It combines the both SELECT statement and removes duplicate rows between them./p>

Each SELECT statement within the UNION operator must have the same number of fields in the result sets with similar data types.

Syntax

SELECT expression1, expression2, ... expression_n

FROM table1

WHERE conditions

UNION

SELECT expression1, expression2, ... expression_n

FROM table2

WHERE conditions;

Example

SELECT supplier_id

FROM suppliers

UNION

SELECT supplier_id

FROM order_details

UNION ALL Operator

In Oracle, the UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It is different from UNION operator in a way that it does not remove duplicate rows between the various SELECT statements. It returns all of the rows.

Each SELECT statement within the UNION ALL must have the same number of fields in the result sets with similar data types.

Difference between UNION and UNION ALL operators

UNION operator removes duplicate rows while UNION ALL operator does not remove duplicate rows.

Syntax

SELECT expression1, expression2, ... expression_n

FROM table1

WHERE conditions

UNION ALL

SELECT expression1, expression2, ... expression_n

FROM table2

WHERE conditions;

Example

```
SELECT supplier_id
FROM suppliers
UNION ALL
SELECT supplier_id
FROM order_details;
```

INTERSECT Operator

In Oracle, INTERSECT Operator is used to return the results of 2 or more SELECT statement. It picks the common or intersecting records from compound SELECT queries.

Syntax

SELECT expression1, expression2, ... expression_n
FROM table1
WHERE conditions
INTERSECT
SELECT expression1, expression2, ... expression_n
FROM table2

Example

```
SELECT supplier_id
```

```
FROM suppliers
INTERSECT
SELECT supplier_id
FROM order_details;
```

MINUS operator

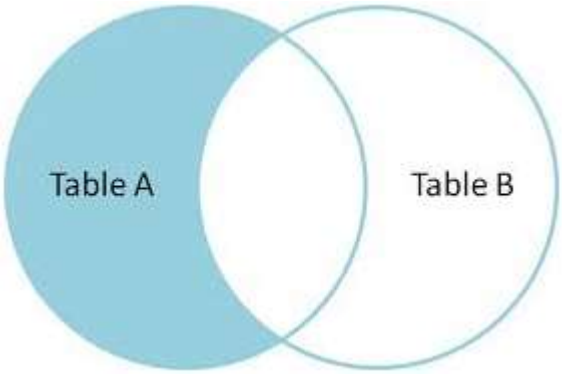
In Oracle, MINUS operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement.

Each SELECT statement has a dataset and the MINUS operator returns all documents from the first dataset and then removes all documents from the second dataset.

For example

Syntax

```
SELECT expression1, expression2, ... expression_n
FROM table1
WHERE conditions
MINUS
SELECT expression1, expression2, ... expression_n
FROM table2
WHERE conditions;
```



Example

```
SELECT supplier_id
FROM suppliers
MINUS
SELECT supplier_id
```

FROM order_details;

Data Constraints

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.
Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

Following are some of the most commonly used constraints available in SQL.

- NOT NULL Constraint – Ensures that a column cannot have NULL value.
- DEFAULT Constraint – Provides a default value for a column when none is specified.
- UNIQUE Constraint – Ensures that all values in a column are different.
- PRIMARY Key – Uniquely identifies each row/record in a database table.
- FOREIGN Key – Uniquely identifies a row/record in any of the given database table.
- CHECK Constraint – The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- INDEX – Used to create and retrieve data from the database very quickly.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use the ALTER TABLE statement to create constraints even after the table is created.

Dropping Constraints

Any constraint that you have defined can be dropped using the ALTER TABLE command with the DROP CONSTRAINT option.
For example, to drop the primary key constraint in the EMPLOYEES table, you can use the following command.
ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;

Some implementations may provide shortcuts for dropping certain constraints. For example, to drop the primary key constraint for a table in Oracle, you can use the following command.
ALTER TABLE EMPLOYEES DROP PRIMARY KEY;

Some implementations allow you to disable constraints. Instead of permanently dropping a constraint from the database, you may want to temporarily disable the constraint and then enable it later.

Integrity Constraints

Integrity constraints are used to ensure accuracy and consistency of the data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in **Referential Integrity (RI)**. These constraints include Primary Key, Foreign Key, Unique Constraints and other constraints which are mentioned above.

Grouping data from table
The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Syntax

The basic syntax of a GROUP BY clause is shown in the following code block. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

```
SELECT column1, column2
FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2
```

Example

Consider the CUSTOMERS table is having the following records –

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

If you want to know the total amount of the salary on each customer, then the GROUP BY query would be as follows.

```
SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS
  GROUP BY NAME;
```

This would produce the following result –

```
+----------+-------------+
| NAME     | SUM(SALARY) |
+----------+-------------+
| Chaitali |     6500.00 |
| Hardik   |     8500.00 |
```

kaushik	2000.00
Khilan	1500.00
Komal	4500.00
Muffy	10000.00
Ramesh	2000.00
+----------+-------------+

Sub queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow –

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

view

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following –

- Structure data in a way that users or classes of users find natural or intuitive.

- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows −

CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];

Example

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS;
```

Sequences

A sequence is a set of integers 1, 2, 3, ... that are generated in order on demand. Sequences are frequently used in databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.

synonym

A **synonym** is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.

You generally use synonyms when you are granting access to an object from another schema and you don't want the users to have to worry about knowing which schema owns the object.

Create Synonym (or Replace)

You may wish to create a synonym so that users do not have to prefix the table name with the schema name when using the table in a query.

Syntax

The syntax to create a synonym in Oracle is:

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema .] synonym_name
 FOR [schema .] object_name [@ dblink];
```

OR REPLACE
Allows you to recreate the synonym (if it already exists) without having to issue a DROP synonym command.

PUBLIC
It means that the synonym is a public synonym and is accessible to all users. Remember though that the user must first have the appropriate privileges to the object to use the synonym.

schema
The appropriate schema. If this phrase is omitted, Oracle assumes that you are referring to your own schema.

object_name
The name of the object for which you are creating the synonym. It can be one of the following:

- table
- view
- sequence
- stored procedure
- function
- package
- materialized view
- java class schema object
- user-defined object
- synonym

Example

```
CREATE PUBLIC SYNONYM suppliers
FOR app.suppliers;
```

The **Data Control Language (DCL)** component of the SQL language is used to create privileges to allow users access to, and manipulation of, the database. There are two main commands:

GRANT to grant a privilege to a user
REVOKE to revoke (remove) a privilege from a user

GRANT command

In order to do anything within an Oracle database you must be given the appropriate privileges. Oracle operates a closed system in that you cannot perform any action at all unless you have been authorised to do so. This includes logging onto the database, creating tables, views, indexes and synonyms, manipulating data (ie select, insert, update and delete) in tables created by other users, etc.

The SQL command to grant a privilege on a table is:

GRANT SELECT, INSERT, UPDATE, DELETE ON tablename TO username;

There are many more forms of the GRANT command, but this is sufficient for this Unit.

Any combination of the above privileges is allowed. You can issue this command on any tables that you have created. For example:

GRANT SELECT ON employee TO hn23;
GRANT SELECT, UPDATE, DELETE ON employee TO hn44;

REVOKE command

The SQL command to revoke a privilege on a table is:

REVOKE SELECT, INSERT, UPDATE, DELETE ON tablename FROM username;

For example:

REVOKE SELECT ON employee FROM hn23;
REVOKE SELECT, UPDATE, DELETE FROM hn44;