

ARBAMINCH UNIVERSITY



FACULTY OF COMPUTING AND SOFTWARE ENGINEERING

Fundamental of database administrator

Tutorials Module

Prepared by

Amin Tuni (Asst Professor)



March, 2023

ARBA MINCH, ETHIOPIA

Objective of the course

Upon successful completion of the course, the student is expected to be able to:

- Describe the basic principles of the relational model of data. In which it includes modeling concepts, notations, operations, and how relations can be declared in a database system.
- Learn how to manipulate relations and specify queries using SQL.
- Delve into an important technique used in logical database design methodology using Entity Relationship and Enhanced Entity Relationship models.
- Demonstrate how to transform a design from one normal form to another form and be able to show the advantages of having a logical database design that conforms to a particular normal form.
- Apply step-by-step methodology for conceptual database design, logical database design, and physical database design for relational systems.
- Know locking mechanisms for controlling the concurrency and also knowing the execution of different types of transactions and their schedules.
- Employ proper technique on how users can access the database at their own site and also access data stored at remote sites.

Chapter One

1. Introduction to Database System

Database systems are designed to manage large data set in an organization. The data management involves both definition and the manipulation of the data which ranges from simple representation of the data to considerations of structures for the storage of information. The data management also considers the provision of mechanisms for the manipulation of information.

Today, Databases are essential to every business.

What is a database? In essence a database is nothing more than a collection of shared information that exists over a long period of time, often many years. In common dialect, the term database refers to a collection of data that is managed by a DBMS.

Data management passes through the different levels of development along with the development in technology and services.

The major three levels are;

1. Manual Approach
2. Traditional File Based Approach
3. Database Approach

Manual Approach

In the manual approach, data storage and retrieval follows the primitive and traditional way of information handling where cards and paper are used for the purpose. The data storage and retrieval will be performed using human labor.

- ✦ Files for as many event and objects as the organization has are used to store information.
- ✦ Each of the files containing various kinds of information is labeled and stored in one ore more cabinets.
- ✦ The cabinets could be kept in safe places for security purpose based on the sensitivity of the information contained in it.
- ✦ Insertion and retrieval is done by searching first for the right cabinet then for the right the file then the information.
- ✦ One could have an indexing system to facilitate access to the data

Limitations of the Manual approach

- ✓ Prone to error.
- ✓ Difficult to update, retrieve, integrate.

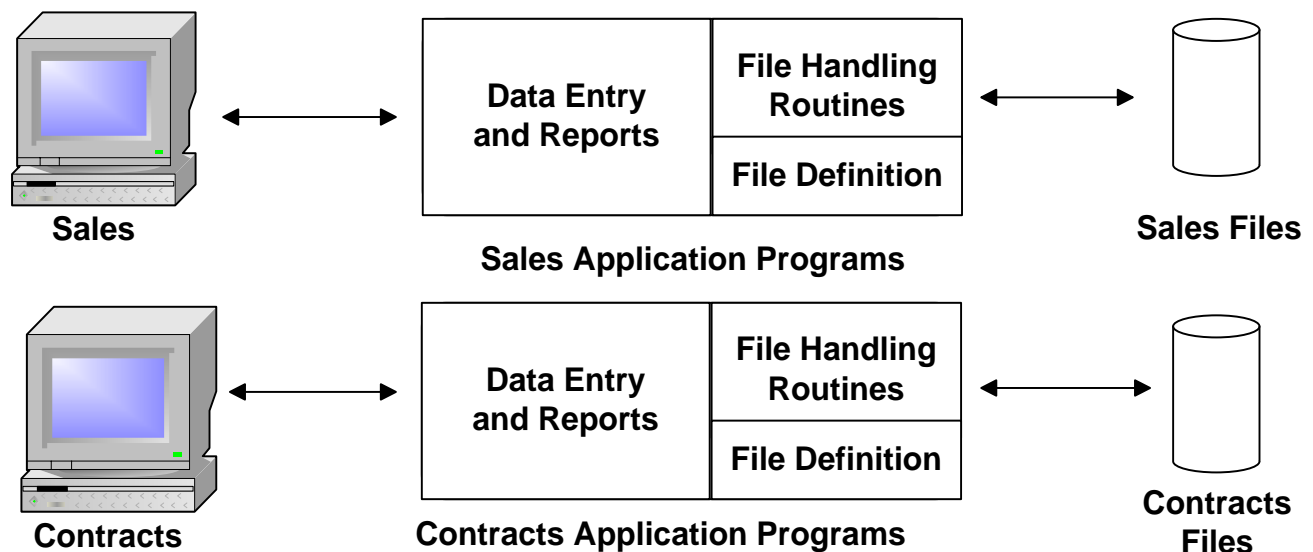
- ✓ You have the data but it is difficult to compile the information.
- ✓ Limited to small size information.
- ✓ Cross referencing is difficult

An alternative approach of data handling is a computerized way of dealing with the information. The computerized approach could also be either decentralized or centralized base on where the data resides in the system.

Traditional File Based Approach

There were, and still are, several computer applications with file based processing used for the purpose of data handling. Even though the approach evolved over time, the basic structure is still similar if not identical.

- ✦ File based systems were an early attempt to computerize the manual filing system.
- ✦ This approach is the decentralized computerized data handling method.
- ✦ A collection of application programs perform services for the end-users. In such systems, every application program that provides service to end users defines and manages its own data.
- ✦ Such systems have number of programs for each of the different applications in the organization.
- ✦ Since every application defines and manages its own data, the system is subjected to serious data duplication problem.
- ✦ File, in traditional file based approach, is a collection of records which contains logically related data.



Limitations of the Traditional File Based approach

As business application become more complex demanding more flexible and reliable data handling methods, the shortcomings of the file based system became evident. These shortcomings include, but not limited to:

- ▲ Separation or Isolation of Data: Available information in one application may not be known.
- ▲ Limited data sharing.
- ▲ Lengthy development and maintenance time.
- ▲ Duplication or redundancy of data.
- ▲ Data dependency on the application.
- ▲ Incompatible file formats between different applications and programs creating inconsistency.
- ▲ Fixed query processing which is defined during application development.

The limitations for the traditional file based data handling approach arise from two basic reasons.

1. Definition of the data is embedded in the application program which makes it difficult to modify the database definition easily.
2. No control over the access and manipulation of the data beyond that imposed by the application programs.

The most significant problem experienced by the traditional file based approach of data handling is the “**update anomalies**”. We have three types of update anomalies;

1. **Modification Anomalies**: a problem experienced when one or more data value is modified on one application program but not on others containing the same data set.
2. **Deletion Anomalies**: a problem encountered where one record set is deleted from one application but remain untouched in other application programs.
3. **Insertion Anomalies**: a problem experienced whenever there is new data item to be recorded, and the recording is not made in all the applications. And when same data item is inserted at different applications, there could be errors in encoding which makes the new data item to be considered as a totally different object.

Database Approach

Queries could be expressed in a very high-level language, which greatly increased the efficiency of database programmers. The database approach emphasizes the integration and sharing of data throughout the organization.

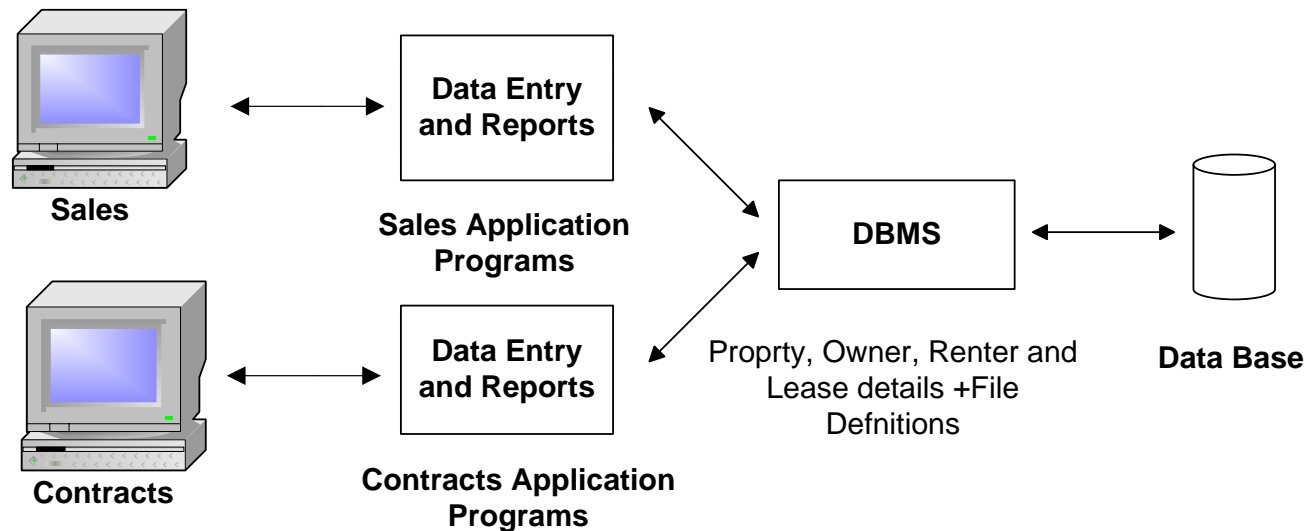
Thus in Database Approach:

- ✓ Database is just a computerized record keeping system or a kind of electronic filing cabinet.

- ✓ Database is a repository for collection of computerized data files.
- ✓ Database is a shared collection of logically related data designed to meet the information needs of an organization. Since it is a shared corporate resource, the database is integrated with minimum amount of or no duplication.
- ✓ Database is a collection of logically related data where these logically related data comprises entities, attributes, relationships, and business rules of an organization's information.

Benefits of the database approach

- ★ *Data can be shared:* two or more users can access and use same data instead of storing data in redundant manner for each user.
- ★ *Improved accessibility of data:* by using structured query languages, the users can easily access data without programming experience.
- ★ *Redundancy can be reduced:* isolated data is integrated in database to decrease the redundant data stored at different applications.
- ★ *Quality data can be maintained:* the different integrity constraints in the database approach will maintain the quality leading to better decision making.
- ★ *Inconsistency can be avoided:* controlled data redundancy will avoid inconsistency of the data in the database to some extent.
- ★ *Transaction support can be provided:* basic demands of any transaction support systems are implanted in a full scale DBMS.
- ★ *Integrity can be maintained:* data at different applications will be integrated together with additional constraints to facilitate shared data resource.
- ★ *Security majors can be enforced:* the shared data can be secured by having different levels of clearance and other data security mechanisms.
- ★ *Improved decision support:* the database will provide information useful for decision making.
- ★ *Standards can be enforced:* the different ways of using and dealing with data by different unite of an organization can be balanced and standardized by using database approach.
- ★ *Compactness:* since it is an electronic data handling method, the data is stored compactly (no voluminous papers).
- ★ *Speed:* data storage and retrieval is fast as it will be using the modern fast computer systems.
- ★ *Less labor:* unlike the other data handling methods, data maintenance will not demand much resource.
- ★ *Centralized information control:* since relevant data in the organization will be stored at one repository, it can be controlled and managed at the central level.



Limitations and risk of Database Approach

- ✓ Introduction of new professional and specialized personnel.
- ✓ Complexity in designing and managing data.
- ✓ The cost and risk during conversion from the old to the new system.
- ✓ High cost to be incurred to develop and maintain the system.
- ✓ Complex backup and recovery services from the user's perspective.
- ✓ Reduced performance due to centralization and data independency.
- ✓ High impact on the system when failure occurs to the central system.

Database Management System (DBMS)

Database Management System (DBMS) is a Software package used for providing EFFICIENT, CONVENIENT and SAFE MULTI-USER (many people/programs accessing same database, or even same data, simultaneously) storage of and access to MASSIVE amounts of PERSISTENT (data outlives programs that operate on it) data.

DBMS and Components of DBMS Environment

A DBMS is software package used to design, manage, and maintain databases. It provides the following facilities:

Data Definition Language (DDL):

- ✦ Language used to define each data element required by the organization.
- ✦ Commands for setting up schema or the intension of database.
- ✦ These commands are used to setup a database, create, delete and alter table with the facility of handling constraints

Data Manipulation Language (DML):

- ✦ Is a core command used by end-users and programmers to store, retrieve, and access the data in the database e.g. SQL.
- ✦ Since the required data or Query by the user will be extracted using this type of language, it is also called "Query Language". 3360

Data Dictionary:

- ✦ Due to the fact that a database is a self describing system, this tool, Data Dictionary, is used to store and organize information about the data stored in the database.

Data Control Language:

- ✦ Database is a shared resource that demands control of data access and usage. The database administrator should have the facility to control the overall operation of the system.
- ✦ Data Control Languages are commands that will help the Database Administrator to control the database.
- ✦ The commands include grant or revoke privileges to access the database or particular object within the database and to store or remove database transactions

DBMS environment has five components

The DBMS environment has five components. To design and use a database, there will be the interaction or integration of Hardware, Software, Data, Procedure and People.

1. **Hardware:** are components that one can touch and feel.
2. **Software:** are collection of commands and programs used to manipulate the hardware to perform a function.
3. **Data:** Operational and Metadata.
4. **Procedure:** this is the rules and regulations on how to design and use a database.
5. **People:** the people that are responsible or play a role in designing, implementing, managing, administering and using the resources in the database.

Roles in Database Design and Use**1. Database Administrator (DBA)**

- ✦ Responsible to oversee, control and manage the database resources (the database itself, the DBMS and other related software).
- ✦ Authorizing access to the database.
- ✦ Coordinating and monitoring the use of the database.

- ▲ Responsible for determining and acquiring hardware and software resources.
- ▲ Accountable for problems like poor security, poor performance of the system.
- ▲ Involves in all steps of database development

We can have further classifications of this role in big organizations having huge amount of data and user requirement.

1. **Data Administrator (DA)**: is responsible on management of data resources. It involves in database planning, development, maintenance of standards policies and procedures at the conceptual and logical design phases.

2. **Database Administrator (DBA)**: is more technically oriented role. He/she is Responsible for the physical realization of the database. It involves in physical design, implementation, security and integrity control of the database.

2. **Database Designer (DBD)**

- ▲ Identifies the data to be stored and choose the appropriate structures to represent and store the data.
- ▲ Should understand the user requirement and should choose how the user views the database.
- ▲ Involve on the design phase before the implementation of the database system.

We have two distinctions of database designers, one involving in the logical and conceptual design and another involving in physical design.

1. Logical and Conceptual DBD

- ▲ Identifies data (entity, attributes and relationship) relevant to the organization.
- ▲ Identifies constraints on each data.
- ▲ Understand data and business rules in the organization.
- ▲ Sees the database independent of any data model at conceptual level and consider one specific data model at logical design phase.

2. Physical DBD

- ▲ Take logical design specification as input and decide how it should be physically realized.
- ▲ Map the logical data model on the specified DBMS with respect to tables and integrity constraints. (DBMS dependent designing).
- ▲ Select specific storage structure and access path to the database.
- ▲ Design security measures required on the database.

3. **Application Programmer and Systems Analyst**

- ▶ System analyst determines the user requirement and how the user wants to view the database.
- ▶ The application programmer implements these specifications as programs; code, test, debug, document and maintain the application program.
- ▶ Determines the interface on how to retrieve, insert, update and delete data in the database.
- ▶ The application could use any high level programming language according to the availability, the facility and the required service.

4. End Users

Workers, whose job requires accessing the database frequently for various purposes, there are different group of users in this category.

1. Naive Users:

- ▶ Sizable proportion of users.
- ▶ Unaware of the DBMS.
- ▶ Only access the database based on their access level and demand.
- ▶ Use standard and pre-specified types of queries.

2. Sophisticated Users

- ▶ Are users familiar with the structure of the Database and facilities of the DBMS?
- ▶ Have complex requirements.
- ▶ Have higher level queries.
- ▶ Are most of the time engineers, scientists, business analysts, etc.

3. Casual Users

- ▶ Users who access the database occasionally.
- ▶ Need different information from the database each time.
- ▶ Use sophisticated database queries to satisfy their needs.
- ▶ Are most of the time middle to high level managers.

Chapter Two

ANSI-SPARC Architecture

The purpose and origin of the Three-Level database architecture

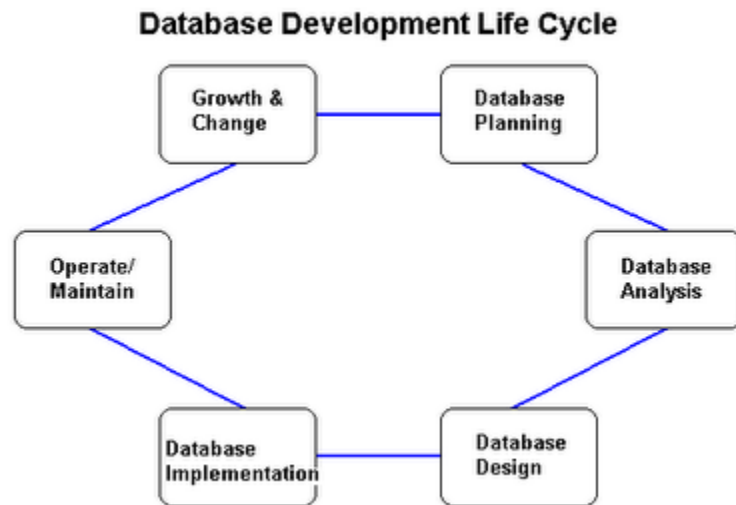
- ▲ All users should be able to access same data. This is important since the database is having a shared data feature where all the data is stored in one location and all users will have their own customized way of interacting with the data.
- ▲ A user's view is unaffected or immune to changes made in other views. Since the requirement of one user is independent of the other, a change made in one user's view should not affect other users.
- ▲ Users should not need to know physical database storage details. As there are naïve users of the system, hardware level or physical details should be a black-box for such users.
- ▲ DBA should be able to change database storage structures without affecting the users' views. A change in file organization, access method should not affect the structure of the data which in turn will have no effect on the users.
- ▲ Internal structure of database should be unaffected by changes to physical aspects of storage.
- ▲ DBA should be able to change conceptual structure of database without affecting all users. In any database system, the DBA will have the privilege to change the structure of the database, like adding tables, adding and deleting an attribute, changing the specification of the objects in the database.

All the above and much other functionality are possible due to the three levels ANSI-SPARC architecture.

Database Development Life Cycle.

The database development process comprises a series of phases. The major phases in information engineering are:

- | | | |
|-------------|-------------------|-------------------|
| 1) Planning | 3) Design | 5) Implementation |
| 2) Analysis | 4) DBMS Selection | 6) Maintenance |



Database planning

The database-planning phase begins when a customer requests to develop a database project. During planning phase, four major activities are performed.

- Review and approve the database project request.
- Prioritize the database project request.
- Allocate resources such as money, people and tools.
- Arrange a development team to develop the database project.

Database planning should also include the development of standards that govern how data will be collected, how the format should be specified, what necessary documentation will be needed.

Requirements Analysis

Requirements analysis is done in order to understand the problem, which is to be solved. It is very important activity for the development of database system. The person responsible for the requirements analysis is often called "Analyst".

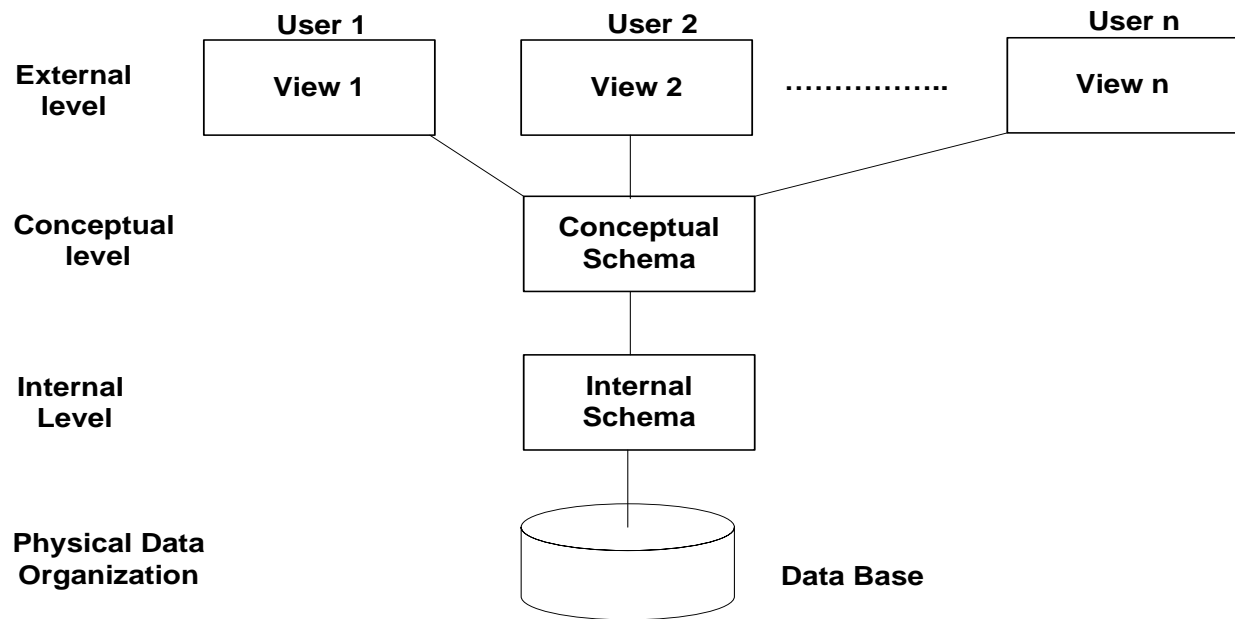
There are two major activities in requirements analysis.

- Problem understanding or analysis
- Requirement specifications.

Design

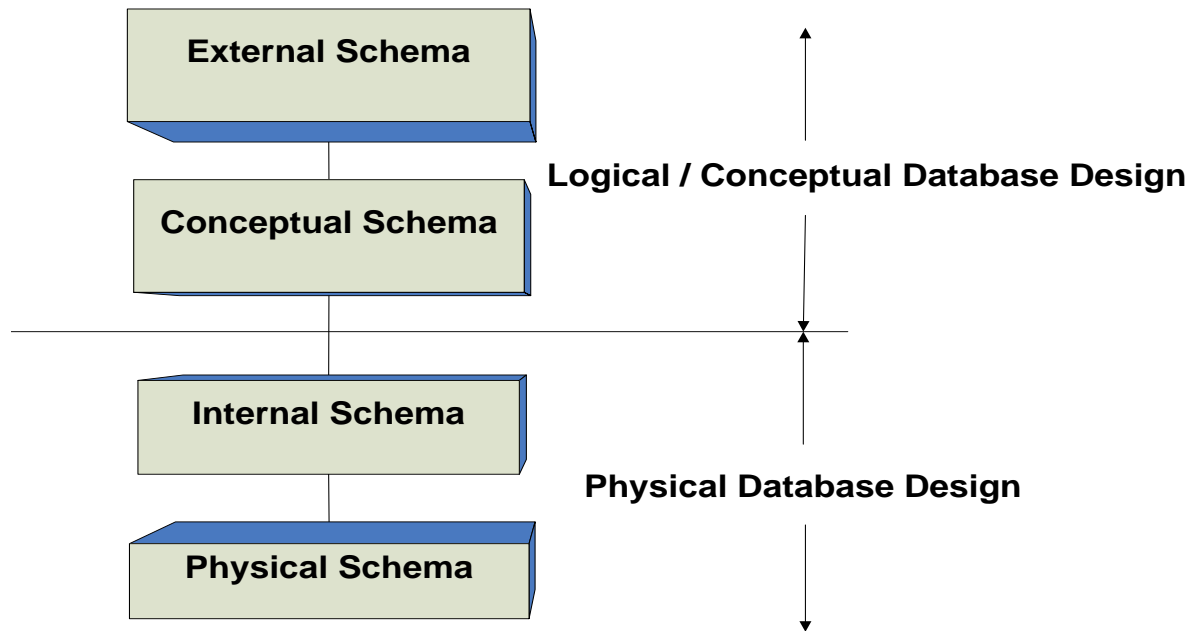
The database design is the major phase of information engineering. In this phase, the information models that were developed during analysis are used to design a conceptual schema for the database and to design transaction and application.

- **In conceptual schema design**, the data requirements collected in Requirement Analysis phase are examined and a conceptual database schema is produced.
- **In transaction and application design**, the database applications analyzed in Requirement Analysis phase are examined and specifications of these applications are produced. There are two major steps in design phase:
 - ♣ Database Design
 - ♣ Process Design



Three-level ANSI-SPARC Architecture of a Database

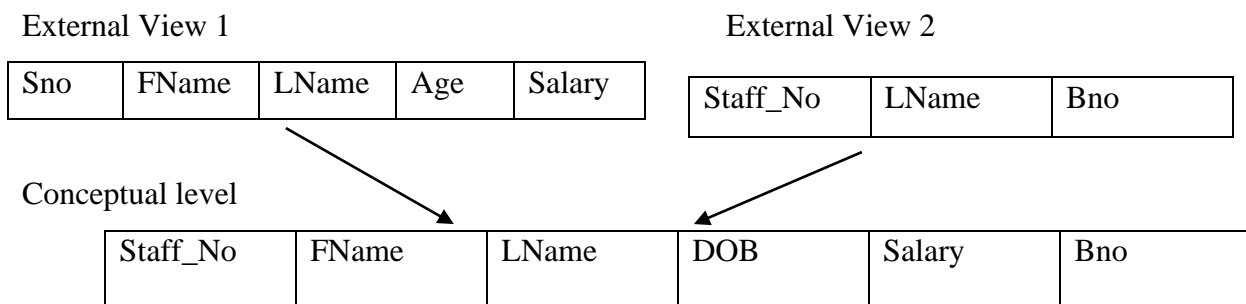
ANSI-SPARC Architecture and Database Design Phases



External Level: Users' view of the database. It describes that part of database that is relevant to a particular user. Different users have their own customized view of the database independent of other users.

Conceptual Level: Community view of the database. It describes what data is stored in database and relationships among the data.

Internal Level: Physical representation of the database on the computer. It describes how the data is stored in the database.



Internal level

```

Struct STAFF
{
    Int Staff_No;
    Int Branch_No;
    Char FName[15];
    Char LName[15];
    Date Date_of_Birth;
    Float Salary;
    Strcut STAFF *next;
};

```

Differences between Three Levels of ANSI-SPARC Architecture

Defines DBMS schemas at three levels:

Internal schema: at the internal level to describe physical storage structures and access paths.
Typically uses a physical data model.

Conceptual schema: at the conceptual level to describe the structure and constraints for the whole database for a community of users. It uses a conceptual or an implementation data model.

External schema: at the external level to describe the various user views. It usually uses the same data model as the conceptual level.

Data Independence

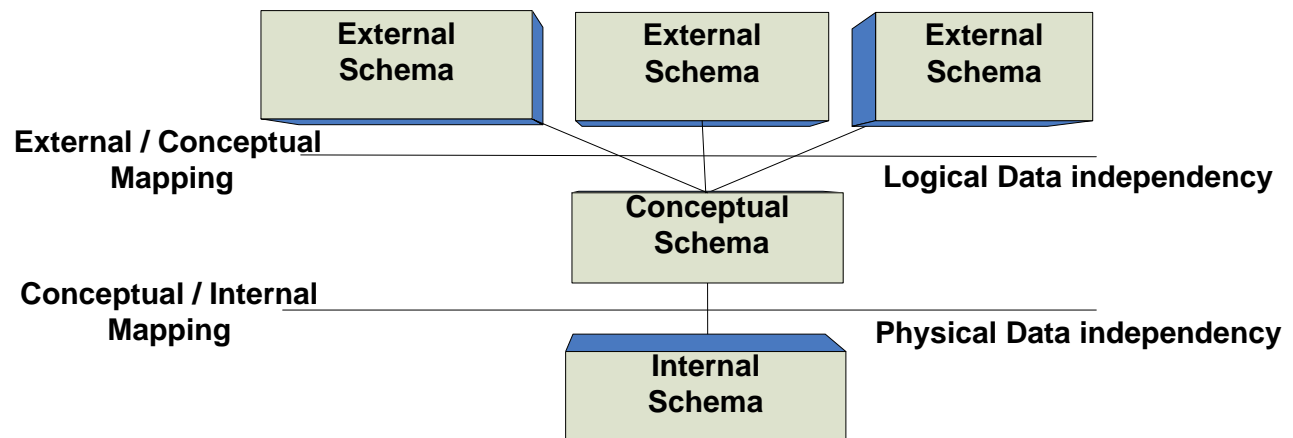
Logical Data Independence:

- Refers to immunity of external schemas to changes in conceptual schema.
- Conceptual schema changes e.g. addition/removal of entities should not require changes to external schema or rewrites of application programs.
- The capacity to change the conceptual schema without having to change the external schemas and their application programs.

Physical Data Independence

- The ability to modify the physical schema without changing the logical schema
- Applications depend on the logical schema.
- In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
- The capacity to change the internal schema without having to change the conceptual schema.

- ★ Refers to immunity of conceptual schema to changes in the internal schema.
- ★ Internal schema changes e.g. using different file organizations, storage structures/devices should not require change to conceptual or external schemas.



Data Independence and the ANSI-SPARC Three-level Architecture

The distinction between a Data Definition Language (DDL) and a Data Manipulation Language (DML)

Database Languages

Data Definition Language (DDL)

- ★ Allows DBA or user to describe and name entities, attributes and relationships required for the application.
- ★ Specification notation for defining the database schema.
- ★ It consists of the SQL commands that can be used to define the database schema. These commands are used for creating, modifying, and dropping the structure of database objects. The database automatically commits the current transaction before and after every DDL Command. Some DDL commands are CREATE, ALTER, and DROP
 - ★ CREATE: It is used to create the database and its objects (like tables, functions, views, indexes, procedures, triggers).

Examples: CREATE DATABASE database_name; This query will create a new database in SQL and name the database as the provided

- ❖ ALTER: Alter command is used to modify the existing database objects.

Data Manipulation Language (DML)

- ★ Provides basic data manipulation operations on data held in the database.
- ★ Language for accessing and manipulating the data organized by the appropriate data model.

- ★ DML also known as query language
 - ★ Procedural DML: user specifies what data is required and how to get the data.
 - ★ Non-Procedural DML: user specifies what data is required but not how it is to be retrieved.

SQL is the most widely used non-procedural language query language

Fourth Generation Language (4GL)

- ★ Query Languages
 - ★ Forms Generators
 - ★ Report Generators
 - ★ Graphics Generators
 - ★ Application Generators
- **DML Commands** deals with the manipulation or managing of data present in the objects of a relational database. These commands are used by end-users and programmers for retrieving, inserting/storing, modifying and deleting the data stored in the database. Some DML commands are SELECT, INSERT, UPDATE, and DELETE commands.

Transaction Control Language (TCL) Commands

TCL commands deal with the transaction within the database. A transaction is a unit of work that is performed against a database. Transaction results in a change in the state of data. TCL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.

- ✓ TCL commands are used with DML Commands only. B/c DDL Commands automatically saves the state of the data.
- ✓ **COMMIT** command: is a TCL command used to save changes invoked by a transaction to the database. It saves all the transactions or changes to the database since the last COMMIT or ROLLBACK command. After COMMIT the changes cannot be undone.

Data control language

DCL SQL commands are used for providing security to database objects (like table, views, procedures, etc.) and allow us to control access to data within the database. They are also used to create objects related to user access and also control the distribution of privileges among users. These commands are GRANT and REVOKE.

- ✚ To perform any operation in the database, such as for creating tables, sequences or views, a user needs privilege. Privileges are of two types:

- 1) **System:** This includes permissions for creating sessions, tables, etc and all types of other system privileges.
 - 2) **Object:** This includes permissions for any command or query to perform any operation on the database tables.
- ❖ **GRANT:** This command is used to give access or permission to specific users or roles on specific database objects like table, view, etc.

Chapter Three

Data Model

Data Model: a set of concepts to describe the structure of a database, and certain constraints that the database should obey.

A **Data model** is a description of the way that data is stored in a database. Data model helps to understand the relationship between entities and to create the most effective structure to hold data.

Data Model is a collection of tools or concepts for describing

- ✓ Data
- ✓ Data semantics
- ✓ Data relationships
- ✓ Data constraints

The main purpose of Data Model is to represent the data in an understandable way.

Categories of data models include:

- ♣ Object-based
- ♣ Record-based
- ♣ Physical

Record-based Data Models

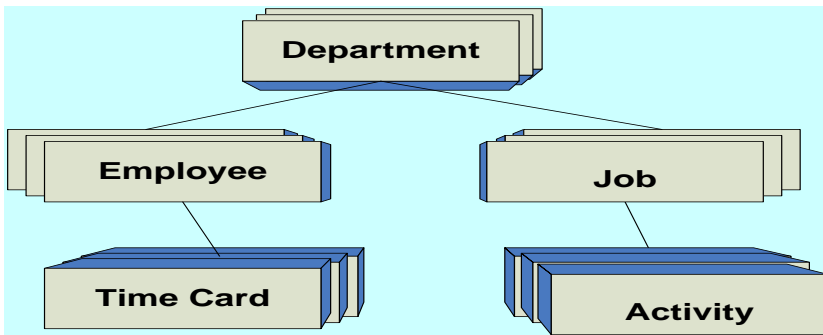
Consist of a number of fixed format records. Each record type defines a fixed number of fields; each field is typically of a fixed length.

- ♣ Hierarchical Data Model
- ♣ Network Data Model
- ♣ Relational Data Model

Hierarchical Model

- ♣ The simplest data model.
- ♣ Record type is referred to as node or segment.
- ♣ The top node is the root node.
- ♣ Nodes are arranged in a hierarchical structure as sort of upside-down tree.
- ♣ A parent node can have more than one child node.
- ♣ A child node can only have one parent node.
- ♣ The relationship between parent and child is one-to-many.
- ♣ Relation is established by creating physical link between stored records (each is stored with a predefined access path to other records).

- ▶ To add new record type or relationship, the database must be redefined and then stored in a new form.



Advantages of Hierarchical Data Model

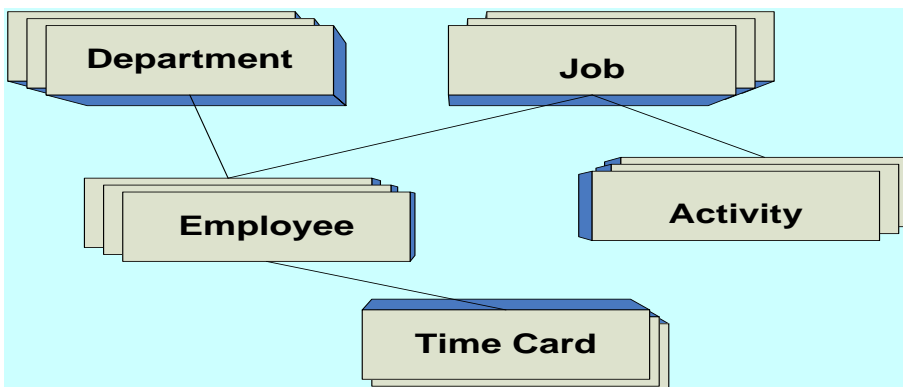
- ▶ Hierarchical Model is simple to construct and operate on.
- ▶ Corresponds to a number of natural hierarchically organized domains e.g., assemblies in manufacturing, personnel organization in companies
- ▶ Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.

Disadvantages of Hierarchical Data Model

- ▶ Navigational and procedural nature of processing
- ▶ Database is visualized as a linear arrangement of records
- ▶ Little scope for "query optimization"

Network Model

- ▶ Allows record types to have more than one parent unlike hierarchical model.
- ▶ A network data model sees records as set members.
- ▶ Each set has an owner and one or more members.
- ▶ Do not allow many to many relationships between entities.
- ▶ Like hierarchical model network model is a collection of physically linked records.
- ▶ Allow member records to have more than one owner.



Advantages of Network Data Model

- Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.
- Can handle most situations for modeling using record types and relationship types.
- Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can do optimal navigation through the database.

Disadvantages of Network Data Model

- Navigational and procedural nature of processing
- Database contains a complex array of pointers that thread through a set of records.
- Little scope for automated "query optimization"

Relational Data Model

- Can define more flexible and complex relationship.
- Viewed as a collection of tables called "Relations" equivalent to collection of record types.
- *Relation* Two dimensional table.
- Stores information or data in the form of table's → rows and columns.
- Records are related by the data stored jointly in the fields of records in two tables or files. The related tables contain information that creates the relation.

Alternative terminologies		
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Chapter Four

Relational Data Model

Properties of Relational Databases

- Each row of a table is uniquely identified by a Primary Key composed of one or more columns.
- Each tuple in a relation must be unique.
- Group of columns that uniquely identifies a row in a table is called a ***Candidate Key***.
- ***Entity Integrity Rule*** of the model states that no component of the primary key may contain a NULL value.
- A column or combination of columns that matches the primary key of another table is called a ***Foreign Key***. Foreign key is used for cross-reference tables.
- The ***Referential Integrity Rule*** of the model states that, for every foreign key value in a table there must be a corresponding primary key value in another table in the database or it should be NULL.
- All tables are logical entities.
- A table is either a base tables (Named Relations) or views (Unnamed Relations).
- Only Base Tables are physically stores.
- It is the collection of tables.
 - ✓ Each entity in one table.
 - ✓ Attributes are fields (columns) in table.

All values in a column represent the same attribute and have the same data format.

Building Blocks of the Relational Data Model

The building blocks of the relational data model are:

- **Entities**: real world physical or logical object.
 - **Attributes**: properties used to describe each Entity or real world object.
 - **Relationship**: the association between Entities.
 - **Constraints**: rules that should be obeyed while manipulating the data.
1. The Entities (persons, places, things etc.) which the organization has to deal with. Relations can also describe relationships.
 - The name given to an entity should always be a singular noun descriptive of each item to be stored in it. E.g.: student NOT students.
 - Every relation has a schema, which describes the columns, or fields the relation itself corresponds to our familiar notion of a table.

- A relation is a collection of *tuples*, each of which contains values for a fixed number of *attributes*.
 - *Existence Dependency*: the dependence of an entity on the existence of one or more entities.
 - *Weak entity* : an entity that can not exist without the entity with which it has a relationship – it is indicated by a double rectangle
2. The Attributes - the items of information which characterize and describe these entities.
- Attributes are pieces of information ABOUT entities. The analysis must of course identify those which are actually relevant to the proposed application. Attributes will give rise to recorded items of data in the database.
 - At this level we need to know such things as:
 - ✓ *Attribute name* (be explanatory words or phrases)
 - ✓ *The domain* from which attribute values are taken (A DOMAIN is a set of values from which attribute values may be taken.) Each attribute has values taken from a domain. For example, the domain of Name is string and that for salary is real.
 - ✓ Whether the attribute is part of the *entity identifier* (attributes which just describe an entity and those which help to identify it uniquely).
 - ✓ Whether it is *permanent or time-varying* (which attributes may change their values over time).
 - ✓ Whether it is *required or optional* for the entity (whose values will sometimes be unknown or irrelevant).

Types of Attributes

- (1) Simple (atomic) Vs Composite attributes
 - ✓ Simple : contains a single value (not divided into sub parts)
E.g. Age, gender
 - ✓ Composite: Divided into sub parts (composed of other attributes)
E.g. Name, address
- (2) Single-valued Vs multi-valued attributes
 - ✓ Single-valued: have only single value (the value may change but has only one value at one time).
E.g. Name, Sex, Id. No. color_of_eyes
 - ✓ Multi-Valued: have more than one value.
E.g. Address, dependent-name

Person may have several college degrees

(3) Stored vs. Derived Attribute

- ✓ Stored: not possible to derive or compute.
E.g. Name, Address
- ✓ Derived: The value may be derived or computed from the values of other attributes.
E.g. Age (current year – year of birth)
Length of employment (current date- starts date)
Profit (earning-cost)
G.P.A (grade point/credit hours)

(4) Null Values

- ✓ NULL applies to attributes which are not applicable or which do not have values.
- ✓ You may enter the value NA (meaning not applicable).
- ✓ Value of a key attribute can not be null.

Default value - assumed value if no explicit value.

Entity versus Attributes

When designing the conceptual specification of the database, one should pay attention to the distinction between an Entity and an Attribute.

- ✓ Consider designing a database of employees for an organization.
 - ✓ Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
 - If we have several addresses per employee, address must be an entity (*attributes cannot be set-valued/multi valued*).
 - ✓ If the structure (city, Woreda, Kebele, etc) is important, e.g. want to retrieve employees in a given city, address must be modelled as an entity (*attribute values are atomic*)
3. The Relationships between entities which exist and must be taken into account when processing information. In any business processing one object may be associated with another object due to some event. Such kind of association is what we call a Relationship between entity objects.
- ✓ One external event or process may affect several related entities.
 - ✓ Related entities require setting of LINKS from one part of the database to another.
 - ✓ A relationship should be named by a word or phrase which explains its function.
 - ✓ Role names are different from the names of entities forming the relationship: one entity may take on many roles; the same role may be played by different entities.

- ✓ For each Relationship, one can talk about the Number of Entities and the Number of Tuples participating in the association. These two concepts are called *Degree* and *Cardinality* of a relationship respectively.

Degree of a Relationship

- An important point about a relationship is how many entities participate in it. The number of entities participating in a relationship is called the Degree of the relationship.
- Among the Degrees of relationship, the following are the basic:
 - ✓ Unary/Recursive Relationship: Tuples/records of a Single entity are related with each other.
 - ✓ Binary Relationships: Tuples/records of two entities are associated in a relationship.
 - ✓ Ternary Relationship: Tuples/records of three different entities are associated.
 - ✓ And a generalized one N-Nary Relationship: Tuples from arbitrary number of entity sets are participating in a relationship.

Cardinality of a Relationship

- Another important concept about relationship is the number of instances/tuples that can be associated with a single instance from one entity in a single relationship. The number of instances participating or associated with a single instance from an entity in a relationship is called the Cardinality of the relationship. The major cardinalities of a relationship are:
 - ✓ *ONE-TO-ONE*: one tuple is associated with only one other tuple.
 - E.g. Building – Location → as a single building will be located in a single location and as a single location will only accommodate a single Building.
 - ✓ *ONE-TO-MANY*, one tuple can be associated with many other tuples, but not the reverse.
 - E.g. Department-Student → as one department can have multiple students.
 - ✓ *MANY-TO-ONE*, many tuples are associated with one tuple but not the reverse.
 - E.g. Employee – Department → as many employees belong to a single department.
 - ✓ *MANY-TO-MANY*: one tuple is associated with many other tuples and from the other side, with a different role name one tuple will be associated with many tuples
 - E.g. Student – Course → as a student can take many courses and a single course can be attended by many students.

4. Relational Constraints/Integrity Rules

➤ Relational Integrity

- ✓ Domain Integrity: No value of the attribute should be beyond the allowable limits.

- ✓ Entity Integrity: In a base relation, no attribute of a Primary Key can assume a value of NULL.
- ✓ Referential Integrity: If a Foreign Key exists in a relation, either the Foreign Key value must match a Candidate Key value in its home relation or the Foreign Key value must be NULL.
- ✓ Enterprise Integrity: Additional rules specified by the users or database administrators of a database are incorporated.

🔑 Key constraints

If tuples are need to be unique in the database, and then we need to make each tuple distinct. To do this we need to have relational keys that uniquely identify each relation.

- ✓ **Super Key:** an attribute or set of attributes that uniquely identifies a tuple within a relation.
- ✓ **Candidate Key:** a super key such that no proper subset of that collection is a Super Key within the relation.

A candidate key has two properties:

1. Uniqueness
2. Irreducibility

If a super key is having only one attribute, it is automatically a Candidate key.

If a candidate key consists of more than one attribute it is called Composite Key.

- ✓ **Primary Key:** the candidate key that is selected to identify tuples uniquely within the relation.

The entire set of attributes in a relation can be considered as a primary case in a worst case.

- ✓ **Foreign Key:** an attribute, or set of attributes, within one relation that matches the candidate key of some relation.

A foreign key is a link between different relations to create the view or the unnamed relation

4.3. Relational Views

Relations are perceived as a Table from the users' perspective. Actually, there are two kinds of relation in relational database. The two categories or tyapes of Relations are Named and Unnamed Relations. The basic difference is on how the relation is created, used and updated:

Base Relation

A *Named Relation* corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.

View (Unnamed Relation)

A View is the dynamic result of one or more relational operations operating on the base relations to produce another virtual relation that does not actually exist as presented. So a view is virtually derived relation that does not necessarily exist in the database but can be produced

upon request by a particular user at the time of request. The virtual table or relation can be created from single or different relations by extracting some attributes and records with or without conditions.

Purpose of a view

- Hides unnecessary information from users: since only part of the base relation (Some collection of attributes, not necessarily all) are to be included in the virtual table.
- Provide powerful flexibility and security: since unnecessary information will be hidden from the user there will be some sort of data security.
- Provide customized view of the database for users: each user is going to be interfaced with their own preferred data set and format by making use of the Views.
- A view of one base relation can be updated.
- Update on views derived from various relations is not allowed since it may violate the integrity of the database.
- Update on view with aggregation and summary is not allowed. Since aggregation and summary results are computed from a base relation and does not exist actually.

Schemas and Instances and Database State

When a database is designed using a Relational data model, all the data is represented in a form of a table. In such definitions and representation, there are two basic components of the database. The two components are the definition of the Relation or the Table and the actual data stored in each table. The data definition is what we call the Schema or the skeleton of the database and the Relations with some information at some point in time is the Instance or the flesh of the database.

Schemas

- ✓ Schema describes how data is to be structured, defined at setup/Design time (also called "metadata").
- ✓ Since it is used during the database development phase, there is rare tendency of changing the schema unless there is a need for system maintenance which demands change to the definition of a relation.
- ✓ *Database Schema (Intension)*: specifies name of relation and the collection of the attributes (specifically the Name of attributes).
 - Refer to a description of database (or intention).
 - Specified during database design.
 - Should not be changed unless during maintenance.

✓ ***Schema Diagrams***

- Convention to display some aspect of a schema visually.

✓ ***Schema Construct***

- Refers to each object in the schema (e.g. STUDENT).
- E.g. STUNEDT (FName, LName, Id, Year, Dept, Sex)

Instances

- ✓ *Instance*: is the collection of data in the database at a particular point of time (snap-shot).
 - Also called *State* or *Snap Shot* or *Extension of the database*.
 - Refers to the actual data in the database at a specific point in time.
 - State of database is changed any time we add, delete or update an item.
 - *Valid state*: the state that satisfies the structure and constraints specified in the schema and is enforced by DBMS.
- ✓ Since Instance is actual data of database at some point in time, changes rapidly.
- ✓ To define a new database, we specify its database schema to the DBMS (database is empty).
- ✓ Database is initialized when we first load it with data.

Chapter Five

Database design

Database design is the process of coming up with different kinds of specification for the data to be stored in the database.

Information System with Database application consists of several tasks which include:

- ✓ Planning of Information systems Design
- ✓ Requirements Analysis,
- ✓ Design (Conceptual, Logical and Physical Design)
- ✓ Testing
- ✓ Implementation
- ✓ Operation and Support

From these different phases, the prime interest of a database system will be the Design part which is again sub divided into other three sub-phases.

These sub-phases are:

- Conceptual Design
- Logical Design, and
- Physical Design
- ✓ In general, one has to go back and forth between these tasks to refine a database design, and decisions in one task can influence the choices in another task.

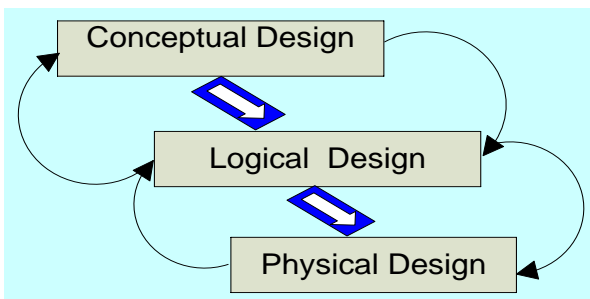


Figure 5.1 The Three levels of Database Design

Conceptual Database Design

- ✓ Conceptual design is the process of constructing a model of the information used in an enterprise, independent of any physical considerations.
 - It is the source of information for the logical design phase.
 - Mostly uses an Entity Relationship Model to describe the data at this level.
- ✓ After the completion of Conceptual Design one has to go for refinement of the schema, which is verification of Entities, Attributes, and Relationships.

Logical Database Design

- ✓ Logical design is the process of constructing a model of the information used in an enterprise based on a specific data model (e.g. relational, hierarchical or network or object), but independent of a particular DBMS and other physical considerations.
 - Normalization process
 - ✓ Collection of Rules to be maintained.
 - ✓ Discover new entities in the process.
 - ✓ Revise attributes based on the rules and the discovered Entities

Physical Database Design

- ✓ Physical design is the process of producing a description of the implementation of the database on secondary storage, defines specific storage or access methods used by database
 - Describe the storage structures and access methods used to achieve efficient access to the data.
 - Tailored to a specific DBMS system, characteristics are function of DBMS and operating systems.
 - Includes estimate of storage space

Conceptual Database Design

- ✓ Conceptual design revolves around discovering and analyzing organizational and user data requirements.
- ✓ The important activities are to identify
 - Entities
 - Attributes
 - Relationships
 - Constraints
- ✓ And based on these components develop the ER model using ER diagrams

The Entity Relationship (E-R) Model

- ✓ Entity-Relationship modelling is used to represent conceptual view of the database.
- ✓ The main components of ER Modelling are:
 - Entities
 - ✓ Corresponds to entire table, not row.
 - ✓ Represented by Rectangle
 - Attributes
 - ✓ Represents the property used to describe an entity or a relationship

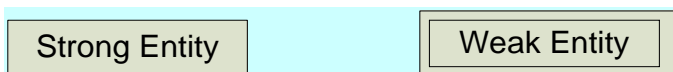
- ✓ Represented by Oval
- Relationships
 - ✓ Represents the association that exist between entities
 - ✓ Represented by Diamond
- Constraints
 - ✓ Represent the constraint in the data

Developing an E-R Diagram

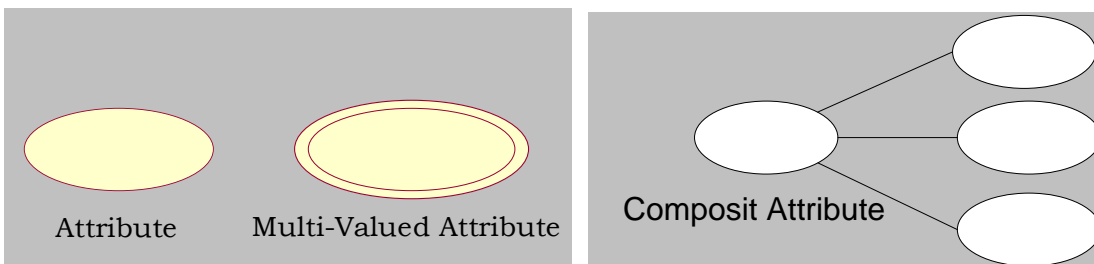
- ✓ Designing conceptual model for the database is not a one linear process but an iterative activity where the design is refined again and again.
- ✓ To identify the entities, attributes, relationships, and constraints on the data, there are different set of methods used during the analysis phase.
- ✓ These include information gathered by...
 - Interviewing end users individually and in a group
 - Questionnaire survey
 - Direct observation
 - Examining different documents
- ✓ The basic E-R model is graphically depicted and presented for review.

Graphical Representations in ER Diagramming

- ✓ Entity is represented by a RECTANGLE containing the name of the entity.



- ✓ Connected entities are called relationship participants.
- ✓ Attributes are represented by OVALS and are connected to the entity by a line.



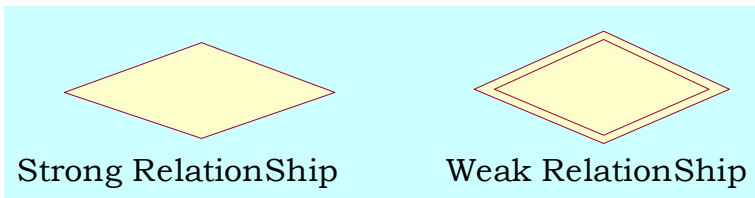
- ✓ A derived attribute is indicated by a DOTTED LINE. (.....)



- ✓ PRIMARY KEYS are underlined.

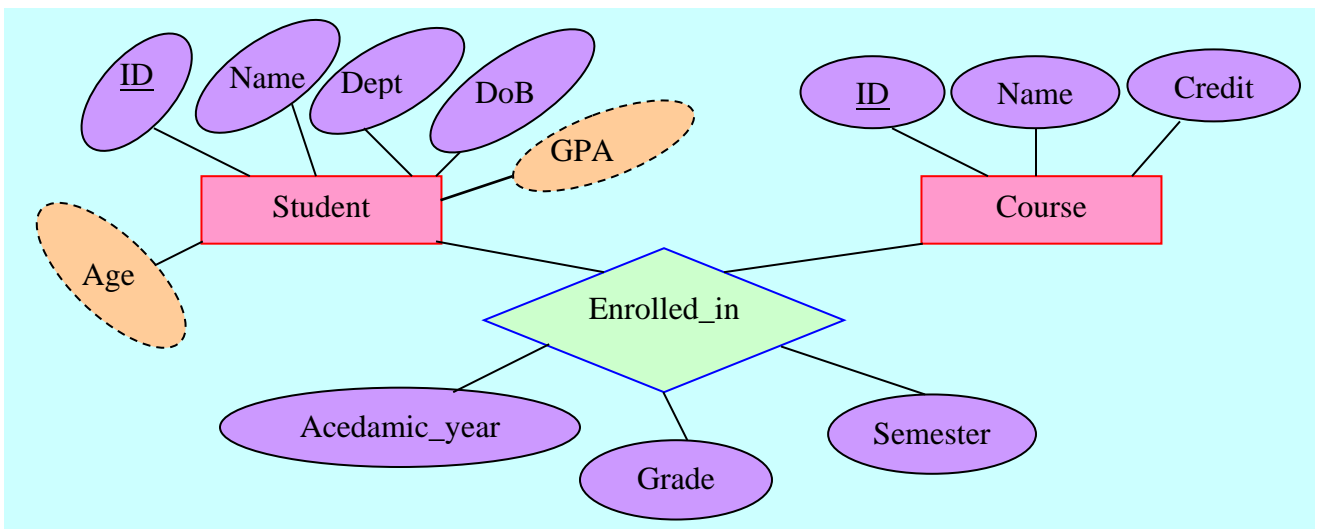
Key

- ✓ Relationships are represented by DIAMOND shaped symbols
 - Weak Relationship is a relationship between Weak and Strong Entities.
 - Strong Relationship is a relationship between two strong Entities



Example 1: Build an ER Diagram for the following information:

- ✓ A student record management system will have the following two basic data object categories with their own features or properties. Students will have an Id, Name, Dept, Age, GPA and Course will have an Id, Name, Credit Hours
 - Whenever a student enroll in a course in a specific Academic Year and Semester, the Student will have a grade for the course



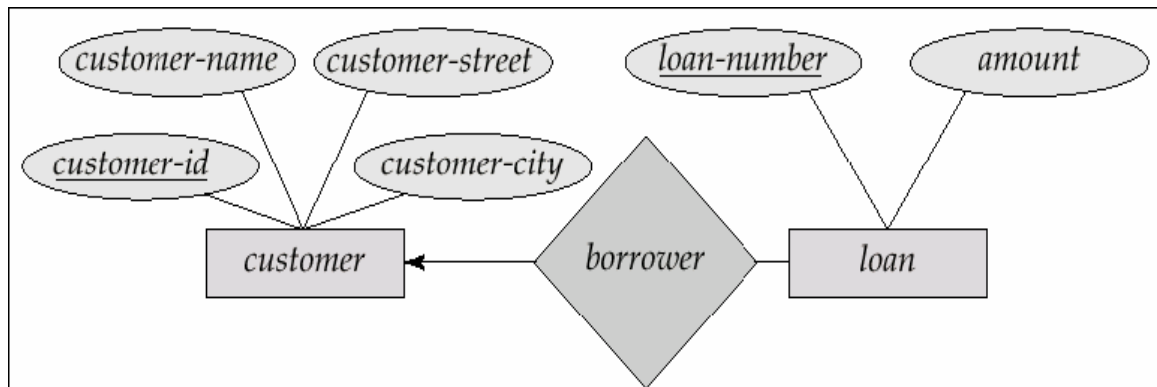
Example 2: Build an ER Diagram for the following information:

Structural Constraints on Relationship

1. Constraints on Relationship / Multiplicity/ Cardinality Constraints
 - ✓ Multiplicity constraint is the number or range of possible occurrence of an entity type/relation that may relate to a single occurrence/tuple of an entity type/relation through a particular relationship.
 - ✓ Mostly used to insure appropriate enterprise constraints.

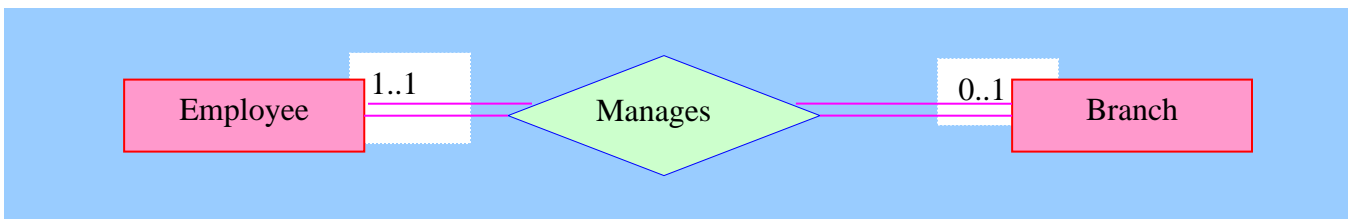
One-to-one relationship

- ✓ A customer is associated with at most one loan via the relationship borrower
- ✓ A loan is associated with at most one customer via borrower



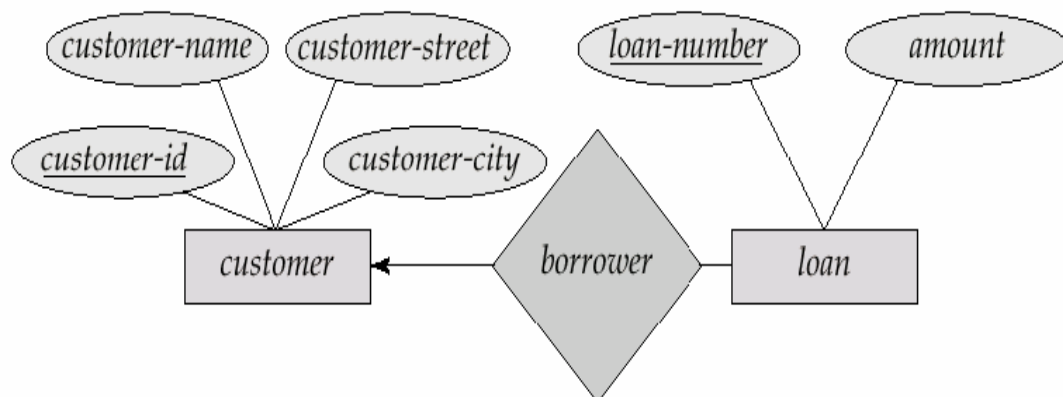
E.g. Relationship Manages between Staff and Branch

- ✓ The multiplicity of the relationship is:
 - One branch can only have one manager.
 - One Employee could Manages either one or no branches.



One-To-Many Relationships

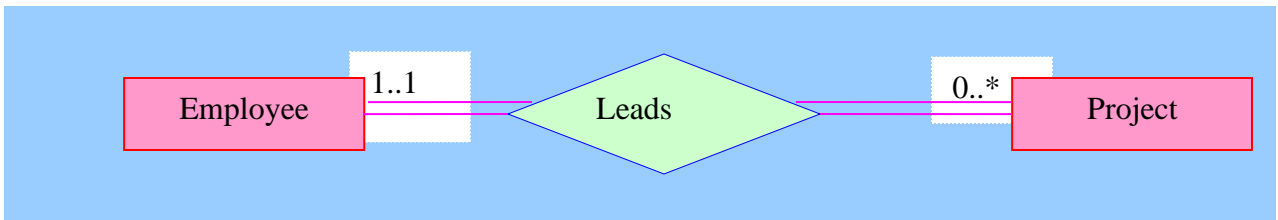
- ✓ In the one-to-many relationship a loan is associated with at most one customer via borrower, a customer is associated with several (including 0) loans via borrower



E.g. Re

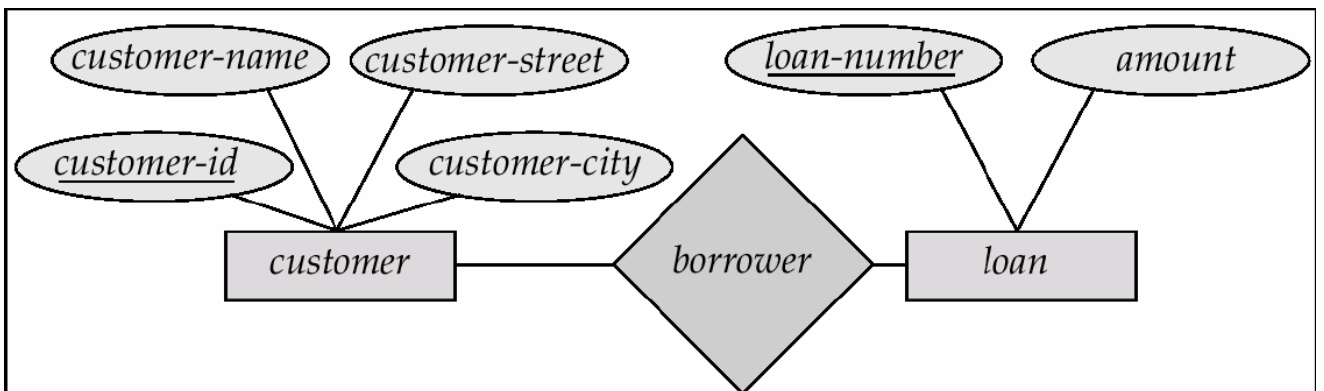
- ✓ The

- One project is leaf by one staff.



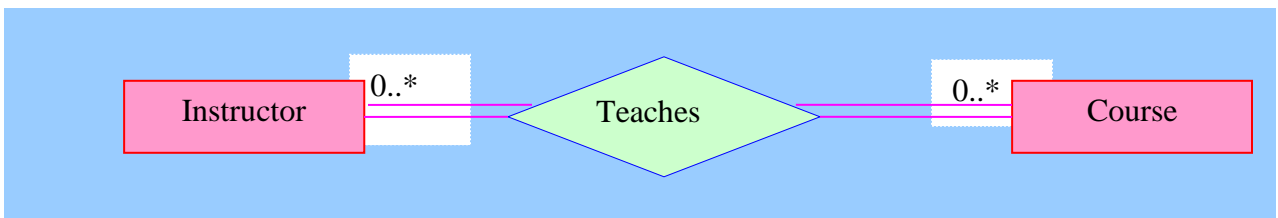
Many-To-Many Relationship

- ✓ A customer is associated with several (possibly 0) loans via borrower.
- ✓ A loan is associated with several (possibly 0) customers via borrower.



E.g.: Relationship Teaches between Instructors and Course.

- ✓ The multiplicity of the relationship
 - One Instructor teaches one or more Course(s).
 - One course taught by zero or more Instructor(s).



Logical Database Design

Logical design is the process of constructing a model of the information used in an enterprise based on a specific data model (e.g. relational, hierarchical or network or object), but independent of a particular DBMS and other physical considerations.

Normalization process

- Collection of Rules to be maintained.
- Discover new entities in the process.
- Revise attributes based on the rules and the discovered Entities

The first step before applying the rules in relational data model is converting the conceptual design to a form suitable for relational logical model, which is in a form of tables.

Converting ER Diagram to Relational Tables

Three basic rules to convert ER into tables or relations:

1. For a relationship with One-to-One Cardinality:
 - All the attributes are merged into a single table. Which means one can post the primary key or candidate key of one of the relations to the other as a foreign key.
2. For a relationship with One-to-Many Cardinality:
 - Post the primary key or candidate key from the “one” side as a foreign key attribute to the “many” side. E.g.: For a relationship called “Belongs To” between Employee (Many) and Department (One)
3. For a relationship with Many-to-Many Cardinality:
 - Create a new table (which is the associative entity) and post primary key or candidate key from each entity as attributes in the new table along with some additional attributes (if applicable)

After converting the ER diagram in to table forms, the next phase is implementing the process of normalization, which is a collection of rules each table should satisfy.

Normalization

Normalization is the process of identifying the logical associations between data items and designing a database that will represent such associations but without suffering the update anomalies which are;

1. Insertion Anomalies
2. Deletion Anomalies
3. Modification Anomalies

Normalization may reduce system performance since data will be cross referenced from many tables. Thus de-normalization is sometimes used to improve performance, at the cost of reduced consistency guarantees.

Normalization normally is considered as good if it is lossless decomposition.

Insertion anomalies

An "insertion anomaly" is a failure to place information about a new database entry into all the places in the database where information about that new entry needs to be stored. In a properly normalized database, information about a new entry needs to be inserted into only one place in the database; in an inadequately normalized database, information about a new entry may need to be inserted into more than one place and, human fallibility being what it is, some of the needed additional insertions may be missed.

Deletion anomalies

A "deletion anomaly" is a failure to remove information about an existing database entry when it is time to remove that entry. In a properly normalized database, information about an old, to-be-gotten-rid-of entry needs to be deleted from only one place in the database; in an inadequately normalized database, information about that old entry may need to be deleted from more than one place, and, human fallibility being what it is, some of the needed additional deletions may be missed.

Modification anomalies

A modification of a database involves changing some value of the attribute of a table. In a properly normalized database table, whatever information is modified by the user, the change will be effected and used accordingly.

The purpose of normalization is to reduce the chances for anomalies to occur in a database.

Functional Dependency (FD)

Before moving to the definition and application of normalization, it is important to have an understanding of "functional dependency."

Data Dependency

The logical associations between data items that point the database designer in the direction of a good database design are referred to as determinant or dependent relationships.

Two data items A and B are said to be in a determinant or dependent relationship if certain values of data item B always appears with certain values of data item A. if the data item A is the

determinant data item and B the dependent data item then the direction of the association is from A to B and not vice versa.

$X \rightarrow Y$ holds if whenever two tuples have the same value for X, they must have the same value for Y

The notation is: $A \rightarrow B$ which is read as; B is functionally dependent on A.

In general, a *functional dependency* is a relationship among attributes. In relational databases, we can have a determinant that governs one other attribute or several other attributes.

FDs are derived from the real-world constraints on the attributes.

Example

<i>Dinner</i>	<i>Type of Wine</i>
Meat	Red
Fish	White
Cheese	Rose

Since the type of *Wine* served depends on the type of *Dinner*, we say *Wine* is functionally dependent on *Dinner*.

$Dinner \rightarrow Wine$

<i>Dinner</i>	<i>Type of Wine</i>	<i>Type of Fork</i>
Meat	Red	Meat fork
Fish	White	Fish fork
Cheese	Rose	Cheese fork

Since both *Wine* type and *Fork* type are determined by the *Dinner* type, we say *Wine* is functionally dependent on *Dinner* and *Fork* is functionally dependent on *Dinner*.

$Dinner \rightarrow Wine$

$Dinner \rightarrow Fork$

Partial Dependency

If an attribute which is not a member of the primary key is dependent on some part of the primary key (if we have composite primary key) then that attribute is partially functionally dependent on the primary key.

Let $\{A, B\}$ is the Primary Key and C is no key attribute.

Then if $\{A, B\} \rightarrow C$ and $B \rightarrow C$ or $A \rightarrow C$

Then C is partially functionally dependent on {A, B}

Full Dependency

If an attribute which is not a member of the primary key is not dependent on some part of the primary key but the whole key (if we have composite primary key) then that attribute is fully functionally dependent on the primary key.

Let {A, B} is the Primary Key and C is no key attribute

Then if $\{A, B\} \rightarrow C$ and $B \rightarrow C$ and $A \rightarrow C$ doesn't hold (if B can not determine C and B can not determine C)

Then C Fully functionally dependent on {A, B}

Transitive Dependency

In mathematics and logic, a transitive relationship is a relationship of the following form: "If A implies B, and if also B implies C, then A implies C."

Example:

If Mr X is a Human, and if every Human is an Animal, then Mr X must be an Animal.

Generalized way of describing transitive dependency is that:

If A functionally governs B, AND

If B functionally governs C

THEN A functionally governs C

Provided that neither C nor B determines A i.e. $(B \not\rightarrow A \text{ and } C \not\rightarrow A)$

In the normal notation:

$\{(A \rightarrow B) \text{ AND } (B \rightarrow C)\} \Rightarrow A \rightarrow C$ provided that $B \not\rightarrow A$ and $C \not\rightarrow A$

Steps of Normalization:

We have various levels or steps in normalization called Normal Forms. The level of complexity, strength of the rule and decomposition increases as we move from one lower level Normal Form to the higher.

A table in a relational database is said to be in a certain normal form if it satisfies certain constraints.

Normal form below represents a stronger condition than the previous one

Normalization towards a logical design consists of the following steps:

★ *Un-Normalized Form:*

Identify all data elements

★ *First Normal Form:*

Find *the key* with which you can find *all* data

★ *Second Normal Form:*

Remove part-key dependencies. Make all data dependent on *the whole key*.

★ *Third Normal Form*

Remove non-key dependencies. Make all data dependent on *nothing but the key*.

For most practical purposes, databases are considered normalized if they adhere to third normal form.

First Normal Form (1NF)

Requires that all column values in a table are *atomic* (e.g., a number is an atomic value, while a list or a set is not). We have two ways of achieving this: -

1. Putting each repeating group into a separate table and connecting them with a primary key-foreign key relationship.
2. Moving these repeating groups to a new row by repeating the common attributes. If so then find the key with which you can find all data

Definition: a table (relation) is in 1NF

If

- ★ There are no duplicated rows in the table. Unique identifier.
- ★ Each cell is single-valued (i.e., there are no repeating groups).
- ★ Entries in a column (attribute, field) are of the same kind.

Example for First Normal form (1NF)

UNNORMALIZED

<i>EmpID</i>	<i>FName</i>	<i>LName</i>	<i>SkillID</i>	<i>Skill</i>	<i>SkillType</i>	<i>School</i>	<i>SchoolAdd</i>	<i>Skill level</i>
12	tamiru	tessema	2	SQL	Database	AU	Sidist_killo	5
			6	VB.6	Programming	Helico	Piazza	8
16	Lemma	Alemu	5	C++	Programming	NAC	Saris	6
			1	IP	Programming	Jimma	Jimma_city	4
28	Mesfin	Taye	2	SQL	Database	AAU	Sidist_killo	10
65	Almaz	Abera	2	SQL	Database	Helico	Piazza	9
			4	Prolog	Programming	Jimma	Jimma_city	8
			7	Java	Programming	AAU	Sidist_killo	6
24	Teddy	Tamiru	8	Oracle	Database	NAC	Saris	5
94	Taye	Gizaw	3	Cisco	Networking	AAU	Sidist_killo	7

FIRST NORMAL FORM (1NF)

Remove all repeating groups. Distribute the multi-valued attributes into different rows and identify a unique identifier for the relation so that it can be said is a relation in relational database.

<i>EmpID</i>	<i>FName</i>	<i>LName</i>	<i>SkillID</i>	<i>Skill</i>	<i>SkillType</i>	<i>School</i>	<i>SchoolAdd</i>	<i>Skill level</i>
12	tamiru	tessema	2	SQL	Database	AU	Sidist_killo	5
12	tamiru	tessema	6	VB.6	Programming	Helico	Piazza	8
16	Lemma	Alemu	5	C++	Programming	NAC	Saris	6
16	Lemma	Alemu	1	IP	Programming	Jimma	Jimma_city	4
28	Mesfin	Taye	2	SQL	Database	AAU	Sidist_killo	10
65	Almaz	Abera	2	SQL	Database	Helico	Piazza	9
65	Almaz	Abera	4	Prolog	Programming	Jimma	Jimma_city	8
65	Almaz	Abera	7	Java	Programming	AAU	Sidist_killo	6
24	Teddy	Tamiru	8	Oracle	Database	NAC	Saris	5
94	Taye	Gizaw	3	Cisco	Networking	AAU	Sidist_killo	7

SECOND NORMAL FORM (2NF)

No partial dependency of a non key attribute on part of the primary key. This will result in a set of relations with a level of Second Normal Form.

Any table that is in 1NF and has a single-attribute (i.e., a non-composite) primary key is automatically in 2NF.

Definition: a table (relation) is in 2NF

If

- *It is in 1NF and*
- *If all non-key attributes are dependent on the entire primary key. i.e. no partial dependency.*

Example for 2NF:

EMP_PROJ

<u>EmpID</u>	EmpName	<u>ProjNo</u>	ProjName	ProjLoc	ProjFund	ProjMangID	Incentive
--------------	---------	---------------	----------	---------	----------	------------	-----------

EMP_PROJ rearranged

<u>EmpID</u>	<u>ProjNo</u>	EmpName	ProjName	ProjLoc	ProjFund	ProjMangID	Incentive
--------------	---------------	---------	----------	---------	----------	------------	-----------

Business rule: Whenever an employee participates in a project, he/she will be entitled for an incentive.

This schema is in its 1NF since we don't have any repeating groups or attributes with multi-valued property. To convert it to a 2NF we need to remove all partial dependencies of non key attributes on part of the primary key.

$\{EmpID, ProjNo\} \rightarrow EmpName, ProjName, ProjLoc, ProjFund, ProjMangID, Incentive$

But in addition to this we have the following dependencies

$FD1: \{EmpID\} \rightarrow EmpName$

$FD2: \{ProjNo\} \rightarrow ProjName, ProjLoc, ProjFund, ProjMangID$

$FD3: \{EmpID, ProjNo\} \rightarrow Incentive$

As we can see, some non key attributes are partially dependent on some part of the primary key. This can be witnessed by analyzing the first two functional dependencies (FD1 and FD2). Thus, each Functional Dependencies, with their dependent attributes should be moved to a new relation where the Determinant will be the Primary Key for each.

EMPLOYEE

<u>EmpID</u>	EmpName
--------------	---------

PROJECT

<u>ProjNo</u>	ProjName	ProjLoc	ProjFund	ProjMangID
---------------	----------	---------	----------	------------

EMP_PROJ

EmpID	ProjNo	Incentive
-------	--------	-----------

THIRD NORMAL FORM (3NF)

Eliminate Columns Dependent on another non-Primary Key - If attributes do not contribute to a description of the key, remove them to a separate table. These levels avoid update and delete anomalies.

Definition: a Table (Relation) is in 3NF

If

- *It is in 2NF and*
- *There are no transitive dependencies between a primary key and non-primary key attributes.*

Example for (3NF)

Assumption: Students of same batch (same year) live in one building or dormitory

STUDENT

<u>StudID</u>	Stud_F_Name	Stud_L_Name	Dept	Year	Dormitory
125/97	tamiru	tessema	Info Sc	1	401
654/95	Lemma	Alemu	Geog	3	403
842/95	Mesfin	Taye	Comp. Sc	3	403
165/97	Abera	Belay	Info Sc	1	401
985/95	Almaz	Abera	Geog	3	403

This schema is in its 2NF since the primary key is a single attribute.

Let's take *StudID*, *Year* and *Dormitory* and see the dependencies.

StudID→*Year* AND *Year*→*Dormitory*

And *Year* cannot determine *StudID* and *Dormitory* cannot determine *StudID*

Then transitively *StudID*→*Dormitory*

To convert it to a 3NF we need to remove all transitive dependencies of non key attributes on another non-key attribute.

The non-primary key attributes, dependent on each other will be moved to another table and linked with the main table using Candidate Key- Foreign Key relationship.

STUDENT

<u>StudID</u>	Stud_F_Name	Stud_L_Name	Dept	Year
125/97	tamiru	tessema	Info Sc	1
654/95	Lemma	Alemu	Geog	3
842/95	Mesfin	Taye	Comp. Sc	3
165/97	Abera	Belay	Info Sc	1
985/95	Almaz	Abera	Geog	3

DORM

<u>Year</u>	Dormitory
1	401
3	403

Generally, even though there are other four additional levels of Normalization, a table is said to be normalized if it reaches 3NF. A database with all tables in the 3NF is said to be Normalized Database.

Mnemonic for remembering the rationale for normalization up to 3NF could be the following:

1. No Repeating or Redundancy: - no repeating fields in the table.
2. The Fields Depend Upon the Key: - the table should solely depend on the key.
3. The Whole Key: - no partial key dependency.
4. And Nothing But the Key: - no inter data dependency.
5. So Help Me Codd: - since Codd came up with these rules.

5.1.3. Other Levels of Normalization

Boyce-Codd Normal Form (BCNF):

Isolate Independent Multiple Relationships - No table may contain two or more 1: n or N: M relationships that are not directly related.

The correct solution, to cause the model to be in 4th normal form, is to ensure that all M: M relationships are resolved independently if they are indeed independent, as shown below.

Def: A table is in BCNF if it is in 3NF and if every determinant is a candidate key.

Forth Normal form (4NF)

Isolate Semantically Related Multiple Relationships - There may be practical constraints on information that justify separating logically related many-to-many relationships.

Def: A table is in 4NF if it is in BCNF and if it has no multi-valued dependencies.

Fifth Normal Form (5NF)

A model limited to only simple (elemental) facts, as expressed in ORM.

Def: A table is in 5NF, also called "Projection-Join Normal Form" (PJNF), if it is in 4NF and if every join dependency in the table is a consequence of the candidate keys of the table.

Domain-Key Normal Form (DKNF)

Models are free from all modification anomalies.

Def: A table is in DKNF if every constraint on the table is a logical consequence of the definition of keys and domains.

The underlying ideas in normalization are simple enough. Through normalization we want to design for our relational database a set of tables that;

- (1) Contain all the data necessary for the purposes that the database is to serve.
- (2) have as little redundancy as possible.
- (3) Accommodate multiple values for types of data that require them.
- (4) Permit efficient updates of the data in the database, and
- (5) Avoid the danger of losing data unknowingly.

Pitfalls of Normalization

- ✦ Requires data to see the problems.
- ✦ May reduce performance of the system.
- ✦ Is time consuming.
- ✦ Difficult to design and apply and.
- ✦ Prone to human error.

Chapter 6

The Relational Data Model and the Relational Algebra

The relational model used the basic concept of a relation or table. The columns or fields in the table identify the attributes such as name, age, and so. A tuple or row contains all the data of a single instance of the table such as a person named Doug. In the relational model, every tuple must have a unique identification or key based on the data. In this figure, a social security account number (SSAN) is the key that uniquely identifies each tuple in the relation. Often, keys are used to join data from two or more relations based on matching identification. The relational model also includes concepts such as foreign keys, which are primary keys in one relation that are kept in another relation to allow for the joining of data. As an example of foreign keys is storing your mother's and father's SSAN in the tuple that represent you. Your parents' SSANs are keys for the tuples that represent them and they are foreign keys in the tuple that represents you.

Relational Data Model Concepts

Tables – In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where a row represents records and columns represent the attributes.

Tuple – A single row of a table, which contains a single record for that relation is called a tuple.

Relation instance – A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

Relation schema – A relation schema describes the relation name (table name), attributes, and their names.

Relation key – each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

Attribute domain – every attribute has some pre-defined value scope, known as attribute domain.

Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity Constraints. There are three main integrity constraints –

- Key constraints
- Domain constraints
- Referential integrity constraints

Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called key for that relation. If there is more than one such minimal subset, these are called *candidate keys*.

Key constraints force that –

- In a relation with a key attribute, no two tuples can have identical values for key attributes.
- A key attribute cannot have NULL values.

Key constraints are also referred to as Entity Constraints.

Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Referential integrity Constraints

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

The Relational Operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface. Many Web sites, such as Wikipedia, perform SQL queries when generating pages.

In response to a query, the database returns a result set, which is just a list of rows containing the answers. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted.

Often, data from multiple tables are combined into one, by doing a join. Conceptually, this is done by taking all possible combinations of rows (the Cartesian product), and then filtering out everything except the answer. In practice, relational database management systems rewrite ("optimize") queries to perform faster, using a variety of techniques.

There are a number of relational operations in addition to join. These include project (the process of eliminating some of the columns), restrict (the process of eliminating some of the rows), union (a way of combining two tables with similar structures), difference (that lists the rows in one table that are not found in the other), intersect (that lists the rows found in both tables), and product (mentioned above, which combines each row of one table with each row of the other). Depending on which other sources you consult, there are a number of other operators – many of which can be defined in terms of those listed above. These include semi-join, outer operators such as outer join and outer union, and various forms of division. Then there are operators to rename columns, and summarizing or aggregating operators, and if you permit relation values as attributes (relation-valued attribute), then operators such as group and ungroup. The SELECT statement in SQL serves to handle all of these except for the group and ungroup operators.

Chapter 7

Record Storage and Primary File Organization

File organization: is the organization of the data of a file into records, blocks and access structures. This includes the way records and blocks are stored on the storage medium and linked. Access method: provides a group of operations that can be applied to a file.

File Organization

- The File is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
- File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

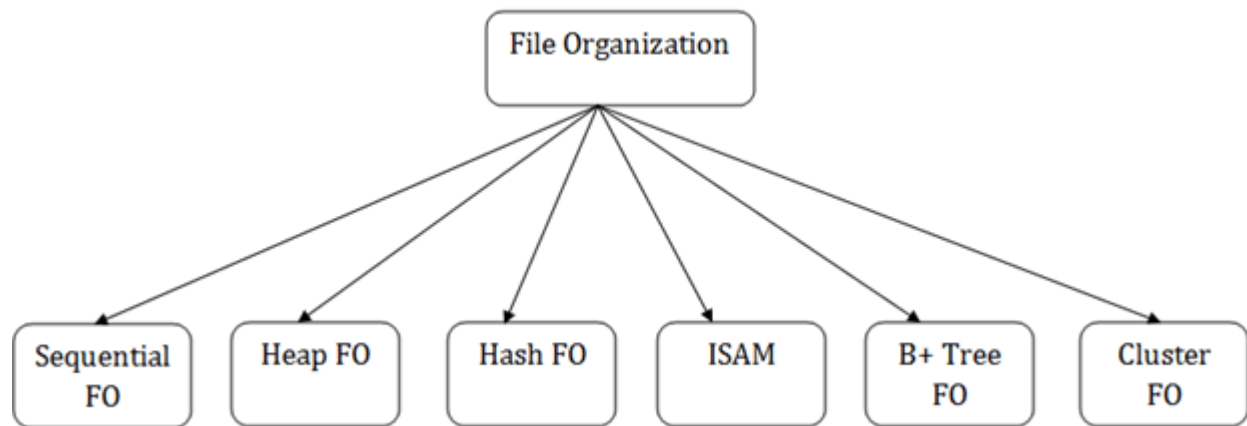
Objective of file organization

- It contains an optimal selection of records, i.e., records can be selected as fast as possible.
- To perform insert, delete or update transaction on the records should be quick and easy.
- The duplicate records cannot be induced as a result of insert, update or delete.
- For the minimal cost of storage, records should be stored efficiently.

Types of file organization:

File organization contains various methods. These particular methods have pros and cons on the basis of access or selection. In the file organization, the programmer decides the best-suited file organization method according to his requirement.

Types of file organization are as follows:



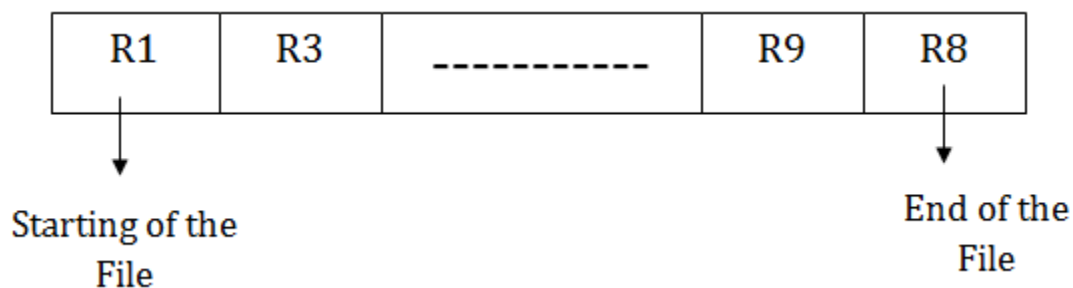
- ♣ Sequential file organization
- ♣ Heap file organization
- ♣ Hash file organization
- ♣ B+ file organization
- ♣ Indexed sequential access method (ISAM)
- ♣ Cluster file organization

Sequential File Organization

This method is the easiest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:

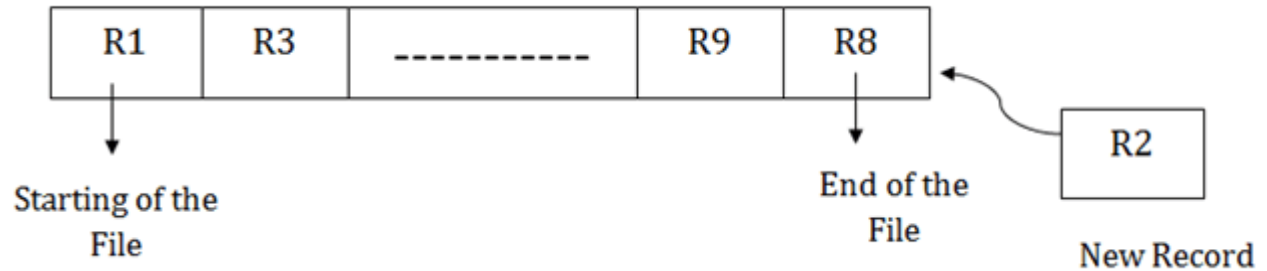
1. Pile File Method:

- It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.
- In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



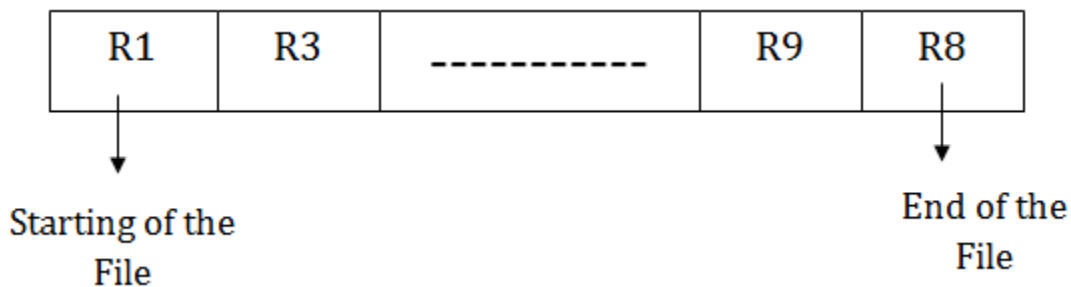
Insertion of the new record:

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.



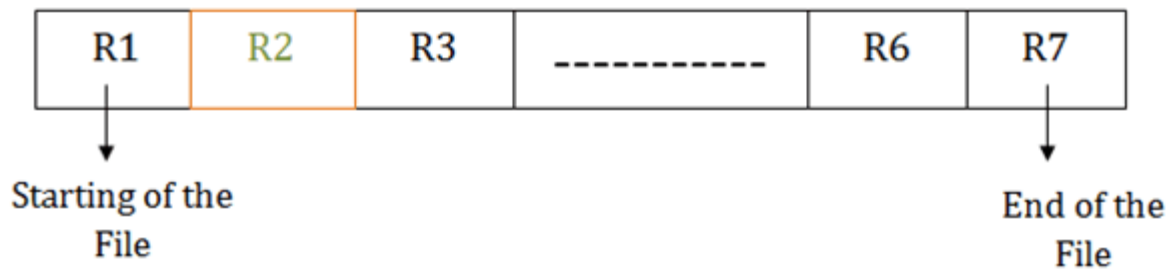
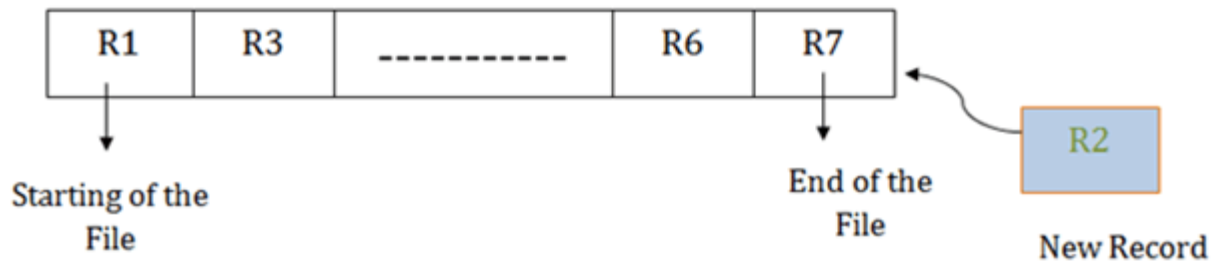
2. Sorted File Method:

- In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on any primary key or any other key.
- In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



Insertion of the new record:

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.



Pros of sequential file organization

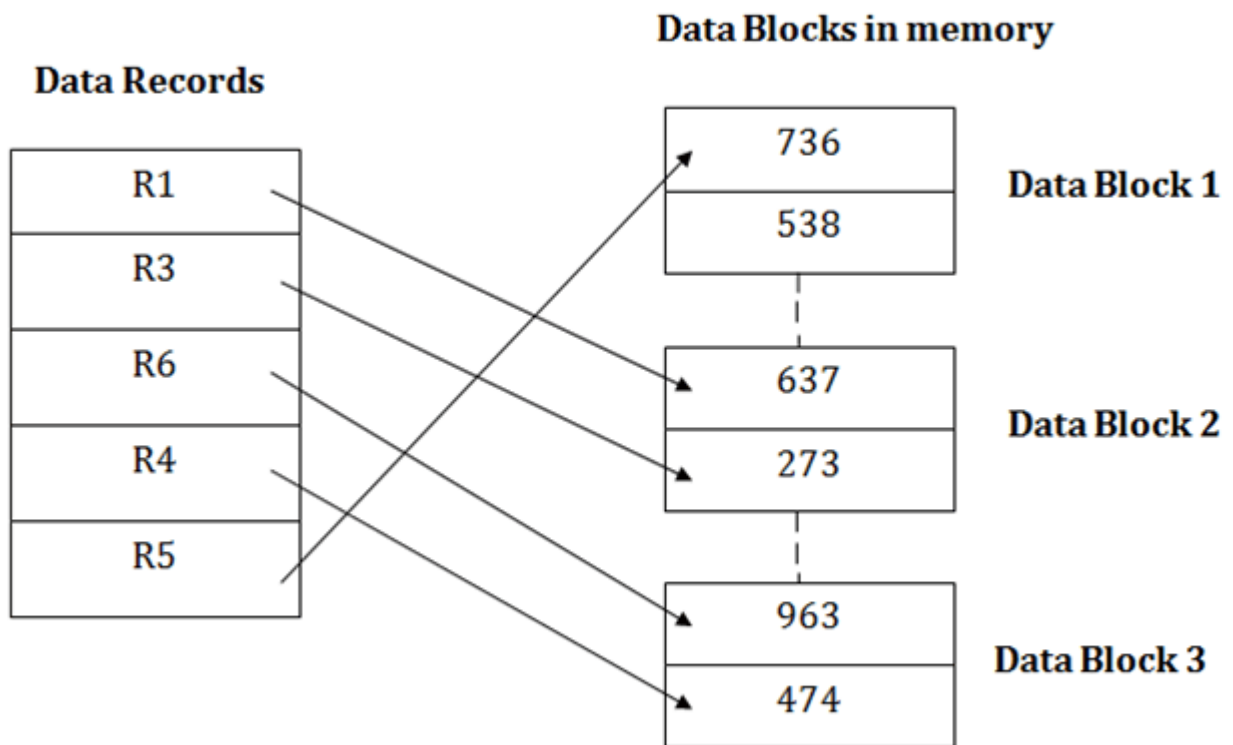
- It contains a fast and efficient method for the huge amount of data.
- In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
- It is simple in design. It requires no much effort to store the data.
- This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.
- This method is used for report generation or statistical calculations.

Cons of sequential file organization

- It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
- Sorted file method takes more time and space for sorting the records.

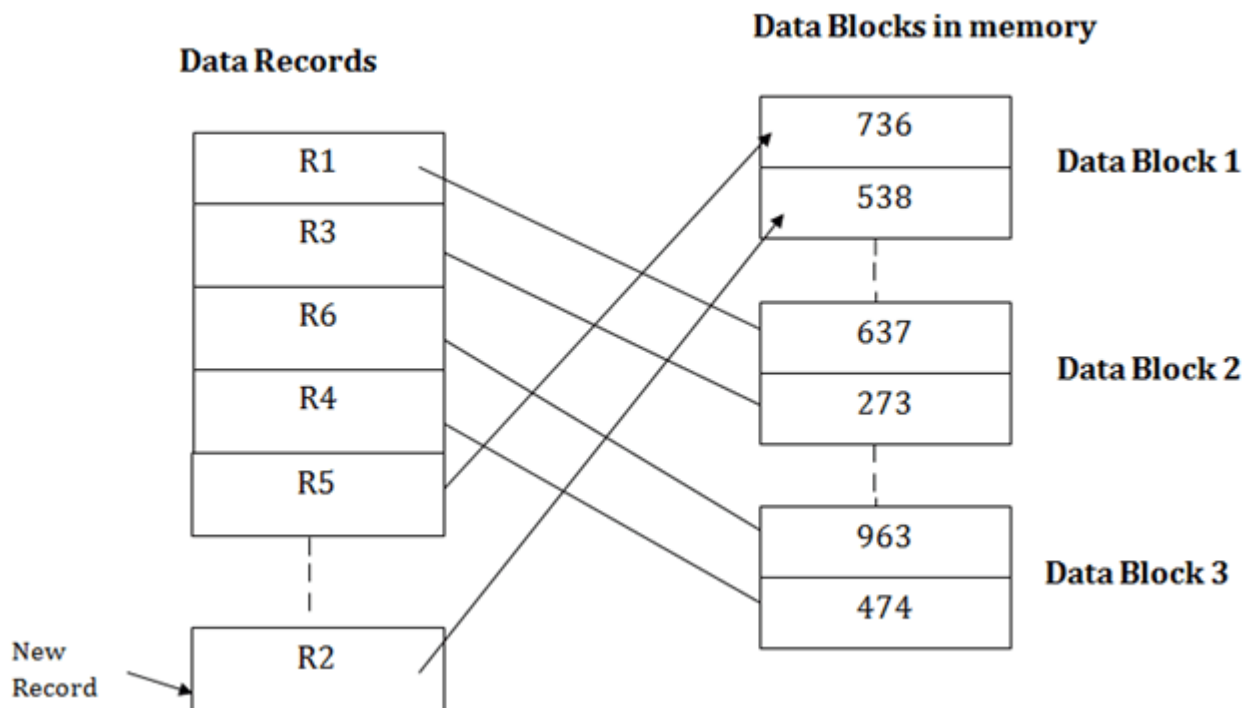
Heap file organization

- It is the simplest and most basic type of organization. It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
- In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.



Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS, let's say data block 1.



If we want to search, update or delete the data in heap file organization, then we need to traverse the data from starting of the file till we get the requested record.

If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records. In the heap file organization, we need to check all the data until we get the requested record.

Pros of Heap file organization

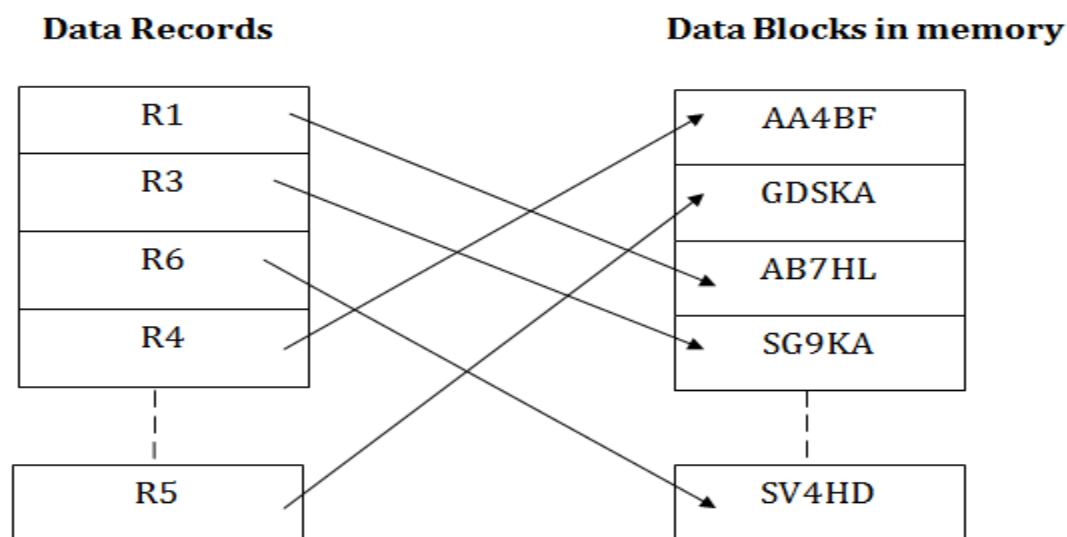
- It is a very good method of file organization for bulk insertion. If there is a large number of data which needs to load into the database at a time, then this method is best suited.
- In case of a small database, fetching and retrieving of records is faster than the sequential record.

Cons of Heap file organization

- This method is inefficient for the large database because it takes time to search or modify the record.
- This method is inefficient for large databases.

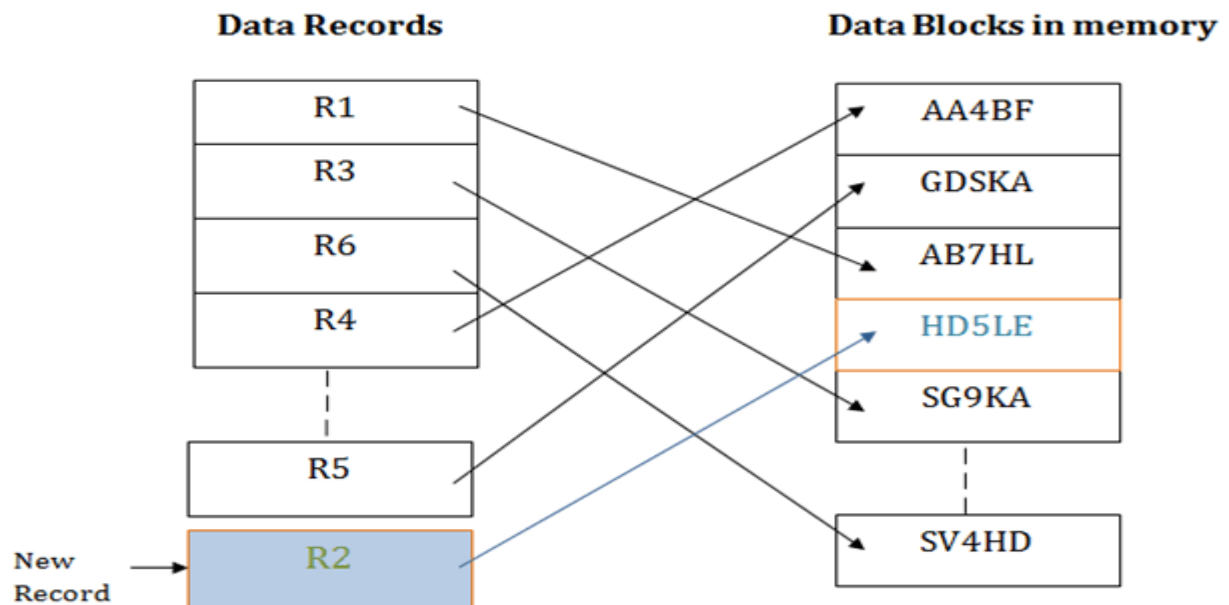
Hash File Organization

Hash File Organization uses the computation of hash function on some fields of the records. The hash function's output determines the location of disk block where the records are to be placed.



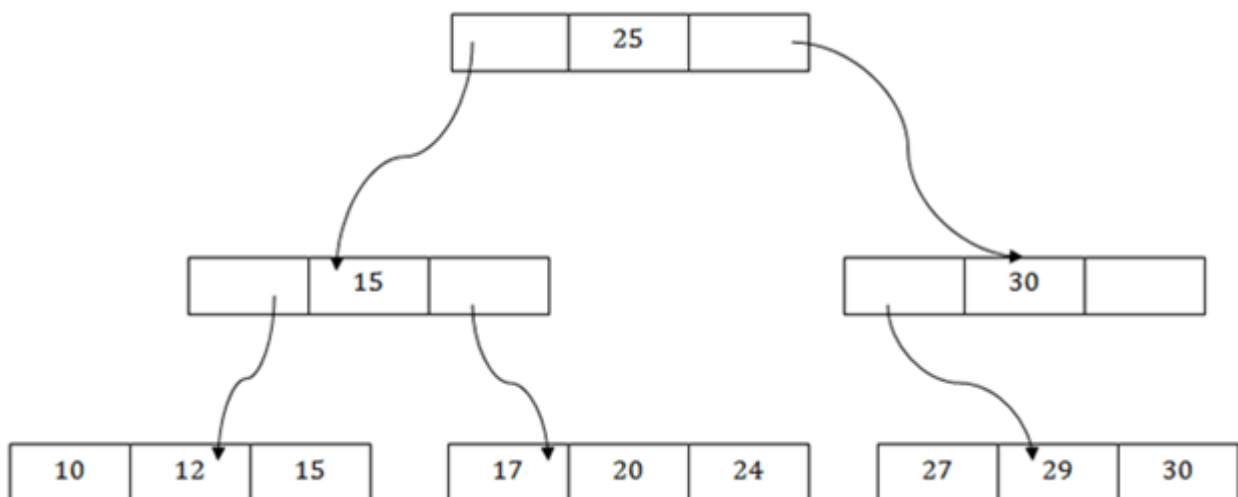
When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address. In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted. The same process is applied in the case of delete and update.

In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.



B+ File Organization

- B+ tree file organization is the advanced method of an indexed sequential access method. It uses a tree-like structure to store records in File.
- It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.
- The B+ tree is similar to a binary search tree (BST), but it can have more than two children. In this method, all the records are stored only at the leaf node. Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.



The above B+ tree shows that:

- There is one root node of the tree, i.e., 25.
- There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.
- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
- There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
- Searching for any record is easier as all the leaf nodes are balanced.
- In this method, searching any record can be traversed through the single path and accessed easily.

Pros of B+ tree file organization

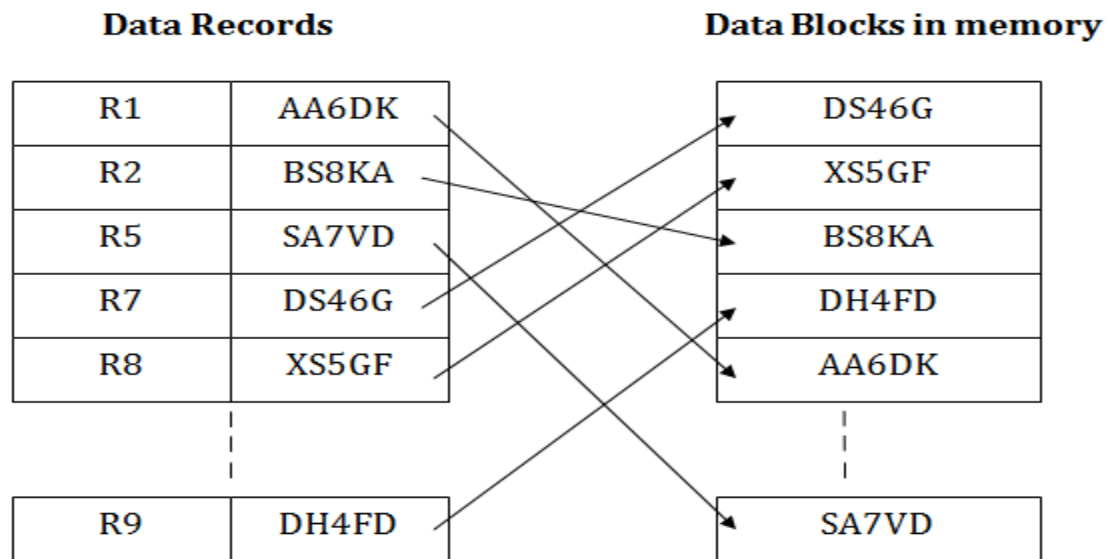
- ✚ In this method, searching becomes very easy as all the records are stored only in the leaf nodes and sorted the sequential linked list.
- ✚ Traversing through the tree structure is easier and faster.
- ✚ The size of the B+ tree has no restrictions, so the number of records can increase or decrease and the B+ tree structure can also grow or shrink.
- ✚ It is a balanced tree structure, and any insert/update/delete does not affect the performance of tree.

Cons of B+ tree file organization

- This method is inefficient for the static method.

Indexed sequential access method (ISAM)

ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.



If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

Pros of ISAM:

- In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.
- This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

Cons of ISAM

- This method requires extra space in the disk to store the index value.
- When the new records are inserted, then these files have to be reconstructed to maintain the sequence.
- When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

Cluster file organization

- When the two or more records are stored in the same file, it is known as clusters. These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
- This method reduces the cost of searching for various records in different files.

- The cluster file organization is used when there is a frequent need for joining the tables with the same condition. These joins will give only a few records from both tables. In the given example, we are retrieving the record for only particular departments. This method can't be used to retrieve the record for the entire department.


EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEP_ID
1	John	Delhi	14
2	Robert	Gujarat	12
3	David	Mumbai	15
4	Amelia	Meerut	11
5	Kristen	Noida	14
6	Jackson	Delhi	13
7	Amy	Bihar	10
8	Sonoo	UP	12

DEPARTMENT

DEP_ID	DEP_NAME
10	Math
11	English
12	Java
13	Physics
14	Civil
15	Chemistry

Cluster Key



DEP_ID	DEP_NAME	EMP_ID	EMP_NAME	ADDRESS
10	Math	7	Amy	Bihar
11	English	4	Amelia	Meerut
12	Java	2	Robert	Gujarat
12		8	Sonoo	UP
13	Physics	6	Jackson	Delhi
14	Civil	1	John	Delhi
14		5	Kristen	Noida
15	Chemistry	3	David	Mumbai

In this method, we can directly insert, update or delete any record. Data is sorted based on the key with which searching is done. Cluster key is a type of key with which joining of the table is performed.

Types of Cluster file organization:

Cluster file organization is of two types:

1. Indexed Clusters:

In indexed cluster, records are grouped based on the cluster key and stored together. The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key- DEP_ID and all the records are grouped.

2. Hash Clusters:

It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

Pros of Cluster file organization

- The cluster file organization is used when there is a frequent request for joining the tables with same joining condition.
- It provides the efficient result when there is a 1:M mapping between the tables.

Cons of Cluster file organization

- This method has the low performance for the very large database.
- If there is any change in joining condition, then this method cannot use. If we change the condition of joining then traversing the file takes a lot of time.
- This method is not suitable for a table with a 1:1 condition.

Index Structure for Files

An index is an ordered list of headings that points to relevant information in materials that are organized in a different order. Generally, whenever an index exists, that index is necessary for being able to find a record within a record series.

Indexes

- indexes are used to speed up record retrieval in response to certain search conditions
- indexes structures provide secondary access paths
- any field can be used as an index
 - multiple indexes can be constructed
- most indexes are based on ordered files
- index files are small and can be retrieved with less block accesses

Single-Level Ordered Indexes

- ordered index similar to an index on a book
- the indexing field stores the value of the index and a pointer to the block where the record can be retrieved
- values in index are ordered
- types of single-level indexes
 - primary index
 - specified on the ordering key field of ordered file of records
 - index file is an ordered file with two keys
 - primary key
 - pointer to a disk block
 - one index entry in the index file for each block in the data file
 - disadvantage: insertion and deletion of records

- move records around and change index values
 - solutions: use unordered overflow file, use linked list of overflow records
- clustering index
 - used if numerous records can have the same value for the ordering field
 - file records are physically ordered on a non-key field without a distinct value for each record
 - index file is an ordered file with two keys
 - clustering field value
 - pointer to a disk block of the first appearance of the field value
- secondary index
 - can be specified on any non ordering field
 - index file is file with two keys
 - indexing field
 - block pointer or record pointer
 - usually needs more storage and longer search time than primary index (because it's dense)
- indexes may be dense or sparse
 - dense index has an index entry for every search key value in the data file
 - sparse index has entries for only some search values
 - applicable when records are sequentially ordered on search-key

Other Types of Indexes

- multilevel Indexes
 - designed to reduce remaining search space as search is conducted
 - index file
 - first level of the multilevel index
 - second level
 - primary index to the first level
 - third level
 - primary index to the second level
- hash indexes
 - secondary structure for file access
 - uses hashing on a search key other than the one used for the primary data file organization
 - index entries of form (k, pr) or (k, p)
 - pr: pointer to the record containing the key
 - p: pointer to the block containing the record for that key

Some General Issues Concerning Indexing

- physical index
 - pointer specifies physical record address
 - disadvantage: pointer must be changed if record is moved
- secondary indexes can be created for any primary record organization
 - complements other primary access methods
- tuning indexes
 - tuning goals
 - dynamically evaluate requirements
 - reorganize indexes for best performance

- reasons for revising initial index choice
 - certain queries may take too long to run due to lack of an index
 - certain indexes may not get utilized
 - certain indexes may undergo too much updating if based on an attribute that undergoes frequent changes

Chapter 8

Introduction to Structural query language (SQL)

SQL is a database language designed for the retrieval and management of data in a relational database. SQL is the standard language for database management. All the RDBMS systems like MySQL, MS Access, Oracle, Sybase, Postgres, and SQL Server use SQL as their standard database language. SQL programming language uses various commands for different operations. We will learn about the like DCL, TCL, DQL, DDL and DML commands in SQL with examples.

Why Use SQL?

Here, are important reasons for using SQL

- It helps users to access data in the RDBMS system.
- It helps you to describe the data.
- It allows you to define the data in a database and manipulate that specific data.
- With the help of SQL commands in DBMS, you can create and drop databases and tables.
- SQL offers you to use the function in a database, create a view, and stored procedure.
- You can set permissions on tables, procedures, and views.

Brief History of SQL

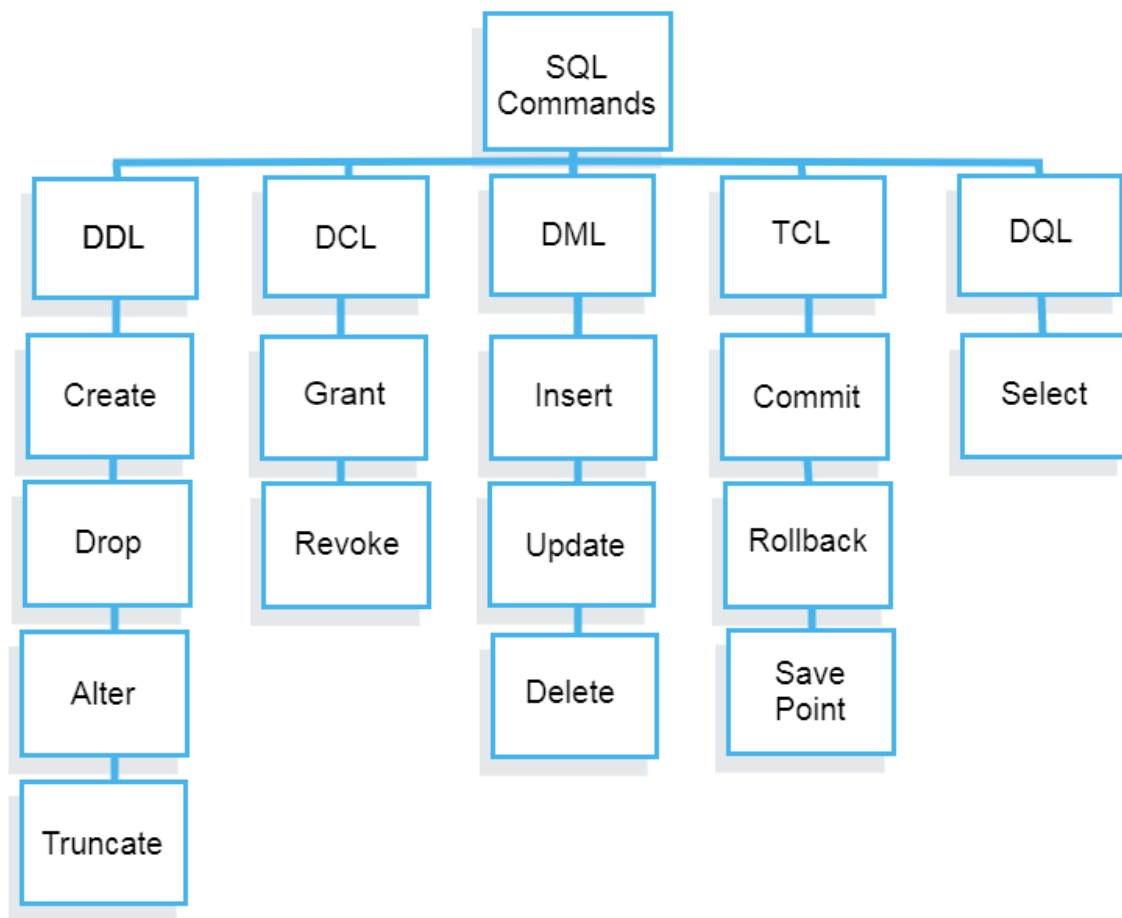
Here, are important landmarks from the history of SQL:

- 1970 – Dr. Edgar F. “Ted” Codd described a relational model for databases.
- 1974 – Structured Query Language appeared.
- 1978 – IBM released a product called System/R.
- 1986 – IBM developed the prototype of a relational database, which is standardized by ANSI.
- 1989- First ever version launched of SQL
- 1999 – SQL 3 launched with features like triggers, object-orientation, etc.
- SQL2003- window functions, XML-related features, etc.
- SQL2006- Support for XML Query Language
- SQL2011-improved support for temporal databases

Types of SQL

Here are five types of widely used SQL queries.

- 1) Data Definition Language (DDL)
- 2) Data Manipulation Language (DML)
- 3) Data Control Language(DCL)
- 4) Transaction Control Language(TCL)
- 5) Data Query Language (DQL)



Data Definition Language

Data Definition Language helps you to define the database structure or schema. Let's learn about DDL commands with syntax.

Five types of DDL commands in SQL are:

CREATE

CREATE statements is used to define the database structure schema:

Syntax:

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES [...]);

For example:

- Create database university;
- Create table students;
- Create view for_students;

DROP

Drops commands remove tables and databases from RDBMS.

Syntax

DROP TABLE ;

For example:

- Drop object_type object_name;
- Drop database university;

- Drop table student;

ALTER

Alter command allows you to alter the structure of the database.

Syntax:

To add a new column in the table

ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify an existing column in the table:

ALTER TABLE MODIFY(COLUMN DEFINITION....);

For example:

Alter table guru99 add subject varchar;

TRUNCATE:

This command used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE table students;

Data Manipulation Language

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database.

There are three basic constructs which allow database program and user to enter data and information are:

Here are some important DML commands in SQL:

- INSERT
- UPDATE
- DELETE

INSERT:

This is a statement is a SQL query. This command is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME (col1, col2, col3,.... col N)

VALUES (value1, value2, value3, valueN);

Or

INSERT INTO TABLE_NAME

VALUES (value1, value2, value3, valueN);

For example:

INSERT INTO students (RollNo, FirstName, LastName) VALUES ('60', 'Tom', 'Erichsen');

UPDATE:

This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

For example:

```
UPDATE students
```

```
SET FirstName = 'Jhon', LastName= 'Wick'
```

```
WHERE StudID = 3;
```

DELETE:

This command is used to remove one or more rows from a table.

Syntax:

```
DELETE FROM table_name [WHERE condition];
```

For example:

```
DELETE FROM students
```

```
WHERE FirstName = 'Jhon';
```

Data Query Language (DQL)

Data Query Language (DQL) is used to fetch the data from the database. It uses only one command:

SELECT:

This command helps you to select the attribute based on the condition described by the WHERE clause.

Syntax:

```
SELECT expressions FROM TABLES WHERE conditions;
```

For example:

```
SELECT FirstName FROM Student WHERE RollNo > 15;
```

Data Control Language

The *Data Control Language* is a subset of the Structured Query Language. Database administrators use DCL to configure security access to relational databases. It complements the *Data Definition Language*, which adds and deletes database objects, and the *Data Manipulation Language*, which retrieves, inserts, and modifies the contents of a database.

DCL is the simplest of the SQL subsets, as it consists of only three commands: GRANT, REVOKE, and DENY. Combined, these three commands provide administrators with the flexibility to set and remove database permissions in granular fashion.

Adding Permissions with the GRANT Command

The GRANT command adds new permissions to a database user. It has a very simple syntax, defined as follows:

```
GRANT [privilege]
```

```
ON [object]
```

```
TO [user]
```

```
[WITH GRANT OPTION]
```

Here's the rundown on each of the parameters you can supply with this command:

- Privilege — can be either the keyword ALL (to grant a wide variety of permissions) or a specific database permission or set of permissions. Examples include CREATE DATABASE, SELECT, INSERT, UPDATE, DELETE, EXECUTE and CREATE VIEW.
- Object — can be any database object. The valid privilege options vary based on the type of database object you include in this clause. Typically, the object will be either a database, function, stored procedure, table or view.
- User — can be any database user. You can also substitute a role for the user in this clause if you wish to make use of role-based database security.
- If you include the optional WITH GRANT OPTION clause at the end of the GRANT command, you not only grant the specified user the permissions defined in the SQL statement but also give the user permission to further grant those same permissions to *other* database users. For this reason, use this clause with care.

For example, assume you wish to grant the user *Joe* the ability to retrieve information from the *employee* table in a database called *HR*. Use the following SQL command:

```
GRANT SELECT
ON HR.employees
TO Joe
```

Joe can retrieve information from the employees' table. He will not, however, be able to grant other users permission to retrieve information from that table because the DCL script did not include the WITH GRANT OPTION clause.

Revoking Database Access

The REVOKE command removes database access from a user previously granted such access. The syntax for this command is defined as follows:

```
REVOKE [GRANT OPTION FOR] [permission]
ON [object]
FROM [user]
[CASCADE]
```

Here's the rundown on the parameters for the REVOKE command:

- Permission — specifies the database permissions to remove from the identified user. The command revokes both GRANT and DENY assertions previously made for the identified permission.
- Object — can be any database object. The valid privilege options vary based on the type of database object you include in this clause. Typically, the object will be either a database, function, stored procedure, table, or view.
- User — can be any database user. You can also substitute a role for the user in this clause if you wish to make use of role-based database security.
- The GRANT OPTION FOR clause removes the specified user's ability to grant the specified permission to other users. If you include the GRANT OPTION FOR clause in a REVOKE

statement, the primary permission is not revoked. This clause revokes only the granting ability.

- The CASCADE option also revokes the specified permission from any users that the specified user granted the permission.

The following command revokes the permission granted to Joe in the previous example:

```
REVOKE SELECT
ON HR.employees
FROM Joe
```

Explicitly Denying Database Access

The DENY command explicitly prevents a user from receiving a particular permission. This feature is helpful when a user is a member of a role or group that is granted a permission, and you want to prevent that individual user from inheriting the permission by creating an exception. The syntax for this command is as follows:

```
DENY [permission]
ON [object]
TO [user]
```

The parameters for the DENY command are identical to those used for the GRANT command. For example, if you wished to ensure that Matthew would never receive the ability to delete information from the employees' table, issue the following command:

```
DENY DELETE
ON HR.employees
TO Matthew
```

Examples of DCL commands:

Commands that come under DCL:

- 1) Grant
- 2) Revoke

Grant:

This command is used to give user access privileges to a database.

Syntax:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

For example:

```
GRANT SELECT ON Users TO 'Tom'@'localhost';
```

Revoke:

It is useful to back permissions from the user.

Syntax:

```
REVOKE privilege_name ON object_name FROM {user_name | PUBLIC | role_name}
```

For example:

REVOKE SELECT, UPDATE ON student FROM BCA, MCA;

Using GRANT and REVOKE

Data Control Language (DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

- System: This includes permissions for creating session, table, etc and all types of other system privileges.
- Object: This includes permissions for any command or query to perform any operation on the database tables.

In DCL we have two commands,

- GRANT: Used to provide any user access privileges or other privileges for the database.
- REVOKE: Used to take back permissions from any user.

8.1. Transaction control language

Transaction control language or TCL commands deal with the transaction within the database.

Commit

This command is used to save all the transactions to the database.

Syntax:

Commit;

For example:

DELETE FROM Students WHERE RollNo =25;

COMMIT;

Rollback

Rollback command allows you to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

Example:

DELETE FROM Students WHERE RollNo =25;

SAVEPOINT

This command helps you to set a savepoint within a transaction.

Syntax:

SAVEPOINT SAVEPOINT_NAME;

Example:

SAVEPOINT RollNo;

Summary:

- SQL is a database language designed for the retrieval and management of data in a relational database.
- It helps users to access data in the RDBMS system
- In the year 1974, the term Structured Query Language appeared
- Five types of SQL queries are 1) Data Definition Language (DDL) 2) Data Manipulation Language (DML) 3) Data Control Language(DCL) 4) Transaction Control Language(TCL) and, 5) Data Query Language (DQL)
- Data Definition Language (DDL) helps you to define the database structure or schema.
- Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data.
- DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give “rights & permissions.”
- Transaction control language or TCL commands deal with the transaction within the database.
- Data Query Language (DQL) is used to fetch the data from the database.

Summary of questions for Fundamentals of Database system

1. A collection of facts and figures is known to be as
 - A. **Data**
 - B. Database
 - C. Sequence data
 - D. Structured data
2. A pool in the connection pooling refers to
 - A. A set of open ODBC/JDBC connections
 - B. A set of available ODBC/JDBC connections
 - C. A set of closed ODBC/JDBC connections
 - D. A set of similar ODBC/JDBC connections
3. Records of a Log consists of
 - A. Data Value
 - B. Acknowledgement Value
 - C. New value and old value
 - D. Error value
4. Data model that preceded relational data model is known to be
 - A. Network Data Model
 - B. Structured Data Model
 - C. Hierarchical Data Model
 - D. A and B
5. A DDL operation referring to some object is protected by
 - A. Encapsulation
 - B. Locking
 - C. Encryption
 - D. Views log
6. Commanding unstructured textual data is referred to as
 - A. Information Access
 - B. Information Manipulation
 - C. Information Updating
 - D. Information Retrieval
7. A collection of tables to represent data and relationship among data is represented through model
 - A. ER data model
 - B. Semi-structure data model
 - C. Relational data model
 - D. Object oriented data model
8. A component of storage manager managing the most critical part of the database system is known to be
 - A. Transaction Manager
 - B. File Manager
 - C. Authorization and integrity manager
 - D. Buffer manager

9. The terminology referring to data dictionary cache is known to be
 - A. Row cache
 - B. Tuple cache
 - C. Attribute cache
 - D. Column cache
10. A combination of Cartesian product followed by a selection process is called
 - A. Association
 - B. Joins
 - C. Product formulations
 - D. Protocols
11. A type database data models that allows the specification of data is said to be
 - A. Relational data model
 - B. Object-based data model
 - C. Entity relationship data model
 - D. Semi-structured data model
12. The most widely used structure for recording database modifications is called
 - A. Scheduling
 - B. Log
 - C. Buffering
 - D. Locking
13. A SQL statement comprising a logical entity is called
 - A. Database Object
 - B. Database Definition
 - C. Database Operation
 - D. Database Declaration
14. The process of database in which the entity-relationship model is widely used is said to be
 - A. Testing
 - B. Retrieval
 - C. Implementation
 - D. Design
15. Ensuring the properties of atomicity and durability is the responsibility of
 - A. Data Manager
 - B. Transaction Manager
 - C. Recovery manager
 - D. Buffer manager
16. The number of keys used in the encryption and decryption process of asymmetric-key encryption technique are
 - A. 2
 - B. 4
 - C. 3
 - D. 5
17. A combination of Cartesian product followed by a selection process is called
 - A. Join algorithm
 - B. Merge-join algorithm
 - C. Split-join algorithm
 - D. Union-join algorithm
18. The set of commands, like SELECT/FROM/WHERE lies at the section
 - A. DDL
 - B. D-DDL
 - C. DML
 - D. P-DML
19. Functional dependencies are considered poor detection method for
 - A. ER-Diagrams
 - B. UML
 - C. Relational diagrams
 - D. Object-based diagrams
20. The connection created upon the request received at the web server is said to be
 - A. Temporarily
 - B. Uniformly
 - C. Permanently
 - D. Linearly
21. Relational data model can represent an exemplary element named to be
 - A. Logical based models
 - B. Record based models
 - C. View based models
 - D. Entity based model
22. Various data values have permitted types of access in the database which is settled
 - A. Assertions
 - B. Referential integrity
 - C. Domain constraints
 - D. Authorization
23. The access of types differentiations are expressed in terms of
 - A. Authorization
 - B. Domain constraints
 - C. Referential integrity
 - D. Assertions
24. Any future modification to the database when the assertion is valid is considered to be
 - A. Violated
 - B. Allowed
 - C. Not allowed
 - D. Not violated

25. At the creation of an assertion the system it against

- A. Query
- B. Errors

- C. Validity
- D. Invalidity

Answer Key

- 1. A
- 2. A
- 3. B
- 4. D
- 5. B
- 6. D
- 7. C
- 8. D

- 9. A
- 10. B
- 11. D
- 12. B
- 13. D
- 14. D
- 15. C
- 16. A
- 17. B
- 18. C

- 19. A
- 20. A
- 21. B
- 22. D
- 23. A
- 24. B
- 25. C

References

1. Introduction to Database systems- RameezElmasri and Shamakanth B. Navathe
2. Introduction to Relational Databases and SQL Programming - Christopher Allen, Simon Chatwin,` Catherine A. Creary
3. Fundamentals of Database systems - C.J. Data
4. Database – Models, Language and Design - James L. Johnson
5. Joseph M. Hellerstein, Michael Stonebraker and James *Hamilton Architecture of a Database System*
6. <https://www.javatpoint.com/dbms-cluster-file-organization>
7. <https://www.javatpoint.com/dbms-tutorial>