Chapter one

1. Introduction to Database systems

After going through this chapter, students will be able to:

- ♣ Define what is data, database, database system, database management system (DBMS) and information
- ♣ Identify the alternative data handling (manual approach, file based approach, database approach)
- ♣ Identify database development life cycle
- ♣ Differentiate actors on scene and workers behind scene

Data: is a collection of raw facts.
Information: is a processed data in the form that is meaningful to the user.
A database system is basically a computerized record keeping 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. They are used to maintain internal records, to present data to customers and clients on the World-Wide-Web, and to support many other commercial processes. Databases are likewise found at the core of many modern organizations. The power of databases comes from a body of knowledge and technology that has developed over several decades and is embodied in specialized software called a database management system, or DBMS. A DBMS is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely. These systems are among the most complex types of software available.
Thus, for our question: 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. Users of the database can perform a variety of operations. Such as:

- ♦ Adding new data to empty file
- ♦ Adding new data to existing file
- ♦ Retrieving data from existing file
- ♦ Modifying data to existing file
- ♦ Deleting data from existing file
- ♦ Searching for target information

Thus the DB course is about:

- How to organize data
- Supporting multiple users
- Efficient and effective data retrieval
- Secured and reliable storage of data
- Maintaining consistent data
- Making information useful for decision making

Data management passes through the different levels of development along with the development in technology and services. These levels could best be described by categorizing the levels into three levels of development. Even though there is an advantage and a problem overcome at each new level, all methods of data handling are in use to some extent. The major three levels are;

- Manual approach
- File based approach
- Database approach

1.1. 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 labour.

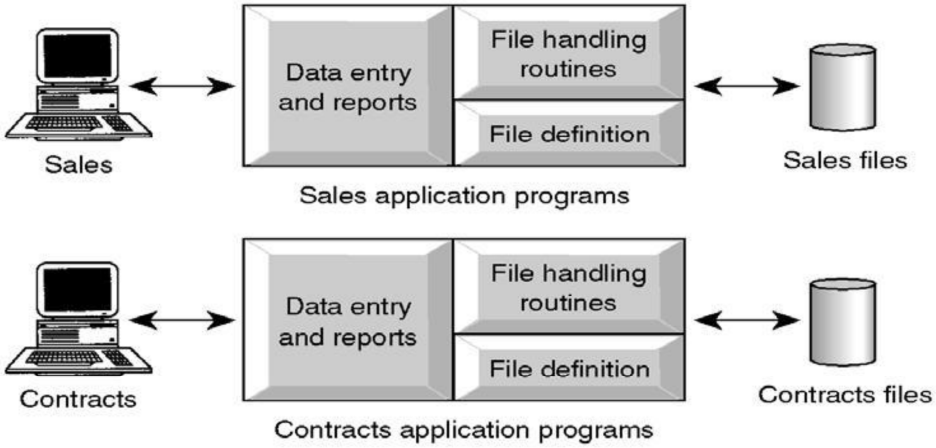
- Files for as many event and objects as the organization has are used to store information.
- Each of the files containing various kinds of information is labeled and stored in one or more cabinets.
- The cabinets could be kept in safe places for security purpose based on the sensitivity of the information 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

1.2. File based approach

- File based approaches were an early attempt to computerize the manual filing system.
- This approach is the decentralized computerized data handling method.
- It is s a collection of application programs that performs services for the end users.
- Each program defines and manages its own data.
- Since every application defines and manages its own data, the system is subjected to serious data duplication problem.
- Since every application defines and manages its own data, the system is subjected to serious data duplication problem.
- Programming languages were used.
- Example of programming languages: C++, Cobol, Pascal



Sales Files

Property_for_Rent(Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, Owner Number)

Owner(Owner Number, First Name, Last Name, Address, Telephone Number)

Renter(Renter Number, First Name, Last Name, Address, Telephone Number, Preferred Type, Maximum Rent)

Contracts Files

Lease(Lease Number, Property Number, Renter Number, Monthly Rent, Payment Method, Deposit, Paid, Rent Start Date, Rent Finish Date, Duration)

Property_for_Rent(Property Number, Street, Area, City, Post Code, Monthly Rent)

Renter(Renter Number, First Name, Last Name, Address, Telephone Number)

Limitations of File Based approach

1) Separation/Isolation of data

When data is isolated in separate files, it is difficult to access data that should be available. This is because; there is no concept of relationship between files. Therefore, we need to create a temporary file for the participating files.

2) Duplication of data (Redundancy)

This is concerning with storage of similar information in multiple files
Disadvantage of redundancy:
a) It costs time and money to enter the data
b) It takes up additional storage space (memory space)
c) Inconsistency: this is loss of data integrity.

3) Data Dependence

- Changes to an existing structure are difficult to make.
- Example: change in the size of Student Name (from 20 characters to 30 characters) requires a new program to convert student file to a new format.
- The new program opens original student file, open a temporary file, read records from original student file and write to the temporary file, delete the original student file and finally rename the temporary file as student file.
- It is time consuming
- Prone to error

4) Incompatible file formats

- The structure of file is dependent on the application programs.
- Incompatibility of files makes them difficult to process jointly.
- Example: consider two files with in the same enterprise but in different departments, or in different branches:
- If the first file is constructed using COBOL and the second file is written using C++, then there will be a problem of integrity.

1.3. Database approach

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 collection of related data in an organized way.
- Most of the time, Organization is in tabular form.
- E.g. book database

Call no	Title	Author	Publisher	No of copies
QA46	Introduction to dbase	Colony Addison	Wesley	15

The organization of the database becomes necessary when the data is voluminous.
Otherwise, managing data will be very difficult.

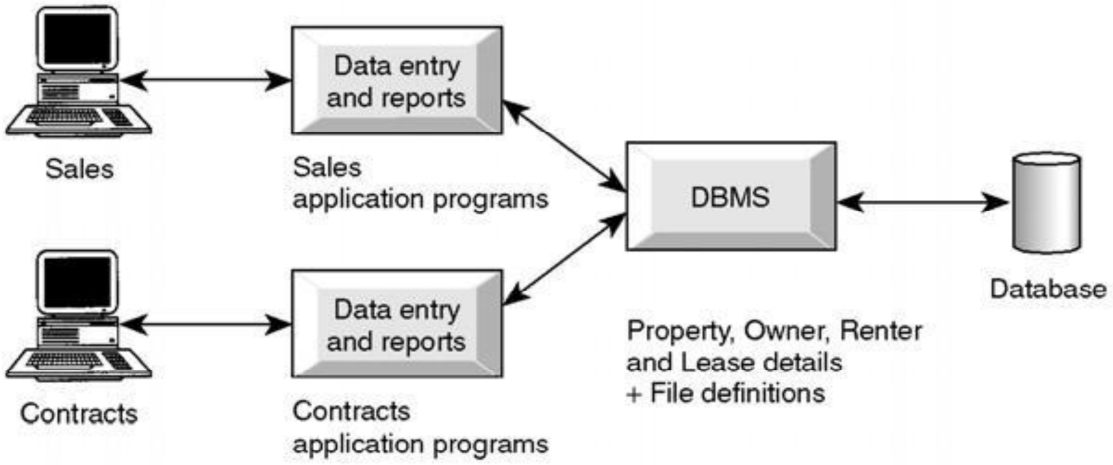
E.g. A Manufacturing Company with product data
A Bank with account data
A Hospital with patients
A University with Students
A government with planning data

- 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.
- In addition to containing data required by an organization, database also contains a description of the data which called as "Metadata" or "Data Dictionary" or "Systems Catalogue" or "Data about Data".
- Since a database contains information about the data (metadata), it is called a self descriptive collection on integrated records.
- The purpose of a database is to store information and to allow users to retrieve and update that information on demand.
- Database is designed once and used simultaneously by many users.
- Unlike the traditional file based approach in database approach there is program data independence. That is the separation of the data definition from the application. Thus the application is not affected by changes made in the data structure and file organization.
- Each database application will perform the combination of: Creating database, Reading, Updating and Deleting data.

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 labour*: 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.



Property_for_Rent(Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, Owner Number)

Owner(Owner Number, First Name, Last Name, Address, Telephone Number)

Renter(Renter Number, First Name, Last Name, Address, Telephone Number), Preferred Type, Maximum Rent)

Lease(Lease Number, Property Number, Renter Number, Payment Method, Deposit, Paid, Rent Start Date, Rent Finish Date)

Limitations and risk of Database Approach

- ♣ Introduction of new professional and specialized personnel.
- ♣ Complexity in designing and managing data
- ♣ To 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 users perspective
- ♣ Reduced performance due to centralization and data independency
- ♣ High impact on the system when failure occurs to the central system.

1.4. 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. A DBMS also provides a systematic method for creating, updating, storing, retrieving data in a database. Example: Ms Access, FoxPro, SQL Server, MySQL, Oracle… DBMS also provides the service of controlling data access, enforcing data integrity, managing concurrency control, and recovery. Having this in mind, a full scale DBMS should at least have the following services to provide to the user.

1. Data ***storage***, ***retrieval*** and ***update*** in the database
2. A user accessible ***catalogue***
3. ***Transaction support service***: ALL or NONE transaction, which minimize data inconsistency.
4. ***Concurrency Control Services***: access and update on the database by different users simultaneously should be implemented correctly.
5. ***Recovery Services***: a mechanism for recovering the database after a failure must be available.
6. ***Authorization Services*** (Security): must support the implementation of access and authorization service to database administrator and users.
7. ***Support for Data Communication***: should provide the facility to integrate with data transfer software or data communication managers.
8. ***Integrity Services***: rules about data and the change that took place on the data, correctness and consistency of stored data, and quality of data based on business constraints.
9. Services to promote ***data independency*** between the data and the application
10. ***Utility services***: sets of utility service facilities like
    - Importing data
    - Statistical analysis support
    - Index reorganization
    - Garbage collection

1.4.1 DBMS and Components of DBMS Environment

A DBMS is software package used to design, manage, and maintain databases. Each DBMS should have facilities to define the database, manipulate the content of the database and control the database. These facilities will help the designer, the user as well as the database administrator to discharge their responsibility in designing, using and managing the database. 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"

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.

The DBMS is software package that helps to design, manage, and use data using the database approach. Taking a DBMS as a system, one can describe it with respect to it environment or other systems interacting with the DBMS. 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. These components are comprised of various types of personal computers, mainframe or any server computers to be used in multi-user system, network infrastructure, and other peripherals required in the system.
2. Software: are collection of commands and programs used to manipulate the hardware to perform a function. These include components like the DBMS software, application programs, operating systems, network software, language software and other relevant software.
3. Data: since the goal of any database system is to have better control of the data and making data useful, Data is the most important component to the user of the database. There are two categories of data in any database system: that is *Operational* and *Metadata*. Operational data is the data actually stored in the system to be used by the user. Metadata is the data that is used to store information about the database itself. The structure of the data in the database is called the ***schema***, which is composed of the *Entities*, *Properties of entities*, and *relationship between entities*.
4. Procedure: this is the rules and regulations on *how to design and use* a database. It includes procedures like how to log on to the DBMS, how to use facilities, how to start and stop transaction, how to make backup, how to treat hardware and software failure, how to change the structure of the database.
5. People: this component is composed of the people in the organization that are responsible or play a role in designing, implementing, managing, administering and using the resources in the database. This component includes group of people with high level of knowledge about the database and the design technology to other with no knowledge of the system except using the data in the database.

1.5. Database Development Life Cycle

As it is one component in most information system development tasks, there are several steps in designing a database system. Here more emphasis is given to the design phases of the system development life cycle. The major steps in database design are;

1. **Planning**: that is identifying information gap in an organization and propose a database solution to solve the problem.
2. **Analysis**: that concentrates more on fact finding about the problem or the opportunity. Feasibility analysis, requirement determination and structuring, and selection of best design method are also performed at this phase.
3. 3. **Design**: in database designing more emphasis is given to this phase. The phase is further divided into three sub-phases.
    A. **Conceptual Design**: concise description of the data, data type, relationship between data and constraints on the data.
        • There is no implementation or physical detail consideration.
        • Used to elicit and structure all information requirements
    B. **Logical Design**: a higher level conceptual abstraction with selected specific data model to implement the data structure.
        • It is particular DBMS independent and with no other physical considerations.
    C. **Physical Design**: physical implementation of the upper level design of the database with respect to internal storage and file structure of the database for the selected DBMS.
        • To develop all technology and organizational specification.
4. **Implementation**: the testing and deployment of the designed database for use.
5. **Operation and Support**: administering and maintaining the operation of the database system and providing support to users.

1.6. Roles in Database Design and Use

As people are one of the components in DBMS environment, there are group of roles played by different stakeholders of the designing and operation of a database system.

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. It is Responsible for the physical realization of the database. It also 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.

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.

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.

A. Naïve 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.

B. 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

C. Casual Users

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

These users can be again classified as "Actors on the Scene" and "Workers Behind the Scene".

Actors on the Scene:

- ♣ Data Administrator
- ♣ Database Administrator
- ♣ Database Designer
- ♣ End Users

Workers Behind the Scene

- ♣ **DBMS** designers and implementers: who design and implement different DBMS software.
- ♣ **Tool Developers**: experts who develop software packages that facilitates database system designing and use. Prototype, simulation, code generator developers could be an example. Independent software vendors could also be categorized in this group.
- ♣ **Operators and Maintenance Personnel**: system administrators who are responsible for actually running and maintaining the hardware and software of the database system and the information technology facilities.

Chapter two

2. Database System Concepts and Architecture

2.1 Data models

A specific DBMS has its own specific Data Definition Language, but this type of language is too low level to describe the data requirements of an organization in a way that is readily understandable by a variety of users. We need a higher-level language. Such a higher-level is called data-model.

A **model** is a representation of real world objects and events and their associations.

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

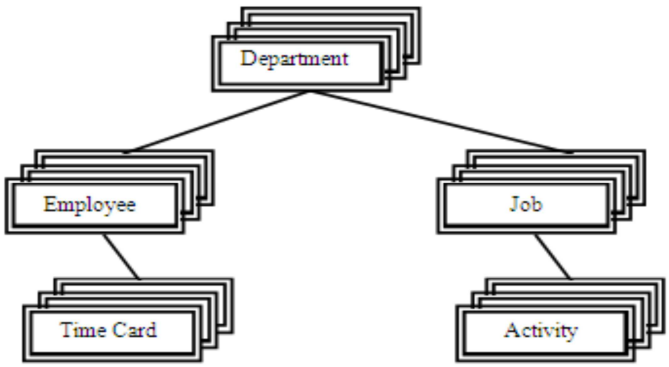
- Data
- Data relationships
- Data semantics
- Data constraints

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

2.1.1 Basic Data Models

2.1.1.1 Hierarchical Model

• The simplest data model
• Records are represented by rectangles.
• 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:

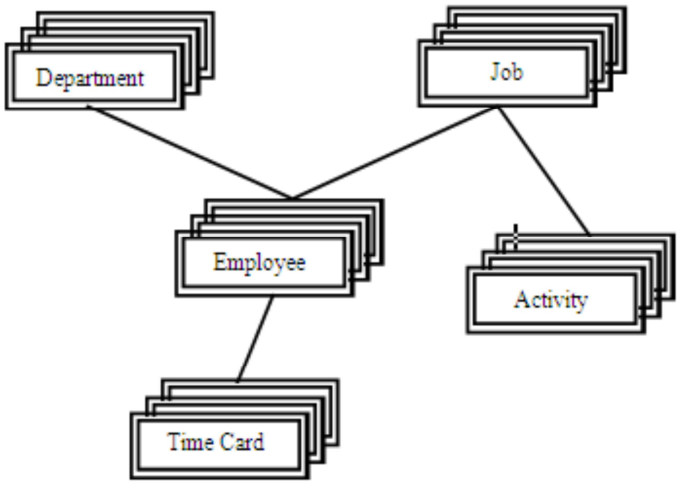
- Hierarchical Model is simple to construct and operate on
- Corresponds to a number of natural hierarchically organized domains
  - e.g., assemblies in manufacturing, personnel organization in companies
- Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.

DISADVANTAGES of Hierarchical Data Model:

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

2.1.1.2 Network Model

• Allows record types to have more than one parent unlike hierarchical model
• A network data models sees records as set members
• Each set has an owner and one or more members
• Allow no many to many relationship 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"

2.1.1.3 Relational Data Model

A relational model is a model that is perceived by the user as table.

- Developed by **Dr. Edgar Frank Codd in 1970** (famous paper, 'A Relational Model for Large Shared Data Banks')
- Terminologies originates from the branch of mathematics called set theory and relation
- 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 tables- **rows and columns**.
- A **row** of the table is called tuple equivalent to **record**.
- A **column** of a table is called attribute equivalent to **fields**.
- Data value is the value of the Attribute.
- 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.
- The tables seem to be independent but are related somehow.
- No physical consideration of the storage is required by the user.
- Many tables are merged together to come up with a new virtual view of the relationship.

Alternative terminologies

- ❖ Relation=Table = File
- ❖ Tuple = Row = Record
- ❖ Attribute = Column = Field

- The rows represent records (collections of information about separate items).
- The columns represent fields (particular attributes of a record)
- Conducts searches by using data in specified columns of one table to find additional data in another table.
- In conducting searches, a relational database matches information from a field in one table with information in a corresponding field of another table to produce a third table that combines requested data from both tables.

2.1.1.3.1 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.
- A 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**.
- 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
- Views are derived from BASE TABLES with SQL instructions like: [SELECT ... FROM... WHERE... ORDER BY]

2.1.1.3.2 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.
   Existence Dependency: the dependence of an entity on the existence of one or more entities.
   Weak entity: an entity that cannot 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.

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 (computed) from the values of other attributes.
E.g. Age (current year – year of birth)
Length of employment (current date- start 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 cannot be null.

Default value - assumed value if no explicit value

3. The **relationships** between entities which exist and must be taken into account when processing information. A ***relationship*** is an association between entity types. 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. For 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.

I. 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.
  E.g. Employee and Manager
- Binary relationships: tuples/records of two entities are associated in a relationship.
  E.g. Book and Publisher, Employee and Department
- Ternary relationship: tuples/records of three different entities are associated.
  E.g. Course, Student, Grade
- And a generalized one:
  - N-nary relationship: Tuples from arbitrary number of entity sets are participating in a relationship.

II. 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.

2.1.1.3.3 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.

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 types of Relations are Named and Unnamed Relations. The basic difference is on how the relation is created, used and updated:

1. **Base Relation**
   A ***Named Relation*** corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.
2. **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.

2.2 Schemas and Instances

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.

2.2.0 Schemas

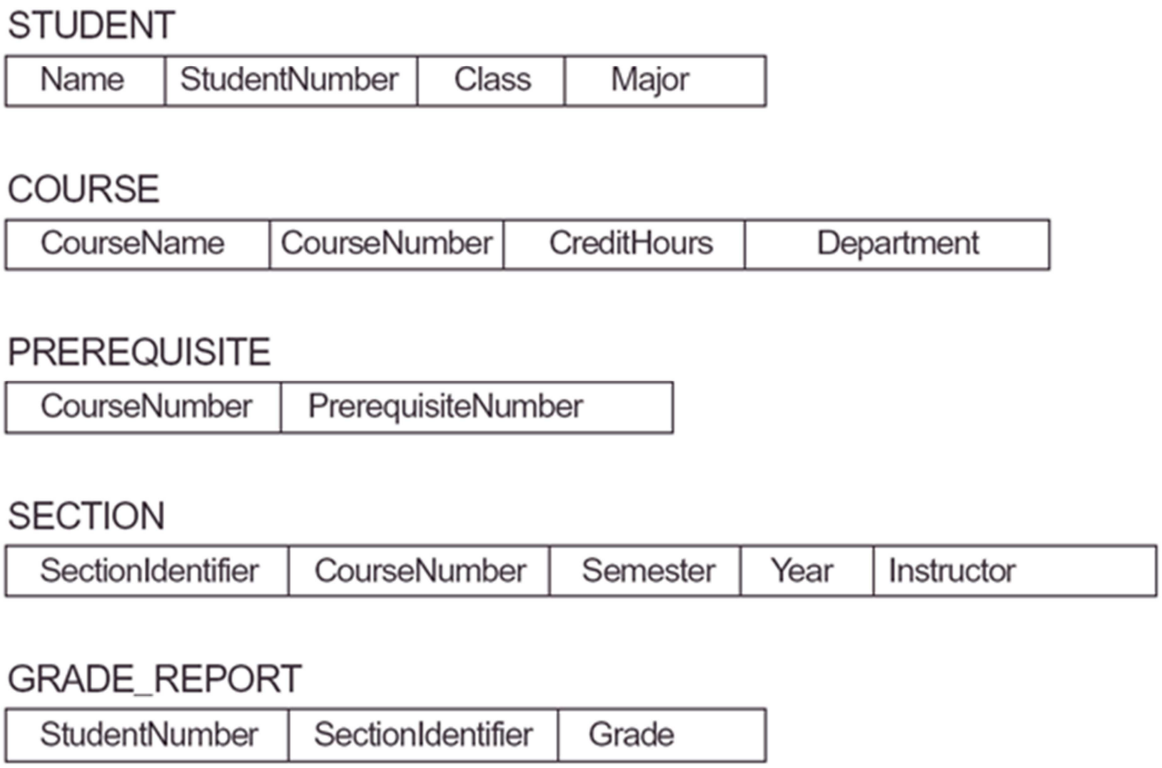
- Schema describes how data is to be structured, defined at setup/Design time (also called "metadata")
- Since it is used during the database development phase, there is rare tendency of changing the schema unless there is a need for system maintenance which demands change to the definition of a relation.

Database Schema (I**ntension**): 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)

2.2.1 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

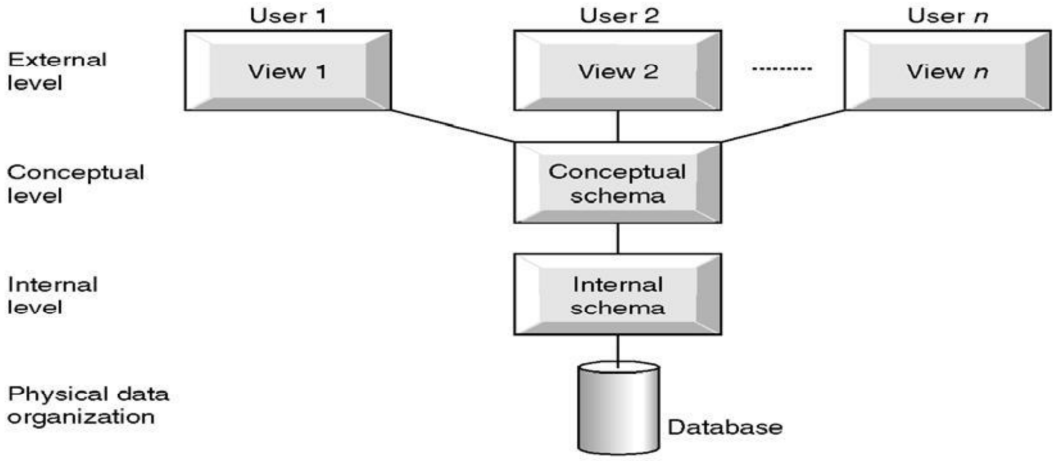
- Refers to the actual data in the database at a specific point in time
- State of database is changed any time we add, delete or update an item.
- **Valid state**: the state that satisfies the structure and constraints specified in the schema and is enforced by DBMS.

2.3 DBMS Architecture (ANSI-SPARC)

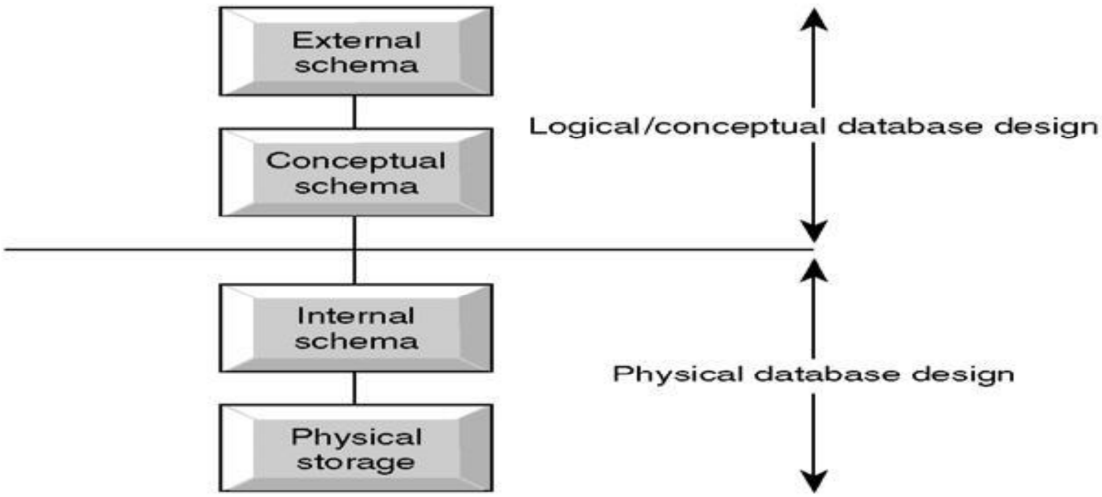
The purpose and origin of the Three-Level database architecture

- 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.



Three-level ANSI-SPARC Architecture of a Database



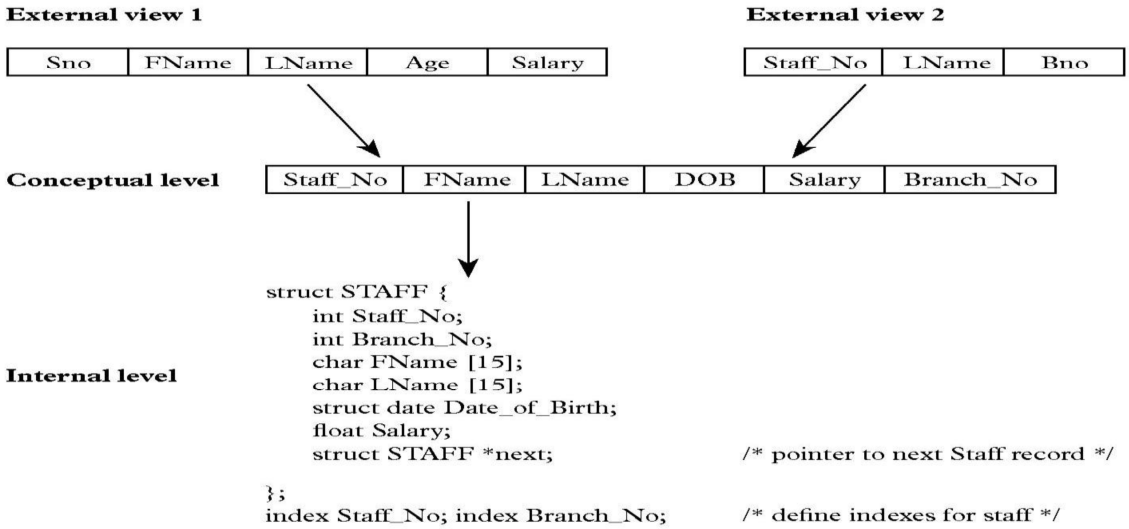
External Level: Users' view of the database. It describes that part of database that is relevant to a particular user. Different users have their own customized view of the database independent of other users.

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

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

The following example can be taken as an illustration for the difference between the three levels in the ANSI-SPARC database Architecture. Where:

- The first level is concerned about the group of users and their respective data requirement independent of the other.
- The second level is describing the whole content of the database where one piece of information will be represented once.
- The third level



2.4 Data Independence

Data independence is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Logical data independence: capacity to change the conceptual schema without having to change external schemas or application programs.

Physical data independence: capacity to change the internal schema without having to change the conceptual (or external) schemas.

2.5 DBMS Languages

Data Definition Language (DDL): Used by the DBA and database designers to specify the conceptual schema of a database. It is a language to specify conceptual and internal schemas for the database and any mappings between the two.

Storage definition language (SDL): used when clear distinction between conceptual and internal schema.

View definition language (VDL): specify user views and their mappings to the conceptual schema.

Data manipulation language (DML): retrieval, insertion, deletion, and modification of the data

Most DBMS's combine the various capabilities of the DDL, VDL, DML and SDL into a single high-level DML (e.g. SQL relational database language.)

There are two main types of DMLs:

1. A high-level or nonprocedural DML: specify complex DB operations.
   Example: SQL (set-at-a-time).
   Are set-oriented and specify what data to retrieve than how to retrieve. It also called declarative languages.
2. A low-level or procedural DML: retrieve individual records or objects from DB and process each separately (record-at-a-time). They specify *how* to retrieve data and include constructs such as looping.

2.6 DBMS Interfaces

Menu-Based Interfaces for Browsing

- menus leads to formulation of a request

Forms-Based Interfaces

- Display a form for each user (insert, select)
- Designed for naïve users.

Graphical User Interfaces (GUI)

- Display schema as diagram.
- Utilize both menu and forms.

Natural Language Interfaces

- Accept requests in native language and attempt to understand them.
- Refers to words in the schema and (standard words) to interpret the request.

Interfaces for Parametric Users (e.g. bank tellers) using function keys.

Interfaces for the DBA

- Creating accounts, system privileges, changing schema, etc.

2.7 The Database System Environment

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

2.7.1 DBMS Component Modules

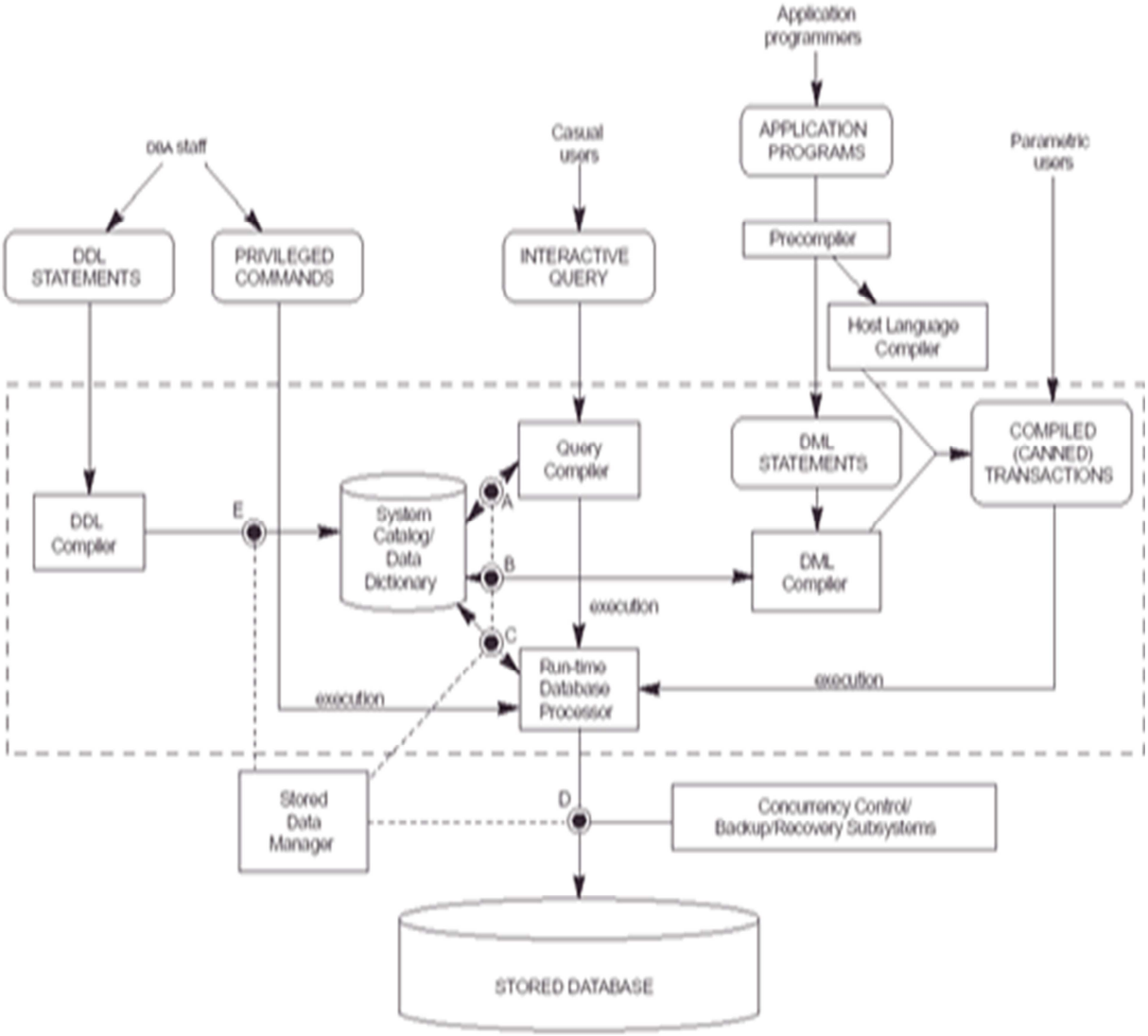


Figure 01

Figure 01 illustrates, in a simplified form, the typical DBMS components. The database and the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the **operating system (OS),** which schedules disk input/output. A higher-level **stored data manager** module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog. The dotted lines and circles marked A, B, C, D, and E in Figure 01 illustrates accesses that are under the control of this stored data manager. The stored data manager may use basic OS services for carrying out low-level data transfer between the disk and computer main storage, but it controls other aspects of data transfer, such as

handling buffers in main memory. Once the data is in main memory buffers, it can be processed by other DBMS modules, as well as by application programs.
The **DDL compiler** processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog. The catalog includes information such as the names of files, data items, storage details of each file, mapping information among schemas, and constraints, in addition to many other types of information that are needed by the DBMS modules. DBMS software modules then look up the catalog information as needed.

The **run-time database processor** handles database accesses at run time; it receives retrieval or update operations and carries them out on the database. Access to disk goes through the stored data manager. The **query compiler** handles high-level queries that are entered interactively. It parses, analyzes, and compiles or interprets a query by creating database access code, and then generates calls to the run-time processor for executing the code.

The **pre-compiler** extracts DML commands from an application program written in a host programming language. These commands are sent to the **DML compiler** for compilation into object code for database access. The rest of the program is sent to the host language compiler. The object codes for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.

Figure 01 is not meant to describe a specific DBMS; rather it illustrates typical DBMS modules. The DBMS interacts with the operating system when disk accesses—to the database or to the catalog—are needed. If the computer system is shared by many users, the OS will schedule DBMS disk access requests and DBMS processing along with other processes. The DBMS also interfaces with compilers for general-purpose host programming languages. User-friendly interfaces to the DBMS can be provided to help any of the user types shown in Figure 01 to specify their requests.

2.7.2 Database System Utilities

In addition to possessing the software modules just described, most DBMSs have **database utilities** that help the DBA in managing the database system. Common utilities have the following types of functions:

1. **Loading:** A loading utility is used to load existing data files—such as text files or sequential files—into the database.
2. **Backup:** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape.
3. **File reorganization:** This utility can be used to reorganize a database file into a different file organization to improve performance.
4. **Performance monitoring:** Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files to improve performance.

2.7.3 Tools, Application Environments, and Communications Facilities

- **CASE tools:** are used in the design phase of database systems.
- **data (information) repository:** store catalog info, design decisions, usage, application program description, user information
  - ✓ Useful in large organizations.

- **Application Developer:** e.g. power builder. Help in development of DB design, GUI, query, update etc.
- **Communications Software:** allow users remotely to access the DB.

2.8 Classification of DBMS

Several criteria are normally used to classify DBMSs.

- Data model:
  - ✓ Relational
  - ✓ Network
  - ✓ Hierarchical
  - ✓ Object-oriented
- Number of users supported by the system.
  - ✓ Single-user systems and Multiuser systems
- Number of sites over which the database is distributed.
  - ✓ Centralized, distributed DBMS (DDBMS), Homogeneous DDBMSs, federated DBMS (develop software to access several autonomous preexisting databases stored under heterogeneous DBMSs.)

Chapter three

3. Data Modelling using Entity Relationship Model

3.1 Database Design

Database design is the process of coming up with different kinds of specification for the data to be stored in the database. The database design part is one of the middle phases we have in information systems development where the system uses a database approach. Design is the part on which we would be engaged to describe how the data should be perceived at different levels and finally how it is going to be stored in a computer system. The prime interest of a database system will be the Design part which is divided into other three phases.

These phases are:

1. **Conceptual Design**
2. **Logical Design, and**
3. **Physical Design**

In developing a good design, one should answer such questions as:

- ✓ What are the relevant Entities for the Organization
- ✓ What are the important features of each Entity
- ✓ What are the important Relationships
- ✓ What are the important queries from the user
- ✓ What are the other requirements of the Organization and the Users

3.1.1 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.
- ❖ 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

3.1.2 The Entity Relationship (E-R) Model

Entity-Relationship modeling is used to represent conceptual view of the database.
The main components of ER Modeling 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

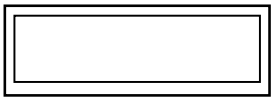
Developing an E-R Diagram

- 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.
- The process is repeated until the end users and designers agree that the E-R diagram is a fair representation of the organization's activities and functions.
  - Checking for Redundant Relationships in the ER Diagram. Relationships between entities indicate access from one entity to another - it is therefore possible to access one entity occurrence from another entity occurrence even if there are other entities and relationships that separate them - this is often referred to as *Navigation* of the ER diagram
  - The last phase in ER modeling is validating an ER Model against requirement of the user.

Graphical Representations in ER Diagramming

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

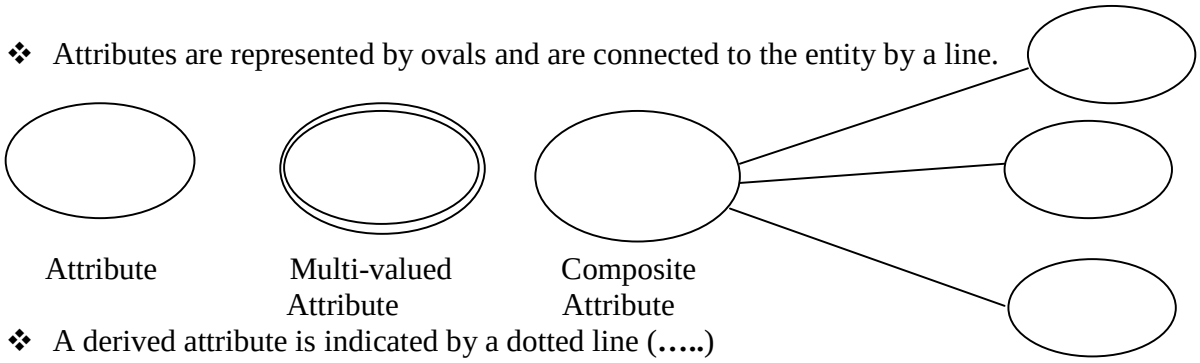
Strong Entity



Weak Entity

- Connected entities are called relation participant.

- Attributes are represented by ovals and are connected to the entity by a line.

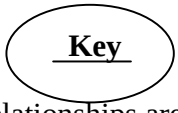


Attribute | Multi-valued Attribute | Composite Attribute

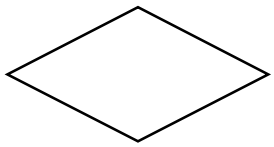
- A derived attribute is indicated by a dotted line (**…..**)

Derived attribute

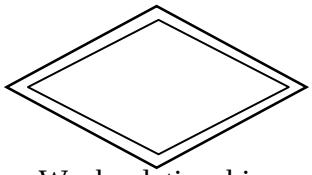
- Primary keys are underlined.



- 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.

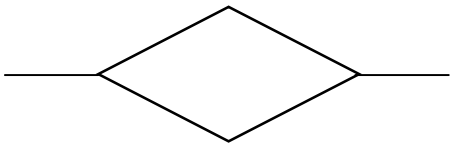


Strong relationship



Weak relationship

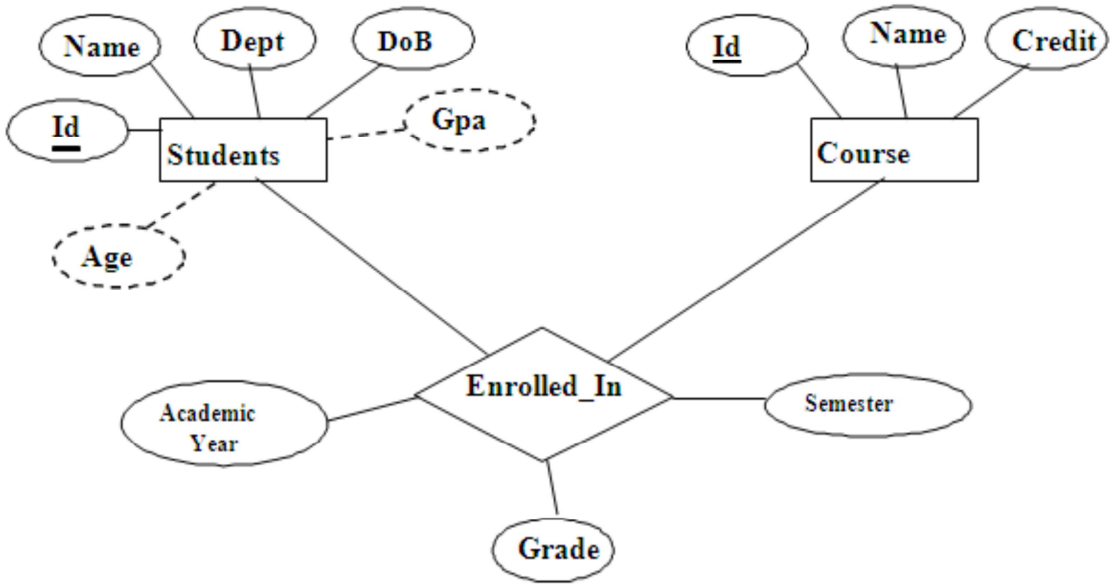
- Cardinality



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, Dep't, 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:

- A Personnel record management system will have the following two basic data object categories with their own features or properties:
- Employee will have an Id, Name, DoB, Age, Tel and Department will have an Id, Name, Location
- Whenever an Employee is assigned in one Department, the duration of his stay in the respective department should be registered.

Exercise: Build an ER Diagram for the following information:

The Company Employee Administrative Database (COMPANY) stores information about the employees, the departments and the projects of a company. The following data have been identified in the requirements collection and analysis phase and they are to be represented in the enterprise.

- The company is organized in to departments. Each department has a unique name, a unique number, and a particular employee who manages the department. The database keeps track of the start date when an employee began managing the department. A department may have several locations.
- A department controls a number of projects; each project has a unique name, a unique number, and a single location. The database also stores the number of work-hours budgeted for each project.
- The department stores each employee's name, social security number, address, salary, sex, birthdate, date hired and date he/she terminates employment. An employee is

assigned to one department but he/she may work on several projects, which are not necessarily controlled by the same department. The database also keeps track of the number of hours an employee already worked on each project, and the direct supervisor of each employee if he/she has one. (Note that an employee need not have an assigned supervisor.)

- For insurance purposes, the database keeps track of the dependents of each employee. Each dependent's name, sex, birthdate, and relationships to the employee is recorded in the database.

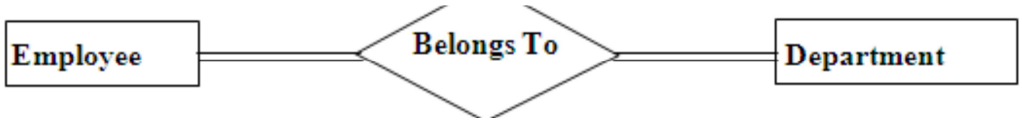
3.1.3 Participation of an Entity Set in a Relationship Set

Participation constraint of a relationship is involved in identifying and setting the mandatory or optional feature of an entity occurrence to take a role in a relationship. There are two distinct participation constraints with this respect, namely: ***Total Participation*** and ***Partial Participation***

Total participation: every tuple in the entity or relation participates in at least one relationship by taking a role. This means, every tuple in a relation will be attached with at least one other tuple. The entity with total participation in a relationship will be connected to the relationship using a double line.

Partial participation: some tuple in the entity or relation may not participate in the relationship. This means, there is at least one tuple from that Relation not taking any role in that specific relationship. The entity with partial participation in a relationship will be connected to the relationship using a single line.

E.g. 1: Participation of EMPLOYEE in "belongs to" relationship with DEPARTMENT is total since every employee should belong to a department. Participation of DEPARTMENT in "belongs to" relationship with EMPLOYEE is total since every department should have more than one employee.



E.g. 2: Participation of EMPLOYEE in "manages" relationship with DEPARTMENT is partial participation since not all employees are managers. Participation of DEPARTMENT in "Manages" relationship with EMPLOYEE is total since every department should have a manager.

Exercises

What is the cardinality and existence of each of the following relationships in just the direction given? State any assumptions you have to make.

1. Husband to wife
2. Student to degree
3. Child to parent
4. Player to team

Chapter four

4 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.

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.

4.1 Converting ER Diagram to Relational Tables

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.

4.2 Normalization

A relational database is merely a collection of data, organized in a particular manner. As the father of the relational database approach, **Codd** created a series of rules called ***normal forms*** that help define that organization.

One of the best ways to determine what information should be stored in a database is to clarify what questions will be asked of it and what data would be included in the answers.

Database normalization is a series of steps followed to obtain a database design that allows for consistent storage and efficient access of data in a relational database. These steps reduce data redundancy and the risk of data becoming inconsistent.

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.

All the normalization rules will eventually remove the update anomalies that may exist during data manipulation after the implementation.
The type of problems that could occur in insufficiently normalized table is called update anomalies which includes;

(1) 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.

(2) 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.

(3) 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.

Example of problems related with Anomalies

EmpID	*FName*	*LName*	*SkillID*	*Skill*	*SkillType*	*School*	*SchoolAdd*	*Skill Level*
12	Abebe	Mekuria	2	SQL	Database	AAU	Sidist_Kilo	5
16	Lemma	Alemu	5	C++	Programming	Unity	Gerji	6
28	Chane	Kebede	2	SQL	Database	AAU	Sidist_Kilo	10
25	Abera	Taye	6	VB6	Programming	Helico	Piazza	8
65	Almaz	Belay	2	SQL	Database	Helico	Piazza	9
24	Dereje	Tamiru	8	Oracle	Database	Unity	Gerji	5
51	Selam	Belay	4	Prolog	Programming	Jimma	Jimma City	8
94	Alem	Kebede	3	Cisco	Networking	AAU	Sidist_Kilo	7
18	Girma	Dereje	1	IP	Programming	Jimma	Jimma City	4
13	Yared	Gizaw	7	Java	Programming	AAU	Sidist_Kilo	6

Deletion Anomalies:
If employee with ***ID 16*** is deleted then ever information about skill ***C++*** and the type of skill is deleted from the database. Then we will not have any information about ***C++*** and its skill type.
Insertion Anomalies:
What if we have a new employee with a skill called ***Pascal***? We cannot decide whether ***Pascal*** is allowed as a value for skill and we have no clue about the ***type of skill*** that ***Pascal*** should be categorized as.
Modification Anomalies:
What if the address for ***Helico*** is changed from ***Piazza*** to ***Mexico***? We need to look for every occurrence of ***Helico*** and change the value of ***School_Add*** from ***Piazza*** to ***Mexico***, which is prone to error.

4.3 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.

The essence of this idea is that if the existence of something, call it A, implies that B must exist and have a certain value, and then we say that ***"B is functionally dependent on A."*** We also often express this idea by saying that "A determines B," or that "B is a function of A, "or that "A functionally governs B." Often, the notions of functionality and functional dependency are expressed briefly by the statement, "If A, then B." It is important to note that the value B must be *unique* for a given value of A, i.e., any given value of A must imply just one and only one value of B, in order for the relationship to qualify for the name "function." (However, this does not necessarily prevent different values of A from implying the same value of B.)

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

Dinner	**Type of Wine**
Meat	Red
Fish	White

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

Dinner	**Type of Wine**	**Types of Fork**
Meat	Red	Meat fork
Fish	White	Fish 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 →Wine
Dinner → 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} →C and B→C or A→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}→C and B→C and A→C** does not hold (if B cannot determine C and A cannot 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 **/→** A and C **/→** A)

In the normal notation:

{(A→B) AND **(B→C)}** ==> **A→C** provided that **B /→ A** and **C /→ A**

4.4 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.

Normalization towards a logical design consists of the following steps:

UnNormalized 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.

UNNORMALIZED

- A table that contains one or more repeating groups.
- A repeating group is a field or group of fields that hold multiple values for a single occurrence of a field.

EmpID	*FirstName*	*LastName*	*Skill*	*SkillType*	*School*	*SchoolAdd*	*SkillLevel*
12	Abebe	Mekuria	SQL, VB6	Database, Programming	AAU, Helico	Sidist_Kilo Piazza	5 8
16	Lemma	Alemu	C++ IP	Programming Programming	Unity Jimma	Gerji Jimma City	6 4
28	Chane	Kebede	SQL	Database	AAU	Sidist_Kilo	10
65	Almaz	Belay	SQL Prolog Java	Database Programming Programming	Helico Jimma AAU	Piazza Jimma City Sidist_Kilo	9 8 6
24	Dereje	Tamiru	Oracle	Database	Unity	Gerji	5
94	Alem	Kebede	Cisco	Networking	AAU	Sidist_Kilo	7

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 this repeating group 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)

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

EmpID	*FirstName*	*LastName*	*SkillID*	*Skill*	*SkillType*	*School*	*SchoolAdd*	*SkillLevel*
12	Abebe	Mekuria	1	SQL	Database	AAU	Sidist_Kilo	5
12	Abebe	Mekuria	3	VB6	Programming	Helico	Piazza	8
16	Lemma	Alemu	2	C++	Programming	Unity	Gerji	6
16	Lemma	Alemu	7	IP	Programming	Jimma	Jimma City	4
28	Chane	Kebede	1	SQL	Database	AAU	Sidist_Kilo	10
65	Almaz	Belay	1	SQL	Database	Helico	Piazza	9
65	Almaz	Belay	5	Prolog	Programming	Jimma	Jimma City	8
65	Almaz	Belay	8	Java	Programming	AAU	Sidist_Kilo	6
24	Dereje	Tamiru	4	Oracle	Database	Unity	Gerji	5
94	Alem	Kebede	6	Cisco	Networking	AAU	Sidist_Kilo	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 First Normal form (2NF)

EMP_PROJ

EmpID	EmpName	ProjNo	ProjName	ProjLoc	ProjFund	ProjMangID	Incentive

EMP_PROJ rearranged

EmpID	ProjNo	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}→EmpName, ProjName, ProjLoc, ProjFund, ProjMangID, Incentive

But in addition to this we have the following dependencies:

FD1: {EmpID}→EmpName
FD2: {ProjNo}→ProjName, ProjLoc, ProjFund, ProjMangID
FD3: {EmpID, ProjNo}→ 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

EmpID	EmpName

PROJECT

ProjNo	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 First Normal form (3NF)

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

STUDENT

StudID	*Stud_F_Name*	*Stud_L_Name*	*Dept*	*Year*	*Dormitary*
125/97	Abebe	Mekuria	*Info Sc*	*1*	*401*
654/95	Lemma	Alemu	*Geog*	*3*	*403*
842/95	Chane	Kebede	*CompSc*	*3*	*403*
165/97	Alem	Kebede	*InfoSc*	*1*	*401*
985/95	Almaz	Belay	*Geog*	*3*	*403*

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

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

StudID→Year AND **Year→Dormitary**

And **Year** cannot determine **StudID** and **Dormitary** cannot determine **StudID**

Then transitively **StudID→Dormitary**

- 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

StudID	*Stud F_Name*	*Stud L_Name*	*Dept*	*Year*
125/97	Abebe	Mekuria	*Info Sc*	*1*
654/95	Lemma	Alemu	*Geog*	*3*
842/95	Chane	Kebede	*CompSc*	*3*
165/97	Alem	Kebede	*InfoSc*	*1*
985/95	Almaz	Belay	*Geog*	*3*

DORM

Year	*Dormitary*
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.*

Other Levels of Normalization

1. Boyce-Codd Normal Form (BCNF)
2. Forth Normal form (4NF)
3. Fifth Normal Form (5NF)
4. Domain-Key Normal Form (DKNF)

Graphical illustration of different phases of normalization