UNIT 01:

Purpose of DBMS - Applications - Views of data - Data Abstraction - Instances and Schemas – Data Models - Database Design - Relational Databases - Database Architecture - Database users and administrators - History of Database systems. Entity- Relationship (E-R) Model: Basic concepts - Constraints - E-R Diagram - Weak Entity Sets - Reduction of Relational schemas.

Introduction to Database Management System

As the name suggests, the database management system consists of two parts. They are:

1. Database and

2. Management System

What is a Database?

To find out what database is, we have to start from data, which is the basic building block of any DBMS.

Data: Facts, figures, statistics etc. having no particular meaning (e.g. 1, ABC, 19 etc).

Record: Collection of related data items, e.g. in the above example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll No	Name	Age
1	ABC	19

Table or Relation: Collection of related records.

The **columns** of this relation are called **Fields**, Attributes or Domains. The **rows** are called **Tuples** or Records.

Database: Collection of related relations. Consider the following collection of tables:

T1

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

T2

Roll	Address
1	KOL
2	DEL
3	MUM

T3

Roll	Year
1	I
2	II
3	I

T4

Year	Hostel
I	H1
II	H2

We now have a collection of 4 tables. They can be called a “related collection” because we can clearly find out that there are some common attributes existing in a selected pair of tables. Because of these common attributes we may combine the data of two or more tables together to find out the complete details of a student. **Questions like “Which hostel does the youngest student live in?”**

Age and **Hostel** attributes are in different tables.

A database in a DBMS could be viewed by lots of different people with different responsibilities.

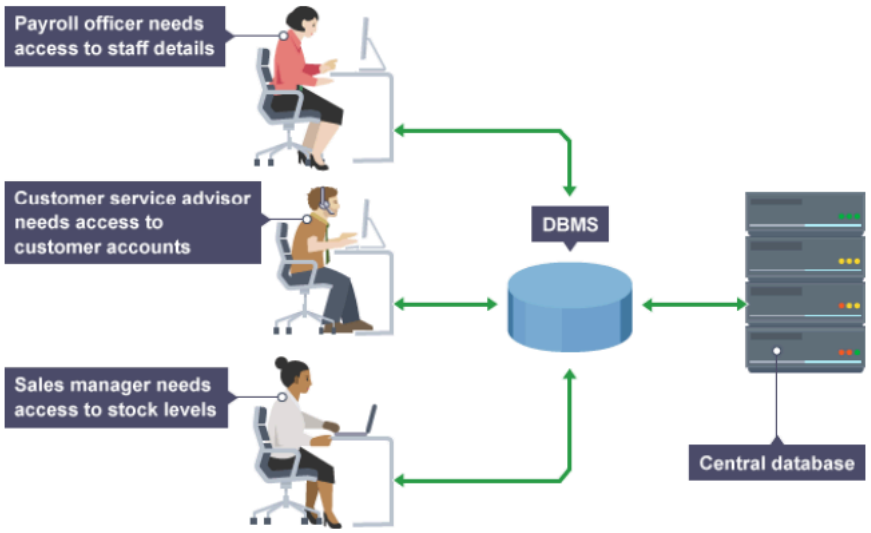


Figure 1.1: Empolyees are accessing Data through DBMS

For example, within a company there are different departments, as well as customers, who each need to see different kinds of data. Each employee in the company will have different levels of access to the database with their own customized front-end application.

In a database, data is organized strictly in row and column format. The rows are called Tuple or Record. The data items within one row may belong to different data types. On the other hand, the columns are often called Domain or Attribute. All the data items within a single attribute are of the same data type.

What is Management System?

- **A database-management system (DBMS)** is a collection of **interrelated data** and a set of programs to access those data. This is a collection of related data with an implicit meaning and hence is a database. The collection of data, usually referred to as the **database, contains information relevant to an enterprise.** The primary goal of a DBMS is to provide a way to **store and retrieve database information** that is both **convenient and efficient**. By data, we mean known facts that can be recorded and that have implicit meaning.
- Database systems are designed to **manage large bodies of information.** Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at

unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

Database Management System (DBMS) and Its Applications:

A Database management system is a computerized record-keeping system. It is a repository or a container for collection of computerized data files. The overall purpose of DBMS is to allow the users to define, store, retrieve and update the information contained in the database on demand. Information can be anything that is of significance to an individual or organization.

Databases touch all aspects of our lives. Some of the major areas of application are as follows:

1. Banking
2. Airlines
3. Universities
4. Manufacturing and selling
5. Human resources

Enterprise Information

- **Sales:** For customer, product, and purchase information.
- **Accounting**: For payments, receipts, account balances, assets and other accounting information.
- **Human resources**: For information about employees, salaries, payroll taxes, and benefits, and for generationof paychecks.
- **Manufacturing:** For management of the supply chain and for tracking production of items in factories,inventories of items inwarehouses and stores, and orders for items.
- **Online retailers:** For sales data noted above plus online order tracking,generation of recommendation lists,and maintenance of online product evaluations.

Banking and Finance

- **Banking:** For customer information, accounts, loans, and banking transactions.
- **Credit card transactions: For** purchases on credit cards and generation of monthly statements.
- **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
    - **Universities:** For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
    - **Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
    - **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Purpose of Database Systems

Database systems arose in response to early methods of computerized management of commercial data. As an example of such methods, typical of the 1960s, consider part of a

university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information,the system has a number of application programs that manipulate the files, including programs to:

- Add new students, instructors, and courses
- Register students for courses and generate class rosters
- Assign grades to students, compute grade point averages (GPA), and generate transcripts

System programmers wrote these application programs to meet the needs of the university.New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science). As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

This typical **file-processing system** is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced,organizations usually stored information in such systems. Keeping organizational information in a fileprocessing system has a number of major disadvantages:

Data redundancy and inconsistency. Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files).

For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system.

Difficulty in accessing data. Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all students.

The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same clerk

needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

Data isolation. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

Integrity problems. The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

Atomicity problems. A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.

Consider a program to transfer $500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur.

That is, the funds transfer must be atomic—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

Concurrent-access anomalies. For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. Consider department A, with an account balance of $10,000. If two department clerks debit the account balance (by say $500 and $100, respectively) of department A at almost exactly the same time, the result of the concurrent executions may leave the budget in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $10,000, and write back $9500 and $9900, respectively. Depending on which one writes the value last, the account balance of department A may contain either $9500 or $9900, rather than the correct value of $9400. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data

may be accessed by many different application programs that have not been coordinated previously.

As another example, suppose a registration program maintains a count of students registered for a course, in order to enforce limits on the number of students registered. When a student registers, the program reads the current count for the courses, verifies that the count is not already at the limit, adds one to the count, and stores the count back in the database. Suppose two students register concurrently, with the count at (say) 39. The two program executions may both read the value 39, and both would then write back 40, leading to an incorrect increase of only 1, even though two students successfully registered for the course and the count should be 41.

Furthermore, suppose the course registration limit was 40; in the above case both students would be able to register, leading to a violation of the limit of 40 students.

Security problems. Not every user of the database system should be able to access all the data. For example,in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

These difficulties, among others, prompted the development of database systems. In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems.

Advantages of DBMS:

Controlling of Redundancy: Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.

Improved Data Sharing : DBMS allows a user to share the data in any number of application programs.

Data Integrity : Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database. For example: in customer database we can can enforce an integrity that it must accept the customer only from Noida and Meerut city.

Security : Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.

Data Consistency : By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.

Efficient Data Access : In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing Enforcements of Standards : With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.

Data Independence : Ina database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the meta data obtained by the DBMS is changed but the DBMS is continues to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.

Reduced Application Development and Maintenance Time : DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

Disadvantages of DBMS

1) It is bit complex. Since it supports multiple functionality to give the user the best, the underlying software has become complex. The designers and developers should have thorough knowledge about the software to get the most out of it.

2) Because of its complexity and functionality, it uses large amount of memory. It also needs large memory to run efficiently.

3) DBMS system works on the centralized system, i.e.; all the users from all over the world access this database. Hence any failure of the DBMS, will impact all the users.

4) DBMS is generalized software, i.e.; it is written work on the entire systems rather specific one. Hence some of the application will run slow.

View of Data

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

Data Abstraction

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

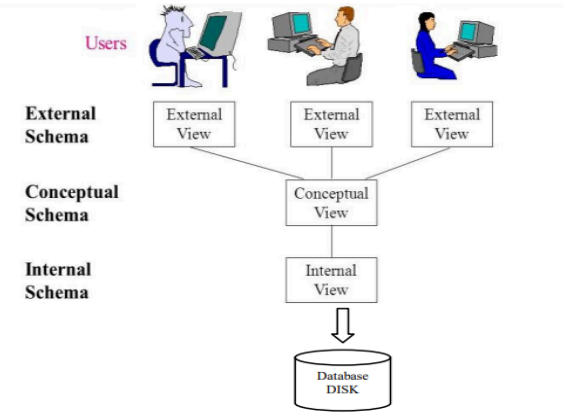


Figure 1.2 : Levels of Abstraction in a DBMS

• **Physical level (or Internal View / Schema):** The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

• **Logical level (or Conceptual View / Schema):** The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

• **View level (or External View / Schema):** The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database. Figure 1.2 shows the relationship among the three levels of abstraction. An analogy to the concept of data types in programming languages may clarify the distinction among levels of abstraction. Many high-level programming languages support the notion of a structured type. For example, we may describe a record as follows:

type instructor = record

ID : char (5);

name : char (20);

dept name : char (20);

salary : numeric (8,2);

end;

This code defines a new record type called instructor with four fields. Each field has a name and a type associated with it. A university organization may have several such record types, including Database DISK

• department, with fields dept_name, building, and budget

• course, with fields course_id, title, dept_name, and credits

• student, with fields ID, name, dept_name, and tot_cred

At the physical level, an instructor, department, or student record can be described as a block of consecutive storage locations. The compiler hides this level of detail from programmers. Similarly, the database system hides many of the lowest-level storage details from database programmers. Database administrators, on the other hand, may be aware of certain details of the physical organization of the data.

At the logical level, each such record is described by a type definition, as in the previous code segment, and the interrelationship of these record types is defined as well. Programmers using a programming language work at this level of abstraction. Similarly, database administrators usually work at this level of abstraction. Finally, at the view level, computer users see a set of application programs that hide details of the data types.

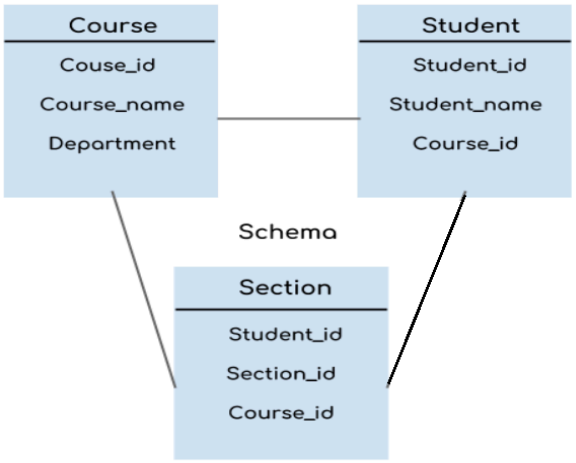
At the view level, several views of the database are defined, and a database user sees some or all of these views. In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing certain parts of the database. For example, clerks in the university registrar office can see only that part of the database that has information about students; they cannot access information about salaries of instructors.

Instances

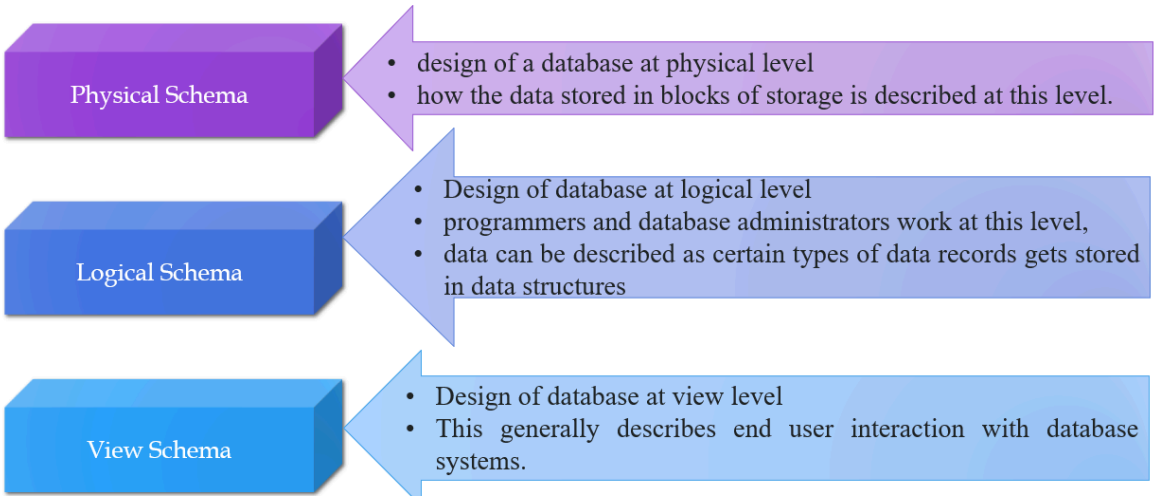
- Databases change over time as information is inserted and deleted.
- The collection of information stored in the database at a particular moment is called an instance of the database.
- The overall design of the database is called the database schema.

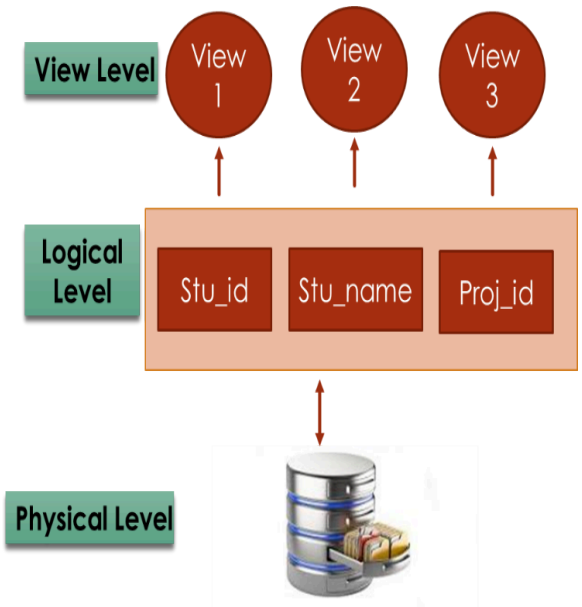
Schemas

- Design of a database is called the schema
- Shows the design of the database, it doesn't show the data present in those tables
- **For example**



Schema Architecture





- Records of each student which each user sees view level of information
- Columns, their data types, their mapping and constraints logical level of information
- Actual structure of table and data stored in the server's memory physical level

Instance

- Data stored in database at a particular moment of time is called instance of database

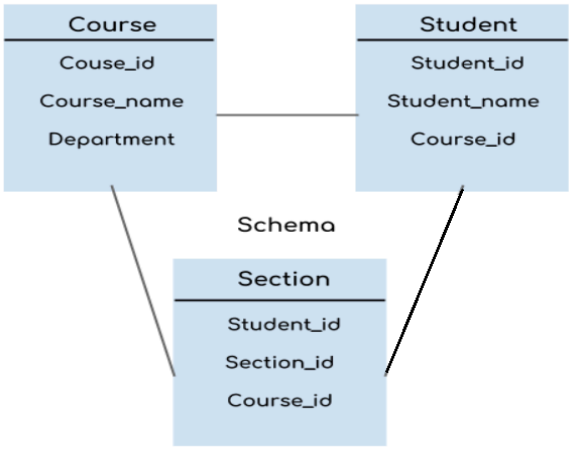
  For Example:

  - Course_id :20
  - Course_name: DBMS
  - Department : CSE

- Value of these variables at a moment of time is called the instance of that database

  For Example:

  - Today, table has 100 records
  - Tomorrow, Adding another 100 records, so instance of database tomorrow will have 200 records



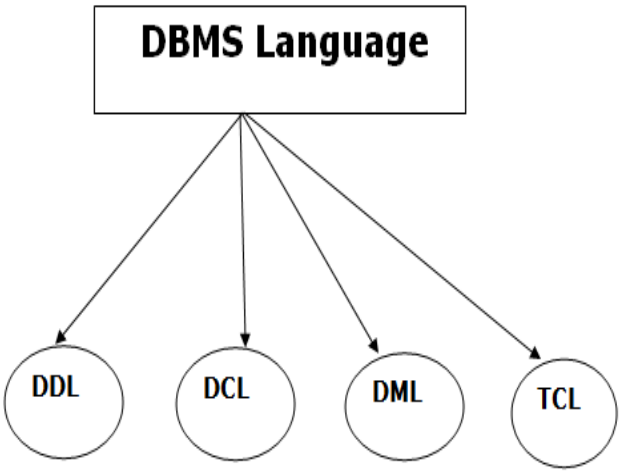
Data Models

The structure of a database is the data model: a collection of conceptual tools for describing data,data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at the physical, logical, and view levels.

The data models can be classified into four different categories:

- **Relational Model.** The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-basedmodel. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.
- **Entity-Relationship Model.** The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. The entityrelationship model is widely used in database design.
- **Object-Based Data Model.** Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.
- **Semi-structured Data Model.** The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes.
- **The Extensible Markup Language (XML)** is widely used to represent semi-structured data.
- **Database Languages** A database system provides a data-definition language to specify the database schema and a data-manipulation language to express database

queries and updates. In practice, the data definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.



Database Definition Language (DDL)

- DDL stands for Data Definition Language. It is used to define database structure or pattern.
- It is used to create schema, tables, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

Create: It is used to create tables in the database.

Alter: It is used to alter the structure of the database.

Drop: It is used to delete tables from the database**.**

Truncate: It is used to remove all records from a table.

Rename: It is used to rename an object.

Database Manipulation Language (DML)

- DML stands for Data Manipulation Language. It is used for accessing and manipulating data in a database. It handles user requests.

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.

Data Control Language (DCL)

- **DCL** stands for **D**ata **C**ontrol **L**anguage. DCL is used for **granting and revoking user access** on a database.
- **Here are some tasks that come under DCL:**
- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

Transaction Control Language (TCL)

- TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.
- **Here are some tasks that come under TCL:**
- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit

Database Design

- Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of data.
- Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space.
- It helps produce database systems

a. That meet the requirements of the users

b. Have high performance.

c. **Two Types of Database Techniques:**

    a. ER Modeling

    b. Normalization

ER Modeling

- **Entity Relationship Model** (ER Modeling) is a **graphical approach** to database design.

- It is a high-level data model that **defines data elements and their relationship** for a specified software system.
- An ER diagram is used to represent **real-world objects**.
- **Design or blueprint** of a database that can later be implemented as a database
- Helps you to analyze data requirements systematically to produce a well-designed database
- Represents real-world entities and the relationship between them.
- Considered as a **best practice** to complete **ER modeling before implementing your database.**
- **Steps to Create an ERD**

Components of ER diagram

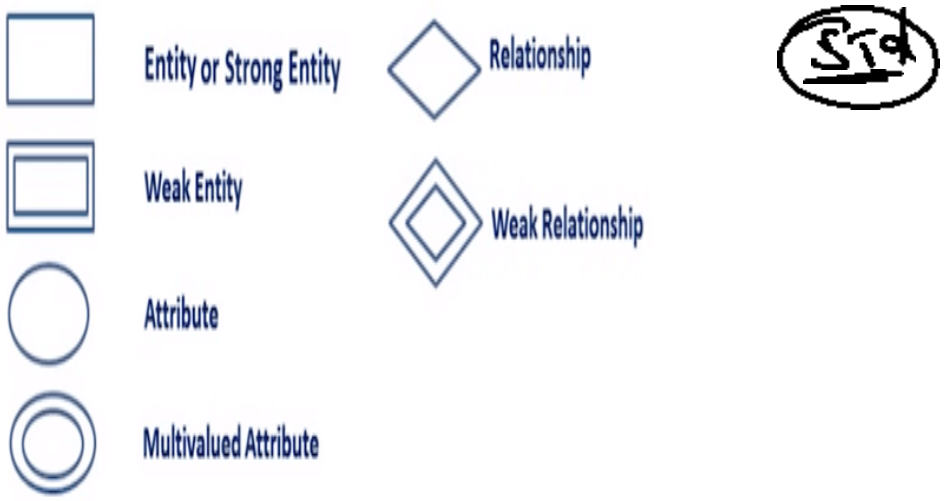
ER diagram depends on below three concepts:

- Entity
- Attributes
- Relationships

ER Diagram

Symbols in ER Diagrams:

- Rectangles: This Entity Relationship Diagram symbol represents entity types
- Ellipses : Symbol represent attributes
- Diamonds: This symbol represents relationship types
- Lines: It links attributes to entity types and entity types with other relationship types
- Primary key: attributes are underlined
- Double Ellipses: Represent multi-valued attributes



Normalization

- Another method for designing a relational database is to use a process commonly known as normalization.
- The goal is to generate a set of relation schemas that allows us to store information without unnecessary redundancy.
- It also allows us to retrieve information easily. The approach is to design schemas that are in an appropriate normal form.
- **Normal Forms in SQL:**

a. 1NF (First Normal Form)

b. 2NF (Second Normal Form)

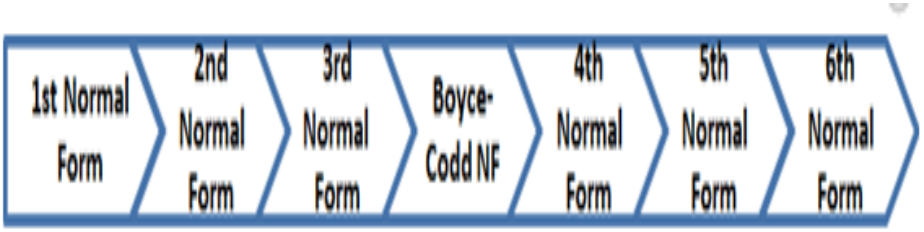
c. 3NF (Third Normal Form)

d. BCNF (Boyce-Codd Normal Form)

e. 4NF (Fourth Normal Form)

f. 5NF (Fifth Normal Form)

g. 6NF (Sixth Normal Form) Still in construction phase



Relational Database

- **RDBMS** stands for relational database management system. A relational model can be represented as a table of rows and columns.
- **A relational database has following major components:**

1. Table
2. Record or Tuple
3. Field or Column name or Attribute
4. Domain
5. Instance
6. Schema
7. Keys

- A table is a collection of data represented in rows and columns. Each table has a name in database. For example, the following table "STUDENT" stores the information of students in database.

Table: STUDENT

Student_Id	Student_Name	Student_Addr	Student_Age
101	Chaitanya	Dayal Bagh, Agra	27
102	Ajeet	Delhi	26
103	Rahul	Gurgaon	24
104	Shubham	Chennai	25

- Each row of a table is known as record. It is also known as tuple. For example, the following row is a record that we have taken from the above table.

102	Ajeet	Delhi	26

- In the table "STUDENT" has four fields (or attributes): Student_Id, Student_Name, Student_Addr & Student_Age.

Table: STUDENT

Student_Id	Student_Name	Student_Addr	Student_Age
101	Chaitanya	Dayal Bagh, Agra	27
102	Ajeet	Delhi	26
103	Rahul	Gurgaon	24
104	Shubham	Chennai	25

RDBMS -Domain

- A domain is a set of permitted values for an attribute in table. For example, a domain of month-of-year can accept January, February,…December as values, a domain of dates can accept all possible valid dates etc. We specify domain of attribute while creating a table. An attribute cannot accept values that are outside of their domains.
- For example, In the above table "STUDENT", the Student_Id field has integer domain so that field cannot accept values that are not integers for example, Student_Id cannot has values like, "First", 10.11 etc.

RDBMS – Instance And Schema

- Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.
- The data stored in database at a particular moment of time is called instance of database.

RDBMS – Keys

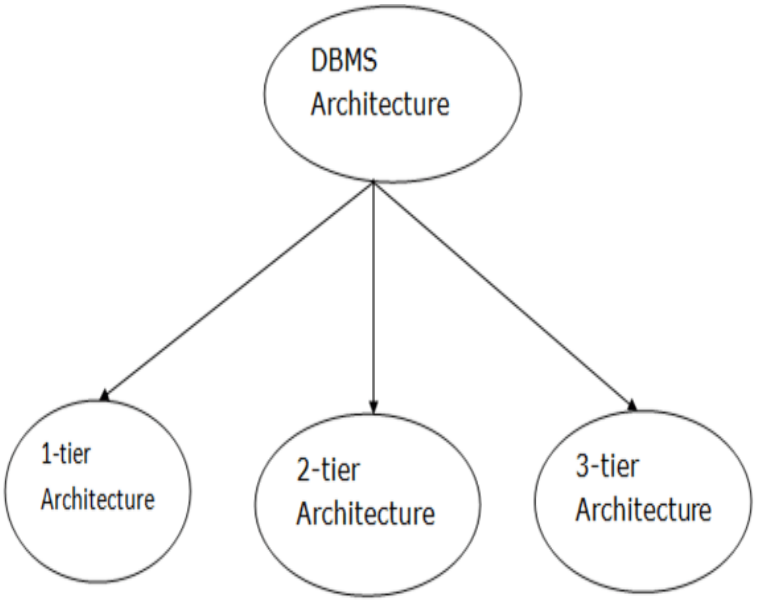
- **Key** plays an important role in relational database; it is used for **identifying unique rows from table**. It also **establishes relationship** among tables.

  Types of keys in DBMS

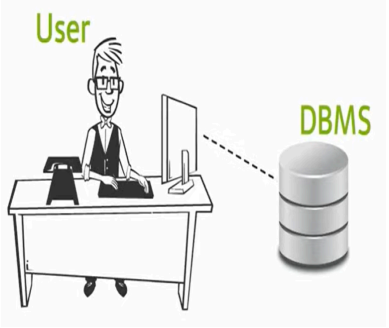
  1. Primary Key
  2. Foreign Key
  3. Super Key
  4. Composite Key
  5. Candidate Key

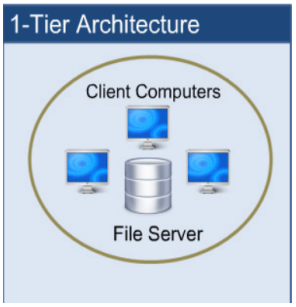
Database Architecture

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.



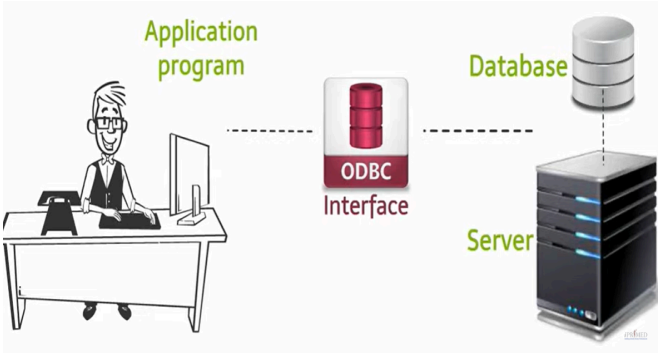
1 – tier



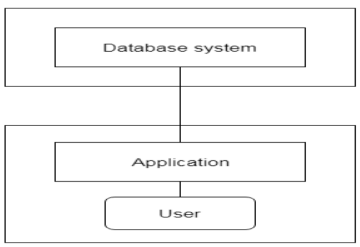


- Involves putting all of the required components for a software application or technology on a single server or platform
- Database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.
- These are centralized systems

2 – tier

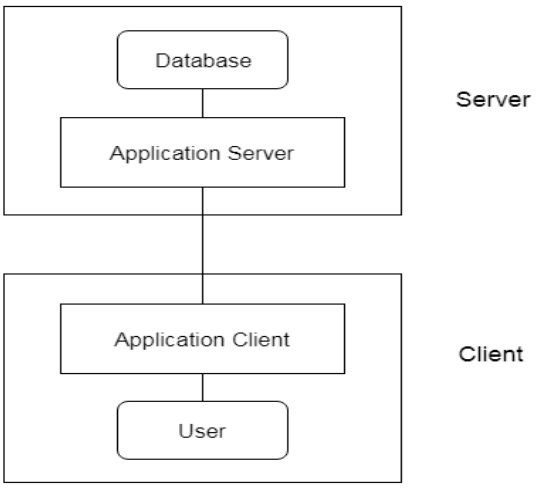


- The two-tier is based on Client Server architecture.
- The direct communication takes place between client and server. There is no intermediate between client and server.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.

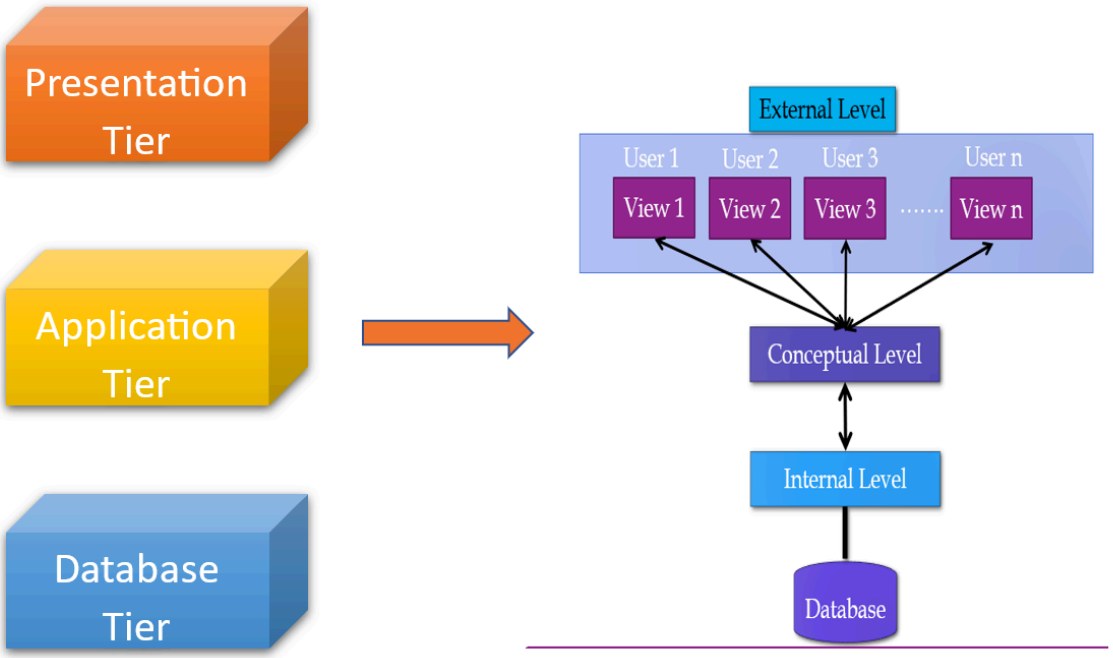


3 – tier

- 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.



Three levels of DBMS Architecture



Internal Level or physical level:

- Physical Storage structure of Database
- This level describes how the data is actually stored in the storage devices.
- This level is also responsible for allocating space to the data.
- This is the lowest level of the architecture.

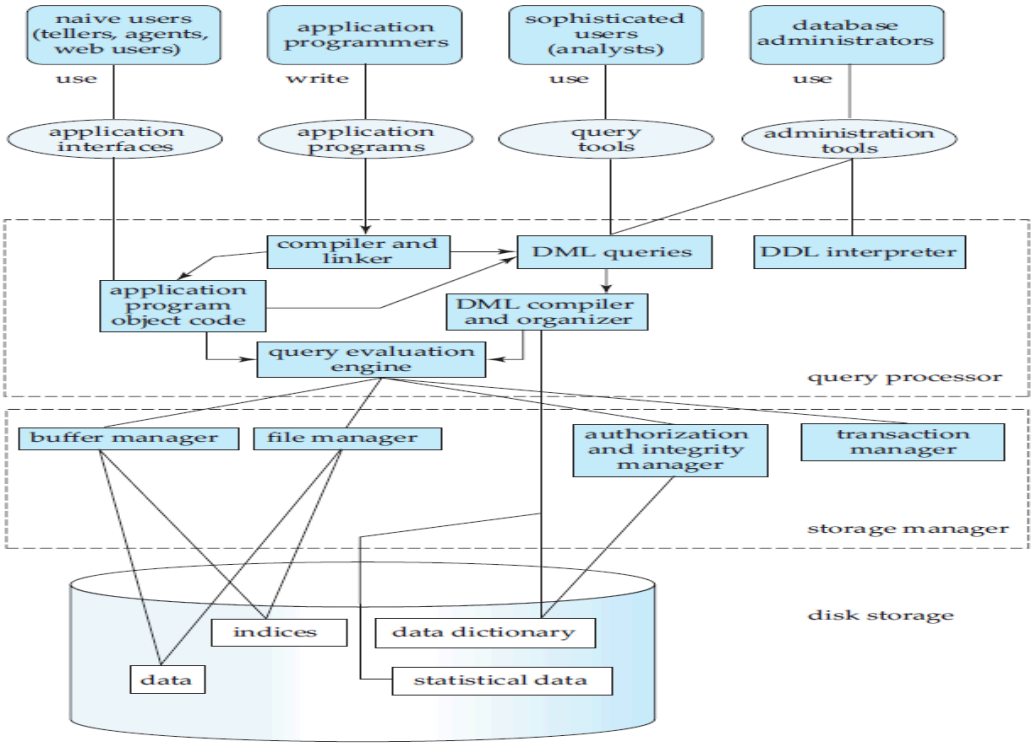
Conceptual Level or logical level:

- The whole design of the database such as relationship among data, schema of data etc. are described in this level.
- Database constraints and security are also implemented in this level of architecture.
- Programmers and Database administrators work at this level

External Level or view level:

- Several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping
- User doesn't need to know the database schema details such as data structure, table definition etc.
- User is only concerned about data which is what returned back to the view level after it has been fetched from database

Architecture



Database Engine

- A database engine (or storage engine) is the underlying software component that a database management system (DBMS) uses to create, update, read and delete (CURD) data from a database.
- The functional components of a database system can be broadly divided into two components:
    a. The storage manager
    b. The query processor

Storage Manager

- The storage manager is responsible for **storing, retrieving, and updating** data in the database.
- **The storage manager components include:**
    - ❖ Authorization and integrity manager
    - ❖ Transaction manager
    - ❖ File manager
    - ❖ Buffer manager

- **Authorization and integrity manager:** which tests for the satisfaction of integrity constraints and **checks the authority of users to access data.**
- **Transaction manager:** which ensures that the database remains in a consistent (correct) state despite system failures, and that **concurrent transaction executions proceed without conflicting.**
- **File manager:** which manages the **allocation of space on disks** to rage and the data structures used to represent information stored on disk.
- **Buffer manager:** which is responsible for **fetching data from disk storage** into **main memory**, and deciding what data to cache in main memory. The buffer manager is a **critical part** of the database system, since it enables the database to **handle data sizes that are much larger than the size of main memory.**

The Query Processor

- As query is very much necessary to find out only the data user need from tons of data of the database, query processor is very important to process these query requests.

  The query processor components include:

  DDL interpreter: which interprets DDL statements and records the definitions in the data dictionary.

  DML compiler: which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

  Query evaluation engine: which executes low-level instructions generated by the DML compiler.

Database Users and Administrators

Database users

- There are four different types of database – system users, differentiated by the way they expect to interact with the system.
- Different types of user interfaces have been designed for the different types of users.

  i. Naïve Users
  ii. Application programmers
  iii. Sophisticated users
  iv. Specialized users

1. Naïve Users :

- Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results.

- This type of user does not need to be aware of the presence of the database system. They interact with the database only with the help of the application interface. They are just accessing the application whatever they are using.

EXAMPLES:

1. The person who is working in the bank will check and tell the current balance after the transaction.
2. Reservation clerks for airlines, railways, hotels etc.,. they will check the current availability and make reservation based upon our request.
3. Ticket booking users are naïve users.

2.Application Programmers:

- Application programmers are computer professionals (back end programmers) who write application programs. These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc and reports with minimal programming effort. Application programmers can choose from many tools to develop user interfaces.

3. Sophisticated users :

- Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. They can develop their own database applications according to their requirement. They don't write the program code but they interact the database by writing SQL queries directly through the query processor

4. Specialized users :

- Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. They are the developers who develop the complex programs to the requirement. These applications store data with complex datatypes (for example: graphics data and audio data) and environment-modeling systems.

Database Administrators

- Database Administrator (DBA) is a **person/team** who defines the schema and also controls the 3 levels of database. The DBA will then create a **new account id and password** for the user if he/she need to access the database.
- DBA is also responsible for **providing security** to the database and he allows only the authorized users to access/modify the database.
- DBA also monitors the **recovery and back up** and provide technical support.

- The DBA has a **DBA account** in the DBMS which called a **system or super user account.**
- DBA **repairs damage** caused due to **hardware and/or software failures.**

DBA – Functions

1. Schema Definition:

The DBA define the logical Schema of the database. A Schema refers to the overall logical structure of the database.

According to this schema, database will be developed to store required data for an organization.

2. Storage Structure and Access Method Definition:

The DBA decides how the data is to be represented in the stored database.

3. Assisting Application Programmers:

The DBA provides assistance to application programmers to develop application programs.

4. Physical Organization Modification:

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

5. Approving Data Access:

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

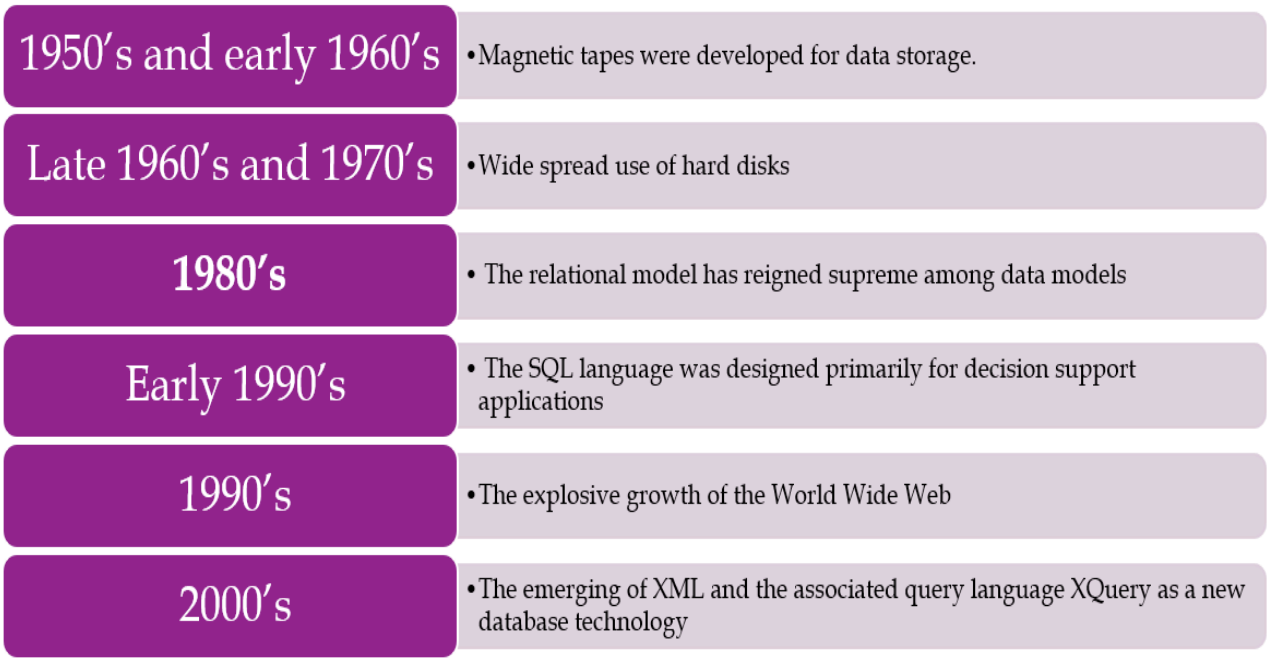
6. Monitoring Performance:

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

7. Backup and Recovery:

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

History of Database System

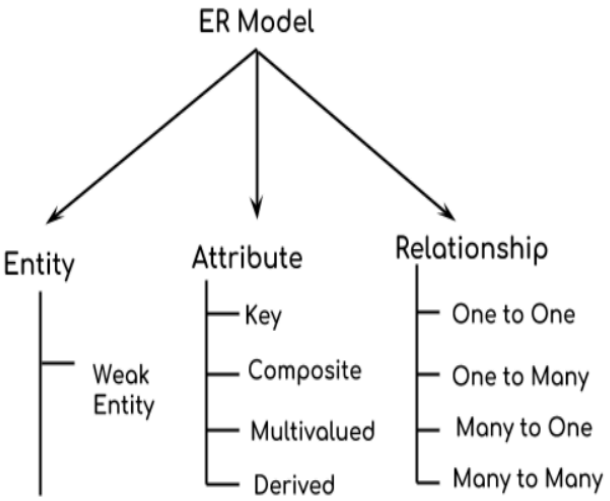


ENTITY – RELATIONSHIP MODEL

Components of ER Diagram

ER diagram depends on below three concepts:

- Entity
- Attributes
- Relationships



Entities (Table)

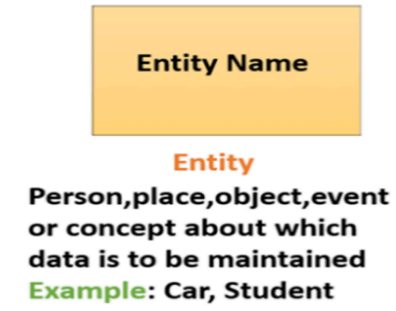
- Entity – basic object of the E-R model
  - Represented as rectangle in an ER diagram
  - Represents a “thing” with an independent existence
  - Can exist physically or conceptually
    - a professor, a student, a course

Entities

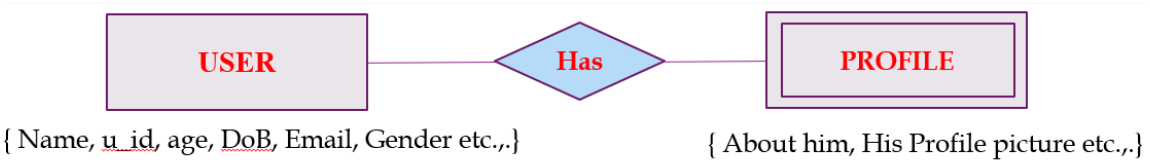
Examples of entities:

- Person: Employee, Student, Patient
- Place: Store, Building
- Object: Machine, product, and Car
- Event: Sale, Registration, Renewal

- Concept: Account, Course



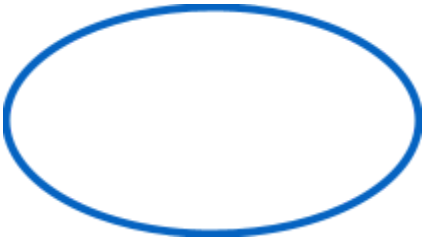
Weak Entities



- Profile belongs to the user, it cannot independently exist without an user.
- With no user there will be no profile.
- Weak entity will not have any primary key of its own.
- Weak entities always depend on other entities.
- Profile only exist because of user, so u_id will be referred in profile.
- An weak entity do not have key attribute of their own so that cannot be uniquely identified.
- Weak entities cannot exist without another relationship with other entity
- It is represented by a double rectangle
- A partial key is the portion of the key that comes from the weak entity. The rest of the key comes from the other entity in the relationship.
- Weak entities *always* have total participation as they cannot exist without the identifying relationship.

Attributes (Column)

- Each entity has a set of associated properties that describes the entity. These properties are known as ***attributes***.
- Attributes can be:
  - Simple or Composite



- Single or Multi-valued
- Stored or Derived
- NULL

Simple Attribute:

- Can't be divided any further
- Cannot be divided into subparts
- Atomic value

For example:

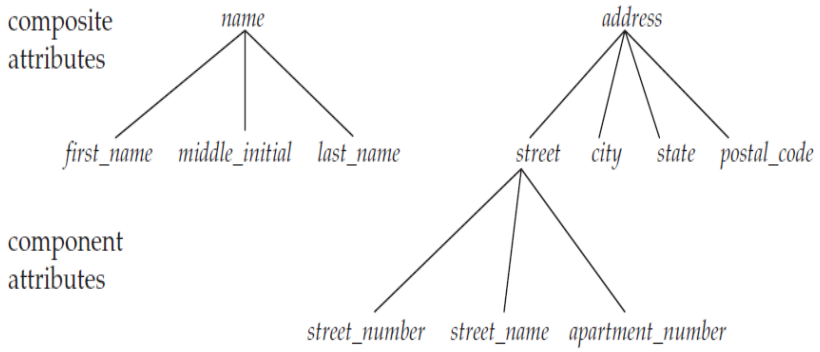
- Student roll number
- Bank account number

Attributes : Composite Attribute

Composite attribute:

- An attribute **composed of many other attribute** is called as composite attribute
- They are further **divided in a tree like structure.** Every node is then connected to its attribute

For example: student address



Attributes – Single & Multivalued attributes

Single-valued attribute:

Holds a single value for a particular entity

For example:

- Student : roll no

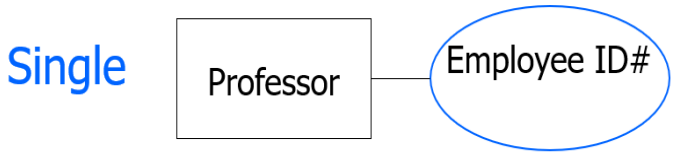
Multi-valued attribute

Multivalued attributes can have more than one values

Multivalued attributes are depicted by double ellipse.

For example:

- Mobile Number
- Email address



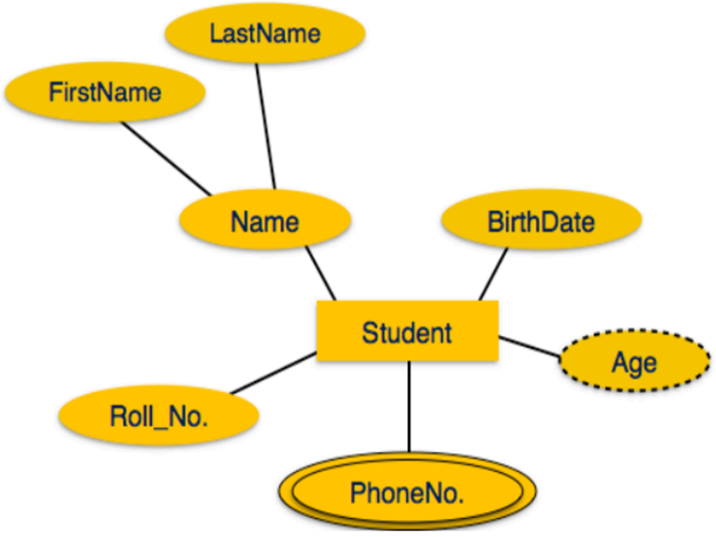
Attributes : Derived Attributes

Derived attribute:

- The value for this type of attribute can be derived from the values of other related attributes or entities
- Derived attributes are depicted by dashed ellipse.

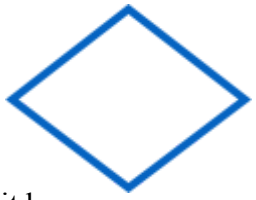
For example:

- Age (can be derived from DOB)



Relationships

- Shows the relationship among entities
- Association among two or more entities
- We can often identify relationships with verbs or verb phrases
- Represented using diamond shape
- All the entities (rectangles) participating in a relationship, are connected to it by a line.



limited by:

  - Participation
  - Cardinality Ratio

Relation

Association between the instances of one or more entity types
Example: Blue Car Belongs to Student Jack

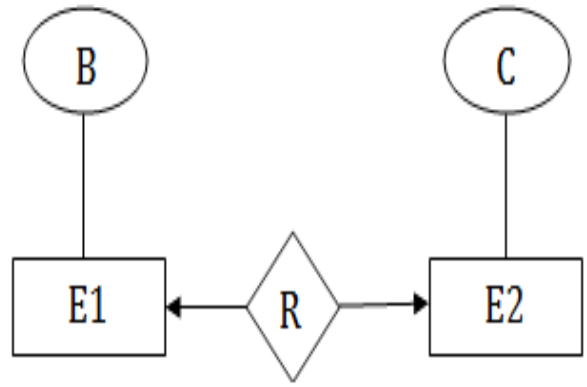
Mapping Cardinalities

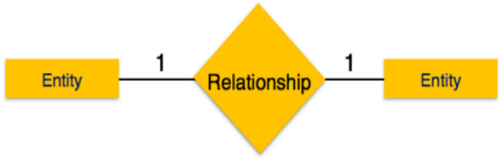
- Expresses the number of entities to which another entity can be related via a relationship set
- It is most useful in describing the relationship sets that involve more than two entity sets.
- For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:
- Types:
  - ✔ One-to-one
  - ✔ One-to-many
  - ✔ Many to one

✔ Many-to-many

Mapping Cardinalities : One to One

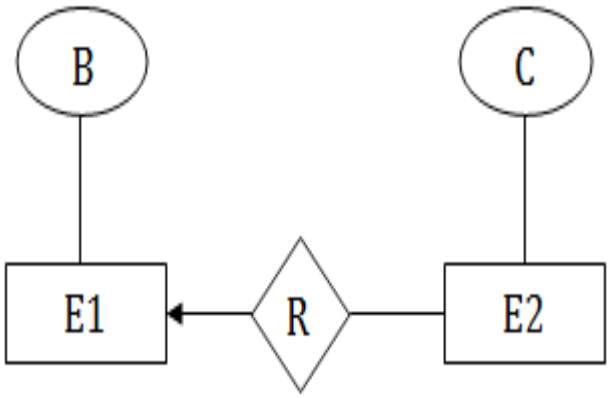
- In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1.

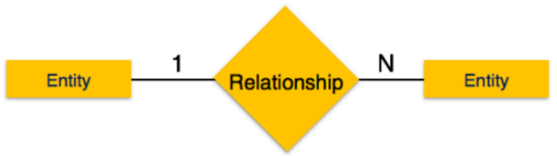




Mapping Cardinalities : One to Many

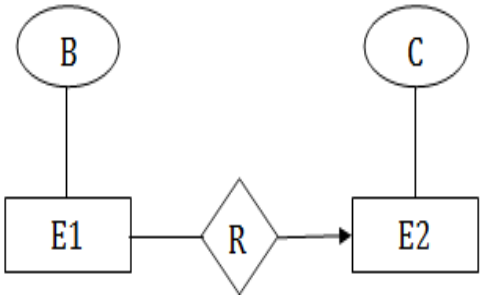
- In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.

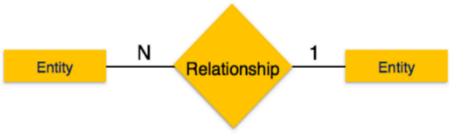




Mapping Cardinalities : Many to One

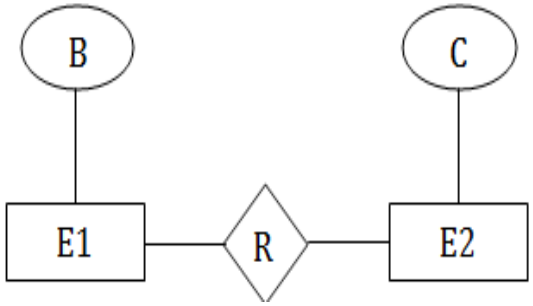
- In one-to-many mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.

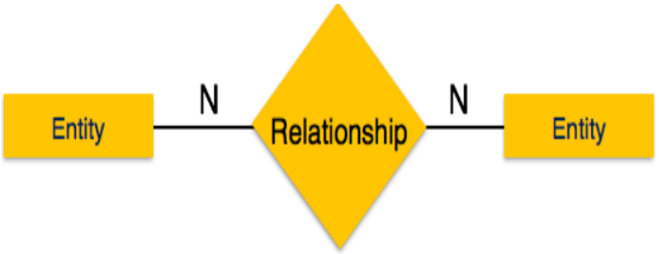


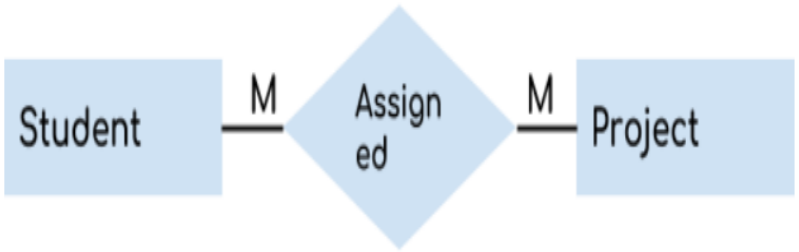


Mapping Cardinalities : Many to Many

- In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1.

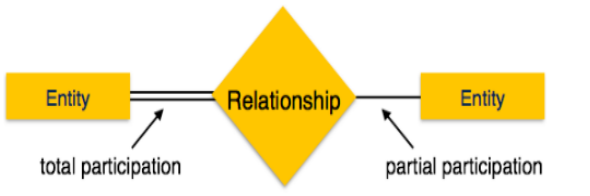


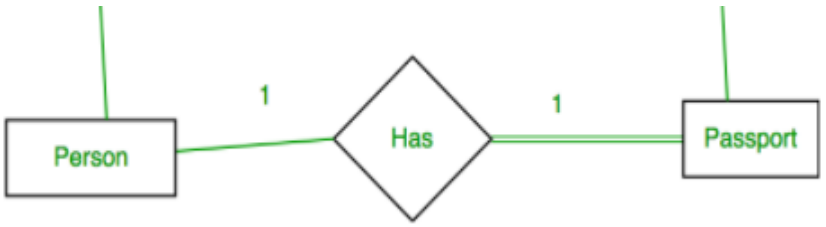




Participation Constraints

- **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.





Steps to draw a ER-Diagram

1. Identify the entities required

2. Identify the attributes, primary key
3. Identify the relationship needed.
4. Identify the cardinality ratio and participation constraints.

Example – ER Diagram for Bank

1. Entities :

- Bank
- Branch
- Employee
- Customer
- Loan
- Account

ER Diagram for Bank

2. Attributes :

ATTRIBUTES	PRIMARY KEY
BANK : Bname, code	code
BRANCH : Bname, Blocation, Branchcode	Branchcode
EMPLOYEE : Eid, Designation, Salary, Name	Eid
CUSTOMER : Cid, Cname, Address, DoB	Cid
LOAN : Loan_no, Amount, Rate	Loan_no
ACCOUNT : Acc_no, Type	Acc_no

ER Diagram for Bank

BANK

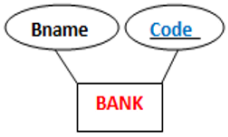
BRANCH

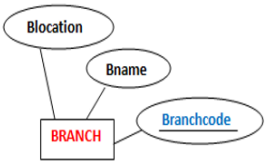
LOAN

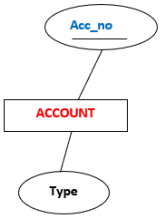
ACCOUNT

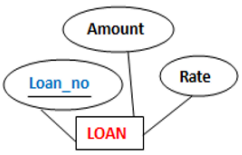
EMPLOYEE

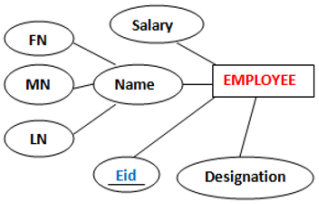
CUSTOMER

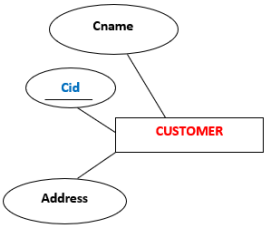


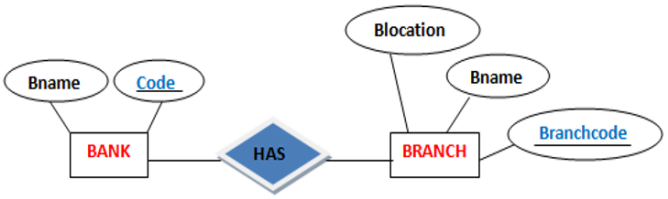


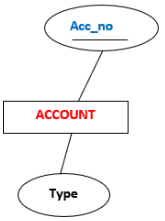


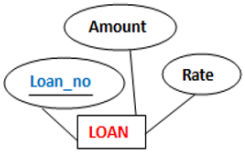


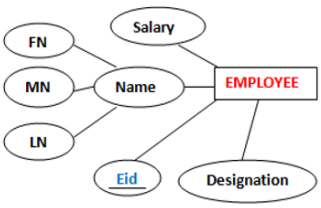


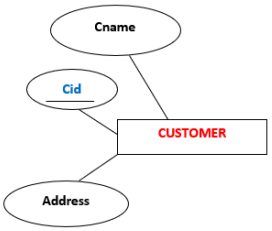


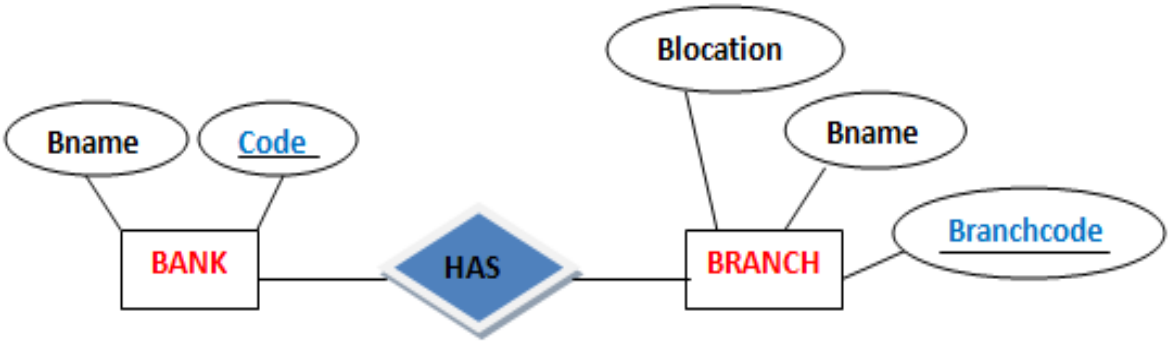






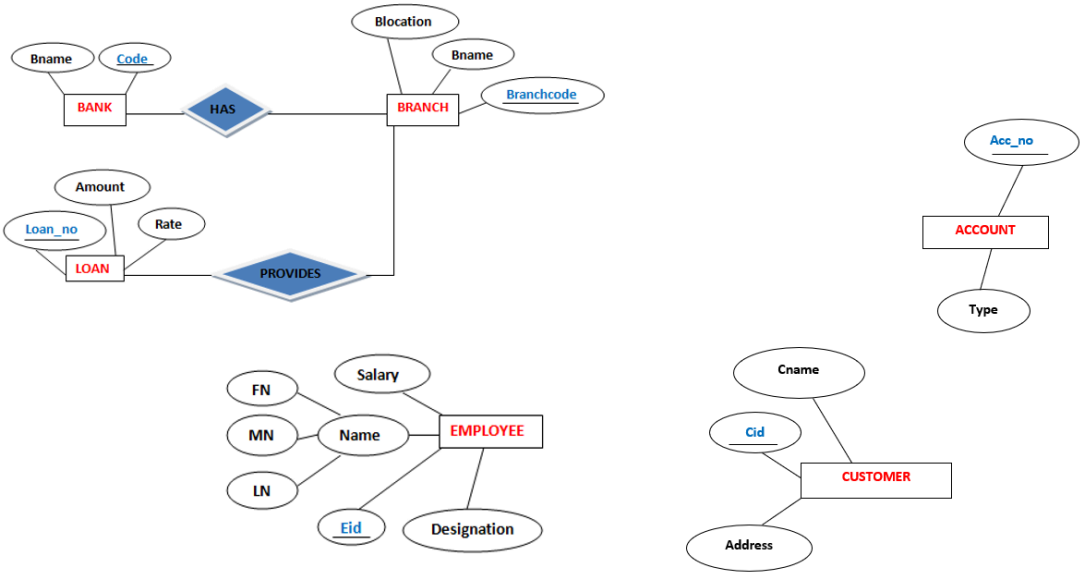


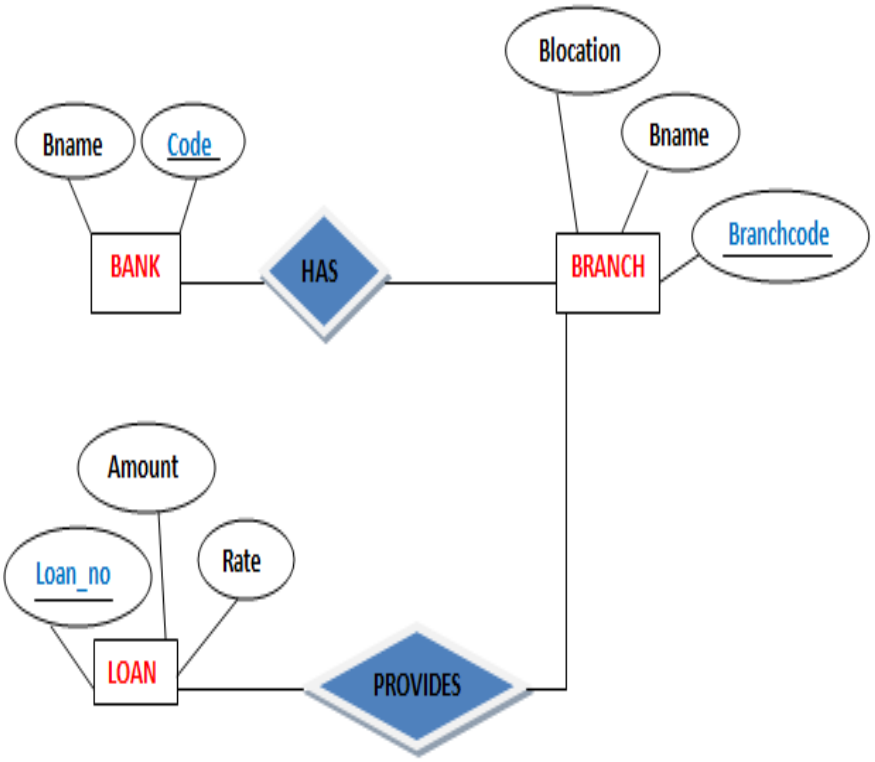




1. **Bank has a branch:**

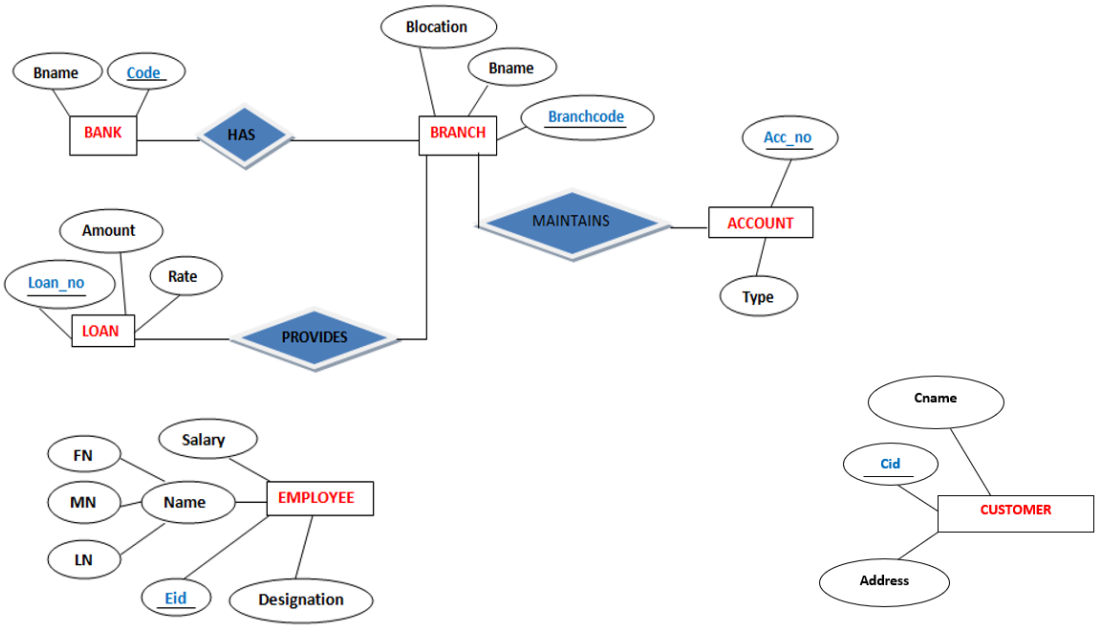
- Bank has “N” number of branches.
- But a Branch belongs to only one bank.
- Cardinality ratio: 1: N

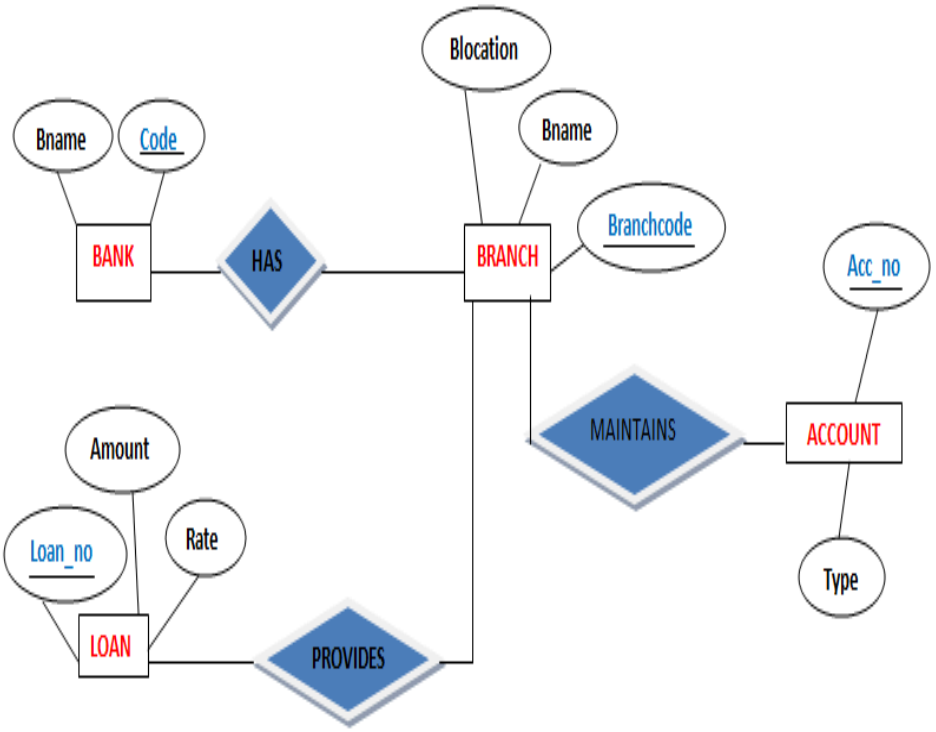




2. Branch provides loan

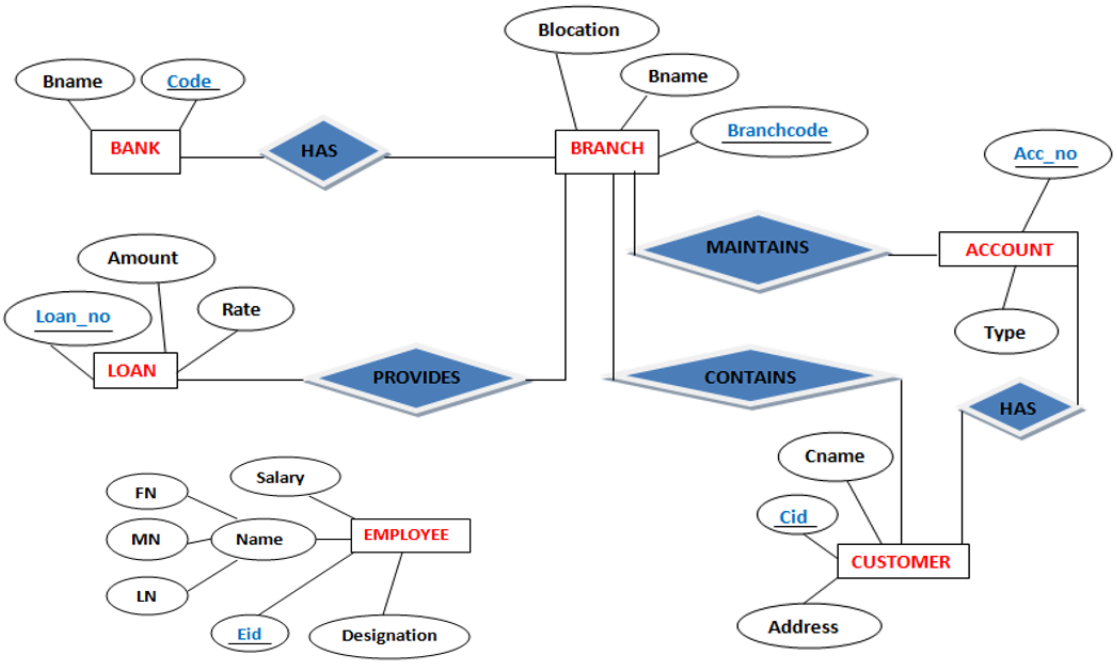
- A Branch provides “n” number of loan
- But a loan belongs to only “1” branch
- Cardinality ratio : 1: N





3. A branch maintains account

- A branch maintains “n” number of A/C
- An A/C belongs to only one branch
- Cardinality Ratio : 1: N

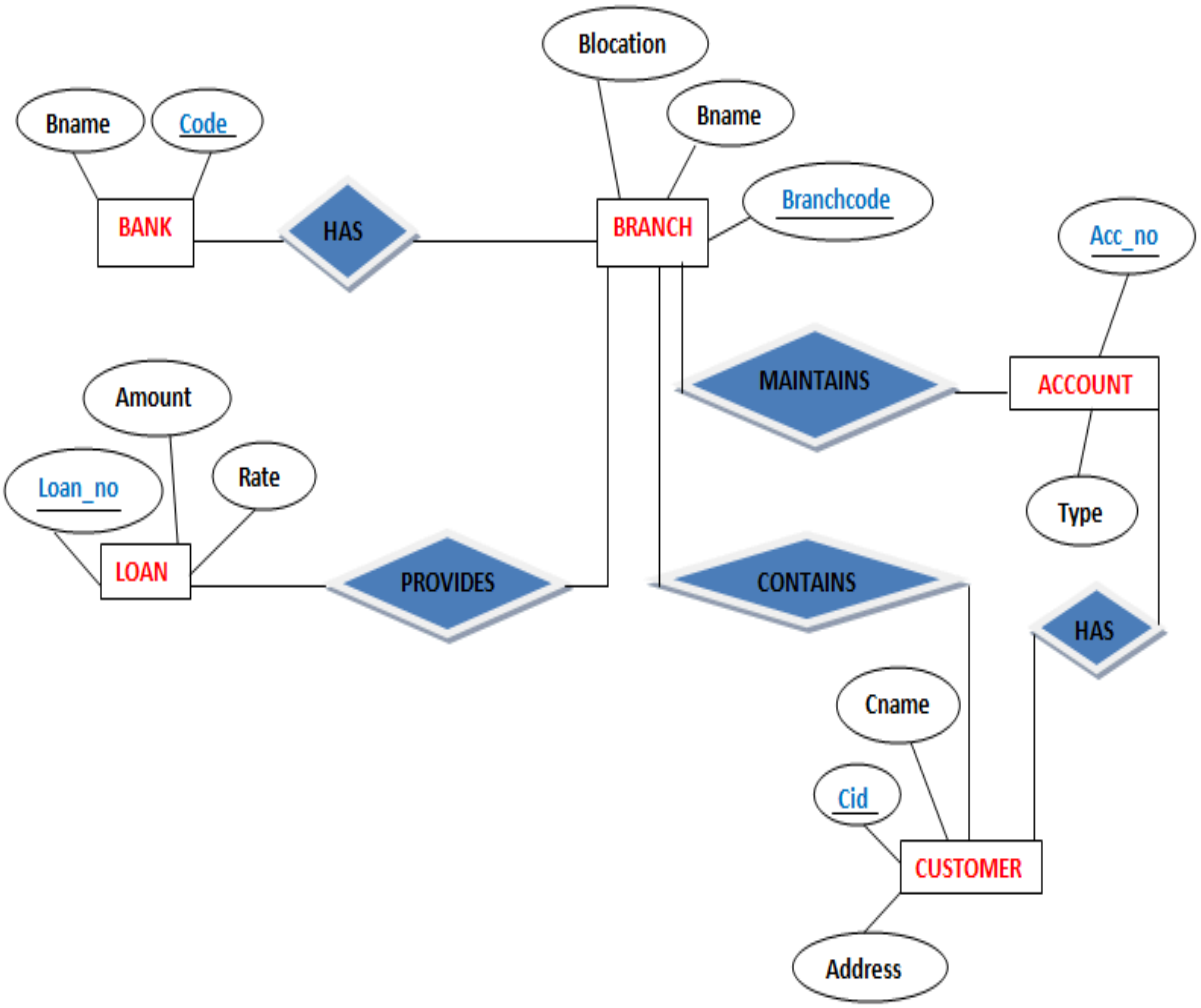


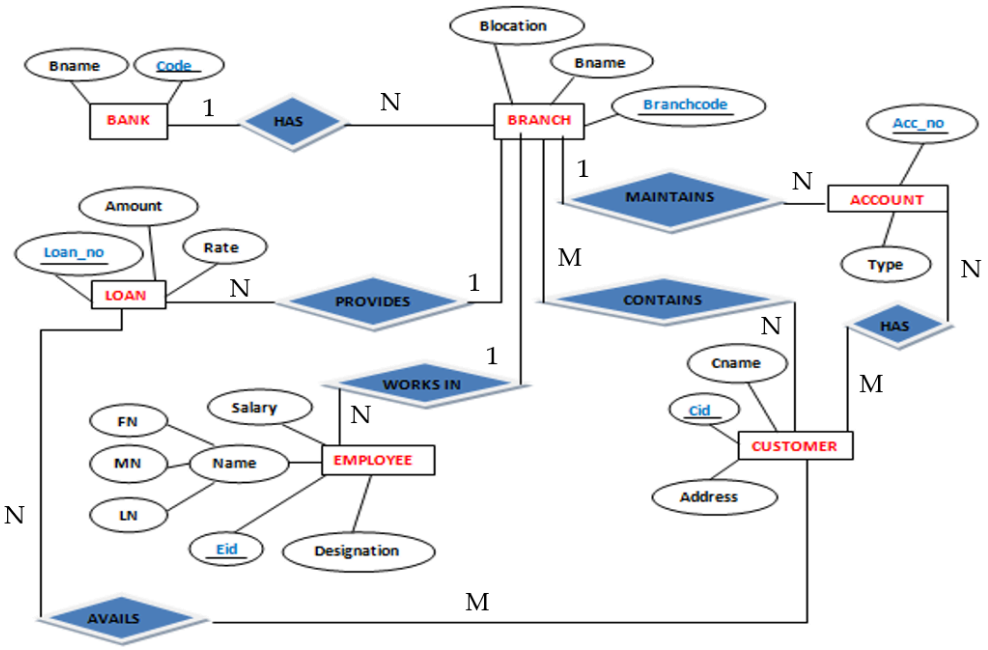
4. Branch contains customers

- A Branch contains “n” number of customers
- A customer can have A/C in more than one branch “M”.
- **Cardinality Ratio: M:N**

5. Customer has account

- Customer can have “n” number of A/C
- A/C belongs to “m” customers
- **Cardinality Ratio : M:N**



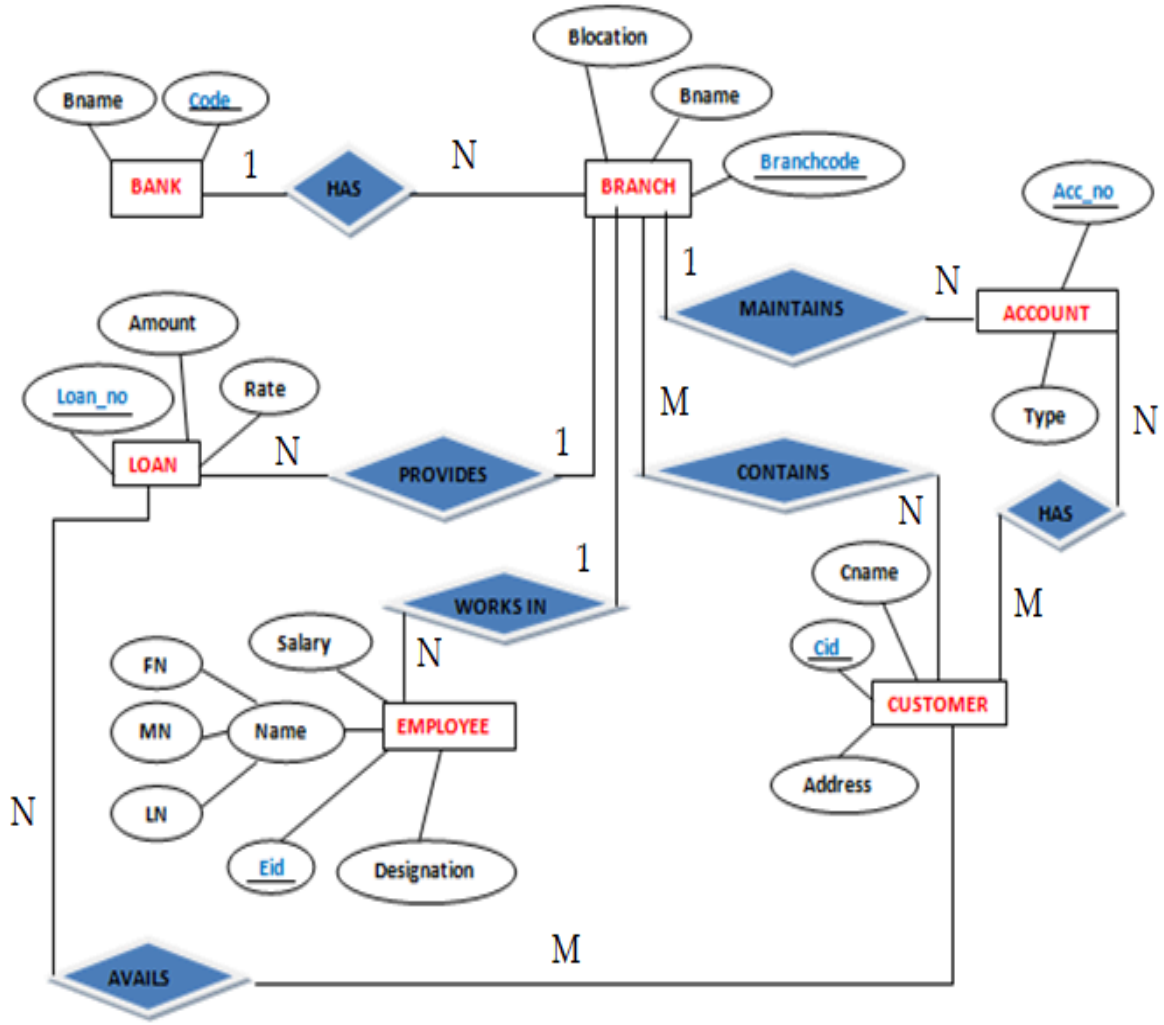


6. Employee works in a branch:

- A employee works for only one branch.
- A branch can have “n” number of employees.
- Cardinality ratio : N:1

7. Customer avails loan:

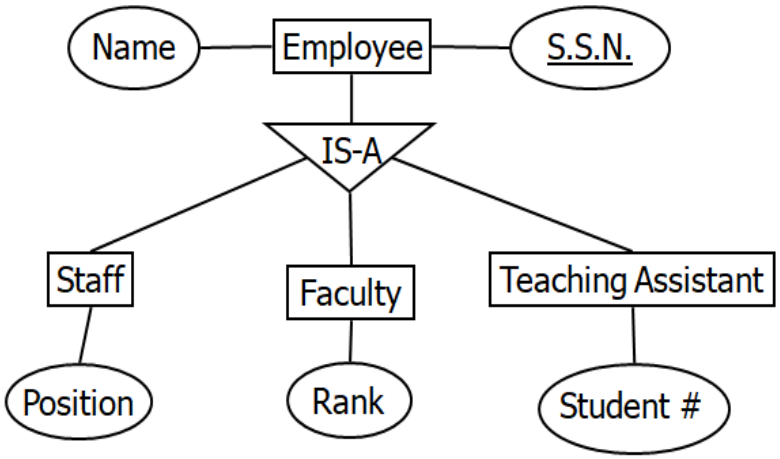
- A customer can have “N” number of loans.
- A loan can be jointly availed for “m” customers
- Cardinality ratio : M:N



Extended ER Model

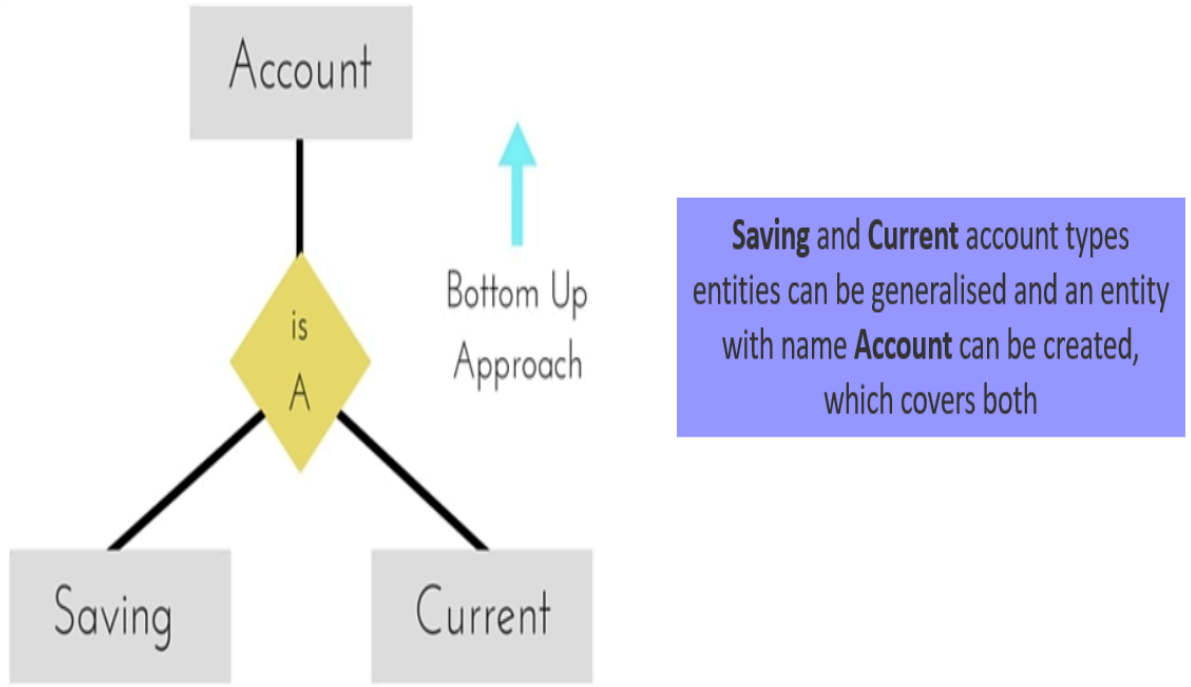
- ❖ E-R model is sufficient for traditional database applications
- ❖ Non-traditional applications (CAD, multimedia) have more complex requirements
- ❖ Can extend traditional E-R diagrams with *semantic data modeling* concepts

IS – A Relationship



Generalization

- **Bottom-up** approach
- **Two lower level** entities **combine** to form a **higher-level entity**
- It's more like superclass and subclass system
- Entities are combined to form a more generalised entity, in other words, sub-classes are combined to form a super-class.



Specialization

- Opposite to generalization

- Top-down approach
- One higher level entity can be broken down into two lower level entity