Dayananda Sagar College of Engineering
Kumarswamy Layout, Bengaluru – 560111
Department of Computer Science and Engineering

IV Semester

Database Management System (22CS44)

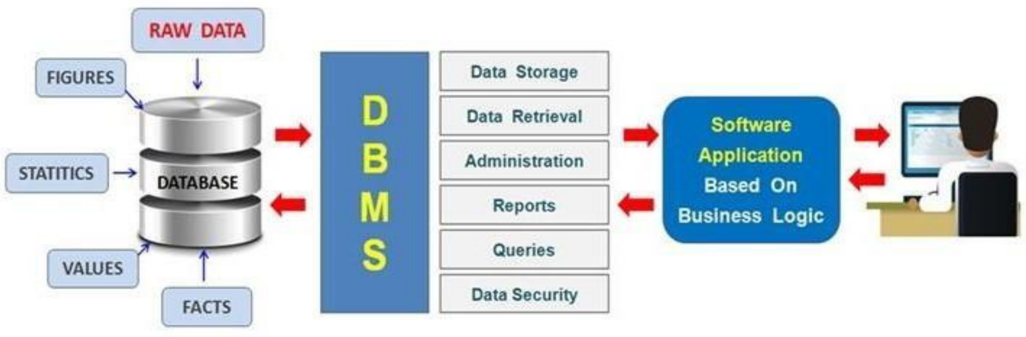
Introduction to DBMS

1. **Data**
   Data is any sort of information which is stored in computer memory. This information can later be used for a website, an application or any other client to store for future purpose. Examples : names, telephone numbers, addresses.

2. **Database**
   An organized collection of interrelated data.
   Data in the database
   - Is integrated
   - Can be shared
   - Can be concurrently accessed.

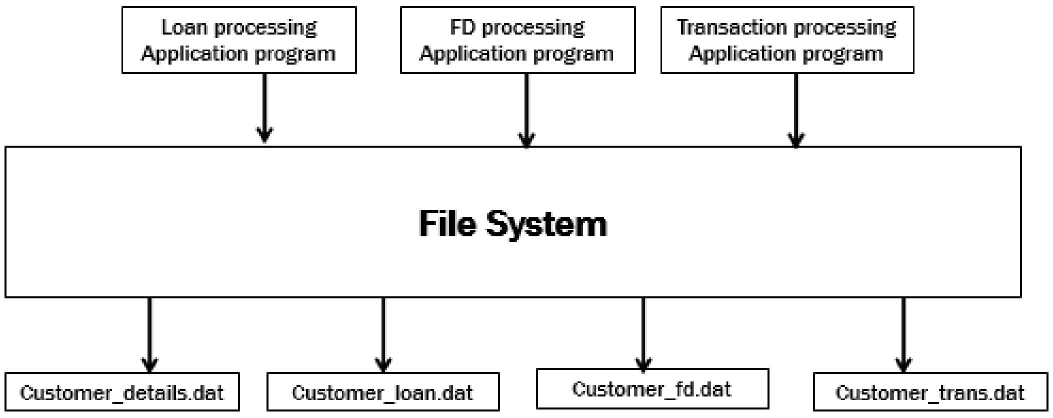
   Example : Student database.

3. **DBMS**
   Collection of interrelated files and a set of programs that allow users to access and modify these files.
   Example : Oracle, Sybase



Traditional File System vs DBMS

Traditional File System

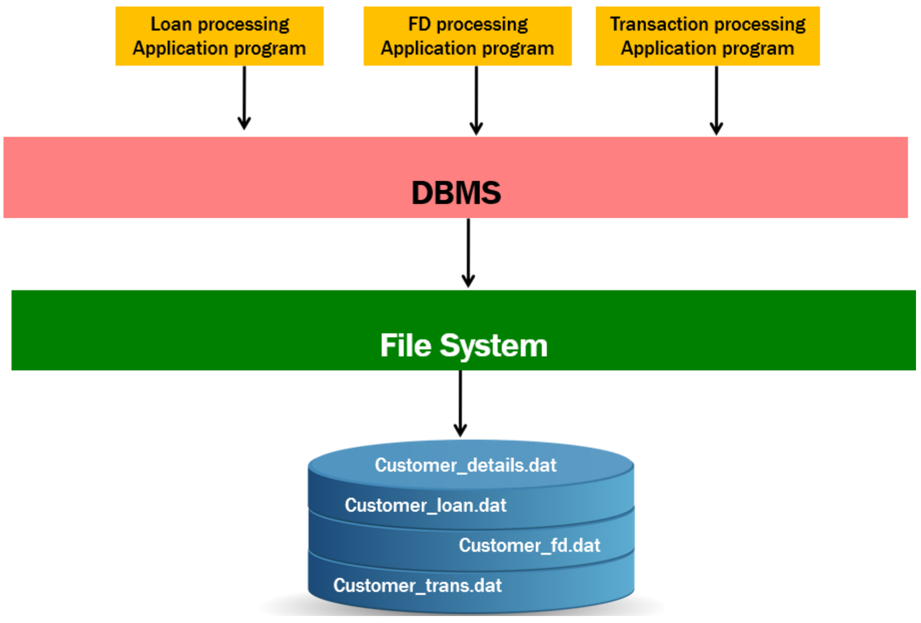

Loan processing Application program
FD processing Application program
Transaction processing Application program
File System
Customer_details.dat
Customer_loan.dat
Customer_fd.dat
Customer_trans.dat

In traditional approach, information is stored in flat files which are maintained by file system. Flat file is a data file that does not contain links to other files.

Disadvantages

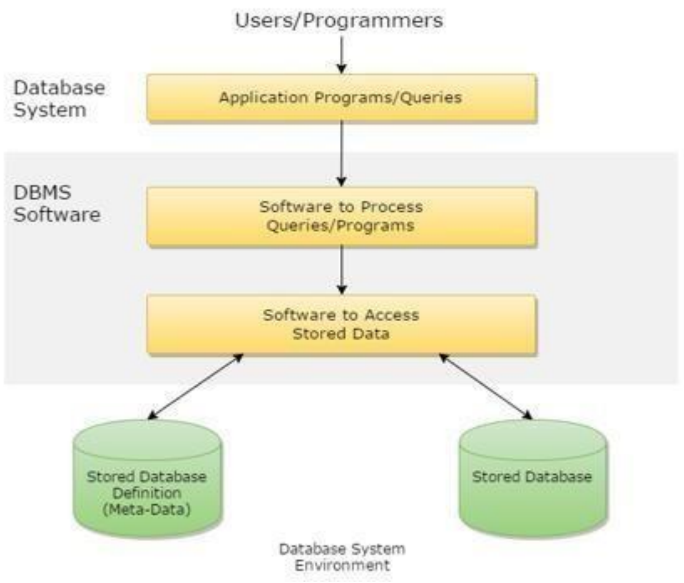
1. **Data security:** As data is stored in flat file,it is easily accessible and not secure.
2. **Data redundancy:** Same information can be duplicated in two or more files.duplication of data leads to high storage cost and data inconsistency.
3. **Data isolation:** It means all related data is not available in one file. Generally, data is scattered in various files and files may be in different format.
4. **Program/data independence:** Application programs are data dependent. It is impossible to change physical representation or access technique without affecting application.

DBMS


Loan processing Application program
FD processing Application program
Transaction processing Application program
DBMS
File System
Customer_details.dat
Customer_loan.dat
Customer_fd.dat
Customer_trans.dat

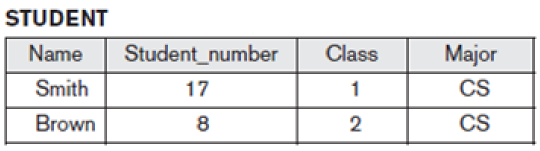
- DBMS is a collection of interrelated files and set of programs that allow users to access and modify these files.
- It is a general-purpose software system that facilitates the process of *defining, constructing, manipulating and sharing* database among various users and applications.
- *Defining* a database involves specifying the data types, structures and constraints of the data to be stored in the data base. The data base description is also stored in the data base in the form of a catalog and this data is called as metadata.
- *Constructing* a database is the process of storing the data on some storage medium that is controlled by DBMS.
- *Manipulating* a database includes functions such as querying, updating, deleting a database.
- *Sharing* a database allows multiple users and programs to access the database simultaneously.

Simplified database system environment

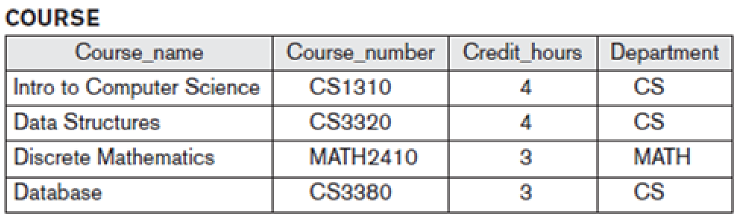

Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database
Database System Environment

A database is an organized collection of interrelated data. A database is built, designed and populated with data for a specific purpose and has intended user.

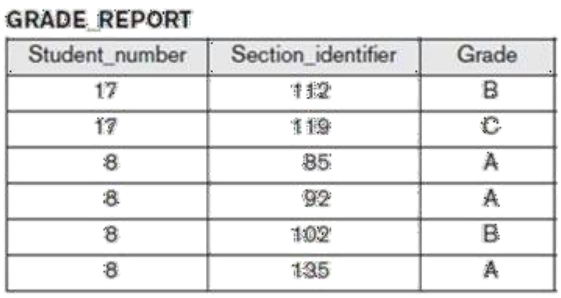
Example : Consider **University database.** To construct university database, we store data to represent each student details, course details, section details, grade report and prerequisite details.


STUDENT

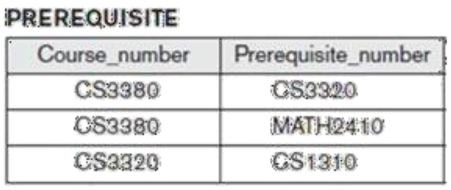
Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS


COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS


GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A


PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Database Management System

- DBMS is a collection of interrelated files and set of programs that allow users to access and modify these files.
- It is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing database among various users and applications.
- Defining a database involves specifying the data types, structures and constraints of the data to be stored in the data base. The data base description is also stored in the data base in the form of a catalog and this data is called as metadata.

Catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Characteristics of Database Approach

1. Self-describing nature of a database system
2. Insulation between programs and data, and data abstraction
3. Support of multiple views of the data
4. Sharing of data and multiuser transaction processing

1. Self describing nature :
A general purpose DBMS is not written for specific database application. It must work equally with any number of database application.
Example : University database, Banking database, company database etc. as long as the database definition is stored in the catalog.

2. Insulation between programs and data, and Data Abstraction:
In traditional file processing, the structure of data files is embedded in the application programs. So, any changes to the structure of a file may require changing all programs that access that file. In DBMS, the structure of the data files is stored in the DBMS catalog separately from the access programs. Any change made to the structure of data does not require changing the access programs. This property is called program-data independence.

3. Support of multiple views of data:
A database has many users, each of whom may require a different perspective or view of the database.
A view may contain virtual data that is derived from the database files but is not explicitly stored.

A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views. Example : one user may be interested in checking student address details while another user may be interested in checking that all the students have taken elective subject or not.

4. Sharing of data and multiuser transaction processing:

- A Multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database.
- Example : when many persons try to book a seat in a movie theatre, DBMS should see that each/specified no. of seats can be accessed by only one person at a time for booking. Such applications are called Online Transaction Processing (OLTP) applications. A transaction is an executing program or process that includes one or more database access such as reading or updating database records.

Advantages of DBMS

1. **Controlling redundancy**
   Redundancy means storing same data multiple times. It creates data inconsistency and requires more storage space. DBMS integrates different user groups during database design. It stores each logical item in only one place of database. This ensures consistency and saves storage space. Sometimes controlled redundancy helps to improve performance of queries.
2. **Restricting unauthorized access**
   When multiple user share a large database, it is likely that most users will not be authorized to access all information in database.DBMS should provide security and authorization subsystem. Most of the users will not be authorized to access all information in the database. Example : Financial data. Some can be permitted to retrieve data whereas others are allowed to retrieve and update.
   Example :
   global.contineo.in
   Users are given amount numbers protected by passwords.
   DBA creates accounts and specifies account restrictions.
3. **Providing persistent storage for program object**
   Database is used to provide persistant storage for program object and data structure. This is known as Object Oriented Database System. Programming languages have complex data structures such as class definitions in C++ and Java. When a program terminates, the value of program variables are discarded. To store them permanently, files are used. Use of files involves converting these complex data structures into a format suitable for file storage. When the information is to be read, programmer must convert file format to variable structure.
   OODB are compatible with programming languages C++ and Java. DBMS software automatically performs necessary conversions. Hence a complex object in C++ can be stored permanently in OODBMS. This object is called persistent since it survives the termination of program execution and later directly retrieved by another C++ program.

4. **Providing storage structure for efficient query processing**
   Database system provide capabilities for efficiently executing queries and updates. DBMS provides specilaized data structure called indexes to speed up disk search for desired records. Indexes are based on tree data structures or hash data structures. Indexes allow database application to find data without reading the whole table. Query processing and optimization module of DBMS choose an efficient query execution plan for each query based on existing structure.
5. **Providing backup and recovery**
   Backup and recovery subsystem is responsible for recovery. For example if a computer fails in middle of a complex transaction, recovery system restore to its previous state. Alternatively recovery system ensures that transaction is resumed from the point at which it was interrupted.
6. **Providing multiple user interface**
   DBMS provides a variety of user interfaces. These include query language, programming language interface and menu driven interface. Both form style and menu driven interface are known as graphical user interfaces.
7. **Representing complex relationship among data**
   DBMS represents variety of complex relationship among data, defines new relationship, retrieves and updates related data.
8. **Enforcing integrity constraints**
   DBMS should provide capabilities for defining and enforcing integrity constraints. The simplest type of integrity constraints involves specifying a data type for each data item.
   Examples :
   - USN must be 10 character value : DAADDAADDD
   - Semester must be within the range 1-8
   - Each USN must be unique
   - Name cannot be null
9. **Drawing inferences and actions**
   Some database systems provide capabilities for defining deduction rules for inferencing new information from the stored database facts.
10. **Potential for enforcing standards**
    Standards refer to data item names, display formats, screens etc

When NOT to use a DBMS

1. If the database and applications are simple, well-defined and not expected to change.
2. If access to data by multiple users is not required.
3. If the database system is not able to handle the complexity of data because of modeling limitations.
4. If the database users need special operations not supported by the DBMS.

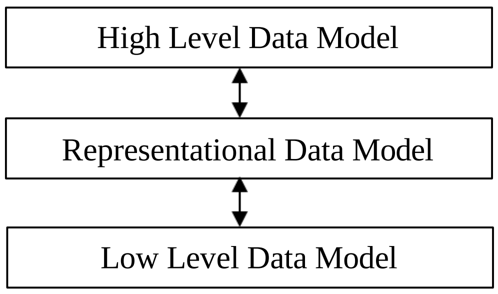
Data Model

It is a collection of concepts that describes

- **Structure of the database :** Data types, Relationships, Constraints
- **Basic operations for retrieving and updating :** Insert, delete, update

Categories of Data Model

1. High level or Conceptual data models
2. Representational or Implementation data models
3. Low level or Physical data models



1. High level or Conceptual data models

- Provides concepts that are close to the way many users perceive data.
- Use concepts such as entities, attributes and relationships.

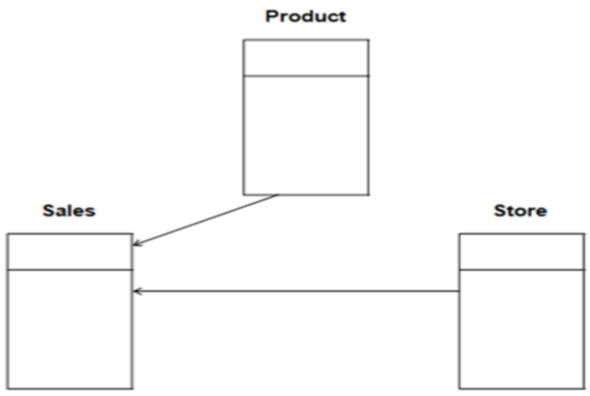
Entity :
Real world Object or concept is an entity.
Examples : In a College database, students, teachers, classes and courses offered can be considered as entities.

Attribute :
Entities are represented by means of their properties, called **attributes**.
Examples : A student entity may have name, class, and age as attributes.

Relationship :
Represents association among two or more entities.
Example: EMPLOYEE ***Works on a*** PROJECT
In high level model,

- Only entities and relationships are specified.
- No attribute is specified.

No primary key specified.

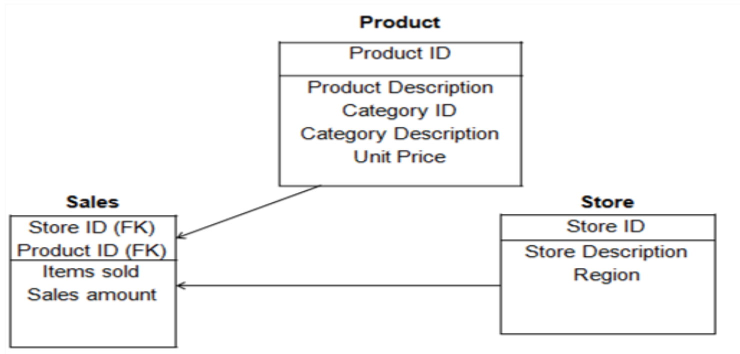


2. Representational or Implementation data models

- They provide concepts understood by end users.
- Also known as **record-based data models** because they represent data by using record structures.

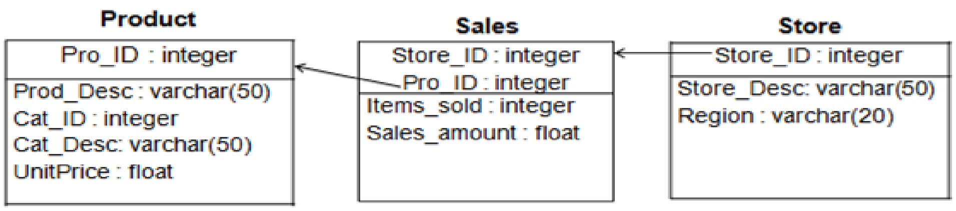
This model includes

- Entities and relationships between them.
- Attributes for each entity.
- Primary key for each entity.
- Foreign key to identify relationships between different entities.



3. Low level or Physical data models

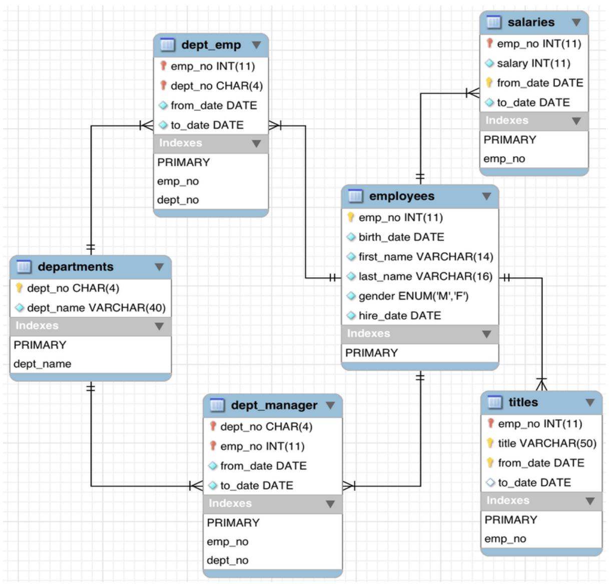
- They describe how the data is stored as files.
- They show Record formats, Record ordering, Access paths such as indexes.
- Concepts provided by Low level data models are generally meant for computer specialists and NOT for end users.



Schema, Instance and Database state

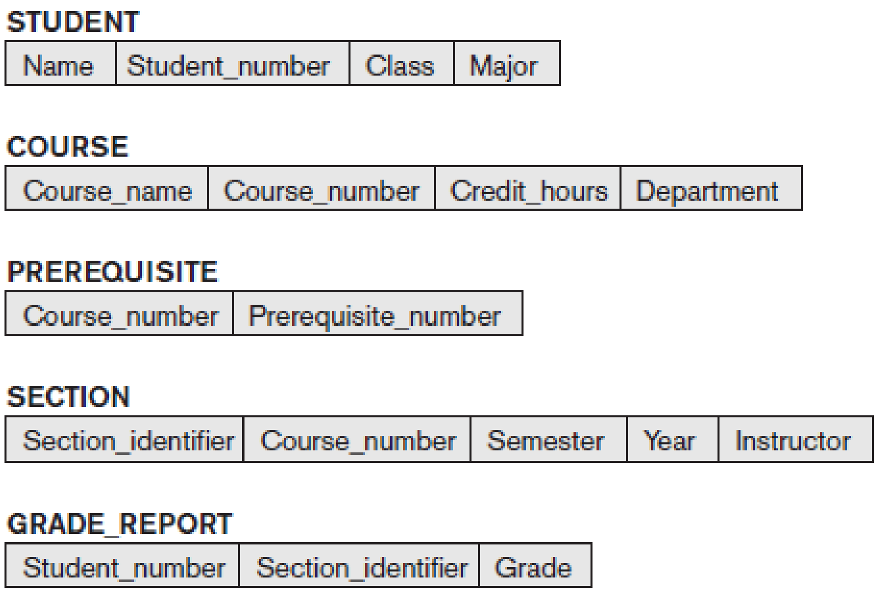
1. Database Schema (Intension)

- It is the description of a database which is specified during the database design and is not expected to change frequently.
- It includes descriptions of the database structure, data types, and the constraints on the database.
- It represents Representational data model.



2. Schema Diagram

The schema diagram displays the structure of each record type but not the actual data.



STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

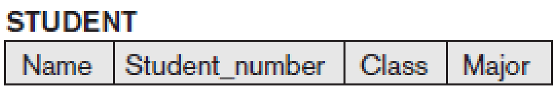
Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

3. Schema Construct

It is a component of the schema or an object within the schema.
Examples : STUDENT, COURSE



STUDENT

Name	Student_number	Class	Major

4. Database state (Extension OR Database snapshot)

- It is the actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database.
- The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*.

Differences between Schema and State

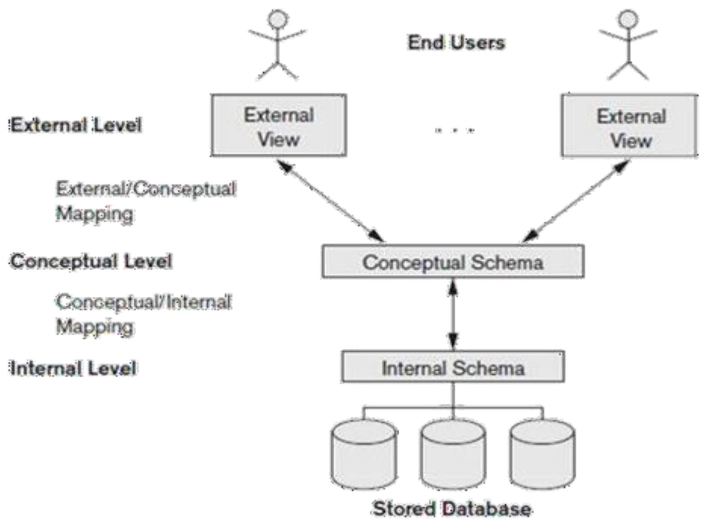
- Database Schema does not change frequently.
- Database State changes every time the table is updated. Example: when a record is added or deleted.

5. Metadata

- DBMS definition is stored in DBMS catalog, which contains information such as structure of each file, type, storage format of each data item and various constraints on data.
- This information stored in the catalog is called as metadata and it describes the structure of the primary database.
- Metadata is by definition "data about other data".

Name	Type	Length	Description
SName	Character	50	Student's Name
SID	Number	5	Unique Identification Number for a student
DOB	Date	8	Student's Date of Birth in the format 01.01.80

Three Schema Architecture of Database System



1. Internal schema

- It uses the physical data model.
- Describes the physical storage of the data.
- Also provides information about the access paths (Ex: indexes)

2. Conceptual schema

- It uses the representational data model.
- Describes the structure of the database for users.
- Hides the details of the physical storage structure.
- Details about entities, relationships, data types, constraints etc…

3. External schemas

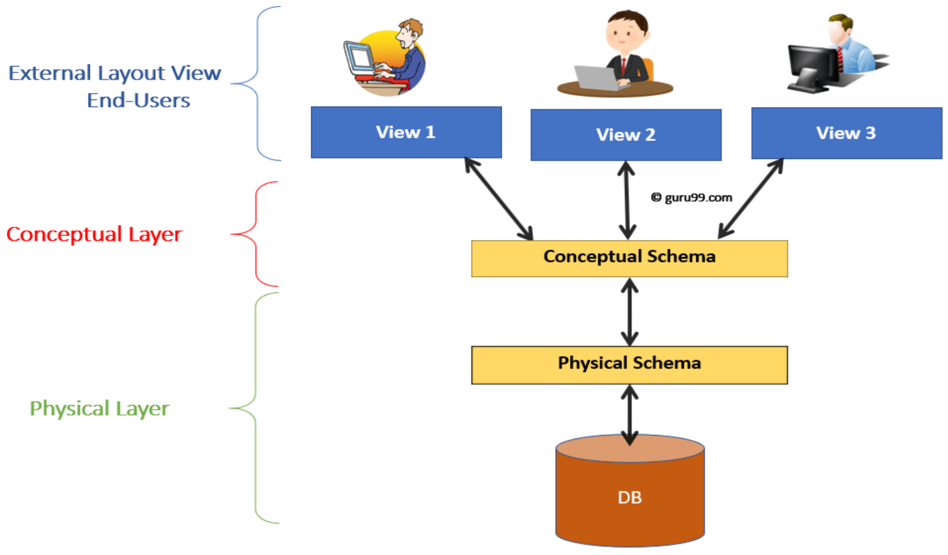
- Describes the database for a particular user group.
- The rest of the database is hidden from that user group.
- Usually uses the same data model as the conceptual schema.

Data Independence

The schema at one level can be changed without having to change the schema at the ***next higher level***.

Two types of data independence

1. Logical Data Independence
2. Physical Data Independence



1. Logical Data Independence:

The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

Examples

- Add/Modify/Delete a new attribute without a rewrite of existing application programs
- Merging two records into one, Breaking an existing record into two or more records

2. Physical Data Independence:

The capacity to change the internal schema without having to change the conceptual schema.

Examples

- Using a new storage device like Hard Drive or Magnetic Tapes
- Switching to different data structures.
- Change of Location of Database from say C drive to D Drive

Database Languages and Interfaces

1. Data Definition Language (DDL)

It is used by the DBA and database designers to specify the conceptual schema of a database.
In many DBMSs, the DDL is also used to define internal and external schemas (views).
In some DBMSs, separate storage definition language (SDL) and view definition language (VDL) are used to define internal and external schemas.
Examples : CREATE, ALTER, DROP commands in SQL

2. Data Manipulation Language (DML)

It is used to specify database retrievals and updates.
DML commands (data sublanguage) can be *embedded* in a general-purpose programming language such as COBOL, C, C++, or Java.
A library of functions can also be provided to access the DBMS from a programming language.
Examples : INSERT, SELECT, UPDATE, DELETE commands in SQL

3. Data Control Language (DCL)

It is used to assign roles, permission and referential integrity.

Examples: GRANT, REVOKE

Classification of DBMS

The different criteria used to classify the DBMS are

1. **Based on the data model used**
    a) Relational data model
    b) Network data model
    c) Hierarchical data model
    d) Object-oriented data model
    e) Object-relational data model
2. **Based on the no. of users**
    a) Single user system
    b) Multiuser system
3. **Based on the number of sites**
    a) Centralized
    b) Distributed
4. **Based on the cost**

1. Based on the data model used

a) Relational data model

Currently it is the most dominant model for developing database applications. The basic relational data model represents a database as a collection of tables.

Examples DB2, Oracle, MS SQL Server, Sybase, Informix

STUDENT

Name	Student_number	Class	Major

COURSE

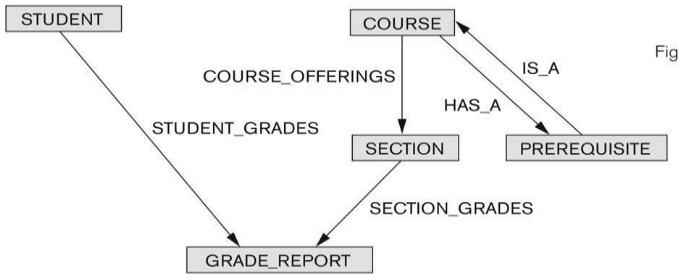
Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

b) Network data model

- The first network DBMS was implemented by Honeywell in 1964-65 (IDS System).
- The network model represents data as record types and also represents a limited type of 1:N relationship. A 1:N relationship relates one instance of a record to many record instances using some pointer linking mechanism in these models.
- Examples : IDMS, DMS 1100 (Unisys), IMAGE (HP)



c) Hierarchical data model

It represents data as hierarchical tree structures. Each hierarchy represents a number of related records. There is no standard language for this model. Examples : IMS (IBM)

d) Object-oriented data model

It defines a database in terms of objects, their properties and their operations. Objects with the same structure and bahaviour belong to a class and classes are organized into hierarchies. The operations of each class are specified in terms of predefined procedures called methods. Example : O2, ORION, IRIS

e) Object-relational model

Relational DBMS incorporate concepts from object databases leading to object-relational DBMS. Examples : Oracle 10i, DB2, SQL server

2. Based on the number of users

a) Single user system : supports only one user at a time are mostly used with PCs.

b) Multiuser system : supports concurrent multiple users.

3. Based on the number of sites over which the database is distributed

a) Centralized : It can support multiple users but the DBMS and database reside totally at a single computer site.

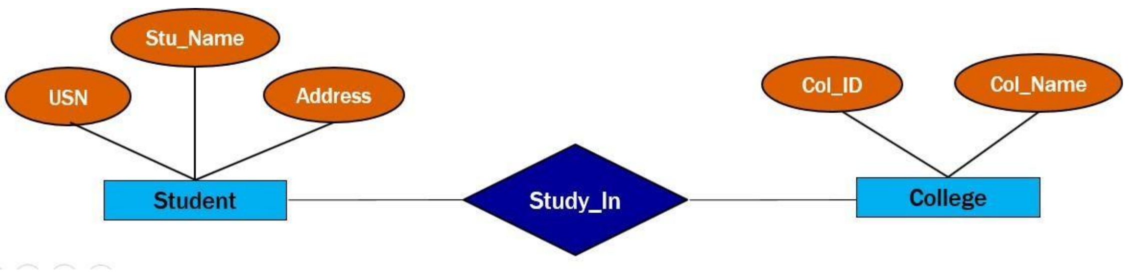
b) Distributed DBMS : It can have the actual database and DBMS software distributed over many sites, connected by a computer network.

4. Based on the cost

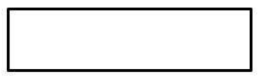
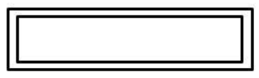
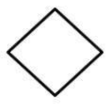
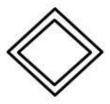
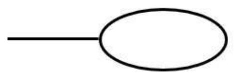
The cost for DBMS ranges from free open-source systems to configurations costing millions of dollars. Examples of free software : MySQL

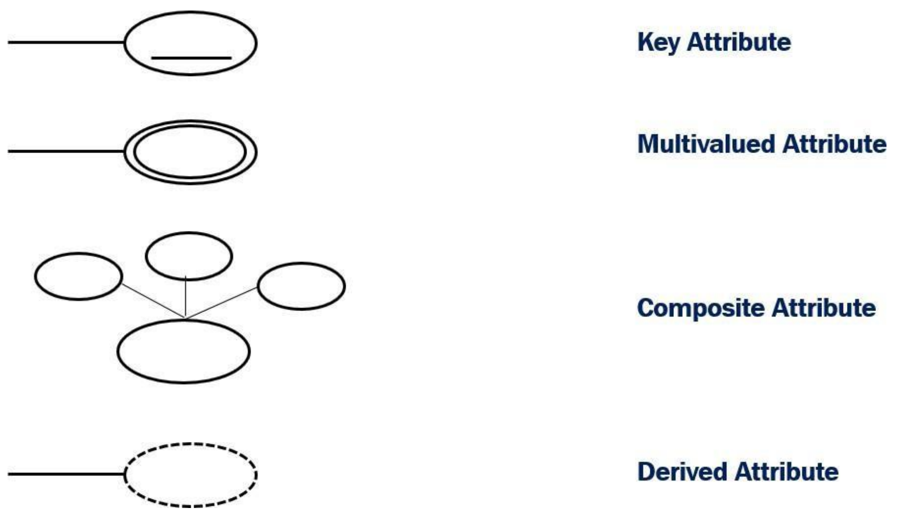
Data Modeling using the Entity-Relationship (ER) Model

- ER Model is based on the concept of real-world entities and relationships among them.
- Visual representation of data that describes how data is related to each other.
- Used for the conceptual design of a database.
- based on:
  - **Entities** and their attributes.
  - **Relationships** among entities.



ER Diagram Notation

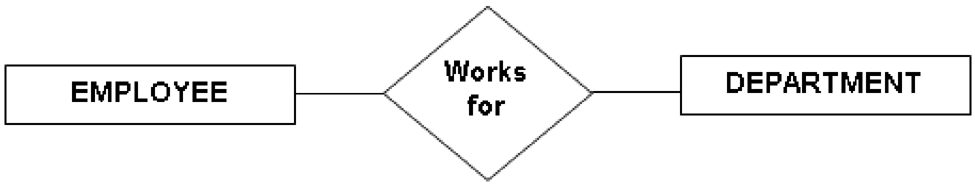
Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying relationship
	Attribute



Entity Types, Entity Sets, Attributes and Keys

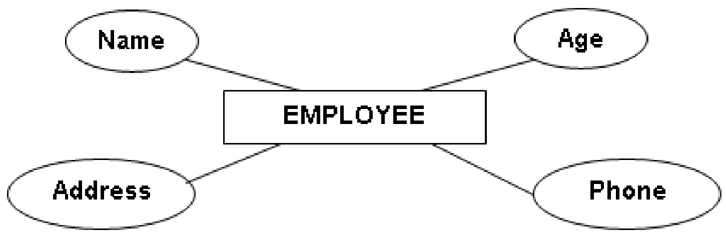
Entity

- Entity is an object or a thing in the real world with an independent existence.
- An entity may be an object with the physical existence.
  Example : person, car, house, employee etc.
- An entity may be an object with the conceptual existence.
  Example : company, job, university etc.



Attribute

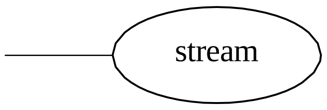
- Attributes are the properties used to describe an entity. **Example :** EMPLOYEE entity may be described by the employee's name, age, address, job.
- The entities have some value for each of the attributes. **Example :** The EMPLOYEE entity has four attributes: Name, Address, Age, and Phone; Their values are 'John Smith,' '2311 Kirby, Houston, Texas 77001', '55', and '713-749-2630', respectively.



Types of attributes

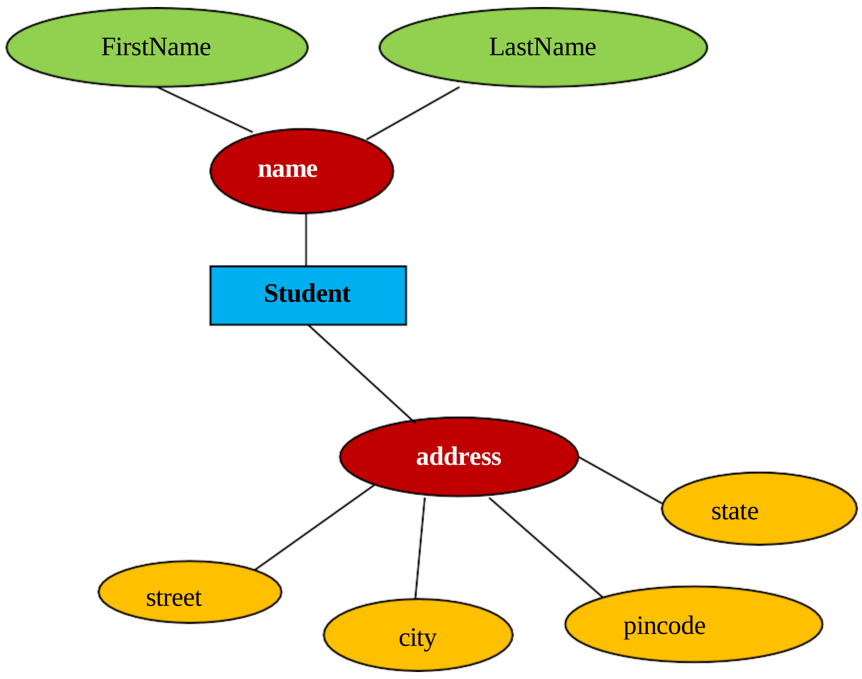
1. Simple (Atomic) attribute

- Simple attributes are atomic values, which cannot be divided further.
- *Representation :* Ellipse connected directly to an entity.
- *Example :* A student’s stream is an atomic value. It can be CSE or ISE



2. Composite attribute

- are composed of many other simple attributes.
- *Representation :* ellipse connected with an ellipse.
- *Example :* address (street, city, state, country)

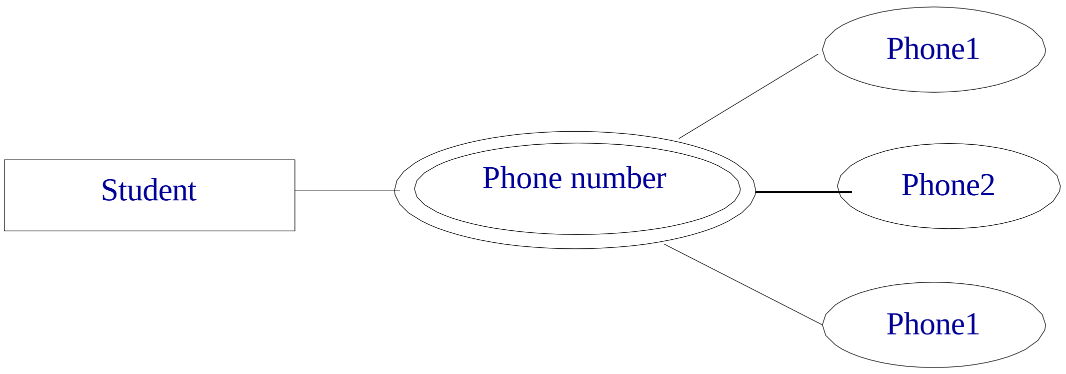


3. Single valued attribute

- Single valued attributes are those attributes which can take only one value for a given entity from an entity set.
- *Example:* Age, USN of student, PlaceOfBirth

Simple	**Single**
It cannot be subdivided into parts.	It has only one value.
Examples : marital status, gender	**Example :** Social Security Number (SSN)
	Single valued attribute is not necessarily a simple attribute. **Example :** Product serial number : SE-08-02-189935, is single-valued, but it is a composite attribute because it can be subdivided into the region in which the part was produced (SE), the plant within that region (08), the shift within the plant (02), and the product number (189935).

4. **Multivalued attribute**
   A multivalued attribute can have more than one value at a time for an attribute.
   Example : A person can have more than one phone number, email_address, skills, degree etc.

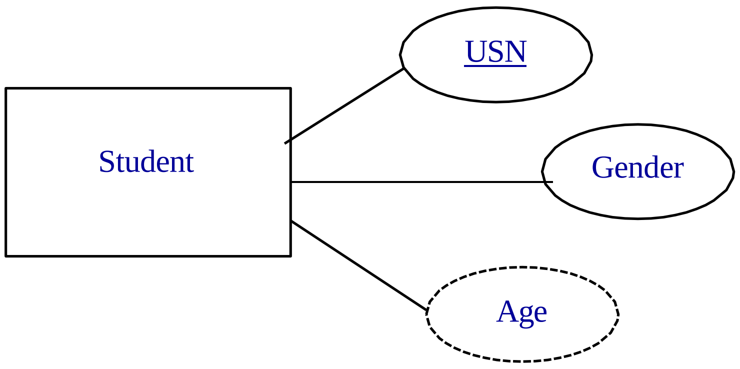


5. **Stored and derived attribute**
   Stored Attribute :
   - Stored attribute is an attribute which are physically stored in the database.
   - Attributes like student_id, name, roll_no in Student.
   - We cannot derive value of these attribute using other attributes.

   Derived attribute :
   - Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
   - *Example :* average_salary in a department should not be saved directly in the database, instead it can be derived.
   - age can be derived from date_of_birth.

6. **Null value**
   If a particular entity does not have an applicable value for an attribute or value is unknown or value is missing, Null value is used.
   Example: College degree for non college person.

7. **Key attribute**
   Key attributes are those attributes which can identify an entity uniquely in an entity set.



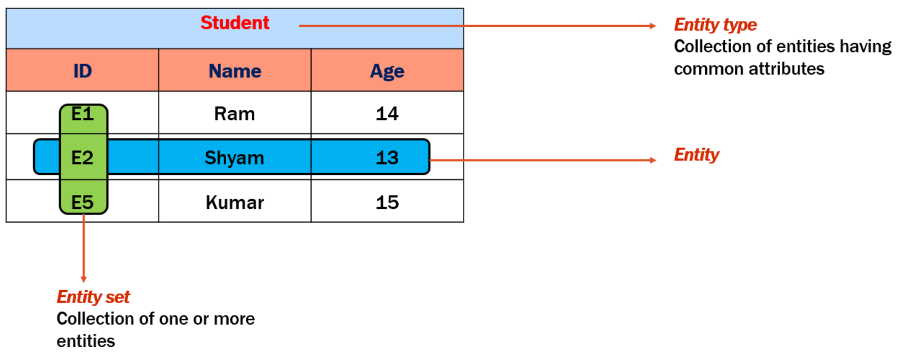
8. **Complex attribute**
   For an entity, if an attribute is made using the multi valued attributes and composite attributes then it is known as complex attributes.
   Example: A person can have more than one residence; each residence can have more than one phone.

Entity Type

- An entity type defines a collection of entities that have the same attributes.
- Each entity type is described by its name and attributes.
- An entity type describes the schema or intension for a set of entities that share the same structure. Example : employee

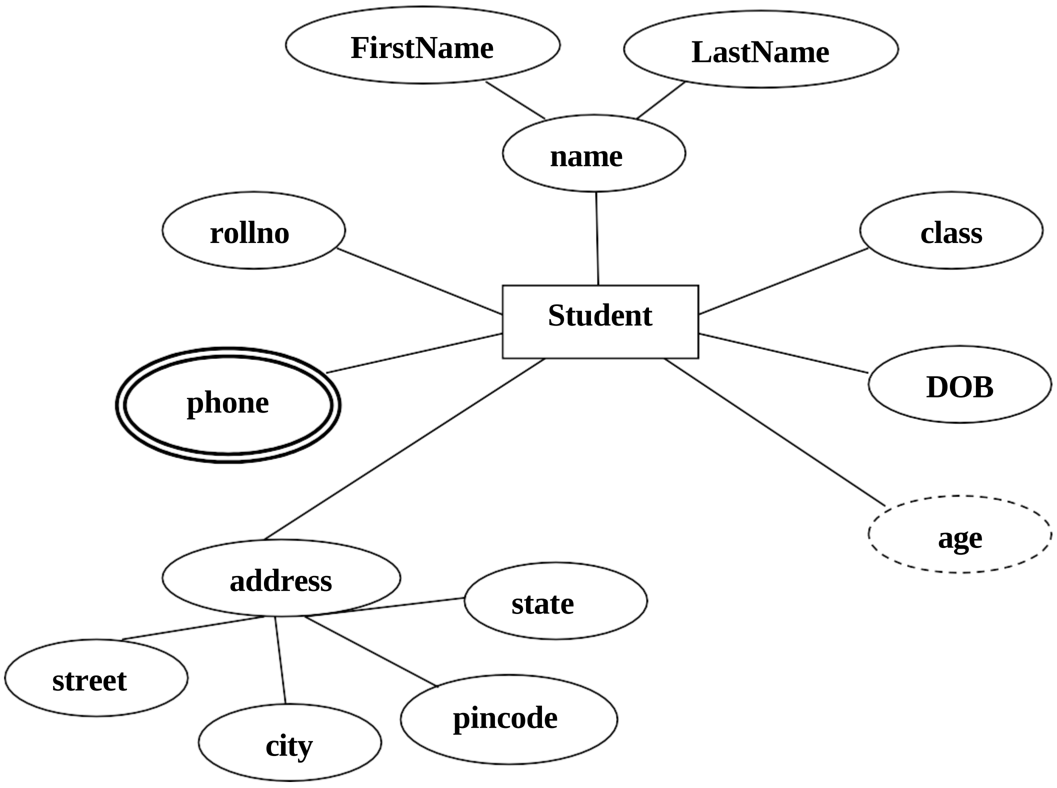
Entity set

- Each entity type will have a collection of entities stored in the database.
- Entity set is the current state of the entities of that type that are stored in the database.
- Entity set is a set of entities of same type.


Student
Entity type
Collection of entities having common attributes
Entity
Entity set
Collection of one or more entities

ID	Name	Age
E1	Ram	14
E2	Shyam	13
E5	Kumar	15

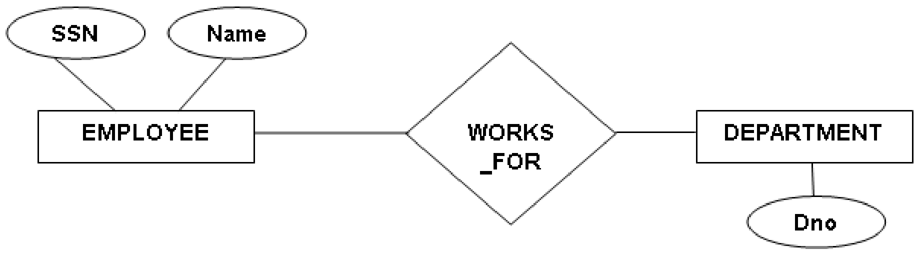
Identify the attributes in the following ER diagram



Simple/Single attribute	FirstName, LastName, Street, city, pincode, state, DOB, class
Composite attribute	Name, address
Multivalued attribute	Phone
Key attribute	rollno
Derived attribute	age

Relationships

- A relationship relates two or more distinct entities with a specific meaning.
- **Example :** EMPLOYEE John Smith ***works for*** the DEPARTMENT
- Relationship is represented using diamonds.



Relationship type

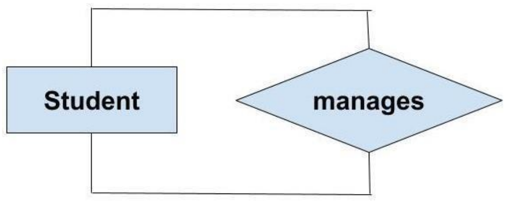
- It is the schema description of a relationship.
- It identifies the relationship name and the participating entity types.
- It also identifies certain relationship constraints.

Degree of a Relationship Type

The Degree of a relationship type is the number of participating entity types.
There can be Binary, ternary, unary..n-ary relationship

Unary relationship :

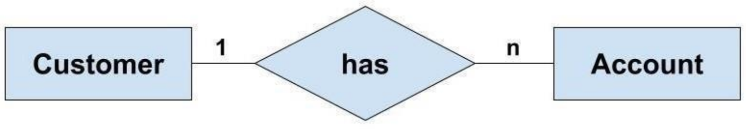
A unary relationship exists when both the participating entity type are the same. When such a relationship is present, we say that the degree of relationship is 1.
For example, Suppose in a classroom, we have many students who belong to a particular club-like dance club, basketball club etc. and some of them are club leads. So, a particular group of student is managed by their respective club lead. Here, the group is formed from students and also, the club leads are chosen from students. So, the ‘Student’ is the only entity participating here. We can represent this relationship using the E-R diagram as follows:



Unary Relationship
(degree 1)

Binary relationship :

A binary relationship exists when exactly two entity type participates. When such a relationship is present we say that the degree is 2. This is the most common degree of relationship. It is easy to deal with such relationship as these can be easily converted into relational tables.
For example, We have two entity type ‘Customer’ and ‘Account’ where each ‘Customer’ has an ‘Account’ which stores the account details of the ‘Customer’. Since we have two entity types participating we call it a binary relationship. Also, one ‘Customer’ can have many ‘Account’ but each ‘Account’ should belong to only one ‘Customer’. We can say that it is a one-to-many binary relationship.

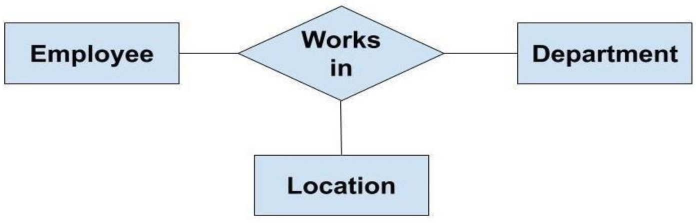


Binary Relationship
(degree 2)

Ternary relationship :

A ternary relationship exists when exactly three entity type participates. When such a relationship is present we say that the degree is 3. As the number of entity increases in the relationship, it becomes complex to convert them into relational tables.

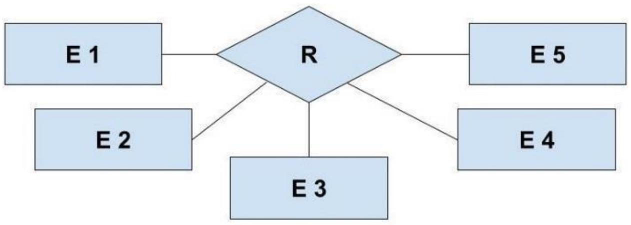
For example, We have three entity type 'Employee', 'Department' and 'Location'. The relationship between these entities are defined as an employee works in a department, an employee works at a particular location. So, we can see we have three entities participating in a relationship so it is a ternary relationship. The degree of this relation is 3.



Ternary Relationship (degree 3)

N-ary relationship :

An N-ary relationship exists when 'n' number of entities are participating. So, any number of entities can participate in a relationship. There is no limitation to the maximum number of entities that can participate. But, relations with a higher degree are not common. This is because the conversion of higher degree relations to relational tables gets complex. We are making an E-R model because it can be easily be converted into any other model for implementing the database. But, this benefit is not available if we use higher degree relations. So, binary relations are more popular and widely used. Though we can make a relationship with any number of entity types but we don't do that.



N-ary Relationship

Constraints on Relationship Types

Structural Constraints

1. Cardinality Ratio Constraint
2. Participation Constraint

1. Cardinality Ratio

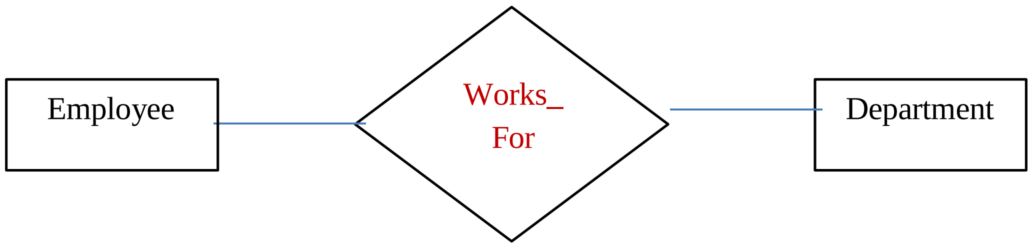
The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.

Examples :

WORKS_FOR binary relationship type

EMPLOYEE: DEPARTMENTS of cardinality ratio N:1,

Each department can be related to any number of employees, but employee can be related to (work for) only one department



Possible ratios are 1:1, 1:N, N:1, and M:N

One-to-One cardinality (1:1)

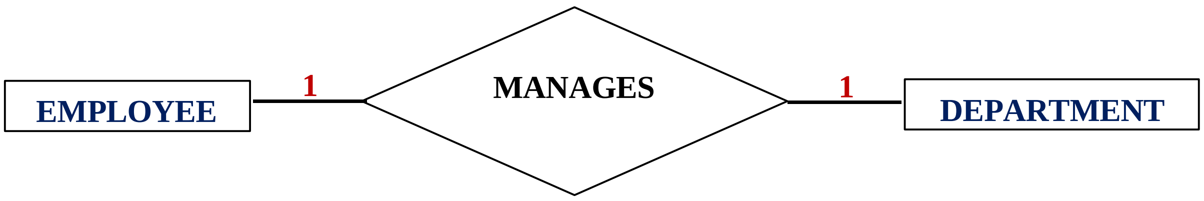
One instance of entity is related to one instance of another entity.

Example 1 :

One country can have only one citizen as president and one citizen can become president of only one country.

Example 2 :

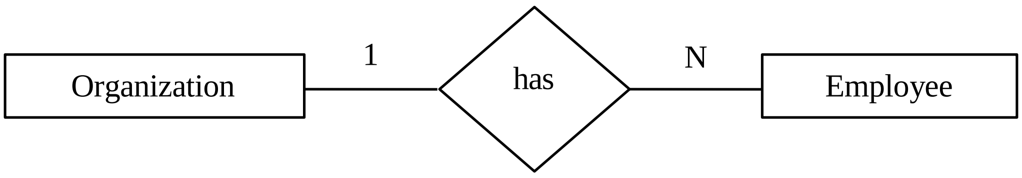
One employee can manage only one department and a department has only one manager.



One-to-many cardinality (1:N)

One instance of entity is related to multiple instances of another entity.

Example :

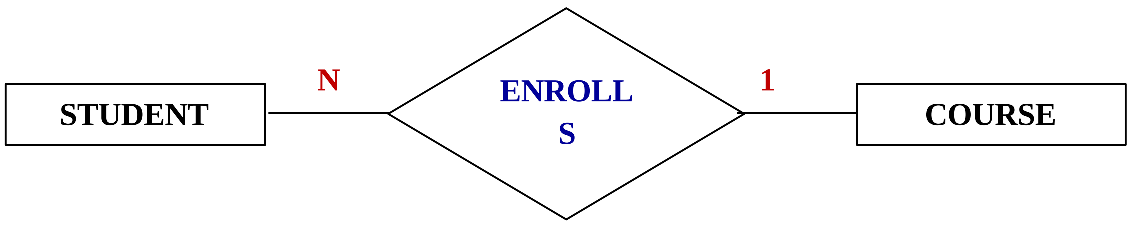


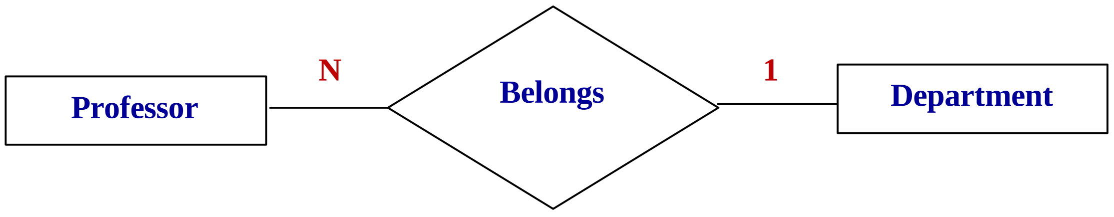
Many-to-one cardinality (N:1)

This is the reverse of 1:N relationship.

Multiple instances of entity are related to one instance of another entity.

Examples :



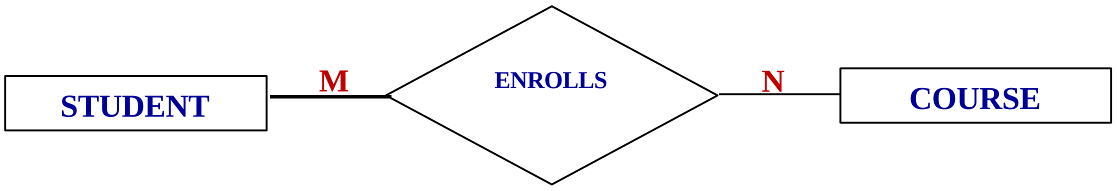


Many-to-many cardinality (M:N)

Multiple instances of entity are related to multiple instances of another entity.

Examples :

- One student can enroll in any number of courses.
- An employee can work on several projects and a project can have several employees.



Note :

1) One-to-one:

Implemented using single table by establishing relationship between same type of columns in a table.

2) One-to-many:

Implemented using two tables with primary key and foreign key relationships.

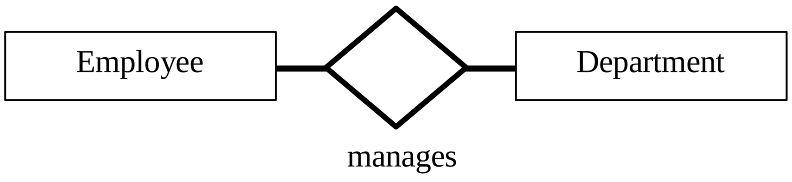
3) Many-to-many:

Implemented using a junction table. The keys from both the tables form composite primary key of the junction table.

2. Participation Constraints

It specifies whether the existence of an entity instance depends on being related to another entity. It is the minimum no. of relationship instances that each entity can participate in.

Example :

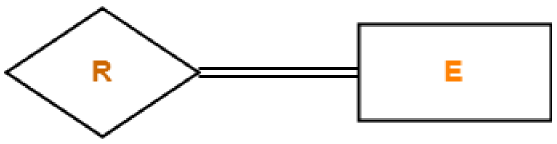


As soon as a new department D_x is created, we need to create a manager M_x to manage it. Department exists only with manager.

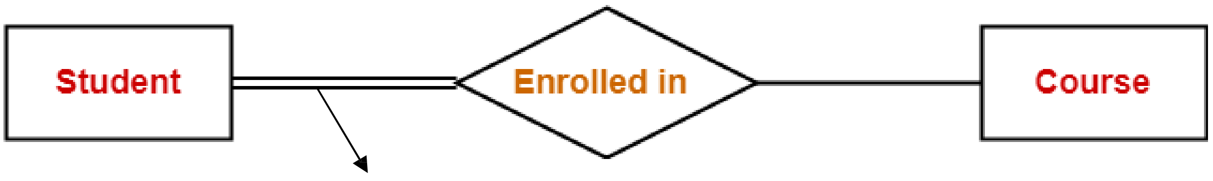
Types : Total and Partial

1. Total participation (existence dependency):

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- Total participation is represented using a double line between the entity set and relationship set.
- Example : Every student must be enrolled in a course. It means the student can exist only if it participates in atleast one "enrolled" relationship instance.



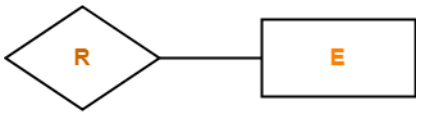
Total Participation



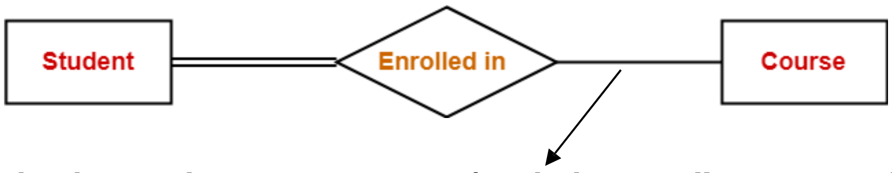
It specifies that each student must be enrolled in at least one course.

2. Partial participation:

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.



Partial Participation



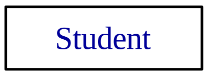
It specifies that there might exist some courses for which no enrollments are made.

Entity Types

1. Strong entity
2. Weak entity / Owner entity
3. Identifying relationship type
4. Partial key in a weak entity

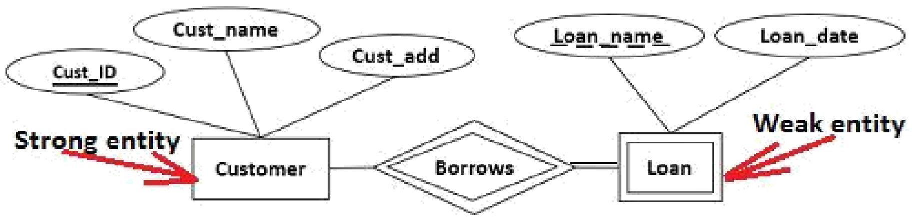
1. Strong entity

- The strong entity has a primary key.
- Its existence is not dependent on any other entity.
- Strong Entity is represented by a single rectangle.



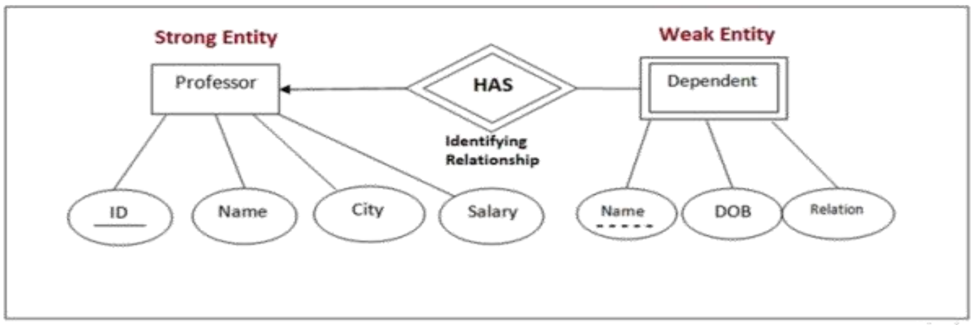
2. Weak entity

- Entity types that do not have key attributes of their own are called weak entity types. Weak entity is an entity that depends on another entity.
- Double rectangle represents weak entity.



3. Identifying relationship type

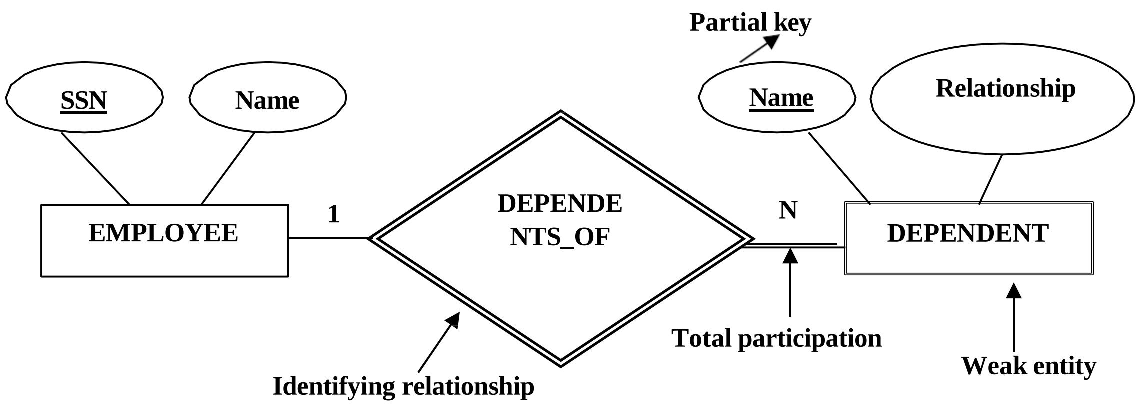
It links the strong and weak entity and is represented by a double diamond sign.



Dependent Name could not exist on its own but in relationship to a Professor.

4. Partial key in a weak entity

The set of attributes of a weak entity set that uniquely identify a weak entity for a given owner entity is called partial key.



Modelling ER Diagram

Steps to model ER Diagram

1. Identify the entities
2. Find relationships and cardinalities
3. Identify key attributes for each entity
4. Identify other relevant attributes
5. Complete ER Diagram

Example 1 : University Database Application

- One University has many departments.
- Each department has multiple professors and one of them is HOD.
- A professor belongs to only one department.
- Each department offers multiple courses. Each one is taught by single professor.
- A student may enroll for many courses offered by different departments.

Step 1: Identify the entities

Entities must have multiple instances.

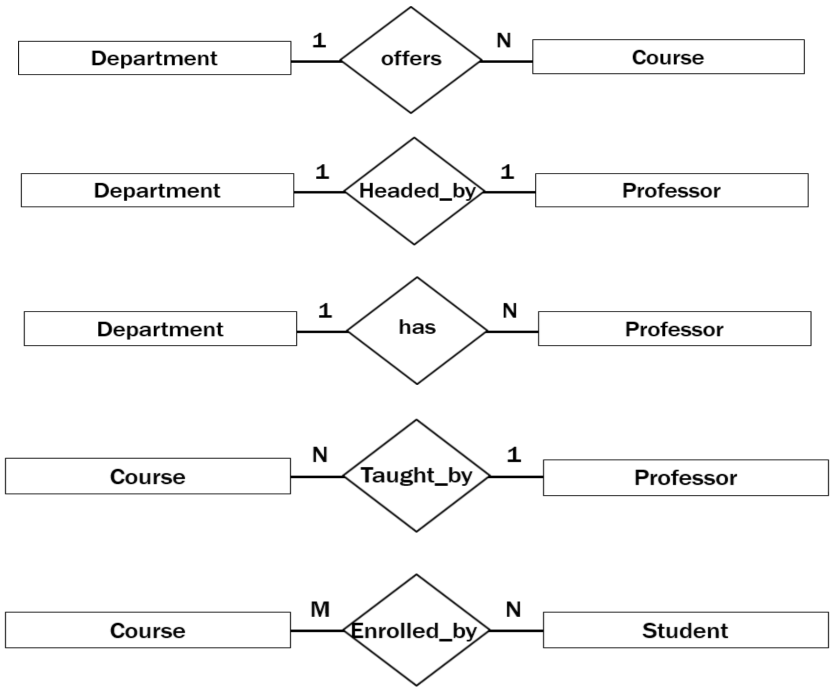
The entities are

- Department
- Course
- Professor
- Student

HOD is not entity. It is the relationship between the professor and department.

Step 2 : Find relationships and cardinalities

Entities : Department, Course, Professor, Student



We can derive the relationship between Professor and Student indirectly through Course and Professor and Course and Students.

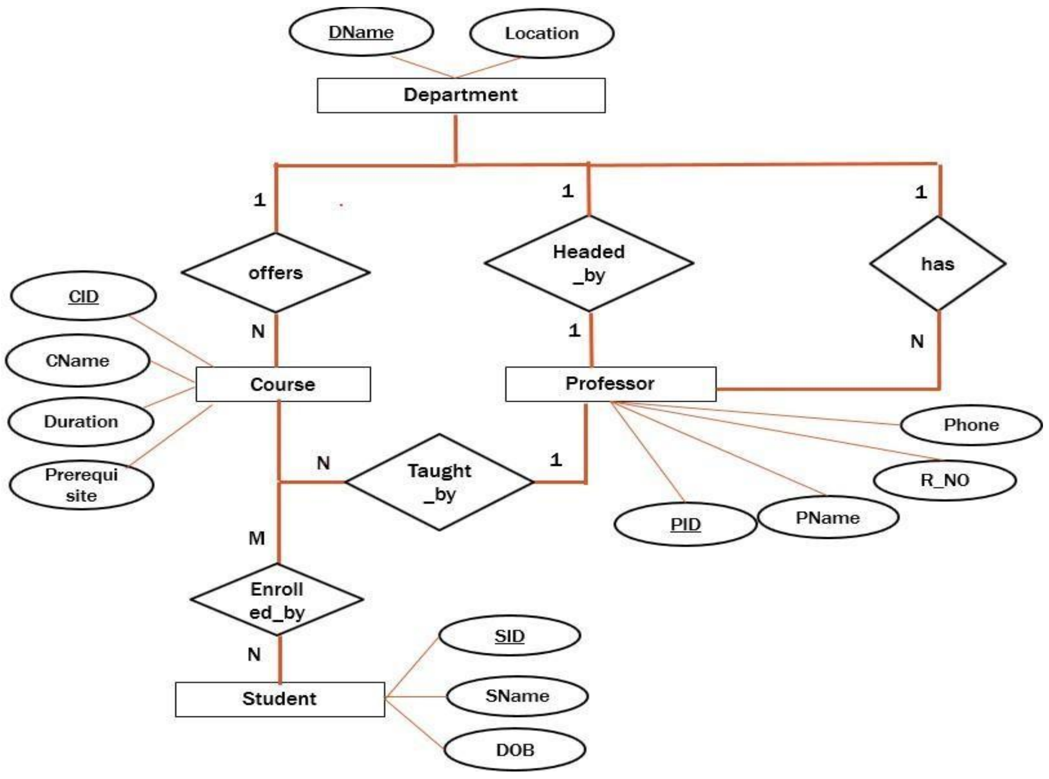
Step 3: Identify key attributes.

Department :	Dept Name
Course :	Course Number
Professor :	Professor ID
Student :	Student Number

Step 4 : Identify other relevant attributes.

Department :	Location
Course :	Course name, duration, prerequisite
Professor :	Professor Name, Room Number, Phone
Student :	Student Name, DOB, address, Phone

Step 5 : Complete the ER diagram



Example 2 : Library Database

Assume a university has

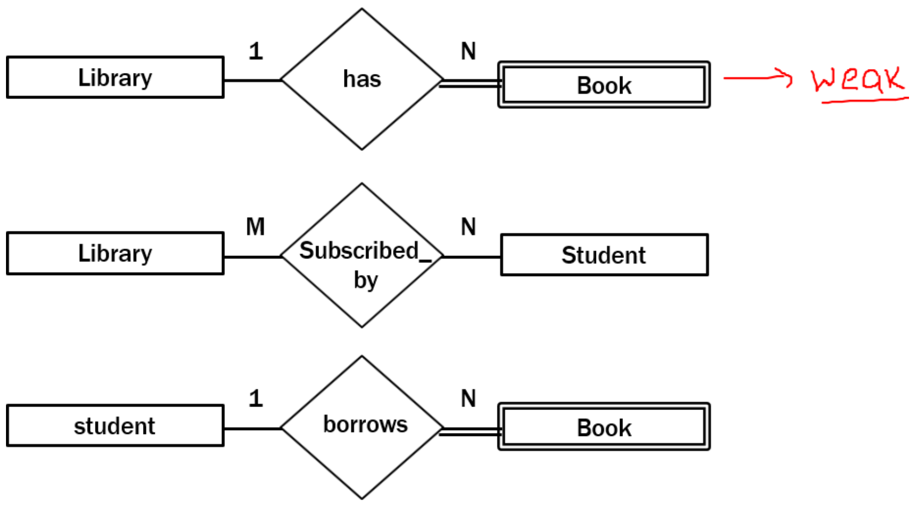
- Multiple libraries, each library has multiple student members
- Students can become members to multiple libraries by paying appropriate membership fee.
- Each library has it own sets of books, within the library, these books are identified by a unique number.
- Student can borrow multiple books from subscribed library.

Step 1: Identify the entities.

Entities must have multiple instances.

1. Library
2. Book -----> Weak entity because without knowing the library details book cannot be identified independently. Book is always associated with its library.
3. Student

Step 2 : Find relationships and cardinalities.



Step 3: Identify key attributes.

Library : Library ID

Book : Book ID

Student : Student Number

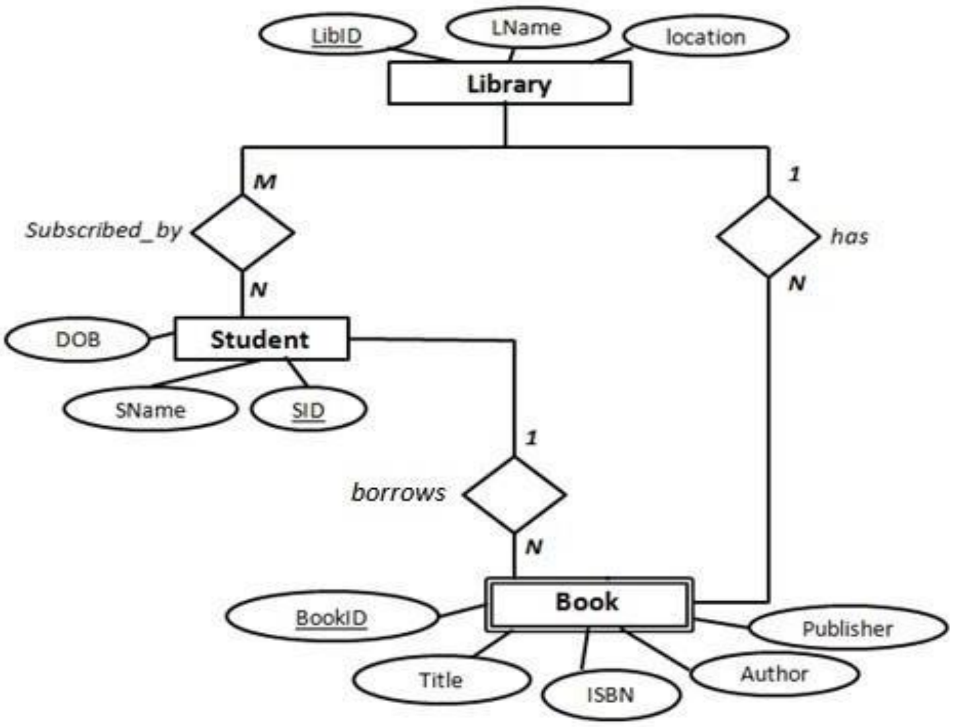
Step 4 : Identify other relevant attributes.

Library : Library Name, Location

Book : Book Title, ISBN, Author, Publisher

Student : Student Name, DOB

Step 5 : Complete the ER diagram



Example 3 : Bank Database

Assume in a city

- There are multiple banks and each bank has many branches. Each branch has multiple customers.
- Customers have various types of accounts.
- Some customers also had taken different types of loans from these bank branches.
- One customer can have multiple accounts and loans.

Step 1: Identify the entities.

- Bank
- Branch -----> Weak entity
- Loan
- Account
- Customer

Step 2 : Find relationships and cardinalities.

1. 1 Bank ---------> Many branches (1:N)
2. 1 Branch --------> Many Loans (1 : N)
3. 1 Branch --------> Multiple Accounts (1:N)
4. 1 Loan -------> Multiple Customers and each customer can avail multiple Loans (M:N)
5. 1 Customer------ > multiple accounts and each account can be held by multiple customers (M:N)

Step 3: Identify key attributes.

Bank : Bank Code

Branch : Branch Number + Bank Code

Loan: Loan Number

Account : Account Number

Customer : Customer Number

Step 4 : Identify other relevant attributes.

Bank : Bank Name, address

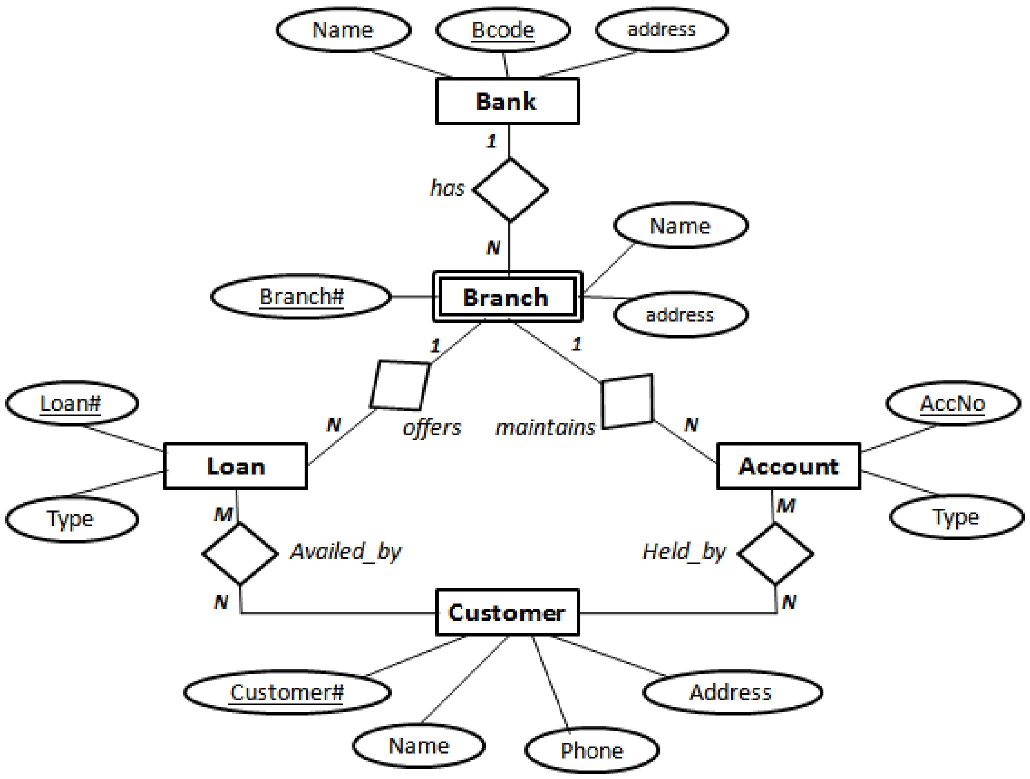
Branch : Branch Name, address

Loan: Loan type

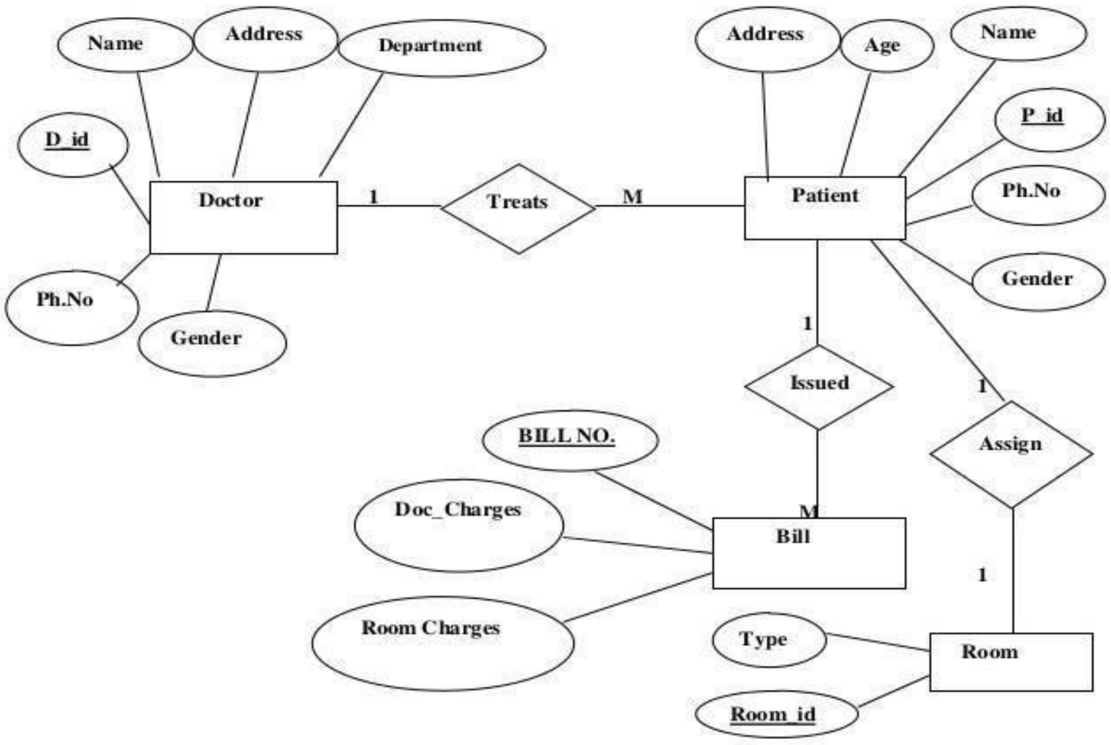
Account : Account type

Customer : Customer Name, address, phone

Step 5 : Complete the ER diagram.



Example 4 : Hospital Management


Name
Address
Department
D_id
Doctor
Ph.No
Gender
1
Treats
M
Address
Age
Name
P_id
Patient
Ph.No
Gender
1
Issued
1
Assign
BILL NO.
Doc_Charges
M
Bill
Room Charges
1
Type
Room
Room_id